UNIVERZITA PARDUBICE
DOPRAVNÍ FAKULTA JANA PERNERA

SOFTWAROVÁ PODPORA PRO TECHNICKOU PŘÍPRAVU PROJEKTU V ELEKTRIZACI ŽELEZNIC PRAHA A.S.

BC. LUKÁŠ HRON

DIPLOMOVÁ PRÁCE
2008

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Katedra informatiky v dopravě
Akademický rok: **2007/2008**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Lukáš HRON**

Studijní program: **N3708 Dopravní inženýrství a spoje**

Studijní obor: **Aplikovaná informatika v dopravě**

Název tématu: **Softwarová podpora pro technickou přípravu projektu v Elektrizaci železnic Praha a. s.**

Zásady pro vypracování:

Diplomová práce bude sloužit pro podporu zpracování technické dokumentace stavby v elektronické podobě. Bude respektovat aktuální metodiku zadávání číselníků používanou v Elektrizaci Železnic Praha a. s. a bude obsahovat kontrolní mechanismy pro ověření konzistencí datových zdrojů sestavení a součástí. Práce má následující cíle:

a. analýza objektového modelu technické přípravy projektu,

b. navržení a analýza komunikace mezi objekty při rozpisu projektu,

c. zpracování části projektu v prostředí JAVA nad databází MySQL.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **KISZKA, B.** *1001 tipů a triků pro programování v jazyce Java.* **Brno : Computer press, 2003. ISBN 80-7226-989-5.**
2. **HEROUT, P.** *Učebnice jazyka Java.* **České Budějovice : Kopp, 2001. ISBN 80-7232-323-4.**
3. **Materiály k technické dokumentaci staveb Elektrizace železnic Praha a. s.**

Vedoucí diplomové práce: **Ing. Miroslav Klobasa**
Elektrizace železnic, a.s.

Konzultant diplomové práce: **Mgr. Jan Němec**
Elektrizace železnic, a.s.

Datum zadání diplomové práce: **4. prosince 2007**

Termín odevzdání diplomové práce: **4. června 2008**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 30. listopadu 2007

Souhrn

Tato práce si klade za cíl vysvětlit čtenářům problematiku postupu při rozpisu stavebních objektů v Elektrizaci železnic Praha a.s., navrhnout objektový model a vysvětlit související pojmy. V práci je vytvořeno uživatelské rozhraní pro elektronickou správu číselníků, které je jednou ze samostatných částí budoucího systému TPP.

Klíčová slova

technická příprava projektu, java, databáze, sestavení, katalogový list, kusovník, součást, číselník, rozpis, montáž

Title

Software support for Technical Project Preparation department in Elektrizace železnic Praha a.s.

Abstract

This work explains basic prosesses on Technical Project Preparation department in Elektrizace železnic Praha a.s. and related basic terms.

Keywords

technical project preparation, java, database, composition, catalog list, item list, component, code list, building, engineering

Poděkování:

Na tomto místě bych rád poděkoval Ing. Miroslavu Klobasovi za odborné rady a vedení při zpracování této diplomové práce.

Obsah

Seznam ilustrací

Seznam tabulek

Seznam zkratek

JSP	JavaServer Pages
TPP	Technická příprava projektu
TP	Technická příprava
SW	Software
UML	Unified Modeling Language
EŽ	Elektrizace železnic Praha a.s.
ČSD	Československé státní dráhy
CDDL	Common Development and Distribution Licence
GPL	General Public Licence
JDBC	Java Database Connectivity
XML	Extensible Markup Language
XSL	eXtensible Stylesheet Language
JSON	JavaScript Object Notation
AJAX	Asynchronous JavaScript and XML
XHTML	Extensible Hypertext Markup Language
CSS	Cascading Style Sheets
SOAP	Simple Object Access Protocol
WDSL	Web Services Description Language

1 Úvod

Společnost Elektrizace železnic byla založena 1.ledna 1954, jako státní podnik, jehož hlavní náplní byla elektrizace tratí v síti tehdejších ČSD. Podnik vyvíjí a realizuje nové systémy trakčního vedení nejen pro dráhy, ale i pro městskou hromadnou dopravu, tj. pro tramvaje a trolejbusy.

Trakční vedení, nebo také jinak trolejové vedení, slouží k napájení kolejových vozidel nebo trolejbusů elektrickou energií. Ve většině případů je tvořeno jedním nebo dvěma měděnými vodiči, které jsou zavěšeny ve stanovené poloze nad jízdní dráhou. Kontakt vozidla s trolejovým vedením se udržuje prostřednictvím sběrače.

Hlavním cílem této práce je vytvořit softwarovou podporu pro technickou přípravu projektu (dále TPP). Vstupy do aplikace budou pro jednotlivé stavby zadávat pracovníci oddělení technické přípravy na základě projektů a dokumentace vypracované projektanty. Výstupy pak budou sloužit jako podklad pro samotnou výrobu i montáž jednotlivých dílů a sestavení na jednotlivých zakázkách.

2 Projektové zadání

2.1 Předmět zadání

Vytvoření elektronického systému pro podporu technické přípravy projektu TPP v Elektrizaci železnic Praha a.s. Program bude nasazený jako samostatná součást informačního systému společnosti.

Činnost TP spočívá ve zpracování a sdílení technické dokumentace k jednotlivým realizovaným stavbám společností Elektrizace železnic Praha a.s. Technickou dokumentaci vytváří projektanti, kteří ji předávají dále nejčastěji v elektronické podobě (v aplikaci Microsoft Excel), částečně pak i v tištěné podobě. Tato dokumentace slouží následně jako podklad k výrobě a montáži jednotlivých součástí pro konkrétní projektovanou stavbu.

Z dokumentace je nutné získat informace o použitých sestaveních, což jsou montované nebo složené celky. Případně pak i přímo o jednotlivých součástech a parametrech jejich použití. Sestavení jsou navržena obecně, tak aby vyhovovala použití daného kompletu v různých variantách. Obsahují neučené součásti, které se dopočítávají/dourčují podle dalších známých parametrů. Jsou to například izolátory, které se použijí podle vedeného napětí, nebo délka lišty upevnění, která je určena na základě šířky stožáru v místě uchycení.

Aplikace bude respektovat aktuální metodiku zadávání číselníku používanou v Elektrizaci železnic Praha a.s., jejíž součástí budou kontrolní mechanismy pro ověření konzistencí datových zdrojů sestavení (včetně kusovníků) a součástí. Vytvořený model by měl být navržen tak, aby velká část těchto neurčených součástí mohla být doplněna ze známých parametrů. Správa pravidel pro určování součástí by měla být co nejjednodušší.

Hlavním úkolem vytvoření softwarové podpory TPP je sjednotit tvorbu a sdílení těchto dokumentů v rámci společnosti Elektrizace železnic Praha a.s. Na základě těchto vstupních informací budou aplikací doplněny neurčené parametry a výsledné výstupy budou použity jak při výrobě jednotlivých dílů a sestavení, tak také při samotné montáži na stavbě.

2.2 Rozsah projektu

Práce je rozdělena do třech hlavních bodů. První je zaměřený na analýzu objektového modelu TPP. To znamená vysvětlení principu projektování staveb v Elektrizaci železnic Praha a.s. Seznámení se základními termíny, které s touto problematikou souvisí a samotný návrh základních datových struktur pro správu číselníků, jejich popis a funkce.

Obsahem druhé části je analýza a navržení komunikace mezi objekty při rozpisu nového projektu. To znamená, jakým způsobem si jednotlivé objekty navržené v předchozí kapitole budou mezi sebou předávat informace. S tím také souvisí návrh kontrolních mechanismů konzistence dat při doplňování neurčených parametrů, podle kterých se budou posílané informace řídit.

Poslední část se zabývá zpracováním části projektu v prostředí JAVA a databázi MySQL. Je to praktická ukázka návrhu jednotlivých objektu a jejich vzájemná komunikace prostřednictvím kontrolních mechanismů. V tomto dokumentu jsou vysvětleny základní funkce programu a jeho ovládání.

2.3 Technologické požadavky:

Program TPP bude vytvořený jako dynamická webová aplikace. Realizovaná bude pomocí technologie Java Server Pages (JSP), ve vývojovém prostředí Netbeans 6.x. Část programu uchovávající data bude nasazena na databázovém serveru MySQL 5.x. Běh aplikace zajistí webový server Apache Tomcat, který podporuje technologii JSP. Jako znaková sada bude v aplikaci i databázi použita utf-8.

2.4 Specifikace požadavků na aplikaci TPP

Aplikace TPP bude splňovat následující funkční požadavky:

- Uživatel musí být pro práci s aplikací do systému přihlášen.
- Při změně záznamů v databázi se musí zaznamenat kdo je modifikoval a kdy.
- Vkládání, editace, aktualizace a mazání záznamů v číselnících sestavení.
- Vkládání, editace, aktualizace a mazání záznamů v číselnících součástí.
- Vkládání, editace, aktualizace a mazání záznamů v číselnících katalogového listu.
- Vkládání, editace, aktualizace a mazání záznamů v číselnících kusovníku.
- Prohlížení číselníků sestavení, součástí a katalogových listů.
- Založení, editace a smazání stavby.
- Založení, editace a smazání objektu.
- Založení, editace, smazání a doplnění dodatku.
- Pro uživatele bude vytvořena nápověda.
- Zálohování databáze.
- Komunikace uživatelů se správcem systému přes email.

2.5 Použité technologie pro vývoj TPP

2.5.1 Netbeans

Open Source projekt založený a sponzorovaný firmou Sun Microsystems s velmi rozsáhlou uživatelskou základnou a rostoucí komunitou vývojářů byl založen v druhé polovině roku 2000.

V dnešní době existují dva produkty, kterými se projekt zabývá. Prvním z nich je vývojové prostředí NetBeans (NetBeans IDE) a ve druhém případě se jedná o vývojovou platformu NetBeans (The NetBeans Platform).

Vývojové prostředí NetBeans IDE je nástroj, který nabízí programátorům pohodlnou práci při psaní, překladu a ladění aplikací. Vývojářům nabízí širokou škálu modulů, kterými lze, dle potřeby prostředí jednoduše rozšířit. Netbeans IDE je bezplatně šířeným produktem.

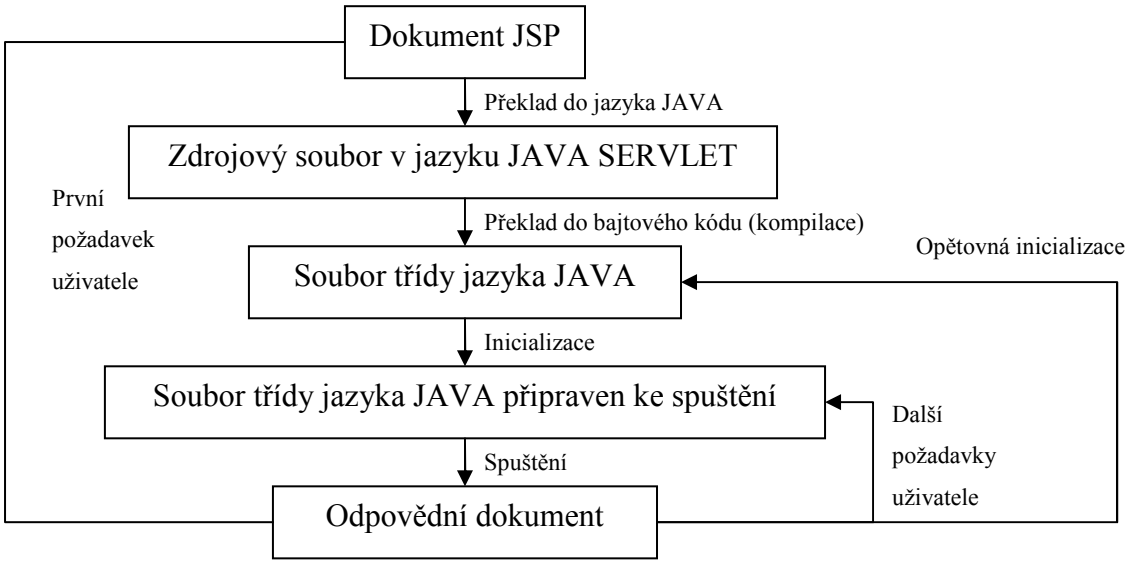
Vývojová platforma NetBeans Platform je modulární a rozšiřitelný základ pro použití při vytváření rozsáhlých aplikací. Nezávislí dodavatelé softwaru nabízejí moduly pro integraci do této platformy. Tyto moduly slouží pro vývoj jejich vlastních nástrojů a řešení.

Oba produkty jsou vyvíjeny pod licencí Open Source a je možné je bezplatně používat v komerčním i nekomerčním prostředí. Zdrojový kód je dostupný pod licencí Common Development and Distribution Licence (CDDL).

2.5.2 JSP (JavaServer Pages)

JSP je jednou z několika technologií, které lze použít k tvorbě dynamických webových aplikací. V JSP dokumentech je HTML kód spojován s programovacím jazykem JAVA. Ten pomáhá počítači rozhodnout o dalším postupu. Jedná se o serverovou technologii. To znamená že data dokumentu JSP, přístup k databázi a další součásti vyžadující zásah ze strany serveru může sdílet více uživatelů. JSP má vůči ostatním technologiím velkou výhodu. Dokument JSP není při každém požadavku interpretován úplně od začátku. Při prvním použití je dokument JSP převeden do jazyku JAVA na tzv. SERVLET a ten předkompilován (přeložen) do pseudokódu. Ten umožňuje rychlé a efektivní spuštění kódu. JSP dokumenty lze přenést na jinou platformu i po přeložení do bytového kódu.

V následujících odstavcích a na Obr. 1 je znázorněný postup odesílání JSP dokumentu do klientského počítače. Více informací o této technologii a způsobu vytváření JSP stránek se čtenář může dozvědět v knize [1].



Obr. 1 Postup odesílání dokumentu JSP, zdroj [1]

Princip zpracování JSP dokumentu je následující. Napíšeme JSP dokument a uložíme jej například pod názvem novy_dokument.jsp. Poté tento dokument umístíme na počítač, na kterém běží webový server. Ten musí mít spuštěný i software, který umí zpracovávat JSP dokumenty. Tento software se nazývá kontejner JSP. V našem případě to je webový server Apache Tomcat, kde kontejner JSP je již jeho součástí.

Pokud uživatel navštíví naší webovou aplikaci a zadá první požadavek na naši stránku, přeloží kontejner JSP tento dokument na straně serveru a vytvoří z něj program v jazyce JAVA. Název tohoto programu bude _novy_dokumet.java. V původním dokumentu JSP se smíchaly HTML značky a příkazy v jazyce JAVA. V přeloženém programu se tyto značky, stejně jako další texty stanou součástí příkazů JAVA. Úkolem je sdělit serveru, aby uživateli zaslal daný text nebo značky HTML. Takový program, tedy program spuštěný na serveru a odpovídající na příchozí požadavky uživatelů, nazýváme servlet. Ten je uložený do zvláštního adresáře, který je rezervován pro všechny soubory vznikající v důsledku překladu dokumentu JSP.

Každý program v jazyce JAVA musí být před spuštěním přeložen do bytového kódu. Z pohledu vývojáře to je další z transformačních kroků. Překladač přečte soubor _novy_dokument.java a vytvoří nový soubor pojmenovaný _novy_dokument.class. Ten je automaticky umístěn do adresáře pro soubory tříd vzniklých přeložením dokumentů JSP. Soubory .class jsou binární a nelze je editovat a prohlížet v běžných programech. Pokud

uživatel navštíví náš webový server, najde server v těchto souborech tříd všechny příkazy potřebné ke správnému chodu stránek. Příkazy obsahují pravidla tvorby odpovědních dokumentů a pravidla jejich odesílání do klientských prohlížečů. Kontejner načte soubor třídy a na základě zjištěné definice vytvoří nový objekt. Mimo jiné inicializuje všechny proměnné, které jsou součástí nového objektu. Jedná se o tzv. inicializaci dokumentu JSP.

Kontejner vytvoří odpovědní dokument, který odešle zpět k uživateli. Výsledný odpovědní dokument neobsahuje žádné zbytky kódu JSP.

