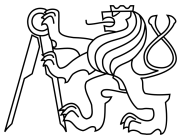
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Systém Akrmat verze 3 |
| **Student:** | Bc. Jakub Štefan |
| **Vedoucí:** | Ing. Michal Valenta, Ph.D. |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2016/17 |

## Pokyny pro vypracování

1. Seznamte se s aktuálním stavem projektu Akrmat, což je komplexní webový informační systém pro podporu tvorby akreditačních materiálů, který používá technologie webových služeb, XML nativní databázi a automatizovanou tvorbu formulářů na základě XML schéma.
2. Shromážděte a analyzujte požadavky na rozšíření a změny funkcí aplikace (zavedení rolí, přívětivější uživatelské rozhraní, kontroly, importy dat z jiných systémů apod.).
3. Proveďte revizi architektury aplikace a použitých technologií.
4. Na základě bodů 2 a 3 proveďte nový návrh.
5. Návrh implementujte, zdokumentujte, otestujte.

## Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 24. prosince 2015

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

Diplomová práce

# Systém Akrmat verze 3

***Bc. Jakub Štefan***

Vedoucí práce: Ing. Michal Valenta, Ph.D.

9. května 2016

# Poděkování

Rád bych v první řadě poděkoval vedoucímu diplomové práce, Ing. Michalu Valentovi, Ph.D., za čas, který mi věnoval. Dále děkuji své rodině za podporu.

# 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, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 9. května 2016 .....

### Odkaz na tuto práci

Štefan, Jakub. *Systém Akrmat verze 3.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

# Abstrakt

Tato diplomová práce se zabývá rozšířením funkcionality systému AKRMAT pro tvorbu a správu akreditačních materiálů na Fakultě informačních technologií ČVUT v Praze. V rámci práce je provedena analýza stávající verze systému a jsou identifikovány jeho nedostatky. Na základě získaných informací je proveden návrh a implementace vylepšení systému, včetně nového uživatelského rozhraní. Výsledkem práce je nová, plně funkční verze systému AKRMAT, která je výrazně lepším nástrojem pro správu akreditačních materiálů než jeho předchozí verze.

**Klíčová slova** AKRMAT, akreditace, akreditační materiály, XML, PHP, REST, API, KOS, VVVS, webová aplikace, uživatelské rozhraní

# Abstract

The main goal of this master's thesis is extending the functionality of an information system called AKRMAT, which is used for creating and managing accreditation documents at Faculty of Information Technology CTU in Prague. This thesis focuses on conducting an analysis of the current implementation of the system and on discovering its issues. Based on the gathered information new system upgrades have been designed and implemented. One of those upgrades is a brand new user interface. The output of this thesis is a new, fully functional version of the information system AKRMAT, which is a more powerful tool for managing accreditation documents than its previous version.

**Keywords** AKRMAT, accreditation, accreditation documents, XML, PHP, REST, API, KOS, VVVS, web application, user interface

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Vyučování na vysoké škole je podmíněno udělením akreditace alespoň jednoho studijního programu. Akreditaci uděluje MŠMT[1] schválením tzv. akreditačního spisu. Jedná se o komplexní dokument, který, po jeho schválení, opravňuje vysokou školu k vyučování konkrétního studijního programu. Akreditační spis je nezbytné schválit nejen při tvorbě nového programu, ale taktéž při změně stávajícího platného programu. Akreditace se uděluje na dobu určitou (v řádech několika let), poté se musí studijní program znovu reakreditovat.

Dle informací získaných konzultací s vedoucím diplomové práce začíná celý proces udělení akreditace studijního programu vytvořením a vyplněním mnoha formulářů s jasně definovaným obsahem. Dokument obsahující tyto formuláře je právě výše zmiňovaný akreditační spis. Formuláře ve spisu jsou několika typů a jsou od sebe odlišeny písmeny abecedy. Každý typ formuláře obsahuje data týkající se různých aspektů studijního programu. Například formulář C obsahuje podrobné informace o jednotlivých oborech spadajících pod studijní program a formulář G zase reprezentuje životopis jednoho pedagoga. Z toho vyplývá, že některé formuláře se nacházejí ve spisu více než jedenkrát.

Po vytvoření se akreditační spis přesune ke schválení do akademického senátu. Pokud je spis v senátu v pořádku schválen, putuje dále k finálnímu schvalování na MŠMT. Až po konečném souhlasu z ministerstva lze studijní program začít vyučovat.

Celý akreditační spis je velice obsáhlý dokument. Jeho absolutní velikost závisí na obsáhlosti studijního programu. Může se pohybovat i v několika stovkách stran. Když se vezme v úvahu, že každá stránka reprezentuje jeden formulář, který musí být vyplněn, ruční vyplnění se jeví jako nesmírně zdlouhavá a pracná činnost, při které může navíc snadno dojít k chybám.

Pro usnadnění vytváření akreditačního spisu využívá Fakulta informačních technologií ČVUT v Praze vlastní informační systém. Systém nese název AKRMAT (zkratka pro slovní spojení „akreditační materiály“) a má za úkol

[1] Ministerstvo školství, mládeže a tělovýchovy

pomocí webových služeb integrovat data z jiných školních systémů, poskytnout možnost tato data spravovat a modifikovat a následně umožnit vygenerovat celý, kompletně vyplněný akreditační spis připravený do procesu schvalování.

Od zavedení současné verze (verze 2) v roce 2012 byl systém v rámci několika diplomových a bakalářských prací postupně vylepšován. Momentálně se nachází, co se týče funkcionality, v uspokojivém, ale nikoli ideálním stavu. Naposledy byl použit v roce 2015, kdy bylo nutné reakreditovat magisterský studijní program Informatika. Pro dosažení správného výstupu bylo nutné systému pomoci různými externími skripty pro import a manipulaci s daty. Výsledné vygenerované formuláře navíc nebyly kompletní a musely být ručně doplňovány.

Hlavním cílem této diplomové práce je posunout celý systém, co se týče použitelnosti, od uspokojivého co nejblíže k ideálnímu stavu tak, aby byla při další práci se systémem eliminována nebo alespoň velmi minimalizována potřeba externích zásahů.

Kapitola 1

# Cíl a struktura práce

Tato diplomová práce je součástí projektu AKRMAT a navazuje na diplomové práce Bc. Michala Kabelky, Bc. Luboše Růžičky a Bc. Jakuba Jambora a na bakalářskou práci Jakuba Houta. V textu se budou objevovat odkazy na tyto práce, a to zejména v kapitole 2, zabývající se analýzou současné verze systému.

## 1.1 Motivace

V roce 2015 bylo nutné provést reakreditaci Magisterského studijního programu na Fakultě informačních technologií ČVUT v Praze. Při ní vyšlo najevo několik problémů se systémem AKRMAT, který je na fakultě používán pro správu akreditačních materiálů.

Poslední vývoj na systému probíhal před dvěma lety v roce 2014. Některá funkcionalita, jejíž analýza byla provedena již v roce 2012, navíc stále není implementována. Implementace chybějící funkcionality a odstranění všech zaznamenaných problémů by znamenalo pro FIT[2] opět krok kupředu v rámci optimalizace celého procesu schvalování akreditačního spisu.

## 1.2 Vymezení zadání

Zadáním práce je analýza možností vylepšení stávající verze systému AKRMAT a jejich realizace. To zahrnuje sběr požadavků na vylepšení od vedoucího práce a podrobnou analýzu současné verze systému. Dále na základě nasbíraných informací provést návrh revize architektury, návrh revize datového modelu, návrh nového uživatelského rozhraní, průzkum možností modernizace použitých technologií a rozšíření funkcionality systému, např. o importy dat z jiných systémů (KOS, VVVS). Následně tento komplexní návrh implementovat a otestovat.

[2]Fakulta informačních technologií

## 1.3 Výstup práce

Konečným výstupem práce je nová, plně funkční verze systému AKRMAT, která odstraňuje nedokonalosti předchozí verze a bude k dispozici při dalším procesu schvalování akreditačního spisu na Fakultě informačních technologií ČVUT v Praze. Součástí výstupu je i aktualizovaná instalační příručka.

## 1.4 Struktura práce

- **Kapitola Analýza současného řešení** – V této kapitole se nejprve budu věnovat analýze stávajícího řešení. Stručně popíši historii vývoje, jak vlastně systém pracuje a co je jeho přínosem. Budu také popisovat architekturu systému a jaké jsou použité technologie. Pokusím se shrnout hlavní nedostatky a identifikovat další prostory ke zlepšení. Na závěr této kapitoly provedu stručnou analýzu stávajícího uživatelského rozhraní webové aplikace.
- **Kapitola Použité technologie** – V této kapitole uvedu informace o důležitých použitých technologiích jak v současné verzi, tak při implementaci verze nové.
- **Kapitola Návrh a metodika nového řešení** – Zde podrobně popíši navrhované změny a opravy identifikovaných problémů. Dále okomentuji filozofii, která stojí za návrhem nového uživatelského rozhraní.
- **Kapitola Implementace** – V této kapitole se budu věnovat popisu implementačních postupů. Zmíním problémová místa, na která jsem při práci narazil a jakým způsobem jsem se s těmito problémy vypořádal.
- **Kapitola Testování** – Tato kapitola je zaměřena především na uživatelské testy. Popíši zde metodiku testování, použité testovací scénáře a výsledky testování.
- V závěru je ještě uvedeno mé celkové hodnocení odvedené práce a zmíněn další možný rozvoj systému AKRMAT.

V textu jsou často použity termíny „systém AKRMAT“ nebo „aplikace AKRMAT“. V kontextu této práce budou tyto termíny považovány za synonyma. Pokud bude nutné zmínit konkrétně webovou aplikaci (webové rozhraní) AKRMAT, bude použit explicitní termín „webová aplikace“.

Kapitola 2

# Analýza současného řešení

## 2.1 Historie vývoje systému AKRMAT

První verze aplikace vznikla už před rokem 2012. V této sekci se budu nicméně věnovat až druhé verzi, jejíž vývoj začal právě v roce 2012 a postupně se na ní od té doby podíleli čtyři studenti v rámci svých diplomových a bakalářských prací. Vývoj druhé verze započaly dvě souběžně běžící a navzájem se doplňující diplomové práce Bc. Michala Kabelky a Bc. Luboše Růžičky.

V rámci těchto prací došlo ke změně typu databáze pro uložení dat o akreditačních materiálech. Z do té doby používané relační databáze se přešlo na XML nativní databázi Sedna. To úplně změnilo podstatu uchovávaných dat, protože v rámci XML nativní databáze se data neuchovávají v tabulkách, ale v podobě kolekcí XML dokumentů.

Obě práce se tedy zabývaly migračním procesem na novou databázovou platformu, zároveň ještě každý autor přispěl konceptuální analýzou určité části systému. M. Kabelka se zabýval především správou a generováním životopisů. Dále zkoumal možnosti propojení aplikace AKRMAT pomocí webové služby se školním systémem VVVS, který úzce souvisí se správou životopisů, protože poskytuje informace o vědeckých publikacích osob. Okrajově se také zmínil o možnostech propojení se školním KOS API.[1]

L. Růžička se ve své diplomové práci zabýval problematikou správy a generování předmětů. Provedl analýzu a návrh propojení aplikace AKRMAT s KOS API a navrhl REST API pro manipulaci se zdroji reprezentující data uložená v XML nativní databázi. Navíc vytvořil velmi jednoduchý frontend[3], který umožnil aplikaci pohodlněji používat a testovat.[2] Tento frontend se i přes různé úpravy používá i nyní.

Výsledkem těchto dvou prací nicméně nebyl kompletně funkční systém, ale spíše jakási kostra, na kterou bylo třeba ještě v budoucnu navázat.

[3] klientská část aplikace

Tohoto úkolu se zhostil Bc. Jakub Jambor. V rámci své diplomové práce měl za úkol konsolidovat výstupy předchozích dvou prací a uvést systém do provozuschopného stavu. Zaměřil se na implementaci samotného generování formulářů akreditačního spisu a umožnil uživatelům přihlašovat se do aplikace pomocí LDAP[4]. Ve finále ale fungovalo generování pouze některých formulářů, nikoliv všech potřebných pro kompletní akreditační spis.[3]

Posledním, kdo na systému AKRMAT pracoval, byl Jakub Hout. Jakub Hout při vypracovávání své bakalářské práce zavedl novou entitu reprezentující studijní program. Do té doby šlo generovat akreditaci pouze v rámci studijních oborů, nikoli studijního programu jako celku. Dokončil také funkcionalitu generování zbylých formulářů a vylepšil možnosti frontendu tak, aby mohli uživatelé pracovat s nově zavedenou entitou Program.[4]

Po práci Jakuba Houta je tedy systém v provozuschopném stavu, ale stále má mnoho nedostatků a chybějících funkcionalit, viz sekce 2.5.

## 2.2 Architektura

Systém je implementován v jazyce PHP a běží na serveru Apache pod operačním systémem Linux. Jeho architektura je názorně ilustrována na obrázku 2.1.

### 2.2.1 Databázové technologie

V pozadí se nachází XML nativní databáze Sedna. V ní jsou uložena data o třech entitách vystupujících v systému: Předmět, Životopis a Program. Data jsou umístěna v kolekcích XML dokumentů, kde každá kolekce připadá jedné entitě. Vedle databáze Sedna se ještě používá relační databáze MySQL pro autentizaci a autorizaci uživatelů.

### 2.2.2 Databázová vrstva

Pro spojení s databází Sedna slouží třída `SingleXMLConnection`, která využívá implementaci rozhraní `XMLPDO` pro operace vkládání a čtení dat z databáze. To zaručuje v budoucnu možnost jednoduché výměny databáze Sedna za jinou XML nativní databázi. Pro spojení s MySQL databází není použito žádné externí třídy. Tato databáze je využita pouze k uložení informací o uživatelích (uživatelské jméno, heslo a role), proto se spojení provádí přímo ve třídě `Authentication`. Pro získání dat z této databáze se využívá nativní PHP třída `PDO`.

[4]Lightweight directory access protocol

### 2.2.3 XML vrstva

Získané XML dokumenty z databáze je nutné nějak v aplikaci reprezentovat. K tomu slouží třídy `DocumentCollection` pro celé kolekce dokumentů a `Document` pro jediný XML dokument. K validaci ukládaných XML dokumentů slouží třída `XMLValidator`.

### 2.2.4 REST vrstva

Nad XML vrstvou se nachází vrstva implementující RESTful webovou službu pomocí PHP frameworku[5] Tonic. Tato vrstva poskytuje zdroje, reprezentované třídou `Resource`, které jsou identifikovány pomocí URI[6] a dostupné přes metody protokolu HTTP. Zdroje poskytují rozhraní pro provádění CRUD[7] operací nad daty uloženými v databázi Sedna.

Do REST vrstvy patří ještě další důležité třídy. Třída `Atom`, která se stará o sestavení dat pro prezentaci ve formátu Atom XML, třída `Generator` zajišťující transformaci dat do různých výstupních formátů a `StatusXML` pro reprezentaci zpráv zobrazených uživateli.

#### 2.2.4.1 Příklady HTTP požadavků na zdroje REST API systému AKRMAT

- Seznam předmětů - `GET /predmety`
- Informace o konkrétním předmětu - `GET /predmety/{kod_predmetu}`
- Vytvoření nového předmětu - `PUT /predmety`
- Úprava konkrétního předmětu - `POST /predmety/{kod_predmetu}`
- Smazání konkrétního předmětu - `DELETE /predmety/{kod_predmetu}`

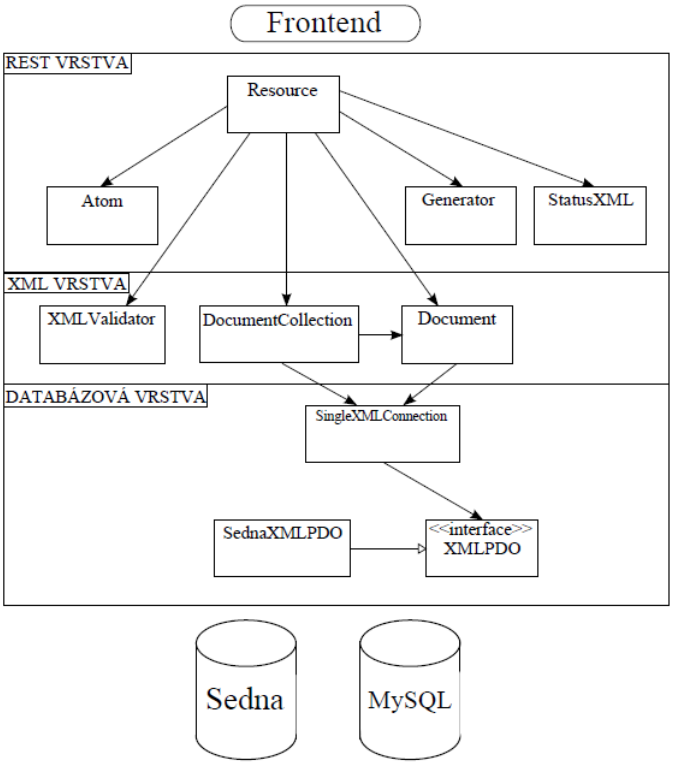
Výše uvedený příklad lze analogicky aplikovat i na ostatní entity. V implementaci AKRMAT REST API jsou chybně prohozené metody POST a PUT. Tento fakt je dále okomentován v sekci 2.5.7.2.

### 2.2.5 Webová aplikace

Na vrcholu pomyslného stromu architektury systému stojí webová aplikace. Ta je implementovaná v jazyce PHP na serverové straně a v klasických webových technologiích HTML, CSS a JavaScript na straně klientské. V pozadí se používá vestavěná PHP knihovna libcurl k získání výše uvedených zdrojů

[5] softwarová struktura, která podporuje vývoj softwarových projektů
[6] Uniform Resource Identifier
[7] Create, Read, Update, Delete

REST API. Hlavním úkolem této webové aplikace je umožnit uživatelům snadnou manipulaci s daty uloženými v databázi. Aplikace se také stará o finální generování akreditačního spisu.




Obrázek 2.1: Architektura systému AKRMAT [4]

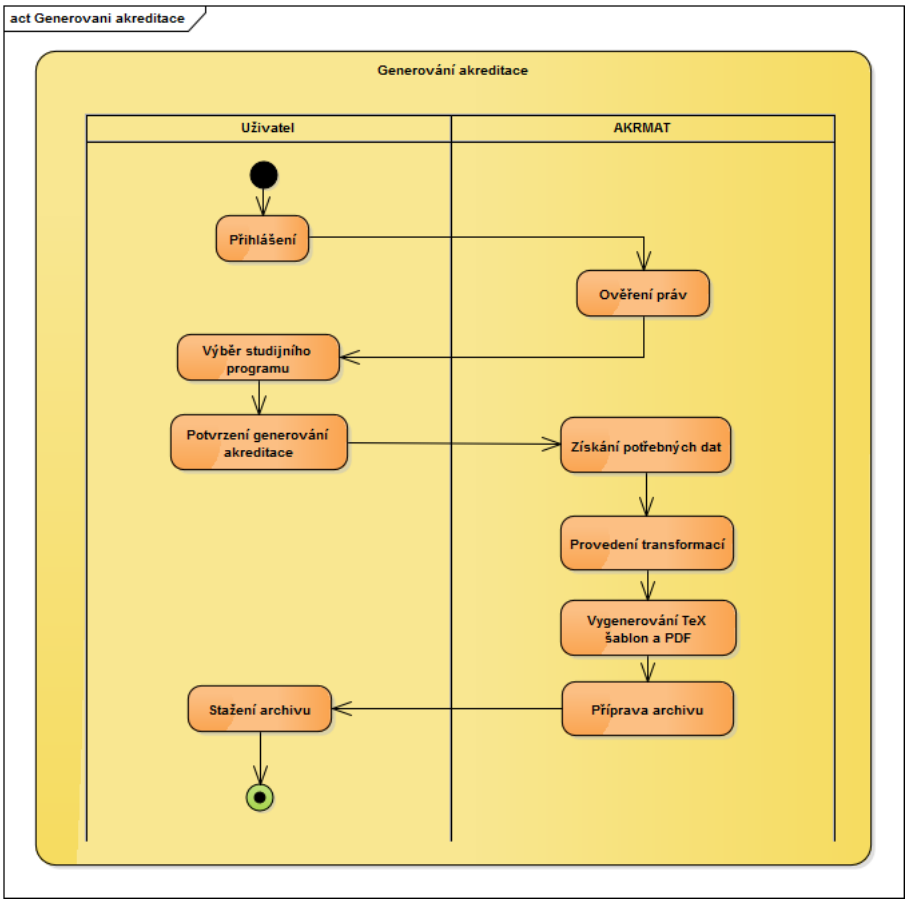
## 2.3 Proces generování akreditačního spisu

Hlavním případem užití systému je generování akreditačního spisu. Celý proces popsal ve své diplomové práci Bc. Jakub Jambor a lze ho vidět na UML[8] diagramu 2.2.

Nejprve dojde k autentizaci uživatele. Pokud je oprávněn pro vstup do aplikace, může přejít na stránku s formulářem vybraného studijního programu. Zde se nachází tlačítko „Generovat akreditaci“. Po jeho stisknutí se spustí samotný proces generování. Webová aplikace v pozadí spustí shell skript, který pomocí programu cURL posílá HTTP požadavky na příslušné zdroje poskytované AKRMAT REST API.

Příklad URL zdroje při generování akreditačního spisu pro studijní program BI – Informatika:

[8]Unified Modeling Language




Obrázek 2.2: Diagram aktivit pro proces generování akreditačního spisu

```
/programy/BI.akr?form=formD
```

Z URL lze poznat, že se jedná o zdroj obsahující data o programech, konkrétně o programu s kódem `BI`, což je zároveň v rámci systému AKRMAT jeho unikátní identifikátor. Přípona obecně v rámci AKRMAT REST API indikuje, v jakém formátu jsou data požadována. Přípona `.akr` říká, že je požadován formát pro generování akreditace, konkrétně se pod tím ukrývá formát `.tex`.

Parametr `form` v rámci URL s hodnotou `formD` říká, že se bude generovat formulář typu D z akreditačního spisu. Podle logiky popsané ve třídě `FormDGenerator` jsou poté poskládána potřebná data a pomocí XSLT transformace na výstup vygenerována `.tex` šablona pro formulář D.

Postupně jsou provedeny požadavky na všechny typy formulářů a shell skript všechny obdržené šablony spojí do jedné, na kterou spustí program pdflatex, který šablonu zkompiluje do formátu `.pdf`. Výsledek, včetně všech zdrojových `.tex` šablon, se ještě zabalí do `.tar.gz` archivu a poskytne se

uživateli ke stažení. Zdrojové šablony jsou přiloženy pro případ potřeby ruční korekce.

## 2.4 Generování HTML formulářů

HTML formuláře pro jednotlivé entity jsou generovány dynamicky z XML schématu pomocí dvou XSLT transformací, nejprve do formátu RELAX NG (RNG) a následně do HTML. Poté aplikace pomocí REST API stáhne potřebná data a formulář jimi s využitím JavaScriptu naplní. Uživatel může následně formulář libovolně upravovat.

## 2.5 Problémy a nedostatky

Bylo zjištěno, že systém AKRMAT obsahuje určité nedostatky a postrádá důležitou funkcionalitu. Tato sekce se blíže těmto problémům věnuje. Při identifikaci problémů bylo využito tří informačních pramenů: zadání práce, konzultace s vedoucím práce a osobní zkušenost se systémem. V kapitole 4 je pro každý zde uvedený problém navrženo řešení.

### 2.5.1 Seznam nedostatků a problémových míst

- Studijní obory součástí programů
- Chybějící import dat
- Uživatelské role
  - Nastavení uživatelských rolí
  - Využití uživatelských rolí v systému
- České názvy v datovém modelu
- Kvalita zdrojového kódu
- REST API
  - Nefunkční zamykání editace dokumentů
  - Prohození sémantiky metod POST a PUT
  - Použití PHP session
- Webová aplikace
  - Šablonovací systém htmltmpl
  - Javascriptové chyby
  - Adresářová struktura

- Struktura URL
- Uživatelské rozhraní

### 2.5.2 Studijní obory součástí programů

Studijní obory jsou nyní evidovány v rámci studijního programu, do kterého patří. Jakub Hout v kapitole 5.1.1 ve své práci uvádí, že předměty oboru nemohou být evidovány v rámci XML elementu `obor`. Těchto předmětů totiž může být teoreticky nekonečně mnoho a to samé platí i o oborech umístěných v elementu `obory_programu`. V praxi se tedy jedná o dvě vnořené nekonečné sekvence XML elementů a tuto konfiguraci z nějakého důvodu nelze uložit do databáze Sedna.[4]

Momentálně je tento problém řešen extrakcí předmětů oboru do samostatného elementu `predmety_oboru`, který se nachází na stejné úrovni jako element `obory_programu` v rámci dokumentu programu. Všechny předměty všech oborů studijního programu se tedy nachází uvnitř jednoho velkého XML elementu `predmety_oboru`.

