Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Využití technologií JEE při vývoji webové aplikace

Bakalářská práce

Eva Adámková

Školitel: Ing. František Drdák, CSc.

České Budějovice 2013

Adámková, E., 2013: Využití technologií JEE při vývoji webové aplikace.

[Utilization of technology JEE in development of a web application. Bc. Thesis, in Czech.] – 81 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

Tato bakalářská práce se zabývá tvorbou webové aplikace s využitím technologií JEE. Má za úkol zhodnotit výhody a nevýhody použití této technologie.

Abstract:

This bachelor thesis deals with the web application development by using the JEE technology. Its objective is to assess the advantages and disadvantages of using this technology.

Klíčová slova:

JEE, PrimeFaces, Glassfish, webová aplikace

Keywords:

JEE, PrimeFaces, Glassfish, web application

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 26. 4. 2013

Podpis studenta

Poděkování

Ráda bych poděkovala svému vedoucímu bakalářské práce panu Ing. Františkovi Drdákovi, CSc. za ochotu, věnovaný čas a poskytování odborných a podnětných rad.

OBSAH

1. Úvod

S rozvojem internetu se vyvíjí nové technologie pro tvorbu internetových stránek.

Zpočátku se jednalo pouze o stránky statické, kdy se používaly značkovací jazyky sloužící zejména k zobrazování textu. Rozvoj webových aplikací přinesl potřebu generovat obsah stránek dynamicky, tj. zobrazovat výsledky zpracování konkrétního klientského požadavku a jeho parametrů. Na podporu tohoto základního požadavku vznikla a stále vzniká celá řada technologií a frameworků.

Při prohlížení nabídek práce [11], souvisejících s vývojem webových aplikací, je vždy vyžadována schopnost vyvíjet aplikaci pomocí jedné ze tří žádaných technologií – PHP, ASP.NET a Java Enterprise Edition (JEE). Lze se tedy domnívat, že jednou z nejpoužívanějších moderních technologií pro vývoj webové aplikace je právě technologie JEE, pomocí níž byla vyvinuta aplikace předpokládaná pro tvorbu a využití databáze dětských školek v Jihočeském kraji.

Nejen Jihočeský kraj, ale celá Česká republika má problém s nadměrnou obsazeností školek a nalezením té optimální volné. Sice existují stránky ve formě databáze všech školek České republiky, ale nenabízejí možnost vyhledávat podle vyžadovaných služeb, nebo přihlásit dítě do vybrané školky. A to je problém, jímž se částečně zabývá tato bakalářská práce.

První kapitola upřesňuje motivaci tvorby této práce. Dále vysvětluje její cíle a kroky, které je nutno k dosažení těchto cílů podstoupit.

Rodinu technologií JEE, využitou v praktické realizaci webové aplikace, popisuje druhá kapitola. Zabývá se technologiemi potřebnými pro tvorbu komplexních webových aplikací s využitím jazyka Java.

Kapitola třetí se zabývá dalšími technologiemi nezbytnými pro tvorbu diskutované aplikace. Pojednává především o aplikačním serveru a řešení databáze.

Následují kapitoly obsahující vlastní realizaci webové aplikace. Vývoj a implementace zahrnuje několik fází inspirovanými inkrementálně-iteračním modelem, počínaje zadáním, analýzou, návrhem, následuje vlastní implementace a testování. Práce neřeší provozní

podmínky aplikace s ohledem na skutečnost, že cílem práce není vytvoření profesionální webové aplikace, ale pouze ověření vhodnosti a přínosů zvolených technologií.

Poslední kapitola téma této práce uzavírá. Rekapituluje a shrnuje získané zkušenosti.

1.1. Motivace a cíle práce

Tato práce vznikla z několika důvodů. Základní motivací je identifikovat a ověřit vlastnosti technologie JEE, které ji činí velmi oblíbenou na trhu. Na půdě Ústavu aplikované informatiky Jihočeské univerzity v Č. Budějovicích dosud nebyla vytvořena podobná bakalářská práce pomocí JEE. Lze ji tedy využít jako případovou studii pro výukové účely. Dalším důvodem je zájem o tvorbu databáze školek, jež by mohla přispět k centralizaci informací a manipulaci s datovou základnou, týkající se školek. Podobnou problematikou se zabývá webová stránka http://www.materskeskolky.cz/ [9].

1.2. Cíle práce

1) Navrhnout jednoduchou webovou aplikaci pro tvorbu a využití databáze dětských školek v JČ kraji
2) Implementovat jádro této aplikace pomocí technologií JEE
3) Zhodnotit výhody a nevýhody použití technologie na základě zkušeností získaných při vývoji aplikace
4) Napsat testovací scénáře a provést testy

V počáteční fázi byly navrženy use-case diagram a er-diagram k databázi, s níž aplikace pracuje. Vzhledem k rozsahu aplikace není cílem implementace kompletní profesionální aplikace. Jedná se o implementaci pouze vybraných modulů se zaměřením na pořízení a zpracování dat.

Aplikace byla naprogramována s využitím technologií JEE. Během programování aplikace byly získávány poznatky a zkušenosti pro účely vyhodnocení výhod a nevýhod zvolené technologie a implementačního postupu, a následně provedeno testování podle vytvořených testovacích scénářů.

2. Technologie Java Enterprise Edition

2.1. Historie JEE

Sun Microsystem představila v roce 1999 JSP [32]. Zanedlouho vznikla standardizovaná knihovna značek JSTL, zahrnující základní funkce společné pro mnoho JSP aplikací, a výrazový jazyk EL. Se zavedením JSP 2.0 byl EL v roce 2003 začleněn do specifikace JSP.

Paralelně s vývojem JSP se začaly vyvíjet i další frameworky pro tvorbu webových aplikací. Sun Microsystem představila v roce 2004 framework JSF, zaměřující se na uživatelské prostředí a používající JSP ve výchozím nastavení jako základní skriptovací jazyk.

Mezitím se vyvíjely servlety. V září 2005 vydal Sun Microsystems Servlet 2.5. V květnu 2006 zveřejňuje JCP Java specification request (JSR), tzv. požadavek na změnu, který popisuje vývoj technologií JSP a JSF. Sun Microsystems zahrnuje JSP 2.1, konzistentní s JSF 1.2, Servlet 2.5 a EJB 3.0 do své platformy JEE v témže roce.

Platforma JEE je tedy podporována programem Java Community Process, JCP. Jedná se o mechanismus zaštiťující rozvoj standardů, technických specifikací a API pro technologii Java.

Nejnovější verze JEE používaná v současné době je JEE 6 z roku 2009. Zveřejnění technologie JEE 7, spolupracující s technologiemi Cloud a HTML5, je naplánováno na rok 2013.

2.2. Java EE

Cílem JEE [2, 28] je poskytnout sadu dobře integrovaných technologií, významně snižujících náklady a složitost vývoje, a nasazení a správu vícevrstvé, serverové aplikace. Java EE, v návaznosti na platformu Java Standard Edition (JSE), přidává technologie pro vytváření kompletní serverové aplikace, která je:

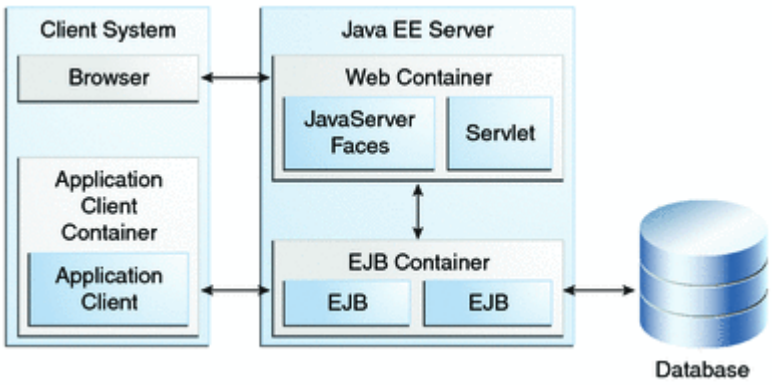
- Robustní
- Stabilní, spolehlivá a bezpečná

- Přenositelná, multiplatformní
- Vysoce výkonná
- Škálovatelná

Navíc je variabilní z pohledu implementace jednotlivých vrstev.

JEE aplikace jsou složeny z komponent, z nichž každá je samostatnou funkční softwarovou jednotkou a komunikuje s ostatními složkami. JEE specifikace definuje především následující komponenty:

- Na straně klienta
  - Tenký klient (včetně appletů), tlustý klient
- Na straně serveru
  - Java Servlet, JavaServer Pages (JSP), JavaServer Faces (JSF)
  - Enterprise JavaBeans (EJB)



Obrázek 1. Dostupnost JEE 6 API ve webovém kontejneru. [28]

Pro podrobnější přehled technologií JEE viz kapitolu 2.2.1 Profily. [2, 28]

2.2.1. *Profily*

Profily JEE [10] jsou tvořeny v procesu JCP v souladu s normou Oracle White Paper a přispívají k pružnosti aplikace, co do rozsahu požadované podpory na AS.

Profily poskytují možnost nasazení standardizovaného souboru technologií JEE platformy, technologií zaměřených na konkrétní třídy aplikací.

V pravém sloupci tabulky, viz Tabulka 1.: Profily platformy JEE 6., lze vidět profil zahrnující kompletní sadu aplikačního rozhraní JEE. Levý sloupec reprezentuje webový

profil JEE, určený pro vývoj moderních webových aplikací. Webový profil poskytuje všechny technologie potřebné pro vývoj webové aplikace s databází. [10]

JAVA EE 6 WEB PROFILE	JAVA EE 6 FULL PLATFORM
Servlet 3.0	Servlet 3.0
JavaServer Pages (JSP) 2.2	JavaServer Pages (JSP) 2.2
Expression Language (EL) 2.2	Expression Language (EL) 2.2
Debugging Support for Other Languages (JSR 045)	Debugging Support for Other Languages (JSR 045)
Standard Tag Library for JavaServer Pages (JSTL) 1.2	Standard Tag Library for JavaServer Pages (JSTL) 1.2
JavaServer Faces (JSF) 2.0	JavaServer Faces (JSF) 2.0
Common Annotations for Java Platform 1.1 (JSR 250)	Common Annotations for Java Platform 1.1 (JSR 250)
EJB Lite	EJB 3.1 EJB 3.1 Lite
Java Transaction API (JTA) 1.1	Java Transaction API (JTA) 1.1
Java Persistence API (JPA) 2.0	Java Persistence API (JPA) 2.0
Bean Validation 1.0	Bean Validation 1.0
Managed Beans 1.0	Managed Beans 1.0
Interceptors 1.1	Interceptors 1.1
Contexts and Dependency Injection for the Java EE Platform 1.0 (JSR 299)	Contexts and Dependency Injection for the Java EE Platform 1.0 (JSR 299) JavaMail 1.4 Connector 1.6 (JAX-WS, JAX-RS 1.1, JAX-RPC 1.1, JAXB 2.2, and JAXR 1.0) Java EE Management 1.1 Java EE Deployment 1.2 JACC 1.3 JASPIC 1.0 JSP Debugging 1.0 Web Services Metadata 2.0 SAAJ 1.3 Web Services 1.3

Tabulka 1.: Profily platformy JEE 6. [10]

V bakalářské práci jsou, z důvodů důležitosti a využití v aplikaci, popsány následující technologie:

- Java Servlet
- Java Server Pages
- Java Server Faces
- Enterprise JavaBean
- Java Persistence API

2.3. Java Servlet

Servlety [28] jsou programy Java platformy, které pro spuštění vyžadují webový, nebo aplikační server.

Servlety poskytují, bez omezení výkonu CGI programů, metody pro vytváření webových aplikací, které jsou nezávislé na platformě. Zároveň mají přístup k celé rodině Java API, včetně rozhraní JDBC API pro přístup k databázím. Zpravidla přes http protokol přijímají a reagují na požadavky webových klientů.

Servlety s sebou přinášejí vlastnosti jazyka Java, tj. přenositelnost, výkon, znuvupoužitelnost a ochranu před spadnutím.

Stávající JEE 6 platforma pracuje se Servlet 3.0. [28]

2.4. JavaServer Pages (JSP)

JSP [4, 28] je rozšířením technologie Java Servlets a poskytuje zjednodušený a rychlý způsob tvorby dynamické webové stránky nezávislé na platformě a serveru. Jedná se o technologii umožňující umístit fragmenty kódu servletu přímo do textového souboru. JSP stránka je textový dokument obsahující 2 typy textu – statická data, vyjádřena v HTML, popř. XML, a elementy JSP zastupující dynamickou část stránky. Uživatelské rozhraní je tak odděleno od generování obsahu, a tudíž mohou návrháři změnit celkový vzhled stránek beze změny dynamického obsahu.

JSP nabízí využití široké škály dynamicky zpracovávaných úloh zahrnujících propojení s databází, užití emailu, nebo řízení toku dat pomocí komponenty JavaBean. Důsledkem je eliminace opakování a zajištění větší kompaktnosti aplikací.

Stávající JEE 6 platforma pracuje, z důvodu kompatibility s předchozími verzemi, s JavaServer Pages 2.2, v nových aplikacích však doporučuje použití technologie JSF. [4, 28]

2.5. JavaServer Faces (JSF)

JSF [28] vychází z technologií JSP a Servlet. Primárním cílem technologie JSF je snadné použití.

Obrázek 2. Vztah mezi technologiemi Java Servlet, JavaServer Pages a JavaServer Faces [28]

Jednoduchost JSF spočívá v možnosti vytvoření vlastního uživatelského rozhraní za pomoci komponent UI napojených na objektech na straně serveru. K veškeré konfiguraci aplikace dochází prostřednictvím anotací, případně pomocí XML souboru faces-config.xml.

Architektura JSF je založena na MVC modelu a jasně definuje oddělení aplikační (business), prezentační a datové vrstvy a zároveň poskytuje možnost připojit ke kódu aplikace prezentační vrstvu. Po domluvě na rozhraní mohou tedy jednotliví vývojáři pracovat odděleně na své části vývojového procesu a potom tyto části spojit do jedné aplikace. Výsledkem je, že se vývojáři mohou zaměřit na práci v oboru své specializace a tím je zajištěn efektivní vývoj aplikace.

Mezi hlavní složky technologie JSF patří:

- Komponenty uživatelského rozhraní UI jsou reprezentovány sadou rozhraní API. Dále jsou k dispozici nástroje k řízení jejich stavu, definování navigace mezi stránkami, zpracování událostí, vstupní validaci, podporu přístupnosti, internacionalizace atp.
- Flexibilní model UI umožňuje vykreslovat komponenty různých verzí HTML, nebo jiných značkovacích jazyků. Objekt Renderer generuje značky pro vykreslení a převedení dat uložených v objektu na zobrazitelný druh.
- Standardní RenderKit generuje značky HTML. RenderKit lze zaměřit na konkrétní zařízení, např. PC, telefon, značkovací jazyk a díky tomu není JSF omezeno na jakékoliv zařízení, protokol, či značkovací jazyk.

Technologie JSF je schopna fungovat jako samostatný webový rámec, stejně jako je možné do beanu připojit kontejner, např. Spring, Prime Faces, Maven.

Stávající JEE 6 platforma pracuje s JSF 2.0 a EL 2.2. [28]

2.5.1. *Facelets*

Facelets je mocný, ale odlehčený jazyk, sloužící k tvorbě prezentační vrstvy JSF.

Původně tuto funkci plnila technologie JSP. JSP však nepodporuje všechny nové funkce dostupné v JSF 2.0 patřící pod platformu JEE 6, a tudíž není schopna poskytnout dostatečně sofistikovanou funkcionalitu. Nekonzistence v podobě různých koncepcí obou technologií vedly k vývoji Facelets. Dnes je zobrazování pomocí JSP považováno za zastaralý postup, který je v JEE 6 podporován pouze z důvodů zpětné kompatibility. Facelets je součástí specifikace JSF a současně upřednostňovanou technologií pro tvorbu prezentační vrstvy.

Facelets využívá ke tvorbě webových stránek XHTML, podporuje jazyk EL a nabízí šablony komponent a stránek, díky nimž může být kód stránky opakovaně použit a rozšířen. JSF, s využitím Facelets, poskytuje bohatou architekturu pro řízení stavu komponent, zpracování dílčích údajů, událostí a ověřování vstupu uživatele. Další výhodou, urychlující vývoj aplikace, je možnost použití implicitní navigace. [28]

2.5.2. *Životní cyklus požadavku faceletu*

Běžné úkoly, jako vykreslení webové stránky do prohlížeče, nebo zpracování příchozích požadavků, jsou prováděny v rámci životních cyklů facelets. Některé z nich jsou vývojáři aplikací utajeny, jiné musí sám spravovat.

JSF spravuje většinu činností životního cyklu automaticky, ale zároveň odhaluje jednotlivé fáze, které si může vývojář upravit. Podrobněji lze najít, např. v [28].

2.5.3. *PrimeFaces*

PrimeFaces je knihovna s mnoha rozšířeními pod licencí open source využívající technologii JSF. PrimeFaces pomáhá vytvářet bohaté uživatelské rozhraní. Nachází se v jednom nezávislém souboru typu jar primefaces-{verze}.jar, který není nutné dále konfigurovat. Kromě PrimeFaces existuje více podobných knihoven, snažících se zjednodušit a zpříjemnit vývojářům implementaci JEE aplikací, př. IceFaces, RichFaces. Knihovna PrimeFaces se ale

v současné době těší největší popularitě, viz kapitolu 5, a je vyvíjena ve spolupráci s vývojáři aplikací. [22]

PrimeFaces zahrnuje následující rozšíření:

- Bohatá sada komponent (HTML editor, Dialog, automatické dokončování, grafy a mnoho dalších)
- Vestavěný Ajax založený na standardním JSF 2.0 Ajax API
- Nulová konfigurace a žádné požadované závislosti
- Podpora AJAX Push Engine (APE) přes websockets
- Mobile UI kit pro tvorbu mobilních webových aplikací pro kapesní zařízení
- Prezentační vrstva pomocí Skinning Frameworku s 30 vestavěnými tématy a podporou pro nástroj Theme designer tool
- Rozsáhlá dokumentace
- Velká a aktivní uživatelská komunita

2.6. Enterprise JavaBean (EJB)

EJB je součástí komponentové architektury pro vývoj a nasazení distribuovaných podnikových aplikací. Aplikace napsané pomocí EJB jsou škálovatelné, transakční a zabezpečené pro více uživatelů. EJB nabízí standardní model pro vytváření komponent na straně serveru, které představují aplikační (business) procesy.

EJB Java třídy implementují business logiku aplikace oddělenou od datové vrstvy, vývojář aplikace se proto může soustředit na řešení vlastní aplikace a službu napojení na datovou vrstvu obstarává kontejner.

Existují 2 typy EJB:

- Session Bean – vykonává úkol pro klienta a provádí úkoly uvnitř serveru.
  Stavový - není sdílen, může mít pouze jednoho klienta
  Bezstavový – je sdílen. Po dokončení metody je dokončen i případ užití. Není tedy potřeba ukládat stavy mezi použitími jednotlivých metod.
- Message-Driven Bean – slouží jako posluchač konkrétního druhu zpráv, jako je například JavaMessage Service API

Umožňuje asynchronní zpracování zpráv. Na rozdíl od Session Beanu, do Message-Driven Beanu nemají klienti přímý přístup přes rozhraní, tzn., Nemohou volat jeho metody.

Součástí EJB byla dříve i persistentní vrstva, tzv. Entity Bean. Od verze EJB 3.0 se oddělila a nazývá se Java Persistence API (JPA).

Stávající JEE 6 platforma pracuje s EJB 3.1. [28]

2.7. Java Persistence API (JPA)

JPA standardizuje objektově–relační mapování pro správu relačních dat v Java aplikacích a zjednodušuje entity persistence model. Zabývá se způsobem, jakým jsou relační data namapována na Java objekty a automatizuje realizaci CRUD operací s těmito daty.

JPA čerpá z idejí předních perzistentních frameworků a API, jako je Hibernate, Oracle TopLink, Java Data Objects (JDO), anebo starší verze EJB kontejneru řízeného perzistencí. [28]

3. Další technologie

3.1. Aplikační server (AS)

Práce byla tvořena ve vývojovém prostředí Netbeans IDE 7.2.1, které implicitně nabízí využití 2 různých serverů.