Když později o stejný dokument požádá stejný nebo nový uživatel, vytvoří kontejner nový odpovědní dokument. To se děje stále znovu a znovu dokud kontejner příslušný soubor tříd neodstraní. Jestliže potřebuje kontejner uvolnit část paměti a nikdo dokument JSP nevyžaduje, může tento soubor z paměti vymazat. Pokud by se správce serveru rozhodl kontejner úplně vypnout, vymaže tím automaticky všechny soubory tříd z operační paměti. Jestliže poté navštíví server další uživatel, kontejner načte soubor třídy znovu do operační paměti a inicializuje všechny proměnné. Vytvoří nový odpovědní dokument a odešle jej do klientského počítače a bude čekat na další požadavky.

2.5.3 Apache Tomcat

Jakarta, nebo také Apache Jakarta je projektem organizace Apache Software Foundation. Hlavní portál projektu najdeme na adrese http://jakarta.apache.org. Cílem projektu Jakarta je vytvářet kvalitní open-source řešení serverových aplikací pro Java platformu. V současné době se jedná o jedno z nejkomplexnějších volně šiřitelných řešení pro serverové aplikace.

Jedním ze stěžejních podprojektů projektu Jakarta je projekt Jakarta Tomcat, také nazývaný jako Apache Tomcat. Tomcat je oficiální referenční implementace technologií Java Servlet a Java Server Pages (JSP), obojí vyvíjené pod záštitou společnosti SUN. Ve své stabilní řadě 6.0 Tomcat nabízí podporu pro Java Servlet verze 2.5 a JSP verze 2.1.

2.5.4 MySQL

Databázový systém, který vytvořila švédská firma MySQL AB. Tento software je k dispozici jak pod bezplatnou licencí GPL (General Public License), tak pod komerční placenou licencí.

MySQL je multiplatformní databáze, která nalezla své hlavní uplatnění jako jedna ze softwarových součástí webového serveru. Lze ji instalovat na Linux, MS Windows, tak i na

další operační systémy. Komunikace s tímto databázovým serverem probíhá pomocí jazyka SQL. Zpočátku byl server optimalizovaný především na rychlost i za cenu absence některých funkcí. V současnosti je již doplněna o pokročilé databázové funkce jako vnořené dotazy, pohledy, procedury, funkce atd.

2.5.5 UML (Unified Modeling Language)

Grafický jazyk používaný v softwarovém inženýrství pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. UML podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. UML neobsahuje způsob, jak se má používat, ani neobsahuje metodiku(y), jak analyzovat, specifikovat či navrhovat programové systémy. Standard UML definuje standardizační skupina Object Management Group (OMG).

Způsob použití UML lze rozdělit do několika kategorií. Prvním je kreslení konceptů. Při tomto použití je UML podpůrným nástrojem pro komunikaci mezi vývojáři a pro zaznamenání myšlenek a návrhů. Do diagramů se kreslí pouze věci podstatné pro grafické vyjádření návrhu, části návrhu před tím, než se začne programovat. Důležitá je srozumitelnost, rychlost nakreslení a snadnost změny či navržení alternativ řešení.

Cílem kreslení detailních návrhů je zaznamenat kompletní návrh či kompletní realizaci. Při kreslení návrhu by měl analytik obsáhnout všechny prvky tak, aby programátor byl schopen vytvořit program bez velkého přemýšlení nad věcnou oblastí. To znamená, že by pro programátora neměla vzniknout potřeba konzultace s uživatelem. Případně by měla být jen nezbytně nutná. Při kreslení detailních návrhů se obvykle používají specializované programy (CASE), které jsou schopny sdílet informace mezi jednotlivými modely a kontrolovat konzistenci návrhu.

UML lze použít jako programovací jazyk. Vývojář nakreslí UML diagramy, ze kterých se vygeneruje přímo spustitelný kód. Toto vyžaduje specializované nástroje a velmi přesné vyjadřování v UML diagramech.

3 Analýza a návrh objektového modelu TPP

První část této kapitoly se zabývá systémem jako celkem. Jsou zde vysvětleny základní termíny související s problematikou technické přípravy projektu. Pomocí diagramu aktivit je zobrazený reálný postup rozpisu stavebního objektu. Pomocí use case diagramu jsou naznačeny role a činnosti všech uživatelů, kteří se systémem pracují. Poslední část kapitoly se zabývá analýzou a návrhem základních tříd a metod pro správu číselníků.

3.1 Vysvětlení základních pojmů

Stavba

Pokud se v souvislosti s TPP hovoří o stavbě většinou se má na mysli nějaký větší úsek železniční tratě, nádraží nebo také celý koridor. Každá stavba je v projektové dokumentaci uvedena pod vlastním identifikačním číslem. Rozsah stavby určuje investor, zadavatel stavby (zakázky).

Stavební objekt

Každá stavba se skládá z několika samostatných stavebních objektů. Je to například železniční stanice nebo širá trať, případně jejich části. Stavební objekty jsou ucelené celky, které určil již investor a které mají stanovený harmonogram jednotlivých stavebních prací.

Dodatek

Stavební objekty se dále dělí na tzv. dodatky – odpovídají úsekům, které se dělají jako ucelená část omezená např. výlukou. Bývá to např. kolej na nádraží, úsek několika stožárů a bran na širé trati apod. Někdy se používá termín „kalkulace“. Dodatky jsou z části určeny rozpisem výluk od investora, částečně si je určuje stavbyvedoucí podle aktuálního stavu prací na stavbě a operativního plánu.

Součást

Je samostatný díl, který se nedá dále dělit. V číselníku je každá součást identifikována unikátním označením. Buďto číselným, nebo kombinací čísel a písmen.

Katalogový list

Je výkres skupiny součástí, které se liší v rozměrech, parametru. Příkladem může být šroub S15/I – katalogový list/výkres, který popisuje šroub M20 dvou různých délek

50 a 60 mm. Každý katalogový list je identifikován unikátním označením tzv. číslem katalogového listu. Je to kombinace číslic a písmen. Součásti příslušné katalogovému listu mají zpravidla číslo součásti ve tvaru číslo katalogového listu/označení. Např. katalogový list S15/I – součásti S15/I/50 a S15/I/60.

Sestavení

Jsou montované nebo složené celky, skládající se z samostatných součástí. Každé sestavení je identifikovatelné unikátním číslem sestavení.

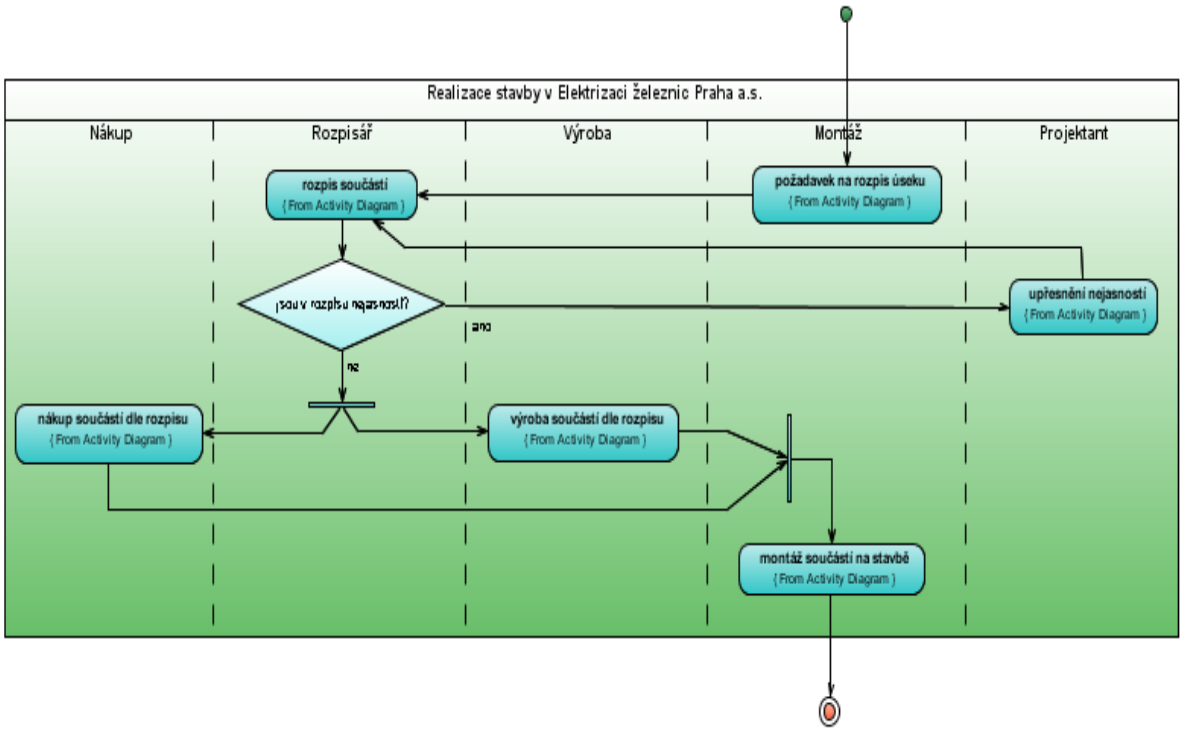
Kusovník

Každé sestavení se sestává alespoň z jedné části. Obsah sestavení je tzv. kusovník – seznam položek. Položkou kusovníku může být součást, katalogový list nebo podsestavení. Důležité je, že část položek není v sestavení přímo určena, ale je variabilní a závisí na konkrétním použití sestavení – délce konzoly, šířce stožáru v místě uchycení, stejnosměrném/střídavém napájení atd. Tyto neurčené součásti se částečně dopočítávají v programu TPP, částečně je dourčují pracovníci Technické Přípravy Projektu z dokumentace dodané projektantem, případně podle konkrétního zaměření ze stavby. Položka kusovníku je tedy navázána na odpovídající sestavení a dále na položku z množiny (součást, katalogový list, sestavení). Dalším důležitým parametrem je určení proměnná/neproměnná a množství, které je pro konkrétní sestavení potřeba.

3.2 Postup realizace stavby v Elektrizaci železnic Praha a.s.

Projektant vytvoří návrh celé stavby. Ta se však nerealizuje celá najednou, ale je rozdělena na tzv. stavební objekty. Před tím než bude montážní středisko realizovat konkrétní stavební objekt, musí zadat požadavek na rozpis součástí do oddělení technické přípravy. To vytvoří podrobný soupis všech součástí použitých na dané části stavby. Pokud by se při vytváření rozpisu vyskytly nějaké nejasnosti, jsou prokonzultovány s projektantem a rozpis je poté upraven nebo doplněn. V dalším kroku se rozepsané součásti rozdělí podle toho, zda si je firma bude vyrábět sama nebo je objedná u svých subdodavatelů na součásti vyráběné/nakupované. Toto rozdělení se provádí mimo program TPP v hlavním informačním systému, v modulu plánování MRP. Jsou-li všechny potřebné součásti nakoupené nebo vyrobené, může následovat samotná montáž součástí na stavebním objektu. Tento postup je zobrazen pomocí diagramu aktivit na Obr. 2.

Jako hlavní zdroj informací pro rozpis součásti stavebního objektu využívá oddělení technické přípravy číselníky, vycházející z technické dokumentace firmy. Existují tři druhy. Je to číselník pro katalogové listy, součásti a sestavení. Jejich význam je vysvětlený v podkapitole 3.1.



Obr. 2 Postup realizace stavby v Elektrizaci železnic Praha a.s.

3.3 Případy užití systému pro technickou přípravu

Uživatelé, kteří budou se systémem pracovat můžeme rozdělit do pěti kategorií. Vstupy do systému zadává vedoucí projektu, rozpisář a administrátor. Výstupy z aplikace poté pro svou činnost využívá výrobní, montážní středisko a nákup. Grafický návrh případů užití zobrazuje use case diagram, který je součástí přílohy A.

Systém z pohledu vedoucího projektu

Vedoucím projektu je osoba, která dohlíží na rozpis jednotlivých stavebních objektu konkrétní stavby. Zakládá do systému informace o stavbách, o stavebních objektech a dodatcích. O samotný rozpis dodatků se starají rozpisáři. Činnosti, které vedoucí projektu provádí v systému si vysvětlíme v následujícím odstavci.

Use case „Přihlášení do systému“ se zobrazí uživateli vždy při spuštění aplikace. Po zadání platného uživatelského jména a hesla má uživatel k dispozici další funkce. V případě

vedoucího projektu to je use case „Editovat stavbu“. Ten v sobě zahrnuje další činnosti, kterými jsou „Založit novou stavbu“, „Vymazat stavbu“ a „Upravit informace o stavbě“. Za určitých podmínek může být use case „Editovat stavbu“ rozšířen o „Editovat objekt“, který v sobě zahrnuje činnosti „Přidat nový objekt“, „Upravit objekt“ a „Odstranit objekt“. Use case „Editovat objekt“ je rozšířen o „Editovat dodatek“, který v sobě zahrnuje use case „Založit dodatek“, „Smazat dodatek“ a „Upravit dodatek“. Vedoucí projektu může nahlížet do vytvořených staveb (use case „Prohlížet stavbu“) a prohlížet číselníky (use case „Prohlížet číselníky“). Prohlížení staveb je rozšířeno o use case „Prohlížení objektů“ a „Prohlížení dodatků“.

Systém z pohledu rozpisáře

Dalším aktérem, který s aplikací pracuje je rozpisář z oddělení Technické Přípravy Projektu. Na základě vypracovaného návrhu stavby od projektanta zadává do systému požadavky a vytváří rozpis pro jednotlivé stavební objekty.

Po spuštění aplikace musí rozpisář pro práci v systému zadat své přihlašovací údaje (use case „Přihlášení do systému“). Asi nejdůležitější činností rozpisáře je use case „Doplnit dodatek“, který rozšiřuje use case „Upravit dodatek“. Další činností, kterou může provádět je use case „Editovat číselníky“, ta v sobě zahrnuje use case „Aktualizovat záznam“, „Vložit záznam“ a „Smazat záznam“. Stejně jako vedoucí projektu může i rozpisář nahlížet do vytvořených staveb (use case „Prohlížet stavbu“) a prohlížet číselníky (use case „Prohlížet číselníky“).

Systém z pohledu nákupu

Úkolem oddělení nákupu je podle vytvořeného rozpisu nakoupit součásti, které si firma sama nevyrábí, ale objednává je u svých subdodavatelů. Oddělení nákupu může v systému prohlížet realizované stavby a prohlížet číselníky. K tomu, aby mohl uživatel provádět tyto dvě činnosti musí být v systému přihlášený.

Systém z pohledu výroby

Podle rozpisu stavebního objektu toto středisko připraví a vyrobí konstrukce a součásti, které firma nenakupuje u svých subdodavatelů.

Pro výrobní středisko jsou na diagramu zobrazeny dva případy užití, je to use case „Prohlížet stavbu“ a use case „Prohlížet číselníky“. Samozřejmě pro nahlížení do systému musí být uživatel přihlášený.

Systém z pohledu montáže

Výstupy z aplikace jsou hlavním podkladem pro montážní středisko. Podle těchto podkladů se realizuje samotná montáž konstrukcí na stavebním objektu.

Pro montážní středisko platí podobné případy užití, jako pro výrobu.

Systém z pohledu administrátora

Hlavním úkolem administrátora je zajistit chod a dostupnost aplikace pro všechna střediska společnosti EŽ, která spolupracují na technické přípravě projektu. Administrátor řeší úkoly spojené se správou uživatelů a zajišťuje pravidelné zálohování databáze.

Tyto činnosti jsou na diagramu zobrazeny pomocí use case „Správa uživatelů“ a „Zálohování databáze“.

3.4 Návrh objektového modelu pro správu číselníků

Systém bude rozdělený do několika samostatných částí, které spolu budou komunikovat prostřednictvím zasílání zpráv. Následující objektový model je určený pro správu číselníků, která je jednou ze samostatných částí systému. Základní objekty si můžeme odvodit z výše uvedených informací. Třídy v tomto modelu můžeme rozdělit do dvou balíčků. V prvním budou třídy, které nám zajišťují přístup k databázi a správu uživatelů. Tento balíček pojmenujeme cz.elzel.system. Ve druhém, který je pojmenovaný cz.elzel.tpp_base, budou zařazeny třídy zajišťující práci s jednotlivými záznamy tabulek a třídy sloužící pro přístup k jednotlivým tabulkám databáze. V následující části této kapitoly si popíšeme jednotlivé třídy, jejich atributy a metody. Diagram tříd naleznete v příloze B a C.