Jelikož generované HTML formuláře v aplikaci reflektují strukturu XML dokumentů, má skutečnost popsaná v předchozím odstavci jeden nepříjemný vedlejší účinek. Formulář programu je velmi rozsáhlý a značně nepohodlný pro prohlížení a úpravu. Dekompozice studijních oborů do samostatné entity řeší oba tyto problémy a tedy zefektivňuje práci se systémem. Tento krok už navrhl ve své práci Jakub Hout, ale z nedostatku času a také kvůli jeho náročnosti ho nerealizoval.[4]

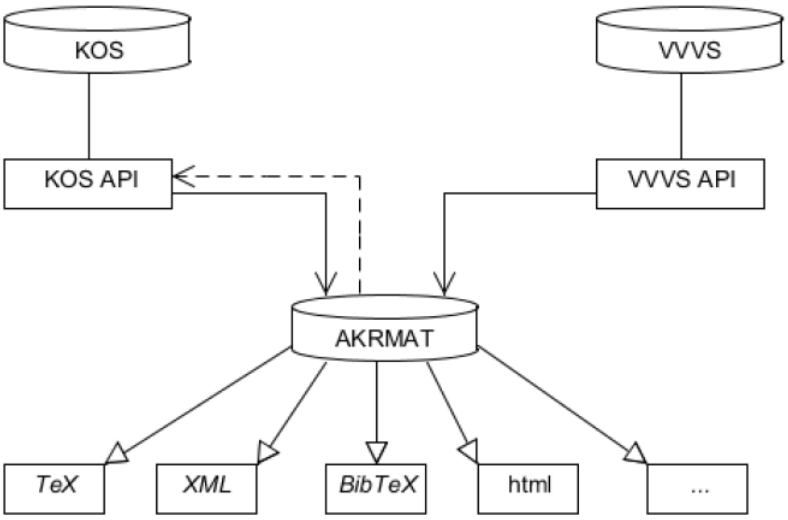
### 2.5.3 Chybějící import dat

Dalším velkým nedostatkem je chybějící, do aplikace plně integrovaná, možnost importu dat z jiných školních systémů. Přestože analýza této funkcionality byla provedena už v roce 2012 v diplomové práci Bc. Michala Kabelky, viz obrázek 2.3, není stále implementována.

Pro potřeby aplikace AKRMAT připadá v úvahu import dat z KOS API a VVVS API. Obě webové služby jsou implementovány v rámci standardu webových služeb REST a jsou přístupné ke čtení pro všechny uživatele se zřízeným ČVUT účtem.

Do této doby byl na začátku procesu vytváření akreditačních materiálů prováděn import ručně pomocí různých skriptů a datových transformací. Dle konzultace s vedoucím práce, který tyto importy prováděl, to fungovalo sice uspokojivě, ale zároveň to bylo nepohodlné a časově náročné.

KOS API poskytuje informace o všech entitách vystupujících v systému AKRMAT. Do životopisů lze získat osobní informace pedagogů, např. jméno, příjmení nebo akademické tituly. O předmětech je v KOS API dostupných více informací, např. kód předmětu, název, požadavky na studenta, zakončení




Obrázek 2.3: Schéma importu dat ze systémů KOS API a VVVS API [1]

předmětu, náplň přednášek nebo doporučená literatura. Pro programy a obory lze importovat jejich kódy, názvy, garanty a obecné slovní charakteristiky.

VVVS API poskytuje informace o publikacích jednotlivých osob v rámci ČVUT. Tato data lze využít u životopisů, kde se eviduje několik posledních publikací.

V součtu se jedná o mnoho dat, a proto je zcela zřejmé, že pokud se import integruje přímo do aplikace a automatizuje, ušetří to velké množství práce.

### 2.5.4 Uživatelské role

| Činnost \ Role | Nepřihlášený | Čtenář | Autor | Editor | Admin |
|---|---|---|---|---|---|
| Čtení záznamů | × | ✓ | ✓ | ✓ | ✓ |
| Vytváření záznamů | × | × | ✓ | ✓ | ✓ |
| Editace vlastních záznamů | × | × | ✓ | ✓ | ✓ |
| Editace cizích záznamů | × | × | × | ✓ | ✓ |
| Generování studijního plánu | × | ✓ | ✓ | ✓ | ✓ |
| Generování akreditace | × | ✓ | ✓ | ✓ | ✓ |
| Zálohování databáze | × | × | × | ✓ | ✓ |

Obrázek 2.4: Přehled uživatelských rolí a jejich oprávnění [3]

#### 2.5.4.1 Nastavení uživatelských rolí

Bc. Jakub Jambor ve své práci provedl analýzu uživatelských rolí. Stanovil, že v aplikaci budou vystupovat čtyři hlavní uživatelské role. Výsledek této analýzy lze vidět na obrázku 2.4. Těmito rolemi jsou Čtenář, který má právo prohlížet formuláře, generovat akreditaci a studijní plán a Autor, který má navíc právo vytvářet, editovat a mazat jím vytvořené záznamy. Další dvě role Editor a Admin jsou ekvivalentní a mají oprávnění ke všem úkonům v aplikaci.[3]

Zde vidím dvě menší nedokonalosti. Čtenář by dle mého názoru neměl mít možnost generovat akreditaci a studijní plán. Z čistě sémantických, nikoli bezpečnostních důvodů. Všechna data uvedená v akreditačním spisu lze totiž získat v aplikaci i jiným způsobem. Dalším menším problémem je redundance rolí Admin a Editor. V současné době není důvod je od sebe odlišovat, tudíž je role Editora nadbytečná.

#### 2.5.4.2 Využití uživatelských rolí v systému

Úplná kontrola přístupových práv se provádí pouze uvnitř webové aplikace ve skriptu `edit.php`. Na úrovni API je její kontrole věnováno pouze několik řádků kódu. Konkrétně podmínka v metodě pro mazání dokumentů:

```
if ($_SESSION["user"]["prava"] != "admin" ) {
   $errXML = new ERRORstatusXML(
      "Nemáte práva k~odstranění životopisu",
      Response::UNAUTHORIZED
   );
// ...
```

Zde lze vidět, že role Admin a Editor ve skutečnosti nejsou ekvivalentní, což odporuje předchozímu návrhu. Mazání je povoleno pouze pro roli Admin.

Tato kontrola na straně API je dle mého názoru nedostatečná. I pouhý Čtenář může přímým HTTP požadavkem na API v podstatě provádět jakékoliv úkony kromě mazání dokumentů, což odporuje nastavení rolí z předchozí sekci.

### 2.5.5 České názvy v datovém modelu

Bc. Luboš Růžička ve své diplomové práci v sekci 3.1.2 uvedl, že názvy v rámci datového modelu jsou převzaté z původní relační databáze, kde byly v českém jazyce. Dle vývojové koncepce FIT by ale měly být v jazyce anglickém. Tento požadavek nebyl při původní migraci systému z relační databáze do XML nativní databáze včas předán vývojářům.[2] V průběhu vývoje bylo do aplikace přidáváno mnoho další funkcionality, ale tento problém zůstal neřešen.

Nyní by přeložení názvů všech XML elementů v celém datovém modelu znamenalo příliš mnoho práce. V rámci téměř každé funkcionality se totiž v nějaké fázi používá jazyk XPath, který využívá názvy XML elementů k procha-

zení stromu elementů XML dokumentu. Musely by se mimo jiné tedy upravit všechny XPath dotazy.

Na přesných názvech XML elementů závisí i webová aplikace. Jelikož je HTML formulář dynamicky generovaný z XML schématu, musí být upraven pomocí JavaScriptu do podoby vhodné k prezentaci uživateli. Většina modifikací formuláře je nějak závislá na názvech vstupních polí, které se opět odvíjejí od názvů XML elementů.

### 2.5.6 Kvalita zdrojového kódu

Když jsem se dozvěděl, že na systému pracovalo během několika let mnoho lidí, navíc v rámci svých závěrečných prací, kdy často vítězí rychlost vývoje nad kvalitou, měl jsem trochu obavy o kvalitu zdrojového kódu. Jestli bude dobře čitelný, jak rychle mu dokáži porozumět a jak snadno bude rozšiřitelný.

Mé obavy se částečně potvrdily. Ve zdrojovém kódu se nachází mnoho zakomentovaných částí. Některé jsou pouhé ladící výpisy, u ostatních ale často není uveden důvod, proč jsou zakomentované a zda nejsou nějak důležité.

Implementace AKRMAT REST API je dobře strukturovaná do tříd, čitelná a relativně dobře okomentovaná. U webové aplikace už je situace horší, zde je zakomentovaných částí více. Dále by, dle mého názoru, mohly být použity výstižnější názvy proměnných.

Celý zdrojový kód webové aplikace kromě tříd obalujících šablonovací systém htmltmpl a tříd realizujících HTTP komunikaci je umístěn v rámci dvou velkých PHP skriptů bez jakékoli strukturalizace kódu do tříd nebo alespoň funkcí.

Tyto aspekty mi velmi ztížily proces seznámení se systémem a způsobily, že mi to trvalo mnohem déle, než jsem očekával. Také přispěly k rozhodnutí vytvořit celou webovou aplikaci úplně od začátku.

### 2.5.7 REST API

Zde popíši několik problémů, na které jsem narazil při práci s AKRMAT REST API. Při analýze jsem čerpal ze zkušeností nabraných v praxi, informací získaných absolvováním předmětu Web 2.0 (MI-W20) na FIT a také z článku Best Practices for Designing a Pragmatic RESTful API od Vinay Sahni.[5]

#### 2.5.7.1 Nefunkční zamykaní editace dokumentů

Jedno z vylepšení, které jsem dostal za úkol do aplikace implementovat, je funkční zamykání editace dokumentů. V praxi by to mělo zajišťovat, že jakýkoliv dokument nemohou editovat současně dva uživatelé. V rámci REST API je pro zamykání dokumentů vytvořena pouze kostra, a proto nebyla tato funkcionalita dosud využívána.

Zamknout editaci dokumentu lze nyní přidáním parametru `zamknout` s hodnotou `true` do HTTP GET požadavku na URL zdroje dokumentu. Ode-

mykání funguje analogicky pomocí parametru `odemknout`. Dle [5] by HTTP GET parametry by měly být použity výhradně k filtrování, dotazování a řazení.

Hlavním nedostatkem této funkcionality je ale absence samovolného odemykání dokumentů. Pokud někdo dokument zamkne a explicitně neodemkne, zůstane dokument navždy zamčený. Dalším problémem je, pokud uživatel odešle požadavek na zamknutí už na jím zamčený dokument, vrátí se mu chyba, že dokument má údajně zamčený někdo jiný. Při zamykání dokumentu se totiž pouze kontroluje, zda byl už dokument zamčen, ale ne kým.

#### 2.5.7.2 Prohození sémantiky metod POST a PUT

POST a PUT jsou dva typy HTTP metod a slouží k posílání dat unvitř těla HTTP požadavku. V rámci RESTful API mají obě metody odlišný sémantický význam. Dle [5] by metoda POST měla sloužit k vytváření nového zdroje a metoda PUT k modifikace zdroje existujícího. V zásadě se obě metody liší pouze v URL, na kterou se odesílají data. URL při požadavku POST může vypadat následovně:

```
http://www.example.com/api/predmety
```

Zde se požadavek posílá na URL zdroje kolekce předmětů. Tím je určeno, že se do této kolekce přidává nový předmět. URL při požadavku PUT může naopak vypadat následovně:

```
http://www.example.com/api/predmety/test
```

Zde se požadavek posílá také do kolekce předmětů, ale na konkrétní předmět s identifikátorem `test`. Tím je určeno, že se tento předmět modifikuje.

V AKRMAT REST API jsou tyto dvě metody prohozené. Tedy POST slouží k modifikaci dokumentu a PUT k přidávání nových dokumentů do kolekcí. Je to čistě sémantická chyba, která nemá žádný vliv na funkcionalitu, ale jsou tím porušeny principy RESTful webové služby.

#### 2.5.7.3 Použití PHP session

Dle [5] by mělo být RESTful API bezstavové. Veškeré informace potřebné k rekonstrukci stavu a autentizační údaje by měly být obsažené v každém požadavku.

V implementaci AKRMAT REST API se používá vestavěná PHP proměnná `$_SESSION` k uložení informací o přihlášeném uživateli. Ukládá se jeho uživatelské jméno, MD5 hash hesla a uživatelská role. Neporušuje to pravidlo uvedené výše, protože je autentizační informace stejně vyžadována při každém požadavku. V tomto případě je proměnná `$_SESSION` použita k přenosu dat mezi PHP skripty. Ukládání hesla do session, i když je to v tomto případě pouze jeho MD5 hash, není dobrý nápad z hlediska bezpečnosti.

Hlavním využitím PHP session je přenos dat mezi několika různými HTTP požadavky. V případě AKRMAT REST API se veškerá logika provádí v rámci jednoho konkrétního HTTP požadavku, takže k přenosu dat mezi PHP skripty není nutné využívat proměnné `$_SESSION`.

### 2.5.8 Webová aplikace

#### 2.5.8.1 Šablonovací systém htmltmpl

Pro separaci prezentace (HTML kódu) a logiky (PHP kódu) je ve webové aplikaci AKRMAT použit šablonovací systém htmltmpl. Ten byl vytvořen už v roce 2001 Tomášem Stýblem a jeho poslední aktualizace se datuje do roku 2007.[6] Jedná se tedy o velmi starou technologii s takřka nulovou šancí, že by ještě někdy mohla být aktualizována. Dnes se na trhu pohybuje daleko více konkurenčních řešení, která jsou, co se týče funkcionality, vyspělejší.

Vedle dávno ukončeného vývoje je dalším problémem omezená funkcionalita htmltmpl, kde mi chybí zejména podpora dědičnosti šablon. Podporována je sice inkluze šablon, ta ale může dědičnost nahradit jen částečně. Moderní šablonovací systémy navíc nabízí další přidanou hodnotu v podobě různých formátovacích funkcí a filtrů, díky kterým má programátor daleko větší flexibilitu při prezentaci dat. Dalším nedostatkem htmltmpl je, dle mého názoru, mírně nepřehledná syntaxe. Použití špičatých závorek pří deklaraci htmltmpl výrazu není úplně nejšťastnější. Špičaté závorky jsou totiž také součástí HTML syntaxe. Při letmém pohledu do zdrojového kódu může HTML splývat s htmltmpl kódem.

Zdrojový kód 2.1: Ukázka kombinace HTML a htmltmpl

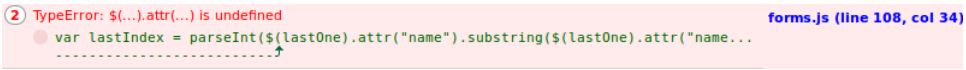
```
<TMPL_ELSE>
   <TMPL_IF exception>
      <TMPL_INCLUDE _exceptionMessage.tmpl>
   <TMPL_ELSE>
      <form action="" id="delete" method="post">
         <div>
            <input type="hidden" name="delete" value="true" />
         </div>
         <a href="?page=edit&amp;id=<TMPL_VAR id>">storno</a>
      </form>
   </TMPL_IF>
</TMPL_IF>
```

Je správné, že je v současném řešení použit alespoň nějaký způsob separace prezentace od logiky. Nicméně si myslím, že nastal čas současný šablonovací systém vyměnit za modernější řešení.

#### 2.5.8.2 Javascriptové chyby

Používání současné webové aplikace v určitých situacích vyvolá JS chyby. Standardně po vyvolání JS chyby se další kód v daném bloku přestane provádět, a to má za následek zamrznutí některých ovládacích prvků UI.

Není například ošetřena mezní hodnota, kdy se ve vícepoložkové sekci v některém z HTML formulářů nenachází žádná položka. Odebrání poslední položky a opětovné přidání nové způsobí chybu, viz 2.5. To znemožní další přidávání položek do této sekce.




Obrázek 2.5: JS chyba při přidání položky do prázdné vícepoložkové sekce

Další chyba se vyvolá při každém příchodu na hlavní stránku. Jedná se o použití nedefinovaných proměnných při řazení zobrazovaných seznamů záznamů. Zde je náprava velmi jednoduchá. Oprava předchozí chyby už vyžaduje větší zásah.

#### 2.5.8.3 Adresářová struktura

Dle mého názoru je špatně zvolená adresářová struktura systému AKRMAT, konkrétně umístění celého REST API společně s webovou aplikací, která je oddělena pouze podadresářem `aplikace`.

Pokud se nyní uživatel potřebuje dostat do webové aplikace, musí ještě do URL za adresu serveru, na kterém AKRMAT běží, přidat `/aplikace`. Mělo by dojít k automatickému přesměrování na reálnou URL `/aplikace/index.php`. Toto přesměrování z nějakého důvodu ale nefunguje v každém prohlížeči. V kořenu serveru běží AKRMAT REST API, pokud tedy nedojde ke správnému přesměrování, je HTTP požadavek interpretován v rámci REST API a vrátí se chyba neexistujícího zdroje `/aplikace`. To samé platí, pokud jde uživatel na stránku, která neexistuje. Opět je požadavek interpretován v rámci REST API.

#### 2.5.8.4 Struktura URL

V aplikaci není využit potenciál modulu `mod_rewrite` serveru Apache. URL v rámci webové aplikace jsou dlouhé a těžko zapamatovatelné.

**Příklad** Uživatel chce zobrazit životopis Ing. Martina Kohlíka. URL této stránky vypadá následovně:

```
/aplikace/index.php?page=edit&id=zivotopisy/kohlimar
```

Pomocí modulu `mod_rewrite` může být tato URL zjednodušena na následující tvar:

```
/aplikace/zivotopisy/kohlimar
```

#### 2.5.8.5 Uživatelské rozhraní

Současný vzhled uživatelského rozhraní není moderní a rozhraní samotné obsahuje několik problémů. Na druhou stranu je velmi jednoduché a nelze říci, že je nepoužitelné. Nicméně se určitě vyplatí přemýšlet nad návrhem nového.

Jedná se o stejný design UI jako v jiných školních systémech, např. Anketa FIT a Úvazkostroj. Ukázku lze vidět na obrázku D.4. Chyby uživatelského rozhraní aplikace Úvazkostroj jsem analyzoval ve své bakalářské práci v roce 2014.[7] Některé chyby jsou sdílené oběma aplikacemi. V následujících odstavcích je popsáno několik nejviditelnějších.

V rámci UI je ve vnitřní sekci použit barevný přechod ze světlé do tmavší modré barvy. Ve spodních částech stránky se dostává uživatel do situace, kdy musí číst černé písmo na tmavě modrém pozadí, což není barevná kombinace, která by byla příjemná pro čtení. Tento barevný přechod je navíc vytvořen pomocí obrázků. Několik let už je dostupný CSS3 atribut `linear-gradient` a stejný barevný přechod tak dnes je možné vytvořit jedním řádkem CSS kódu.

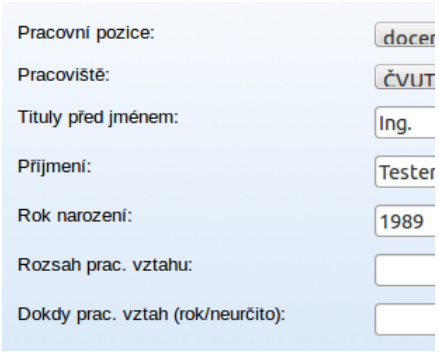
Další drobností je chyba zobrazení titulku stránky. Například ve formuláři životopisu se do titulku stránky dostane i neinterpretovaný HTML kód, viz obrázek 2.6.


Obrázek 2.6: HTML kód v titulku stránky

Samotné formuláře obsahují také několik problémů. Na Linuxu mají v prohlížečích Firefox i Chrome vstupy typu `<select>` malou výšku, takže se do nich nevejde celý text, viz obrázek 2.7.

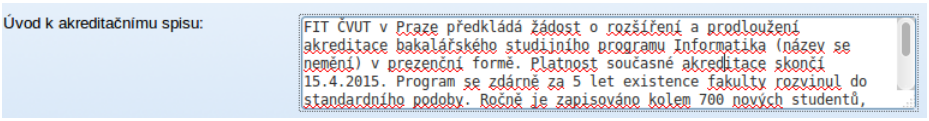

Obrázek 2.7: Nedostatečná výška vstupů typu select

Popisky u formulářových vstupů jsou umístěny vlevo, navíc jsou ještě zarovnány doleva. Uživatel tedy ve většině případů nemůže jedním pohledem zachytit text popisku a zároveň obsah vstupu, viz obrázek 2.8.




Obrázek 2.8: Doleva zarovnané popisky u vstupů

Ve formulářích se také nachází mnoho vstupních polí typu `<textarea>`, zejména ve formuláři studijního programu. Většinou tyto vstupy obsahují velmi dlouhé texty, které se do těchto vstupů nevejdou a uživatel musí při čtení nebo editaci scrollovat myší, viz obrázek 2.9. Vstupy `textarea` lze v každém prohlížeči roztahovat, ale poté většinou začnou zasahovat do jiných prvků formuláře.




Obrázek 2.9: Nedostatečná výška vstupů typu textarea

Kapitola 3

# Použité technologie

Cílem této kapitoly je stručně popsat nejdůležitější technologie a technologické principy, které jsou součástí implementace současné i nové verze systému AKRMAT.

## 3.1 Webový server Apache

V současnosti celý systém AKRMAT běží na webovém serveru Apache. Jedná se o populární webový server implementovaný v jazyce C/C++, který může běžet na širokém spektru operačních systémů. Podporována je většina OS založených na bázi Unixu, dále OS X i Windows.

Apache je rozšiřitelný pomocí modulů, které přidávají další rozmanitou funkcionalitu. Moduly jsou vyvíjeny jak vývojáři samotného webového serveru, tak i třetími stranami. Pomocí Apache Module API si lze vytvořit modul vlastní nebo modifikovat nějaký stávající.

Moduly mohou například přidávat podporu programovacích jazyků. Nejpoužívanější jsou moduly pro jazyky PHP, Python nebo Perl. Další důležité moduly jsou `mod_ssl` pro podporu bezpečnostních protokolů SSL[9] a TLS[10] a `mod_rewrite` pro přepisování URL na jiné URL.[8][9]

### 3.1.1 Modul mod_rewrite

Modul umožňuje přepisování URL do jiných tvarů podle pravidel specifikovaných v konfiguračním souboru `apache2.conf` (`httpd.conf` na Windows) nebo v souboru `.htaccess` přímo v adresáři webové aplikace.

Při vytváření pravidel se používají regulární výrazy pro určení tvaru vstupních URL. Modul za běhu parsuje příchozí URL a kontroluje, zda pro ni neplatí nějaké pravidlo. Pokud ano, URL je dle pravidla přepsána a pokračuje se kontrolou dalších pravidel nebo lze přepisování explicitně ukončit.[10]

[9] Secure Sockets Layer
[10] Transport Layer Security

Díky tomuto modulu lze přepisovat nevzhledné, dlouhé nebo špatně zapamatovatelné URL na kratší a lépe zapamatovatelné varianty. Například URL

```
http://example.com/user.php?id=15587
```

lze přepsat na

```
http://example.com/user/15587/
```

Pravidla pro výše uvedený přepis URL vypadají následovně:

```
RewriteEngine on
RewriteRule ^user/(\w+)/?$ user.php?id=$1
```

První řádek povoluje použití přepisovacího systému a druhý řádek je samotné pravidlo. Prvním parametrem pravidla je tvar příchozí URL popsaný pomocí regulárního výrazu. Druhým parametrem je přesný výstup po přepsání. Zde lze vidět, že přepis ve skutečnosti probíhá obráceně, než je uvedeno v příkladu výše. Z krátké URL ve tvaru `user/15587/` dostáváme delší a složitější URL ve tvaru `/user.php?id=15587`. Kratší tvar je tím, který zadává uživatel např. do prohlížeče, ale tomuto tvaru cílová webová aplikace nerozumí. Proto musí být přepsán na tvar, který už bude webové aplikaci vyhovovat. Díky tomu dojde k uspokojení obou stran. Celý proces přepisování je navíc před koncovým uživatelem skryt.

### 3.1.2 Virtuální servery

Apache také obsahuje podporu tzv. virtuálních serverů. Jedná se o způsob jak provozovat více webových stránek/aplikací v rámci jedné instalace serveru. Apache podporuje dva typy virtuálních serverů.

- tzv. **IP-based** virtuální servery
- tzv. **name-based** virtuální servery

**IP-based** Tento způsob je založený na reprezentaci jednotlivých virtuálních serverů pomocí vlastních IP adres. Tudíž je nutné mít k dispozici tolik různých IP adres, kolik je plánováno virtuálních serverů. Je to složitější způsob, proto se většinou používá druhá varianta: name-based virtuální servery.

**Name-based** Zde se nemusí řešit více rozdílných IP adres, stačí pouze jediná. V tom je velká výhoda oproti předchozímu způsobu. Tento způsob závisí na tom, že klient v rámci HTTP požadavku specifikuje jméno serveru v HTTP hlavičce `HOST`. Dále je potřeba nakonfigurovat DNS[11] server tak, aby namapoval jméno virtuálního serveru na správnou IP adresu, nebo přidat ono mapovací pravidlo do souboru `hosts` v operačním systému. Je také nutné upravit konfigurační soubor serveru Apache `vhosts.conf` tak, aby rozpoznal požadovaná jména virtuálních serverů.[11]

Příklad direktivy v souboru `vhosts.conf` pro jeden virtuální server:

```
<VirtualHost *:80>
    DocumentRoot "/www/example2"
    ServerName www.example.org
</VirtualHost>
```

První řádek obsahuje pravidlo pro filtrování možností pomocí IP adres. V příkladu je použit žolík ve formě znaku „*“. Z toho důvodu pravidlo splňují všechny IP adresy na portu 80, a tudíž se prakticky toto filtrování neprovádí. Dalším krokem je porovnání hodnoty v hlavičce HOST příchozího HTTP požadavku s hodnotou v pravidle `ServerName`. Pravidlo `DocumentRoot` pak odkazuje na fyzické umístění webové stránky/aplikace v systému souborů.

