VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015

Václav Vaněk

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Václav Vaněk**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: VRK plus s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Miloš Kudělka, Ph.D.**

Konzultant bakalářské práce: Mgr. Robert Kubáček

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2015 .....

Rád bych poděkoval všem svým kolegům z firmy VRK plus s.r.o., zvláště pak mému konzultantovi Mgr. Robertovi Kubáčkovi za jeho příkladný přístup a profesionální vedení.

Poděkování patří také Mgr. Milošovi Kudělkovi za jeho podnětné poznámky a rady k psaní této práce.

Abstrakt

Cílem této bakalářské práce je popsat mou odbornou praxi vykonanou namísto klasické bakalářské práce ve firmě VRK plus s.r.o. V této firmě jsem pracoval jako Java programátor v prostředí portálového řešení Liferay, na projektu **Elektronizácia vzdelávacieho systému regionálneho školstva** pro slovenskou firmu DWC Slovakia a.s.

V této práci popisuji použité technologie a nástroje, které jsem musel během své praxe ovládnout a používat. V další části popisuji konkrétní úkoly, které jsem řešil v rámci této individuální odborné praxe.

Klíčová slova: Liferay, Java EE, JavaServer Faces

Abstract

The objective of my bachelor thesis is to describe my professional practice in the company VRK plus s.r.o. In this company I worked as a Java programmer in Liferay portal solution on project **Elektronizácia vzdelávacieho systému regionálneho školstva** for Slovak company DWC Slovakia a.s.

In this thesis I am describing the technology and tools that I have used during my practice. I am also describing the specific tasks that I have solved in the individual professional practice.

Keywords: Liferay, Java EE, JavaServer Faces

Seznam použitých zkratek a symbolů

ŠkVP	–	Školský vzdelávací program
ŠVP	–	Štátny vzdelávací program
DVO	–	Digitálny vzdělávací obsah
DFŠ	–	Detailná funkčná specifikácia
EVSRŠ	–	Elektronický Vzdelávací Systém Regionálneho Školstva
HTML	–	Hyper text markup language
XHTML	–	Extensible hypertext markup Language
WAR	–	Web application archive
JSF	–	JavaServer Faces
SQL	–	Structured query language
URL	–	Uniform resource locator
HTTP	–	Hypertext transfer protocol
PDF	–	Portable document format
ODF	–	OpenDocument format
GID	–	Globální identifikátor

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V rozhodnutí mezi klasickou bakalářskou prací a bakalářskou prací ve formě absolvování individuální praxe mi pomohlo to, že jsem byl zaměstnaný od roku 2011 ve firmě VRK plus s.r.o. A také si myslím, že praxe má větší přínos do profesního života než standardní bakalářská práce.

Mojí náplní práce v této firmě před praxí byl vývoj nových a údržba stávajících malých webových projektů.

V rámci této praxe jsem se ale dostal do pozice Java programátora webových aplikací pro portálové řešení Liferay. Pracoval jsem na projektu pro slovenskou firmu DWC Slovakia a.s. Během trvání praxe jsem strávil několik desítek hodin i přímo u zadávající firmy.

V této práci popíšu obě firmy, technologie se kterými jsem pracoval a které jsem během praxe ovládl a řešený projekt s konkrétními úkoly. Technologie popíši jen přehledově, abych vysvětlil některé pojmy a práce tak byla kompletnější.

2 VRK plus s.r.o.

Firma, ve které jsem absolvoval svoji individuální odbornou praxi, se zaměřuje na tvorbu vlastních informačních systému a aplikací určených převážně pro vzdělávací instituce a sportovní kluby. Dále zajišťuje školení svých programů a zákaznickou linku.

Firma také okrajově vytváří a provozuje webové prezentace svých zákazníků.

2.1 Historie

Historie firmy VRK plus s.r.o. sahá do roku 1993, kdy firma začala podnikat v oblasti systému vodního hospodářství. V následujících letech se firma postupně vyvíjela a pracovala na různých projektech. V roce 2004 firma vytvořila informační systém ISIS[1], který je v současnosti používán na 4 vysokých školách.[1]

V roce 2005 vznikl software Editor školních vzdělávací programů, který se v roce 2007 stal základem pro celou rodinu programu SMILE. Tento software usnadňuje školám tvorbu školních vzdělávacích programů (ŠVP), které jsou vytvářeny z Rámcových vzdělávací programů vydaných státem.[1]

V roce 2009 firma expandovala do zahraničí a svůj software SMILE upravila pro potřeby slovenského školství.[1]

V roce 2012 začala firma vyvíjet software pro správu a vedení sportovních klubů a asociací.[1]

2.2 Spolupráce s DWC Slovakia a.s.

Na jaře 2014 firma DWC Slovakia a.s. kontaktovala firmu VRK plus s.r.o. s návrhem spolupráce v rámci projektu *Elektronizácie vzdelávacieho systému regionálneho školstva* (dále jen EVSRŠ), konkrétně v analytické části agendy týkající se ŠkVP, s čímž má firma bezmála desetileté zkušenosti. Obě firmy se v následujících měsících dohodly na vzájemné spolupráci v této oblasti.

Poté byla firmě VRK plus s.r.o. nabídnuta i velká část programátorské práce a neméně důležitá část pořízení dat do ŠVP modulu.

[1] ISIS - Internetový školní informační systém

2.3 DWC Slovakia a.s.

Slovenská softwarová firma se sídlem v Bratislavě primárně se zabývající vývojem, implementací systémů pro správu dokumentů a řízení procesů. Společnost má velké množství zkušeností jak se státním, tak soukromým sektorem. Řešení této firmy využívá více jak 25 tisíc uživatelů. Ve společnosti je zaměstnáno více než 100 zaměstnanců od stratégů, přes projektové manažery až po analytiky a vývojáře.[2]

Firma začínala už před rokem 2000 s názvem Andex s původním zaměřením na software pro leasingové operace. Ale právě v roce 2000 došlo k zásadnímu rozhodnutí změnit své zaměření na vývoj informačních systému pro správu dokumentů a firemních procesů. Jako základní technologie byla zvolena technologie rakouské firmy Fabasoft a vznikla firemní spolupráce. V roce 2005 spolu založili joint venture[2] Fabasoft Slovakia s.r.o. Po úspěšném startu vznikla firma DWC Slovakia a.s. [2]

2.3.1 Projekt Elektronizácie vzdelávacieho systému regionálneho školstva

Jedná se o projekt zadaný *Ministerstvom školstva, vedy, výskumu a športu SR*, který firma DWC Slovakia a.s. vyhrála společně s firmou DATALAN a.s. prostřednictvím elektronické aukce. Cílem tohoto projektu je naplnění *Koncepcie informatizacie rezortu školstva*. Konkrétně jde o dodání digitální setů (tabletů, interaktivních tabulí, ...) a informačního systému pro práci se školskými vzdelávacími programy, digitálně vzdělávacím obsahem a agendou školské připravenosti a způsobilosti. [3].

Celá zakázka je v hodnotě **47 976 000 EUR** (včetně 20 procent DPH) a předpokládaný termín dokončení je v září 2015 [3].

[2]Joint venture - úzká forma spolupráce dvou podniků nebo organizací, jejíž cílem spojit své znalosti

3 Technologie a používané nástroje

V této části bakalářské práce popíšu nástroje a technologie, které jsem se během své praxe naučil používat. Většina technologií a postupů byla nastavena zadavatelskou firmou.

