Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Specializované CMS pro jazykové vzdělávání

Plzeň, 2012 Martin Průša

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 19.7.2012, Martin Průša

Abstract

Specialized CMS for Language Curricula

This thesis introduces revers to the CMS system created on the Java EE platform and system customization capabilities for saving language course. The thesis analyzes the needs of language school for keeping content of language courses for frontal and e-learning teaching. In the thesis, the design of CMS system that will be used for keeping language content according to the analysis is described in words and by diagrams. There are also described various user roles figuring in the system, its features, capabilities and the entire functionality of the system administration. In the last section, the thesis describes in detail the implementation of the proposed system, storing language content and its administration too.

Poděkování

Děkuji panu Ing. Janu Valdmanovi, Ph.D. za hodnotné vedení mé diplomové práce včetně cenných připomínek. Dále bych rád poděkoval panu Ing. Miroslavu Morávkovi, který byl nápomocen při výkladu zadání a dolaďování přesné funkčnosti výsledné internetové aplikace.

Obsah

1 Úvod

Specializovaný CMS neboli Content Management System je systém pro správu obsahu. Cílem diplomové práce je specializovat již vytvořený CMS či vytvořit úplně nový CMS pro účely správy jazykového vzdělávání. Jaká možnost se zvolí, se rozhodlo dle proběhlé analýzy požadavků jazykové školy na frontální i e-learningovou výuku. Frontální výukou je myšleno statické prohlížení studijního obsahu, kdežto e-learningová část interaguje s uživatelem, tedy studentem. Tato práce se bude zabývat pouze frontálním typem výuky, e-learningová část byla tvořena zvlášť. Vzniklý CMS slouží jazykové škole pro rozšíření možností výuky. CMS umožňuje spravovat celý studijní obsah, tedy vytvářet, editovat či mazat a zobrazovat ho. Dále zvládá správu všech uživatelů i uživatelských rolí. Aplikace je lokalizována do několika jazyků, zabezpečená proti neoprávněnému přístupu a klade důraz na jednoduchost ovládání.

2 CMS systémy na platformě Java EE

V současné době existuje široká škála volně dostupných CMS systémů založených na platformě Java EE. Po proběhlém seznamování se s několika z nich, byly dva vybrány, které zde budou představeny. Oba systémy jsou schopné plně pokrýt potřeby správy obsahu. Mají rozsáhlá a poměrně složitá nastavení. Pokud je potřeba systém využít pro účel, který nenabízí jeho obecná implementace, je nutné do systému udělat poměrně veliký zásah. Na takovéto zásahy nejsou systémy zcela připraveny. Na druhou stranu nabízejí širokou škálu funkcí. Tyto funkce ovšem běžná aplikace požadující pouze správu dokumentů ani nevyužije. Jejich jednoduché použití je omezeno použitím systému dle jejich pravidel, což vždy není možné.
Mezi základní funkce CMS systémů patří:

- Tvorba, modifikace a publikace dokumentů (článků) zpravidla prostřednictvím webového rozhraní, často s využitím jednoduchého online WYSIWYG editoru nebo jednoduchého systému formátování textu (není nutná znalost HTML)
- řízení přístupu k dokumentům, zpravidla se správou uživatelů a přístupových práv
- správa diskusí či komentářů, ať už k publikovaným dokumentům nebo obecných
- správa souborů
- správa obrázků či galerií
- kalendářní funkce
- statistika přístupů

2.1 OpenCMS

OpenCms je profesionální open source Web Content Management System. OpenCms pomáhá vytvářet a spravovat komplexní webové stránky snadno bez znalosti HTML. Integrovaný WYSIWYG editor s uživatelským rozhraním, který je podobný známým kancelářským aplikacím, umožňuje uživateli vytvářet obsah, zatímco sofistikované šablony udržují jednotný vzhled celého webu. OpenCMS je využíván katedrou KIV již několik let.

Logo: 

Licence: GNU General Public License

Poslední vydaná verze: 16.11.2011 – 8.0.3

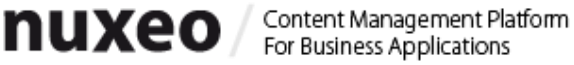
Web: http://www.opencms.org/

2.2 Nuxeo

Nuxeo 5 je inovativní, na standardech založené open source platforma pro aplikace ECM (Enterprise Content Management). Zabývá se aplikací domén jako: Dokument řízení, spolupráce, správu záznamů, správa obsahu a další. Nabízí ucelené a komplexní komponenty k vytváření, zpracování, spravování, publikování a archivování všeho obsahu. To umožňuje organizacím zvýšit efektivitu podnikových procesů. Jeho součástí je na služby orientovaná architektura.

Skládá se ze dvou částí:

- Nuxeo EP (Enterprise Platform) je založena na Java EE a poskytuje ucelený soubor prvků zaměřených na celý rozsah ECM a správu celého životního cyklu obsahu. Využívá moderní technologie Java, včetně Jackrabbit, Lucene, JSF, Seam, jBPM a JRules atd.
- Nuxeo RCP (Rich Client Platform) pro klientské aplikace. Je založen na platformě Eclipse RCP.

Logo: 

Licence: GNU Library or Lesser General Public License (LGPL)

Poslední vydaná verze: 13.12.2011 – 5.5

Web: http://www.opencms.org/

3 Analýza potřeb jazykové školy

Jazyková škola vznesla požadavek na vytvoření e-learningové výuky studentů. Projekt, který má ucelovat jazykové vzdělání v odborné praxi, získal podporu Plzeňského kraje. Aplikace musí umět spravovat studijní obsah, spravovat uživatele a umožňovat studentům vybrat si požadované studium, zaplatit a zahájit vzdělávání. Na závěr student skládá testy, aby dosáhl úspěšného zakončení studia. Následuje výsledek analýzy všech požadavků jazykové školy.

3.1 Metody výuky

Metody výuky jazykové školy se dělí na dva hlavní bloky. Jsou to frontální a e-learningová výuka.

Frontální výuka probíhá prohlížením si obsahů minilekce studovaného submodulu studentem. Minilekce může obsahovat slovíčka a fráze. Student si může obsah prohlížet libovolně dlouho až do vypršení svého studia či složení certifikačního testu. Tato práce se zabývá pouze frontální výukou.

Elearningová výuka probíhá pomocí vyplňování pracovních listů. Každý submodul má vytvořené pracovní listy. Ty jsou interaktivní, tzn., že student může například doplňovat text z viditelného či neviditelného seznamu, překládat slovíčka či fráze atd. Tímto způsobem získá potřebné znalosti a navíc splněním všech povinných pracovních listů se dostane k testovací části, kde již může složit kontrolní a poté certifikační test. Touto částí se tato práce nezabývá.

3.2 Obsah studia

Obsah studia, které bude tato aplikace umožňovat spravovat, je tvořen moduly, submoduly, minilekcemi, gramatickými minilekcemi a pracovními listy a testy. Pracovní listy a testy nejsou zahrnuty v zadání této práce, nebudou tedy detailněji rozebírány.

3.2.1 Minilekce

Minilekce, která je také nazývána jako bloček, je základní vzdělávací jednotka učiva pro studenta. Novou minilekci vytvoří autor, garant novou minilekci schválí a následně editor provede grafickou úpravu. Až poté je garantovi umožněno schválit minilekci pro studium (zveřejnit jí). Minilekce obsahuje text v podobě slovní zásoby či frází a může obsahovat i multimédia, myšleno obrázky, audio či video. Minilekce se vytváří vždy pro jeden určitý modul a přiřazuje se do submodulů příslušného modulu. Může být obsažena v několika submodulech nebo také v žádném. Minilekce uchovává svůj název, textový popis a úroveň obtížnosti, dle které je minilekce přiřazována do submodulů se stejně nastavenou obtížností. Celkem existují tři úrovně obtížnosti. Dále minilekci určuje její jedinečný NACE[1] kód. Ten má následující podobu XX.XX.XY.YY_LN. Prvních pět znaků 'X' určují NACE kód modulu, kam minilekce patří. Tři znaky 'Y' definují jedinečný kód minilekce v rámci jednoho modulu. Kód minilekce se skládá z číslic a jedná se o pořadové číslo minilekce v modulu. Pokud se vyčerpají všechny kombinace číslic, lze použít i písmena. Poslední dva znaky za podtržítkem NACE kódu minilekce označují jazyk modulu a tudíž i dané minilekce. Jazyk je stejný jako jazyk modulu, pro který byla minilekce vytvořena. Dále má minilekce určeno, zda se jedná o obecnou či odbornou minilekci a zda se jedná o minilekci s metodickými pokyny pro lektory či klasickou minilekci. Posledními údaji uchovávanými u minilekce je odkaz na autora minilekce, čas vytvoření, a pokud byla minilekce upravena, pak odkaz na autora úpravy a čas úpravy. Také je uchován schvalující garant.

3.2.2 Gramatická minilekce

Speciálním typem minilekce je gramatická minilekce. Jedná se o všeobecný přehled gramatiky. Uchovává stejné hodnoty jako klasická minilekce, ale má tu zásadní změnu, že není příslušná k modulu, ale k jazyku. Přiřazuje se k submodulům a může se vyskytovat v několika modulech, ovšem vždy v těch, které mají shodný jazyk jako gramatická minilekce. Jelikož gramatická minilekce nepřísluší k modulu, má pozměněný NACE kód: prvních pět znaků NACE kódu jsou vždy "GR.AM.M".

[1] NACE kód je označení nebo klasifikace ekonomických činností pokrývající širokou škálu nejrůznějších činností.

3.2.3 Submodul

Submodul je tvořen množinou minilekcí. Je to studijní jednotka, kterou si student vybírá ke studiu a studuje pak všechny minilekce, které jsou v daném submodulu obsaženy. Submodul průměrně obsahuje dvě až pět minilekcí. Mimo minilekcí obsahuje submodul také definice pracovního listu, kontrolního a certifikačního testu. Submodul vytváří autor vždy pro konkrétní modul. Po schválení garantem je submodul zveřejněn. Submodul uchovává svůj název, textový popis a úroveň obtížnosti, která je totožná i pro všechen studijní obsah submodulu. Dále definuje čas potřebný pro nastudování, resp. za jak dlouho vyprší studium daného submodulu. Submodul má určený také svůj NACE kód, který má podobu XX.XX.X_YYY_LN. Prvních pět znaků ’X’ určují NACE kód modulu, kam submodul patří. Tři znaky ’Y’ definují jedinečný kód submodulu v rámci jednoho modulu. Kód submodulu se skládá z číslic a jedná se o pořadové číslo submodulu v modulu. Pokud se vyčerpají všechny kombinace číslic, lze použít i písmena. Poslední dva znaky za podtržítkem NACE kódu submodulu označují jazyk modulu a tudíž i daného submodulu. Submodul dále stanoví ceny za jeho studium. Jedná se o tři typy cen dle třech typů studia a každá z těchto tří cen je ještě rozdělena na cenu pro studenta, na (vnitřní) cenu pro lektora a na cenu speciální a cenu při promotérských akcích zadavatele projektu. Dále submodul určuje specifickou cenu, která je určena pro studenta, který řádně studoval, ale nesložil úspěšně certifikační test. V takovém případě je mu nabídnuta tato specifická cena pro opakování studia. Další jinou specifickou cenou je cena pro získání necertifikovaného certifikátu. Při běžném skládání certifikačního testu se student nachází v certifikované místnosti s dohlížející osobou. Pokud student zaplatí za necertifikovaný certifikát, může certifikační test skládat doma. Znamená to však, že může používat pomůcky a certifikát nebude tolik věrohodný. Proto je tento certifikát levnější. U submodulu je uchováván autor submodulu, čas vytvoření, a pokud byl upraven, pak autor úpravy a čas úpravy. Je také uchováván schvalující garant.

3.2.4 Modul

Modul je ucelující částí celého studijního obsahu. Uchovává spolu související submoduly a minilekce a je vytvářen vždy pro určitý jazyk. Student při výběru

submodulu prochází nejdříve přes modul a pak se dostává na submoduly. Jeden modul obsahuje řádově desítky až stovky minilekcí; submodulů složených z minilekcí neomezeně. Modul vytváří autor a schvaluje ho garant, až po schválení je modul veřejný. Modul uchovává svůj název, textový popis a označení NACE kód, které má podobu XX.XX.X_LN. Za znaky 'X' se dosazuje NACE kód dle klasifikace ekonomických činností. Pokud NACE kód neobsahuje páté číslo, doplňuje se nula. Pokud je pro jeden čtyřmístný NACE kód v systému více modulů, značí se pátý znak od nuly do devítky, dále pak velkými písmeny bez diakritiky. Poslední dva znaky za podtržítkem NACE kódu modulu označují jeho jazyk. Modul uchovává autora, jímž byl vytvořen, stejně jako čas vytvoření. Při úpravě modulu je uchován autor úpravy a čas úpravy. Je také uchováván schvalující garant.

3.3 Aktoři figurující v systému

Z pohledu jazykové školy je zapotřebí uchovávat celkem osm uživatelských rolí, které pracují se systémem. Pro přehled je následně uveden výčet všech funkcí, práv a povinností, které jednotlivé role zahrnují. U každého uživatele bude uchováván jeho přihlašovací email, heslo, celé jméno, kontaktní údaje, datum narození, kontrolní otázka a odpověď na ni (sloužící pro obnovu hesla při jeho ztrátě) a datum založení uživatelského účtu.

3.3.1 Student

Systém je zaměřen právě na tuto uživatelskou roli, tedy na cílovou skupinu studentů, kteří konzumují studijní obsah. Student si vybírá studium na základě svých preferencí. Vybere si konkrétní submodul a bude studovat jeho minilekce. Minilekce může komentovat a tím dávat zpětnou vazbu autorovi. Dále bude vyplňovat pracovní listy a skládat kontrolní a certifikační testy. Při výběru studia si také volí formu studia. Může zvolit prezenční formu, kde bude docházet pravidelně do školy a samotný e-learning nebude využívat. Bude mít přiděleného lektora, který po splnění studia studentem nastaví tuto skutečnost v systému, a student tak může přejít k testům. Další formou studia je kombinovaná forma. V této formě může student docházet do školy, ale i studovat e-learningově. K testům se dostane buď splněním všech potřebných pracovních listů, nebo po nastavení splnění studia od přiděleného lektora. Poslední,

třetí formou studia je e-learning. Student nemá přiděleného lektora a sám prochází studiem. Využívá frontální i e-learningovou formu výuky, nedochází do školy a po splnění všech potřebných pracovních listů se dostává k testům.

3.3.2 Autor

Autor studijního obsahu je hlavním strůjcem studia. Vytváří a upravuje obsah minilekcí, celou strukturu studijního obsahu (moduly, submoduly a minilekce), otázky, pracovní listy a testy. Všechno, co vytvoří, musí projít schválením garantem, do té doby nebude studijní obsah zveřejněn (zpřístupněn studentům). Autor má určenou specializaci na moduly, což znamená, že může vytvářet a upravovat pouze studijní obsah příslušný daným modulům. Do modulů, na které nemá přidělenou specializaci, nemá přístup. Může podat žádost o přidělení specializace, kterou administrátor schválí nebo zamítne.

3.3.3 Lektor

Lektor se stará o osobní výuku studentů v prostorách školy. Má přidělené studenty, kteří zvolili prezenční nebo kombinovanou formu výuky. Přidělení studentů lektorovi provádí administrátor, a to dle zvoleného submodulu studia. Pokud vlastní lektor na daný submodul certifikát, může se stát lektorem daného studia. Lektor má přístup k výsledkům testů svých studentů. K tomu, aby se stal lektorem určitého submodulu, musí splnit certifikační test daného submodulu na 100 %. Svůj certifikát si musí pravidelně obnovovat před vypršením platnosti certifikátu.

3.3.4 Editor

Editor se stará o správné zobrazení obsahů minilekcí, a to výhradně dle grafické stránky. Má přístup k celému studijnímu obsahu a provádí úpravu a schvalování obsahu minilekcí po schválení obsahu minilekce garantem. Je zodpovědný za grafickou úpravu i spisovnost.

3.3.5 Garant

Garant má na starosti schvalování veškerého vytvořeného studijního obsahu od autora. Stejně jako autor má specializaci dle modulu. O přidělení specializace může

požádat, administrátor posléze žádost buď potvrdí, nebo zamítne. Garant může editovat a poté schválit pouze takový studijní obsah, na který má specializaci.

3.3.6 Certifikovaná osoba

Nemá až tak významnou funkci v systému jako takovém, jako spíš mimo systém. Účastní se totiž skládání certifikačních testů v certifikované místnosti a dohlíží na dodržování pravidel, resp. neopisování. V systému zadává platbu za studium, pokud například někdo přijde jen na certifikační test, přebere od něj certifikovaná osoba peníze a zadá do systému, že jeho studium bylo zaplaceno. Dále opravňuje studenty k zahájení certifikačního testu.

3.3.7 Administrátor

Má na starosti správu celé aplikace. Spravuje uživatele, jejich uživatelské role, u autorů a garantů jejich specializace a schvaluje nové registrace uživatelů (kromě studentů, kteří nepodléhají schvalování). Nemá však přístup k prohlížení, vytváření, úpravě ani jiné správě obsahu studia. Dále přiděluje a mění lektory studentům prezenčního/kombinovaného studia, zadává platbu za studium, spravuje měnové kurzy, jazyky studijního obsahu a partnery zadavatele projektu.

3.3.8 Super administrátor

Má stejná práva i přístupy jako administrátor. Navíc však přiděluje práva aktorům (kromě studenta) k tisku obsahu studia. Má také privilegium nastavit administrátora na super administrátora a naopak.

3.4 Typy studia

Student má v rámci vzniklého systému možnost zvolit si ze tří forem studia. Jsou to prezenční, e-learningové a kombinované.

Prezenční studium nabízí studentovi možnost docházet do školy a učit se klasickou formou ve třídě s ostatními. K takovému studiu je přidělen lektor, který vyučuje své studenty. V systému si může lektor zobrazit své studenty a těm, kteří splnili výuku, nastavit v systému, že mají splněnou výuku a tím jim umožní přejít k testům. Výsledky testů si lektor může u svých studentů v systému prohlédnout.

E-learningové studium je opakem prezenčního. Student vůbec nedochází do školy a ani nemá přiděleného lektora. Prochází si minilekce, učí se a vyplňuje pracovní listy. Po správném složení určeného počtu pracovních listů je student připuštěn k testům. Student si může nastavit i přeskočení studia a v takové chvíli jde k testům okamžitě.

Posledním typem studia je kombinované studium. To v sobě spojuje obě předešlé formy studia. Student může některé dny docházet do školy a některé dny se učit z domova pomocí e-learningu. Má přiděleného lektora, který mu může nastavit splnění studijní části, či splnit všechny potřebné pracovní listy. I zde si student může nastavit přeskočení studia.

3.5 Správa obsahu studia

Správa obsahu dle požadavků probíhá přes tři uživatelské role. Autor vytvoří daný studijní obsah a garant ho schválí, případně před schválením upraví. Pokud se bude jednat o obsah minilekce, musí projít ještě schválením editorem, který se stará o grafickou podobu obsahu minilekce. Po schválení obsahu editorem nastaví garant obsah jako platný, tudíž se obsah minilekce zveřejní. Pokud není potřeba schválení editorem, studijní obsah se při schválení garantem ihned nastavuje jako platný.

