UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Systém pro rezervaci a hodnocení témat prezentací

Bc. Josef Kudláček

Diplomová práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Josef Kudláček**
Osobní číslo:	**I16230**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém pro rezervaci a hodnocení témat prezentací**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Teoretická část se bude zabývat návrhem informačního systému (databáze a webového rozhraní). Pozornost bude věnována též možnostem hromadného importu a exportu dat a možných formátů (např. CSV, JSON, XML) a možnosti interoperability s jiným IS pomocí definovaných rozhraní (REST, SOAP).
Bude provedena analýza potřeb pro rezervaci a přidělování témat prezentací a jejich hodnocení v předmětu Správa operačních systémů (ISOSY a ISOSR) vyučovaného na FEI UPa. Na základě požadavků garanta předmětu bude navržen a implementován systém rezervací splňující jím požadované vlastnosti:
- import zapsaných studentů ze systému STAG (JSON či CSV) včetně rozvrhových skupin a cvičících,
- import a export témat a alternativ (v CSV) pro akademický rok, vyřazování a přeznačení témat a alternativ pro nový akademický rok,
- zadání nového (upřesňujícího) jména alternativy a poznámky studentem (při rezervaci) i přednášejícím či garantem (při přidělování témat i později, např. upřesnění zadání či důvody zamítnutí rezervace),
- řešení přednosti (dle časové známky rezervace a priority stanovené přednášejícím či garantem) a restrikcí rezervace alternativ (dle dovoleného limitu a rozvrhových skupin studentů),
- schvalování rezervací a přidělení tématu (i když nesplňuje restrikce) přednášejícím či garantem,
- přidělování data prezentace přednášejícím či garantem (nezávisle na schválení rezervace),
- bodové hodnocení prezentace cvičícím včetně slovních připomínek,
- zvlášť evidované hodnocení a připomínek přednášejícího či garanta (korekce hodnocení cvičícího),
- uložení rozpracovaného hodnocení i v případě vypršení platnosti autentizace (např. do cookie nebo lokálního úložiště prohlížeče),
- export studentů s hodnocením (v CSV),
- export seznamu referátů (rok, autor, název s odkazem, datum, hodnocení).

Systém bude implementován nad databází MySQL (MariaDB) v PHP s webovým rozhraním. Příslušné informace (seznamy a tabulky) budou vhodně a přehledně zobrazovány (studentům, cvičícím a přednášejícímu či garantovi) a bude je možné řadit dle zvolených kritérií pomocí databáze (s podporou řazení dle více sloupců); výchozí a alternativní řazení bude možné změnit a uložit (dle uživatele). Výsledný web musí splňovat příslušné normy W3C.

Příloha zadání diplomové práce

Seznam odborné literatury:

HUDEC, Tomáš: *Správa operačních systémů* **[online]. 2017 [cit. 2017-10-05]. Stránky k vyučovanému předmětu. URL: http://fei-as.upceucebny.cz/usr/hudec/vyuka/sos/**

PŘENOSIL, Vít: *Rezervační systém témat prezentací.* **Univerzita Pardubice, 2011. Bakalářská práce, vedoucí Tomáš Hudec. Dostupné též online z vnitřního IS UPa.**

PŘENOSIL, Vít – HUDEC, Tomáš: *Rezervační systém předmětu Správa operačních systémů* **[online]. 2011–2017 [cit. 2017-10-05]. Stávající systém pro rezervace (modifikace systému BP).**

Webové služby nad IS/STAG **[online]. Univerzita Pardubice, 2017 [cit. 2017-10-05]. URL: https://stag-ws.upce.cz/**

LACKO, Luboslav: *PHP 5 a MySQL 5: Hotová řešení.* **Computer Press, 2007. ISBN 978-80-251-1695-1.**

MOLINARO, Anthony: *SQL: Kuchařka programátora.* **Computer Press, 2009. ISBN 978-80-251-2617-2.**

KOFLER, Michael: *Mistrovství v MySQL 5: Kompletní průvodce webového vývojáře.* **Computer Press, 2007. ISBN 978-80-251-1502-2.**

KOFLER, Michael – ÖGGL, Bernd: *PHP 5 a MySQL 5: Průvodce webového programátora.* **Computer Press, 2007. ISBN 978-80-251-1813-9.**

ZeldMAN, Jeffrey: *Tvorba webů podle standardů: XHTML, CSS, DOM, ECMASkript a dalších.* **Computer Press, 2011. ISBN 978-80-251-0347-0.**

LUBBERS, Peter – ALBERS, Brian – SALIM, Frank: *HTML5: Programujeme moderní webové aplikace.* **Computer Press, 2011. ISBN 978-80-251-3539-6.**

CHAFFLER, Jonathan – SWEDBERG, Karl: *Mistrovství v jQuery: Kompletní průvodce vývojáře.* **Computer Press, 2013. ISBN 978-80-251-4103-8.**

BÖHMER, Marian: *Návrhové vzory v PHP.* **Computer Press, 2012. ISBN 978-80-251-3338-5.**

Rozsah grafických prací:	**10**
Rozsah pracovní zprávy:	**60**
Forma zpracování diplomové práce:	**tištěná**
Seznam odborné literatury:	**viz příloha**

Vedoucí diplomové práce:	**Mgr. Tomáš Hudec** Katedra informačních technologií
Datum zadání diplomové práce:	**30. října 2017**
Termín odevzdání diplomové práce:	**18. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 25. 5. 2018

Bc. Josef Kudláček

Poděkování

Rád bych poděkoval svému vedoucímu práce, Mgr. Tomáši Hudcovi, za nezbytné konzultace pro možné další využití v rámci funkcionalit a také za vedení diplomové práce. Mé poděkování také patří Ing. Romanu Divišovi, který připravil webový server pro provoz rezervačního systému. Nakonec bych rád poděkoval svým blízkým za podporu při studiu a zpracování této práce.

Anotace

Diplomová práce se zabývá vytvořením infomačního systému pro rezervaci témat prezentací a jejich možné hodnocení. Informační systém je navržen pro účely předmětu Správa operačních systémů vyučovaném na Fakultě elektrotechniky a informatiky Univerzity Pardubice.

Práce popisuje proces vývoje informačního systému na základě metodik zmíněných v teoretických východiscích. Stěžejní částí vývoje je analýza požadavků a funkcionalit systému. Na základě analýzy je poté systém implementován.

Součástí práce jsou analytické diagramy, navržený E-R diagram a implementovaný systém ve frameworku PHP Nette společně s využitím javascriptové knihovny jQuery nad databází MySQL.

Klíčová slova

Informační systém, analýza, Nette framework, PHP, MySQL.

Title

The System for Reservation and Assessment of Presentations

Annotation

This master's thesis deals with creating an information system for reservation of presentation topics and subsequent recording of teacher's assessment. The information system is designed for purposes of the Management of Operating Systems course taught at the Faculty of Electrical Engineering and Informatics, University of Pardubice.

The thesis describes the development of an information system based on the methods mentioned in the theoretical part. The key part of development is the analysis of requirements and system functionalities. The system implementation is based on the analysis.

Parts of the thesis include the analytical diagrams, the E-R diagram and information system itself implemented in the PHP framework Nette with usage of the jQuery JavaScript library on the top of the MySQL database.

Keywords

Information system, analysis, Nette framework, PHP, MySQL.

Obsah

Seznam zkratek

1NF	První normální forma
2NF	Druhá normální forma
3NF	Třetí normální forma
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CMS	Content Management System
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DDL	Data Definition Language
EA	Enterprise Architect
E-R	Entity-Relationship
FTP	File Transfer Protocol
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IS	Informační systém
IT	Informační technologie
ISOSY	Správa operačních systémů (obor IT), kód předmětu v IS UPa
ISOSR	Správa operačních systémů (obor ŘP), kód předmětu v IS UPa

jQuery	JavaScriptová knihovna
JSON	Javascript Object Notation
LAMP	Linux, Apache, MySQL, PHP
LS	Letní semestr
MVC	Model, View, Controller
NetID	Uživatelské jméno v IS Univerzity Pardubice
PHP	PHP: Hypertext Preprocessor
REQ	Requirement
REST	Representational State Transfer
ŘP	Řízení procesů (obor)
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
STAG	Studijní agenda, IS Univerzity Pardubice
SW	Software
UC	Use Case
UDDI	Universal Description, Delivery and Integration
UP	Unified Process
UPa	Univerzita Pardubice
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VPN	Virtual Private Network

W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	Extensible Markup Language
XP	Extrémní programování
XSS	Cross-Site Scripting
ZS	Zimní semestr

Seznam obrázků

Seznam ukázek kódu

Seznam tabulek

Úvod

Metody vývoje informačních systémů se neustále mění. Některé principy a metody však přesto stále zůstávají s případnými obměnami nedílnou součástí vývoje ve společnostech zaměřených na vývoj či jednotlivci. Neexistuje vhodný univerzální způsob, jak vyvinout informační systém.

Vždy je zapotřebí zvolit takový princip či metody, které budou odpovídat stylu vývoje a cílovému systému. Ať už se bude jednat o flexibilní vývoj či deterministický, projekt korporátních rozměrů nebo práci jednotlivce.

Máme-li jasno v tom, jakým způsobem se bude vývoj ubírat, může proces vývoje začít. Zpravidla je tento proces dle tradičního přístupu rozdělen do několika etap: specifikace požadavků, analýzy, návrhu, implementace, testování a nasazení. Etapy na sebe přímo navazují, jelikož na počátku každé se pracuje s výsledky předchozí.

První etapa se zaměřuje na to, co vlastně od výsledného systému očekáváme a jakými funkcionalitami by měl disponovat. O tuto etapu se primárně stará analytik, který při komunikaci se zákazníkem sbírá a shromažďuje požadavky. Z výsledných požadavků získáváme přibližnou představu nejen o funkcionalitách systému, ale také o jeho omezeních.

Druhou etapu tvoří analýza získaných požadavků. Cílem etapy je zachytit a popsat jednotlivé procesy funkcionalit systému. Docílíme toho kupříkladu modelováním případů užití a jednotlivých scénářů. Na základě případu užití lze následně navrhnout analytický model, jehož úkolem je popsat požadované chování systému pro snazší implementaci. Závěrem analýzy jsou pak diagramy aktivit, které znázorňují interakce vzniklých analytických tříd mezi sebou v rámci realizace případů užití.

Ve třetí etapě máme již k dispozici detailnější přehled nejen o tom, co má systém dělat, ale i jakým způsobem. Dle omezení systému či našich možností proto zvolíme technologie, ve kterých se bude systém implementovat a zpracuje se návrh implementace systému. V této práci pod návrh systému spadá E-R diagram na základě analytického modelu a vytvoření webového rozhraní.

Čtvrtou etapou začíná implementace systému. Implementace probíhá různými způsoby, základními jsou například od datové vrstvy po vykreslovací či naopak (bottom-up a top-down

development). Mimo to se také můžeme setkat s inkrementálními či iterativními přístupy i v rámci implementace.

V závislosti na výsledcích implementace pak přichází pátá etapa, jejímž cílem je ověřit funkčnost implementovaných funkcionalit. Například na základě stanovených scénářů během analýzy. Se zjištěnými nedostatky se pak pracuje dle jejich původu.

U implementačních chyb dochází k opravě v rámci implementace a opětovnému testování. V horších případech se jedná o chyby při analýze. Ty by se měly u rozsáhlých projektů opravit nejen implementací, ale také opětovným provedením analýzy nad zasaženou kritickou oblastí.

Poslední etapou je nasazení funkčního řešení do provozu. To bývá spojené především s potřebnými konfiguracemi serveru, na kterém naše řešení bude fungovat, a možnému přizpůsobení řešení pro daný server. Proces vývoje je tak ukončen a nahrazen procesem údržby.

Text práce je členěn do čtyř kapitol, přičemž jejím cílem je implementovat informační systém na základě analýzy definovaných požadavků garantem předmětu Správy operačních systémů, pro jehož potřeby je vyvíjen.

První kapitola seznamuje čtenáře s možnými principy a metodami při vývoji systému. Zároveň obsahuje seznámení s použitými technologiemi zvolenými pro implementaci a vybrané partie problematiky.

Druhá kapitola dokumentuje sběr požadavků a jejích následnou analýzu. Při analýze je využito modelování případů užití včetně ukázky jejich realizace za pomoci sekvenčních diagramů. Součástí analýzy je také stanovení aktérů systému, demonstrace některých scénářů případů užití a analytický model.

Třetí kapitola seznamuje čtenáře s problematikou normalizace a procesem vytvoření fyzického návrhu databáze od E-R diagramu, vycházejícího z analytického modelu, po generování výsledných skriptů DDL.

Čtvrtá kapitola mimo návrhu webového rozhraní demonstruje základní konstrukce frameworku Nette a ukázky kódu vybraných částí pro implementaci systému. Závěr kapitoly je věnován zajištění bezpečnosti systému v rámci implementace.

1 Teoretická východiska práce

První kapitola pojednává o stěžejních teoretických východiscích, na kterých je stavěna samotná diplomová práce. Oddíl 1.1 definuje pojem informační systém a jeho možné rozdělení zejména dle vynaloženého úsilí při implementaci či z hlediska finanční stránky. V oddíle 1.2 jsou stručně nastíněny metodiky vývoje při samotné tvorbě zvoleného typu IS. Kapitola je zakončena oddílem 1.3, která představuje použité technologie.

1.1 Informační systém (IS)

Informačním systémem bychom mohli nazvat koncept, který souhrnně označuje celek složitých věcí. Tento celek je však více než pouhým souhrnem a očekává se od něj jistá kvalita. Kvalita do takové míry, aby bylo možné určit podstatu celku, účel a cíl či charakteristické chování.

Označením IS si tedy hrubě můžeme představit systém pracující nad množinami objektů, uchovávající důležité informace. Jako hlavní cíl takového IS, pak lze označit přístup ke správným požadovaným informacím a jejich zobrazení na správném místě a ve správný čas. Jako správné místo si můžeme představit v tomto případě uživatelům dostupné webové stránky. Správný čas pak můžeme označit jako čas reálný, v dnešní době ideálně „ihned“. Z toho důvodu se může vývoj a použité technologie IS zaměřovat na přesunutí režie do uživatelské části v rámci jeho zařízení. [1]

Rozdělení

Níže uvedené rozdělení je pro čtenářovu představu o směrech, podle kterých lze získat výsledný IS.

- **Vlastní informační systém** je asi nejvíce používanou variantou. To je způsobeno zejména volností při vlastním vývoji a implementaci IS. Mezi silné stránky takového systému lze uvést především jeho snazší správu a kontrolu nad ním. Slabou stránkou je pak potřebný čas a značné vynaložené úsilí pro jeho vznik. Ve výsledku jsou pak potřebné finanční náklady a termín provozuschopnosti závislé pouze na našich možnostech a dovednostech.
- **Informační systém na míru** vychází „de facto“ z vlastního IS. Rozdíl však spočívá v přenechání vývoje IS někomu jinému. To s sebou přináší značné výhody i nevýhody. Kontrola nad vývojem je omezena na domluvě se zpracovatelem. Tuto domluvu je

vhodné definovat do písemné formy, jelikož tak lze předejít nejasnostem výsledného řešení. Řešení pak i více odpovídá našim představám. Dalším sporným specifikem jsou finanční náklady. Náklady se budou pochopitelně lišit jak ve zkušenostech zpracovatele, v rozsáhlosti funkcionalit IS či požadovaném termínu předání provozuschopnosti, tak i v aktualizacích či pozdějších úpravách.

- **Hotové řešení** v rámci různých systémů pro správu obsahu či řešení open-source[1]. Hlavní výhodou je již hotový produkt, který lze rovnou použít většinou za předpokladu drobných úprav. Finanční náklady jsou individuální dle produktu, ve většině případech ale nejsou nijak vysoké, pokud se nejedná o vzácný produkt s nízkou poptávkou na trhu. Nevýhodou mohou být omezené funkcionality onoho řešení či čas strávený nad prvotními potřebnými úpravami.
- **Přizpůsobení** (kastomizace) je posledním možným směrem, který vychází z hotového řešení. Požadavky na funkcionality či úpravy mohou být různého rázu. Většinou však v tomto případě již překračují svou složitostí drobné úpravy a je také třeba těmto úpravám věnovat více času. Extrémním případem může být přeměna produktu pro jiné účely, než pro jaké byl vytvořen. Důležitou roli při úpravách hraje znalost produktu hotového řešení. Proto může být v mnoha případech přenechána opět někomu jinému. Míra nákladů se pak obdobně jako u předchozích liší dle produktu a jeho funkcionalit, zpracovateli úprav a námi požadovaném výsledku.

S ohledem na specifikum vytvářeného IS a uplatnění vybraných partií znalostí vysokoškolského studia bude IS zpracován formou vlastního informačního systému s použitím vhodné metodiky vývoje z následující oddíl 1.2.

1.2 Metody vývoje IS

Úspěšnost tvorby IS projektů by se dala charakterizovat třemi kritérii – dokončení včas do stanoveného termínu provozuschopnosti, nepřekročení hranice vyhrazených nákladů na tvorbu a disponování kladenými funkcionalitami. Snahou zvýšit tuto úspěšnost se zaobírají tradiční a agilní metodické přístupy, z nichž každý k tomu přistupuje jiným způsobem.

[1] SW s otevřeným zdrojovým kódem, který lze dle jeho licenčních podmínek použít či upravit.

Tradiční způsoby předpokládají tvorbu IS jako daný proces, jenž lze přesně definovat, popsat a dle toho opakovatelně realizovat, případně modifikovat. Naproti tomu agilní metody jsou přesvědčeny o tvorbě IS založené na zkušenostech a pozorováních, kde nemá smysl tvorbu popisovat, ale je třeba ji monitorovat a přizpůsobovat realitě. [1]

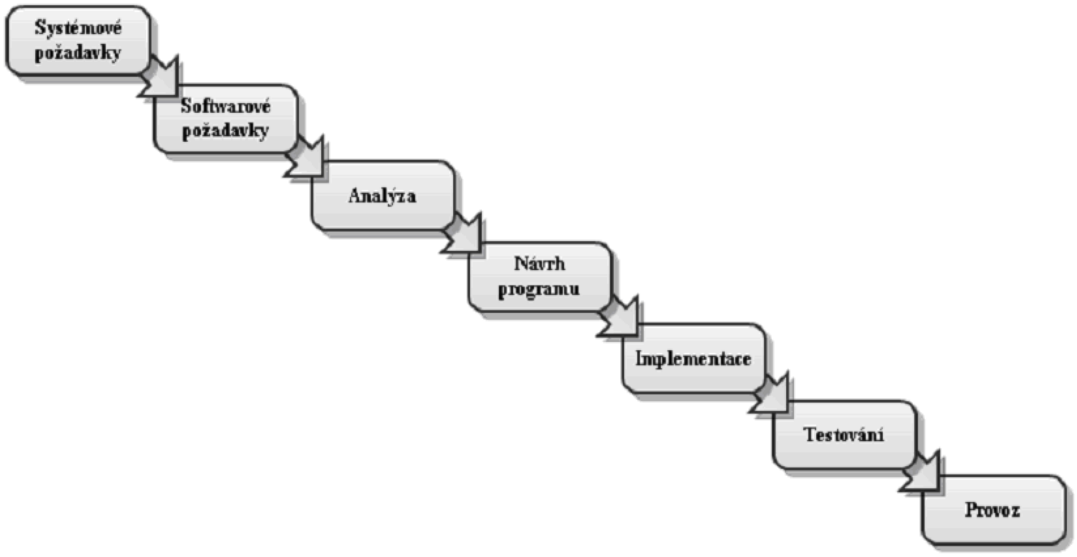
1.2.1 Tradiční přístup

Zjednodušeným způsobem lze tradiční přístup vycházející z [1] popsat jako rozdělení tvorby IS do etap. Na jednotlivé etapy lze uplatnit role zpracovatelů. Tyto role jsou specifické svou činností a pohledem na tvorbu. Životní cyklus etapy lze definovat vymezeným časovým úsekem. Pro návaznost etap by na konci každého úseku měla být práce na IS z pohledu dané etapy vývoje ukončena. Výstupem každé etapy je produkt použitý v následujících etapách. Nejvýznamnějšími modely jsou pak vodopádový, iterativní a inkrementální.

