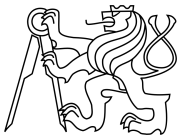
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Revize a dokončení aplikace pro správu reagencií a kitů
Student:	Martin Hampl
Vedoucí:	Ing. Josef Vogel, CSc.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Pomocí metod softwarového inženýrství revidujte a poté rozšiřte webovou aplikaci pro správu reagencií a kitů pro Ústav hematologie a krevní transfuze.

Podrobné pokyny pro zpracování:
1. Navažte na rozpracovaný projekt "Aplikace pro správu reagencií a kitů", popište aktuální stav.
2. Revidujte analýzu, která vznikla v rámci předmětů SP1 a SP2. V případě zjištěných problémů navrhněte změny.
3. Je-li to nutné pro návrh a implementaci rozšíření, implementujte změny navržené v předchozím bodu.
4. Navrhněte následující nové moduly a části systému: správa objednávek kitů, export dat do CSV souborů, archivace, import dat ze současných xls souborů, správa uživatelů a varování.
5. Na základě návrhu proveďte implementaci nové funkcionality, otestujte ji a nasaďte v infrastruktuře zadavatele.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 17. listopadu 2015

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

Bakalářská práce

Revize a dokončení aplikace pro správu reagencií a kitů

Vedoucí práce: Ing. Josef Vogel

8. května 2017

Poděkování

Rád bych tímto poděkoval svému vedoucímu práce panu Ing. Josefu Vogelovi a paní Ing. Vrané z ÚHKT za umožnění pracovat na tomto projektu a získat mnoho zkušeností, které jistě brzy využiji v pracovním prostředí.

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, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2017 .....

Odkaz na tuto práci

Hampl, Martin. *Revize a dokončení aplikace pro správu reagencií a kitů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá revizí analýzy, dokumentace a implementováním rozšiřujících modulů do aplikace „Správa reagencií a kitů“, jejíž vývoj započal v předmětech Softwarový projekt 1 a 2 (SP1 a SP2). Cílem aplikace bude umožnit pracovníkům z Ústavu hematologie a krevní transfuze (ÚHKT) kontrolu a evidenci těchto látek a další činností s tím spojené.

Klíčová slova databáze, javaEE, webová aplikace, mySQL, ÚHKT, správa reagencií a kitů

Abstract

This bachelor thesis is based on revision of analysis, documentation and implementation of new modules into the „Reagent and kit management“ web application, the development of which started in Software project 1 (SP1) and Software project 2 (SP2) subjects. The goal of the web application will be to enable workers of Institute of Haematology and Blood Transfusion (in Czech ÚHKT) control and registration of these substances and other related activities.

Keywords database, javaEE, web application, mySQL, ÚHKT, application for management reagents and kits

Obsah

Seznam obrázků

Úvod

Jedním z nejvýznamnějších zdravotnických zařízení v Praze je Ústav hematologie a krevní transfuze (ÚHKT).

„*Již téměř šedesát let toto pracoviště propojuje specializovanou léčebnou péči a diagnostiku s intenzivní výzkumnou činností v oboru hematologie a krevní transfuze. Významnou součástí práce Výzkumného úseku je vývoj a zavádění nových diagnostických metod pro klinickou praxi.*“[1]

Samostatným oddělením v rámci Výzkumného úseku je oddělení HLA, jehož hlavní náplní je vyhledávání vhodných dárců pro transplantace hematopoetických kmenových buněk. Za tímto účelem jsou využívány kontrolní diagnostické soupravy (KITy) a roztoky pro laboratorní analýzu tzv. reagencia. V současné době tomuto oddělení schází vhodný program pro správu kitů a reagencií, se kterými pracuje. Proto v roce 2015 začal ústav spolupracovat z Českým vysokým učením technickým. Cílem této spolupráce mělo být vytvoření aplikace pro správu výše uvedených látek. Měl jsem možnost na aplikaci pracovat jako člen týmu v rámci studia v předmětu Softwarový projekt 1 (SP1) a následně i v jeho pokračování Softwarový projekt 2 (SP2). Bohužel se ukázalo, že tato aplikace je na tyto předměty příliš rozsáhlá. Rád bych poskytl tamním odborníkům, jejichž práce si velmi vážím, práci ulehčující software. Rozhodl jsem se proto dokončit tuto aplikaci ve své bakalářské práci.

Struktura práce

V první kapitole se věnuji popisu problematiky pro danou doménu. Zaměřuji se především na popis a náplň práce ÚHKT a na problematiku získávání vhodných dárců. V krátkosti vysvětlím mechanismus tzv. histokompatibility a princip určování HLA molekul pomocí diagnostických souprav a reagencií. Nakonec se zaměřím na to, jakým způsobem funguje správa kitů a reagencií nyní. Ve druhé kapitole jasně vymezuji stav projektu před započetím vývoje

modulu v rámci bakalářské práce. Ve třetí kapitole se zabývám revizí a doplněním předešlé softwarové analýzy. Z tohoto tématu jsem vyzdvihl pouze zajímavé nebo revidované části. Následuje kapitola o implementaci, která se věnuje použitým technologiím. Je v ní uveden jednak základní popis technologie, za jakým účelem se používá a ukázka kódu. Další kapitola se zabývá testováním softwaru, jak v rovině teoretické (členění testu), tak i konkrétním testováním samotné aplikace. Poslední kapitola je stručným návodem pro nasazení aplikace na webový server.

Kapitola 1

Popis problematiky

V kapitole jsou uvedeny významy jednotlivých pojmů týkajících se této problémové domény. Jsou zde vysvětleny postupy, které uživatelé využívají při práci s daty a potíže s tím spojené.

1.1 Ústav hematologie a krevní transfuze

„ÚHKT je největší hematologické centrum v Česku a už více než 60 let tu je pro všechny, kteří potřebují opravdu specializovanou léčbu. Tvoří ho totiž nejen špičkoví lékaři, ale také tým osmdesáti vědců, kteří pátrají po tom, proč vůbec poruchy krvetvorby v těle vznikají a jak je co nejúčinněji napravit. Se svými výsledky patří k nejlepším v oboru.“[1]

Ústav se nachází v Praze na Karlově náměstí a byl založen v roce 1952 tehdejším ministerstvem zdravotnictví. Zpočátku se zabýval téměř výhradně problematikou transfuze krve. Postupně rozšiřoval svoji působnost na rozsáhlou diagnostiku a léčbu hematologických onemocnění. Zároveň se zde rozvinula i intenzivní výzkumná činnost.

„Vznikla laboratoř na typizaci HLA antigenů a tematika řešená na tomto oddělení plynule přešla k současnému vyhledávání vhodných dárců kostní dřeně. Bakteriologická laboratoř kromě kontroly sterility krevních produktů a bakteriologické kontroly místností a zařízení prováděla také kultivační vyšetřování a citlivost na antibiotika pro pacienty klinického oddělení. Výzkumně se zabývala studiem protilátek proti tkáňovým antigenům.“ [1]

1.2 Histokompatibilita

Aby byla transplantace úspěšná, musí být co největší tzv. histokompatibilita, neboli tkáňová slučitelnost. Je to určitá podobnost specifických znaků,

které se nacházejí na povrchu téměř všech buněk. Tyto znaky jsou většinou bílkovinné povahy a imunitní systém je dokáže rozlišit. Všeobecně je označujeme jako antigeny. Vlastní antigeny imunitní systém tzv. toleruje a proti cizím vyvolává imunitní reakci s cílem jejich zničení. Tato reakce je velmi důležitá pro ochranu organismů proti různým původcům nemocí, například proti virům a bakteriím. V oblasti transplantační medicíny je však nežádoucím jevem. Proto je snaha o co nejpřesnější zmapování těchto struktur a transplantovat tkáně, které, laicky řečeno, mají co nejpodobnější znaky.

Antigeny, které jsou nejvíce zodpovědné za neúspěšné transplantace, se označují jako tzv. Hlavní histokompatibilní komplex (MHC z angl. major histocompatibility complex). Nejlépe byly MHC struktury prozkoumány u myší, kde je nazýváme H-2 komplex. V roce 1939 byl zahájen výzkum MHC u člověka a koncem padesátých let minulého století byly tyto struktury objeveny a zmapovány. Protože byly detekovány na buňkách bílých krvinek (tzv. leukocytů), byly označeny jako Human Leucocyte Antigens (HLA). HLA antigeny jsou rozděleny do tříd a je jich opravdu mnoho. Pro tkáňovou kompatibilitu jsou důležité zejména antigeny I. a II. třídy. Antigeny I. třídy jsou přítomné téměř na všech buňkách s výjimkou červených krvinek, kde prokázány nebyly. Antigeny II. třídy se nacházejí spíše na buňkách podílejících se přímo na imunitních reakcích.

Tyto třídy ještě dělíme na podskupiny, které označujeme písmeny abecedy. Nejvýznamnější z hlediska transplantací jsou HLA-A, HLA-B, HLA-C. I u těchto podskupin známe velké množství variant, které potom označujeme číslicí připojenou za písmeno. Např u HLA-A je to 59 variant, HLA-B 111 variant a u HLA-C 37 variant. Podtřídy HLA antigenů II. třídy zase označujeme dvěma písmeny. Nejdůležitější je HLA-DR se 122 variantami, HLA-DP s 62 variantami a HLA-DQ s 25 variantami.