Balíček cz.elzel.tpp_system

Zde jsou zařazeny následující třídy Dbf a MySqlDbf zajišťující připojení k databázovému serveru, konkrétně k MySQL verze 5. Hlavním úkolem třídy UserLogin je zajistit správu uživatelů pro aplikaci TPP. DAO a rozhraní IDAO zprostředkovávají přístup k tabulkám databáze TPP.

Balíček cz.elzel.tpp_base

Třídy v tomto balíčku bychom mohli rozdělit do dalších svou skupin. Třídy, které mají v názvu označení DAO poskytují metody pro práci nad jednotlivými tabulkami databáze TPP. Například třída KatListDAO poskytuje operace pro práci s tabulkou katlist. Seznam a popis těchto metod bude uvedený níže pro jednotlivé třídy zařazené v tomto balíčku. Druhou

skupinou jsou třídy bez tohoto označení. Jsou určené pro uchování jednotlivých záznamů tabulek v paměti počítače. To znamená, že jedna instance objektu KatList odpovídá jednomu záznamu tabulky katlist. Všechny třídy mají implementované metody označované jako getter a setter pro přístup k atributů.

Dbf.java

Abstraktní třída, která je předkem třídy MySqlDbf a zajišťuje spojení s databázovým serverem na obecné úrovni. Obsahuje základní metody pro navázání spojení s databází a metody pro volání SQL dotazů.

Atributy třídy Dbf:

- String url – adresa databázového serveru
- String dbfName – název databáze
- String userName – uživatelské jméno
- String password - heslo
- String driver – ovladač pro přístup k serveru
- String connectionType – typ databázového spojení
- boolean connected – je navázáno spojení se serverem
- ResutlSet resultSet – ukládá výsledek posledního SQL dotazu

Metody třídy Dbf:

- boolean connect() – připojení k databázovému serveru
- boolean disConnect() – odpojení od databázového serveru
- int executeInsert(String sql) – zavolání SQL dotazu INSERT
- int executeInsertStatement(PreparedStatement stm) – SQL dotaz INSERT pomocí předkompilovaného dotazu
- boolean executeQuery(String sql) – zavolání obecného SQL dotazu (nejčastěji SELECT)
- ResultSet executeQueryP(String sql) - zavolání obecného SQL dotazu (nejčastěji SELECT)
- boolean executeStatement(PreparedStatement stm) – Provede předkompilovaný SQL dotaz na daném spojení
- ResultSet executeStatementP(PrepareStatement) – Provede předkompilovaný SQL dotaz na daném spojení
- int executeUpdate(String sql) – Zavolání SQL dotazu typu UPDATE

- int executeUpdateStatement(PreparedStatement stm) – Provede UPDATE pomocí předkompilovaného dotazu

Třída obsahuje další metody pro práci s jednotlivými datovými typy viz. dokumentace JavaDoc k projektu TPP. Tato dokumentace je soušástí přílohy D.

MySqlDbf.java

Třída, ve které jsou definované základní metody a atributy pro přístup k MySQL databázovému serveru. Tato třídy je potomkem Dbf.java dědí metody a atributy této třídy. V této třídě jsou přetížené některé atributy a metody zajišťují registraci JDBC ovladače databáze v aplikaci.

Atributy třídy MySqlDbf:

- String connectionType – typ databázového spojení, nastavená hodnota je ("com.mysql.jdbc.Driver")
- String driver – ovladač pro přístup k serveru, nastavená hodnota je ("mysql")

Metody třídy MySqlDbf:

- int executeInsert(String sql) – metoda pro vložení záznamu do databáze
- int executeInsertStatement(PreparedStatement stm) – provede zápis do databáze pomocí překompilovaného SQL dotazu.

IDAO.java

Interface pro práci s databázovými záznamy. Toto rozhraní implementuje třída DAO a její potomci. To znamená třídy KatListDAO, VlastnostDAO, SestDAO, SoucDAO a TypmaDAO. Toto rozhraní nabízí výše zmíněným třídám jednotný přístup pro práci s tabulkami databáze.

Metody rozhraní IDAO:

- int addItem(Object item) – založení položky do databáze
- int copyItem(int id) – kopie položky s daným id do nové
- int copyItem(Object item) – kopie položky do nové
- boolean delItem(int id) – smazání položky s daným id
- boolean delItem(Object id) – smazání položky
- boolean updateItem(Object item) – aktualizace položky
- Object getItem(int id) – vrací položku s daným id
- Object[] getItemList() – vrací seznam položek

DAO.java

Abstraktní třída pro práci s databázovými tabulkami implementující rozhraní IDAO. Je předkem pro třídy, které poskytují funkce pro práci s jednotlivými tabulkami v databázi TPP.

Atributy třídy DAO:

- String className – název třídy, která je používána
- String tableName – název tabulky, se kterou třída pracuje
- Dbf dbf – databázové spojení
- String userName – uživatelské jméno (slouží také pro zaznamenávání poslední změny)

Metody třídy DAO:

- int addItem(Object item) – založení položky do databáze
- int copyItem(int id) – kopie položky s daným id do nové
- int copyItem(Object item) – kopie položky do nové
- boolean delItem(int id) – smazání položky s daným id
- boolean delItem(Object id) – smazání položky
- boolean updateItem(Object item) – aktualizace položky
- Object getItem(int id) – vrací položku s daným id
- Object[] getItemList() – vrací seznam položek

UserLogin.java

Třída poskytuje základní metody pro správu uživatelů. Obsahuje metody pro přidání, smazání, modifikaci uživatelského účtu, změnu práv přístupu k aplikaci TPP a verifikace uživatele.

Atributy třídy UserLogin:

- String className – název třídy, se kterou pracujeme
- Dbf dbf – databázové spojení
- String tableName – tabulka uživatelů
- String tableNameRights – tabulka s uživatelským oprávněním
- String userName – uživatelské jméno

Metody třídy UserLogin:

- int addUser(String userName, String password) – přidá nového uživatele
- String countHash(String password) – přepočet hashe ze zadaného hesla
- boolean delUser(int id) – vymaže uživatele s daným id

- boolean delUser(String userName) – vymaže uživatele s daným jménem
- String getComment(int id) – dej komentář k uživateli s daným id
- String getEmail(int id) – dej email uživatele s daným id
- String getFullName(int id) – dej celé jméno uživatele s daným id
- String getHash(int id) - dej hash uživatele s daným id
- int getId(String userName) – dej id uživatele
- int getStatus(int id) – dej status uživatele s daným id
- HashMap<String, String> getUserLogin(int id) – dej uživatelský login podle id uživatele
- String getUserName(int id) – dej uživatelské jméno podle id uživatele
- String getUserName() – dej uživatelské jméno připojeného uživatele
- int getUserPageRight(int id, String page) – vrací oprávnění uživatele k souboru/složce
- String getUserRight(int id, String name) – dej uživatelská práva
- int getUserRightId(int id, String name) – dej uživatelské oprávnění
- HashMap<String,String> getUserRights(int id) – uživatelská práva
- void setDbf(Dbf dbf) - nastav databázové spojení
- boolean setUserLogin(HashMap<String,String> userLogin) – nastav uživatelský login
- void setUserName(string userName) – nastav uživatelské jméno
- int setUserRight(int user_id, String name, String value) – nastav uživatelské oprávnění
- int verifyUser(String userName, String password) – ověřit uživatele

Vlastnost.java

Třída pro práci se záznamy v tabulce vlastnost. Ta je určena pro nastavení specifických vlastností určité součásti, sestavení, katalogového listu. Jednomu záznamu tabulky vlastnost odpovídá jedna instance objektu Vlastnost. Třída obsahuje stejné atributy jako tabulka vlastnost. Nejdůležitější metody této třídy zajišťují výpis atributů do formátu JSON a XML.

Atributy třídy Vlastnost:

- int id – identifikátor záznamu v databázi
- String type – typ vlastnosti (pro jaký typ záznamu je vlastnost určená)
- int owner_id – identifikátor vlastníka
- String name – jméno vlastnosti
- String value – hodnota vlastnosti

- Date lmdt – datum poslední modifikace vlastnosti
- String lmchid – modifikoval (uchovává login uživatele)
- Metody třídy Vlastnost:
- JSONObject to JSON() – výpis hodnot atributů objektu do formátu JSON
- String toXML() – výpis hodnot atributů objektu do formátu XML
- String toString() – výpis hodnot atributů do textového řetězce

VlastnostDAO.java

Třída implementuje operace pro přístup k tabulce vlastnost. Je potomkem třídy DAO. Obsahuje základní metody jako přídání, odebrání, mazání a aktualizace záznamu.

Atributy třídy VlastnostDAO:

- String className – jméno třídy se kterou pracujeme
- String tableName – jméno tabulky databáze, se kterou třída pracuje

Tyto atributy jsou statické. To znamená, že k nim můžeme přistupovat i v případě, kdy není vytvořena žádná instance objektu.

Metody třídy VlastnostDAO:

- int addItem(Object item) - přidej položku
- int addItem(Object item, boolean setDt) - přidej položku
- int addVlastnost(Vlastnost item) - založení nové vlastnosti do databáze, datum a autor poslední změny je nastaveno
- int addVlastnost(Vlastnost item, boolean setDt) - založení nové vlastnosti do databáze
- int copyItem(int id) - kopie položky
- int copyItem(Object item) - kopie položky
- int copyVlastnost(int id) - vytvoření kopie vlastnosti, datum a autor poslední změny budou nastaveny
- int copyVlastnost(int id, boolean setDt) - vytvoření kopie vlastnosti
- int copyVlastnost(Vlastnost item) - vytvoření kopie vlastnosti, datum a autor poslední změny bude nastaveno
- int copyVlastnost(Vlastnost item, boolean setDt) - vytvoření kopie vlastnosti
- boolean delItem(int id) - vymaže položku
- boolean delItem(Object item) - vymaž položku
- boolean delVlastnost(int id) - smazání vlastnosti z databáze
- boolean delVlastnost(Vlastnost item) - smazání vlastnosti z databáze

- boolean delVlastnostValue(String type, int owner_id, String value) - vymaže vlastnost
- Object getItem(int id) - dej položku
- Object[] getItemsList() - vrací seznam položek
- Vlastnost getVlastnost(int id) - načtení položky s daným id z databáze
- Vlastnost getVlastnost(String type, int owner_id, String name) - načtení položky daného typu pro vlastníka se jménem
- Vlastnost[] getVlastnostList() - vrací seznam vlastností jako array
- Vlastnost[] getVlastnostList(int first) - vrací seznam vlastností jako pole od zadané pozice
- Vlastnost[] getVlastnostList(int first, int amount) - vrací seznam součástí jako array od určité pozice počet záznamů
- Vlastnost[] getVlastnostList(String type) - vrací seznam vlastností určitého typu
- Vlastnost[] getVlastnostList(String type, int owner_id) - vrací seznam vlastností určitého typu pro určitého vlastníka
- JSONArray getVlastnostListJSON() - vrací seznam vlastností pro formát JSON
- JSONArray getVlastnostListJSON(int first) - vrací seznam vlastností pro formát JSON od pozice
- JSONArray getVlastnostListJSON(int first, int amount) - vrací seznam vlastností pro formát JSON od určité pozice a počet záznamů
- String getVlastnostListXML() - vrací seznam vlastností pro formát XML
- String getVlastnostListXML(int first) - vrací seznam vlastností pro formát XML od zadané pozice
- String getVlastnostListXML(int first, int amount) - vrací seznam vlastností pro formát XML od určité pozice a počet záznamů
- String getVlastnostValue(String type, int owner_id, String name) - vrací hodnotu parametru value na základě zadaných parametru
- String getVlastnostValue(Vlastnost item) - vrací parametr value vlastnosti, která je parametrem
- int setVlastnost(Vlastnost vl) - nastavení vlastnosti - kontroluje, jestli existuje, pokud ano modifikuje ji, jinak přidá novou vlastnost do databáze
- boolean setVlastnostValue(String type, int owner_id, String name, String value) - nastaví vlastnost dle parametru
- boolean setVlastnostValue(Vlastnost item) - nastaví vlastnost dle parametru

- boolean updateItem(Object item) - aktualizace položky
- boolean updateVlastnost(Vlastnost item) - aktualizace vlastnosti v databázi, bude aktualizováno datum a autor poslední změny
- boolean updateVlastnost(Vlastnost item, boolean setDt) - aktualizace vlastnosti v databázi, pokud bude setDt nastaveno na hodnotu false, nebude uložen autor poslední změny

Sest.java

Třída pro práci se záznamy sestavení tabulky sest. Jeden záznam odpovídá jedné instanci objektu Sest. Třída obsahuje stejné atributy jako tabulka sest a navíc má ještě implementován atribut kusovník, který je typu ArrayList<Typma>. Ten v sobě uchovává kusovník daného sestavení. Třída obsahuje metody pro výpis hodnot atributů do formátu JSON a XML.

Atributy třídy Sest:

- int id – identifikátor záznamu v databázi
- String cest – číslo sestavení
- String nsest – název sestavení
- String fs – soubor
- String skup - skupina
- String comment – komentář
- java.util.ArrayList<Typma> kusovnik – kusovník pro sestavení
- Date rgdt – datum registrace
- String rgchid – registroval (uchovává login uživatele)
- Date lmdt – datum poslední modifikace
- String lmchid - modifikoval (uchovává login uživatele)

Metody třídy Sest:

- JSONObject to JSON() – výpis hodnot atributů objektu do formátu JSON
- String toXML() – výpis hodnot atributů objektu do formátu XML
- String toString() – výpis hodnot atributů do textového řetězce

SestDAO.java

Třída implementuje operace pro přístup k tabulce sest. Je potomkem třídy DAO.java. Poskytuje základní metody pro přidání, odebrání a aktualizaci záznamu tabulky sest.

Atributy třídy SestDAO:

- String className – jméno třídy se kterou pracujeme
- String tableName – jméno tabulky databáze, se kterou třída pracuje

Oba tyto atributy jsou statické.

Metody třídy SestDAO:

- int addItem(Object item) - přidá novou položku do databáze
- int addSest(Sest item) - přidá nové sestavení do tabulky sest
- int addSest(Sest item, boolean setDt) - přidá nové sestavení do tabulky šest, pokud je setDt nastaveno na hodnotu false neuloží se informace, kdo a kdy provedl poslední změnu
- int copyItem(int id) - vytvoří kopii položky
- int copyItem(Object item) - vytvoří kopii položky
- int copyItem(Object sest, String novy_nsest) - vytvoří kopii položky
- int copySest(int id) - vytvoří kopii sestavení
- int copySest(int id, String novy_nsest) - vytvoří kopii sestavení
- int copySest(String nsest, String novy_nsest) - vytvoří kopii sestavení
- boolean delItem(int id) - smaže sestavení podle id
- boolean delItem(Object item) - smaže položku
- boolean delSest(int id) - smaže sestavení podle id
- boolean delSest(Sest item) - smaže sestavení
- boolean delVlastnost(int owner_id, String value) - smaže vlastnost s daným vlastníkem a hodnotou
- Object getItem(int id) - dej položku s id
- Object[] getItemsList() - vrací seznam položek jako pole
- Sest getSest(int id) - vrací součást podle id sestavení
- Sest getSest(String csest) - vrací sestavení podle čísla sestavení
- Sest[] getSestList() - vrací seznam sestavení jako array
- Sest[] getSestList(int first) - vrací seznam sestavení od dané pozice
- Sest[] getSestList(int first, int amount) - vrací seznam sestavení jako array, od zadané pozice daný počet záznamů
- Sest[] getSestList(String cse) - vrací seznam sestavení určitého čísla sestavení
- JSONArray getSestListJSON() - vrací seznam sestavení ve formátu pro JSON

- JSONArray getSestListJSON(int first) - vrací seznam sestavení ve formátu pro JSON od zadané pozice
- JSONArray getSestListJSON(int first, int amount) - vrací seznam sestavení ve formátu pro JSON, od zadané pozice daný počet záznamů
- String getSestListXML() - vrací seznam sestavení ve formátu pro XML
- String getSestListXML(int first) - vrací seznam sestavení ve formátu pro XML
- String getSestListXML(int first, int amount) - vrací seznam sestavení ve formátu pro XML, od zadané pozice daný počet záznamů
- Vlastnost getVlastnost(int id, String name) - dej vlastnost
- Vlastnost getVlastnost(Sest item, String name) - dej vlastnost
- Vlastnost getVlastnost(String csest, String name) - dej vlastnost
- Vlastnost[] getVlastnostList(int owner_id) - dej seznam vlastností
- int readVlastnost(Sest s) - čti vlastnosti daného sestavení
- void setVlastnost(int id, String name, String value) - nastav vlastnost pro sestavení s id
- void setVlastnost(Sest sest, String name, String value) - nastav vlastnost pro sestavení
- boolean updateItem(Object item) - aktualizace položky item
- boolean updateSest(Sest item) - aktualizace sestavení item
- boolean updateSest(Sest item, boolean setDt) - aktualizace sestavení item, pokud je setDt nastaveno na hodnotu false neuloží se informace o poslední změně
- boolean updateSestField(int id, String field, String value, boolean setDt) - aktualizace sestavení item, pokud je setDt nastaveno na hodnotu false neuloží se informace o poslední změně

Souc.java

Třída pro práci se záznamy součástí tabulky souc. Uchovává data z tabulky souc v paměti počítače. Atributy třídy jsou shodné s atributy tabulky souc. Jeden záznam tabulky souc odpovídá jedné instanci objektu typu Souc. Metody vypisují hodnoty atributů do formátu JSON a XML.