Vodopádový model

Nejstarší model byl definovaný již v roce 1970 Winstonem W. Roycem ve článku „Managing the Development of Large Software System“. Nejvíce byl rozšířen v 70. a 80. letech a inspiroval se postupy z průmyslu. Pojmenování vychází z grafického zpodobnění jednotlivých fází v posloupnost protékání vody vodopádem viz obrázek 1. [1], [2], [4]

Proces začíná fází, kdy se specifikují systémové, případně i softwarové požadavky. Následuje fáze analýzy těchto požadavků, od které se očekává odhalení případných problémů či rizik spojených s IS. Na základě analýzy se navrhne možné řešení a to se následně implementuje. Jakmile je implementace ukončena, je produkt vystaven testování dle funkčnosti požadavků. Pokud je po testování vyhodnocen produkt jako provozuschopný, splňující dané funkcionality, bývá nasazen. Na konci každé jednotlivé fáze lze vyhodnotit neúspěch a dle zjištěného důvodu traverzovat zpět na počátek dané fáze či do některé z fází předchozích. [1], [2], [3], [4]

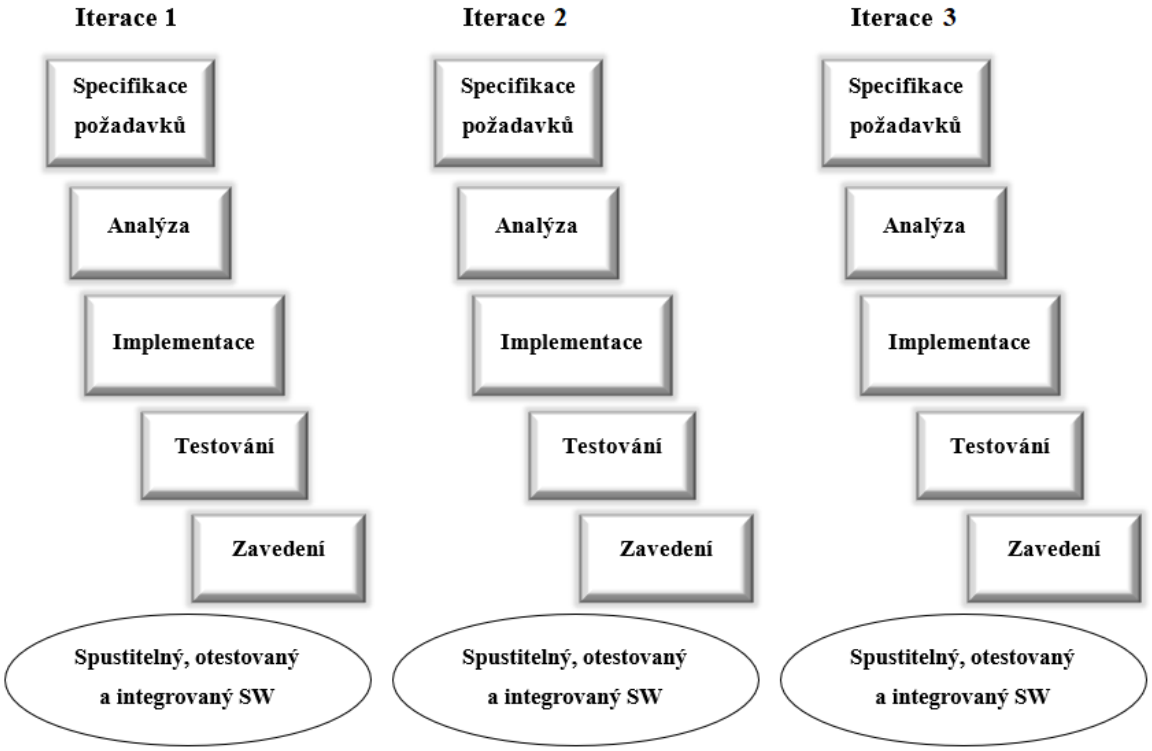


Obrázek 1: Vodopádový model životního cyklu vývoje [2]

Nejčastější důvod pro použití tohoto modelu je v případech, kdy se jedná o malý IS, u kterého se nepředpokládají změny požadavků a lze je všechny jednoznačně specifikovat v první fázi vývoje. V tom spočívá i jeho nevýhoda. Pokud není možné specifikovat požadavky na IS při počátku vývoje, bude docházet během vývoje k častým změnám a problémům v rámci konzistence mezi jednotlivými fázemi. Nevýhodou je také skutečnost, že je zákazník zapojen do vývoje zejména na začátku a na konci procesu vývoje. Nemá tak kontrolu nad průběhem a výsledným řešením. Jako další nevýhodu lze označit umístění fáze testování až mezi poslední fáze, kdy se mohou odhalit některé problémy v rámci požadavků a analýzy. V dnešní době se proto tento model v praxi tolik, alespoň ne bez určitých úprav. Slouží však jako vhodný výukový demonstrační model pro životní cyklus softwaru. [1], [3], [4]

Iterativní model

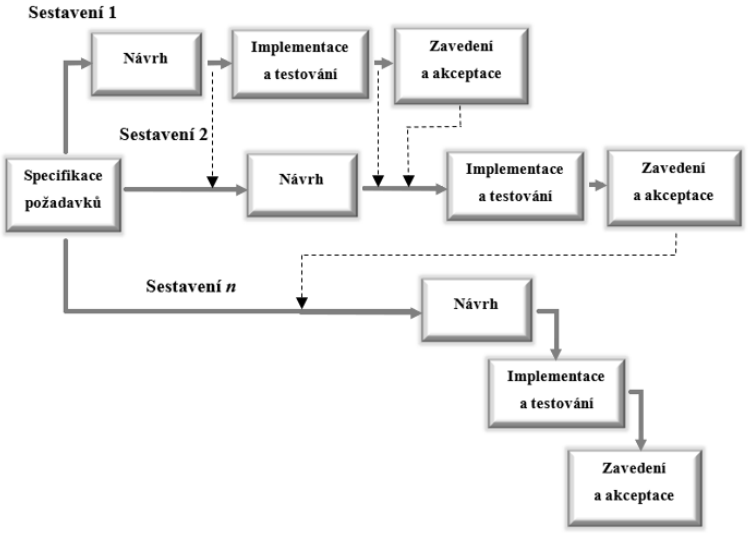
Značné nevýhody vodopádového modelu odstraňuje model iterativní. Ten je i dnes považován za moderní způsob vývoje. Již z názvu je patrné, že vývoj bude rozdělen do iterací. Myšlenkou modelu je rozdělit vyvíjený IS do menších částí, které bude možné vyvinout snadněji. Každá iterace však opět zahrnuje fáze vývoje z vodopádového modelu: plánování, analýza, návrh, implementace, testování a zavedení (obrázek 2). Výsledkem každé iterace musí být fungující část IS společně s výsledky předchozích iterací. [1]



Obrázek 2: Iterativní vývoj [1]

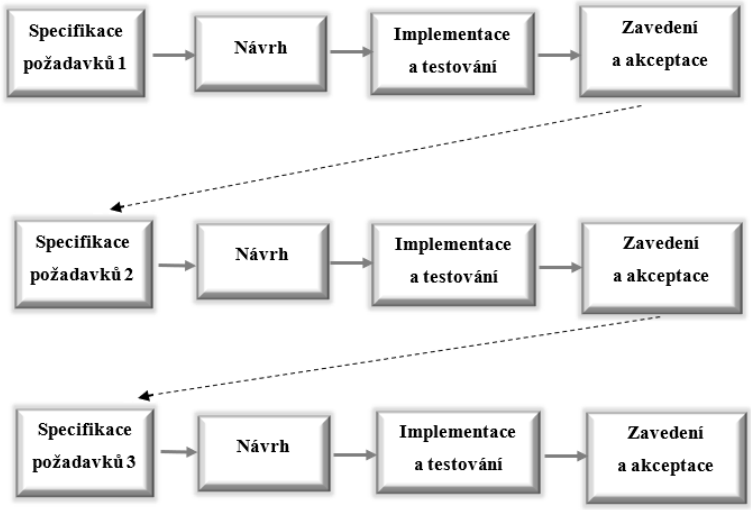
Iterativní vývoj je pak schopen zejména odhalit problémy (na rozdíl od vodopádového) ve schopnostech týmu naprogramovat funkcionalitu a také ji integrovat. Práce v týmu je pak vyřešena hned v první iteraci. Výstupem každé iterace by mělo být vyhodnocení práce týmu a uvést případné jednotlivé podněty k efektivnější práci a výsledkům v dalších iteracích. Také je zajištěna častější zpětná vazba se zákazníkem v rámci každé iterace, což vede ke zlepšení vývoje na základě spokojenosti zákazníka. Dále je pak vyřešen i problém změn požadavků během vývoje. Celkově lze pak rozlišit iterativní vývoj do dvou modelů: inkrementální a evoluční. [1]

Při vývoji inkrementálním modelem hrubě definujeme požadavky na systém. Poté jej rozdělíme do několika realizovatelných částí (přírůstků), kde každý přírůstek prochází všemi fázemi samostatně. Hlavní výhodou je rovnoměrné zavádění částí plánovaného systému. V průběhu je tak snazší sledovat, zda se vývoj ubírá požadovaným směrem a zpětnou vazbu lze zkoumat po realizaci každého přírůstku. Klíčové je pak správné vymezení přírůstků, nejen co do velikosti, ale i samostatnosti a bez nutné závislosti na doposud nerealizovaných částech systému. Znázornění je vidět na obrázku 3, kde přerušované čáry představují synchronizace mezi verzemi. [1]



Obrázek 3: Inkrementální model [1]

Evoluční model se od inkrementálního liší definováním požadavků. Všechny požadavky nejsou předem definovány, ale definují se postupně při každé iteraci (viz obrázek 4). [1]



Obrázek 4: Evoluční model [1]

1.2.2 Agilní metody

Jak bylo zmíněno v oddíle 1.2, agilní metody berou vývojový proces jako specifický a nelze jej předem dostatečně popsat. Z tohoto důvodu tyto metody nepopisují proces vývoje, ale definují principy a praktiky pro vývoj. I když je každá z metod svým způsobem specifická, všechny jsou postaveny na stejných hodnotách, které byly v roce 2001 definovány na Manifestu agilního vývoje. Hodnoty jsou popsány v následujících dvanácti bodech: [1], [3]

- Nejvyšší prioritou je kontinuálně a včas dodávat zákazníkovi SW, který mu přináší přidanou hodnotu.
- Změny požadavků jsou vítány i v pozdějších fázích vývoje. Poskytují zákazníkovi konkurenční výhodu.
- Fungující SW je třeba dodávat často a ideálně v krátkých intervalech.
- Denně by měli vývojáři a lidé z businessu (případně businessoví analytici) pracovat společně na projektu.
- Klíčovým faktorem úspěchu projektu jsou motivovaní pracovníci s vhodně vytvořenými podmínkami pro práci a s podporou jejich vedení.
- Osobní komunikace je nejefektivnějším způsobem pro přenos informací v týmu.
- Základním kamenem úspěchu je dobře fungující SW.
- Agilní procesy podporují udržitelný vývoj. Sponzoři, vývojáři a zákazníci by měli dodržovat pevnou pracovní dobu.
- Neustále je třeba se zaměřovat na dokonalé technické řešení a návrh posilující agilitu.
- Základní hodnotou je dělat věci jednoduché, což lze brát jako schopnost minimalizovat objem výsledného produktu (menší míra chybovosti).
- Nejlepší architektury, požadavky a návrhy vznikají od týmů, které jsou schopny se samy organizovat.
- V pravidelných intervalech se tým zabývá efektivitou své práce a upravuje na základě toho svou činnost.

Agilní metody jsou tedy obecně používané pro inkrementální přístup nejen k vývoji softwaru, ale také způsobu jeho dodávání. Nejvhodnější je jejich využití pro vývoj, kde se rapidně mění požadavky na SW během samotného vývojového procesu. Souvisí s tím i časté dodávky zákazníkovi, který může vznést nové požadavky do další iterace. Cílem je také omezit procesní byrokracii, která má pochybnou dlouhodobou hodnotu, a odstranit dokumentaci, která pravděpodobně nebude vůbec použita. [1], [5]

Extrémní programování (XP)

Pravděpodobně nejznámější a nejvíce používanou agilní metodou je extrémní programování. Vymyslel a publikoval ji v roce 1999 Kent Beck v knize s názvem „Extreme Programming Explained“. Vychází z prosazování a rozvinutí osvědčených postupů, jakými je například

iterativní vývoj na „extrémní“ úrovni. Podle XP lze například vyvinout, integrovat a otestovat v jeden den několik verzí softwaru. [3], [5]

Obrázek 5: Cyklus vydání SW při použití extrémního programování [5]

V XP jsou požadavky vyjádřeny jako scénáře (uživatelské příběhy), které lze implementovat jako sérii úkolů. Vývojáři pracují v párech a vytváří testy pro každý úkol před psaním samotného kódu. Všechny testy musejí úspěšně projít před integrováním nového kódu do systému. Obrázek 5 popisuje inkrementální procesy, které jsou vyvíjeny. Použití XP zahrnuje řadu principů, které reflektují hodnoty agilních metod [5], [6]:

- Inkrementální vývoj je podpořen řadou malých, frekventovaných vydání SW. Požadavky jsou založeny na jednoduchých scénářích, které jsou použity jako základ pro rozhodování, co bude zahrnuto v dalším vydání.
- Zákazník je nepřetržitě zapojován do vývoje ve spolupráci s vývojářským týmem. Zákazník se účastní vývoje a je zodpovědný za definování akceptačních testů IS.
- Lidé, nikoliv procesy, jsou součástí párového programování a podílí se kolektivně na udržitelném kódu. Udržitelný rozvoj nezabírá většinu pracovní doby.
- Změna je přijímána prostřednictvím pravidelných vydání SW zákazníkovi, vývojem řízeným testy, refaktorováním kódu, které předchází jeho degeneraci, a nepřetržité integraci nových funkcionalit.
- Zachování jednoduchosti je založeno na refaktorování kódu, které zlepšuje kvalitu, a používání jednoduchých návrhů, které se zbytečně nesnaží nepředvídat budoucí změny v IS.

V praxi se většina firem, ve kterých se používá XP, se neřídí všemi jeho principy a většinou je volí dle způsobu práce. Například párové programování se může nahrazovat individuálním programováním a posuzováním kódu (code review). Napříč různými zkušenostmi

někteří programátoři nerefaktorují v částech systému, které nevyvinuli nebo jsou namísto scénářů použity konvenční požadavky. Většinou se tak u použití XP setkáme s malými vydáními, kde se upřednostňují testy a průběžná integrace před vývojem. [3], [5]

Scrum

Scrum je nejpoužívanější agilní metoda založená hlavně na řízení projektu. Vývoj probíhá v iteracích nazvaných sprint, ve kterých se dodává množina užitečných funkcionalit IS. Jeden sprint trvá obvykle 2 až 4 týdny. Pravidla pro něj jsou jednoduchá: členové týmu se jednotlivě přihlásí k úkolům, každý pracuje na splnění cíle sprintu a každý se také účastní denních schůzek. [1], [3], [6]

Cílem denních schůzek je monitorovat stav projektu a identifikovat možné překážky či problémy, které mohou ohrozit úspěšnost projektu. Schůzka se koná ve stejnou dobu na stejném místě a trvá přibližně 15 minut. Je řízena Scrum Mastery a účastní se jí jak členové týmu, tak i manažeři. Každý člen týmu na ní zodpovídá na tři jednoduché otázky: jaké úkoly dokončil od předchozí porady, jaké má nově řešit a jaké problémy či překážky vidí v řešení úkolů. [1], [6], [7]

Scrum dále definuje role: vlastníka produktu (product owner) a vývojový tým (development team). Vlastník produktu spravuje seznam požadavků pro maximalizaci hodnoty projektu. Vývojový tým je skupina lidí s různou specializací, kteří se sami řídí tak, aby v rámci každého sprintu dodali fungující SW. Scrum Master zodpovídá za řízení a správné uplatnění metody k co největšímu užitku. V rámci metody jsou také definována pravidla pro schůzku zaměřenou na plánování sprintu, schůzku na konci každého sprintu pro předvedení hotového přírůstku a schůzka týmu pro zhodnocení práce během sprintu. [1], [7]

Efektivním použitím agilních metod je účast zákazníka. Ideálně by měl být součástí týmu či k dispozici dle potřeby týmu. Dalším klíčovým požadavkem pro agilní metody je velikost týmu. Tým by měl mít ideálně 8 lidí a pracovat v jedné místnosti z důvodu nutnosti osobní komunikace. Třetím uvedeným požadavkem dle [1] jsou znalosti, zkušenosti a morální hodnoty vývojářů.

Při rostoucím používání agilních metod však dochází také k jejich úpravám, kdy je snahou je aplikovat na distribuované týmy či projekty větší důležitosti. Za tím účelem nejsou však jen upravovány stávající, ale jsou také doplňovány nové agilní metody. Dvě výše uvedené

techniky jsou sice nejpoužívanější, ale můžeme se setkat samozřejmě s řadou dalších, například vývoj řízený testy, adaptivní vývoj softwaru, vývoj řízený vlastnostmi, lean development či metody crystal. [1], [3], [6]

1.3 Použité technologie

Cílem oddílu 1.3 je popsat technologie, knihovny a architektury použité v implementační části. Následující část oddílu se zaměřuje na použité technologie v rámci základní softwarové části. Další část je věnována javascriptové knihovně jQuery, frameworku Nette pro PHP a responsivnímu frameworku Boostrap.

Poslední část je věnována návrhovým architekturám Model, View, Controller (MVC) a webovým službám Simple Object Access Protocol (SOAP) a Representational State Transfer (REST). Oba protokoly jsou také použity pro webové služby IS studijní agendy (dále označováno jako IS/STAG).

LAMP

Linux, Apache, MySQL a PHP jsou často používaná technologie či spíše sada SW, která umožňuje po vzájemné kombinaci vytvořit stabilní systém. Využívá se zejména pro bezplatné licencování a také díky open-source. Propaguje se již od roku 1997. Technologie od té doby však prošla uplynulými roky mnohými změnami. Ať v rámci vývoje používáním různých nadstaveb formou frameworků nad PHP, tak koupí práv na MySQL, vlastněných Sun Microsystems, Oraclem v roce 2008. Základní princip a konkurenceschopnost vůči Javě a .NET však zůstává stále stejná. [8], [9] Jednotlivé části technologie jsou popsány níže.

Linux

Pod prvním pojmem technologie LAMP je operační systém Linux. Je to především stabilita, rychlost, minimální hardwarové požadavky, cena a bezpečnost, co činí operační systém Linux velkým konkurentem Windows v rámci webhostingu. Můžeme se setkat s mnoha vhodnými distribucemi Linuxu při využití pro LAMP (například Ubuntu, Fedora, Debian a další). Výsledný produkt této práce je provozován v Dockeru na distribuci Debian. [8]

Apache

Apache je webový server schopný rychlé spolupráce s operačním systémem Linux. Jeden webový server může provozovat více webových domén díky technologii virtuálních hostitelů. Mimo jiné také nabízí logování, omezování šířky pásma, ochranu přístupu k adresářům

a podporu různých vestavěných modulů. Popularita využití Apache je dle Netcraft stále vysoká, i když zaznamenává klesající trend vůči NGINX. [8], [10]

MySQL

Databázový správce pro práci s databází. Pomocí něj lze provádět základní operace nad daty formou dotazovacího jazyka SQL, a to i s využitím programovacích jazyků, zejména PHP, Java, ale i jiné. Podobně jako Oracle SQL a SQL Server, patří mezi funkcionality MySQL uzamykání tabulek, uživatelské účty, uložené procedury, triggery, pohledy či transakce.

Na rozdíl od používaných verzí 5.5 až 5.7 se lze již také setkávat s nástupcem MariaDB, což je otevřený projekt původních vývojářů MySQL vzniklý po zmíněném odkoupení Sun Microsystems, a to ve verzi 10.x. Ve verzi 10 již není zaručená zpětná kompatibilita s používanými verzemi MySQL od Oracle a obsahuje nové funkcionality. Tato práce bude využívat MySQL ve verzi 5.7, která je součástí připraveného serveru. [8], [11]

PHP

PHP (PHP: Hypertext Preprocessor) je skriptovací jazyk na straně serveru. Obecně je využitelný zejména pro svou nezávislost na platformě a velkou rozšířenost pro vývoj IS na straně serveru. Základem jazyka byl roku 1995 interpretovaný programovací jazyk Perl. Původně jsme se mohli setkat s názvem PHP/FI[2], který měl více odrážet jeho využití v té době. Mimo verze 3, která byla kompletním přepsáním předchozí verze, je asi nejvíce zlomová verze 5, ve které je již implementováno i objektově orientované programování. [8]

Aktuálně se doporučuje využívat verzi 7.x, která přináší nejen větší rychlost, lepší podporu 64bitového operačního systému, ale také mnoho dalších funkcionalit oproti starší, často používané verzi 5.6. Pro starší verze (< 5.6) již nejsou vydávány bezpečnostní aktualizace. [12]

1.3.1 Knihovna a frameworky

Moderním trendem programování je využití různých frameworků a knihoven. Setkat se s knihovnou však lze při programování již od nepaměti – například při programování složitějších matematických operací nebo při práci se souborem či databází. Ve většině jazyků

[2] Personal Home Page / Forms Interpreter – interpret pro osobní stránky a formuláře.

budeme proto zahrnovat do našeho kódu knihovnu, kterou již pro tyto často používané operace někdo vytvořil.

Technologie však podléhají neustálému vývoji. Proto se můžeme již častěji setkat s připravenými rozsáhlejšími knihovnami, které mají usnadnit náš vývoj v daném programovém jazyce (např. jQuery). Na druhou stranu se můžeme setkat i s tak rozsáhlými knihovnami, které abstrahují nad jazykem samotným (Nette, Bootstrap). O takové knihovně mluvíme jako o frameworku. V některých případech při jejich použití (Nette) dochází až k upozadění samotného jazyka – je využit pro jiné účely než byl původně stvořen. Hlavní rozdíl je ale také v tom, že zatímco knihovna je využívána při psaní v daném programovém jazyce, ve frameworku přímo píšeme.

jQuery

jQuery je knihovna JavaScriptu. Předností této knihovny je rychlost, sortiment nabízených funkcionalit a kompatibilita s webovými prohlížeči. Některé prohlížeče totiž implementují interpret JavaScriptu nekorektně (odlišné od normy), což jQuery umí ošetřit/obejít. Umožňuje webovým stránkám různé manipulace s obsahem, animace a zachycení událostí, vše jednotně pro všechny prohlížeče bez nutnosti ošetřovat chybné implementace. Tvorba pomocí technologie Ajax[3] je díky dostupnému API snadná napříč mnohými webovými prohlížeči. Při vývoji velmi usnadňuje programování v JavaScriptu, což se však negativně může odrážet na složitějších řešeních. Nejnovější dostupnou verzí jQuery je verze 3.3.1. [13]

Nette

Nette (Framework), je framework PHP cílený pro psaní různých webových řešení (od CMS po blogy). Přidaná hodnota frameworku je menší, čitelnější a tím i přehlednější kód. Obsahuje také různé funkcionality již vhodně připravené pro potřeby tvorby webových aplikací (ladicí nástroje, propojení s databází, tvorba formulářů a jiné). Jeho popularita, rozšíření a komunita jsou především v České republice. Aktuální verzí Nette je 2.4. [14]

[3] Označení pro tvorbu interaktivních webových aplikací.

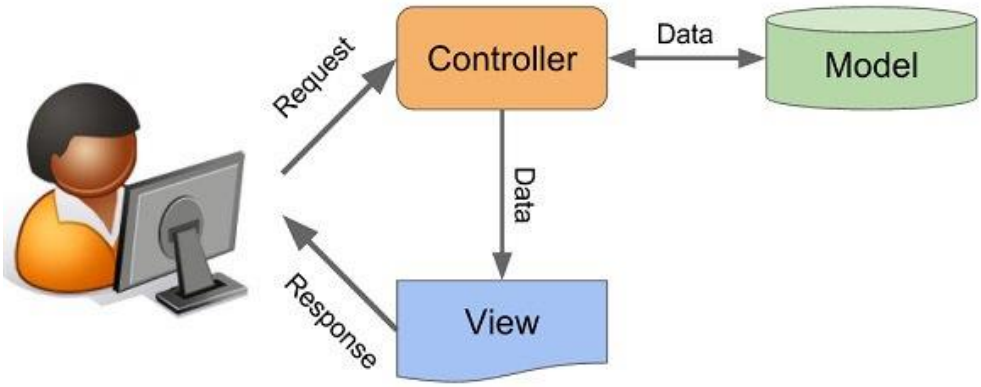
Bootstrap

Bootstrap je responsivní framework. Pracuje především s technologiemi pro vykreslení stránky (HTML, CSS a JavaScript). Použitím a dodržením konvencí Bootstrapu při vývoji webové aplikace je možné ji zobrazit korektně na jakémkoliv klientském zařízení (mobil, tablet). Usnadňuje vytváření vzhledu a obsahuje různé předem připravené šablony. Nabízí také již hotové prvky jako například způsob a zobrazení validace u formulářů. Momentálně je možné setkat se s verzí 4.0. [15]

1.3.2 MVC

Model, View a Controller je označení pro softwarovou architekturu, jejímž cílem je rozdělit aplikaci do zmíněných tří vrstev popsaných níže. Cílem architektury je nejen učinit kód přehlednějším, ale také usnadnit budoucí možný vývoj jednotlivých vrstev. Ve výsledku lze testovat a vyvíjet jednotlivé vrstvy zvlášť, ovšem s ohledem na kooperaci mezi nimi. [14]

Při vývoji se také můžeme často setkat s rozdělením na dvě části: front-end a back-end, které odrážejí princip založený na MVC. Rozdělení pak spíše štěpí vývoj či vývojářský tým do dvou větví, kde každá větev má svou zodpovědnost za přidělenou část a příslušné technologie. Opět je žádoucí, aby docházelo k nějaké kooperaci, v tomto případě při vývoji v rámci týmu.