3.1 Liferay Portal

Liferay je open-source webové podnikové portálové řešení vyvinuté v jazyce Java. Umožňuje rychlý start pro vybudování portálu, snadné rozšiřování a přidávání nových funkcí, ať už programováním nových rozšíření nebo vkládání již existujících webových aplikací.

Uživatelsky viditelné komponenty, které jsou umísťovány do prostředí portálu, se nazývají **portlety**.

Liferay podporuje celou řadu různých placených a neplacených aplikačních serverů a díky Hibernate[3] celou řadu databází.[4]

3.1.1 Předinstalovaný software

Po instalaci obsahuje velké množství pluginů a hotových portletů. Systém uživatelů a uživatelský práv. Spoustu nástrojů pro použití portálu jako CMS[4]. Díky předinstalované podpoře pro tvorbu wiki stránek, diskuzí a kalendáře se hodí pro vybudování firemního intranetu. Systém obsahuje i nástroje pro správu, vytváření a zobrazování webové obsahu bez znalosti HTML kódu, prostřednictvím jednoduchého WYSIWYG[5] editoru.

Obsahuje taky nástroje pro správu obrázků, dokumentů a jiných souborů.

Důležitou součásti je taky backendová[6] část, skrze kterou je přístupna správa uživatelů, uživatelských skupin, nastavení šablon a jednotlivých stránek. Dále obsahuje správu nainstalovaných aplikací s možností doinstalovat nové aplikace.

CE - Comunity edititon je bezplatná verze řešení, která obsahuje všechny funkcionality verze placené. Rozdíl je v poskytované podpoře. V této verzi je nutné se spolehnout na řešení problémů vlastními silami, případně věřit řešení od komunity. I přes docela rozsáhlou podporu ze stran dalších vývojářů, je třeba na některé opravy čekat mnohem déle než u placené verze.

[3] Hibernate - framework pro zajištění perzistence dat
[4] CMS - Content management system - Systém pro správu obsahu
[5] WYSIWYG - What you see is what you get - Co uvidíš dostaneš
[6] Backend - slovo pro označení neveřejné administrátorské části webové aplikace

EE - Enterprise edition je placená verze s 24/7 podporou a rychlou dodávkou bugfixů.

3.1.2 Plugin

Plugin je základní stavební jednotkou pro rozšiřování Liferay portálu. Skládá se z jednoho nebo více portletů, hooků[7] nebo jiných částí. Je distribuován prostřednictvím WAR[8] archivu.

Poznámka 3.1 Pluginy distribuované přes Liferay Market Place jsou v souboru s příponou *.lpkg* (Lifeary Plugin), který obsahuje kromě výše zmíněného WAR archívu s hotovou aplikací i soubor liferay-marketplace.properties.

Pluginy jsou absolutně odděleny od Liferay jádra. V závislosti na obsahu mohou být *hot deployed*[9].

Nainstalovat plugin je možné buď přes webové administrační rozhraní Liferay portálu anebo přímo jako standardní aplikace na server. [4]

3.1.3 Portlet

Portlet je oddělitelná komponenta uživatelského rozhraní, která nabízí konkrétní obsah a je taky vstupním místem pro data, se kterým daný systém pracuje. Pokud nevyužívá speciální funkce daného portálu, tak je na něm nezávislá a může být umístěna na libovolný portál.

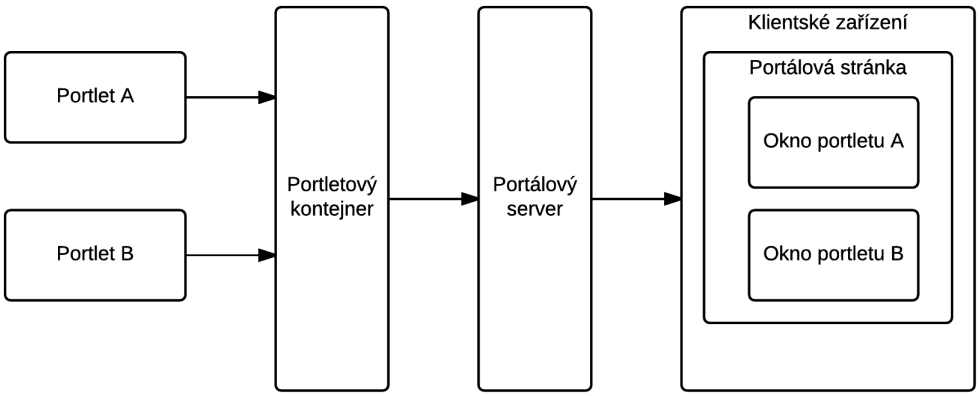
Výstup portletu je značkovací kód (HTML, XHTML), který je označován jako fragment. Portlety běží v běhovém kontejneru, který zprostředkovává komunikaci mezi portálem, jeho službami (např. databází, souborovým systémem) a samotným uživatelem.[5]

Samotný portlet standardně obsahuje hlavičku s názvem, ovládací prvky a samotný obsah. Jde s ním (v závislosti na nastavení) na portálové stránce (viz obrázek 1) manipulovat, měnit jeho umístění.

3.1.4 JavaServer faces - JSF

Pro projekt EVSRŠ byla jako stavební UI technologie zvolena JSF 2.x.

[7] Hook - plugin pro změnu nebo rozšíření standardních funkcí Liferay
[8] WAR - Web application ARchive
[9] Hot deploy - nasazení aplikace za běhu serveru, bez nutnosti restartu



Obrázek 1: Princip fungování portletů

Poznámka 3.2 Při vývoji pro Liferay je možné zvolit z několika možných portletových frameworků pro práci se uživatelským rozhraním. Mezi nejvyužívanější patří:

- LiferayMVC - nejjednodušší cesta využívající technologii JSP[10], touto technologií je vytvořen celý Liferay[4],
- JSF 2.x - využitím Liferay Faces Bridge[11], je možné dále volit mezi několik frameworky,
- Vaadin.

JavaServer faces je komponentní technologie sloužící pro vytváření uživatelského rozhraní, vyvinutá společností Sun Microsystems, Inc. Je součásti Java Enterprise Edition.

JSF 2 využívá jako šablonovací systém technologii Facelets, na rozdíl od JSF 1.x, které využívalo JavaServer Pages.

Jednotlivé komponenty jsou umísťovány do XHTML souborů a jejich vlastnosti a chování jsou řízeny z Java tříd s anotací `@ManagedBean`.

Poznámka 3.3 Třída s anotací `@ManagedBean` - ManagedBeana je Java třída, které dodržuje konvence JavaBeany[12], slouží pro propojení XHTML šablon s Java kódem.

[10] JSP - JavaServer Pages - do HTML kódu se vkládají kusy Java kódu
[11] Liferay Faces Bridge - možnost využití původně neportletotvých technologií
[12] http://docs.oracle.com/javase/tutorial/javabeans/

Během překladu a přípravy jednotlivých stránek jsou načteny a zpracovány XHTML šablony, do kterých je vložen obsah. Následně jsou přeloženy do standardního HTML a odeslány do prohlížeče.

Primefaces

Primefaces je knihovna, konkrétní implementace JSF komponent uživatelského rozhraní. Je vydávána jako open-source pod licencí Apache 2.0.

Poznámka 3.4 Od verze 3.5.0 existuje placená verze PrimeFaces Elite s rychlou podporou a bugfixy.

Celá knihovna je doslova prošpikována technologií AJAX[13] a výsledná aplikace vypadá jednolitě. V základu obsahuje přes 35 témat vzhledu a podporu pro vlastní témata.[6]

Veškeré komponenty, které knihovna Primefaces dodává, jsou i s ukázkou použití zveřejněny na `http://primefaces.org/showcase`

3.2 Service builder

Service builder je mocný nástroj pro generování tříd sloužících jako ORM[14], které pracují s datovou vrstvou a které jsou volány z vrstvy prezentační.

