Inovace databázového interface aplikace PAVEZA

Bc. Ondřej Vršan

Diplomová práce
2014

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Vršan**
Osobní číslo:	**A12487**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Inovace databázového interface aplikace PAVEZA**

Zásady pro vypracování:

1. **Seznamte se s webovou aplikací PAVEZA a s technologií TFS.**
2. **Analyzujte stávající databázový engine a současný návrh architektury nového engine.**
3. **Upravte databázové dotazy pro engine UniDB.**
4. **Navrhněte případné doplnění funkcionality nového engine.**
5. **Proveďte zpětnou podporu odevzdaného kódu v součinnosti se zaměstnanci firmy NWT a.s.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **SHARP, John. Microsoft Visual C# 2010: krok za krokem. Vyd. 1. Brno: Computer Press, 2010, 696 s. ISBN 978-80-251-3147-3.**
2. **LACKO, Luboslav. Oracle: správa, programování a použití databázového systému. 2. dopl. vyd. Překlad Marek Kocan. Brno: Computer Press, 2007, 576 s. ISBN 978-80-251-1490-2.**
3. **AGARWAL, Vidya Vrat a James HUDDLESTON. Databáze v C# 2008: průvodce programátora. Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2009, 424 s. ISBN 978-80-251-2309-6.**
4. **MARTIN, Robert C. Čistý kód. Vyd. 1. Brno: Computer Press, 2009, 423 s. ISBN 978-80-251-2285-3.**
5. **PECINOVSKÝ, Rudolf. Návrhové vzory. Vyd. 1. Brno: Computer Press, 2007, 527 s. ISBN 978-80-251-1582-4.**

Vedoucí diplomové práce: **Ing. Bc. Pavel Vařacha, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **21. února 2014**

Termín odevzdání diplomové práce: **20. května 2014**

Ve Zlíně dne 21. února 2014

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- jsem byl seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně .....

Bc. Ondřej Vršan

ABSTRAKT

Tato práce se zabývá inovací databázového *interface* aplikace PAVEZA. Inovace má za cíl umožnit aplikaci fungovat na volitelném databázovém *engine* a minimalizovat náklady pro zavedení podpory dosud nepodporovaného *engine*. Inovace byla provedena v rámci Inovačního voucheru Zlínského kraje dotovaného Evropskou unií.

Klíčová slova:

NWT, PAVEZA, UniDB, databázvý interface, SQL, veřejná zakázka

ABSTRACT

This thesis deals with database interface innovation of PAVEZA application. Innovation aims to allow application to work with optional database engine and to minimize costs when implementing support for other database engines, which are not currently supported. Innovation was performed as a part of the Innovation voucher of Zlin region subsidized by the Europan Union.

Keywords:

NWT, PAVEZA, UniDB, database interface, SQL, government procurement

Poděkování

Děkuji svému vedoucímu, Ing. Bc. Pavlovi Vařachovi PhD., za vedení mé práce a užitečné rady pro její vypracování, zaměstnancům firmy NWT a.s. za dodané podklady pro inovaci aplikace PAVEZA a součinnost při dopracování potřebných funkcionalit a také za jejich čas na konzultačních schůzkách. Poděkování také patří mým rodičům a přítelkyni za to, že mne podporovali po celou dobu mých studií.

OBSAH

ÚVOD

Tato práce se zabývá inovací a rozšířením funkcionality produktu PAVEZA pro použití s dalším typem databázového *engine* a tím následně možnost nasadit produkt na více typech databázových systémů. Inovace povede ke zvýšení konkurenceschopnosti produktu na komerčním trhu.

Aplikace PAVEZA je komerční softwarové řešení od firmy NWT a.s. pro plánování nákupu potřeb organizace a následně spojený s evidencí veřejných zakázek včetně publikování těchto zakázek v profilu organizace.

Inovace byla provedena v rámci Inovačního voucheru Zlínského kraje dotovaného Evropskou unií. Během prací na projektu byl přepsán databázový *interface* tak, aby bylo možné aplikaci PAVEZA nasadit na libovolný databázový *engine*. Doposud bylo podporováno pouze databázové řešení od firmy Oracle.

I. TEORETICKÁ ČÁST

1 NWT A.S.

Společnost NWT a.s. se řadí mezi nejvýznamnější technologické a inovační firmy v České republice. Zákazníkovi nabízí kompletní portfolio služeb od dodávky HW a SW, přes implementaci, až po následný servis v režimu 24/7/365. Její filozofií je nabízet komplexní služby na profesionální úrovni s vysokou přidanou hodnotou pro zákazníka, cílem je dlouhodobý vztah s klienty založený na vzájemné důvěře a zkracování dodavatelsko–odběratelského řetězce. Vizí společnosti je být číslo jedna v Evropě v šíři nabídky alternativních zdrojů. [2]

Následující informace o firmě NWT a.s. byly převzaty z [3]

Vize

- Být jedničkou v šíři nabídky alternativních zdrojů energie.
- Rozvíjet nezávislou, pro partnery finančně motivující, energetiku, která bude v souladu se životním prostředím a celosvětovou enviromentální politikou.
- Být garantem kvality pro obchodní partnery a zákazníky.

Filozofie

- Budování dlouhodobých vztahů s partnery založených na vzájemné důvěře.
- Nabídka profesionálních služeb s vysokou přidanou hodnotou pro zákazníka.

Cíle

- Udržení pozice největšího dodavatele pro obnovitelné zdroje energie v rámci ČR.
- Prohloubení zákaznického portfolia.
- Posílení pozitivního povědomí o značce ve spojení s dlouholetou tradicí společnosti.
- Realizace nových projektů inteligentních a ekologických staveb.
- Navyšování kvalifikace pracovních týmů a vytvoření nových pracovních příležitostí.
- Splnění závazných ukazatelů pro získání dotace z projektu Centrum strategických služeb Zlín.
- Spolupráce v rámci partnerství a sítí mezi vzdělávacími institucemi.

Certifikace

- ČSN EN ISO 14001:2005; 9001:2009
- ČSN OHSAS 18001:2008
- ČEKIA Stability Rating A

1.1 Hlavní oblasti činnosti

Energetika

- fotovoltaické elektrárny
- bioplynové stanice
- pyrolýza
- energetické využití skládkového plynu
- termotlaká příprava biomasy pro BPS
- kotle na biomasu
- LED - průmyslové haly a veřejné osvětlení
- komplexní poradenství energetických úspor
- spořící regulátory napětí

Realizace – Specializované stavby

- technologické celky a stavby
- elektrárny na klíč (fotovoltaika, bioplyn)
- zpracování PD ve všech stupních
- inženýrská činnost
- elektromontáže (slaboproud, silnoproud NN a VN, měření a regulace)

Výroba

- bowdeny (plastové trubičky) pro automobilový průmysl
- výroba elektřiny a tepla

Prodej

- dodávky IT infrastruktury
- internetový obchod `Patro.cz`
- věrnostní systémy

Výzkum a vývoj

- tištěné polymerní spoje
- nanotechnologie
- vertikální rotor a vícepólový generátor větrných elektráren
- studie energetické účinnosti domů

Služby

- telekomunikace
- komplexní IT služby vč. outsourcingu
- správa elektráren a nemovitostí
- cloudové technologie
- vedení účetnictví
- komplexní finanční poradenství
- evropské strukturální fondy
- callcenturm, telefonní operátor
- internet service provider

2 APLIKACE PAVEZA

Informace o aplikaci PAVEZA byly převzaty z [4].

Intranetová aplikace PAVEZA (a její odlehčenější verze PAVEZA LIGHT) jako velmi efektivní elektronický nástroj usnadňuje zadavatelům přípravu a administraci zadávání veřejných zakázek při současném zvýšení transparentnosti celého procesu a s možností vnitřního i vnějšího dohledu nad přípravou a definováním zadávacích podmínek veřejné zakázky, a to již od okamžiku vzniku záměru. Systém je také průběžně aktualizován v souvislosti s novelami zákona.