Obrázek 6: Znázornění MVC architektury [16]

Model

Pokud si představíme MVC jako jablko, vrstva model by spíše představovala skryté jadérko jablka. Jedná se o vrstvu, která uživateli není přímo dostupná. Cílem této vrstvy je spravovat bázi dat a provádět operace nad ní. Model nemá povědomí o ostatních vrstvách. Poskytuje

však rozhraní pro bázi dat, díky čemuž je možné s ní pracovat a přistupovat k datům i dále. [14], [16]

View

Zůstaneme-li u metafory s jablkem. Vrstva view představuje samotný vzhled, jak jablko vypadá zvenčí – barvu, tvar a stav slupky. Je to tedy to, co se uživateli zobrazuje a umožňuje mu interagovat s aplikací. Lze ji také popsat jako způsob vykreslení dat na výstup uživateli. [14], [16]

Controller

Poslední vrstva zpracovává požadavky uživatele. Dle typu požadavku zpravidla pracuje s daty skrze model nebo nechá vykreslit požadovanou či výslednou stránku vrstvou view. [14], [16]

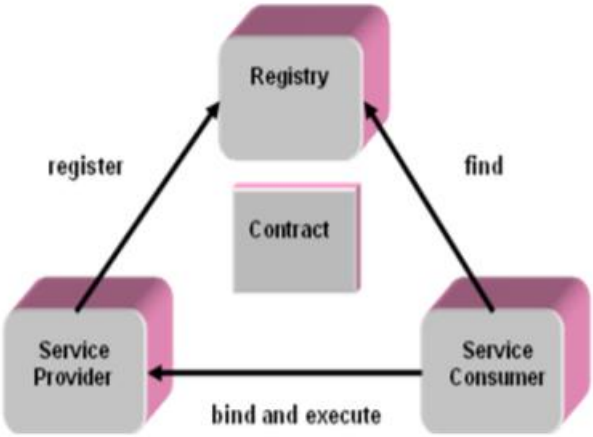
Při použití Nette se lze setkat s presenterem, obdobou controlleru. V případě presenteru se jedná o objekt, který zpracuje uživatelův požadavek a forma odpovědi může mít různé podoby – od stránek, přesměrování po objekty či soubory. [14]

1.3.3 Webové služby

Webovými službami můžeme označit standard pro způsob zprostředkování služeb po internetu napříč různými technologiemi a platformami. S webovou službou je možné komunikovat pomocí dvou principů: SOAP a REST, které budou popsány níže. [17]

SOAP

Základem při použití principu SOAP je soubor Extensible Markup Language (XML), který slouží ke komunikaci. Cílová strana serveru popisuje dostupné webové služby pomocí souboru Web Services Description Language (WSDL), díky čemuž je dána forma zasílaného a obdrženého souboru. Pro nalezení služby se dále využívá registru Universal Description, Delivery and Integration (UDDI). Registr UDDI je schopen díky popisu ve WSDL naleznout dostupné webové služby.



Obrázek 7: Architektura webové služby SOAP [17]

SOAP tak můžeme definovat jako tři entity (viz obrázek 7): poskytovatele webové služby, uživatele či konzumenta webové služby a registr. Poskytovatel přijímá a zpracovává požadavky od konzumenta. Konzumentem je myšlena jiná služba, aplikace či SW. Registr je síť dostupných služeb.

Počátkem SOAP je událost, kdy konzument vyhledá dostupnou službu dle popisu v registru, kam ji zveřejnil poskytovatel služby. Komunikace těchto entit probíhá formou zpráv SOAP pomocí protokolu SOAP. Zpráva SOAP se skládá z obálky, hlavičky a těla. Obálku identifikujeme jako zmíněný XML soubor. Hlavička obsahuje informace o volání definovaných metod a odpovědi. Tělo je pak samotný obsah souboru XML. Zprávy a volání metod jsou poté též definovány jako soubory XML a posílány skrze protokoly Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) či Hypertext Transfer Protocol (HTTP) ke konzumentovi. [17]

REST

Druhým principem pro používání webových služeb je REST. REST obdobně jako SOAP odráží zasílání požadavků. V případě REST se však jedná čistě o komunikaci mezi serverem a klientem. Server tedy rovnou na základě požadavků generuje odpovědi. Aby však mohlo dojít k požadavku, je třeba jej přesunout do správného zdroje. K tomu se využívá Uniform

Resource Identifier[4] (URI). Na rozdíl od SOAP, REST nevyžaduje daný formát zpráv v rámci hlaviček a obálek. Z toho vychází, že není potřeba ani parsování XML, což vede k menší režii. [17]

Principem je adresovatelnost, bezestavovost a jednotné rozhraní dostupné pro URI. Adresovatelnost souvisí s modelováním množiny dat určené pro práci jako zdroj, který je označen pomocí URI. Jednotné rozhraní je použito pro přístup prostředků skrze metody HTTP. [17]

Jelikož je každá transakce nezávislá a nesouvisející s předchozími, data zpracovaná požadavkem jsou obsažena pouze v odpovědi na tento požadavek. Data ze sezení (session) uživatele nadále nezatěžují server. [17]

Aplikace využívající principu REST nazýváme RESTful. Služby RESTful využívají pro získání, vytvoření, změnu a smazání prostředků HTTP-metod – GET, POST, PUT a DELETE. [17]

V případě porovnání SOAP a REST je pak REST jasnou volbou. Způsobuje znatelně menší kapacitní zátěž zpráv bez nutnosti zpracovávat data ve formátu XML. Služby RESTful je snadnější popsat a využívají čistě protokolu HTTP bez potřeby registru. [17]

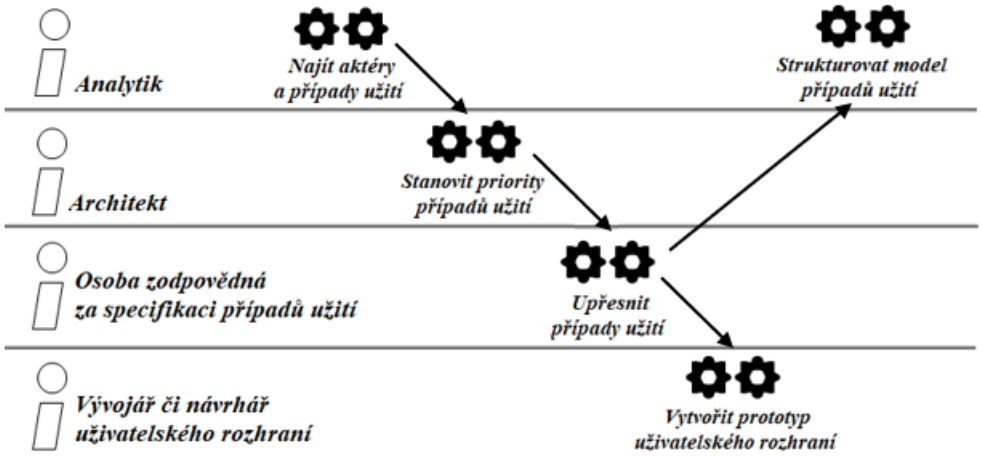
[4] URI identifikuje dokument či zdroj na internetu. Obecně se jedná o pojmenování souboru či zdroje.

2 Analýza

Další kapitoly se dle tradičního způsobu již zaměřují na tvorbu IS. V druhé kapitole se jedná o fáze, kdy se specifikují požadavky na systém a se následně provede analýza systému. Smyslem druhé kapitoly je zmapovat především funkční požadavky, aby na ně mohl být kladen důraz při fázi implementace systému. Pro zpracování analýzy byl použit nástroj Enterprise Architect (EA) verze 12.

2.1 Specifikace požadavků

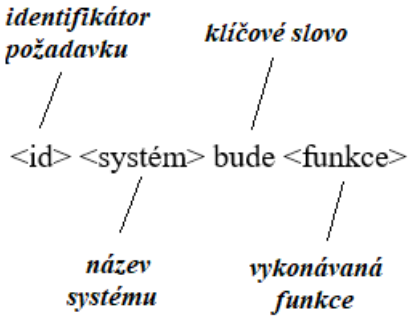
Specifikací požadavků dochází k úplnému začátku procesu tvorby. Pracovní postup při postupu a definici požadavků dle metody Unified Process[5] (UP) popisuje následující obrázek 8. Důraz je kladen na nalezení aktérů, případů užití a stanovení jejich priorit. Při tomto postupu se ale neřeší nefunkční aspekty požadavků a je proto nutné zařadit též vyhledání a rozlišení funkčních a nefunkčních požadavků. [18]



Obrázek 8: Postup při definici a specifikaci požadavků dle UP [18]

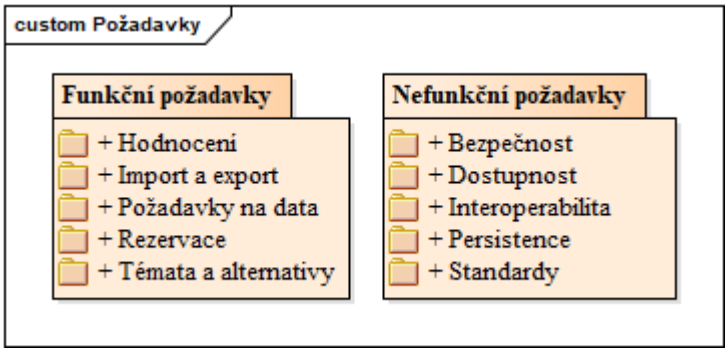
Požadavek by měl být napsán tak, aby z něj bylo srozumitelně jasné, co se bude realizovat. Využíváme proto jednoduchého tvaru, kterým je každý požadavek popsán (viz obrázek 9). [19]

[5] Proces vývoje SW, který při vývoji definuje kdo, co, kdy a jak realizuje.



Obrázek 9: Popis požadavku [18]

Při stanovení požadavků, ať už funkčních či nefunkčních, je vhodné dané požadavky mimo základního dělení rozčlenit také do kategorií. Přispějeme tím k detailnější klasifikaci. Nejčastěji se můžeme s kategorizací požadavků setkat v případech vysokého počtu požadavků. [18], [19]

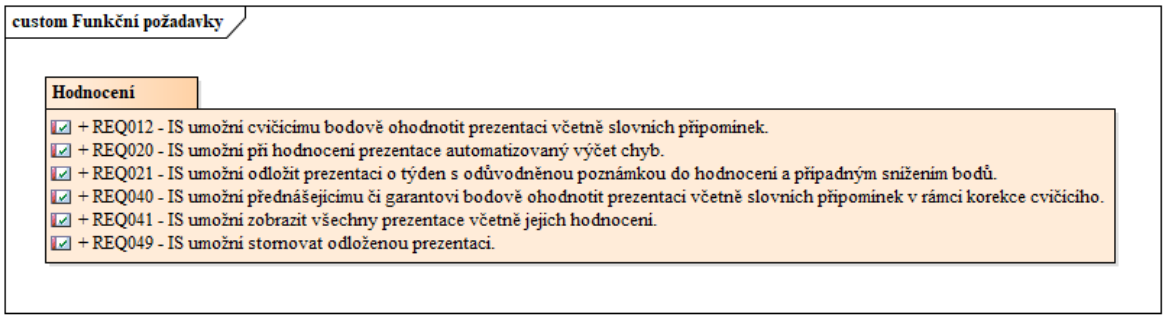


Obrázek 10: Požadavky na systém [vlastní]

Funkční požadavky

Funkční požadavky formulují to, co by měl systém dělat. Popisují tak požadované funkcionality systému k implementaci. [18], [19]

Funkční požadavky jsou rozděleny do 5 kategorií, které zastupují specifické oblasti zpracování. Jednotlivé kategorie budou popsány níže.



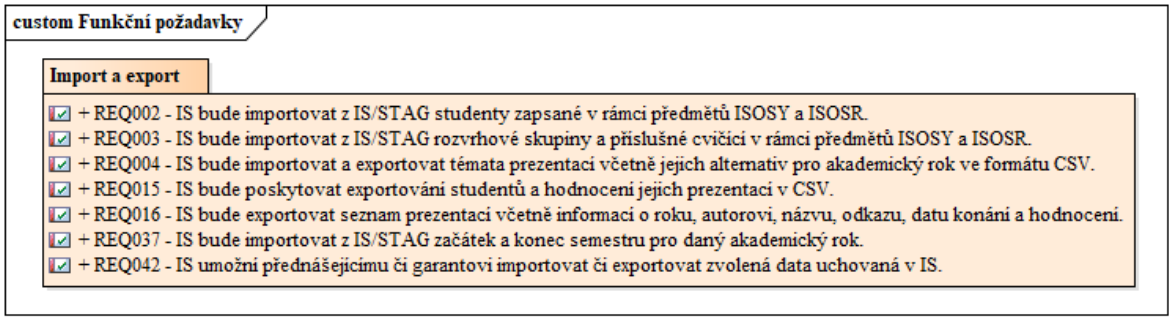
Obrázek 11: Kategorie hodnocení [vlastní]

Kategorie hodnocení (viz obrázek 11) v sobě obsahuje požadavky ohledně procesu hodnocení. Hodnocení by mělo obsahovat získané body a slovní připomínky. IS bude umožňovat zobrazit seznam všech hodnocení včetně jejich hodnocení.

V rámci možné korekce cvičícího je umožněno hodnocení i přednášejícímu či garantovi, pokud cvičícím není zároveň i přednášející či garant.

Při hodnocení by měla být možnost automatizace při výčtu častých chyb. Především se jedná o pravopisné chyby či jiné chyby definované v pravidlech hodnocení ISOSY/ISOSR. [20]

Pro důvod odkladu prezentace o týden bude tato informace zahrnuta v hodnocením s případným snížením bodů. V případě změny jako například pozdního příchodu prezentujícího, bude možné tuto operaci stornovat a vytvořit nové hodnocení.



Obrázek 12: Kategorie import a export [vlastní]

Kategorie import a export (viz obrázek 12) specifikují požadavky pro dané operace s daty uchovanými v IS. Je to ať už z důvodu získání dat z IS/STAG nebo uchování a využití dat IS pro další administrativní operace přednášejícím či garantem.

V rámci interoperability s IS/STAG je nutné do IS získat informace o tom, kdo ze studentů je aktuálně zapsaný na předmět ISOSY/ISOSR. Definovat cvičící, pod které jednotliví

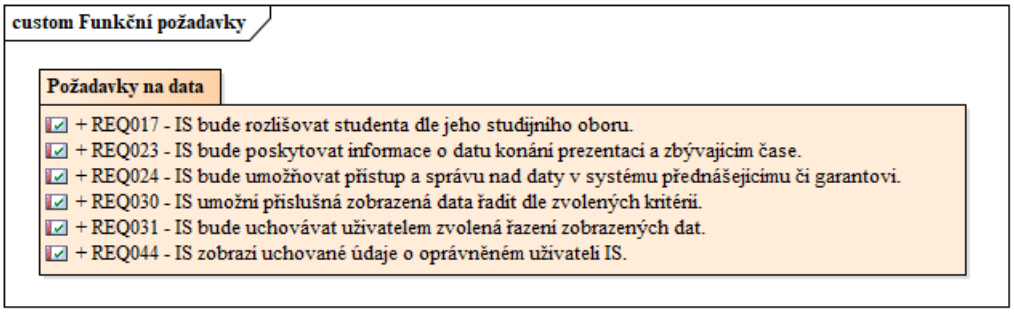
studenti spadají, a jejich rozvrhové skupiny pro další možné plánování prezentací. Dále je nutné získat informaci o tom, kdy začíná a končí jednotlivé semestry pro daný akademický rok pro rozvržení témat.

Kvůli možnému pozdějšímu zápisu a odhlášení studentů na předmět ISOSY/ISOSR bude importování zapsaných studentů prováděno zvlášť od importování rozvrhových skupin a cvičících.

Při importu rozvrhových skupin bude docházet k přiřazení orientačního čísla dané skupiny dle konání rozvrhové akce. V případě stejného dne a času více rozvrhových skupin bude číslování provedeno dle příjmení cvičícího. Jelikož jedna skupina může být společná pro studenty z více oborů, je třeba je sloučit dle konání rozvrhové akce a cvičícího.

IS bude exportovat pro administrativní potřeby přednášejícího či garanta jednotlivé informace o prezentaci jako rok, autor, název, odkaz na prezentaci, datum konání a hodnocení. Exportován pro stejné účely bude i pouhý seznam studentů s jejich hodnocením, a to ve formátu Comma-Separated Values (CSV).

IS bude importovat i exportovat témata prezentací včetně jejich alternativ pro daný akademický rok, opět ve formátu CSV.



Obrázek 13: Kategorie požadavky na data [vlastní]

Kategorie požadavky na data se vztahuje k požadavkům pro zobrazená data IS, která nespadají ani do jedné z vytvořených kategorií (viz obrázek 13).

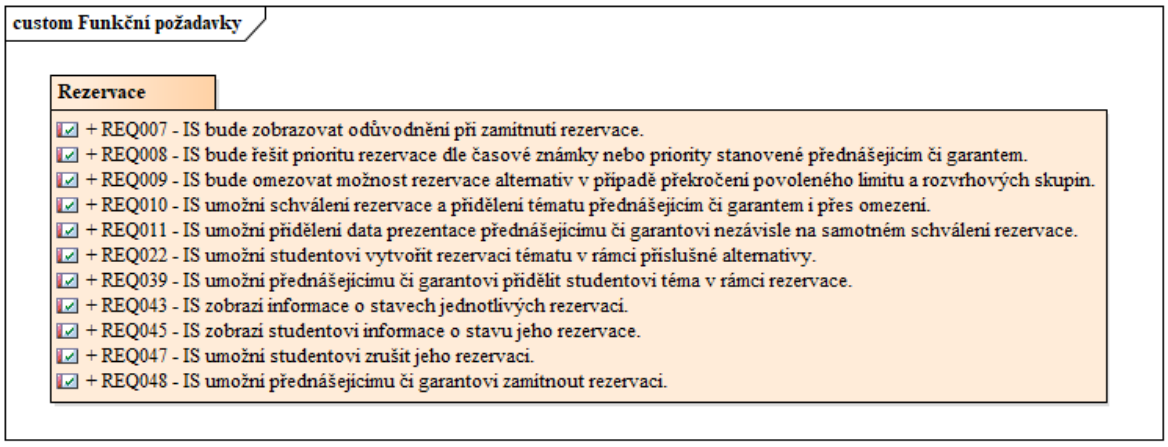
Požadavek v rámci REQ017 umožňuje IS rozlišit, do jakého studijního oboru patří daný student. S touto informací se může dále pracovat například při hodnocení, ale především pak u zobrazení témat a alternativ. Zobrazení bude popsáno v příslušné kategorii.

Z hlediska účelu IS je žádoucí zobrazení informace o datu konání jednotlivých prezentací a zbývajícím čase. Tato informace je důležitá napříč všemi rolemi IS.

Přednášející či garant by měl mít pro úspěšný chod a správu systému přístup nad všemi daty udržovanými v systému. Ať už v rámci zobrazení či úprav.

Zadavatelem je kladen důraz na možnost řazení zobrazených dat dle volených kritérií. Případně bude možné uživatelem zvolené řazení uložit.

IS umožní uživateli zobrazit o něm uchované informace získané z IS/STAG.



Obrázek 14: Kategorie rezervace [vlastní]

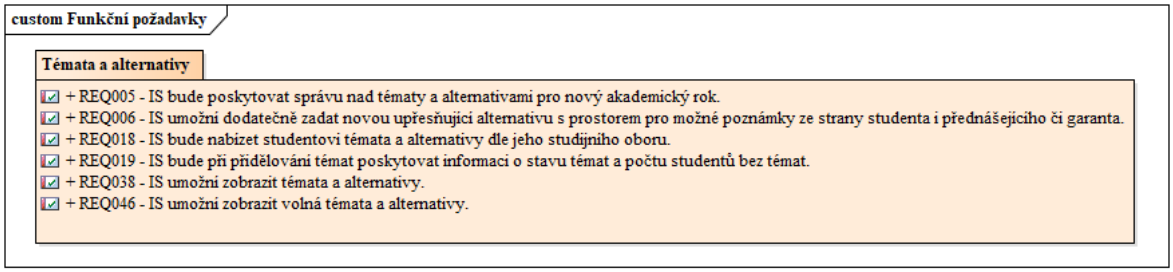
Pod kategorii rezervace spadají požadavky vztahující se na rezervace od studentů (viz obrázek 14). Základní rutinou je studentem vytvořená rezervace tématu. Vytvořenou rezervaci může přednášející či garant v rámci schválení přidělit či zamítnout.

IS bude při procesu zamítnutí rezervace vyžadovat odůvodnění. Odůvodnění se bude poté u zamítnuté rezervace informačně zobrazovat.

IS bude řešit prioritu rezervace. Priorita je dána časovou známkou nebo přednášejícím či garantem. K této prioritě bude přihlíženo v rámci schvalování rezervací, a tím i přidělení alternativy jako téma prezentace.

Rezervace daných alternativ bude omezena počtem možných rezervací. V rámci překročení stanoveného počtu, nebude možné vytvářet další rezervace. Přednášející či garant bude moci i přes omezení počtu přidělit téma prezentace. Obdobně bude moci přidělit datum prezentace nezávisle na schválení rezervace.

IS dále umožní zobrazit informace o stavech jednotlivých rezervací studentovi i přednášejícímu či garantovi. Student si může také zobrazit stav své vlastní rezervace, případně ji zrušit, pokud nebyla schválena.



Obrázek 15: Kategorie témata a alternativy [vlastní]

Pod kategorii témat a alternativ (viz obrázek 15) spadají požadavky ohledně správy témat a alternativ nebo jejich zobrazení.

Témata a alternativy lze spravovat souhrnně vždy pro nový akademický rok, kdy může dojít k přidání nových témat a alternativ na základě předchozích prezentací nebo s ohledem na nové technologie. Z podobného důvodu může docházet i k odebírání. Odebrání lze řešit snížením počtu rezervací na záporné číslo, často se používá −1.

Při rezervaci může být ze strany studenta i přednášejícího či garanta alternativa upřesněna. IS umožní zadat novou upřesňující alternativu s prostorem pro možné poznámky, jak ze strany studenta, tak i přednášejícího či garanta.

IS umožní studentovi výběr tématu a tím i příslušné alternativy pro téma prezentace. Zadavatel požaduje, aby bylo možné témata a alternativy nabízet dle studijních oborů pro možné dělení témat a alternativ.

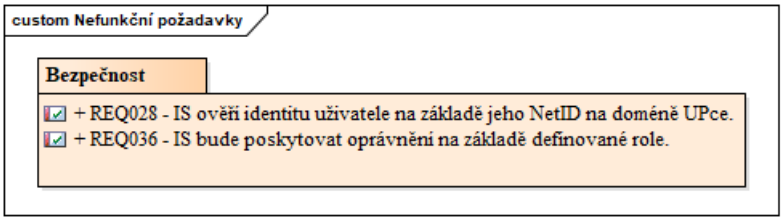
Při přidělování tématu, rezervace, bude IS zobrazovat informace o stavech témat a počtu studentů bez témat.

IS bude dále poskytovat zobrazení nejen všech témat a alternativ, ale také témata a alternativy dostupné k rezervaci. Bude se jednat o především o alternativy, jejichž počet rezervací nebyl naplněn.

Nefunkční požadavky

Nefunkční požadavky formulují omezení, která jsou uvalena na systém. Specifikují tak způsob, kterým bude systém implementován. [18]

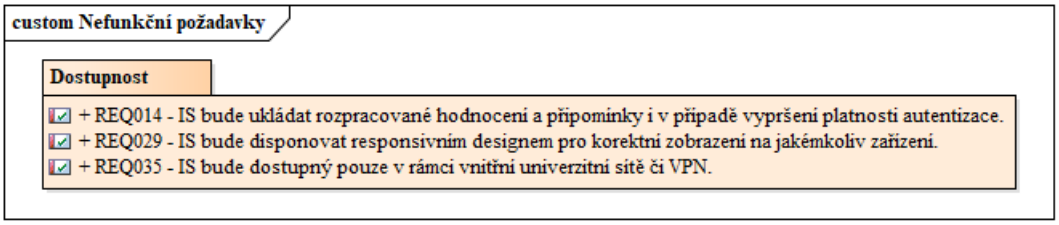
Obdobně jako u funkčních požadavků, jsou i nefunkční požadavky rozděleny do kategorií dle účelu požadavku.