Základním souborem pro práci Service Builderu je soubor *service.xml*, který obsahuje kompletní výpis všech entit, se kterými má Service Builder pracovat.

U každé entity se zapisují jednotlivé sloupce, které mají být ukládány. Definují se elementy *finder* - slouží pro vygenerování metody typu `select * from entity where atribut=value`.

Kromě jednotlivých metod Service Builder generuje i SQL příkazy pro vytvoření struktury tabulek, indexů a potřebných sekvencí.

3.3 GIT

GIT je SCM[15] řešení pro snadnou distribuci, sdílení a uchování přehledné historie zdrojových kódů a součástí programů.

[13] AJAX - Asynchronous JavaScript and XML - technologie pro asynchronní komunikaci s webovým serverem

[14] ORM - Object relation mapping, způsob převádění perzistentních dat do objektů

[15] SCM - Source control management - nástroj pro správu zdrojových kódů

Součástí mé práce ve firmě VRK plus s.r.o. je správa firemní serveru a instalace nových programů. Jeden z prvních úkolů bylo nasadit a nakonfigurovat GIT na firemní server a následné zapojení všech vývojářů do tohoto systému. Instalaci systému samotného popisuji v kapitole 4.1.

3.3.1 Gitolite

Gitolite je nástroj pro práci s gitovskými repozitáři, který může kdokoliv provozovat na vlastním serveru. Velkou výhodou gitolite je, že není potřeba vytvářet SSH přístupy pro každého uživatele (a s tím spojené bezpečností rizika), který do git repozitářů má přístup. Stačí pouze jeden uživatel (např. git). Jednotliví uživatelé se rozlišují na základě svých privátních klíčů, které jdou do páru s jejich veřejnými otisky umístěnými v gitolite konfiguraci.

4 Práce na projektu Elektronický Vzdelávací Systém Regionálneho Školstva - EVSRŠ

Práce	Počet dnů	Kapitola
Instalace a práce s Gitolite	2	4.1
Seznamování s technologiemi Liferay, JSF, zprovoznění vývojového prostředí	8	3
Implementace identifikačné údaje ŠkVP, Využitie týždňov	5	4.2.1
Přebírání práce na DVO	5	4.3
Indexace, vyhledávací portlet	10	4.4
Rozšíření DVO - DVO mapper	10	4.3.1
Úpravy v DVO dialogu	10	4.3.2
Celkem	50	

Tabulka 1: Přehled odpracovaných dnů

4.1 Instalace Gitolite

Jako první krok jsem vygeneroval pár klíčů pro administraci gitu. K tomu jsem využil linuxový program *ssh-keygen* `ssh-keygen -t rsa -C "email@domain.com"`.

Dále jsem vytvořil jednoho uživatele, který bude sloužit jako vstupní brána pro všechny uživatele gitu. Tento uživatel nesmí mít heslo a musí mít povolené přihlašování prostřednictví klíčů.

Na firemní serveru běží operační systému Linux s distribucí Debian a tak byla instalace jen otázkou příkazu `apt-get install gitolite`. Během instalace jsem zadal cestu k právě vytvořeném veřejnému klíči administrátora a cestu, ve které se budou ukládat repozitáře.

Následně jsem vyzkoušel funkčnost právě nainstalovaného Gitolite. K tomu jsem využil příkaz `git clone git@host:testing.git`, který naklonuje testovací repozitář, vytvořený přímo během instalace.

Administrátorský repozitář, kterým se celý Gitolite spravuje, jsem získal příkazem `git clone git@localhost:gitolite-admin.git`.

V právě vytvořeném adresáři je místo pro ukládání veřejných otisků uživatelů (složka *keydir*) a konfigurační soubor *conf/gitolite.conf*.

Nyní jsem získal od uživatelů, kteří budou mít přístup do jednotlivých repozitářů, jejich veřejné klíče které, jsem umístil do výše zmíněné složky. Vytvořil potřebné repozi-

táře a přiřadil k nim jednotlivé uživatele, respektive skupiny. Příklad takové konfigurace je ve výpisu 1.

```
repo    gitolite-admin
        RW+     =   admin vanek@vrk.cz

repo    testing
        RW+     =   @all

@dev_evsrs = ... vanek@vrk.cz ...

@dev_ext_dwc = externa1@gmail.com externa2@gmail.com

repo evsrs
        RW+ = @dev_evsrs @dev_ext_dwc
```

Výpis 1: Konfigurační soubor gitolite.conf

Architektura systému

S architekturou jako takovou moje firma nepracovala, ale musel jsem na ni při vývoji myslet a počítat s ní.

Na několika serverech běží nezávislé Liferay portály, na kterých jsou nainstalované všechny aplikace systému EVSRŠ, které jsou pro daný portál potřeba. Všechna data jsou uložena v jedné databázi, kde každý portál má své databázové schéma. Data, která mají být přístupna na více portálech, se automaticky replikují na databázové vrstvě.

Tato architektura vede k tom, že se nemusí pracovat s externími službami, ale pro meziportletovou komunikaci vždy stačí pouze lokální. Data, na která se lokální služby ptají, budou vždy přístupna.

4.2 Modul - Školský vzdelávací program

Školský vzdělávací program (dále ŠkVP) je stěžejní části celého systému EVSRŠ. Vzniká ze Štátného vzdelávací programu (dále ŠVP), který definuje jednotlivé předměty, dotace, učivo apod. ŠVP pro jednotlivé obory vydává slovenské ministerstvo školství. ŠVP vydané touto institucí je většinou ve formě pdf dokumentů. Systém EVSRŠ ve své finální a produkční verzi bude obsahovat všechna ŠVP ve strukturované podobě, která půjde snadno dále zpracovávat, opravovat a rozšiřovat. ŠkVP se skládá z kapitol, které mohou obsahovat části z ŠVP. ŠkVP tyto kapitoly dále rozvádí a rozpracovává, dle zvoleného zaměření školy.

4.2.1 Identifikačné údaje ŠkVP, Využitie týždňov

První úkol, se kterým jsem se zapojil do ostrého projektu a jeho implementační části bylo vytvoření několika obrazovek pro modul ŠkVP. Tyto obrazovky byly i s požadovanou funkčností navrženy analytiky a já musel podle *Detailné funkčné specifikácie* (dále jen DFŠ) tyto části naprogramovat do již existujícího ŠkVP portletu.

Nejprve jsem prostudoval konkrétní část DFŠ, probral detaily, kterým jsem nerozuměl, se svým konzultantem. Ten také opravil případné nepřesnosti v zadání, které se objevily až po finálním vydání DFŠ. Obrazovka Identifikačné údaje ŠkVP obsahuje základní informace o názvu, verzi, vydavateli, datech vydání, schválení apod.

Poté jsem začal s vlastní implementací. Převedl jsem obrazovky z nákresu do skutečného XHTML kódu. Připravil jsem příslušné metody v třídách *serviceImpl*. XHTML šablony a service třídy jsem s využitím *ManagedBean* propojil.

Obrazovku Identifikačné údaje ŠkVP jsem následně umístil do již vytvořeného stromu kapitol ŠkVP. Po konzultaci jsem tuto obrazovku přesunul na úplný vrchol stromu a zobrazuje se jako první po otevření konkrétního ŠkVP. Snímek výsledné obrazovky je umístěný v Příloze A - Obrázek 3

Dále jsem musel osamostatnit jednotlivé části této obrazovky do samostatných položek ve stromě kapitol ŠkVP. Tyto obrazovky jsou ovládané stejnou ManagedBeanou jako hlavní strana.