2.1 Cílová skupina uživatelů

Systém PAVEZA je určen pro:

- města, obce, územně samosprávní celky
- podniky a organizace se státní účastí 1–100%
- komerční subjekty
- školství, univerzity, policie, armáda

2.2 Modul plánování

Modul plánování umožňuje vytvořit plán nákupních potřeb ve věcné struktuře (NIPEZ), a to buď z aktuálního finančního plánu organizace (například rozpočtu města) anebo přímo bez vazby na finanční plány. Díky tomu je možné snadno získat přehled o plánovaných nákupních potřebách na dané období za celou organizaci. Tímto postupem lze snadno určit hodnoty věcně setříděných budoucích nákupních akcí a také agregovat nákup do větších nákupních celků. Dosažení finančních úspor je zase snazší. Proces plánování je nastavitelné workflow s možnostmi schvalování na jednotlivých organizačních úrovních. Automatické načítání rozpočtu z účetního systému lze nastavit pouze v případě serverové verze PAVEZA instalované u zákazníka. Systém lze nastavit i opačně, tzn. pomocí tohoto modulu lze vytvořit poklady pro tvorbu finančního plánu. Dostupné funkce Modulu plánování v aplikaci PAVEZA a PAVEZA LIGHT zobrazuje tab. 1/str. 15.

Tab. 1. Modul plánování v aplikaci PAVEZA a PAVEZA LIGHT [4]

Modul plánování	**PAVEZA**	**PAVEZA LIGHT**
Import rozpočtu města nebo jiného finančního plánu organizace	•	
Rozpad vlastního finančního plánu do věcné struktury NIPEZ	•	
Vkládání nákupních potřeb ve struktuře NIPEZ bez vazby na finanční plán	•	•
Uživatelská oprávnění dle organizační struktury	•	•

2.3 Modul veřejných zakázek

Modul veřejných zakázek je vhodným prostředkem pro elektronickou podporu „klasických papírových“ veřejných zakázek. Umožňuje evidenci, přípravu a administraci všech nákupních aktivit, které probíhají podle předem stanovených postupů. Jednotlivé nákupní akce (například typicky veřejná zakázka v režimu zákona o veřejných zakázkách) jsou uživatelem zadávány do připravených formulářů. Vyžadovány jsou jen podstatné informace.
Uživatel zvolí pouze takový postup výběru dodavatele, který mu systém „povolí“, a to v souladu s interní směrnicí, případně se zákonem.

Proces přípravy veřejné zakázky (nákupní akce) může být elektronicky schvalován. Pravidla procesu a schvalování lze nastavit dle organizačních zvyklostí. Uživatel má možnost z aplikace přímo vytvářet dokumenty ve formátu PDF, které jsou pro daný zadávací postup předepsané, což zaručuje jednotnost a obsahovou správnost. Po vyhlášení VZ má uživatel k dispozici formuláře pro administraci, kdy pouze zadává informace do polí v aplikaci. Postupně je tak s pomocí generovaných PDF vytvářena požadovaná dokumentace (zápisy, protokoly apod.). Pokud se jedná o veřejnou zakázku s povinným uveřejňováním na profilu zadavatele, může být tento profil součástí dodávky řešení. Uveřejňování dokumentů a informací pak probíhá poloautomaticky, bez nutnosti uživatele vstupovat na profil a dokumenty vkládat. Dostupné funkce Modulu veřejných zakázek v aplikaci PAVEZA a PAVEZA LIGHT zobrazuje tab. 2/str. 16. [4]

Tab. 2. Modul veřejných zakázek v aplikaci PAVEZA a PAVEZA LIGHT [4]

Modul veřejných zakázek	**PAVEZA**	**PAVEZA LIGHT**
Generování dokumentů PDF dle uložených šablon	•	•
Customizace předdefinovaných PDF šablon dle potřeb zákazníka	•	omezeně
Import originálů (skenů) dokumentů	•	•
Vedení elektronického spisu dokumentů k VZ	•	•
Workflow v procesu přípravy a administrace VZ s možností schvalování	•	•
Customizace workflow dle potřeb zákazníka	•	omezeně
Integrované propojení na certifikovaný profil zadavatele EVEZA (součást)	•	•
Integrace na aukční portál proe.biz (licence proe.biz není součástí)	•	•
Uživatelská oprávnění dle organizační struktury	•	•

2.3.1 Veřejná zakázka

Veřejná zakázka je způsob nákupu zboží, objednání práce, díla nebo služby organizací se státní účastí, který hospodaří s penězi, pocházejících z daní, poplatků nebo jiných zdrojů veřejného bohatství.

Na veřejnou zakázku se vztahuje zákon č. 137/2006 Sb. o veřejných zakázkách. Tento zákon zapracovává příslušné předpisy Evropské unie a upravuje postupy při zadávání veřejných zakázek, soutěž o návrh, dohled nad dodržováním tohoto zákona a podmínky vedení a funkce seznamu kvalifikovaných dodavatelů a systému certifikovaných dodavatelů. [10]

Výhoda veřejné zakázky mj. tkví v tom, že jednotlivé subjekty spadající pod organizaci neobjednávají pohledávku samostatně u různých dodavatelů. Centrálně jsou zjištěny stejné pohledávky v rámci organizace a pro ty je vypsána veřejná zakázka. Může tak být vybrán dodavatel, který např. zboží hromadně dodá, čímž se sníží jednotková cena. Při výběru dodavatele kromě ceny hraje samozřejmě roli také kvalita služeb, provedení apod.

2.4 Výhody řešení

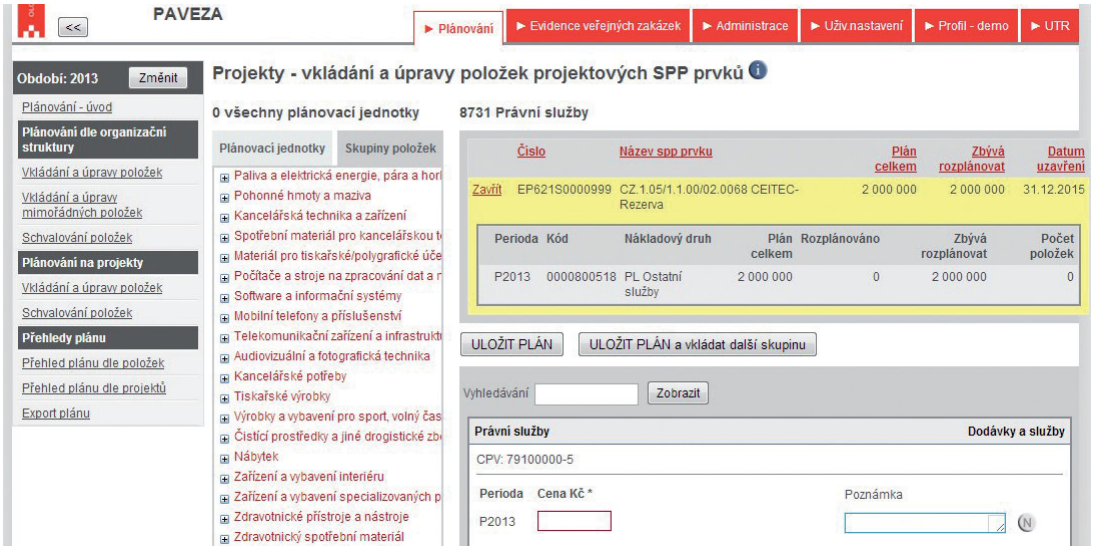
- Zavedení jednotného systému a postupů na všech organizačních úrovních zadavatele.
- Efektivní řízení procesu zadávání a správy veřejných zakázek.
- Úspora času při přípravě a administraci zadávacích řízení.
- Zjednodušení a zpřehlednění celého procesu zadávání veřejných zakázek.
- Zefektivnění rutinních administrativních úkonů při zadávaní a hodnocení VZ.
- Kompletní vedení agendy dokumentace celého zadávacího řízení.
- Správa a evidence všech veřejných zakázek v jednotném softwarovém prostředí.
- Předdefinované šablony dokumentů.
- Možné modifikace specifických potřeb dle požadavků konkrétního zadavatele.
- Manažerské přehledy a statistiky.
- Možnost integrace na datová úložiště zadavatele.