„*Každý člověk dědí HLA antigeny po svých rodičích. To znamená, že má vždy 2 varianty od každé podskupiny antigenů (například HLA-A 1 a HLA-A 3). Vzhledem k velkému množství variant u jednotlivých HLA antigenů je zřejmé, že těžko nalezneme 2 lidi, jejichž buňky se shodují ve všech HLA antigenech. Výjimku tvoří jednovaječná dvojčata. Pro úspěšnou transplantaci musíme ale nalézt dárce, který se shoduje v co největším množství. To, jaké HLA determinanty se na povrchu buněk vytvoří, je zakódováno v genech, které se nacházejí na 6. chromozomu. Právě tyto geny je snaha rozpoznat, mluvíme pak o tzv. genotypizaci.*“[2]

„*Určení přesného genotypu HLA u pacienta a jeho dárce je jednou ze základních podmínek pro úspěšnost allogenní (v překladu allogenní -jiný organismus vlastního druhu) transplantaci hematopoetických kmenových buněk. Neshoda*

v HLA oblasti mezi dárcem a příjemcem transplantátu může navodit odhojení tohoto štěpu, popřípadě způsobit nemoc štěpu proti hostiteli GvHD (Graft Versus Host Disease), která je vážnou, popřípadě smrtelnou komplikací.“ [3]

1.3 Kity a Reagencie

Genotypizace HLA se provádí pomocí specifických diagnostických souprav (KITů), kde dochází ke specifickým reakcím, které se různými metodami detekují. HLA oddělení v současné době tyto kity nakupuje od tří komerčních výrobců. Tyto soupravy se skládají z jednotlivých látek (částí) zvaných reagencie.

1.4 Stav před nasazením aplikace

Evidence kitů před nasazením programu v ÚHKT probíhala pomocí nepřehledných Office Open XML (XLS) dokumentů, které jsou uloženy na sdíleném disku. Vznikají zde problémy s vícenásobným přístupem (zaměstnanci se vždy musí domlouvat, kdo bude zapisovat, aby si práci navzájem nevymazali) a neposkytují žádnou podporu pro objednávky.

Kapitola 2

Počáteční stav aplikace

Tato bakalářská práce navazuje na softwarový týmový projekt, na kterém jsem pracoval s dalšími třemi spolužáky v předmětech Softwarový týmový projekt 1 a 2. V rámci těchto předmětů byla vytvořena detailní analýza, kde jsou popsané moduly, které umožňují zobrazovat nebo editovat kity a reagencie.

2.1 Použité technologie

Pro výběr základní technologie jsme měli na výběr mezi dvěma nejpoužívanějšími jazyky, tedy Java EE a PHP. Vybrali jsme si Java EE, která nám přináší více možnosti než PHP. Po zkontaktování informačních techniků ÚHKT jsme dostali příkazem použít server Apache Tomcat. Jako databázový server jsme vybrali MySQL, a to hlavně zásluhou prověřenosti, dlouhodobé spolehlivosti.

Pro naprogramování webové aplikace bylo třeba ještě vybrat framework (Dnešní aplikace jsou už tak složité, že pro většinu funkcí se používají již abstraktně napsané části kódu známe jako frameworky, které se pouze konfigurují pro daný účel.) pro MVC (Model View Controller) a pro Grafické uživatelské prostředí. Jako MVC jsme zvolili JSF 2.2 (Java Server Faces) hlavně kvůli tomu, že je to normativní Java technologie a má velkou uživatelskou podporu. Jako GUI (Graphical User Interface) framework jsme zvolili Primefaces, které jsme považovali pro toto použití za nejvhodnější z hlediska přehlednosti.

Pro propojení Java kódu a databáze jsme použili systém Objektové relační mapování (ORM), jehož nejpoužívanější implementace je Hibernate. Tento systém umožňuje převést řádek tabulky přímo na objekt, což ušetřilo spoustu práce při programování.

2.2 Počáteční stav

Kity a reagencie jsou zobrazeny pomocí komponenty DataTable, která je součástí frameworku Primefaces. Do této tabulky jsou přidány funkce pro filtrování a řazení podle všech sloupců. Pro přehlednost je zde přidána funkce stránkování. Pro úpravy je třeba vybrat položky pomocí zaškrtávacích políček nebo lze pro výběr použít klávesové zkratky Ctrl a Shift (známe z Windows).

Implementována je základní podpora přihlašování, avšak je nepříliš bezpečná. Heslo je chráněné pouze funkcí MD5 a to bez použití solení, což by mohlo v případě napadení databáze znamenat prolomení většiny hesel. Tento proces jsem vylepšil použitím mnohem modernějšího nástroje BCRYPT. Systém také dokáže zobrazit základní statistiku o vlastnostech a počtu kitů v databázi.

Kapitola 3

Revidování Analýzy

3.1 Cíl aplikace

Cílem aplikace je poskytnout pracovníkům ÚHKT kvalitní podporu při evidování kitů a reagencií. Následující podkapitoly popisují softwarovou analýzu tohoto projektu. Úvodní část je věnována požadavkům, které mají na tuto aplikaci zaměstnanci oddělení v čele s Ing. Milenou Vranou. Dále se zabývá případy užití (doplněno o užití nových modulů) a je zde popsán revidovaný doménový model.

3.2 Analýza požadavků

Vývoj každé aplikace nebo její části by měl vždy začínat analýzou požadavků. Zajímá nás tedy, co od tohoto softwaru jeho zákazník očekává. Tato část byla řešena nemalým počtem schůzek s koncovým zákazníkem. Nároky na aplikaci jsme popsali jako soubor funkčních a nefunkčních požadavků. Bohužel se zde projevilo, že požadavky se sbíraly od uživatelů, kteří nejsou v IT oblasti příliš zdatní, což zavinilo nedorozumění a následné úpravy v této analýze.

3.2.1 Dohodnutá funkcionalita

Do aplikace budou mít přístup pouze určení pracovníci. Ty bude moci přidávat administrátor. Každý uživatel musí být jednoznačně určen svým uživatelským jménem. Pro přihlášení bude požadováno heslo. Uživatel bude moci své údaje upravovat (včetně změny hesla), pouze uživatelské jméno je fixní a nelze ho změnit. Aby bylo možné spravovat evidované položky, je třeba je zobrazit, umožnit jejich upravování a přidávání. V případě kitů a reagencií aplikace umožní následující možnosti.

- Přidávání položek

- Upravení položek
- Otevření položek (Automatické předvyplnění data otevření na dnešní datum)
- Spotřebování položek (Automatické předvyplnění data, kdy byla položka spotřebována)
- Při přijmutí objednávky vygenerování a automatické přidání položek

Aplikace musí být přehledná a intuitivní především pro uživatele navyklé jen na kancelářské balíky (Microsoft Office). Kvůli přehlednosti je třeba umožnit uživatelům skrývat jednotlivé vlastnosti (sloupce) s vyjímkou těch potřebných pro jednoznačnou identifikaci řádku. Protože položky mají velký počet atributů, aplikace skryje pro danou operaci nerelevantní sloupce kvůli přehlednosti. Pro jednoduché použití aplikace odfiltruje již spotřebované položky (fyzicky již neexistující) a zobrazí je pouze na přání uživatele. Každý uživatel by měl mít k dispozici detailní příručku obsahující podrobný manuál k aplikaci.

Otázka zálohování je dnes pro každou instituci naprosto zásadní, proto aplikace musí umožňovat zálohovat veškerá data na žádost uživatele i automaticky. Automatická záloha bude ukládána do předem definované cesty (path) jednou denně. Je již na provozovateli, aby zajistil, zda aplikace bude mít dostatečná práva pro zápis do této složky. Pokud dojde k problému, který smaže nebo poškodí data v databázi, umožní aplikační modul importér všechna zálohovaná data znovu nahrát přes grafické uživatelské rozhraní.

V ÚHKT by se nemělo stát, že kity dojdou nebo projdou (expirují). Proto další požadavek je varování před blížící se expirací a před spotřebováním všech látek. Tato varování budou moci spravovat běžní uživatelé, kteří je mohou skrýt, aby nemuseli rozeznávat mezi novým varováním a těmi, co již systém vygeneroval před týdnem, ale nová objednávka ještě nedorazila. Kity nyní ústav objednává od tří dodavatelů. Do aplikace bude možné tohoto dodavatele přidat a také importovat jeho katalog látek, které prodává. Při vytváření objednávky aplikace poskytne katalog, který bude možno filtrovat podle určitého výrobce. Z tohoto katalogu může pracovník vybrat položky, které chce objednat, vytvořit tiskovou sestavu všech objednávaných položek pro účel písemné objednávky zboží a celou objednávku vytisknout.

3.2.2 Uživatelé systému

Budeme rozlišovat 3 druhy uživatelů. U následující uživatelských rolí platí dědičnost. Tedy například správce je obdařen všemi vlastnosti uživatele.

3.2.3 Uživatel s právem číst

Případ uživatele, který nerozumí zcela problematice kitů a reagencií, ale z určitého důvodu potřebuje prohlížet či kontrolovat uložená data.

3.2.4 Uživatel s právem zapisovat

Uživatel z právem zapisovat je nejčastější případ využití této aplikace. Toto právo umožní uživateli přidávat nebo upravovat data v systému. (Reagencie, Kity a Objednávky)

3.2.5 Administrátoři systému (Správci)

Administrátor je uživatel, který bude zodpovídat za nastavení systému. Tento uživatel je zodpovědný za přípravu systému před tím, než ho začnou používat běžní uživatelé, což zahrnuje:

- Nastavení "Důvodů vyřazení"nebo "Výrobců kitů".
- Importovaní dat do systému přes webové rozhraní. (Kity, Reagencie a Katalog produktů)
- Vytvoření uživatelských účtů.

Pokud uživateli odebereme právo číst, stává se z něj zablokovaný uživatel. (Nemůže se přihlásit do systému.) Systém si však jeho údaje dále pamatuje pro případ auditu.

Aplikace musí podporovat webové prohlížeče Mozilla Firefox (verze 34.0 a vyšší) a Internet Explorer (verze 8.0 a vyšší). Musí být spustitelná na webovém serveru Apache Tomcat a musí využívat databázi, která běží pod otevřenou licencí.