Pro vytváření a úpravu obsahů studia je vyžadován editor, který umožňuje jednoduché úpravy textu, jako je nastavení fontu, velikosti písma apod. Tento požadavek řeší WYSIWYG editor. Wysiwyg editor je zkratka pro „what you see is what you get“. Jedná se o editor typu Microsoft Word, který uloží grafickou podobu textu tak, jak ho uživatel při práci s ním vidí. Tento editor navíc umožňuje vkládat a nastavovat velikost obrázků, vkládat a přehrávat video a audio, což jazyková škola také využije.

U obsahu minilekcí je požadováno zachování historie verzí obsahu. Aplikace tudíž musí po každé změně obsahu zachovat rovněž původní verzi. Autor i garant při každé úpravě obsahu a potvrzení úpravy vytvoří nový obsah, který není schválený, a proto se stále používá původní obsah. Poté, co obsah upraví editor, který ovšem svojí úpravou nevytvoří nový obsah, jelikož pouze upravuje grafickou podobu obsahu, a schválí nový obsah, je nový obsah připraven k použití. Nyní na řadu přichází opět garant, který nejnověji vytvořený obsah schválený garantem i editorem nastaví na

aktuální. Tím se aktuální obsah nastaví na neaktuální a zveřejněn bude pouze nový aktuální obsah. Obsah právě nastavený jako neaktuální a všechny ostatní verze obsahu však zůstávají zachovány a přítomny v systému. Tudíž je možné se kdykoli navrátit k původním obsahům.

3.6 Správa uživatelů

Je důležité připravit v aplikaci administrátorské rozhraní, které umožní spravovat uživatele. Dle požadavků jazykové školy by se tak měla umožnit správa uživatelských rolí a správa uživatelských specializací na moduly. Uživatel s uživatelskou rolí administrátor nebo super administrátor může schvalovat nové registrace uživatelů. Super administrátor může schválit registraci i super administrátora, administrátor nikoli. Dále mohou obě administrátorské role editovat uživatelův profil s tou výjimkou, že administrátor nemůže editovat profil super administrátora, ovšem super administrátor smí editovat profil jiného super administrátora. Co žádný administrátor editovat nemůže, je heslo uživatele. Heslo může editovat pouze daný uživatel. Každý uživatel může zároveň editovat svůj profil. Obě uživatelské role smějí editovat uživatelovy specializace (platí pouze pro role autor a garant), schvalovat žádosti o specializace či přímo specializaci uživatelům přidat či odebrat. Administrátoři mohou také nastavovat a měnit uživatelské role. Tato funkčnost se nejvíce využije při žádosti lektora o roli studenta, aby si mohl zvýšit počet certifikátů na submoduly a tím pádem vyučovat více studentů.

3.7 Mimofunkční požadavky

Jazyková škola požadovala, aby systém nejméně v modulu assessment podporoval CS, EN a DE rozhraní a byl připraven na další lokalizace. Díky lokalizaci zabudované ve Springu a lokalizačním textům uložených v properties souborech nebude problém lokalizovat celou aplikaci, a to minimálně do CS, EN a DE rozhraní. Aplikace bude dále podporovat běžné standardy včetně SQL, HTML4 a CSS3. Systém bude tvořen pro jednoduché a intuitivní ovládání, aby byl snadno použitelný.

3.8 Funkční požadavky

Výčet všech identifikovaných funkčních požadavků zobrazuje následující seznam. Detaily požadavků i s popisem se nachází na přiloženém DVD.

REQ1.1	Registrace do systému - student
REQ1.2	Registrace do systému - ostatní aktoři
REQ1.3	Registrace do systému - student, který píše certifikační test
REQ1.4	Přihlášení aktora
REQ1.5	Editace profilu
REQ1.6	Zapomenuté heslo
REQ1.7	Odhlášení
REQ1.8	Vytvoření obsahu studia minilekce
REQ1.9	Úprava obsahu studia minilekce
REQ1.10	Formátování obsahu
REQ1.11	Nahrání multimediálního souboru
REQ1.12	Vytvoření minilekce
REQ1.13	Úprava minilekce
REQ1.14	Zneplatnění minilekce
REQ1.15	Vytvoření submodulu
REQ1.16	Úprava submodulu
REQ1.17	Zneplatnění submodulu
REQ1.18	Vytvoření modulu
REQ1.19	Úprava modulu
REQ1.20	Zneplatnění modulu
REQ1.26	Přidání komentáře k minilekci
REQ1.27	Zneplatnění komentáře k minilekci
REQ1.28	Příjem platby za certifikační test
REQ1.29	Vydání dokladu o platbě
REQ1.30	Prohlížení obsahu minilekce
REQ1.31	Jazykové mutace
REQ1.32	Zabezpečení systému
REQ1.34	Tisk obsahu

REQ1.35	Žádost o změnu specializace
REQ1.36	Zadávání středních/vysokých škol
REQ2.1	Výběr studijního obsahu
REQ2.8	Přeskočení studijní části
REQ4.1	Autorizace aktorů kromě studentů
REQ4.2	Autorizace obsahu
REQ4.3	Autorizace plateb
REQ4.4	Správa studentů
REQ4.5	Správa aktorů
REQ4.6	Stanovení ceny za certifikační test

4 Návrh CMS systému

4.1 Použité technologie

4.1.1 Programovací jazyk

Celá aplikace je založena na platformě přenositelného jazyka Java. Program bude fungovat na internetu, proto je využita verze Javy pro distribuované systémy – Java Enterprise Edition (J2EE). Java je vyvinuta firmou Sun Microsystems a od roku 2008 se jedná o open-source, což znamená, že Java nyní může být vylepšována kýmkoli. Následuje výčet základních vlastností jazyka. Java je objektově orientovaný jazyk vycházející z C++, vyznačuje se ovšem mnohem jednodušší správou. Nemusí se alokovat a uvolňovat paměť pro objekty. Namísto ukazatelů se používají reference, je implementován mechanismus vláken (více-procesové aplikace), objekty lze serializovat (možnost ukládání do souborů, posílání po síti). Dále je implementován mechanismus výjimek, tudíž je možné zachycovat veškeré chyby při běhu programu a efektivně je zpracovávat. Výjimky jsou objektové, proto je možné zachytit celou hierarchii výjimek v hlídaném bloku. Java také poskytuje široké množství knihoven, které značně usnadňují a urychlují práci. Důraz je kladen i na bezpečnost jazyka (přidělování práv, podepisování a bezpečný překlad kódu). [1]

4.1.2 Webový server

Webový server neboli aplikace vyřizuje HTTP[2] požadavky od klienta, což povětšinou bývá internetový prohlížeč. Přijímá požadavky a počítači, který požadavek vznesl, vrací odpověď. Ta je většinou tvořena HTML[3] stránkou. Pro účely aplikace je využit WebSphere Application Server Community Edition od společnosti IBM. Tento server disponuje i potřebným webovým kontejnerem, který vyžaduje každá webová aplikace. Webový kontejner je komponenta zmiňovaného webového serveru, která pracuje s HTTP požadavky a odpověďmi. Tato funkčnost velmi usnadňuje práci,

[2] Hypertext Transfer Protocol (HTTP) je protokol určený k výměně dokumentů mezi klientem a webovým serverem.
[3] Hypertext Markup Language (HTML) je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web (WWW), který umožňuje publikaci dokumentů na internetu.

jelikož po zpracování požadavku webový kontejner předá řízení přímo příslušné metodě, která požadavek obslouží. Není proto nutné implementovat tuto obsluhu požadavků. Samozřejmě je ale možné tuto funkčnost překrýt využitím konfiguračních souborů pro mapování požadavků na příslušné metody, které požadavek obslouží. Webový kontejner také zajišťuje alokování zdrojů a běh servletů (popsány v další kapitole).

4.1.3 Databáze

Pro účely ukládání dat spojených s během webové aplikace byla zvolena databáze DB2 Express-C Community Edition od společnosti IBM. Pracuje na principu klient-server a může být použita na různých operačních systémech. Komunikace probíhá pomocí jazyka SQL[4]. DB2 Express-C je rychlá, bezpečná, spolehlivá a výborně škálovatelná databáze. Ideální pro nastartování většiny malých a středních obchodních aktivit. Je dostupná pro Linux, Unix, Windows a nyní i Mac OS X.

4.1.4 Servlety a Java Server Pages

Základem internetové aplikace jsou servlety, což jsou objekty psané v Javě používající balík *javax.servlet*. Protože servlety musí reagovat na požadavky klientů, které jsou vyřizovány HTTP protokolem, vždy dědí od rodičovské třídy *javax.servlet.http.HTTPServlet*, čímž se z Java třídy stává servlet. Ten je poté jednou zkompilován a načten do paměti, kde přetrvává a reaguje na požadavky vznesené většinou od internetového prohlížeče (klienta). Tím je zaručena velmi rychlá reakce, protože nemusí docházet k opětovné kompilaci. Servlet by měl implementovat metody *doGet* nebo *doPost*, které jsou ekvivalentní s požadavkem od klienta (*get* pro požadavky vyvolané kliknutím na odkaz, *post* vyvolané odesláním webového formuláře). Těmto metodám se předá řízení po vyvolání požadavku. Druhou možností (používanou v aplikaci) je mapování URL[5] pomocí dispatcher servletů (popsány v další kapitole) na metody v kontrolérech (popsány v další kapitole), které mají za vstupní parametry objekty *HTTPServletRequest* a *HTTPServletResponse*. Oba objekty

[4] Structured Query Language (SQL), česky strukturovaný dotazovací jazyk, je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.
[5] Uniform Resource Locator (URL) je řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na internetu.

jsou z balíku *javax.servlet.http*. *HTTPServletRequest* nese informace o uživatelově požadavku a do *HTTPServletResponse* je možné zapisovat odpověď serveru na požadavek.

Java Server Pages (JSP) jsou ve své podstatě také servlety, které zjednodušují zobrazování stránek v podobě HTML. Jejich velkou výhodou je možnost oddělení obsahu stránky od programového řešení. K tomuto dopomáhají JavaBeans (popsány v další kapitole), ve kterých jsou uložená data. JSP využívají speciální tagy (většinou z JSTL[6] knihovny), díky kterým dochází k pravému oddělení obsahu stránky od programu a JSP stránky může připravovat i HTML kodér, který nemá zkušenosti s Javou.

4.1.5 Dispatcher servlet

Dispatcher servlet je servlet (dědí z *HttpServlet* bázové třídy) a jako takový je deklarován ve web.xml webové aplikace. Žádosti klienta, které mají být směrovány na konkrétní Dispatcher servlet pro zpracování, musí být mapovány pomocí URL mapování v konfiguračním souboru web.xml. Jedná se o standardní J2EE servlet konfiguraci. Ukázka mapování požadavků na cms Dispatcher serlvet:

```
<!-- propojeni s cms servletem -->
<servlet>
     <servlet-name>cms</servlet-name>
     <servlet-class>
         org.springframework.web.servlet.DispatcherServlet
     </servlet-class>
     <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
     <servlet-name>cms</servlet-name>
     <url-pattern>/cms/*</url-pattern>
</servlet-mapping>
```

[6] JavaServer Pages Standard Tag Library (JSTL) zapouzdřuje jednoduché tagy základních funkcí společných pro mnoho webových aplikací. JSTL má podporu pro společné úkoly, jako jsou iterace, podmínky a mnoho dalších. Díky nim není nutné využívat scriptlety, které nabourávají filosofii JSP stránek.

4.1.6 Spring framework

Spring framework je open-source aplikační framework šířený pod licencí Apache 2.0. Spring je velice rozsáhlý a nabízí širokou škálu technologií. Díky svým zásadním vlastnostem, a to především jeho modularitě, si můžeme vybrat, kterou část z něho použijeme, aniž bychom museli používat zbytek. Základním stavebním prvkem springu je rozhraní. V aplikaci zjednodušuje přístup do databáze (provádění SQL dotazů). Umí například mapovat atributy databázové tabulky na atributy tříd (tzv. JavaBean). Hlavní výhodou a zároveň zásadním konceptem, kterým se Spring vyznačuje, je Inversion of Control/Dependency Injection (IoC/DI). Jde o velmi jednoduchý koncept, který způsobuje, jak z názvu vyplývá, nadefinování třídy ve Spring-beans konfiguraci, které se vloží do cílové třídy, aniž by je cílová třída musela volat. Děje se tak pomocí setrů cílové třídy, kam Spring automaticky vloží instanci požadované třídy. Slouží k tomu, aby se aplikace zbavila vzájemných vazeb mezi jednotlivými částmi (vrstvami) aplikace vyjádřených ve zdrojovém textu a také vazeb na aplikační framework. Implementace tohoto konceptu se často nazývá lightweight container a právě slovo kontejner je klíčové. Kontejner, patřičně nakonfigurovaný, slouží jako chytrý sklad objektů, který se postará o všechny závislosti, které mezi objekty panují. Příklad konfigurace/definice UserProfileControlleru (cílová třída):

```
<bean
    id="userProfileController"
    class="cz.rcjk.controller.user.UserProfileController">
      <property name="userService" ref="userService" />
      <property
          name="userAuthentication"
          ref="userAuthentication"
      />
      <!-- slouží pro mapování URL požadavků přímo
      na metody třídy (viz dále) -->
      <property
          name="methodNameResolver"
          ref="userProfileActions" />
```

```
</bean>
```

Dle této konfigurace se při spuštění aplikace do cílové třídy UserProfileControlleru vloží potřebné instance tříd UserSerivce a UserAuthentication, které jsou již dříve definovány stejným způsobem (příklad použití vzoru Dependency Injection).

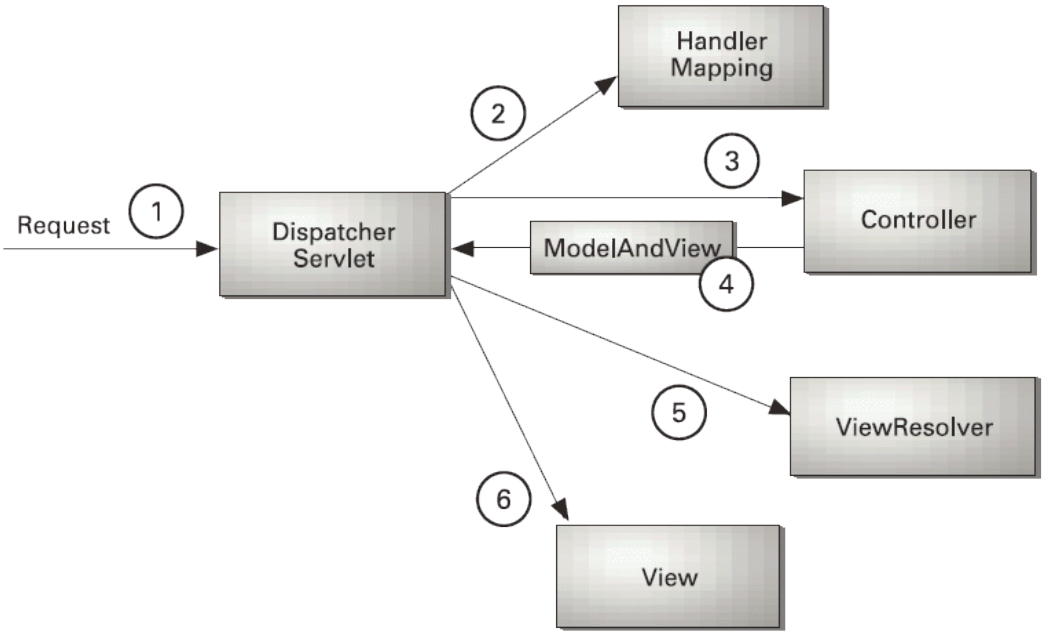
Z modulárního Spring frameworku se využije Spring core, Spring context, Spring Web MVC, Spring ORM (Hibernate) a Spring Security.

4.1.7 Spring Web MVC

Jeho základem je výše zmíněný Dispatcher servlet, který přijímá a rozděluje klientské požadavky. Po obdržení požadavku musí rozhodnout, který kontrolér má požadavek obsloužit. Využije k tomu všechny objekty ve svém kontextu, které implementují rozhraní HandlerMapping. Taková třída se v naprosté většině případů nemusí psát, protože Spring dává k dispozici dvě mocné implementace: *BeanNameUrlHandlerMapping* a zejména *SimpleUrlHandlerMapping*, který je v aplikaci použit.

```
<bean
     id="urlHandlerMappingCms"
     class="org.springframework.web.servlet.handler.
     SimpleUrlHandlerMapping">
      <!--  url mapping  -->
      <property name="urlMap">
          <map>
               <entry
                    key="/introduction"
                    value-ref="introductionController" />
               <entry
                    key="/browsing"
                    value-ref="browsingController" />
               <entry
                    key="/create"
                    value-ref="studyCreateController" />
```

Po rozhodnutí, kam požadavek náleží, je požadavek odeslán na příslušný kontrolér. Ten požadavek obslouží a vrátí tzv. *ModelAndView*, kde je informace, na jaké JSP stránce výsledek zobrazit a parametry k zobrazení. O zobrazení a vybrání správné cesty k JSP stránce se stará *ViewReslover* a *View*. Popsaný postup zpracování požadavku klienta znázorňuje obrázek 1.



Obrázek 1 - Postup zpracování klientského požadavku dle Spring Web MVC

4.1.7.1 Kontroléry

Kontrolér je třída, která má na starosti skutečné zpracování požadavku, tedy obvykle volání nějaké služby aplikační logiky. Ve Springu kontroléry implementují rozhraní *Controller*, pro zpracování stránky s formulářem dědí od SimpleForm *Controller* a při implementování kontroléru s několika obsluhami dědí od *MultiActionController*. Kontrolér po dokončení své činnosti vrátí servletu objekt typu ModelAndView, podle nějž servlet zařídí vygenerování odpovědi. Ta obsahuje jednak jméno toho, co se má odeslat (View), a jednak Mapu objektů, které mají být při generování odpovědi dostupné. Většinou se jedná o data, která se mají uživateli zobrazit. O Mapu se jedná proto, že právě pod tím jménem, pod kterým do ní bude objekt uložen, bude později dostupný.