2.5 Přínosy

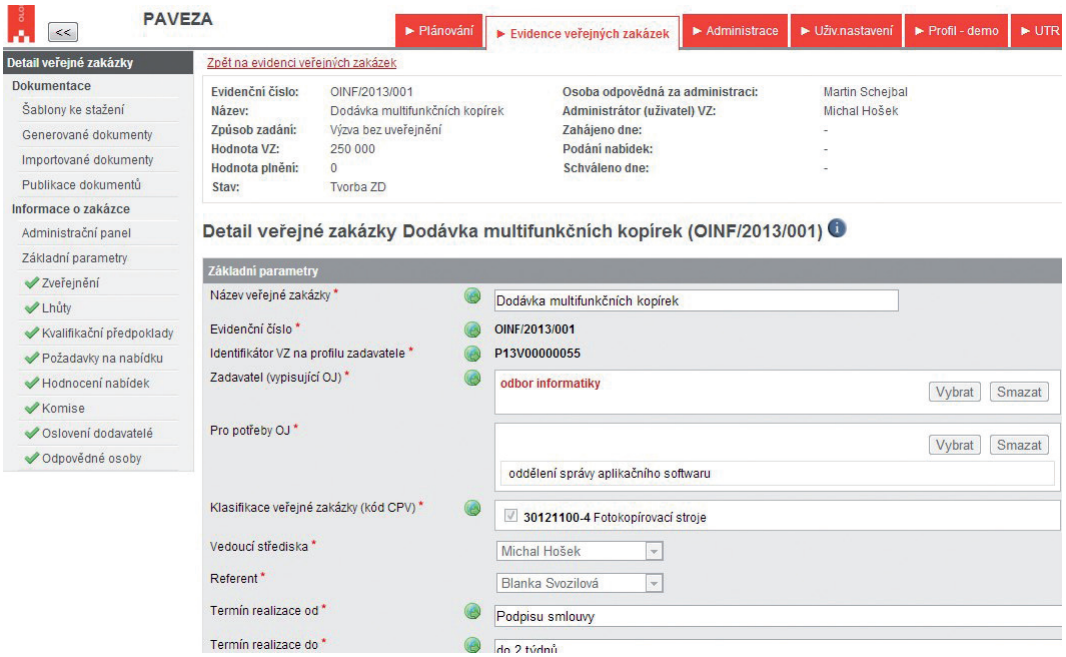
- Přímé úspory v nákupu (sdružení nákupu = výhodnější ceny).
- Sekundární úspory (efektivnější správa VZ, nižší personální náklady).
- Eliminace pokut vlivem nižšího rizika porušení zákona o veřejných zakázkách.
- Zprůhlednění systému nákupu v celé organizaci včetně organizačních složek.
- Naplnění zákonných povinností v zadávání VZ na všech org. úrovních i pro VZMR.
- Implementaci softwaru (k automatizaci nákupního procesu) provázaného na stávající systémy organizace.

2.6 Obrazové ukázky aplikace

Ukázky aplikace PAVEZA jsou vyobrazeny na obr. 1/str. 18, obr. 2/str. 18 a obr. 3/str. 19.



Obr. 1. PAVEZA (a) [4]

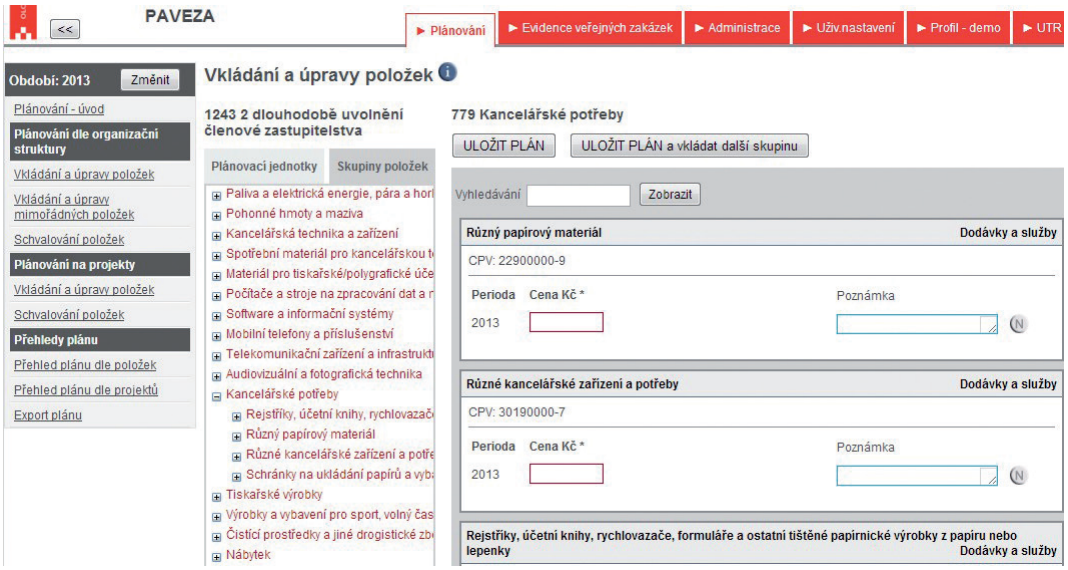


Obr. 2. PAVEZA (b) [4]

2.7 Použití v praxi

Systém PAVEZA je aktuálně nasazen např. v organizacích

- Lesy ČR
- VUT v Brně
- Magistrát města Olomouce



Obr. 3. PAVEZA (c) [4]

2.8 Použité technologie v původní verzi

Z hlediska použitých technologií v původní verzi aplikace se tato práce s ohledem na téma bude zabývat technologií databázovou. Zabývat se veškerými technologiemi napříč celou aplikací by bylo mimo rozsah této práce.

2.8.1 Databáze obecně

S nárůstem objemu dat a potřeby jejich rychlého a spolehlivého zpracování narůstala potřeba oddělit vlastní data od programové části aplikace. Za databázi lze označit např. i jednoduchý CSV soubor, ale oproti komplexním databázovým systémům nám nenabídne mechanismy a funkce jako jsou rychlé vyhledávání, správa uživatelů a oprávnění, řešení přístupu více uživatelů, atomičnost transakcí, datové typy, metadata a další.

Komplexní databázový systém bývá označován jako SŘBD (systém řízení báze dat) nebo také DBMS (z anglického DataBase Management System). Pod pojmem databáze se běžně rozumí jak samotná uložená data, tak i software, který nám poskytuje možnosti a funkce popsané výše.

Databáze je organizovaná struktura dat, která je modelem reálného systému. Model používá přípustné abstrakce, což znamená, že se jedná vždy o vztah model : realita – 1 : N prvkům. Databáze lze rozdělit dle typu modelu dat na hierarchické, síťové a relační. Relační model je nejmladší a v současné době asi nejpoužívanější. Data jsou organizována v tabulkách do sloupců a řádků a tyto tabulky mezi sebou tvoří relační vztahy.