Jedním z nich je webový kontejner Tomcat [1], jenž je sice schopen pracovat jako server a má menší požadavky na paměť, nicméně je vybaven menším souborem funkcí nutných pro vývoj komplexní aplikace.

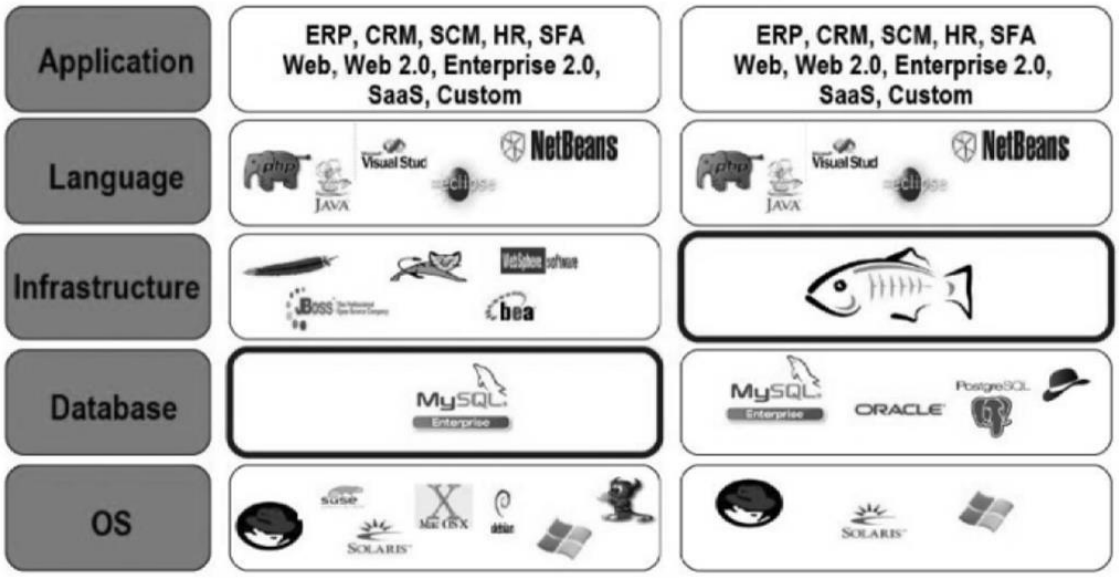
3.1.1. *Glassfish*

Druhým je aplikační server Oracle Glassfish [18]. Server Glassfish je nativním serverem pro platformu JEE. Referenční implementací k JEE 6 je Glassfish v3, slouží především pro předvedení nejnovějších rysů poslední specifikace JEE.

Mezi pozoruhodné rysy Server Oracle Glassfish v3 [10, 24] patří modulární architektura založená na OSGi pro snížení využití zdrojů a rychlý iterativní vývoj. Glassfish 3 nadále poskytuje podporu pro třetí strany dynamických jazyků, např. JRuby, Jython a Groovy.

Při užívání AS Glassfish v integrovaných vývojových prostředích (IDE), př. NetBeans, Eclipse, zahrnuje proces redeployment (opakované umístění aplikace na AS) 3 úkony – editace, uložení a aktualizace prohlížení, a to bez ztráty stavu aplikace. Standardně vyžaduje tento proces 6 časově náročných kroků – editace, uložení kódu, kompilace, zabalení, nasazení a opětovné naplnění dat relace.

Oracle Glassfish Server přináší podporu pro robustní open source vývojová prostředí a umožnuje vzájemnou integraci aplikací, včetně MySQL a předních platforem aplikačního prostředí, viz Obrázek 3.: Oracle GlassFish Server – podpora a integrace napříč technologiemi. [24].



Obrázek 3.: Oracle GlassFish Server – podpora a integrace napříč technologiemi. [24]

Zde odladěnou aplikaci lze přenést do cílového prostředí jiných kompatibilních aplikačních serverů, např. Weblogic, WebSphere, nebo JBoss. Stejným způsobem lze nahradit relační databázi jiným produktem atp.

Aplikační server Glassfish plně vyhovuje pro účely předkládané práce a byl proto použit pro vývoj diskutované aplikace, viz kapitola 5.1.

3.2. Databáze

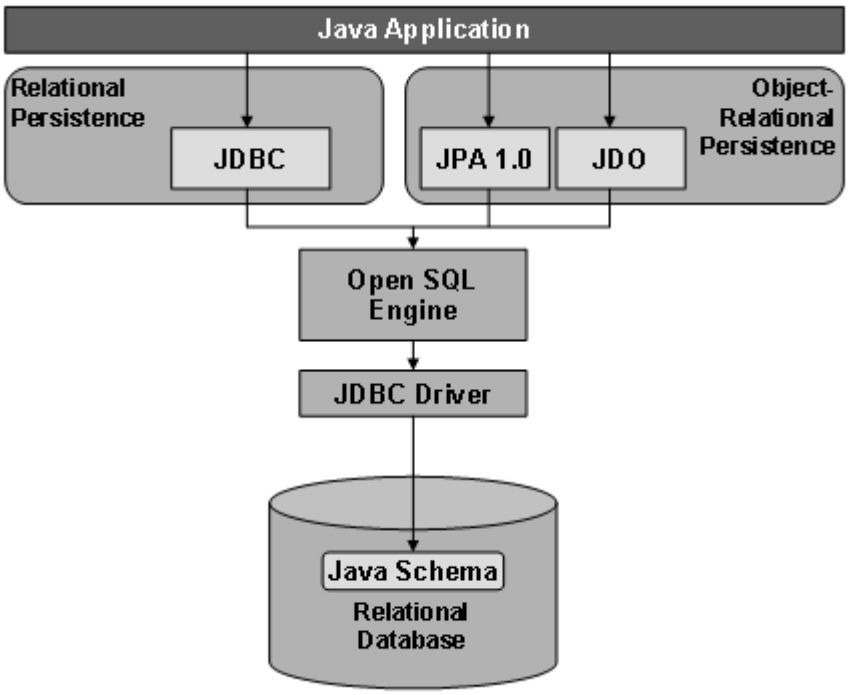
3.2.1. *MySQL*

MySQL [16] je světově nejpopulárnější open source databáze, viz kapitola 5.1, umožňující nákladově efektivní poskytování spolehlivých, vysoce výkonných a škálovatelných databázových aplikací.

3.2.2. *JDBC a JPA*

JDBC API poskytuje unifikovaný přístup k relačním databázím v Javě. Jedná se o nezávislé propojení mezi programovacím jazykem Java a širokou škálou databází. Vývojáři mohou komunikovat s různými relačními databázemi bez jejich bližších znalostí databází, neboť k nim přistupují pomocí universálního aplikačního rozhraní. Užitím stejné Java syntaxe lze připojit data i z heterogenního prostředí databáze. Podmínkou je nutnost naprogramování mapování objektového modelu pro reprezentaci dat k relačnímu modelu.

Frameworky, umožňující objektově-relační mapování, tento problém řeší. Patří mezi ně frameworky JPA (viz kapitola 2.7) a Hibernate. Na obrázku Obrázek 4.: JPA a JDBC [5] je znázorněno relační i objektově-relační mapování.



Obrázek 4.: JPA a JDBC [5]

Lze se domnívat, že technologie JPA může výrazně snížit čas potřebný pro vývoj aplikace, jinak strávený manuální manipulací dat v SQL a JDBC. [19]

3.3. Unified Modeling Language (UML)

Od svého zavedení v roce 1997 je UML [13] důležitou součástí procesu vývoje softwaru. UML specifikace byla počínaje verzí 2.0 rozdělena do dvou komplementárních specifikací – infrastruktury a její nadstavby. V roce 2011 byly vydány specifikace UML 2.4.1.

S UML se během tvorby modelů setkávají především analytici a návrháři aplikací. UML umožňuje modelovat aplikace pomocí standardizované formální syntaxe, a tak sdílet modely s ostatními analytiky a návrháři. Modely jsou srozumitelné i pro zákazníka, čímž je dosaženo snazšího pochopení aspektů aplikace. Jedná se o jazyk vhodný i pro specifikaci, dokumentaci, vizualizaci a stavbu aplikací.

UML nenabízí pouze jeden typ diagramu. Systém není zobrazen jako celek, ale jednotlivý typ diagramu se soustředí na jeden pohled na vyvíjený systém.

Tato práce operuje s následujícími UML diagramy.

Use-case diagram

Use-case diagramy, také diagramy případů užití, reprezentují vnější pohled na systém, zachycují celkovou funkčnost vyvíjené aplikace a vymezují tak jednoznačně rozsah prací.

Activity diagram

Diagramy aktivit jsou objektovou obdobou vývojových diagramů. Diagramy aktivit popisují chování aplikace – sekvenční i paralelní.

Entity-relationship diagram (ERD)

ER-diagramy upřesňují, jak jsou informace v informačním systému ukládány a organizovány. Pomocí diagramů a symbolů se pomocí této metody vytvoří schéma databáze

Class diagram

Class diagram, nebo také diagram tříd, graficky znázorňuje statickou strukturu vyvíjeného systému. Zejména se jedná o třídy, jejich obsah a vztahy mezi nimi.

Pro bližší informace viz [13].

4. Životní cyklus aplikace

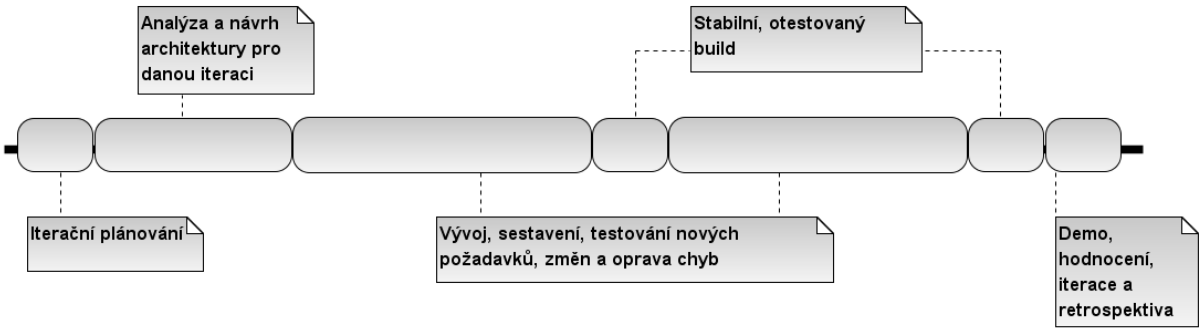
Všechny vyvíjené softwarové produkty procházejí určitým životním cyklem. Aplikace prošla těmito etapami:

- Analýza
- Návrh
- Implementace
- Pozorování
- Testování

U reálných aplikací obsahuje životní cyklus navíc zavedení aplikace do systému, její údržba a vyřazení. Práce se těmito fázemi nezabývá, neboť se jí netýkají.

Existuje značné množství metodik a přístupů pomáhajících určit cestu, jakou se bude software ubírat. Proces tvorby aplikace byl inspirován agilní metodikou, konkrétně se jedná o iterativně-inkrementální metodiku, pro více informací viz [23].

Proces vývoje systému byl tedy rozdělen do iterací, viz Obrázek 5.: Struktura iterace – prováděné aktivity. [23]. Každou iterací vznikne ucelená spustitelná verze systému, lze ji tedy vyzkoušet a otestovat.



Obrázek 5.: Struktura iterace – prováděné aktivity. [23]

5. Analýza

Analýza specifikuje zadání. Je nezanedbatelnou součástí životního cyklu jakékoliv aplikace, podle ní se totiž řídí všechny navazující fáze.

Není všeobecný univerzální návod popisující, jaké metody vybrat pro analýzu webové aplikace. Z tohoto důvodu je nutné zvolit optimální metody, které tvoří logický a souvislý celek.

Diskutovaná bakalářská práce se zaměřila na následující metody:

- Analýza konkurence
- Analýza požadavků na systém
  - Funkční
  - Nefunkční
- Analýza obsahu aplikace
- Use-case diagram

5.1. Analýza konkurence

5.1.1. *Databáze mateřských školek*

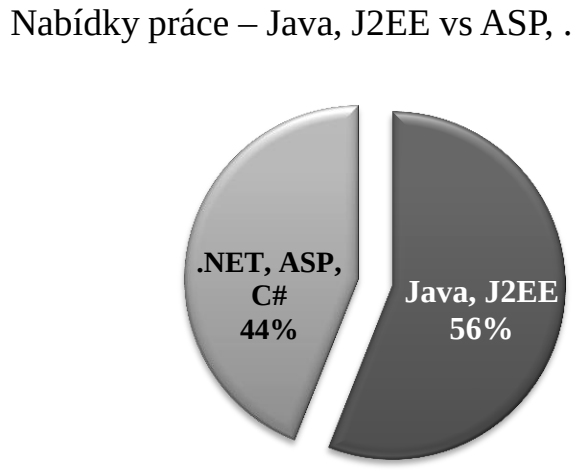
Nejprve bylo nutné si ověřit, zda již nebyl vyvíjen podobný systém dotýkající se diskutované problematiky. Bylo zjištěno, že již existuje internetová stránka disponující databází školek v České republice, viz [9]. Nejedná se však o aplikaci, která by umožnila s informacemi o školkách dále pracovat, tj. zjistit volná místa ve školce, přihlásit dítě do školky, na akci apod.

Aplikace tvořená v rámci této bakalářské práce obsahuje podrobné informace týkající se mateřských školek. Jedná se o bakalářskou práci zaměřenou primárně na použité technologie, nikoliv o práci zabývající se mateřskými školkami. Nebyl podán žádný podnět na tvorbu takové aplikace, a proto se zde v aplikaci nevyskytují reálná data, nýbrž pouze trénovací množina dat.

5.1.2. *Vybrané technologie a jejich trendy*

JEE

Jazyk Java, spolu s technologií JEE, jsou podle serveru Indeed.com [11] nejvíce vyžadovanými technologiemi na trhu práce v r. 2010, viz Obrázek 6.: Podíl nabídek práce – Java, J2EE vs ASP, .NET, C#. [23]

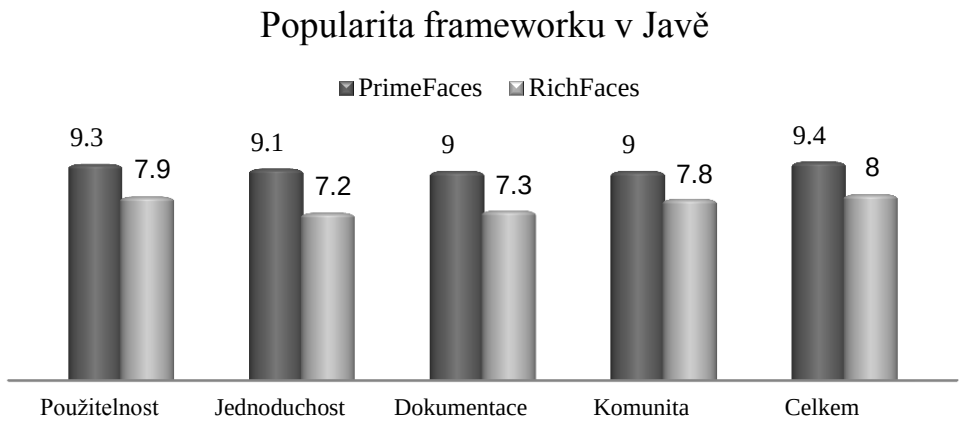


Obrázek 6.: Podíl nabídek práce – Java, J2EE vs ASP, .NET, C#. [23]

PrimeFaces

Výzkum DevRates [3] tvrdí, že PrimeFaces je nejoblíbenějším frameworkem pro vytváření bohatého uživatelského rozhraní v Javě, viz Obrázek 7.: Popularita frameworku v Javě. [3].

Navíc je technologie PrimeFaces šířena s referenčním vývojovým prostředím NetBeans v. 7.2.1.



Obrázek 7.: Popularita frameworku v Javě. [3]

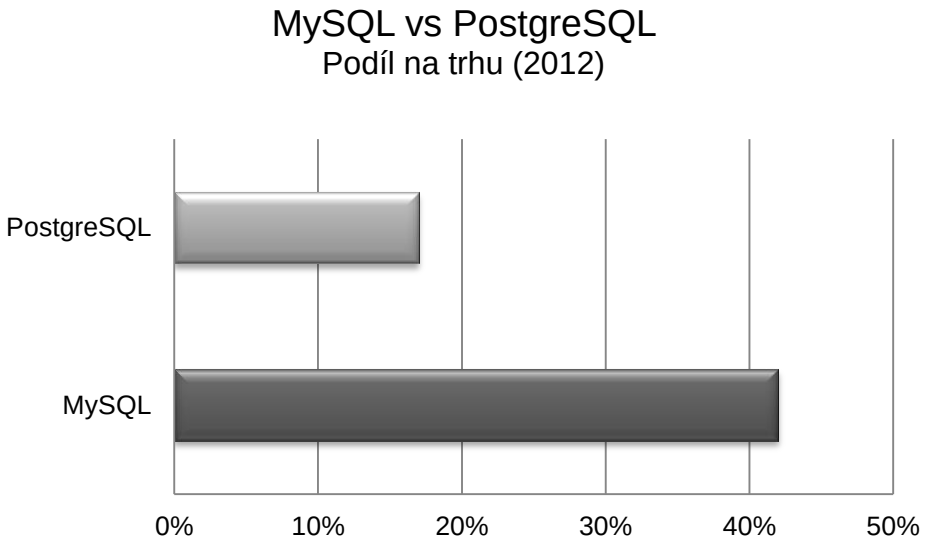
AS Glassfish

Aplikační server Glassfish byl pro tvorbu této aplikace vybrán z několika důvodů. Hlavním argumentem je skutečnost, že je server Glassfish nativním serverem k technologii JEE, a proto podporuje všechny její funkčnosti a je zajištěna kompatibilita.

Zároveň je AS Glassfish integrovaný ve vývojových prostředích IDE, jako jsou NetBeans a Eclipse.

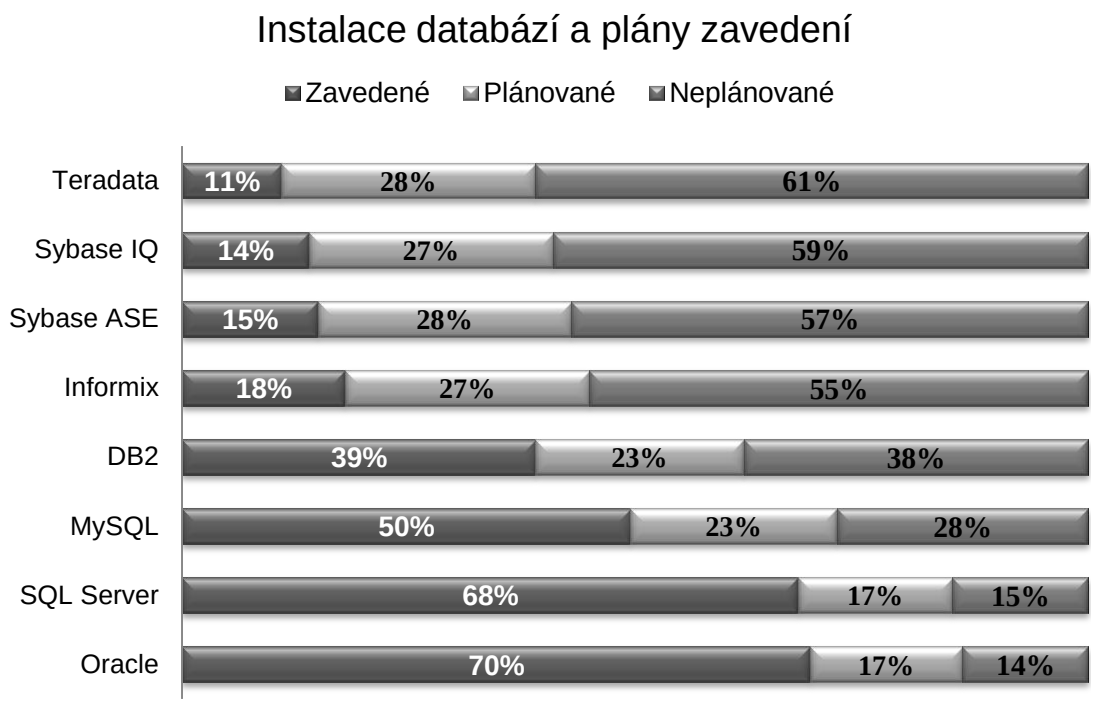
MySQL

Podle serveru Jelastic.com [6] zaujímá databáze MySQL mezi open source databázemi největší podíl na trhu, viz Obrázek 8.: MySQL vs PotgreSQL (Podíl na trhu). [6].