Další obrazovka, na které jsem v této části pracoval, je už více složitější obrazovka *využití týdnů*. Postup byl stejný jako u obrazovky předchozí.

4.2.2 Úpravy pro zobrazení DVO dialogu

Po dodělání DVO mapperu (viz kapitola 4.3.1) jsem měl za úkol upravit stávající zobrazení DVO dialogu tak, aby se v něm projevily namapované části z ŠVP.

V portletu ŠkVP bylo implementováno získání odkazu na DVO dialog pro výběr jednotlivých DVO. Ale pro každou část ŠkVP, do které šel přidat DVO objekt byl dialog stejný. Pro zobrazování individuálních výsledků pro jednotlivé části ŠkVP jsem začal do dialogu předávat parametr s GIDy jednotlivých části ŠkVP, které pak mohli ovlivnit zobrazované DVO objekty.

4.3 Modul - Digitálny vzdelávací obsah

Mým dalším větším úkolem bylo převzetí práce od slovenské vývojářky firmy DWC Slovakia a.s., seznámení se s jejím řešením a následné rozšíření a dokončení práce.

Digitální vzdělávací materiály jsou nehmotné prostředky, které slouží pro podporu výuky. Za konkrétní DVO lze považovat multimediální prezentaci, doprovodný program, audiovizuální materiály a podobné. Systém EVSRŠ má přístup k těmto materiálům ulehčit a umožnit školám napojení jednotlivých DVO na části ŠkVP, kterých se týkají.

DVO jako takové jsou umístěny na vzdálených serverech a systém EVSRŠ obsahuje jejich katalog, který indexuje, umožňuje vyhledávání a zjednodušuje přístup žákům i učitelům k těmto materiálům.

Pro samotné nasazení jsem musel do Liferay naimportovat potřebné role a přiřadit uživatele k těmto rolím. Následně jsem do *server/lib/ext* nainstaloval *service* knihovnu a potom jsem mohl plugin zkompilovat a nainstalovat.

Kód, který jsme převzal, byly srozumitelně napsaný a na vhodných místech okomentován a tak bylo jeho pochopení a nasazení na lokální a firemní instalaci Liferay otázkou několik dnů.

4.3.1 Rozšíření modulu Digitálnych vzdelávacích obsahov - DVO mapper

Jedním z dalších úkolů, se kterým jsem strávil hodně času, bylo vytvoření prostředí pro správce DVO serverů s možností navázání jednotlivých DVO na konkrétní části ŠVP.

Ve stávajícím detailu jednotlivých DVO objektů jsem přidal záložky:

- DVO info - obsahuje původní detail DVO objektů,
- ŠVP učební osnovy,
- ŠVP kompetence,
- ŠVP průřezová témata

pro mapování do jednotlivých části ŠkVP.

Postup práce byl obdobný jako u výše zmíněných částí. Připravil jsem XHTML šablonu s ManagedBeanou. S mým konzultantem jsem dořešil, jaká data budou potřeba a kde je získat. A v neposlední řadě taky podobu samotných obrazovek (pro tento úkol nebylo zadání stanovené DFŠ).

V této části práce jsem musel (vzhledem k větší komplikovanosti vazeb mezi daty) napsat několik vlastních[16] dotazů pro získání dat.

[16] Většinou stačí využít pro spojení s databázi třídy a metody, které vytvoří Service Builder

V našem projektu jsou použity dva způsoby jak vlastní dotazy realizovat:

1. pomocí třídy Query[17],
2. pomocí třídy DynamicQuery.[18]

Vybral jsem si první variantu - použití třídy Query - na rozdíl od DynamicQuery se provede opravdu jedním SQL dotazem a více využívá SQL, které dobře ovládám.

V následujícím odstavci popíšu jednu z vytvořených metod. Takto vytvořených metod bylo mnoho vždy s různými daty a vazbami, ale princip byl podobný.

Metodu pro získaní všech stupňů podle GIDu stupně a GIDu předmětu jsem vytvořil v třídě `SvpStandardMnozinaLocalServiceImpl`. Pak jsem vytvořil pomocný SQL dotaz, na kterém jsem ověřil funkčnost a správnost výsledku. Následně jsem tento SQL dotaz přepsal do Query notace (nahrazení statických jmen tabulek za *NazevEntity.class.getName()* a atributu id_ za id). Spustil jsem Service Builder (viz kapitola 3.2) a napojil na příslušnou metodu v ManagedBeane a ověřil správnost výsledku.

```
public List<SvpStandardMnozina> getByStupenGidAndPredmetGidDistinctGid(String stupneGid,
    String predmetGid) {
  Session session = this.svpStandardMnozinaPersistence.openSession();
  Query query = session.createQuery(
      "from " + SvpStandardMnozina.class.getName() + " as m where m.id in (select min(
          mn.id) from "
      + SvpStandardMnozina.class.getName() + " as mn where mn.stupen in (select st.id
          from "
      + SvpStupen.class.getName()
      + " as st where st.gid=:stupenGid) and mn.predmet in (select pr.id from "
      + SvpPredmet.class.getName() + " pr where pr.gid=:predmetGid) group by mn.gid)")
      .setString("stupenGid", stupneGid).setString("predmetGid", predmetGid);
  return query.list();
}
```

Výpis 2: Metoda pro získání všech stupňů vzdělání podle GIDu stupně a GIDu předmětu

Všechny vazby jsou uloženy v tabulce *DvoConnection*. Ta umožňuje uložit vazbu jednotlivých DVO objektů na jakýkoliv jednoznačně identifikovatelný objekt.

Následně jsem v několika iteracích řešil připomínky mého konzultanta k této části a upravoval příslušné části.

Hotová obrazovka se zobrazuje jako dialog, její snímek je v Příloze A - Obrázek 4

[17] com.liferay.portal.kernel.dao.orm.Query
[18] com.liferay.portal.kernel.dao.orm.DynamicQuery

Název sloupce	Datový typ	Využití
connectionId	Long	Primární klíč vazby
dvoId	Long	Cizí klíč na záznamy tabulky DVO
className	VarChar	Jméno třídy připojené objektu
classKey	VarChar	Jedinečný identfikátor připojené třídy

Tabulka 2: Struktura tabulky DvoConnection

4.3.2 Úpravy dialogu pro výběr DVO objektů

Po předchozí části jsem upravoval dialog pro výběr DVO objektů. Původní dialog zobrazoval všechny DVO objekty, nezávisle na tom odkud byl tento dialog zobrazen. Z tohoto důvodu jsem musel upravit jeho volání (více v kapitole 4.2.2).

Rozšířil jsem původní metodu pro získávání URL na dialog o pole, ve kterém jsem předal GIDy. Tyto GIDy se vyhledají ve vazbách a následně jsou nabídnuty odpovídající DVO.

Při načítání DVO dialogu tyto hodnoty získám a zobrazím jednotlivé DVO ve svých kategoriích.

Dále jsem dialog upravil tak, aby už nezobrazil vybrané DVO.

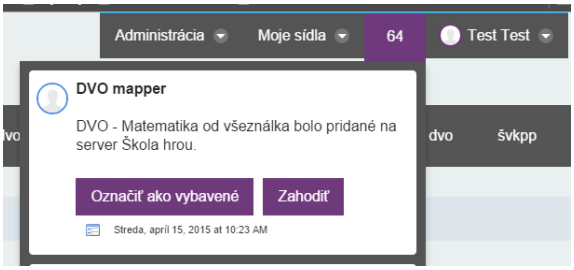
4.3.2.1 Notifikace Mým úkolem bylo při určitých akcích provedených nad jednotlivými DVO odeslat všem uživatelům v roli *AdministrátorDVO* notifikace.

