VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

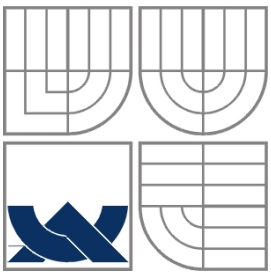
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROGRAMOVÁ PODPORA PRO VÁŽNÍ SYSTÉM

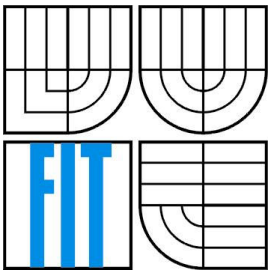
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE Milan Meisl
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROGRAMOVÁ PODPORA PRO VÁŽNÍ SYSTÉM

WEIGHTING SYSTEM PROGRAM SUPPORT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Milan Meisl
AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Chmelař
SUPERVISOR

BRNO 2008

Abstrakt

Tématem bakalářské práce je programová podpora pro vážní systém. Hlavním cílem práce je vytvořit aplikaci, jež je schopna rychle a efektivně zaznamenávat data o navážených hodnotách. Dílčím cílem práce je naprogramovat komunikační modul programu pro komunikaci váhy s osobním počítačem. Aby proces vážení mohl probíhat dostatečně rychle, je nutné zajistit jeho automatické vykonávání. Aplikace je naprogramována prostřednictvím moderních programovacích technologií.

Klíčová slova

silniční vážní systémy, automatické zpracování vážních dat, MS SQL 2005, C#, platforma dotNET, RS 232

Abstract

Topic of the bachelor's thesis is the weighting system programme support. The main aim of the thesis is to create an application that is able to record data on weighted elements quickly and effectively. Partial aim of the thesis is to programme a communicative module of a programme for communication between the scale and the personal computer. In order to make the weighting process quick enough, it is necessary to ensure its automatic running. The application is programmed by means of modern programming technologies.

Keywords

transport weighting system, automatic processing of weighted data, MS SQL 2005, C#, platform dotNET, RS 232.

Citace

Meisl Milan: Programová podpora pro vážní systém. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Programová podpora pro vážní systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Chmelaře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Milan Meisl
11. května 2008

Poděkování

Velmi rád bych poděkoval Ing. Petru Chmelařovi za jeho ochotu, obohacující připomínky a přínosná doporučení.

Obsah

Úvod

Tématem bakalářské práce je programová podpora pro vážní systém. Toto téma jsem si vybral proto, že většina softwaru, který se dnes pro tento účel využívá, je již zastaralá a nemoderní. Rád bych tedy alespoň částečně přispěl ke zkvalitnění softwaru v dané oblasti.

Hlavní cíl práce spočívá ve vytvoření aplikace, jež by byla schopna rychle a efektivně zaznamenávat data o navážených hodnotách. Doba, která je potřeba ke zvážení jedné položky, musí byt co nejkratší, a to i při zadávání velkého množství důležitých údajů potřebných k vedení vážního deníku. Výhody softwaru, který se v současnosti používá, lze spatřovat nejen v jeho jednoduchosti a rychlosti, ale také ve skutečnosti, že k jeho ovládání není potřeba polohovací zařízení myš. Uživatel tak pomocí klávesnice a klávesových zkratek dokáže poměrně rychle zadávat a vybírat všechny hodnoty. Jak již však bylo předestřeno, hlavní nevýhodou softwaru je jeho zastaralost, která neumožňuje příjemné uživatelské rozhraní.

Dílčím cílem bakalářské práce je naprogramovat komunikační modul programu pro komunikaci váhy s osobním počítačem (PC). Aby vážení mohlo probíhat dostatečně rychle, je nutné, aby bylo vykonáváno automaticky.

V rámci bakalářské práce jsem se zaměřil na vývoj softwaru, který bude grafickou aplikací podobný dnešnímu standardu, na který jsou uživatelé osobních počítačů zvyklí. Aplikace by měla být naprogramována prostřednictvím moderních programovacích technologií.

Bakalářská práce je rozčleněna do dvou hlavních částí – na část teoretickou a praktickou. Předmětem první kapitoly teoretické části jsou silniční vážní systémy. Kapitola se zabývá základním rozdělením vah, přičemž hlavní důraz je kladen na kategorii silničních vah, jejich vývoj a přínos. Druhá kapitola se věnuje objektově orientovanému programování. Po nastínění problematiky daného přístupu k programování jsou definovány jeho nejdůležitější pojmy. V pořadí třetí kapitola představí základní definici databází, jejich vývoj, stěžejní pojmy a architekturu databázových systémů. Tématem čtvrté kapitoly je platforma RS232.

Praktická část práce se soustředí na architekturu návrhu aplikace. Po popisu procesu vytvoření databáze následuje detailní analýza tvorby samotné aplikace. Její klíčové prvky tvoří strukčný popis aplikace, přihlášení do aplikace, hlavní okno aplikace, záložkové menu, rychlá nápověda, nastavení aplikace a komunikace s váhou. Závěrečná kapitola následně obsahuje zhodnocení dosažených výsledků.

1 Teoretický rámec práce

Předmětem první kapitoly teoretické části jsou silniční vážní systém, jejich zařazení do kategorie vah, vývoj a přínos. Druhá kapitola se zabývá objektově orientovaným programováním. Po nastínění problematiky daného přístupu k programování následuje definice jeho základních pojmů. Třetí kapitola představí definici databází, jejich vývoj, stěžejní pojmy a architekturu databázových systémů. Tématem čtvrté kapitoly je platforma RS232.

1.1 Vážní silniční systémy

1.1.1 Zařazení do kategorie vah

Silniční váhy se řadí do průmyslových vah. Průmyslové váhy jsou definovány jako *„přístroje, které slouží k měření hmotnosti definovaně ohraničených objektů vyjádřené v g, kg, t a dále k průběžnému měření hmotnosti toku přepravovaných materiálů za jednotku času vyjádřené v jednotkách přepravního výkonu hmotnostních (g.h-1, kg.h-1, t.h-1) nebo objemových (l.h-1)“*.[4] Jak bude rozebíráno později, průmyslové váhy v současnosti neslouží již pouze k zjištění hmotnosti či přepravního výkony, ale také mj. také k modernizaci logistických a dalších technologických procesů. Díky svému napojení na automatizační a počítačovou techniku je lze využít k mnoha rozmanitým účelům.

Na základě realizace vážení lze rozlišit dva typy vážících technik: váhy diskontinuální a váhy kontinuální. Zatímco diskontinuální váhy jsou známy a využívány již velmi dlouhou dobu, druhý typ vah je novější, neboť jeho vznik se datuje do poloviny 19. století. Kontinuální váhy tvoří:

- Průtokoměry sypkých hmot
- Dávkovací pásové váhy
- Diferenciální dávkovací váhy
- Elektronické řídicí jednotky pro kontinuální vážení.
- Elektronické řídicí jednotky pro dávkování

Mezi diskontinuální váhy patří:

- Váhy pro silniční vozidla
- Váhy pro kolejová vozidla
- Zásobníkové váhy
- Váhy ve válečkových tratích
- Jeřábové váhy

Podle kritéria účelu vážení lze diskontinuální a kontinuální váhy rozdělit do dvou skupin: na váhy technologické a váhy určené k obchodnímu styku. Jeden z rozdílů mezi nimi lze nalézt v míře přesnosti jejich vážení. Zatímco u technologických vah je nejvyšší povolená hodnota možné chyby definována relativní chybou vztaženou k rozsahu nebo k okamžité hodnotě vážení a pohybuje se v rozsahu 0,1 až 5 %, u vah používaných v obchodním styku, ekologii, zdravotnictví a v některých dalších oborech se musí pro přesnost vážení dodržovat národní a evropské normy doporučené organizací OIML (Organisation de Métrologie Légale).[4]

Silniční váhy spadají do kategorie diskontinuálních vah. Vyrábí se dva základní typy vah pro silniční vozidla: úrovňové a nájezdové. Úrovňové (jinak též zapuštěné) váhy považují mnozí odborníci za vhodnější; rozměry vážních mostů se u nich pohybují v rozmezí 8 × 3 m až 22 × 3 m, přičemž za evropský standard se považuje rozměr 18 × 3 m. Váživost je volena v rozmezí 30 až 60 tun. Přesnost statického vážení podle normy obchodního vážení ČSN EN 45501 splňuje požadavky třídy III s dělením na 3000 dílků.[5] Obecně lze tvrdit, že se silniční váhy považují za méně přesné a ponejvíce se využívají ke kontrole osových nápravových tlaků. Vážní mosty i základové vany se zpravidla odlévají betonu jako prefabrikáty; jejich elektronika je většinou vybavena vyhodnocovací jednotkou s osobním počítačem.[4]

Vozidla lze vážit za klidu či za jízdy. Výzkumy poukazují na skutečnost, že váhy pro vážení silničních vozidel za pohybu bývají méně přesné a ačkoli se původně očekávalo, že proces vážení výrazně urychlí, děje se tak spíše pouze ve výjimečných případech.