Obrázek 8.: MySQL vs PotgreSQL (Podíl na trhu). [6]

Výzkum Gartner Study (2008) zveřejnil graf, viz Obrázek 9.: Instalace databází a plány zavedení (2008). [16], podle něhož je databáze MySQL jednou z nejvíce využívaných databází na trhu.



Obrázek 9.: Instalace databází a plány zavedení (2008). [16]

5.2. Use-case diagram

5.2.1. *Uvažovaná plná verze*

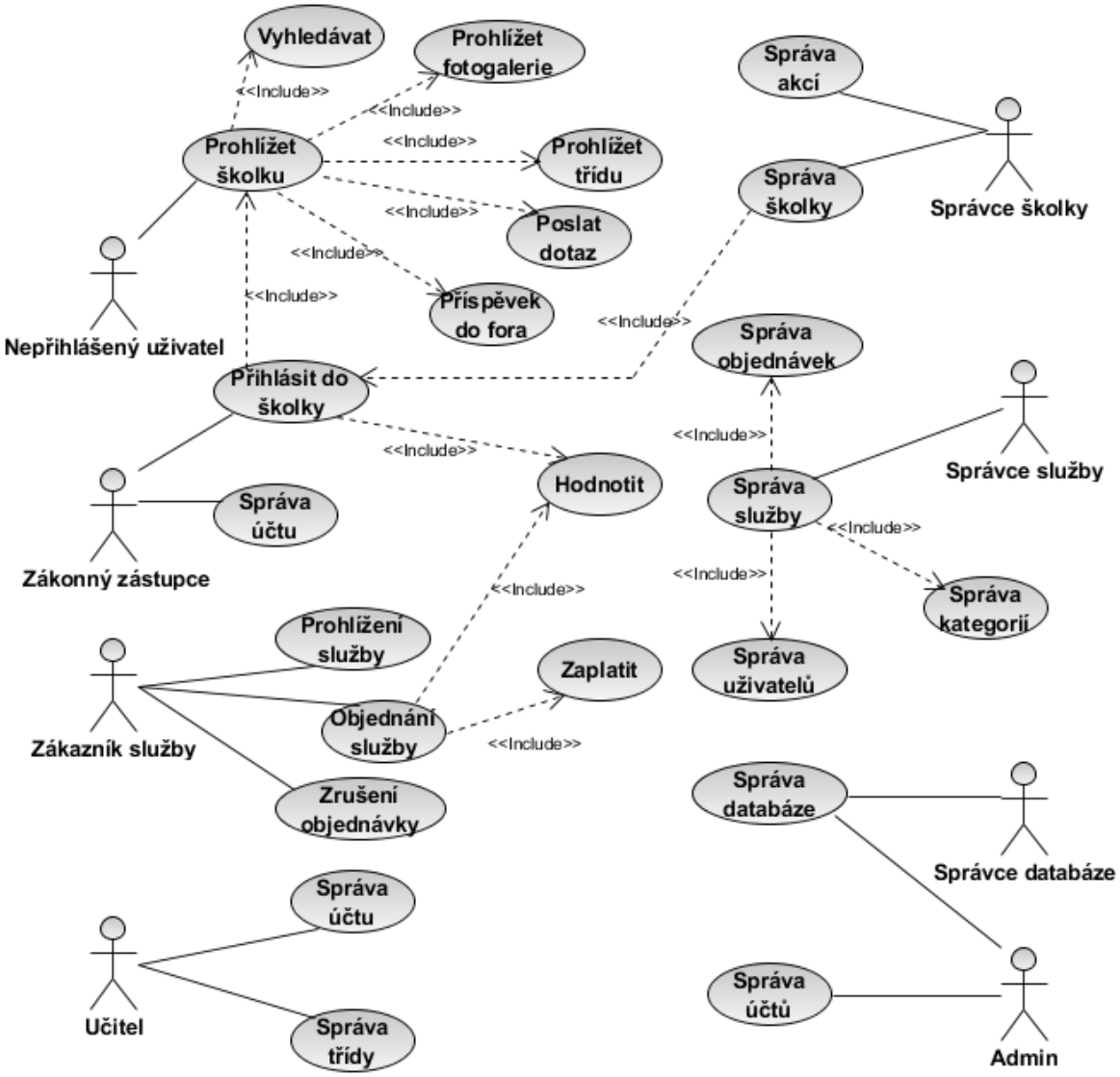
Analýza chování uvažované plné verze aplikace databáze školek je znázorněna na obrázku Obrázek 10.: Use-case (Uvažovaná plná verze). Jedná se o vizi systému spravujícího databázi školek a služeb s nimi spojenými.

Takovou aplikaci může obsluhovat několik typů aktérů. Prvním je nepřihlášený uživatel. Nepřihlášenému uživateli není umožněno měnit obsah databáze, může se však zaregistrovat, přihlásit a zaujímat jednu z dále zmíněných rolí příslušející běžnému uživateli:

- Zákonný zástupce – rodič, opatrovník dítěte
- Zákazník služby – uživatel využívající službu
- Správce školky – osoba pověřená spravovat databázi určité školky
- Učitel
- Správce služby – osoba pověřená spravovat službu

Běžný uživatel nemůže získat roli správce databáze, nebo administrátora, a tudíž ani vykonávat jejich funkci. Úkolem správce databáze je naplnit databázi daty a tato data

spravovat. Administrátor má, zjednodušeně řečeno, neomezená práva k nakládání s celou aplikací a zároveň ověřuje správnost dat, např. ověřuje, zda daná školka opravdu existuje. Z bezpečnostních důvodů není administrátorovi umožněno nahlížet do přihlašovacích údajů.



Obrázek 10.: Use-case (Uvažovaná plná verze)

Pro lepší orientaci v diagramu use-case, viz Obrázek 10.: Use-case (Uvažovaná plná verze), a pro upřesnění rolí jednotlivých uživatelů byla vytvořena tabulka, viz Tabulka 2.: Upřesnění use-case diagramu.

Aktér	č.	Případ užití	Popis	<<include>>
Nepřihlášený uživatel	1	Prohlížet školku	Nepřihlášený uživatel si může školky prohlížet.	V rámci toho je mu umožněno si prohlížet fotografie a třídy, vyhledávat v nich, posílat školce dotazy a psát příspěvky do fóra.
Zákonný zástupce	2	Přihlásit do školky	Zákonný zástupce může dítě do školky přihlásit.	V rámci toho může školku hodnotit.
	3	Správa účtu	Zákonnému zástupci je umožněno spravovat založený účet. *	V rámci správy účtu může účet editovat a smazat.
Správce školky	4	Správa školky	Správce školky spravuje všechny instance databáze patřící pod školku, tj. informace o školce, třídách, službách, učitelích, fotografiích a přihlášených dětech. *	V rámci správy školky je správci školky umožněno přihlašovat dítě do systému.
	5	Správa akcí	Správci školky je umožněno spravovat akce. *	
Učitel	6	Správa účtu	Viz případ užití č. 3.	
	7	Správa třídy	Učiteli je umožněno spravovat třídu, ve které vyučuje. *	
Zákazník služby	8	Prohlížení služby	Zákazníkovi je umožněno si prohlížet nabízené služby a vyhledávat v nich.	
	9	Objednání služby	Zákazníkovi je umožněno si službu objednat.	V rámci této možnosti může zákazník využitou službu hodnotit a pomocí aplikace ji i zaplatit.
	10	Zrušení objednávky	Objednanou službu lze stornovat.	
Správce služby	11	Správa služby	Správce služby spravuje danou službu. *	Správa služby zahrnuje správu objednávek, uživatelů a kategorií.
Správce databáze	12	Správa databáze	Správci databáze je umožněno spravovat celou databázi, až na správu osobních účtů. *	
Administrátor	13	Správa databáze	Viz případ užití č. 12.	
	14	Správa účtů	Administrátorovi je umožněno spravovat celou databázi včetně osobních účtů. *	

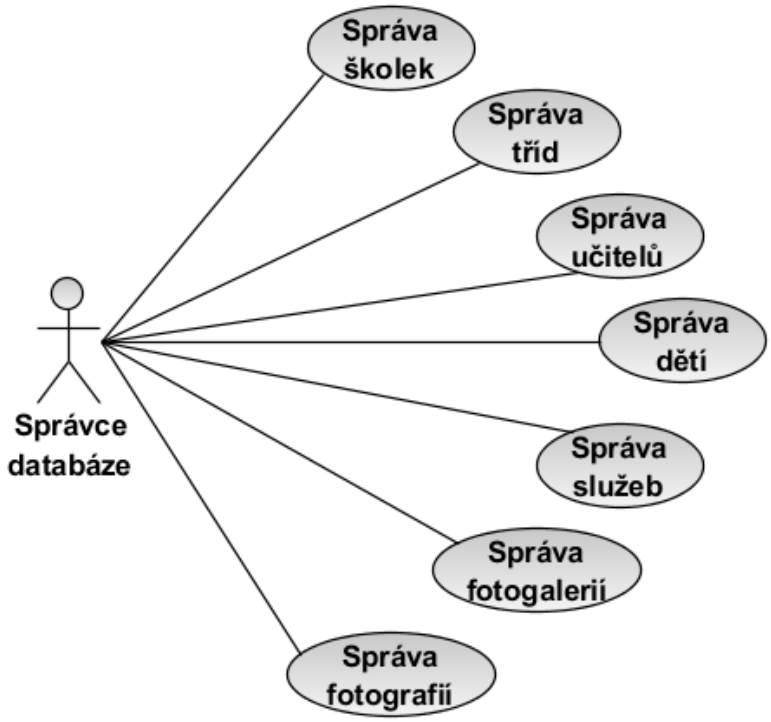
Tabulka 2.: Upřesnění use-case diagramu

* Správou všeobecně je myšleno přidání, úprava, vyhledávání a smazání položky.
Všech uživatelům je umožněno obsluhovat aplikaci podle případu užití č. 1.

5.2.2. *Implementovaný modul*

Pro vlastní implementaci byla vybrána podmnožina funkcí uvažované plné verze. Jedná se o tu část množiny, jež dovoluje pracovat a obsluhovat databázi příslušnému typu uživatele – správci dat. Z těchto důvodů není nutné rozeznávat role uživatelů v systému, a proto není řešena otázka přihlášení a správy rolí.

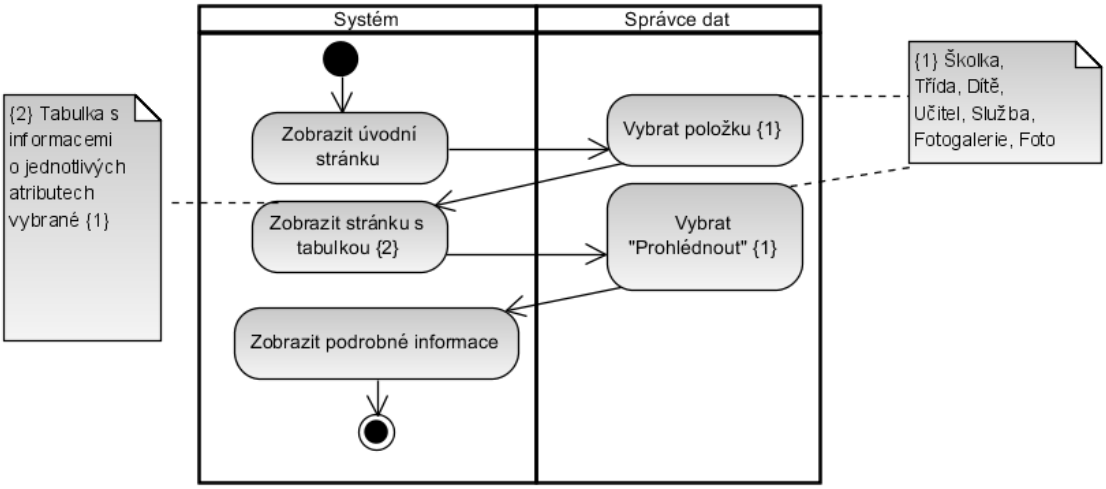
Správce dat si pomocí aplikace může prohlížet jednotlivé prvky databáze, přidávat nové, upravovat je, vyhledávat v nich, nebo je mazat, viz Obrázek 11.: Use-case diagram.



Obrázek 11.: Use-case diagram

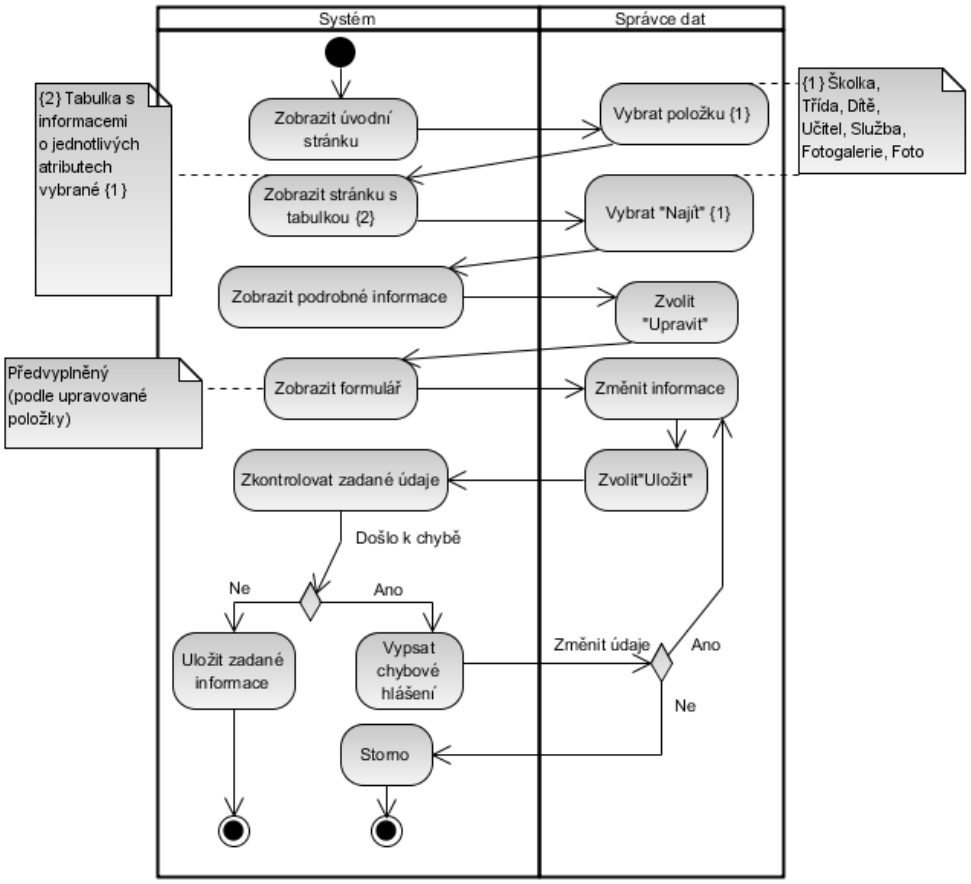
K jednotlivým případům užití byly vytvořeny scénáře případů užití a namodelovány diagramy aktivit.

Prohlížení je znázorněno na obrázku Obrázek 12.: Diagram aktivit - prohlížení. Patří sem případy užití „Prohlížej školku“, „Prohlížej třídu“, „Prohlížej učitele“, „Prohlížej dítě“, „Prohlížej službu“, „Prohlížej fotogalerii“ a „Prohlížej foto“.



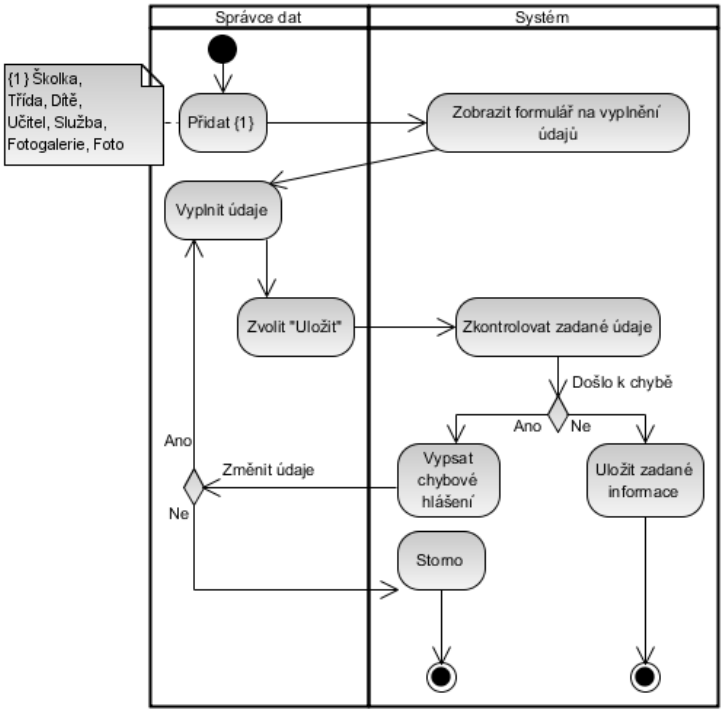
Obrázek 12.: Diagram aktivit - prohlížení

Diagram na obrázku Obrázek 13.: Diagram aktivit – úprava znázorňuje případy užití „Uprav“, tj. „Uprav školku“, „Uprav třídu“, „Uprav učitele“, „Uprav dítě“, „Uprav službu“, „Uprav fotogalerii“ a „Uprav foto“.



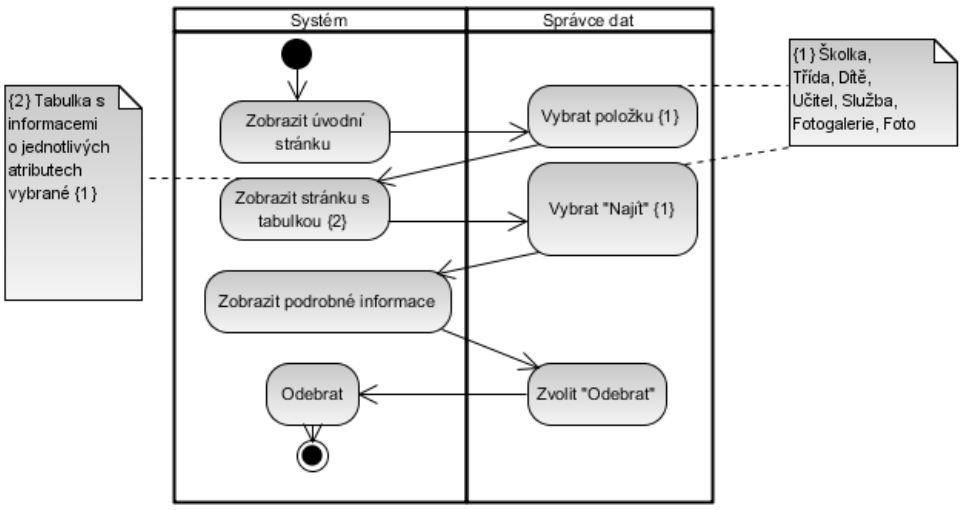
Obrázek 13.: Diagram aktivit – úprava

Diagram na obrázku Obrázek 14.: Diagram aktivit – přidání znázorňuje případy užití „Přidej“, tj. „Přidej školku“, „Přidej třídu“, „Přidej učitele“, „Přidej dítě“, „Přidej službu“, „Přidej fotogalerii“ a „Přidej foto“.