## 3.2 XML

XML je jednou z nejdůležitějších technologií, které systém AKRMAT využívá. Téměř veškerá data jsou totiž uložena ve formě XML dokumentů v XML nativní databázi Sedna.

XML je, jak už z názvu může být zřejmé, rozšiřitelný značkovací jazyk. Obsah v rámci XML je tvořen pomocí značek (tzv. tagů), které mohou být párové i nepárové. Používá se zejména k serializaci nebo ukládání dat. Jazyk je hojně rozšířen a má podporu v širokém spektru programovacích jazyků. Další jeho výhodou je, že je čitelný jak pro člověka, tak pro stroj. XML dokumenty navíc mohou být bez problémů editovány v jednoduchém textovém editoru.[12]

Pokud je potřeba určit, jaké elementy budou v konkrétním XML dokumentu povoleny, jsou na výběr dvě možnosti validace: DTD[12] a XML schéma. DTD je považováno v dnešní době za zastaralé a často velmi nepřehledné, proto se většinou používá XML schéma.

[11] Domain Name System
[12] Document Type Definition

### 3.2.1 Syntaxe XML

Kódování XML dokumentu je vždy ve formátu Unicode, často UTF-8. Pokud XML dokument dodržuje určitá pravidla, může být označen jako správně sestavený (tzv. well-formed).[13] Některá z těchto pravidel jsou:

- XML dokument musí mít právě jeden kořenový element.
- Neprázdné XML elementy musí mít koncovou značku, např. `<element>Obsah</element>`.
- Elementy mohou být do sebe vnořeny, nesmí se nicméně překrývat.
- Prázdné elementy musí být zakončeny pomocí `/>`, např. `<img src="obrazek.jpg" />`.
- Hodnoty atributů musí být v uvozovkách.
- Syntaxe je case sensitive, záleží tedy na velkých a malých písmenech. Např. konstrukce `<Element>Obsah</element>` není validní.

## 3.3 XML schéma

XML schéma poskytuje soustavu specifikací a pravidel, podle kterých se určuje struktura a obsah XML dokumentu. Pokud jsou tato pravidla dodržena, lze konkrétní XML dokument prohlásit za validní podle konkrétního XML schématu. Samotné XML schéma používá soubory s příponou `.xsd`, které jsou také XML dokumenty, takže pro ně platí veškeré informace z předchozí sekce. Pomocí XML schématu mohou být určeny následující vlastnosti XML dokumentu:[14]

- Elementy a jejich atributy
- Vnoření elementů
- Pořadí a počet elementů
- Obsah a jeho datový typ
- Výchozí a fixní hodnoty elementů

Alternativou k XML schématu je dnes už zastaralé DTD. XML schéma oproti DTD má mnoho výhod:

- XML schéma je současně XML dokument.
- XML schéma podporuje datové typy.

- XML schéma podporuje jmenné prostory.
- XML schéma podporuje určení pořadí elementů.
- XML schéma je rozšiřitelné.

V systému AKRMAT se XML schéma používá ke kontrole integrity dat, která jsou ukládána do databáze. Ukázka zdrojového kódu části XML schématu:

Zdrojový kód 3.1: Část souboru programy.xsd pro validaci entity Program

```
<!-- název studijního programu -->
<xs:element name="nazev_programu" type="xs:string"/>
<!-- kód studijního programu - jedná se o~jednoznačný identifikátor!
    -->
<xs:element name="kod_programu" type="xs:string"/>
<!-- typ studijního programu -->
<xs:element name="typ_programu">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="bakalářský"/>
         <xs:enumeration value="magisterský"/>
         <xs:enumeration value="navazující magisterský"/>
         <xs:enumeration value="doktorandský"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```

## 3.4 XML nativní databáze Sedna

XML nativní databáze využívá jazyka XML v procesu manipulace s daty. Na rozdíl od tzv. XML-enabled databáze, která pouze přijímá XML na vstup a poté ho transformuje do jiných struktur pro samotné uložení, XML nativní databáze přímo ukládá XML dokumenty.[15]

Pro projekt AKRMAT byla vybrána XML nativní databáze Sedna. Výběrem databáze se detailně zabýval Bc. Luboš Růžička ve své diplomové práci. Sedna byla vybrána z důvodu podpory ACID[13] transakcí a dotazovacího jazyka XQuery. Dále je k ní dostupné API pro jazyk PHP, ve kterém je současná verze systému AKRMAT implementována.[2]

## 3.5 PHP

Celý systém AKRMAT je v současnosti na serverové straně implementován v jazyce PHP. Celým názvem Hypertext-Preprocessor, je populární objektově

[13] Atomicity, Consistency, Isolation, Durability

orientovaný skriptovací jazyk pro vývoj dynamických webových stránek nebo aplikací.

### 3.5.1 Výběr programovacího jazyka

Volba PHP pro implementaci současné verze systému AKRMAT se jeví jako dobrá. Jazyk je jednoduchý, má přehlednou syntaxi odvozenou z jazyků C nebo Java a nepotřebuje žádné běžně nedostupné prostředí k nasazení. Důležité je, že pro něj existuje také API k XML nativní databázi Sedna.

Jelikož jsem se rozhodl rozdělit celý systém na dvě části: REST API a webovou aplikaci, která bude zcela nově implementována (důvody jsou uvedeny např. v sekcích 2.5.8.3 a 2.5.6), nabízí se možnost právě pro webovou aplikaci vybrat jiný programovací jazyk. Lze uvažovat např. o jazyku C#. Na rozdíl od PHP, mám nicméně s tímto jazykem velmi malé zkušenosti. Navíc budou nejspíše obě části běžet na stejném stroji a bylo by vhodné, aby nemusela být výrazně upravována jeho konfigurace při nasazení nové verze. Proto i nadále v rámci webové aplikace bude na serverové straně použit programovací jazyk PHP.

### 3.5.2 Výběr PHP verze

Důležité je také zmínit, na které verzi PHP bude probíhat nový vývoj. V současné době běží systém AKRMAT na PHP 5.3. Většina nových knihoven, které hodlám při implementaci použít, podporují také minimálně verzi 5.3. Nejnovější verze knihovny Guzzle nicméně vyžaduje minimálně verzi 5.4. Obecně se vyplatí používat novější verze PHP, které mají opravené chyby nalezené v předchozích verzích a často obsahují navíc i novou funkcionalitu. Rozhodl jsem se prozkoumat na jakou nejnovější verzi by mohlo být PHP bezpečně aktualizováno, aniž by byla ovlivněna funkčnost systému.

Většina použitých knihoven je podporována i v druhé nejnovější verzi PHP 5.6 (v době psaní diplomové práce, duben 2016, je vydána už i verze 7). Výjimkou je knihovna poskytující API pro databázi Sedna. Pro verzi 5.5 se mi tuto knihovnu podařilo zkompilovat ve funkčním stavu, pro verzi 5.6 už nikoli. Bylo tedy rozhodnuto, že systém AKRMAT bude podporovat nově minimálně PHP 5.4, ideálně PHP 5.5.

### 3.5.3 Výběr PHP frameworku

Prvotní plán zahrnoval vybrat vhodný PHP framework pro implementaci nové webové aplikace. Nicméně jsem si uvědomil, že žádný neznám úplně na 100 % a měl jsem obavy, že v průběhu vývoje narazím na nějaký složitý problém, který bude příliš časově náročné řešit. Chtěl jsem mít co největší kontrolu nad zdrojovým kódem, aby případné řešení problémů probíhalo co nejrychleji. Rozhodl jsem tedy žádný ucelený PHP framework nepoužít a pouze využít několika málo knihoven.

### 3.5.4 Použité PHP knihovny

Při implementaci nové verze webové aplikace bude použito několik nových PHP knihoven, které v současné verzi nefigurují. Proto je v této sekci zmíním a pokusím se přiblížit důvody, proč jsem se pro ně rozhodl.

#### 3.5.4.1 Klein

Klein[14] je open source[15] PHP router, tedy knihovna, která zajišťuje směrování (routing). Termín směrování v rámci webových aplikací znamená mapování URL na části zdrojového kódu, které se po příchodu na tuto URL provedou. Routing je nedílnou součástí architektury MVC[16], kde jedna URL je většinou namapována na jednu metodu (akci) určité třídy (Controlleru). Klein tuto funkcionalitu poskytuje s využitím modulu `mod_rewrite` webového serveru Apache, který všechny příchozí požadavky směruje na hlavní stránku, např. `index.php`, kde už aplikační logika routeru Klein zařídí zavolání požadované metody.

Jednotlivá směrovací pravidla se musí nejprve registrovat voláním metody `respond`. To může být provedeno následujícím způsobem:

```
// inicializace knihovny
$router = new Klein();
$router->respond($method, $url, $callback);
```

Parametr `$method` určuje, jaká HTTP metoda je použita v požadavku, `$url` obsahuje samotnou URL požadavku a `$callback` obsahuje definici funkce, která se zavolá po obdržení HTTP požadavku. Na hlavní stránce `index.php` už jen stačí zajistit volání metody `dispatch`.

```
$router->dispatch();
```

Tato metoda zařídí parsování URL pomocí regulárních výrazů a následné přiřazení správnému směrovacímu pravidlu.

#### 3.5.4.2 Guzzle

Posílat HTTP požadavky z PHP kódu lze pomocí vestavěné knihovny libcurl. Ta je sice bez problémů použitelná, nicméně správné nastavení a odeslání HTTP požadavku zahrnuje někdy i více než 10 volání různých funkcí této knihovny. Je zřejmé, že využití nějaké třídy obalující tuto funkcionalitu je klíčem k úspěchu. Řešením může být právě knihovna Guzzle[17]. Jedná se o HTTP klienta, který obaluje v PHP vestavěnou knihovnu libcurl a poskytuje rozhraní

[14] `https://github.com/klein/klein.php`
[15] software s otevřeným zdrojovým kódem
[16] Model-View-Controller
[17] `https://github.com/guzzle/guzzle`

vyšší úrovně pro odesílání HTTP požadavků. Použití této knihovny umožní z webové aplikace pohodlně posílat HTTP požadavky jak na AKRMAT REST API, tak na REST API systémů KOS a VVVS.

Zdrojový kód 3.2: Příklad použití knihovny Guzzle

```
// nejprve je nutné vytvořit samotného klienta
// lze mu ihned nastavit základ URI
$client = new Client([
   "base_uri" => $baseUri
]);

// poté už lze posílat HTTP požadavky
// do druhého parametru se vkládají další informace
// např. samotný datový obsah metody POST či HTTP hlavičky
$response = $client->post($uri, [
   "form_params" => [
      "grant_type" => "client_credentials"
   ],
   "headers" => [
      "Authorization" => "Basic ".$this->getAuthorizationInfo()
   ]
]);

// obdržená data v~podobě stringu lze získat následovně
$data = $response->getBody(true);
```

#### 3.5.4.3 Kint

Při ladění PHP kódu často vývojáři používají ladící výpisy, které se většinou vytváří následující klasickou konstrukcí nebo její obdobou:

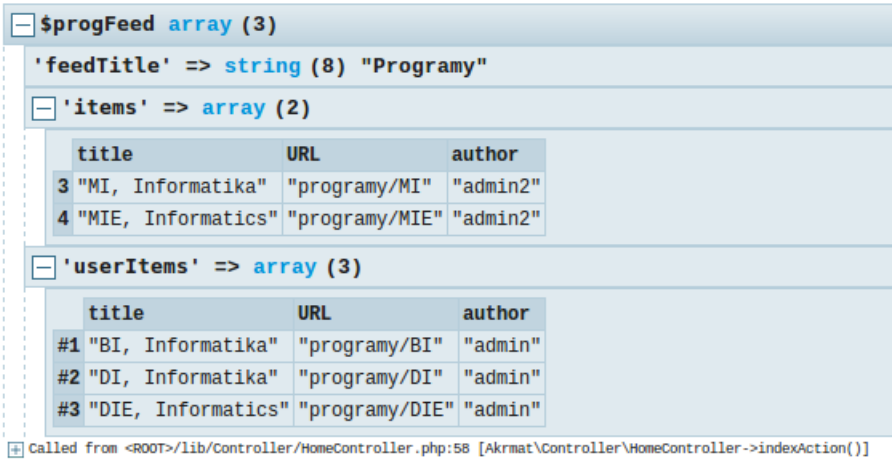
```
echo "<pre>"; var_dump($variable); die;
```

Psát tento řádek kódu vždy, když je potřeba ladit nějakou proměnnou, není úplně pohodlné. Výsledný výpis navíc není nijak zvlášť dobře formátovaný a nemusí být vždy úplně snadno čitelný.

Knihovna Kint[18] poskytuje pohodlnější a rychlejší vytváření ladících výpisů, které jsou navíc lépe formátované, čitelnější a obsahují další užitečné informace. Umí také vypsat obsah zásobníku, ve kterém lze přehledně vidět posloupnost volání jednotlivých metod.

Výše uvedený často používaný řádek ladícího zdrojového kódu lze pomocí této knihovny zapsat jednoduše `ddd($variable)`. Pro pokračování běhu zdrojového kódů po výpisu se použije `d($variable)`. Příklad ladícího výpisu knihovny Kint lze vidět na obrázku 3.1.

[18] `http://raveren.github.io/kint/`




Obrázek 3.1: Ladící výpis pomocí knihovny Kint

## 3.6 REST

REST neboli Representational State Transfer je webová softwarová architektura. Definuje rozhraní, přes které klienti mohou přistupovat ke zdrojům (datům). Zakládá se na několika principech.[5][16] Těmi nejdůležitějšími jsou:

- **Reprezentace zdrojů** – Jednotlivé zdroje jsou identifikovány pomocí URI. Klient při požadavku o získání zdroje nezíská zdroj samotný, ale pouze jeho reprezentaci, např. ve formátu XML nebo JSON.
- **Manipulace se zdroji** – Klient musí mít k dispozici veškerá metadata potřebná k případné modifikaci zdroje, pokud k tomu má oprávnění.
- **Bezstavovost** – Server nesmí udržovat žádné informace o stavu. Všechny informace potřebné k rekonstrukci stavu musí být obsaženy v rámci každého HTTP požadavku, např. v těle nebo v HTTP hlavičkách.
- **Client-server architektura** – Pomocí uniformního rozhraní dochází k separaci klienta a serveru. Server se například nezajímá o stav klienta a jeho uživatelské rozhraní a klient zase o manipulaci s daty v databázi. Znamená to, že obě části mohou být vyvíjeny individuálně a v případě potřeby okamžitě vyměněny.

Rozhraní REST je použito v systému AKRMAT pro umožnění přístupu k dokumentům uloženým v databázi Sedna. Toto rozhraní navrhli v rámci svých diplomových prací Bc. Luboš Ružička a Bc. Jakub Jambor.[2][3] Dalším místem, kde se v kontextu systému AKRMAT technologie REST vyskytuje, jsou služby KOS API a VVVS API. Obě služby poskytují rozhraní REST pro přístup ke svým datům.

### 3.6.1 HTTP metody

**GET** Metoda GET slouží k samotnému získání reprezentace zdroje. Za URL lze přidat otazník a tzv. query parametry, které typicky v rámci implementací REST API slouží k filtraci, či řazení dat na výstupu. Při úspěchu vrací HTTP odpověď `200 OK`.

**POST** Metoda POST slouží k vytvoření nového zdroje. Data o zdroji jsou umístěna v těle HTTP požadavku. HTTP odpovědí bývá `201 Created` s hlavičkou `Location` obsahující odkaz na nově vytvořený zdroj.

**PUT** Je velmi podobná metodě POST. Data se posílají stejným způsobem, ale metoda PUT slouží k modifikaci existujícího zdroje. HTTP odpovědí na PUT bývá `200 OK` s upravenou reprezentací zdroje.

**DELETE** Slouží ke smazání existujícího zdroje. Při úspěchu vrací `200 OK` s reprezentací odstraněného zdroje, nebo `204 No Content` bez reprezentace, většinou pro úsporu síťových prostředků.

## 3.7 OAuth 2.0

Jedná se o druhou verzi otevřeného standardu pro autorizaci OAuth, který se stal prakticky standardem zabezpečení webových služeb typu REST. Hlavní jeho výhodou je, že třetí straně se může umožnit omezený přístup k HTTP službě, aniž by ji musely být předány veškeré přístupové údaje a s nimi prakticky plný přístup ke službě. OAuth 2.0 podporuje také tzv. scopes. Mohou se jimi regulovat přesné pravomoci jednotlivých aplikací třetích stran, kterým byl povolen přístup k datům. Například mobilní aplikaci se posktyne přístup k osobním informacím v profilu na Facebooku, ale nikoli už k samotným příspěvkům.[17]

V procesu autorizace vystupují čtyři role:

- **Resource owner** – typicky uživatel
- **Resource server** – poskytovatel zdroje
- **Client** – aplikace, která požaduje přístup k obsahu
- **Authorization server** – server, který klientovi vydává přístupový token

OAuth 2.0 podporuje čtyři typy tzv. grant types (povolení přístupu):

- **Client credentials** – Aplikace se autentizuje sama za sebe.

- **Authorization code** – Aplikace se autentizuje za uživatele, který k tomu musí dát explicitní souhlas.
- **Implicit** – Podobné jako Authorization code, ale nepodporuje obnovování tokenů. Klient v tomto případě není webový server, ale běží např. v prohlížeči.
- **Resource owner password credentials** – Aplikace získá kompletní autentizační informaci. Používá se v případě, že panuje naprostá důvěra mezi aplikací a serverem poskytujícím službu, např. v případě, že jsou vyvíjeny stejnou firmou.

Obě služby pro import dat, které se budou v systému AKRMAT využívat, KOS API a VVVS API, podporují autorizaci pomocí standardu OAuth 2.0. V tomto případě bude o přístup žádat webová aplikace sama za sebe, a proto bude reprezentovat role Client i Resource owner. Využije se zde autorizace pomocí Client credentials. Proces tohoto typu autorizace je popsán v následující posloupnosti kroků. Jeho diagram lze vidět na obrázku 3.2.

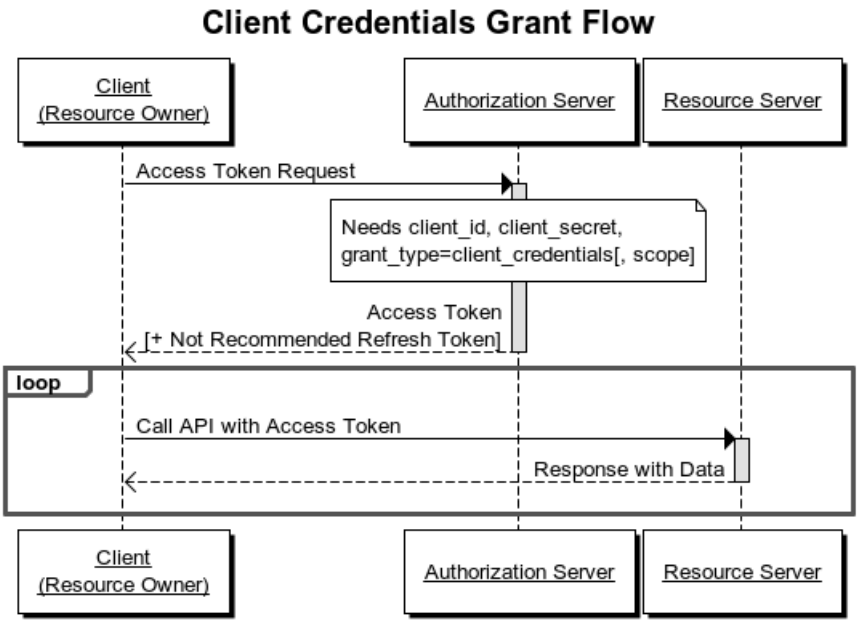
1. Aplikace (Client/Resource Owner) požádá o přístupový token autorizační server (Authorization server). Do HTTP požadavku vloží své client ID a client secret. Jsou to dva řetězce, pomocí kterých autorizační server pozná identitu aplikace.
2. V případě, že je aplikace oprávněna obdržet přístupový token, vrátí ho autorizační server v těle odpovědi.
3. Tento token se přiloží do HTTP požadavku o data od poskytovatele služby (Resource server).
4. V případě, že je přístupový token validní, jsou v odpovědi vrácena požadovaná data.

Přesný proces autorizace pomocí OAuth 2.0 pro konkrétní systémy KOS API a VVVS API je popsán v implementační kapitole v sekci 5.5.1.

## 3.8 MVC

MVC je architektonický softwarový vzor, který rozděluje datový model, uživatelské rozhraní a řídicí logiku do tří samostatných částí: Model, View a Controller. Tento vzor se stal nesmírně populárním ve vývoji webových aplikací. Diagram MVC lze vidět na obrázku 3.3.

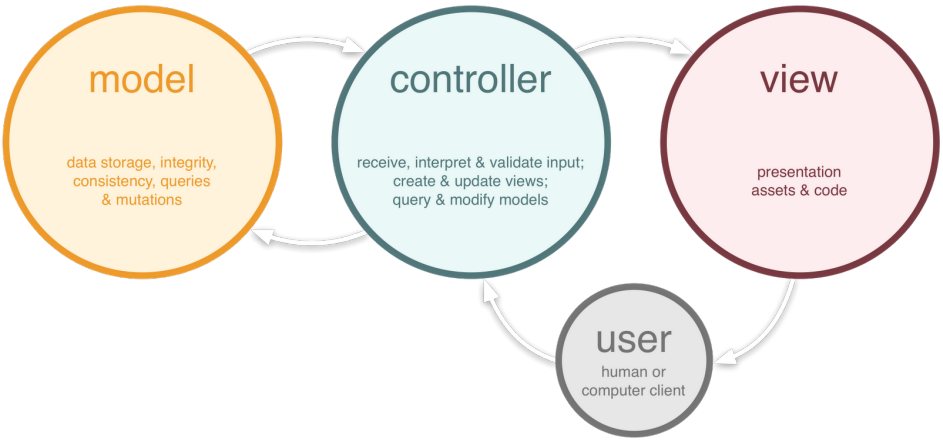
**Model** Reprezentuje datovou vrstvu a obsahuje logiku týkající se manipulace s daty, jako je např. ukládání a dotazování do databáze nebo datové transformace. Tato data předává na vyžádaní Controlleru.




Obrázek 3.2: Autorizace pomocí Client credentials standardu OAuth 2.0 [17]

**View** Spadá sem veškerá prezentační logika. View se stará o prezentaci dat, získaných od Controlleru, uživateli.

**Controller** Controller je prostředníkem mezi Modelem a View. Obsahuje veškerou aplikační (business) logiku a reaguje na události vyvolané uživatelem. Manipuluje pomocí Modelu s daty a těmito daty aktualizuje View.




Obrázek 3.3: Schéma architektury MVC [18]

V současné verzi systému AKRMAT není tento vzor použit. Využil jsem jej až při návrhu architektury nové webové aplikace.

## 3.9 Výběr šablonovacího systému

Cílem šablonovacího systému je umožnit snadné vytváření HTML stránek s dynamickým obsahem. Veškerá prezentace (HTML kód) se separuje od aplikační logiky do samostatných šablon, ve kterých se kombinuje HTML a jazyk použitého šablonovacího systému. Samotné PHP je možné svým způsobem použít také jako šabolonovací systém. Nicméně kombinace PHP a HTML kódu ve velké míře může vést ke značné nepřehlednosti zdrojového kódu šablony. Syntaxe specializovaných šablonovacích systému bývá navržena tak, aby tento problém nenastal. Tyto systémy implementují i přidanou funkcionalitu, např. dědičnost šablon, filtry, datové transformace nebo bloky.

Současná webová aplikace systému AKRMAT využívá šablonovací systém htmltmpl. Jak jsem už dříve popsal v sekci 2.5.8.1, je tento systém velmi zastaralý, má nepřehlednou syntaxi a chybí mu funkcionalita, kterou dnešní moderní šablonovací systémy standardně mívají. Pro vývoj nové verze aplikace jsem se tedy rozhodl vybrat nový šablonovací systém.

Rozhodování panovalo mezi šablonovacím systémem Smarty, který jsem například použil v rámci implementace své bakalářské práce[7] a šablonovacím systémem Twig, který je součástí PHP frameworku Symfony, ale je i dostupný jako samostatná knihovna. S tímto systémem mám také nějaké zkušenosti.

Po pečlivém zvážení situace jsem zvolil Twig, který řeší všechny problémy identifikované u htmltmpl. Při poslední práci s Twigem jsem byl velmi spokojen. Naopak Smarty, ač zajisté také kvalitní systém s dlouhou vývojovou tradicí, mě při poslední práci s ním trochu zklamal. Narazil jsem totiž na několik chyb, které jsem nebyl schopen opravit. Jednalo se například o samovolnou změnu rozložení stránky. Jsem přesvědčen, že tyto chyby už nejspíše byly odstraněny v novějších verzích, ale přesto jsem se rozhodl dát šanci konkurenčnímu Twigu.

Zdrojový kód 3.3: Ukázka šablony a syntaxe šablonovacího systému Twig

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <ul id="navigation">
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
        </ul>
```

```
        <h1>My Webpage</h1>
        {{ a_variable }}
    </body>
</html>
```

## 3.10 HTML5

V rámci nové webové aplikace bude HTML kód implementován ve standardu HTML5. Je to nejnovější specifikace značkovacího jazyka HTML, která obsahuje mnoho výhod a vylepšení oproti starším specifikacím XHTML 1.0 nebo HTML 4.01. Některé z nich jsou:

- **Nové sémantické elementy** - HTML5 umožňuje použití nových elementů s vyšším sémantickým významem než standardní HTML elementy `<div>` nebo `<span>`. Těmito novými elementy jsou např. `<header>`, `<footer>`, `<nav>`, `<section>` nebo `<main>`. Zlepšují mimo jiné čitelnost zdrojového kódu.
- **Validace formulářů** - HTML5 zavádí možnost validovat formuláře přímo v prohlížeči bez použití JavaScriptu. Validace je možná díky novým typům formulářových polí jako `email`, `search`, `number` nebo `date`. Dále jsou k dispozici nové atributy `placeholder` nebo `required`.
- **Vlastní atributy** - Nově lze zcela validním způsobem zařídit, aby HTML element nesl navíc nějakou informaci, která je poté dostupná v JS kódu. Docílí se toho přidáním vlastního atributu elementu s prefixem `data-`.