Dalším typem vah jsou digitální silniční mostové váhy, které jsou dostupné také již na českém trhu. Lze je definovat následujícím způsobem: *„U digitální technologie probíhá digitalizace signálu již na úrovni jednotlivých snímačů (převodník A/D je zabudován v tělese každého snímače). Digitální signál snímačů lze pak již přímo zpracovávat na běžném personálním počítači PC (bez nutnosti začlenění klasické vyhodnocovací/indikační jednotky V/IJ do řetězce vážního systému).“* [18]

Další typologie silničních vah může být následující:

- Silniční mostové váhy s železobetonovým mostem s prefabrikovaným základem/ v nájezdovém i úrovňovém provedení
- Silniční mostové váhy s ocelovým mostem
- Silniční dynamické váhy pro vážení vozidel za jízdy (ke zjištění nápravových zatížení i celkové hmotnosti)
- Mobilní přenosné vážící plošiny pro zjišťování kolových/nápravových/podvozkových zatížení i celkové hmotnosti.

1.1.2 Vývoj silničních vážních systémů

Vážní systémy jsou v současné době na velmi vysoké úrovni; mnohé využívají moderní technologii dynamického vážení vozidel za jízdy s předvýběrem potenciálně přetížených vozidel a jejich

následným převážením na cejchované silniční váze.[6] Jejich modulárnost a široký výběr doplňkových zařízení umožňují přizpůsobit se konkrétnímu účelu a potřebám vážení.

Počátkem 80. let 20. století byla publikována teorie, že účinek zatížení na komunikaci vyvolaný přejezdem jednoho těžkého nákladního vozidla odpovídá přibližně účinku, který by vyvolalo 20 000 přejezdů vozidel osobních.[6] Současné výzkumy závěry této teorie potvrzují. Podle odhadů jezdí v České republice více než 20 % přetížených nákladních vozidel, což je třikrát více než ve vyspělých zemích EU. Většina evropských zemí disponuje koncepcemi na budování automatických systémů sběru dopravních informací.

V naší zemi se s vážením silničních vozidel lze setkat od přelomu 70. a 80. let 20. století. První projekt, v rámci něhož byly na odpočívkách silnic 1. třídy a dálnici D1 vybudovány vážící místa se žlaby na uložení přenosných vážících plošinek pro statickou kontrolu zatížení nápravy, však skončil – v důsledku podcenění přípravy vážících zón – neúspěšně. [22] V současné době lze již v ČR vážit vozidla i za jízdy; vývoj je ovšem v tomto ohledu pomalejší než ve vyspělejších zemích (Švýcarsko, Rakousko či Francii) a systém vážení se potýká s celou řadou těžkostí. Na rozdíl od zmíněných zemí v ČR např. stále chybí jednotný řídicí systém pro vážení přetížených vozidel, který by využíval moderní informační techniku.

V České republice se počet vážních zařízení každoročně zvyšuje. Na konci roku 2005 byly evidovány následující počty: Ministerstvo dopravy 15 vážních zařízení, Česká policie 11, celní správa 17, kraje (Správa a údržba silnic) 5, město Lovosice 1, což dohromady činí 49 vážních zařízení.[3] Převážná většina přenosných stacionárních vah výrobkem švýcarské firmy HAENNI.

1.1.3 Důvody a možnosti použití vážních systémů

Silniční váhy se uplatňují v mnoha oblastech lidské činnosti. Taktéž platí, že podíváme-li se na strukturu potenciálních uživatelů informací generovaných vážními systémy, zjistíme, že je velmi široká (dopravní policie, silniční databanky apod.) Na základě zákona o pozemních komunikacích mohou kontrolní vážení a měření na dálnicích, silnicích a místních komunikacích provádět i celníci.

Důvodů, proč se vážní systémy používají, lze vysledovat několik. První z nich spočívá ve snaze eliminovat přetěžování silničních vozidel a chránit tak pozemní komunikace, a to zejména s ohledem na dlouhodobě se zvyšující pohyb a míru nákladní dopravy na českých silnicích. Podle odhadů vzrostl objem nákladní dopravy v letech 1990 až 2005 o 177%; v letech 1995 až 2000 o 40 %. V roce 2003 činil podíl nákladní dopravy u silniční dopravy 74 %, železniční 25 % a letecké a vodní dopravy dohromady 1 %.[8]

Následkem přetěžování náprav silničních vozidel dochází k postupné degradaci silničních komunikací a jejich konstrukcí (jako jsou např. tunely či mosty) a snižování jejich životnosti. Kontrolní vážení a měření se zpravidla soustředí na celkovou hmotnost vozidla (tj. i se soupravou),

kontrolu nápravových tlaků a vnější rozměry vozidla a nákladu. V současnosti lze již zaznamenat některé pozitivní trendy. Např. v kraji Vysočina bylo v období od ledna 2006 do listopadu 2006 evidováno 129 přetížených vozidel, což představuje 5,54 % z celkového počtu vážených vozidel. V porovnání s rokem 2005, kdy bylo zváženo celkem 1938 vozidel a 8,77 % z nich bylo přetíženo, se jedná o 16,72 % nárůst v počtu zvážených vozidel a naopak o 3,23 % snížení přetížených vozidel.[11]

Váhy jsou také využívány ke sběru údajů o aktuálním dopravním proudu vozidel, které se následně uplatňují v oblastech, jako je organizace a řízení dopravy, dopravní inženýrství, systém hospodaření s vozovkou (*Pavement Management System*), dimenzování vozovek a mostů či dokonce dopravní statistika (např. periodicky se opakující celostátní sčítání vozů). Díky vážním systémům lze analyzovat a komplexně registrovat proud vozidel na dané silnici nebo zjistit skutečné zatížení silniční komunikace.

Na základě úplného či alespoň částečného vyhodnocení informací lze aktuálně a v reálném čase ovlivňovat proud vozidel (tzv. liniová a plošná koordinace, např. zabránění tvoření přílišných front či odklon dopravy). Pro tuto činnost se vžily pojmy telematika, organizace a řízení dopravy či anglický *Inteligent Transport System*. Vážní systémy nacházejí uplatnění i v rámci mechanismu na výběr mýtného *Electronic Toll Collection*, který spravuje poplatky za užívání silniční komunikace v závislosti na celkové hmotnosti vozidel a ujeté dráze.

V neposlední řadě přispívají silniční váhy ke zvýšení bezpečnosti silničního provozu. Získaná on-line data mohou být doplněna kontrolním programem, jenž okamžitě upozorní na potenciálně přetížené vozidlo, zaregistruje jej a informaci odešle příslušnému orgánu (řídící dopravní centrále). Stanice WIM jsou totiž napojeny na všechny druhy dopravních central (městské, regionální, dálniční či zimní údržba). WIM de facto představují *„indukční smyčky, doplněné jedním (vedlejší komunikace) nebo dvěma (hlavní komunikace) senzory pro zjišťování hmotností vozidel (kol, náprav). Hlavním výhoda tohoto uspořádání tkví v možnosti dokonalého rozlišení jednotlivých tříd (kategorií) vozidel. Vozidla je možno rozlišit dle počtu náprav, uspořádání náprav, jejich odstupů, hmotností, a celé řady dalších kritérií.“*[6]

Informace o dopravním proudu vozidel mohou být přenášeny buď on-line, anebo ve formě vyhodnocených souborů (špičkové intervaly či rychlostní intervaly).[6] Po automatickém (proměnná dopravní signalizace) či manuálním (policie) vyřazení nebezpečného vozidla z provozu následuje kontrola hmotností vozidla na cejchované váze (stabilní nebo přenosné).[6]

1.2 Objektově orientované programování

1.2.1 Definice základních pojmů

Pod pojmem objektově orientované programování (často se používá zkratka OOP, z anglického Object-Oriented Programming) se skrývá metodika vývoje softwaru, jejíž počátky spadají do 70. let 20. století. Jedná se o novou metodu vytváření programů, která doplňuje klasickou metodu strukturovaného programování. V rámci strukturovaného programování se využíval postup odshora dolů, kdy se jednotlivé celky rozčleňovaly na stále nižší úrovně. Objektově orientované programování postupuje opačně (tedy odzdola nahoru): základní prvky se nejprve definují a až poté se z nich postaví celý program. Programátoři při použití této techniky nedefinují pouze typ struktury dat, nýbrž rovněž typy operací (funkcí), která na strukturu dat mohou být aplikována.