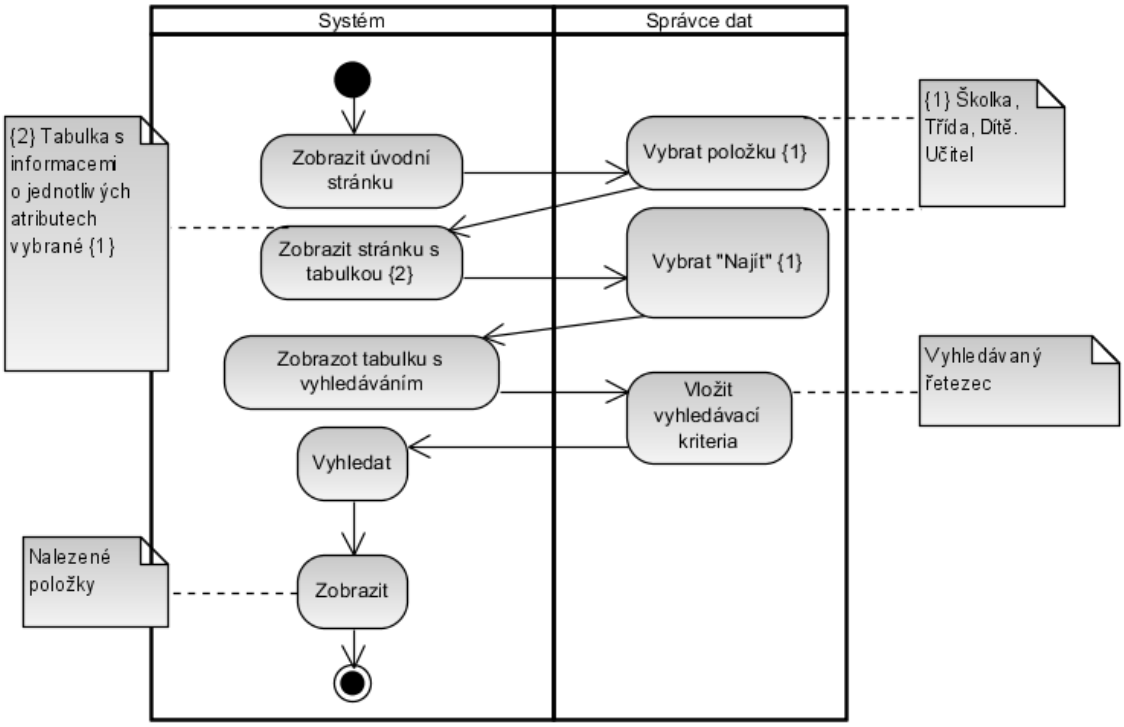
Obrázek 14.: Diagram aktivit – přidání

Diagram na obrázku Obrázek 15.: Diagram aktivit – odebrání znázorňuje případy užití „Odeber“, tj. „Odeber školku“, „Odeber třídu“, „Odeber učitele“, „Odeber dítě“, „Odeber službu“, „Odeber fotogalerii“ a „Odeber foto“.



Obrázek 15.: Diagram aktivit – odebrání

Diagram na obrázku Obrázek 16.: Diagram aktivit – vyhledávání znázorňuje případy užití „Vyhledej“, tj. „Vyhledej školku“, „Vyhledej třídu“, „Vyhledej učitele“ a „Vyhledej dítě“.



Obrázek 16.: Diagram aktivit – vyhledávání

5.3. Analýza požadavků na systém

Analýza požadavků na systém byla provedena podle metodiky uvedené v knize Requirements analysis [7].

5.3.1. *Funkční požadavky na systém*

Funkcionalita

Uvažovaná plná verze

Přístup k aplikaci je umožněn přihlášeným i nepřihlášeným uživatelům. Nepřihlášený uživatel si může pouze vyhledat a prohlédnout veřejně přístupné informace, popř. s vybranou školkou prostřednictvím aplikace komunikovat a přispívat do fóra. Přihlášení uživatelé mají rozšířené kompetence, viz kapitolu 5.2.1 Use-case diagram: Uvažovaná plná verze.

Implementovaný modul
Základní funkčním požadavkem na implementovanou část aplikace je obsluha databáze školek. Je vyžadováno, aby mohl uživatel danou databázi prohlížet, přidávat položky, opravovat je, vyhledávat v nich a mazat je, viz kapitolu 5.2.2 Use-case: Implementovaný modul.

5.3.2. *Nefunkční požadavky na systém*

Uživatelé

Uvažovaná plná verze je navržena pro více typů uživatelů, implementovaný modul pro jeden typ. Pro bližší popis viz kapitolu 5.2.

Uvažovaná plná verze
U všech uživatelů, kromě správců a administrátorů, je očekávána pouze základní znalost práce na počítači. Aplikace je tvořena s ohledem na uživatelskou přívětivost a lze ji ovládat intuitivně, a proto není nutné tyto uživatele školit. V případě potíží lze využít nápovědy, či návodu k použití.

U správců se předpokládá určitá znalost práce s počítačem, optimálně vzdělání se zaměřením na výpočetní technologie. U administrátora je navíc vyžadována praxe na podobné pozici. Tito uživatelé mají povinnost projít odpovídajícím školením.

Legislativní požadavky

Uvažovaná plná verze
Vzhledem k tomu, že analyzovaná aplikace nakládá s osobními údaji svých uživatelů, je třeba postupovat v souladu se zákonem č. 101/2000 Sb., o ochraně osobních údajů.

Zároveň je nutné dodržovat pravidla přístupnosti, protože žádný z občanů ČR nesmí být diskriminován v přístupu k informacím veřejné správy, viz vyhláška č. 64/2008 Sb., o formě uveřejňování informací souvisejících s výkonem veřejné správy prostřednictvím webových stránek pro osoby se zdravotním postižením (vyhláška o přístupnosti) [21].

Dále se aplikace musí řídit zákony, o nichž se zmiňuje odstavec Implementovaný modul.

Implementovaný modul
Mimo jiné je nezbytné, aby se aplikace řídila zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Bezpečnost

Uvažovaná plná verze

Přístup k aplikaci musí být monitorován. Aplikace přijímá opatření proti poškození, ztrátě a loupeži dat. K webovým stránkám se přistupuje prostřednictvím šifrovaného https protokolu, tím je zabezpečená komunikace, a přihlašovací údaje uživatelů aplikace jsou zabezpečeny pomocí hash funkce. Aplikace zároveň vyžaduje pravidelnou změnu hesla.

Implementovaný modul

V aplikaci je zajištěna validace vkládaných dat s ohledem na nebezpečí nechtěného smazání celé databáze.

Rozhraní

Uvažovaná plná verze

Nebyl veden řízený rozhovor se zákazníkem profesionální aplikace, a proto nelze určit rozhraní pro její implementaci.

Nepochybně se ovšem jedná o aplikaci nabízející plnohodnotné využití nejen ve webovém prohlížeči na počítači, ale i např. v mobilním telefonu.

Implementovaný modul

Základním požadavkem práce je využití technologie JEE.

Data

Persistentní data jsou vložena v databázi. K datům mohou uživatelé přistupovat souběžně a je zajištěna snadná přenositelnost a rozšiřitelnost.

Uvažovaná plná verze

Aplikace pracuje s velkou databází školek nacházející se na centrálním serveru. Jedná se o citlivá a aktuální data, proto je nezbytné je zálohovat a komunikaci s aplikací šifrovat.

Implementovaný modul

Aplikace obsluhuje jednoduchou databázi, viz 6.2 Entity-relationship model (ERD). Databáze je naplněna testovací množinou dat.

5.4. Analýza obsahu aplikace

Návrh uživatelského rozhraní aplikace.

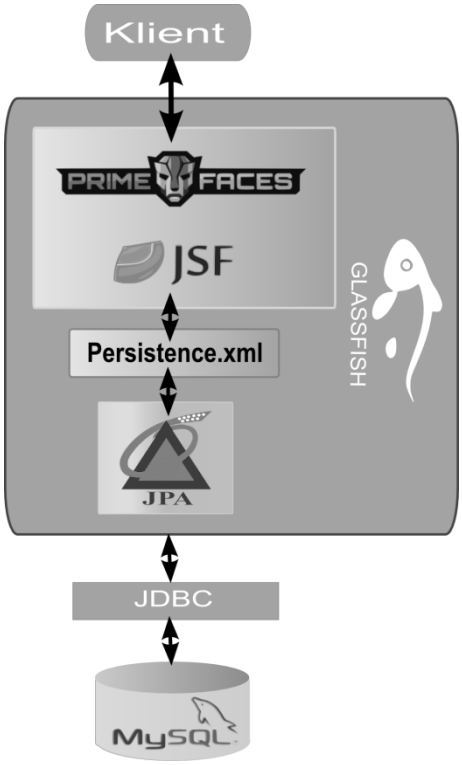
1. Obsah úvodní stránky
   - Rozřazovací menu
2. Obsah dílčích stránek
   - Menu
   - Informace k právě otevřené položce
3. Obsah menu
   - Odkaz na úvodní stránku
   - Odkazy na jednotlivé prvky databáze
     - Školka, třída, učitel, dítě, služba, fotogalerie

6. Návrh

Kapitola se zabývá návrhem aplikace, konkrétně její architekturou, využitým rozhraním a databází. Kromě toho je pomocí drátového modelu navržena struktura webové stránky aplikace.

6.1. Architektura aplikace

Na základě analýzy, kde byly vybrány technologie pro další tvorbu aplikace, byla namodelována architektura aplikace, viz Obrázek 17.: Model architektury aplikace.

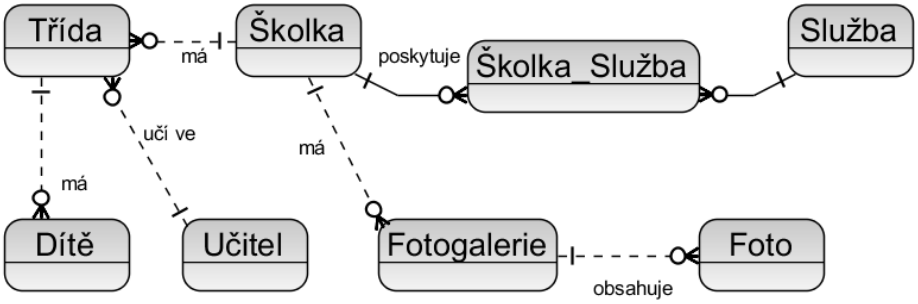


Obrázek 17.: Model architektury aplikace

Na úrovni databáze byla použita technologie MySQL. Datová vrstva je k databázi připojena pomocí JDBC a je realizována technologií JPA. V rámci MVC modelu se jedná o vrstvu Model (M). Následuje Controller (C), nebo také aplikační vrstva, která je realizována pomocí EJB beanů a JSF. Prezentační vrstva, podle MVC modelu View (V), je postavena na frameworku PrimeFaces. Klient vidí standardní webovou stránku napsanou jazyky HTML a CSS.

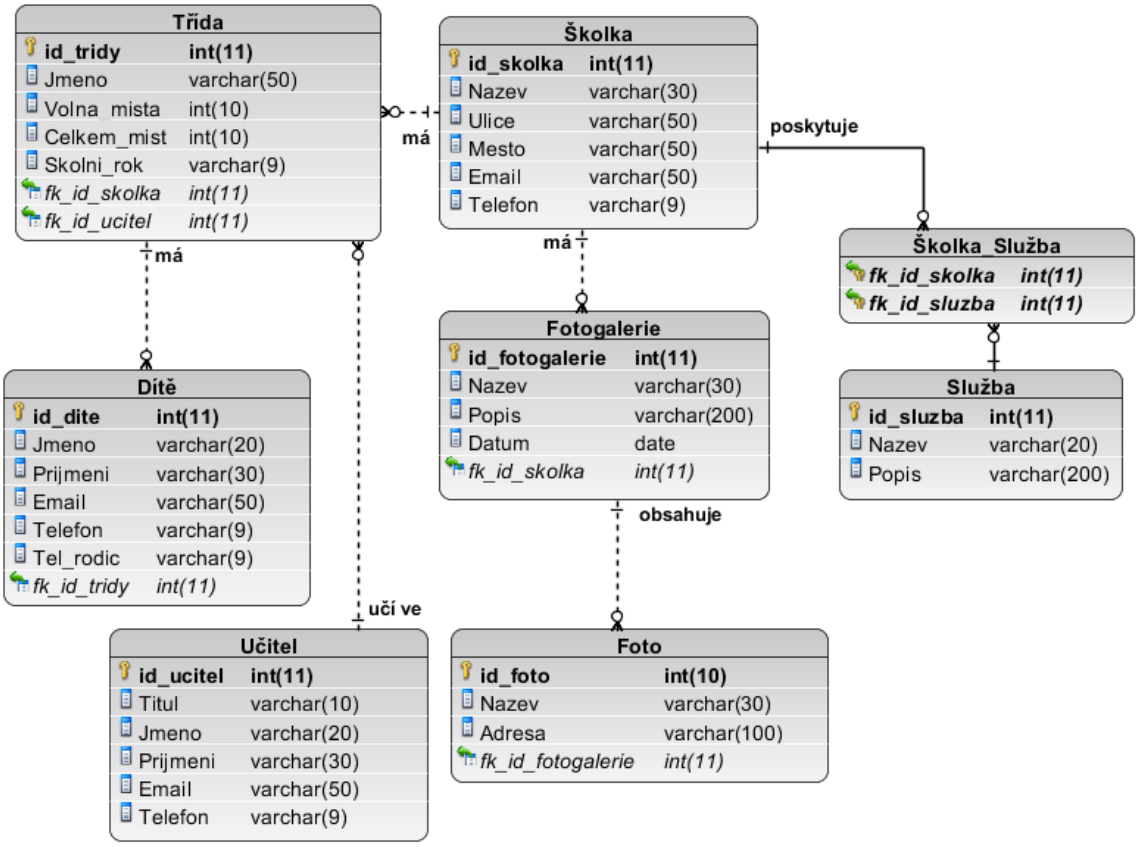
6.2. Entity-relationship model (ERD)

V průběhu tvorby návrhu byl nejprve namodelován zjednodušený er-diagram, sloužící pro lepší orientaci v databázi, viz Obrázek 18.: ER-diagram k databázi školek, zjednodušená verze.



Obrázek 18.: ER-diagram k databázi školek, zjednodušená verze

Následoval ucelený a podrobný model, viz Obrázek 19. V podrobném návrhu databáze lze navíc vidět jednotlivé atributy položek, zároveň jsou mezi entitami upřesněny vztahy.



Obrázek 19.: ERD – podrobná architektura databáze

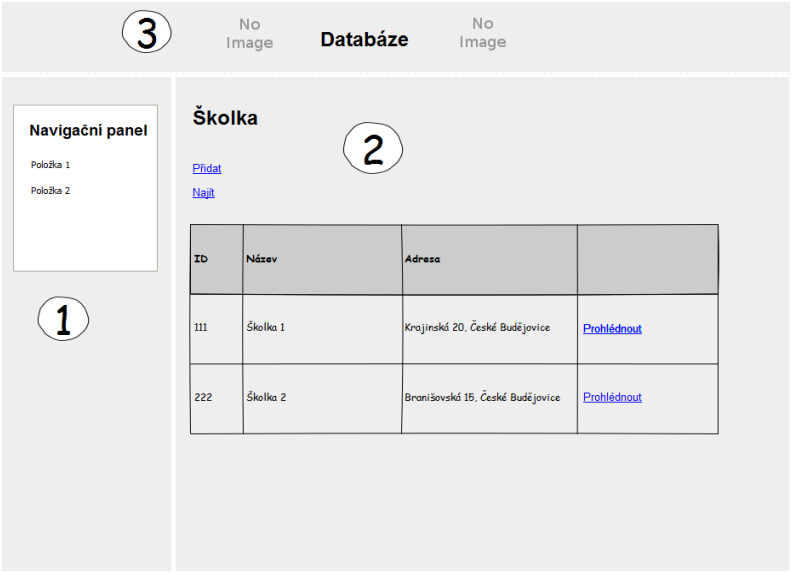
6.2.1. *Možná rozšíření databáze*

Technologie JPA umožňuje provádět v databázi jednoduše změny. Při zavedení více typů uživatelů systému a pro účely implementace uživatelských rolí by byla úprava databáze nezbytná – minimálně pro instalaci funkcí aplikace, které jsou s takovou realizací úzce propojeny.

Jedním z rozšíření by bylo přidání přihlašovacích údajů jednotlivých aktérů do databáze, dalším realizace komunikace mezi rodiči a školkami, možnost přihlašování dětí do školek, informování o akcích konaných školkou atp. V návaznosti na služby lze upravit databázi pro realizaci jednotlivých služeb, např. letní školka a akce s ní spojené, jídelna s možností objednání jídel apod.

6.3. Drátový model (Wireframe)

Během návrhu byl naskicován tzv. Wireframe, také drátový model, nebo blueprint. Wireframe odhaluje základní strukturu webové stránky, její uspořádání, a definuje její obsah. Pro bližší informace viz [3].



Obrázek 20.: Wireframe aplikace

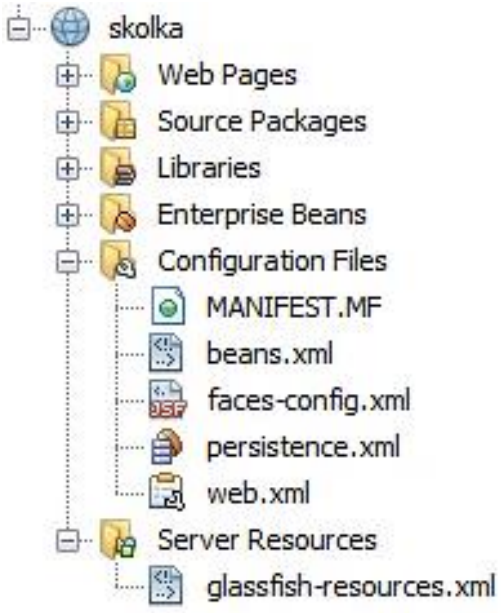
Wireframe na obrázku Obrázek 20.: Wireframe aplikace znázorňuje zjednodušený pohled na webovou stránku aplikace. Stránka je rozčleněna na 3 části. První částí je navigační panel, kde se nachází menu. Prostor s číslem 2 reprezentuje hlavní obsah stránek. Zde se zobrazují informace z databáze. Nadpis se nachází v oblasti č. 3.

7. Implementace

7.1. Základní konfigurace

Práce byla implementována ve vývojovém prostředí NetBeans IDE 7.2.1. Nejprve byl založen nový projekt ve variantě jednoduché JSF aplikace a přidány knihovny, např. k frameworku PrimeFaces.

V konfiguračních souborech byly nastaveny parametry pro správný běh aplikace. Názvy konfiguračních souborů lze na obrázku Obrázek 21.: Konfigurační soubory vidět ve složkách *Configuration Files* a *Server Resources*.



Obrázek 21.: Konfigurační soubory

7.1.1. *Konfigurace JSF*

Pro správnou funkčnost aplikace ji bylo nutné nakonfigurovat. Konfigurace aplikace v JSF je obsažena ve dvou souborech – web.xml a faces-config.xml.

Faces-config.xml

Konfigurace týkající se, např. přidávání deklarace spravovaných beanů a bundle a doplnění navigačních pravidel, se nacházejí v konfiguračním souboru *faces-config.xml*.

V aplikaci nastavuje tento soubor především `resource-bundle`. Resource bundle pomáhá oddělovat zdrojový kód od textu, viz kapitolu 7.4 Resource Bundle. Nastavení diskutovaného konfiguračního souboru ukazuje Příklad 1.: Konfigurační soubor faces-config.xml.