## 3.11 Javascript

JavaScript je dnes hlavním programovacím jazykem na webu. Používá se většinou k přidání interaktivity do webové stránky. Může reagovat na události po provedení různých akcí uživatelského rozhraní, např. kliknutí na tlačítko, vyplnění textového pole nebo odeslání formuláře.

Ačkoli lze JavaScript používat samostatně, existuje mnoho knihoven a frameworků, které vývojářům poskytují nespočet užitečných funkcí. Například usnadňují práci s DOM nebo použití techniky AJAX[19].[7]

### 3.11.1 JQuery

Podle [19] je knihovna jQuery tou nejpopulárnější. Jednou z největších výhod je její kompatibilita mezi prohlížeči. JQuery se sama stará o to, aby se stránka chovala stejně ve všech prohlížečích a operačních systémech.

[19] Asynchronous JavaScript and XML

JQuery umožňuje pro vybírání elementů z DOMu použít CSS selektory, včetně nových ze specifikace CSS3. Vývojář, který umí CSS, umí zároveň vybrat DOM element pomocí jQuery a naopak.

Další výhodou je snadná rozšiřitelnost. Pro jQuery existuje mnoho volně dostupných pluginů. Je navíc velice snadné si vytvořit plugin vlastní.[7]

### 3.11.2 Bootstrap

Bootstrap je populární framework, který poskytuje základ pro rapidní vývoj webových stránek a aplikací v HTML, CSS a JavaScriptu. Součástí Bootstrapu jsou různé komponenty a funkcionality, které velmi usnadňují a urychlují vývoj.

Většina JS komponentů Bootstrapu funguje jako pluginy do knihovny jQuery. Jedná se např. o zabalující se sekce (komponenta collapse.js) nebo vyskakovací okna (komponenta modal.js).

Většinu komponent lze inicializovat dvojím způsobem. První způsob spočívá v přidání speciálních atributů do HTML elementů s prefixem `data-`, například `data-toggle="toggle"` nebo `data-placement="top"`. Druhým způsobem je inicializace přímo v JavaScriptu.[7]

## 3.12 CSS3

CSS3 je nejnovější specifikací kaskádových stylů (CSS). Je plně zpětně kompatibilní a přináší nové techniky pro implementaci designu webových stránek. CSS3 umožňuje využívat mnoha nových selektorů pro výběr HTML elementů, dále přináší mnoho nových vlastností, díky kterým lze přímo v CSS implementovat animace, barevné přechody nebo stínování.

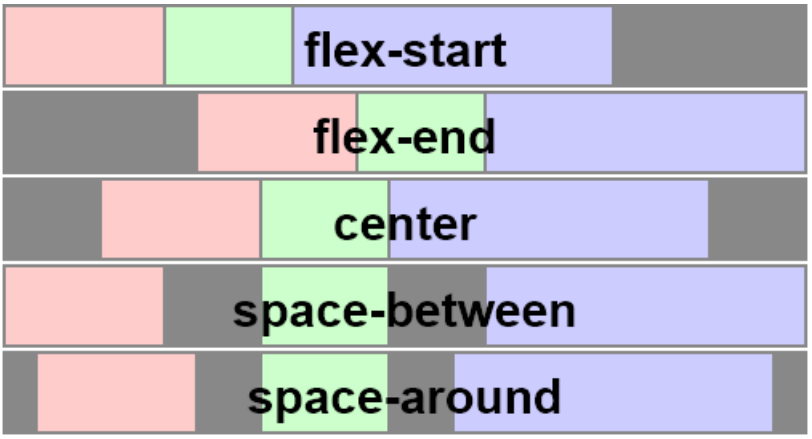
**Vlastní písma** Dříve bylo nutné na webu používat pouze úzkou skupinu fontů (písem), tzv. web-safe fonty. U nich se předpokládalo, že jsou na počítači klienta nainstalovány a ve většině případů tomu tak bylo. Web-safe fonty jsou např. Arial, Times nebo Helvetica.

Díky pravidlu `@font-face` lze nyní v CSS3 nastavit vlastní font, který nemusí být na klientském počítači nainstalován. V pravidle se určí cesta k souboru s fontem, který může být nahrán buď na serveru webové aplikace, nebo jinde na internetu.[7]

V novém návrhu webové aplikace plánuji použít dva druhy fontů, jejichž použití bude implementováno právě pomocí pravidla `@font-face`.

**Flexbox** Jedna z nových vlastností obsažená ve standardu CSS3, pomocí které HTML elementy mohou samy upravovat rozložení svého obsahu tak, aby byl co nejlépe vyplněn celý dostupný prostor.

Na obrázku 3.4 lze vidět jednotlivé módy flexboxu a jaký mají vliv na uspořádání obsahu uvnitř HTML elementu.




Obrázek 3.4: Módy distribuce obsahu CSS3 vlastnosti flexbox [20]

Tato technika se ve webové aplikaci AKRMAT může hodit zejména pro modifikaci vzhledu formulářů, kdy lze velmi jednoduše zařídit libovolný počet stejně širokých sloupců obsahujících formulářové vstupy.

Kapitola 4

# Návrh a metodika nového řešení

## 4.1 Požadavky

Informace použité k určení požadavků na vylepšení systému byly získány konzultacemi s vedoucím diplomové práce a osobními zkušenostmi nasbíranými prací se systémem. Následující požadavky lze klasifikovat jako funkční požadavky.

### 4.1.1 Požadavky na systém

- Systém bude ukládat studijní obory separátně mimo studijní programy.
- Systém bude evidovat zamykání dokumentu, včetně autora zámku a časové značky.
- Systém bude automaticky rušit platnost zámku dokumentů po uplynutí stanoveného časového intervalu.
- Systém bude kontrolovat uživatelská práva při veškerých činnostech.
- Systém umožní import dat o předmětech z KOS API.
- Systém umožní import dat o životopisech z KOS API.
- Systém umožní import dat o programech z KOS API.
- Systém umožní import dat o oborech z KOS API.
- Systém umožní import dat o publikacích z VVVS API.

### 4.1.2 Požadavky na uživatelské rozhraní

- Uživatel bude mít možnost výběru importovaných položek z KOS API.
- Rozhraní zobrazí výsledek importu z KOS API.

- Uživatel bude mít možnost importu publikací z VVVS API v rámci formuláře životopisu.
- Na hlavní stránce se vedle ostatních seznamů zobrazí i seznam oborů.
- Uživatel bude moci vytvářet nové obory.
- Uživatel bude moci editovat existující obory.
- Rozhraní poskytne uživatelům pohodlnější editaci elementů `textarea`.
- Rozhraní zobrazí uživatelům informaci o stavu zámku dokumentu na stránce s formulářem dokumentu.

### 4.1.3 Nefunkční požadavky

- Webová aplikace bude na serverové straně implementována v PHP.
- Webová aplikace poběží na webovém serveru Apache 2.
- Systém poskytne v rámci REST API rozhraní pro manipulaci s novou entitou Obor.

## 4.2 Nová adresářová struktura

Hlavní motivací je odstranění problému s chybným přesměrováním z webové aplikace do REST API popsaný v sekci 2.5.8.3. Vedlejší motivací je zlepšení struktury zdrojového kódu a zároveň jeho čitelnosti.

Celý systém se rozdělí na dvě samostatné části: webovou aplikaci a REST API. Pojem REST API nyní mírně zobecníme. V souladu s architekturou systému popsanou v sekci 2.2 do této části zahrneme nejen REST vrstvu, ale také XML a databázové vrstvy.

Obě části systému budou mít vlastní adresáře umístěné v kořenu serveru. Pokud tyto adresáře nazveme `api` a `aplikace`, URL mohou mít následující podobu: `akrmat.fit.cvut.cz/api` pro REST API a pro webovou aplikaci `akrmat.fit.cvut.cz/aplikace`.

Ještě lepším řešením by bylo pro každý adresář v rámci serveru Apache vytvořit vlastní virtuální server. Díky tomu by webová aplikace mohla být dostupná přímo na hlavní doméně `akrmat.fit.cvut.cz` a REST API na subdoméně `api.akrmat.fit.cvut.cz`.

## 4.3 Nová architektura webové aplikace

Z důvodu nekvality zdrojového kódu současné webové aplikace a pochybnostech o jeho snadné rozšiřitelnosti bylo rozhodnuto, že bude tato část navržena

a implementována úplně od začátku. Součástí toho je i návrh nové architektury.

Volba architektury byla velmi jednoduchá. V dnešní době mnoho webových aplikací využívá architektonického vzoru MVC, většinou implementovaného jako součást nějakého frameworku. Tento vzor rozděluje aplikaci na tři oddělené segmenty: Model, View, Controller a každý segment má svůj vlastní úkol.

Model má na starosti práci s daty, View prezentuje tato data uživateli a Controller zpracovává podněty od uživatele a příslušně na ně reaguje voláním vhodných metod, většinou v rámci části Model. Toto rozdělení na segmenty zajistí dobrou strukturu zdrojového kódu a separaci prezentační a aplikační (business) logiky.

Rozhodl jsem se nepoužit žádný framework, který by měl už MVC v sobě implementované. Více informací o tomto rozhodnutí je uvedeno v sekci 3.5.3. Proto je nutné navrhnout vlastní třídy realizující MVC architekturu.

### 4.3.1 Model

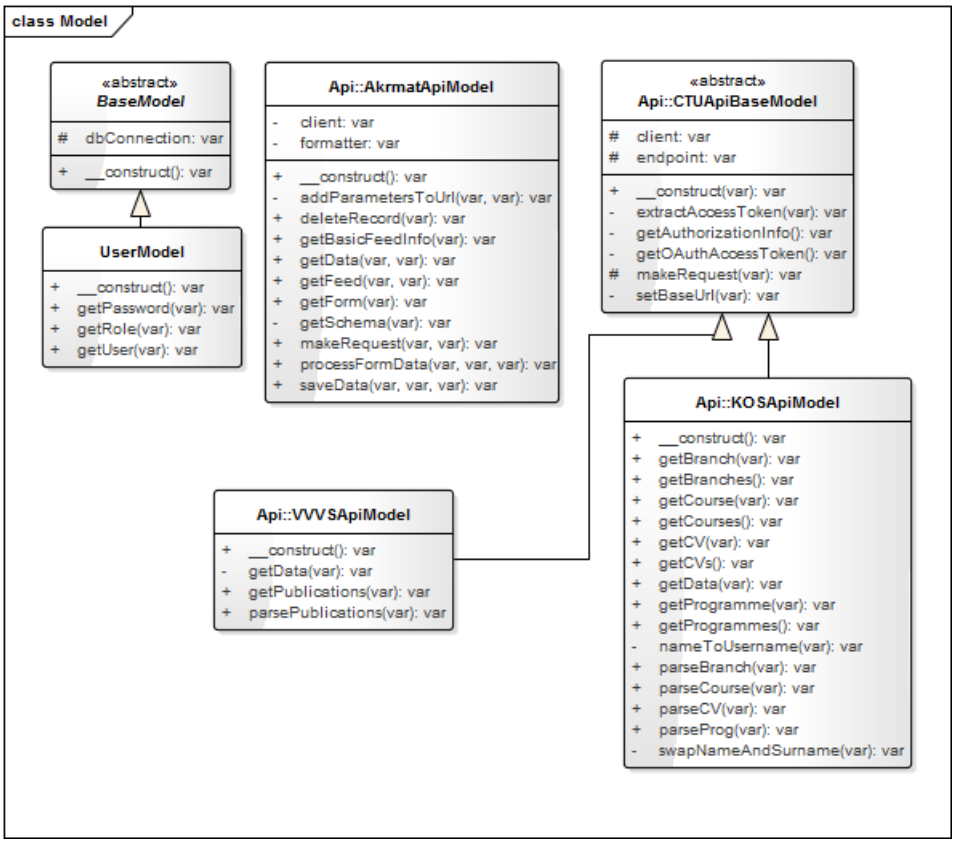
Základem části Model, která pracuje s daty z různých zdrojů, jsou následující tři třídy:

- `BaseModel` pro přistup k datům v MySQL databázi
- `AkrmatApiModel` pro přístup k datům v XML nativní databázi Sedna
- `CTUApiBaseModel` pro přístup k datům z jiných školních systémů

Od abstraktní třídy `BaseModel` dědí třída `UserModel`, která manipuluje s daty o uživateli uloženými v MySQL databázi. `CTUApiBaseModel` je další abstraktní třída a zařizuje přístup k REST API jiných školních systémů. Obě třídy, které od ní dědí, se starají o zisk dat z KOS API, respektive VVVS API. `AkrmatApiModel` je samostatnou entitou, která poskytuje rozhraní pro manipulaci s dokumenty v databázi Sedna. Diagram Modelu lze vidět na obrázku 4.1.

### 4.3.2 Controller

Hlavní třídou této části je abstraktní třída `BaseController`. Ta je základem ostatních tříd controllerů, které od ní dědí. Každý controller reprezentuje prakticky jednu stránku webové aplikace a obsahuje metody pro akce, které na této stránce může uživatel vykonávat. Diagram Controlleru lze vidět na obrázku 4.2.




Obrázek 4.1: Model - Diagram tříd

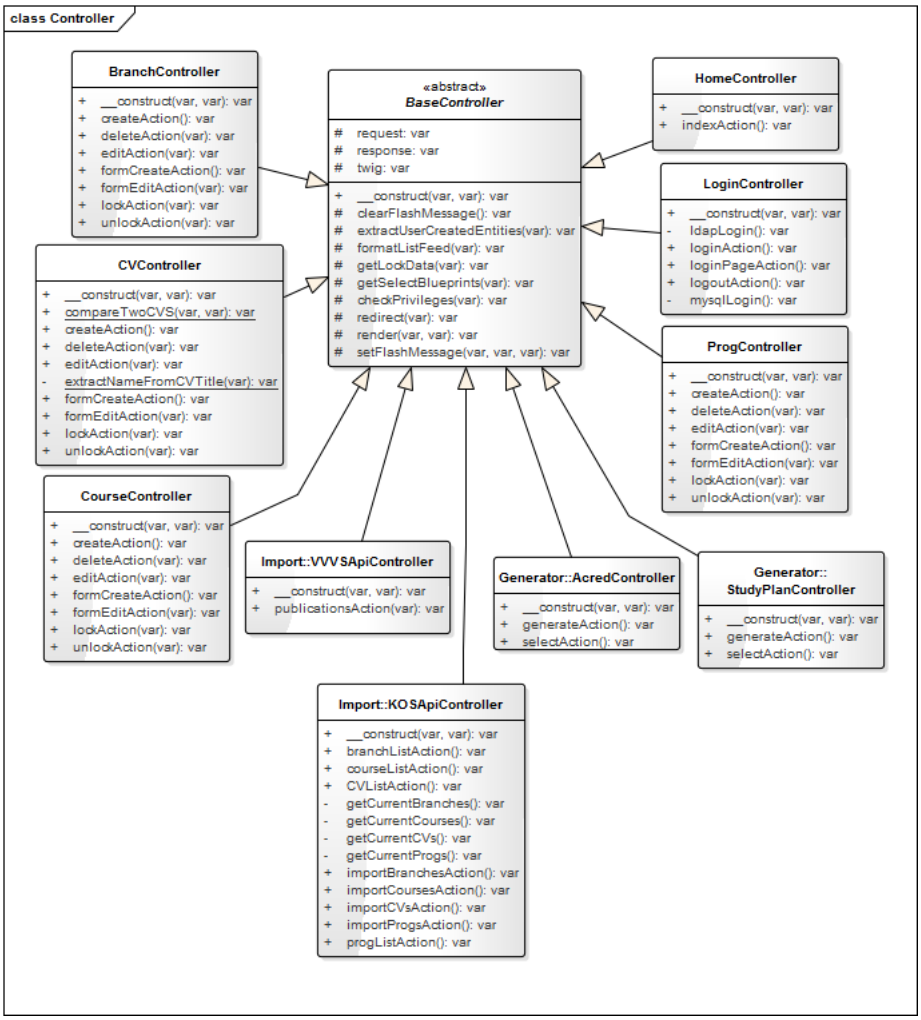
### 4.3.3 View

Část View reprezentují šablony obsahující HTML kód, které jsou vykreslovány dle logiky popsané v jednotlivých controllerech a plněny daty získanými pomocí modelů.

## 4.4 Aktualizace datového modelu

V současné verzi jsou všechny informace o studijních oborech vedeny v rámci studijních programů. To má za následek několik problémů popsaných v sekci 2.5.2. Řešením je upravit datový model systému a extrahovat informace o oborech do nové samostatné entity. Pro obory se vytvoří nové XML schéma a do něj se přesunou některé XML elementy ze schématu programu. Extrahovány budou tyto elementy:

- Obsah elementu `obory_programu`
- Celý element `predmety_oboru`



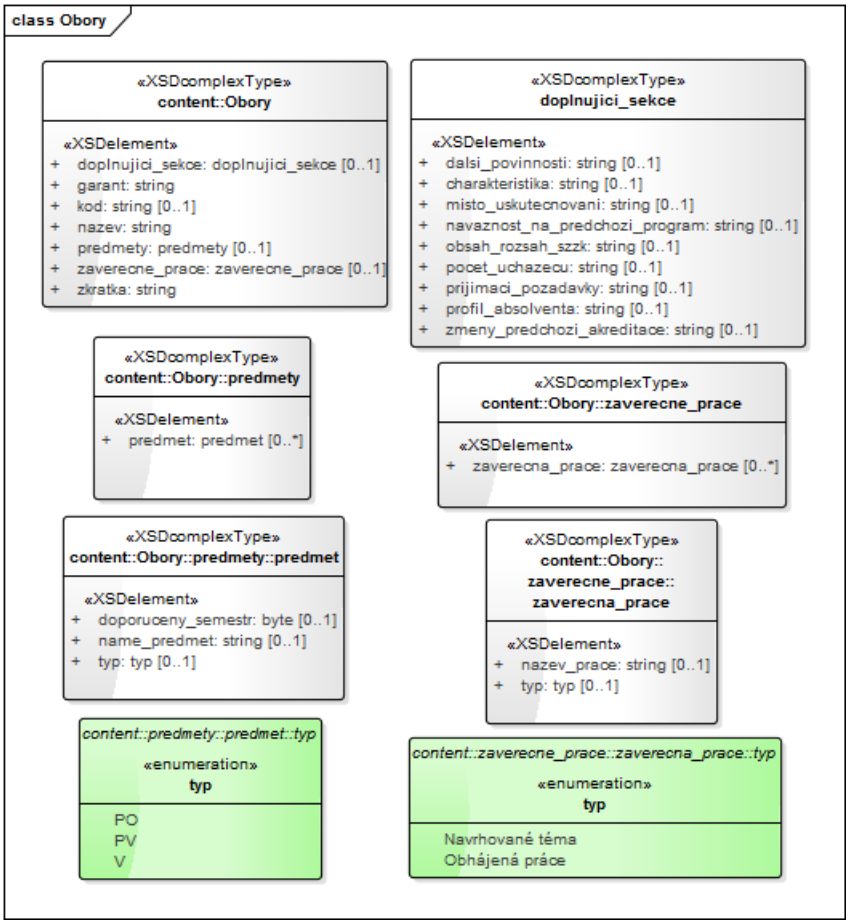

Obrázek 4.2: Controller - Diagram tříd

- Celý element `zaverecne_prace`

Element `obory_programu` obsahuje obecné informace o jednotlivých oborech, jako jsou zkratka oboru, název, jméno garanta a další textové popisy. Celý tento obsah se přesune do elementu `doplnujici_sekce` v rámci nového schématu oboru a ve schématu programu bude nahrazen jedním elementem `obor`, který bude obsahovat kombinaci zkratky a jména oboru. Díky tomu bude zachována informace, které obory do programu patří.

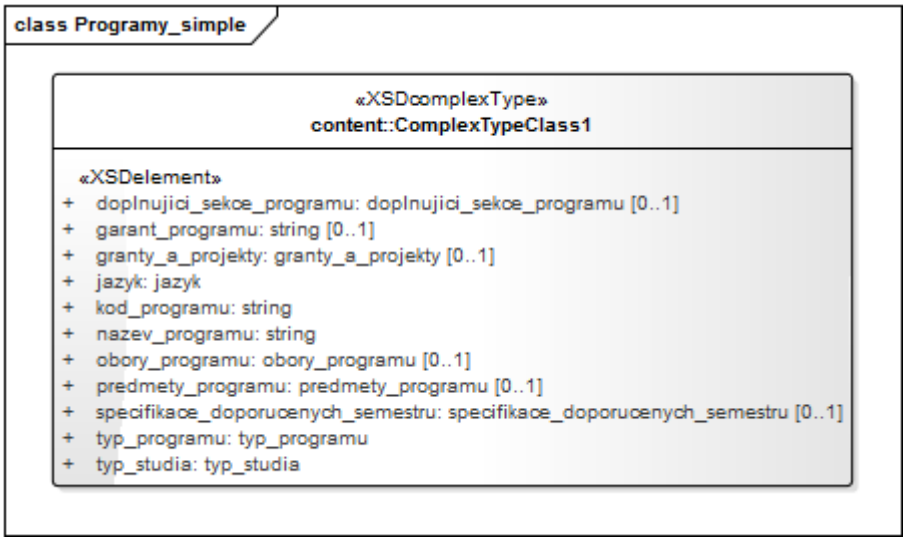
Element `predmety_oboru`, obsahující seznam oborových předmětů, a element `zaverecne_prace`, obsahující závěrečné práce v jednotlivých oborech, se kompletně přesunou ze schématu programu do schématu oboru. Element `predmety_oboru` bude v rámci schématu oboru přejmenován na `predmety`.

Zbývá nastavit povinné elementy ve schématu oboru. Budou jimi zkratka oboru, název oboru a garant oboru. Zkratka bude v rámci systému AKRMAT unikátním identifikátorem oboru, název a garant oboru budou povinné pro zajištění správného generování formulářů akreditačních materiálů. Kompletní datový model entity Obor lze vidět na obrázku 4.3.




Obrázek 4.3: Kompletní datový model entity Obor

Další změna entity program už se netýká oborů. V současné verzi jsou informace o grantech a projektech evidovány v elementu `granty_a_projekty`, který je vnořený do elementu `doplnujici_sekce_programu`. Je to jediná situace v celém datovém modelu, kdy vícepoložková sekce (element obsahující sekvenci jiných elementů) je uvnitř jiného rozsáhlého elementu. Pro jednodušší a přehlednější strukturu vygenerovaného HTML formuláře se element `granty_a_projekty` přesune z `doplnujici_sekce_programu` o úroveň výše vedle ostatních vícepoložkových sekcí. Upravený datový model entity program je ukázán na obrázku 4.4.




Obrázek 4.4: Upravený datový model entity Program

## 4.5 Nové nastavení uživatelských rolí

Po konzultaci s vedoucím diplomové práce bylo rozhodnuto, že v aplikaci budou nadále vystupovat pouze tři uživatelské role a jejich názvy budou převedeny do anglického jazyka. Rolí, která bude z nové verze aplikace vyjmuta, je Editor. Dle návrhu současné verze aplikace je tato role ekvivalentní s rolí Admin a doposud nebyla využívána. Nyní budou tedy v systému vystupovat pouze tři uživatelské role s názvy: Admin, Author a Reader.

**Reader** Vychází z původní role Čtenář, ale přijde o možnost generovat akreditační spis a studijní plán. Reader tuto funkcionalitu, která sémanticky neladí s názvem role, nepotřebuje. Role Reader bude přiřazena běžnému uživateli, který může pouze prohlížet data uložená v databázi pomocí HTML formulářů.