Obrázek 16: Kategorie bezpečnost [vlastní]

Kategorie bezpečnost specifikuje požadavky kladené na IS (viz obrázek 16).

IS bude ověřovat identitu uživatele na základě jeho NetID v doméně UPCE. Ověřuje se tak, zda má uživatel přístup do IS, jaká role je mu v rámci IS poskytnuta a jaká oprávnění mu jsou umožněna v rámci dané role.

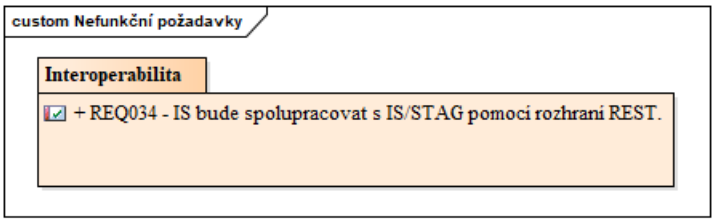


Obrázek 17: Kategorie dostupnost [vlastní]

Pod kategorii dostupnost (viz obrázek 17) spadají požadavky cílené na to, komu bude IS dostupný. S ohledem na kategorii bezpečnost je tak dostupnost vymezena pouze v rámci vnitřní univerzitní sítě či skrze Virtual Private Network (VPN) i pro připojení mimo dosah univerzitní sítě.

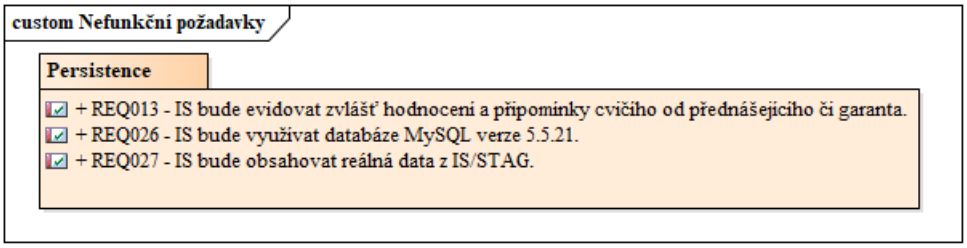
Z důvodu rozpracovaného hodnocení klade zadavatel důraz na jeho ukládání a pozdější možnosti v něm pokračovat i po vypršení platnosti autentizace.

V rámci zpracovaní je poté i kladen požadavek, aby IS disponovalo responsivním designem. Jedná se o požadavek pro možnost zobrazit IS korektně na jakémkoliv zařízení s různou úhlopříčkou.



Obrázek 18: Kategorie interoperabilita [vlastní]

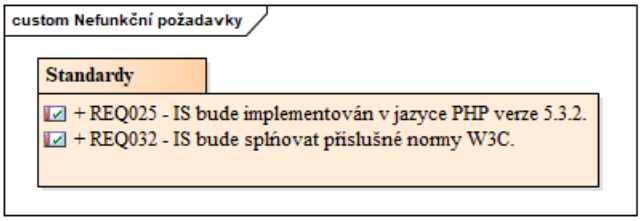
Kategorie interoperabilita (viz obrázek 18) obsahuje důležitý požadavek na IS, kdy bude možné v rámci něj spolupracovat s IS/STAG skrze rozhraní REST. Díky tomu bude možné získat potřebná data pro funkčnost IS.



Obrázek 19: Kategorie persistence [vlastní]

Kategorie persistence (viz obrázek 19) specifikuje, jaká data a jak je IS bude uchovávat. Pro možnou korekci cvičícího bude hodnocení přednášejícího či garanta evidováno odděleně.

V rámci dostupného cílového serveru se bude jednat o systém správy databáze MySQL verze 5.5.21. Databáze bude obsahovat mimo jiné i reálná data importovaná skrze rozhraní REST z IS/STAG.



Obrázek 20: Kategorie standardy [vlastní]

Kategorie standardy (viz obrázek 20) definuje, jakým standardům bude IS podléhat.

V rámci implementace se bude jednat o programovací jazyk PHP. Dle dostupného cílového serveru je jeho verze 5.3.2. Odpovídající verze Nette k dané verzi PHP je 2.3.[6]

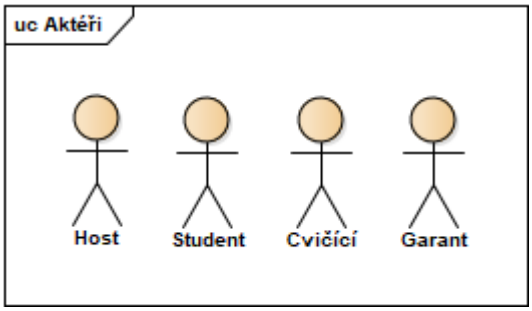
Výsledné webové rozhraní IS má splňovat příslušné normy dané standardy World Wide Web Consortium (W3C).

2.2 Modelování případů užití

Modelování případů užití je dle [18] doplňkovým způsobem pro získání a dokumentování požadavků. V rámci oddílu 2.2 bude dokumentováno nalezení aktérů IS a případů užití. Cílem oddílu je vymezit dle požadavků role uživatelům systému a popsat jejich činnosti, které mohou se systémem vykonávat.

2.2.1 Aktéři

Dle požadavků v předchozím oddíle lze definovat tři aktéry: student, cvičící a přednášející či garant. S ohledem na rutinu z předchozí systému je třetí aktér jednotně pojmenován jako garant. Zároveň je k tomu také přidán aktér, reprezentující entitu uživatele, který nespadá ani do jedné množiny z aktérů, ale má díky NetID do IS přístup. Jedná se o nepřihlášeného uživatele zvaného zkráceně host. Získané aktéry znázorňuje následující obrázek. [21]



Obrázek 21: Aktéři IS [vlastní]

Host

Jedná se o nepřihlášeného uživatele s přístupem k IS v rámci univerzitní sítě skrze NetID. Může se jednat zejména o studenty, kteří nemají zapsán předmět ISOSY/ISOSR. Případně hostem mohou být akademičtí pracovníci, již nevystupují v rámci předmětu jako cvičící,

[6] Zdrojem je Nette sandbox pro verzi 2.3 dostupný na GitHubu, viz https://goo.gl/oMKqD8.

přednášející či garant. V rámci pozdního importu se může ale jednat i o potenciální uživatele systému, kteří prozatím nebyli nahráni do systému.

Host si může dle [21] zobrazit dostupná témata a alternativy. Zároveň také pravidla pro rezervaci a podmínky pro přístup do IS.

Student

Aktér student představuje studenta zapsaného na předmět ISOSY/ISOSR.

V rámci IS je studentovi poskytnuto zobrazení dostupných témat a jejich alternativ. Dále pak zobrazení rezervací, prezentací, volných témat k prezentaci a všechna doposud prezentovaná témata z předchozích semestrů a let. Může si také zobrazit informaci o svých údajích získaných z IS/STAG. [21]

Student realizuje v rámci IS rezervaci vlastní zvolené alternativy jako téma prezentace. K rezervaci může připojit poznámku vztahující se k upřesnění alternativy. Může si také zobrazit stav své rezervace. [21]

Cvičící

Aktér představuje akademického pracovníka organizujícího libovolný počet rozvrhových akcí v rámci předmětu ISOSY/ISOSR, označeného též jako cvičící. Účel cvičícího je v rámci IS hodnotit proběhlé prezentace, popřípadě zaznamenávat její odložení či neproběhnutí.

Cvičící si může zobrazit informace o tématech a alternativách a seznamu prezentací včetně hodnocení. Na rozdíl od studenta si ale může také zobrazit souhrnné informace o blížících se prezentacích v rámci jeho cvičení. Také může hodnotit prezentace v rámci své skupiny. [21]

Garant

Aktér představuje v rámci IS vyšší roli připodobitelné k administrátorovi. Může, ale nemusí se jednat o akademického pracovníka, který organizuje rozvrhované akce v rámci předmětu ISOSY/ISOSR. Působení aktéra je z pohledu nejen IS, ale také řízení předmětu, klíčové. Proto se jedná buď přímo o garanta předmětu nebo přednášejícího.

Účel garanta je v IS dán následujícím výčtem činností [21]:

- správa témat a alternativ,

- správa vyučujících (aktéři typu cvičící a garant),
- správa rozvrhovaných skupin,
- správa učeben,
- import a export potřebných dat,
- spravovat rezervace studentů,
- udělit téma studentovi bez nutnosti rezervace,
- napravit v případě potřeby hodnocení cvičícího vlastním hodnocením,
- hodnotit prezentace v rámci svých rozvrhových akcí.

2.2.2 Případy užití

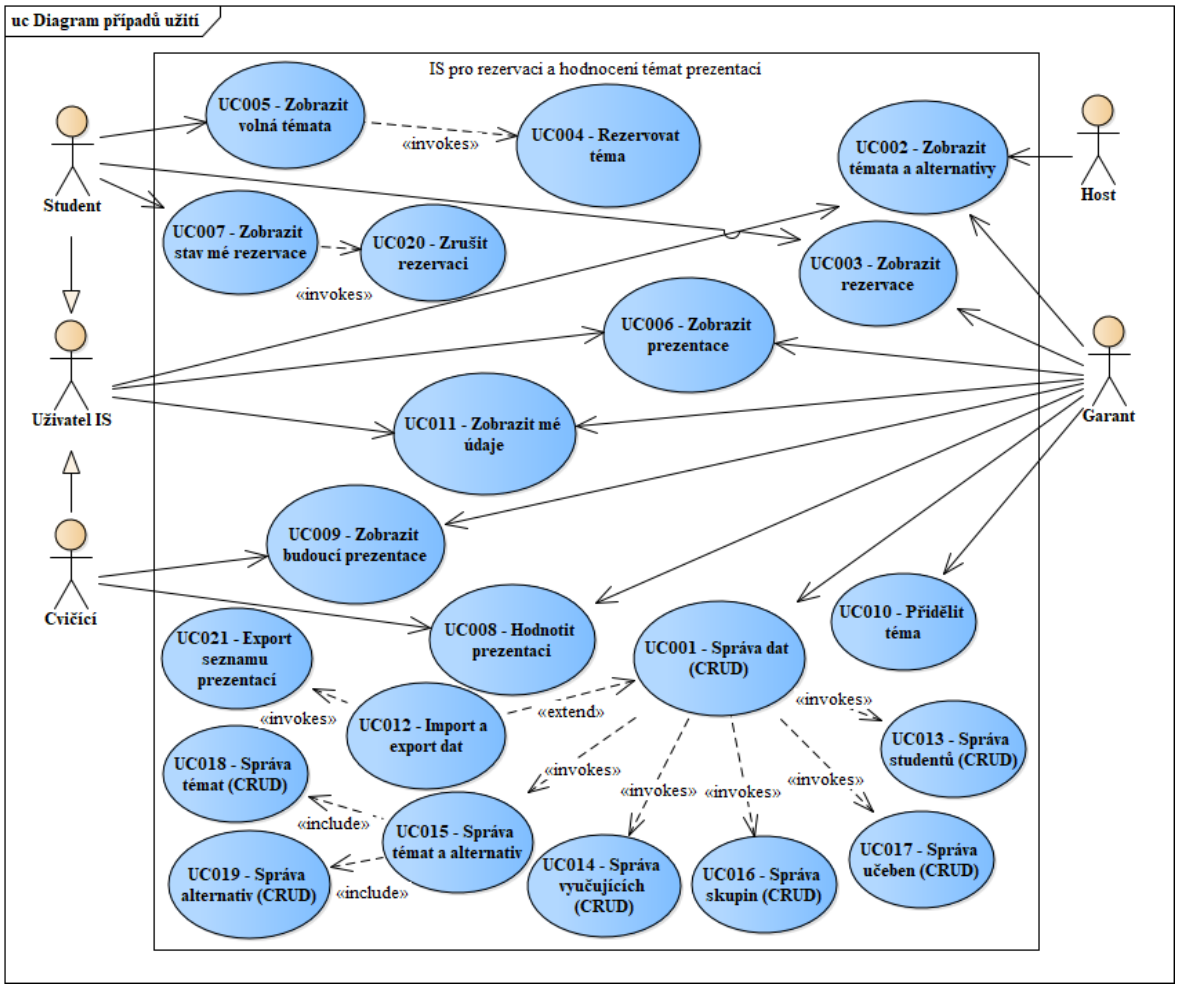
Případy užití můžeme definovat jako činnosti, které jsou prováděny v IS včetně různých chybových či proměnných posloupností. Činnosti jsou následkem interakce vnějšího aktéra s IS. [18]

Z pohledu aktéra je to činnost, kterou od systému očekává. Vhodným způsobem pro znázornění případů užití je pak položení z pohledu aktéra důležitých otázek: „Co mi bude systém poskytovat?“ a „K čemu systém budu využívat?“ Doporučuje se proto projít jednotlivé aktéry, zodpovědět si zmíněné otázky a sepsat ke každému možné případy užití. Při tomto procesu může dojít k objevení nových aktérů. [18]

Při znázornění diagramu případů použití je neméně důležité zmínit znázornění hranic IS rámečkem. Vyjadřuje se tím oddělení objektů uvnitř a mimo IS. Ze zmíněných souvislostí vyplývá, že aktéři patří do skupiny objektů patřící mimo IS a případy užití mezi objekty uvnitř IS.

Diagram případů užití

Vymodelovaný diagram obsahuje mimo případů užití všechny zmíněné aktéry z předchozího pododdílu 2.2.1. V modelovaných případech užití jsou znázorněny jak funkční požadavky na systém vycházející z oddílu 2.1, tak očekáváné činnost popsané u jednotlivých aktérů. Ve snaze zjednodušit výsledný diagram, je přidán aktér uživatel IS, který slouží jako předek k aktérům student a cvičící. Diagram zachycuje následující obrázek.

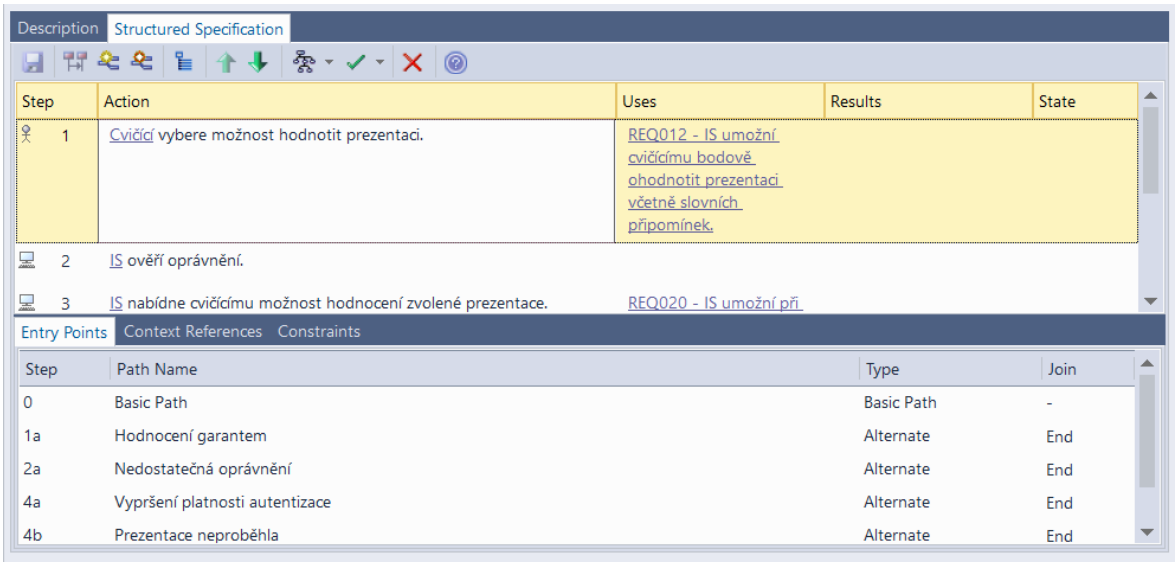


Obrázek 22: Diagram případů užití [vlastní]

Specifikace případů užití

Pro specifikaci případu užití neexistuje žádný standard. Obvykle se využívá šablony, která dle [18] obsahuje o případu užití jeho název, identifikátor, slušný popis, aktéry, vstupní podmínky jako stav systému před, hlavní scénář a jeho kroky, výstupní podmínky jako stav systému po a možné alternativní scénáře.

Při použití nástroje EA pro analýzu a návrh IS lze specifikovat případ užití v rámci strukturovaného scénáře (viz obrázek 23) a následně vygenerovat diagram aktivit samotného případu užití, včetně zobrazení alternativních kroků.



Obrázek 23: Specifikace scénáře v nástroji EA [vlastní]

V rámci obsahu diplomové práce jsou znázorněny pouze klíčové případy užití a to ve formě zmíněné šablony. Všechny diagramy jsou dostupné v rámci přílohy A, souboru s projektem EA.

Tabulka 1: Případ užití pro rezervování tématu [vlastní]

UC004 – Rezervovat téma
ID: 4
Stručný popis: Student si rezervuje téma prezentace.
Hlavní aktéři: Student
Vedlejší aktéři: Žádní.
Vstupní podmínky: 1. Student má zobrazený seznam volných témat.
Hlavní scénář: 1. Student zvolí možnost rezervovat volnou alternativu jako téma své prezentace. 2. IS ověří oprávnění.

3. IS zobrazí formulář pro možnost vyplnění poznámky. 4. Student potvrdí a odešle formulář. 5. IS zkontroluje počet rezervací. 6. IS vytvoří novou rezervaci. 7. IS zobrazí stav rezervace studenta.
Výstupní podmínky: Vytvořena nová rezervace.
Alternativní scénáře: Nedostatečná oprávnění. Omezený počet rezervací.

Prvním demonstrovaným případem užití je rezervování tématu (viz tabulka 1), který realizuje REQ022. Student si v rámci seznamu volných alternativ a jejich témat zvolí volnou alternativu. IS poté ověří jeho oprávnění a zobrazí mu formulář pro možnost vyplnit poznámky k rezervaci (REQ006 upřesňující alternativa). Student vyplní potřebné údaje, potvrdí a odešle formulář. IS zkontroluje počet rezervací a vytvoří novou rezervaci. Nakonec zobrazí studentovi stav jeho rezervace.

Alternativní scénáře jsou: nedostatečná oprávnění a omezený počet rezervací. Nedostatečná oprávnění se vztahují ke 2. kroku, kdy proces rezervování tématu skončí, pokud student není oprávněn rezervaci uskutečnit (např. jiný studijní obor). Omezený počet rezervací je kontrolován před vytvořením nové rezervace v rámci 6. kroku a v případě omezení ukončuje proces rezervace tématu. Počet rezervací se totiž může během jednoho procesu měnit v souvislosti s jinými procesy.

Tabulka 2: Případ užití pro přidělení tématu [vlastní]

UC010 – Přidělit téma
ID: 10
Stručný popis: Garant přiděluje téma studentovi v rámci rezervace.
Hlavní aktéři: Garant

Vedlejší aktéři: Žádní.
Vstupní podmínky: Žádná.
Hlavní scénář: 1. Garant vybere možnost přidělit téma prezentace. 2. IS zobrazí seznam rezervací dle uložených kritérií. 3. Garant u vybrané rezervace zvolí možnost přidělit. 4. IS zobrazí informace o stavu témat a studentů bez témat. 5. IS zobrazí formulář s možnými připomínkami k rezervaci. 6. Garant vybere možnost přidělit prezentaci a odešle formulář. 7. IS změní stav rezervace. 8. IS vytvoří novou prezentaci. 9. IS zobrazí seznam rezervací.
Výstupní podmínky: IS vytvoří novou prezentaci.
Alternativní scénáře: Nová kritéria řazení. Přidělení bez rezervace. Zamítnutí rezervace.

Případ užití pro přidělení tématu (viz tabulka 2) popisuje scénář pro činnost garanta při přidělování tématu. Na zobrazený seznam rezervací je použito, jako u předchozího případu užití, řazení dle zvolených kritérií a jejich možné uložení.

U jednotlivé rezervace jsou zobrazeny informace o studentovi a jeho skupině, tématu a alternativy (obojí s popisem), název prezentace, studentova poznámka při rezervaci, skupina pro rezervaci a předpokládaný datum prezentace. Rezervace jsou dále rozděleny do skupin témat.

Garant u zvolené rezervace zobrazí formulář pro možné přidělení tématu. Nastaví potřebné údaje, případně připojí připomínku k rezervaci a odešle formulář IS. IS poté změní stav

zvolené rezervace. Na základě rezervace a vstupních dat vytvoří novou prezentaci a zobrazí seznam rezervací.

Alternativními scénáři jsou nová kritéria řazení, přidělení bez rezervace a zamítnutí rezervace. Nová kritéria řazení představují změnu a uložení kritérií podle kterých je seznam rezervací řazen dle zmíněných požadavků. Přidělení bez rezervace umožňuje garantovi přidělit téma bez schvalovacího procesu rezervace (REQ011), případně obejít omezení stanovená v REQ010.

Posledním alternativním scénářem je zamítnutí rezervace, kde dochází ke změně rezervace. Není tak vytvořena nová prezentace.

Tabulka 3: Případ užití pro hodnocení prezentace [vlastní]

UC008 – Hodnotit prezentaci
ID: 10
Stručný popis: Cvičící hodnotí proběhnutou prezentaci.
Hlavní aktéři: Cvičící
Vedlejší aktéři: Žádní.
Vstupní podmínky: Student prezentoval přidělené téma prezentace.
Hlavní scénář: 1. IS zobrazí seznam prezentací. 2. Cvičící vybere možnost řadit dle zvolených kritérií a uloží. 3. IS uloží zvolená kritéria pro řazení. 4. Cvičící vybere možnost hodnotit prezentaci. 5. IS ověří oprávnění. 6. IS nabídne cvičícímu možnost hodnocení zvolené prezentace. 7. Cvičící zvolí počet bodů, zapíše připomínky a zvolí uložit hodnocení. 8. IS uloží hodnocení prezentace. 9. IS zobrazí seznam prezentací.

Výstupní podmínky: IS změní informace o prezentaci.
Alternativní scénáře: Hodnocení garantem. Nedostatečná oprávnění. Vypršení platnosti autentizace. Prezentace neproběhla. Stornování neproběhnuté prezentace.

Případ užití pro hodnocení prezentace (viz tabulka 3) popisuje scénář pro činnost cvičícího při hodnocení proběhnuté prezentace. Na zobrazený seznam prezentací je použito řazení dle zvolených kritérií definovaných v rámci REQ031 nebo možnost je měnit v rámci REQ030 definovaných ve funkčních požadavcích oddílu 2.1.

Cvičící u zvolené prezentace vybere možnost hodnotit prezentaci. IS ověří oprávnění cvičícího, zda může hodnotit zvolenou prezentaci. V případě souhlasu mu zobrazí možnost ohodnotit prezentaci. Po zvolení počtu bodů a zapsání připomínek cvičící zvolí možnost uložit hodnocení. IS uloží hodnocení a znovu zobrazí seznam prezentací.