3.2.6 Funkční požadavky

- Webová aplikace - Aplikace musí běžet na serveru a být přístupná na lokální síti.
- Podrobná uživatelská příručka - V příručce by měl každý uživatel aplikace najít vše, co potřebuje.
- Uložení informace o poslední změně u jednotlivé položky v databázi v podobě uživatelského jména a času změny.
- Správa položek
  - Kity

    - Přidání
    - Upravení vybraných
    - Otevření vybraných
    - Spotřebování vybraných
    - Zobrazení spotřebovaných (Prezentováno jako archiv)
    - Vytvoření varování pro vybraný typ
- Reagencie
    - Přidání
    - Upravení vybraných
    - Otevření vybraných
    - Spotřebování vybraných
    - Zobrazení spotřebovaných (Prezentováno jako archiv)
    - Vytvoření varování pro vybraný typ
- Objednávky
    - Vytvoření
    - Upravení
    - Přijmutí (Automaticky předvyplní doručené kity, s možností dodatečné změny.)
    - Zobrazení již přijatých
    - Vytištění objednávky - Realizované exportem do XLSX formátu
- Katalog
    - Importování ze souboru
    - Manuální přidání
    - Odstranění
- Varování
    - Vytvoření nastavení pro varování
    - Upravení nastavení pro varování
    - Skrýt aktuální varování
    - Zobrazit skrytá varování
- Nastavení
    - Nastavení (Pouze administrátor)
        - Přidání/Odebrání dodavatele kitů
        - Přidání/Odebrání důvodu vyřazení
    - Export dat
    - Import dat
- Správa uživatelů (Pouze administrátor)

    * Vytvoření
    * Upravení
  - Nastavení účtu
    * Upravení vlastních údajů

3.2.7 Nefunkční požadavky

- Přívětivé uživatelské rozhraní.
- Intuitivní uživatelské prostředí.
- Výkon - Systém musí pracovat dostatečně rychle, aby nezdržoval pracovníky při práci.
- Spolehlivost - Aplikace musí být především spolehlivá v oblasti persistence. Systém bude pravidelně zálohovat všechna data.
- Rozšiřitelnost - Systém by měl jít snadno rozšířit o další funkcionalitu, aniž by se muselo zasahovat do funkčnosti již funkčních komponent.
- Bezpečnost - Systém bude ověřovat identitu jeho uživatelů.
- Podpora výše specifikovaných webových prohlížečů.
- Aplikace bude realizovatelná pomocí softwaru s otevřenou licencí.

3.3 Případy užití

Každý uživatel má svojí roli. Nejběžnější případy užití, které platí pro všechny (kromě zablokovaného) uživatele, jsou

- Přihlášení do systému
- Odhlášení
- Zobrazení dat

3.3.1 Uživatel s právem zapisovat

Běžný uživatel má v tomto systému právo spravovat:

- Kity
- Reagencie
- Varování
- Objednávky

Dále má možnost upravit svoje osobní údaje kromě fixního přihlašovacího jména.

3.3.2 Administrátor systému (Správce)

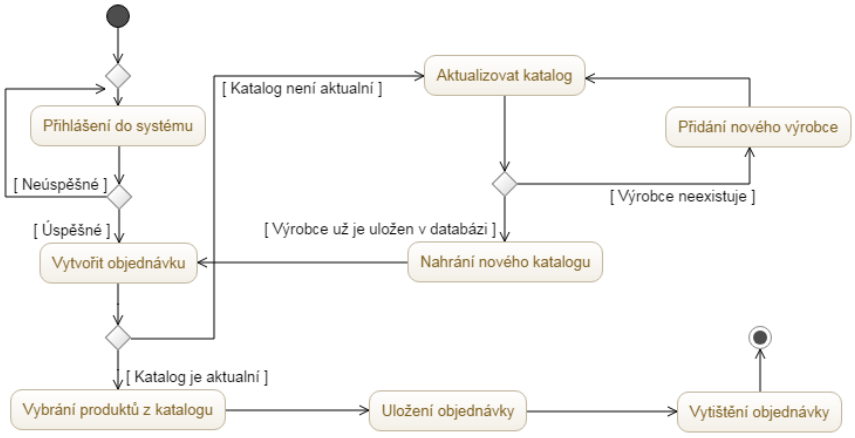
Role administrátora je rozšířením běžného uživatele, tzn. dědí všechny jeho případy užití. Navíc má tyto možnosti

- Nastavení systému
  - Přidat nebo odebrat dodavatele kitů.
  - Přidat nebo odebrat důvod vyřazení. (Například jestli byl kit spotřebován nebo jinak vyřazen.)
  - Importovat kity
  - Importovat reagencie
- Správa uživatelů
- Správa nastavení varování

3.4 Diagram aktivit

„Diagram aktivit je jeden z UML diagramů, které popisují chování. Tento diagram se používá pro modelování procedurální logiky, procesů a zachycení workflow.“ [4] Diagramů aktivit bylo v původním projektu vytvořeno mnoho. Protože tato kapitola slouží spíše jako výčet jednotlivých kroků analýzy, vybral jsem pouze jeden z těchto diagramů, který popisuje přidání objednávky do systému.

Obrázek 3.1: Diagram aktivit - Vytvoření objednávky

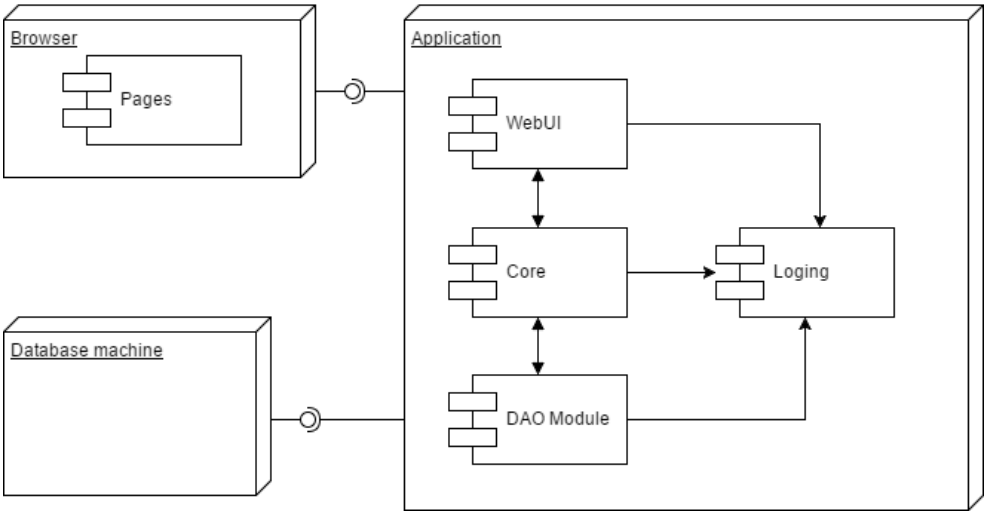


3.5 Doménový model

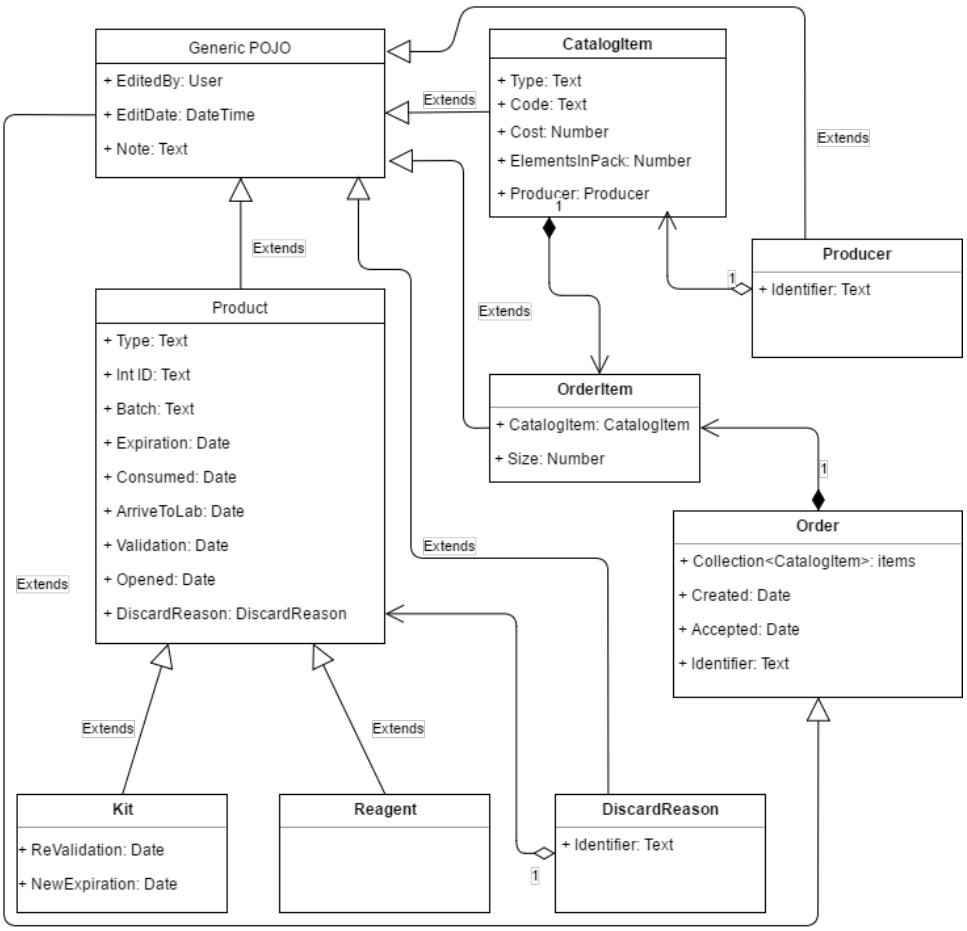
V doménovém modelu se analytik snaží postihnout základní entity systému. Model je platformě nezávislý, tedy neváže se k určitému programovacímu jazyku. Neměly by se tedy zde uvádět ani datové typy. Nesmí být příliš složitý, protože se většinou diskutuje se zákazníkem, pro kterého by mohlo být velké množství entit matoucí. V tomto modelu se pro zjednodušení zakreslují entity jako třídy. Mezi těmito třídami se tvoří vztahy (asociace) zakreslené pomocí čar.