**Author** Přijde také o možnost generování akreditačního spisu a studijního plánu. Tuto roli budou mít uživatelé, kteří mají v rámci tohoto systému veden svůj životopis, tedy garanti předmětu nebo oborů. Author bude moci editovat záznamy, které sám vytvoří, nebo u kterých mu bude přiděleno autorství. Uživatelé s touto rolí budou mít možnost importovat publikace z VVVS API v rámci formuláře svého životopisu.

**Admin** K této roli samozřejmě patří přístup ke všem funkcionalitám systému. Oproti roli Author může Admin navíc editovat cizí záznamy, generovat akreditační spis a studijní plán a importovat data z KOS API. Touto rolí bude disponovat pouze několik hlavních uživatelů systému.

Tabulka 4.1: Nové nastavení uživatelských rolí

| Akce | Reader | Author | Admin |
|---|---|---|---|
| Čtení záznamů | Ano | Ano | Ano |
| Vytváření záznamů | Ne | Ano | Ano |
| Editace vlastních záznamů | Ne | Ano | Ano |
| Editace cizích záznamů | Ne | Ne | Ano |
| Import z VVVS API | Ne | Ano | Ano |
| Import z KOS API | Ne | Ne | Ano |
| Generování studijního plánu | Ne | Ne | Ano |
| Generování akreditačního spisu | Ne | Ne | Ano |

### 4.5.1 Nastavení autora záznamu

Při importu záznamů z KOS API se jako jejich autor nastaví uživatel, který import provedl. Vždy to tedy bude někdo s rolí Admin. Aby si mohli uživatelé s rolí Author upravovat své záznamy, je nutné nějakým způsobem zařídit změnu autora (vlastníka) záznamu.

Nabízí se zde tři možnosti. První možností je povolit roli Author importovat záznamy z KOS API. Každý by si importoval svůj životopis, předměty a obory. Druhou možností je přímo v aplikaci umožnit roli Admin libovolně měnit autora dokumentů. Třetí možností je tento problém ignorovat a zachovat způsob změny autora ze současné verze systému, který se provádí pomocí modifikace dokumentu přímo v databázi Sedna.

## 4.6 Metodika importu z KOS API a VVVS API

Import dat z jiných systémů je zásadním prvkem při vytváření akreditačních materiálů, jelikož odstraňuje nutnost vytvářet veškeré záznamy ručně. Z dostupných zdrojů sice nelze získat veškerá potřebná data, ale i přesto se jedná o dobrý způsob, jak celý proces vytváření akreditačních materiálů optimalizovat. Dosud byl import prováděn pomocí různých externích skriptů a jeho integrace přímo do aplikace je mnohem pohodlnějším řešením.

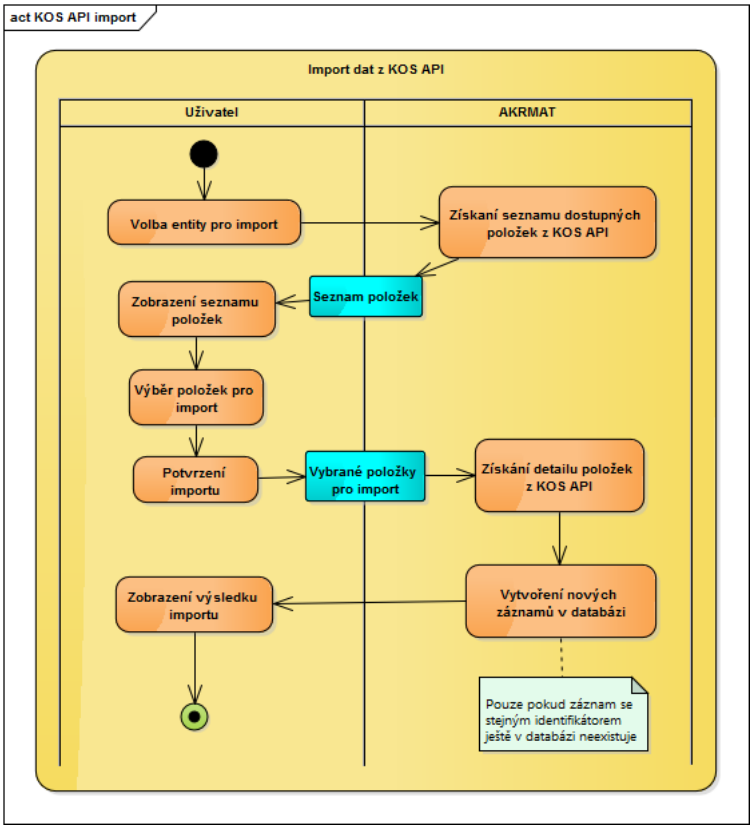
### 4.6.1 Specifikace importu

- Import ze systému KOS API bude dostupný pro uživatelskou roli Admin.
- Import ze systému VVVS API bude dostupný pro uživatelskou roli Admin a Author.
- Import ze systému KOS API nebude přepisovat existující záznamy.

### 4.6.2 Proces importu dat z KOS API

Ve webové aplikaci si uživatel bude moci z hlavní nabídky vybrat možnost „Importovat“ a následně zvolit, kterou entitu si přeje importovat. Poté třída `KOSApiModel` provede HTTP požadavek a získá seznam dostupných záznamů v rámci kategorie vybrané entity.

Tento seznam se zobrazí uživateli, který si v něm přesně zvolí, které konkrétní záznamy si přeje importovat. Po spuštění importu třída `KOSApiModel` provede několik HTTP požadavků a získá detaily vybraných záznamů. Pro ně se systém pokusí vytvořit nové dokumenty a uložit je do databáze. Pokud dokument se stejným identifikátorem už existuje, pokračuje se dalším dokumentem. Po provedení importu se uživateli zobrazí seznam úspěšně a neúspěšně importovaných záznamů. Celý proces je ilustrován v UML diagramu 4.5.




Obrázek 4.5: Diagram aktivit pro proces importu dat z KOS API

### 4.6.3 Proces importu publikací z VVVS API

Import publikací se bude odehrávat na stránce formuláře životopisu. Po odemčení formuláře k editaci se v sekci „Publikace“ zobrazí tlačítko pro import. Po kliknutí na toto tlačítko je uživateli prezentován seznam jeho nalezených publikací, které lze importovat. Ty budou seřazeny od nejnovější po nejstarší.

Seznam publikací získá třída `VVVSApiModel` pomocí HTTP požadavku na službu VVVS API. Uživatel si v seznamu může zvolit, které publikace se budou importovat. Po potvrzení importu se publikace vloží jako položky do sekce „Publikace“. Zde se neprovede okamžitě žádné uložení do databáze. Uživatel bude mít možnost informace ještě upravit. Uložení se provede až odesláním celého formuláře životopisu.

## 4.7 Metodika zamykání editace dokumentů

Na tvorbě akreditačního spisu typicky pracuje více lidí. Může tedy dojít k situaci, kdy bude jeden dokument editován více uživateli najednou. Zabránit potenciálnímu konfliktu při ukládání do databáze může tzv. zamykání dokumentů. V praxi to bude fungovat tak, že uživatel vždy požádá systém o uzamčení dokumentu k editaci a tento dokument bude poté pro ostatní uživatele dostupný pouze pro čtení.

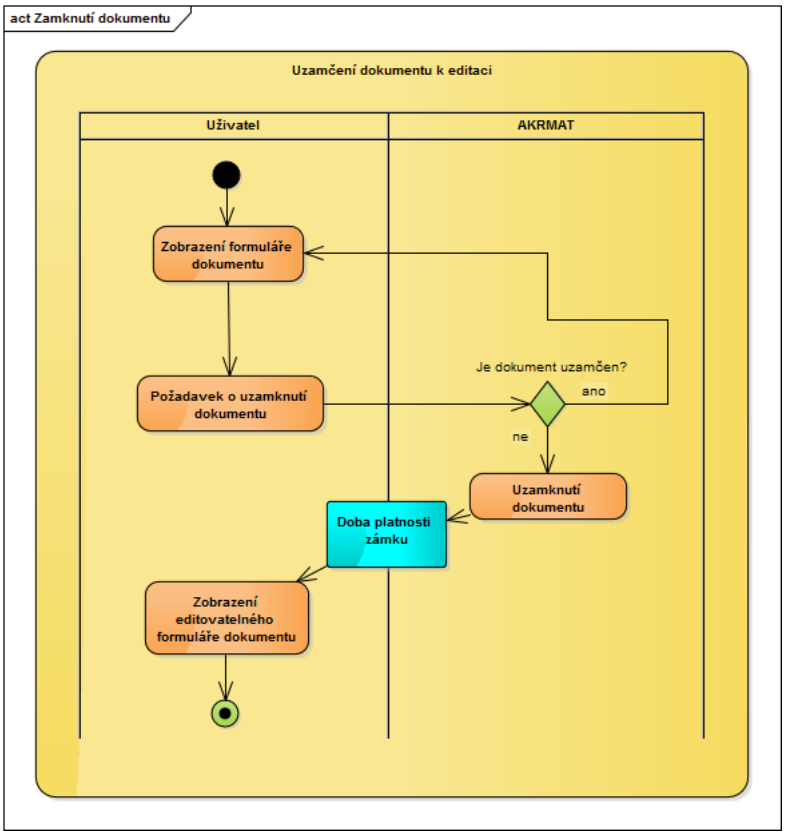
### 4.7.1 Specifikace zamykání dokumentů

- Zamykat dokumenty mohou pouze uživatelské role Admin a Author.
- Zámek není možné programově zrušit nikým jiným, než jeho autorem.
- Platnost zámku vyprší automaticky po uplynutí nastavitelného časového intervalu.

### 4.7.2 Proces zamykaní dokumentu

Při příchodu na stránku s HTML formulářem bude tento formulář ve výchozím stavu uživateli dostupný pouze pro čtení. Po stisku tlačítka „Editovat“ se provede požadavek o uzamčení dokumentu. Pokud je uživatel oprávněn k jeho uzamknutí a zároveň už není dokument uzamčen jiným uživatelem, je tento požadavek přijat. Dokument se uzamkne a v HTML formuláři se povolí editace. Uživateli bude na stránce také zobrazena doba platnosti zámku dokumentu. Po tuto dobu může uživatel bezpečně dokument editovat.

V případě vypršení platnosti zámku je uživateli o tom zobrazena zpráva. Editovat dokument je nadále možné, nicméně není zaručeno, že někdo jiný tento dokument mezitím nezamkne. Pro obnovení zámku musí uživatel uložit změny odesláním formuláře a požádat opět o nový zámek. Proces je ilustrován v UML diagramu 4.6.




Obrázek 4.6: Diagram aktivit pro proces zamykání dokumentu

### 4.7.3 Proces odemykaní dokumentu

Pokud je dokument uzamčen k editaci, postupně se zkracuje nastavená doba platnosti zámku. Uživatel bude moci stisknutím tlačítka „Zrušit editaci“ explicitně dokument odemknout a zámek uvolnit. Po uložení změn odesláním HTML formuláře se také zámek uvolní. Pokud platnost zámku dokumentu vyprší, při jakémkoliv dalším požadavku na tento dokument se zámek automaticky uvolní. K uvolnění zámku nebude docházet tedy úplně přesně po uplynutí nastaveného intervalu platnosti.

## 4.8 Nové uživatelské rozhraní

Ke zlepšení použitelnosti webové aplikace je navrženo nové uživatelské rozhraní, do kterého je také zakomponována podpora nových funkcionalit implementovaných v rámci této práce. Cílem je, aby nová verze webové aplikace měla modernější, přehlednější vzhled s jasnými a srozumitelnými ovládacími prvky.

### 4.8.1 Barvy a Písma

V novém rozhraní dojde ke změně z modrého/světle modrého pozadí a černého textu ke klasické barevné kombinaci: bílé/světle šedé pozadí a černý text. Dále budou použity dva typy písma místo jediného, jak je to v současné verzi. Tyto dvě změny by měly přispět k lepší a pohodlnější čitelnosti zobrazovaného obsahu.

### 4.8.2 Cílová zařízení

Webovou aplikaci AKRMAT lze klasifikovat jako určitý typ administračního rozhraní. Bude používána především v prostředí kanceláří na FIT. Lze tedy bezpečně vyloučit potřebu podpory mobilních zařízení. Cílovými zařízeními byly stanoveny notebooky a stolní počítače.

Většina moderních notebooků má rozlišení obrazovky o velikosti alespoň 1366 x 768 pixelů a ve většině kanceláří na FIT jsou monitory s Full HD[20] rozlišením. Podle [21] má navíc 97 % uživatelů rozlišení větší než 1024 x 768 pixelů. Z výše uvedených důvodů bude minimální šířka obrazovky, pro kterou bude rozhraní optimalizováno, 1024 pixelů.

### 4.8.3 Hlavní stránka

Oproti současné verzi aplikace, kde byly na hlavní stránce zobrazeny dlouhé seznamy, ve kterých byly jednotlivé položky velmi blízko u sebe a nebyly nijak oddělené, prezentuje nové rozhraní tyto seznamy pomocí tabulek s jasně oddělenými položkami pro větší přehlednost.

Na hlavní stránku je nutné navíc přidat seznam záznamů nové entity Obor. Proto jsou zde vedle sebe umístěné celkem čtyři tabulky. Pro přidání nového záznamu slouží výrazné tlačítko ve tvaru plus umístěné v hlavičce každé tabulky. Wireframe hlavní stránky lze vidět na obrázku D.1.

### 4.8.4 Formuláře

Hlavním cílem při návrhu nového UI bylo zvýšit přehlednost HTML formulářů. Stránky s formuláři jsou místem, na kterém uživatel stráví nejvíce času při práci s aplikací. Wireframe stránky s formulářem lze vidět na obrázku D.2.

**Přesunutí popisků vstupů** Prvním krokem k větší přehlednosti je přesunutí popisků u jednotlivých formulářových vstupů z levé strany nahoru nad vstupy a zvětšení velikosti jejich písma. To umožní uživateli rychleji najít vstup, který hledá. To platí zejména v rozsáhlých formulářích, jaké obsahuje webová aplikace AKRMAT.

[20] 1920 x 1080 pixelů

**Rozložení vstupů** Získaný horizontální prostor přesunutím popisků se využije k přidání dalšího sloupce vstupů. Z nynějšího počtu dvou sloupců se přejde na tři stejně široké sloupce. Vstupy budou navíc všechny zarovnané i v rámci vnořených sekcí. Díky tomu formulář vypadá více konzistentní a jednotná linie zarovnání vstupů umožní rychlejší prohlížení obsahu vstupů očima.

**Úprava zabalitelných sekcí** Další úpravou, směřující k lepší přehlednosti, je zavedení zabalitelných sekcí i do ostatních formulářů (v současné verzi jsou jen ve formuláři programu). Tyto sekce jsou ve výchozím stavu zabalené, tím se zmenšuje vertikální rozměr formuláře, a to přispívá k jeho lepší přehlednosti. V zabalitelných vícepoložkových sekcích bude vylepšena funkcionalita řazení prvků pomocí tahu myší (Drag & Drop), kdy se více zvýrazní místo, na které lze položku přetáhnout.

**Editace vstupů typu textarea** Ještě bych chtěl uvést vylepšení editace HTML elementů `textarea`. Při kliknutí na tento element se nově uživateli zobrazí vyskakovací okno s mnohem větším prostorem na vyplňování. Toto okno bude navíc libovolně vertikálně roztažitelné. Vstupy `textarea` totiž většinou v aplikaci AKRMAT obsahují rozsáhlé texty a jejich velikost v rámci formulářů na pohodlné čtení nebo vyplňování nestačí. Lze sice standardně ve všech prohlížečích myší upravovat jejich rozměry, ale to má většinou za následek rozhození konzistentního vzhledu formuláře.

### 4.8.5 Zpětná vazba uživateli

Součástí každého kvalitního administračního rozhraní je jasné a viditelné oznamování zpráv uživateli. Proto po provedení jakékoliv důležité akce bude v malém vyskakovacím okně v horní části obrazovky oznámen její výsledek. Toto okno může uživatel vždy okamžitě zavřít pomocí křížku v pravém horním rohu. Zprávy, které oznamují úspěch akce se zavřou po chvíli automaticky. Chybové zprávy se automaticky nezavírají, ale musí být zavřeny explicitně. To je z důvodu, aby měl uživatel dostatek času k přečtení chybové zprávy. Wireframe zobrazení zpětné vazby lze vidět na obrázku D.2.

## 4.9 Řešení dalších problémů

V této jsou popsány návrhy řešení dalších problémů, které byly uvedeny v sekci 2.5.

### 4.9.1 České názvy v datovém modelu

Změnit názvy v datovém modelu není úplně triviální záležitost. V současné verzi systému jsou přesné názvy totiž využity na několika místech ve zdrojo-

vém kódu. Moderní IDE[21] a textové editory s tímto úkolem mohou pomoci. Poskytují různé funkce hromadného refactoringu[22], ale jejich použití často vede k zavedení nových chyb a musí se tedy používat opatrně.

Po konzultaci s vedoucím diplomové práce bylo rozhodnuto, že pro řešení tohoto problému bude provedena pouze analýza bez implementace. Nejedná se o kritickou chybu a její oprava by mohla být příliš časově náročná. Prioritu tedy dostaly jiné úpravy. V následujících odstavcích je popsáno, kde všude ve zdrojovém kódu by při změnách názvů elementů v datovém modelu muselo dojít k úpravě.

**XML schéma** Nejprve by se musely změnit samotné názvy elementů uvnitř XML schématu, které určuje formu, v jaké se XML dokumenty ukládají do databáze.

**Třídy pro generování formulářů akred. spisu** Dalším místem, kde by muselo dojít k úpravám, jsou všechny třídy typu `FormXGenerator`, kde X je typ formuláře akreditačního spisu. Tyto třídy obsahují metody, které využívají XPath výrazy nebo XML selektory třídy `SimpleXMLElement` k formátování dat. Oba dva případy využívají přesných názvů elementů.

Zdrojový kód 4.1: Příklad výskytu názvů elementů ve třídě FormAGenerator

```
$content->addChild("vysoka_skola",
    (string)$doplnujiciSekce->vysoka_skola);
$content->addChild("fakulta", (string)$doplnujiciSekce->fakulta);
$content->addChild("doba_studia",
    (string)$doplnujiciSekce->doba_studia);
$content->addChild("vysledny_titul",
    (string)$doplnujiciSekce->vysledny_titul);
\\ ...
```

**Transformace HTML formulářů** Názvy elementů se díky dynamicky generovaným HTML formulářům z XML schémat projeví i v názvech HTML elementů uvnitř těchto formulářů. Právě toho využívají transformace formulářů specifikované v souborech JavaScriptu `forms.js` a `adjustments.js`. Poslední místo, které by se muselo upravit, také souvisí s těmito transformacemi. Pomocí klíčů polí obsahujících názvy HTML elementů v souborech `form-labels.php`, `form-textareas.php` a `form-tooltips.php` se určuje, na který HTML element bude konkrétní transformace aplikována.

[21] Integrated Development Environment
[22] proces určité restrukturalizace zdrojového kódu

Zdrojový kód 4.2: Příklad výskytu názvů elementů v souboru adjustments.js

```
$(elementType+'[name*="predmety_programu"][name$="name_predmet"]')
.each(function () {
   fields = fields
   + '<option value="'+$(this).val()+'">'+$(this).val()+'</option>';
});
```

Zdrojový kód 4.3: Příklad výskytu názvů elementů v souboru form-labels.php

```
'/content/nazev' => "Název předmětu",
'/content/url' => "URL předmětu",
'/content/nazev_eng' => "Název předmětu v~AJ",
'/content/rozliseni' => "Rozlišení předmětu",
'/content/kredity' => "Kredity",
'/content/zakonceni' => "Způsob zakončení",
```

### 4.9.2 Kvalita zdrojového kódu

Při psaní zdrojového kódu se vždy snažím držet rad popsaných v knize Clean Code od Roberta C. Martina.[22] Použití MVC mi zajistí logické rozdělení zdrojového kódu do tříd a i v JS kódu se pokusím dodržet přehlednou strukturu pomocí objektů. Budu používat metody a proměnné s jasnými názvy tak, aby bylo zapotřebí co nejméně komentářů. Kde budou komentáře nutné, použiji co nejjasnější popis.

### 4.9.3 Prohození sémantiky POST a PUT

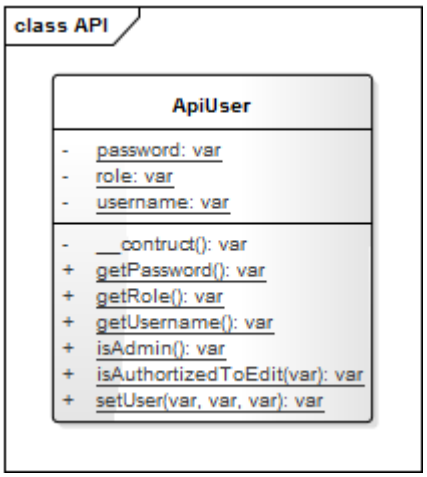
V současné verzi REST API je prohozena sémantika těchto metod. Tato chyba nijak neovlivňuje funkcionalitu systému a její náprava je navíc velmi jednoduchá. Ve všech třídách, které dědí od třídy `Resource`, se prohodí názvy metod `put` a `post`. Na tuto změnu musí reagovat i webová aplikace, odkud se tyto metody volají. U volání metody `exec` třídy `httpClient` se změní pouze hodnota parametru, který určuje HTTP metodu požadavku.

### 4.9.4 Použití PHP session v rámci REST API

PHP session se používá hlavně v případě, kdy je potřeba přenést data mezi několika HTTP požadavky. PHP prostředí po dokončení HTTP požadavku všechna ostatní data zahodí. V případě AKRMAT REST API se nicméně všechny potřebné operace provádějí v rámci jednoho požadavku. To plyne ze samotné podstaty REST API. V tomto případě se tedy PHP session nemusí používat. Pro přesun dat mezi PHP skripty místo toho lze využít třídy se statickými vlastnostmi.

Využije se tedy nové třídy `ApiUser`, která obsahuje tři statické vlastnosti pro uživatelské jméno, heslo a roli. Tyto informace se nastaví při autentizaci

uživatele a budou poté dostupné z kteréhokoli PHP skriptu v rámci jednoho HTTP požadavku. Navíc se jedná o bezpečnější řešení než uložení uživatelské informace do PHP session.




Obrázek 4.7: Diagram tříd pro třídu ApiUser

### 4.9.5 Javascriptové chyby

Všechny nalezené javascriptové chyby budou odstraněny implementací nové webové aplikace. Při implementaci bude nicméně důležité předcházet dalším novým chybám. Toho lze docílit pečlivým ošetřováním všech možných mezních a nulových stavů, které mohou nastat.

Co se týče řešení problému prázdné vícepoložkové sekce, který byl popsán v sekci 2.5.8.2, bude muset být vyřešen i pro novou webovou aplikaci. V současné verzi se po stisknutí tlačítka pro přidání další položky vytvoří nová položka kopírováním nějaké existující. Pokud žádná neexistuje, nelze vytvořit kopii a nastane chyba.

Řešením by mohlo být, už při načtení formuláře, z každé vícepoložkové sekce vytvořit kopii položky a udělat z ní jakýsi vzor, ze kterého budou nové položky vytvářeny. Tento vzor by byl uložen v rámci DOMu[23], ale byl by uživateli skrytý. Pokud by tedy nastala situace prázdné vícepoložkové sekce, bylo by stále odkud kopírovat nové položky.

### 4.9.6 Struktura URL

Problém delších a nevzhledných URL lze vyřešit použitím modulu `mod_rewrite` webového serveru Apache, na kterém webová aplikace běží. Do konfiguračního souboru `apache2.conf` nebo do souboru `.htaccess` přímo v adresáři aplikace lze pak popisovat pravidla pro přepis URL na jiné tvary. Modul parsuje URL

[23]Document Object Model

za běhu pomocí regulárních výrazů a podle specifikovaných pravidel je přepisuje na jiné tvary.[10] Díky tomu lze použít URL, které jsou kratší a lépe zapamatovatelné.

Kapitola 5

# Implementace

V této kapitole popíši implementační postupy nejdůležitějších úprav, které byly v systému AKRMAT provedeny. Dále zde uvedu také několik problémů, na které jsem při implementaci narazil a vysvětlím, jakým způsobem jsem je řešil.

## 5.1 Vývojové fáze

Vývoj probíhal v několika inkrementálních fázích. Každá přidala nebo upravila nějakou funkcionalitu a ta byla důkladně otestována, než se přešlo k další fázi. Posloupnost vývojových fází byla následující:

1. úprava datového modelu
2. nová webová aplikace
3. zamykání editace dokumentů
4. import dat z KOS API a VVVS API
5. generování akreditace a studijního plánu

Nejvíce času zabralo vytvoření nové webové aplikace, naopak překvapivě snadno a rychle proběhla úprava tříd generátorů akreditace tak, aby podporovala novou verzi datového modelu.