Alternativními scénáři jsou: hodnocení garantem, nedostatečná oprávnění, vypršení platnosti autentizace, prezentace neproběhla a stornování neproběhnuté prezentace.

REQ040 definuje možnost hodnotit prezentaci garantem v rámci korekce cvičícího, pokud cvičícím není zároveň garant. Alternativní scénář probíhá obdobně jako hlavní scénář.

Druhý alternativní scénář vychází z hlavního scénáře hodnocení prezentace kroku 5, kdy cvičící nemá dostatečná oprávnění k hodnocení. K tomu může dojít v případě, kdy se cvičící pokouší hodnotit prezentaci, která nespadá pod jeho rozvrhovou akci.

REQ014 požaduje, aby bylo možné přistoupit k rozpracovanému hodnocení i přes vypršení platnosti autentizace. Požadavek odpovídá kroku 7, z hlavního scénáře a vztahuje se i na alternativní scénář při hodnocení garantem.

V případě, kdy nedojde k prezentování studentem, cvičící zvolí možnost odložení prezentace o týden definovanou v REQ021, přidá případné připomínky a uloží hodnocení. Stornování

pak realizuje REQ049, kde je třeba zmíněné odložení stornovat. Cvičící potvrdí stornování a je mu umožněno znovu hodnotit prezentaci.

Tabulka 4: Případ užití pro export seznamu prezentací [vlastní]

UC021 – Export seznamu prezentací
ID: 21
Stručný popis: Garant exportuje seznam prezentací do CSV soboru.
Hlavní aktéři: Garant
Vedlejší aktéři: Žádní.
Vstupní podmínky: 1. Garant má zobrazený seznam prezentací. **Hlavní scénář:** 1. Garant zvolí možnost exportovat prezentace. 2. IS exportuje prezentace včetně informací o roku, autorovi, názvu, odkazu, datu konání a hodnocení. 3. IS nabídne soubor s daty k uložení na disk. 4. Garant potvrdí uložení souboru. 5. Soubor s daty je uložen. **Výstupní podmínky:** Žádné.
Alternativní scénáře: Žádné.

Případ užití pro export seznamu prezentací (viz tabulka 4) popisuje scénář činnosti garanta při exportu dat o prezentacích do CSV souboru. Garant při zobrazení seznamu prezentací zvolí možnost exportovat prezentace, kdy jsou souhrnně do souboru uloženy údaje o prezentaci. Následně je nabídnuto garantovi uložit soubor na disk a v případě potvrzení je soubor uložen.

Matice relací

EA nabízí nástroj matice relací (v originále Relationship Matrix) pro zachycení formou vizualizace relací mezi požadavky a případy užití. Pomocí něj je možné zkontrolovat, zda jsou všechny funkční požadavky realizované. Matice relací je dostupná v rámci přílohy B.

2.3 Tvorba analytického modelu

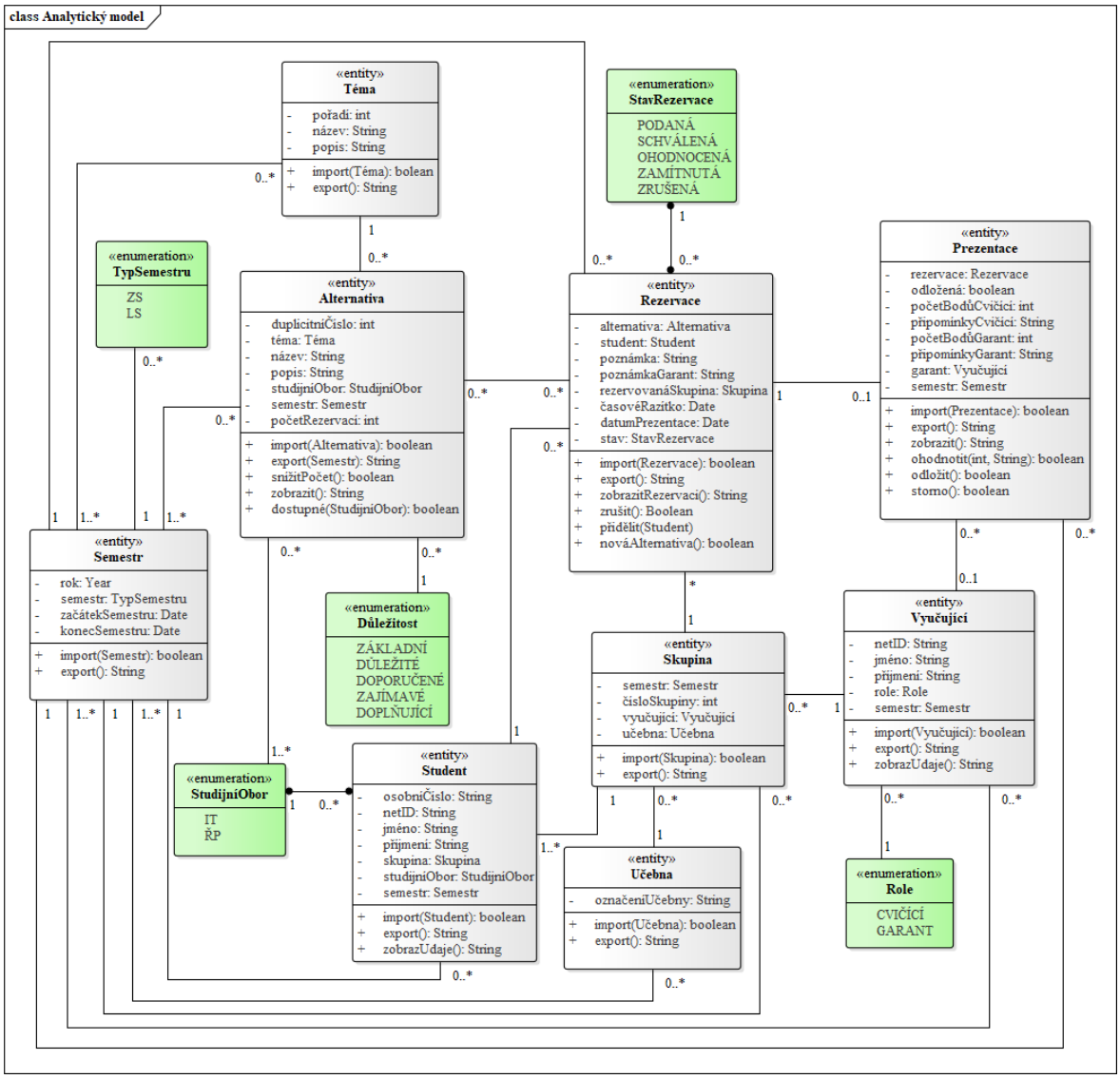
Účelem analytického modelu je zmapovat činnosti IS. Důležité při tvorbě modelu je omezit se pouze na třídy, které jsou součástí problémové domény. Snahou je při vší stručnosti a jednoduchosti formulovat struktury a chování IS. [18]

Základní pravidla tvorby analytického modelu jsou dle [18] shrnuta v následujícím seznamu.

- Vytvořit analytický model v obchodním (terminologickém) jazyce, veškeré abstrakce jsou zahrnuty do slovníku pojmů.
- Popsat určitou část požadovaného chování IS diagramem.
- Zachytit případy užití z určité perspektivy.
- Zaměřit se na abstrakci v rámci problémové domény (obchodní požadavky), nikoliv doménou řešení.
- Minimalizovat vzájemné vazby mezi třídami. Každá asociace vytváří určitou vazbu.
- V analýze nikdy neaplikovat dědičnost pro opětovné použití kódu. Dědičnost je totiž nejsilnější formou spojení mezi třídami.
- Stále zkoumat užitečnost modelu pro všechny uživatele IS a zainteresované osoby. Nejhorším případem je analytický model, který není využit či je dokonce ignorován.
- Snažit se udržovat model co nejjednodušší. Pohlížet na obecný případ modelu.

2.3.1 Analytický model

Analytický model (viz obrázek 24) znázorňuje třídy entit a objekty výčtů (enumeration). Model tříd byl vytvořen formou hledání podstatných jmen ve funkčních požadavcích. Následovalo rozlišení, zda se jedná o třídu nebo jen výčtový typ. Když se zdálo, že jsou nalezené všechny třídy, začal proces vytváření vazeb mezi nimi, pokud tedy nějaké byly. Po vytvoření vazby bylo nutné její přítomnost odůvodnit, zpravidla je dostatečným odůvodněním označení násobnosti.



Obrázek 24: Analytický model [vlastní]

Při „finální“ verzi je vhodné celý model projít, zkontrolovat požadavky a případně celý postup opakovat při nalezení nových tříd. Nejčastěji k tomu může dojít při snaze abstraktně aplikovat případy užití nebo pouze sloves z požadavků. Poté aplikovat doplnění metody u tříd.

V modelu se vyskytuje možnost použití dědičnosti u tříd „student a vyučující“, kde by rodičem těchto tříd mohl být uživatel. Společnými atributy jsou totiž NetID, jméno, příjmení a semestr. Většinou se v IS můžeme setkat i s třídami „uživatel“ a přiřazením příslušné role. Pro větší jednoznačnost odlišnosti kompetencí a uchovaných dat, je však použito oddělených tříd.

Následuje popis jednotlivých tříd modelu, kde je kladen důraz na informace, které nemusely být doposud zřejmé a poslouží čtenáři v lepší orientaci modelem.

Semestr

Třída uchovává informace o importovaném akademickém roce dle požadavku REQ037. Společně s informací o roce, je požadováno uchovávat začátek a konec semestru. Třída je navrhnuta tak, aby bylo možné odlišit semestry za pomoci vazby na výčet typ semestru.

Vazby třídy-semestr se dále vztahují na všechny ostatní třídy. Každý objekt z těchto tříd může, ale nemusí být v daném semestru. Cílem těchto vazeb je naznačit možnost archivace údajů. Předpokládá se, že kromě vyučujícího a učebny, není objektů, které by patřily do více semestrů.

Alternativa

Třída uchovává informace o alternativách, které je možné si studentem rezervovat pro svou prezentaci. Alternativa je požadavkem vymezena pro studijní obory (IT, ŘP), kde ale může být dostupná pro oba studované obory. Výčtem u alternativy je také obsahová důležitost, dle které se dělí její priorita při rezervaci. V případu třídy alternativ je třeba věnovat pozornost atributu duplicitní číslo. Tento atribut je součástí řešení omezení u rezervací, kde by se pro daný semestr nemělo vyskytnout vícekrát.

Další vazby třídy alternativa vedou k třídám téma a rezervace. Skupina alternativ spadá pod jedno téma, které má dané pořadí v rámci průběhu semestru. Tomu odpovídá i násobnost asociace mezi těmito třídami. Alternativa může, ale nemusí být rezervována studentem.

Téma

Třída téma uchovává informace o jednotlivých tématech. Hierarchicky se jedná o kategorii, do které patří podkategorie typu alternativ. Jedno téma obsahově naplňuje několik alternativ, z čehož vychází daná vazba.

Rezervace

Třída rezervace v sobě uchovává informace o vytvořených rezervacích studenty. Rezervace může mít několik stavů, do kterých se může dostat. Tyto stavy jsou definovány výčtem stav rezervace. Rezervace v sobě však uchovává vždy poslední stav a nosí v sobě časové razítko svého vzniku.

Důležitost této třídy je v IS zásadní. Vazby má s třídami prezentace, skupina a student. Na základě rezervace může, ale nemusí vzniknout prezentace. Rezervace se váže k dané skupině, kde bude prezentována. Skupina nemusí odpovídat skupině, do které patří student. Její vytvoření pak má na svědomí určitý student, čemuž odpovídá poslední vazba.

Prezentace

Třída prezentace uchovává informace o prezentaci. Prezentace mohla být z nějakého důvodu odložena dle požadavku REQ021. Může také ale dojít ke stornování této informace. Zpravidla však k druhému odložení nedochází.

Prezentace má mimo semestru vazbu na vyučujícího. Je to z důvodu hodnocení garanta. Informace o cvičícím je dohledatelná z rezervace.

Vyučující

Třída vyučující uchovává informaci o vyučujícím. Vyučujícím může být souhrnně cvičící či rutinou stanovený garant. Pochopitelně je jeho doposud nezmíněná vazba na skupinu, kde však nemusí působit v žádné.

Skupina

Třída skupina uchovává informaci o skupině, která představuje rozvrhovou akci studentů. Kromě zmíněných vazeb má také vazbu na učebnu, kde probíhá výuka, a samotné studenty, kterých může být nespočet.

Učebna

Třída učebna uchovává pouze údaj o tom, kde probíhá rozvrhová akce skupiny.

Student

Poslední nezmíněnou třídou je student. Na rozdíl od vyučujícího v sobě obsahuje dva identifikátory: osobní číslo a NetID. Nejvhodnějším unikátním identifikátorem bude při mapování na ER diagram rozhodně osobní číslo. NetID se navazováním či opakováním studia většinou nemění.

2.4 Sekvenční diagramy

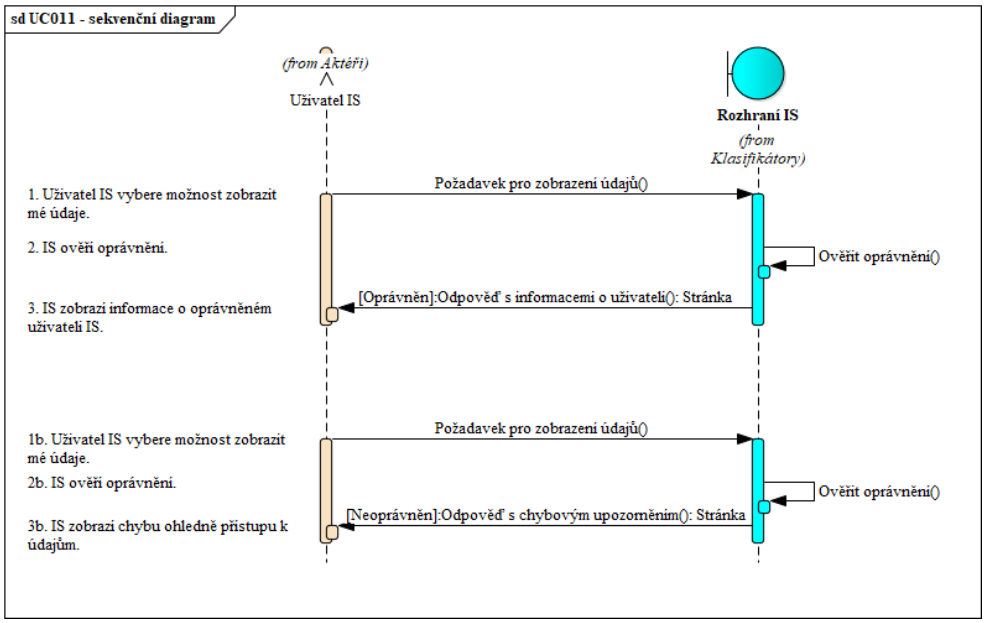
Sekvenční diagramy znázorňují interakce jako časově uspořádanou posloupnost událostí. Jedná se o nejbohatší a nejpružnější formu interakcí mezi instancemi analytických tříd či aktéry. [18]

Cílem sekvenčních diagramů je nejen otestovat dosavadní závěry analýzy, ale případně definovat nové požadavky či poupravit samotné případy užití. Je to součástí analytického procesu, kde se případy užití, diagram analytických tříd a sekvenční diagramy postupně vyvíjejí. [18]

V tomto oddíle budou popsány pouze vybrané sekvenční diagramy. Ostatní a případné změny v předchozích diagramech budou dostupné v rámci přílohy A, souboru s EA projektem. Sekvenční diagramy jsou zpracovány v rámci balíčku „Realizace případů užití."

UC011 Zobrazit mé údaje

První sekvenční diagram (viz obrázek 25) znázorňuje komunikaci mezi aktérem uživatel IS a rozhraním IS. Uživatel chce v rámci případu užití zobrazit jeho údaje uložené v IS. Po ověření oprávnění se diagram štěpí v „oprávněn a neoprávněn" (štěpení alternativních scénářů je znázorněno s postfixem). Symbol v hranatých závorkách znázorňuje podmínku pro scénář.



Obrázek 25: Sekvenční diagram pro zobrazení údajů o uživateli IS [vlastní]

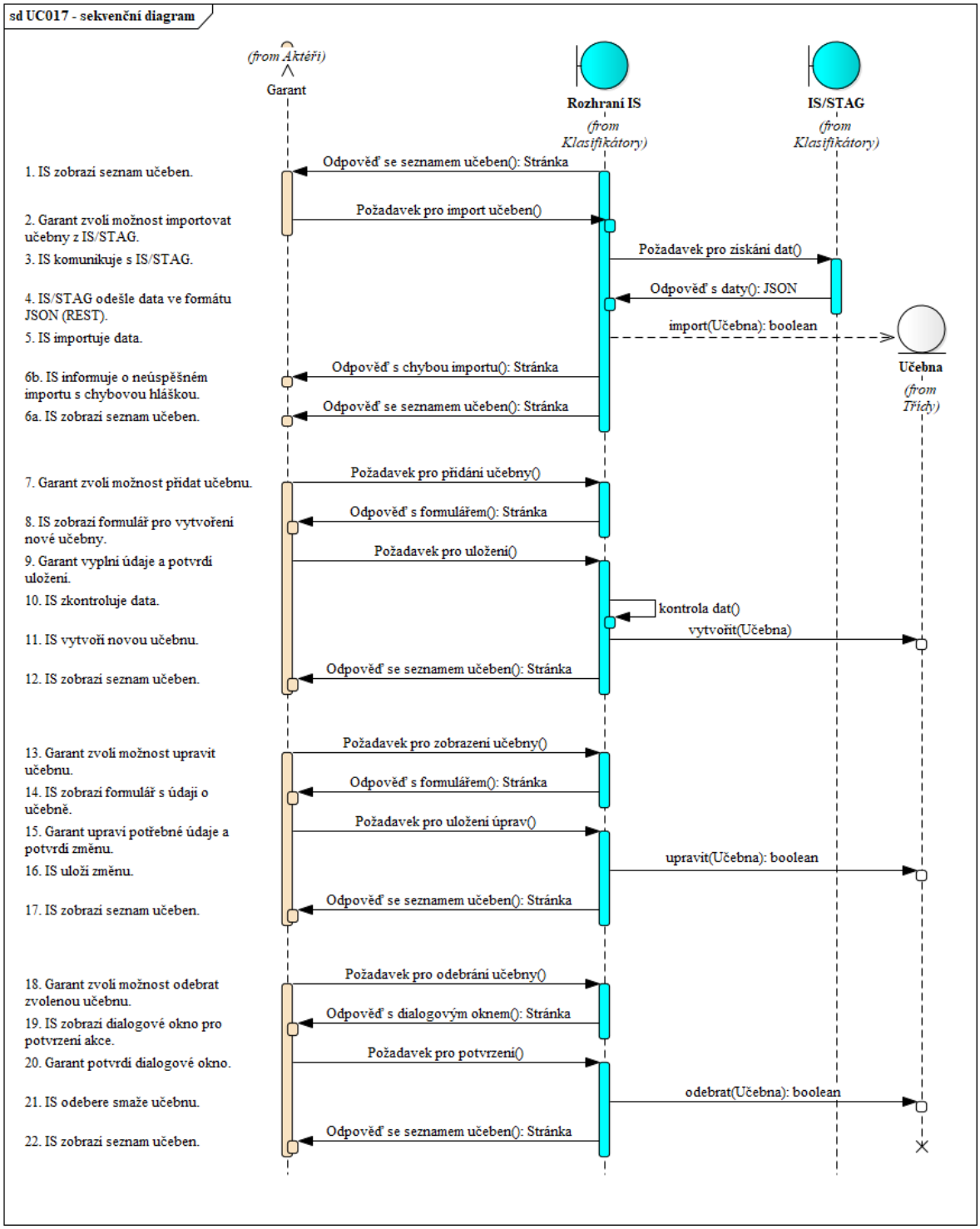
UC017 Správa učeben (CRUD)

Složitější sekvenční diagram pro znázornění správy učeben (viz obrázek 26) souhrnně představuje průchod jednotlivými operacemi včetně 4 alternativních scénářů v rámci diagramu aktivit v příloze A.

Aktér garant přistupuje k rozhraní IS, kde volá jednotlivé operace nutné pro správu dat. Operace jsou oddělené podobně jako u předchozího diagramu čárkovanou čarou. Posloupnost operací je dána diagramem aktivit, který začíná zobrazením seznamu učeben.

Garant nejprve zvolí operaci pro import učeben z IS/STAG. IS proto musí interoperabilovat s IS/STAG k získání dat. Jakmile mu IS/STAG poskytne požadovaná data, IS je importuje a opět zobrazí garantovi seznam učeben. Import se nemusí podařit. V tom případě o tom IS garanta informuje.

První operací CRUD je vytvoření. Garant nad zobrazeným seznamem zvolí možnost přidat učebnu. IS mu poskytne formulář pro vyplnění potřebných údajů. Jakmile je garant odešle pro uložení, IS zkontroluje data a vytvoří novou učebnu. Nakonec zobrazí garantovi zpět seznam učeben.



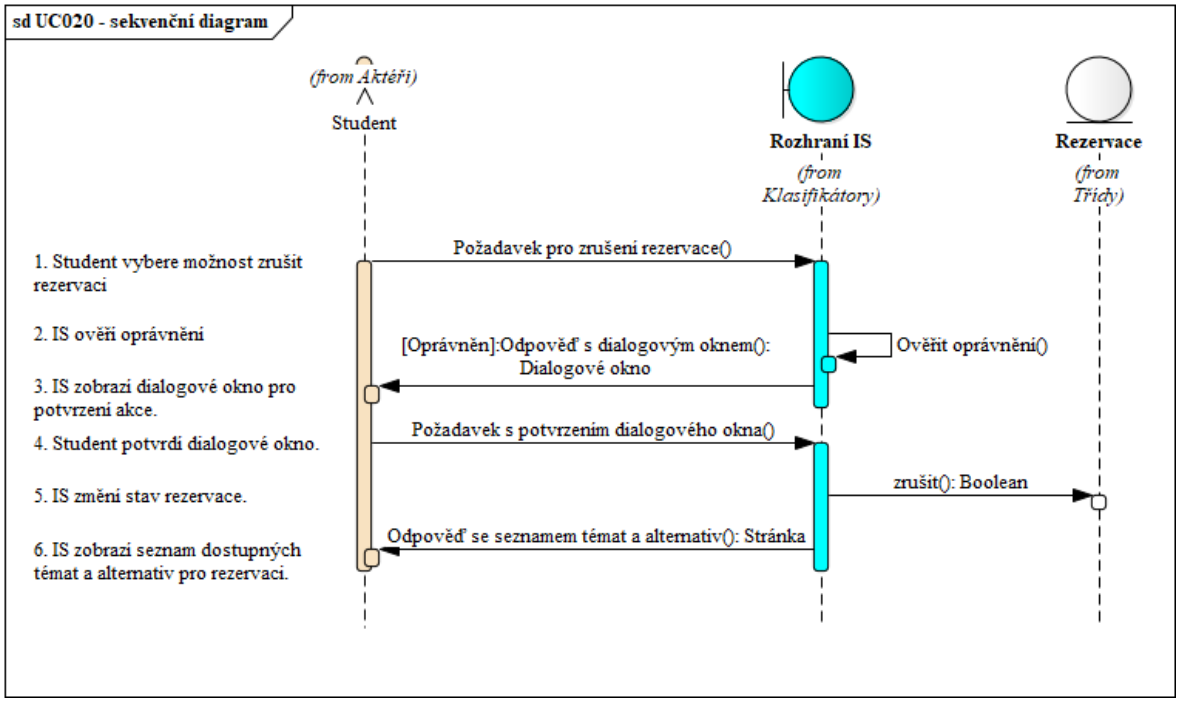
Obrázek 26: Sekvenční diagram pro správu učeben (CRUD) [vlastní]

Druhá operace CRUD je zobrazování seznamu učeben, proto bude popsána třetí operace, úprava. Garant ze seznamu zvolí učebnu, kterou potřebuje spravovat. IS mu zobrazí formulář pro danou učebnu. Po vyplnění a odeslání údajů, se změny uloží a garantovi je opět zobrazen seznam učeben.