Doménový diagram této aplikace byl příliš velký a nepřehledný. Proto jsem z této analýzy dvě části, které zde popíši. První model se zabývá rozdělením projektu na jednotlivé moduly. Druhý se věnuje POJO třídám, které v projektu zastávají velkou část funkcionality.

Obrázek 3.2: Doménový model rozhraní



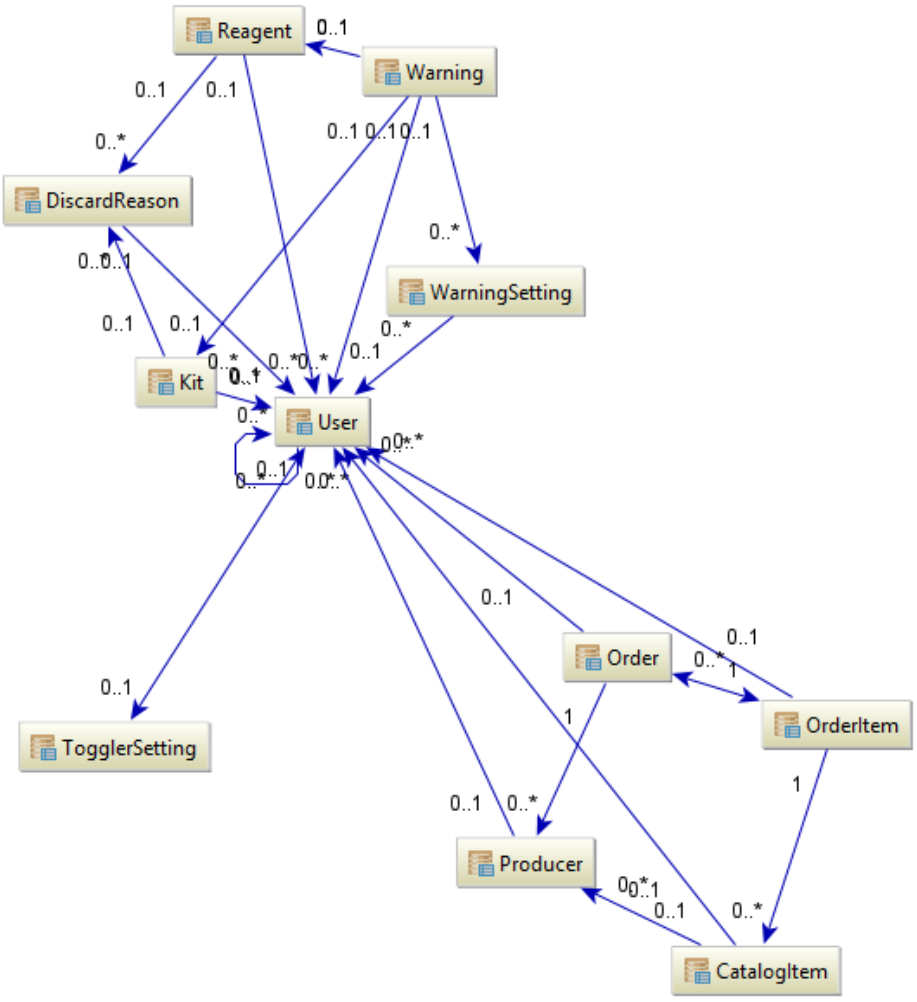
Obrázek 3.3: Doménový model POJO tříd



3.6 Datová vrstva

Databázová vrstva je zde odpovědná za uložení dat. Je implementována libovolnou SQL databází. Připojení do databáze zajišťuje JDBC (Java Database Connectivity) driver. Most mezi datovou a logickou vrstvou zajišťuje framework implementující JPA (Java Persistence API).

Obrázek 3.4: Schéma databáze

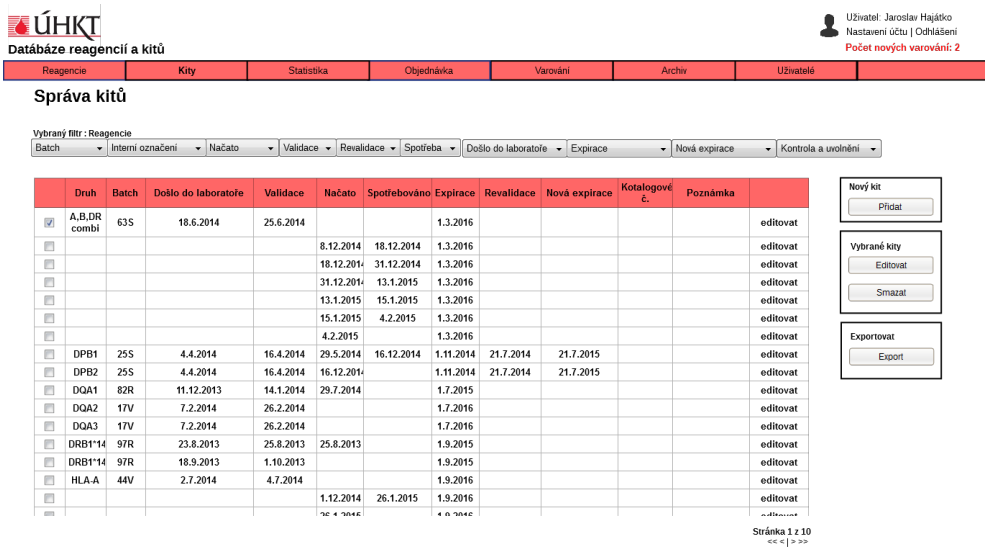


3.7 Návrh grafického uživatelského rozhraní

V předmětu SP1 a SP2 byl vypracován kompletní návrh Web GUI. Tohoto návrhu se projekt v rámci možností držel. Návrh kladl velký důraz na to, aby prostředí připomínalo Microsoft Excel, který jsou uživatelé zvyklí používat.

Obrázek 3.5: Návrh přihlašovací stránky

Obrázek 3.6: Návrh stránky pro správu kitů



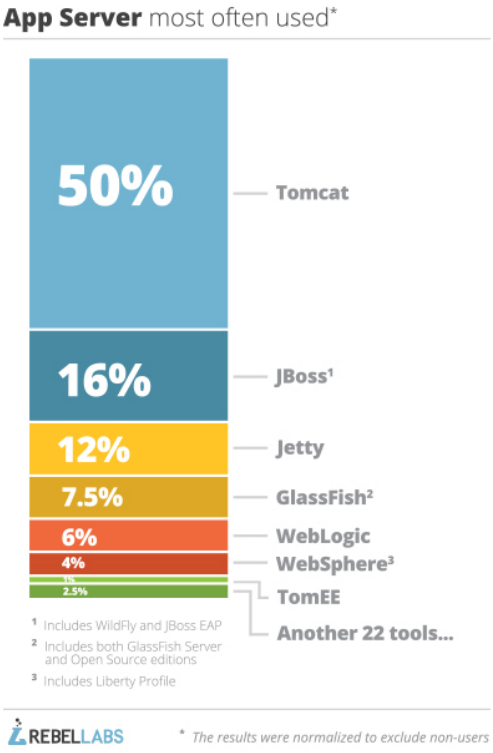
Kapitola 4

Implementace

V této kapitole se budu zabývat zajímavostmi z implementace a také uvedu a vysvětlím zjednodušené ukázky kódů.

4.1 Aplikační server

Obrázek 4.1: Využití Java aplikačních serverů v grafu



Tomcat je aplikační server od Apache Software Foundation, který spouští Java servlety a vykresluje webové stránky, pokud obsahují Java Server Page kódování. Je popsaný jako „referenční implementace“ pro Java Servlety a Java Server Page specifikaci. Tomcat je výsledek otevřené spolupráce vývojářů a je dostupný na webových stránkách Apache, jak v binární spustitelné verzi, tak i v té určené pro kompilaci. Tomcat může být použit buď jako samostatný Web Server nebo společně s dalšími webovými službami včetně Apache, Netscape Enterprise Server, Microsoft Internet Information Server (IIS), and Microsoft Personal Web Server. Tomcat vyžaduje Java Runtime Enterprise Environment (JRE) ve verzi 1.1. a vyšší.[5]

4.2 Grafické uživatelské rozhraní

GUI bylo vytvořeno pomocí frameworku PrimeFaces 5.3, který je založen na JSF. Na následujících stránkách jsou uvedeny a vysvětleny ukázky z již fungujícího GUI a k nim přiložené ukázky zdrojového kódu. Při psaní GUI jsem čerpal především z materiálu Primefaces User Guide.[6]

4.2.1 Zobrazení dat

Pro zobrazení dat byl použit p:dataTable tag, který je vyobrazen v následující ukázce. V ukázce je vidět, že tato XHTML stránka pracuje s třídou KitBean, která se stará o všechny operace s kity. Definice sloupců je zde vcelku primitivní.

```
1  <p:dataTable  var="kit"
2               value="#{kitBean.all}"
3               filtered="#{kitBean.filtered}"
4               selection="#{kitBean.selected}">
5
6    <p:column filterBy="#{kit.type}"
7              sortBy="#{kit.type}"
8              headerText="#{txt.type}"
9              filterMatchMode="contains">
10     <h:outputText value="#{kit.type}" />
11   </p:column>
12
13   <p:column filterBy="#{kit.internalId}"
14             sortBy="#{kit.internalId}"
15             headerText="#{txt.intId}"
16             filterMatchMode="contains">
17     <h:outputText value="#{kit.internalId}" />
18   </p:column>
19
20 </p:dataTable>
```

4.2.2 Výhody a nevýhody kombinace JSF + Primefaces

Tato kombinace frameworků přináší mnoho výhod i nevýhod. Mezi největší výhody patří velmi jednoduché používaní a prudce stoupající učební křivka na počátku užívání této technologie. I když vývoj Primefaces ještě neskončil, verze 6.0 byla vydána 7. 6. 2016, naráží už tato technologie na svá omezení způsobená především stářím JSF, i když jde o normativní Java technologii s velkou uživatelskou podporou. V dnešní době se intenzivně prosazují frameworky AngularJS2 a React (zejména kvůli Facebooku a Instagramu, které jsou na nich postavené), založené na TypeScriptu.