Při použití třídy *MultiActionController* se přímo na jednotlivé metody daného kontroléru mapují URL požadavky, tudíž jednotlivé metody se starají o obsluhu požadavků. Takové mapování zajišťuje třída *PropertiesMethodNameResolver*. Ukázka mapování pro kontrolér *UserProfileController*:

```
<bean
     id="userProfileActions"
     class="org.springframework.web.servlet.mvc.
     multiaction.PropertiesMethodNameResolver">
      <property name="mappings">
          <props>
               <prop key="/profile">detail</prop>
               <prop key="/profile/edit/password">
                    passwordChangeForm
               </prop>
          </props>
      </property>
</bean>
```

Při použití třídy *SimpleFormController* pro správu formuláře dokáže Spring Web MVC mapovat na jednotlivé prvky formuláře atributy doménového objektu[7]. V kontroléru se nastaví třída, která bude Springem spravována a mapována a také název, pod kterým bude přístupná v JSP stránce. Spring na tento název namapuje danou třídu, tudíž díky EL jazyku se dá přistupovat v JSP stránce k atributům třídy. Po odeslání formuláře se nové hodnoty z políček formuláře namapují na jednotlivé atributy daného doménového objektu. V Dispatcher servletu se nakonfigurují JSP stránky, které budou sloužit pro zobrazení formuláře (formView) a stránky po odeslání formuláře (successView). Pokud připojíme k formuláři validátor, formulář se úspěšně odešle až po úspěšné validaci všech hodnot. Definice kontroléru *UserProfileEditController*:

[7] Doménový objekt koresponduje vždy s jednou databázovou tabulkou. Je popsán v doménovém modelu, který je příbuzný k databázovému modelu aplikace. Doménové objekty reprezentují databázové tabulky.

```
<bean
     name="userProfileEditController"
     class="cz.rcjk.controller.user.
     UserProfileEditController">
      <property name="formView" value="userProfileEdit" />
      <property name="successView" value="userProfile" />
      <property name="userService" ref="userService" />
      <!-- připojení validátoru odesílaných hodnot -->
      <property name="validator" ref="userEditValidator"/>
</bean>
```

4.1.7.2 ViewResolver a View

Když kontrolér vykonal všechno, co vykonat měl, je na čase dát uživateli aplikace najevo, k čemu vlastně došlo. Spring definuje rozhraní *View*, které představuje jakoukoli odpověď klientovi, tedy například JSP stránku, ale i PDF dokument či tabulku v Excelu. V *ModelAndView* se nachází jméno, ze kterého musí servlet získat samotný objekt *View*. K tomu použije všechny objekty implementující rozhraní *ViewResolver*, které nalezne ve svém kontextu. Příklad definice *ViewResolveru* pro CMS:

```
<bean
     id="viewResolverCms"
     class="org.springframework.web.servlet.view.
     InternalResourceViewResolver">
      <property
          name="viewClass"
          value="org.springframework.web.servlet.view.
          JstlView" />
     <property name="prefix" value="/WEB-INF/jsp/cms/" />
     <property name="suffix" value=".jsp" />
</bean>
```

4.1.8 Spring Security

Pro zabezpečení celé aplikace je využito modulu Springu s názvem Spring Security. Ten pomocí jednoho konfiguračního souboru nastavuje kompletní zabezpečení aplikace proti přístupu nepřihlášených uživatelů ke stránkám podléhajícím ověření přihlášeného uživatele. Pomocí Spring Security se definuje každá URL webové aplikace a k ní se přímo nastaví, které uživatelské role (tedy konkrétní přihlášený uživatel) mají ke stránce přístup. Dále se nastaví, zda je URL zabezpečená šifrovaným přenosem dat, tedy nastavení zda se má použít protokol HTTPS[8].

```
<intercept-url
     pattern="/cms/browsing"
     access="ROLE_AUTHOR, ROLE_GARANT, ROLE_EDITOR"
     requires-channel="https"
/>
<intercept-url
     pattern="/cms/profile/administrate"
     access="ROLE_ADMINISTRATOR, ROLE_SUPER_ADMINISTRATOR"
     requires-channel="https"
/>
```

Dále konfigurační soubor definuje mapování portů z nezabezpečených na zabezpečené:

```
<port-mappings>
     <port-mapping http="80" https="443" />
     <port-mapping http="8080" https="8443"/>
</port-mappings>
```

[8] Hypertext Transfer Protocol Secure (HTTPS) je nadstavba síťového protokolu HTTP, která umožňuje zabezpečit spojení mezi webovým prohlížečem a webovým serverem. Zabrání tím odposlouchávání, podvržení dat a umožňuje též ověřit identitu protistrany. HTTPS používá protokol HTTP, přičemž přenášená data jsou šifrována pomocí SSL nebo TLS a standardní port na straně serveru je 443.

Další definicí jsou požadavky odesílané webové aplikaci v závislosti na požadování přihlašovacího formuláře, úspěšném či neúspěšném přihlášení či požadavek na zobrazení stránky při zamítnutém přístupu ke stránce a stránka s odhlášením:

```
<form-login
     login-page="/public/login"
     default-target-url="/public/welcome"
     authentication-failure-url="/public/loginfailed"
/>
<access-denied-handler error-page="/public/403" />
<logout logout-success-url="/public/logout" />
```

Poslední a zásadní konfigurace jest SQL dotaz do databáze, kde se dle předaného loginu (pro účely aplikace se využívá email), vybere heslo uživatele a zjistí se, zda je uživatel platný. Pokud je uživatel nalezen a je platný, pokračuje se zjištěním uživatelových rolí, které určují jeho přístup ke stránkám dle výše uvedených „intercept-url“. Také se zde nastaví, zda je použito kódování hesel. Přesné SQL dotazy jsou uvedeny v kapitole 5 Realizační část, v podkapitole 5.3 Konfigurační soubory Spring frameworku.

Pro přihlašovací formulář se využívají speciální kódová hesla, kterým „rozumí“ pouze Spring Security. Formulář s přihlašovacími údaji se odesílá na URL „j_spring_security_check“ a odhlašování probíhá pomocí zavolání odkazu s kódovým heslem „j_spring_security_logout“. V přihlašovacím formuláři jsou identifikovány položky uživatelské jméno a heslo pomocí „j_username“ a „j_password“.

4.1.9 Hibernate

Hibernate je framework napsaný v jazyce Java, který umožňuje tzv. objektově-relační mapování (ORM). Usnadňuje řešení otázky zachování dat z objektů uložením do relační databáze. Hibernate poskytuje způsob, pomocí nějž je možné zachovat stav objektů mezi dvěma spuštěními aplikace. Říkáme tedy, že udržuje data persistentní. Dosahuje toho pomocí ORM, což znamená, že mapuje javovské objekty na entity v relační databázi. K tomu používá tzv. mapovací soubory, ve kterých je popsáno, jakým

způsobem se mají data z objektu transformovat do databáze a naopak, jakým způsobem se z databázových tabulek mají vytvořit objekty. Poté, co jsou objekty uložené v databázi, se na ně může dotazovat jazykem Hibernate Query Language (HQL), který je odvozen z SQL a je mu tedy velice podobný. ID, NAME a SHORTCUT jsou atributy databázové tabulky namapované na atributy Java třídy. Atribut kolekce *modules* se nachází pouze v Java třídě a vznikl díky vazbě 1:N mezi jazykem a moduly. V Java třídě *Module* tedy bude atribut *language* odkazující na jazyk modulu. Ukázka mapování atributů databázové tabulky LANGUAGE na atributy Java třídy Language.

```
<class
    name="cz.rcjk.data.domain.Language"
    table="LANGUAGE"
    schema="DB2ADMIN">
     <id name="idLanguage" type="int">
         <column name="ID_LANGUAGE" />
         <generator class="assigned" />
     </id>
     <property name="name" type="string">
         <column name="NAME" length="40" not-null="true"/>
     </property>
     <property name="shortcut" type="string">
         <column
             name="SHORTCUT" length="5" not-null="true" />
     </property>
     <set
         name="modules" table="MODULE"
         inverse="true" lazy="true" fetch="select">
          <key>
               <column name="FK_LANGUAGE" not-null="true">
          </key>
          <one-to-many class="cz.rcjk.data.domain.Module">
     </set>
</class>
```

4.1.10 Bean

Beany jsou definovány v konfiguračních souborech Spring frameworku, jsou spravovány Springem a uchovávány ve Spring kontejneru. Beana zastupuje vždy určitou třídu, ze které Spring při spuštění vytvoří právě jeden objekt, který je poté přístupný celé webové aplikaci běžící na Spring frameworku. Obvykle jsou do Beany (tedy třídy) nastaveny vstupní parametry pomocí setterů. Můžou to být přímo hodnoty, anebo odkazy na jiné beany. Díky těmto jedinečným objektům definovaným pomocí bean nejsou v aplikaci zbytečně instanciované třídy několikrát, když postačí jeden objekt dané třídy využívaný všemi ostatními. Příklad beany:

```
<bean id="userController" class="cz.rcjk.UserController">
     <!-- vkládání do beany pomocí odkazů na jiné beany-->
     <property name="userService" ref="userService" />
     <property name="sendEmail" ref="sendEmail" />
     <property name="messageSource" ref="messageSource" />

     <!-- vkládání do beany přímou hodnotou -->
     <property name="publicKey" value="ds5g54fsd4fd5g2" />
</bean>
```

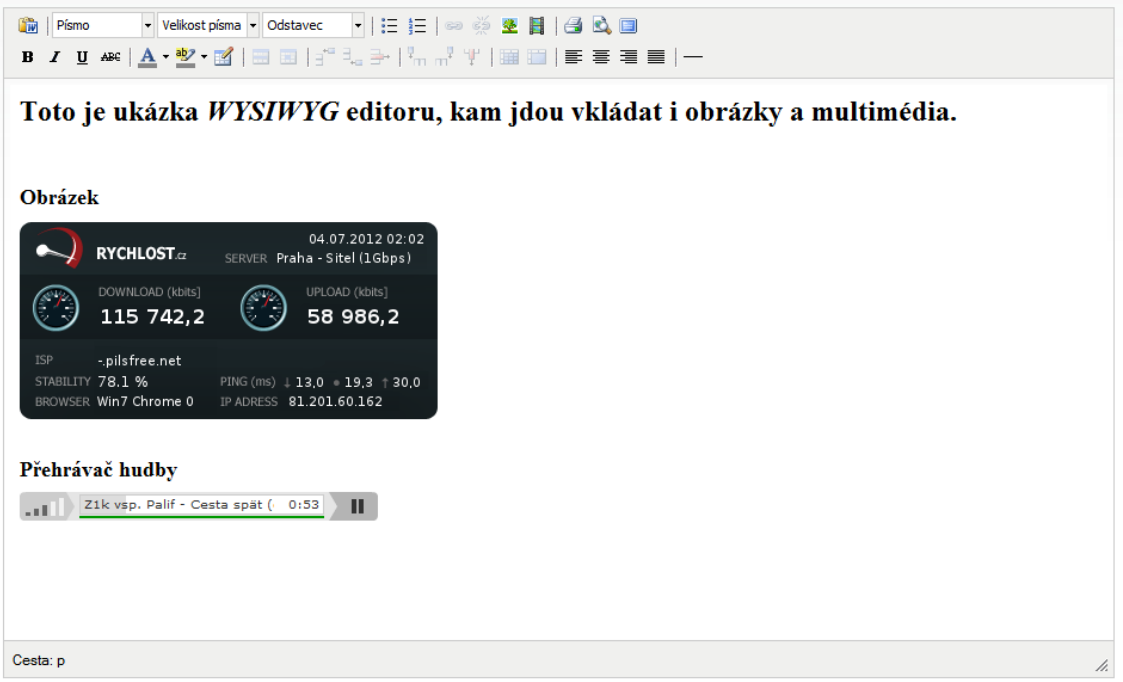
4.1.11 JavaBean

JavaBean jsou třídy psané v Javě dle daných specifikací. Jejich instance jsou uchovávány v kontejneru webového serveru. Je tedy jednoduché se dostat k jejím atributům ze všech JSP stránek pomocí Expression Language (EL) jazyka (lze využít tečkovou notaci). Z Java tříd se k atributům přistupuje pouze pomocí getterů a setterů. Pokud je atribut typu boolean, musí se místo metody *get* implementovat metoda *is*.

4.1.12 TinyMCE

TinyMCE je na platformě nezávislý webový Javascriptový Open Source HTML WYSIWYG editor pod licencí LGPL. TinyMCE má schopnost vylepšit HTML tag textarea o WYSIWYG editor. Prostředí editoru je podobné jako u standardních textových editorů (např. Microsoft Word). Umožňuje vlastní nastavení vzhledu a

tlačítek, které budou uživateli dostupné. Aplikace je dostupná v českém jazyce, který stačí pouze dohrát a nastavit ve zdrojovém kódu. TinyMCE lze velice snadno integrovat do nejrůznějších systémů pro správu obsahu. Ukázka editoru na Obrázku 2.



Obrázek 2 - WYSIWYG editor pro úpravu textu,
vkládání obrázků a přehrávání multimédií

4.1.13 CAPTCHA

CAPTCHA je postup, při kterém se ověřuje, zda je daný uživatel člověk či stroj. Většinou se tak děje opsáním nějakého textu z obrázku. Pro vyvíjenou webovou aplikaci bude využito reCaptchy od společnosti Google. Ta zobrazuje vždy dvě slova a uživatel pro ověření své humanity musí obě slova správně opsat. ReCaptcha umožňuje i místo dvou slov použít přehrání zvukového záznamu, na základě kterého uživatel napíše, co slyší. ReCaptcha má i využití v digitalizaci knih. Slova, která uživatel opisuje, jsou z větší části nerozpoznaná slova při automatické digitalizaci knih. Takto opsané slovo pomůže v rozpoznání dalších slov a rychlejšímu a automatizovanému rozpoznávání knih. Stačí si zaregistrovat svůj web na recaptcha.com a uživatel dostane public a private key. Poté stačí vložit do HTML stránky Javascritpový kód, vyplnit public key, který se odesílá serveru pro generování slov pro reCaptchu a vložit několik

řádků kódu do kontroléru ověřujícího správnost zadaných slov. Od té chvíle se dá využít reCaptcha ověření humanity na vlastním serveru.

4.2 Architektura webové aplikace

Aplikace bude založena na několikavrstvé architektuře. O zpracování dat se bude starat datová vrstva řízená Hibernatem, výpočetní logiku bude zajišťovat aplikační vrstva implementující Spring Web MVC model a vizuální stránku aplikace a interakci s uživatelem bude umožňovat prezentační vrstva tvořená JSP stránkami.

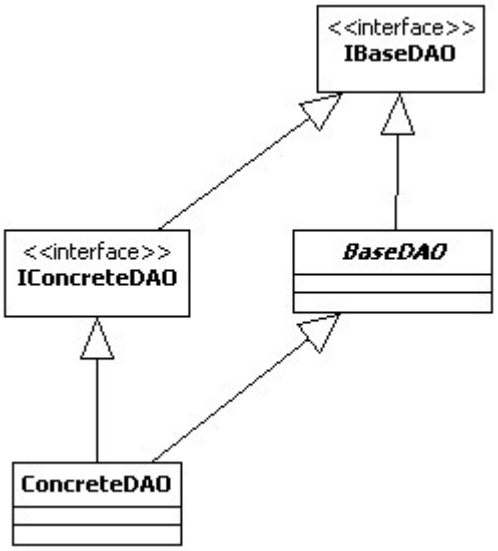
4.2.1 Datová vrstva

Datová vrstva disponuje metodami pro práci s daty uloženými v databázi. Zajišťuje načítání a ukládání dat, případně modifikace a mazání stávajících dat. Veškeré metody pracující s daty jsou uloženy v DAO[9] objektech. Metody pracují s doménovými objekty, které jsou vytvořeny dle tabulek v databázi a jsou perzistentní. O všechny perzistentní doménové objekty se stará Hibernate, jehož metody jsou volány z připravených DAO metod.

Jelikož je v každém DAO objektu, který náleží jednomu doménovému objektu, zapotřebí implementovat téměř identický kód CRUD[10] operací, vzniklo rozhraní, které využívá generičnosti Javy (*IBaseDAO* dle Obrázku 3). To znamená, že toto rozhraní nemá určený doménový objekt, se kterým pracuje, je tedy obecné. Rozhraní implementuje abstraktní generická třída (*BaseDAO* dle Obrázku 3), a to pomocí volání metod Hibernatu. Tímto jsou implementovány CRUD operace a není zapotřebí je v každém DAO objektu implementovat. Vytvořené rozhraní pro každý jednotlivý doménový objekt (*IConcreteDAO* dle Obrázku 3) dědí z generického rozhraní a nastavuje mu již konkrétní doménový objekt, se kterým bude pracovat. Toto konkrétní rozhraní ještě přidává specifické metody, které se provádějí s daným doménovým objektem. Třída implementující konkrétní rozhraní určitého doménového objektu (*ConcreteDAO* dle Obrázku 3) dědí z abstraktní generické třídy, čímž dosáhne implementace všech metod z generického rozhraní. Zbývá tedy již pouze

[9] Tyto objekty tvoří jakousi mezivrstvu mezi databází a aplikační logikou. Díky jejich metodám je zcela odstíněna databáze od zbytku programu, jelikož se program na data dotazuje pouze přes tyto DAO objekty. Znamená to, že data mohou být uložena v libovolné databázi nebo třeba v souborech na disku.
[10] Create, read, update, delete (CRUD) je soubor čtyř základních metod, které musí implementovat každá aplikace pracující s daty. Je nezbytný pro to, aby bylo možné data vytvářet, číst, editovat a mazat.

implementovat specifické metody z konkrétního rozhraní. Tento návrh řešení nejen CRUD operací, které se opakují ve všech DAO objektech, značně zjednodušuje a zpřehledňuje všechny DAO objekty. Následující Obrázek 3 znázorňuje popsaný postup implementace DAO.



Obrázek 3 - Způsob implementace DAO tříd

4.2.2 Servisní vrstva

Servisní vrstva stojí na pomezí mezi aplikační logikou a datovou vrstvou. Přijímá požadavky od kontrolérů a na základě daného požadavku volá metody z DAO objektů. Vrácená data přeposílá zpět aplikační logice. Může dělat i určité úkony nad daty bez volání metod z DAO objektů a výsledek této činnosti opět vrátí aplikační logice. Následující obrázek zobrazuje

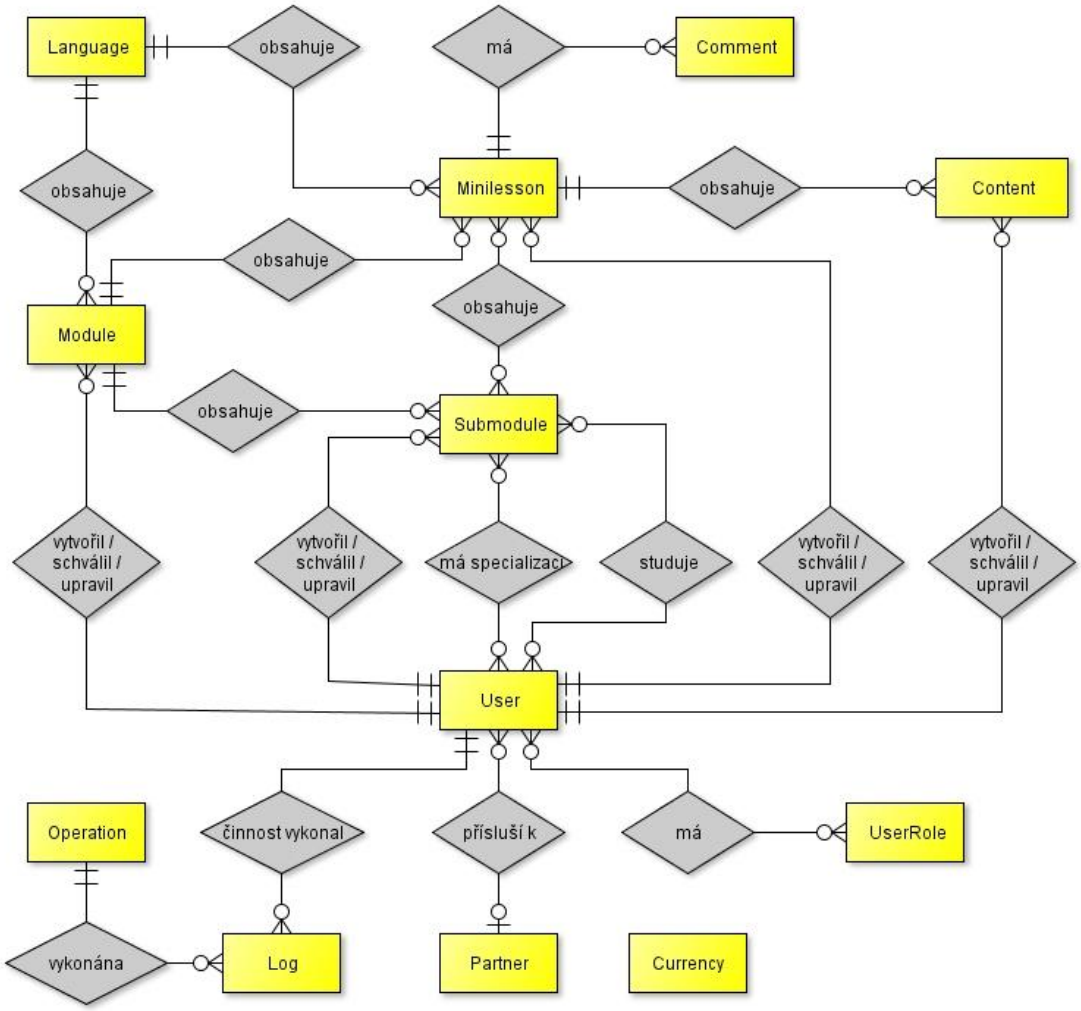
4.2.3 Aplikační vrstva

Aplikační vrstva je celkově tvořena kontroléry. Dle Spring Web MVC modelu kontroléry zpracovávají požadavky od uživatele, řídí postup zpracování požadavku, volají metody ze servisní vrstvy a výsledek posílají v podobě kolekce mapa k zobrazení JSP stránce.