## 5.2 Vývojové prostředí

Vývoj probíhal na platformě LAMP, což je zkratka pro spojení technologií Linux, Apache, MySQL a PHP. V rámci serveru Apache jsem si vytvořil dva virtuální servery. Jeden pro část s REST API, který běžel na URL `http://api.akrmat` a druhý pro webovou aplikace dostupný na URL `http://akrmat`. Pro ostré nasazení se předpokládá podobné nastavení.

## 5.3 Aktualizace datového modelu

První fází vývoje byla úprava datového modelu systému. Požadavkem bylo extrahovat informace o oborech z entity Program do samostatné entity Obor. Všechna data, která byla extrahována jsou uvedena v sekci 4.4. Aktualizace datového modelu vyžadovala dva kroky:

- vytvoření nového XML schématu pro entitu Obor
- vytvoření nových tříd pro manipulaci s entitou Obor v rámci REST API

### 5.3.1 Vytvoření nového XML schématu

Nejprve bylo nutné vytvořit nové XML schéma pro entitu Obor, podle kterého se bude kontrolovat integrita dat ukládaných do databáze. Pojmenováno bylo `obory.xsd` a umístěno vedle současných schémat pro ostatní entity. Níže lze vidět ukázku tohoto schématu. Z prezentačních důvodů mají elementy `doplnujici_sekce`, `predmety` a `zaverecne_prace` skrytý obsah.

Zdrojový kód 5.1: Zjednodušená ukázka XML schématu obory.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

<!-- obsah -->
<xs:element name="content">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nazev" type="xs:string"/>
      <!--timto elementem bude kazdy zaznam identifikovan-->
      <xs:element name="zkratka" type="xs:string"/>
      <xs:element name="garant" type="xs:string"/>
      <xs:element name="kod" minOccurs="0" type="xs:string"/>
      <xs:element name="doplnujici_sekce" minOccurs="0"></xs:element>
      <xs:element name="predmety" minOccurs="0"></xs:element>
      <xs:element name="zaverecne_prace" minOccurs="0"></xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

### 5.3.2 Vytvoření nových tříd v rámci REST API

Pro možnost manipulace s novou entitou Obor bylo nutné v rámci REST API vytvořit několik nových tříd.

Nejprve jsem vytvořil dvě třídy: `OborResource` a `OboryResource`. První jmenovaná třída reprezentuje zdroj jednoho konkrétního oboru, druhá jmenovaná reprezentuje celou kolekci oborů. Obě mají analogickou strukturu a umístění jako podobné třídy pro ostatní entity.

Do těchto tříd byly přidány metody, které reprezentují jednotlivé HTTP metody REST API. `OborResource` obsahuje metody `get`, `put` a `delete` a třída `OboryResource` obsahuje `get` a `post`.

Novou entitu Obor je nutné v systému interně nějak reprezentovat. O to se stará nová třída `oborDocument`, která stejně jako obdobné třídy pro ostatní entity rozšiřuje obecnou třídu `Document` a slouží k reprezentaci jednoho XML dokumentu oboru. Dále obsahuje logiku pro ukládání oboru do databáze.

Před uložením do databáze je zapotřebí dokument nejprve validovat. Nová třída `oborXMLValidator` tuto validaci pro dokumenty oborů zařídí. Tato třída si pouze načte nově vytvořené XML schéma a samotná logika validace už je popsána v rodičovské třídě společně pro všechny entity.

Po implementaci těchto úprav už lze pomocí přímého přístupu na REST API, např. z prohlížeče, manipulovat s novou entitou Obor. Ukázka XML dokumentu oboru BI-WM je níže.

Zdrojový kód 5.2: XML dokument pro obor BI-WM

```
<obor>
  <metadata>
    <vytvoreno>2016-04-03T16:00:29Z</vytvoreno>
    <vytvoril>admin</vytvoril>
    <zmeneno>2016-04-28T15:17:11Z</zmeneno>
    <zmenil>admin</zmenil>
    <zamkl>admin</zamkl>
    <zamceno>29.04.2016 21:23:37</zamceno>
    <platnostZamku>572</platnostZamku>
  </metadata>
  <content>
    <nazev>Web a multimédia</nazev>
    <zkratka>BI-WM</zkratka>
    <garant>vitvatom</garant>
    <doplnujici_sekce></doplnujici_sekce>
    <predmety></predmety>
    <zaverecne_prace></zaverecne_prace>
  </content>
</obor>
```

## 5.4 Webová aplikace

### 5.4.1 MVC

Jak jsem uvedl v sekci 3.5.3, při implementaci jsem nepoužil žádný ucelený PHP framework, který by využíval architektonického vzoru MVC. Jelikož jsem přesto chtěl tuto architekturu použít, bylo nutné implementovat její vlastní verzi.

Třídy umístěné v adresáři `web-app/lib/model` reprezentují vrstvu Model, třídy umístěné v `web-app/lib/controller` vrstvu Controller a šablony uvnitř `web-app/view` vrstvu View.

Důležité je zmínit proces směrování jednotlivých URL na metody Controllerů. Všechna směrovací pravidla jsou definována ve třídě `Routing`. Ta obsahuje statické pole `$routes`, ve kterém jsou uvedena jednotlivá směrovací pravidla. Každé pravidlo je asociativní pole o pěti prvcích.

Zdrojový kód 5.3: Příklad směrovacího pravidla ve třídě Routing

```
array(
   "method" => "GET",
   "url" => "/vytvorit-predmet",
   "controller" => "CourseController",
   "action" => "formCreateAction",
   "roles" => array("admin", "author")
);
```

Prvek `method` říká, pomocí které HTTP metody byl požadavek odeslán, `url` obsahuje relativní URL požadavku, `controller` název cílového Controlleru, `action` název konkrétní metody a `roles` povolené uživatelské role pro tuto metodu (akci).

Samotné směrování zařizuje PHP router Klein. Nejprve je ale nutné do routeru registrovat směrovací pravidla. Třída `KleinDispatcher`, obalující router Klein, obsahuje metodu `registerRoutes`, která zajistí registraci všech směrovacích pravidel. Pomocí PHP funkce `call_user_func_array` se pak při každém požadavku dynamicky zavolá příslušná metoda Controlleru. Samotné použití této funkce vypadá následovně:

```
call_user_func_array(array($ctrl, $action),array($request->params()));
```

Proměnná `$ctrl` už obsahuje instanci konkrétního Controlleru a v `$action` je uložen název metody. Obě tyto informace jsou získány z příslušného směrovacího pravidla. Dále se metodě předají URL parametry požadavku pomocí volání `$request->params()`.

### 5.4.2 Přihlášení do aplikace

Uživatele nyní přímo autentizuje webová aplikace pomocí MySQL databáze. V předchozí verzi probíhala autentizace nepřímo přes AKRMAT REST API. Aplikace odeslala HTTP požadavek na URL `/user` s příslušnou autentizační informací. Heslo se navíc posílalo v plain-textovém formátu. V odpovědi byl pak odeslán zpět výsledek autentizace ve formátu XML.

Tato změna způsobu autentizace byla realizována, protože v původním návrhu bylo plánováno z webové aplikace do MySQL databáze ukládat více informací (např. informace o zamykání editace dokumentů). Proto bylo nezbytné, aby k ní aplikace měla přímý přístup. Později byl nicméně návrh upraven a MySQL databáze tak slouží nadále pouze k uložení informací o uživatelích. Pokud by v budoucnu došlo k rozhodnutí, že je zapotřebí do této databáze ukládat více informací, je k tomu připraveno funkční rozhraní.

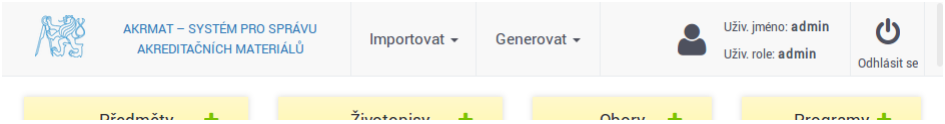
V nové webové aplikaci byla také zachována podpora přihlášení přes LDAP. Za podmínky, že je uživatel veden také v MySQL databázi a má přiřazenou nějakou uživatelskou roli.

### 5.4.3 Nové uživatelské rozhraní

V následujících sekcích budou popsány informace o implementaci jednotlivých stránek webové aplikace. Snímky obrazovky těch nejdůležitějších jsou k nalezení v příloze D.

#### 5.4.3.1 Společná hlavička

Společným prvkem všech stránek je hlavička stránky. Ta obsahuje hlavní nabídku, která je viditelná pouze pro roli Admin, jelikož obsahuje odkazy na stránky generátoru akreditace/studijního plánu a importu dat z KOS API. V hlavičce se dále na levé straně nachází logo ČVUT a název aplikace. Oba tyto prvky obsahují odkaz, který vede zpět na hlavní stránku. Po pravé straně je umístěna informace o přihlášeném uživateli: uživatelské jméno a role. Ještě více vpravo se nachází tlačítko pro odhlášení z aplikace.

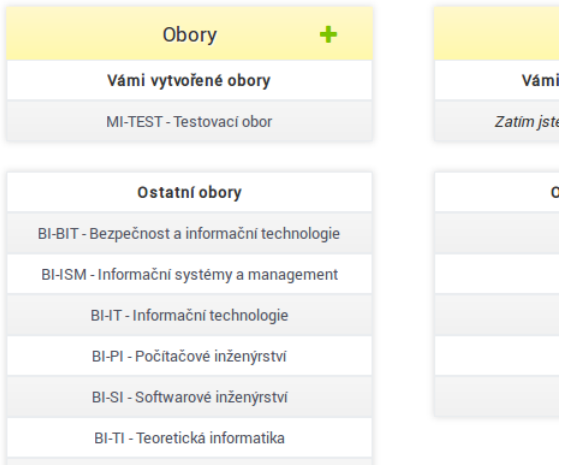



Obrázek 5.1: Společná hlavička pro všechny stránky

#### 5.4.3.2 Hlavní stránka

Na hlavní stránce se zobrazují vedle sebe čtyři tabulky, jedna pro každou hlavní entitu vystupující v systému. Nově zde samozřejmě je i tabulka oborů.

Každá tabulka je rozdělena na dvě části. Horní část obsahuje ty položky, jejichž autorem je přihlášený uživatel. Ten bude totiž pracovat zejména s těmito položkami a díky tomuto rozdělení je snáze nalezne. Ve spodní části jsou umístěné ostatní položky. Pro přidání nové položky slouží zelené tlačítko ve tvaru plus v hlavičce každé tabulky.




Obrázek 5.2: Seznam oborů na hlavní stránce

#### 5.4.3.3 Stránka pro import dat z KOS API

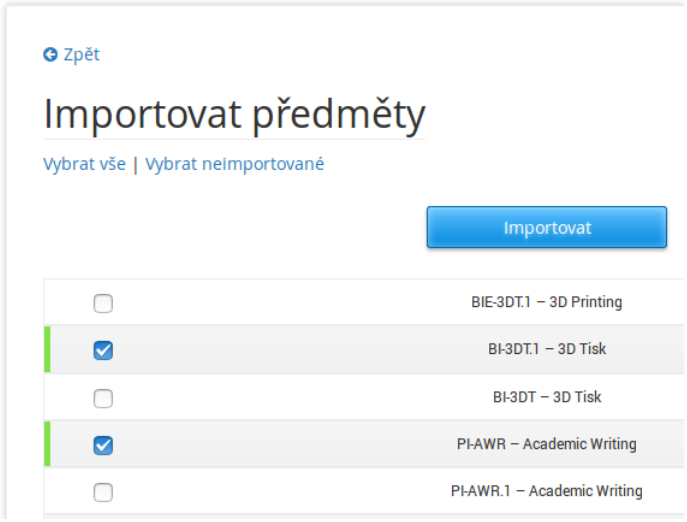
Tato stránka je dostupná z hlavní nabídky, kde uživatel zvolí, kterou entitu si přeje importovat. Na stránce samotné se zobrazí seznam dostupných záznamů pro import a lze si přesně zvolit, které se budou importovat. Dostupná jsou také tlačítka „Označit vše“ a „Označit neimportované“, viz obrázek 5.3. Po provedení importu se zobrazí přehled s výsledkem importu, který je vidět na obrázku 5.4.

#### 5.4.3.4 Stránka pro generování

Samostatné stránky mají i generátory akreditace a studijního plánu, které jsou dostupné z hlavní nabídky. Obě se obsahem liší pouze v nadpisu. Nachází se zde pouze výběr studijního programu, ze kterého se provádí generování.

#### 5.4.3.5 Stránka s formulářem

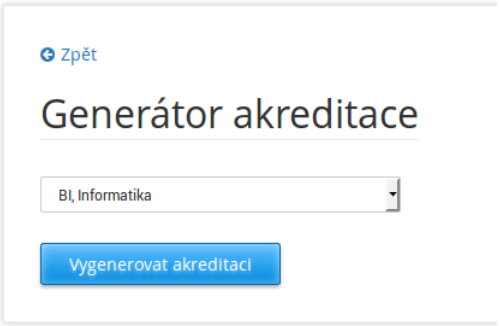
**Modifikace formuláře** Formuláře se generují dynamicky z XML schémat jednotlivých entit. Proto jakékoli úpravy HTML struktury formuláře je nejlepší provádět v JavaScriptu. Veškerá logika týkající se formulářů se nachází ve skriptu `form.js`. Ten obsahuje tři objekty:




Obrázek 5.3: Výběr předmětů k importování z KOS API


Obrázek 5.4: Přehled výsledku importu předmětů z KOS API

- `Form` obsahuje logiku veškerých modifikací formuláře: plnění daty, přidávání dalších HTML elementů (např. tlačítka), úprava atributů elementů, transformace vstupů na jiné typy, přidávání popisků vstupů nebo tooltipů nápovědy. Jednotlivé modifikace jsou aplikovány v rámci metody `setup`. Na pořadí, v jakém jsou aplikovány, záleží.




Obrázek 5.5: Výběr programu na stránce generátoru akreditace

- `Timer` reprezentuje časovač, který informuje uživatele o zbývající platnosti zámku editace dokumentu.
- `Sortable` zařizuje řazení položek pomocí tahu myší (Drag & Drop) uvnitř každé vícepoložkové sekce.

**Vzhled formuláře** Díky přidaným HTML třídám v rámci výše zmíněných modifikací formuláře, bylo celkem snadné pomocí CSS nastylovat formulář do požadované podoby. Zvolil jsem rozložení vstupních polí do tří stejně širokých sloupců, které je realizováno pomocí nové CSS3 vlastnosti flexbox aplikované na obalující elementy jednotlivých sekcí. O možnostech vlastnosti flexbox jsem hovořil v rámci sekce 3.12. V tomto případě CSS pravidlo pro nastavení flexboxu v obalujícím elementu vypadá následovně:

```
display: flex;
flex-wrap: wrap;
```

První řádek určuje samotné použití techniky `flexbox`. Druhý říká, že pokud se obsah nevejde do jednoho řádku, má se pokračovat novým řádkem. Pro jednotlivé vstupy se aplikuje následující CSS pravidlo:

```
width: calc(100% / 3);
padding: 10px;
```

Šířka vstupů se pomocí funkce `calc` nastaví přesně na třetinu šířky rodiče (obalující sekce) a ještě se přidá menší `padding` tak, aby mezi sebou vstupy měly trochu prostoru.

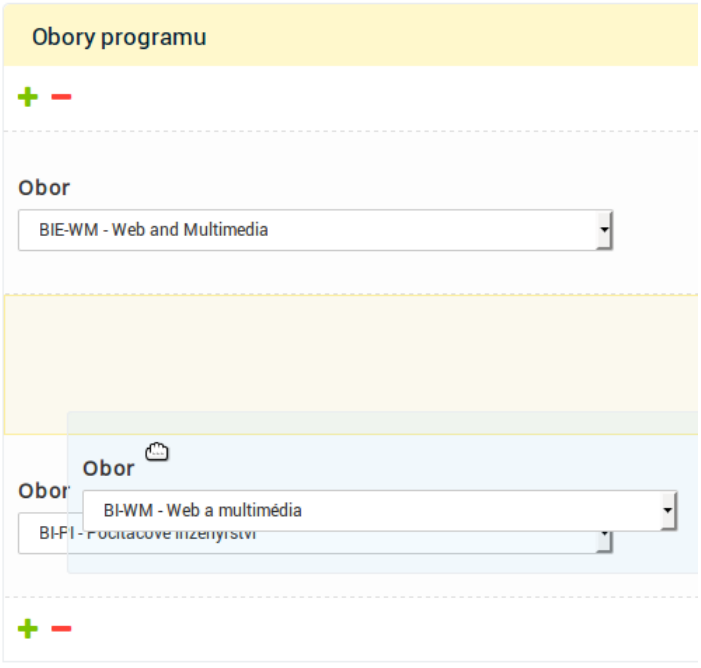
**Zabalitelné sekce** V předchozí verzi aplikace byly zabalitelné sekce použité pouze ve formuláři programu. Rozhodl jsem se je implementovat i do ostatních typů formulářů. Ušetří se spousta vertikálního prostoru a formuláře tak vypadají, dle mého názoru, daleko přehledněji.

Zabalitelné sekce jsou realizované pomocí JS komponenty knihovny Bootstrap `collapse.js`, což je v jádře plugin do knihovny jQuery. Inicializace tohoto pluginu probíhá v metodě `setupCollapisbleSections` v rámci objektu `Form`. Sekce jsou ve výchozím stavu zabalené. Po kliknutí na jejich hlavičku dojde k zabalení/rozbalení.

**Vícepoložkové sekce** Do kategorie zabalitelných sekcí patří i sekce vícepoložkové. Položky jsou do těchto sekcí přidávány pomocí zeleného tlačítka ve tvaru plus v horní a ve spodní části sekce. Horní tlačítko přidává položku na začátek a spodní na konec. Analogicky fungují i obě tlačítka ve tvaru mínus, ale ta naopak položky odstraňují. Libovolnou položku lze odstranit i kliknutím na šedý křížek v pravém horním roku položky.

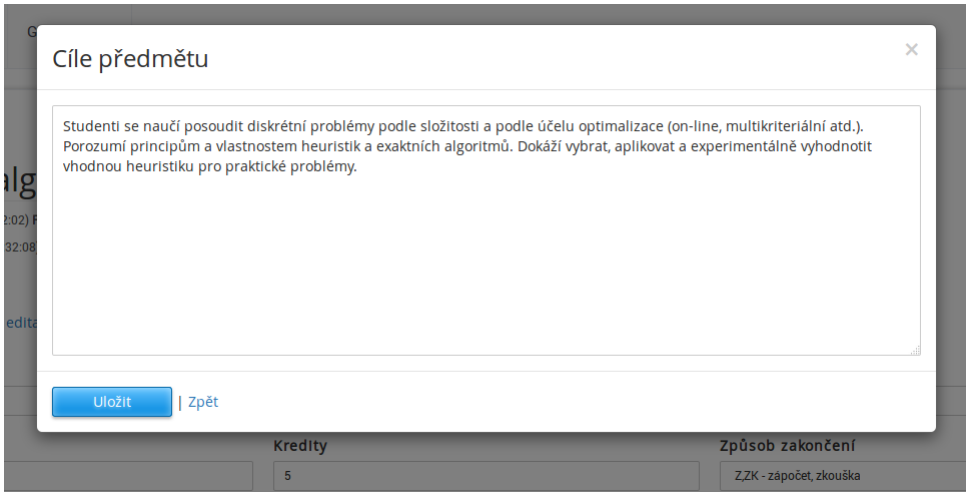
Položky v těchto sekcích lze libovolně řadit pomocí tahu myší (Drag & Drop). Pořadí položek ve formuláři je totiž plně reflektováno do pořadí položek, vypsaných v jednotlivých formulářích výsledného akreditačním spisu.

Tato funkcionalita je realizována knihovnou jQuery UI, což je rozšíření standardní knihovny jQuery o podporu různých ovládacích prvků uživatelského rozhraní. Inicializace řazení se nachází v objektu `Sortable`, který obsahuje i metodu pro zákaz řazení (pro verzi formuláře pouze pro čtení). Samotnou logiku řazení zajišťuje knihovna jQuery UI. Ukázku řazení lze vidět na obrázku 5.6.




Obrázek 5.6: Řazení uvnitř vícepoložkové sekce

**Editace vstupů textarea** Pro pohodlnější editaci vstupů typu `textarea` se při kliknutí na tento element objeví vyskakovací okno, které obsahuje daleko větší prostor pro editaci. Uživatel upravuje text uvnitř tohoto okna a po kliknutí na tlačítko „Uložit“ se změny projeví i v původním vstupu. Vyskakovací okno je implementováno pomocí Bootstrap komponenty `modal.js`. Ukázka je na obrázku 5.7.




Obrázek 5.7: Vyskakovací okno pro pohodlnější editaci vstupů textarea

**Import publikací** Na stránce s formulářem životopisu se v sekci Publikace nachází v horní části tlačítko „Importovat“, viz obrázek 5.8. Po kliknutí na toto tlačítko se uživateli otevře podobný seznam jako na obrázku 5.3. V tomto seznamu jsou publikace osoby, které patří otevřený životopis, získané ze služby VVVS API, seřazené od nejnovější po nejstarší. Vybrané publikace se ihned neukládají do databáze, ale pouze se jimi naplní sekce Publikace. Samotné uložení probíhá až po odeslání formuláře celého životopisu.


Obrázek 5.8: Tlačítko pro import publikací

## 5.5 Import dat z KOS a VVVS API

### 5.5.1 OAuth 2.0

Obě služby podporují autentizaci přes protokol OAuth 2.0. KOS API ještě podporuje jednodušší autentizaci HTTP Basic, její podpora bude ale v budoucnu zrušena. VVVS API podporuje pouze OAuth 2.0.

V sekci 3.7 jsem uvedl, že v případě importu dat ze školních systémů se při autentizaci použije tzv. grant type Client credentials, kde se aplikace (webová aplikace AKRMAT) autentizuje sama za sebe.

**Získání přístupového tokenu** Nejprve je nutné v aplikaci ČVUT Apps Manager[24] zaregistrovat aplikaci AKRMAT. Při registraci se vybere „Service Account“ jako typ aplikace. Automaticky se nově registrované aplikaci přiřadí dva důležité řetězce: `Client ID` (unikátní identifikátor) a `Client secret` (obdoba hesla). Tyto dva řetězce jsou potřeba k úspěšnému získání přístupového tokenu a jsou uvedeny v konfiguračním souboru aplikace `config.php`.

Třída `CTUApiModel` slouží k posílání HTTP požadavků na obě služby a také k získání přístupového tokenu. Ten lze získat odesláním požadavku na autorizační server dostupný na URL `https://auth.fit.cvut.cz/oauth/authorize`. Do těla tohoto požadavku se přiloží parametr `grant_type` s hodnotou `client_credentials` a ještě je nutné požadavek doplnit o HTTP hlavičku `Authorization`. Hodnotou této hlavičky je spojení řetězců `"Basic"` a řetězce, který vznikne spojením dvojtečkou oddělených řetězců `Client ID` a `Client secret` v kódování Base64. Hlavička může vypadat např. takto:

```
Authorization: Basic ZWE2YzY0ZDctYjg0ZS00ODg0LTljMjMtYyZGRhYz...
```

Přístupový token je odeslán zpět v těle HTTP odpovědi, v tomto konkrétním případě ve formátu JSON. Obsahem odpovědi je samotný token, jeho typ, platnost a tzv. scope, tj. vymezení systémových oprávnění aplikace.

Zdrojový kód 5.4: Tělo HTTP odpovědi s přístupovým tokenem

```
{
  "access_token": "3f06d53f-658a-4b96-bac6-52f575b66a00",
  "token_type": "bearer",
  "expires_in": 3240,
  "scope": "urn:ctu:oauth:vvvsapi.read
     urn:ctu:oauth:kosapi:public.readonly"
}
```

Získaný token platí dle aktuálního nastavení autorizačního serveru 1 hodinu. Zde jsem si implementaci mírně zjednodušil a jeho platnost v aplikaci

[24] `https://auth.fit.cvut.cz/manager/`

nekontroluji. Při každém požadavku o data se posílá i požadavek o token. Jelikož import dat není často používaná funkcionalita, jeden HTTP požadavek navíc nemá výrazný vliv na plynulost práce s aplikací.

**Získání dat** Přístupový token je nutné přiložit ke každému požadavku o data. Přikládá se opět do HTTP hlavičky `Authorization`, tentokrát s hodnotou `Bearer` a přístupovým tokenem. Hlavička může vypadat např. takto:

```
Authorization: Bearer 3f06d53f-658a-4b96-bac6-52f575b66a00
```

Tento HTTP požadavek se posílá na URL požadovaného zdroje KOS nebo VVVS API. Pro KOS API k tomu slouží třída `KOSApiModel`, pro VVVS API `VVVSApiModel`. Odpovědí jsou data ve formátu XML Atom Feed. Níže je uvedený seznam URL zdrojů, které jsou relevantní pro potřeby systému AKRMAT.