Objektově orientované programování modeluje reálnou skutečnost a napodobuje vzhled a chování objektu ve skutečném světě. Jeho hlavní přínos spočívá v zajištění možnosti větší strukturovanosti a modularity programu a rovněž ve vysoké znuvupoužitelnosti objektů.

Jako objekt označujeme jednotlivé datové prvky modelované reality. Objekt je definován jako základní, jedinečná a jednoznačně identifikovatelná entita, která je dána identitou (parametry, které odlišují objekt od ostatních objektů) a chováním (služby, které objekt poskytuje vzhledem ke svému okolí).[1] Jedná se tedy o jistou skutečnost, o níž uchováváme data (např. firmu či osobu), a operace pro práci s danými daty. Každý objekt definuje svou vlastní datovou strukturu a algoritmy, má své vlastnosti, může vykonávat určité činnosti a spadá do nějaké třídy objektů. Třída zahrnuje množinu činností, které jsou společné všem objektům dané třídy, takže jednotlivé objekty stejné třídy se od sebe navzájem odlišují pouze svými vlastnostmi/atributy.[13] Pojem instance znamená volání třídy.

Obecně se rozlišují dva typy metod: metody objektů a metody tříd. Zatímco metody objektů se vztahují vždy k danému objektu, metody tříd zahrnují všechny objekty v příslušné třídě. Mezi speciální metody třídy patří konstruktor a destruktor. Destruktor poté, co je existence objektu ukončena, provede jeho zrušení. Konstruktor se naopak volá při vytváření objektu a slouží k jeho inicializaci. Konstruktor může být rovněž i metodou objektu, a to tehdy, je-li nový objekt tvořen dle vlastností jiného konkrétního objektu. Konstruktor vykazuje následující vlastnosti:

- Každá třída obsahuje nejméně jeden konstruktor.
- Jméno konstruktoru je vždy shodné se jménem třídy.
- Konstruktor automaticky vrací objekt, který je instancí uvedené třídy.[1]

U objektově orientovaného programování je dalším důležitým termínem pojem abstrakce. Tento termín odkazuje na unikátní skutečnost, že v rámci objektově orientovaného programování objekty pracují, aniž by bylo nutné znát vnitřní mechanismy jejich práce.

1.2.2 Základní vlastnosti

Objektově orientovaný přístup tvorby programu je charakterizován třemi základními vlastnostmi:

- Obalení (zapouzdření)
- Dědičnost
- Polymorfismus

1.2.2.1 Obalení

Obalení (anglicky encapsulation) zabezpečuje komunikaci mezi objekty pomocí zpráv. Tento princip je realizován prostřednictvím objektu, jenž vznikl pomocí kombinace standardního datového typu záznam (record) a datového typu procedura (funkce). Kromě datových položek obsahuje tento princip také řídící struktury/chování (metody). Obalení tedy označuje proces, kdy jsou datové položky nebo datové struktury (představující prvek reálného světa) obaleny metodami, jež umožňují přistoupit k datům objektu. Objekt navenek zpřístupňuje pouze své rozhraní, pomocí kterého se s ním následně pracuje, čímž je zaručena určitá míra konzistence.

1.2.2.2 Dědičnost

Při programově orientovaném přístupu jsou jednotlivé objekty organizovány stromovým způsobem. Tento způsob umožňuje, aby objekty mohly dědit vlastnosti jiného druhu objektů, a určuje vztahy mezi nižší a vyšší třídou. Jinými slovy dědičnost umožňuje objektu získat strukturu lokálních dat a možné operace s nimi od jim nadřazených objektů.[1] Děje se tak pomocí již zmíněného dělení objektů do tříd, přičemž každá třída může (přímo či nepřímo) dědit od jiné třídy. Dědící třída pak se nazývá podtřídou; původní třída rodičovskou třídou. Praktický význam této vlastnosti je, že programátoři mohou daný kód vícenásobně použít a vytvářet hierarchie objektů, a tím zvyšovat či snižovat jejich specializaci.

1.2.2.3 Polymorfizmus

Polyformismus představuje další ze základních objektových praktik. Tento princip zabezpečuje skutečnost, že zpráva s jedním určitým jménem, která je zaslána rozdílným objektům (resp. od různých objektů), způsobuje rozdílné chování. Pokud tedy několik objektů poskytuje stejné rozhraní, reagují na stejný podnět a pracuje se s nimi stejným způsobem, avšak jejich konkrétní chování se liší. Při této akci se programátor nemusí zabývat implementací metody a vnitřní strukturou objektu.[1] Existuje dva druhy polyformismu: polymorfismus nad dvěma různými objekty či nad dvěma různými stavy téhož objektu.

1.2.3 Objektově orientované jazyky

Definovat lze tři následující úrovně objektových orientací:

- Jazyky založené na objektech a podporující objekty jako prvky s množinou a stavem.
- Jazyky založené na třídě obsahují jak objekty, tak třídy.
- Objektově orientované jazyky obsahují také dědičnost a polymorfismus. [12]

Existuje velké množství programovacích jazyků umožňujících objektově orientované programování (často jsou označovány jako jazyk 3 a půlté generace). Jedná se např. o Smalltalk, Java, C++, Object Pascal, C#, Visual Basic .NET, Lisp, PHP, Python, Ruby. Ačkoli mnozí odborníci považují jazyk Simula67 za vůbec první z nich, byl to Smalltalk, který zavedl úplnou implementaci pojmu OOP.[12]

Dané jazyky lze dále dělit do dalších podskupin. Zatímco čistě objektové jazyky (často se používá název objektové) využívají pouze objekty (např. Ruby či Smalltalk), hybridní jazyky připouštějí i jiné programovací modely a lze u nich OOP prvky zcela vypustit (např. C++, Python, Object Pascal). Dalším rozlišovacím kritériem může být jejich dynamika, resp. statika. Statické jazyky jsou založeny na znalosti datového typu a provádějí četné typové kontroly v době překladu. Dynamické jazyky mají slabší znalost typu, provádějí většinu kontroly za běhu a jsou obvykle interpretovány (např. Smalltalk). Naproti tomu statické jazyky jsou založeny na znalosti datového typu, provádějí četné typové kontroly v době překladu a jsou kompilovány (např. Object Pascal).[12]

1.2.4 C# a .Net

Architektura .Net výrazným způsobem zjednodušuje vývoj aplikací. Software vytvořený touto technologií není překládán do kódu pro daný procesor, nýbrž do mezijazyka MSIL (Microsoft Intermediate Language). Důvodem je zajištění možnosti, aby kolekce objektově orientovaných komponent mohla byt napsána v jakémkoliv jazyce. V tuto chvíli tedy nezáleží na tom, v jakém jazyce program píšeme a jaké k němu používáme komponenty. MSIL běží ve stroji CLR (Common Language Runtime), jenž se stará o správu paměti, vykonávání procesů, vykonávání kódu, kontrolu bezpečnosti kódu apod.

Ačkoli platforma .Net nepředepisuje žádný programovací jazyk, hlavním jazykem se stal C#, který byl se z 80% vyvinul z jazyka Java. Jeho kořeny dále představuje jazyka c++ a Visual Basic. Dohromady platforma podporuje 27 programovacích jazyků. Dodejme ještě, že Microsoft nabízí pro vývoj aplikací na platformě .Net vývojové prostředí Visual Studio.

1.3 Databáze

1.3.1 Základní definice

Databázi (neboli Datovou základnu) lze definovat jako určitou uspořádanou množinu informací (dat), která slouží pro popis reálného světa a je uložena na paměťovém médiu. V širším smyslu je součástí databáze kromě uložených dat i software umožňující správu dat. Systém pro řízení databáze se nazývá systém řízení báze dat (anglicky Relational database management systém). Databázové systémy nachází ve společnosti široké uplatnění, které sahá od zdravotnictví a školství až po evidenci obyvatel a letectví. K ovládání databáze je dnes většinou používán obvykle dotazovací strukturovaný jazyk SQL.

