Sem vložte zadání Vaší práce.

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

Bakalářská práce

Webová aplikace pro doučování matematiky

Pavel Brabec

Vedoucí práce: Ing. Zdeněk Rybola

3. května 2015

Poděkování

Zde bych rád poděkoval vedoucímu práce Ing. Zdeňku Rybolovi za cenné rady a podnětné připomínky. Velké díky patří také rodině a přátelům za podporu při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Černé u Bohdanče dne 3. května 2015 .....

Odkaz na tuto práci

Brabec, Pavel. *Webová aplikace pro doučování matematiky.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato bakalářská práce se zabývá tvorbou webové aplikace pro doučování matematiky. Uživateli této aplikace budou žáci základních škol a jejich rodiče. Aplikace bude usnadňovat zadávání a opravování příkladů z matematiky a žákům nabídne neomezenou sbírku příkladů. Součástí práce je analýza, návrh, implementace, testování a nasazení. Výsledkem bude prototyp aplikace nasazený na veřejně dostupném serveru v testovacím provozu.

Klíčová slova doučování matematiky, webová aplikace, Java, JSF, Spring, Hibernate.

Abstract

This bachelor thesis deals with development web application for math tutoring. Users of this application will be students of primary school and their parents. Application will facilite assigns and correcting of mathematics examples and for students provides unlimited collection of examples. This thesis describes analysis, design, implementation, testing and deployment. Result of thesis is application prototype deployed on public available server in test mode.

Keywords tutoring in mathematics, web application, Java, JSF, Spring, Hibernate.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Již delší dobu si všímám, jak rodiče žáků základních škol neumí řešit některé příklady z matematiky. Toto je problém hlavně ve chvíli, když potomek potřebuje doma matematiku procvičovat. Rodiče však neumí příklady opravit a proto jsou ochotni za doučování potomků zaplatit nemalé peníze. Z těchto důvodů jsem se rozhodl vytvořit webovou aplikaci, která by měla tyto problémy řešit.

Aplikace Doučmat bude umožňovat rodičům zadávat úkoly na procvičování online a zobrazovat přehledy o výsledcích testů. Úkoly bude možno sestavovat z libovolného počtu typových příkladů. Díky tomu bude umožněno zadávat úkoly přesně podle potřeb žáka. Příklady budou náhodně generovány a automaticky vyřešeny. Žáci tak budou mít k dispozici prakticky neomezenou sbírku příkladů. Důraz bude kladen na jednoduché a přehledné uživatelské rozhraní tak, aby jej zvládali i rodiče, kteří mají méně zkušeností s internetem, nebo nejmladší žáci.

V první části bude čtenář seznámen s aktuálním řešením tohoto problému. Tato část odkazuje na několik aplikací s podobnou funkčností, srovnává je se zamýšlenou aplikací a hodnotí jejich klady a zápory.

Druhá část této práce se zabývá analýzou problému. Jsou zde sepsány požadavky na aplikaci, vytvořen model případů užití a navržen analytický doménový model aplikace.

Třetí část se věnuje návrhu aplikace. V této kapitole jsou popsány technologie, které bude aplikace využívat. Dále je zde popsán návrhový, databázový model a architektura aplikace.

Čtvrtá část popisuje implementaci aplikace. Jsou zde stručně popsány komponenty, ze kterých se aplikace skládá, a způsob vykreslování matematických výrazů a grafů.

V páté části je popsán způsob testování aplikace a technologie použité k testování. Dále jsou zde zmíněny podporované prohlížeče.

Šestá část krátce popisuje proces nasazení aplikace.

Poslední částí práce je uživatelská příručka, která poskytuje uživatelům stručný návod k používání aplikace.

Kapitola 1

Současný stav řešení problematiky

Na internetu bylo nalezeno několik podobně zaměřených aplikací, ale žádná z nich nebyla zcela podle představ zamýšlené aplikace. Nejčastějšími nedostatky jsou složité a nepřehledné uživatelské rozhraní, nedostupnost české lokalizace aplikace nebo absence přehledu rodiče nad výsledky potomka. Aplikace Doučmat je cílena především na české uživatele, žáky základních škol a jejich rodiče. Pro tuto skupinu uživatelů je české rozhraní aplikace a přehledné uživatelské rozhraní naprosto nezbytné.

1.1 WolframAlpha

WolframAlpha [1] je webová služba, která odpovídá na otázky uživatele. Odlišnost od vyhledávačů spočívá v tom, že WolframAlpha se snaží nabídnout uživateli přímo odpověď. Výborně ovládá všechny disciplíny matematiky, ale umí odpovědět i na další otázky. Např: *weather 22.12.1992 in Prague* nebo *How old was Queen Elizabeth II in 1980?*. Konkurencí zamýšlené aplikace je hlavně Wolfram Problem Generator, který nabízí širokou škálu typových příkladů včetně postupu řešení. Problem Generator nenabízí přehled výsledků pro rodiče ani zadávání úkolů. Další nevýhodou může být absence českého překladu a to, že služba je zpoplatněna $5.49 měsíčně.

1.2 KhanAcademy

KhanAcademy [2] je komplexní služba pro doučování. Nabízí doučování mimo jiné matematiky, fyziky, chemie. Implementován je zde i dozor rodiče nad dítětem. Konkrétně pro základní školu nabízí velmi omezený počet typových příkladů. Další nevýhodou je pěkné ale nepřehledné uživatelské rozhraní a čás-

tečná absence českého překladu. Přeložena jsou pouze některá výuková videa s titulky.

1.3 MathGoodies

Služba MathGoodies [3] je pouhý generátor zadaní bez možnosti registrace a sledování statistik. Využít by se dala jako generátor testů pro učitele. Služba nabízí nepřehledné a zastaralé uživatelské rozhraní. Chybí česká lokalizace.

1.4 e-matematika

Projekt e-matematika [4] je jedním ze zástupců s českou lokalizací. Jako ostatní podobné české služby nabízí pouze omezený počet příkladů. Většina výsledků a postupů řešení je zpoplatněna.

1.5 Doučování se soukromým lektorem

Doučování se soukromým lektorem je asi nejefektivnější metodou, ale zároveň má spoustu nevýhod. Výhodou je, že lektor může dítě opravovat již při výpočtu, přesně najít chybu v postupu nebo vymýšlet příklady „na míru“. Doučování s lektorem je časově poměrně náročné, protože se za lektorem musí často dojíždět, a zároveň se musí oba účastníci shodnout na časovém termínu. Další nevýhodou je vysoká cena za doučování.

1.6 Shrnutí současného stavu řešení problematiky

Aplikace Doučmat by měla skloubit výhody služeb WolframAlpha a Khan-Academy, které jsou považovány za největší konkurenty. Navíc přidá zadávání úkolu dítěti online, kontrolu rodiče nad dítětem a zobrazování statistik. Ostatní weby jsou pouze sbírky, které nabízejí pouze omezený počet příkladů.

Kapitola 2

Analýza

2.1 Požadavky na aplikaci

Požadavky na aplikaci nám poskytují rámcový přehled o budoucí funkčnosti aplikace. Hlavním úkolem požadavků je zachytit, jakou funkčnost bude budoucí informační systém poskytovat. Zpřesňují tak představu o aplikaci jak na straně zákazníka tak na straně dodavatele. Požadavek je definován jako „specifikace toho, co by mělo být implementováno“, proto by měl odpovídat na otázku, *co* bude systém dělat, nikoliv *jak*. Rozlišujeme dva základní typy požadavků:

- **Funkční** - určují, jakou funkčnost bude budoucí systém nabízet
- **Nefunkční** - specifikují vlastnosti budoucího systému např: bezpečnost, škálovatelnost, dostupnost

Sběru požadavků je přikládána velká důležitost také proto, že jsou hlavním podkladem pro odhad časové i finanční náročnosti vývoje aplikace. Podle několika studií [5, 6] jsou chyby při sběru požadavků velice častým faktorem selhání projektu. Jazykem UML není specifikováno, jak bychom měli požadavky zapisovat. Autoři [7] doporučují následující formát zápisu požadavku:

<id> <název systému> **bude** <popis funkce>

Důležité je všechny požadavky jednoznačně definovat proto, aby nedocházelo ke sporům se zákazníkem při akceptačních testech. Ke sporům může docházet na příklad u požadavku na rychlost systému, kde velice pravděpodobně bude mít každá ze stran jinou představu o tom, co znamená rychlý systém. K popisu aktivit spojených se sběrem požadavků, jejich dokumentováním a údržbou používáme termín inženýrství požadavků (requirements engineering). [7, 8]

2.1.1 Funkční požadavky

Seznam funkčních požadavků aplikace Doučmat vypadá takto:

F01 Registrace
Aplikace bude umožňovat uživateli se registrovat ve dvou rolích, a to buď jako rodič nebo jako žák. Při registraci bude vyžadováno jméno uživatele, e-mail, uživatelské jméno a heslo dlouhé alespoň 5 znaků.

F02 Přihlášení se
Aplikace bude umožňovat registrovanému uživateli se přihlásit. Pro přihlášení bude potřeba uživatelské jméno a heslo.

F03 Odhlášení se
Aplikace bude umožňovat přihlášenému uživateli se odhlásit.

F04 Zadávání úkolů
Aplikace bude umožňovat uživatelům registrovaným jako rodič zadávání úkolů žákům, které mají pod svým dohledem. Uživatelům registrovaným jako žák bude umožněno zadávat si dobrovolné úkoly.

F05 Řešení úkolů
Aplikace bude umožňovat uživatelům registrovaným jako žák řešit úkoly dobrovolné i zadané rodičem. Historie řešených úkolů bude ukládána.

F06 Generování příkladů
Aplikace bude generovat náhodné příklady z algebry v rozsahu základní školy a nalezne jejich správné výsledky.

F07 Zobrazování statistik
Aplikace bude umožňovat uživateli zobrazit statistiky o plnění úkolů a úspěšnosti.

F08 Správa úkolů
Aplikace bude umožňovat zobrazení přehledu vypracovaných a nevypracovaných úkolů u každého žáka.

F09 Správa žáků
Aplikace bude umožňovat přiřadit žáka k rodiči na základě rodičovy žádosti. Aplikace rodiči umožní přiřazení odstranit.

F10 Notifikace
Aplikace bude umožňovat odesílání notifikací uživatelům e-mailem.

2.1.2 Nefunkční požadavky

Seznam nefunkčních požadavků aplikace Doučmat vypadá takto:

N01 Webové rozhraní
Aplikace bude dostupná přes webové rozhraní pomocí prohlížečů Google Chrome 41 [9], Mozilla Firefox 37 [10], Internet Explorer 11 [11] a pomocí novějších verzí těchto prohlížečů. Speciální rozhraní pro mobilní telefony nebo tablety není vyžadováno.

N02 Bezpečnost
Citlivé údaje jako např. hesla k uživatelským účtům, nebude aplikace uchovávat jako prostý text, ale jako jejich otisk neboli hash vhodné hashovací funkce.

N03 Lokalizace
Aplikace bude v českém jazyce. Překlad do ostatních jazyků nebude vyžadován.

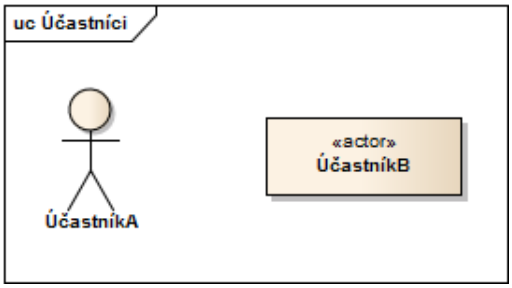
2.2 Model případů užití

Další činností inženýrství požadavků je modelování případů užití. Modelování případů užití je dalším doplňkovým způsobem zachycení požadavků na systém. Výstupem modelování případů užití by měly být následující komponenty: [7]

- **Účastníci** - množina rolí uživatelů, kteří budou využívat systém. Účastníkem může být také jiná aplikace, která s vyvíjenou aplikací komunikuje nebo spolupracuje. V případě, že systém spouští některé operace závisle na čase např: každý den o půlnoci, může být účastníkem také čas.
- **Případy užití** - činnosti, které mohou uživatelé systému vykonávat.
- **Relace** - vztahy mezi účastníky a případy užití.
- **Hranice systému** - Ohraničení kolem případů užití, které znázorňuje hranice funkčnosti systému. Účastníci jsou vždy znázorněni až za hranicí systému.

2.2.1 Účastníci

Jazyk UML znázorňuje účastníky buď jako obrázek figurky nebo jako obdélník. UML nedává žádné doporučení jaký ze symbolů používat. Přesto je zvykem pro přidělení rolí osobám používat figurku a pro přidělení rolí jiným systémům obdélník. [7] Ukázku zachycení uživatelů v jazyce UML vidíme na obrázku 2.1.