Pojmy

- **Data** – údaje nesoucí informaci
- **Datové entity** – objekty v databázi (tabulky, procedury, pohledy, indexy). Datové entity mají také přiřazen datový typ, např. text, celé číslo, desetinné číslo, logická hodnota, datum a čas.
- **Atribut** – Sloupec dané tabulky. Např. Jméno, Příjmení, Tel. číslo, ...
- **Záznam** – jeden řádek tabulky, skládá se z jednotlivých atributů
- **Primární klíč** – unikátní hodnota jednoznačně identifikující daný záznam v rámci tabulky
- **Cizí klíč** – Primární klíč pocházející z jiné tabulky. Tímto je realizováno odkázání na záznam v jiné tabulce, který souvisí s aktuálním záznamem.

Integritní omezení

Úkolem SŘBD je zajistit integritu databáze, tzn. že data uložená v databázi vyhovují definovaným pravidlům (např. že sloupec jméno bude vždy obsahovat text, telefonní číslo bude vždy v daném formátu, rok narození bude vždy číslo větší než 1800 apod.) Integrita se zajišťuje pomocí integritních omezení, která tato pravidla definují.

Vztahy mezi tabulkami

V zásadě se rozišují tyto druhy vztahů:

- **1:1** – záznamu z tabulky A odpovídá právě jeden záznam tabulky B
- **1:N** – záznamu z tabulky A odpovídá N záznamů z tabulky B. Jde o nejpoužívanější typ relace, protože odpovídá vztahům z reálného života
  (např. 1 výrobce vyrábí N výrobků atp.)
- **M:N** – M záznamům z tabulky A odpovídá N záznamů z tabulky B. Tento vztah se nejčastěji modeluje pomocí samostatné tabulky, která nám tuto vazbu rozdělí na 1:N a 1:M. Jako příklad vztahu M:N lze uvést disponenty účtů v bance, kde M účtů má N disponentů. Realizace v samostatné tabulce pak udává N disponentů k danému účtu a naopak M účtů, kterými disponuje daný disponent.

Normální formy

Protože databáze obsahují velké množství dat, vyplatí se jejich strukturu organizovat tak, abychom se vyhnuli ukládání redundantních dat. To pro nás ve výsledku také znamená snazší manipulaci s daty a zachování jejich konzistence. K dosažení tohoto stavu se používájí normální formy, které udávají pravidla, jaká by data v relační databázi měla splňovat.

- **0.NF** – Ve sloupci tabulky se nachází více hodnot.
- **1.NF** – Sloupce obsahují atomické informace, nelze je dále dělit.
- **2.NF** – Každý neklíčový sloupec v tabulky je závislý na primárním klíči.
- **3.NF** – Všechny neklíčové sloupce musí být vzájemně nezávislé.
- **4.NF** – Relace popisují pouze příčinnou souvislosti mezi klíčem a sloupci
- **5.NF** – Relace je již dále nedělitelná, přidání nového sloupce by způsobilo rozpad na dvě tabulky.

ACID

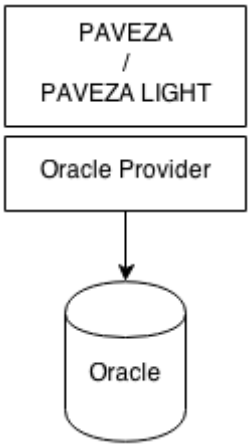
V databázových systémech je ACID mnemotechnický akronym pro vlastnosti, které musí splňovat transakce v databázích

- **A** – *Atomicity* – Atomicita = Transakce musí být nedělitelná, tzn. buď se provede jako celek, nebo se neprovede vůbec.
- **C** – *Consistency* – Konzistence = Transakce neporuší žádné integritní omezení.
- **I** – *Isolation* – Izolovanost = Transakce je izolována od ostatních transakcí, tzn. neovlivňuje ostatní transakce ani není žádnou transakcí ovlivněna.
- **D** – *Duralibility* – Trvalost = Po proběhnutí transakce jsou změny uloženy do databáze a nemohou být již ztraceny.

2.8.2 Oracle

Vůbec prvním zákazníkem a organizací, pro kterou se aplikace PAVEZA (resp. její historický předchůdce JASAN – umožňovala pouze evidenci veřejných zakázek) vyvíjela, byly Lesy ČR. Protože tato organizace měla již zakoupenou licenci na databázový *engine* od firmy Oracle, vyvíjela se aplikace pro něj, aby odpadla nutnost licencovat engine jiný.

Původní verzi resp. způsob komunikace s databází zobrazuje obr. 4/str. 22.



Obr. 4. Původní způsob komunikace s DB

Tab. 3. Srovnání ceny licencí Oracle DB a MS SQL Server [5]

Cena licence pro 4 jádrový x86 CPU	**Oracle DB 11gR2**	**SQL Server 2012**
Enterprise	$95.000	$27.496
Standard	$17.500	$17.142

2.9 Odůvodnění migrace na nové technologie

Podobně jako s případem organizace Lesy ČR může být potenciální zákazník, který již vlastní licenci na jiný databázový engine než Oracle – např. na Microsoft SQL Server. V takovém případě nákup licence Oracle znamená výdaje navíc a zvyšují tak dobu návratnosti investice do aplikace.

Dalším důvodem je vyšší pořizovací cena licence pro Oracle 11g než pro Microsoft SQL Server viz tab. 3/str. 22.

Další problematikou je aplikace PAVEZA LIGHT. Protože se jedná o placenou webovou službu, je zde problém (dle společnosti NWT a.s.) s tzv. „přeprodejem licence“ další straně, která není licenčními podmínkami dovolena. V případě, že aplikace poběží na Microsoft SQL Serveru tento problém odpadá.

2.10 Použité technologie v nové verzi

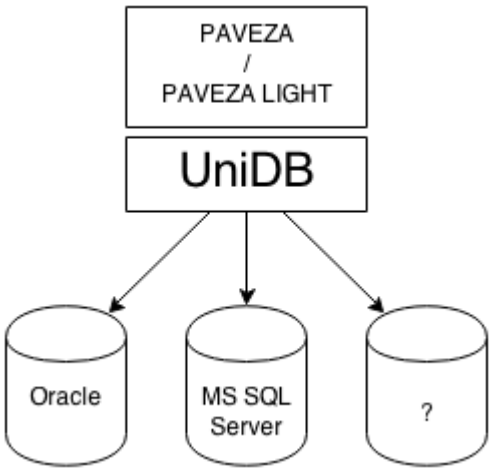
2.10.1 Microsoft SQL Server

Jak je již uvedeno výše, přechod na Microsoft SQL Server se jeví jako výhodný. Stávající zákazníci však používají už engine Oracle a není tedy možné nevratně změnit databázový *interface* přímočaře. Také SQL dotazy nejsou mezi různými databázovými enginy vzájemně kompatibilní. Musí však být podporovány obě řešení.

Protože jde o rozsáhlou úpravu, je vhodné ji navrhnout tak, aby v budoucnu bylo možné jednoduše přidat podporu dalšího databázového engine.

2.10.2 UniDB

Speciálně pro účel podpory různých databázových *enginů* byl firmou NWT a.s. navrhnut *interface* **UniDB**. Tento *interface* umožňuje zapsat dotazy do univerzální podoby, kdy aplikace až za svého běhu dle databázového engine, na kterém běží, rozhodne, jaká bude podoba výsledného SQL dotazu. Způsob funkce **UniDB** ilustruje obr. 5/str. 23.