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="2.1"
      xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_2_1.xsd">

<application>
      <resource-bundle>
            <base-name>/Bundle</base-name>
            <var>bundle</var>
      </resource-bundle>
      <resource-bundle>
            <base-name>zprava.Bundle</base-name>
            <var>bundle</var>
      </resource-bundle>
      <message-bundle>
            zprava.Moje_zpravy
      </message-bundle>
</application>
</faces-config>
```

Příklad 1.: Konfigurační soubor faces-config.xml

Web.xml

Konfigurační soubor web.xml popisuje webové nasazení. V rámci této kapitoly jsou popsány parametry a atributy použité v naimplementovaném modulu.

V prvním parametru `context-param`, viz Příklad 2.: Konfigurace JSF, je tzv. `PROJECT-STAGE` (fáze projektu), pomocí které je určeno, ve které roli současného nasazení se projekt vyskytuje. Mezi platné hodnoty patří `Development`, `Production`, `SystemTest` a `UnitTest`. Aplikace se nachází ve fázi `Development`, která ve spuštěné aplikaci vypisuje nápovědu a varování, a tím přispívá k odhalení chyb a optimalizaci systému [28].

Následuje parametr `primefaces.THEME` ustanovuje tzv. téma určující vzhled aplikace. Knihovna komponent PrimeFaces nabízí 37 vlastních témat [22], která lze libovolně upravovat. Pro aplikaci bylo vybráno téma `hot-sneaks`.

Parametr `STATE_SAVING_METHOD` určuje, kam bude uložen stav. Jsou 2 možnosti – server a client. Pokud je stav uložen na straně klienta, je vykreslen do skrytého pole na stránce. Výchozím nastavením je `server`.

Aplikace JSF [28] může spustit aplikaci a zobrazit úvodní webovou stránku `index.xhtml`, dle `<welcome-file>` v Příklad 2.: Konfigurace JSF – soubor web.xml, až potom, co webový server vyvolá Servlet instanci. V konfiguračním souboru je zanecháno defaultní nastavení servletu a je jím zajištěno vyvolání Servlet instance. Servlet-mapping určuje tzv. mapování prefixu. Atribut `servlet-mapping` slouží pro lokalizaci stránek na AS.

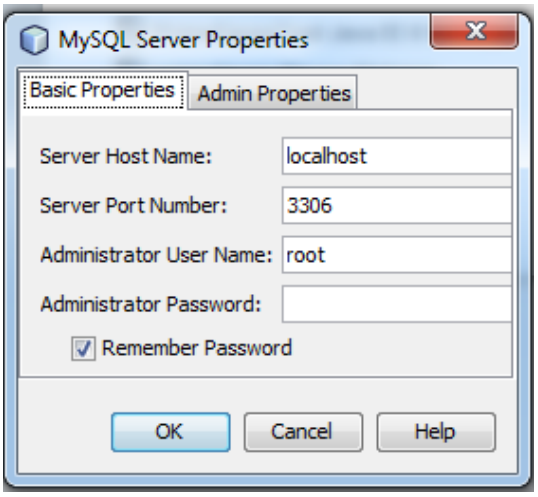
```
<!– fáze projektu -->
<context-param>
      <param-name>javax.faces.PROJECT_STAGE</param-name>
      <param-value>Development</param-value>
</context-param>
<!-- konfigurace PrimeFaces -->
<context-param>
      <param-name>primefaces.THEME</param-name>
      <param-value>hot-sneaks</param-value>
</context-param>
<context-param>
      <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
      <param-value>server</param-value>
</context-param>
<!-- konfigurace Servletu -->
<servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>
            javax.faces.webapp.FacesServlet
      </servlet-class>
      <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>/faces/*</url-pattern>
</servlet-mapping>

<!– konfigurace úvodní strany -->
<welcome-file-list>
      <welcome-file>faces/index.xhtml</welcome-file>
</welcome-file-list>
```

Příklad 2.: Konfigurace JSF – soubor web.xml

7.1.2. *Vytvoření a připojení databáze*

Podle ERD, viz Obrázek 19.: ERD – podrobná architektura databáze, byla v programu phpMyAdmin (program EasyPHP 12.1) vytvořena databáze s kódováním UTF-8.

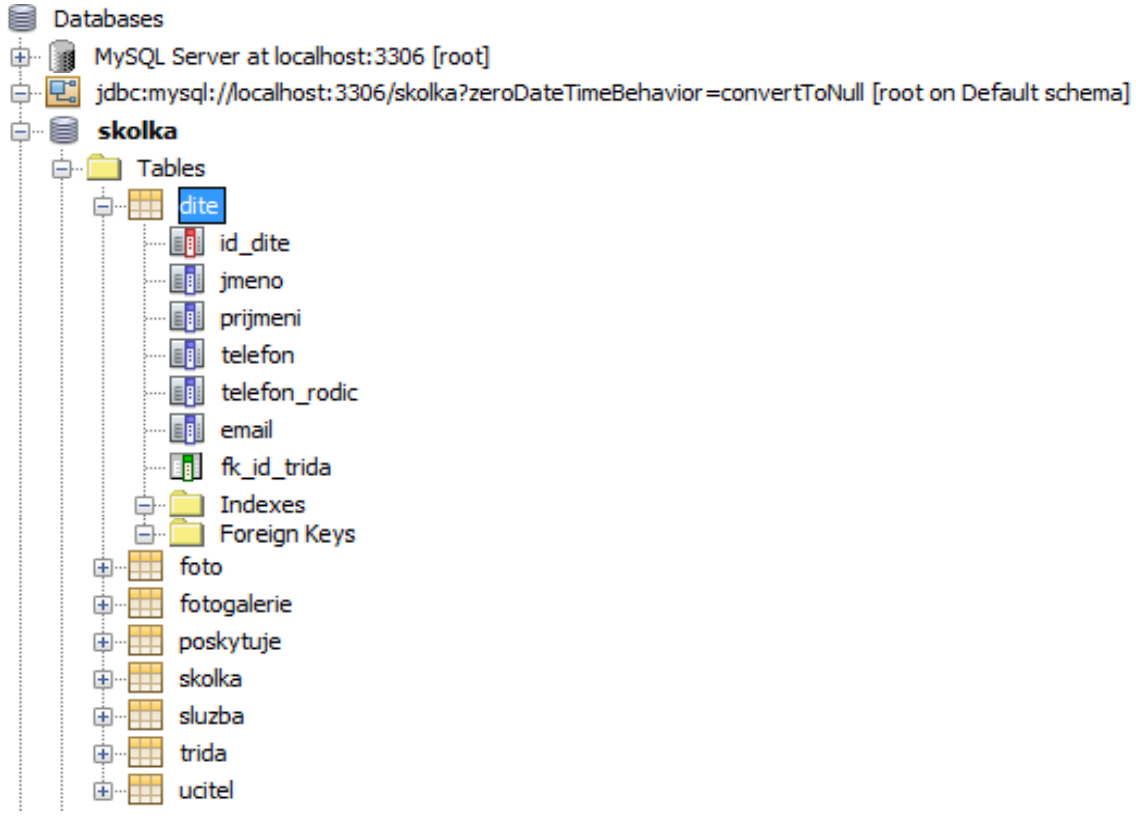
Aby bylo možné vytvořenou databázi připojit k projektu ve vývojovém prostředí Netbeans, byly nastaveny vlastnosti MySQL serveru. Na obrázku Obrázek 22.: MySQL Server Properties – základní vlastnosti je zobrazena konfigurace základních vlastností, v záložce *Admin Properties* je určena cesta k MySQL serveru.



Obrázek 22.: MySQL Server Properties – základní vlastnosti

Po kliknutí pravým tlačítkem myši na nakonfigurovaný MySQL server se rozbalila nabídka. Příkazem „*Connect*“ (Připojit) byl MySQL server připojen a zobrazil databáze na něm umístěné, viz Obrázek 23.: Databáze v NetBeans IDE.

V implementované aplikaci již byla celá databáze, včetně tabulek, vytvořena, ale NetBeans IDE umožňuje takovou databázi vytvořit a pracovat s ní.



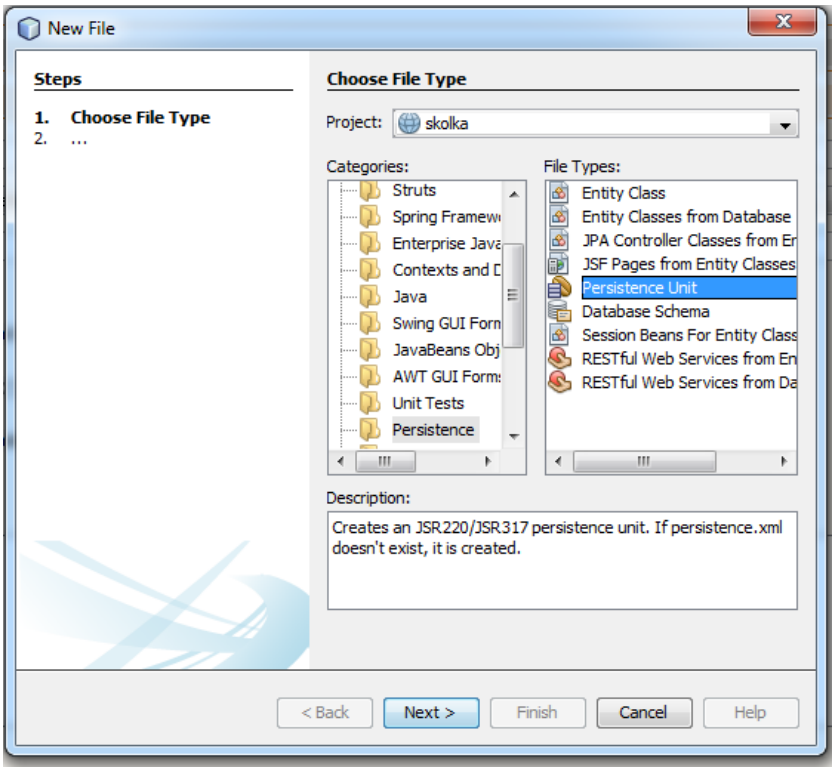
Obrázek 23.: Databáze v NetBeans IDE

Kromě toho byla databáze připojena k serveru Glassfish, tj. byl vytvořen JDBC Connection Pool, viz Příklad 3.: Konfigurační soubor glassfish-resources.xml, a ve vlastnostech serveru (*Tools → Servers → Glassfish*) byl povolen *JDBC Driver Deployment*.

```
<jdbc-connection-pool ...další atributy...>
      <property name="serverName" value="localhost"/>
      <property name="portNumber" value="3306"/>
      <!-- Přihlašovací údaje-->
      <property name="User" value="root"/>
      <property name="Password" value=""/>
      <!-- Adresa k databázi -->
      <property name="URL"
      value="jdbc:mysql://localhost:3306/skolka?characterEncoding=
UTF-8"/>
      <property name="driverClass" value="com.mysql.jdbc.Driver"/>
</jdbc-connection-pool>

<jdbc-resource enabled="true" jndi-name="skolka" object-
type="user" pool-name="mysql_skolka_rootPool"/>
```

Příklad 3.: Konfigurační soubor glassfish-resources.xml

V této fázi byla přidána tzv. persistence unit, viz Obrázek 24.: Přidání nové *Persistence Unit*.



Obrázek 24.: Přidání nové *Persistence Unit*

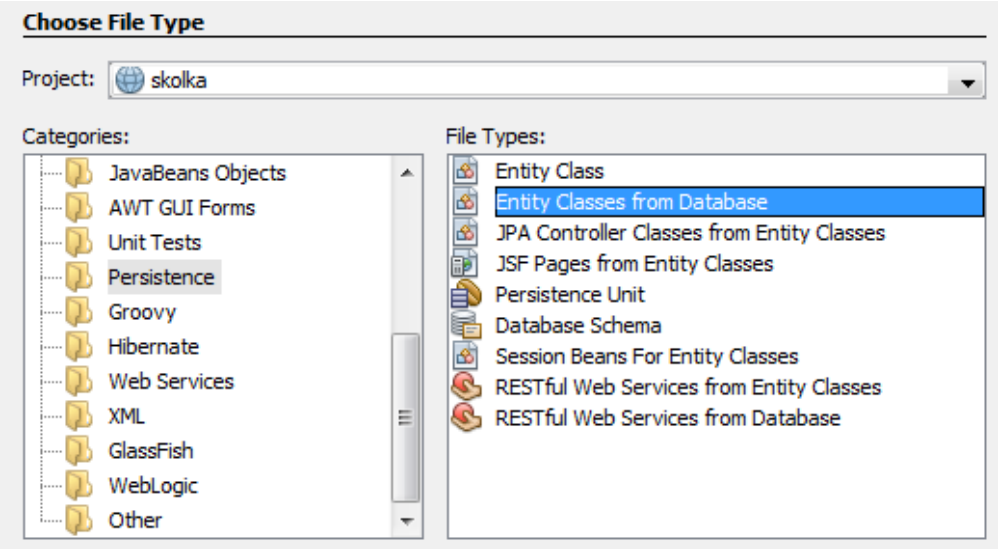
Jak lze vidět v ukázce Příklad 4.: Část konfiguračního souboru persistence.xml, byla vytvořena persistence unit `skolkaPU`. Poskytovatelem je, z důvodů defaultního nastavení NetBeans, implementace EclipseLink JPA 2.0, která je referenční implementací JPA. Zdrojem dat je databáze `skolka` a je nastaveno kódování v UTF-8.

```
<persistence-unit name="skolkaPU" transaction-type="JTA">
    <provider>
          org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <jta-data-source>skolka</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
        <property name="useUnicode" value="true"/>
        <property name="characterEncoding" value="UTF-8"/>
    </properties>
</persistence-unit>
```

Příklad 4.: Část konfiguračního souboru persistence.xml

Generování

Z databáze byly automaticky vygenerovány třídy entit (viz Obrázek 25.: Nabídka NetBeans pro generování nových komponent) a přidány do balíčku *org.skolka*. Podobně byly vygenerovány i Session beany a JPA Controllery pro tyto třídy (současně s xhtml soubory – viz kapitola 7.8). U všech těchto komponent je na následujících kapitolách popsáno jejich využití a modifikace.

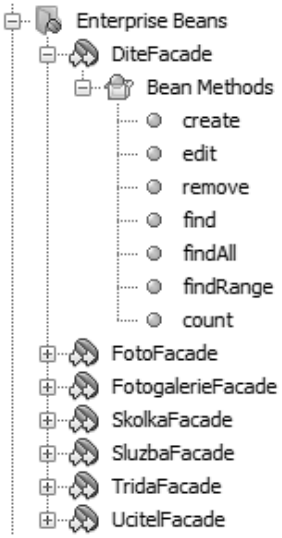


Obrázek 25.: Nabídka NetBeans pro generování nových komponent

7.2. EJB

7.2.1. *Stateless session bean*

Aplikace užívá bezstavové session beany [28], viz Obrázek 26.: EJB v NetBeans, konkrétně se jedná o EJB DiteFacade, FotoFacade, FotogalerieFacade, SkolkaFacade, SluzbaFacade, TridaFacade a UcitelFacade.



Obrázek 26.: EJB v NetBeans

Všechny EJB definují metody – *create*, *edit*, *remove*, *find*, *findAll*, *findRange* a *count*. Tyto metody jsou obsaženy v abstraktní třídě *AbstractFacade.java*. *AbstractFacade.java* pracuje s generickým typem <T> a implementuje pro něj společné operace, kde <T> je entita JPA. Viz Příklad 5.: Ukázka deklarace konstruktoru ze třídy AbstractFacade.java. EJB beany pracují s entitami, viz 7.3 JPA.

```
public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;    }
```

Příklad 5.: Ukázka deklarace konstruktoru ze třídy AbstractFacade.java

Následně bylo implementováno rozšíření abstraktní třídy *AbstractFacade* za využití ProductBeanů [28]. Pro demonstraci byla vybrána ukázka zdrojového kódu patřícího k objektu dítě, viz Příklad 6.: Ukázka entitní třídy DiteFacade.java, kde se nachází 3 anotace:

- `@Stateless` – určuje stav beanu (bezstavový)
- `@PersistenceContext` – vyjadřuje závislost na kontejneru řízeného *EntityManager*em a přidruženém persistentním obsahu. Je možné mít více konfigurací, poté se jen vybere `unitName` – skolkaPU.
- `@Override` – překrývá již deklarovanou třídu předka a plní funkci kontroly typografických chyb

```
@Stateless
public class DiteFacade extends AbstractFacade<Dite> {
	@PersistenceContext(unitName = "skolkaPU")
	private EntityManager em;

	@Override
	protected EntityManager getEntityManager() {
		return em;}

	public DiteFacade() {
		super(Dite.class);}
}
```

Příklad 6.: Ukázka entitní třídy DiteFacade.java

Pro bližší informace o EJB viz 2.6 Enterprise JavaBean (EJB).

7.3. JPA Entity

JPA [28] je velmi komplexní záležitostí, viz 2.7 Java Persistence API (JPA).

JPA využívá entit, neboli POJO (Plain Old Java Object) tříd reprezentující persistentní datové objekty. Během vývoje aplikace byly vygenerovány a následně upraveny entity Dite, Foto, Fotogalerie, Skolka, Sluzba, Trida a Ucitel. Entity byly popsány prostřednictvím anotací odchylek, jak je tomu v ukázce Příklad 7.: Dite.java, poněvadž mají nadefinované smysluplné defaultní podmínky.

```
@Entity
@Table(name = "dite")
@XmlRootElement
@NamedQueries({
	@NamedQuery(name = "Dite.findAll",
		query = "SELECT d FROM Dite d")})
	. . .

public class Dite implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Basic(optional = false)
	@Column(name = "id_dite")
	private Integer idDite;
	. . .
```

Příklad 7.: Dite.java

V rámci Entity jsou pro každý atribut vytvořeny getter a setter, viz Příklad 8.: Dite.java – ukázka get a set.

```
public Dite() { }

public Dite(Integer idDite) {
      this.idDite = idDite;}

public Integer getIdDite() {
      return idDite;}

public void setIdDite(Integer idDite) {
      this.idDite = idDite;}
```

Příklad 8.: Dite.java – ukázka get a set

Anotace

Byly použity následující anotace:

Povinné

`@Entity`	specifikuje třídu na entitu
`@Id`	určuje primární klíč entity

Použité nepovinné

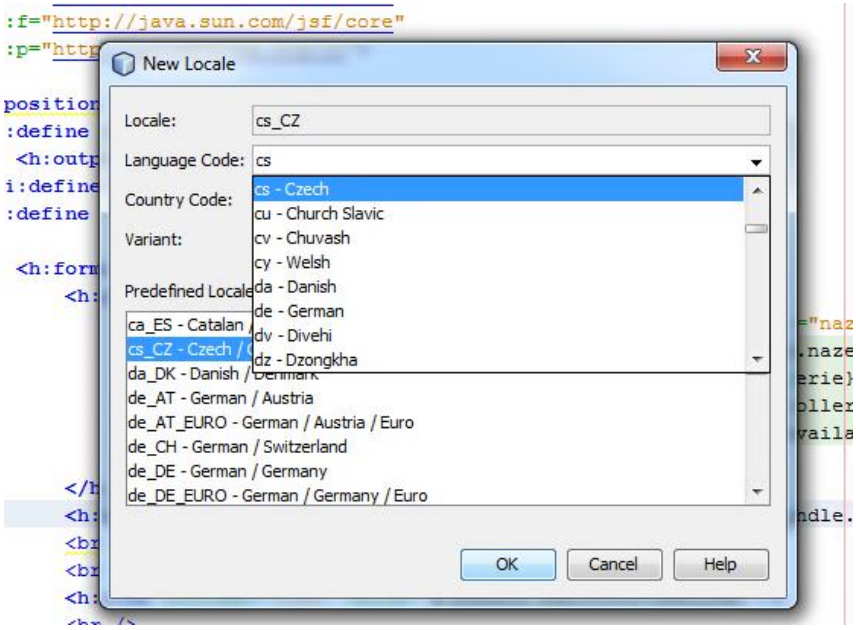
Jsou popsány pouze ty anotace a jejich funkčnosti, které jsou využity v aplikaci.

`@Table`	specifikuje primární tabulku
`@XmlRootElement`	mapuje třídu, popř. enum, do prvku XML
`@GeneratedValue`	zajišťuje automatické generování primárních klíčů
`@Basic`	určuje, zda může být položka nulová
`@NotNull`	komentovaný element nesmí být null, přijme libovolný typ
`@Column`	specifikuje mapovaný sloupec
`@Pattern`	kontroluje obsah stringu pomocí regulárního výrazu
`@Size`	určuje velikost, délku elementu

`@Temporal`	je povinnou poznámkou pro vlastnosti typu date a calendar
`@JoinColumn`	definuje mapování pro složené cizí klíče
`@OneToMany`	definuje asociaci one to many

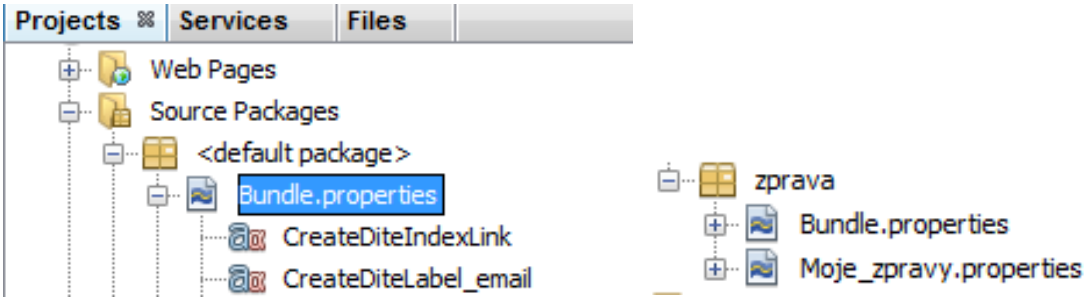
7.4. Resource Bundle

V implementovaném systému jsou využity tzv. Resource bundle, jež lze použít pro lokalizaci aplikace. Výrazně zjednodušují návrh stránek pro vícejazyčné aplikace. Diskutovaná aplikace počítá sice s jedním jazykem, ale bundle využívá pro centrální správu textů, tj. pro případy rozšíření a modifikaci textů. Automatické přidání češtiny do bundle umožňuje příkaz *Customize*, který zobrazí nabídku viz Obrázek 27.: Customize bundle.



Obrázek 27.: Customize bundle

Konfigurace resource bundle je popsána kapitole 7.1.1 Konfigurace JSF. Na obrázku Obrázek 28.: Bundle – umístění je zobrazeno umístění *Bundle.properties* v aplikaci.



Obrázek 28.: Bundle – umístění

Každá tzv. Property v bundle.properties se skládá z klíče a hodnoty, která se vyobrazuje konečnému uživateli aplikace. Klíč s hodnotou se zaznamenávají ve formátu `klíč=hodnota`. Viz Příklad 9.: Bundle – příklad zápisu.