4.2.4 Prezentační vrstva

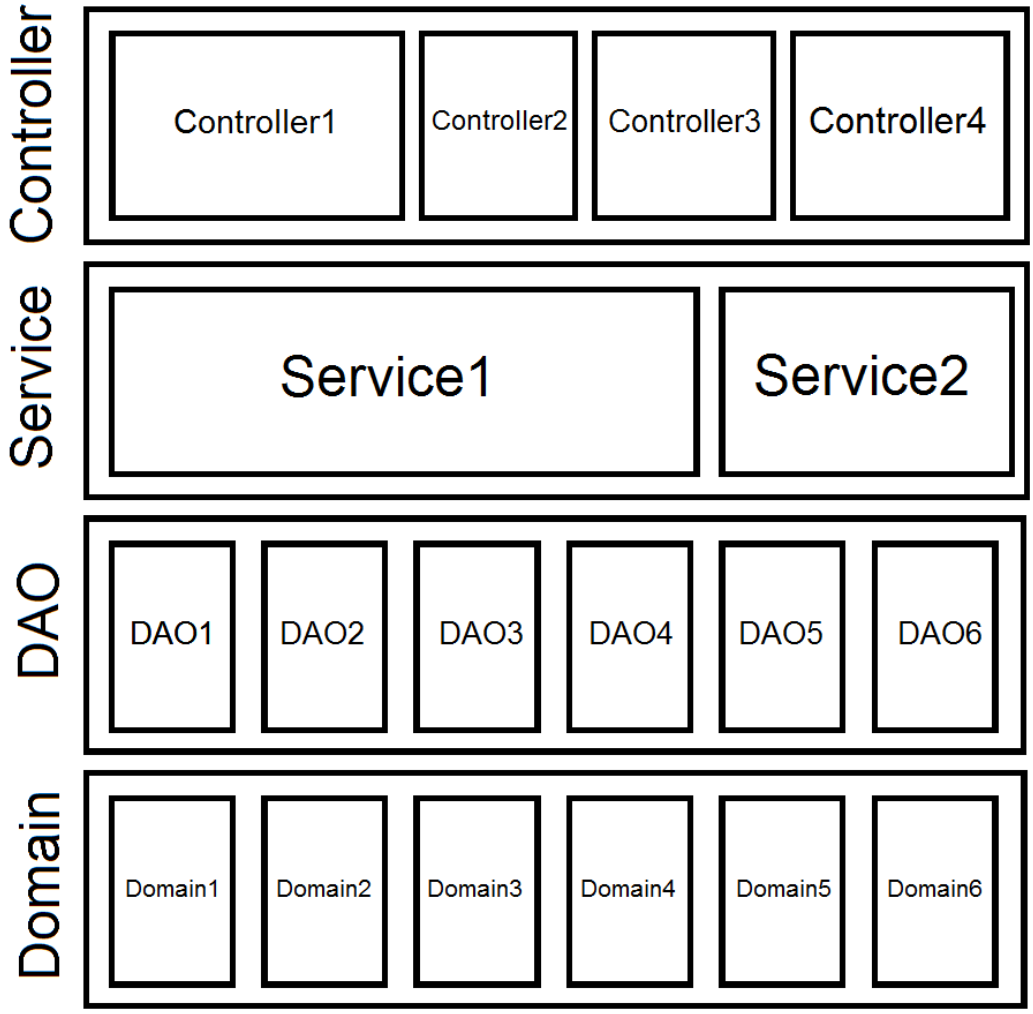
Pomocí prezentační vrstvy je umožněna interakce uživatele s programem. Interakce je zajištěna zobrazením JSP stránek v uživatelově internetovém prohlížeči. Uživatel má možnost zadávat data do připravených formulářů a tím interagovat se systémem. JSP stránky umí zobrazit informační texty po odeslání vyplněného formuláře. Zobrazují také chybové zprávy po vložení chybných údajů.

4.3 ER model databáze



Obrázek 4 - ER model databáze

4.4 Nástin spolupráce tříd



Obrázek 5 - Nástin hierarchie tříd

5 Realizační část

Navržený systém v předchozích bodech této práce byl implementován bez použití a potřebných úprav již hotového CMS systému. Celá práce vznikala od začátku, tedy byl vytvořen zcela nový systém. Tento postup byl zvolen z důvodu specifických uživatelských rolí, které by se těžko zanášely do existujících CMS systémů. Navíc správě obsahu bude podléhat pouze jeden doménový objekt, tudíž využívat malocha CMS systém bylo vyhodnoceno jako neefektivní.

5.1 Příprava vývojového prostředí

Systém je postaven na platformě Java EE. Jeho běh zajišťuje aplikační server WebSphere Application Server Community Edition s databázovým serverem DB2 Express-C (obojí od firmy IBM). Vývoj probíhal ve vývojovém nástroji SpringSource Tool Suite, který je vystavěn na známém prostředí Eclipse. Vývoj aplikace probíhal na operačním systému Windows 7 64bit.

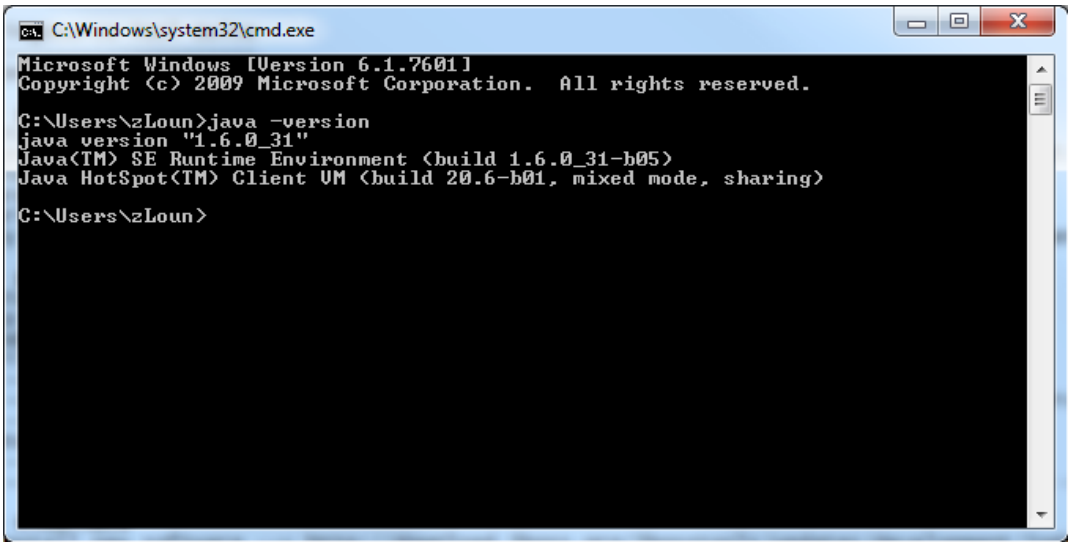
5.1.1 Instalace vývojového prostředí

5.1.1.1 Java

Nejdříve je třeba zajistit, aby byla na počítači nainstalována Java. Ta musí podporovat vývoj aplikací, tj. Java Development Kit (JDK). Tato verze Javy obsahuje nejen Java Runtime Environment (JRE) nezbytný pro běh programů programovaných v Javě, ale i knihovny umožňující překlad vyvíjených aplikací (JDK). Pokud JDK (případně ani Java) instalován není, je možné jej stáhnout z internetových stránek firmy Oracle. Na stránce Java SE Downloads je na výběr z několika možností. Pro účely vývoje aplikace postačí samotné JDK.

Je vhodné také přidat proměnnou s cestou k adresáři s nainstalovanou Javou do systémových proměnných a do systémové cesty. Do systémových proměnných se dá dostat kliknutím pravým tlačítkem na Tento počítač a vybráním položky Vlastnosti. Dále se klikne v levém menu nově otevřeného okna na Upřesnění nastavení systému, čímž se otevře nové okno Vlastnosti systému. Zde se vybere záložka Upřesnit a dále

se klikne na tlačítko Systémové proměnné. Otevře se okno se systémovými proměnnými. Jsou zde možnosti přidat proměnné pro aktuálního uživatele i pro celý systém. Důležité je přidat proměnnou do systémových proměnných. Klikne se na Nová a jako jméno proměnné se vyplní JAVA_HOME a hodnota proměnné cesta k instalovanému JDK (např.: c:\Program Files (x86)\Java\jdk1.6.0_31). Proměnná se uloží stisknutím tlačítka OK. Nakonec je ještě potřeba editovat proměnnou Path, přidat do ní řetězec "%JAVA_HOME%\bin" a pozavírat otevřená okna pomocí tlačítek OK. Tím je přidání proměnné do systémové cesty hotové. Správnost vložení proměnné se zkontroluje zadáním řetězce "java -version" do příkazové řádky systému Windows. Měla by se vypsat verze nainstalované Javy (viz obrázek 6).



Obrázek 6 – Ukázka výpisu po příkazu "java –version"

5.1.1.2 Databázový server DB2 Express-C

Ukládání dat zajišťuje databáze DB2 Express-C od firmy IBM. Instalace se provede pomocí samorozbalovacího archivu nacházejícího se na přiloženém DVD (db2exc_975_WIN_x86.exe). Ten se po spuštění dotáže, kam se má rozbalit, a po potvrzení cesty dojde k rozbalení archivu a spuštění informačního okna. V okně se vybere Instalovat, což způsobí spuštění běžného instalátoru. V něm se na uvítacím okně klikne na tlačítko Další, na dalším okně je Licenční smlouva, kterou je nezbytné přijmout. Po přijetí se klikne opět na tlačítko Další, čímž se průvodce instalací přesune na výběr typu instalace. Zde postačí vybrat typickou instalaci (či jinou v závislosti na

uživatelových preferencích) a pokračuje se opět stiskem tlačítka Další. Ocitáme se na stránce, kde se vybere volba Instalovat produkt DB2 Express-C do počítače a zároveň uložit nastavené parametry do souboru odpovědí (opět ovšem záleží na preferencích uživatele) a vybere se cesta, kam se uloží soubor s odpověďmi. Po pokračování v průvodci instalací se dostane na stránku s výběrem instalační složky. Zadá se složka a pokračuje se na další stránku, kde se zadá jedinečný název kopie produktu DB2. Po přejití na další stránku již proběhne samotná instalace produktu. Administrátor má přihlašovací údaje k databázovému serveru: login – db2admin a heslo – db2admin.

5.1.1.3 Instalace WebSphere Application Server Community Edition

Na řadu přichází instalace webového serveru s webovým kontejnerem, který umožňuje běh servletů a JSP stránek. Po dohodě se zadavatelem byl zvolen produkt od stejné firmy jako databázový server, tedy od IBM, WebSphere Application Server Community Edition. Instalační soubor se nalézá na přiloženém DVD (wasce_setup-3.0.0.0-win.exe). Po jeho spuštění se zobrazí instalační průvodce s uvítací stranou, kliknutím na tlačítko Next se instalace posune na další stranu. Zde je potřeba zaškrtnout souhlas s licenčními podmínkami a pokračovat dále tlačítkem Next. Na další stránce se zvolí instalační adresář a klikne se na Next. Další stránka nabízí možnost spolu s instalací aplikačního serveru nainstalovat i plugin pro Eclipse. Tato instalace se odmítne pomocí zaškrtnutí "No, thanks", jelikož k instalaci pluginu do Eclipsu pro WebSphere dojde později. Pokračuje se tlačítkem Next a po prohlédnutí souhrnných informací se kliknutím na tlačítko Install zahájí instalace.

Důležité je nastavit na složku, kam byl WepSphere nainstalován, plná přístupová práva pro přihlášeného uživatele, aby v ní WebSphere mohl dělat změny, jinak se ani nepodaří server nastartovat z prostředí SpringSource Tool Suite. Pokud server bude používán na localhostu, jeho URL jest: http://localhost:8080/<nazev_webove_aplikace> nebo pro zabezpečený přenos: https://localhost:8443/<nazev_webove_aplikace>.

5.1.1.4 Instalace SpringSource Tool Suite

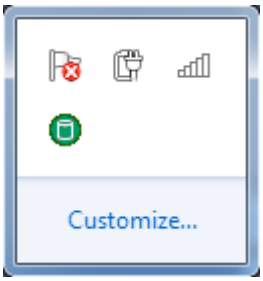
Poslední nutnou položkou, která je potřeba pro samotný vývoj aplikace, je SpringSource Tool Suite. Jedná se o oblíbené a hojně používané vývojové prostředí

Eclipse, které je již v základu rozšířeno o nástroje pro snadnou práci a správu Spring frameworku. Na přiloženém DVD se nachází jak instalační soubor (springsource-tool-suite-2.9.2.RELEASE-e3.7.2-win32-installer.exe), tak archiv (springsource-tool-suite-2.9.1.RELEASE-e3.7.2-win32.zip). Pokud není žádoucí nebo umožněno provádět na cílovém počítači instalaci, postačí rozbalit archiv a ihned se dá se SpringSource Tool Suitem pracovat. Tuto možnost nabízí Eclipse ve všech svých verzích. Je-li možné instalaci provést, spustí se instalační soubor, čímž se spustí instalační průvodce. Z úvodní stránky se pokračuje tlačítkem Next, na další stránce se zaškrtne souhlas s licenčními podmínkami a opět se pokračuje stiskem tlačítka Next. Následující stránka instalačního průvodce nabízí zadání cílové cesty pro instalaci, po jejím zadání se opět stiskne tlačítko Next. Průvodce dále nabídne možnost výběru balíků, které se mají nainstalovat, balík SprinSource Tool Suite se samozřejmě nainstalovat musí. Po kliknutí na Next je třeba zadat cestu k instalaci JDK (např. výše zmíněná: c:\Program Files (x86)\Java\jdk1.6.0_31) a pokračovat tlačítkem Next, čímž spustíme instalaci. Nyní je vše připraveno pro počáteční nastavení nainstalovaných systémů.

5.1.2 Konfigurace vývojového prostředí

5.1.2.1 Vytvoření databáze

Pro vytvoření a naplnění databáze se využije nástroj Řídicí centrum DB2. Tento nástroj umožňuje kompletní správu všech databází běžících na nainstalovaném databázovém serveru. Nástroj byl nainstalován společně s instalací databázového serveru, tudíž není potřeba instalace žádného databázového správce třetích stran. Po instalaci byl databázový server přidán mezi služby Windows, spouští se při startu systému a je přístupný přes zelenou ikonu v oznamovací oblasti Windows (viz obrázek 7).



Obrázek 7 – Zelená ikona DB2 Express-C

Na ikonu se klikne levým tlačítkem a z kontextového menu se vybere položka Řídicí centrum DB2. Po naběhnutí Řídicího centra se klikne na ikonu označující Editor příkazů (viz obrázek 8). V editoru se zadá příkaz pro vytvoření databáze:

```
CREATE DATABASE RCJK USING CODESET UTF-8 TERRITORY CS_CZ;
```

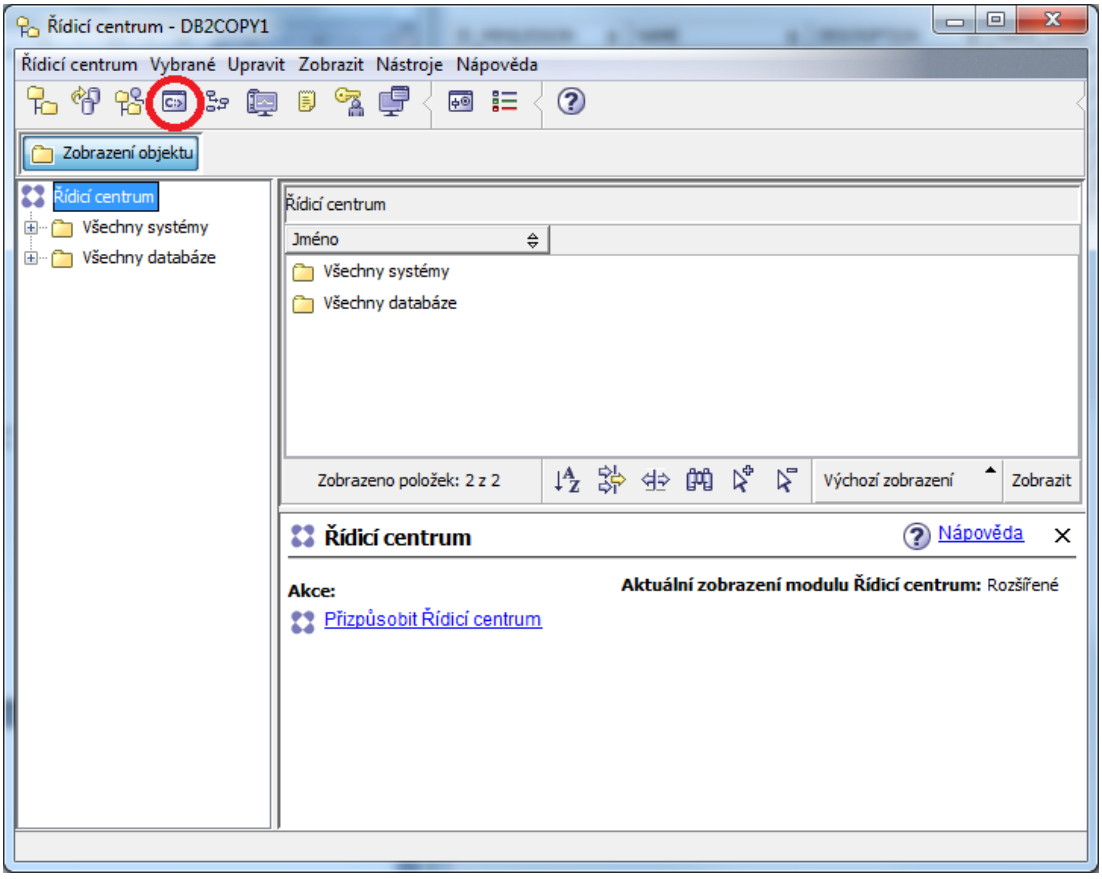
Tím dojde k vytvoření nové databáze s názvem RCJK, kódováním UTF-8 a českou lokalizací. Pro vytvoření a naplnění tabulek musí nejprve dojít k připojení k databázi, což se provede příkazem:

```
CONNECT TO RCJK USER db2admin USING db2admin;
```

Nyní je vše připraveno k naplnění databáze tabulkami a daty. To se provede pomocí zkopírování obsahu SQL skriptů připravených přesně k tomuto účelu, nalézajících se na přiloženém DVD. Nejprve se provede skript s názvem database_tables.sql, který vytvoří v databázi všechny potřebné tabulky. Poté se spustí skript database_data.sql, který naplní tabulky daty, a nakonec se spustí skript database_constraints.sql, který zavede v databázi doménová omezení, resp. určí, které cizí klíče patří k jakým tabulkám (doménám).