Atributy třídy Souc:

- int id - identifikátor záznamu v databázi
- String csouc – číslo součásti
- String nsouc – název součásti
- String ckat – číslo katalogu

- String jkpov – číslo položky (jednoznačné v IS)
- float price – cena
- String mj – měrná jednotka
- String from1 – rozsah od
- String to1 – rozsah do
- String from2 – rozsah od
- String to2 – rozsah do
- String comment – komentář
- Date rgdt – datum registrace
- String rgchid – registroval (uchovává login uživatele)
- Date lmdt – datum poslední modifikace
- String lmchid – modifikoval (uchovává login uživatele)

Metody třídy Souc:

- JSONObject to JSON() – výpis hodnot atributů objektu do formátu JSON
- String toXML() – výpis hodnot atributů objektu do formátu XML
- String toString() – výpis hodnot atributů do textového řetězce

SoucDAO.java

Třída implementuje operace pro přístup k tabulce souc. Je potomkem třídy DAO.java. Implementuje základní operace pro přidání, mazání a aktualizaci záznamů v této tabulce.

Atributy třídy SoucDAO:

- String className – jméno třídy se kterou pracujeme
- String tableName – jméno tabulky databáze, se kterou třída pracuje

Tyto atributy jsou statické, to znamená, že k nim můžeme přistupovat i v případě, kdy není vytvořena žádná instance tohoto objektu.

Metody třídy SoucDAO:

- int addItem(Object item) - přidá novou položku do databáze
- int addSouc(Souc item) - přidá novou součást do tabulky souc
- int addSouc(Souc item, boolean setDt) - přidá novou součást do tabulky souc
- int copyItem(int id) - vytvoří kopii položky
- int copyItem(Object item) - vytvoří kopii položky
- int copyItem(Object souc, String nove_csouc) - vytvoří kopii položky s novým číslem součásti

- int copySouc(int id) - vytvoří kopii součásti
- int copySouc(int id, String nove_csouc) - vytvoří kopii součásti s novým číslem součásti
- int copySouc(String csouc, String nove_csouc) - vytvoří kopii součásti
- boolean delItem(int id) - smaže položku podle id
- boolean delItem(Object item) - smaže položku
- boolean delSouc(int id) - smaže položku podle id
- boolean delSouc(Souc item) - smaže součást
- boolean delVlastnost(int owner_id, String value) - smaže vlastnost s daným vlastníkem a hodnotou
- Object getItem(int id) - dej položku s id
- Object[] getItemsList() - vrací seznam položek jako pole
- Souc getSouc(int id) - vrací součást podle id součásti
- Souc getSouc(String csouc) - vrací součást podle čísla součásti
- Souc[] getSoucKatList(String ckat) - vrací seznam součástí pro daný katalogový list
- Souc[] getSoucList() - vrací seznam součástí jako array
- Souc[] getSoucList(int first) - vrací seznam součástí jako array
- Souc[] getSoucList(int first, int amount) - vrací seznam součástí jako array, od dané pozice daný počet záznamů
- JSONArray getSoucListJSON() - vrací výpis součástí ve formátu JSON
- JSONArray getSoucListJSON(int first) - vrací výpis součástí ve formátu JSON
- JSONArray getSoucListJSON(int first, int amount) - vrací výpis součástí ve formátu JSON, od dané pozice daný počet záznamů
- String getSoucListXML() - vrací výpis součástí ve formátu XML
- String getSoucListXML(int first) - vrací výpis součástí ve formátu XML
- String getSoucListXML(int first, int amount) - vrací výpis součástí ve formátu XML, od dané pozice daný počet záznamů
- Vlastnost getVlastnost(int id, String name) - dej vlastnost
- Vlastnost getVlastnost(Souc item, String name) - dej vlastnost
- Vlastnost getVlastnost(String csouc, String name) - dej vlastnost
- Vlastnost[] getVlastnostList(int owner_id) - dej seznam vlastností
- void setVlastnost(int id, String name, String value) - nastav vlastnost pro součást s id

- void setVlastnost(Souc souc, String name, String value) - nastav vlastnost pro součást
- boolean updateCkat(String ckat, String newCkat) - nahrazení katalogového listu u součástí
- boolean updateItem(Object item) - aktualizace položky item
- boolean updateSouc(Souc item) - aktualizace součásti item
- boolean updateSouc(Souc item, boolean setDt) - aktualizace součásti item, pokud je setDt nastaveno na hodnotu false neuloží se informace o poslední změně
- boolean updateSoucField(int id, String field, String value, boolean setDt) - aktualizace součásti item

KatList.java

Třída pro práci se záznamy katalogových listů tabulky katlist. Obsahuje stejné atributy jako tabulka katlist. Metody této třídy vypisují hodnoty atributů do formátu JSON a XML. Atribut soucs je typu Souc[], jeto pole součástí daného katalogového listu.

Atributy třídy KatList:

- int id - identifikátor záznamu v databázi
- String nkat – název katalogu
- String ckat – číslo katalogu
- float defval- představitel ceny
- Souc[] soucs – pole součástí pro katalogový list
- String comment – komentář
- Date rgdt – datum registrace
- String rgchid – registroval (uchovává login uživatele)
- Date lmdt – datum poslední modifikace
- String lmchid – modifikoval (uchovává login uživatele)

Metody třídy KatList:

- JSONObject to JSON() – výpis hodnot atributů objektu do formátu JSON
- String toXML() – výpis hodnot atributů objektu do formátu XML
- String toString() – výpis hodnot atributů do textového řetězce

KatListDAO.java

Třída implementuje operace pro přístup k tabulce katlist. Je potomkem třídy DAO.java. Základní metody této třídy zajišťují přidání, odebírání a aktualizaci záznamů v této tabulce.

Atributy třídy KatListDAO:

- String className – jméno třídy se kterou pracujeme
- String tableName – jméno tabulky databáze, se kterou třída pracuje

Tyto atributy jsou statické, to znamená, že k nim můžeme přistupovat i v případě, kdy není vytvořena žádná instance tohoto objektu.

Metody třídy KatListDAO:

- int addItem(Object item) - přidej položku
- int addItem(Object item, boolean setDt) - přidej položku, možnost nastavení poslední změny
- int addKatList(KatList item) - založení nového katalogového listu do databáze, datum a autor poslední změny je nastaveno
- int addKatList(KatList item, boolean setDt) - založení nového katalogového listu do databáze, možnost nastavení poslední změny
- int copyItem(int id) - kopie položky
- int copyItem(int id, String newCkat) - kopie položky s daným id, vytvořená kopie má nastaveno nové katalogové číslo
- int copyItem(java.lang.Object item) - kopie položky
- int copyKatList(int id, String newCkat) - vytvoření kopie katalogového listu, datum a autor poslední změny bude nastaveno
- int copyKatList(int id, String newCkat, boolean setDt) - vytvoření kopie katalogového listu, možnost nastavení poslední změny
- int copyKatList(KatList item, String newCkat) - vytvoření kopie katalogového listu, datum a autor poslední změny bude nastaveno
- int copyKatList(KatList item, String newCkat, boolean setDt) - vytvoření kopie katalogového listu, možnost nastavení poslední změny
- int copyKatList(String ckat, String newCkat) - vytvoření kopie katalogového listu, datum a autor poslední změny bude nastaveno

- int copyKatList(String ckat, String newCkat, boolean setDt) - vytvoření kopie katalogového listu, možnost nastavení poslední změny
- boolean delItem(int id) - vymaž položku s id
- boolean delItem(Object item) - vymaž položku
- boolean delKatList(int id) - smazání katalogového listu z databáze
- boolean delKatList(KatList item) - smazání katalogového listu z databáze
- boolean delVlastnost(int owner_id, String value, VlastnostDAO vlDAO) - smaže vlastnost s daným vlastníkem a hodnotou
- Object getItem(int id) - dej položku
- Object[] getItemsList() - vrací seznam položek
- KatList getKatList(int id) - načtení položky s daným id z databáze
- KatList getKatList(String ckat) - načtení položky daného typu pro vlastníka se jménem
- KatList[] getKatListList() - vrací seznam katalogových listů jako array
- KatList[] getKatListList(int first) - vrací seznam katalogových listů jako array, od dané pozice
- KatList[] getKatListList(int first, int amount) - vrací seznam katalogových listů jako array od dané pozice daný počet záznamů
- JSONArray getKatListListJSON() - vrací seznam katalogových listů ve formátu JSON
- JSONArray getKatListListJSON(int first) - vrací seznam katalogových listů ve formátu JSON od dané pozice
- JSONArray getKatListListJSON(int first, int amount) - vrací seznam katalogových listů ve formátu JSON od dané pozice daný počet záznamů
- String getKatListListXML() - vrací seznam katalogových listů ve formátu XML
- String getKatListListXML(int first) - vrací seznam katalogových listů ve formátu XML
- String getKatListListXML(int first, int amount) - vrací seznam katalogových listů ve formátu XML od dané pozice daný počet záznamů
- Souc[] getSoucKatList(String ckat) - vrací seznam součástí pro daný katalogový list
- Vlastnost getVlastnost(int id, String name, VlastnostDAO vlDAO) - dej vlastnost
- Vlastnost getVlastnost(Souc item, String name, VlastnostDAO vlDAO) - dej vlastnost
- Vlastnost getVlastnost(String nkat, String name, VlastnostDAO vlDAO) - dej vlastnost

- Vlastnost[] getVlastnostList(int owner_id) - dej seznam vlastností
- void setVlastnost(int id, String name, String value, VlastnostDAO vlDAO) - nastav vlastnost pro součást s id
- void setVlastnost(KatList katlist, String name, String value, VlastnostDAO vlDAO) - nastav vlastnost pro součást
- boolean updateItem(Object item) - aktualizace položky item
- boolean updateKatList(KatList item) - aktualizace součásti item
- boolean updateKatList(KatList item, boolean setDt) - aktualizace katalogového listu item, možnost nastavení podlesní změny

Typma.java

Třída pro práci se záznamy kusovníků tabulky typma (kusovník). Jeden záznam tabulky typma odpovídá jedné instanci objektu Typma. Tyto záznamy poskytují bližší informace o přesném složení daného sestavení.

Atributy třídy Typma:

- int id - identifikátor záznamu v databázi
- int sest_id – odkaz na sestavení
- int object_id - odkaz na položku – sestavení, součást, katalogový list
- String prom – proměnná („P“ – proměnná / „“ - neproměnná)
- String typf – typ souboru („S“ - sestavení, „K“ - katalogový list, „“ - součást)
- float amount – množství použité položky v sestavení
- ITypma typma – položka použita v sestavení (může být součást, katalogový list, sestavení)
- Date rgdt – datum registrace
- String rgchid – registroval (uchovává login uživatele)
- Date lmdt – datum poslední modifikace
- String lmchid – modifikoval (uchovává login uživatele)

Metody třídy Typma:

- JSONObject to JSON() – výpis hodnot atributů objektu do formátu JSON
- String toXML() – výpis hodnot atributů objektu do formátu XML
- String toString() – výpis hodnot atributů do textového řetězce

TypmaDAO.java

Třída implementuje operace pro přístup k tabulce typma (kusovník). Je potomkem třídy DAO.java.

Atributy třídy TypmaDAO:

- String className – jméno třídy se kterou pracujeme
- String tableName – jméno tabulky databáze, se kterou třída pracuje

Tyto atributy jsou statické, to znamená, že k nim můžeme přistupovat i v případě, kdy není vytvořena žádná instance tohoto objektu.

Metody třídy TypmaDAO:

- int addItem(Object item) - přidá novou položku do databáze
- int addTypma(Typma item) - přidá nový kusovník do tabulky typma
- int addTypma(Typma item, boolean setDt) - přidá novou položku kusovníku do tabulky, možnost nastavení poslední změny
- int copyItem(int id) - vytvoří kopii položky
- int copyItem(Object item) - vytvoří kopii položky
- int copyTypma(int id) - vytvoří kopii kusovníku
- boolean delItem(int id) - smaže položku podle id
- boolean delItem(Object item) - smaže položku
- boolean delTypma(int id) - smaže položku podle id
- boolean delTypma(Typma item) - smaže kusovník
- boolean delVlastnost(int owner_id, String value) - smaže vlastnost s daným vlastníkem a hodnotou
- Object getItem(int id) - dej položku s id
- Object[] getItemsList() - vrací seznam položek jako pole
- String getSoucListXML(int first, int amount) - vrací seznam kusovníku ve formátu XML, od dané pozice daný počet záznamů
- Typma getTypma(int id) - vrací kusovník podle id kusovníku
- Typma[] getTypmaList() - vrací seznam kusovníků jako array
- Typma[] getTypmaList(int first) - vrací seznam kusovníků jako array
- Typma[] getTypmaList(int first, int amount) - vrací seznam kusovníků jako array
- JSONArray getTypmaListJSON() - vrací seznam kusovníku ve formátu JSON
- JSONArray getTypmaListJSON(int first) - vrací seznam kusovníku ve formátu JSON

- JSONArray getTypmaListJSON(int first, int amount) - vrací seznam kusovníku ve formátu JSON, od dané pozice daný počet záznamů
- String getTypmaListXML() - vrací seznam kusovníku ve formátu
- String getTypmaListXML(int first) - vrací seznam kusovníku ve formátu XML
- Typma[] getTypmaObjectList(int object_id, String typf) - vrací seznam kusovníků podle object_id a typf
- Typma[] getTypmaSestList(int sest_id) - vrací seznam kusovníků podle sest_id
- Vlastnost getVlastnost(int id, String name) - dej vlastnost
- Vlastnost getVlastnost(Typma item, String name) dej vlastnost
- Vlastnost[] getVlastnostList(int owner_id) - dej seznam vlastností
- void setVlastnost(int id, String name, String value) - nastav vlastnost pro kusovník s id
- void setVlastnost(Typma typma, String name, String value) - nastav vlastnost pro kusovník
- boolean updateItem(java.lang.Object item) - aktualizace položky item
- boolean updateTypma(Typma item) - aktualizace kusovníku item
- boolean updateTypma(Typma item, boolean setDt) - aktualizace kusovníku item, možnost nastavení poslední změny

ITypma.java

Implementuje rozhraní pro práci s kusovníkem. Toto rozhraní implementují třídy Sest, Souc, KatList.

Metody rozhraní ITypma:

- int getId() – dej id kusovníku
- String getCislo() – dej číslo kusovníku
- float getPriceItem() – dej cenu položky
- String getName() - dej jméno kusovníku
- void setCislo(String cislo) – nastav číslo kusovníku
- void setName(String name) – nastav jméno kusovníku
- void setId(int id) – nastav id kusovníku

Nyní si popíšeme závislosti při mazání a aktualizaci položek jednotlivých tabulek. Tyto vazby jsou uvedeny v následující tabulce. Všechny tyto závislosti jsou ošetřené v metodách jednotlivých tříd s označením DAO, určených pro práci s tabulkami.