```
PersistenceErrorOccured=Došlo k chybě.
Previous=Zpět
Next=Vpřed
CreateDiteTitle=Přidat dítě
CreateDiteShowAllLink=Prohlédnout celou databázi dětí
CreateDiteRequiredMessage_jmeno=Jméno je povinná položka.
CreateDiteLabel_email=Email:
```

Příklad 9.: Bundle – příklad zápisu

Konkrétní použití bundle lze pozorovat ve zdrojovém kódu Příklad 10.: Bundle – příklad použití. PrimeFaces komponenta `h:outputlabel` bude po spuštění aplikace zobrazovat text „Email:“.

```
<h:outputLabel
     value="#{bundle.CreateDiteLabel_email}" for="email" />
```

Příklad 10.: Bundle – příklad použití

7.5. Validace

Existuje více postupů, prostřednictvím kterých lze v JEE řešit validaci. Tato bakalářská práce se zabývá postupy použitými v aplikaci.

Validace byla ošetřena převážně anotacemi. Aplikace využila předdefinované anotace `@Basic`, `@NotNull` a `@Size` a nepředddefinovanou anotaci `@Pattern`, viz Příklad 11.: Validace za pomoci anotací.

```
@Basic(optional = false)    // Řetězec nesmí být prázdný
@NotNull                    // Řetězec nesmí být nulový
@Size(min = 1, max = 30)    // Řetězec musí obsahovat mit 1 a max
                            // 30 znaků
```

Příklad 11.: Validace za pomoci anotací

Anotace `@Pattern` umožňuje i definování vlastních validátorů a to pomocí regulárních výrazů. V ukázce Příklad 12.: Validace pomocí @Pattern lze pozorovat dva atributy. Atribut `regexp` obsahuje regulární výraz, pro bližší informace viz [20]. Atribut `message` pojímá chybové hlášení, které se zobrazí, pokud vstupní text neodpovídá regulárnímu výrazu.

```
@Pattern( // kontrola, zda nejsou ve jméně nepovolené znaky
      regexp="([A-Za-z ]{1,20})",
      message="Chybný formát jména!")
```

Příklad 12.: Validace pomocí @Pattern

JSF řeší validaci pomocí JSF tagu `f:validator`, nebo ve starších verzích JSF v konfigurační souboru *faces-config.xml*. Těmito postupy se diskutovaná aplikace nezabývá.

7.6. Navigace

Navigace uživatelského rozhraní aplikace byla řešena několika způsoby.

7.6.1. *Standardní navigace*

Při standardní navigaci je zadaná relativní adresa s celým názvem souboru, který bude vyobrazen. Je využita v menu, v příkladu Příklad 13.: Standardní navigace se taková adresa nachází v atributu `url`.

```
<p:menuitem value="Děti"
      icon="/images/index/dite.png"
      url="faces/Database/dite/List.xhtml"/>
```

Příklad 13.: Standardní navigace

7.6.2. *Automatická navigace*

Dále byla aplikována automatická navigace, viz Příklad 14.: Automatická navigace, kde atribut `outcome` obsahuje relativní adresu webové stránky s jejím názvem bez koncovky xhml. Jedná se především o navigaci na úvodní stránku.

```
<h:link
      outcome="/index" value="#{bundle.CreateDiteIndexLink}"/>
```

Příklad 14.: Automatická navigace

7.6.3. *Podmíněná navigace za pomoci beanů*

Dalším způsobem navigace, který se v aplikaci vyskytuje, je navigace podmíněná za pomoci beanů. V příkladu Příklad 15.: Podmíněná navigace (JSF) je přidávána nová položka, konkrétně dítě.

```
<h:commandLink
      action="#{diteController.create}"
      value="#{bundle.CreateDiteSaveLink}" />
```

Příklad 15.: Podmíněná navigace (JSF)

Atribut `action` spustí metodu `create()`.

Pokud jsou všechny údaje vyplněné správně a nedošlo k chybě, nová položka se vytvoří a uživateli se zobrazí stránka s prázdným formulářem pro vytvoření další položky a informace o úspěšném vytvoření předchozí položky.

Pokud ale došlo k chybě, uživatel uvidí stále stejnou stránku, včetně jím vyplněných údajů, a chybové hlášení udávající, proč se položka nevytvořila.

```
public String create() {
      try { // správně vyplněné údaje
          getFacade().create(current);
          JsfUtil.addSuccessMessage
                (ResourceBundle.getBundle
                ("/Bundle").getString("DiteCreated"));
           // informace o úspěchu
          return prepareCreate(); // vytvoření a nová stránka
      } catch (Exception e) { // chyba, zůstává na stránce
          JsfUtil.addErrorMessage
                (e,ResourceBundle.getBundle
                ("/Bundle").getString("PersistenceErrorOccured"));
           // chybové hlášení
          return null;}
}

public String prepareCreate() {
        current = new Dite();
      // vytvoří novou položku Ditě, pokud je naplněna
        selectedItemIndex = -1;
        return "Create";
      // vrací String určující výstup, kterým je stránka
      Create.xhtml ve složce Dítě
}
```

Příklad 16.: Podmíněná navigace (bean)

Dále lze v JEE využít navigaci za použití konfiguračního souboru *faces-config.xml*, pro bližší informace viz [24, 26, 28.]

7.7. Šablony

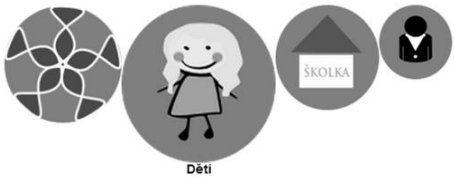
Index.xhtml

Index.xhtml, viz Příklad 17.: Zjednodušený pohled na zdrojový kód index.xhtml, je úvodní stránkou aplikace a neslouží jako šablona. Úvodní stránka obsahuje jedno rozřazovací menu pro snadnou orientaci v aplikaci.

```
<p:dock position="top" itemWidth="90" maxWidth="100" >
      <p:menuitem value="Fotogalerie"
            icon="/images/index/guest.png"
            url="faces/fotogalerie.xhtml" /></p:dock>
```

Příklad 17.: Zjednodušený pohled na zdrojový kód index.xhtml

Rozřazovací menu [22] je naimplementováno pomocí dock komponenty PrimeFaces, která výrazně připomíná dock rozhraní systému Mac OS X, viz Obrázek 29.: Index.xhtml ve webovém prohlížeči. Důvodem je snaha o uživatelsky přívětivou aplikaci.



Obrázek 29.: Index.xhtml ve webovém prohlížeči

Template.xhtml a Fotogalerie.xhtml

Jedná se o hlavní uživatelské rozhraní. Stránka je členěna podle drátového modelu, viz Obrázek 20.: Wireframe aplikace, do několika oblastí. Pro každou z těchto oblastí je vyobrazení určeno odděleně. Má tedy funkci šablony pro další stránky, viz Příklad 18.: Zjednodušený pohled na Template.xhtml. Fotogalerie.xhtml se od Template.xhtml liší nadpisem a zápatím. Tato šablona je užita při práci s fotografiemi a fotogaleriemi.

```
<p:layout fullPage="true" >
      <p:layoutUnit position="north" …další atrubity…>
            // záhlaví
      </p:layoutUnit>
      <p:layoutUnit position="west" …další atrubity…>
            // menu
      </p:layoutUnit>
      <p:layoutUnit position="center" …další atrubity…>
            // stěžejní část
            <h1>
                  <ui:insert name="title">Default Title</ui:insert>
            </h1>
            <p>
                  <ui:insert name="body">Default Body</ui:insert>
            </p>
      </p:layoutUnit>
</p:layout>
```

Příklad 18.: Zjednodušený pohled na Template.xhtml

7.8. Základní funkčnosti aplikace

Tato kapitola popisuje základní funkčnosti implementované aplikace. Metody přidat, upravit, odebrat a zobrazit položku z balíčku `org.skolka` a `org.skolka.util` jsou zčásti automaticky vygenerovány. Byla přidána funkce vyhledávání.

7.8.1. *Přidat položku*

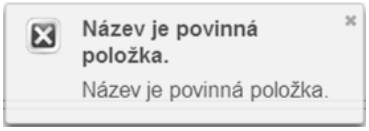
Přidání nové položky je realizováno ve formuláři `h:form`. V ukázce Příklad 19.: Zjednodušený pohled na soubor služba/Create.xhtml jsou prezentovány základní využité komponenty.

V komponentě `h:panelGrid` se nachází 2 položky – `h:outputLabel` (štítek) a `h:inputText` (vstupní políčko). Oběma položkám byl přiřazen titulek prostřednictvím bundle, viz Resource Bundle. Vstupní políčko navíc obsahuje volání třídy `sluzbaController` a jeho vyplnění je povinné.

```
<h:form>
      <h:panelGrid columns="2">
        <h:outputLabel
           value="#{bundle.CreateSluzbaLabel_nazev}"
           for="nazev" />
        <h:inputText id="nazev"
           value="#{sluzbaController.selected.nazev}"
           title="#{bundle.CreateSluzbaTitle_nazev}"
           required="true" requiredMessage=
           "#{bundle.CreateSluzbaRequiredMessage_nazev}"/>
      </h:panelGrid>

      <h:commandLink
           action="#{sluzbaController.create}"
           value="#{bundle.CreateSluzbaSaveLink}" />

      <h:commandLink
           action="#{sluzbaController.prepareList}"
           value="#{bundle.CreateSluzbaShowAllLink}"
           immediate="true"/>

      <h:link outcome="/index"
           value="#{bundle.CreateSluzbaIndexLink}"/>
      <p:growl id="growl" showDetail="true" sticky="true" />
</h:form>
```

Příklad 19.: Zjednodušený pohled na soubor služba/Create.xhtml

Atribut `requireMessage` udává, jaká zpráva se uživateli zobrazí, pokud nesplní předešlou podmínku. Tato zpráva se zobrazí pomocí komponenty `p:growl`, viz Obrázek 30.: Growl. Jedná se o prvek podobný vyskakovacímu oknu a lze jej zavřít.



Obrázek 30.: Growl

Ostatní tagy a atributy jsou popsány v kapitole 7.6 Navigace zároveň s popisem procesu Přidat.

7.8.2. *Upravit položku*

Funkce *Upravit* je v aplikaci řešena podobně jako funkce *Přidat*, proto je v ukázce Příklad 20.: Upravit (zjednodušený pohled) – SluzbaController.java metoda `update` pouze naznačena. Znatelný rozdíl lze pozorovat v metodě `prepareEdit`, protože program potřebuje znát, kterou položku bude editovat.

```
public String prepareEdit() {
      current = (Sluzba) getItems().getRowData();
      selectedItemIndex = pagination.getPageFirstItem() +
                  getItems().getRowIndex();
      return "Edit";}

public String update() {
      ... // podobně jako create()
      }
```

Příklad 20.: Upravit (zjednodušený pohled) – SluzbaController.java

7.8.3. *Odebrat položku*

Poslední ze základních operací s databází je smazání položky representované funkcemi `destroy` a `destroyAndView`. Uživateli je umožněno položky mazat pouze na stránce s podrobným náhledem na onu položku.

Funkce `destroy` konkrétní položku smaže (`performDestroy`), aktualizuje tabulku a vrátí uživatele na stránku zobrazující všechny položky z dané tabulky, viz Příklad 21.: Destroy a performDestroy.

```
public String destroy() {
   current = (Sluzba) getItems().getRowData();
   selectedItemIndex = pagination.getPageFirstItem() +
      getItems().getRowIndex();
   performDestroy();
   recreatePagination();
   recreateModel();
   return "List";}

private void performDestroy() { // provede smazání položky
   try {
      getFacade().remove(current);
      JsfUtil.addSuccessMessage(ResourceBundle.getBundle
      ("/Bundle").getString("SluzbaDeleted"));
   } catch (Exception e) {
      JsfUtil.addErrorMessage
      (e, ResourceBundle.getBundle
      ("/Bundle").getString("PersistenceErrorOccured"));}}
```

Příklad 21.: Destroy a performDestroy – SluzbaController.java

Funkce `destroyAndView`, viz Příklad 22.: DestroyAndView, postupuje obdobně jako funkce `destroy`. Odlišnost spočívá ve stránce, která se uživateli objeví po smazání položky. Pokud totiž nesmazal poslední položku tabulky, funkce se zachová rozdílně a poskytne mu náhled na jinou existující položku (`return View`).

```
public String destroyAndView() {
      performDestroy(); recreateModel();
      updateCurrentItem(); // aktualizuje nově zobrazenou položku
      if (selectedItemIndex >= 0) {
            return "View";
      } else { // všechny položky jsou smazány
            recreateModel();
            return "List";}}
```

Příklad 22.: DestroyAndView – SluzbaController.java

7.8.4. Vyhledat položku

Vyhledávání, stejně jako třídění, je v aplikaci řešeno za použití komponenty `p:dataTable`.

Ve výňatku z kódu Příklad 23.: Vyhledávání ve všech sloupcích – skolka/View.xhtml implementuje vyhledávání funkce `tridy.filter()`, která prohledává celou tabulku tříd.

Na této konkrétní stránce je potřeba vyhledat třídy a jejich učitele patřící do aktuální prohlížené školky, a z tohoto důvodu je název této školky defaultní hodnotou `value`.

```
<p:dataTable … další atributy … >
...
<h:outputText value="Třídy a jejich učitelé ze školky " />
     <p:inputText id="globalFilter" onkeyup="tridy.filter()"
     value="#{skolkaController.selected.nazev}"
     style="width:150px" />
</p:outputPanel>
...
</p:dataTable>
```

Příklad 23.: Vyhledávání ve všech sloupcích – skolka/View.xhtml

Dále je vyhledávání vyřešeno atributem `filterBy`, který umožňuje vyhledávání v jednom sloupci. Viz Příklad 24.: Vyhledávání a třídění v konkrétním sloupci – skolka/View.xhtml, kde se vyhledává podle názvu školky. Dalším zajímavým atributem je `sortBy`, který umožňuje setřídit celou tabulku podle vybraného sloupce.

```
<p:column headerText="Školka" style="width:200px;"
     sortBy="#{item.fkIdSkolka.nazev}"
     filterBy="#{item.fkIdSkolka.nazev}">

     <h:outputText value="#{item.fkIdSkolka.nazev}" />
</p:column>
```

Příklad 24.: Vyhledávání a třídění v konkrétním sloupci – skolka/View.xhtml

7.9. Další implementované funkčnosti

7.9.1. *Fotogalerie*

V aplikaci jsou v souladu se zákonem, viz Legislativní požadavky, použity výhradně volně šiřitelné fotografie a obrázky z tzv. free fotogalerií. Viz [8, 15, 27, 29].

Fotografie učitele

Při prohlížení profilu učitele se uživateli zobrazí i jeho fotografie. To je zajištěno tagem `p:graphicImage` a jeho atributem `value`, který obsahuje adresu fotografie. Fotografie je i s dalšími informacemi ohraničená za použití tagu `p:fieldset`.

```
<p:fieldset // ohraničení s nadpisem (Příjmení Jméno)
     legend="#{ucitelController.selected.prijmeni}
          #{ucitelController.selected.jmeno}">

<h:panelGrid column="2" cellpadding="10">
     <p:graphicImage // fotografie zobrazovaného učitele
          value="#{ucitelController.selected.foto}"
          height="252" />
</h:panelGrid>
 ... // informace o učiteli, tj. Jméno, příjmení, adresa, …
</p:fieldset>
```

Příklad 25.: Fotografie – ucitel/View.xhtml

V případě, že fotografie není u učitele k dispozici, zobrazí se obrázek sovy, který je na adrese proměnné foto, viz Příklad 26.: Ošetření chybějící fotografie – Ucitel.java. Pokud je zadána neexistující adresa fotografie, zůstane plocha určená pro fotografii prázdná.

```
public String getFoto() {
     if ("".equals(foto)){
          foto = "../../images/ucitel/neni.jpg";
     }
return foto; }
```

Příklad 26.: Ošetření chybějící fotografie – Ucitel.java

Fotogalerie

V rámci práce byla vytvořena fotogalerie – v databázi jsou uloženy adresy umístění jednotlivých fotografií. Jednotlivé fotogalerie mají stejný název složky s fotografiemi jako `id_fotogalerie`.

Implementace

Fotogalerie[22] byla implementována za použití tagu `p:galleria` v dialogu, viz Příklad 27.: Fotogalerie – View.xhtml. Stránka *fotogalerie/View.xhtml* zobrazí informace o dané fotogalerii a tlačítko, při jehož kliknutí se objeví vyskakovací okno s fotografiemi, kter do fotogalerie patří.

Parametr `header` v dialogu určuje nadpis ve vyskakovacím okně, hodnota položky `widgetVar` je použita v atributu tlačítka – `onclick`. Jednotlivé fotografie jsou načítány z adresy `value` v tagu `p:graphicImage`.

```
<h:panelGrid columns="1" cellpadding="5">
      // tlačítko pro zobrazení fotogalerie
      <p:commandButton id="obr2" value="Náhled"
            onclick="fotogalerie.show();" type="button" />
</h:panelGrid>

<p:dialog id="basicDialog2" header="Fotogalerie
      #{fotogalerieController.selected.nazev}"
      width="620" height="400" widgetVar="fotogalerie" >

      <p:galleria value="#{galleriaBean.images}" var="image">
      <p:graphicImage
            value="../../images/fotogalerie/
            #{fotogalerieController.selected.idFotogalerie}/
            #{image}" />
</p:galleria>
```

Příklad 27.: Fotogalerie – View.xhtml

7.9.2. *Upload*

Pro nahrání fotografie učitele byl za pomocí tagu `p:fileUpload` naimplementován upload fotografie, viz Příklad 28.: Upload – ucitel/Create.xhtml.