5.1.2.2 Přidání a konfigurace pluginů pro SpringSource Tool Suite

Posledním krokem před zahájením vývoje aplikace je nastavení pluginů ve SpringSource Tool Suite. Pro správu projektu i programového kódu je používán Rational Team Concert od firmy IBM běžící na serveru https://jazz.kiv.zcu.cz. Plugin je potřeba pro správu projektu, aplikační server a i Hibernate. Nejprve je nutné spustit SpringSource Tool Suite a nastavit domácí adresář pro ukládání projektů. Vybere se adresář, kde budou uloženy všechny projekty včetně nastavení postupně vytvářeného při práci s vývojovým prostředím.



Obrázek 8 – Ikona Editoru příkazů v Řídicím centru DB2

Instalace RTC pluginu

Instalace pluginu pro správu projektu probíhá tak, že ve SpringSource Tool Suite se klikne v menu na Help a dále na Install New Software. V nově otevřeném okně se klikne na tlačítko Add, zadá se název (např.: RTC) a klikne se na Archive. To způsobí otevření Open Dialogu, kde se z přiloženého DVD vybere plugin pro RTC (RTC-Client-p2Repo-3.0.1.1.zip), klikne se na Open a poté na OK. Tím se přidají dvě položky do seznamu se softwarem k instalaci. Zaškrtnutím se vybere položka s názvem Rational Team Concert Client (extend an Eclipse installation) a klikne se na tlačítko Next. Dojde k zjištění Requirements and dependencies. Vybere se souhlas s podmínkami a zahájí se instalace. Během instalace může dojít i k instalaci nepodepsaného softwaru, před jehož instalací bude uživatel varován a dotázán zda, se má pokračovat. Zvolí se pokračování a plugin se v pořádku nainstaluje.

Nyní již bude možné přepnout se do perspektivy Work Items (pracovní položky). To se provede vybráním položky Window z menu, dále Open Perspective a Other. V nově otevřeném okně se vyhledá a zvolí Work Items a potvrdí stiskem OK.

Vývojové prostředí se přepne do nové perspektivy, k původní je možno se vrátit pomocí tlačítek v pravém horním rohu. Připojení k projektu se provede kliknutím na odkaz Accept Team Invitation ve View Team Organization. Předpokládá se, že byla předem poslána pozvánka od člena týmu. Zároveň člen týmu upozorní, že poslal pozvánku a předá přihlašovací údaje. Po úspěšném připojení k Jazz serveru zbývá stáhnout aktuální kód z Repository Workspace. Klikne se na My Repository Workspace, dále na RCJK Team Stream Workspace, poté RCJK Team Default Component, dále pravým tlačítkem na Load a nakonec se klikne na Find and load Eclipse projects, čímž se dostaneme do nabídky založení nového projektu se zdrojovými kódy z Jazz serveru.

Instalace WebSphere pluginu

Na řadu přichází instalace pluginu pro aplikační server WebSphere, která umožní přidat definici WebSphere aplikačního serveru přímo do SpringSource Tool Suite a tím jednoduše publikovat nový kód na server. Instalace je jednodušší než předchozí, stačí rozbalit archiv (wasce_eclipse-plugin-3.0.0.1-deployable.zip) a nakopírovat ho do složky sts-2.9.2.RELEASE, která se nachází v adresáři s instalací SpringSource Tool Suite. Restartuje se vývojové prostředí a plugin je připraven k použití.

Nyní je potřeba přidat nový server, který je již nainstalován a připraven k použití. Z hlavního menu se vybere File, dále New, poté se klikne na Other a zobrazí se nové okno. V okně se vybere Server a pokračuje se tlačítkem Next. V novém okně se rozbalí IBM a zvolí se IBM WASCE v3.0 Server. Jako Server's host name se ponechá localhost, Server name se také ponechá a pro Server runtime environment se vytvoří nová konfigurace. Za JRE se zvolí nainstalované JDK a jako Application Server Installation Directory se zvolí cesta k nainstalovanému WebSphere (např: C:\Program Files\IBM\WebSphere\AppServerCommunityEdition). Nová konfigurace se potvrdí pomocí tlačítka Finish a vytváření definice serveru pokračuje stiskem tlačítka Next. Na další stránce se nastaví Hostname na localhost, Administration id na system a Administration password na manager. Definice serveru se vytvoří pomocí tlačítka Finish.

Dále je doporučeno vypnout automatické publikování na server. Jelikož jedno publikování trvá cca 1 minutu, je dosti nepraktické nechat server automaticky

publikovat po pár změnách. Také je vhodné navýšit maximální hodnoty používané operační paměti serverem, potažmo Java Virtual Machinem. Pokud se tak nestalo automaticky, pomocí hlavního menu Window a dále Show View se zobrazí Servers. V tomto View se následně dvakrát poklepe myší na dříve vytvořený server (dle návodu o bod výše) a rozbalí se roletkové menu Publishing. Ze zobrazených možností se vybere Never publish automatically a pomocí klávesové zkratky ctrl+s se nové nastavení uloží. Dále se v General Information klikne na odkaz Open launch configuration, což otevře nové okno s možnostmi konfigurace serveru. Ze zobrazených možností se zvolí záložka Arguments. V textovém poli nadepsaném VM arguments se nastaví následující hodnoty:

-Xms256m -Xmx1024m -XX:MaxPermSize=512m

Nakonec se klikne na Apply a na OK, čímž je nastavení definice serveru hotovo.

Instalace Hibernate pluginu

Posledním pluginem, který je nutné přidat do SpringSource Tool Suite, je plugin pro Hibernate pro umožňující generování Java doménových tříd dle tabulek v databázi. Ve SpringSource Tool Suite se klikne v menu na Help a dále na Install New Software. V nově otevřeném okně se do políčka Work With vyplní cesta k balíčkům s Hibernate Tools (http://download.jboss.org/jbosstools/updates/development/indigo), což způsobí přidání několika položek do seznamu se softwarem k instalaci. Do filtrovacího políčka, ve kterém je šedě napsáno “type filter text“, se napíše Hibernate Tools, což způsobí zobrazení pouze těch položek, které jsou vyžadovány. Zaškrtnutím se vybere položka s názvem a klikne se na tlačítko Next. Dojde ke kalkulaci Requirements and dependencies. Vybere se souhlas s licenčními podmínkami a zahájí se instalace. Během instalace může dojít i k instalaci nepodepsaného softwaru, před jehož instalací bude uživatel varován a dotázán, zda se má pokračovat. Zvolí se pokračování a plugin se v pořádku nainstaluje.

Nyní již bude možné se přepnout do perspektivy Hibernate. To se provede vybráním položky Window z menu, dále Open Perspective a Other. V nově otevřeném okně se vyhledá a zvolí Hibernate a potvrdí stiskem OK. Vývojové prostředí se přepne do nové perspektivy. Pro generování doménových tříd je nyní potřeba nastavit konfiguraci Hibernate. Ve View Hibernate Configuration se klikne pravým tlačítkem myši na prázdnou plochu a vybere Add Configuration. V nově otevřeném okně se

vyplní Name (může se ponechat základní), Project se vybere dříve naloadovaný z RTC Workspacu a dále je zapotřebí vytvořit Database Connection. Klikne se na New a v nově otevřeném okně se vybere DB2 for Linux, UNIX, and Windows a pokračuje se stisknutím tlačítka Next. Na další stránce průvodce se zadá název databáze RCJK, Host se nastaví na localhost a Port number na 50000. Do User name se zadá db2admin a do Passwordu také db2admin. Default schema se ponechá prázdné a v Connection URL se ponechá předvyplněné:
jdbc:db2://localhost:50000/RCJK:retrieveMessagesFromServerOnGetMessage=true;
Vytváření nového Database connection se potvrdí tlačítkem Finish. Dále se vytvoří Property file kliknutím na tlačítko Setup a dále na Create new. V novém okně se pouze vybere umístění souboru (doporučuje se použít resource složku projektu) a potvrdí se stiskem tlačítka Finish. Posledním konfiguračním souborem je Configuration file. Opět se klikne na tlačítko Setup a dále na Create new, v novém okně se vybere umístění souboru a pokračuje se tlačítkem Next. Na další stránce se vyplní Database dialect, kde se z nabídky zvolí DB2, dále se nastaví Driver class na com.ibm.db2.jcc.DB2Driver, Connection URL na jdbc:db2://localhost:50000/RCJK a nakonec Username a Password opět obojí na db2admin. Kliknutím na Finish a v předchozím okně na OK se vytvoří nová konfigurace Hibernate.

5.2 Konfigurační soubory webové aplikace

Framework Spring disponuje rozsáhlou škálou nastavení, které se provádí přes XML konfigurační soubory. Je možné všechna nastavení provést v jednom souboru nebo vytvořit několik souborů s logickým pojmenováním. Projekt disponuje několika oddělenými konfiguračními soubory, které jsou spojeny pomocí souboru web.xml.

5.2.1 Geronimo-web.xml

Geronimo-web.xml, neboli Geronimo Deployment je plán nasazení pro webové aplikace, které jsou obvykle zabaleny do WAR souboru. Geronimo-web.xml je používán ve spojení s web.xml (Java EE deployment descriptor) pro nasazení webových aplikací skládajících se z Java Servlet Pages (JSP) a servletů do aplikačního serveru Geronimo. Volitelně může být použit ke konfiguraci Geronimo webového

serveru, kde bude webová aplikace nasazena. Část geronimo-web.xml představuje následující ukázka:

```
<!-- ukázka základního nastavení -->
<dep:environment>
     <dep:moduleId>
          <dep:groupId>default</dep:groupId>
          <dep:artifactId>RCJK</dep:artifactId>
          <dep:version>1.0</dep:version>
          <dep:type>car</dep:type>
     </dep:moduleId>
</dep:environment>
<web:context-root>/RCJK</web:context-root>
```

Zde je zásadní část v nastavení context-root, který určuje URL cestu k aplikaci.

5.2.2 Web.xml

Navazuje na geronimo-web.xml a obsahuje základní konfigurace webové aplikace společně s jednotlivými subjekty rozmístěnými na serveru. Definuje umístění všech konfiguračních souborů celé webové aplikace včetně konfiguračních souborů Spring Frameworku, dále rozděluje obsluhu uživatelských požadavků na jednotlivé Dispatcher servlety[11] dle URL cesty požadavku uživatele. Dále nastavuje správné kódování znaků v celé aplikaci, obzvláště pak se toto nastavení využije v odesílaných formulářích. V neposlední řadě povoluje použití Spring Security pro celou webovou aplikaci a nastavuje uvítací stránku. Ukázka představuje propojení všech konfiguračních souborů webové aplikace a filtraci uživatelových požadavků dle URL

```
<!-- propojení s konfiguračními soubory a servlety -->
<context-param>
     <param-name>contextConfigLocation</param-name>
     <param-value>/WEB-INF/applicationContext.xml,
```

[11] Servlety starající se o předávání obsluhy uživatelských požadavků kontrolerům (např.: cms-servlet.xml).

```
            /WEB-INF/applicationContext-hibernate.xml,
            /WEB-INF/applicationContext-serviceAndDao.xml,
            /WEB-INF/applicationContext-security.xml,
            /WEB-INF/public-servlet.xml,
            /WEB-INF/cms-servlet.xml,
            /WEB-INF/assessment-servlet.xml
      </param-value>
</context-param>

<!-- propojení s cms servletem -->
<servlet>
      <servlet-name>cms</servlet-name>
      <servletclass>
         org.springframework.web.servlet.DispatcherServlet
      </servlet-class>
      <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>cms</servlet-name>
      <url-pattern>/cms/*</url-pattern>
</servlet-mapping>
```

5.3 Konfigurační soubory Spring frameworku

Konfigurace Spring frameworku se skládá ze čtyř konfiguračních souborů, a to pro samotný Spring framework, pro Hibernate, pro Spring Security a pro Service a DAO. Dále pak obsahuje tři Dispatcher servlety.

5.3.1 ApplicationContext.xml

ApplicationContext.xml je základní konfigurací aplikačního kontextu Spring frameworku. Obsahuje nastavení AOP[12], které zajišťuje správnou funkčnost

[12] Aspektově orientované programování, které pomáhá rozložit problém na menší opakované části. Dá se říct, že AOP se zaměřuje na modularizaci a zapouzdření průřezových koncernů.

databázových transakcí[13]. Dále nastavuje potřebné hodnoty pro správnou lokalizaci aplikace včetně defaultního nastavení lokalizace, nastavení, kde jsou uloženy lokalizační texty a nastavení názvu parametru v URL, jehož obsah bude měnit lokalizaci. Obsah může být například cs, en, de a další, pokud jsou k nim příslušné lokalizační texty. Lokalizační texty jsou uložené ve složce src/main/resources v balíku *cz.rcjk.messages* s názvy *RCJK_<zkratka_lokalizace>.properties*. Aplikační kontext dále obsahuje definování kontrolérů pro obsluhu chybových hlášení s kódy 403[14] a 404[15] při zpracování požadavků uživatele. Také nastavuje kontrolu autentizace přihlášeného uživatele, odesílání emailů a kontrolu humanity pomocí systému captcha. Všechny kontroly a odesílání emailů nastavuje na vytvořené třídy. V poslední řadě se aplikačnímu kontextu také nastaví umístění souborů s vlastnostmi aplikace (application.properties) a údaji sloužícími k následnému připojení k databázi (jdbc.properties).

```
<!-- lokalizace aplikace – defaultní nastavení -->
<bean
    id="localeResolver"
    class="org.springframework.web.servlet.i18n.
    SessionLocaleResolver">
     <property name="defaultLocale" value="cs" />
</bean>

<!-- lokalizace aplikace – změna lokalizace dle hodnoty
parametru s názvem lang -->
<bean
    class="org.springframework.web.servlet.mvc.support.
    ControllerClassNameHandlerMapping" >
     <property name="interceptors">
          <list>
```

[13] Databázová transakce je množina několika příkazů, obvykle v SQL jazyce, které převedou databázi z jednoho konzistentního stavu do druhého.
[14] Kód chyby 403, neboli přístup odmítnut znamená pokus uživatele o přístup k části webu, ke kterému nemá přístup.
[15] Kód chyby 404, neboli stránka nenalezena znamená pokus uživatele o přístup ke stránce webu, která neexistuje.

```
                <ref bean="localeChangeInterceptor" />
            </list>
        </property>
</bean>
<bean
    id="localeChangeInterceptor"
    class="org.springframework.web.servlet.i18n.
    LocaleChangeInterceptor">
     <property name="paramName" value="lang" />
</bean>

<!-- definuje, kde jsou umístěny lokalizační texty -->
<bean
    id="messageSource"
    class="org.springframework.context.support.
    ResourceBundleMessageSource">
     <property
          name="basename"
          value="cz.rcjk.messages.RCJK"
     />
</bean>
```

5.3.2 ApplicationContext-hibernate.xml

Jak již z názvu tohoto konfiguračního souboru vyplývá, je v něm kompletní konfigurace Hibernatu, tedy nastavení práce s databází. Nejprve je potřeba nastavit správu zdroje dat (dataSource), kde se nastavuje jméno třídy ovladače (driverClassName), URL pro přístup k databázi, uživatelské jméno (username) a heslo (password). Všechny tyto údaje se nastaví prostřednictvím souboru jdbc.properties. Díky tomuto nastavenému zdroji dat se již nastaví správa transakcí (transactionManager) a dále továrna sezení (sessionFactory). Továrna sezení vytváří sezení pro práci s databází. Její pomocí se nastavují všechny doménové třídy reprezentující databázové tabulky a také nastavení hibernatu jako jsou dialect, zda cachovat, zobrazovat SQL do konzole při každém přístupu do databáze apod.

```
<!-- nastavení správy zdroje dat -->
<bean
    id="dataSource"
    class="org.springframework.jdbc.datasource.
    DriverManagerDataSource">
     <property
          name="driverClassName"
          value="${jdbc.driver}"
     />
     <property name="url" value="${jdbc.url}"/>
     <property name="username" value="${jdbc.username}"/>
     <property name="password" value="${jdbc.password}"/>
</bean>
```

5.3.3 ApplicationContext-security.xml

Tento konfigurační soubor definuje zabezpečení webové aplikace. Definuje přesně dle URL, na které stránky má přístup jaká uživatelská role, a zda je přenos stránky šifrován pomocí HTTPS[16]. Také definuje porty pro přenos zabezpečeného obsahu a finálně SQL příkazy pro vybrání uživatele z databáze a zjištění jeho role v systému, na což navazují definice přesných URL.

```
<!--definice jedné URL a k ní povolené uživatelské role->
<intercept-url
    pattern="/cms/module/list"
    access="ROLE_AUTHOR, ROLE_GARANT, ROLE_EDITOR"
    requires-channel="https"
/>
```

[16] HTTPS (Hypertext Transfer Protocol Secure) je nadstavba síťového protokolu HTTP, která umožňuje zabezpečit spojení mezi webovým prohlížečem a webovým serverem před odposloucháváním či podvržením dat a umožňuje též ověřit identitu protistrany. HTTPS používá protokol HTTP, přičemž přenášená data jsou šifrována pomocí SSL nebo TLS a standardní port na straně serveru je 443.

```
<!-- definice SQL dotazů do databáze -->
<authentication-manager>
     <authentication-provider>
          <password-encoder hash="sha" />
          <jdbc-user-service
               data-source-ref="dataSource"
               users-by-username-query=
"select EMAIL, PASSWORD, VALID from USER where EMAIL=?"
               authorities-by-username-query=
"select U.EMAIL, UR.ROLE
 from USER U, MN_USER_USER_ROLE MN, USER_ROLE UR
 where U.ID_USER = MN.FK_USER AND
       MN.FK_USER_ROLE = UR.ID_USER_ROLE AND U.EMAIL=?"
          />
     </authentication-provider>
</authentication-manager>
```

5.3.4 ApplicationContext-serviceAdnDao.xml

Další konfigurační soubor definuje všechny použité service a DAO objekty v aplikaci. Zároveň nastavuje odkazy na potřebné DAO objekty service objektům a tím připravuje service objekty na použití (vložení) do kontrolérů.

```
<bean
     id="userDao"
     class="cz.rcjk.data.dao.impl.UserDaoHibernate">
      <property
          name="sessionFactory" ref="sessionFactory" />
</bean>
<bean
     name="userService"
     class="cz.rcjk.data.service.UserService">
      <property name="userDao" ref="userDao" />
      <property name="userRoleDao" ref="userRoleDao" />
</bean>
```

5.3.5 Cms-servlet.xml

Posledním konfiguračním souborem je dispatcher servlet pro URL požadavky uživatele začínající “cms“ (další soubory jsou pro “public“ a “assessment“). Dispatcher nejprve nastavuje prefix a sufix JSP souborů, na které budou odkazovat kontroléry. Bude tedy stačit napsat pouze jméno JSP stránky místo celé cesty. Dále dispatcher definuje URL mapping. Nejdříve se nastaví měnič lokalizace, který reaguje na každou URL a mění lokalizaci, pokud URL obsahuje výše nastavený název parametru. Dále již přiřazuje k přesným URL adresám kontroléry, které daný požadavek obslouží. Každý kontrolér má přiřazené potřebné service objekty a jiné další objekty, které potřebuje ke své činnosti a které se kontroléru „nasetují“ při nahrání webové aplikace na server. Kontroléry také mohou obsahovat mapování jednotlivých URL přímo na své metody, které obslouží požadavek.

```
<!-- přiřazení kontroléru k URL požadavku -->
<entry key="/module/list" value-ref="moduleController" />

<!-- nastavení potřebných service objektů, objektu pro
uživatelskou autentizaci a rozhodovač (resolver) dle URL
adres, kterou metodu z kontroléru spustit  -->
<bean
    name="moduleController"
    class="cz.rcjk.controller.cms.ModuleController">
     <property name="cmsService" ref="cmsService" />
     <property name="userService" ref="userService" />
     <property
         name="userAuthentication"
         ref="userAuthentication" />
     <property
          name="methodNameResolver"
          ref="moduleActions" />
</bean>
```

```
<bean
    id="moduleActions"
    class="org.springframework.web.servlet.mvc.
    multiaction.PropertiesMethodNameResolver">
     <property name="mappings">
          <props>
               <prop key="/module/list">moduleList</prop>
               <prop key="/module/detail">detail</prop>
               <prop key="/module/hide">hide</prop>
          </props>
     </property>
</bean>
```

5.4 Datová vrstva

Spodní vrstva architektury této webové aplikace obsahuje doménové objekty, které jsou generovány dle tabulek databáze, DAO objekty, které přímo pracují s doménovými objekty a databází. Nad těmito objekty stojí service objekty, které mohou přeposílat jejich volání DAO objektům nebo z dat vydolovat konkrétnější informace a ty předat volajícímu objektu.