Poslední operací CRUD je mazání, kde garant zvolí ze seznamu učebnu pro odebrání. IS vynutí potvrzení dialogovým oknem. Po potvrzení garantem je provedeno smazání učebny a zobrazení seznamu učeben.

UC020 Zrušení rezervace

Třetím sekvenčním diagramem (viz obrázek 27) je případ užití, kdy aktér student ruší svoji rezervaci. Student při zobrazení své rezervace zvolí možnost zrušit rezervaci. IS ověří jeho oprávnění k rušení dané rezervace a vynutí potvrzení dialogového okna pro zrušení. Potvrzením dialogového okna je změněn stav u rezervace na zrušená.



Obrázek 27: Sekvenční diagram pro zrušení rezervace [vlastní]

3 Návrh relační databáze

Třetí kapitola se zabývá návrhem relační databáze. Návrh vychází z předchozí kapitoly, kde byly stanoveny požadavky na IS a případy užití společně s návrhem analytických tříd.

Základ návrhu databáze vychází především z analytického modelu. Součástí této kapitoly bude jeho mapování na Entity-Relationship (E-R) diagram. Postupně pak z E-R diagramu vytvoříme logický návrh, kde budeme entity mapovat do tabulek, a nakonec vytvoříme cílový fyzický návrh databáze.

K tomu bude využito nástroje pro datové modelování SQL Developer Data Modeleru verze 18.1. Nejprve si však v následujícím oddíle definujeme normalizaci ke zefektivnění použití relačního modelu databáze.

3.1 Normalizace

Normalizace je brána jako technika pro vytvoření sady tabulek s minimální redundancí, která podporuje požadavky organizace. [22]

Tato technika probíhá jako řada testů nad tabulkami pro určení, zda splňují pravidla určité normální formy. Normálních forem se definuje obvykle pět, ale nejpoužívanější jsou první tři, které budou v rámci oddílu 3.1 představeny.

Redundance dat

Hlavní cíl při návrhu relační databáze je vytvořit takové uskupení sloupců v tabulkách, aby se minimalizoval počet zbytečných informací. Zredukuje se tím také potřebné uložení implementovaných podkladů tabulek. [22]

Redundantní data poznáme tak, že se opakují informace ve sloupcích zkoumané tabulky. Slabým místem takových tabulek je modifikace těchto dat, kde by se musela procházet celá tabulka a měnit každý výskyt, což v případech miliónu záznamů může být již výpočetní problém nerealizovatelný v reálném čase. [22]

Tento problém se ale netýká jen modifikace dat, ale také vkládání či mazání dat. Problém u vkládání je především způsobem vynucením vkládání dat, která ještě neexistují.

Představme si to na příkladu, kdy bychom udržovali tabulku rozvrhových akcí (skupinu) a v ní také informace o jednotlivých studentech. Část sloupců by reprezentovala skupinu

a další část údaje o studentovi. Pokud by nastalo, že skupina zrovna nebude mít žádné studenty, svádělo by nás to pro vkládání prázdných hodnot (null) do sloupců s údaji o studentovi. To by však dle integrity dat nebylo možné kvůli primárnímu klíči. [22]

První normální forma

První normální forma (1NF) je kritickou normální formou pro tvorbu vhodných tabulek pro relační databáze. Následující normy jsou volitelné, ale pro eliminaci zmíněné redundance je doporučeno používat třetí normální formy (3NF). [22]

Tabulka je v 1NF, pokud každý záznam ve sloupci obsahuje pouze jednu hodnotu. Například v případech telefonních čísel distribučních center (viz tabulka 5) nebude záznam obsahovat více telefonních čísel. Vyskytne-li se v tabulce porušení 1NF, je třeba všechny související sloupce (telefonní čísla) převést do nové tabulky s telefonními čísly a určit telefonní číslo jako primární klíč. [22]

Tabulka 5: Porušení 1NF v tabulce distribučních center [22]

dCentrČíslo	distrAdresa	distrTelČíslo
D001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
D002	City Center Plaza, Seattle WA 981 22	206-555-6756, 206-555-8836
D003	14 – 8 Avenue, New York, NY 10012	212-371-3000
D004	2 W. El Camino, San Francisco, CA 940 84	822-555-3131, 822-555-4112

Druhá normální forma

Druhá normální forma (2NF) se vztahuje na tabulky se složenými primárními klíči. Tabulka se nachází v 2NF, pokud se nachází v 1NF a zároveň platí, že všechny hodnoty sloupců, které nejsou primárním klíčem, jsou jednoznačně definované hodnotami sloupců, které tvoří primární klíč. [22]

Porušení 2NF poznáme dle příkladu z [22], kde máme tabulku, která uchovává záznamy o zaměstnancích, kteří odpracovali týdně v rámci více distribučních center určitý počet hodin (viz tabulka 6).

Tabulka 6: Porušení 2NF v tabulce odpracovaných hodin zaměstnanců [22]

číslo**Zam**	dCentr**Číslo**	**jméno**	**pozice**	**odpracováno**
54555	D002	Ellen Layman	Asistent	16
54555	D004	Ellen Layman	Asistent	9
54612	D002	Dave Sinclair	Asistent	14
54612	D004	Dave Sinclair	Asistent	10

V tomto případě můžeme problém vyřešit vytvořením nové tabulky pro zaměstnance, čímž odstraníme z tabulky 6 sloupce jméno a pozici. Důležité je při procesu normalizace do vyšších normálních forem, zachovat funkční závislost z původní tabulky. To znamená, že v tabulce odpracovaných hodin necháme údaj o číslu zaměstnance, které bude tak sloužit jako cizí klíč do tabulky zaměstnanců a zachová funkční závislost[7]. [22]

Třetí normální forma

Tabulka je ve třetí normální formě (3NF), pokud je již v 1NF, 2NF a všechny hodnoty ve sloupcích, které nepatří mezi primární klíče, jsou definovatelné pouze sloupci primárního klíče a nejsou definovatelné žádnými jinými sloupci. [22]

Porušení 3NF je zřejmé z tabulky 7. Tabulka slouží pro uchování údajů o zaměstnancích distribučních center společně s údaji o distribučních centrech. O této anomálii byla zmínka v pododdíle o redundanci dat. [22]

[7] Funkční závislost je označení pro souvislost mezi sloupci, které se v tabulce nachází: číslo zaměstnance, distribuční centrum a počet odpracovaných hodin. Hodnotu odpracovaných hodin totiž bez znalosti předchozích nemůžeme určit. [22]

Tabulka 7: Porušení 3NF v tabulce pro uchování zaměstnanců [22]

čís-loZam	jméno	pozice	příjem	dCentr-Číslo	dAdresa	dTelefon
S1500	Tom Da-niels	Manažer	48 000	D001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0003	Sally Adams	Asistent	30 000	D001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0010	Mary Marti-nez	Manažer	51 000	D002	City Center Plaza, Se-attle WA 981 22	206-555-6756
S3250	Robert Chin	Asistent	33 000	D002	City Center Plaza, Se-attle WA 981 22	206-555-6756
S2250	Sally Stern	Manažer	48 000	D004	2 W. El Ca-mino, San Francisco, CA 940 87	822-555-3131
S0415	Art Peters	Manažer	42 000	D003	14 – 8 Ave-nue, New York, NY 10012	212-371-3000

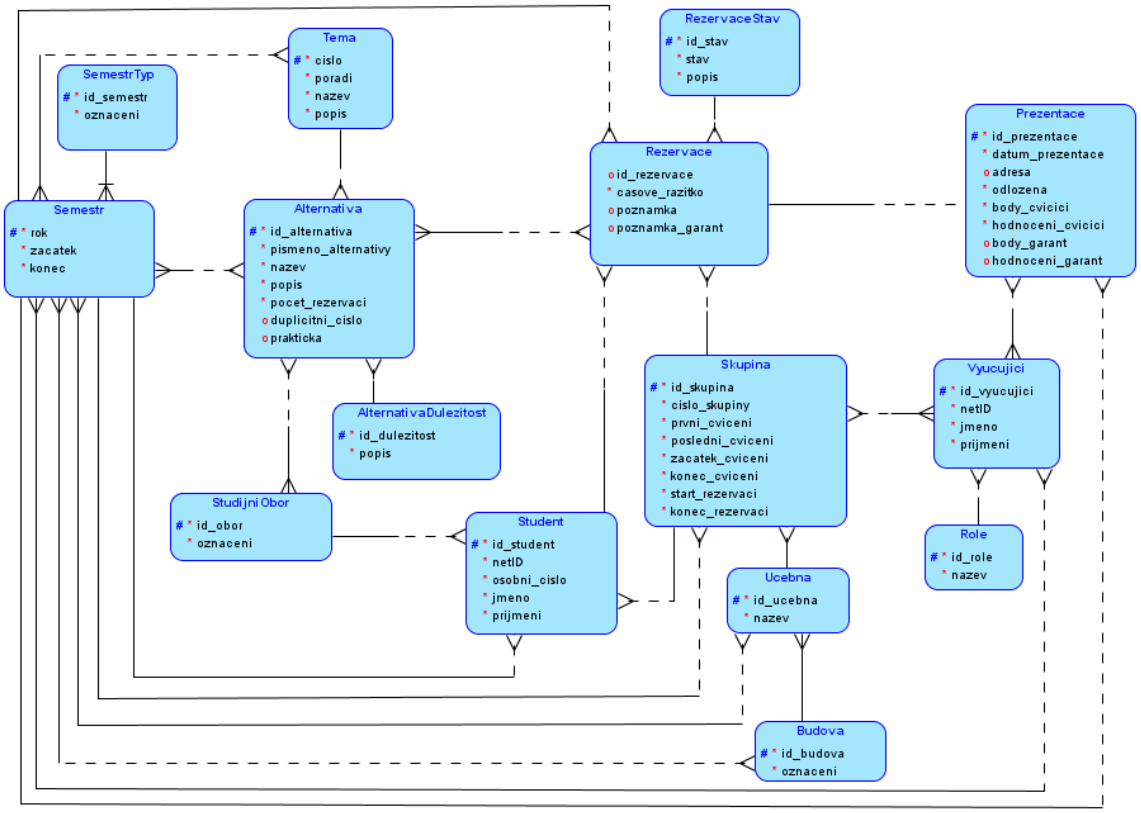
Anomálii vyřešíme vytvořením tabulek pro zaměstnance a distribuční centra. V tabulce 7 poté uchováme pouze sloupce s čísly zaměstnanců a distribučních čísel, které nyní budou sloužit jako cizí klíče a zachovají díky tomu funkční závislost odkazem na data v nově vzniklých tabulkách. [22]

3.2 E-R diagram

Vytvoření E-R diagramu je v rámci fáze konceptuálního návrhu, kde se snažíme zachytit dle [22]: entity, jejich relace, atributy entit a jejich domény, primární i cizí klíče a integritní omezení.

V rámci pododdílu 2.3.1 analytického modelu byla vybudována hrubá představa, kterou lze použít pro výsledný návrh. Diagram bude také doplněn o již existující databázové řešení [21]. V něm se totiž mohou skrývat data, na které nebyl při analýze kladen důraz a zadavatel by je mohl později přeci jen využít.

Následující obrázek představuje navržený E-R diagram. Na rozdíl od analytického modelu v sobě již obsahuje informace o primárních klíčích (symbol křížku) u jednotlivých entit, nepovinné atributy (symbol kroužku) a nepovinné vazby (čárkovaná čárka). Vidlice pak značí kardinalitu vazeb (u analytického modelu ji představovala násobnost).



Obrázek 28: E-R diagram [vlastní]

Oproti analytickému modelu je přidána entita představující budovu pro učebnu. V původní databázi [21] bylo označení pro budovu a učebnu uchováno společně v jednom záznamu. Pro redundanci dat je vhodnější uchovávat informace odděleně.

V následujícím výčtu budou zmíněny pouze entity, které byly s ohledem na [21] pozměněny.

Alternativa

Entita alternativy obsahuje navíc označení písmene pro alternativu a praktickou ukázku. Písmeno je pro odlišení alternativ v rámci tématu a rutinně se používalo doposud. Praktická ukázka je u většiny alternativ povinná (např. instalace operačního systému Debian), ale může být i nepovinná, jedná-li se o teoretické téma (např. žurnálování).

Skupina

Entita skupina obsahuje i informace o tom, kdy probíhá výuka. Je to pro přehled pro přidělování prezentací a časových možnostech. Dále obsahuje informace o začátku a konci cvičení a časové možnosti rezervace témat.

Prezentace

Entita prezentace v sobě navíc uchovává údaj o Uniform Resource Locator (URL) adrese. Je to pro dostupnost garantovi v případě korekce hodnocení, ale také pro ostatní studenty z důvodů studování obsahu. Uchovává také atribut data prezentace, které bylo přesunuto z entity rezervací.

Rovněž byla pozměněna vazba s entitou vyučující, kde může dojít k vícero hodnocením od cvičícího a také garanta.

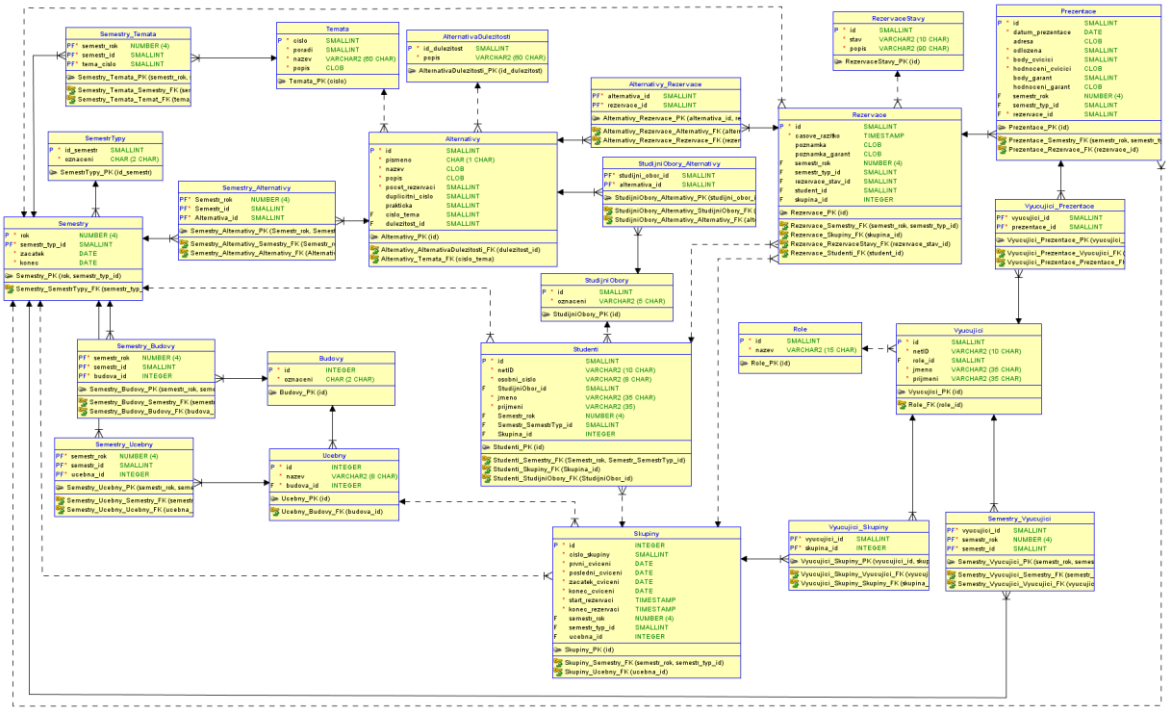
3.3 Fyzický návrh databáze

Cílem oddílu 3.3 je nejprve zdokumentovat mapování E-R diagramu na logický. Diagram přetransformuje jednotlivé entity v tabulky, které je vhodné zkontrolovat dle normalizace. Následně provést generování DDL[8] skriptů pro přípravu tabulek.

[8] Množina SQL příkazů pro definování datových struktur (tabulek).

Mapování na logický diagram

Pomocí Data Modeleru (zkrácený název) je transformace provedena jednoduchým použitím „Engineer to Relation Model“ za předpokladu, máme-li předpřipravený E-R diagram a logický model[9]. Výsledný diagram je pak patrný z následujícího obrázku.



Obrázek 29: Diagram logického návrhu [vlastní]

Diagram logického návrhu (viz obrázek 29) již zobrazuje i domény atributů (datové typy) jednotlivých entit, které nebyly z E-R diagramu patrné. Bez jejich definování by ale nebylo možné výsledný logický návrh získat.

Vznikly také nové tabulky uchovávající kardinalitu M:N. Hovorově se také často označují jako „mezitabulky“, jejichž cílem je uchovávat záznamy o vazbách „každý s každým“. Většinou jejich název bývá ve tvaru: „Relational_Target_Source“.

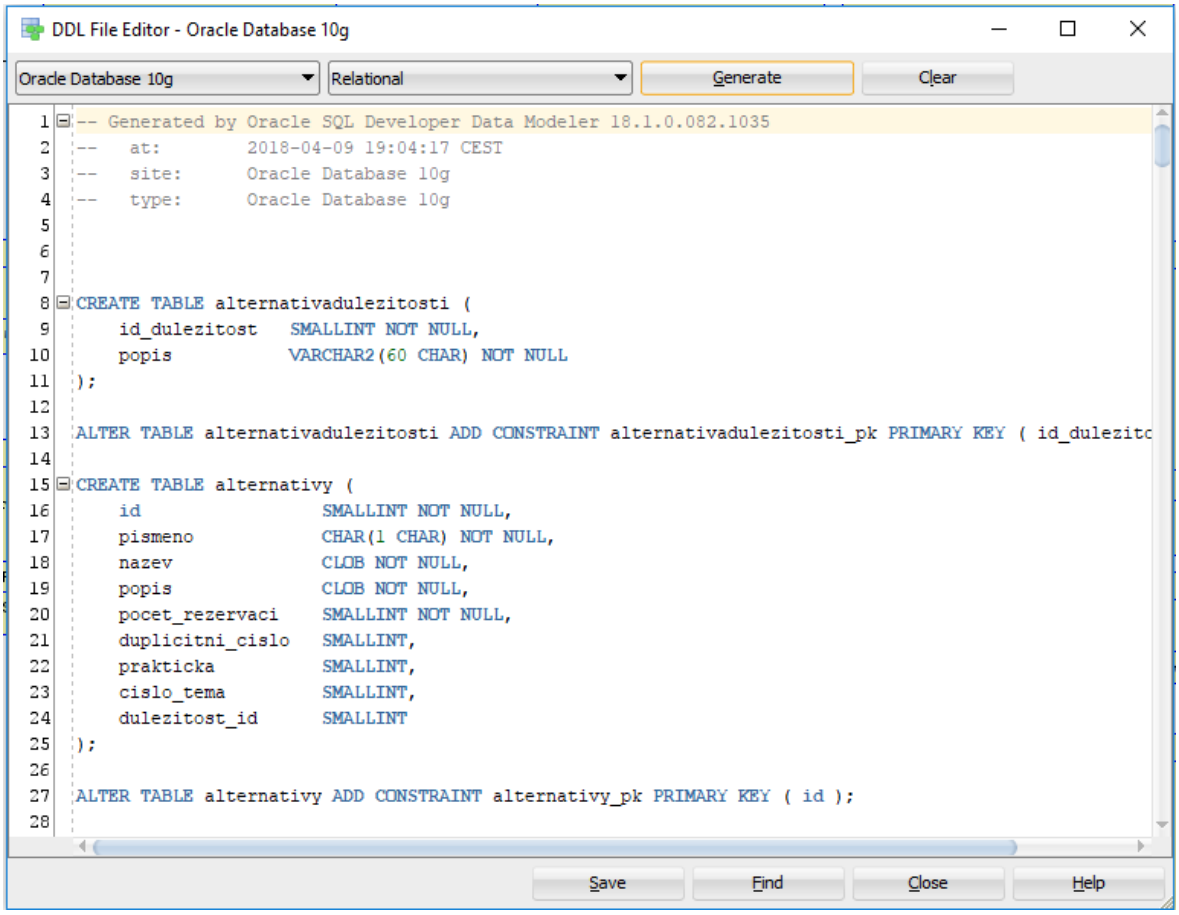
Problémem je ale délka názvu. Proto je v diagramu relational odstraněno a z podobného důvodu došlo k přejmenování některých atributů, sloužících jako cizí klíče v tabulkách.

[9] V Data Modeleru se setkáme s označením Relation Model. Logical Model je označením pro E-R diagram.

U názvů cizích klíčů jsou totiž vygenerovány názvy jejich tabulek, aby bylo pro databázového specialistu jednodušší s nimi pracovat.

Fyzický návrh

Po zkontrolování logického diagramu můžeme díky Data Modeleru provést generování DDL. Formulář s vygenerovanými DDL skripty pak zachycuje následující obrázek, kde byla cílová platforma zvolena jako Oracle Database 10g (pro MySQL bude třeba skripty konvertovat).



Obrázek 30: Generování DDL skriptů pro fyzický návrh [vlastní]

Při generování se často můžeme setkat s chybami, které je třeba vyladit pro správnou funkčnost.

Po úpravě chyb v souboru je vhodné jej zkusit nahrát na server a otestovat, zda se něco nepřehlédlo. Poté chyby upravit a znovu soubor nahrát. K tomu se také využívá vytvoření souboru s množinou dotazů SQL pro mazání vytvořených objektů, jinak při znovunahrání bude mezi chyby patřit pokus o vložení již existujícího objektu.

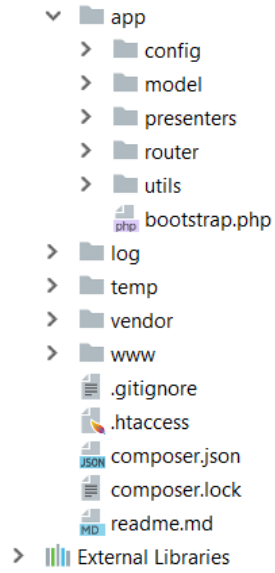
Nakonec nastane stav, kdy se podaří původní soubor DDL nasadit bez chyb a je možné pokračovat ke konverzi do MySQL. U konverzace je nutné hlídat si změnu datových typů a syntaxe u primárních klíčů. Všechny výsledné soubory jsou dostupné v příloze A.

4 Implementace systému

Čtvrtá kapitola popisuje implementovaný systém a navazuje na předchozí. Cílem této kapitoly je představit čtenáři vybrané části implementace pro možnou lepší orientaci v systému, případně obeznámit čtenáře s některými z použitých technologií.

Kapitola je rozdělena do tří pododdílů, které zastřešují vybrané části implementace: frontendovou část (co a jak systém vykresluje svému uživateli), backendovou část (logika, dle které systém vybere, co má být vykresleno) a bezpečnost systému.