```
<p:fileUpload
      fileUploadListener="#{tryUploadBean.handleFileUpload}"
      // určuje bean, který upload provede – převzato z [24]
      value="Upload" id="file"
      mode="advanced" update="messages"
      sizeLimit="1073741824"
      // povolený velikostní limit
      label="Vybrat" uploadLabel="Nahrát" cancelLabel="Zrušit"
      // tlačítka
      allowTypes="/(\.|\/)(jpe?g)$/"/> // povolené typy souborů
```

Příklad 28.: Upload – ucitel/Create.xhtml

Při řešení funkce uploadu nejprve docházelo k ukládání pouze dočasných souborů, což pomohla vyřešit kniha *JSF 2.0 cookbook* [24]. Přetrvává ovšem problém při nahrávání velkého souboru, kdy aplikace soubor nenahraje (kvůli limitům), ale nenahlásí chybu. Uživatel potom neví, zda bylo nahrání souboru provedeno.

7.9.3. *Export*

Správci databáze je umožněno si vybrané tabulky exportovat do formátů xls a pdf. Export je proveden za pomoci funkce PrimeFaces tagu `<p:dataExporter>`, viz Příklad 29.:

Export, a nachází se vždy na stránce vyhledávání v tabulce. Tím je zajištěno, že je možné exportovat i vyfiltrované části tabulky.

Formát, do něhož bude tabulka exportována, stanovuje atribut `type`. Může nabývat hodnot `xls`, `pdf`, `csv` a `xml`. Hodnota atributu `target` je shodná s hodnotou atributu `id` náležejícího tabulce. Atribut `fileName` udává název exportovaného souboru. Kódování určuje atribut `encoding`.