5.4.1 Doménové objekty

Doménové objekty přesně korespondují s databázovými tabulkami. O jejich vygenerování se stará plugin Hibernate Tools, který se přidal do SpringSource Tool Suitu. Doménové objekty po vygenerování obsahují všechny atributy jako databázová tabulka a navíc obsahují kolekce ostatních doménových objektů dle vazeb v databázi. Pokud například databáze obsahuje rozkladovou vazbu M:N (promítnutou do tabulkyMN) mezi tabulkou1 a tabulkou2 a vazba (tabulkaMN) neobsahuje žádné další atributy kromě dvou cizích klíčů z obou tabulek, tato tabulka nebude mít ekvivalentní doménový objekt. Hibernate Tools tuto situaci vyřeší přidáním do doménového objektu vzniklého z tabulky1 kolekci s doménovými tabulkami vzniklé z tabulky2. To samé provede analogicky pro tabulku2. Tím jsou doménové objekty propojeny vazbou

M:N. Pokud ovšem vazba (tabulkaMN) obsahuje další atributy, dojde již k vytvoření doménového objektu dle tabulkyMN.

Vygenerování doménových objektů popisuje v bodech následující text. Při každé změně databáze je třeba provést nové vygenerování, přičemž je nutné předem smazat původní vygenerované doménové objekty.

- Připravení databázových tabulek
- Přepnutí do perspektivy Hibernate
- Vybrání z hlavního menu Run, dále Hibernate Code Generation a pak Hibernate Code Generation Configurations a vybrat dříve vytvořenou konfiguraci
- Zaškrtnutí Reverse engineer from JDBC Connection, kliknutí na Setup u reveng.xml a v novém okně vybrání Create new
- Nastavení umístění Hibernate Reverse Engineering filu (reveng.xml) většinou na src/main/resource a pokračovat stiskem tlačítka Next
- Kliknutí na tlačítko Refresh, rozbalení schématu DB2ADMIN, vybrání všech tabulek z něj, kliknutí na Include a ukončení vytváření tlačítkem Finish
- Na záložce Exporters zaškrtnutí Use Java 5 syntax, Generate EJB3 annotations, Domain Code a Hibernate XML Mappings
- Na záložce Refresh zaškrtnutí Refresh resources upon completion a vybrání The project containing the selected resource
- Zahájení generování pomocí tlačítka Run
- Otevřít doménové třídy a do všech doplnit pod anotaci @Id anotaci: @GeneratedValue(strategy = GenerationType.IDENTITY)

- Pokud dochází k novému generování, je potřeba nejprve provést:
  - smazat původní doménové třídy včetně mapovacích xml souborů (*.hbm.xml) a souboru s definicí Hibernate Reverse Engineeringu (hibernate.reveng.xml)
  - Ve view Hibernate Configuration kliknout pravým tlačítkem myši na dříve vytvořenou Hibernate konfiguraci a vybrat Edit Configuration
  - V nově otevřeném okně vybrat záložku Mappings a smazat všechny přítomné xml soubory ze seznamu

Pro účely diplomové práce slouží následující doménové třídy uložené v balíku cz.rcjk.data.domain:

- Module, Submodule, Minilesson, Content
  - Uchovávají studijní obsah
  - Definují obsahy minilekce, minilekce submodulu a submoduly modulu, tím vytváří kompletní přehled učiva
- Language

    - Uchovává jazyk modulu či jazyk gramatické minilekce
- MnSubmoduleUserStudy
    - Uchovává studium studenta
    - Nese odkazy na submodul a studenta, typ studia a další parametry
- MnModuleUserSpecialization
    - Uchovávání specializací autora a garanta dle modulu a jazyka
- User, UserRole
    - Uchovává všechny uživatele a uživatelské role figurující v systému
- Partner
    - Uchovává partnery RCJK - střední školy či jiné subjekty
- Comment
    - Uchovává komentáře minilekcí od studentů nebo lektorů
- Currency
    - Uchovává měny k jednotlivým lokalizacím aplikace a jejich kurzy pro převod z českých korun
- Log, Operation
    - Uchovává všechny možné operace, které může kterýkoli uživatel provést a zaznamenává, kdy a kým byla operace provedena

5.4.2 DAO objekty

DAO objekty úzce spolupracují s Hibernatem, konkrétněji řečeno volají metody Hibernatu, který se stará o získávání, ukládání, aktualizování a mazání záznamů v databázi. Splňuje tak zažité a nutné 4 metody se zkratkou CRUD (create, read, update, delete). DAO objekt je připraven pro každou z doménových tříd uvedených výše. Vždy je nadefinováno rozhraní, které dědí z generické DAO třídy *GenericDao.java* (popsáno níže) a doplňuje metody, které jsou pro danou doménovou třídu specifické. Rozhraní jsou uložena v balíku *cz.rcjk.data.dao*. Každé rozhraní je následně implementováno, přičemž dědí z abstraktní DAO třídy *AbstractDaoHibernate.java* (popsáno níže) a implementuje specifické metody pro danou doménovou třídu. Implementace DAO rozhraní se nachází v balíku *cz.rcjk.data.dao.impl*. Díky tomuto návrhu s implementováním rozhraní nebude problém kdykoli vyměnit implementované třídy za například klasické JDBC[17] místo Hibernatu.

[17] Java Database Connectivity (JDBC) je API, které definuje jednotné rozhraní pro přístup k relačním databázím. JDBC je součástí Javy SE („Standard Edition“) od JDK 1.1. Pro přístup ke konkrétnímu databázovému serveru je potřeba JDBC driver (ovladač), který poskytuje tvůrce databázového serveru.

Rozhraní generické DAO definuje všechny potřebné metody nutné pro práci s databází, přičemž nedefinuje, s jakým doménovým objektem se tyto operace budou provádět. Tím vzniká obecný předpis využitelný pro všechny doménové objekty a použitelný i do dalších aplikací. Až při dědění tohoto rozhraní se určí, s jakým doménovým objektem se budou definované metody provádět. Abstraktní DAO třída implementuje generické DAO, čímž jsou připraveny všechny základní metody pro práci s databází a nemusejí se implementovat v každém jednotlivém DAO objektu pro každý doménový objekt. Opět je abstraktní DAO třída generická, tudíž neurčuje, s jakým doménovým objektem budou operace prováděny. Až při dědění z abstraktní DAO třídy se určí doménový objekt, se kterým budou databázové operace prováděny.

5.4.3 Service objekty

Jsou to objekty, které leží nad DAO objekty a tvoří mezivrstvu mezi datovou vrstvou a aplikační logikou (tedy kontroléry). Dochází v nich k ošetření chybových stavů, jako jsou špatně zadané parametry, které se předávají dále do DAO objektů. V rámci diplomové práce vznikly tři service objekty, které obsluhují vícero DAO objektů. Service objekty jsou umístěny v balíku *cz.rcjk.data.service*.

- CMS service
  - Spolupracuje s DAO objekty, které obsluhují následující tabulky
    - CONTENT, MINILESSON, SUBMODULE, MODULE, LANGUAGE, MN_SUBMODULE_USER_STUDY, MN_MODULE_USER_SPECIALIZATION
- CMS Tool service
  - Spolupracuje s DAO objekty, které obsluhují následující tabulky
    - LANGUAGE, PARTNER, COMMENT, CURRENCY, LOG, OPERATION
- User service
  - Spolupracuje s DAO objekty, které obsluhují následující tabulky
    - USER, USER_ROLE

5.5 Aplikační vrstva

Aplikační vrstva je „mozkem“ celé aplikace. Přijímá požadavky od uživatele a na základě jejich vyhodnocení požádá datovou vrstvu o informace a ty následně odešle k zobrazení JSP stránkám. Tyto třídy přijímající a zpracovávající požadavky od uživatele se nazývají kontroléry a patří do MVC modelu Springu. Dále sem patří

podpůrné, neboli pomocné třídy, např. třída pro odesílání emailu či pro zakódování hesla pomocí hashe SHA1, jež budou rozebrány dále. Také jsou zde komparátory, které slouží pro správné seřazení kolekcí s daty (doménovými objekty) a v poslední řadě je zde balík s výjimkami.

5.5.1 Kontroléry

Přijímají požadavky od uživatele a na základě požadavků odesílají data předané z datové vrstvy k zobrazení do JSP stránek. Zároveň volají metody z datové vrstvy, čímž obstarávají vytváření, editaci, výpis i nastavování na neplatný či mazání doménových objektů z databáze. Kontroléry se nacházejí v balíku *cz.rcjk.controller*.

5.5.1.1 Spravující studijní obsah

Jedná se o kontroléry, které se starají o přidávání, editování, prohlížení, schvalování a nastavování na validní či nevalidní studijní obsah. Také se starají o vybrání aktuálního obsahu minilekce. Nachází se v balíku *cz.rcjk.controller.cms*.

ModuleController dědí od *MultiActionControlleru*, což umožňuje výše zmíněné mapování URL požadavku pomocí dispatcher servletu přesně na danou metodu kontroléru. Obsahuje metody pro prohlížení seznamu modulů, detailu modulu a nastavení na validní či nevalidní modul. Dále je umožněno schvalování modulu. *ModuleCreateController* dědí od *SimpleFormControlleru*, který umožňuje chytře spravovat formulář. Kontrolér dokáže mapovat atributy doménového objektu Modul přesně na políčka formuláře, případně s pomocí editoru (popsán níže). Nový modul se vytváří vždy pro určitý předem zvolený jazyk. Kontrolér umožňuje uživateli vytvořit nový modul, který musí po odeslání formuláře projít validátorem (popsán níže). Pokud jsou hodnoty nevyhovující, zobrazí se na stránce s vyplněným formulářem chybové hlášení upozorňující na špatně vložené hodnoty. Jsou-li hodnoty validní, dojde k vytvoření nového modulu a jeho vložení do databáze. *ModuleEditController* dědí také od *SimpleFormControlleru* a umožňuje uživateli pomocí formuláře editovat již vytvořený modul. Po odeslání upravených hodnot formuláře musí opět dojít ke kontrole hodnot validátorem.

SubmoduleController má analogickou logiku a chování jako *ModuleController*, až na to, že pracuje se submodulem. *SubmoduleCreateController* má opět analogickou logiku jako *ModuleCreateController* a jediným rozdílem je, že dochází k vytváření submodulu. Při vytváření se vybere modul, ke kterému nově vytvářený submodul bude patřit. Hodnoty musí projít validátorem. *SubmoduleEditController* je analogií k *ModuleEditControlleru*. Upravené hodnoty musí projít validátorem. Pro submodul existuje navíc kontrolér s názvem *SubmoduleConnectController*, který se stará o propojování submodulů s minilekcemi. Propojení probíhá vybráním (zaškrtnutím) minilekcí z výpisu minilekcí, které se mají se submodulem propojit. Vypsány jsou všechny minilekce, které přísluší rodičovskému modulu propojovaného submodulu a gramatické minilekce, které přísluší k jazyku rodičovského modulu.

Dále jsou zde kontroléry pro minilekce a gramatické minilekce, které fungují zcela analogicky jako kontroléry pro moduly a submoduly. Vytváření minilekce probíhá vždy pro daný modul a vytváření gramatické minilekce pro daný jazyk. Kontroléry se nazývají *MinilessonController*, *MinilessonCreateController*, *MinilessonEditController*, *MinilessonGrammaticalController*, *MinilessonGrammaticalCreateController* a *MinilessonGrammaticalEditController*.

Poslední skupinou kontrolérů spravujících studijní obsah jsou kontroléry pro správu obsahu minilekcí a gramatických minilekcí. *MinilessonContentController* slouží pro prohlížení všech vytvořených obsahů jedné minilekce, pro zobrazení jednoho detailu obsahu minilekce a pro nastavení aktuálního obsahu, což znamená, že tento obsah uvidí studenti, ostatní jej neuvidí. Dále je umožněno obsahy schvalovat. *MinilessonContentCreateController* obstarává vytváření nového obsahu minilekce. Opět díky třídě *SimpleFormController*, ze které dědí, je správa vytváření poměrně snadná. Zadané hodnoty aspirující na stání se novým obsahem minilekce, musí nejprve projít validátorem obsahů minilekce. Po úspěšné validaci je vytvořen nový obsah minilekce a je uložen do databáze. *MinilessonContentEditController* zajišťuje editaci obsahu minilekce. Po odeslání upravených hodnot musí opět dojít k validaci hodnot, a pokud budou hodnoty validní, dojde k úpravě hodnot i v databázi.

Do této podkapitoly ještě spadá kontrolér *StudyContentCreateController*, který připravuje data pro stránku s odkazy na vytvoření kteréhokoli studijního kontentu. Tedy odešle JSP stránce k zobrazení seznam jazyků, pro výběr jazyka před zahájením

vytváření modulu či gramatické minilekce a také seznam modulů pro vytváření nového submodulu či minilekce. O další zpracování vytváření studijního obsahu se již postarají příslušné kontroléry uvedené výše.

Spadá sem také kontrolér *ApproveStudyContentController*. Tento kontrolér připraví data pro stránku, která zobrazuje dosud neschválený studijní obsah. Ihned na stránce se dá kliknout na schválení, což zpracuje také tento kontrolér a nastaví daný studijní obsah na schváleno. Opět vrátí seznam neschváleného studijního obsahu k zobrazení, v seznamu se však již nenachází právě schválený studijní obsah.

5.5.1.2 Spravující studium

Zde se nalézají dva kontroléry. První z nich, *StudyChooseController*, se zaměřuje na výběr studia již registrovaným studentem. Zároveň také slouží pro přehled studia pro dosud nezaregistrovaného studenta. Kontrolér umožňuje vyhledávání v obsahu studia dle NACE kódu či názvu obsahu studia nebo filtrování modulů dle jazyka. Dále nabízí metody prohlížení detailu modulů, submodulů i minilekcí, ale obsah minilekce není přístupný. Po výběru submodulu pro studium je možno pomocí připravené metody zvolit typ studia a provést platbu. Platba je umožněna pomocí platební brány nebo osobně v centru zadavatele projektu. Pokud se zvolí druhá možnost, je připraven formulář pro administrátora aplikace, který zadá, že studium bylo zaplaceno.

Druhý kontrolér *StudyController* je využíván pro studenty, kteří již studium zaplatili, a tudíž již mohou studovat. Jsou zde metody, které připraví pro zobrazení na JSP stránce submoduly, které student studuje, jejich detail, výpis minilekcí a obsahu minilekce. Student studuje obsah minilekce. Dále je studentovi umožněno zadávat komentáře k jednotlivým minilekcím. Poslední možností studenta v rámci tohoto kontroléru je umožnění přeskočení studijní části studia a přesunutí se rovnou k testům.

5.5.1.3 Spravující uživatele

Tyto kontroléry zajišťují registraci uživatelů, jejich správu administrátorem i uživatelskou editaci svého profilu. Dále žádosti o nové heslo, schvalování nových registrací, správu specializací a další, které budou nyní detailněji rozebrány. Tyto kontroléry se nacházejí v balíku *cz.rcjk.controller.user*.

O registraci nových uživatelů se starají dva kontroléry. První z nich s názvem *UserStudentRegistrationController* obstarává registraci studentů. Kterýkoli neregistrovaný uživatel internetu, který klikne na stránky projektu, má umožněno se registrovat. Po úspěšné registraci (včetně validace všech zadaných hodnot, obzvláště emailu), zašle systém na uvedený email zprávu s vygenerovaným aktivačním kódem, který je zanesený do odkazu, uvedeného ve zprávě. Uživatel po obdržení emailu musí kliknout na uvedený odkaz, čímž potvrdí funkčnost, a že je vlastníkem emailu. V té chvíli se stává potvrzeným uživatelem systému a může se přihlásit.

Druhým kontrolérem je *UserOtherAktorRegistrationController*, který se stará o registraci všech ostatních uživatelských rolí figurujících v systému kromě studenta. Na internetovou stránku s registrací nevede z webu projektu žádný odkaz. Pokud se chce uživatel internetu stát spolupracovníkem zadavatele projektu a zastávat jednu z možných uživatelských rolí (kromě studenta), musí požádat člena zadavatelova týmu o poskytnutí přímého odkazu pro registraci. Následující postup registrace je stejný, jen si uchazeč o registraci navíc vybere uživatelskou roli, kterou chce zastávat. Další postup registrace je stejný až po potvrzení emailu kliknutím na odkaz v příchozí zprávě. Nyní má registrovaný uživatel potvrzený email, ale stále se nemůže přihlásit. Musí vyčkat schválení jeho registrace administrátorem, o čemž bude opět informován emailem. Až po schválení se může přihlásit do systému.

Pro účely ověření správnosti emailu dle ověřovacího kódu slouží kontrolér *EmailVerificationController*. Po kliknutí na odkaz v emailu se zavolá metoda v kontroléru, která pomocí předaného emailu nalezne v databázi konkrétního uživatele včetně jeho ověřovacího kódu. Ten pak porovná s kódem předaným pomocí odkazu a, když souhlasí, nastaví uživatelovi email na validní/ověřený.

Posledním bodem v registraci uživatele (kromě studenta) je schválení administrátorem, o což se stará *ApproveUserController*. Po vybrání odkazu pro schvalování nových uživatelů se administrátorovi vypíší všechny neschválené registrace, které již mají ověřený email. Pokud administrátor klikne na schválit, uživatel se nastaví jako platný.

Kontrolér *UserLoginController* obstarává přihlašování a odhlašování uživatelů do/ze systému. Navazuje na Spring Security, který je zavolán při každém přihlášení i odhlášení. Po úspěšném přihlášení zavolá Spring Security pomocí URL odkazu

metodu *welcome*, která zajistí zobrazení uvítacího textu přihlášeného uživatele pomocí příslušné JSP stránky. Při neúspěšném přihlášení zavolá metodu *loginFailed*, která vyvolá JSP stránku pro neúspěšné přihlášení. Při odhlášení zavolá metodu *logout*.