4.3 Import a export dat ze souborů

Protože jsou nyní data uložena v souborech s příponou xls a xlsx, bylo je třeba importovat do aplikace. Následně pak i exportovat, což je určeno i jako druhý způsob zálohování. Byla zvážena následující možnosti.

- Samostatná aplikace pro importovaní dat v JavaFX
- Import pomocí Webového rozhraní aplikace.
- Skripty pro převod do SQL insert.

Jako nejvíce uživatelsky přívětivou metodu jsem zvolil import pomocí Webového rozhraní.

4.3.1 Apache POI framework

K těmto operacím jsem vybral a použil Apache POI. Ten zajišťuje jednoduché čtení a zápis do prakticky všech souborů z rodiny Microsoft Office. Při importu a exportu dat je samozřejmě potřeba přesouvat data mezi klientem a serverem. Tuto funkci zajišťují Primefaces s komponentami Download a Upload file. Následuje zjednodušená ukázka čtení souboru v Apache POI.

Ukázka čtení XLS/XLSX souboru

```
InputStream fis = file.getInputstream();
Workbook wb = new HSSFWorkbook(fis); // for XLS format
// Workbook wb = new XSSFWorkbook(fis); // for XLS format
Sheet ws = wb.getSheetAt(0); // Open first sheet
int rowNum = ws.getLastRowNum() + 1;
for (int i=1; i<rowNum; i++){
   Row row = ws.getRow(i);
   String textInColumnA =
       row.getCell(0,row.CREATE_NULL_AS_BLANK).getStringCellValue();
```

```
   Integer numberInColumnB =
       row.getCell(1,row.CREATE_NULL_AS_BLANK).getNumberCellValue();
   Date dateInColumnC =
       row.getCell(2,row.CREATE_NULL_AS_BLANK).getDateCellValue();
}
```

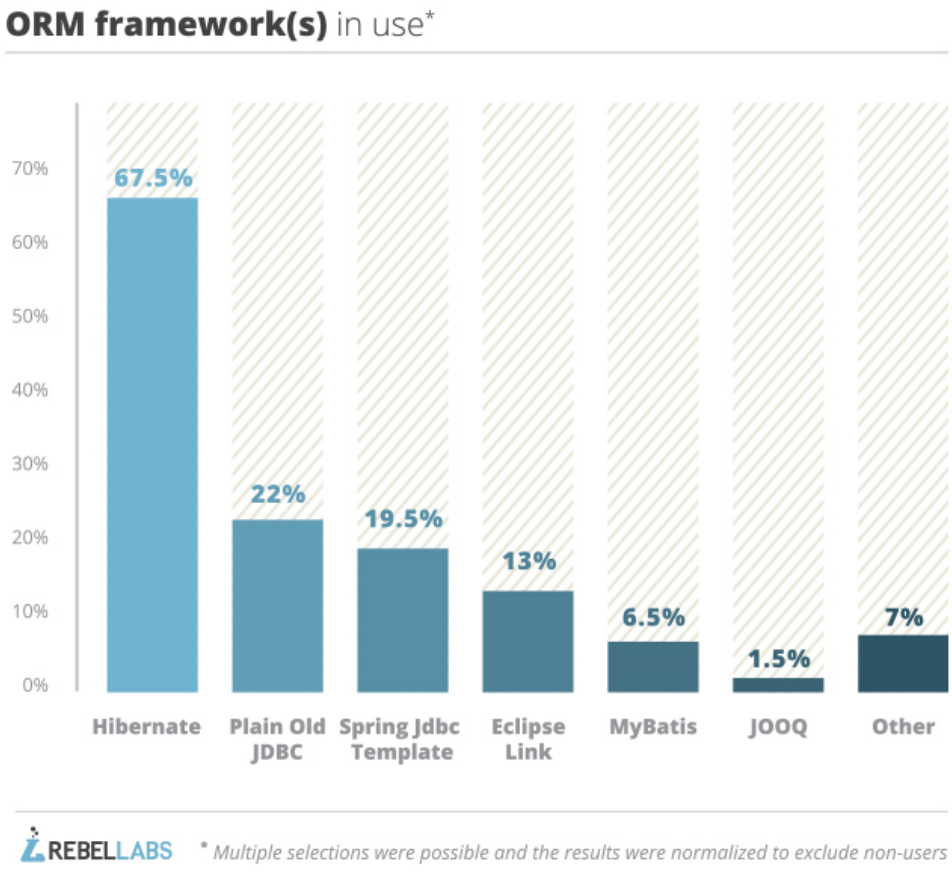
4.4 Objektově relační mapování

ORM zajišťuje most mezi relační databází a programovacím jazykem. Ve většině moderních programovacích jazyků se používá objektový model. Ten však nekoresponduje s relačními databázemi, kde jsou vlastně jenom řádky v tabulce. ORM zajistí přeformátování řádků tabulky nebo jakéhokoli výstupu (SELECT) na instance třídy v programu.

Konverze mezi relační databází a modelovými objekty není bezproblémová. Je spojena s řadou komplikací souhrnně označovaných jako object-relational impedance mismatch. Tyto problémy, resp. jejich možné řešení, jsou důvodem pro existenci návrhových vzorů pro ORM.[7]

Pro výběr ORM frameworku jsem použil následující graf od RebelsLabs. Z grafu jasně vyplývá převaha Hibernate frameworku nad ostatními.

Obrázek 4.2: Využití objektově relačních frameworků v grafu



4.4.1 Hibernate ORM framework

Hibernate ORM je databázově nezávislý. To znamená, že je možné vyměnit používanou databázi pouze změněním konfiguračního souboru (viz ukázka). Hibernate implementuje Java Persistence API (JPA), což nám umožňuje tento framework nahradit velmi snadno jiným. Hibernate umožňuje programátorovi zmapovat i velmi složité vztahy mezi jednotlivými tabulkami. Mezi ně patří:

- @OneToOne - Vytvoří vztah 1 ku 1.
- @ManyToOne - Vytvoří vztah * ku 1.
- @ManyToMany - Vytvoří vztah * ku *.
- @ForeignKey - Označí proměnnou jako cizí klíč.
- @JoinColumn - Připojí určený sloupec.

Další věc, kterou za nás Hibernate může dělat, je pojmenovávání tabulek a sloupců. Je označována NamingStrategy. Existují dva základní typy.

- Jméno sloupce a řádku je vyplněno implicitně v anotacích @Column a @Table.
- Jméno sloupce a řádku není vyplněno a program ho vygeneruje pomocí názvu třídy nebo proměnné.

Následující ukázka kódu znázorňuje použití ORM v praxi. Pro každou tabulku lze vytvořit POJO třídu, do které aplikace transformuje jeden řádek tabulky. Využívá automaticky generované ID a první způsob NamingStrategy.[8]

```
@Entity
@Table(name="users")
public class User extends ItemTemplate {

   @Id
   @GeneratedValue
   @Column(name="id", unique=true)
   private int id;

   @Column(name="username", length=64, unique=true, nullable=false)
   private String username;

   @Column(name="email", length=64, unique=true, nullable=false)
   private String email;

   @Column(name="firstName", length=64, nullable=false)
   private String firstName;

   @Column(name="lastName", length=64, nullable=false)
   private String lastName;

   public User(){}
}
```

Jak můžete vidět v ukázce kódu pro ORM, potřebuje Hibernate prázdný konstruktor. Po vytvoření objektu jsou do něj zapsána data pomocí referencí. Tato metoda víceméně popírá objektový princip, protože může bez setter metody zapisovat i do privátních proměnných tříd. V další ukázce si ukážeme, jak jednoduše je možné přečíst data z databáze. Tato funkce je generická, může se tedy použít na jakoukoli třídu, která je zavedená ve třídě Hibernate Frameworku.

```
public <T> List<T> getAll(final Class<T> type) throws DbIOException {
```

```
        final CriteriaBuilder criteriaBuilder =
            session.getCriteriaBuilder();
        final CriteriaQuery<T> criteriaQuery =
            criteriaBuilder.createQuery(type).distinct(true);
        final Root<T> root = criteriaQuery.from(type);
        criteriaQuery.select(root);
        try {
            return session.createQuery(criteriaQuery).getResultList();
        } catch (Exception ex) {
            throw new DbIOException(ex);
        }
    }
```

4.4.2 Konfigurační soubor

V tomto souboru musíme definovat typ databáze, ke které se připojujeme. Pro tento účel zde slouží JDBC driver (Java Database Connectivity), který tvoří most mezi programem a databází. Protože v SQL jazycích pro jednotlivé databáze jsou malé rozdíly, je třeba definovat dialect, kterému bude databázový server rozumět. Jaký dialect vybrat pro určitou databázi se dozvíme v dokumentaci Hibernate.