**KOS API** Základ URL je `https://kosapi.fit.cvut.cz/api/3`.

- **Předmět** – `/courses/{kod_predmetu}?detail=1`
- **Životopis** – `/teachers/{username}`
- **Obor** – `/branches/{kod_oboru}`
- **Program** – `/programmes/{kod_programu}`

**VVVS API** Základ URL je `https://vvvsapi.fit.cvut.cz/api/v2`.

- **Publikace** – `/people/{username}/publishedDocuments`

### 5.5.2 KOS API

#### 5.5.2.1 Parsování získaných dat

Získaná data je nutné ještě parsovat, tedy převést do určitého formátu tak, aby je bylo možné snadno uložit do databáze. Nejprve se obdržený obsah nahraje do vestavěné PHP třídy `SimpleXMLElement`, která umožňuje pohodlně pracovat s XML dokumentem. Vybraná data se poté transformují přímo v příslušné třídě Modelu do stejné podoby, ve které jsou data odesílána z formulářů webové aplikace. Díky tomu lze použít pro obě činnosti stejné rozhraní pro ukládání.

**Mapování jména na uživatelské jméno** Pokud je v datech získaných z KOS API nějaká vazba na určitou osobu, např. garant předmětu, je tato osoba identifikována pomocí celého jména včetně titulů, nikoli pomocí uživatelského jména. Pro uložení dat do databáze je ale nezbytné tato jména nějakým způsobem transformovat na uživatelská jména.

Nejjednodušším způsobem se mi jevilo mapování na životopisy uložené v databázi systému AKRMAT. Metoda `nameToUsername` ve třídě `KOSApiModel` zajistí prohledání všech uložených životopisů a porovná celé jméno včetně titulů s tím obdrženým z KOS API. Pokud je nalezena shoda, je z této funkce vráceno uživatelské jméno osoby. Jelikož je v některých případech hledané jméno součástí povinného elementu, i v případě neúspěšného hledání je nutné nějaké uživatelské jméno vrátit. Zvolil jsem v tomto případě jednoduše poslední kontrolované. Tím se uspokojí podmínka případné povinnosti elementu, nicméně je poté samozřejmě nutné provést ruční úpravu.

**Parsování přednášek** Data o přednáškách jsou v rámci předmětu dostupná jako dlouhý řetězec, který naštěstí obsahuje oddělovače nových řádků `\n`. Řetězec lze pomocí tohoto znaku rozdělit do pole, kde prvky pole reprezentují jednotlivé přednášky. Obsah jednoho prvku tohoto pole může vypadat např. takto:

```
4. Bezpečnost, autentizační moduly.
```

Z tohoto řetězce je ještě potřeba extrahovat pouze název přednášky. Pomocí PHP funkce `preg_split` lze řetězec rozdělit podle regulárního výrazu. V tomto případě bude regulární výraz vypadat následovně:

```
^[0-9]{1,2}[).]
```

Chceme oddělit číslo přednášky a následující tečku. Bylo vypozorováno, že některé předměty mají za číslem přednášky místo tečky kulatou závorku, proto jsou do regulárního výrazu zahrnuty obě varianty.

**Parsování literatury** Data o doporučené literatuře předmětů jsou sice dostupná, ale nemají vůbec žádnou ucelenou strukturu. Při důkladné analýze problému, jsem narazil minimálně na 5 různých formátů popisu doporučené literatury.

Jediná informace, která relativně dobře šla extrahovat, bylo ISBN[25] publikace. Nicméně i zde se projevila špatná konzistence formátu literatury napříč různými předměty. U většiny předmětů ale extrakce ISBN funguje.

[25] International Standard Book Number

#### 5.5.2.2 Problém s importy předmětů

Kódy některých předmětů obsahují tečku, např. BI-ADU.1 nebo BI-3DT.1. Po výběru předmětů, které se budou importovat a potvrzení importu dojde k odeslání formuláře a seznam vybraných předmětů je poté dostupný v PHP proměnné `$_POST`. Tato proměnná je asociativní pole, které ale nesmí obsahovat klíče s tečkou a všechny tečky automaticky nahrazuje za podtržítka. Proto je nutné před odesláním požadavku na KOS API upravit kódy všech předmětů a změnit všechna podtržítka zpět na tečky.

### 5.5.3 VVVS API

Parsování dat získaných z VVVS API probíhá velmi podobně jako parsování dat KOS API. Nicméně data z VVVS API nejsou okamžitě ukládána do databáze, proto je není potřeba transformovat do struktury pro uložení do databáze. Data jsou naopak připravena na vkládání do HTML formuláře, proto jsou klíče jednotlivých hodnot shodné s názvy vstupních polí HTML formuláře tak, aby vložení bylo co nejjednodušší.

Na konci parsovacího procesu jsou data pro jednu publikaci pomocí PHP funkce `json_encode` převedena do formátu JSON a vložena jako HTML atribut `data-json` elementu reprezentujícího řádek seznamu s výběrem publikací pro import. Po potvrzení importu se tato data pomocí JavaScriptu pouze přečtou a velmi jednoduše vloží jako nová položka sekce Publikace.

Získání seznamu publikací z VVVS API navíc probíhá asynchronně na pozadí pomocí technologie AJAX. Seznam se začne stahovat ihned po příchodu uživatele na stránku s formulářem životopisu. Po kliknutí na tlačítko „Importovat“, nemusí už uživatel čekat na načtení seznamu.

## 5.6 Manipulace s předměty s tečkou v kódu

Jak už jsem uvedl výše, některé předměty obsahují tečku v jejich kódu. Tato skutečnost způsobila problém pří přístupu ke zdrojům těchto předmětů pomocí AKRMAT REST API. URL zdroje takového předmětu může vypadat takto:

```
http://api.akrmat/predmety/BI-3DT.1
```

PHP framework Tonic, na kterém je AKRMAT REST API postaveno, totiž při parsování URL s tímto tvarem nepočítá a obsah ihned za první nalezenou tečkou považuje za doménu. V databázi je poté hledán předmět s chybným kódem, např. BI-3DT místo BI-3DT.1, a proto požadavek na výše uvedenou URL nebude správně fungovat.

Při konzultaci s vedoucím práce jsem se navíc dozvěděl, že v budoucnu na FIT je plánováno používat tečku v kódu předmětu mnohem častěji, proto bylo důležité vymyslet řešení tohoto problému. Tím je malá úprava konstruktoru

třídy `Request` v rámci frameworku Tonic. Explicitně se z URL extrahuje část s tečkou a číslem a připojí se ke kódu předmětu. Úprava konstruktoru spočívá v přidání následujících několika řádků zdrojového kódu na specifické místo:

```
// fix pro predmety s~teckou a cislem v~kodu
// napr. BI-3DT.1
if (is_numeric($parts[1])) {
  // jedna se o predmet s teckou a cislem v kodu
  // pripojime ji k url a odstranime tuto cast z pole casti
  $this->uri .= ".".$parts[1];
  unset($parts[1]);
  $parts = array_values($parts);
}
```

## 5.7 Zamykání editace dokumentů

**Nová metadata** Jak jsem uvedl už dříve v sekci 2.5.7.1, implementace zamykání editace dokumentů nebyla v předchozí verzi systému dotažena do konce. Každý dokument obsahoval v rámci metadat dva elementy, které sloužily k tomuto účelu. Jednalo se o elementy `zamceno` a `zmrazeno`. Druhý jmenovaný element nebyl vůbec používán a po konzultaci s vedoucím práce bylo rozhodnuto o jeho odstranění. Zbývající jeden element mi nicméně pro mou implementaci zamykání nestačil, proto byl obsah metadat rozšířen o další dva elementy. Nyní jsou pro tuto funkcionalitu vyhrazeny v rámci metadat dokumentu následující elementy:

- `zamkl`
- `zamceno`
- `platnostZamku`

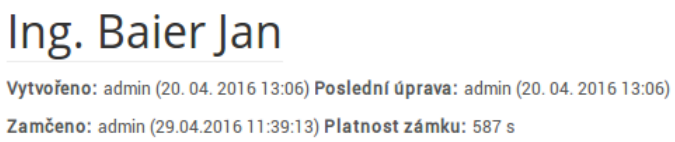
První element obsahuje uživatelské jméno osoby, která zámek vytvořila, druhý časovou značku, kdy byl zámek vytvořen a poslední element obsahuje zbývající platnost zámku v sekundách.

**Implementace zamykání/odemykání** Celý proces zamykání začíná kliknutím na tlačítko „Editovat“ na stránce formuláře. Samotné zamknutí spočívá v odeslání HTTP GET požadavku na zdroj dokumentu s parametrem `zamknout=true`. Na straně API proběhne při každém požadavku (i při obyčejném) na zdroj kontrola platnosti zámku. K tomu slouží nová pomocná třída `EditLockChecker`, ve které metoda `isLockTimedOut` provádí samotnou kontrolu. Pokud platnost zámku vypršela, vymaže se obsah relevantních metadat, a tím se zámek uvolní. Pokud stále zámek platí, je aktualizována jeho zbývající platnost v metadatech.

Zdrojový kód 5.5: Kontrola platnosti zámku editace dokumentu

```
// kontrola platnosti casoveho zamku editace
if ($document->getZamceno() != "") {
  $checker = new EditLockChecker();
  $timedOut = $checker->isLockTimedOut($document->getZamceno());
  if ($timedOut == true) {
    // zamek vyprsel, vymazeme ho
    $document->setZamkl("");
    $document->setZamceno("");
    $document->setPlatnostZamku("");
  } else {
    // jinak aktualizujeme jeho platnost
    $timeRemaining =
        $checker->getRemainingLockTime($document->getZamceno());
    $document->setPlatnostZamku($timeRemaining);
  }
}
```

Pokud dokument není uzamčen, kontroluje se, zda je uživatel oprávněn dokument editovat, tedy zda má přiřazenou roli Admin nebo je autorem dokumentu. Pokud je editace dokumentu uzamčena, jsou na stránce s formulářem tohoto dokumentu dostupné informace o platnosti zámku, viz obrázek 5.9




Obrázek 5.9: Informace na stránce formuláře o zamčení dokumentu

Odemknutí dokumentu lze provést explicitně odesláním HTTP GET požadavek na zdroj dokumentu s parametrem `odemeknout=true`, což lze provést stisknutím tlačítka „Zrušit editaci“, nebo se dokument automaticky odemkne při kontrole platnosti v rámci prvního HTTP požadavku po vypršení platnosti.

## 5.8 Generování akreditace a studijního plánu

Po úpravách datového modelu přestalo samozřejmě fungovat generování akreditačního spisu a studijního plánu. Pro opravení této funkcionality bylo potřeba reflektovat změny datového modelu ve většině tříd, které dědí od třídy `Generator`. Bylo nutné upravit některé selektory třídy `SimpleXMLElement` a XPath výrazy tak, aby obsahovaly správné cesty k XML elementům a jejich názvy. Na některých místech bylo také nezbytné přidat čtení nových XML dokumentů oborů.

**Jmenné konflikty v PHP 5.5** V sekci 3.5.2 jsem zmínil, že jsem zkoumal možnosti přechodu na novější verzi PHP. Při testování aplikace na verzi PHP 5.5 jsem narazil na několik chyb. Jejich příčinou byla třída `Generator`, která je základem pro logiku generování akreditačních materiálů v systému AKRMAT.

PHP 5.5 totiž obsahuje novou vestavěnou třídu, která nese stejný název a slouží k paměťově optimálnímu iterování. Nejjednodušším řešením se jevilo třídu `Generator` v systému AKRMAT přejmenovat. Protože hlavním úkolem této třídy je generování určitých souborů (převážně TeX šablon), nově se jmenuje `FileGenerator`.

**Parametrizace skriptu pro generování akreditace** Generování akreditace se spustí stisknutím tlačítka „Generovat akreditaci“ na stránce generátoru akreditace. V metodě `generateAction` třídy `AcredController` se pomocí PHP funkce `exec` spustí shell skript `gen_akred.sh`, který zajistí celkové vytvoření archivu s akreditačními materiály. Více o tomto procesu jsem uvedl v sekci 2.3.

Oproti předchozí verzi tohoto skriptu se jeho nová verze příliš neliší. Bylo odstraněno několik zakomentovaných částí a provedena větší parametrizace vstupů skriptu. V předchozí verzi byl parametrizován pouze kód studijního programu, ke kterému se generovaly akreditační materiály. Ve skriptu vystupuje i několik dalších parametrizovatelných informací. Jedná se o přihlašovací jméno a heslo uživatele a URL pro AKRMAT REST API. V předchozí verzi byly tyto informace uvedeny přímo ve skriptu a na mnoha místech se opakovaly. V nové verzi jsou nastavitelné v rámci konfiguračního souboru webové aplikace `config.php` a lze je tak snadno měnit.

Dále byl pro účely generování akreditace vytvořen speciální uživatel s uživatelským jménem generator a rolí Admin. To odstraní nutnost mít v konfiguračním souboru uvedené přístupové informace k nějakému reálnému uživateli.

## 5.9 Nastavení parametrů systému

V této sekci uvedu místa, kde lze nastavit určité parametry systému. Detailní popis těchto parametrů je k nalezení v instalační příručce v příloze C.

### 5.9.1 Konfigurační soubory

Celý systém využívá tří konfiguračních souborů.

#### REST API

- `serverSettings.php` – Obsahuje obecné nastavení prostředí PHP, přihlašovací údaje do obou databází a nastavení připojení k LDAP serveru.

- `settings.php` – Obsahuje veškeré nastavení načtení souborů s použitými třídami. Jediným parametrem, s jehož hodnotou by mohlo být častěji pohybováno, je doba platnosti zámku editace dokumentu nastavitelná v konstantě `EDIT_LOCK_TIMEOUT_IN_SECONDS`.

**Webová aplikace**

- `config.php` – Obsahuje veškerá nastavení týkající se webové aplikace. Například přihlašovací údaje k MySQL databázi, nastavení LDAP připojení, nastavení generování akreditace nebo různá nastavení cest k jednotlivým adresářům. Mezi další parametry patří nastavení pro OAuth 2.0, kde lze upravit hodnoty řetězců `Client ID` a `Client secret` a URL autorizačního serveru. Libovolně lze také měnit maximální počet povolených publikací v rámci životopisu v konstantě `MAX_CV_PUBLICATIONS`.

### 5.9.2 Transformace HTML formulářů

Aby byly HTML formuláře čitelné a maximálně použitelné, jsou na ně aplikovány pomocí JavaScriptu různé transformace. Používají se 4 typy transformací:

- Přepis popisků vstupních polí
- Změna vybraných textových vstupů na vstupy typu `textarea`
- Změna vybraných textových vstupů na vstupy typu `select`
- Přidání tooltipů nápovědy

Všechny tyto transformace jsou definovány ve třídě `FormModifications`. Každá je umístěna ve svém statickém asociativním poli, kde klíče pole slouží k vybrání správných transformovaných HTML elementů. Změna nebo přidání nových položek pro tyto transformace je tedy velmi snadným úkolem.

Kapitola 6

# Testování

Tato kapitola je věnována testování nové verze systému AKRMAT. Obsahem kapitoly je popis použitých testovacích technik, celkové vyhodnocení testů a výpis seznamu identifikovaných problémů.

## 6.1 Druhy prováděných testů

Během vývoje bylo využito dvou druhů testování. Manuální testy, které se prováděly po implementaci každé nové dílčí funkcionality a uživatelské testy, které byly provedeny na samém konci vývojového procesu. Nabízelo se i využití jednotkových testů na ověření správnosti jednotlivých tříd a metod. Jelikož aplikace není příliš komplexní, bylo rozhodnuto, že se tento druh testů provádět nebude a místo toho bude kladen důraz na důkladnější manuální testování dílčích funkcionalit během vývoje.

## 6.2 Průběžné manuální testy

Vždy po implementaci dílčí funkcionality byl proveden její důkladný test na různé mezní situace a případné nalezené chyby ihned opraveny. Dalšímu rychlému otestování byly podrobeny i dříve implementované funkcionality tak, aby se vyloučilo zavedení nových chyb. Do další fáze se vývoj posunul až v případě, že testy neodhalily žádné problémy. Cílem tohoto vývojového postupu bylo minimalizovat případnou akumulaci chyb.

## 6.3 Uživatelské testy

Tento druh testování byl proveden na samém konci vývojového procesu nové verze systému AKRMAT. Náplní uživatelských testů je zkoumat přívětivost uživatelského rozhraní a celkovou použitelnost aplikace. Hodnocení probíhá

na základě vyplněných dotazníků uživateli nebo vyhodnocením testovacích scénářů. V tomto případě bylo využito pouze hodnocení scénářů.

Testování pomocí scénářů spočívá v tom, že uživateli (testerovi) je předložen testovací scénář, což je posloupnost dílčích úkolů, které většinou vedou k nějakému ucelenému cíli. Ten by měl reprezentovat jeden z hlavních případů užití testované aplikace. Tester plní jednotlivé dílčí úkoly, je pozorován a poskytuje zpětnou vazbu vývojářům. Na základě nasbíraných informací jsou identifikovány chyby aplikace a ohodnocena jejich závažnost. Oprava poté probíhá většinou prioritně podle závažnosti chyb.

### 6.3.1 Testovací scénář – test hlavních funkcionalit systému

Vzhledem k tomu, že systém není příliš komplexní, lze veškerou hlavní funkcionalitu otestovat pomocí jediného delšího scénáře. Cílem je ověřit funkčnost následujících akcí:

- vytvoření nového dokumentu pomocí importu dat z KOS API
- vytvoření nového dokumentu ručně v aplikaci
- zobrazení, zamykání a editace dokumentu
- import publikací z VVVS API
- generování studijního plánu
- generování akreditace
- mazání dokumentu

**Scénář:**

1. Uživatel se přihlásí do aplikace.
2. Uživatel vytvoří nový životopis Ing. Lukáše Bařinky (barinkl) pomocí importu z KOS API.
3. Uživatel zobrazí importovaný životopis a uzamkne ho pro editaci.
4. Uživatel zkontroluje platnost zámku životopisu.
5. Uživatel provede následující úpravy životopisu:
   - import několika publikací z VVVS API
   - odstranění poslední importované publikace
   - prohození pořadí dvou publikací
   - přidání položky do sekce Vzdělání

6. Uživatel uloží provedené úpravy životopisu.

7. Uživatel ručně vytvoří nový předmět MI-TEST a jako garanta nastaví Ing. Bařinku.

8. Uživatel vytvoří nový obor Počítačové sítě a systémy (MI-PSS) pomocí importu z KOS API.

9. Uživatel přiřadí oboru MI-PSS předmět MI-TEST jako povinný oborový a nastaví jeho doporučený semestr.

10. Uživatel přiřadí obor MI-PSS do magisterského studijního programu Informatika.

11. Uživatel vygeneruje studijní plán výše uvedeného programu a zkontroluje, zda jsou v něm obsaženy informace o oboru MI-PSS a předmětu MI-TEST.

12. Uživatel vygeneruje akreditační spis k výše uvedenému programu a provede kontrolu, zda spis obsahuje vyplněné formuláře C pro obor MI-PSS, D pro předmět MI-TEST a G pro Ing. Bařinku.

13. Uživatel odstraní předmět MI-TEST.

14. Uživatel se odhlásí z aplikace.

### 6.3.2 Průběh testování

Uživatelské testy probíhaly v prostorách FIT na notebooku nad testovací databází a zúčastnily se jich celkem 3 osoby (testeři). Každý tester dostal za úkol projít výše uvedený scénář. V průběhu testování jsem si dělal poznámky a zodpovídal případné otázky. Pro testovaní byli vybráni dva uživatelé, kteří už mají zkušenosti s předchozí verzí systému AKRMAT a jeden, který s tímto systémem nikdy nepřišel do kontaktu. Díky zařazení neznalého uživatele do testování lze, dle mého názoru, lépe vyhodnotit použitelnost uživatelského rozhraní. Testery konkrétně byli:

- **Ing. Michal Valenta, Ph.D.** – vedoucí Katedry softwarového inženýrství na FIT, administrátor předchozí verze systému AKRMAT

- **Markéta Loučková** – sekretářka děkana FIT, administrátor předchozí verze systému AKRMAT

- **Bc. Josef Němeček** – student FIT, neznalý uživatel

## 6.4 Závěry z testování

Zde jsou zmíněny veškeré chyby, které byly identifikovány při uživatelských testech. U každé chyby je určen stupeň závažnosti a uvedeno, zda byla opravena. Chyb odhalených v rámci průběžných manuálních testů bylo velké množství a ihned se opravovaly, proto nejsou do této sekce zahrnuty.

### 6.4.1 Identifikované chyby

1. **Na hlavní stránce chybí vyhledávání v tabulkách.**
   - **Závažnost:** nízká
   - **Opraveno:** ne
2. **V seznamu výběru položek pro import chybí vyhledávání.**
   - **Závažnost:** nízká
   - **Opraveno:** ne
3. **Po dvojím kliknutí na tlačítko „Upravit“ na stránce s formulářem se odešle dvakrát požadavek o zamknutí dokumentu.**
   - **Závažnost:** nízká
   - **Opraveno:** ano
4. **Pro pedagoga, jehož životopis obsahuje prázdnou sekci Vzdělání, se nevytvoří formulář G v akreditačním spisu.**
   - **Závažnost:** vysoká
   - **Opraveno:** ano
5. **Pokud v rámci životopisu jsou v sekci Publikace prázdné položky, import publikací z VVVS API tyto položky nevyplní.**
   - **Závažnost:** nízká
   - **Opraveno:** ano
6. **Výpis výsledku importu neobsahuje odkazy na importované položky.**
   - **Závažnost:** nízká
   - **Opraveno:** ano
7. **Ve formulářových vstupech typu `select` nelze vyhledávat.**
   - **Závažnost:** nízká
   - **Opraveno:** ne

8. **Vygenerovaný studijní plán se neotvírá v rámci nového okna prohlížeče.**
   - **Závažnost:** nízká
   - **Opraveno:** ano

# Závěr

V rámci vypracování své diplomové práce jsem navrhl celkové vylepšení systému AKRMAT pro vytváření a správu akreditačních materiálů na FIT ČVUT. Návrh vylepšení zahrnuje přidání nové funkcionality a také novou webovou aplikaci, včetně nového uživatelského rozhraní. Navržená vylepšení jsem implementoval a důkladně otestoval. Při testování nebyly odhaleny žádné chyby, které by bránily použitelnosti systému.

Výsledkem je funkční systém, který, oproti jeho předchozí verzi, obsahuje mnoho vylepšení, např. dekompozici datového modelu na samostatné studijní obory a programy, integrovaný import dat nebo funkční zamykání editace dokumentů. Nové uživatelské rozhraní webové aplikace by navíc mělo zvýšit použitelnost systému a nová architektura webové aplikace společně s důrazem na kvalitu a přehlednost zdrojového kódu by měly usnadnit rozšiřitelnost v budoucnosti.

Mohu konstatovat, že zadání diplomové práce se podařilo splnit ve všech bodech. Implementována byla i většina požadavků nasbíraných mimo oficiální zadání. Nová verze systému by měla brzy nahradit verzi současnou a bude tedy k dispozici pro další schvalování akreditačních spisů studijních programů.

Osobně hodnotím vykonanou práci a její výsledek velmi kladně. Vyzkoušel jsem si práci s XML nativní databází Sedna a obecně si vylepšil své znalosti, které se týkají manipulace s XML dokumenty. Dále jsem si osvojil implementaci architektonického vzoru MVC bez použití PHP frameworku. Dle mého názoru jsem navíc získal další implementační zkušenosti, které mě posunou opět o kousek dál jako programátora.

## Budoucí rozvoj systému

Bohužel i přesto, že se povedlo splnit zadání diplomové práce a implementovat většinu dalších požadavků, na některé úpravy nezbyl časový prostor. V následujících odstavcích popíši úpravy, které by bylo vhodné v budoucnu realizovat

a ke každé též uvedu stručný návrh možného řešení. Tato sekce by měla sloužit jako jeden z podkladů pro případný další rozvoj systému AKRMAT.

### Parsování literatury při importu dat z KOS API

Z důvodu špatné konzistence formátování informací o doporučené literatuře v rámci předmětů v KOS API nebylo snadné realizovat jejich parsování. Relativně spolehlivě se podařilo parsovat pouze ISBN. Zde by možná mohl být aplikován komplexní regulární výraz, který by pokryl většinu typu formátování. Další možností by mohlo být rozdělení parosvání na více případů a na ty pak aplikovat jednodušší regulární výrazy. Případně by mohla být také provedena analýza možností nápravy na straně služby KOS API. Jsem přesvědčen, že tato skutečnost není problémem jen pro systém AKRMAT.

### Prodloužení platnosti zámku editace dokumentu

Aktuálně nelze žádným způsobem prodloužit platnost zámku editace dokumentu. Uživatel může pouze požádat o vytvoření nového zámku, ale ještě předtím musí předchozí zámek zrušit uložením nebo zrušením rozpracovaných změn ve formuláři. Navrhoval bych přidání tlačítka pro prodloužení platnosti zámku do uživatelského rozhraní.

### Změna autora dokumentu v aplikaci

Po importu dat z KOS API je jako autor vytvořených dokumentů označen uživatel, který import provedl. Aby si uživatelé s uživatelskou rolí Author mohli upravovat své životopisy, předměty a obory, musí být nejprve ručně změněn autor (vlastník) dokumentů. V ideálním stavu by mělo jít pohodlně měnit autora dokumentu přímo v aplikaci na stránce s formulářem. Také by mohla být přidána možnost nastavovat autora dokumentu už při importu dat.