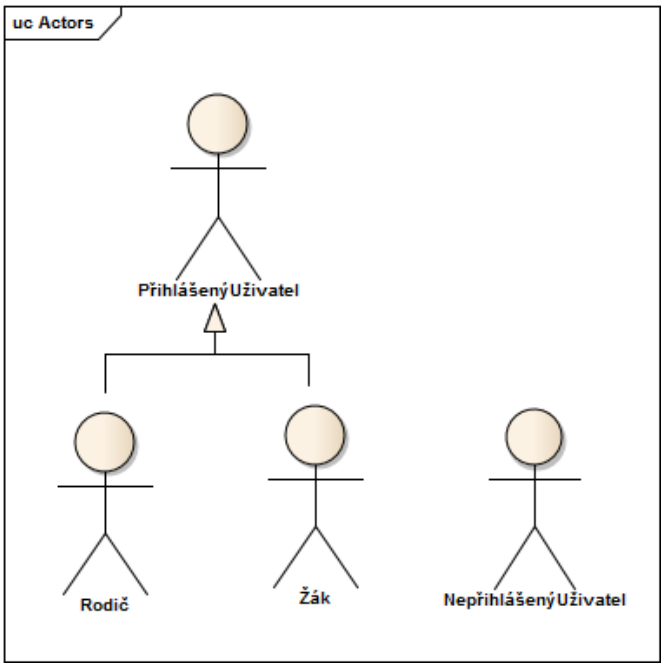
Obrázek 2.1: Znázornění uživatelů v jazyce UML

Aplikaci Doučmat budou užívat účastníci rodič, žák a neregistrovaný uživatel. Účatníci jsou zachyceni na obrázku 2.2.

Rodič
: Rodič může svým žákům zadávat úkoly a sledovat jejich výsledky formou souhrnných statistik nebo zobrazením výsledků vypracovaných úkolů.

Žák
: Žák vypracovává úkoly zadané rodičem, ale může vypracovávat i dobrovolné úkoly. Historie jeho úkolů je ukládána. Každý žák má maximálně jednoho rodiče. Rodič je přiřazen na základě své žádosti, kterou žák musí potvrdit.

Neregistrovaný uživatel
: Neregistrovaný uživatel se může registrovat jako žák nebo rodič. Po registraci a přihlášení mu je zpřístupněna funkčnost aplikace pro vybranou roli.



Obrázek 2.2: Uživatelé aplikace Doučmat

2.2.2 Případy užití

Každý případ užití (use case) by měl odpovídat jedné funkcionalitě systému. Měl by být popsán jako posloupnost zpráv, které si vyměňují systém a jeho uživatel za účelem dosažení požadovaného výsledku. Účelem případu užití je popsat část souvislého chování systému bez odhalení vnitřní implementace. [12] Pro přehlednost byly případy užití systému Doučmat rozděleny do několika balíčků.

2.2.2.1 Správa žáků

Balíček Správa žáků obsahuje případy užití, které se týkají dozoru rodiče nad žákem a žádostí o dozor.

UC01 Zobrazit žáky

Aplikace zobrazí všechny žáky pod kontrolou rodiče.

UC02 Vymazat žáka

1. Případ užití začíná, když rodič chce ukončit kontrolu nad jedním ze svých žáků.
2. Rodič vybere položku Přehled studentů.
3. INCLUDE „Zobrazit žáky“.
4. Klepne na odkaz Odebrat kontrolu u vybraného žáka.

5. Aplikace ukončí kontrolu rodiče nad tímto žákem.
6. Žák je informován e-maillem.

UC03 Zaslání žádosti o dohled

1. Případ užití začíná, když rodič chce zadávat úkoly a sledovat statistiky svého dítěte.
2. Rodič vybere položku Žádosti o dozor.
3. Aplikace zobrazí formulář pro vyplnění uživatelského jména dítěte.
4. Rodič vyplní uživatelské jméno studenta a stiskne tlačítko Odeslat.
5. Aplikace odešle dítěti notifikaci.

UC04 Přijmout žádost rodiče

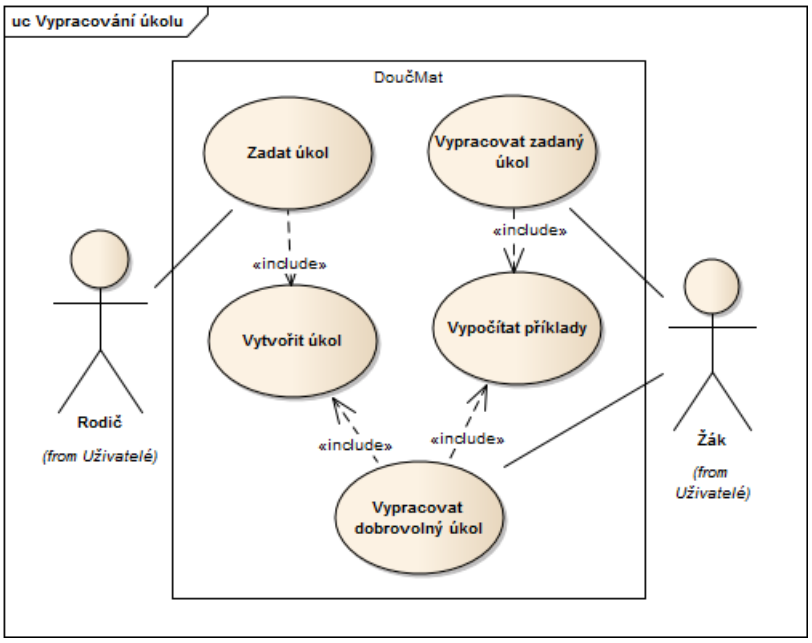
Žák může schválit žádost rodiče o dohled. Schválením umožňuje rodiči zadávat úkoly a sledovat jeho statistiky.

UC05 Zamítnout žádost rodiče

Žák může zamítnout žádost rodiče o dohled. Zamítnutím žádosti zamezí sledovaním vlastních výsledků cizí osobou.

2.2.2.2 Vypracování úkolu

Balíček případů užití Vypracování úkolu je vyobrazen na obrázku 2.3.



Obrázek 2.3: Případ užití - Vypracování úkolu

UC06 Zadat úkol

1. Případ užití začíná, když se rodič rozhodne zadat úkol.
2. Z přehledu svých studentů rodič zvolí žáka a vybere Zadat úkol.
3. INCLUDE „Vytvořit úkol“.
4. Žák obdrží notifikaci o tom, že dostal nový úkol.

UC07 Vytvořit úkol

1. Aplikace zobrazí seznam dostupných typových příkladů.
2. Uživatel přidá do košíku vybrané typové příklady.
3. Aplikace zobrazí vygenerované příklady.
4. Aplikace umožní uživateli smazat nevhodné příklady.
5. Pokud chce uživatel přidat další příklady, pokračuje bodem 4.
6. Uživatel vybere možnost Zadat úkol a tím je úkol vytvořen.

UC08 Vypočítat příklady

1. Aplikace zobrazí zadání příkladu a pole pro vyplnění výsledku.
2. Žák příklad vyřeší a výsledek napíše do pole pro výsledek.
3. Žák stiskne tlačítko uložit.
4. Aplikace výsledek uloží, ale zároveň zachová možnost pozdější editace.
5. Pokud není vyplněný příklad poslední, žák stiskne tlačítko Další a pokračuje bodem 1.
6. Aplikace zobrazí tlačítko Odevzdat úkol.
7. Klepnutím na tlačítko Odevzdat úkol žák odevzdá úkol a přijde o možnost editace výsledků.
8. Aplikace zobrazí přehled příkladů z úkolu včetně správných výsledků.

UC09 Vypracovat zadaný úkol

1. Případ užití začíná, když žák obdrží notifikaci o zadání úkolu.
2. Žák se přihlásí do systému.
3. Aplikace zobrazí seznam nevypracovaných úkolů.
4. Uživatel klepnutím vybere, který úkol chce vypracovat.
5. INCLUDE „Vypočítat příklady“. 6. Rodič je upozorněn e-mailem.

UC10 Vypracovat dobrovolný úkol

1. Případ užití začíná, když se žák rozhodne vypracovat dobrovolný úkol.
2. Žák vybere možnost Vytvořit si úkol.
3. INCLUDE „Vytvořit úkol“.
4. INCLUDE „Vypočítat příklady“.

2.2.2.3 Zobrazení statistik

UC11 Zobrazení statistik

Aplikace zobrazí statisky formou grafů o příkladech, které student vypracoval.

2.2.2.4 Správa úkolů

UC12 Zobrazení vypracovaných úkolů

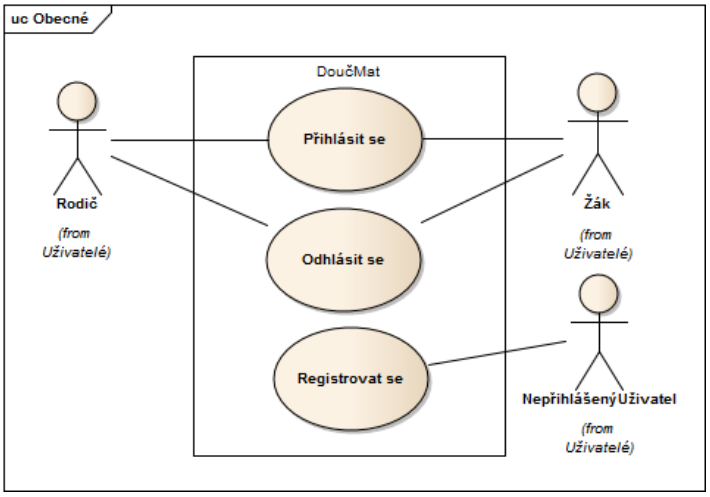
Aplikace zobrazí přehled vypracovaných úkolů. U každého úkolu bude dostupný detail všech příkladů. Detail příkladů bude obsahovat zadání, správné řešení a studentův výsledek.

UC13 Zobrazení nevypracovaných úkolů

Aplikace zobrazí seznam úkolů, které student nevypracoval.

2.2.2.5 Obecné

Balíček případů užití Obecné je vyobrazen na obrázku 2.4.



Obrázek 2.4: Případ užití - Obecné

UC14 Registrovat se

1. Případ užití začíná, když neregistrovaný uživatel chce začít využívat systém.
2. Uživatel vybere možnost Registrovat se.
3. Aplikace zobrazí registrační formulář.
4. Uživatel vyplní registrační formulář a stiskne tlačítko Registrovat se.
5. Aplikace zobrazí formulář pro přihlášení uživatele.
6. Uživatel se přihlásí vyplněním formuláře a po přihlášení může plně využívat aplikaci.

UC15 Přihlásit se

1. Případ užití začíná, když registrovaný uživatel chce začít pracovat s aplikací.
2. Uživatel vybere možnost Přihlásit se.

3. Aplikace zobrazí formulář přihlášení.
4. Uživatel vyplní formulář a po stisknutí tlačítka Přihlásit se může začít plně využívat aplikaci.

UC16 Odhlásit se

1. Případ užití začíná, když uživatel chce ukončit práci s aplikací.
2. Uživatel vybere možnost Odhlásit se.
3. Aplikace uživatele odhlásí a zobrazí formulář přihlášení.

2.3 Tabulka pokrytí funkčních požadavků

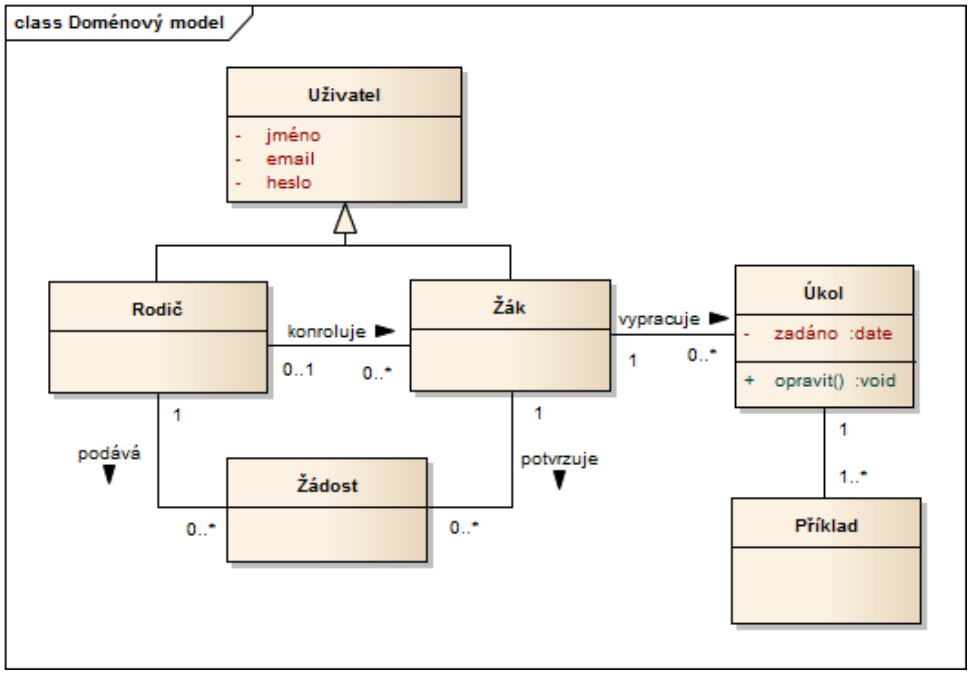
Tabulka pokrytí funkčních požadavků slouží k tomu, abychom si ověřili skutečnost, že každý funkční požadavek je pokryt alespoň jedním případem užití a naopak. Tabulku vytvoříme tak, že do řádků vložíme jednotlivé případy užití a do sloupců funkční požadavky. Skutečnost, že funkční požadavek je pokryt případem užití vyznačíme tak, že políčko na jejich průniku označíme. Pokud pomocí tabulky pokrytí funkčních požadavků najdeme funkční požadevek, ke kterému není přiřazen žádný případ užití, znamená to, že model případů užití nemáme kompletní. Toto tvrzení platí i opačně. Pokud nalezneme případ užití, ke kterému není přiřazen žádný funkční požadavek, tak víme, že soupis funkčních požadavků není kompletní. [7]

Tabulka 2.1: Pokrytí funkčních požadavků

	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10
UC01									+	
UC02									+	+
UC03									+	+
UC04									+	+
UC05									+	+
UC06				+		+				+
UC07				+		+				
UC08					+					
UC09					+					+
UC10				+	+	+				
UC11							+			
UC12								+		
UC13								+		
UC14	+									+
UC15		+								
UC16			+							

2.4 Analytický doménový model

Na základě předchozích kroků analýzy byl vytvořen analytický doménový model. Doménový model by měl sloužit jako náhled na daný problém z vyšší úrovně abstrakce. Skládá se z doménových tříd a asociací mezi nimi. Doménové třídy obsahují pouze nejdůležitější atributy. Implementační detaily tyto třídy nezachycují. Důležitým atributem doménové třídy je její název, který by měl co nejpřesněji odrážet její účel. [7] Doménový model aplikace Doučmat je znázorněn na obrázku 2.5.