Strukturu implementovaného systému popisuje následující obrázek. Struktura vychází z doporučené hierarchie dle [14].

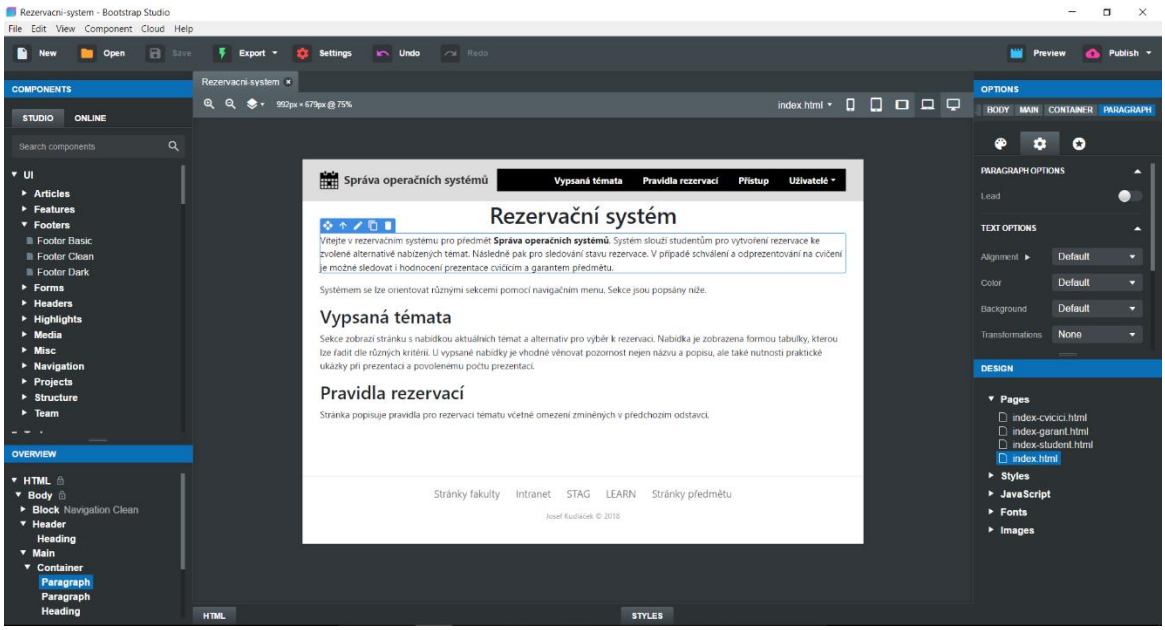


Obrázek 31: Struktura implementovaného systému v Nette [vlastní]

4.1 Frontendová část

Návrh webového rozhraní vychází z využití frameworku Bootstrap pro snadnější a rychlejší vývoj responsivního vzhledu. Pro efektivnější použití Bootstrapu je možné použít placeného nástroje Bootstrap Studio, který poskytuje vývojáři rozsáhlé rozhraní pro návrh vzhledu či vytváření prototypů aplikací. Bootstrap Studio bylo použito ve verzi 4.1.6. [23]

Výsledný návrh společně s rozhraním Bootstrap Studia je na následujícím obrázku.



Obrázek 32: Návrh webového rozhraní pomocí nástroje Bootstrap Studio [vlastní]

Při práci s Bootstrap Studiem kromě jeho nesporných výhod díky rozhraní bylo objeveno i několik nevýhod. Pro snadnější a rychlejší práci se nepodařilo znovu použít definování některých vlastních CSS. Nástroj pracuje především nad předpřipravenými sadami Bootstrapu. Po prohlédnutí vygenerovaného CSS byly pak vlastní definice zahrnuty do značek div a span.

Samotná práce s Bootstrapem často vyžaduje vlastní definici upravující Bootstrap pro vlastní potřeby. Pro vytvoření rychlého návrhu při menší znalosti CSS se však v kombinaci s Bootstrap Studiem jedná o účinný nástroj.

Latte

Pro vývoj v Nette lze využít i systému šablon Latte [24] pro příjemnější vykreslování stránek společně s PHP. Pomocí Latte lze konstrukce jinak psané v PHP, často i nepřehledně, optimalizovat kód view vrstvy a snadněji jej spravovat (viz následující ukázka 1).

```
{foreach $reservationStates as $state}
    {include "../ReservationStates/column.latte",
      state => $state->id}
    <td>{$state->stav}</td>
    <td>{$state->popis}</td>
    </tr>
{/foreach}
```

Ukázka kódu 1: Vykreslení stavů rezervací v Latte [vlastní]

Latte nabízí mimo čitelnější zápis také makra a různé filtry. To usnadňuje vykreslování některých řešení. Ve výchozím stavu nabízí Latte několik filtrů. Při implementaci však nemusí nabídka filtrů stačit a je tedy vhodné si vytvořit vlastní filtry.

Použití filtrů je pak definováno pomocí svislé čáry. Následující ukázka demonstruje použití při výpisu počtu zbývajících dní do prezentace.

```
<td title="{$reservation->predpokladane_datum|date:"j. n. Y"}">
    {$reservation->predpokladane_datum|countDays|negativeNumber}
</td>
```

Ukázka kódu 2: Použití filtrů v Latte [vlastní]

V titulku tagu `<td>` můžeme vidět použití výchozího filtru pro formátování data, zatímco v obsahu tagu je zobrazený počet dní dle vlastního filtru. Počet dní může být záporný a je proto ošetřeno, aby znaménko minus nebylo vykresleno spojovníkem.

Při použití vlastních filtrů je nutné provést konfiguraci. Toho lze docílit například přidáním filtrů do šablony Nette v `BasePresenteru`. Konfigurování filtru `countDays` znázorňuje následující ukázka.

```
$this->template->addFilter('weekDayCZ', function ($number) {
    return Filters::weekDayCZ($number);
});
```

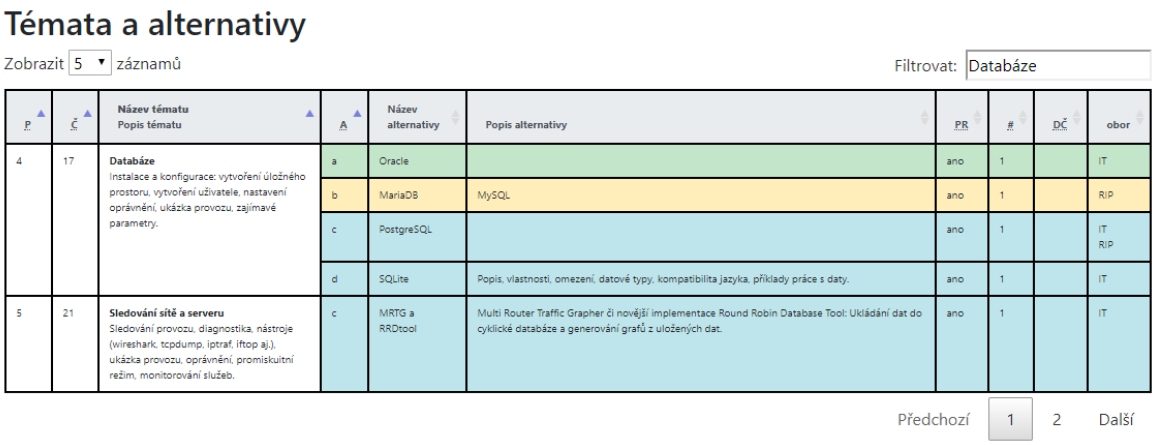
Ukázka kódu 3: Konfigurace vlastního filtru do Latte [vlastní]

S přibývajícím počtem vlastních filtrů je vhodné uvažovat o vytvoření vlastní třídy, která bude funkce jednotlivých filtrů poskytovat. V implementaci systému je proto umístěna v adresáři `../app/utils/` třída `Filters.php`, obsahující použité vlastní filtry.

Soubory Latte je pak možné nalézt v adresáři `../app/presenters/templates/*`, kde každý adresář seskupuje vytvořené soubory pro vykreslení dané problematiky.

DataTables

Společně s využitím jQuery bylo použito i několika doplňků pro dynamičtější obsah. Mezi ně patří doplněk DataTables [25]. DataTables umožňují uživateli především filtrovat data dle zvolených kritérií a ukládat stav tabulky. Při kladených požadavcích a možném rozsahu zobrazovaných dat bylo tedy vhodné užití tohoto doplňku (viz obrázek 33).



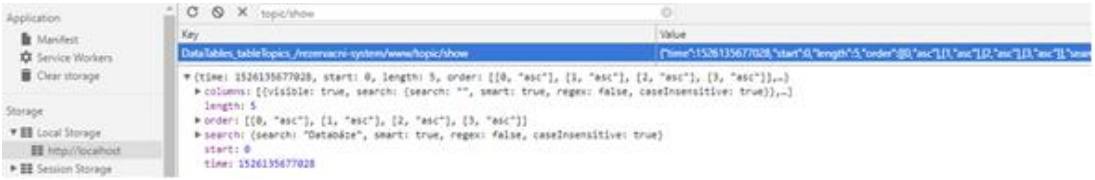
Obrázek 33: Zobrazení témat a alternativ pomocí doplňku DataTables [vlastní]

Nastavení vlastností tabulky pak probíhá v souboru Latte, části pro skript (viz ukázka 4).

```
$(document).ready(function () {
      $('#tableTopics').DataTable({
         stateSave: true,
         info: false,
         stateDuration: 60 * 60 * 24 * 365,
         rowsGroup: [
              0, 1, 2
         ],
         "order": [
              [0, 'asc'],
              [1, 'asc']
         ]
      });
});
```

Ukázka kódu 4: Nastavení DataTables [vlastní]

Ukládání stavu tabulky pak probíhá skrze tzv. Local Storage[10] v daném prohlížeči (viz obrázek 34). Stejného principu je využito i u hodnocení prezentací, kdy jsou data též ukládána do Local Storage.



Obrázek 34: Uložení stavu tabulky do Local Storage doplňkem DataTables [vlastní]

4.2 Backendová část

Implementace backendové části vychází z kapitol 2 a 3. Podle MVC architektury představené v 1. kapitole, bychom ji mohli rozdělit na databázovou vrstvu (Model) a logickou vrstvu (Controller).

Při práci s Nette se v případě logické vrstvy setkáme s terminologickým pojmem tzv. presenter, zmíněn v předchozím oddíle 4.1 při konfigurace vlastních filtrů do šablony.

4.2.1 Databázová vrstva

Databázová vrstva vychází z kapitoly 3, na jejímž závěru byly vygenerovány skripty DDL fyzického návrhu a poupraveny pro databázový systém MySQL. Nejprve je však nutné nastavit připojení k databázi v souboru `../app/config/config.local.neon`, aby bylo s databází možné pracovat.

Adresář `../app/model/*` obsahuje veškeré modely, které poskytují data z potřebných tabulek dle názvu PHP třídy (například `Topic.php`). Pro funkčnost v rámci Nette musí být každý model korektně nastaven pomocí těchto kroků:

1. U vytvořené třídy zajistíme použití jmenných prostorů `Nette`, `Nette\SmartObject` a `App\Model`.
2. Pomocí dependency injection si připravíme globální proměnnou kontextu databáze.

[10] Local Storage (místní uložiště) je prostředkem HTML5 pro možné ukládání dat na straně klienta v rámci jeho prohlížeče. Jedná se o způsob, který nahrazuje řešení pro ukládání dat do cookies či session.

3. Vytvořený model zaregistruje do `../app/config/config.neon`, aby k němu bylo možné přistupovat v rámci aplikace.
4. V modelu použijeme globální proměnné pro vykonávání dotazů nad databází.

Dotazy se vytváří v rámci metod modelu, takže je možné jich volat při vložení objektu modelu do presenteru. Následující ukázka demonstruje vykonání selectu nad tabulkou témat v rámci metody `getTopics()` ve zmíněném modelu `Topic.php`.

```
public function getTopics()
{
    return $this->database->query('
        SELECT * FROM temata
        ORDER BY poradi, cislo');
}
```

Ukázka kódu 5: Vykonání dotazu select nad kontextem databáze [vlastní]

Transakce

Při práci s tabulkami v režimu uložiště InnoDB u MySQL je vhodné u četnějších dotazů pro zrychlení práce s databází a zajištění konzistence dat zajistit správu transakcí. Pro tyto účely byl vytvořen model `Transaction.php`, který obsahuje dvě základní metody k docílení správy: `startTransaction()` a `endTransaction()` (jejich definice je na následující ukázce).

```
public function startTransaction()
{
    $this->database->query("SET autocommit = 0;");
}

public function endTransaction()
{
    $this->database->query("COMMIT;");
}
```

Ukázka kódu 6: Metody pro správu transakcí [vlastní]

Metody, včetně vložení modelu, je vhodné použít s rozvahou v místech presenteru, kdy dochází k četnější práci s databází. K tomu dochází například při importu studentů z rozvrhových akcí v rámci `StagPresenteru`, kde je prováděna řada dotazů viz následující ukázka.

```
$this->transaction->startTransaction();

$scheduleActions = $this->scheduleAction->getActions();
$checkActions = $scheduleActions->getRowCount();
```

```php
if ($checkActions == 0) {
    unset($_SESSION["stagUserTicket"]);
    $this->transaction->endTransaction();

    $this->flashMessage('Semestr neobsahuje žádné rozvrhové akce.',
     'error');
    $this->redirect('importStudents');
}

$opts = array('http' => array(
    'header' => 'Authorization: Basic ' . base64_encode(
     $_SESSION["stagUserTicket"] . ":")
));

$context = stream_context_create($opts);

$this->processScheduleActions($scheduleActions, $context);
$this->student->clearTemporaryTableWithIndex();

$this->template->surplusStudents = $this->student
     ->getTemporaryTableData()->fetchAll();

$this->flashMessage('Import studentů proběhl úspěšně.', 'success');
$this->transaction->endTransaction();
```

Ukázka kódu 7: Použití transakce při importu studentů [vlastní]

Ukázka 7 v sobě obsahuje také použití temporary table při kontrole, zda není v rezervačním systému student, který již není dle IS/STAG zapsán na předmět.

Temporary table

Cílem použití temporary table je v rámci session vytvořit dočasnou tabulku, která bude pro naše potřeby uchovávat vybraná data. Dočasná tabulka existuje pouze pro session, ve které byla vytvořena, a to do doby, než je session ukončena, případně je v rámci session smazána.

Dočasné tabulky je v systému celkem použito ve dvou situacích. První je při zmíněném importu zapsaných studentů z webových služeb IS/STAG. Druhá při importu skupin. Vytvoření dočasné tabulky při importu studentů znázorňuje následující ukázka.

```php
$this->database->query("
    CREATE TEMPORARY TABLE IF NOT EXISTS students_temp (
    `id` SMALLINT(5) UNSIGNED NULL DEFAULT NULL,
    `skupina_id` SMALLINT(5) UNSIGNED NULL DEFAULT NULL,
    `netid` VARCHAR(10) NOT NULL,
    ...
    `priznak` TINYINT(1) UNSIGNED NOT NULL DEFAULT 1,
    PRIMARY KEY (`netid`));");
```

Ukázka kódu 8: Vytvoření dočasné tabulky [vlastní]

Popíšeme si využití tabulky při importu studentů. Tabulka je vytvořena v rámci metody `prepareTemporaryTable()`, kde do ní budou vloženi studenti aktuálního semestru, kteří jsou uchováni v systému a nastaví se jim výchozí příznak na 1. Tím jsme získali množinu studentů v systému.

Následuje vložení studentů, které jsme získali importem z IS/STAG, kdy při výskytu studenta v tabulce, je mu změněn příznak či při vložení vkládán s hodnotou 0. V tabulce pak máme skupinu studentů s příznakem 1, kteří již nejsou zapsaní a můžeme nabídnout garantovi jejich smazání ze systému.

Výhodou tohoto řešení je, že nemusíme v databázi mít další tabulku. Můžeme také změnit primární klíč, kdy nebudeme zkoumat id studentů, ale jejich NetID. Nakonec pomocí příznaků můžeme rozlišit, do jaké množiny student patří pro další potřebné operace.

Zamykání tabulek

Při implementaci a zkoumání stávajícího systému [21] bylo použito i principu zamykání tabulek. Cílem bylo řešit problém souběhu u rezervací, kde v jeden okamžik bude prováděno mnoho operací nad tabulkou rezervace.

Problém souběhu nastává především při UC004, kdy si studenti chtějí rezervovat téma, ale mohou si rezervovat pouze volné téma dle stanovených omezení. Při vytváření diagramu se nepočítalo se situací implementace, kdy se nejprve musí provést dotaz pro zjištění, zda je studentem rezervované téma opravdu dostupné, a až poté dochází k jeho rezervaci.

Může proto nastat situace, kdy se k obsluze přidělení rezervace dostane i proces studenta B. Proces studenta A totiž prozatím zdárně nedospěl ke změně dostupnosti tématu na konci rezervace. Pouze proběhlo vyhodnocení, že je téma dostupné, a po tomto vyhodnocení byl přerušen. Téma tak bylo vyhodnoceno jako stále dostupné i pro proces studenta B. Proces rezervace studenta B tedy postoupí do stavu rezervování, ke kterému nemělo dojít. Ve výsledku pak bude mít téma rezervován student A i student B.

Situaci lze řešit dotazem pro zamknutí tabulky na počátku realizace UC004 a odemčením tabulky na jeho zdárném konci (viz ukázka 9) v metodě `insertNewReservation()`.

```
$this->database
    ->query("
        LOCK TABLES rezervace WRITE, rezervace AS r1 READ;");
```

```
// Kontrola dostupnosti témat
$resultArray = $this->alternative
     ->getAvailableAlternativesForReservations($semester)
             ->fetchPairs('id', 'alternativa');

// Vytvoření rezervace
$result =  $this->database
    ->query("
    INSERT INTO rezervace ... ;");
$this->database
    ->query("
    UNLOCK TABLES;");
```

Ukázka kódu 9: Řešení problému souběhu zamykáním tabulek [vlastní]

Pro předejití vyhladovění čekajících procesů na odemčení tabulky rezervace, je celá realizace ještě zahrnuta do transakce, kdy při jejím neúspěšném provedení bude provedena operace rollback.

4.2.2 Logická vrstva

Cílem logické vrstvy je poskytnout data z vybraných modelů pro vykreslení na frontendové části. Jak bylo zmíněno, v Nette se k tomu používá presenterů. Všechny presentery jsou umístěny v adresáři `../app/presenters/`. Každý presenter zastřešuje danou problematiku pro vykreslení.

Základním presenterem je `BasePresenter`. Při použití jakéhokoliv presenteru se pracuje zároveň i s tímto předkem, od kterého ostatní presentery dědí. Toho se dá vhodně využít nejen při použití vlastních filtrů, ale rutin potřebných na každém presenteru (například nastavení semestru pro garanta nebo získání identity uživatele).

Důležité také je, že `BasePresenter` dědí od třídy presenteru, díky čemuž máme zprostředkovány metody zejména pro obsluhu vykreslení (render) odpovědí či zpracování akcí. Při tvorbě nového presenteru je tedy vhodné dědit právě presenteru `BasePresenter`.

Pokud nebudeme chtít obstarávat žádnou logiku nad vykreslením, nemusíme pro vykreslení Latte šablony v presenteru nutně definovat metodu. Jedná se zejména o případ, kdy pouze zobrazujeme stránku s jejím obsahem.

Jak bylo zmíněno u modelů, pro práci s modelem v presenteru je třeba jej do presenteru vložit. K tomu nabízí Nette několik způsobů. V systému je využito vložení přes konstruktor a injektování, viz následující ukázka.

```php
private $building;

/** @var Model\Semester
 * @inject
 */
public $semester;

public function __construct(Model\Building $building)
{
    $this->building = $building;
}
```

Ukázka kódu 10: Práce s modelem v presenteru [vlastní]

V ukázce 10 můžeme vidět použití vložení objektu modelu budov v presenteru `Building-Presenter`. Zpravidla je využito principu, kdy model odpovídající problematice presenteru je vložen skrze konstruktor a ostatní potřebné modely pomocí injektování.

Takto vložené modely pak představují v příslušném presenteru globální proměnné, ke kterým lze v případě potřeby přistupovat v metodě presenteru. Metoda presenteru pak funguje jako klasický soubor PHP, ve které provádět potřebné PHP příkazy.

V presenteru si také můžeme vytvořit privátní metody, které budou využity například při přípravě vykreslování odpovědi presenteru. Můžeme si také vytvořit opět třídu znovupoužitelných konstrukcí – jakožto bylo provedeno v případě statické třídy `Utils.php`, umístěné v adresáři `../app/utils/`.

Jelikož se jedná o statickou třídu, nemusíme ji injektovat, ale musíme použít jmenného prostoru `App/Utis/Utils` a zaregistrování do souboru `../app/config/config.local.neon`. Jedná se totiž o třídu mimo adresář a je třeba ji lokalizovat.

Příkladem je pak následující ukázka 11 při exportování souboru CSV s alternativami.

```php
public function actionExport()
{
    if ($_SESSION["role"] === 1) {
        $currentSemester = Utils::currentSemester($this->semester
            ->getCurrentSemester());
        $alternatives = $this->alternative
            ->prepareAlternativesWithTopicsForExport(
                $currentSemester)->fetchAll();
        $header = array('id_tematu', 'poradi_tematu',
            'cislo_tematu', 'nazev_tematu', 'popis_tematu',
            'dulezitost', 'id_dulezitost', 'id_alternativy',
            'pismeno_alternativy', 'nazev_alternativy',
            'popis_alternativy', 'pocet_rezervaci',
```

```
                'duplicitni_cislo', 'obor_IT', 'obor_RIP');

            Utils::exportCSV($header, $alternatives, "alternativy");

        } else {
            $this->flashMessage('Nedostatečná oprávnění!', "error");
            $this->redirect('Homepage:default');
        }
}
```

Ukázka kódu 11: Export souboru alternativ [vlastní]

V ukázce můžeme vidět zpracování akce export, která je volaná při zvolení exportu alternativ při jejich správě. V metodě můžeme vidět použití znovupoužitelných metod `currentSemester()`, pro zjištění aktuálního semestru, a `exportCSV()`, pro přípravu a odeslání souboru CSV. Mimo to obsahuje i práci se session klasickou konstrukcí PHP a přesměrování s výpisem chyby v případě nedostatečných oprávnění.

V těch jednodušších případech pak bude vykreslovací metoda presenteru pouze předávat data do šablony k vykreslení v Latte (viz ukázka 12 s vykreslením témat).

```
public function renderShow()
{
    $this->template->alternativeImportance = $this
            ->alternativeImportance->getAlternativeImportances();
    $currentSemester = Utils::currentSemester($this->semester
            ->getCurrentSemester());
    $this->template->alternatives = $this->alternative
            ->getAlternativesWithTopics($currentSemester);
}
```

Ukázka kódu 12: Předání dat z modelu do šablony [vlastní]

V Latte pak přistupujeme k proměnným `alternativeImportance` a `alternatives`, které v sobě uchovávají data z modelů `AlternativeImportance` a `Alternative`.

Presentery také slouží pro přípravu komponenty formuláře s potřebnými parametry a zpracování jejích výsledků. V Latte pak obstaráváme konkrétní podobu komponenty. Jako příklad může sloužit vytváření stavu rezervace pomocí formuláře `newStateForm`.