Tab. 1 Tabulka závislostí mezi objekty při mazání a změně položek

záznam tabulky	mazání	změna názvu
sest	smaž kusovník z typma smaž vlastnost	změna v sest
katlist	smaž vlastnost v souc nastav ckat na null	změna v katlist
souc	smaž vlastnost	změna v souc
typma	smaž vlastnost	změna v typma
vlastnost		změna ve vlastnost

Pro součást platí následující pravidla. Pokud budeme z databáze vymazávat jakoukoliv položku z číselníku souc, musíme spolu s touto položkou také vymazat všechny její specifické vlastnosti. Aktualizace součásti nijak neovlivní ostatní objekty.

V případě mazaní katalogového listu, musíme u všech součástí, které do tohoto katalogového listu patří nastavit položku ckat na null. Poté vymažeme specifické vlastnosti, které byly pro katalogový list doplněny v tabulce vlastnost. Pokud budeme aktualizovat položku ckat katalogového listu, musíme změnu provést i u jednotlivých součástí, které sem patří.

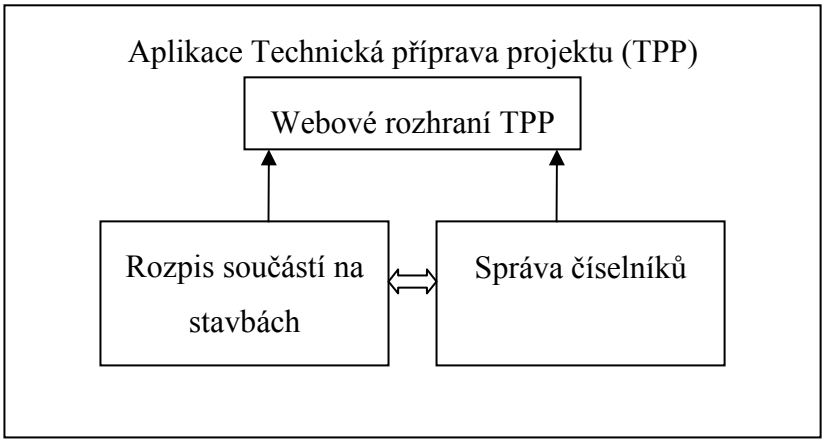
Změny v tabulce sestavení nám ovlivní záznamy v tabulce typma a vlastnost. Pokud vymažeme nějakou položku sestavení, spolu s ní odstraníme i kusovník z tabulky typma a nastavené vlastnosti z tabulky vlastnost. Při aktualizaci záznamu není potřeba provádět změny v jiné části databáze.

Při mazání položky z tabulky typma (kusovník) se odstraní i položky pro tento kusovník z tabulky vlastnost. Změna názvu kusovníku se provede pouze u položky tabulky typma. Jiné záznamy neovlivní.

Smazání nebo změna parametrů vlastnosti nemá vliv na záznamy v jiných tabulkách.

4 Analýza a návrh komunikace mezi objekty TPP

Systém je rozdělený do několika samostatných částí, které spolu komunikují prostřednictvím zasílání zpráv. Samostatné části systému jsou zobrazeny na následujícím obrázku.



Obr. 3 Samostatné moduly aplikace TPP

V této kapitole si vysvětlíme technologie, které jsou použity v systému pro komunikaci mezi jednotlivými objekty a částmi systému.

XML (eXtensible Markup Language)

Obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Používá se k vytváření konkrétních značkovacích jazyků pro různé účely. Nejčastěji se využívá pro výměnu dat mezi aplikacemi a pro publikování dokumentů. XML popisuje strukturu dokumentu z hlediska věcného obsahu jednotlivých částí. Nezabývá se vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se definuje připojeným stylem. Ten je vytvořený pomocí jazyka XSL (eXtensible Stylesheet Language). XSL je jazyk, kterým vytváříme pravidla pro transformaci jedné třídy XML dokumentů na jiný XML dokument.

V dnešní době, již není vhodné zasílat dokumenty ve formátech, které vyžadují pro své přečtení a editaci speciální software konkrétní firmy. Dnes se používána celá řada operačních a informačních systémů a není záruka, že každý uživatel má k dispozici příslušný software.

Z tohoto důvodu vznikla potřeba vytvořit jednoduchý otevřený formát, který není úzce svázán s nějakou platformou. Toto nabízí jazyk XML, který je založen na jednoduchém textu nejčastěji v kódování utf a je zpracovatelný (v případě potřeby) libovolným textovým editorem.

Specifikace XML konsorcia W3C je zdarma přístupná všem. Každý tak může bez problémů do svých aplikací implementovat podporu XML. Více informaci se lze dočíst viz. [12].

AJAX (Asynchronous JavaScript and XML)

Touto zkratkou se označuje technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.

AJAX je pojem označující použití několika technologií dohromady s určitým cílem. Pro prezentaci informací se využívá HTML (nebo XHTML) a CSS. Pro zobrazení a dynamické změny se využívá DOM a JavaScript. Objekt XMLHttpRequest se používá k asynchronní výměně dat s webovým serverem. Pro formátování zasílaných zpráv a odpovědí se nejčastěji využívá formát XML, ale je možné použít libovolný jiný formát například HTML, prostý text, JSON nebo EBML.

K hlavním výhodám patří odstranění nutnosti znovunačtení a překreslení celé stránky při každé operaci jako je tomu u klasického modelu WWW stránek. Například pokud si uživatel prohlíží na webu fotogalerii, která je vytvořena prostřednictvím AJAXu. Klikne-li na přechod na další fotografii, proběhne odeslání tohoto požadavku na pozadí a server vrátí zpět jen ty části stránky, které se změnily. Ty se uživateli aktualizují a překreslí. Tento způsob navozuje pocit mnohem větší plynulosti práce s webovými aplikacemi.

Další nespornou výhodou je, snížení zátěže na webové servery a síť obecně. Jelikož není potřeba při každém požadavku sestavit celý HTML dokument, je množství vyměňovaných dat výrazně nižší. To může mít příznivý vliv i na zátěž databázových serverů. Na druhou stranu, může AJAX při nevhodné implementaci zvýšit počet vyměňovaných HTTP požadavků. Objem dat je sice nižší, ale zátěž díky velkému počtu požadavků neklesne.

Nevýhodou použití toto technologie je, že webové stránky se chovají jako plnohodnotná aplikace se složitou vnitřní logikou. Není to už posloupnost stránek, mezi kterými se lze navigovat i pomocí tlačítek Zpět a Další. Moderní AJAXové aplikace jsou

schopny funkce těchto tlačítek (přinejmenším částečně) obnovit za použití různých technik (např. využití části adresy za znakem # či pomocí neviditelných IFRAMEs).

Problémem AJAXových aplikací také může být síťová latence. Komunikace přes Internet má negativní dopady na rychlost odezvy a interaktivitu uživatelského rozhraní. Pokud uživateli není jasně signalizováno, že aplikace zpracovává jeho požadavek (a na pozadí komunikuje se serverem), jediné, co zaregistruje, je zpožděná reakce (mezitím se dokonce může snažit operaci spustit znovu, neboť se domnívá, že systém jeho příkaz ignoroval).

Další nevýhodou AJAXu je nutnost používat moderní grafické prohlížeče, které podporují potřebné technologie a ty musí být povoleny. Všechny dnešní běžné prohlížeče však tyto technologie alespoň v základu podporují. Více informací se lze dozvědět ve zdroji [17].

JSON (JavaScript Object Notation)

Je textový formát, sloužící pro výměnu dat na webu, který je snadno čitelný strojově i člověkem. JSON není vázaný na konkrétní platformu. Na Webu se JSON používá pro výměnu dat AJAXových aplikací. Je založený na dvou strukturách.

První je kolekce párů název/hodnota. Ta bývá v různých programovacích jazycích realizována jako objekt, záznam (record), struktura (struct), slovník (dictionary), hash tabulka, klíčový seznam (keyed list) nebo asociativní pole. Formát zápisu je uvedený na příkladu níže.

Příklad 1 Zápis atributu číslo sestavení ve formátu JSON

```
{label : 'Číslo', id : 'csest', key : 'csest'}
```

Druhou strukturou je tříděný seznam hodnot. Ten je u většině jazyků realizován jako pole, vektor nebo seznam (list). Formát zápisu takové struktury je uvedený na příkladu 2 viz. níže.

Příklad 2 Zápis výpisu tabulky pro sestavení pomocí JSON

```
columns :[    {label : 'Id', id : 'id', key : 'id', 'className' : 'nr'}
, {label : 'Číslo', id : 'csest', key : 'csest'}
, {label : 'Název', id : 'nsest', key : 'nsest'}
, {label : 'Skupina', id : 'skup', key : 'skup'}
, {label : 'FS', id : 'fs', key : 'fs'}
, {label : 'Poznámka', id : 'comment', key : 'comment'}
]
```

Přestože JSON je původně odvozen z JavaScriptu, je jazykově nezávislý. Implementace JSON například existuje také pro PHP (od verze 5.2 nativně), Ruby nebo

Python a existují volně dostupné knihovny pro většinu jazyků používaných pro tvorbu webových aplikací. Více informací k této technologii může čtenář nalézt ve zdrojích [13],[14].

V našem projektu tuto technologii využíváme při zobrazení číselníků do tabulky. Výpis těchto tabulek je realizovaný pomocí komponenty yahoo.dataTable AJAXové knihovny jMaki.

Příklad 3 Ukázka zdrojového kódu komponenty DataTable knihovny jMaki

```
<a:widget name="yahoo.dataTable"
        subscribe="/table/units_katlistlist_table"
        args="{
        columns :[
        {label : 'Id', id : 'id', key : 'id', 'className' : 'nr'}
        , {label : 'Číslo', id : 'ckat', key : 'ckat'}
        , {label : 'Název', id : 'nkat', key : 'nkat'}
        , {label : 'Def. hodnota', id : 'defval', key : 'defval'}
        , {label : 'Poznámka', id : 'comment', key : 'comment'}
        ]}"
        service="/KatListServlet?action=list&rowsonly=0&format=units/xml/katListList.xsl"
        id="units_katListlist_table"
/>
```

Webové služby (web service)

Je souhrnné označení pro sadu technologií umožňující komunikaci mezi objekty. Webová služba je jednoduchá komponenta nabízející určitou službu. Například pomocí webových služeb můžeme zjistit aktuální kurz měn, aktuální počasí v konkrétním místě nebo překlad textu. V našem projektu si jako příklad můžeme uvést webovou službu pro nastavení vlastnosti viz. Příklad 4.

Příklad 4 Zdrojový kód webové služby VlastnostWS

```
package cz.elzel.tpp_ws;
import cz.elzel.system.MySqlDbf;
import cz.elzel.tpp_base.VlastnostDAO;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
@WebService()
public class VlastnostWS {
```

```
    /**
    * Web service operation
    */
    @WebMethod(operationName = "setVlastnostValue")
    public String setVlastnostValue(
        @WebParam(name = "type") String type
        , @WebParam(name = "owner_id") int owner_id
        , @WebParam(name = "name") String name
        , @WebParam(name = "value") String value
        , @WebParam(name = "loginname") String loginname) {
    //TODO write your implementation code here:
        MySqlDbf dbf = new MySqlDbf("tpp", "tppuser", "localhost:3306", "elzel2008");
        dbf.connect();
        VlastnostDAO vlDAO = new VlastnostDAO(dbf, loginname);
        return(vlDAO.setVlastnostValue(type, owner_id, name, value) ? "1" : "0");
        }
}
```

Tato webová služba nabízí nastavení vlastnosti prostřednictvím metody setVlastnostValue se zadanými parametry type, owner_id, name, value a loginname. Jako odpověď vrací 1 nebo 0, podle toho zda byla vlastnost nastavena nebo ne. Pomocí této služby mohou další části systému TPP (např. pro rozpis stavebního objektu) komunikovat se správou číselníků, pokud požadují založení nové vlastnosti.

Komunikace webových služeb a protokol SOAP

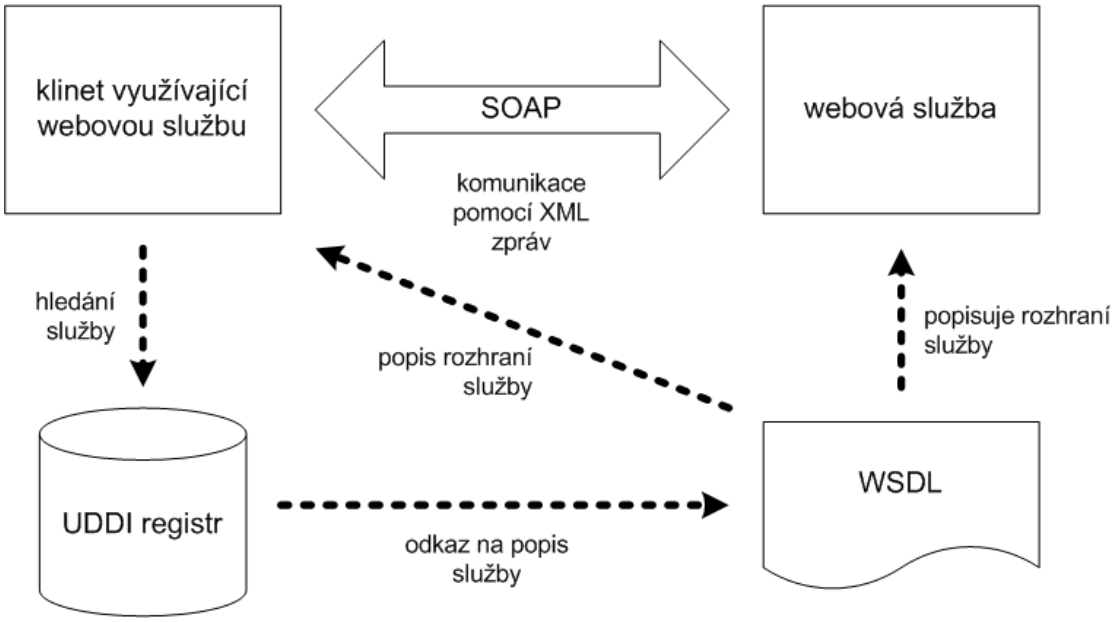
Webové služby umožňují jednoduchou komunikaci mezi aplikacemi, která je založena na nezávislých standardech, především na jazyce XML a protokolu HTTP. Jednotlivé aplikace si mezi sebou zasílají XML zprávy, které ve svém těle přenášejí dotazy a odpovědi jednotlivých aplikací. Infrastruktura webových služeb je založena na třech základních technologiích.

- **SOAP (Simple Object Access Protocol)** – protokol používaný pro komunikaci
- **WSDL (Web Services Description Language)** – standardní formát pro popis rozhraní webové služby
- **UDDI (Universal Description, Discovery and Integration)** – standardní mechanismus, který umožňuje registraci a vyhledávání webových služeb

Na obrázku Obr. 4 je znázorněný vztah mezi těmito technologiemi. Každá webová služba by měla mít k dispozici svůj formální popis v jazyce WSDL. Z něho jde automaticky vygenerovat soapový požadavek. Ve větších systémech nebo přímo v otevřeném prostředí

Internetu se popis služby může zaregistrovat do UDDI registru. Ten slouží jako jakýsi telefonní seznam („zlaté stránky“), který umožňuje vyhledávání služeb s určitými parametry.

Klient, který chce využít webovou službu, získá buď přes UDDI nebo přímo její popis. Z něj je jasné, jakou strukturu mají mít soapové zprávy (volání i odpověď) a kam se má webové službě poslat, aby ji rozpoznala.



Obr. 4 Vazby tří technologii při komunikaci webových služeb, zdroj [15]

SOAP

SOAP je protokol pro posílání zpráv XML a je základem webových služeb. SOAP umožňuje zaslání XML zprávy mezi dvěma aplikacemi. Pracuje na principu peer-to-peer. Nejčastěji se SOAP používá jako náhrada vzdáleného volání procedur (RPC), tedy v modelu požadavek/odpověď. Jedna aplikace pošle ve formátu XML požadavek (volání) druhé aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu (odpověď) zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde soapová zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď.