### Vyhledávání v tabulkách, seznamech, apod.

V tabulkách na hlavní stránce a v seznamech výběru položek pro import nelze nijak vyhledávat. Nad každou tabulku nebo seznam by mohlo být umístěno vyhledávací pole a samotné vyhledávání zajištěno pomocí nějaké JS knihovny, např. DataTables[26].

Dále nelze vyhledávat ve formulářových vstupech typu `select`. Tato funkcionalita by byla vhodná zejména pro výběr předmětů, jelikož jich je v systému používáno velké množství.

[26] `https://www.datatables.net/`

### Export dat pro jiné účely

V minulosti bylo plánováno, že data ze systému AKRMAT budou použita i pro jiné účely. V rámci REST API bylo zřízeno rozhraní pro export dat do různých formátů vedle používaného `.xml`. Jedná se např. o formáty `.html`, `.tex` nebo `.bib`. Tato funkcionalita nebyla nikdy dokončena a je otázkou, zda by pro ni bylo vůbec nějaké využití.

### Překlad datového modelu do anglického jazyka

Dle vývojových koncepcí na FIT by datový model neměl obsahovat názvy v českém jazyce. V samotné implementaci se pak mísí české názvy z datového modelu s názvy proměnných, často v angličtině. Bohužel na řešení tohoto problému nebyl dostatečný časový prostor. V sekci 4.9.1 jsem alespoň provedl analýzu, na kterých místech by musela být provedena úprava, pokud by se názvy v datovém modelu přeložily do anglického jazyka.

# Literatura


[1] Kabelka, M.: *Aplikace pro správu akreditačních materiálů - část odborné životopisy učitelů*. Diplomová práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2012.

[2] Růžička, L.: *Aplikace pro správu akreditačních materiálů - část anotace předmětů*. Diplomová práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2012.

[3] Jambor, J.: *IS pro tvorbu a generování akreditačních materiálů*. Diplomová práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[4] Hout, J.: *IS pro akreditační materiály - verze 3.0 - úpravy a rozšíření*. Bakalářská práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

[5] Sahni, V.: *Best Practices for Designing a Pragmatic RESTful API*. [online]. [cit. 2016-04-15]. Dostupné z: `http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api`

[6] Stýblo, T.: *htmltmpl: templating engine for separation of code and HTML*. [online]. [cit. 2016-04-16]. Dostupné z: `http://htmltmpl.sourceforge.net/`

[7] Štefan, J.: *Systém pro hodnocení pedagogického výkonu - verze 3.0 - rozhraní pro úvazkáře a učitele*. Bakalářská práce, Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

[8] The Apache Software Foundation: *Httpd Wiki FAQ*. [online]. [cit. 2016-04-23]. Dostupné z: `http://wiki.apache.org/httpd/FAQ`

[9] Wikipedia - The Free Encyclopedia: *Apache HTTP Server*. [online]. [cit. 2016-04-23]. Dostupné z: `https://en.wikipedia.org/wiki/Apache_HTTP_Server`

[10] The Apache Software Foundation: *Apache Module mod_rewrite*. [online]. [cit. 2016-04-22]. Dostupné z: http://httpd.apache.org/docs/current/mod/mod_rewrite.html

[11] The Apache Software Foundation: *Name-based Virtual Host Support*. [online]. [cit. 2016-04-23]. Dostupné z: https://httpd.apache.org/docs/current/vhosts/name-based.html

[12] Wikipedia - The Free Encyclopedia: *Extensible Markup Language*. [online]. [cit. 2016-04-23]. Dostupné z: https://en.wikipedia.org/wiki/XML

[13] Tutorialspoint: *XML - Syntax*. [online]. [cit. 2016-04-23]. Dostupné z: http://www.tutorialspoint.com/xml/xml_syntax.htm

[14] Wikipedia - The Free Encyclopedia: *XML Schema (W3C)*. [online]. [cit. 2016-04-24]. Dostupné z: https://en.wikipedia.org/wiki/XML_Schema_(W3C)

[15] Wikipedia - The Free Encyclopedia: *XML database*. [online]. [cit. 2016-04-24]. Dostupné z: https://en.wikipedia.org/wiki/XML_database

[16] RestApiTutorial.com: *What Is REST?* [online]. [cit. 2016-04-24]. Dostupné z: http://www.restapitutorial.com/lessons/whatisrest.html

[17] Reinke, J.: *Understanding OAuth2*. [online]. [cit. 2016-04-24]. Dostupné z: http://www.bubblecode.net/en/2016/01/22/understanding-oauth2/

[18] Bean Software: *Easy Intro to ASP.NET MVC*. [online]. [cit. 2016-04-20]. Dostupné z: http://www.beansoftware.com/ASP.NET-Tutorials/Intro-ASP.NET-MVC.aspx

[19] W3Techs: *Usage of JavaScript libraries for websites*. [online]. [cit. 2016-04-26]. Dostupné z: http://w3techs.com/technologies/overview/javascript_library/all

[20] W3C: *CSS Flexible Box Layout Module Level 1*. [online]. [cit. 2016-04-26]. Dostupné z: https://www.w3.org/TR/css-flexbox-1/

[21] W3Schools: *Browser Display Statistics*. [online]. [cit. 2016-04-20]. Dostupné z: http://www.w3schools.com/browsers/browsers_display.asp

[22] Martin, R.: *Clean code : a handbook of agile software craftsmanship*. Upper Saddle River, NJ: Prentice Hall, 2009, ISBN 978-0-13-235088-4.

Příloha **A**

# Seznam použitých zkratek

**MŠMT** Ministerstvo školství, mládeže a tělovýchovy

**AKRMAT** Zkratka pro slovní spojení „akreditační materiály“

**KOS** Komponenta studium

**VVVS** Věda, výzkum a vedlejší styky

**XML** Extensible Markup Language

**API** Application Programming Interface

**REST** Representational State Transfer

**LDAP** Lightweight Directory Access Protocol

**PHP** Hypertext Preprocessor

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**HTTP** Hypertext Transfer Protocol

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

**CSS** Cascading Sytle Sheets

**JS** JavaScript

**UML** Unified Modeling Language

**XSD** XML Schema Definition

**XSLT** Extensible Stylesheet Language Transformations

**UI** User Interface

**MVC** Model, View, Controller

**IDE** Integrated Development Environment

**SSL** Secure Sockets Layer

**TLS** Transport Layer Security

**DTD** Document Type Definition

**ACID** Atomicity, Consistency, Isolation, Durability

**AJAX** Asynchronous Javascript and XML

**ISBN** International Standard Book Number

Příloha **B**

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `extras` ..... doplňkové soubory pro instalaci
- `src`
  - `impl`
    - `api` ..... zdrojové kódy AKRMAT REST API
    - `web-app` ..... zdrojové kódy webové aplikace AKRMAT
    - `dokuWiki` ..... zdrojové kódy pro dokuWiki
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

# Informace pro nasazení

Proces instalace nové verze systému je téměř totožný s předchozí verzí, ke které vytvořil instalační příručku ve své diplomové práci Bc. Jakub Jambor. Já na tuto příručku pouze navazuji. Její aktualizovanou podobu, ve formě dokuWiki stránky, lze nalézt v adresáři `dokuWiki` na přiloženém CD. V následujících sekcích jsou popsány nejdůležitější informace týkající se instalace nové verze systému.

## C.1 Instalace správné PHP verze

Budeme uvažovat instalaci PHP na linuxové distribuci Ubuntu. PHP lze nainstalovat z terminálu přes balíčkový systém `apt-get` pomocí příkazu:

```
apt-get install php5
```

Nainstalována bude nějaká verze PHP 5.x. Konkrétní verze pak záleží na tom, z jakého repozitáře se balíček s PHP stahuje a jaká verze Ubuntu je použita.

Pro správné fungování nové verze systému AKRMAT je nutné nainstalovat PHP 5.4 nebo 5.5. Na Ubuntu 14.04 se z výchozího repozitáře instaluje právě PHP 5.5, na novějších distribucích už PHP 5.6. Pokud je nezbytné systém nasadit na novější nebo naopak výrazně starší distribuci, je nutné buď změnit repozitář, ze kterého se PHP stahuje, nebo správnou verzi PHP ručně zkompilovat a nainstalovat.

Repozitář `ppa:ondrej/php5` obsahuje balíčky pro PHP 5.5 pro různé linuxové distribuce. Do programu `apt-get` se registruje pomocí příkazu:

```
add-apt-repository ppa:ondrej/php5
```

## C.2 Nastavení webového serveru Apache

Systém je rozdělen na dvě hlavní části a každá má svůj vlastní adresář (`api` a `web-app`). Pro každou část je vhodné vytvořit vlastní virtuální server. To lze zařídit přidáním několika direktiv do konfiguračního souboru Apache. Pokud se postupuje podle pokynů instalační příručky, je to soubor:

```
/etc/apache2/sites-available/001-AKRMAT.conf
```

Do tohoto souboru je nutné přidat následující řádky a upravit cesty do jednotlivých adresářů.

```
<VirtualHost *:80>
    DocumentRoot "{cesta k adresari s webovou aplikaci}"
    ServerName akrmat
    ServerAlias akrmat
    <Directory "{cesta k adresari s webovou aplikaci}">
        Options Indexes FollowSymLinks Includes ExecCGI
        AllowOverride All
        Order allow,deny
        Allow from all
   </Directory>
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "{cesta k adresari s api}"
    ServerName api.akrmat
    ServerAlias api.akrmat
    <Directory "{cesta k adresari s api}">
        Options Indexes FollowSymLinks Includes ExecCGI
        AllowOverride All
        Order allow,deny
        Allow from all
   </Directory>
</VirtualHost>
```

Dále je nutné do souboru `/etc/hosts` přidat následující řádky:

```
127.0.0.1 akrmat
127.0.0.1 api.akrmat
```

Po restartování serveru Apache příkazem `service apache2 restart` už by mělo být AKRMAT REST API dostupné na `http://api.akrmat` a webová aplikace na `http://akrmat`.

## C.3 Nastavitelné konstanty konfiguračních souborů

Systém používá 3 konfigurační soubory: `serverSettings.php`, `settings.php` v adresáři `api` a `config.php` v adresáři `web-app`. Tyto soubory obsahují různé definice konstant. Zde popíši pouze ty, které byly přidány v nové verzi systému.

**Konfigurační soubor settings.php**

- `EDIT_LOCK_TIMEOUT_IN_SECONDS` – Doba platnosti zámku editace dokumentu v sekundách. Po uplynutí tohoto intervalu se jeho zámek při dalším požadavku na uzamčený dokument automaticky uvolní.

**Konfigurační soubor config.php**

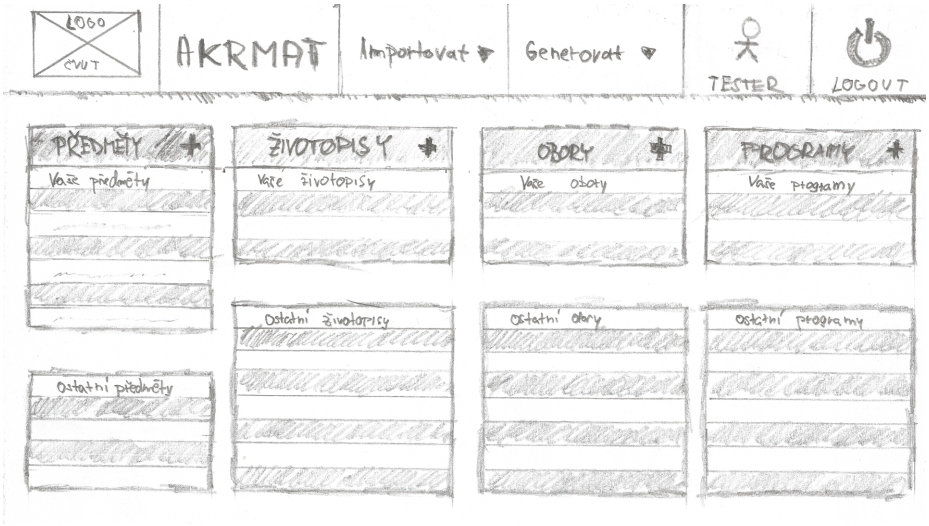
- `APP_DIR` – Relativní cesta do adresáře z kořenu webového serveru. Pokud aplikace běží v kořenu, bude hodnotou této konstanty prázdný řetězec.
- `AKRMAT_API_ENDPOINT` – URL, na které je dostupné AKRMAT REST API.
- `CONTROLLER_NAMESPACE` – PHP namespace pro třídy vrstvy Controller.
- `MAX_CV_PUBLICATIONS` – Maximální počet povolených publikací ve formuláři životopisu.
- `OAUTH_CLIENT_ID` – Řetězec `Client ID` pro protokol OAuth 2.0. Nový lze získat registrací aplikace na portálu Apps Manager ČVUT[27] a výběrem „Service Account“ jako typ aplikace.
- `OAUTH_CLIENT_SECRET` – Řetězec `Client secret` pro protokol OAuth 2.0. Postup pro získání nového řetězce je stejný jako u `Client ID`.
- `OAUTH_ENDPOINT` – URL, na které je dostupný autorizační server ČVUT pro protokol OAuth 2.0.
- `OAUTH_TOKEN_URL` – Relativní URL v rámci autorizačního serveru, na které lze získat přístupový token v protokolu OAuth 2.0.
- `KOSAPI_ENDPOINT` – URL, na které je dostupná služba KOS API.
- `VVVSAPI_ENDPOINT` – URL, na které je dostupná služba VVVS API.
- `ABOSLUTE_ROOT_FILESYSTEM_PATH` – Absolutní cesta v rámci systému souborů do kořene webové aplikace.

[27] `https://auth.fit.cvut.cz/manager/`

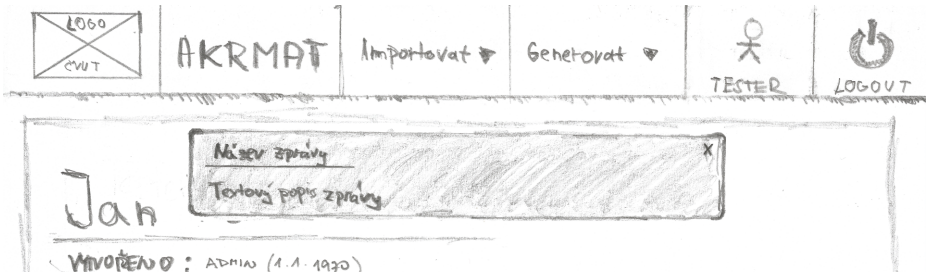
- `ACRED_GEN_DIR` – Cesta k adresáři, kde se nachází skript pro generování akreditace.
- `ACRED_GEN_SCRIPT_FILENAME` – Jméno skriptu pro generování akreditace.
- `ACRED_GEN_USER` – Uživatelský účet, pomocí kterého se bude provádět generování akreditace.
- `ACRED_GEN_PASSWORD` – Heslo k uživatelskému účtu, pomocí kterého se bude provádět generování akreditace.
- `ACRED_GEN_PROGRAMY_URL` – URL, na které je dostupný seznam studijních programů v rámci AKRMAT REST API.

Příloha D

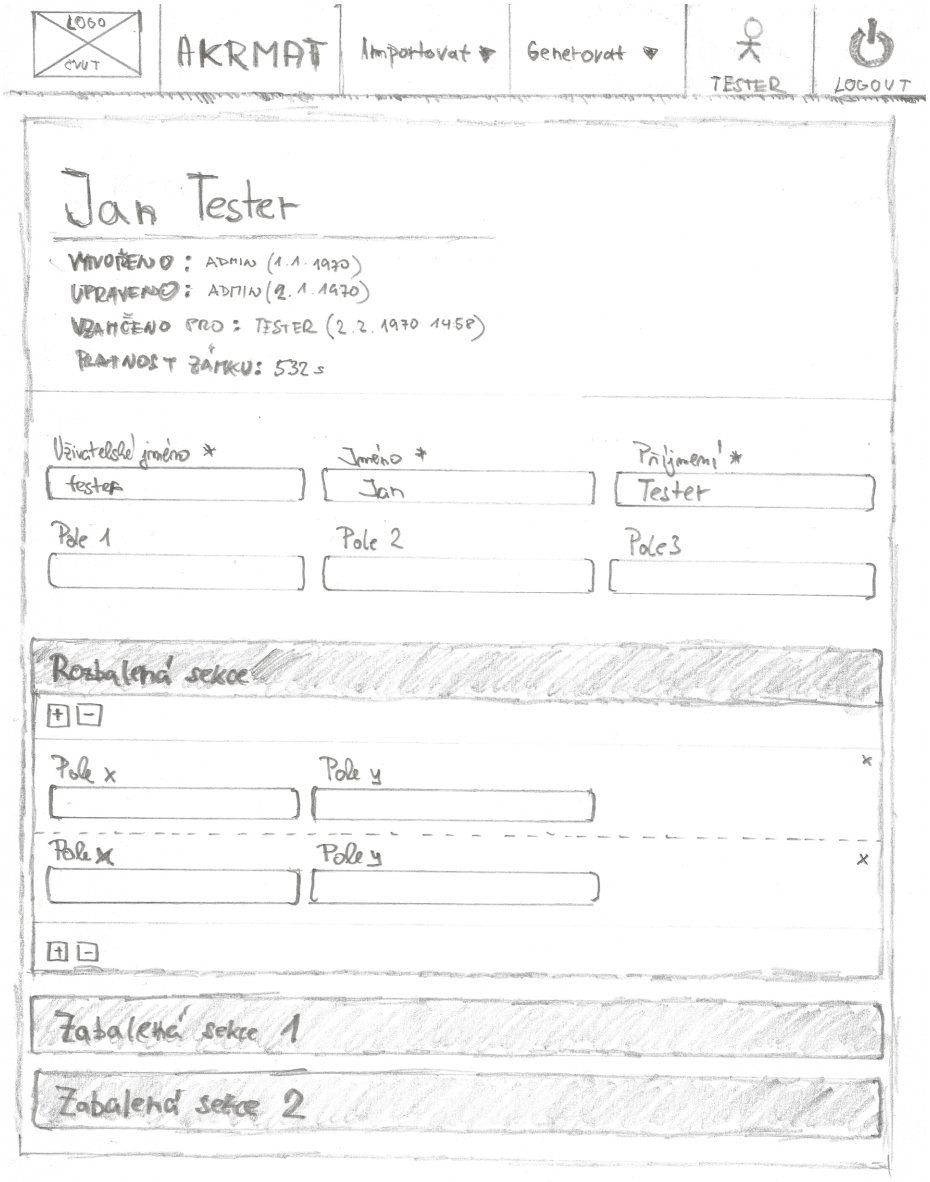
# Obrázkové přílohy



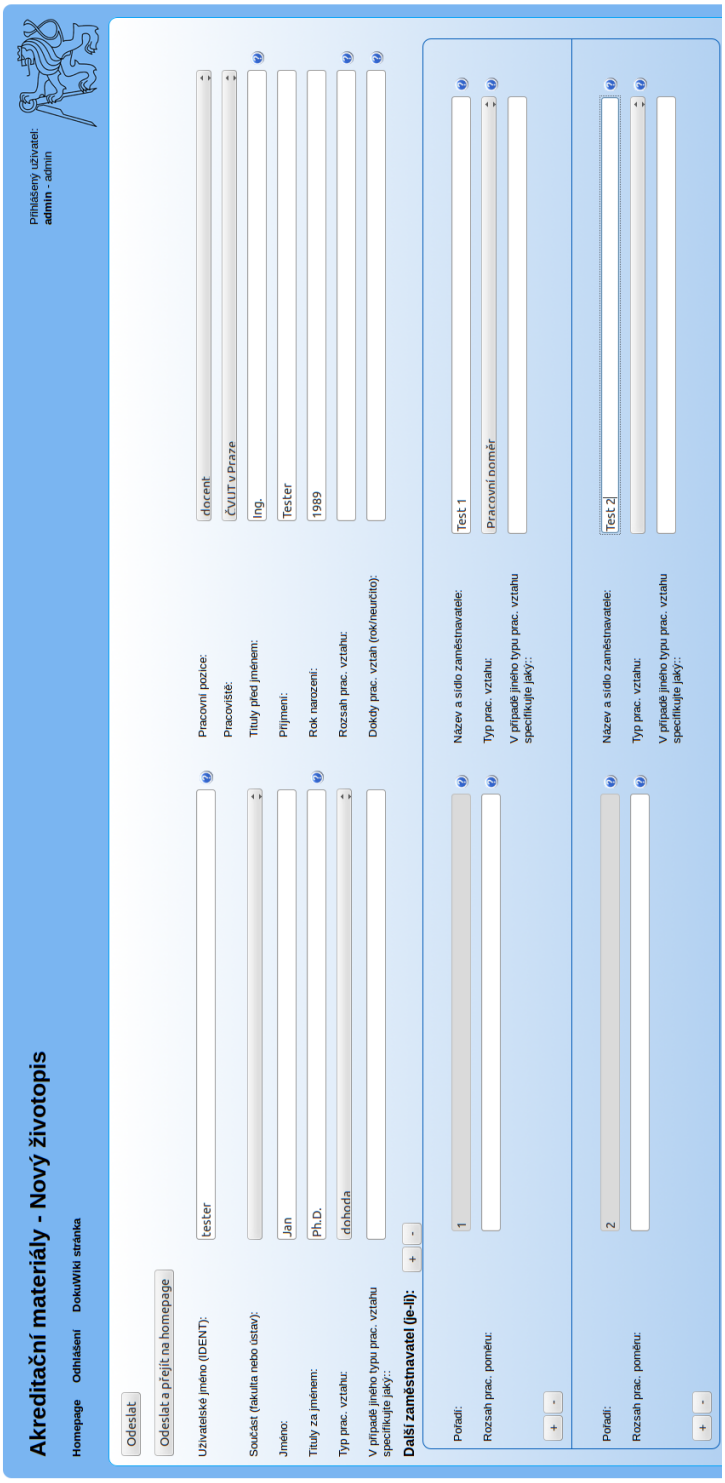

Obrázek D.1: Wireframe (návrh) hlavní stránky




Obrázek D.2: Wireframe (návrh) zobrazení zprávy uživateli




Obrázek D.3: Wireframe (návrh) stránky s formulářem



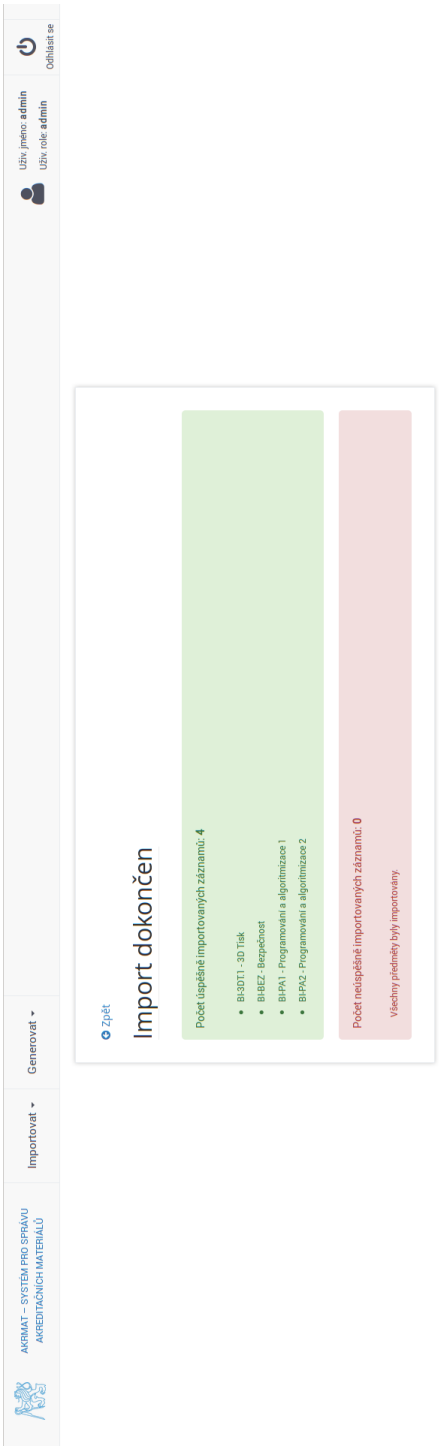

Obrázek D.4: Stránka s formulářem předchozí verze webové aplikace AKR-MAT


Obrázek D.5: Stránka pro přihlášení do nové webové aplikace AKRMAT.

Obrázek D.6: Hlavní stránka nové webové aplikace. Stránka obsahuje vedle sebe 4 tabulky, jednu pro každou entitu vystupující v systému.


Obrázek D.7: Stránka s formulářem studijního programu v nové webové aplikace. Dokument studijního programu je uzamčen uživatelem pro editaci, a proto je formulář editovatelný.

Obrázek D.8: Stránka se seznamem předmětů pro import z KOS API v nové webové aplikaci. Uživatel si může ze seznamu vybrat, které položky se boudou importovat.




Obrázek D.9: Stránka se souhrnem právě proběhlého importu v nové webové aplikaci. Horní sekce (zelená) obsahuje seznam úspěšně importovaných a spodní (červená) obsahuje seznam neúspěšně importovaných položek.