Obr. 5. Komunikace s DB přes interface UniDB

2.11 Rozdíly mezi Oracle a MS SQL Server

Hlavním rozdílem je způsob, jakým jsou ve *výchozím* nastavení řešeny transakce. U Oracle je vše transakce a změny nejsou permanentní, dokud není spuštěn příkaz `COMMIT` (až na pár výjimek souvisejících s DDL příkazy, u kterých je `COMMIT` implicitní). U SQL Serveru tomu je naopak, tedy neohraničené bloky kódu `BEGIN TRANSACTION` a `COMMIT TRANSACTION` nejsou transakcí a může se tedy stát, že při chybě uprostřed procedury se část kódu vykoná (a změny se uloží) a část kódu zůstane nevykonána. Data pak mohou zůstat v nekonzistentním stavu. [7]

Dalším podstatným rozdílem je MVCC[1)]. Ve *výchozím* nastavení MS SQL Sever dovoluje *dirty reads*[2)] a zápisy mohou blokovat čtení, narozdíl od Oracle, kde se načítají jen trvale uložená data. [11]

Další rozdíly jsou v syntaxi SQL. Základní syntaxe SQL jsou u obou databázových systémů stejné (oba podporují ANSI-SQL code standard), ale liší se deklarace proměnných,

[1)] *Multiversion concurrency control* – metoda řešení konkurenčního přístupu v databázových a programovacích jazycích pro implementaci transakční paměti

[2)] Způsob čtení paměti, kdy načtená data mohou obsahovat řádky uložené aktuálně probíhající transakcí, kdy tato transakce ještě nebyla dokončena.

procedur, pohledů (*view*) a také integrované funkce. Například u MS SQL Serveru lze omezit počet vrácených výsledků pomocí klíčového slova `TOP`. Stejná věc se u Oracle musí řešit pomocí vrácení čísel řádků a jejich omezení v klauzuli `WHERE`. [12]

Výčet rozdílů v SQL lze nalézt v [13] a [14]. Některé z nich ale již nemusí platit a naopak některé mohou chybět, protože tyto zdroje jsou několik let staré. Oracle [15] i Microsoft [16] poskytují migrační nástroje, které dokážou migrovat celé databáze i SQL *queries*. Tato práce se však spíše než na migraci zaměřuje na univerzální použití pro obě (a v budoucnu případně pro více) databázové řešení.

II. PRAKTICKÁ ČÁST

3 POPIS ŘEŠENÉHO PROJEKTU

Aplikace PAVEZA je velice rozsáhlá a její databáze obsahuje mnoho tabulek. S tím je tedy také spjato množství dotazů. Všechny SQL dotazy byly v kódu zapsány v podobě pro Oracle. Bylo tedy nutné tyto dotazy přepsat pro *interface* UniDB. Odhad času na přepsání byl firmou NWT a.s. stanoven přibližně na 500 hodin. V komerční sféře však není snadné zastavit všechen ostatní vývoj (který je nutný) a věnovat se pouze přepisu databázového *interface*. To byl také jeden z důvodů vypsání Inovačního voucheru Zlínského kraje.

3.1 Inovační voucher Zlínského kraje

Inovační voucher je nástroj, který podporuje spolupráci podnikatelských subjektů s vědeckovýzkumnými (dále jen „VaV“) institucemi. Tato spolupráce probíhá jako zakoupení služeb podnikatelským subjektem od VaV institucí, čímž tento subjekt získá vyšší inovační potenciál.

Inovační voucher se vydává v hodnotě 60 – 149 tisíc Kč, který lze využít k nákupu služeb od VaV v hodnotě 80 – 199 tisíc Kč. Podporována je prvotní spolupráce podnikatelského subjektu s konkrétním VaV pracovištěm na daném typu služby pro konkrétní inovaci produktu.

Voucher je financován z Regionálního operačního programu (ROP) Střední Morava a prostředků Zlínského kraje. Jedná se o veřejnou podporu podnikatelskému subjektu poskytovanou v režimu „de minimis“. [8]

> **Podpora de minimis (neboli podpora malého rozsahu)** [9]
>
> De minimis představuje takovou podporu, která nesmí spolu s ostatními podporami „de minimis“ poskytnutými jednomu příjemci za dobu předchozích tří let přesáhnout výši odpovídající částce 200 000 €. Tento finanční strop platí bez ohledu na formu či účel podpory de minimis poskytnuté v předchozím tříletém období. Za tříleté období se považují fiskální roky používané k daňovým účelům.
>
> Podporu de minimis není možno kumulovat s jinou veřejnou podporou na stejné způsobilé výdaje, jestliže by kumulací došlo k poskytnutí vyšší míry podpory než je stanovena dle regionální mapy podpory (resp. v nařízení o blokových výjimkách nebo v rozhodnutí EK).
>
> Poskytovatel podpory de minimis je povinen před jejím poskytnutím písemně sdělit podniku zamýšlenou částku podpory, upozornit jej na charakter podpory, dále je poskytovatel povinen vyžádat si od daného podniku prohlášení o všech dalších podporách de minimis, které tento podnik v předchozích

dvou fiskálních letech a v současném fiskálním roce obdržel. Poskytnutím podpory nesmí být překročen limit podpory de minimis (200 000 €). U Rozhodnutí vydaných do 31. 12. 2009 se pro přepočet používá měnový kurz Evropské centrální banky platný pro aktuální měsíc dle data vydání Rozhodnutí o poskytnutí dotace (tj. kurz platný v den před posledním pracovním dnem v měsíci, který předchází měsíci, ve kterém bylo vydáno Rozhodnutí o poskytnutí dotace).

Od 1. 1. 2010, tedy od počátku fungování Registru de minimis, se pro přepočet použije měnový kurz Evropské centrální banky platný v den vydání Rozhodnutí o poskytnutí dotace. Tento kurz se použije až u Rozhodnutí vydaných po datu 1. 1. 2010, nikoliv zpětně.

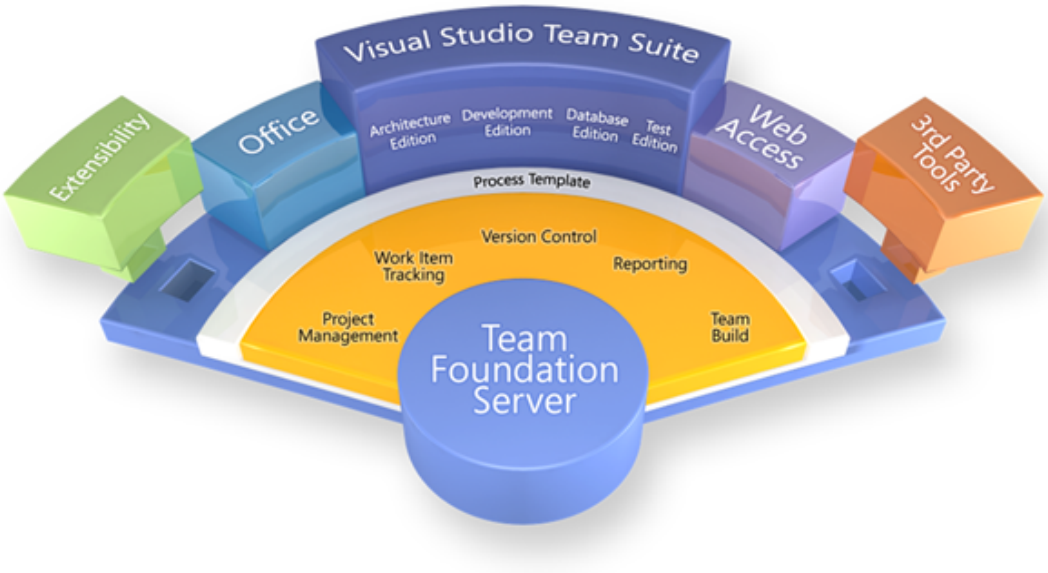
3.2 Seznámení se zdrojovým kódem aplikace

Po úvodní schůzce v sídle NWT a.s. ve Zlíně byly získány přístupové údaje pro Team Foundation Server (TFS). Vrámci programu MSDN AA bylo z portálu DreamSpark staženo Visual Studio 2010 a pomocí klienta TFS stažen repozitář zdrojových kódů. Firma NWT a.s. také poskytla instrukce pro přepis databázových dotazů.