Obrázek 2.5: Analytický doménový model aplikace Doučmat

Z analytického doménového modelu vyplývá, že aplikace nebude připouštět existenci uživatele v roli žáka a zároveň rodiče. Aplikace bude evidovat pouze nevyřízené žádosti o dozor, po kladném nebo záporném vyjádření k žádosti ze strany žáka nebo stažení žádosti ze strany rodiče aplikace žádost nebude dále evidovat.

Uživatel
: Doménová třída reprezentující registrovaného uživatele.

Rodič
: Doménová třída reprezentující uživatele registrovaného jako rodič.

Žák
: Doménová třída reprezentující uživatele registrovaného jako student.

Žádost
: Doménová třída reprezentující nevyřízený rodičův požadavek na kontrolu žáka.

Úkol
: Doménová třída reprezentující zadání úkolu.

Příklad
: Doménová třída reprezentující jeden příklad.

Kapitola 3

Návrh

3.1 Použité technologie

Aplikace Doučmat využívá open-source nebo volně dostupných technologií založených na platformě Java. Nezpochybnitelnou výhodou těchto technologií je jejich pořizovací cena. U open-source technologií musíme ale počítat se složitější konfigurací a neucelenostií jednotlivých řešení, takže při vývoji strávíme více času konfigurací, než by tomu pravděpodobně bylo u komplexních komerčních řešení.

3.1.1 Java

Java [13] je objektově orientovaný, interpretovaný programovací jazyk vyšší úrovně s vysokým stupněm zabezpečení. Veřejnosti byl poprvé představen v květnu 1995 firmou SunMicrosystems, v současnosti jej spravuje Oracle Corporation. Syntaxe tohoto jazyka vychází z jazyka C. Programy v Javě se kompilují do tzv. *byte kódu*, který lze sputit všude tam, kde je dostupné JRE (Java Runtime Enviroment). V Javě je implementována automatická správa paměti tzv. *garbage collector*, díky kterému se programátoři nemusí starat o uvolňování paměti.[14] Java se dělí do několika edicí, např: [15]

- **Java SE** - edice určená pro vývoj desktopových a serverových aplikací.
- **Java EE** - edice určená pro vývoj podnikových systémů.
- **Java ME** - edice určená pro vývoj aplikací pro mobilní telefony a vestavná zařízení.

3.1.2 Apache Tomcat

Apache Tomcat [16] je lightweight servlet kontejner, je vyvíjený pod licencí Apache Licence version 2 [17]. Mezi jednu z jeho nevýhod oproti klasickým aplikačním serverům patří absence Enterprise Java Beans kontejneru. Tato

skutečnost je při použití Springu zcela zanedbatelná. Mezi výhody patří nízké nároky na webhosting, díky které je Apache Tomcat často nabízen u zdarma dostupných hostingů.

3.1.3 JavaSever Faces

JavaServer Faces [18] je framework pro vývoj webových aplikací založených na platformě Java. Byl navržen tak, aby odděloval uživatelské rozhraní od aplikační logiky. Součástí JSF jsou mechanizmy pro konverzi a validaci vstupních hodnot, internacionalizaci a navigaci. JSF umožňuje snadné vytváření šablon webových stránek pomocí tzv. *templates*. [19]

3.1.4 Spring

Spring [20] je lightweight kontejner kontrolující životní cyklus Spring bean. Za Spring beanu považujeme jakýkoli objekt ovládaný Springem. Tyto objekty jsou někdy nazývány jako Spring managed object. Hlavním důvodem využití Springu je injekce závislostí tzv. *dependency injection*, která je někdy také označována jako Inversion of Control. Konfigurace injekce závislostí se obvykle ukládá do externího XML souboru. Protože konfigurace injekce závislostí není uložená ve zdrojovém kódu, aplikace se rozpadne do nezávislých částí, které můžeme znovu používat nebo testovat. Další pozitivní vlastnost Springu je jeho neinvazivnost. Spring beany nemusí implementovat žádný interface nebo dědit od jiné třídy. [21]

3.1.5 Hibernate

Hibernate [22] je framework pro objektově-relační mapování (ORM). Mapuje objekty do tabulek relačních databází, ale i naopak umožňuje vytvářet objekty z dat uložených v databázích. Mapování můžeme konfigurovat XML souborem nebo anotacemi. Další vlastností tohoto frameworku je odstínění vývojáře od databáze. Vývojář využívající Hibernate nemusí vědět, s jakou databází pracuje. Pro změnu databáze stačí v konfiguračním souboru změnit Hibernate dialekt. Hibernate obsahuje implementaci Java Persistent API (JPA).

3.1.6 MySQL

Data aplikace Doučmat budou uchovávána v MySQL [23] databázi. Podle [24] je MySQL nejpopulárnější open-source databáze na světě. Vývoj MySQL byl započat již v roce 1980 ve Švédsku. Nyní projekt MySQL spravuje Oracle Corporation.

3.1.7 Symja

Symja [25] je CAS (Computer Algebra System) zajišťující matematické výpočty Doučmatu. Dokumentace tohoto projektu neobsahuje žádný návod k použití knihovny. Všechny informace pro užívání je nutné vyčíst z dostupných testů. Vývoj tohoto projektu je poměrně aktivní, což by mohl být příslib do budoucna. Projekt je rozdělen na několik částí, všechny tyto části jsou licencovány licencí LGPL [26] nebo Apache Licence version 2. [17]

3.2 Architektura aplikace

Softwarová architektura zachycuje realizaci nefunkčních požadavků. Při jejím návrhu bychom měli dbát zvýšené opatrnosti. Chyby při návrhu softwarové architektury mohou v budoucnu velice komplikovat a prodražovat údržbu celého projektu. [27]

3.2.1 Architektura Doučmatu

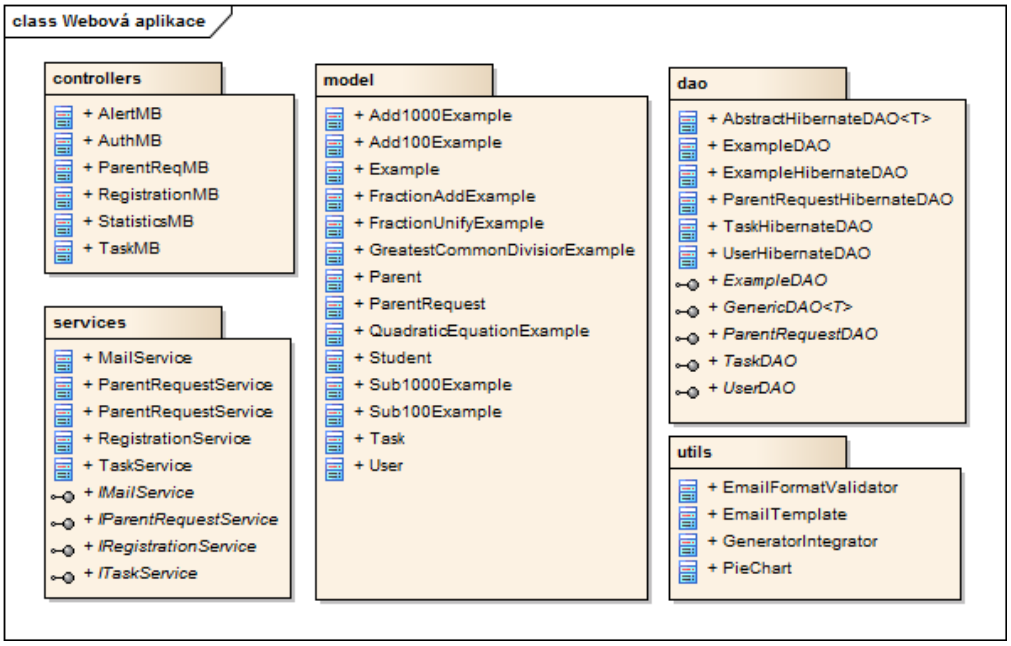
Architektura aplikace Doučmat částečně vychází z návrhového vzoru Model View Controller. Odlišnost spočívá ve vyčlenění logiky aplikace do zvláštní části Business Logic. Oddělením části Business Logic se zpřehlední zdrojový kód a aplikace se rozdělí na více zcela nezávislých a lépe testovatelných částí. Podrobná architektura aplikace Doučmat je zachycena na obrázku v příloze C.

3.3 Návrhový model tříd

Základ návrhového modelu tříd vychází z analytického doménového modelu, který byl popsán v kapitole Analýza. Tento model již zachycuje více detailů. Začínáme rozlišovat typy atributů včetně jejich viditelnosti. Dále v návrhovém modelu tříd přiřazujeme třídám jejich zodpovědnosti definováním jednotlivých metod včetně vstupních i výstupních paramametrů. Pomocí vhodného CASE (Computer-Aid Software Engineering) nástroje např: Enterprise Architect [28] můžeme vygenerovat předlohy modelovaných tříd pro programátory. Proto je vhodné tento model modelovat co nejdetailněji.
Oproti analytickému datovému modelu přibyli potomci třídy Example, kteří zachycují jednotlivé typové příklady, se kterými bude aplikace pracovat. Návrhový model tříd je zachycen na obrázku v příloze D.

3.3.1 Popis tříd a balíčků

Třídy Doučmatu byly rozděleny do několika balíčků tak, aby se zdrojový kód stal co nejpřehlednější. Přehled balíčků a tříd je znázorněn na obrázku 3.1.



Obrázek 3.1: Přehled balíčků a tříd webové aplikace

3.3.1.1 Balíček controllers

Tento balíček obsahuje 7 tříd, které podle MVC zastupují controllery. Jsou implementovány jako JSF ManagedBeany. Úkolem těchto tříd je odebírat požadavky uživatele a předat je části Business Logic nebo aktualizovat model. Tyto třídy se také starají o udržování aktualních dat ve view.

3.3.1.2 Balíček dao

Dalším balíčkem je balíček dao. Tento balíček obsahuje rozhraní a jejich implementace, pomocí kterých Doučmat přistupuje k datům v databázi. Třídy v tomto balíčku jsou navrženy podle návrhového vzoru Data Access Object.

3.3.1.3 Balíček model

Balíček model obsahuje třídy, které reprezentují doménový model aplikace. Tyto třídy jsou pomocí tříd z balíčku dao mapovány do tabulek databáze.

3.3.1.4 Balíček services

Třídy v balíčku services zajišťují logiku aplikace Doučmat. Jejich úkolem je zajistit funkčnost týkající se žádostí rodičů o kontrolu, registraci nových uži-

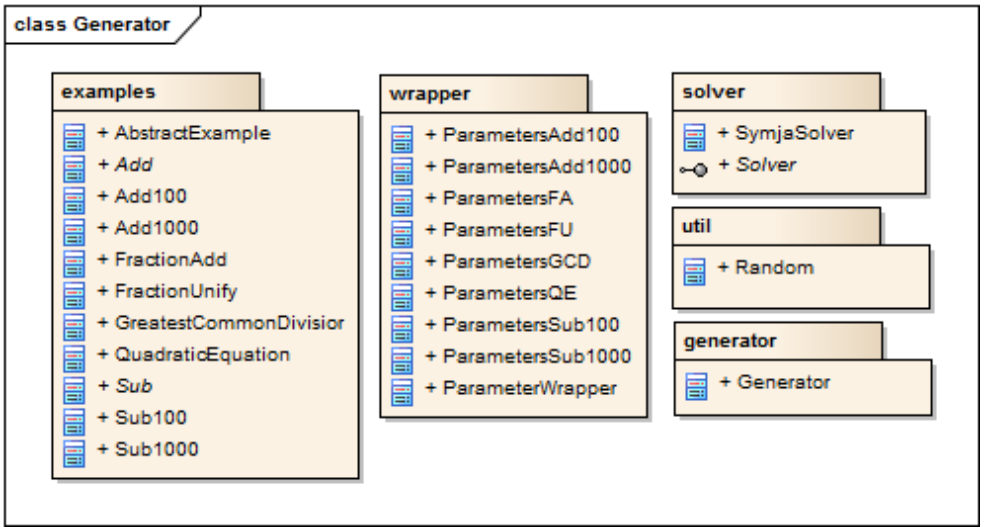
vatelů a funkčnost spojenou s řešením úkolů. V tomto balíčku je také obsažena třída MailService, která zajišťuje odesílání notifikací e-mailem.