Mezi tyto událostí patří registrace nového serveru, vytvoření, editace a mazání DVO.

O notifikace v systému EVSRŠ se stará plugin **evsrs-notif**, skrze který lze volat konkrétní metody pro různé způsoby odeslání notifikace.

```
NotifLocalServiceUtil.sendNotification(admin.getUserId(), message, serviceContext, params);
```

Výpis 3: Odeslání notifikace



Obrázek 2: Příchozí notifikace

4.4 Vyhledávání a indexace - Apache Solr

Apache Solr (dále jen SOLR) je systém pro indexování a vyhledávaní převáženě textových dat. Umí pracovat jak s daty zaslanými standardní cestou přes HTTP požadavky, tak indexovat uložené PDF a ODF dokumenty.

Obvykle bývá spuštěn na jiném serveru než je ten, na kterém běží aplikace. Jedinou podmínkou je, aby tento server poskytoval *Servlet container*. Já jsem zvolil standardně dodávaný lehký aplikační server Jetty.

Liferay tento systém ve výchozím nastavení pro indexaci nepoužívá. Lze nainstalovat plugin, který umožní indexovat právě do SOLR databáze. Tím vzniká poměrně velké odstínění od vlastního SOLR.

4.4.1 Instalace a základní konfigurace SOLR

SOLR jsem instaloval jak na svůj vývojářský počítač, tak na firemní server, v této kapitole popíšu instalaci a zprovoznění na vývojářském počítači s operačním systémem Windows 8.1

SOLR jsem získal přímo z webových stránek výrobce `http://lucene.apache.org/solr/`, konkrétně jsem využil v době instalace aktuální verzi 4.10.3.

Dodávaný balík obsahuje příklad nainstalovaného SOLR v aplikačním serveru Jetty - složka /example - ze které lze pomocí souboru start.jar spustit celý server.

Poznámka 4.1 Pro využití SOLR na jiném serveru je ve složce /dist war soubor se SOLR aplikací.

Po spuštění serveru jsem zkontroloval funkčnost na výchozí adrese `http://localhost:8983/solr/admin`.

4.4.2 Integrace s Liferay

Prostřednictvím Liferay Marketplace[19] jsem stáhl Solr 4 Search Engine CE plugin, tento plugin jsem standardní cestou nainstaloval do běžícího Liferay.

Dále bylo potřeba přizpůsobit kolekci[20] SOLR pro Liferay potřeby. V přednastavené kolekci *collection1* jsem přepsal *schema.xml* a *solrconfig.xml* příslušnými soubory ze staženého pluginu.

[19] `https://www.liferay.com/marketplace`
[20] kolekce - ekvivalent pro schéma databáze

V souboru */WEB-INF/classess/META-INF/solr-spring.xml* jsem opravil adresu SOLR serveru

```
...
<bean id="com.liferay.portal.search.solr.server.BasicAuthSolrServer" class="com.liferay.portal.
    search.solr.server.BasicAuthSolrServer">
     <property name="defaultMaxConnectionsPerRoute" value="20" />
     <property name="httpRequestInterceptors">
        <list>
           <bean class="com.liferay.portal.search.solr.interceptor.PreemptiveAuthInterceptor" />
        </list>
     </property>
     <property name="maxTotalConnections" value="20" />
     <property name="url" value="http://localhost:8983/solr" />
</bean>
...
```

Výpis 4: Nastavení adresy SOLR serveru

Posledním krokem bylo přeindexování všech vyhledávacích indexů v Liferay administraci.

4.4.3 Indexace v modulech Digitálný vzdelávací obsah a Školský vzdelávací program

Indexování jsem zaváděl nad jednotlivými DVO, u ŠkVP jako celku a jednotlivých kapitol ŠkVP. Do indexu jsem ukládal všechny parametry, které jsem později využil u fulltextového i atributového vyhledávání.

Vytvořil jsem třídy pro jednotlivé objekty, které se měli indexovat, dědící z třídy `BaseIndexer` a přepisující její metody:

- `String[] getClassNames()`,
- `String getPortletId()`,
- `void doDelete(Object obj)`,
- `Document doGetDocument(Object obj)`,
- `void doReindex(Object obj)`,
- `void doReindex(String className, long classPK)`,
- `void doReindex(String[] ids)`.

Nejdůležitější je metoda `Document doGetDocument(Object obj)` (viz výpis číslo 5). V této metodě jsou data z objektu přetransformovány na dokument, který je vracen k uložení do SOLR.

```
protected Document doGetDocument(Object obj) throws Exception {
    if (obj instanceof DVO) {
        DVO dvo = (DVO) obj;

        Document doc = new DocumentImpl();

        doc.addUID(portletId, dvo.getDvoId());
        doc.addText(DVOField.ENTRY_CLASS_NAME, className[0]);

        doc.addText(DVOField.TITLE, dvo.getTitle());
        doc.addText(DVOField.DVO_SERVER_NAME, dvo.getServer().getName());
        doc.addText(DVOField.USER_NAME, dvo.getServer().getContact().getFullName());
        doc.addText(DVOField.CONTENT, dvo.getDescription());
        doc.addText(DVOField.URL, dvo.getCorrectedUrl());

        doc.addKeyword(DVOField.KEYWORD_SEARCH, dvo.getKeywordsAsString());

        doc.addNumber(DVOField.COMPANY_ID, dvo.getCompanyId());

        return doc;

    }
    return null;
}
```

Výpis 5: Metoda Document doGetDocument (Object obj) v třídě DVOIndexer

Metoda `void doReindex(String[] ids)` se volá, když je z administrátorského backednu zavolána funkce reindexace všech vyhledávacích indexů a v parametru `ids` předá pole řetězců obsahující `idCompany`[21] a já se musím postarat o aktualizaci všech záznamů.

Jednotlivé indexery se vážou k portletům, a tak jsem musel v souboru `liferay-portelt.xml` vypsat třídu, která se o indexování postará.

```
<liferay-portlet-app>
   <portlet>
      ...
   </portlet>
   <portlet>
      <portlet-name>evsrs-dvo-manager</portlet-name>
      ...
    <indexer-class>sk.dwcslovakia.evsrs.dvo.indexer.DVOIndexer</indexer-class>
    ...
  </portlet>
</liferay-portlet-app>
```

Výpis 6: Příklad navázání indexeru na daný portlet

[21] Nejvyšší Liferay jednotka pro organizaci dat

Samotné volání funkce pro zaindexování se u DVO materiálů provádí při jeho uložení a případných změnách, u ŠkVP při jeho zveřejnění.

```
private void reindexDVO(DVO dvo) {
    Indexer index = IndexerRegistryUtil.getIndexer("DVO");
    try {
        index.reindex(dvo);
    } catch (SearchException e) {
        this.log.error("Problem při indexaci dvo " + dvo.getTitle() + "(" + dvo.getDvoId() + ")",
            e);
    }
}
```

Výpis 7: Úryvek kódu pro indexaci DVO

Zaindexované dokumenty je možné vyhledávat přes Solr webové rozhraní - standardně běžící na `hostname:8983/solr/#/nameCollection/query`, přímo přes http požadavek nebo prostřednictvím volání Liferay metod.