3.2.1 Team Foundation Server

Team Foundation Server (TFS) je nástroj pro týmovou spolupráci při vývoji aplikací a řízení jejich životního cyklu. Poskytuje správu zdrojového kódu, ať už přes Team Foundation Version Control nebo přes Git, správu požadavků přizpůsobený agilnímu i vodopádovému modelu vývoje, správu chyb, automatickou kompilaci sestavení projektu (*build*), správu testů a uzavřených verzí.

TFS využuje celý tým od projektového manažera, přes programátory, testery, architekty až po business analysty. Jak je již zmíněno výše, TFS složí nejen pro správu projektu, ale i pro správu zdrojových kódů. Důležitou funkcí TFS je také schopnost vytvářet reporty o událostech v TFS, což dává okamžitou zpětnou vazbu o stavu projektu.



Obr. 1. Team Foundation Server [17]

4 PŮVODNÍ ŘEŠENÍ DATABÁZOVÉHO INTERFACE

Třídy business logiky projektu **BusinessLogic** komunikovaly s databází přes databázový *interface* implementovaný v třídách `Oracle*Provider` projektu **DataAccess.Oracle**. Např. třída business logiky `ItemManager` využívala pro komunikaci s databází třídu `OracleItemProvider`. Pomocí jejích metod bylo realizováno volání konkrétních databázových dotazů pro databázi Oracle.

Protože metody tříd `Oracle*Provider`, načítající data z databáze, vracely kromě obecných `DataSet` objektů také objekty typu `DB*` (např. v případě třídy `OracleItemProvider` to byl objekt `DBItem`), musely třídy business logiky navíc obsahovat *mappery*, které přemapovaly jednotlivé *property* objektu na *properties* objektu business logiky. Analogicky potom fungoval i zápis do databáze, kde se objekty business logiky mapovaly na objekt třídy `DB*`, se kterým pracoval daný `Oracle*Provider`.

Konkrétní případ metody třídy `ItemManager` ilustruje kód 1/str. 29. Na řádku 7 se získá objekt třídy `OracleItemProvider`, na kterém je volána metoda `GetItem(int)`. Právě ta zprostředkuje získání dat z databáze a sestavení objektu. Na řádku 8 je pak zmíněné mapování na objekt business logiky `Item`.

Kód 1. Příklad komunikace s databázovým interface v třídě `ItemManager`

```
1  class ItemManager
2  {
3      // ...
4      public static Item GetItem(int itemID)
5      {
6          Item oItem = new Item();
7          DBItem dbItem = DBProviderManager<DBItemProvider>.Provider.GetItem(itemID);
8          oItem = DBMapping(dbItem);
9          return oItem;
10     }
11     // ...
12 }
```

Dále si můžeme povšimnout dvou menších nedostatků v kódu, které se vyskytovaly poměrně často. Nejedná se však o kritické výkonové či bezpečností chyby, ale z hlediska zásad čistého kódu by měly být odstraněny [6]. Oba nedostatky se nachází na řádku 6.

Prvním nedostatkem je použití maďarské notace – `'o'` na začátku názvu proměnné `oItem` naznačuje, že se jedná o objekt. Jde o historický pozůstatek z doby, kdy neexistovala silná typová kontrola. Programátoři tak potřebovali berličku, která by jim pomohla si datové typy proměnných zapamatovat a tak si je kódovali do názvu proměnných. Tento způsob někteří programátoři používají dodnes. Jazyk C# je silně typovaný jazyk a pokud bychom se pokoušeli přiřadit do proměnné data jiného typu než je její deklarace, vývojové prostředí detekuje chybu ještě dříve, než spustíme překlad. S ohledem na platnost proměnné 4 řádky máme zajištěno, že vždy uvidíme její deklaraci.

Maďarská notace se dnes nedoporučuje z důvodu, že pokud bychom např. v tomto případě (z jakýchkoliv důvodů) chtěli objekt Item převést na strukturu, již se nebude jednat o referenční typ, ale o typ hodnotový. Pak bychom tedy pro zachování konzistence museli měnit i název proměnné. V dnešní době to sice zvládne vývojové prostředí za nás, ale je zde citelné, že nejde o vhodný přístup.

Druhým nedostatkem je zbytečná inicializace objektu. Každý nadbytečný kód je potencionální prostor pro budoucí zavedení chyby. Pokud by navíc konstruktor třídy `Item` obsahoval časově nákladné operace, má tato inicializace objektu i negativní dopad na výkon přímo úměrný četnosti jejího volání.

Metodu databázového *interface*, kterou volá kód 1/str. 29, ukazuje kód 2/str. 30. Zde se vytvoří spojení pro databázi, sestaví se databázový dotaz, spustí se a načtou se vrácené výsledky, které jsou vráceny jako objekt třídy `DBItem`.

Kód 2. Metoda z databázového interface třídy `OracleItemProvider`

```
1  class OracleItemProvider : DBItemProvider
2  {
3      // ...
4      public override DBItem GetItem(int itemID)
5      {
6          DBItem oItem = new DBItem();
7          using (OracleConnection conn =
8              OracleHelper.CreateConnection(_oracleConnectionString))
9          {
10             StringBuilder sqlPrikaz = new StringBuilder();
11             sqlPrikaz.Append(
12                 "SELECT I.* FROM Items I WHERE I.ID = :ID ");
13
14             OracleCommand dbCommand = conn.CreateCommand();
15             dbCommand.CommandType = CommandType.Text;
16             dbCommand.CommandText = sqlPrikaz.ToString();
17             dbCommand.Parameters.Add(":ID", OracleType.Int32).Value = itemID;
18
19             conn.Open();
20             using (OracleReader dbReader = dbCommand.ExecuteReader())
21             {
22                 if (dbReader.Read())
23                 {
24                     oItem = GetItemFromReader(dbReader);
25                 }
26             }
27             conn.Close();
28         }
29         return oItem;
30     }
31     // ...
32 }
```

4.1 Analýza rozsahu nutných změn

Aby mohla aplikace PAVEZA fungovat s *libovolným*[1)] databázovým enginem, je nutné přepsat všechny volání metod z tříd `Oracle*Provider` tak, aby se místo nich použil *interface* **UniDB**.

Způsob potřebných úprav demonstruje kód 3/str. 31 na příkladu uvedené metody z třídy `ItemManager` resp. `OracleItemProvider`. Místo volání metody pro `OracleItemProvider` a následné mapování objektu je zde přímo implementace dotazu pro *interface* **UniDB**.

Kód 3. Implementace metody třídy `ItemManager` s použitím UniDB

```
1  using UniDB.Base;
2
3  class ItemManager : BaseManager
4  {
5      // ...
6      public static Item GetItem(int itemID)
7      {
8          using (Select dbs = db.NewSelect("Items"))
9          {
10             dbs.Where.AddColEq("ID", itemID);
11             return dbs.FetchObject(new Item()) as Item;
12         }
13
14         #region old code
15         /*
16         Item oItem = new Item();
17         DBItem dbItem = DBProviderManager<DBItemProvider>.Provider.GetItem(itemID);
18         oItem = DBMapping(dbItem);
19         return oItem;
20         */
21         #endregion
22     }
23     // ...
24 }
```

Potřebné úpravy:

- Abychom mohli použít *property* `db`, která obsahuje inicializovaný objekt třídy `Database` z knihovny **UniDB**, musíme na řádku 3 doplnit dědění z třídy `BaseManager`.
- Původní kód zakomentujeme a zabalíme do *regionu* `old code`, abychom jej mohli schovat. Tento krok se může zdát zbytečný – máme přece správu zdrojových kódů, která si historii pamatuje. Protože se ale jedná o přepis celého databázového *interface* napříč celou aplikací, je velice pravděpodobné zavedení chyb. Aby se při ladění a testování v případě chyb nemusel kód dohledávat v historii zdrojových kódů, byl ponechán zde. Tento postup plynul z požadavků firmy NWT a.s. na

[1)] Podpora pro každý nový engine musí být do UniDB doprogramována. Termínem *libovolný* je myšlen kterýkoliv databázový engine založený na relační SQL databázi, který splňuje nároky aplikace.

přepis. Poté, co testování bude ukončeno a funkčnost ověřena, lze díky pojmenovanému *regionu* najít snadno všechny výskyty a kód odstranit.