První verze (1.0) protokolu SOAP vznikla na konci roku 1999 jako výsledek společné práce firem DevelopMentor, Microsoft a UserLand, které chtěly vytvořit protokol pro vzdálené volání procedur (RPC) založený na XML. V průběhu roku 2000 se k podpoře přihlásila i firma IBM a nová verze SOAPu 1.1 byla zaslána W3C konsorciu. Dnes je již verze protokolu 1.2.

Zpráva v SOAPu je jednoduchý XML dokument, který má kořenový element Envelope. V této obálce jsou pak uzavřeny dva elementy Header (hlavička) a Body (tělo). Hlavička je přitom nepovinná a používá se pro přenos pomocných informací pro zpracování zprávy. Například identifikaci uživatele, autentizační informace (jméno, heslo) atp.

O to nejdůležitější se stará tělo zprávy, v němž se přenášejí informace identifikující volanou službu a předávané parametry, resp. návratové hodnoty služby. Zpráv může být více pro volání více funkcí/metod najednou. SOAP používá jmenné prostory pro identifikování jednotlivých částí XML zprávy. Obálka, hlavičky a tělo zprávy patří do jmenného prostoru http://schemas.xmlsoap.org/soap/envelope/.

Pomocí technologie SOAP a webových služeb je zajištěna komunikace uživatelského rozhraní správy číselníků se systémem TPP. Výhodou zvoleného způsobu je snazší údržba a modifikace částí hlavního systému.

WSDL (Web Services Description Language)

Jazyk WSDL je určený k popisu síťových služeb jako množiny koncových bodů zpracovávajících zprávy. Operace a zprávy jsou popisovány na abstraktní úrovni a teprve poté jsou svázány s konkrétním síťovým protokolem a datovým formátem. To umožňuje snadné vytvoření popisu rozhraní, které nabízí jednu službu několika způsoby. V praxi WSDL popisy nejčastěji popisují služby, které si posílají zprávy pomocí SOAP a protokolu HTTP. WSDL soubor s definicí rozhraní služby je XML dokument. Skládá se zejména z následujících elementů, které tvoří základní části každého WSDL popisu.

- **Types** - obsahuje definici datových struktur používaných ve zprávách. K definici lze použít teoreticky libovolný typový systém, ale nejčastěji se používají XML schémata. Nástroje pro webové služby se starají o mapování datových typů podle XML schémat na nativní datové typy použitého jazyka.
- **Message** - definuje formát předávaných zpráv pomocí dříve definovaných datových typů. Zprávy fungují jako vstupní anebo výstupní struktury pro operace. Každá zpráva se může skládat z několika logických částí s vlastním datovým typem. Při použití SOAPu pro RPC odpovídá jedna část zprávy jednomu parametru vzdálené metody.
- **Operation** - abstraktní definice operací, které jsou službou podporovány. U operace se definuje jaké má vstupy a výstupy. Vstup a výstup je popsán již existující zprávou (message). V SOAP RPC modelu odpovídá operace metodě.
- **PortType** - sdružuje dohromady několik operací.

- **Binding** -slouží pro navázání určitého typu portu (portType) na konkrétní protokol a formát přenosu zpráv.
- **Port** - jeden koncový bod služby definovaný jako kombinace síťové adresy a dříve definované vazby (binding).
- **Service** - Sdružuje několik koncových bodů (portů) do jedné služby.

Pro ukázku je níže uvedený popis webové služby Vlastnost WSService pomocí jazyka WDSL. Tyto informace jsou čerpány ze zdroje [16].

Příklad 5 Ukázka popisu webové služby VlastnostWSService pomocí WDSL

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2-b05-RC1.
-->
<definitions targetNamespace="http://tpp_ws.elzel.cz/" name="VlastnostWSService"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://tpp_ws.elzel.cz/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
<types>
    <xsd:schema>
        <xsd:import namespace=http://tpp_ws.elzel.cz/
        schemaLocation="VlastnostWSService_schema1.xsd"/>
    </xsd:schema>
</types>
<message name="setVlastnostValue">
    <part name="parameters" element="tns:setVlastnostValue"/>
</message>
<message name="setVlastnostValueResponse">
    <part name="parameters" element="tns:setVlastnostValueResponse"/>
</message>
<portType name="VlastnostWS">
    <operation name="setVlastnostValue">
        <input message="tns:setVlastnostValue"/>
        <output message="tns:setVlastnostValueResponse"/>
    </operation>
</portType>
<binding name="VlastnostWSPortBinding" type="tns:VlastnostWS">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
        <operation name="setVlastnostValue">
    <soap:operation soapAction=""/>
    <input>
```

```
        <soap:body use="literal"/>
    </input>
    <output>
        <soap:body use="literal"/>
    </output>
    </operation>
</binding>
<service name="VlastnostWSService">
<port name="VlastnostWSPort" binding="tns:VlastnostWSPortBinding">
    <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
</port>
</service>
</definitions>
```

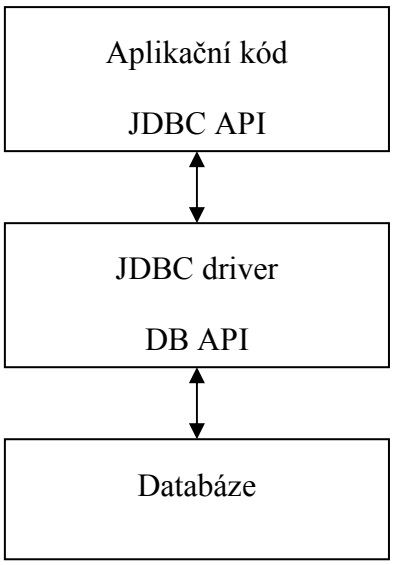
Příklad 6 Popis metody setVlastnostValue webové služby VlastnostWService

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" targetNamespace="http://tpp_ws.elzel.cz/"
xmlns:tns="http://tpp_ws.elzel.cz/" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="setVlastnostValue" type="tns:setVlastnostValue"/>
<xs:element name="setVlastnostValueResponse" type="tns:setVlastnostValueResponse"/>
<xs:complexType name="setVlastnostValue">
    <xs:sequence>
        <xs:element name="type" type="xs:string" minOccurs="0"/>
        <xs:element name="owner_id" type="xs:int"/>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="value" type="xs:string" minOccurs="0"/>
        <xs:element name="loginname" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="setVlastnostValueResponse">
        <xs:sequence>
            <xs:element name="return" type="xs:string" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>
</xs:schema>
```

JDBC

Samotná databáze číselníků je nasazena na databázovém serveru MySQL 5.x a spojení s ní je zajištěno pomocí JDBC (Java Database Connectivity) technologie. Ta poskytuje základní rozhraní pro unifikovaný přístup k různým typům databází. Základem konceptu

JDBC je využití funkčnosti poskytované JDBC ovladačem, který je následně překládá do nativních volání dané databáze. Díky tomu je aplikační programátor odstíněn od specifického API databáze a může se naučit jednotné rozhraní JDBC, které pak použije pro přístup do libovolné databáze, která poskytuje JDBC ovladač. Zjednodušené schéma je zobrazeno na následujícím obrázku.



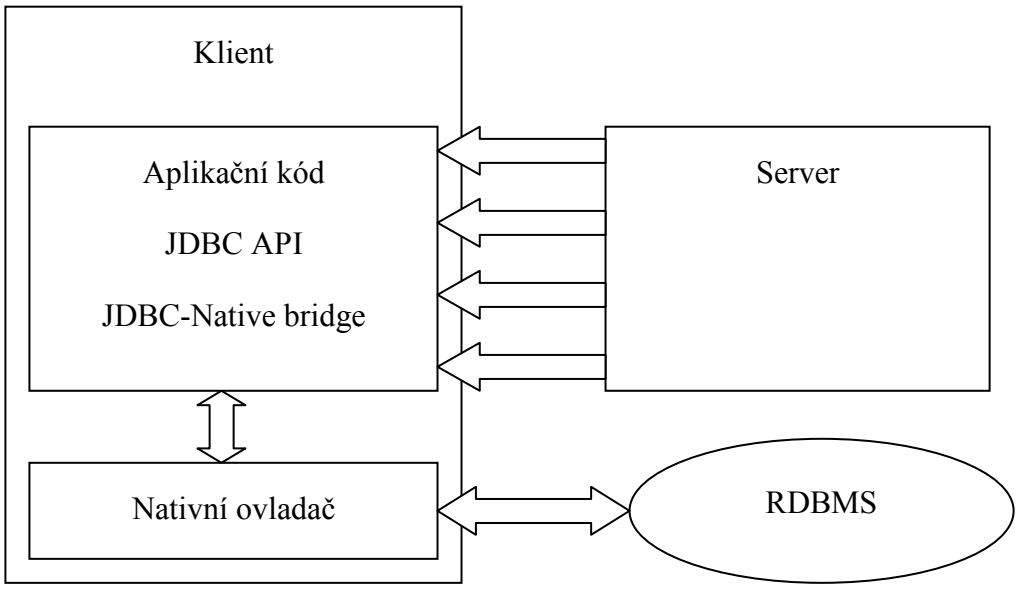
Obr. 5 Zjednodušené schéma JDBC architektury, zdroj [11]

JDBC specifikace rozděluje JDBC ovladače do čtyř skupin. Můžeme je označit jako typ 1 až 4.

Ovladač typu 1 využívá lokální ODBC ovladač a přistupuje k němu přes „JDBC-ODBC bridge“. Tento způsob komunikace s databází vyžaduje nainstalování a nastavení lokálního ODBC ovladače společně s aplikací v Javě, která ODBC ovladač bude využívat. ODBC ovladače jsou specifické pro každého výrobce databáze, a proto je s nimi práce poměrně složitá. Většinou se nevyhneme instalaci lokálních DLL knihoven, které musí být synchronizovány s použitou databází. Administrace těchto knihoven je časově náročná vzhledem k nejednotnosti rozhraní, použití a nastavení. Tento typ ovladače nachází uplatnění hlavně pro testování v prostředí Windows nebo na internetové/intranetové aplikace využívající JSP/Servlety. Hodí se tedy spíše na serverově orientované aplikace. Použití v čistě klientsky orientovaných aplikacích je komplikované kvůli náročnosti na konfiguraci na klientském počítači.

Druhý typ ovladače překládá požadavky z JDBC do určitého specifického ovladače, který je v nativní podobě nainstalovaný na počítači a je určen pro konkrétní typ databáze.

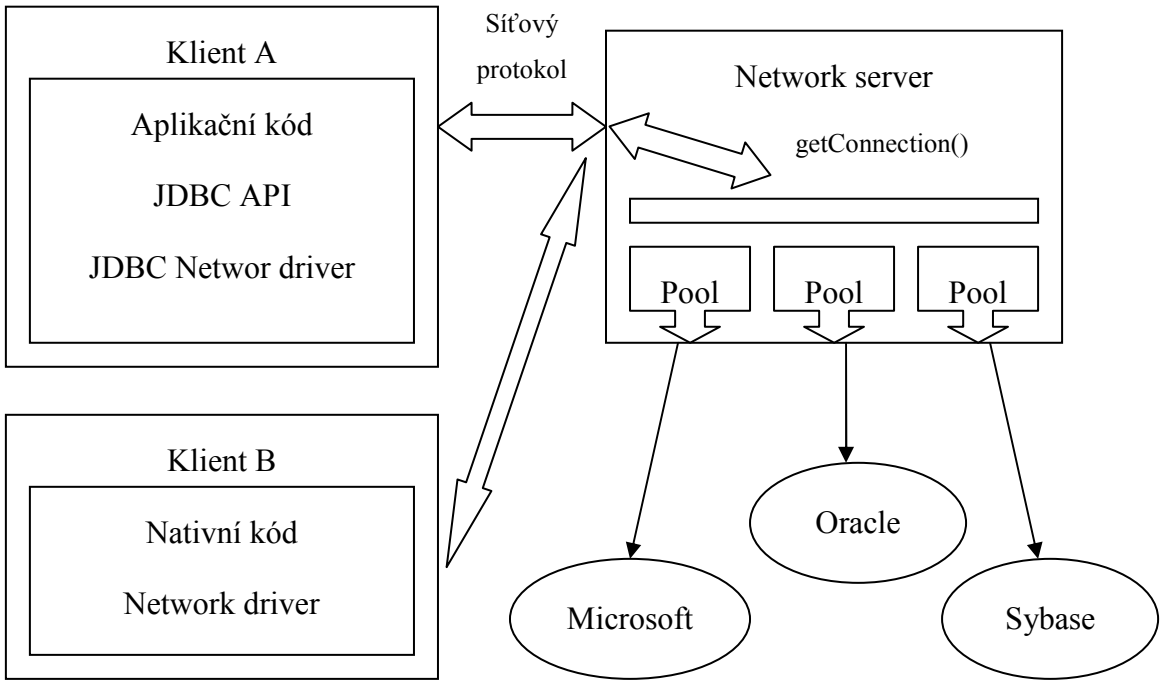
Tento typ ovladače má podobné výhody a nevýhody jako ovladač JDBC-ODBC. Opět se jedná o variantu ovladače, která se hodí spíše na serverové aplikace. Největší nevýhodou je časová náročnost při administraci a při nasazení.



Obr. 6 Ovladač JDBC typu 2, zdroj [11]

Ovladač kategorie 3 již nepoužívá žádný nativní kód pro ovladač, ale je založen čistě na Javě a JDBC. Konvertuje svoji komunikaci do síťového protokolu, který se spojuje s centrálním serverem, který poskytuje připojení k databázi. Tento server konvertuje síťový protokol, kterým komunikuje s klienty, do databázově specifického protokolu, jemuž již databáze rozumí. Tento typ ovladače je vysoce efektivní. Především s ohledem na možnost „poolingu“ připojení. Velice často nastává situace, kdy mnoho klientů vytváří fyzické připojení do databáze, které vyžadují velké režijní náklady. Technika pooling připojení (connection pooling) vytváří jakýsi zásobník (pool), kam se ukládají jednotlivá spojení, která se dají využít vícenásobně během jedné session. To výrazně zvyšuje výkon, především ve třívrstvé architektuře, kde klienti mohou sdílet méně fyzických připojení k databázi než bez poolingu. Další výhodou tohoto typu ovladače je možnost připojení k sadě heterogenních databázových systémů. Tato architektura se obvykle používá v případě rozsáhlých systémů, kde z historických důvodů mohou být rozdílné databázové produkty. Network server v podstatě nabízí unifikované rozhraní a pro aplikační programátory se tváří jako jedna databáze. Tento typ network serveru stačí nastavit administrátorem pouze jednou a následně ho můžeme provozovat. Kromě toho se k network serveru mohou připojovat i jiní než Java klienti. Je to díky tomu, že síťový protokol je platformově nezávislý. Tato architektura

poskytuje nejen lepší možnosti k optimalizaci výkonu („pooling“), spojení heterogenních databází, ale umožňuje i provázání heterogenních klientských platforem.



Obr. 7 Ovladač JDBC typ 3, zdroj [11]

Ovladač typu 4 je celý napsán v Javě s plnou podporou pro optimalizace vzhledem k dané databázi. Výhodou tohoto ovladače je, že klient nemusí být jakkoli konfigurován a nejsou nutné žádné lokální klientské instalace ovladačů.

Základem JDBC ovladače je specifická třída, obvykle poskytována výrobcem databáze. V prvním kroku je nutné ovladač nahrát a registrovat ve své aplikaci. K tomu slouží příkaz Class.forName("jméno_JDBC_ovladače");. V našem projektu využíváme databázový server MySQL. Nahrání tohoto ovladače do našeho projektu je zapsáno pomocí příkazu Class.forName("com.mysql.jdbc.Driver");. Volání této statické metody musí být v uzavřeno do try-catch bloku. To platí i pro ostatní operace prováděné nad databází. V tomto případě nebude vyvolána výjimka SQLException jako u metod pro práci s databází, ale vyvolá se ClassNotFoundException, což je výjimka informující, že systém nedokázal tuto třídu najít.

Registrace JDBC ovladače je vázána na třídu DriverManager, jejíž součástí jsou metody pro správu a práci s JDBC. Jde o vrstvu mezi aplikačním kódem a JDBC ovladačem. Obecně by měl tento ovladač zavolat při nahrání metodu registerDriver() třídy

DriverManager. Ten pak ovladač eviduje a je poskytnut pro práci s databází. To ovšem není pro aplikačního programátora důležité, protože se to děje bez jeho vědomí.