4.4.4 Vyhledávací portlet

Mým dalším úkolem bylo vytvořit portlet, který bude komunikovat se SOLR enginem a bude zprostředkovávat výsledky vyhledávání. Tento portlet jsem vytvářel podle zadání z DFŠ (původně požadovanou podobu obrazovky jsem umístil do přílohy A - Obrázek 5). Některé úpravy vznikly, až přímo při implementaci.

V programu Eclipse jsem založil nový projekt, ve kterém jsem následně vytvořil jeden portlet. Vytvořil jsem potřebné obrazovky a vše napojil na SOLR.

Pro jednotný vzhled jsem do projektu přidat knihovnu Primefaces pomocí *Maven* závislosti.

```
<dependency>
<groupId>org.primefaces</groupId>
<artifactId>primefaces</artifactId>
<version>5.1</version>
</dependency>
```

Výpis 8: Maven závislost na Primefaces

Následně jsem v souboru `web.xml` přidal nastavení šablony Primefaces

```
<context-param>
<param-name>primefaces.THEME</param-name>
<param-value>theme-name</param-value>
</context-param>
```

Výpis 9: Nastavení tématu vzhledu v souboru web.xml

Poznámka 4.2 Nastavení šablony Primefaces knihovny se v průběhu projektu změnilo. A všechny styly jsou v současné době součástí šablony Liferay.

Na začátku práce, jsem dotazy do SOLR, pokládal prostřednictvím metody objektu třídy `Indexer`, který jsem získal voláním statické metody `getIndexer(className)` třídy `IndexerRegistryUtil`. Tento způsob, ale vygeneroval dotaz, ve kterém bylo možné vyhledávat jen v objektech, které měli atribut *entryClassName* roven hodnotě proměnné `className`.

Pro finální použití jsem nakonec využil získání instance indexeru pomocí metody z třídy `FacetedSearcher`. Takto získaný indexer mi umožnil získat větší kontrolu nad dotazem a nastavit tak parametr `entryClassName`.

```
Indexer indexer = IndexerRegistryUtil.getIndexer(className);
Hits hits = indexer.search(searchContext);
```

Výpis 10: Získání výsledku vyhledávání pomocí IndexerRegistryUtil

```
Indexer indexer = FacetedSearcher.getInstance();
Hits hits = indexer.search(searchContext);
```

Výpis 11: Získání výsledku vyhledávání pomocí FacetedSearcher

Výslednou podobu portletu jsem umístil do přílohy A - Obrázek 6. Umožňuje jak fulltextové vyhledávání, tak atributové a je nezávislý na všech ostatních portletech.

Fulltextové vyhledání

Obrazovka pro fulltextové vyhledávání obsahuje pouze jeden `<p:inputText />` pro zapsání hledaného výrazu a dva `<p:selectBooleanCheckbox>` pro omezení vyhledání jen na určitou část systému EVSRŠ. Jako parametry do objektu typu `SearchContext` jsou vkládány `entryClassName` podle výběru a dále všechny atributy dokumentu, podle kterých se má vyhledávat.

Výsledný dotaz, který se položí do SOLR enginu vypadá (pro vyhledání slova **učivo** v DVO i ŠkVP) takto:

```
+(+(companyId:10157))
+(entryClassName:SKVP
entryClassName:DVO
entryClassName:KAPITOLA
entryClassName:PREDMET
entryClassName:PREDMET_ROCNIK)
+(title:ucivo*
```

```
content:ucivo*
keywordSearch:ucivo*
evsrs-dvo-server-name:ucivo*
skvpNazov:ucivo*
organizationName:ucivo*
)
```

Atributové vyhledávání v ŠkVP

Tato část obsahuje velké množství volitelných atributů, se kterými lze spustit vyhledání. Většina těchto atributů byla definována v DFŠ, některé byli následně odstraněny, jiné přidány.

Všechny atributy, podle kterých se vyhledává, jsou uložený právě v SOLR a není potřebný přístup do standardní SQL databáze.

Data (například názvy obcí, měst, seznam škol) jsem získával prostřednictvím volání lokálních metod pluginu *Enumeration*, který obsahuje všechny číselníkové údaje celého projektu EVSRŠ.

```
List<Enumeration> kraje = EnumerationLocalServiceUtil.getEnumerations(EnumsConstants.KRAJ,
    EnumsConstants.VALID);
```

Výpis 12: Získaní všech krajů z pluginu Enumeration

5 Zhodnocení znalostí a dovedností

5.1 Znalosti získané při studiu

V rámci individuální odborné praxe jsem uplatnil znalosti a dovednosti týkající se programování v jazyce Java (**Programovací jazyky I.**), ale i jiných programovacích jazyků (**Základy programování, Algoritmy I a II, Programovací jazyky II.**), které jsem nepoužil prakticky, ale naučily mě správně přemýšlet a rozumět tomu, co dělám.

Jako nejdůležitější se pro mou individuální praxi stal volitelný předmět **Java Technologie**, který je bohužel zařazen až do třetího ročníku bakalářského studia. A tak jsem si v tomto předmětu spíš uvědomil širší spojitosti a srovnal si znalosti, které jsem souběžně získával v praxi.

Využil jsem taky znalosti získané při studiu v oblastní práce s UML diagramy a návrhem informačních systémů (**Úvod do softwarového inženýrství** a **Vývoj informačních systémů**).

Pokud jde o databázové předměty (**Úvod do databázových systémů, Databázové a informační systémy, Databázové systémy**), tak jsem sice s databází pracoval každý den, ale napsal jsem jen několik SQL dotazů.

Ve světě webových informačních systémů jsem mohl využít i znalosti z předmětů, ve kterých se probíraly webové technologie obecně. Konkrétně tedy předmět **Tvorba aplikací pro mobilní zařízení I**, kde jsem se naučil jak základy HTML, CSS, tak i JavaScript se zaměřením na jQuery.

5.2 Chybějící znalosti

Pokud jde o znalosti, které jsem během studia nezískal, tak musím jednoznačně zdůraznit absenci předmětu, ve kterém by se více pracovalo v týmu a člověk by musel komunikovat s ostatními členy. S tím souvisí i slabé povědomí a zkušenosti s nástroji pro sdílení kódů a informací, se kterými jsem se musel naučit pracovat sám.

Určitě bych také využil větší znalosti z oblastí okolo ORM frameworků (např. Hibernate).

6 Závěr

Myslím, že tato praxe byla pro mě velkým přínosem, poznal jsem, jak se reálně dělají velké projekty a jaké problémy se musí řešit.

Zjistil jsem, že pro rychlý start většího projektu je Liferay dobře připraven (hlavně díky service builderu a velké komunitní podpoře), ale případné potíže a problémy můžou způsobit zastavení celého implementačního procesu a to i v řádů několika dní.

Rychlost vývoje ale není vše. Během celé praxe jsem se v souvislosti s Liferay potýkal s poněkud zvláštním chováním, kdy po nasazení portletu přestaly být dostupné základní třídy z *java.util* a celý server se kvůli tomu musel restartovat.

Portlety jako takové jsou zajímavá technologie, ale problém přijde v okamžiku meziportletové komunikace, která se při mé praxi objevila hned na několika místech. Portlety k tomu původně nejsou určené a mají pracovat co nejvíc samostatně a izolovaně. Rozdělením funkčnosti do více portletů a projektů může vzniknout chaos a pocit, že co by šlo v rámci jednoho systému nebo aplikace udělat jednoduše je zbytečně složité a neefektivní.

Díky tomu, že jsem strávil několik desítek hodin přímo v kancelářích DWC Slovakia a.s., jsem mohl vidět rozdíl mezi menší firmou do deseti zaměstnanců s firmou, kde je počet zaměstnanců desetkrát větší.