- Podle kódu v `OracleItemProvider` zapíšeme nový kód, který bude provádět stejný dotaz na databázi přes **UniDB**. Použitím příkazu `using` zajistíme, že vytvořený objekt třídy `Select` bude zrušen na konci těla tohoto příkazu. Tímto se vyhneme potenciálnímu zablokování databázového spojení. Objekt `Select` je vytvořen tovární metodou `NewSelect` z třídy `Database`. Jako parametr tato metoda přejímá název tabulky, nad kterou se dotaz volá. Na vytvořeném objektu pak zavoláme na *property* `Where` metodu `AddColEq(string, object)`, která nám reprezentuje SQL `WHERE` klauzuli ekvivalentní k SQL podmínce na řádku 12 v kódu `OracleItemProvider` (viz kód 2/str. 30).

  Následně jsou pomocí metody `FetchObject(object)` třídy `Database` získána data, která jsou naplněna přímo do instance předaného objektu třídy `Item`. Jak si můžeme povšimnout, není zde již zapotřebí *mapper*. Vše se děje za pomocí *properties* opatřených *atributy* (viz níže). Protože metoda vrací instanci třídy jako datový typ `object`, musíme vrácený objekt přetypovat. V porovnání s původním kódem `ItemManager` a `OracleItemProvider` je přepsaný kód používající **UniDB** mnohem komplexnější a kompaktnější, což mimo jiné znamená lepší čitelnost a menší prostor pro chyby.

- Naplnění vrácených dat do objektu se děje pomocí atributů, kterými jsou opatřeny *properties* této třídy. Každý sloupec v tabulce má stejnojmenný protějšek jako *property* dané třídy. Aby knihovna **UniDB** byla schopna detekovat, které *property* patří sloupcům, je potřeba každou tuto *property* označit atributem `[DBColumn()]`. V parametru atributu lze případně specifikovat, zda sloupec obsahuje primární klíč, zda jde o cizí klíč – a z jaké tabulky, zda může nabývat hodnotu `NULL`, délku řetězcového datového typu apod.

- Na závěr je zapotřebí v přepsané metodě z třídy `OracleItemProvider` doplnit řádek, který vyhodí výjimku, aby se při případném použití této metody poznalo, že již je přepsaná a nemá se tedy používat (viz kód 4/str. 33).

Kód 4. Úprava metody třídy `OraceItemProvider` po přepsání na volání UniDB

```
1  class OracleItemProvider : DBItemProvider
2  {
3      // ...
4      public override DBItem GetItem(int itemID)
5      {
6          throw new Exception("Převedeno");
7
8          // následuje původní kód metody
9      }
10     // ...
11 }
```

Všechny dotazy použité v třídách projektu **DataAccess.Oracle** bylo zapotřebí přepsat podobně jako je demonstrováno na příkladu popsaném výše. Celkově tyto třídy měly 35 921 řádků. Samozřejmě se jedná o počet řádků včetně deklarací, C# kódu a komentářů. Řádky, které obsahují znak ", tvoří cca 40% z celkového počtu řádků. Jedná se však o použití v komentářích i řetězcových konstantách mimo vlastní SQL dotazy. Zde se tedy můžeme dostat odhadem na 30% celkového počtu řádků, což nám dává přibližně 10 776 řádků SQL kódu, který byl potřeba přepsat pro *interface* **UniDB**. A konečně také, jak již bylo zmíněno ve výčtu potřebných úprav, bylo zapotřebí doplnit všem třídám, které měly svůj ekvivalent v databázi, atributy.

5 IMPLEMENTACE DATABÁZOVÝCH DOTAZŮ POMOCÍ UNIDB

Předchozí kapitola nám na demonstračním příkladu ukázala, jak je potřeba postupovat při přepisování jednoduchého databázového dotazu. Analogickým způsobem lze zapsat veškeré ostatní databázové dotazy s použitím knihovny **UniDB**. Bohužel zde nelze zveřejnit konkrétní detaily a příklady složitějších databázových dotazů s ohledem na uchování *know-how* firmy NWT a.s.

Obecně lze alespoň vypíchnout například použití *fluent-interface* v knihovně **UniDB**. *Fluent-interface* je implementace rozhraní objektu, které dovoluje mimo jiné řetězit volání metod na objektu tak, aby výsledek byl dobře čitelný a blížil se tak v rámci možností přirozenému jazyku. V případě **UniDB** se způsob zápisu blížil syntaxi jazyka SQL. Příklad *podobného* zápisu jako v knihovně **UniDB** ukazuje kód 5/str. 34 na řádku 2.

Kód 5. Příklad zápisu SQL JOIN

```
1    var dbs = db.NewSelect("table1 T1");
2    dbs.Join("table2 T2", JoinType.Left).On(T1.column1).ColEq(T2.column2);
```

5.1 Návrhy doplnění funkcionality

Během přepisu se vyskytly situace, kdy knihovna **UniDB** nenabízela potřebnou funkcionalitu (protože se postupně vyvíjela zároveň s přepisem aplikace) pro zapsání původního kódu `Oracle*Provider`. Byl to např. vnořený `SELECT`, kdy se vybírají záznamy nikoliv z tabulky, ale z dalšího vnitřního *selectu*, doplnění podpory pro `DISTINCT`, řešení pro složené podmínky a pro podmínku `BETWEEN` atd.

Dalším případem byly například integrované funkce, která se liší dle použitého databázového *engine*. Např. volání Oracle SQL funkce `NVL(column, 0)`, která v případě, že sloupec `column` má hodnotu `NULL` vrací `0`, má ekvivalent pro MS SQL Server v integrované funkci `ISNULL(column, 0)`.

Tento případ se řešil tak, že se ve *fluent* zápisu použila funkce `Func("NPV")` respektive `Func("ISNULL")`, která vždy dle aktuálně použitého databázového *engine* použila správnou SQL funkci, bez ohledu na předaný název platformě závislého názvu funkce.

Také byla navržena změna deklarace funkce realizující `JOIN`. Zde byla implicitní hodnota parametru určujícího typ spojení `JoinType.Left`. Toto je opět mimo zásady čistého kódu [6]. Funkce by měla dělat přesně to, co od ní programátor očekává. V SQL je implicitní typ `INNER JOIN` pokud není uvedeno jinak. Zde, když programátor znalý SQL neuvede typ spojení, bude výsledkem `LEFT JOIN`, což zajisté nebude to, co programátor od funkce očekával.

6 KONTROLA FUNKCE A PODPORA ODEVZDANÉHO KÓDU

Po odevzdání kódu probíhalo testování aplikace na straně NWT a.s. Dle dohod by případné logické chyby v odevzdaném kódu byly opraveny v nejkratším možném termínu jakou součást této práce. Testování proběhlo v pořádku až na pár překlepů v řetězcových konstantách názvů tabulek či podmínek pro spojování (`JOIN`) tabulek, které nemohl detekovat kompilátor. Tyto překlepy však firma NWT a.s. nezasílala k opravě, protože režie spojená s požadavkem na opravu překlepu (kontaktování, informace kde se chyba nachází, stažení aktuálních zdrojových kódů, oprava, *check-in*) je nákladnější než okamžitá oprava na straně firmy samotné.