3.3.1.5 Balíček utils

Balíček utils obsahuje několik pomocných tříd a složitější validátory vstupních polí. Třída EmailTemplate slouží ke generování obsahu e-mailu z předpřipravených šablon. Třída GeneratorIntegrator zajišťuje integraci webové aplikace a Generátoru. GeneratorIntegrator využívá návrhového vzoru Double dispatch. Třída PieChart reprezentuje data potřebná k vykreslení grafu.

3.3.2 Generátor

Generátor bude samostatná aplikace, kterou bude Doučmat využívat pro generování a výpočet příkladů. Další funkcí Generátoru bude porovnávání výsledků a export zadání příkladu do formátu LaTeX. V některých případech Generátor využívá pro výpočet nebo porovnání výsledků knihovnu Symja [25]. Přehled balíčků a tříd je znázorněn na obrázku 3.2.



Obrázek 3.2: Přehled balíčků a tříd Generátoru

3.4 Databázový model

Relační databázový model Doučmatu vychází z návrhového modelu tříd z kapitoly 3.3. Úkolem tohoto modelu je zachytit strukturu uložení dat v databázi. Z databázového modelu byl vytvořen DDL (Data definition language) skript, pomocí kterého byly vytvořeny potřebné tabulky v databázi. Tvorbu DDL

z databázového modelu lze automatizovat použitím vhodného CASE nástroje. Databázový model Doučmatu je zachycen na obrázku v příloze E.

3.4.1 Mapování dědičnosti

V návrhovém modelu tříd jsou dvě hierarchie dědičnosti, které je nutné namapovat do relační databáze. Podle [29] existují tři způsoby mapování dědičnosti do relační databáze.

- Mapování celé hierarchie do jedné tabulky
- Tabulka pro každý podtyp hierarchie
- Tabulka pro každý typ v hierarchii

Pro mapování obou hierarchií byla zvolena strategie „Mapování celé hierarchie do jedné tabulky“. Tato strategie v jedné databázové tabulce zachytí všechny atributy tříd v dané hierarchii a přidá sloupec, který autoři Hibernate [22] označují jako „discriminator“. Sloupec discriminator označuje, který podtyp hierarchie zachycuje daný řádek. V některých případech může být tato strategie značně nevýhodná pro svoji paměťovou náročnost. Výhodou je, že pokud chceme přidat nový podtyp hierarchie, který obsahuje pouze atributy již zachycené v hierarchii, nemusíme pro tento podtyp tvořit novou tabulku v databázi. Tento důvod byl velice důležitý u volby strategie mapování typových příkladů Doučmatu. U volby strategie pro mapování uživatelů byla důležitá skutečnost, že mechanizmus přihlašování uživatelů u Apache Tomcat [16] požaduje mít všechny uživatele v jedné databázové tabulce.

3.5 Bezpečnost

Jedním z nefunkčních požadavků je požadavek na bezpečnost uložených hesel. Tento požadavek požaduje abychom hesla uživatelů aplikace neukládali jako prostý text ale jako otisk hešovací funkce. Ukládání hesel tímto způsobem je preferováno z důvodu, že hesla jsou nečitelná pro administrátory serveru i potenciální útočníky.

3.5.1 Hešovací funkce

Hešovací funkce (hash function) je funkce chovající se jako *náhodné orákulum*. Orákulum je libovolný stroj, který odpovídá na vstup výstupem. Jedinou jeho vlastností je, že na stejný vstup odpoví stejným výstupem. Náhodné orákulum je orákulum, které na vstup odpoví výstupem, který byl zvolen náhodným výběrem výstupu z množiny výstupů. [30]

```
pavel   fde9ed552da6bd1d107aaf6a05dcf04ae12ba0ad8f4e506e8b...
Pavel   e4a14c8d1a69d1b514a0b112d6290e5101b77a9d04916b0660...
```

V ukázce vidíme vstupy a počátky odpovídajících výstupů hešovací funkce SHA-512. Vidíme, že malá změna na vstupu, jako je změna velikosti prvního písmene, vyvolá velkou změnu na výstupu hešovací funkce.
Protože hešovací funkce je jako zobrazení z nekonečné množiny vstupů na množinu výstupů jasně definované velikosti, nastávají při hešování kolize. O kolizi hovoříme, pokud dva různé vstupy hašovací funkce mají stejný výstup. Najít takové kolize je u moderních hešovacích funkcí početně velice náročný až prakticky neřešitelný problém. Pokud existuje způsob, kterým dokážeme u nějaké hešovací funkce nalézt kolize v krátém čase, hovoříme o prolomení této hešovací funkce. Příkladem prolomené hešovací funkce je funkce MD5. [30]

3.5.2 Navržené řešení

Jedním z možných řešení ukládání hesel do databazí je metoda solení. Tato metoda před zápisem hesla do databáze vygeneruje náhodný řetězec tzv. *sůl*, následně uloží dvojci sůl, heš(heslo+sůl). Takto uložená hesla jsou odolnější proti slovníkovým útokům. Pro zjednodušení implementace aplikace Doučmat metodu solení využívat nebude, bude zcela dostačující ukládat hesla jako heše hešovací funkce SHA-512.

Kapitola 4

Implementace

Aplikace Doučmat byla rozdělena na dvě nezávislé části. První částí je webová aplikace, která zajišťuje webové rozhraní včetně jeho obsluhy, aplikační logiku a ukládání dat do databáze. Druhou částí je Generátor, který pro webovou aplikaci zajišťuje generování a řešení příkladů. Důležitou funkcí Generátoru je převod zadání příkladů do formátu LaTeX, který je využit pro zobrazní matematických výrazů uživateli. Pro vývoj bylo využito vývojové prostředí NetBeans IDE 8.0.2 [31]. Konkrétní verze použitého softwaru popsaného v kapitole 3.1 jsou:

- Java 1.7.0u67
- Apache Tomcat 8.0.18
- JavaServer Faces 2.2.10
- Spring 4.1.5
- Hibernate 3.6.10
- MySQL 5.5.41
- Symja 2015-01-24

4.1 Validace vstupů

Všechny vstupy do aplikace zadané uživatelem je třeba validovat. Validací zamezíme uživateli vložit data, která neodpovídají požadovanému vstupu nebo by mohla způsobit pád aplikace.

4.1.1 Validace vstupních polí

JavaServer Faces nabízí jednoduchý ale velice účinný a pohodlný mechanizmus validace vstupních dat. Připraveny jsou validátory například numerických hodnot, délky vstupního řetězce nebo validátory formátu vstupního řetězce pomocí regulárního výrazu. V případě, že JSF potřebný validátor nenabízí, je možné implementovat vlastní validátor.

```
<h:outputLabel for="name" value="Jméno: "/>
<h:inputText id="name" value="#{registration.name}"
  required="true" requiredMessage="Toto pole je vyžadováno."
  validatorMessage="Toto pole může obsahovat maximálně 30 znaků.">
  <f:validateLength maximum="30" />
</h:inputText>
<h:message for="name" class="error" />
```

Na ukázce vidíme validaci vstupního pole pro zadání jména v registračním formuláři. Validátor kontroluje, jestli je jméno vyplněno, a pokud není, vypíše hlášku „Toto pole je vyžadováno“. Dalším bodem kontroly je kontrola délky vstupního řetězce, která je validátorem omezena na 30 znaků. Tato kontrola je velice důležitá, protože pokud by uživatel zadal řetězec delší než 30 znaků, tak by pokus o uložení takového řetězce způsobil chybu aplikace.

4.1.2 Validace formátu emailu

Validace e-mailu je poměrně složitá záležitost, která si ve finální verzi vyžádala implementaci vlastního JSF validátoru. V průběhu vývoje aplikace bylo vyzkoušeno několik možností, jak e-mail validovat. Prvním pokusem bylo vytvoření regulárního výrazu, který by kontroloval správnost formátu e-mailové adresy. Vzniklý regulární výraz byl však velice složitý a ani z daleka nepokrýval všechny odchylky od požadovaného formátu. Dalším možným řešením byl validator e-mailové adresy z knihovny Hibernate [22], ale ani ten nedosahoval potřebných kvalit. Finální implementace využívá projektu Apache Commons Validator. [32]

```
public boolean validateFormat(String email) {
     return EmailValidator.getInstance().isValid(email);
}
```

Z ukázky vidíme, že validace e-mailové adresy pomocí Apache Commons Validatoru je opravdu pohodlná a snadná.

4.2 Implementace typových příkladů

Všechny třídy reprezentující typové příklady, se kterými Generátor pracuje, jsou potomci abstraktní třídy AbstractExample. Pro implementaci typového

příkladu je nutné vytvořit novou třídu, která bude potomkem třídy AbstractExample a bude implementovat všechny její abstraktní metody.

```
protected abstract void generateExample();
protected abstract void solveExample();
protected abstract boolean compareResult();
```

- Metoda generateExample vygeneruje náhodné zadání příkladu.
- Metoda solveExample vypočítá výsledek příkladu.
- Metoda compareResult testuje ekvivalenci mezi výsledky studenta a Generátoru.

Dále je nutné vytvořit potomka třídy ParametersWrapper. Objekty této nově vzniklé třídy dávají Generátoru informaci o parametrech nově generovaného příkladu.

4.2.1 Implementace typového příkladu kvadratická rovnice

Pro ilustraci implementace typového příkladu byl vybrán příklad na výpočet kořenů kvadratické rovnice.

4.2.1.1 Implementace metody generateExample

Prvním krokem generování zadání kvadratické rovnice je náhodné vygenerování kořenů kvadratické rovnice r_1, r_2 a nenulové konstanty α. Další krok rozhodne o tom, jestli bude mít kvadratická rovnice kořeny z oboru reálných čísel. Následně vznikne následující výraz:

$$\alpha(x + r_1)(x + r_2)$$

Pokud budoucí kvadratická rovnice nebude mít kořeny z oboru reálných čísel, jsou tyto kořeny z oboru komplexních čísel a jsou komplexně sdružené. Posledním krokem je roznásobení výrazu, které zajistí knihovna Symja [25]. Roznásobením výrazu vznikne textový řetězec, který odpovídá vnitřnímu formátu Generátoru. Tento řetězec může vypadat například takto:

```
x^2+3x+2
```

V následující ukázce vidíme implementaci generování kvadratické rovnice.

```
@Override
protected void generateExample() {
  int root1 = Random.generate(
       params.getRootMax(), params.getRootMin());
  int root2 = Random.generate(
       params.getRootMax(), params.getRootMin());
  int alpha;
  do {
     alpha = Random.generate(
          params.getAlphaMax(), params.getAlphaMin());
  } while (alpha == 0);
  if (Math.random() > 0.1) {
     example = String.format(
          "%d(x+%d)(x+%d)", alpha, root1, root2);
  } else {
     example = String.format(
          "%d(x+%d-I)(x+%d+I)", alpha, root1, root1);
  }
  example = solver.expand(example);
}
```

4.2.1.2 Implementace metody solveExample

Ačkoli byly kořeny kvadratické rovnice známy již při jejím generování, je nutné dodržet obecný postup a kvadratickou rovnici vyřešit. Nalezení kořenů kvadratické rovnice zajišťuje knihovna Symja [25].

```
@Override
protected void solveExample() {
    result = solver.roots(example);
}
```

V ukázce vidíme implementaci metody solveExample třídy QuadraticEquation.

4.2.1.3 Implementace metody compareResult

Protože Generátor připouští několik možných formátů zápisu nalezených kořenů, je implementace metody compareResult zdlouhavá. Metoda pracuje tak, že z výsledku Generátoru a řešení žáka vyparsuje všechny kořeny a uloží je do dvou seznamů. Oba tyto seznamy seřadí podle velikosti. Pokud jsou po seřazení seznamy shodné, bylo žákovo řešení správné.

4.3 Vygenerování příkladu

Objeky potomků třídy ParametersWrapper slouží kromě předávání parametrů i k určení typu příkladu, který má Generátor vygenerovat. Samotné genenerování příkladů zajišťuje metoda

```
public AbstractExample generate(ParameterWrapper params)
```

třídy Generator. Typ generovaného příkladu je určen typem objektu params. Pro určení typu objektu params je využit návrhový vzor Double dispatch.

```
Generator generator = new Generator();
AbstractExample gcd =
        generator.generate(new ParametersGCD(100, 1000));
```

Ukázka zachycuje vygenerování příkladu na hledání největšího společného dělitele dvou čísel. Tato čísla budou patřit do intervalu (100;1000).

4.4 Vykreslení matematických výrazů

Vykreslování matematických výrazů je zajištěno pomocí open-source JavaScriptové knihovny MathJax [33]. Tato knihona umožňuje vykreslování matematických výrazů zapsaných např. v LaTeXu nebo MathML přímo do HTML stránky. Jako jazyk zápisu matematických výrazů byl zvolen LaTeX. Hlavním důvodem pro zvolení LaTeXu byla jeho stručnější a přehlednější syntaxe. Generování zdrojového kódu LaTeXu ve většině případů zajišťuje knihovna Symja. Samotné generování je implementováno v metodě

```
public String toLatex()
```

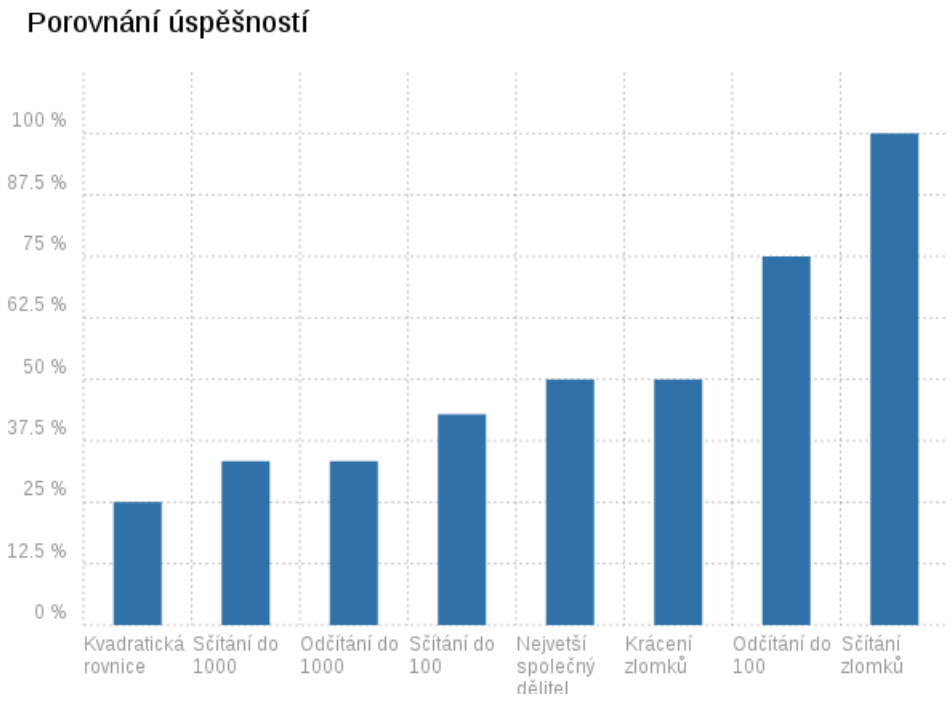
třídy AbstractExample. Tato metoda převede zadání příkladu do zdrojového kódu LaTeXu, pomocí kterého je později vykreslen výraz uživateli. Pokud je tvorba zdrojového kódu LaTeXu odlišná od tohoto postupu, lze metodu toLatex() překrýt a implentovat vlastní způsob generování kódu LaTeXu. Obrázek 4.1 znázorňuje matematický výraz vykreslený pomocí knihovny MathJax.

$$-8 - 9 \cdot x - x^2$$

Obrázek 4.1: Matematický výraz vykreslený pomocí knihovny MathJax

4.5 Vykreslení grafů

Pro vykreslování grafů Doučmat využívá open-source JavaScript knihovnu Chartist-js [34]. Chartist-js podporuje prohlížeče Google Chrome 35, Mozilla Firefox 31, Microsoft Internet Explorer 9 a novější. Podporovány jsou i další prohlížeče, např. Safari. Díky podpoře výše zmíněných prohlížečů knihovna vyhovuje nefunkčnímu požadavku N01. Obrázek 4.2 znázorňuje graf vykreslený pomocí Chartist-js.

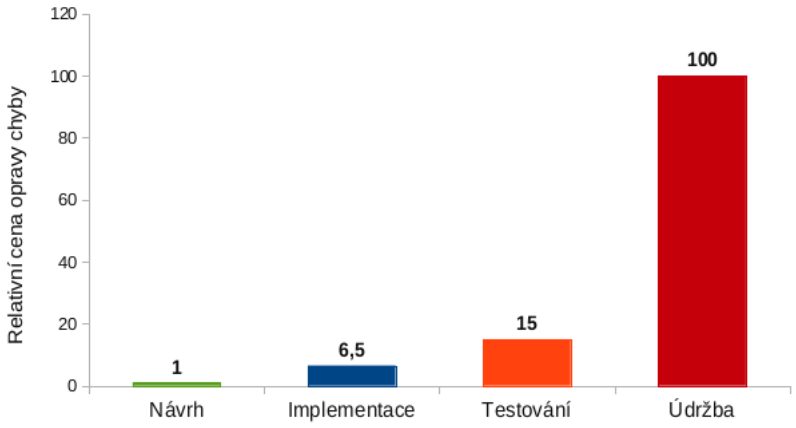


Obrázek 4.2: Ukázka grafu vytvořeného pomocí Chartist-js

Kapitola 5

Testování

Software vyvíjí lidé, a jak je dobře známo, lidé chybují. Chyby při vývoji software nemusí být pouze v implementaci ale i v analýze, návrhu nebo dokonce v samotných testech. Jak vidíme z obrázku 5.1, cena opravení chyby roste s pokračujícími fázemi vývoje projeku. Proto je vhodné chyby nalézt a opravit co nejdříve. Chyby bude s největší pravděpodobností obsahovat každý netriviální software. Cílem testování je tyto chyby najít. Teprve následná oprava chyb vede ke zlepšení kvality software. Disciplínu zabývající se kontrolou kvality software nazývámé Quality Assurance (QA). Hodnoty z obrázku 5.1 byly převzaty z [35].



Obrázek 5.1: Relativní cena opravy chyby

5.1 Použité technologie

5.1.1 JUnit

JUnit [36] je framework pro psaní jednotkových testů v jazyce Java. Patří do rodiny xUnit, která vznikla u jazyka Smalltalk. Nyní se tato rodina rozšířila na více než 30 programovacích jazyků.

5.1.2 Mockito

Mockito [37] je framework pro tvorbu tzv. mock objektů. Mockování je technika izolace testovaných tříd od jejich okolí. Podle oficiálních stránek projektu Mockito zvolila komunita serveru StackOverflow [38] Mockito jako nejlepší nástroj pro tvorbu mock objektů v jazyce Java.

```
import java.util.List;
import org.junit.Test;
import static org.mockito.Mockito.*;
public class ListTest {
    @Test
    public void testAdd(){
        List mockedList = mock(List.class);
        mockedList.add("one");
        mockedList.clear();
        verify(mockedList).add("one");
        verify(mockedList).clear();
        verifyNoMoreInteractions(mockedList);
    }
}
```

V ukázce vidíme ilustrativní ukázku použití frameworků JUnit a Mockito.

5.2 Použité techniky testování

Při vývoji aplikace Doučmat byly použity dvě základní techniky testování. Automatizované jednotkové testování pomocí frameworku JUnit a manuální testování webové vrstvy aplikace pomocí připravených scénářů.

5.2.1 Jednotkové testy

Pro testování aplikace Doučmat bylo vytvořeno zhruba 60 jednotkových testů. U webové části aplikace je většina testů zaměřena na logiku aplikace a vrstvu DAO. Pro odizolování tříd aplikační logiky od jejich okolí je použita technika mockování. Mock objekty jsou vytvářeny pomocí nástroje Mockito [37]. Další testy jsou zaměřeny na složitější validace, jako je validace formátu výsledku přikladu nebo validace emailu.

U generátoru jsou testy z velké části zaměřeny na řešení příkladů a porovnávání výsledků.

5.2.2 Manuální testy

Manuálním testováním byl testován frontend Doučmatu. Aplikace byla proklikávána podle připravených scénářů. Uživatel, který aplikaci manuálně testuje, musí být seznámem s požadavky na aplikaci, aby byl schopný odhalit nedostatky. V ukázce je popsán nejkomplexnější scénář manuálního testování.

1. Registrace uživatele jako rodič.
2. Přihlášení rodiče do aplikace.
3. Odhlášení se rodiče z aplikace.
4. Registrace uživatele jako žák.
5. Přihlášení žáka do aplikace.
6. Kontrola statistik, žádátostí o dozor.
7. Vypracování dobrovolného úkolu.
8. Kontrola statistik a detailu vypracovaného úkolu.
9. Zobrazení nápovědy k vypracování příkladu.
10. Testování tlačítka Zpět.
11. Odhlášení žáka z aplikace.
12. Přihlášení rodiče, odeslání žádosti, odhlášení rodiče.
13. Přihlášení žáka, přijmutí žádosti, odhlášení žáka.
14. Přihlášení rodiče, zadání úkolu, odhlášení rodiče.
15. Přihlášení žáka, vyřešení úkolu, zobrazení statistik odhlášení žáka.

Do budoucna by bylo vhodné proklikávání aplikace Doučmat zautomatizovat. K tomuto účelu by mohl posloužit nástroj Selenium [39].

5.2.3 Testování podpory prohlížečů

Z nefunkčních požadavků vyplývá, že Doučmat má být dostupný přes webové rozhraní pomocí prohlížečů Google Chrome 41 [9], Mozilla Firefox 37 [10], Microsoft Internet Explorer 11 [11] a pomocí novějších verzí těchto prohlížečů. Podpora těchto prohlížečů byla testována manuálně podle scénáře z kapitoly 5.2.2. Testy probíhaly v operačních systémech Linux Debian 7.2, Microsoft Windows 7, Microsoft Windows 8. Výsledky těchto testů jsou zachyceny v příloze F. Nefunkčí požadavky nepožadovaly speciální rozhraní pro mobilní telefony nebo tablety. Přesto lze aplikaci pohodlně ovládat z mobilího telefonu nebo tabletu. Tato skutečnost byla testována podle scénáře z kapitoly 5.2.2 na mobilním telefonu HTC Desire 500 a tabletu Samsung Galaxy TAB 3. V obou případech byl použit prohlížeč Google Chrome.

Kapitola 6

Nasazení

Posledním krokem této práce je nasazení aplikace Doučmat v testovacím provozu na veřejně dostupný server. Aplikace je dostupná z URL:

- `http://doucmat-brabec.rhcloud.com/`

Diagram nasazení naleznete v příloze G.

6.1 Webový hosting

Aplikace byla nasazena na zdarma dostupný webový hosting OpenShift [40] provozovaný firmou RedHat. Po registraci zde může uživatel nasadit až tři různé aplikace. Hosting podporuje širokou škálu technologií, například Javu, PHP, Python nebo Perl. Za databáze můžeme uvést PostgreSQL nebo MySQL včetně nástroje phpMyAdmin. Podporována je i continous integration nástrojem Jenkins.

6.2 Nasazení Doučmatu

Před prvním nasazením Doučmatu na webový hosting byla spuštěna MySQL databáze a vytvořeny potřebné tabulky pro chod aplikace. Dalším krokem nasazaní je upload Web Archive (WAR) na hosting. Tento krok byl automatizován pomocí skriptu, který:

1. Zkompiluje Generátor.
2. Nastaví kontext Doučmatu pro provoz na hostingu.
3. Zkompiluje Doučmat.
4. Nastaví kontext Doučmatu pro lokální vývoj.
5. Uploaduje WAR soubor na hosting.

Celý proces nasazení nové verze aplikace (pokud nebereme v úvahu změnu databáze) je plně automatizovaný. Do budoucna by bylo určitě výhodné pro nasazení začít používat nástroj Jenkins.

Kapitola 7

Uživatelská příručka

Uživatelská příručka slouží uživatelům pro lepší orientaci v aplikaci. Pro přehlednost byla rozdělena na tři části. První část uživatelské příručky je určena pro všechny typy uživatelů. Druhá část je určena pro rodiče a třetí pro žáky.

7.1 Společná část

Společná část uživatelské příručky popisuje úkony společné pro všechny typy uživatelů. Poskytuje návod na registraci a přihlášení. Krátce popisuje úvodní obrazovku.

7.1.1 Úvodní obrazovka

Úvodní stránka nabízí uživateli stručný přehled o funkčnosti aplikace. Na levém okraji obrazovky je menu, ve kterém jsou dvě důležité položky Registrace a Přihlášení.

7.1.2 Registrace

Pokud chce neregistrovaný uživatel začít využívat aplikaci, musí se registrovat. Registrace je možná ve dvou rolích jako rodič nebo dítě. Rodič může zadávat úkoly dětem a sledovat jejich výsledky a statistiky. Dítě řeší úkoly zadané rodičem nebo dobrovolné. Roli po registaci nelze změnit.

K registračnímu formuláři přejdeme z úvodní obrazovky kliknutím na položku menu „Registrace“. V registračním formuláři je nutné vyplnit všechna pole. Po vyplnění formuláře stiskneme tlačítko „Registrovat se“ a pokud vše proběhlo úspěšně, jsme přesměrování na stránku pro přihlášení. Po přihlášení lze začít naplno užívat aplikaci.

Obrázek 7.1: Registrační formulář

7.1.3 Přihlášení

Předtím, než s aplikací začneme naplno pracovat, je nutné se přihlásit.

Na úvodní obrazovce vybereme možnost „Přihlásit se“. Následně se zobrazí přihlašovací formulář, kde vyplníme uživatelské jméno a heslo. Pokud jsme zadali správné údaje, tak jsme po stisku tlačítka „Přihlásit se“ přihlášeni a můžeme začít plně využívat aplikaci.

Obrázek 7.2: Formulář přihlášení

7.1.4 Odhlášení

Pro ukončení práce s aplikací stiskneme tlačítko „Odhlásit se“ v pravém horním rohu obrazovky.

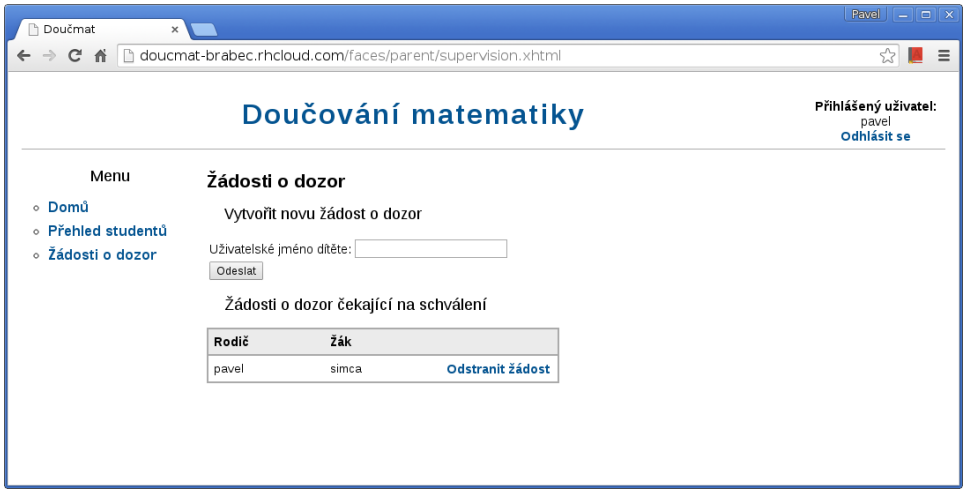
7.2 Pro rodiče

Následující část uživatelské příručky je určena pro uživatele registrované v roli rodiče. Pro následující činnosti je nutné přihlásit se do aplikace.

7.2.1 Zaslání žádosti o dozor

Aby rodič mohl zadávat úkoly dítěti, musí mu zaslat žádost o dozor, kterou musí dítě schválit. Toto je nutné pro ochranu soukromí dítěte, aby nebylo možné sledovat jeho výsledy cizí osobou. Každé dítě má maximálně jednoho rodiče. Rodič může kontrolovat libovolný počet dětí.

Pro zaslání žádosti o dozor vybereme v menu položku „Žádosti o dozor“. Na následující stránce vyplníme uživatelské jméno žáka a stiskneme tlačítko „Odeslat“. Pokud vše proběhlo v pořádku, je nutné počkat na žákovo potvrzení. O výsledku žádosti bude rodič informován e-mailem.

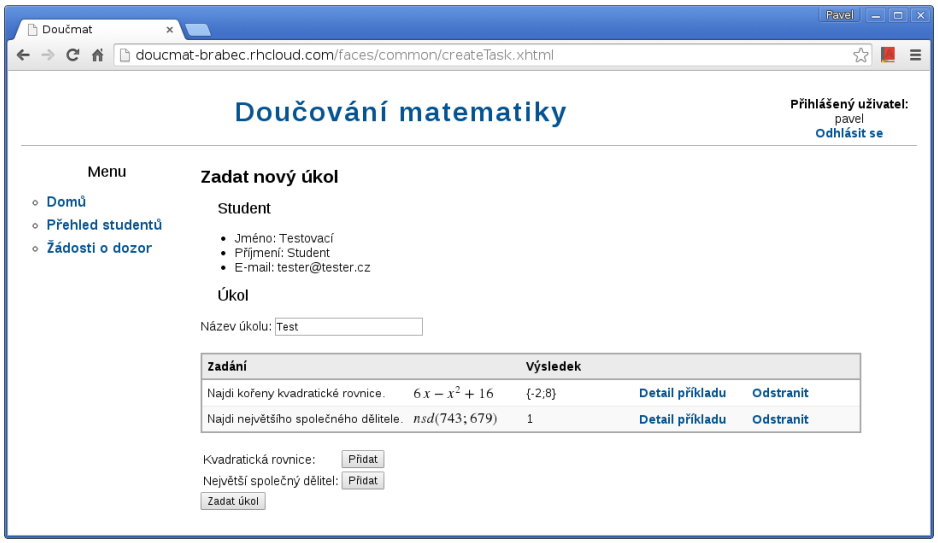


Obrázek 7.3: Odeslání žádosti o dozor

7.2.2 Zadání úkolu

Pro zadání úkolu vybereme v menu položku „Přehled studentů“. Následně se zobrazí přehled studentů pod naším dozorem. U vybraného studenta vybereme možnost „Zadat úkol“. Na následující stránce vyplníme jméno úkolu a pomocí tlačítka „Přidat“ vygenerujeme příklady. Vygenerovaný příklad lze z úkolu odstranit stiskem „Odstranit“. Po vytvoření úkolu stikneme tlačítko „Zadat“, kterým zadáme úkol žákovi. Po vypracování je rodič informován e-mailem.

Obrázek 7.4: Detail studenta



Obrázek 7.5: Zadání nového úkolu

7.2.3 Přehled statistik

Pro zobrazení statistik o výsledcích žáka v menu zvolíme položku „Přehled studentů“. Následně se zobrazí přehled studentů pod naším dozorem. U vybraného studenta vybereme možnost „Detail“. Na následující stránce vidíme přehled vypracovaných a nevypracovaných úkolů daného studenta. Pro zobrazení statistik vybereme možnost „Zobrazit statistiky“. Nyní aplikace zobrazí statistiky vybraného studenta. Pro přehlednost jsou statistiky seřazeny vzestupně podle úspěšnosti v jednotlivých typových příkladech.

7.3 Pro žáky

Následující část uživatelské příručky je určena pro uživatele registrované v roli žák. Pro následující činnosti je nutné přihlásit se do aplikace.

7.3.1 Žádosti o dozor

Pokud si žák chce nechat zadávat úkoly od svého rodiče, musí přijmout jeho žádost. Toto je bezpečnostní opatření, které chrání soukromí žáka. Žádosti o kontrolu zamezují tomu, aby cizí osoba měla přehled o výsledcích a statistikách žáka.

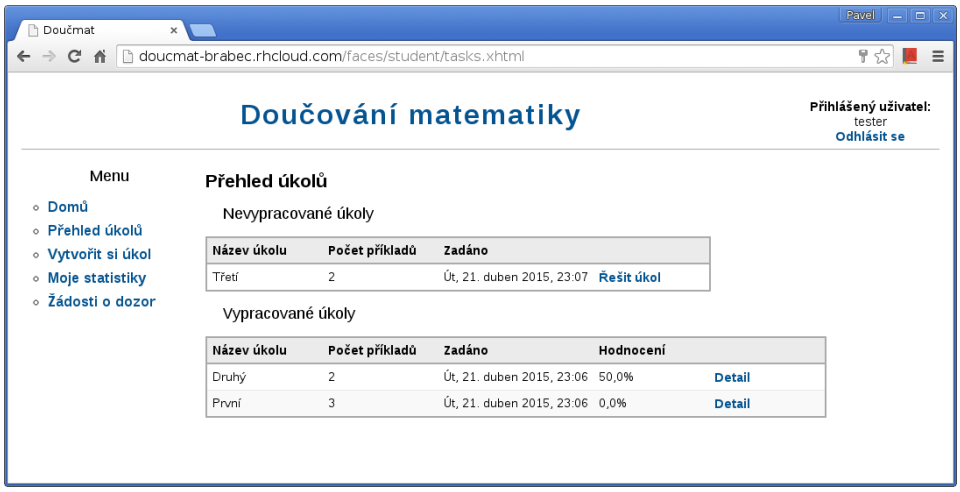
Pro přijmutí nebo zamítnutí žádosti o dozor v menu zvolíme položku „Žádosti o dozor“. Na následující stránce vidíme přehled přijmutých žádostí nebo informace o rodiči. Kliknutím na možnosti „Přijmout žádost“/„Zamítnout žádost“ můžeme žádost přijmout/zamítnout.

Obrázek 7.6: Přijmutí žádosti o dozor

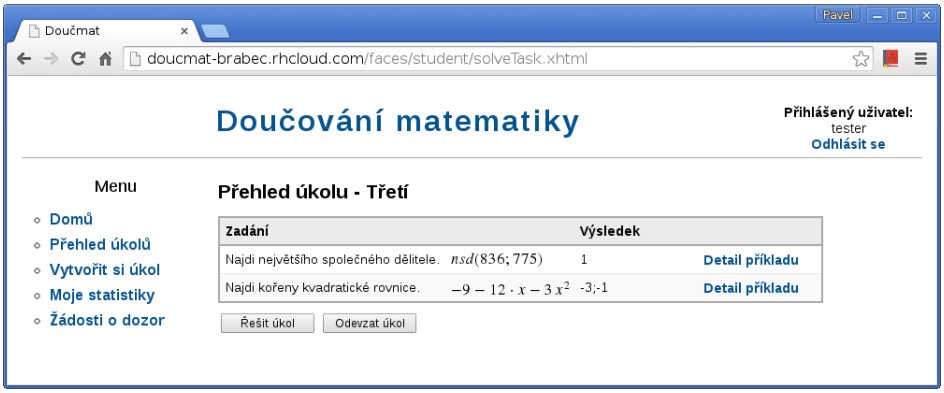
7.3.2 Vypracování úkolu

Pro vypracování úkolu zvolíme v menu položku „Přehled úkolů“. Následně se zobrazí obrazovka s přehledem vypracovaných a nevypracovaných úkolů. U vybraného úkolu vybereme možnost „Řešit úkol“. Nyní se zobrazil detail úkolu. Detail úkolu slouží k tomu, aby žák měl přehled, jaké příklady bude v úkolu řešit, a k ověření toho, jaké výsledky vyplnil. Stiknutím tlačítka „Řešit úkol“ začneme úkol řešit. Žák vypočítá příklad, který mu aplikace zobrazuje. Výsledek příkladu napíše do pole „Řešení“. Výsledek musí být ve správném formátu. Nápověda k popisu formátu, jakým má být řešení vyplněno, se zobrazí po výběru možnosti „Zobrazit nápovědu k formátu výsledku“. Po vyplnění pole „Řešení“, žák stikne tlačítko „Uložit řešení“. Výběrem možnosti „Další“ se žák přesune na řešení dalšího úkolu. Až bude mít žák všechny úkoly vypra-

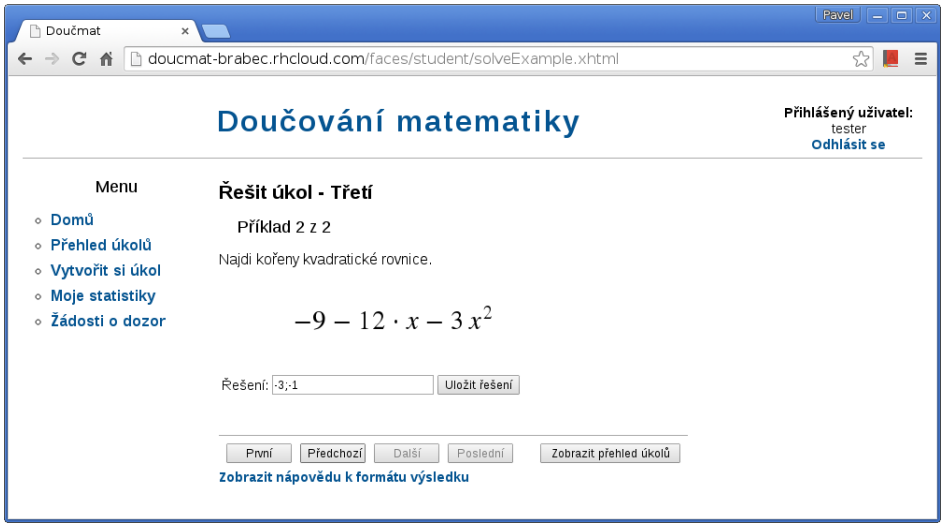
covány, stikne tlačítko „Zobrazit detail úkolu“. Aplikace zobrazí detail úkolu, kde žák překontroluje své výsledky. Pro změnu nebo doplnění výsledku vybere možnost „Detail příkladu“ u vybraného příkladu. Pro odevzání úkolu stikne tlačítko „Odevzdat úkol“. Po stisknutí tohoto tlačítka již není možné měnit odpovědi. Nyní aplikace zobrazí detail úkolu, kde žák vídí správné výsledky a vyhodnocení svých odpovědí.



Obrázek 7.7: Přehled úkolů



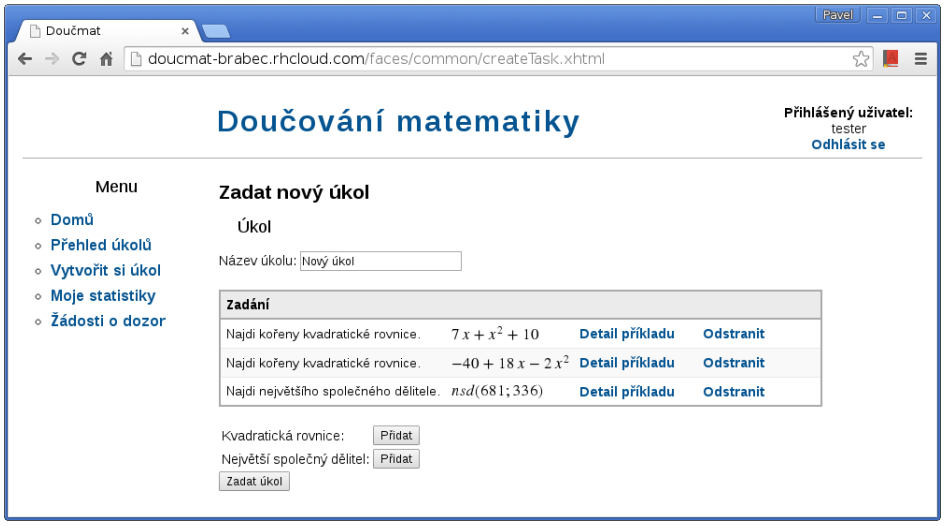
Obrázek 7.8: Detail úkolu



Obrázek 7.9: Řešení příkladu

7.3.3 Vytvoření úkolu

Pro vytvoření dobrovolného úkolu vybereme v menu položku „Vytvořit si úkol“. Na následující obrazovce vyplníme název úkolu a přidáme požadované příklady. Přidané příklady lze odstranit výběrem možnosti „Odstranit“ u požadovaného úkolu. Po stiknutí tlačítka „Zadat úkol“ aplikace zobrazí detail úkolu. Nyní je úkol připraven k vyřešení. Řešení úkolu je popsáno v sekci Vypracování úkolu.