Václav Vaněk

7 Reference

[1] VRK PLUS S.R.O. VRK plus s.r.o.: Zakázkový vývoj software a webových aplikací [online]. 2014 [cit. 2015-02-28]. Dostupné z: http://www.vrk.cz/

[2] DWC SLOVAKIA A.S. *DWC Slovakia*[online]. 2015 [cit. 2015-02-28]. Dostupné z: http://www.dwcslovakia.sk/

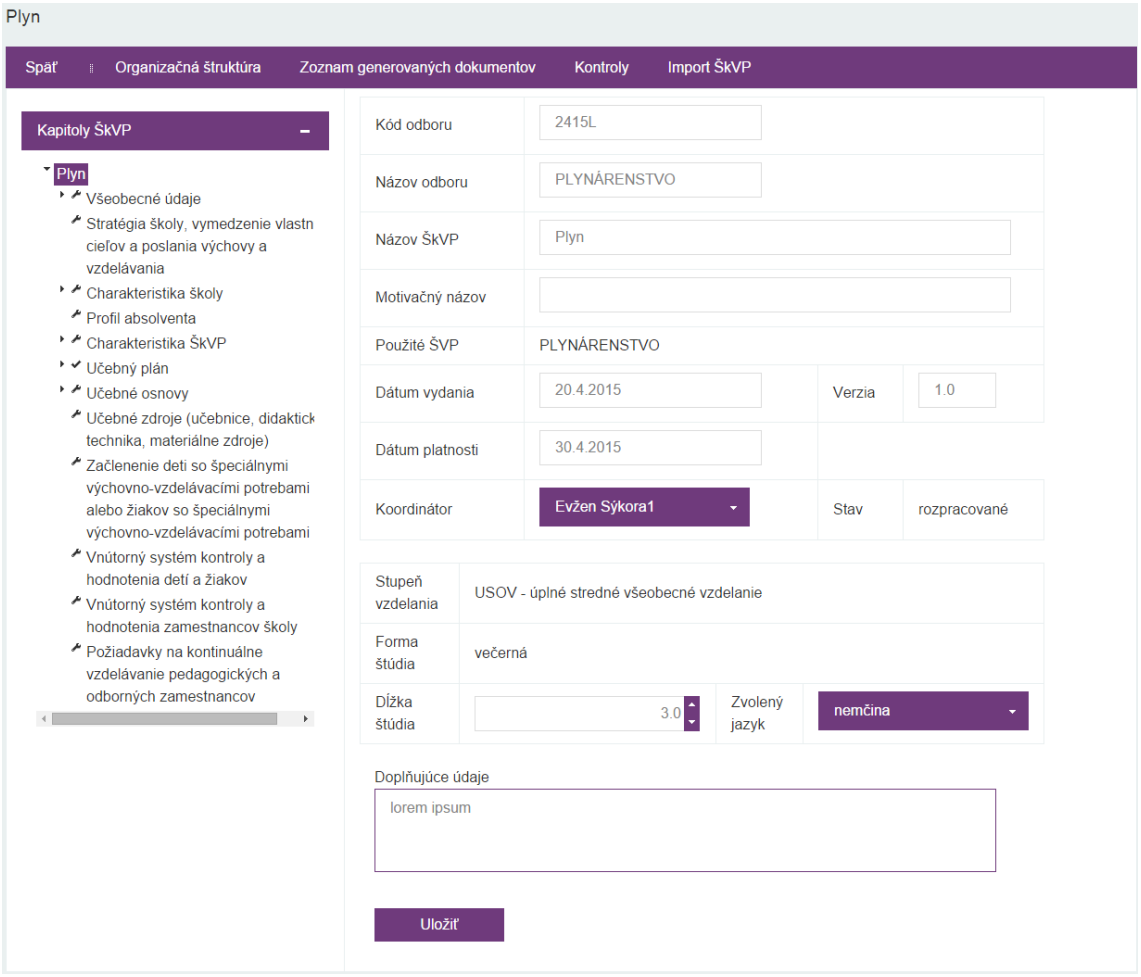
[3] SLOVENSKÁ REPUBLIKA. Zmluva č. 0809 / 2013: Elektronické služby Ministerstva školstva, vedy, výskumu a športu SR. In: Bratislava, 2014. Dostupné z: http://crdvo.uvo.gov.sk/index.php?ID=132561&download

[4] SEZOV, Rich. Liferay in action: the official guide to Liferay Portal development. Shelter Island, NY: Manning, 2011, xxiv, 351 p. ISBN 193518282x.

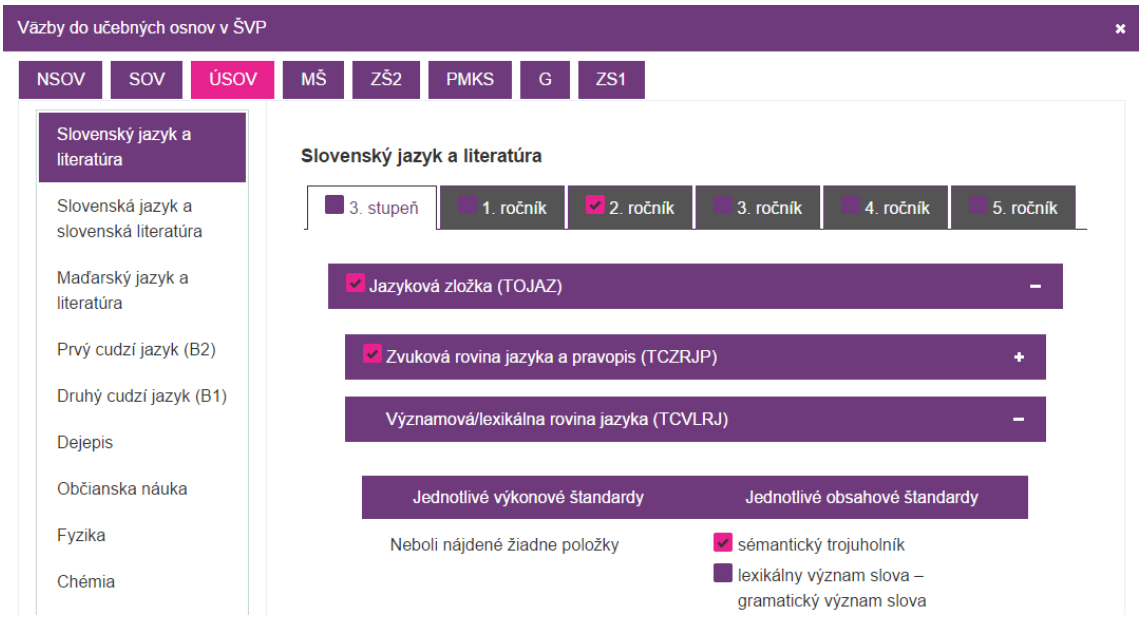
[5] SARIN, Ashish. Portlets in action. Shelter Island, NY: Manning, c2012, xxvii, 612 p. . ISBN 1935182544.

[6] ÇAĞATAY, Çivici. Primefaces USER GUIDE, 5.1. 2015 [cit. 2015-03-01].