Pro použití databází lze nalézt mnoho důvodů. Jsou jimi např. skutečnosti, že

- poskytují rychlejší přístup k datům než soubory
- umožňují přímý přístup k datům
- obsahují zabudovaný mechanismus pro paralelní přístup k datům
- mají zabudovaný systém uživatelských práv
- umožňují pomocí dotazů snadno extrahovat množiny dat, která vyhovují zadaným kritériím.[7]

Databáze je možné rozdělit do několika druhů. Na základě kritéria způsobu ukládání dat a vazeb dělíme databázové modely – které slouží k popisu databáze - následujícím způsobem:

- hierarchická databáze (založen na modelování hierarchie mezi entitami se vztahy podřízenosti a nadřízenosti)
- síťová databáze (vychází z teorie grafů, uzly v grafu odpovídají entitám a orientované hrany definují vztahy mezi entitami)
- relační databáze
- objektová databáze
- objektově relační databáze.

Databáze se vyvinuly z papírových kartoték, které pořádaly data na základě různých kritérií. Za první velké strojové zpracování dat se považuje sčítání lidu ve Spojených státech amerických v roce 1890, kdy získané informace zpracovávaly elektromechanické stroje. Elektromechanické stroje se k podobným účelům využívaly až do poloviny 20. století.

Další rozvoj databázových systémů úzce souvisí s vývojem počítačů v 50. letech. V roce 1960 byla publikována první verze jazyka COBOL, který se stal na dlouhou dobu nejrozšířenějším jazykem pro hromadné zpracování dat. Stalo se tak na základě výsledků konference zástupců firem, uživatelů a amerického ministerstva obrany, která usilovala o vytvoření univerzálního databázového jazyka.

V 60. letech tohoto století jsou poprvé definovány pojmy jako např. databáze, entita, atribut entity či vazba mezi entitami. V roce 1965 byl na konferenci CODASYL založen výbor s názvem *Database Task Group*, díky němuž vznikly první síťové SŘBD na sálových počítačích.

Na počátku 70. let 20. století se začaly vyvíjet hierarchické databáze. Jednou z první z nich byl IMS, který vyvinula firma IBM pro program letu Apollo. I v dnešní době patří systém IMS k nejrozšířenějším na sálových počítačích. Ve stejném období se začínají objevovat i první relační databáze a v roce 1974 přišla na svět první verze dotazovacího jazyka SQL. První objektově orientované databáze založené na principu objektově orientovaných jazyků se objevily v 90. letech. Po prvotních a následně nevyplněných předpokladech, že zcela vytlačí relační systémy, se objevují kompromisní objektově-relační technologie.[14]

1.3.2 Vývoj databází

Databáze se vyvinuly z papírových kartoték, které pořádaly data na základě různých kritérií. Za první velké strojové zpracování dat se považuje sčítání lidu ve Spojených státech amerických v roce 1890, kdy získané informace zpracovávaly elektromechanické stroje. Elektromechanické stroje se k podobným účelům využívaly až do poloviny 20. století.

Další rozvoj databázových systémů úzce souvisí s vývojem počítačů v 50. letech. V roce 1960 byla publikována první verze jazyka COBOL, který se stal na dlouhou dobu nejrozšířenějším jazykem pro hromadné zpracování dat. Stalo se tak na základě výsledků konference zástupců firem, uživatelů a amerického ministerstva obrany, která usilovala o vytvoření univerzálního databázového jazyka.

V 60. letech tohoto století jsou poprvé definovány pojmy jako např. databáze, entita, atribut entity či vazba mezi entitami. V roce 1965 byl na konferenci CODASYL založen výbor s názvem *Database Task Group*, díky němuž vznikly první síťové SŘBD na sálových počítačích.

Na počátku 70. let 20. století se začaly vyvíjet hierarchické databáze. Jednou z první z nich byl IMS, který vyvinula firma IBM pro program letu Apollo. I v dnešní době patří systém IMS k nejrozšířenějším na sálových počítačích. Ve stejném období se začínají objevovat i první relační databáze a v roce 1974 přišla na svět první verze dotazovacího jazyka SQL. První objektově orientované databáze založené na principu objektově orientovaných jazyků se objevily v 90. letech. Po prvotních a následně nevyplněných předpokladech, že zcela vytlačí relační systémy, se objevují kompromisní objektově-relační technologie.[14]

1.3.3 Základní pojmy

Entita je prvek reálného světa (např. člověk, stroj, vyučovaný předmět, město), který je popsán svými charakteristikami (vlastnostmi), které většinou považují za atribut (např. jméno, příjmení, stav, plat, hmotnost).[19]

Zásadním pojmem je vazba mezi entitami. Typy těchto vazeb jsou následující:

- vazba typu 1:1 právě jedny osobní údaje vedené na magistrátě, na oddělení občanských průkazů.
- vazba typu 1:N jeden člověk může vlastnit více kreditních karet (ale jedna kreditní karta může být vlastněna pouze jedním člověkem
- vazba typu M:N Zde není žádné omezení, příkladem by mohla být situace, že student na vysoké škole si může zapsat několik různých předmětů (ale jeden předmět může být zároveň zapsán více studenty).[19]

1.3.4 Architektura databázového systému

Z hlediska architektury lze databázi rozdělit na tři úrovně:

- fyzickou úroveň
- logickou úroveň
- úroveň pohled.

Fyzická úroveň popisuje, jakým způsobem je záznam uložen. Logická úroveň slouží k popisu dat uložených v databázi a vztahů mezi nimi. Logická struktura databáze se označuje jako schéma; ke specifikaci jeho definice slouží jazyk pro definici dat (Data Definition Language). Naproti tomu jazyk pro manipulace s daty (Data Manipulation Language) zpřístupňuje data a umožňuje manipulaci s nimi. Pod pojmem instance rozumíme konkrétní obsah databáze v daném časovém okamžiku. Úroveň pohledu je využívána tehdy, přejeme-li si skrýt nějaké informace. Aplikační programy mohou skrývat detaily o typech dat či – z bezpečnostních důvodů - soukromé informace, jako jsou např. rodná čísla.[14]

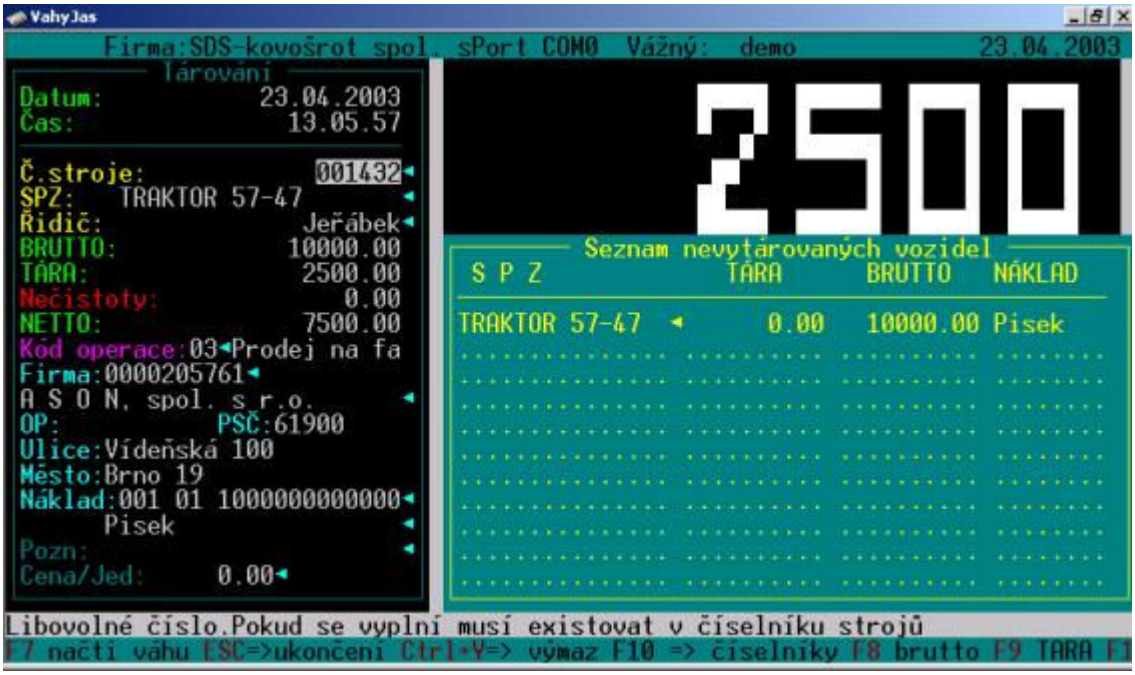
1.4 RS232

Sériové rozhraní RS 232 (někdy označované jako COM) se dnes v počítačích nevyskytuje tak často jako dříve, neboť jej nahradilo rychlejší rozhraní USB Universal Serial Bus. I přesto však velké množství zařízení i dnes stále komunikuje po starším portu RS 232. K připojení takového zařízení k PC slouží převodník USB/COM. Port RS 232 byl v prvopočátku používán výhradně ke komunikaci s modemem tak, aby se data mohla přenášet po telefonní lince. Později port sloužil ke komunikaci s tiskárnou, myší a dalším zařízením.