Při vykreslování odpovědi `ReservationStatePresenter` vytvoří komponentu formuláře pomocí metody `createComponentNewStateForm()` (viz ukázka 13).

```
protected function createComponentNewStateForm()
{
    $form = new Form;
```

```
        $form->addText('stav')
            ->setRequired();
        $form->addText('popis')
            ->setRequired();
        $form->onSuccess[] = [$this, 'newStateFormSucceeded'];
        $form->addSubmit('send', 'Vytvořit');

        return $form;
    }
```

Ukázka kódu 13: Vytvoření komponenty formuláře v presenteru [vlastní]

V metodě jsme mu nastavili potřebné parametry `stav` a `popis`. Stejně tak i metodu `newStateFormSucceeded()`, kterou bude po validním odeslání formuláře volat. Validitu formuláře můžeme definovat v metodě nebo pomocí HTML5 na vykreslené stránce.

Formulář pak vykreslíme například pomocí maker `form` v souboru `create.latte`, kde můžeme definovat výsledný vzhled pomocí Bootstrapu (viz ukázka 14).

```
<form n:name=newStateForm class=form>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Název stavu:</label>
        <div class="col-sm-5">
            <input n:name=stav type="text" maxlength="10"
             class="form-control" />
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Popis stavu:</label>
        <div class="col-sm-5">
            <input n:name=popis type="text" maxlength="90"
              class="form-control" />
        </div>
    </div>
    <div class="form-group row">
        <div class="col-sm-2">
        </div>
        <div class="col-sm-10">
            <input n:name=send id="btn" class="btn btn-primary"
              value="Vytvořit">
            <button onclick="goBack()" type="button"
              class="btn btn-secondary">Zpět</button>
        </div>
    </div>
</form>
```

Ukázka kódu 14: Vykreslení formuláře v Latte [vlastní]

Jakmile je formulář úspěšně odeslán, data jsou zpracována v metodě, kterou jsme definovali při tvorbě komponenty formuláře v presenteru. Bude se tedy jednat o metodu

`newStateFormSucceeded()` v příslušném presenteru `ReservationStatePresenter`. Zpracování pak znázorňuje následující ukázka.

```
public function newStateFormSucceeded($form)
{
    $stateId = $this->getParameter('stateId');

    if ($stateId) {
        $result = $this->reservationState->updateState($stateId,
             $form->getValues());
        if (empty($result)) {
            $this->flashMessage('Nebyl upraven žádný stav
                    rezervace.', 'error');
        } else {
            $this->flashMessage('Stav rezervace upraven.',
                    'success');
        }
    } else {
        $this->reservationState->insertNewState($form->getValues());
        $this->flashMessage('Vytvořen nový stav rezervace',
                    'success');
    }

    $this->redirect('ReservationState:manage');
}
```

Ukázka kódu 15: Zpracování dat z formuláře [vlastní]

Metoda je pro znovupoužitelnost vytvořena tak, že kontroluje i parametr `stateId`. Parametr se vyskytuje pouze v případě, bude-li formulář využit pro úpravu stávajících dat stavu rezervace. Nemusíme tak vytvářet dvakrát stejný formulář. Pouze si v presenteru připravíme data, kterými formulář předvyplníme v případě přístupu na stránku s úpravou (viz ukázka 16).

```
public function renderEdit($stateId)
{
    if ($_SESSION["role"] === 1) {
        $state = $this->reservationState
             ->getReservationState($stateId);

        if (!$state) {
            $this->error('Stav rezervace nenalezen');
            $this->redirect('ReservationState:manage');
        }

        $this['newStateForm']->setDefaults($state);
```

```
} else {

        $this->flashMessage('Nedostatečná oprávnění!', "error");
        $this->redirect('Homepage:default');
    }
}
```

Ukázka kódu 16: Znovu použitelnost formuláře pro úpravu dat [vlastní]

Interoperabilita

Pro potřeby interoperability s IS/STAG byl vytvořen `StagPresenter` a k němu příslušící soubory Latte v adresáři `/templates/Stag/*`. Jejich úlohou je zpracovat a vykreslit činnost garanta při práci s daty, získanými z IS/STAG.

Skrze webové služby nabízí IS/STAG přístup k datům, kterých bude třeba pro chod systému využít. Na výběr je dle [26] mezi použití rozhraní SOAP či REST. Z pododdílu 1.3.3 vyplynulo, že je výhodnější pracovat s rozhraním REST.

Práce s rozhraním je totožná práci se souborem. Skrze potřebné URL získáme z webových služeb soubor dat, se kterým dále pracujeme (viz ukázka 17). Soubor je pochopitelně ve formátu Javascript Object Notation (JSON).

```
$json = file_get_contents(
    'https://stag-ws.upce.cz/ws/services/rest/kalendar/getHarmono
     gramRoku?outputFormat=json&rok='. htmlspecialchars($year));
$stdObject = json_decode($json);
```

Ukázka kódu 17: Získání dat skrze REST rozhraní IS/STAG [vlastní]

Získaný soubor je potřeba zpracovat a vybrat z něj potřebná data. K tomu je potřeba znát strukturu souboru, podle které víme, k jakým atributům budeme přistupovat (např. skrze popis služeb či předzpracování souboru). Soubor je pro tyto potřeby dekódován do generické třídy PHP `stdClass`, se kterou je dále pracováno viz následující ukázka.

```
$harmonogram = $stdObject[0]->harmonogramItem;

foreach ($harmonogram as $key => $place) {
    $description = $place->popis;
    if (strpos($description, 'Začíná: Zimní semestr') !== false) {
        if (property_exists($semester, 'zacatek')) {
            continue;
        }
```

```
$semester = (object)array_merge((array)$semester,
                array('zacatek' => $place->datumOd->value));
    }
}
```

Ukázka kódu 18: Zpracování objektu třídy stdClass [vlastní]

4.3 Bezpečnost systému

Poslední oddíl 4.3 je věnován bezpečnosti systému a využitých technik k jejich zajištění. Úvodní část bude věnována ověřování uživatelů za pomoci Shibboleth a jejich autorizace k požadovaným operacím. Další část se zabývá softwarovou ochrannou proti XSS, SQL injection a snahou o zabezpečení mnohonásobného odeslání požadavku v rámci formuláře.

4.3.1 Uživatelé

Pododdíl se věnuje bezpečnosti z hlediska uživatelů. Popisuje využití Shibbolethu k autentizaci a autorizaci uživatelů. Na závěr poddílu je věnována pasáž ohledně aktuálního tématu s nakládáním citlivých údajů.

Shibboleth

Shibboleth je technologie pro ověření identity uživatele třetí stranou. Slouží pro možné využití principu Single Sign-on. Jedná se o princip centralizovaného procesu pro přihlášení uživatele na vícero aplikací. Díky tomu je po ověření identity uživatele centrálním prvkem (poskytovatelem identity) identita dostupná všem systémům (poskytovatelům služby), které s centrálním prvkem ním spolupracují. [27]

Pro možné použití je třeba na webovém serveru, provozujícím IS, Shibboleth konfigurovat. Po konfiguraci je pak při prvním přístupu na IS vyžadováno ověření identity uživatele. Se získanou identitou lze poté dále pracovat v rámci superglobální proměnné `$_SERVER` viz následující ukázka.

```
if (isset($_SERVER['eppn']) && $_SERVER['eppn'] != "")
{
    $atPosition = strpos($_SERVER['eppn'], '@');
    $netId = substr($_SERVER['eppn'], 0, $atPosition);
}
```

Ukázka kódu 19: Převzetí identity uživatele ze Shibboleth [21]

Autorizace

V pododdíle 2.2.1 byly definovány základní aktéři systému (garant, cvičící, student a host). Jednotlivému aktérovi je po ověření identity přidělena role dle výskytu jeho NetID v databázi. Pro tento účel byl vytvořen model `User`, který je použit v `BasePresenteru`. Použitím metody `prepareRole()`, viz následující ukázka.

```
private function prepareRole($netId)
{
    $roleId = $this->user->identifyUser($netId);
    return $roleId;
}
```

Ukázka kódu 20: Přiřazení role v systému [vlastní]

Metoda `identifyUser()` prochází se získaným NetID tabulku vyučujících, kde hledá záznam s daným NetID. Pokud je nalezeno, je vrácena hodnota dané role (1 pro garanta, 2 pro cvičícího). Pokud není nalezeno, pokračuje hledání v tabulce studentů aktuálního semestru. V případě nalezení záznamu vrací hodnotu 3, jinak vrací hodnotu 4 (reprezentující roli hosta). Hodnotu role poté udržuje session pro potřeby autorizace. Autorizace probíhá na všech presenterech při volání téměř každé metody presenteru.

GDPR

Obecné nařízení o ochraně osobních údajů (GDPR) přicházející v platnost dnem 25. 5. 2018 stanovuje nová pravidla o zpracování online osobních údajů dle Evropské unie. Rezervační systém je vytvořen pro potřeby výuky. Proto se při jeho použití na něj vztahuje dle čl. 13 a 14 nařízení informovat uživatele systému o zpracování jejich osobních údajů pro chod systému. [28]

4.3.2 Softwarové útoky

Pododdíl popisuje možné softwarové útoky na IS a implementační řešení ochrany před nimi. Nejprve je věnována pozornost injektovacím technikám, kdy útočník využívá možnosti vkládat svůj kód kvůli nedostatečnému zabezpečení. Závěr je věnován implementační snaze bránit se mnohonásobnému zpracování formulářů.

XSS

Cross-Site Scripting (XSS) spočívá ve zneužití neošetřených výstupů. Útočník podstrčí do stránek na neošetřená místa vlastí kód a tím pozmění její fungování nebo může získat citlivé údaje o návštěvnících. Proti XSS je třeba se bránit ošetřením výstupních řetězců. [14]

Těmto útokům lze bránit díky Latte, které automaticky při vypisování proměnných předaných v presenteru ošetřuje – převádí speciální znaky HTML na jiné sekvence. V případě potřeby zobrazení HTML kódu lze využít filtru `|noescape`. [24]

SQL injection

Na rozdíl od XSS se jedná o techniku útočníka, kdy se snaží podstrčit svůj kód při komunikaci s databází. K tomu dochází u neošetřených vstupů zejména po odeslání formuláře. Útočník tak může napáchat škody v databázi nebo získat jakákoliv data.

SQL injection se při implementaci v Nette dá bránit správným vázáním parametrů příkazu. Pomocí bindování pak dochází k escapování parametrů před jejich použití v dotazu. Použití bindování demonstruje následující ukázka.

```
$this->database
    ->query("SELECT @year := ?, @semester := ?;",
      $semester->rok, $semester->oznaceni);
```

Ukázka kódu 21: Bindování parametrů v Nette [vlastní]

V ukázce můžeme vidět zástupný symbol (otazník) pro vkládaný parametr. Parametry jsou vkládány do příslušných pozic symbolů dle uvedeného pořadí.

Pro další využití hodnot parametrů je vidět i použití proměnných MySQL. Díky tomu nemusíme při každé potřebě hodnot bindovat parametry, ale můžeme hodnoty použít nativně formou MySQL. Toho lze využít v případech, kdy bychom potřebovali hodnoty bindovat v rámci jednoho dotazu vícekrát nebo v následujících dotazech metody modelu.

Vícenásobné odeslání požadavku

S čím si však většina frameworků neporadí, včetně Nette, je vícenásobné odesílání jednoho požadavku, zejména u formulářů. K tomuto jevu může dojít vícero kliknutím na odesílací tlačítko. Můžeme tak provádět vícenásobně činnost, která nemusí být systémem při znovuprovedení ošetřena. V krajním případě tak může vzniknout nekonzistentní nebo nesmyslný stav systému či dat.

Jedním rozumným řešením je ošetření na frontendové části, kdy se po odeslání tlačítka pomocí jQuery tlačítko na několik milisekund deaktivuje (viz ukázka 22).

```
var button = $(this).find("input[type='submit']").prop('disabled',
     true);

setTimeout(function () {
    button.prop('disabled', false);
}, 1000)
```

Ukázka kódu 22: Ošetření odeslání formuláře [vlastní]

Nevýhodou tohoto řešení je však chvilková deaktivace tlačítek na další stránce a neošetření případu, kdy uživatel nemá povolené využití JavaScriptu.

Závěr

Cílem práce bylo implementovat rezervační systém dle konzultovaných a předložených požadavků. Nad požadavky byla následně provedena analýza pro definování funkcionalit a omezení systému. Po navržení databáze a webového rozhraní byl systém implementován ve frameworku PHP Nette s využitím javascriptové knihovny jQuery. Vykreslení je provedeno za pomoci šablonového systému Latte a knihovny Bootstrap. Potřebná data systému jsou ukládána v databázi MySQL. Cíl je na základě implementace klíčových požadavků garanta splněn.

Implementovaný systém byl nasazen na připravený webový server a podléhá testování funkcionalit pro možné fungování v tzv. „ostrém“ provozu. Pro přístup k funkcionalitám systému je třeba o něj požádat autora práce či garanta předmětu. Systém je dostupný v rámci vnitřní univerzitní sítě, přičemž přiložené zdrojové kódy pouze demonstrují implementaci systému, ale bez možné ověření identity uživatele skrze Shibboleth nejsou zcela funkční.

V práci byly demonstrovány části vývojového procesu informačního systému, které mohou sloužit čtenáři jako opora pro vytvoření vlastního informačního systému s využitím zmíněných technologií. Bohužel kvůli rozsahu a tématu není součástí práce testování systému, konfigurace webového serveru a samotné nasazení. Konfigurace a nasazení proběhlo za pomoci zkušenějšího správce serveru mimo zpracování práce s možností využívat Nette v nejnovější verzi 2.4 při verzi PHP 7.2.

Jelikož se jednalo o první rozsáhlejší projekt autora v Nette, potažmo PHP, je místy v kódu vidět různé typy programových konstrukcí – framework Nette × přímé PHP. U některých konstrukcí totiž dochází k přímému využívání PHP pro nemožnost implementace v rámci Nette. Výsledná implementace si zaslouží refaktoring kódu, který nebyl v rámci mezí možný.

Během implementace požadavků byly nalezeny možné nové funkcionality, které by mohly být zahrnuty v další verzi systému.

Použité zdroje

[1] BRUCKNER, Tomáš. *Tvorba informačních systémů: principy, metodiky, architektury*. Praha: Grada, 2012. ISBN 978-80-247-4153-6.

[2] ROYCE, Winston. *Managing the Development of Large Software Systems* [online]. University of Southern California, release date: 1970. [cit. 3.3.2018]. Dostupné z: https://goo.gl/LmvRQu

[3] KADLEC, Václav. *Agilní programování: metodiky efektivního vývoje softwaru*. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

[4] *Vodopádový model* [online]. Testování softwaru: 2011-2016 [cit. 3.3.2018]. Dostupné z https://goo.gl/8Cr5jW

[5] SOMMERVILLE, Ian. *Softwarové inženýrství*. Brno: Computer Press, 2013. ISBN 978-80-251-3826-7.

[6] BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. Praha: Grada, 2005. ISBN 80-247-1075-7.

[7] MYSLÍN, Josef. *Scrum: průvodce agilním vývojem softwaru*. Brno: Computer Press, 2016. ISBN 978-80-251-4650-7.

[8] ROSEBROCK, Eric a Eric FILSON. *Linux, Apache, MySQL a PHP: instalace a konfigurace prostředí pro pokročilé webové aplikace*. Praha: Grada, 2005. Průvodce (Grada). ISBN 80-247-1260-1.

[9] ŠTRAUCH, Adam. *Databáze MariaDB válcuje MySQL* [online]. Root.cz, 21.12.2012. [cit. 21.3.2018]. ISSN 1212-8309. Dostupné z: https://goo.gl/oLgATQ

[10] MUTTON, Paul. *nginx, Nginx, NGiИX, or NGINX?!* [online]. Netcraft, 20.2.2018. [cit. 23.3.2018]. Dostupné z: https://goo.gl/9ThQ86

[11] METHER, Max. *MariaDB 10 vs MySQL 5.6 - A Feature Comparison Update* [online]. MariaDB, 25.3.2014. [cit. 23.3.2018]. Dostupné z: https://goo.gl/QZN1iw

[12] MONUS, Anna. *PHP 7: 10 Things You Need to Know* [online]. Hongkiat, ©2015-2018. [cit. 23.3.2018]. Dostupné z: https://www.hongkiat.com/blog/php7

[13] *jQuery* [online]. jQuery, ©2018. [cit. 24.3.2018]. Dostupné z: http://jquery.com

[14] *Documentation | Nette Framework* [online]. Nette Foundation, ©2008-2018. [cit. 24.3.2018]. Dostupné z: https://doc.nette.org/cs

[15] *Introduction* [online]. Bootstrap, ©2010-2018. [cit. 24.3.2018]. Dostupné z: https://goo.gl/mJ93Fk

[16] *Model-View-Controller Explained in C++* [online]. Technology of Computing, 24.1.2017. [cit. 24.3.2018]. Dostupné z: https://goo.gl/dfKZWB

[17] MUMBAIKAR Snehal a Puja PADIYA. *Web Services Based On SOAP and REST Principles* [online]. International Journal of Scientific and Research Publications, May 2013. [cit. 24.3.2018]. ISSN 2250-3153. Dostupné z: https://goo.gl/3RCkuX

[18] ARLOW, Jim, Ila NEUSTADT a Bogdan KISZKA. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky.* 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2008. ISBN 978-80-251-1503-9.

[19] WIEGERS, Karl Eugene a Tomáš ZNAMENÁČEK. *Požadavky na software*: [od zadání k architektuře]. Brno: Computer Press, 2008. ISBN 978-80-251-1877-1.

[20] HUDEC, Tomáš. Správa operačních systémů [online]. 2017 [cit. 31.3.2018]. Dostupné z vnitřní univerzitní sítě: http://fei-as.upceucebny.cz/usr/hudec/vyuka/sos

[21] PŘENOSIL, Vít a Tomáš HUDEC. *Rezervační systém předmětu Správa operačních systémů* [online]. 2011–2017 [cit. 31.3.2018]. Dostupné z vnitřní univerzitní sítě: http://fei-sos-rezervace.upceucebny.cz

[22] CONOLLY, Thomas, Carolyn E. BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.

[23] *Bootstrap Studio* [online]. Bootstrap Studio, ©2018. [cit. 12.5.2018]. Dostupné z: https://bootstrapstudio.io/

[24] *Latte | Nette Framework* [online]. Nette Foundation, ©2008-2018. [cit. 12.5.2018]. Dostupné z: https://latte.nette.org/cs

[25] *DataTables | Table plug-in for jQuery* [online]. SpryMedia Ltd. ©2007-2018. [cit. 12.5.2018]. Dostupné z: https://datatables.net/

[26] *Webové služby nad IS/STAG* [online]. Oddělení podpory IS, Fakulta elektrotechniky a informatiky UPa. ©2013-2018 [cit. 12.5.2018]. Dostupné z: https://stag-ws.upce.cz/ws/

[27] *Úvod do Shibboleth SSO* [online]. Oddělení ICT, Fakulta informačních technologií ČVUT. 06.11.2015. [cit. 12.5.2018]. Dostupné z: https://ict.fit.cvut.cz/~web/current/web/identity/shibboleth-sp/

[28] MUŠUTA, Jan. *Metodická pomůcka k aplikaci GDPR* [online]. Ministerstvo školství, mládeže a tělovýchovy. 08.11.2017. [cit. 12.5.2018]. Dostupné z: http://www.msmt.cz/file/44592/

Přílohy

PŘÍLOHA A – OBSAH PŘILOŽENÉHO CD-ROM

Přiložené CD obsahuje dva adresáře. V prvním je textová část diplomové práce uložená ve formátu PDF. Ve druhém je zdrojový kód implementovaného rezervačního systému včetně DDL, diagramy z použitých nástrojů formou obrázků a vytvořené projekty.

PŘÍLOHA B – MATICE RELACÍ

Obrázek níže znázorňuje matici relací, kde sloupce představují funkční požadavky pro realizaci a řádky případy užití. Účel matice relací je zmapovat, kterým případem užití je funkční požadavek realizován a zda dochází k realizaci všech požadavků či je využito všech případů užití pro realizaci požadavků.

Source \ Target	REQ002 - Import studentů z IS/STAG	REQ003 - Importování skupin a cvičících	REQ004 - Import a export témat prezentací a alternativ v CSV	REQ005 - Správa témat a alternativ pro nový AR	REQ006 - Nová upřesňující alternativa	REQ007 - Zobrazení důvodu zamítnutí rezervace	REQ008 - Priorita dle časové známky či jiná dle garanta	REQ009 - Omezení při překročení limitu rezervací	REQ010 - Schválení rezervace garantem i přes omezení	REQ011 - přidělení prezentace bez schválení rezervace	REQ012 - Hodnocení prezentace cvičícím	REQ015 - Export studentů a hodnocení prezentací v CSV	REQ016 - Export seznam prezentací včetně specifických údajů	REQ017 - Rozšíšení studenta dle oboru	REQ018 - Nabídka témat a alternativ dle oboru	REQ019 - Informace o stavu témat a studentů bez témat při přidělování	REQ020 - Automatizace výčtu chyb při hodnocení	REQ021 - Odložení prezentace o týden s odůvodněním	REQ022 - Rezervace tématu v rámci alternativy	REQ023 - Informace o zbývajícím čase do konání prezentace	REQ024 - Přístup a správa nad daty garantovi	REQ030 - Řazení zobrazených dat dle zvolených kritérií	REQ031 - Uchovávat zvolená kritéria pro řazení dat	REQ037 - Import začátek a konec semestru z IS/STAG	REQ038 - Zobrazení témat a alternativ	REQ039 - Přidělení tématu rezervace garantem studentovi	REQ040 - Hodnocení prezentace garantem pro korekci cvičícího	REQ041 - Zobrazení prezentací včetně hodnocení	REQ042 - Import a export zvolených dat z IS	REQ043 - Zobrazení stavu rezervací	REQ044 - Zobrazit údaje o uživateli	REQ045 - Zobrazit studentovi stav jeho rezervace	REQ046 - Zobrazit volná témata a alternativy	REQ047 - Student ruší rezervaci	REQ048 - Zamítnutí rezervace garantem	REQ049 - Stornování odložené prezentace
UC001 - Správa dat (CRUD)																					↑														
UC002 - Zobrazit témata a alternativy																										↑									
UC003 - Zobrazit rezervace																													↑						
UC004 - Rezervovat téma					↑			↑												↑															
UC005 - Zobrazit volná témata														↑	↑																	↑			
UC006 - Zobrazit prezentace																														↑					
UC007 - Zobrazit stav mé rezervace																															↑				
UC008 - Hodnotit prezentaci											↑							↑	↑				↑	↑				↑							↑
UC009 - Zobrazit budoucí prezentace																					↑														
UC010 - Přidělit téma					↑	↑	↑		↑	↑						↑								↑			↑						↑		
UC011 - Zobrazit mé údaje																														↑					
UC012 - Import a export dat																									↑				↑						
UC013 - Správa studentů (CRUD)	↑											↑																							
UC014 - Správa vyučujících (CRUD)		↑																																	
UC015 - Správa témat a alternativ			↑	↑																															
UC016 - Správa skupin (CRUD)		↑																																	
UC017 - Správa učeben (CRUD)		↑																																	
UC018 - Správa témat (CRUD)				↑																															
UC019 - Správa alternativ (CRUD)				↑																															
UC020 - Zrušit rezervaci																																	↑		
UC021 - Export seznamu prezentací													↑																						