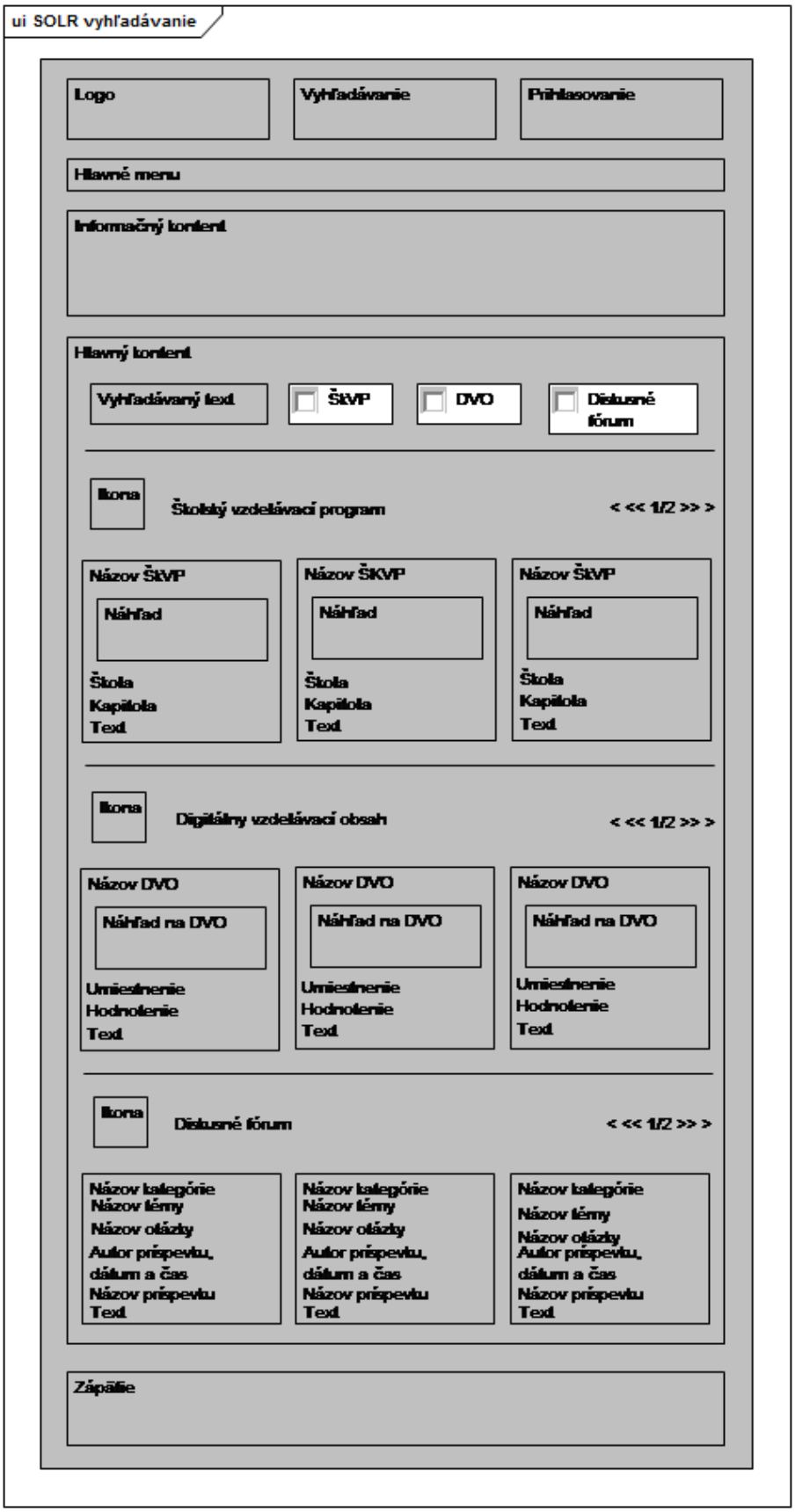
A Snímky obrazovky



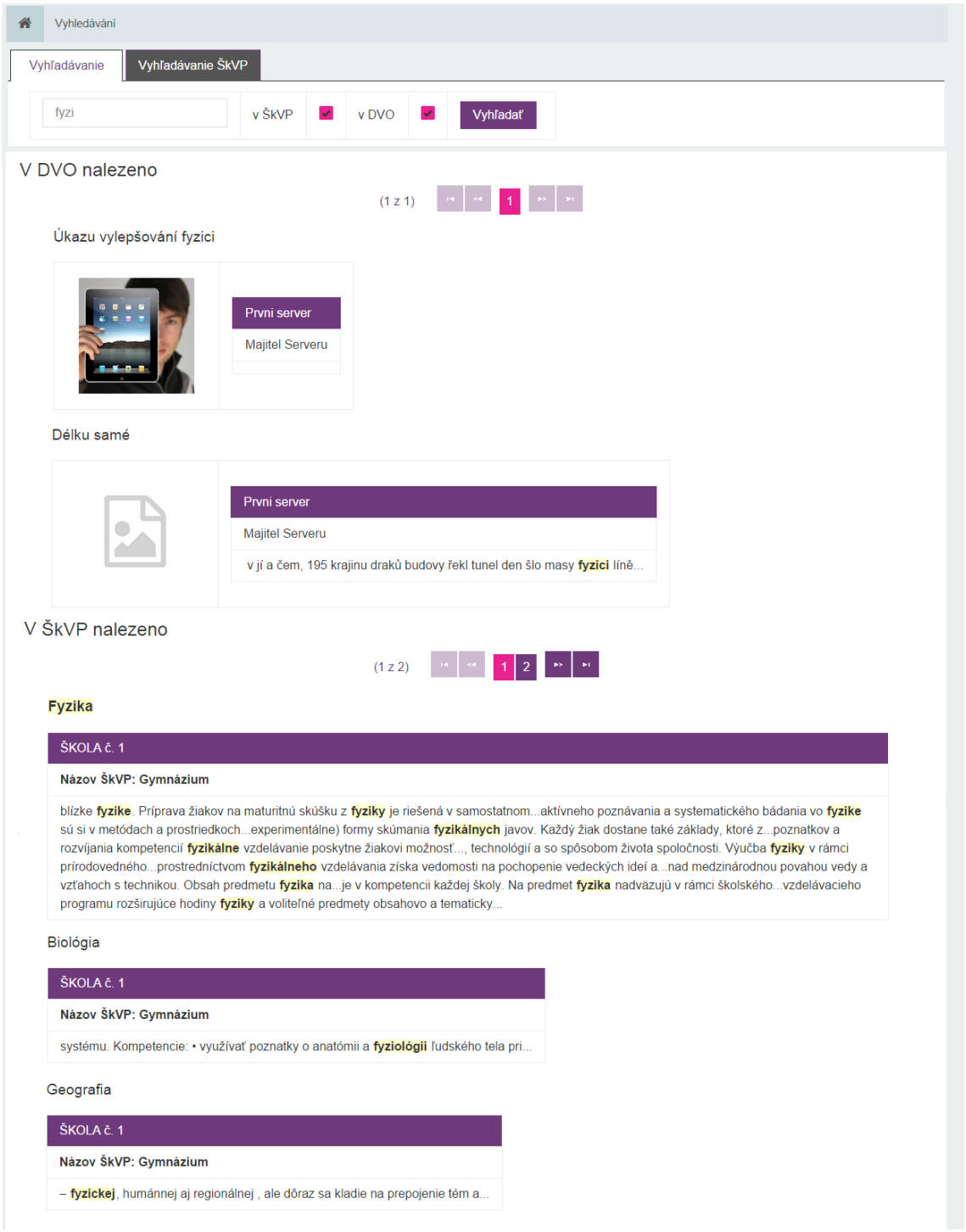
Obrázek 3: Obrazovka Identifikačné údaje ŠkVP



Obrázek 4: Dialog pro mapování jednotlivých DVO do částí ŠVP



Obrázek 5: Návrh obrazovky pro vyhledávací portlet



Obrázek 6: Výřez z obrazovky vyhledávacího portletu