Poslední varianta portu - RS-232C - pochází z roku 1969. Standard definuje asynchronní sériovou komunikaci pro přenos dat. Přenos datových bitů je od LSB (nejméně významný bit) po MSB (nejvíce významný bit). Lze se setkat se sedmi či devíti datovými typy pro přenos dat, ale obvykle se používá osm datových bitu. Logický stav 0-1 je určován bipolárním napájením od +-5V až po +-15V; nejčastěji se však používá +-12V.

2 Architektura návrhu aplikace

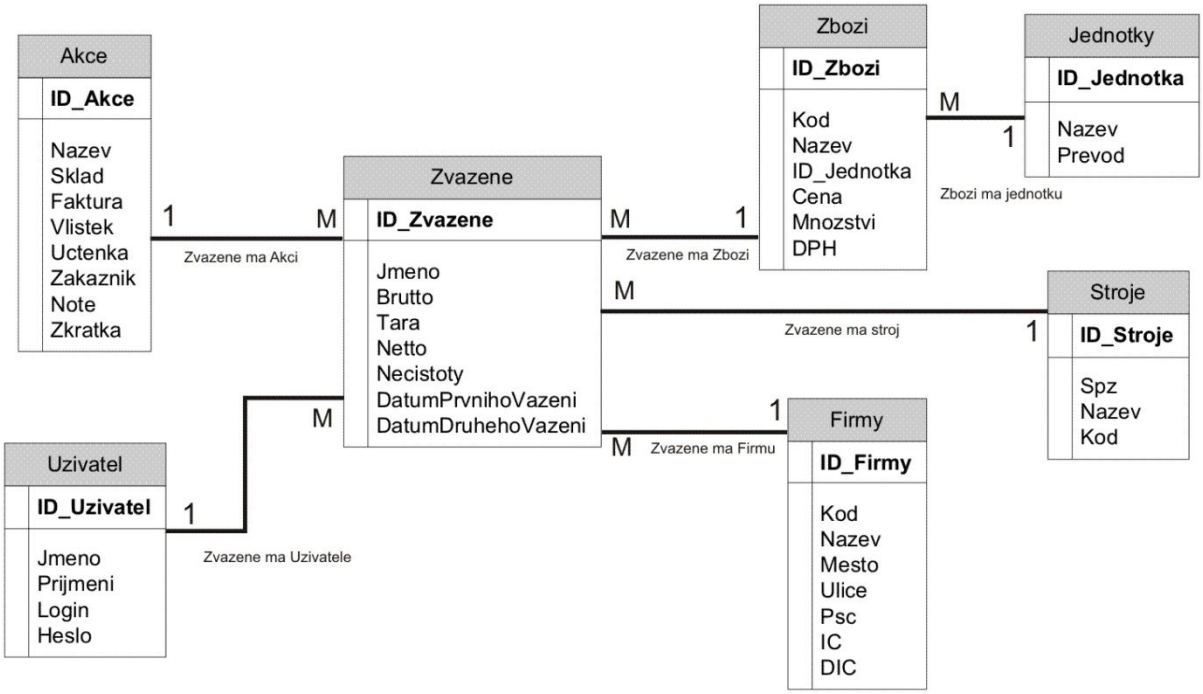
Cílem práce je napsat aplikaci, která bude schopná efektivně zaznamenávat a zpracovávat data získaná z elektronické váhy komunikující po portu RS232. Software byl vyvíjen a optimalizován pro sklady stavebního materiálu, sběrná místa apod. Hlavním úkolem spočíval v napsání aplikace s příjemným uživatelským rozhraním. Důraz byl rovněž kladen na co největší rychlost zvážení vozidla, zadaní všech potřebných údajů a uložení záznamu do vážního deníku. Aplikace by měla být schopna nahradit zastaralý software dodávaný firmou Ekodata.



Obrázek č. 1: Software firmy Ekodata

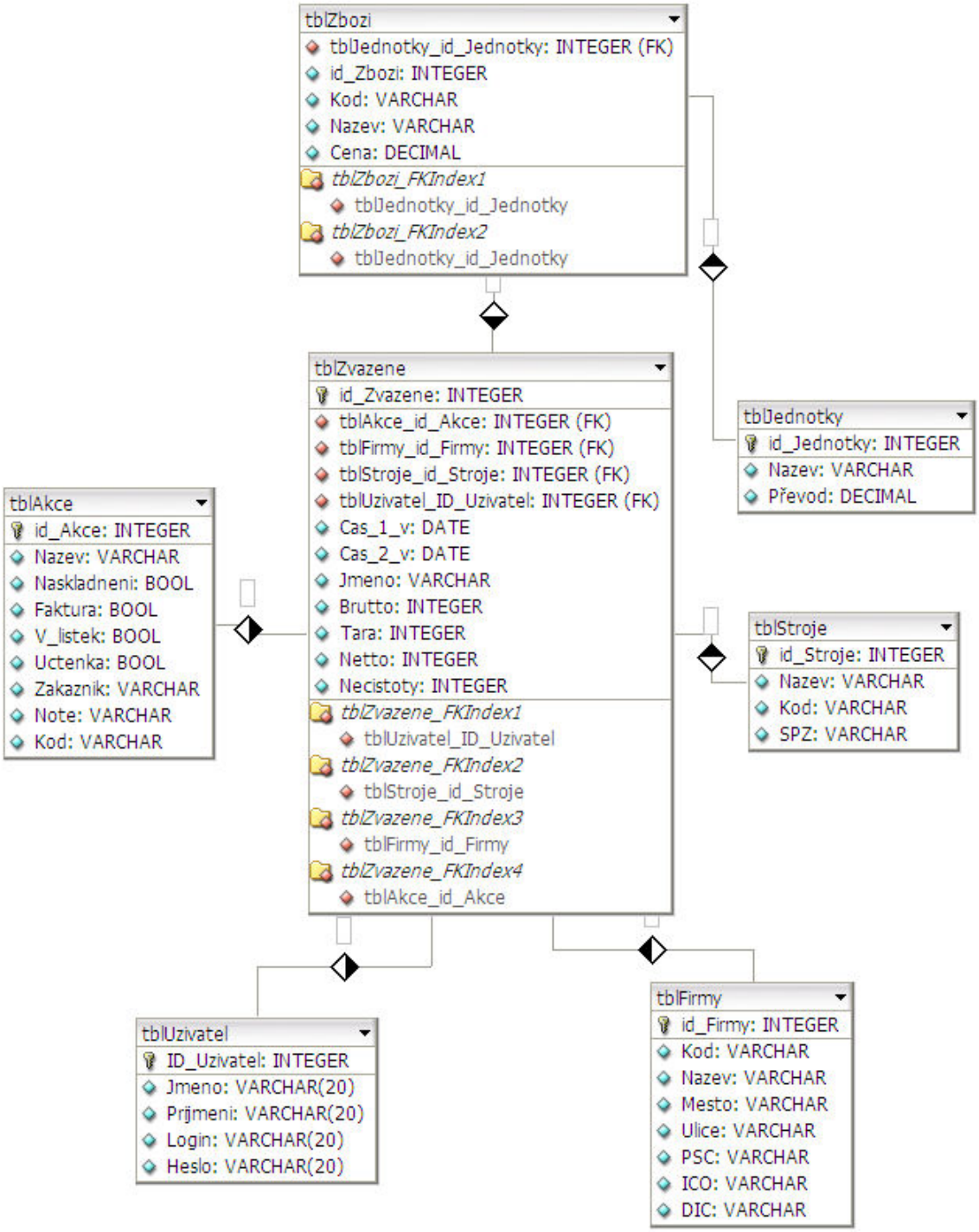
2.1 Vytvoření databáze

Jak již bylo zmíněno, autor si pro svoji bakalářskou práci zvolil databázi MSSQL Express Server 2005, který je součástí VizualStudia 2005. Databáze byla vytvořena podle ER diagramu.