```
Exportovat tabulku:
<h:commandLink>
     <!—Obrázek, který reprezentuje exportovaný formát-->
     <p:graphicImage
          value="../../images/export/xls.jpg" height="40" />
     <!—Tag pro export-->
     <p:dataExporter type="xls"
          target="items" encoding="utf-8"
          fileName="Tabulka-deti"/>
</h:commandLink>
```

Příklad 29.: Export do xls (deti/Filter.xhtml)

8. Testování

Testování [12] je nedílným a velmi důležitým stadiem vývoje aplikací. Navíc je testování implementované aplikace jedním z cílů diskutované práce.

8.1. Uvažovaná plná verze

Při testování uvažované plné verze aplikace se postupuje komplexně podle testovacího plánu [12]. Výslednou aplikaci testuje nejen vývojář, ale především testovací tým a uživatelé. Programátor otestuje svou naimplementovanou část, zda se chová podle jeho očekávání. Poté už je aplikace testována z hlediska uživatele, a to testovacím týmem, nebo samotným zákazníkem.

A jelikož by se jednalo o aplikaci, která má k dispozici velké množství osobních údajů, testování by se nemalou měrou zaměřilo na bezpečnost aplikace, ochranu před únikem informací a možné škody.

8.2. Implementovaný modul

Implementovaný modul aplikace testoval především její autor. Testování [12] bylo z důvodu malého rozsahu aplikace prováděno zejména manuálně, pouze v kapitole 8.5.1 Barvy byl použit testovací software. Aplikace byla testována na funkčnost, kompatibilitu, přístupnost a použitelnost. První tři vlastnosti testoval vývojář aplikace, použitelnost uživatelé neseznámení s aplikací.

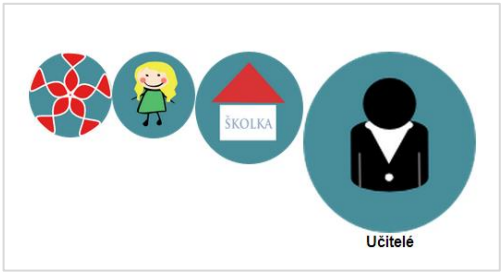
8.3. Funkčnost

Pro testování funkčnosti byly napsány a otestovány testovací scénáře pro vybrané případy.

Přidat učitele – vyplnění všech polí

ID	#1
Název	Přidat učitele
Podtitul	Vyplnění všech polí
Účel	Ověření, zda se nový učitel nahraje do databáze
Podmínky	
Kroky	1. Na úvodní stránce, nebo v menu klikni na položku "Učitelé" 2. Klikni na "Přidat učitele" 3. Vyplň formulář 3.a Vyplň titul (Mgr.), jméno (Alois), příjmení (Kundera), email (kundera@skolka.cz) a telefon (903775231) 3.b Napiš název fotografie (kundera) 3.c Nahraj fotografii (kundera.jpg) 4. Klikní na "Uložit"
Očekávaný výsledek	1. Objeví se zpráva "Podařilo se." 2. Proběhne vložení nového učitele do databáze 3. Na stránce "Učitelé" se objeví nový záznam s námi zadanými údaji a s automaticky vygenerovaným identifikačním číslem 4. U dané položky kliknu na „prohlížet“ a zobrazí se stránka s informacemi o učiteli, včetně jeho fotografie
Provedení testu	Otestováno
Poznámka	Varování – Problém při uploadu fotografie (velikost) – nevypsalo se chybové hlášení, přestože se fotografie na první pokus nenahrála.

1. Kliknout na „Učitelé“:

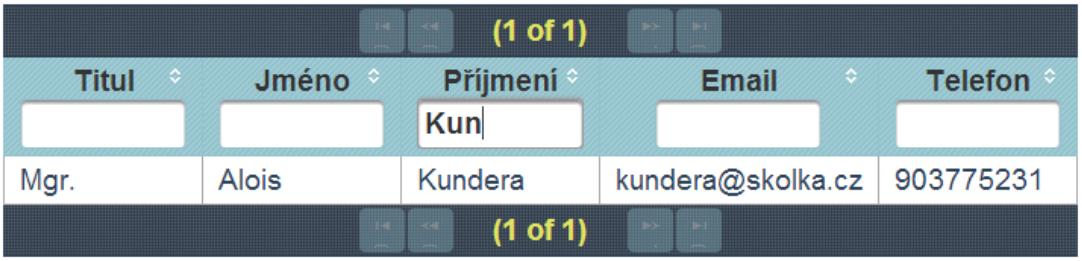


Obrázek 31.: Vlevo „Úvodní stránka“, vpravo „Menu“

2., 3. a 4. Kliknout na „Přidat učitele“, vyplnit formulář dle zadání a uložit

Obrázek 32.: Přidání učitele (Vyplnění všech polí)

Výsledek:



Obrázek 33.: Najít učitele „Kundera“

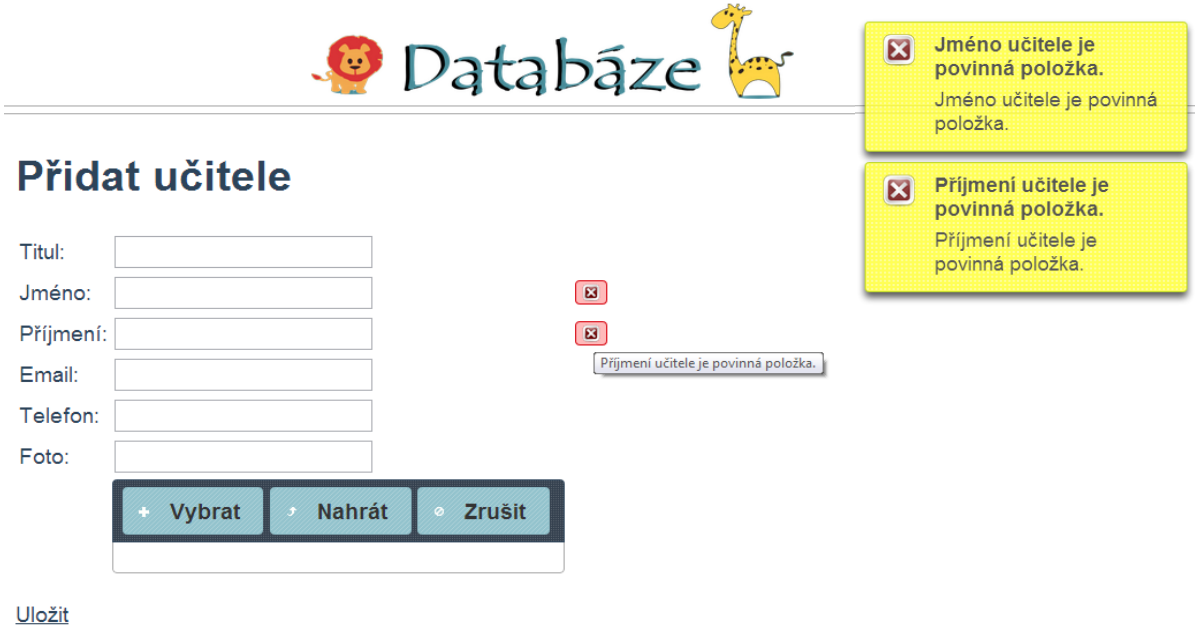
Obrázek 34.: Detail učitele „Alois Kundera“

Přidat položku učitel – nevyplnění polí

ID	#2
Název	Přidat učitele
Podtitul	Nevyplnění polí
Účel	Ověření, zda bude program vyžadovat vyplnění položek formuláře
Podmínky	
Kroky	1. Na úvodní stránce, nebo v menu klikni na položku "Učitelé" 2. Klikni na "Přidat učitele" 3. Klikni na "Uložit"
Očekávaný výsledek	1. Objeví se varovná hlášení "Jméno je povinná položka." a "Příjmení je povinná položka." 2. Na stránce "Učitelé" se v tabulce neobjeví nový záznam.
Provedení testu	Otestováno
Poznámka	OK

Postupujte dle uživatelské příručky, viz Příloha 3.

Výsledek:



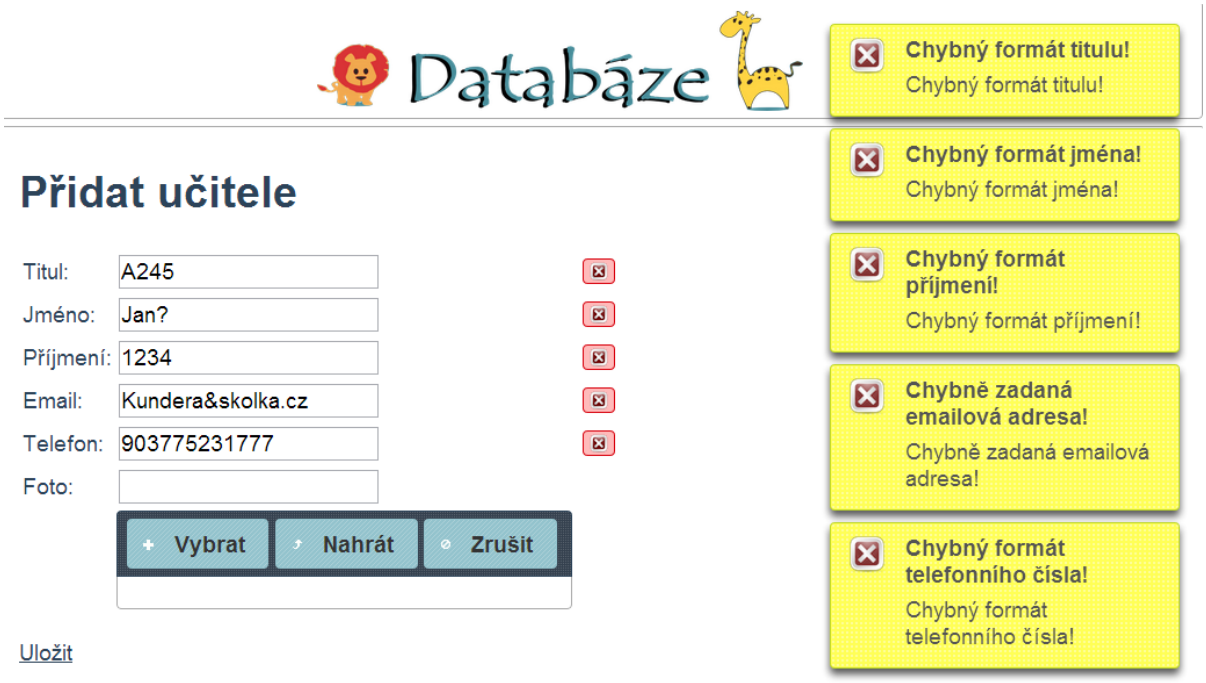
Obrázek 35.: Přidat učitele (Nevyplnění polí)

Přidat položku učitel – chybné vyplnění polí

ID	#3
Název	Přidat učitele
Podtitul	Chybné vyplnění polí
Účel	Ověření, zda program kontroluje položky formuláře
Podmínky	
Kroky	1. Na úvodní stránce, nebo v menu klikni na položku "Učitelé" 2. Klikni na "Přidat učitele" 3. Vyplň formulář 3.a Vyplň titul (A245), jméno (Jan?), příjmení (1234), email (Kundera&skolka.cz) a telefon (903775231777) 4. Klikni na "Uložit"
Očekávaný výsledek	1. Objeví se varovná hlášení "Chybný formát titulu.", "Chybný formát jména.", "Chybný formát příjmení.", "Chybně zadaná emailová adresa." a "Chybný formát telefonního čísla.". 2. Na stránce "Učitelé" se v tabulce neobjeví nový záznam.
Provedení testu	Otestováno
Poznámka	OK

Postupujte dle uživatelské příručky, viz Příloha 3.

Výsledek:



Obrázek 36.: Přidat učitele (Chybné vyplnění polí)

Exportovat tabulku „Učitelé

ID	#4
Název	Exportovat tabulku „Učitelé“
Podtitul	
Účel	Ověření, zda program vygeneruje tabulku s daty o učitelích
Podmínky	
Kroky	1. Na úvodní stránce, nebo v menu klikni na položku "Učitelé" 2. Klikni na "Najít učitele" 3. Klikni na ikonu PDF (vedle textu „Exportovat tabulku“)
Očekávaný výsledek	1. Vygeneruje se soubor „Tabulka-ucitele.pdf“.
Provedení testu	Otestováno
Poznámka	OK

Postupujte dle uživatelské příručky, viz Příloha 3.

Výsledek:

Bc.	Simona	Kopecka	kopecka@skolka.cz	731605231
	Marek	Hudebnicky	hudebnicky@skolka.cz	732775231
	Jarmila	Krajena	krajena.jarmila@email.cz	
Bc.	Zuzana	Sedma	sedma.z@skolka.cz	
Bc.	Jirka	Mensik	mensik.jirka@gmail.com	
Mgr.	Alois	Kundera	kundera@skolka.cz	903775231
	Roman	Adamec	adamec@skolka.cz	906105231
Mgr.	Milada	Navratilova	navratilova@skolka.cz	
Bc.	Karel	Podmanivy	podmanivy@skolka.cz	937875231
Mgr.	Simona	Melounova	simona.melounova@email.cz	623159874
	Andrea	Pavlova		
	Jan	Daniel		

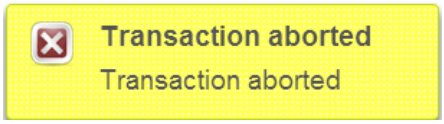
Obrázek 37.: Exportovat tabulku – Obsah vygenerovaného souboru „Tabulka-ucitele.pdf“

Smazat položku – smazání učitele přiřazeného ke školce

ID	#5
Název	Smazat položku
Podtitul	Smazání učitele přiřazeného ke školce
Účel	Ověření, zda program odmítne položku smazat
Podmínky	V databázi existuje učitelka (Bc. Zuzana Sedmá) s vazbou na třídu
Kroky	1. Na úvodní stránce, nebo v menu klikni na položku "Učitelé" 2. Klikni na "Prohlédnout" u učitele/učitelky (Bc. Zuzana Sedma) 3. Klikni na „Smazat“
Očekávaný výsledek	1. Objeví se chybové hlášení „Transaction aborted“. 2. Položka zůstane v databázi.
Provedení testu	Otestováno
Poznámka	OK

Postupujte dle uživatelské příručky, viz Příloha 3.

Výsledek:



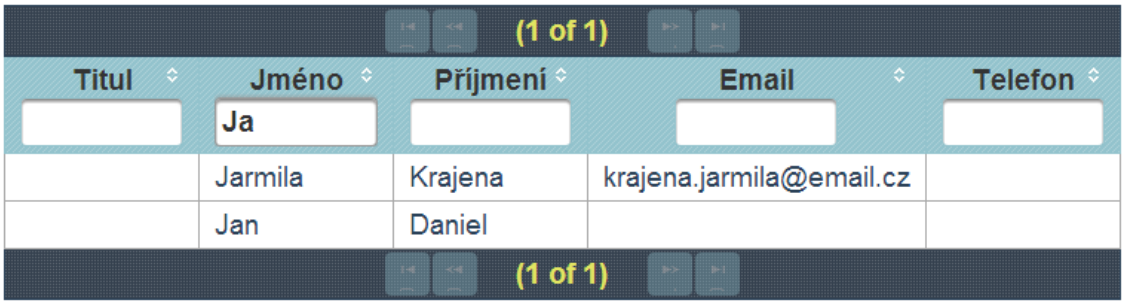
Obrázek 38.: Smazat položku (Smazat učitele přiřazeného ke školce)

Smazat položku – smazání učitele nepřiřazeného ke školce

ID	#6
Název	Smazat položku
Podtitul	Smazání učitele nepřiřazeného ke školce
Účel	Ověření, zda program položku smaže
Podmínky	Existuje učitel (Mgr. Jan Toman) a nemá vazbu k žádné třídě
Kroky	1. Na úvodní stránce, nebo v menu klikni na položku "Učitelé" 2. Klikni na "Prohlédnout" u učitele (Mgr. Jan Toman) 3. Klikni na „Smazat“
Očekávaný výsledek	1. Položka se smaže. 2a V případě, že nešlo o jediného učitele, objeví se stránka s detailem dalšího učitele. 2b Jinak se zobrazí stránka „List.xhtml“ s prázdným seznamem učitelů
Provedení testu	Otestováno
Poznámka	OK

Postupujte dle uživatelské příručky, viz Příloha 3.

Výsledek:



Obrázek 39.: Smazat položku (Smazat učitele nepřiřazeného k položce) – učitel Mgr. Jan Toman se v databázi nevyskytuje.

Výsledek

Během testování funkčnosti aplikace nebyly zjištěny závažné chyby.

8.4. Kompatibilita

Kompatibilita byla testována na zobrazení v různých prohlížečích - Google Chrome, Mozilla Firefox, Internet Explorer a Opera. Ve všech zmíněných prohlížečích byly webové stránky zobrazeny obdobně a disponovaly shodnou funkčností.

8.5. Přístupnost

Přístupnost je neodlučitelnou součástí tvorby všech webových stránek. S ohledem na cílového uživatele naimplementovaného modulu se testování přístupnosti omezuje na několik základních oblastí, viz následující podkapitoly.

8.5.1. *Barvy*

Správné použití barev na webové stránce je podstatným pravidlem v rodině pravidel pro přístupné stránky. Barvy byly testovány programem Colour Contrast Analyser 2.2 for Web Pages [30].

Z tabulky Tabulka 3.: Výsledky testování barev (program Colour Contrast Analyser) lze vyčíst, že při výběru barevné palety pro aplikaci nedošlo ke kritické chybě. Aplikace shledává problémy zejména v barvách nadpisů a záhlaví tabulek, obzvláště pak v tabulce vyhledávací. Podrobný výpis programu je přiložen v příloze Juicy Studio Colour Contrast Analyser. Hodnoty jsou porovnávány v souladu s pravidly W3C (World Wide Web Consortium) [31].

		Počet chyb		
Stránka	Složka	Kontrastní poměr svítivosti	Rozdíl v jasu	Rozdíl v barvě
Index		0	0	0
List	vše	0	0	3
View	Školka	0	0	11
	Třída	0	0	14
	Učitel	0	0	3
	Dítě	0	0	2
	Služba	0	0	2
	Fotogalerie	0	0	4
	Fotografie	0	0	3
Create / Edit	Školka	0	0	2
	Třída	0-2	0-2	2-4
	Učitel	0	0	5
	Dítě	0-1	0-1	2-3
	Služba	0	0	2
	Fotogalerie	0-1	0-1	2-3
	Fotografie	0-1	0-1	2-3
Filter	vše	0	0	17

Tabulka 3.: Výsledky testování barev (program Colour Contrast Analyser)

8.5.2. *Navigace*

Navigace byla taktéž testována kvůli ověření, zda byla implementována v souladu se standardem W3C [31].

Menu se na každé stránce, kromě úvodní stránky, vyskytuje díky použití šablony na stejném místě, tj. v levé části stránky. Menu využívá elementu `frame` (rámce) s nadpisem, jak doporučuje specifikace.

Aplikace má naimplementované jednoduché menu, nikoliv rozbalovací (doporučení standardu). Podle testera je však dostatečně přehledné, neboť není natolik obsáhlé, aby bylo použití doporučeného typu menu nutné.

8.5.3. *Přístupnost za podpory klávesnice*

Následně bylo testováno, do jaké míry aplikace umožňuje se po webové stránce pohybovat pouze za pomocí klávesnice, tj. bez počítačové myši. Podle W3C [31] by měly být všechny funkcionality webové stránky dostupné prostřednictvím klávesnice. Bylo zjištěno, že je uživateli umožněno používat klávesnici pouze při pohybu mezi jednotlivými prvky formuláře. Tuto funkci zajišťuje tlačítko klávesnice TAB.

8.5.4. *Dokumentace*

Dokumentace popisuje způsob užívání aplikace, slouží tedy jako nápověda.

Dokumentaci si prostudovalo a hodnotilo 8 uživatelů, viz Tabulka 4.: Hodnocení dokumentace. Uživatelé klasifikovali dokumentaci podle stupnice od 1 (výborná) do 5 (nedostatečná). Všichni uživatelé hodnotili dokumentaci kladně. Lze se domnívat, že je její zpracování dostatečné.

Kritéria hodnocení	David	Radka	Lenka	Petra	Jana	Robert	Tomáš	Marcela
Uživatelská dokumentace	2	1	1	1	1	1	1	2
Přehlednost	1	1	1	1	1	1	1	2
Srozumitelnost	2	1	1	1	1	2	1	2

Tabulka 4.: Hodnocení dokumentace

V rámci testování dokumentace byla testována i srozumitelnost chybových hlášek. Pokud uživatel nezadá povinnou položku při vyplňování políček ve formuláři, objeví se chybová hláška ve formátu „{název položky} je povinná položka.“ apod. Pokud je položka vyplněna,

ale chybně, objeví se zpráva ve formátu „Chybný formát {název položky}.“. Tester i testující uživatelé považují taková chybová hlášení za dostatečně srozumitelná a pochopitelná.

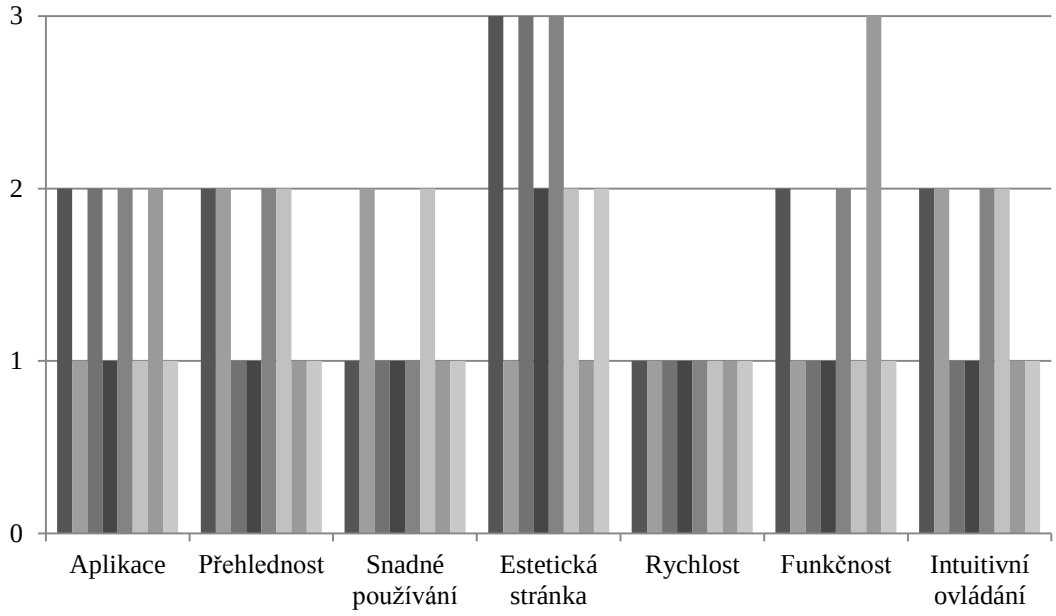
8.6. Použitelnost

Použitelnost byla otestována totožnou skupinou uživatelů a identickým systémem hodnocení, jako tomu bylo u testování uživatelské dokumentace, viz kapitola 8.5.4 Dokumentace. Hodnocení uživatelů lze vidět v tabulce Tabulka 5.: Použitelnost.

	David	Radka	Lenka	Petra	Jana	Robert	Tomáš	Marcela
Aplikace	2	1	2	1	2	1	2	1
Přehlednost	2	2	1	1	2	2	1	1
Snadné používání	1	2	1	1	1	2	1	1
Estetická stránka	3	1	3	2	3	2	1	2
Rychlost	1	1	1	1	1	1	1	1
Funkčnost	2	1	1	1	2	1	3	1
Intuitivní ovládání	2	2	1	1	2	2	1	1

Tabulka 5.: Použitelnost

Na základě hodnocení uživatelů byl vytvořen graf, viz Obrázek 40.: Použitelnost . S nejpřísnějším hodnocením se potýká estetická stránka aplikace. Uživatelé navrhují použití více obrázků na úvodní straně a v rámci celé aplikace více barev a estetických prvků. 37,5% uživatelů vypovědělo, že bylo ovládání aplikace natolik snadné a intuitivní, že by k němu nepotřebovali uživatelskou dokumentaci. Z hlediska funkčnosti je aplikace hodnocena převážně kladně. Nejlépe hodnoceným parametrem je rychlost.



Obrázek 40.: Použitelnost (graf) – barvy sloupců odlišují jednotl. uživatele.

Otázkou je, jaké hodnocení by se aplikaci dostalo při dlouhodobém používání – na internetu a při plném naplnění daty atp.

9. Zhodnocení použitých technologií

V rámci bakalářské práce byla za využití technologií Java EE vytvořena aplikace pro naplnění databáze daty. Samotnou aplikaci, nezávisle na bakalářské práci, je možné řešit jednodušeji na úrovni jiných platforem a za použití odlišných technologií. S ohledem na velikost aplikace není použití technologie typu JEE nejvhodnější volbou, její aplikace byla však jedním ze základních cílů práce.

Autor potvrzuje, že přes náročnost a složitost technologie převládají výhody použití JEE. Velkou výhodou při tvorbě aplikací v JEE je možnost šablonování, znuvupoužitelnost a automatické generování některých metod, zabudované funkce, práce s databází bez použití jazyka SQL a možnost výměny komponent a technologií. Nezanedbatelnou položkou je i výrazné zjednodušení tvorby prezentační vrstvy aplikace, kterou usnadnila především knihovna PrimeFaces.

Vzhledem k zaměření a rozsahu práce nebylo možno využít některé vlastnosti JEE. Jsou to především ty vlastnosti, které hrají velkou roli v rozsáhlých aplikacích, např. škálovatelnost, webové služby, nebo specializace vývojářů.

10. Závěr

Závěry jsou uvedeny k jednotlivým zadaným cílům:

1) Navrhnout jednoduchou webovou aplikaci pro tvorbu a využití databáze dětských školek v JČ kraji

Byla provedena analýza a návrh webové aplikace pro tvorbu a využití databáze školek. S ohledem na konkurenční řešení použité technologie byly stanoveny požadavky na systém a namodelován diagram use-case. Byla navržena architektura aplikace, včetně er-diagramu a drátového modelu navrhujícího rozložení prvků na stránce.

2) Implementovat jádro této aplikace pomocí technologií JEE

Byl proveden výběr funkcí jádra aplikace a technologií JEE a naimplementován systém poskytující všechny funkce navržené v analýze. V rámci popisu implementace je popsán postup a komentovány důležité a zajímavé konstrukce.

3) Zhodnotit výhody a nevýhody použití technologie na základě zkušeností získaných při vývoji aplikace

Výhody a nevýhody využití této technologie byly vyhodnoceny v kapitole 9 Zhodnocení použitých technologií

4) Napsat testovací scénáře a provést testy

Implementovaná aplikace byla testována na kompatibilitu, přístupnost, použitelnost a obzvláště na funkčnost. Pro účely testování funkčnosti byly napsány testovací scénáře a podle nich bylo provedeno testování. Aplikaci vyzkoušelo 8 uživatelů a potvrdilo její přístupnost, uživatelskou přívětivost a funkčnost. Během testování nebyly zjištěny závažné chyby aplikace.

11. Seznam použité literatury

[1] APACHE. *Apache Tomcat* [online]. 1999, 18.2.2013 [cit. 2013-03-20]. Dostupné z: http://tomcat.apache.org/

[2] BODOFF, Stephanie. *The J2EE tutorial*. 2. vyd. Boston, MA: Addison-Wesley, 2002, xxvi, 491 p. ISBN 02-017-9168-4.

[3] BROWN, Daniel M. *Communicating design: developing web site documentation for design and planning*. Berkeley: New Riders, c2007, xiv, 353 s. ISBN 978-032-1392-350. Dostupné z: http://books.google.cz/books?id=6dlFc4iwbuIC&hl=cs&source=gbs_navlinks_s

[4] BURD, Barry. *JSP: Javaserver pages, podrobný průvodce*. Vyd. 1. Praha: Computer Press, 2003, 381 s. ISBN 80-722-6804-X.

[5] Curso: TECN DE PROGRAMACAO APLIC III [turma 06J] 2012/1. *ProgDan Moodle Server: Universidade Presbiteriana Mackenzie* [online]. 2012 [cit. 2013-04-11]. Dostupné z: http://moodle.progdan.com/course/view.php?id=30

[6] Database marketshare: MySQL, PostgreSQL, MariaDB, MongoDB | Jelastic — Top Java and PHP Host, Java and PHP in the Cloud, Java and PHP Server Hosting, Java and PHP Cloud Computing. *Jelastic: Top Java and PHP Host, Java and PHP in the Cloud, Java and PHP Server Hosting, Java and PHP Cloud Computing* [online]. 2011 [cit. 2013-04-07]. Dostupné z: http://blog.jelastic.com/2011/10/19/database-marketshare-mysql-postgresql-mariadb-mongodb/

[7] HAY, David C. *Requirements analysis: from business views to architecture*. Upper Saddle River, NJ: Prentice Hall PTR, c2003, xxxvi, 458 p. ISBN 01-302-8228-6.

[8] HOLMES & COTTRELL GRAPHIC TECHNOLOGIES. *Vector Art by Holmes & Cottrell - highest quality electronic clip art, clipart, for sign industry, vinyl ready, easy to color fill for large format printing, great results for routing and engraving.* [online]. 2004, 2013 [cit. 2013-04-16]. Dostupné z: http://www.vectorart.com/

[9] INTERNET TOP. *Mateřské školky, Mateřské školy, Mateřská škola, Mateřská školka, Seznam škol, Věci pro děti, Pro školy, Základní školy, Střední školy* [online]. [cit. 2013-04-07]. Dostupné z: http://www.materskeskolky.cz/

[10] Introduction to Java Platform, Enterprise Edition 6: An Oracle White Paper. In: *Oracle: Hardware and Software, Engineered to Work Together* [online]. 2010 [cit. 2013-04-07]. Dostupné z: http://www.oracle.com/us/products/middleware/application-server/050871.pdf

[11] Job Trends: Indeed.com. *Job Search: one search. all jobs. Indeed.com* [online]. 2012 [cit. 2013-04-07]. Dostupné z: http://www.indeed.com/jobtrends

[12] KANER, Cem. *Lessons learned in software testing: a context-driven approach*. New York: Wiley, c2002, xxvii, 286 s. :. ISBN 04-710-8112-4.

[13] KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. Brno: Computer Press, 2007. ISBN 80-251-1083-4.

[14] LEONARD, Anghel. *JSF 2.0 cookbook: over 100 simple but incredibly effective recipes for taking control of your JSF applications*. Birmingham, U.K.: Packt Pub., 2010, iv, 382 p. ISBN 978-1-847199-52-2.

[15] LICENIMAGES LTD. *Photl.com – Royalty Free Photo Stock: Download Images For Commercial Use From Free Photostock (Free Stock Photo)* [online]. 2009, 2013 [cit. 2013-04-16]. Dostupné z: http://www.photl.com/

[16] MySQL: Market Share. *MySQL: The world's most popular open source database* [online]. 2008 [cit. 2013-04-07]. Dostupné z: http://www.mysql.com/why-mysql/marketshare/

[17] ORACLE CORPORATION. *Welcome to NetBeans* [online]. 2012 [cit. 2013-01-18]. Dostupné z: http://netbeans.org/

[18] ORACLE. *GlassFish: Open Source Application Server - Java.net* [online]. 2012, 8.2.2013 [cit. 2013-03-20]. Dostupné z: http://glassfish.java.net/

[19] ORACLE. *Java SE Technologies - Database* [online]. [cit. 2013-03-20]. Dostupné z: http://www.oracle.com/technetwork/java/javase/jdbc/index.html

[20] ORACLE. *Lesson: Regular Expressions (The Java™ Tutorials > Essential Classes)* [online]. 1995, 2013 [cit. 2013-04-14]. Dostupné z: http://docs.oracle.com/javase/tutorial/essential/regex/index.html

[21] PAVLÍČEK, Radek. Přístupný web a jak se vyvarovat chyb. In: *Ministerstvo vnitra České republiky* [online]. 2009, 2010 [cit. 2013-04-12]. Dostupné z: http://www.mvcr.cz/clanek/pristupny-web-a-jak-se-vyvarovat-chyb.aspx

[22] PRIME TEKNOLOJI. *PrimeFaces* [online]. 2011 [cit. 2013-01-18]. Dostupné z: http://www.primefaces.org/

[23] PROCHÁZKA, Jaroslav a Cyril KLIMEŠ. *Provozujte IT jinak: agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb*. 1. vyd. Praha: Grada, 2011, 288 s. Průvodce (Grada). ISBN 978-80-247-4137-6.

[24] Ready for Business: Oracle GlassFish Server: An Oracle White Paper. In: *Oracle: Hardware and Software, Engineered to Work Together* [online]. 2013 [cit. 2013-04-07]. Dostupné z: http://www.oracle.com/us/products/middleware/application-server/ready-for-business-glassfish-wp-073701.pdf

[25] REFSNES DATA. *W3Schools Online Web Tutorials* [online]. 2013 [cit. 2013-01-21]. Dostupné z: http://www.w3schools.com/

[26] SCHALK, Chris, Ed BURNS a Neil GRIFFIN. *JavaServer Faces 2.0: The Complete Reference*. New York: McGraw-Hill, 2010. ISBN 978-0-07-162509-8.

[27] STOCKVAULT.NET. *Free Photos - Free Images | Stockvault.net - Free Stock Photos* [online]. 2013 [cit. 2013-04-16]. Dostupné z: http://www.stockvault.net/

[28] The Java EE 6 Tutorial. ORACLE CORPORATION. *Oracle | Hardware and Software, Engineered to Work Together* [online]. 2013 [cit. 2013-01-18]. Dostupné z: http://docs.oracle.com/javaee/6/tutorial/doc/

[29] VECTEEZY.COM. *Vecteezy! - Download Free Vector Art, Stock Graphics & Images...* [online]. 2013 [cit. 2013-04-16]. Dostupné z: http://www.vecteezy.com/

[30] VISION AUSTRALIA. *Colour Contrast Analyser 2-2 for Web Pages* [online]. 2012 [cit. 2013-04-16]. Dostupné z: http://www.visionaustralia.org/business-and-professionals/digital-access/resources/tools-to-download/colour-contrast-analyser-2-2-for-web-pages

[31] W3C. *How to Meet WCAG 2.0* [online]. 1994, 2007 [cit. 2013-04-16]. Dostupné z: http://www.w3.org/WAI/WCAG20/quickref/#qr-visual-audio-contrast-contrast

[32] ZAMBON, Giulio. *Beginning JSP, JSF and Tomcat: Java Web Development*. New York: Apress, 2012. 2. ISBN 978-1-4302-4623-7.

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam příloh

Příloha 1: Obsah CD

Součástí bakalářské práce je přiložené CD, které obsahuje:

1) Text bakalářské práce
2) Přílohy
    a. Příloha 5: Juicy Studio Colour Contrast Analyser
    b. Příloha 6: Class diagram
3) Databáze (skolka.sql)
4) Zdrojový kód aplikace

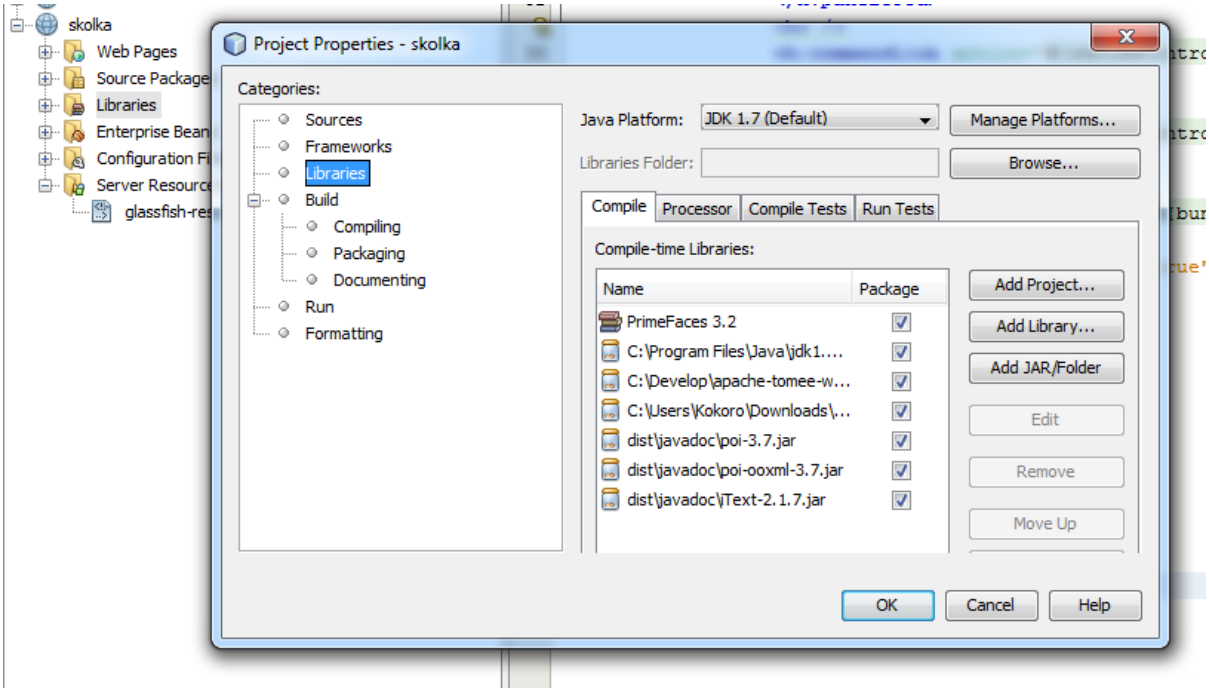
Příloha 2: Zavedení aplikace

Požadavky:

- Netbeans IDE 7.2.1
- Glassfish 3.1
- JSF 2.1
- PrimeFaces 3.2
- Soubor skolka.sql (na CD)

Postup:

1) Ve vývojovém prostředí Netbeans zvolte „Open project“ (Otevřít projekt).
2) Vyberte projekt „skolka“
3) Přidejte potřebné knihovny – pomocí pravého tlačítka myši na složce „Library“ vyberte položku „Properties“.



Obr. 1.: Library

4) Přidejte databázi

   Services → Databases → New Connection → MySQL (Connector /J driver)

V souboru *glassfish-properties.xml* změňte adresu databáze, viz atribut `value` v příkladu Př. 1.: Glassfish-properties.xml.

```
<property name="URL"
      value="jdbc:mysql://localhost:3306/skolka?zeroDateTimeBehavi
or=convertToNull"
/>
```

Př. 1.: Glassfish-properties.xml

5) Spusťte aplikaci.

Příloha 3: Juicy Studio Colour Contrast Analyser

Tato příloha se z důvodu formátu (HTML) nachází na přiloženém CD.

Příloha 4: Class diagram

Tato příloha se z důvodu formátu (HTML) nachází na přiloženém CD.

Příloha 5: Uživatelská příručka