K zobrazení svého profilu pro každého uživatele je připraven kontrolér *UserProfileController*. Uživatel má možnost si prohlédnout všechny údaje, které zadal při registraci, a při potřebě jejich aktualizace může údaje změnit. Dále kontrolér nabízí změnu hesla. Uživatel je vyzván, aby nejprve zadal stávající heslo a poté dvakrát nové heslo (dvě nová hesla se musí shodovat z důvodu vyloučení překlepu). Dojde k ověření původního hesla, a pokud souhlasí, heslo se změní a zapíše do databáze. Při příštím přihlášení již musí uživatel použít nové heslo. Pro změnu uživatelských údajů slouží kontrolér *UserProfileEditController*, který vyvolá formulář s vyplněnými údaji uživatele. Po odeslání formuláře s upravenými hodnotami, musí nové hodnoty projít validací. Pokud jsou nové hodnoty validní, dojde k aktualizaci uživatelských údajů v databázi.

Pokud uživatel zapomene své heslo, má možnost požádat o vygenerování nového pomocí kontroléru *UserLostPasswordController*. Zde zadá svůj email, načež bude uživateli zobrazena otázka, kterou zadal při registraci. Uživatel zadá odpověď a odešle formulář. Kontrolér vyhodnotí odpověď dle zadané odpovědi při registraci. Pokud odpověď souhlasí, kontrolér vygeneruje nové heslo a v zaheslované podobě pomocí SHA1 uloží nové heslo do databáze. Po přihlášení si uživatel dle výše popsaného postupu můžu heslo změnit.

UserProfileAdministrateController slouží pro administrování uživatele. Nabízí administrátorovi zobrazení seznamu registrovaných uživatelů, vyhledávání dle jména a emailu. Kontrolér dle zadaných kritérií bude filtrovat vybrané uživatele z databáze. Poté, co si administrátor vybere uživatele k úpravě, klikne na jeho detail. Kontrolér si převezme ID uživatele a zobrazí jeho profil. Administrátor nyní může měnit uživateli práva pro tisk, uživatelské role a u autora a garanta může měnit specializaci, kterou mají přidělenou na moduly. U studentů je také možno nastavit, kolik uživatel zaplatil a po převzetí peněz potvrdit odesláním formuláře, že platba byla přijata. Kontrolér danou informaci uloží do databáze ke studiu studenta.

Další kontrolér ve správě uživatele, *StudentController*, slouží pro správu studentů, které vyučuje lektor. Ten má možnost si zobrazit seznam svých studentů a

vyhledávat mezi nimi (stejně jako administrátor). U svých studentů má možnost nastavit, že splnili studium. V takovém případě se daná informace zapíše do databáze a studentovi jsou zpřístupněny závěrečné testy.

LectorAdministrateController zajišťuje administrátorovi správu lektorů. Pokud si student zvolí typ studia, který je zajišťován lektorem, zobrazí se tato informace administrátorovi na stránce s přiřazováním lektorů ke studiu. Zde kontrolér připraví seznam studií, které ještě nemají přiřazeného lektora s výběrem z možných lektorů, kteří mají na studovaný submodul osvědčení. Administrátor vybere z roletkového menu lektora a výběr potvrdí. Kontrolér ještě zkontroluje, zda má lektor osvědčení a přiřadí lektora ke studiu. Pro další stránku kontrolér připraví seznam studií s již přiděleným lektorem a možností změnit lektora. Postup pak bude stejný jako při výběru lektora u nového studia. Poslední možností, kterou kontrolér nabízí, je výměna lektora za lektora. Tato možnost byla vytvořena pro případ onemocnění lektora, či rozvázání smlouvy lektora se zadavatelem projektu. Administrátor dostane výpis lektorů, ve kterém má opět umožněno filtrovat pomocí jména nebo emailu lektora. U každého lektora má pak od kontroléru připravený seznam lektorů, za které je možno stávajícího lektora vyměnit. Po vybrání a potvrzení nového lektora proběhne jeho změna u všech studií, u kterých je veden původní lektor.

Autor i garant mají přístup ke studijnímu obsahu pouze dle jejich specializací na modul a jazyk. Správu svých specializací, kterou si provádějí sami, zajišťuje kontrolér *UserSpecializationController*. Kontrolér připraví pro zobrazení na stránce se specializacemi vždy tři seznamy. První je s přidělenými specializacemi, které si autor nebo garant mohou sami odebírat. Kontrolér poté provede odebrání jejich specializací zapsáním do databáze. Druhý seznam zobrazuje specializace, které čekají na schválení administrátora. Tyto specializace si autor nebo garant mohou také libovolně odebírat. Třetí seznam tvoří moduly, na které autor nebo garant nemají doposud specializaci. Mohou požádat o její přidělení. V té chvíli kontrolér uloží daný modul do zvláštní tabulky v databázi, která slouží právě pro tyto žádosti.

Žádosti o specializace, vznesené autorem nebo garantem, se zobrazují administrátorovi na příslušné stránce a obstarává je kontrolér *ApproveSpecializationController*. Ten vidí všechny žádosti seřazené dle autora žádosti. Má možnost buď žádosti vyhovět, nebo ji zamítnout. Pokud učiní tak anebo

tak, žádost se vždy odebere ze speciální databázové tabulky, a buď se autorovi žádosti připíše specializace, nebo nikoli.

5.5.1.4 Spravující další doménové objekty

Každý modul či minilekce musí mít přidělený jazyk, pro který jsou vytvořeny. Aby bylo možno jazyky spravovat, je zapotřebí dalších kontrolérů. *LanguageController* slouží administrátorovi pro vybrání všech jazyků z databáze a jejich odeslání jich JSP stránce k zobrazení. Dále je možné každý jazyk smazat. Kontrolér nejdříve ověří, zda pro jazyk existuje nějaký modul nebo minilekce. Pokud modul nebo minilekci nenalezne, vymaže daný jazyk z databáze. Ze seznamu jazyků se lze také dostat na úpravu jazyka, kterou obstarává *LanguageEditController*. Přebere ID jazyka a zobrazí vyplněný formulář. Administrátor provede úpravy a odešle formulář. Po úspěšné validaci všech hodnot se údaje o jazyku obnoví v databázi. Je zde samozřejmě kontrolér i pro vytváření nových jazyků s názvem *LanguageCreateController*. Ten zobrazí prázdný formulář pro vytvoření jazyka. Po odeslání vyplněného formuláře a úspěšné validaci zadaných hodnot uloží kontrolér nový jazyk do databáze.

Poslední z trojice kontrolérů spravujících jeden doménový objekt jsou kontroléry pro správu partnerů. Každý registrovaný uživatel může zadat, zda má vztah k partnerovi zadavatele projektu. Pro výběr partnerů z databáze, jejich následného zobrazení pomocí JSP stránky a pro vybrání jednoho partnera dle ID partnera slouží kontrolér *PartnerController*. Dále kontrolér umožňuje nastavení partnera na neplatného či opět platného. Pro vytváření a editování partnera slouží kontroléry (analogicky jako pro jazyky) *PartnerCreateController* a *PartnerEditController*.

Jelikož student může pocházet i z jiné země než ČR, může si přepnout lokalizaci např. do angličtiny a ceny se mu zobrazí v eurech. Je tedy důležité udržovat měnové kurzy pro jednotlivé měny, které jsou spojeny s jednotlivými lokalizacemi. Kontrolér *CurrencyController* zajišťuje administrátorovi možnost vytvářet, vypisovat a upravovat měnové kurzy. Změna měnových kurzů je tedy čistě na administrátorovi. Nad rámec zadání by bylo jako jedno vhodné vylepšení stahovat aktuální kurzy pomocí určité webové služby od České národní banky a průběžně kurzy aktualizovat.

Pro potřeby nalezení všech obrázků, případně médií (audio a video) v adresáři, kam se nahrávají multimediální soubory, sloužící pro přidání do WYSIWYG editoru, je připraven *FilesController*. Obsahuje dvě metody. Jedna nalezne všechny obrázky a vrátí seznam cest a jmen všech nalezených obrázků v čistě textové podobě pro použití v Javascriptu ovládajícím WYSIWYG editor. Druhá metoda obdobně najde všechna multimédia.

BrowsingController slouží pouze pro zjištění přihlášeného uživatele a vyvolání JSP stránky, která zobrazí možnosti procházení studijního obsahu.

IntroductionController také zjišťuje přihlášeného uživatele a zavolá JSP stránku, která zobrazí uvítací text dle uživatelské role přihlášeného uživatele. Nad rámec zadání by tento kontrolér mohl posílat k zobrazení údaje o certifikátech, které brzy vyprší. Tato funkčnost by byla připravena pouze pro uživatelskou roli lektor.

Pokud nastanou při práci s webovou aplikací standardní chyby, jsou pro ně připraveny dva kontroléry. Při zavolání URL, pro kterou není připravena obsluha, server klasicky vyvolá chybu 404 - not found. Díky mapování URL, kde na konci mapovaných kontrolérů je řetězec "/*/**" se pro dané volání spustí kontrolér *Error404Controller*. Ten zavolá JSP stránku, která vypíše chybové hlášení o neexistující stránce a umožní dál s webovou aplikací pracovat pomocí odkazů v menu. Jestliže se uživatel snaží přistoupit k URL, která je připravena pro jinou uživatelskou roli, než má momentálně nastavenou, zaznamená to Spring Security a vyvolá kontrolér *Error403Controller*. Kontrolér předá řízení JSP stránce, která zobrazí hlášení o zakázaném přístupu a umožní dál pracovat s aplikací pouze pomocí odkazů z menu, které jsou uživateli přístupné.

5.5.2 Editory

Editory slouží pro správnou identifikaci doménového objektu ve formulářovém políčku. Najdou využití, například při registraci, kdy si student vybírá z roletkového menu, ke kterému patří partnerovy (např. střední škole) spolupracujícímu se zadavatelem projektu. Díky příslušnému editoru, nastavenému v metodě *initBinder* kontroléru pro registraci studenta, který dědí od *SimpleFormController*, se při vyplňování hodnoty prvku ve formuláři vyplní jednoznačný identifikátor objektu získaný z metody *getAsText* editoru. Při odeslání formuláře a zjišťování vybraného

objektu partnera se systém opět „podívá“ do editoru a dle metody *setAsText(String str)*, které předá hodnotu vybraného políčka, zjistí příslušný doménový objekt, který uloží do příslušného atributu v doménovém objektu User. Editory jsou v balíku *cz.rcjk.controller.editor*.

Připraveny jsou rovněž editory pro převod na text a zpět na doménový objekt pro Language (*LanguageEditor*), Minilesson (*MinilessonEditor*), Module (*ModuleEditor*), Partner (*PartnerEditor*) a Submodule (*SubmoduleEditor*). Je zde také možnost převádět časový údaj na jednoznačný řetězec a zpět pomocí editoru *DateEditor*.

5.5.3 Validátory

Kontrolérům se pomocí konfigurace v dispatcher servletu mohou nastavovat validátory. Nacházejí se v balíku *cz.rcjk.validator*. Příklad nastavení validátoru pro doménový objekt Module.

```
<!-- připojení validátoru odesílaných hodnot -->
<property name="validator" ref="moduleValidator" />

<!-- vytvoření validátoru -->
<bean
     name="moduleValidator"
     class="cz.rcjk.validator.ModuleValidator">
      <property name="cmsService" ref="cmsService" />
</bean>
```

Validátory se starají o validaci hodnot vložených do formuláře po jeho odeslání. Jedná se o kontroléry spravující formuláře dědící od *SimpleFormController*, tudíž validátory pracují přímo s atributy daného doménového objektu. “naceCode“ a “name“ jsou atributy modulu, errors je objekt, kam se automaticky ukládají zjištěné chyby a “cms.nace“ a “cms.name“ jsou kódy pro lokalizační texty, které se zobrazí jako informace uživateli, u jakého atributu vznikla chyba.

```
ValidationUtils.rejectIfEmptyOrWhitespace(errors,
     "naceCode", "cms.nace");
ValidationUtils.rejectIfEmptyOrWhitespace(errors,
     "name", "cms.name");
```

Validátory jsou připraveny pro doménové objekty Module (*ModuleValidator*), Submodule (*SubmoduleValidator*), Minilesson (*MinilessonValidator*), MinilessonContent (*MinilessonContentValidator*), Partner (*PartnerValidator*), User (*UserValidator*) a UserRole (*UserRoleValidator*).

5.5.4 Komparátory

Často je třeba vrátit kolekci seřazenou dle požadavku uživatele či dle nastavení kontroléru. Když probíhá dotaz do databáze, rovnou se výsledek seřadí. Pokud se ovšem kolekce nachází v atributu určitého objektu, vrátí volání atributu neseřazenou kolekci. Pro seřazení jsou připraveny komparátory, které se použijí v metodě *Collection.sort*, kde se jako první atribut zadá kolekce k seřazení a druhý objekt komparátor. Komparátor implementuje generické rozhraní *Comparator<T>*, kde se T nahradí doménovým objektem, který se bude řadit. Každá třída komparátoru pak implementuje metodu *compare(T object1, T object2)*, kde se za objekty budou vkládat objekty daného typu. Komparátory jsou připravené pro řazení modulů dle NACE kódu nebo dle názvu (*ModuleByNaceCodeComparator* a *ModuleByNameComparator*), minilekce dle NACE kódu nebo dle názvu (*MinilessonByNaceCodeComparator* a *MinilessonByNameComparator*) a dále pro řazení uživatelů, a to dle emailu, jména nebo města (*UserByEmailComparator*, *UserByNameComparator* a *UserByCityComparator*). Komparátory se nalézají v balíku *cz.rcjk.util.comparator*.

5.5.5 Pomocné třídy

Mezi pomocné třídy se řadí třída zajišťující posílání emailů *SendEmail*.

Druhou pomocnou třídou je SimpleSHA1. Třída slouží pro zakódování hesla uživatele do podoby, která se uloží do databáze. Tímto postupem je heslo chráněno před přímým přečtením hesla administrátorem v databázi. Kódování probíhá pomocí SHA1 hashovacího algoritmu. Obě třídy jsou uloženy v balíku *cz.rcjk.util*.

5.5.6 Výjimky

Pro správu výjimek v rámci celé webové aplikace byla vytvořena výjimka *RCJKException*, která dědí od *Exception*. Nabízí tradiční seznam čtyř konstruktorů - prázdný, s parametrem pro vložení zprávy s chybou, s parametrem pro vložení původce výjimky nebo s dvěma parametry pro zprávu s chybou i s původcem výjimky. Všechny konstruktory volají konstruktor rodiče s předanými parametry. Z této výjimky dědí další dvě výjimky a to *IllegalArgumentException*, která je využívána v metodách service objektů a je vyvolána při špatně vloženém argumentu metody. Druhou výjimkou je *WrongAttributesException*, která je použita v *AbstractDaoHibernate* a je vyvolána při chybě při práci s databází. Výjimky jsou uloženy v balíku *cz.rcjk.exception*.

5.5.7 Konstanty

Konstantní hodnoty prostupující napříč celou webovou aplikací uchovávají třídy v balíku *cz.rcjk.constants*. První z nich s názvem *CmsContants* uchovává konstanty používané pro název emailu, ze kterého jsou odesílány informační emaily a minimální délka hesla. Hodnoty jsou libovolně měnitelné a další konstanty mohou být doplněny. Další třídou uchovávající konstanty je *UserRoleName*, která v sobě nese názvy uživatelských rolí tak, jak jsou uloženy v databázi a jak jim „rozumí“ Spring Security. V aplikaci je jednodušší používat čistě názvy rolí bez přídavného “ROLE_“. Posledním uchovávatelem konstant je třída *TypeOfStudy*, která pod názvy typů studia (prezenční, kombinované a elearningové) uchovává číselné hodnoty, které se ukládají v databázi.

5.6 Prezentační vrstva

Prezentační vrstva je tvořena JSP stránkami, které se nacházejí ve složce WEB-INF/jsp. JSP stránky umožňují uživateli komunikovat s aplikací pomocí příslušných formulářů, čímž do aplikace vkládá data. Dále si může prohlížet data již v aplikaci zanesená. Prezentační vrstva se dělí na dvě základní části, jsou to JSP stránky pro nepřihlášené a přihlášené uživatele. Jediné dvě společné stránky jsou *header* a *footer*.

Hlavička (*header*) je do každé JSP stránky vložena pomocí tagu include. Tento tag dovoluje i vkládat parametry, čímž je do hlavičky předán title stránky, který je předem uložen do proměnné a získán z lokalizačních textů.

```
<jsp:include page="../header.jsp">
    <jsp:param name="title" value="${pageTitle}" />
</jsp:include>
```

V hlavičce je otevřen tag html a nachází se zde celý tag head, kde jsou nastaveny všechny kaskádové styly a javascripty, které se pro zobrazení a funkčnost JSP stránek využívají. Také je otevřen tag body. V hlavičce jsou připraveny odkazy pro přepínání lokalizace aplikace. Jedná se o parametr "lang", jak bylo výše ukázáno v konfiguraci Springu. Dále hlavička na každé stránce zobrazuje přihlášeného uživatele, pokud je uživatel přihlášen, anebo formulář umožňující se přihlásit. Formulář je odeslán na speciální URL, pomocí které Spring Security zjistí, že se přihlašuje uživatel a ověří jeho přihlašovací údaje. Patička (*footer*) se vkládá opět pomocí tagu include na konec všech JSP stránek. Obsahuje pouze loga zadavatele projektu a uzavírací tagy body a html.

Stránky pro nepřihlášeného uživatele tvoří z většiny statické stránky zobrazující informace o webu, kontakty a smlouvu, kterou je nutné přijmout při registraci. Dále je přístupné ověření certifikátu, registrace studenta i registrace dalších uživatelských rolí. Je zde také zpřístupněn přehled studia, kde si návštěvník může prohlédnout moduly, submoduly a seznam minilekcí kromě jejich obsahu.

Přihlášený uživatel má mnohem více možností. U něj skoro všechny JSP stránky slouží k zobrazování dynamického obsahu, který předávají kontroléry k zobrazení. Může procházet studium (autor, garant, editor, student), nebo procházet uživatele (administrátor) či své studenty (lektor). JSP stránky také umožňují přihlášeným uživatelům vyplňovat a odesílat formuláře a tím interagovat se systémem. V rámci prezentační vrstvy je připraven i javascriptový WYSIWYG, který nabízí správu obsahu minilekce v podobě Microsoft Wordu. Také umožňuje vkládat obrázky, videa a audio nahrávky.

5.7 Anomálie

Největší anomálií či nepochopeným chováním byl aplikační server Websphere od IBM. Než se podařilo ustálit konfiguraci aplikačního kontextu, Websphere často po restartu nenaběhla a jediným řešením bylo přeinstalovat celý aplikační server. Bylo to velice zdlouhavé, ale naštěstí po ustálení aplikačního kontextu a konfigurace aplikace se aplikační server Websphere ustálil a již nebylo více třeba reinstalovat. Další zvláštností tohoto aplikačního serveru je, že již v základu obsahuje 68 projektů, které se při spouštění serveru musí nabootovat, a až poté dojde k samotnému publikování vyvíjené aplikace. To nejenže zdržuje celý vývoj, ale hlavně zabírá hodně operační paměti. Po delší době běhu aplikačního serveru si je schopný brát až 1GB operační paměti.