Obrázek č. 2: ER diagram databáze

ER diagram sestává z několika entit (v případě předkládané bakalářské práce se jedná o sedm entit) a vazeb mezi nimi. Každá entita obsahuje jedinečný identifikátor (id) příslušného záznamu v tabulce.Podle předchozího ER diagramu bylo vytvořeno schéma databáze, které odpovídá reálné databázi.



Obrázek č. 3: Schéma databáze

- Základní tabulkou je tabulka *tblUzivatel,* která slouží k evidenci uživatelů softwaru. Tato tabulka obsahuje položky *Jméno*, *Příjmení*, *Přihlašovací jméno* a *Heslo*. Na základě údajů uvedených v tabulce se uživatel přihlašuje do systému a evidují se veškeré záznamy o jeho vážení.
- Další je *tblAkce*. Tato entita slouží k uchování údajů potřebných pro logický běh aplikace. Nachází se zde sloupec *Sklad*, jenž identifikuje, zda se jedná o import či export do skladu.

Záznam v tabulce dále obsahuje možnost tisku faktury, účtenky, vystavení vážního lístku. Následující důležitou položkou je sloupec *Zkratka*, který slouží k rychlému vybrání dané akce při vážení. Dále zde nalezneme sloupec *Zákazník*, jenž specifikuje, jaký typ osoby váží (právnická osoba, fyzická osoba, drobný uživatel nebo skladník).

- Entita *tblFirmy* obsahuje *Kód* pro rychlé vybrání firmy v systému, *Název, Město, Ulici, PSČ, IČ, DIČ.*
- Následuje tabulka *tblStroje*, v níž jsou umístěny záznamy o vozidlech, které byly a budou váženy. Tabulka opět obsahuje *Kód* pro rychlé zadání do systému, *SPZ* (která je unikátní pro každý záznam v tabulce) a *Název* příslušného stroje.
- Entita *tbl.Zbozi* sestává z *Kódu zboží* pro rychlé zadávání do aplikace, *Názvu zboží*, DPH (hodnota, která se vztahuje k danému zboží) a *Ceny*. Cena je uvedena v souladu se zvolenou jednotkou, přičemž *Jednotka* představuje další záznam v tabulce.
- V případě *tbl.Jednotky* se jedná pouze o systémovou entitu, jež obsahuje *Název jednotky* pro tabulku *tbl.Zbozi*, *Zkratku jednotky* a *Převodní parametr* vůči základní jednotce aplikace, tedy kilogram.
- Po prvním vážení je třeba údaje o daném vážení řádně zaznamenat, k čemuž slouží entita *tblZvazene*. Záznam v tabulce obsahuje *Údaj o vybrané akci pro vážení*, *Jméno zákazníka*, *Údaje o stroji*, *firmě* (pokud je potřeba je evidovat), *zboží*, jež se má vážit, a údaje o výsledku vážení (brutto, tára a nečistoty). Umístěn je zde rovněž *Čas prvního vážení*.
- Pokud dojde ke druhému vážení, vybere se předem rozvážené vozidlo, následně se zváží a údaje se uloží do entiti *tblZvazene*. Tyto údaje doplňují první vážení. Jedná se o následující údaje: *Jedinečný identifikátor* zváženého záznamu, *Jméno zákazníka, Údaj o použité akci, Zboží, Firmě a Stroji*; *Brutto, Tára, Netto*, *Nečistoty*; *Datum* prvního a druhého vážení a *Jméno uživatele*, který vážení provedl.

2.2 Vytvoření aplikace

Pro psaní aplikace si autor vybral programovací jazyk C#.net. Jedná se o objektově orientovaný přístup k programování - formulářovou aplikaci, která je řízena událostmi.

2.2.1 Stručný popis aplikace

Aplikace je rozdělena do dvou hlavních částí, a sice na agendu pro první vážení a agendu pro druhé vážení. Při prvním vážení se zadává kód akce, pro kterou má být vážení provedeno. Akce nese informaci, o tom zda se jedná o naskladňování či vyskladňování. Při naskladňování se náklad vybírá při prvním vážení; při vyskladňování až při druhém. Dále je nutné zadat stroj, pro nějž se vážení provádí, a - dle nutnosti - rovněž firmu. Způsob, jak je daná akce nastavena, zaleží opět na kódu akce.

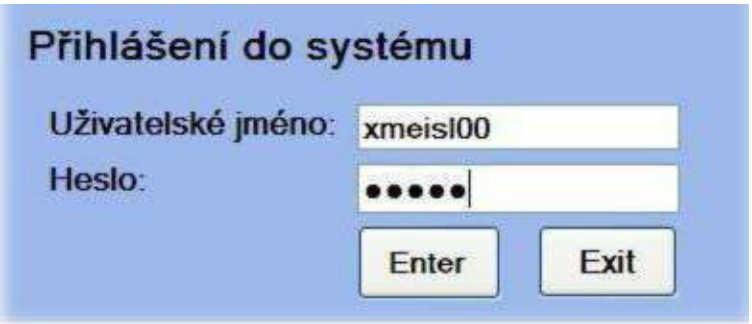
Po provedení prvního vážení se rozvážené vozidlo uloží do tabulky rozvážených vozidel, čímž se proces prvního vážení ukončí. Vybráním vozidla z tabulky *Rozvážených vozidel* započneme proces druhé vážení. Pokud se jedná o vyskladňování, náklad vybereme z tabulky *Zboží*. Po té, co potvrdíme druhé vážení, daný záznam se z tabulky *Rozvážených vozidel* smaže a celé vážení je uloženo do tabulky *Zvážená vozidla*. Jelikož zmíněná tabulka uchovává veškeré informace o provedeném vážení, jedná se de facto o vážní deník.

Další agendy aplikace jsou:

- přihlašovaní a odhlašování ze systému
- správa uživatelů
- správa zboží
- správa akci
- správa firem
- správa strojů
- vážní deník

2.2.2 Přihlášení do aplikace

Po spuštění aplikace se zobrazí dialog pro přihlášení vytvořený podle třídy *LogIn.cs*, která je odvozena od základní třídy *Form*. Na formuláři se nachází komponenty pro přihlášení do systému. Uživatel zadá přihlašovací údaje a pokusí se přihlásit do systému tlačítkem OK, na základě čehož se vyvolá událost *OK.click*. V této události se ověří platnost přihlašovacích údajů a - pakliže jsou korektní - uživatel se nastaví v aplikaci jako přihlášený a dialog se uzavře. Tím pádem se předá řízení programu na hlavní část programu. Pokud se přihlášení nepodaří, aplikace se nespustí.