Jakmile byl JDBC ovladač nahrán a správně zaregistrován třídou DriverManager, je možné navázat spojení s databází. Spojení s databází se identifikuje jako „databázové URL“. V našem projektu je tato adresa zapsána jako textový řetězec jdbc:mysql://localhost:3306/TPP, který se skládá z následujících částí:

- Na začátku je uvedeno slovo „jdbc“, to je určené specifikací.
- Poté může být uveden podprotokol, v našem případě je zapsáno slovo „mysql“, což je záležitost závislá od daného poskytovatele databáze.
- V další části řetězce následuje adresa serveru „//localhost:3306/“ a jméno konkrétní databáze (v našem případě jde o databázi „TPP“), se kterou budeme navazovat spojení.

URL tedy jednoznačně identifikuje datový zdroj, se kterým navazujeme spojení. K tomu slouží metoda getConnetion() třídy DriverManager. Tato metoda vrací objekt typu Connection. Příklad 7 obsahuje ukázku připojení k databázovému serveru v projektu pro správu číselníků.

Příklad 7 Spojení s databázovým serverem MySQL v projektu TPP

```
protected Connection conn;
public boolean connect() {
        boolean result;
        this.conn = null;
        this.connected = false;
        try {
            Class.forName(this.getDriver());
            String connectionString = "jdbc:mysql://localhost:3306/TPP“;
            this.conn = DriverManager.getConnection(connectionString, this.getUserName(),
            this.getPassword());
            this.connected = true;
            result = true;
            this.setResultSet(null);
        } catch (ClassNotFoundException e) {
            System.out.println("Ovladač MySQL nenalezen. " + e.getMessage());
            result = false;
        } catch (SQLException e) {
            System.out.println("Připojení k dbf se nezdařilo. " + e.getMessage());
            result = false;
```

```
        }
    return (result);
}
```

Pokud se podaří navázat spojení s databází, můžeme nad ní provádět další operace. K tomu je určena další třída Statement, kterou nám poskytne vytvořený objekt Connection. Třída Statement implementuje rozhraní java.sql.Statement popsané v dokumentaci JDK. Instance této třídy slouží pro posílání SQL příkazů databázi, kde jsou zpracovány a vrací odpovídající hodnoty. Statement nabízí velké množství metod. Asi nejčastěji využívané metody jsou executeQuery(), executeUpdate().

První metoda, executeQuery() je určena k posílání dotazů typu SELECT do databáze, výsledkem jsou objekty třídy ResultSet (její metody opět určuje rozhraní java.sql.ResultSet, je vhodné se s nimi dobře seznámit z dokumentace). Výsledný „ResultSet“ již obsahuje skutečná data, která vrátil databázový server a s nimiž můžeme dál pracovat.

Druhá metoda, executeUpdate(), slouží k aktualizaci databáze a přijímá znakový řetězec podle standardu SQL92 pro aktualizaci, tedy příkazy jako UPDATE, INSERT, DELETE. Ukázku metody executeQuery() uvádí Příklad 8. Více informací o JDBC se lze dozvědět ve online zdroji [11].

Příklad 8 Metoda pro volání SQL dotazu typu SELECT

```
public boolean executeQuery(String sql) {
        boolean result;
        try {
            Statement s = this.conn.createStatement();
            this.setResultSet(s.executeQuery(sql));
        } catch (SQLException e) {
            System.out.println("Chyba při vykonávání dotazu: " + sql);
            this.setResultSet(null);
        }
        try {
            result = this.getResultSet() != null && this.getResultSet().first();
        } catch (SQLException e) {
            result = false;
        }
    return (result);
}
```

XHTML a CSS

Je značkovací jazyk určený pro tvorbu hypertextových dokumentů v prostředí WWW. Byl vyvinut a standardizován konsorciem W3C. Původně se XHTML mělo stát nástupcem jazyka HTML, jehož vývoj byl ukončen verzí 4.01. V roce 2007 došlo k založení pracovní skupiny, která má za cíl vytvořit novou verzi HTML s označením HTML 5 a její XML variantu XHTML 5. Vedle toho paralelně pokračuje i vývoj XHTML 2.0.

Hlavní výhodou této moderní technologie je oddělení struktury a obsahu dokumentu od jeho vzhledu. Samotný vzhled je definován v souboru s příponou .css pomocí kaskádového stylu. Tento způsob prezentace dokumentů s sebou nese další výhody. Pokud dokumenty dodrží tento standard, jsou co se týče objemu dat menší a přehlednější. Také se snáze upravují. Jeden externí stylový předpis pomocí kaskádových stylů zajistí konzistentní vzhled celému webu. Načítá se jen s první stránkou a u dalších se jen „kešuje“. Další nespornou výhodou je jednoduchá a levnější změna vzhledu (redesign).

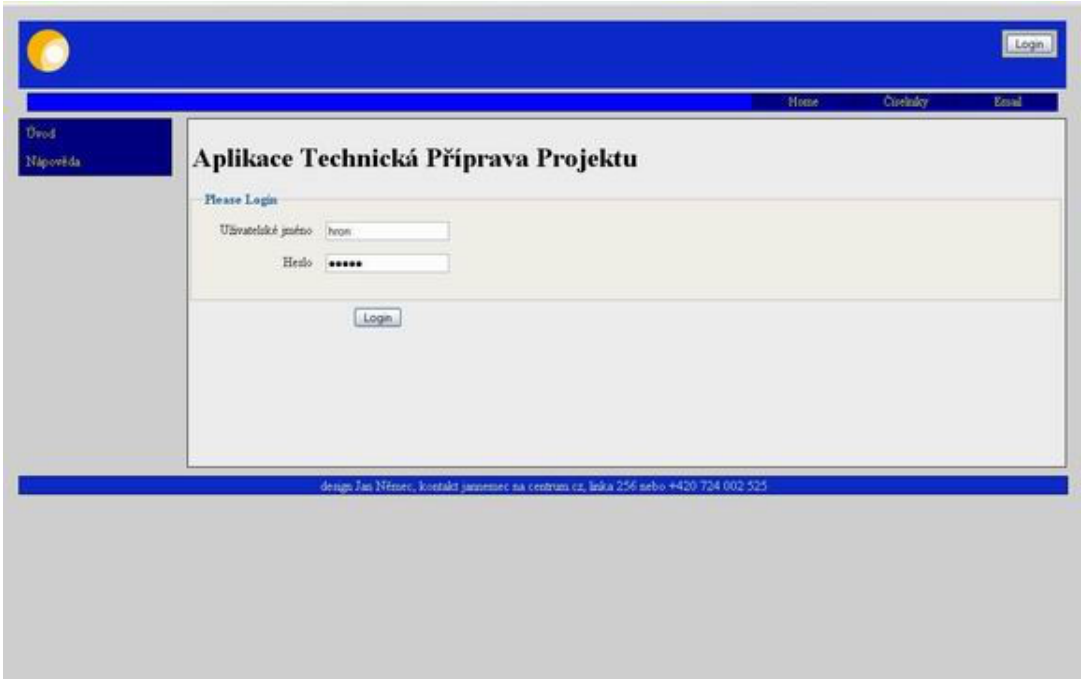
V našem projektu je tato technologie použita jako základ grafického uživatelského rozhraní správy číselníků. Více informací k této technologii se čtenář může dozvědět v knize [19].

5 Softwarové řešení správy číselníků TPP

Samotný systém je rozdělený do jednotlivých částí, které mohou pracovat samostatně. Tento způsob realizace je vhodný zejména pro snadnou údržbu a modifikaci zdrojového kódu. Praktická část této diplomové práce je zaměřena na vytvoření aplikace pro správu číselníků, která je součástí aplikace TPP. Program je od počátku vytvářen s tím, že se bude jednat o webovou aplikaci. Návrh softwarového řešení můžeme rozdělit do dvou etap. Na počátku vývoje jsme si vytvořili projekt nazvaný TPP. Jedná se o knihovnu tříd, ve které jsou definovány základní třídy s atributy a metodami podle objektového modelu z kapitoly 3.4. Nadstavbou nad projektem TPP je uživatelské rozhraní TPP_interface, což je projekt vytvořený jako webová aplikace využívající technologii JSP. Elektronická správa číselníků komunikuje se svým okolím pomocí protokolu SOAP a webových služeb. Bližší informace o komunikaci mezi objekty jsou uvedeny v kapitole 4.

Přihlášení a správa uživatelů

S aplikací TPP mohou pracovat pouze uživatele, kteří mají vytvořený uživatelský učet a mají nastavena patřičná uživatelská oprávnění od administrátora. Po spuštění aplikace se uživateli zobrazí přihlašovací formulář, do kterého se zadá uživatelské jméno a heslo pro vstup do aplikace. Nepřihlášený uživatel má možnost pouze nahlédnout do nápovědy, ostatní funkce nemá přístupné.



Obr. 8 Přihlašovací formulář pro vstup do aplikace TPP

Informace o uživatelích této aplikace jsou uloženy v databázi TPP v tabulce User. Oprávnění pro přístup k aplikaci jednotlivých účtu obsahuje tabulka User_right. Správu nad těmito dvěma tabulkami zajišťuje administrátor systému.

V databázi má každý účet uložený otisk hesla vytvořený pomocí hashovací funkce implementující algoritmus MD5. Což je transformace, u které se na vstupu přijímá řetězec znaků o libovolné délce. Výsledem je pak řetězec znaků s pevnou délkou, tzv. hash nebo česky také otisk. Hashovací funkce se často používají v kryptografii. To znamená, že po zadaní hesla uživatele do systému se provede jeho přepočet pomocí hashovaní funkce a výsledný otisk se porovná s otiskem uloženým v databázi. Výhodou tohoto řešení je, že v databázi není uloženo heslo, ale pouze jeho otisk, ze kterého heslo nelze jednoduše rekonstruovat.

Ukázka práce se správou číselníků

Po úspěšném přihlášení do systému, může uživatel nahlížet do tabulek číselníků. V horním pravém rohu je menu se třemi odkazy „Home“, „Číselníky“ a „Email“. První přesměruje uživatele na úvodní stránku, pomocí druhého přejde uživatel do samotného výpisu číselníků. Poslední odkaz „Email“ je pro odeslaní požadavku uživatele na technickou podporu spravující systém. Také se touto cestou mohou odesílat například upozornění o nestabilním chování systému. V levém menu je odkaz „Nápověda“ určená pro podporu uživatelů.

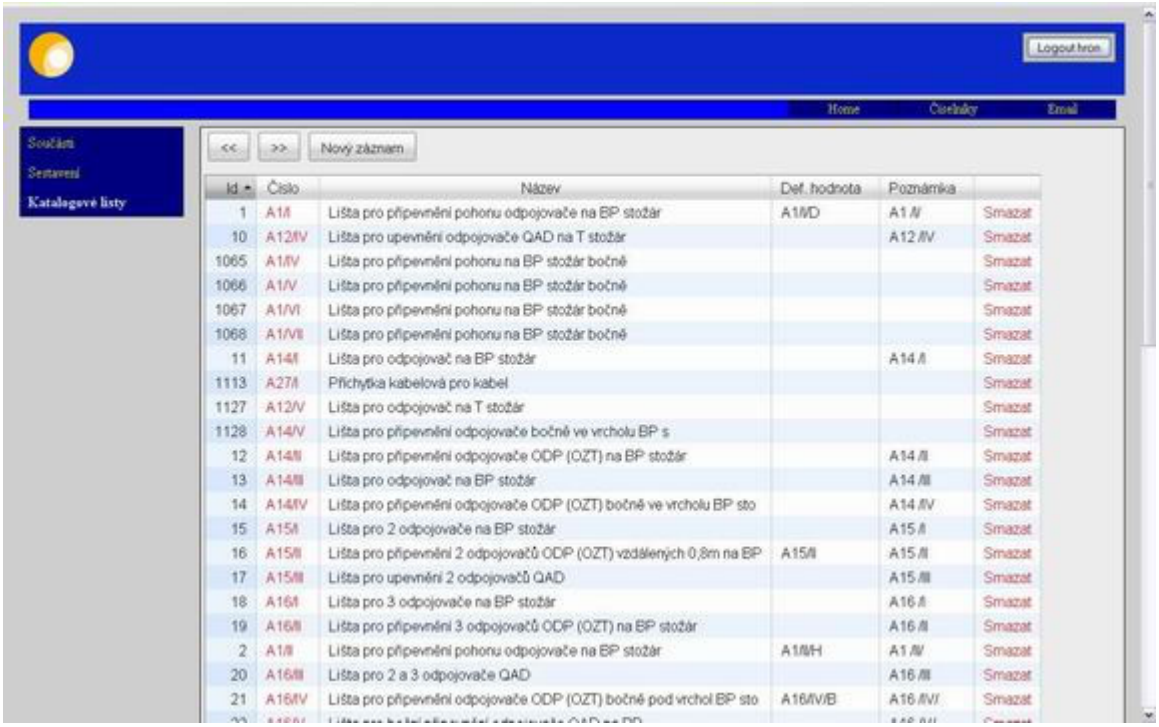
Jak už bylo uvedeno výše k nahlížení a editaci číselníků se uživatel dostane přes odkaz v pravém horním rohu „Číselníky“.

Po jeho rozkliknutí si muže uživatel vybrat jaký číselník bude prohlížet. Může to být sestavení, katalogový list nebo součást. Kusovník je součástí detailního výpisu sestavení a nemá tedy smysl jej zobrazovat samostatně.

Výpis číselníků je realizovaný pomocí tabulek. Ta je vytvořena pomocí komponenty DataTable knihovny jMaki. Do této tabulky jsou pomocí XML vloženy záznamy z databáze. Každou tabulku lze procházet pomocí tlačítek umístěných v horní a dolní části stránky. Jeden výpis na stránku je omezený na 50 záznamů.

K dalším funkcím které tato aplikace nabízí je přídání a odebírání záznamů z databáze. Tyto operace má možnost provádět pouze uživatel, který má oprávnění editovat číselníky. Ostatním uživatelům jsou tyto funkce zakázány. Přidání nového záznamu do databáze se provede kliknutím na tlačítko s název „Nový záznam“, kdy se otevře prázdný formulář pro doplnění hodnot parametrů. U každého jednotlivého záznamu je na konci hypertextový odkaz „Smazat“, pokud na něj uživatel klikne objeví se dialogové okno s textem „Opravdu chcete

tento záznam odebrat?“ Pokud uživatel zvolí „ANO“ záznam bude vymazán, pokud zvolí „NE“ vrátí se zpět výpisu záznamů. Toto ovládání je stejné pro všechny typy číselníků. V praxi se však samotné přidávání a odebírání záznamů tak často neprovádí. Nejčastěji dochází k modifikacím záznamů, ale mohou nastat případy, ve kterých se tyto funkce uplatní. Proto je potřeba, aby byly v systému implementovány. Náhled výpisu číselníku je zobrazený na Obr. 9.



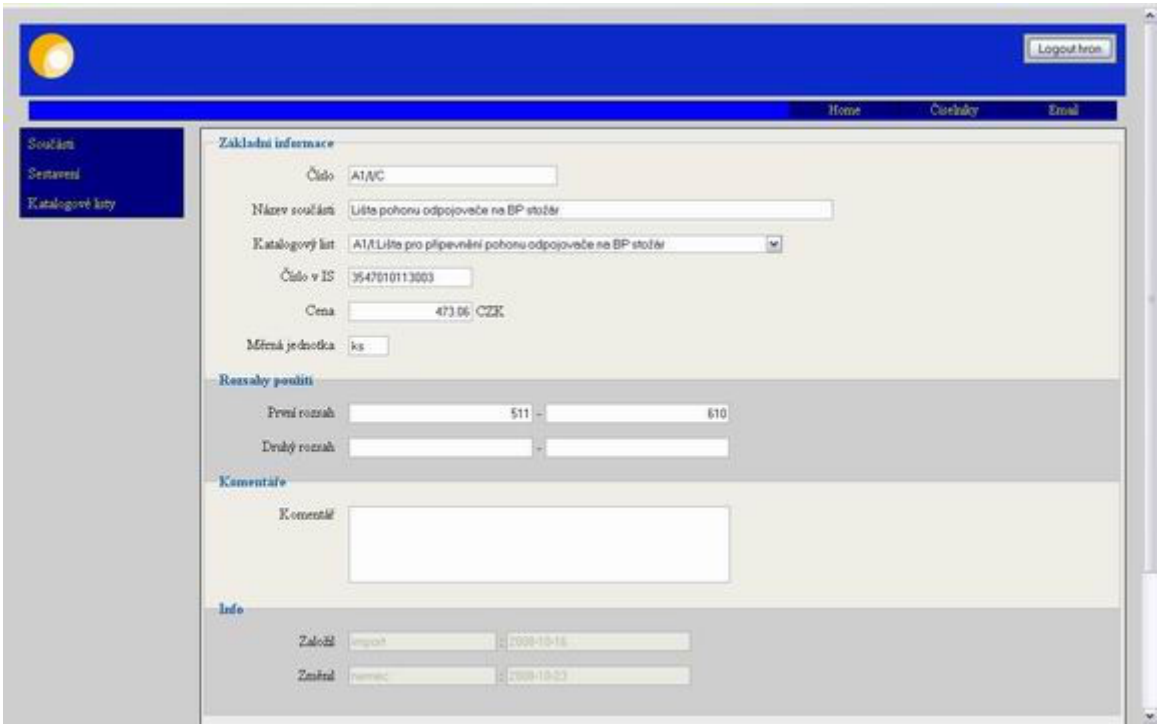
Obr. 9 Výpis katalogových listů

Pokud chce uživatel zobrazit celý detail položky, klikne na odkaz ve sloupci číslo, který je zbarvený červeným písmem.