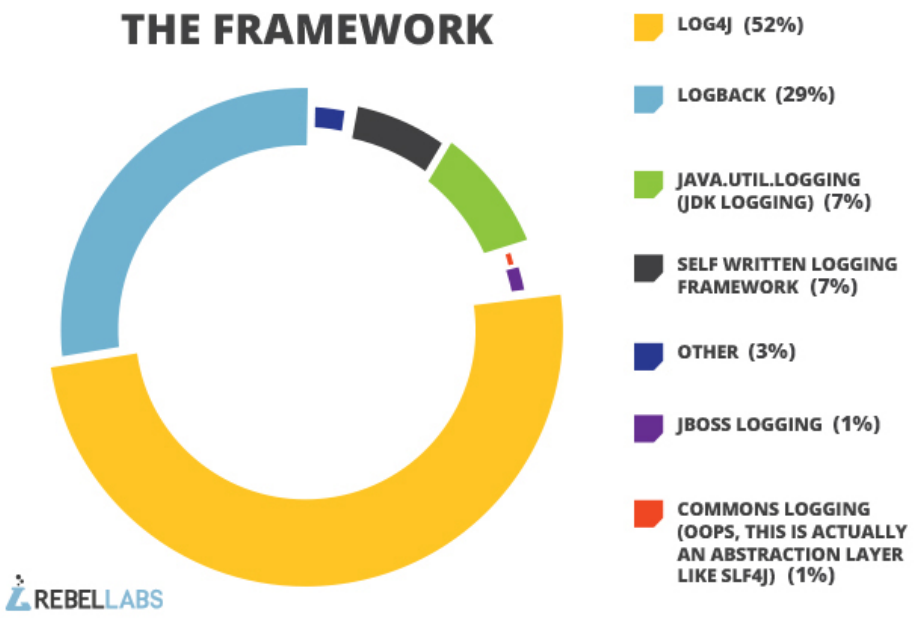
```
1  <hibernate-configuration>
2      <session-factory>
3          <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect<
               /property>
4          <property name="hibernate.connection.driver_class">com.mysql.jdbc.
               Driver</property>
5          <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/
               reagents</property>
6          <property name="hibernate.connection.username">admin</property>
7          <property name="hibernate.connection.password">password</property>
8          <property name="hibernate.hbm2ddl.auto">update</property>
9          <property name="show_sql">false</property>
10         <property name="lazy">false</property>
11         <property name="current_session_context_class">thread</property>
12     </session-factory>
13 </hibernate-configuration>
```

Konfigurační soubor musí být přiložen ve složce zdroje. Ta je bohužel zabalená ve WAR archivu (Web Archive), který se nehodí pro úpravu administrátorem. Naštěstí lze tento konfigurační soubor při startu serveru změnit.

4.5 Logování

„Vytváření logů z aplikací slouží k zaznamenávání informací o průběhu programu, informací sloužících k ladění aplikace a také informací o výskytu chyb v aplikaci. Proč používat zvláštní knihovny pro logování, když můžeme stejných výsledků dosáhnout přímým zápisem záznamů o činnosti aplikace do souboru nebo do standardního výstupu pomocí volání metody System.out.println()? Hlavním důvodem pro používání knihoven pro logování je jednoduchá parametrizovanost a jako další výhodu lze uvést snadnou dostupnost instance Loggeru (viz dále) ze všech tříd aplikace. Další podstatnou výhodou je i vysoká rychlost logování. Co je myšleno tou parametrizovaností? Pokud ladíte pomocí výpisů metody System.out.println(), musíte před dokončením aplikace odstranit všechny nepotřebné výpisy. Při použití specializovaných prostředků pro logování, stačí logování pomocí parametrického souboru vypnout. Stejně tak vám logování ušetří práci s nastavováním logovacích souborů atd.“ [9]

Obrázek 4.3: Využití frameworků pro logování v grafu



Pro vytváření logů jsem si vybral Log4j od Apache. Tento logger pracuje synchronně, takže se nemůže stát, že se zprávy na výstupu přeházejí. Framework také umožňuje nastavit každému řádku logu následující priority.

- DEBUG

- INFO
- WARN
- ERROR
- FATAL

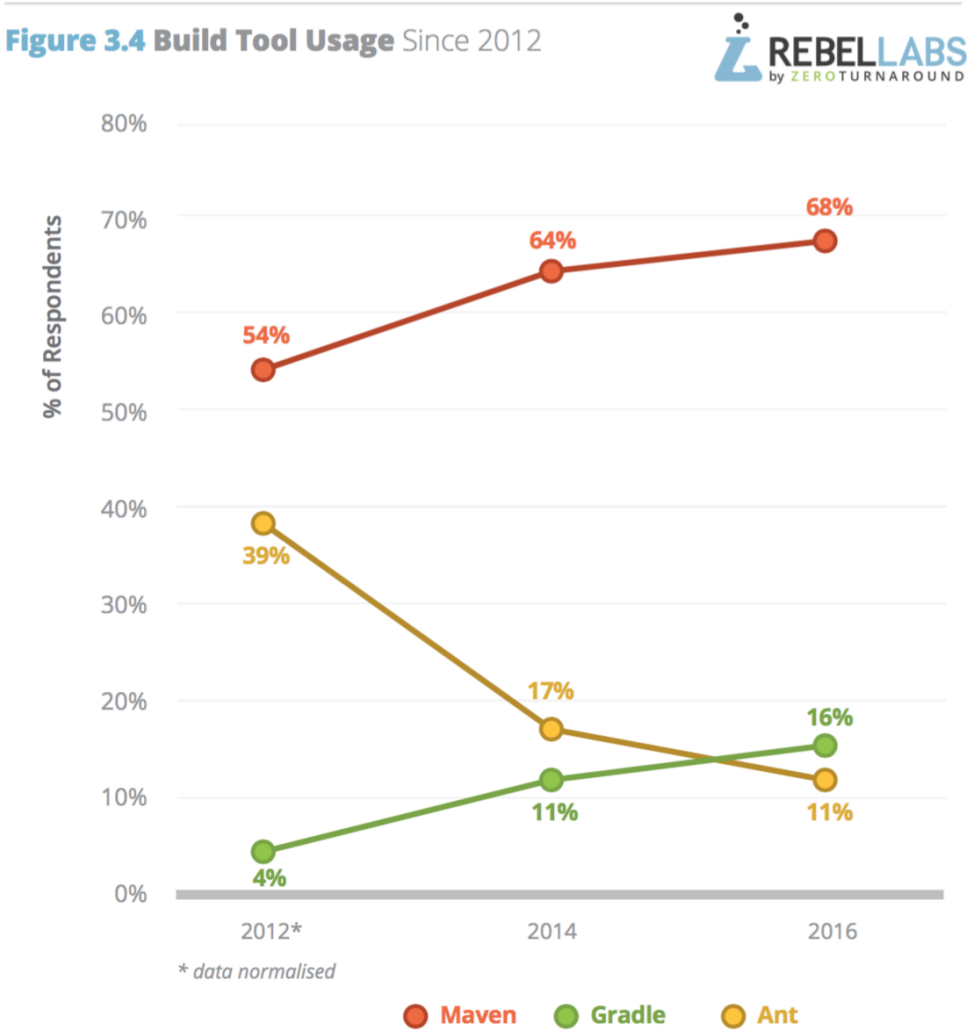
Základní konfiguraci provedeme souborem log4j.properties umístěným ve složce zdroje. Vzniká zde však stejný problém jako u základního nastavení Hibernate. Proto je dobré nastavovat logování do souboru až při běhu programu podle konfiguračního souboru, který se nachází mimo WAR soubor.

```
1 log4j.rootLogger=stdout
2
3 \# Redirect log to console.
4 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
5 log4j.appender.stdout.Target=System.out
6 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
7 log4j.appender.stdout.layout.ConversionPattern=\%d{yyyy-MM-dd HH:mm:ss} \%-5p
      \%c{1}:\%L - \%m\%n
8 log4j.appender.stdout.Threshold=INFO
```

4.6 Sestavení aplikace

„Sestavovací nástroj (Build Tool) je nástroj pro automatizaci vytvoření spustitelné aplikace ze zdrojových kódů. Sestavení zahrnuje kompilaci, pospojování a zabalení kódu do spustitelné formy. V malých projektech vývojáři často ručně spouštějí sestavení. Tato praxe není použitelná u velkých projektů, kde už vývojář nedokáže zajistit kvalitní sestavení. Použitím automatizace zde docílíme konzistentních výsledků.“[10]

Obrázek 4.4: Využití frameworků pro sestavení v grafu



Jako nástroj pro build jsem vybral Maven. Ten umí pomocí souboru pom.xml sestavit celou aplikaci. Nástroj se postará o stažení a importování externích knihoven. Programátor získává většinu informací ze zdrojového kódu a dokumentace. I v této věci nám Maven pomůže, pokud do svého projektu importuje cizí knihovnu. Aplikace pak sama zajistí stažení zdrojového kódu. Následuje ukázka nastavení Maven Buildu.

- repositoty - Repositář sloužící pro stahování externích knihoven.
- dependency - Importuje do projektu externí knihovnu.
- build - Nastavuje, jak se má aplikace sestavit.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>cz.cvut.fit.reagentsApp</groupId>
    <artifactId>reagentsApp</artifactId>
    <version>1.0-SNAPSHOT</version>

    <repositories>
      <repository>
            <id>Central</id>
            <name>Maven Repository</name>
            <url>http://repo1.maven.org/maven2</url>
            <layout>default</layout>
        </repository>
    </repositories>

    <dependencies>
      <dependency>
            <groupId>org.jboss.weld.servlet</groupId>
            <artifactId>weld-servlet</artifactId>
            <version>2.4.3.Final</version>
        </dependency>
    </dependencies>

    <build>
        <finalName>reagentApp</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

4.7 Lombok project

Protože velmi značnou část aplikace tvoří POJO třídy, začala být aplikace poměrně nepřehledná, jelikož 70% POJO tříd tvořily get a set funkce. Hledal jsem proto framework, který mi umožní zredukovat délku kódu. Nalezl jsem

Lombok, který dokáže nahrazovat tyto jednoduché metody pomocí anotací. Při analýze, zda je tento způsob vhodný, jsem narazil na mnoho argumentů odpůrců anotací.