Obrázek č. 4: Přihlašovací formulář

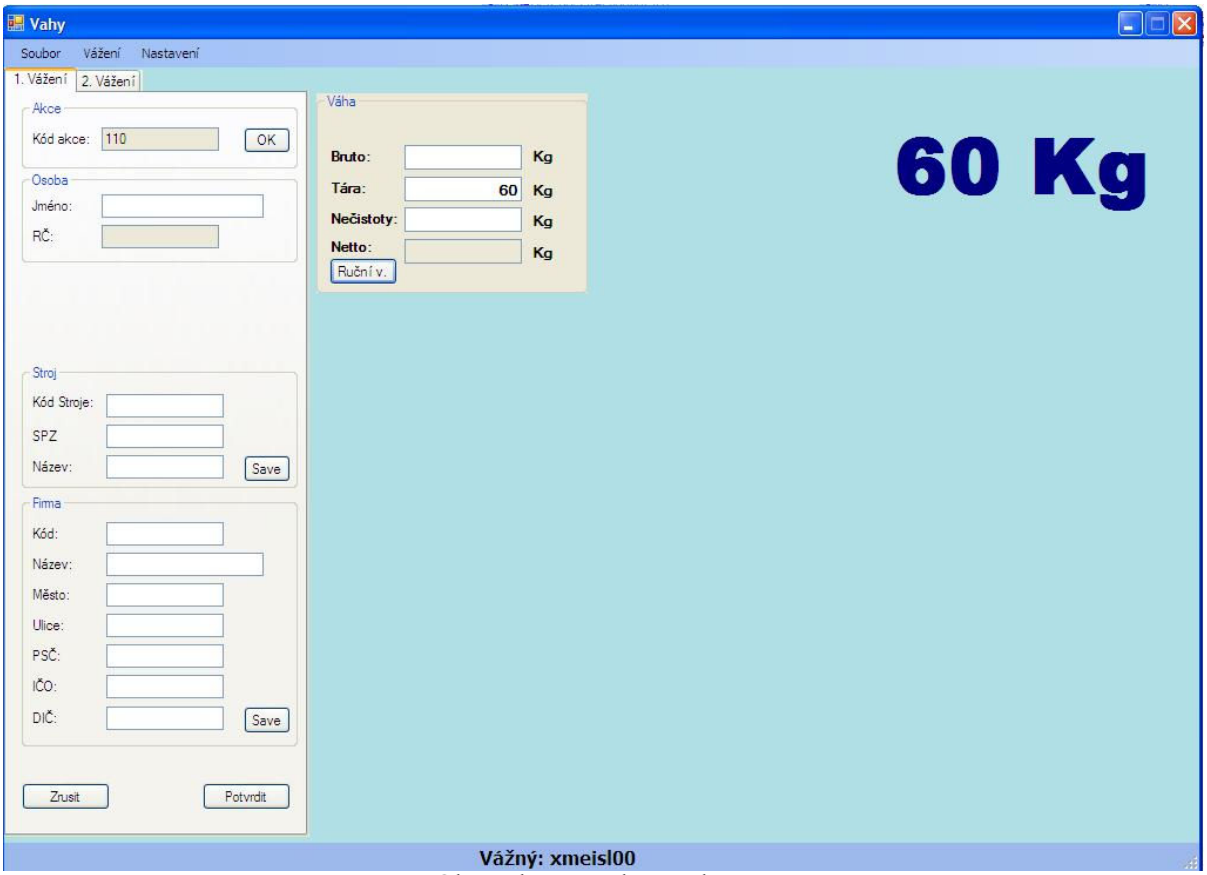
2.2.3 Hlavní okno aplikace

Hlavní okno se skládá z roletového menu vytvořeného podle třídy *Menu.Strip* ve jmenném prostoru *SystemWindows.Forms*. Jeho další součástí je záložkové menu vytvořené podle třídy *TabControl* ve výše uvedeném jmenném prostoru. V neposlední řadě zde nalezneme také komponenty pro váhu a komponenty pro nápovědu. Hlavní okno a rozložení jeho prvků je optimalizováno pro rozlišení 1024x768.

Roletové menu je vytvořeno podle třídy *MenuStrip*. Obsahuje kolekci jednotlivých elementů zobrazovaných v nabídce, která je rozdělena na následující položky:

1. Soubor
    a. Odhlášení
    b. Konec
2. Vážení
    a. První vážení
    b. Druhé vážení
    c. Vážní deník
3. Nastavení
    a. Uživatele
    b. Akce
    c. Stroje
    d. Zboží
    e. Firmy
    f. Váha
4. O aplikaci

Pro rychlý přístup k menu jsou naimplementovány klávesové zkratky. Po stisku klávesy ALT se podtrhne písmeno, které je za účelem vyvolání dané položky nutné stisknout zároveň s klávesou ALT.



Obrázek č. 5: Hlavní okno

2.2.4 Záložkové menu

Záložkové menu obsahuje dvě záložky - *První vážení* a *Druhé vážení* -, které jsou vytvořeny podle třídy *TabPage*. V první záložce nalezneme skupiny objektů pro zadání prvního vážení:

- Akce
- Osoba
- Náklad
- Stroj
- Firma

Jednotlivé prvky daných skupin jsou uloženy v *GroupBox*. Generování vyplňovacího formuláře probíhá na základě nastavení vybrané akce, přičemž nepotřebné skupiny prvků se nezobrazují a jsou neaktivní. U skupiny *Stroj* a *Firma* se nachází tlačítko *SAVE*, jež slouží k rychlému uložení záznamu do databáze, a tudíž není třeba záznam ukládat přes agendu *Firmy* či *Stroje* v hlavním menu. Tato funkce byla naimplementována proto, aby komunikace mezi uživatelem a softwarem byla rychlejší.

Další tlačítko na formuláři slouží k potvrzení prvního vážení. V reakci na jeho stisk se vyvolá událost, ve které je zavolána funkce kontrolující validitu a správný formát zadaných dat. Jestliže jsou data nevalidní či chybně nastavena, funkce zahlásí chybu prvního vážení a barevně zvýrazní nesprávně vyplněné položky. Pakliže je formulář vyplněn dobře, zavolá se funkce pro uložení prvního vážení do *Rozvážených vozidel*. Posléze se formulář nastaví do původního stavu a aplikace je připravena k provedení dalšího vážení.

Ve druhé záložce *Druhé vážení* se nachází téměř stejné skupiny objektů jako v první s tím rozdílem, že většina položek je vyplněna podle vybraného záznamu z tabulky *Rozvážených vozidel*. Pouze náklad je potřeba vybrat v prvním anebo druhém vážení, což opět záleží na vybrané akci. Důležitou skupinou je *Cena* obsahující veškeré informace o ceně nákladu. Dále na formuláři najdeme dvě tlačítka: první přepočítání ceny za zboží o navážené hodnotě a druhé potvrzuje druhého vážení. Toto potvrzení je úspěšné, pakliže jsou zadané a navážené hodnoty správné.

Podstatné je zmínit, že vážení probíhá automaticky. Za předpokladu správné funkce vážícího systému bychom se díky této skutečnosti mohli potenciálně vyhnout špatně naváženým hodnotám. Systém je ovšem schopen detekovat i špatně navážené hodnoty, a to tak, že provede komparaci mezi Bruttem a Tarou. Je-li vše v pořádku, potvrdíme druhé vážení a uložíme navážené hodnoty do *Vážního deníku*.

2.2.5 Rychlá nápověda

Myšlenka *Rychlé nápovědy* vznikla na základě nutnosti moci s aplikací pracovat rychle a efektivně. Při zadávání údajů u prvního a druhého vážení při stisknutí mezerníku v kódu daného objektu se vyvolá nápověda se všemi uloženými informacemi. V ní stačí pouze vybrat daný řádek a stisknout mezerník a všechny údaje dané položky se vyplní. Díky této možnosti je práce s aplikací velmi snadná a rychlá. Prvek *Rychlá nápověda* je univerzální pro všechny skupiny objektů *Akce, Stroje, Náklad a Firmy*; je používán i u tabulky *Rozvážených vozidel*.

Nápověda používá pro zobrazení dotazovaných dat objekt *DataGridView* jménem *dgwHelp*. Jestliže má být nápověda vyvolána (pomocí stisku mezerníku v dané položce), zavolá se funkce pro naplnění nápovědy. Tato funkce provede příslušný dotaz nad danou databází, vytvoří objekt podle třídy *Dataset* a naplní objekt *dgwHelp*.