Tento detailní výpis umožňuje uživatelům, kteří mají patřičná oprávnění, editovat záznamy číselníků. Pokud uživatel změní nějaký atribut položky, zobrazí se vedle textového pole značka, která ho upozorní, zda je zadaná hodnota validní a byla korektně uložena nebo zda je chybná. V případě správné hodnoty se vedle textového pole zobrazí zelený bod se symbolem ✓. Při chybné hodnotě je to červený bod a v něm symbol x. Korektní změna textového pole je ihned odeslána na server a záznam je aktualizován. Spolu se změnou hodnoty některého atributu položky se v databázi číselníků zaznamenává i kdo (tedy jaký uživatel) poslední změnu provedl a v jaký čas a den to bylo. U každého detailního výpisu, ať už se jedná o sestavení, katalogový list nebo součást je vypsána tabulka specifických

vlastností vztahující se k tomuto záznamu. Hodnotu každé z těchto vlastností lze měnit, přidávat, popřípadě mazat.

Ostatní uživatele mají pouze možnost čtení, to znamená že všechna textová pole jsou uzamčená. Praktická ukázka detailního výpisu sestavení je zobrazena na Obr. 10. Pro všechny další typy číselníků je způsob zobrazení stejný.



Obr. 10 Detailní výpis položky A1/I/C z tabulky součástí s možností editace

Pro zkvalitnění práce se správou číselníků budou do aplikace implementovány další funkce dle požadavků uživatelů. Je to například možnost filtrování a vyhledávání záznamů v tabulkách dle zadané hodnoty, přihlašování single sign-on, uchování historie změn a mazání kusovníků.

6 Závěr

Cílem této práce bylo definovat objektový model systému TPP pro rozpis staveb v Elektrizaci železnic Praha a.s. a vytvořit uživatelské rozhraní pro správu základních číselníků.

Ve druhé kapitole teoretické části je popsáno celkové zadání a rozsah projektu. Jsou tu uvedeny požadavky, které bude systém splňovat a technologie, na kterých bude postaven.

Třetí kapitola je rozdělena do dvou částí. Na začátku jsou vysvětleny základní pojmy související s problematikou technické přípravy a je tu graficky i slovně popsaný reálný postup rozpisu stavebního objektu. Dále je tu obecný pohled na systém z pohledu všech účastníků, kteří ho budou v budoucnu využívat. Druhá část této kapitoly je zaměřena na objektový model číselníků a stavebních projektů.

Čtvrtá kapitola je zaměřena na popis komunikace mezi jednotlivými objekty systému. Popisuje technologie využité v projektu a u každé z nich je uvedená praktická ukázka využití.

Poslední kapitola popisuje interface správy číselníků z pohledu uživatele. Vysvětluje, základní ovládání aplikace, způsob aktualizace, přidávání a mazání záznamů v číselnících. Elektronická správa číselníků je podpůrnou součástí systému a základním zdrojem dat pro rozpis staveb.

Spolupráce na vývoji aplikace TPP byla pro mě přínosná z několika důvodu. Vyzkoušel jsem si implementaci moderních technologii a tvorbu většího projektu. Při návrhu projektu jsme pro práci využívali verzovací systém SVN. Tento systém je určen k nasazení tam, kde se předpokládá spolupráce více programátorů na jednom projektu. Systém SVN pomáhá programátorům udržet aktuální verze, spolupracovat na velkém projektu a udržet přehled o postupu prací. Tento software má mechanismy, kterými je schopen uchovávat historii jednotlivých změn. Má také mechanismy, kterými je schopen řešit konflikty v případě modifikace souborů více uživateli. To zároveň umožňuje bezpečně pracovat více vývojářům současně, aniž by si navzájem přepisovali svou práci.

Dalším modulem systému TPP, na kterém se nyní bude pracovat je aplikace pro rozpis staveb a export rozpisu do informačního systému. Elektronická správa číselníků bude sloužit jako hlavní zdroj dat tohoto modulu.

Seznam bibliografických citací

[1] BURD, B. JSP: JavaServer Pages podrobný průvodce. Praha: Computer Press, 2003. 381s. ISBN 80-7226-804-X

[2] WIKIPEDIA - UML - Unified Modeling Language [online] URL: <http://cs.wikipedia.org/wiki/UML> [cit. 2008-10-01]

[3] PAVLÍČKOVÁ J., PAVLÍČEK L.: Úvod do Javy. Oeconomica, 2005. 226 s. ISBN 80-245-0963-6.

[4] NETBEANS IDE 6.1 - Connecting to a MySQL Database [online tutoriál] URL: <http://www.netbeans.org/kb/61/ide/mysql.html> [cit. 2008-07-01]

[5] NETBEANS IDE 6.0 – Introduction to Developing Web Application [online tutoriál] URL: <http://testwww.netbeans.org/kb/60/web/quickstart-webapps.html> [cit. 2008-07-01]

[6] NETBEANS IDE 6.1 – Navigation Pages in a Web Application [online tutoriál] URL: <http://www.netbeans.org/kb/60/web/pagenav.html> [cit. 2008-07-01]

[7] NETBEANS IDE 5.5 - Creating a Simple Web Application in Netbeans IDE Using MySQL [online tutoriál] URL: <http://testwww.netbeans.org/kb/55/mysql-webapp.html> [cit. 2007-11-22]

[8] WIKIPEDIA – MySQL [online] URL: < http://cs.wikipedia.org/wiki/MySQL > [cit. 2008- 09-29]

[9] NETBEANS [online] URL: <http://www.netbeans.org/index_cs.html>

[10] JAKARTA TOMCAT [online] URL: <http://www.fi.muni.cz/~kas/p090/referaty/2003-jaro/skupina10/tomcat.html> [cit. 2007-10-25]

[11] INTERVAL - Úvod do JDBC [online] URL: <http://interval.cz/clanky/uvod-do-jdbc/> [cit. 2003-03-04]

[12] WIKIPEDIA – XML [online] URL: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language> [cit. 2008-10-24]

[13] JSON – Úvod do JSON [online] URL: <http://www.json.org/json-cz.html> [cit. 2008-10-24]

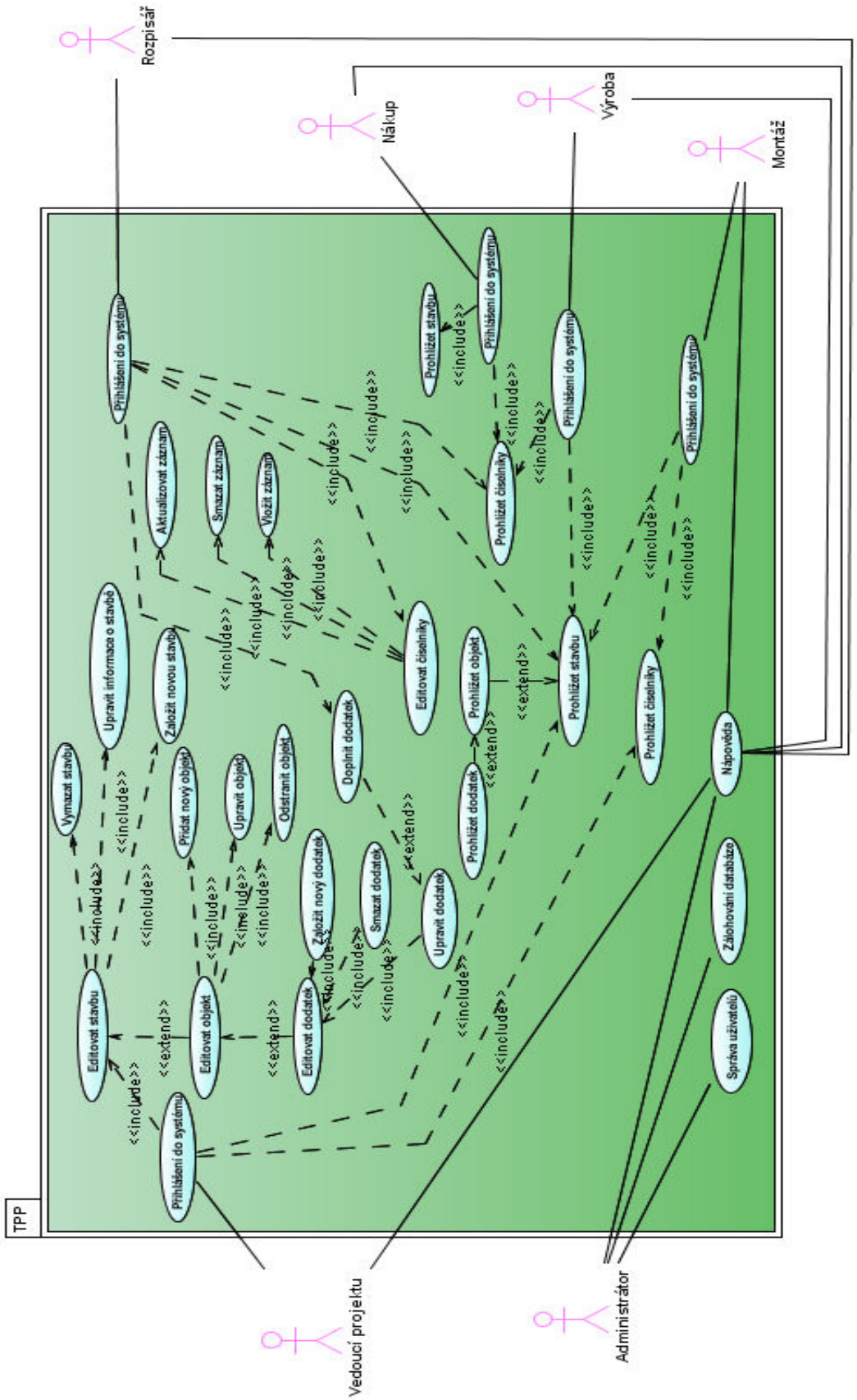
[14] ZDROJÁK – JSON jednotný formát pro výměnu dat URL: <http://zdrojak.root.cz/clanky/json-jednotny-format-pro-vymenu-dat/> [cit. 2008-11-20]

[15] WIKIPEDIA - Hashovaní funkce [online]
URL: <http://cs.wikipedia.org/wiki/Hashovac%C3%AD_funkce> [cit. 2008-11-11]
[16] KOŠEK – Využití webových služeb a protokolu SOAP při komunikaci [online]
URL:<http://www.kosek.cz/diplomka/html/websluzby.html> [cit. 2008-11-11]
[17] WIKIPEDIA - Asynchronous JavaScript and XML [online]
URL:<http://cs.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML>
[cit. 2008-11-11]
[18] ABC LINUX – výkladový slovník
URL: <http://www.abclinuxu.cz/slovnik/hash> [cit 2008-11-28]
[19] MUSCIANO CH., KENNEDY B. HTML a XHTML: kompletní průvodce. Praha: Computer Press, 2000. 633 s. ISBN 80-7226-407-9

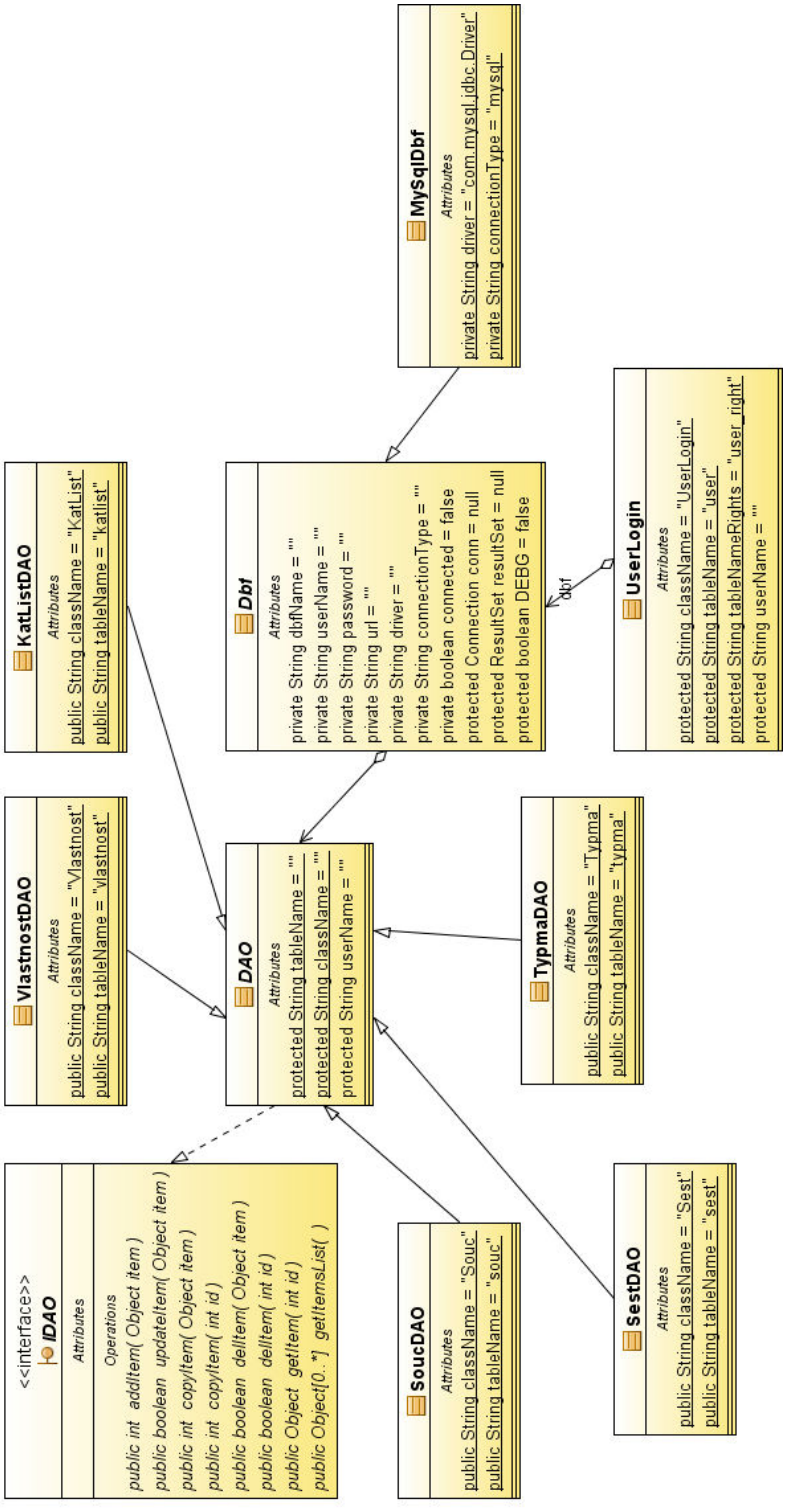
Přílohy

Seznam příloh

Příloha A: Use case diagram systému pro technickou přípravu projektu



Příloha B: Diagram tříd



Příloha C: Diagram tříd pro implementaci kusovníku