5.8 Testování aplikace

Aplikace byla optimalizována díky několikanásobným testům. Veškeré vstupy aplikace (webové formuláře) byly otestovány na různé chybné vstupy a tím bylo docíleno odfiltrování problémů s nesprávným vstupem a zabráněno pádu aplikace. Dále byla aplikace několikrát předváděna zákazníkovi, čímž docházelo k odstraňování nechtěného chování aplikace přímo během vývoje. Zákazníkovi byla také nainstalována webová aplikace na jeho počítač v průběhu vývoje, tudíž si zákazník mohl aplikaci v klidu prohlédnout a ozkoušet. Bohužel kvůli náročnosti aplikačního a databázového serveru si zákazník sám aplikaci nevyzkoušel. Nicméně na pravidelných schůzkách byly vždy řešeny nejasnosti v zadání a zákazníkovi byla pokaždé předváděna nová funkčnost.

6 Závěr

Po zhodnocení všech požadavků od jazykové školy a prostudování současných CMS systémů na platformě Java EE bylo rozhodnuto o navržení a implementaci vlastního CMS systému. Nároky na verzování obsahu nebyly příliš veliké, na druhou stranu byly určeny dosti specifické uživatelské role, proto bylo přistoupeno k variantě přípravy celého vlastního systému. Ze začátku bylo množství času věnováno správné konfiguraci Spring, Hibernate a aplikačního serveru, ale po překonání tohoto problému již vývoj postupoval dobře. Obzvláště rozběhnutí lokalizace aplikace činilo značné potíže. Většina problémů byla, dle mého názoru, způsobena aplikačním serverem Websphere, který není příliš využívaný, a řešení anomálií či dohledávání nejrůznějších konfigurací bylo proto složitější. Tyto těžkosti se například na Tomcatu nevyskytují. Vývoj byl zdržován rovněž dlouhou dobou publikování nového kódu na server Websphere.

Přes všechna popsaná úskalí aplikace splňuje zadání a požadavky jazykové školy. Požadavky sice byly upřesňovány na každé schůzce se zákazníkem, přesto bylo těžké dobrat se přesného znění požadavku. Tím se práce dosti protahovala, jelikož sám zákazník neměl jasnou představu o tom, co chce. Požadavek na placení pomocí platební brány nemohl být implementován vůbec, jelikož se zákazník do poslední chvíle nerozhodl, zda použije platební systém od své banky či od jiných poskytovatelů těchto služeb. Je tedy pouze připravené rozhraní, které implementuje třída s jednou metodou vracející informaci o úspěšném zaplacení.

Ještě před ukončením vývoje přestal fungovat Jazz server, na kterém běžela správa programového kódu, byly zde rozpracované požadavky i případy užití do podrobných detailů. Jelikož jsme aplikaci vyvíjeli ve dvou, ovšem každý svou přesně vymezenou část, bylo nutné průběžně spojovat oba programové kódy dohromady. Pád Jazz serveru dodělání aplikace i postupné sepsání textu práce značně zkomplikoval. Spojování tak probíhalo ručně, což přinášelo riziko možnosti zanesení chyby. Nakonec však došlo k odzkoušení celé aplikace, přičemž přehlédnutí chyby při spojování obou programových kódů byla do velké míry minimalizována.

Přehled zkratek

CMS (Content Management System)

Nástroj ke správě informací.

DAO (Data Access Object)

Třídy obsahující metody pro přístup k databázi.

EL (Expression Language)

Je skriptovací jazyk, který umožňuje přístup k atributům Java objektů (JavaBeans)

přes JSP.

HQL (Hibernate Query Language)

Způsob dotazování se na perzistentní objekty spravované Hibernatem. Syntaxe je velice podobná SQL.

HTML (HyperText Markup Language)

Je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web (WWW).

HTTP (Hypertext Transfer Protocol)

Jedná se o internetový protokol určený k výměně dokumentů mezi klientem a webovým serverem.

HTTPS (Hypertext Transfer Protocol Secure)

Je nadstavba síťového protokolu HTTP, která umožňuje zabezpečit spojení mezi webovým prohlížečem a webovým serverem.

J2EE (Java Eterprise Edition)

Java určená k programování distribuovaných aplikací.

JDBC (Java Database Connectivity)

Je API pro Javu, které definuje jednotné rozhraní pro přístup k relačním databázím.

JDK (Java Development Kit)

Obsahuje JRE, překladač a další vývojové nástroje.

JRE (Java Runtime Environment)

Prostředí pro běh programů psaných v Javě. Obsahuje interpret Javy a standardní knihovny.

JSP (Java Server Pages)

Dynamické internetové stránky využívající speciální tagy, ve kterých se skrývá Java kód.

JSTL (JavaServer Pages Standard Tag Library)

Zapouzdřuje jednoduché tagy základních funkcí společných pro mnoho webových aplikací. JSTL má podporu pro společné úkoly, jako jsou iterace, podmínky a další.

ORM (Object-relational mapping)

Objektově relační mapování je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.

RCJK (Regionální centrum jazykových kompetencí)

Název projektu, pro který je vytvářena tato aplikace.

SAX (Simple API for XML parsing)

Nástroj umožňující parsování XML dle zadaných tagů.

SHA1 (Secure Hash Algorithm)

Je hashovací funkce, která vytváří ze vstupních dat výstup (otisk) fixní délky. Používá se u ukládání hesel.

SQL (Structured Query Language)

Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.

URL (Uniform Resource Locator)

Udává přesnou lokaci zdrojů informací (ve smyslu dokument nebo služba) na Internetu.

XML (Extensible Markup Language)

Značkovací jazyk popisující strukturu dat z hlediska věcného obsahu jednotlivých částí.

Zdroje

Rabinovitch I., Royal J., Hoeber M., Hommel S., Zakhour S., Risser T.: *Java 6*, Computer Press, 2007, 536 s.
ISBN 978-8025115756

Kurniawan B.: *Java for the Web with Servlets, JSP and EJB*, Sams, 2002, 992 s.
ISBN 978-0735711952

Walls C.: *Spring in Action*, Manning, 2011, 424 s.
ISBN 978-1935182351

Konda M.: *Just Spring*, O'Reilly Media, 2011, 80 s.
ISBN 978-1449306403

Bauer C., King G.: *Java Persistence with Hibernate*, Manning, 2006, 904 s.
ISBN 978-1932394887

Elliott J., O'Brien T. M., Fowler R.: *Harnessing Hibernate*, O'Reilly Media, 2008, 382 s.
ISBN 978-0596517724

Bernal J.: *Application Architecture for WebSphere*, IBM Press, 2008, 336 s.
ISBN 978-0137129263

Allen G.: *Beginning DB2: From Novice to Professional*, Apress, 2008, 80 s.
ISBN 159059942X

Chong R. F., Hakes I., Ahuja R.: *DB2 Express-C*, DNS a.s., 2009, 287 s.

Odell D.: *JavaScript*, Computer Press, 2010, 368 s.
ISBN 978-8025127339

Přílohy

A Uživatelská dokumentace

Registrace, přihlášení, odhlášení, zapomenuté heslo, editace profilu

Pro registraci do systému student klikne na odkaz *Vytvoření nového účtu* (Obrázek 9) a vyplní příslušný formulář. Registraci ostatních aktorů probíhá na skryté URL, proto nejdříve musí požádat pracovníka od zadavatele projektu o tuto speciální URL (https://<server>/RCJK/public/newAccount/registerOtherAktor), poté registrace probíhá standardním způsobem jako u studenta, akorát si nový aktor volí uživatelskou roli.

Přihlášení do systému po úspěšné registraci probíhá pomocí zobrazeného formuláře v pravém horním rohu internetové stránky (Obrázek 9). Po úspěšném přihlášení se místo formuláře zobrazí údaje o uživateli a také tlačítko pro odhlášení.

Pokud uživatel zapomene své heslo, klikne na odkaz *Zapomenuté heslo* pod formulářem k přihlášení. Je nutné, aby uživatel vyplnil svůj email a zadal správně odpověď na zobrazenou otázku, kterou zadal při registraci. Poté přijde uživateli na email nové vygenerované heslo.

Editace profilu probíhá po úspěšném přihlášení aktora. V levém hlavním menu, které je stále zobrazeno, se nachází odkaz *Můj profil*. Tímto odkazem se uživatel dostane na zobrazení detailu svého profilu, kde může provést editaci svého profilu a změnu hesla.

Obrázek 9 - Přihlašovací formulář pro registrované uživatele

Minilekce, submodul, modul - prohlížení, vytvoření, úprava a zneplatnění

Po přihlášení do systému s uživatelskou rolí *autor*, *garant* nebo *editor* se zobrazí v levém hlavním menu tlačítka pro správu obsahu. U detailu každého studijního obsahu budou autorovi nabídnuty odkazy pro vytváření a úpravu, garantovi pro úpravu, schvalování a nastavování platnosti studijního obsahu a editorovi úprava a schvalování pouze obsahů minilekcí.

Po zvolení odkazu *Prohlížení* se zobrazí dva odkazy, dle kterých si autor vybere, zda si chce prohlížet celý obsah studia (moduly, submoduly či minilekce) nebo pouze gramatické minilekce. Práce s gramatickými minilekcemi probíhá stejně jako práce s/prohlížení celého obsahu. Zvolí se *Seznam modulů*, což zobrazí seznam modulů, dále se zvolí *Detail*, což zobrazí detail modulu. Na stránce s detailem modulu se nachází výpis všech potřebných atributů modulu a zároveň odkazy pro práci s modulem. Je možno modul upravit i zneplatnit. Je zde také možnost vytvořit podobný modul, což způsobí zobrazení formuláře pro vytváření nového modulu s vyplněnými údaji dle zdrojového modulu. Dále jsou zde odkazy pro vytvoření nového submodulu nebo minilekce pro daný modul. Po kliknutí na tyto odkazy dojde k zobrazení formuláře umožňujícího vytvoření nového studijního obsahu. Dále jsou na detailu modulu odkazy pro zobrazení seznamu submodulů nebo minilekcí daného modulu. Seznam minilekcí zobrazí všechny minilekce patřící modulu (seznamu minilekcí se bude věnovat dále).

Po vybrání seznamu submodulů jsou zobrazeny submoduly daného modulu a po kliknutí na *Detail* se zobrazí stránka s detailem submodulu. Zde je opět možné daný submodul upravit i zneplatnit. Opět je zde také možnost vytvoření podobného submodulu. Je zde odkaz *Přiřaď minilekce k submodulu*, který zobrazí stránku s výpisem všech minilekcí modulu. Minilekce přiřazené danému submodulu jsou modře podsvíceny, ostatní minilekce ne. Přiřazování probíhá pomocí zaškrtávání zaškrtávacích políček. Poslední možností u detailu submodulu je zobrazení seznamu minilekcí.

Po vybrání seznamu minilekcí (buď dle modulu nebo dle submodulu) dojde k zobrazení seznamu minilekcí. Po výběru jedné z nich se klikne na *Detail* a zobrazí se detail minilekce. Opět je možné minilekci upravit i zneplatnit, také je možné vytvořit podobnou minilekci. Pokud je již vytvořený nějaký obsah minilekce, dá se dostat na

seznam obsahů minilekcí pomocí tlačítka pro zobrazení seznamu obsahů minilekcí. Pokud ještě žádný obsah neexistuje, je zobrazen odkaz pro vytvoření nového obsahu minilekce. Vytvoření a správu obsahu minilekce popisuje další kapitola.

Autor má ještě možnost kliknout přímo na odkaz *Vytváření*, který mu zobrazí stránku s možností vytvářet přímo submoduly a minilekce pro daný modul bez proklikávání se obsahem studia. Je zde také umožněno vytvářet nový modul a gramatickou minilekci pro vybraný jazyk.

Vytvoření a úprava obsahu minilekce

Na stránce s detailem obsahu minilekce, kam se autor, garant či editor dostanou po proklikání se studijním obsahem dle předchozí kapitoly, se zobrazí tlačítko pro vytvořené obsahu minilekce (pokud ještš žadný obsah neexistuje) nebo tlačítko pro zobrazení seznamu minilekcí. Pokud ještě obsah není vytvořen, má pouze autor právo nový obsah vytvořit. Pokud již nějaký obsah existuje, klikne se na odkaz pro zobrazení seznamu obsahů a poté na detail obsahu, který bude upravován. Pokud bude obsah minilekce upravovat autor nebo garant, dojde k vytvoření nového obsahu a původní obsah zůstane zachován. Každý nový obsah minilekce musí projít schválením garanta a editora a teprve poté může garant nastavit nový obsah minilekce jako platný. Podrobněji tuto problematiku rozebírá kapitola 3.5 Správa obsahu studia. Pro úpravu obsahu slouží WYSIWYG editor, který je přirovnatelný vzhledem i chováním k Microsoft Wordu. Navíc pomocí dvou tlačítek editoru (viz Obrázek 10) lze přidávat obrázky a multimédia.

Obrázek 10 - Tlačítka pro přidání obrázku a multimédií do obsahu minilekce

Schvalování studijního obsahu garantem nebo editorem

Garantovi i editorovi se v levém menu zobrazují odkazy *Schválení*. Po jeho vybrání se zobrazí všechen studijní obsah čekající na schválení od garanta či editore dle přihlášeného uživatele. Garant i editor mohou schvalovat pomocí příslušných tlačítek v procházení studijního obsahu.

Výběr studijního obsahu

Student se zvolením odkazu *Výběr studia* dostává na seznam všech modulů studia. Může vyhledávat v celém studijním obsahu pomocí názvů a NACE kódů nebo si jednoduše studium prohlídnout postupným výběrem modulu, submodulu a následně prohlídnutí jeho minilekcí. Po zvolení submodulu ke studiu pomocí tlačítka *Studovat submodul* se student ocitá na výběru typu studia (viz kapitola 3.4 Typy studia). Zvolí, zda chce klasický certifikát či necertifikovaný certifikát a pokračuje k platbě. Na této stránce zvolí, zda chce studium zaplatit pomocí platební brány či osobně v centru zadavatele projektu. Pokud zvolil platbu přes platební bránu, dojde k platbě ihned a student může zahájit studium. Pokud zvolil osobní platbu, musí se dostavit do centra zadavatele projektu a zaplatit hotově.

Přeskočení studijní části a studium

Student po vybrání z levého menu odkazu *Studium,* se dostává na seznam všech svých zaplacených i nezaplacených studií. Student si u studia může nastavit přeskočení studia pomocí odkazu *Přeskočení studia*, čímž může rovnou začít skládat testy. Pokud vybere odkaz *Detail*, dostává se na detail svého studia a může přejít ke frontální výuce pomocí odkazu *Zobraz seznam minilekcí*, kde si následně zvolí minilekci pro prohlížení-studování. Má také možnost zvolit *Otevřít pracovní list*, čímž se dostává do e-learningové výuky.

Přidání a zneplatnění komentáře u minilekce

Student u zobrazeného detailu minilekce má i zobrazený formulář pro přidání komentáře. Vepíše komentář a zmáčkne tlačítko *Odešli*. Tím je komentář přidán k minilekci a odešle se zároveň email s informací o novém komentáři autorovi bločku.

Autor i editor vidí seznam komentářů u každého detailu minilekce. Oba mají možnost pomocí tlačítka *Schovej* komentář skrýt.

Žádost o změnu specializace

Autor i garant mohou provádět změny své specializace na moduly. Po přihlášení vyberou z levého menu odkaz *Specializace.* Zobrazí se jim stránka s třemi seznamy modulů. První ze seznamů určuje moduly, které má aktor přiděleny, druhý zobrazuje nepřidělené moduly a třetí obsahuje žádosti o přidělení specializace. Aktor si může sám

odebírat již přidělené specializace a také žádosti o specializace. Ze seznamu nepřidělených modulů může vybírat a žádat o přidělení specializace.

Administrátor má v levém menu v kolonce Schvalování odkaz *Specializace*. Po jeho vybrání se mu zobrazí žádostí o schválení. Administrátor má možnost žádost schválit nebo zamítnout.

Autorizace aktorů kromě studentů

Administrátor má v levém menu v kolonce Schvalování odkaz *Uživatelů*. Po jeho vybrání se mu zobrazí seznam registrovaných aktorů (kromě studenta), kteří žádají o schválení registrace a zároveň o přidělení vybrané uživatelské role. Administrátor má možnost žádost schválit nebo zamítnout.

Správa aktorů

Administrátor má v levém menu v kolonce Správa odkaz Uživatelů. Po jeho vybrání se dostává na seznam aktorů figurujících v systému. Dle jména, příjmení nebo emailu vyhledá aktora, kterého chce spravovat. Zobrazí si detail aktora a má možnost profil aktora nastavit na neplatný i zpět na platný, editovat profil a editovat uživatelské role. Editováním uživatelských rolí se má na mysli změnu (přidání či odebrání) uživatelské role aktora. Pokud spravuje autora nebo garanta, má možnost i spravovat jejich specializace (přidávat či odebírat).

Správa lektorů

Administrátor má v levém menu v kolonce Správa odkaz *Lektorů*. Po jeho vybrání se dostává na možnost výběru ze třech odkazů.

Prvním z nich je změna studentova lektora. Administrátor vyhledá chtěného studenta dle jména, příjmení nebo emailu. Zobrazí se mu studentovi studia a jeho přidělený lektor. Zároveň se ke každému studiu zobrazí roletkové menu s výběrem jiného lektora. Takto může administrátor změnit lektora u jednoho studia.

Dalším odkazem je nastavení přiřazení lektora novým studiím. Zde se administrátorovi zobrazí seznam nových studií, které čekají na přidělení lektora. Další postup je stejný jako v předcházejícím odstavci.

Poslední odkazem je výměna lektora za lektora. Administrátor si vyhledá lektora, kterému budou sebrány všechny výuky a budou vyučovány novým lektorem. Tato

možnost je například z důvodu nemoci lektora. Po nalezení chtěného lektora, administrátor z nabídky v roletkovém menu vybere lektora, který převezme všechny studia původního lektora.

Správa partnerů zadavatele projektu

Administrátor má v levém menu v kolonce Správa odkaz *Partnerů*. Po jeho vybrání se dostává na seznam partnerů, kde má možnost partnery vytvářet, upravovat i mazat.

Správa jazyků

Administrátor má v levém menu v kolonce Správa odkaz *Jazyků*. Po jeho vybrání se dostává na seznam jazyků studijního obsahu, kde má možnost jazyky vytvářet, upravovat i mazat. Pokud má jazyk přidělené moduly nebo gramatické minilekce, nejde smazat.

Správa měnových kurzů

Administrátor má v levém menu v kolonce Správa odkaz *Měnových odkazů*. Po jeho vybrání se dostává na seznam měnových kurzů, kde má možnost měnové kurzy vytvářet, upravovat i mazat. Pod seznamem má ještě zobrazeny aktuální kurzy z České národní banky pro urychlení práci.

Zadávání plateb

Administrátor má v levém menu odkaz *Zaplacení studia*. Po jeho vybrání se dostává na seznam studií, u kterých studenti zvolili platbu v hotovosti. Může mezi nimi vyhledávat pomocí jména, příjmení nebo emailu studenta. Po nalezení správného studenta zadá částku, kterou student v hotovosti předává nebo potvrdí již před vyplněnou částku dle zvoleného studia. Poté klikne na *Proveď*. Tím zároveň umožní studentovi zahájit studium.

Správa studentů lektory

Lektor má v levém menu odkaz *Správa studentů*. Jeho vybráním se mu zobrazí stránka se studii, u kterých je lektorem. Mezi studii může vyhledávat dle studenta dle jména, příjmení nebo emailu. Po nalezení chtěného studia, může lektor studium nastavit jako splněné a také si prohlížet seznam plněných testů.