První sloupec obsahuje id (jedinečný identifikátor) z tabulky, nad kterou se prováděl dotaz. Tento sloupec v aplikaci není viditelný, protože slouží pouze pro běh aplikace. V tuto chvíli se zobrazí v aplikaci rychlá nápověda a lze vybrat či vyhledat daný záznam pro vyplnění formuláře. Vyhledávání se zpravidla provádí podle kódu nebo názvu daného objektu ve formuláři pro vážení. Jestliže je jedna z těchto položek změněna či zadána, automaticky se zavolá funkce pro naplnění nápovědy s parametrem pro podmínku vyhledání. Díky tomu je dosaženo skutečnosti, že nápověda

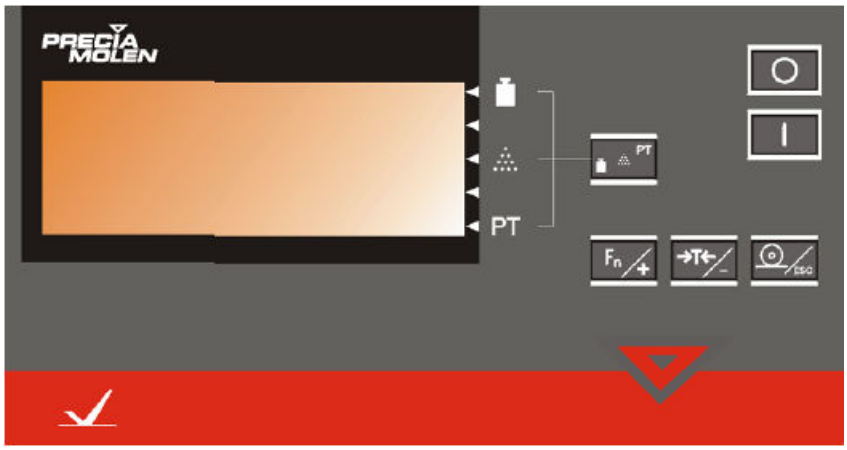
promptně reaguje na změnu vyhledávacího řetězce. Pakliže již máme daný záznam vybrán, potvrdíme svoji volbu mezerníkem. V tento moment se zavolá funkce, které se předá id daného záznamu. Funkce pomocí příslušných tříd zjistí informace o příslušném záznamu a vyplní je do objektu formuláře. Následně nápověda zmizí.

2.2.6 Nastavení aplikace

Nastavení, které je pro každého uživatele software jiné, se uloží do konfiguračního souboru *app.config*. Jedna se o nastavení připojení k databázi tzv. *ConnectionString* a o nastavení parametru připojení váhy přes port RS232. I když soubor *app.config* představuje validní XML soubor, přístup k němu není tvořen pomocí *XmlReaderů* implementovaných ve třídách NET, nýbrž pomocí třídy *Settings* ze jmenného prostoru *Properties*. Přístup je tak velice jednoduchý a efektivní. Při změně nastavení portu, po němž má váha s aplikací komunikovat, není třeba aplikaci restartovat, neboť změněné věci se projeví ihned po vlastním uložení.

2.2.7 Komunikace s váhou

Komunikační modul programu byl napsán pro indikátor I 200 B od firmy Precia Molen.



Obrázek č. 6: Indikátor I 200 B

Indikátor komunikuje s tenzometrem a A/D převodníkem řady ASL od stejné firmy. Indikátor zpracovává digitální signál z tenzometru a převádí ho na signál vhodný ke komunikaci s PC, přičemž komunikace probíhá po portu RS 232. V aplikaci je vytvořen objekt podle třídy *SerialPort.* Konstruktoru této třídy je předán název portu, po němž má komunikace probíhat, spolu s rychlostí komunikace (1200 Bitů za sekundu), datovými bity (8), paritou (žádná) a jedním stopbitem. Jestliže na port přijdou nějaká data, vyvolá se událost, která zpracuje příchozí řetězec a nastaví proměnou na

příslušnou hodnotu odpovídající navážené hodnotě. V případě špatného nastavení portu je hodnota nadefinována na -1000 kg.

Závěr

Tématem předkládané bakalářské práce byla programová podpora pro vážní systém. Hlavní cíl práce, který spočíval ve vytvoření aplikace, jež by byla schopna zaznamenávat data o navážených hodnotách, byl splněn. Přínosem navrženého systému je jeho rychlost a efektivita. Čas potřebný ke zvážení jedné položky je poměrně krátký; velkou úsporu času zabezpečila funkce rychlé nápovědy.

Dílčím cílem bakalářské práce bylo naprogramovat komunikační modul programu pro komunikaci váhy s PC. I tento cíl se podařilo naplnit. Komunikace mezi aplikací a váhou probíhá v pořádku a s dostačující rychlostí, která je potřeba pro získání navážených dat. Potenciální problém by mohl nastat, pokud by se použil převodník USB/Com. Jelikož však tento převodník nebyl k dispozici, nemohla být daná varianta vyzkoušena.

Nosným námětem pro eventuální rozpracování bakalářské práce by mohla být tvorba doplňujících agend aplikace, které by sloužily coby základ pro ekonomický systém (např. vystavení faktur, splácení faktur, účtenek, možnost tisku apod.). Dalším zajímavým vývojem řešeného projektu by byla možnost tvorby modulu pro připojení jiných druh vah, např. tříbodové vážení.

Literatura

[1] Bosák, R., Fanta, M., Peřina, M. Pár kapek Javy. 2000.
http://www.ataco.cz/perina/par-kapek/intro.html

[2] Budoucnost vážení nákladních vozidel.
Http://www.cdv.cz/text/oblasti/legislat/clanek-1.rtf

[3] Černohorský, J. Je měření přetížených silničních vozidel prioritou příslušných ministerstev? Technický deník.
http://www.techtydenik.cz/detail.php?action=show&id=1120&mark=

[4] Černohorský, J. Průmyslová vážící technika pro chemický průmysl. Chemické listy, 2001, č. 2.
http://chemicke-listy.cz/Bulletin/bulletin322/bulletin322.html

[5] Černohorský, J. Váhy pro technologické a logistické procesy. MM průmyslové spektrum.
http://www.mmspektrum.com/clanek/vahy-pro-technologicke-a-logisticke-procesy

[6] Dynamické vážení vozidel. Tara Invest, s.r.o.
http://www.taranisinvest.com/?lg=cs&s=silnicniwim

[7] Horvát, T. *T*eoretický úvod do relačních databází. 08. 11. 2007.
http://programujte.com/index.php?akce=clanek&cl=2007110801-teoreticky-uvod-do-relacnich-databazi

[8] Charvát, H. OECD podruhé hodnotí životní prostředí ČR. Ekolist, 25. října 2005.
http://www.ekolist.cz/zprava.shtml?x=755665

[9] I 200 B/M Reference Manual.PreciaMolen 15. 4. 2001.

[10] Kainka, B. Využití rozhraní PC. Měření, řízení a regulace pomocí standardní portů PC. Ostrava, HEL 1997.

[11] Kamiony ničí silnice Vysočiny. Vysočina News. 1.1.2007.
http://www.vysocina-news.cz/clanek/kamiony-nici-silnice-vysociny/

[12] Kříž, P. Vývoj programování a programovacích jazyků. Fakulta informatiky Masarykovy univerzity.
http://www.fi.muni.cz/usr/jkucera/pv109/2002/xkriz1.htm

[13] Lesson: Object-Oriented Programming Concepts. The Java Tutorials.
http://java.sun.com/docs/books/tutorial/java/concepts/

[14] Medřický, B. Dialogové rozhraní pro správu relačního databázového systému. Fakulta informatiky Masarykova univerzity, Brno 2007.
http://is.muni.cz/th/99026/fi_b/bakalarka.pdf

[15] Moškor, K. Přetížené nákladní auta kontrolují a váží celníci. Celní správa České republiky, 3. 2. 2006.
http://www.cs.mfcr.cz/CmsGrc/Tiskove-centrum/Tiskovy-archiv/2006/060203-kontroly-vazeni-pokuty.htm

[16] Sells, Ch. C# a WinForms. Programování formulářů Windows. Brno, Zoner Press 2005.

[17] Sharp, J. Microsoft Visual C# 2005 krok za krokem. Brno, Computer Press 2006.

[18] Silniční váhy. Tamtron, s.r.o.
http://www.tamtron.cz/produkty.php?kat=silnicni

[19] Skřivan, J. Databáze a jazyk SQL. Vývoj aplikací, Interval.cz.
http://interval.cz/clanky/databaze-a-jazyk-sql/

[20] Price, J. C# programování databází. Praha, Grada Publishing 2005.

[21] Teorie relačních databází: Relační model dat. 12.1. 2006.
http://www.manualy.net/article.php?articleID=9

[22] Vážení silničních vozidel v ČR: historie a současnost. Příspěvek na seminář Preprava nadmernych nakladov. Tenzo, 2003.
http://www.tenzovahy.cz/publikace/clanky/2004-04_vazeni_vozidel_seminar_senec.pdf

[23] Vážením vozidel šetří komunikace.
http://www.mvcr.cz/rs_atlantic/project/article.php?id=25605

[24] Vlach, J., Vlachová V. Počítačová rozhrání. Přenos dat a řídící systémy. Praha, BEN – technická literatura 1995.

[25] Vliv přetížení vozidel na relativníživotnost konstrukce vozovky. http://www.cideas.cz/free/okno/technicke_listy/2tlv/1311-1.pdf

Seznam příloh

Příloha 1. CD se zdrojovými texty