- + Pro zasvěceného uživatele zpřehlední kód
- + Při změně jména třídy se automaticky přegenerují i get a set metody.
- - Pro nezasvěcené znepřehlední kód. Fakticky se volají metody, které tam nejsou.
- - Vyžaduje rozšíření, které přesvědčí IDE, že tam generované metody jsou, i když se do kódu zapíšou až při buildu.

```
public class Kit {
   @Getter
   private Integer id;
   @Setter
   private String name;
}
```

Změní na

```
public class Kit {
   private Integer id;
   private String name;

   public Integer getId(){
      return id;
   }
   public void setName(){
      this.name = name;
   }
}
```

- @Getter Vytvoří metodu get.
- @Setter Vytvoří metodu set.
- @Data Vytvoří pro každé pole třídy get a set metodu.
- @NoArgsConstructor Vytvoří bezparametrický konstruktor.
- A mnoho dalších.

Kapitola 5

Testování

5.1 Proč testovat?

„Automatizované testování softwaru je stále v praxi velmi podceňovanou disciplínou. Přitom jeho využití sebou přináší výrazné úspory prostředků a samozřejmě zvýšení kvality výsledného produktu. Automatizovat lze však nejen samotnou exekuci vybraných manuálních testů, ale také části nebo dokonce celý proces testování softwaru. Některé nástroje, které porovnávají výsledky provedených testů, instalují a konfigurují aplikaci nebo vytváří z analýzy testovací případy. Pokud jsou pro daný software vytvořené stabilní manuální testovací případy, které se opakovaně vykonávají, je nasnadě otázka, zda by nebylo možné tyto vykonávat automaticky (tedy bez zásahu lidského faktoru). Oblast automatizace testů je čím dál populárnější a často vyhledávanou možností, jak software testovat.“ [11]

5.2 Metody testování

Existují různé metody, které mohou být použity pro testování softwaru. Následuje popis metod, které můžeme použít.[12]

5.2.1 Black-Box Testing

Tato metoda se dá použít, když nemáme žádné informace o interních procesech v aplikaci. Obvykle se používá při testování aplikace, kde tester nemá přístup ke zdrojovým kódům. Tester obvykle testuje uživatelský interface aplikace. Zadá vstup a podle toho vyhodnotí výstup, přičemž nemá žádnou znalost toho, jak k této transformaci došlo. Následující seznam popisuje výhody a nevýhody.

- Lze použít velmi snadno i pro rozsáhlé aplikace.
- Není třeba přístup ke zdrojovému kódu.

- Test je prováděn z pohledu uživatele.
- Testy jsou složitější na vytvoření.

5.2.2 White-Box Testing

White-box testing je detailní průzkum interních logických struktur kódu. Pokud chce tester vytvořit tento test, musí znát podrobně interní kód aplikace. Je třeba podívat se do kódu a najít oblasti, kde by se s největší pravděpodobností mohly vyskytnout chyby a tyto oblasti důkladně otestovat.

- Lze lépe odhadnout, jaká testovací data budou nejvhodnější.
- Pomáhá optimalizovat kód.
- Umožňuje maximální pokrytí kódu testy.
- Je třeba opravdu zkušených testerů.

5.2.3 Grey-Box Testing

Grey-box testing je technika na pomezí mezi předešlými dvěma technikami. Používá se pouze, pokud máme omezený pohled na zdrojový kód, jako například přístup do databáze nebo dokumentace. S těmito informacemi mohou testeři lépe vyhodnotit a připravit potřebné testovací scénáře.

- Kombinuje benefity obou předešlých způsobů.
- Otestování všech vstupů/výstupů je nemožné, proto zůstává mnoho oblastí neotestováno.

5.3 Testování webové aplikace

V této kapitole je čerpáno z článku [13] na tutorialspoint.com. Protože se v tomto případě jedná o webovou aplikaci, je třeba jí i tak otestovat. Proto se tato podkapitola bude věnovat právě testům, které je důležité provést před uvolněním aplikace.

5.3.1 Funkční testy

- Kontrola „mrtvých stránek“, tedy stránek s chybou a špatné přesměrování.
- Prověření validace všech vstupních údajů. Test prověřuje, zda systém například nepovolí uživateli přidat příliš dlouhý text, který by se nemohl uložit do databáze.

- Testování zadáváním chybných vstupů.
- Prověření integrity dat.

5.3.2 Testování použitelnosti

Ověřuje se, jak složité je používání aplikace.

- Test navigace a ovládání.
- Test zobrazení správného obsahu.
- Testování intuitivnosti aplikace.

5.3.3 Testování kompatibility

Kompatibilita je doménou především GUI. Musíme zde zajistit, že všechny zdroje pro správné vykreslení stránky budou dostupné a stránka se přijatelně zobrazí na malém i velkém displeji.

- Test kompatibility webového prohlížeče.
- Test kompatibility operačního systému. (Windows, Linux, iOS)
- Test kompatibility se zařízením. (Desktop, Notebook, Mobil)

5.3.4 Testování výkonu

„Proč je testování výkonu důležité?
Při návrhu aplikace, k niž se bude přistupovat přes internet, musí vývojový tým pamatovat na to, že k dané aplikaci mohou přistupovat stovky nebo i tisíce uživatelů. Toto zatížení souběžným používáním může klást na systém extrémní nároky, způsobovat neočekávaná zpoždění a výsledkem může být chabé provádění z hlediska uživatelů. Pokud však svou aplikaci důkladně otestujeme a vyladíme k optimálnímu výkonu při zatížení, manažeři i vývojáři softwaru si budou jisti tím, že jejich kód poběží na optimální úrovni a budou mít potřebné údaje k naplánování kapacitních potřeb sídla.“ [14]
Při programování aplikace není testovací server zatížen velkým počtem požadavků, což může způsobit předtím neobjevené chyby. Proto je třeba aplikaci vytížit a sledovat, co se stane.

- Load test - Je základní test, kterým by měla aplikace projít. Aplikační server zatížíme předpokládaným standardním provozem a sledujeme výsledek.
- Stres test - Test, ve kterém systém zatížíme na maximum. Pomáhá určit například maximální únosné zatížení pro tento aktuální hardware.

- Soak Test - Systém zatížen větším výkonem po delší dobu. Webové servery jsou většinou zatíženy kontinuálně. Tento test zkoumá rychlost po delší době, kdy je server zatížen.

5.4 Testování aplikace pro správu kitů a reagencií

5.4.1 Automatické testování

Pro automatické testování jsem měl na výběr mezi testovacími frameworky, které nabízí Intellij Idea:

- Groovy JUnit
- TestNG
- JUnit3
- JUnit4

Z těchto jsem využil JUnit4, ve kterém jsem otestoval většinu tříd. Následuje ukázka testování. Následující ukázka kódu ukazuje jeden z jednoduchých testů, který testuje, jak se zachová třída KitBean, když jí volá neautorizovaný uživatel.

```
@RunWith(JUnit4.class)
public class KitBeanTest extends TestCase {

    @Test
    public void nonLogedUserTest(){
        try{
            KitBean bean = new KitBean(new CurrentUser());
            assertTrue(false);
        }
        catch (LoginExpiredExeption ex){
            assertTrue(true);
        }
    }
}
```

5.4.2 Testování validity stránek

V tomto projektu validita stránek nebyla testována, protože generování HTML kódu zajišťuje Mojarra 2.2 a PrimeFaces 5.3, a proto v tomto případě jako programátor nemohu tuto část kódu nijak ovlivnit.

5.5 Selenium testy

„Selenium je nástroj pro automatické testování webových aplikací, skládající se z několika navzájem se doplňujících komponent: Selenium IDE, Selenium RC, Selenium WebDriver a Selenium Grid. Vyvinut je v programovacím jazyku Java a je možné jej používat na různých platformách.“[15]

Selenium testování je pro server, který je zaměřený na webové rozhraní, velmi užitečný. Lze provádět automatizované testy z pohledu klienta, což je pro tento typ aplikace velmi výhodné. Umožňuje testovat většinu částí aplikace zároveň, což je u standardních Unit testů velmi složité.

5.5.1 Selenium IDE

„Selenium IDE je integrované vývojové prostředí pro vytváření testů. Je implementováno jako rozšíření do prohlížeče Firefox a dovoluje nahrávat, upravovat a ladit testy. Selenium IDE zahrnuje celé jádro Selenium, což umožňuje jednoduše a rychle nahrávat a přehrávat testy v aktuálním prostředí.“[16]

- Přihlášení uživatele
- Dostupnost stránek
- Přidání kitu
- Upravení kitu specifikovaného pomocí Typu, Batche a Interního Id.
- Načnutí a spotřebování kitu.

Zmíněné testy budou vypracovány a přiloženy k práci jako JUNIT4 test a v HTML formátu importovatelném do Selenium IDE.

5.5.2 Test dostupnosti stránek

Následující ukázka je zkrácený případ vygenerovaného testu ze Selenium IDE. Celý test je dlouhý 216 řádků. Aby ho bylo možné zobrazit na jedné stránce a lépe pochopit, musel jsem ho zkrátit. Test se skládá z přihlášení uživatele, přistoupení ke stránce a kontroly titulku.

```
public class PageAccess {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
```

```
    driver = new FirefoxDriver();
    baseUrl = "http://localhost:8080/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testPageAccess() throws Exception {
    driver.get(baseUrl + "/login.html");
    driver.findElement(By.id("j_username")).clear();
    driver.findElement(By.id("j_username")).sendKeys("admin");
    driver.findElement(By.id("j_password")).clear();
    driver.findElement(By.id("j_password")).sendKeys("admin");
    driver.findElement(By.id("loginButton")).click();
    driver.findElement(By.id("headerForm:kitButton")).click();
    driver.findElement(By.id("headerForm:current")).click();
    try {
      assertEquals("Kity", driver.getTitle());
    } catch (Error e) {
      verificationErrors.append(e.toString());
    }
    driver.findElement(By.id("headerForm:kitButton")).click();
    driver.findElement(By.id("headerForm:archive")).click();
    try {
      assertEquals("Archiv kit", driver.getTitle());
    } catch (Error e) {
      verificationErrors.append(e.toString());
    }
    driver.findElement(By.id("headerForm:kitButton")).click();
    driver.findElement(By.xpath("//a[@id='headerForm:add']/span[2]")).click();
    try {
      assertEquals("Pidat kity", driver.getTitle());
    } catch (Error e) {
      verificationErrors.append(e.toString());
    }
   }

  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }
}
```

Tento test je pro stránky napsané v JSF velmi důležitý, protože XHTML předlohy stránek se kompilují až za běhu a velmi špatně se automaticky kontrolují. I sebemenší chyba na stránce znemožní její vykreslení. Tento automatický test tyto chyby detekuje v přiměřeném čase.