ZÁVĚR

Cíle inovačního voucheru se podařilo splnit v zadaném termínu. Hrubý čas migrace aplikace činil 4 měsíce. Aplikace PAVEZA nyní disponuje novým databázovým *interface*, který poskytuje knihovna **UniDB** pomocí níž jsou skládány databázové dotazy. V sočasné době *interface* podporuje databázový *engine* Oracle 11g a Microsoft SQL Server 2012. V případě potřeby lze podporu dalšího databázového *interface* snadno implementovat za zlomek nákladů bez nutnosti zasahovat do všech databázových dotazů napříč aplikací. Doba implementace podpory dalšího databázového *engine* (např. MySQL) je firmou NWT a.s. odhadována na cca šest týdnů.

Podpora databází Oracle a Microsoft SQL Server zvýšila konkurenceschopnost produktu PAVEZA na trhu. Zákazník má nyní možnost zvolit si, na kterém databázovém *engine* apliakce poběží. To je výhodné v případech, kdy zákazník má již zakoupenou licenci pro nějaký *engine* – nemusí tedy kupovat licenci další. Pokud má zákazník licenci na dosud nepodporovaný databázový *engnine*, může firma NWT a.s. relativně snadno jeho podporu dopracovat.

Tato práce se bohužel nemohla věnovat praktické části v maximálním možném rozsahu s ohledem na *know-how*, které by se odevzdáním této práce zveřejnilo a mohlo by tak poškodit firmu NWT a.s. a tímto snížit její konkurenceschopnost na trhu, což by byl pravý opak cíle inovačního voucheru. Ze stejného důvodu také zdrojové kódy nejsou součástí této diplomové práce. Úryvky kódu, které se nachází v této práci, jsou záměrně vybrány tak, aby se jednalo o triviální případy, které neukazují složitější technické řešení. Některé řádky kódu jsou záměrně pozměněny. Jako doklad o provedené implementační práci slouží předávací protokol, který byl převzat vedoucím této práce Ing. Bc. Pavlem Vařachou PhD.

Za odvedenou práci na inovaci aplikace PAVEZA bylo možné získat studijní stipendium a zároveň získala finanční příspěvek i Fakulta aplikované informatiky UTB ve Zlíně.

SEZNAM POUŽITÉ LITERATURY

[1] SHARP, John. *Microsoft Visual C# 2010: krok za krokem.* Vyd. 1. Brno: Computer Press, 2010, 696 s. ISBN 978-80-251-3147-3.

[2] NWT a.s.: O společnosti. *NWT a.s.* [online]. [cit. 2014-04-18]. Dostupné z: http://www.nwt.cz/lang_cs/clanek/4/5.html

[3] NWT A.S.: *Výroční zpráva NWT a.s. za rok 2012.* 2013, 42 s.

[4] NWT A.S.: *PAVEZA & EVEZA.* Zlín, 2013, 5 s.

[5] MICROSOFT. *SQL Server 2012 Licensing Value vs. Oracle Database.* 2012, 10 s. Dostupné z: http://resources.whymicrosoft.com/ResourceDetail?Title=SQL+Server+2012+Licensing+Value+vs.+Oracle+Database

[6] MARTIN, Robert C. *Čistý kód.* Vyd. 1. Brno: Computer Press, 2009, 423 s. ISBN 978-80-251-2285-3.

[7] LACKO, Luboslav. *Oracle: správa, programování a použití databázového systému.* 2. dopl. vyd. Překlad Marek Kocan. Brno: Computer Press, 2007, 576 s. ISBN 978-80-251-1490-2.

[8] Regionální rada regionu soudržnosti Střední Morava: Inovační vouchery. *Regionální rada regionu soudržnosti Střední Morava* [online]. 2009 [cit. 2014-04-28]. Dostupné z: http://www.rr-strednimorava.cz/rop-sm/inovacni-vouchery

[9] CzechInvest: Veřejná podpora a de minimis. *CzechInvest* [online]. 2007 [cit. 2014-04-28]. Dostupné z: http://www.czechinvest.org/verejna-podpora

[10] Portál o veřejných zakázkách a konsesích. *Portál VZ* [online]. 2006 [cit. 2014-04-30]. Dostupné z: http://www.portal-vz.cz/cs/Jak-na-zadavani-verejnych-zakazek/Legislativa-a-Judikatura/Legislativa/Narodni-legislativa-aktualni-a-uplne-zneni-z-(1)/ZVZ

[11] Segue Technologies: Microsoft SQL Server vs. Oracle: The Same, But Different?. In: *Segue Technologies* [online]. 2014 [cit. 2014-05-01]. Dostupné z: http://www.seguetech.com/blog/2014/03/13/Microsoft-SQL-Server-versus-oracle

[12] Stack Overflow: Basic differences between Oracle and SQL Server. In: *Stack Overflow* [online]. 2008 [cit. 2014-05-01]. Dostupné z: http://stackoverflow.com/questions/2322260/basic-differences-between-oracle-and-sql-server

[13] SQL Syntax differences between Oracle and MS-SQL. *Oracle Consulting, Oracle Support and Oracle Training by BC Oracle Consulting* [online]. 2006 [cit. 2014-05-02]. Dostupné z: http://www.dba-oracle.com/oracle_news/2005_12_16_sql_syntax_differences.htm

[14] Convert SQL Server T-SQL to Oracle PL/SQL. *Oracle Consulting, Oracle Support and Oracle Training by BC Oracle Consulting* [online]. 2009 [cit. 2014-05-02]. Dostupné z: http://www.dba-oracle.com/t_convent_sql_server_tsql_oracle_plsql.htm

[15] Oracle: Database Migration Technology. *Oracle* [online]. 2012 [cit. 2014-05-02]. Dostupné z: http://www.oracle.com/technetwork/database/migration/index-084442.html

[16] Microsoft: SQL Server Database Migration Assistant. *Microsoft* [online]. 2012 [cit. 2014-05-04]. Dostupné z: http://www.microsoft.com/sqlserver/cs/cz/product-info/migration.aspx

[17] Online-TFS Home: Team Foundation Server. *Online-TFS Home* [online]. 2009 [cit. 2014-05-04]. Dostupné z: http://www.online-tfs.com/Default.aspx?pagename=tfs&lng=english

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ANSI	*American National Standards Institute*
BPS	Bioplynová stanice
CPU	*Central Processing Unit*
DB	Databáze
DBMS	*DataBase Management System*
DDL	*Data Definition Language*
EK	Evropská komise
HW	*Hardware*
IT	Informační technologie
LED	*Light-Emitting Diode*
MS	Microsoft
MSDN AA	*Microsoft Developer Network Academic Alliance*
MVCC	*Multiversion concurrency control*
NF	Normální forma
NIPEZ	Návrh modelu národní infrastruktury pro elektronické zadávání veřejných zakázek
NN	Nízké napětí
PD	Pasivní dům
ROP	Regionální operační program
SQL	*Structured Query Language*
SŘBD	Systém řízení báze dat
SW	*Software*
TFS	*Team Foundation Server*
UTB	Univerzita Tomáše Bati
VN	Vysoké napětí
VUT	Vysoké učení technické
VZ	Veřejná zakázka
VZMR	Veřejná zakázka malého rozsahu

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: DVD

- Diplomová práce elektronické podobě (PDF)
- Text v diplomové práce v LaTeXu
- Příloha Inovační voucher Zlínského kraje (PDF)

PŘÍLOHA P II: INOVAČNÍ VOUCHER ZLÍNSKÉHO KRAJE

viz přiložený dokument v deskách práce