Obrázek 7.10: Vytvoření nového úkolu

7.3.4 Přehled statistik

Pro zobrazení statistik vybereme v menu položku „Moje statistiky“. Nyní aplikace zobrazí statistiky. Pro přehlednost jsou statistiky seřazeny vzestupně podle úspěšnosti v jednotlivých typových příkladech.

Závěr

Hlavním cílem této práce bylo vytvořit webovou aplikaci, která by pomohla rodičům žáků základních škol se zadáváním a opravováním úkolů z matematiky a žákům poskytla prakticky neomezenou sbírku příkladů. Součástí práce byla také analýza podobných, již existujících řešení, analýza požadavků na aplikaci, návrh řešení, testování a nasazení aplikace. Posledním krokem bylo vytvoření uživatelské příručky.

Cíl práce byl naplněn. Aplikace byla vytvořena a splňuje všechny funkční i nefunkční požadavky z analýzy. Implementovány jsou rovněž všechny případy užití. Aplikace je nasazená v testovacím provozu na veřejně dostupném serveru a testování aplikace neodhalilo žádné závažné nedostatky.

Vývoj celé aplikace bude pokračovat tak, aby se aplikace stala co nejužitečnější. Do budoucna je naplánováno výrazně rozšířit základnu typových příkladů a alespoň u některých typových příkladů přidat možnost zobrazení správného postupu řešení. Aplikace by také měla být lokalizována do více jazyků, aby se mohla rozšířit základna uživatelů. Jistě bude nutné doplnit některá nastavení uživatelských účtů jako např. nastavení notifikací, dále uživateli umožnit změnu a obnovení hesla, pokud by došlo k jeho zapomenutí.

Literatura

[1] Wolfram Research Company: Wolfram Alpha [online]. 2015, [cit. 2015-02-13]. Dostupné z: http://www.wolframalpha.com/

[2] Khan Academy: Khan Academy [online]. 2015, [cit. 2015-02-13]. Dostupné z: www.khanacademy.org/

[3] Hyco, Inc: MathGoodies [online]. 1998, [cit. 2015-02-13]. Dostupné z: http://www.mathgoodies.com/

[4] Husara, P.: e-matematika [online]. 2015, [cit. 2015-02-13]. Dostupné z: http://www.e-matematika.cz/

[5] Hofmann, H.; Lehner, F.: *Requirements Engineering as a Success Factor in Software Projects [online]*. 2001, [cit. 2015-04-05]. Dostupné z: http://www.ics.uci.edu/~wscacchi/Software-Process/Readings/Req-Engr-SuccessFactors-Software-July01.pdf

[6] Krigsman, M.: *CIO analysis: Why 37 percent of projects fail [online]*. 2011, [cit. 2015-04-05]. Dostupné z: http://www.zdnet.com/article/cio-analysis-why-37-percent-of-projects-fail/

[7] Arlow, J.; Neustadt, I.: *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, vyd. 2. vydání, 2007.

[8] Krátký, T.; Zoubek, B.: *Requirements Engineering [online]*. 2014, [cit. 2015-04-05]. Dostupné z: http://www.profinit.eu/fileadmin/Content/profinit.eu/Academy/NSWI129/prednasky/02_Requirements.pdf

[9] Google, Inc.: *Chrome [software]*. 2015, [cit. 2015-04-13]. Dostupné z: https://www.google.cz/chrome/browser/desktop/

[10] Mozilla Foundation: *Firefox [software]*. 2015, [cit. 2015-04-13]. Dostupné z: https://www.mozilla.org/cs/firefox/new/

[11] Microsoft Corporation: *Internet Expolorer [software]*. 2015, [cit. 2015-04-13]. Dostupné z: http://windows.microsoft.com/cs-CZ/internet-explorer/download-ie

[12] Rumbaugh, J.; Jacobson, I.; Booch, G.: *The Unified Modeling Language Reference Manual [online]*. 1999, [cit. 2015-04-05]. Dostupné z: http://msdl.cs.mcgill.ca/people/tfeng/docs/The%20Unified%20Modeling%20Language%20Reference%20Manual.pdf

[13] Oracle Corp.: *Java SE [software]*. 2015, [cit. 2015-04-09]. Dostupné z: http://www.oracle.com/technetwork/java/javase/downloads/index.html

[14] Balík, M.: *Programování v jazyku Java: historie, úvod [online]*. [cit. 2015-04-09]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-PJV/prednasky/pjv01uvod.pdf

[15] Oracle Corp.: *Java Overview [online]*. 2015, [cit. 2015-04-09]. Dostupné z: http://www.oracle.com/technetwork/java/javaee/overview/index.html

[16] The Apache Software Foundation: *Apache Tomcat [software]*. 2015, [cit. 2015-04-09]. Dostupné z: http://tomcat.apache.org/download-80.cgi

[17] The Apache Software Foundation: *Apache License [online]*. Druhé vydání. Dostupné z: http://www.apache.org/licenses/LICENSE-2.0

[18] Mojarra: *JavaServer Faces [software]*. 2015, [cit. 2015-04-09]. Dostupné z: https://javaserverfaces.java.net/nonav/2.2/download.html

[19] Troníček, Z.: *Enterprise Java BI-EJA, třetí přednáška [online]*. [cit. 2015-04-09]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-EJA/prednasky/EJA3.pdf

[20] Pivotal Software: *Spring [software]*. 2015, [cit. 2015-04-09]. Dostupné z: http://spring.io/

[21] Troníček, Z.: *Spring Framework [online]*. [cit. 2015-04-09]. Dostupné z: http://www.webinare.cz/videozaznamy/videozaznam/94-spring-framework.aspx

[22] Red Hat, Inc.: *Hibernate [software]*. 2015, [cit. 2015-04-09]. Dostupné z: http://hibernate.org/orm/

[23] Oracle Corp.: *MySQL [software]*. 2015, [cit. 2015-04-09]. Dostupné z: http://www.mysql.com/downloads/

[24] Oracle Corp.: *About MySQL [online]*. 2015, [cit. 2015-04-09]. Dostupné z: http://www.mysql.com/about/

[25] Clk, A.: *Symja Android library [software]*. 2015, [cit. 2015-04-16]. Dostupné z: https://bitbucket.org/axelclk/symja_android_library/wiki/Home

[26] Free Software Foundation, Inc.: *GNU Lesser General Public License [online]*. 2015, [cit. 2015-04-10]. Dostupné z: https://www.gnu.org/licenses/lgpl.html

[27] Krátký, T.; Zoubek, B.: *Architecture and Design [online]*. 2014, [cit. 2015-04-05]. Dostupné z: http://www.profinit.eu/fileadmin/Content/profinit.eu/Academy/NSWI129/prednasky/03_ArchitectureDesign.pdf

[28] Sparx Systems Pty Ltd.: *Enterprise Architect [software]*. 2015, [cit. 2015-04-12]. Dostupné z: http://www.sparxsystems.com.au/

[29] Valenta, M.; Pokorný, J.: *Databázové systémy*. Praha: České vysoké učení technické v Praze, první vydání, 2013.

[30] Lórencz, R.: *Bezpečnost: Hašovací funkce [online]*. 2014, [cit. 2015-04-05]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEZ/_media/bez_n5.pdf

[31] Oracle Corp.: *NetBeans IDE [software]*. 2015, [cit. 2015-04-16]. Dostupné z: http://www.netbeans.org/

[32] The Apache Software Foundation: *Commons Validator [software]*. 2015, [cit. 2015-04-10]. Dostupné z: http://commons.apache.org/proper/commons-validator/index.html

[33] The MathJax team: *Math Jax [software]*. 2015, [cit. 2015-04-16]. Dostupné z: http://www.mathjax.org/

[34] Kunz, G.: *chartist-js [software]*. 2015, [cit. 2015-04-16]. Dostupné z: http://gionkunz.github.io/chartist-js/

[35] Dawson, M.: *INTEGRATING SOFTWARE ASSURANCE INTO THE SOFTWAREDEVELOPMENT LIFE CYCLE (SDLC) [online]*. 2010, [cit. 2015-04-13]. Dostupné z: http://tinyurl.com/academia-edu-SoftwareAssurance

[36] JUnit: *JUnit [software]*. 2015, [cit. 2015-04-10]. Dostupné z: http://http://junit.org/

[37] Faber, S.: *Mockito [software]*. 2015, [cit. 2015-04-10]. Dostupné z: http://mockito.org/

[38] Stack Exchange Inc.: StackOverflow [online]. 2015, [cit. 2015-04-11]. Dostupné z: http://stackoverflow.com/

[39] OpenQA: *Selenium [software]*. 2015, [cit. 2015-04-13]. Dostupné z: `http://www.seleniumhq.org/`

[40] Red Hat, Inc.: *OpenShift*. 2014, [cit. 2015-04-22]. Dostupné z: `https://www.openshift.com/`

Příloha **A**

Seznam použitých zkratek

XML Extensible Markup Language

UML Unified Modeling Language

JRE Java Virtual Machine

Java SE Java Standard Edition

Java ME Java Micro Edition

Java EE Jave Enterprise Edition

JSF JavaServer Faces

ORM Objektově Relační Mapování

CAS Computer Algebra System

LGPL Lesser General Public License

MVC Model View Vontroller

CASE Computer-Aided Software Engineering

DDL Data Definition Language

HTML HyperText Markup Language

QA Quality Assurance

DAO Data Access Object

WAR Web ARchive

URL Uniform Resource Locator

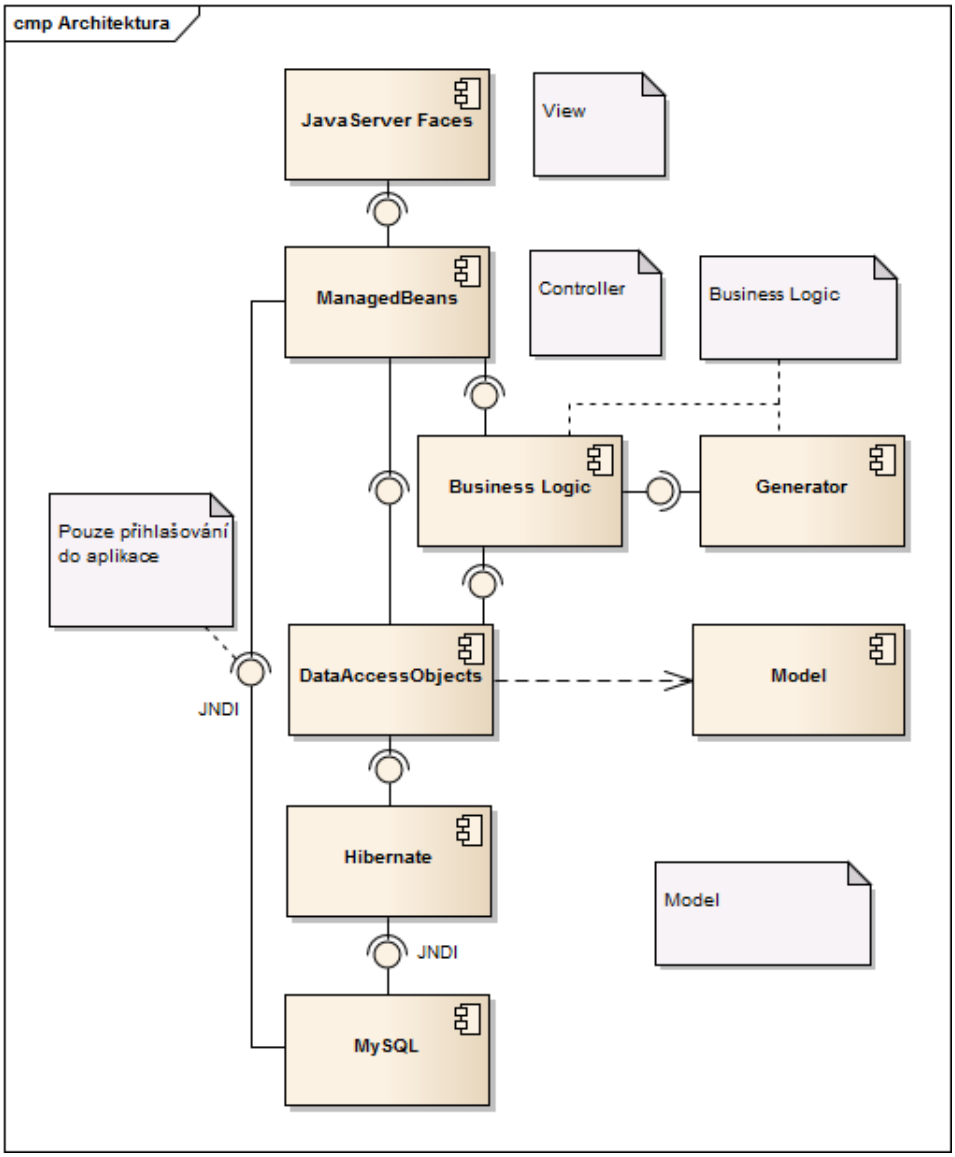
Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `impl`
  - `Doucmat` ..... zdrojové kódy Doučmatu
  - `Generator` ..... zdrojové kódy Generátoru
  - `sql` ..... SQL skripty
- `thesis`
  - `thesis.pdf` ..... text písemné práce ve formátu PDF
  - `src` ..... zdrojové kódy písemné práce ve formátu LaTeX
- `doc`
  - `doucmat.eap` ..... dokumentace v Enterprise Architect
  - `javadoc` ..... dokumentace JavaDoc
- `installation`
  - `doucmat.war` ..... Doučmat WebArchive
  - `installation-manual.txt` ..... stručný popis instalace
  - `doucmat-CREATE.sql` ..... SQL skript pro vytvoření tabulek databáze

Příloha C

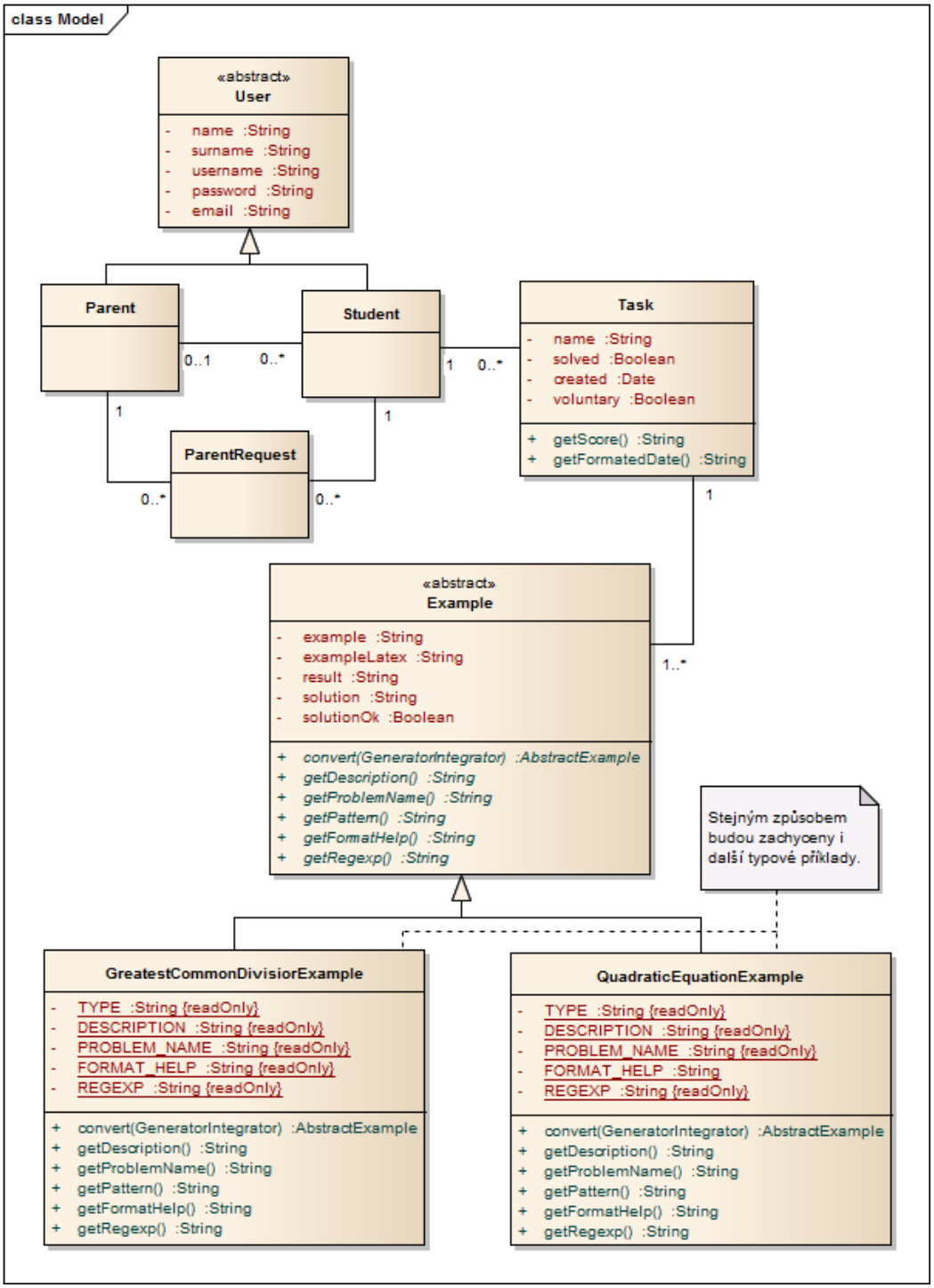
Architektura aplikace



Obrázek C.1: Architektura aplikace Doučmat

Příloha D

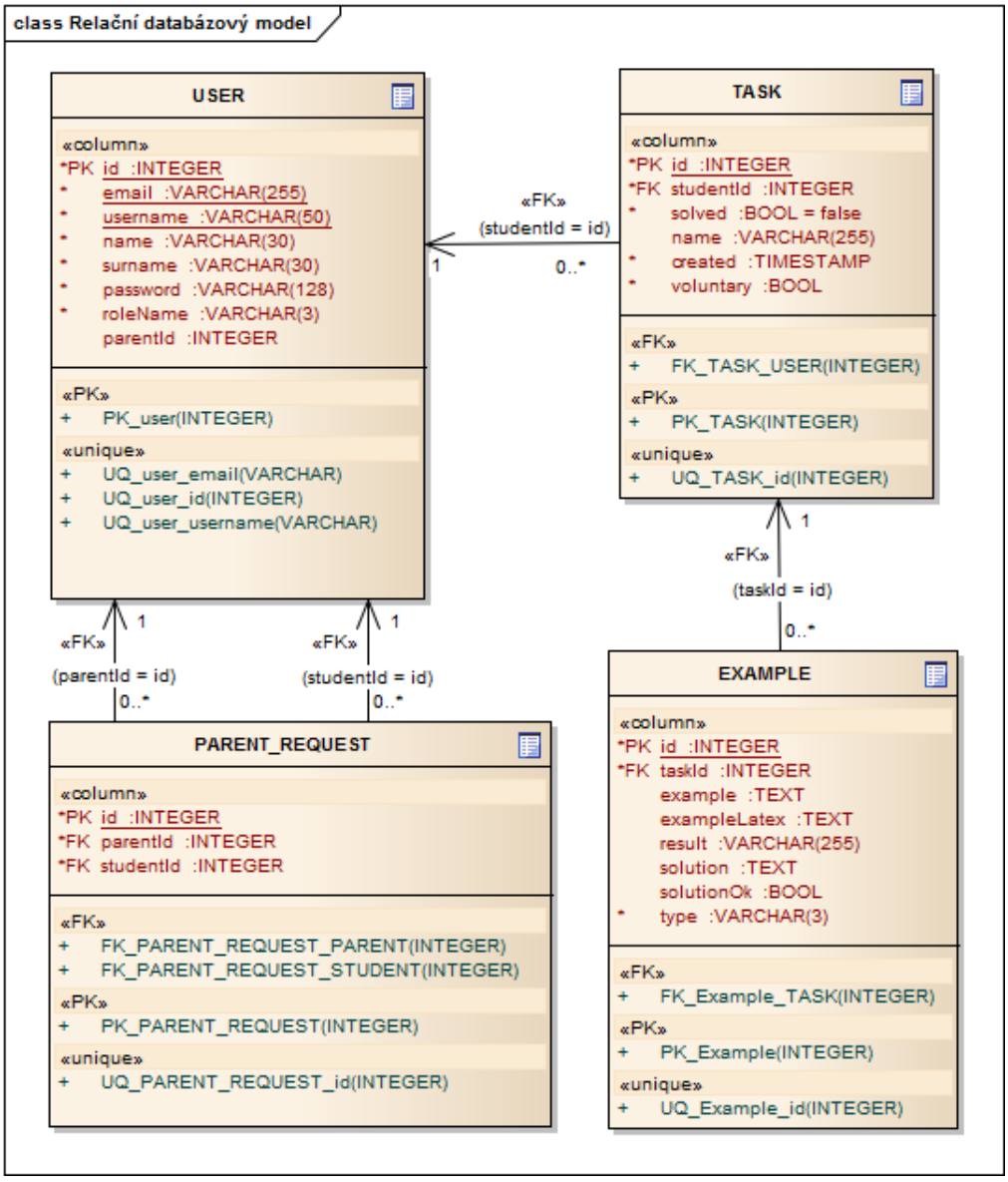
Návrhový model tříd



Obrázek D.1: Návrhový model tříd aplikace Doučmat

Příloha E

Databázový model



Obrázek E.1: Databázový model aplikace Doučmat

Příloha F

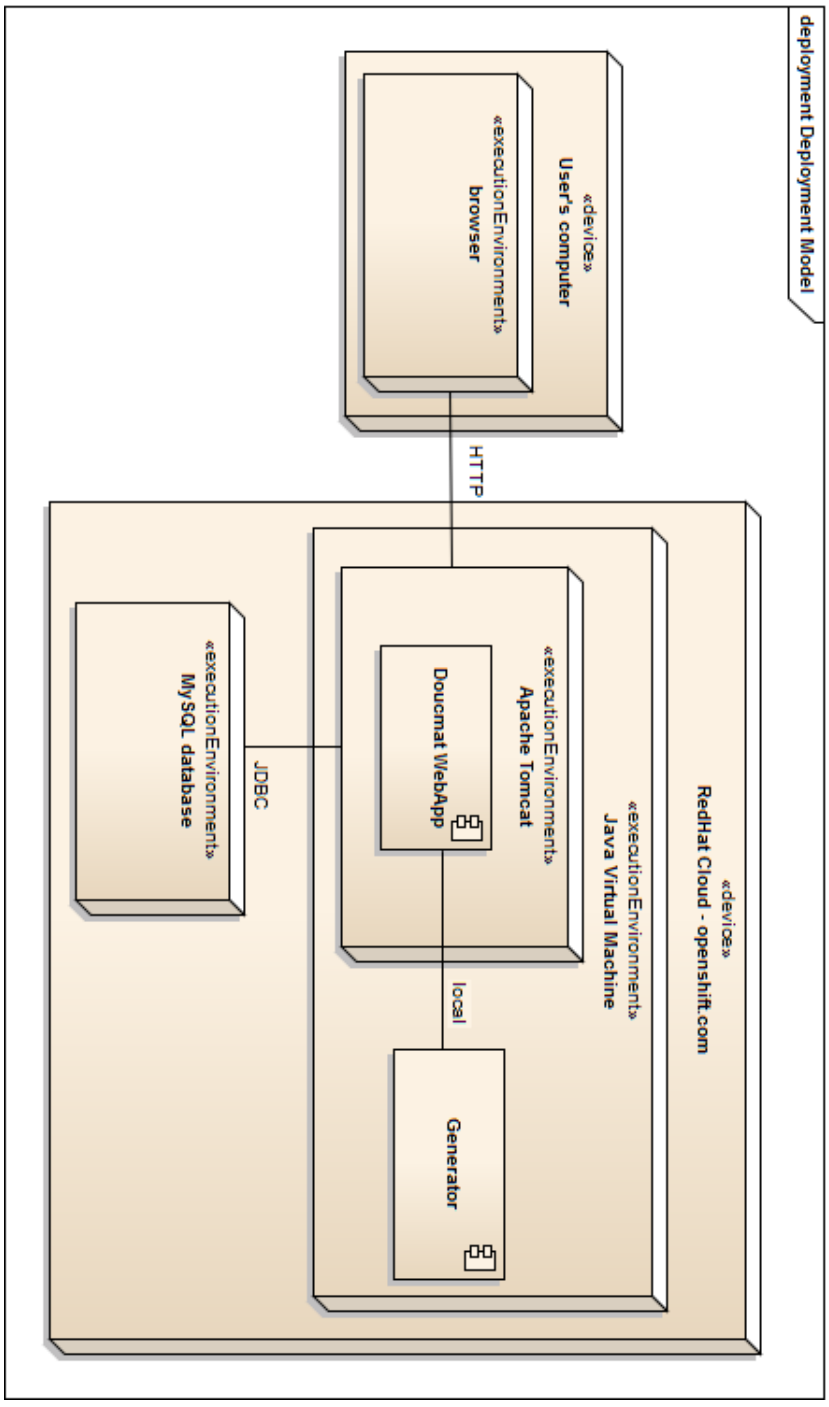
Tabulka testování podpory prohlížečů

Tabulka F.1: Testování podpory prohlížečů

	Debian		Windows 7			Windows 8		
	Chrome 41	Firefox 37	Chrome 41	Firefox 37	IE 11	Chrome 41	Firefox 37	IE 11
Přihlášení užívatele	OK	OK	OK	OK	OK	OK	OK	OK
Odhlášení užívatele	OK	OK	OK	OK	OK	OK	OK	OK
Registrace užívatele	OK	OK	OK	OK	OK	OK	OK	OK
Zaslání žádosti	OK	OK	OK	OK	OK	OK	OK	OK
Zadání úkolu	OK	OK	OK	OK	OK	OK	OK	OK
Vypracování úkolu	OK	OK	OK	OK	OK	OK	OK	OK
Zobrazení grafů	OK	OK	OK	OK	OK	OK	OK	OK
Zobrazení úkolů	OK	OK	OK	OK	OK	OK	OK	OK
Tlačítko Zpět	OK	OK	OK	OK	OK	OK	OK	OK

Příloha G

Diagram nasazení



Obrázek G.1: Diagram nasazení aplikace Doučmat