5.6 Manuální testování uživateli v ÚHKT

S paní Ing. Milenou Vranou v ÚHKT bylo dohodnuto testování aplikace formou používání Beta verze aplikace. Na toto testování bylo vyčleněno 30 kalendářních dnů, ve kterých by uživatelé měli odhalit většinu chyb. Dále budou uživatelé vznášet návrhy na různá vylepšení aplikace. Pokud tyto návrhy budou jednoduše proveditelné, mohou se uživatelé dočkat implementace již v testovacím režimu. Větší úpravy je možné nechat na případnou diplomovou práci. Na konci testování dostanou uživatelé krátký dotazník, jehož účelem je zjistit kvalitu zpracování aplikace.

Obrázek 5.1: Anketa předložená uživatelům

Aplikace pro správu kitů a reagencií

Dotazník pro uživatele, kteří testovali aplikaci pro správu kitů a reagencií. Bude využit jako součást bakalářské práce prezentující její kvalitu.

*Povinné pole

Jak intuitivní je aplikace? *

	1	2	3	4	5	
Neintuitivní	○	○	○	○	○	Intuitivní

Kolik chyb jste našel(a) během testování? *

Vyberte

Jak závažné chyby byly? *

	1	2	3	4	5	
Chyba mě v užívaní neomezila	○	○	○	○	○	Aplikace je nepoužitelná

Jak rychle po nahlášení byly chyby odstraněny? *

- ○ Druhý pracovní den
- ○ Do jednoho týdne
- ○ Nebyly opraveny

Jak moc vám aplikace usnadnila práci? *

	1	2	3	4	5	
Práci mi neusnadnila	○	○	○	○	○	Velmi mi usnadnila práci

Doplňující informace

Vaše odpověď

Kapitola 6

Nasazení

Kapitola se věnuje popisu nasazení aplikace.

6.1 Technologie

Zde jsou uvedeny všechny programy, které je potřeba nainstalovat na server před spuštěním aplikace

- Apache Tomcat 8.5 a vyšší
- Postgress databáze 5.0 a vyšší (MySQL databáze v nejnovější verzi vyžaduje připojení přes SSL, což znesnadňuje konfiguraci.)
- Java Runtime Environment 1.8 a vyšší

6.2 Postup

6.2.1 Konfigurační soubor

Aplikace se konfiguruje pomocí properties souboru. Aplikaci je třeba sdělit, kde se tento soubor nachází pomocí systémové proměnné REAGENTS_CONF. Ukázka konfigurace pro Postgress a logování na Windows.

```
db.username=postgres
db.password=admin
db.uri=jdbc:postgresql://localhost:5432/reagents
db.dialect=org.hibernate.dialect.PostgreSQL94Dialect
db.driver=org.postgresql.Driver
log.file="C:\\ReagentsApp\\" # Start logging to file
    C:\ReagentsApp\reagents.log
log.level=WARN # Log from level warning
```

Je třeba zajistit, aby na databázovém serveru byla dostupná zadaná databáze. Tabulky databáze server založí sám při spuštění.

6.2.2 WAR soubor a Tomcat

Tomcat 8.5 lze stáhnout z `https://tomcat.apache.org/download-80.cgi`. Pokud nebudeme chtít používat HTTPS připojení, stačí nám základní nastavení serveru a nastavení proměnné "JRE_HOME", která musí obsahovat cestu ke složce s JRE\bin. Přiložený WAR soubor jednoduše zkopírujeme do složky „tomcat\webapp“. Spustíme Tomcat pomocí příkazu „catalina.bat“ respektive „catalina.sh -run“. Při správném nastavení aplikace bude aplikace dostupná na adrese `http://localhost:8080/`.

6.2.3 Konfigurace zabezpečeného spojení se serverem

Pro aktivování protokolu HTTPS potřebujeme platný certifikát. Vytvoříme soubor „keystore.jks“ například pomocí programu KeyExplorer a importujeme do něj certifikát. Následně stačí změnit konfigurační soubor Tomcatu, který se nachází v „tomcat\conf\server.xml“ do elementu <Service name="Catalina">

```
<Connector
   port="443"
   protocol="org.apache.coyote.http11.Http11NioProtocol"
   SSLEnabled="true"
   maxThreads="10"
   scheme="https"
   secure="true"
   SSLProtocol="TLSv1+TLSv1.1+TLSv1.2"
   SSLVerifyClient = "optional"
   SSLVerifyDepth = "2"
   URIEncoding="UTF-8"
   keystoreFile="/mnt/c/Tomcat/conf/cskeystore.jks"
   keystorePass="changeit"
   keystoreType="JKS"
   keyAlias="My valid certificate"
   keyPass="Key to private key"
   disableUploadTimeout="true"
   enableLookups="false"/>
```

Závěr

Shrnutí práce

Aplikace byla revidována a obohacena o nové vlastnosti a funkcionalitu. Následně byla aplikace nasazena na testovací server, který jsem zajistil u společnosti OVH, jelikož IT oddělení v ÚHKT nereaguje ani na maily, ani na urgování ze strany paní Ing. Vrané. Protože na otestování aplikace nestačil jeden měsíc, zůstává nadále aplikace v testovacím režimu. Na 22. 5. 2017 je naplánované školení budoucích uživatelů v ÚHKT a účastníci budou požádáni o vyplnění dotazníku kvality aplikace, jejíž výsledky budou prezentovány při obhajobě práce.

Přínos práce

Předpokládám, že by aplikace měla být po konečném úspěšném otestování a zahájení plného provozu přínosem a významně usnadnit práci zaměstnancům ÚKHT se správou reagencií a kitů.

Možná rozšíření

V poslední řadě jsem zjistil, že problematika okolo kitů a reagencií je v této aplikaci silně zjednodušená, a proto předpokládám, že bude možné navázat jejím rozšířením v případné magisterské práci.

Literatura

[1] Ústav hematologie a krevní transfuze: O ÚHKT. [online], [cit 12-04-2016]. Dostupné z: http://www.uhkt.cz/ustav

[2] Jílek, P.: *Základy imunologie*. Ewopharma s.r.o., 21-29 s.

[3] Ústav hematologie a krevní transfuze: Skupina HLA genotypizace. [online], [cit 24-04-2016]. Dostupné z: http://www.uhkt.cz/veda-vyzkum/usek-pro-vedu-a-vyzkum/oddeleni-hla/skupina-hla-genotypizace

[4] FOWLER, M.: *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Třetí vydání, 2003, ISBN 0-321-19368-7.

[5] Rouse, M.: DEFINITION Tomcat. [online], [cit 19-04-2016]. Dostupné z: http://searchsoa.techtarget.com/definition/Tomcat

[6] Çağatay Çivici: *PrimeFaces User Guide 5.3*. Primetek, 2016. Dostupné z: http://www.primefaces.org/docs/guide/primefaces_user_guide_5_3.pdf

[7] FOWLER, M.: *Patterns of Enterprise Application Architecture*. Addison Wesley, 2002.

[8] community, H.: Hibernate User Guide. [online], [cit 20-04-2016]. Dostupné z: http://docs.jboss.org/hibernate/orm/5.1/userguide/html_single/Hibernate_User_Guide.html

[9] Pavlíčková, J.; Pavlíček, L.; škola ekonomická v Praze. Fakulta informatiky a statistiky, V.: *Vývoj klient/server aplikací v Javě*. Oeconomica, 2004, ISBN 9788024507910. Dostupné z: https://books.google.cz/books?id=MyVxAAAACAAJ

[10] https://www.techopedia.com/: Definition - What does Build Tool mean? [online], [cit 22-04-2017]. Dostupné z: https://www.techopedia.com/definition/16359/build-tool

[11] TestovaniSoftwaru.cz: Automatizované testování. [online], [cit 20-04-2016]. Dostupné z: http://testovanisoftwaru.cz/automatizovane-testovani/

[12] Point, T.: Software Testing - Methods. [online], [cit 19-04-2016]. Dostupné z: http://www.tutorialspoint.com/software_testing/software_testing_methods.htm

[13] tutorialspoint.com: Web Application Testing. [online], [cit 24-04-2016]. Dostupné z: http://www.tutorialspoint.com/software_testing_dictionary/web_application_testing.htm

[14] Voráček, K.: *Výkonnostní testování webov.ap.* GRADA Publishing, a.s., 2004, ISBN 80-247-0822-1.

[15] http://seleniumhq.org/: SeleniumHQ Dokumentace. [online], [cit 22-04-2017]. Dostupné z: http://seleniumhq.org/docs/02_selenium_ide.html#selenium-commands-selenese

[16] https://addons.mozilla.org/: Selenium IDE doplněk do Mozzila Firefox. [online], [cit 22-04-2017]. Dostupné z: https://addons.mozilla.org/cs/firefox/addon/selenium-ide/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

DB Database

GUI Graphical user interface

HLA Human Leukocyte Antigen

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDE Integrated Development Environment

IT Information technology

JDBC Java Database Connectivity

JPA Java Persistence API

JRE Java Runtime Envirovement

JSF Java Server Faces

MVC Model View Cotroller

ORM Object-relational mapping

POJO Plain old Java object

SQL Structured Query Language

UML Unified Modeling Language

ÚHKT Ústav hematologie a krevní transfuze

WAR Web application ARchive

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `reagentsApp.war` ..... Zkompilovaný program ve formátu WAR
- `userManual.pdf` ..... Uživatelská Příručka
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF