TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Návrh a realizace informačního systému s databázovou a webovou aplikací

Design and realization of information system with database and web application

Diplomová práce

Autor: **Bc. Lukáš Nesvadba**
Vedoucí práce: Ing. Zuzana Capeková

V Liberci 27. 5. 2009

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Ústav mechatroniky a technické informatiky
Akademický rok: **2008/2009**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Lukáš NESVADBA**

Studijní program: **N2612 Elektrotechnika a informatika**

Studijní obor: **Informační technologie**

Název tématu: **Návrh a realizace informačního systému s databázovou a webovou aplikací**

Zásady pro vypracování:

1. Detailně se seznamte s možnostmi nových funkcí a modulů informačního systému.
2. Pro vytvoření nové verze informačního systému navrhněte databázovou aplikaci, která bude poskytovat vylepšené funkce a nové moduly.
3. Navrhněte webovou aplikaci, která bude součástí informačního systému.
4. Navrhněte algoritmy a vyberte vhodná vývojová prostředí pro tvorbu aplikací.
5. Realizujte své návrhy řešení.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40 - 50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

1. **Pokorný, J., Halaška, I.: Databázové systémy, Praha: Vydavatelství ČVUT, 2001.**

2. **MacDonald, M., Szpuszta, M.: ASP.NET 2.0 a C#, Brno: Zoner Press, 2006**

3. **Stanek, W. R.: Microsoft SQL Server 2000, Brno: Computer Press, 2003**

4. **Price, J.: C# : programování databází, Praha: Grada, 2005**

5. **Petzold, Ch.: Programování Microsoft Windows v jazyce C#. Svazek 1a 2., Praha: SoftPress, 2003**

Vedoucí diplomové práce:	**Ing. Zuzana Capeková** Ústav mechatroniky a technické informatiky
Datum zadání diplomové práce:	**31. října 2008**
Termín odevzdání diplomové práce:	**29. května 2009**

V. Kopecký

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Petr Tůma, CSc.
vedoucí ústavu

V Liberci dne 31. října 2008

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít svou diplomovou práci či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum 27.5.2009

Podpis

Poděkování

V úvodu své práce bych rád poděkoval své vedoucí inženýrce Zuzaně Capekové za metodické pokyny a podnětné návrhy ohledně konceptu a struktury celé práce.

Abstrakt

Cílem diplomové práce je seznámit se s možnostmi stávajícího identifikačního systému a s uživatelskými požadavky na novou verzi informačního systému, dále prostudovat možnosti tvorby databázových a webových aplikací ve vývojovém prostředí Microsoft Visual Studio 2008 ve spojení s databázovým systémem MS SQL 2000 a na základě získaných znalostí vytvořit prostřednictvím programovacího jazyka C# dvouvrstvou desktopovou aplikaci pro evidenci a prodej rostlin a pomocí skriptovacího jazyka ASP.NET 2.0 vytvořit webovou aplikaci pro základní správu rostlin a výstupů prodeje. Aplikace by se ve spojení s identifikačním systémem měly stát stěžejným softwarovým nástrojem zahradnictví Eurogarden.

Abstract

The aim of the graduation thesis is to introduce into the possibilities of an existing identification system and custom demands for a new version of the system, to study the options of database and web applications settings in Microsoft Visual Studio 2008 in connection with the database system MS SQL 2000. On the basis of gathered knowledge and using the programming language C# to create a desktop two-layer application intended for evidence and sale of plants and by means of script language ASP.NET 2.0 to create a web application for basic administration of plants and outcomes of their sale. The application is supposed to be an essential software tool of the Eurogarden horticulture.

Klíčová slova

C#, ASP, SQL, relační databáze, informační systém, ADO, MS Visual Studio, .Net, Chart Controls, pdfSharp, Code 128, Datamax

Key words

C#, ASP, SQL, relational database, information system, ADO, MS Visual Studio, .Net, Chart Controls, pdfSharp, Code 128, Datamax

Obsah

Seznam použitých zkratek

ADSL ..... Asymmetric Digital Subscriber Line

ASP ..... Active Server Pages

CLR..... Common Language Runtime

CLS ..... Common Language Specification

DCL ..... Data Creation Language

DDL ..... Data Definition Language

DFD ..... Data Flow Diagrams

DML..... Data Manipulation Language

DOM ..... Document Object Model

EAN ..... European Article Numbering

ERD ..... Entity-Relationship Diagrams

FCL ..... Framework Class Library

FTP..... File Transfer Protocol

HTML ..... Hypertext Markup Language

HTTP ..... Hypertext Transfer Protocol

IIS..... Internet Information Services

IP ..... Internet Protocol

ISO ..... International Organization for Standardization

MSIL ..... Microsoft Intermediate Language

PDF ..... Portable Document Format

PHP ..... Personal Home Page, Hypertext Preprocessor

SGML ..... Standard Generalized Markup Language

SQL..... Structured Query Language

T-SQL ..... Transact Structured Query Language

UPC..... Universal Product Code

W3C ..... World Wide Web Consortium

WCF..... Windows Communication Foundation

WML..... Wireless Markup Language

WPF ..... Windows Presentation Foundation

XML..... Extensible Markup Language

XSL..... Extensible Stylesheet Language

XSLT ..... Extensible Stylesheet Language Transformations

Úvod

V dnešní době představují informační systémy silný a efektivní nástroj pro sběr, udržování, zpracování a poskytování informací a dat. Lze je nalézt takřka ve všech odvětvích lidské činnosti a v různých podobách. První informační systémy měly výhradně papírovou podobu a až s vývojem informačních technologií se začaly automatizovat pomocí počítačů.

Diplomová práce navazuje na magisterský projekt, jehož hlavním cílem bylo vytvořit identifikační systém pasivních prvků v logistickém řetězci pro potřeby zahradnictví. Součástí řešení projektu byla i realizace databázové aplikace, která obsahovala základní funkce. Cílem diplomové práce je vytvořit novou verzi informačního systému rozšířenou o nové nástroje, která by měla výrazně zefektivnit firemní procesy a tím také zvýšit konkurenceschopnost firmy. Tvorbě konečného softwarového řešení této diplomové práce předcházelo studium kapitol softwarového inženýrství se zaměřením na životní cyklus softwaru.

První část práce se zabývá popisem současného stavu řešení problematiky stávajícího identifikačního systému. Je zde popsán vybraný způsob identifikace rostlin a jeho realizace pomocí jednotlivých prvků identifikačního procesu. Dále je zde popsán samotný proces identifikace, současný stav a využitelnost jednotlivých částí identifikačního systému.

Druhá část práce je věnována seznámení s uživatelskými požadavky na novou verzi informačního systému a podrobnému popisu požadovaných nástrojů.

Ve třetí části jsou popsané nástroje a technologie vhodné pro realizaci řešení. Kromě popisu nejdůležitějších vlastností databázového systému a jazyka SQL se zde nachází i popis technologie Microsoft .NET a formátu XML.

Čtvrtá část práce se zabývá návrhem informačního systémů z pohledu diagramu případů použití a diagramů datových toků. Dále jsou zde popsány návrhy databázové i webové aplikace.

Poslední pátá část práce je věnována realizacím návrhů. Zahrnuje v sobě řešení databázového systému včetně návrhu a implementace databázového schématu a dále zde lze nalézt popis řešení databázové a webové aplikace. V samotném závěru práce je umístěno zhodnocení výsledného řešení a nástin dalších možností rozvoje.

1. Současný stav řešení problematiky

Pro efektivní chod firmy je nezbytné optimalizovat jednotlivé procesy, aby nedocházelo ke zbytečným ztrátám často vedoucích ke snížení konkurenceschopnosti firmy. Jedním z důležitých nástrojů zefektivnění chodu maloobchodního podniku, zabývajícího se prodejem, je implementace kvalitního informačního systému, který bude schopen nabídnout požadované funkce v uživatelsky přívětivém rozhraní.

V dřívějších dobách, kdy informační technologie byly teprve na svém počátku, nebyl k dispozici žádný příliš efektivní aparát, který by procesy maloobchodního prodeje dovedl výrazněji optimalizovat. Dle legislativy České republiky daná povinnost evidence nákupu a prodeje byla často řešena zaváděním finančních denníků, které musel ale někdo spravovat a při kontrole za případné byť neúmyslné chyby nést zodpovědnost. Taková správa byla ale vcelku náročná a pokladní tak museli často zůstávat až do noci ve firmách, aby mohli zaevidovat denní prodeje. Ještě horší podmínky měli účetní, neboť museli zpracovat značné množství účtenek, což bylo a vždy bude velmi časově náročné a v důsledku i dosti nákladné.

Dnes je už ale situace o poznání lepší. Současné nástroje a možnosti informačních technologií umožňují vytvářet pro jakékoliv odvětví lidské činnosti informační systémy s vícevrstvými aplikacemi, jejichž funkce adekvátně nahrazují stereotypní pracovní úkony a umožní tak firmám více využívat potenciál svých zaměstnanců.

V současné době se na trhu vyskytují také informační systémy (KARAT, Altus VARIO atd.) určené přímo pro potřeby zahradnictví. Jejich nástroje a funkce se zaměřují zejména na řešení maloobchodního prodeje, realizace zahrad a logistiky ve vazbě na využití identifikačních systémů s čárovými kódy.

Kromě již vytvořených informačních systémů nabízí většina softwarových firem možnost tvorby vlastního zákaznického systému podle konkrétních požadavků s možností implementace na již vytvořený identifikační systém.

Obě výše zmínění možnosti mají své výhody i nevýhody. Použití tzv. krabicového softwaru je sice o poznání levnější alternativou, ale možnosti a funkce, které nabízí, nemusí vždy zcela vyhovovat. Navíc implementace na již vytvořený identifikační systém se specifickými vlastnostmi může být značně komplikovaná a v některých případech i nerealizovatelná. Výhodnějším řešením proto může být využití

služeb softwarových firem, které ale ve srovnání s krabicovým softwarem bývá finančně náročnější.

Produkt této diplomové práce, informační systém Eurogarden, v sobě skýtá výhody obou přístupů, neboť je navržen dle potřeb zadavatele a specifikací již vytvořeného identifikačního systému. Určitým nedostatkem ve srovnání s komerčními produkty se může zdát neuniverzálnost systému, protože se jedná o zakázkový systém se specifickými vlastnostmi.

1.1 Popis identifikačního systému

Samotnému návrhu a realizaci identifikačního systému předcházelo studium nabídek a možností dostupných prvků a metod identifikace. Nakonec byl systém navržen a zrealizován v lednu roku 2008 ve spolupráci s firmou IKOS Liberec, s.r.o., která se od roku 1995 zabývá komplexním řešením v oblasti identifikačních technologií.

Ve spolupráci s hlavním poradcem firmy IKOS Liberec, s.r.o. s panem inženýrem Pavlem Balounem byl vybrán nejvhodnější a cenově nepříznivější způsob identifikace pomocí jednorozměrných čárových kódů.

1.1.1 Identifikace pomocí čárových kódů

Jednorozměrný čárový kód je tvořen posloupností čar a mezer s přesně definovanými šířkami, které lze pomocí optoelektronických zařízení analyzovat a reprezentovat jako příslušný kód. Čárové kódy jsou v současné době nejúčelnějším, nejrozšířenějším a stále ještě nejlevnějším způsobem identifikace pasivních prvků. Za určitou nevýhodu čárových kódů lze považovat nutnost optického kontaktu nosiče s čárovým kódem a optoelektronického snímacího zařízení. [1]

Nejpoužívanější typy čárových kódů:

- **EAN-13 a EAN-8** (European Article Numbering)
- **UPC-A a UPC-E** (Universal Product Code)
- **Codabar**
- **Code 128**

Vzhledem ke skutečnosti, že prodej rostlin v zahradnictví je určen koncovému zákazníkovi, nebylo nutné ani žádoucí používat mezinárodní čárové kódy typu EAN či UPC. Proto byl pro potřeby zahradnictví z nabídky čárových kódů zvolen číselný kód se zvláštními znaky CODE 128.

CODE 128

Jednorozměrný alfanumerický kód proměnné délky, který byl v roce 1980 vyvinut firmou Computer Identics pro použití v logistice či k označování patentů. Každý znak Code 128 se skládá ze tří čar a tří mezer definované šíře.

Pomocný znak	ID kategorie	Číslo rostliny
Signalizační symbol pro aplikaci, která další znaky detekuje jako kód rostliny.	Identifikační číslo kategorie rostliny.	Automaticky generované číslo, které je unikátní pro každou rostlinu.

Obrázek 1.1 - Význam jednotlivých symbolů v navrženém schématu Code 128

1.1.2 Nosiče čárových kódů

Jako nosiče čárových kódu a dalších informací o rostlině byly použity etikety standardizovaných rozměrů z bílého polyetylénu, které bez problémů odolávají povětrnostním podmínkám. Etikety se jednoduše nalepí na příslušný květináč či kontejner.

Obrázek 1.2 - Etiketa 50 x 25 mm

Obrázek 1.3 - Etiketa 50 x 18 mm

Některé rostliny jsou však umístěny přímo v zemi a nemají tak žádnou plochu k nalepení etikety. Tento problém byl vyřešen použitím visačky ze syntetického papíru, která rovněž odolává nepříznivým vlivům počasí.

Obrázek 1.4 - Visačka 260 x 25 mm (syntetický papír)

1.1.3 Tiskárna etiket, visaček a účtenek

Tiskárna DATAMAX E-4203 je určena zejména pro kancelářské použití v menších tiskových sériích. Ideální pro tisk štítků, etiket a visaček. Tiskárna komunikuje s PC přes paralelní port a USB rozhraní. Součástí balení je i CD obsahující tiskový software LabelView XLT pro návrh etiket, který byl využit pro vlastní návrhy. [2]

Obrázek 1.5 - DMX E-4203

Některé důležité technické parametry

Způsob tisku	Termotransfer nebo tepelný
Max.rychlost tisku	76 mm/s (3"/s)
Rozlišení	203 dpi
Typ média	Etikety v kotoučích, visačky, štítky, nepřetržité pásy

Podporované technologie tisku

Při termotraferové technologii tisku se požadovaný obsah nejprve vytiskne na speciální nosnou fólii a až po té působením tlaku a tepla na povrch finálního nosiče. Nosičem může být kromě papíru i tkaniny, plastické hmoty, porcelán atd. Navíc povrch nosičů může být speciálně upraven, což značně zvyšuje odolnost tisku.

Při tepelném tisku je nosná fólie s barvou v přímém kontaktu s papírem. K samotnému přenosu na papír dochází pouze v místech, kde je barvivo tepelnou tiskovou hlavou roztaveno. Vhodným nosičem je v tomto případě pouze papír.

1.1.4 Čtečka čárových kódů

AveScan AS-8000 od firmy Argox je CCD snímač čárových kódů. Lze jej k PC připojit přes rozhraní KBW, RS232 a USB-KBW. [3]

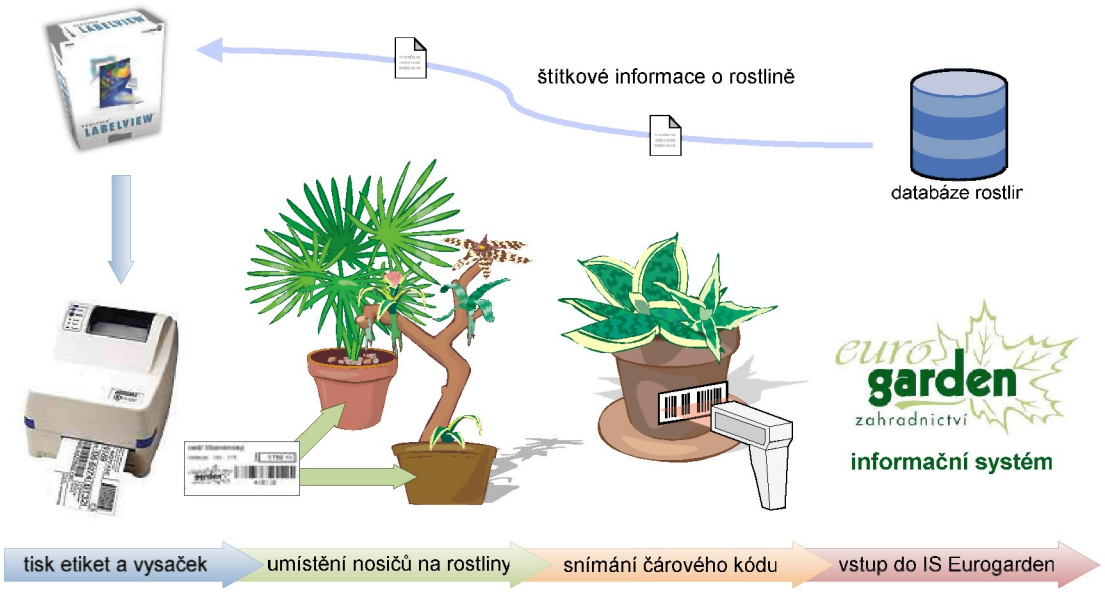
Obrázek 1.6 - AS-8000

Některé důležité optické parametry

Světelný zdroj	viditelná červená LED 660 nm
Optický systém	CCD 2048 pixel
Skenovací rychlost	100 sn./sec.

1.2 Identifikace v systému

Na základě databáze rostlin informačního systému Eurogarden byly prostřednictvím tiskového softwaru LabelView XLT vytvořeny návrhy etiket a visaček s čárovými kódy Code 128, které se následně na tiskárně DATAMAX E-4203 vytiskly na vhodné nosiče. Etikety a visačky pak zaměstnanci zahradnictví upevnili na rostliny, které lze přečtením čárových kódů čtečkou AS-8000 jednoduše identifikovat.



Obrázek 1.7 - Identifikační systém

V průběhu roku začaly etikety z bílého polyetylénu ztrácet své vlastnosti. I přesto že byl tisk realizován termotransferovým procesem, došlo v některých případech, zřejmě vlivem ultrafialového záření a velkým změnám teplot, k poškození popisků a čárových kódů, které se staly pro optoelektronické zařízení nečitelnými. Naproti tomu vlastnosti visaček ze syntetického papíru se zatím jeví jako stálé a teplotně neměnné. Další dotisk nosičů čárových kódů se bude raději realizovat výhradně na visačky.

Optoelektronické zařízení čtečka čárových kódů AveScan 8000 se na pokladně zahradnictví používá zejména při identifikaci většího množství rostlin, které se v pokladním systému automaticky přidají jako položky na účtenku. Při identifikaci několika kusů se využívá většinou jen ručního zadávání.

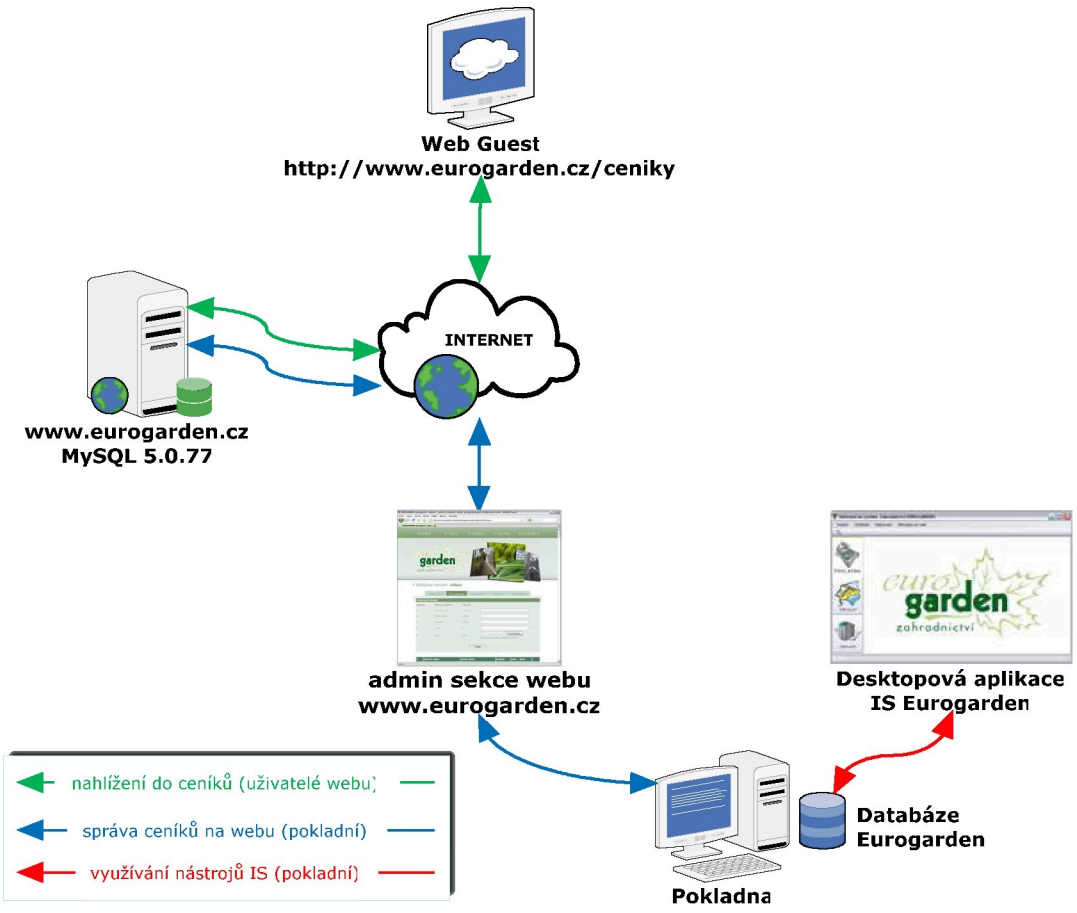
Tiskárna DATAMAX E-4203 je poměrně náročná na obsluhu. Postrádá intuitivní ovládání a pro většinu zaměstnanců zahradnictví je kritickým zařízením celého identifikačního systému.

1.3 Původní informační systém

Původní informační systém zahradnictví Eurogarden byl vytvořen před rokem jako součást řešení magisterského projektu. Obsahoval pouze základní funkce pro správu rostlin a evidenci jejich prodeje. Jednalo se o testovací verzi informačního systému, která měla zjistit, zda je rentabilní vytvořit vlastní plnohodnotný informační systém.

Po úspěšné implementaci identifikačního systému a následném nasazení testovací verze informačního systému proběhlo školení pracovníků zahradnictví, kde bylo vše řádně vysvětleno, názorně ukázáno a odzkoušeno.

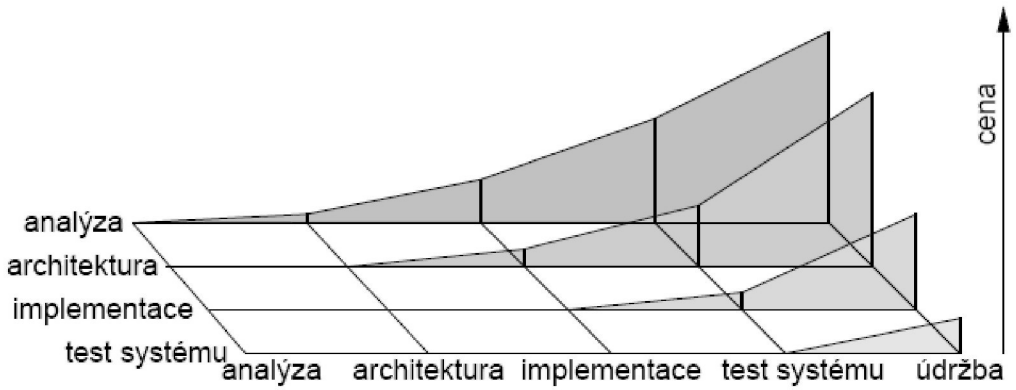
V prvních měsících provozu bylo ještě nutné udělat několik algoritmických úprav, což nikterak neohrozilo chod firmy. Samotná databázová aplikace s databázovým serverem MS SQL 2000 MSDE byla nasazena pouze na pokladně zahradnictví a během používání nadchla svými funkcemi a uživatelsky přívětivým rozhraním pokladní i majitele. Velkým nedostatkem ve spojení s webem byla zdvojená správa rostlin, při které se musely informace o rostlinách zadávat dvakrát, jak je zřejmého z následujícího obrázku.



Obrázek 1.8 - Zdvojené procesy správy rostlin

2. Seznámení s uživatelskými požadavky

Specifikace a analýza požadavků je jedna z nejdůležitějších částí životního cyklu softwaru, neboť případné chyby v této části cyklu vedou k problémům při designu a implementaci. Vyskytnou-li se v koncových fázích chyby, které vznikly defekty v této části životního cyklu, náklady na jejich odstranění zpravidla bývají až dvacetkrát vyšší. Proto obecně platí snaha vyřešit případné problémy a chyby co nejdříve.



Obrázek 2.1 - Náklady na odstranění defektů [4]

Aby bylo možné vytvořit efektivní softwarový nástroj pro potřeby zahradnictví, bylo nezbytné získat od zadavatele podrobný popis reálného světa zahradnictví. K získání podrobného popisu často předcházejí dialogy, ve kterých se vyjasňují rozpory, neúplnosti, nejasnosti zadání a případné reakce na nepřípustná data. Většina zadavatelů totiž často zprvu neví, co přesně požaduje. Mají jen rámcovou představu o požadovaném přínosu softwaru, ale k jednotlivých funkcím a nástrojům málokdy poskytnou dostatečný popis. [4]

Zadavatel požaduje, aby realizovaný softwarový nástroj obsahoval specifikované funkce pro správu rostlin a účtenek. Požadované funkce musí být dostupné v rámci jednoho softwarového řešení a jejich používání by mělo být dostatečně jednoduché a intuitivní.

2.1 Požadované funkce pro správu rostlin

Zadavatel následujícím způsobem specifikoval požadavky na funkce, které souvisejí se správou rostlin. Kromě požadovaných funkcí byly dále popsány i vlastnosti, vztahy a omezení, která se vyskytují v reálním světě zahradnictví.

2.1.1 Uchování informací o rostlinách včetně jejich fotografie

Pro potřeby již vytvořeného identifikačního systému je potřeba uchovávat informace o rostlinách, aby byla možná jejich následná automatická identifikace. Každá rostlina je jednoznačně identifikována unikátním číselným kódem, který obsahuje pětimístné číslo. Kromě čárového kódu je potřeba uchovávat další informace o vlastnostech, které rostlině přísluší, český a latinský název, cenu, počet kusů, výškový interval a další vlastnosti.

Český název se skládá z jednoho či více slov, která obsahují znaky české abecedy a jejichž délka bude maximálně 80 znaků. Jeden český název může být shodný pro více rostlin, které se mohou lišit v kultivaru, velikostech, rozměrech kontejneru atp. Latinský název rostliny nemusí mít pevný formát používaný v botanice. Jedná se spíše o doplňující informaci o daném kultivaru. Z čehož tedy vyplývá, že jeden latinský název může mít více různých rostlin. Latinský název rostliny se skládá z jednoho či více slov, která obsahují znaky s diakritikou a jejichž délka bude maximálně 100 znaků.

Číselná hodnota počtu kusů a ceny musí být pro každou rostlinu známa. Stejně tak informace o datu a času poslední úpravy informací o rostlině. Ke každé rostlině lze připsat poznámku, text s maximálně 500 znaky, který bude blíže popisovat vlastnosti dané rostliny.

Zadavatel rovněž požaduje, aby bylo možné ke každé rostlině kdykoliv dodat její fotografii. Tudíž fotografie rostlin se budou dodávat postupně a je možné stejnou fotografii přiřadit více rostlinám.

2.1.2 Přidávání nových rostlin a úprava stávajících informací

Je nezbytné vytvořit uživatelsky přívětivé rozhraní pro přidávání nových rostlin. Informace budou do systému vkládat zaměstnanci zahradnictví, kteří disponují základními znalostmi a zkušenostmi práce na PC. Realizované řešení proto musí být dostatečně intuitivní, aby nedocházelo ke zbytečným časovým prodlevám při naskladnění nových rostlin. Rozhraní by mělo být realizováno pomocí formulářových

prvků, jejichž hodnoty se pomocí vstupních zařízení vyplní. Formulářová pole určená pro číselné hodnoty musí obsahovat kontrolu vstupních dat, aby nebylo možné vložit nečíselné znaky. Dále je požadováno, aby měl uživatel při zadávání kategorie možnost výběru z již vytvořených kategorií případně možnost vytvořit kategorii novou, při jejíž tvorbě musí uživatel zadat DPH kategorie pro prodej rostliny. Řešení musí obsahovat i možnost načtení obrázku přidávané rostliny a jeho uložení.

Dále je požadováno aby realizace rozhraní pro úpravu informací o stávajících rostlinách byla vytvořena dle vzoru rozhraní pro přidávání nové rostliny. Důvodem je snazší a rychlejší orientace uživatelů v systému.

2.1.3 Automatické generování číselného kódu rostliny

Při přidávání nové rostliny je nezbytné, aby se jí automaticky přiřadilo unikátní číselné označení, pod kterým bude možné rostlinu automaticky identifikovat. Struktura pětimístného číselného kódu vychází s požadavků identifikačního systému, přičemž první dvě číslice specifikují číslo kategorie rostliny a zbylé 3 číslice pak nejnižší neobsazené pořadové číslo rostliny v příslušné kategorii. Je tedy zřejmé, že maximální počet kategorií rostlin je 99 a počet různých rostlin v každé kategorii může být maximálně 999. Maximální celkový počet kódů rostlin je pak tedy 99999.

2.1.4 Vyhledávání rostlin dle zvolených kritérií

Pomocí standardní ovládacích prvků je potřeba obohatit systém o funkce a nástroje pro vyhledávání, aby mohl uživatel informačního systému snadno a rychle získat vložené informace o některé z rostlin. Kromě čárového kódu lze hledat také podle názvů, ceny, počtů kusů, velikosti a kategorií rostlin. Funkce vyhledávání bude často používána pokladní, při řešení zákaznických požadavků a proto musí vyhledávání obsahovat i funkci automatického přidávání nalezených rostlin na účtenku.

2.1.5 Automatická aktualizace ceníků na webu

Nástroj automatické aktualizace ceníků by měl být schopen na základě uživatelského příkazu automaticky zaktualizovat nabídku rostlin na webovém serveru zahradnictví Eurogarden. Není však žádoucí na webu zpřístupňovat příliš podrobné informace o rostlinách či snad synchronizovat celé databáze. Pro potřeby návštěvníků webových stránek postačí pouze zpřístupnit základní informace o rostlinách, jakými jsou český a latinský název, výškové rozmezí, kategorie a případná ilustrace.

2.2 Požadované funkce pro tvorbu a správu účtenek

Je zřejmé, že by samotná evidence rostlin s nástroji pro její správu potřebám zahradnictví nepostačovala, neboť se zahradnictví zabývá převážně maloobchodním prodejem. Z tohoto důvodu zadavatel požaduje, aby výsledné softwarové řešení obsahovalo i nástroje pro tvorbu a správu účtenek, jejichž položkami budou právě rostliny uchované v systému. Uživatel následujícím způsobem specifikoval požadavky pro tvorbu a správu účtenek.

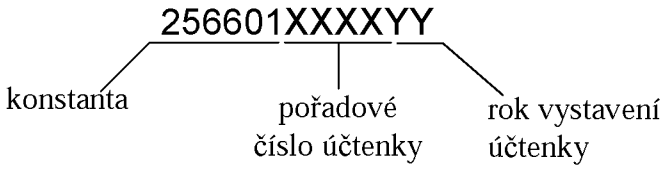
2.2.1 Automatické sestavení, uložení a tisk účtenek

Zadavatel požaduje, aby softwarové řešení implementované na PC pokladny bylo schopné automaticky sestavovat účtenky, vytisknout je na tiskárně DATAMAX E-4203, ukládat je ve formátu pdf a zároveň informace o nich ukládat pro pozdější tvorbu účetních sestav. Stejně jako u požadavků na správu rostlin tak i zde je kladen důraz na jednoduché a intuitivní ovládání.

Při vytváření účtenky je potřeba, aby každá účtenka měla své jednoznačné identifikační číslo, které bude striktně dodržovat posloupnost již vytvořených účtenek. Dále aby účtenka obsahovala datum vystavení a číselné hodnoty, které budou určovat celkovou cenu, celkový počet prodaných kusů a případnou slevu. Rovněž je kladen požadavek na uchování informací o zaměstnancích, kteří účtenky vystavili.

2.2.2 Automatické generování číselného kódu dle účetní specifikace

Při tvorbě nové účtenky je nutné, aby se každé účtence přiřadil unikátní číselný kód, který bude přesně odpovídat účetní specifikaci a striktně dodržovat číselnou posloupnost účtenek.



Obrázek 2.2 - Struktura kódu účtenky

Jak je zřejmé z ilustrace, číselný kód označující účtenku obsahuje 12 cifer, kde prvních šest představuje účetní konstantu, další 4 pořadové číslo účtenky a poslední dvě cifry reprezentují označení roku, ve kterém byla účtenka vystavena. Maximální počet účtenek vystavený v jednom roce je pak tedy 9 999. Podle testovací verze informačního systému má tento rozsah dostatečnou rezervu, neboť během roku 2008 počet vystavených

účtenek nepřesáhl 1500. Informace o roku vystavené účtenky je nutná, protože s novým rokem se začne generovat nová posloupnost účtenek.

2.2.3 Vyhledávání účtenek a jejich zobrazení

Pro případné zpětné ověřování vystavených účtenek je potřeba vytvořit nástroj pro jejich vyhledávání, který bude nabízet možnost vyhledávání dle základních kritérií. Nejjednoznačnějším vyhledávacím kritériem je číselný kód účtenky, podle kterého musí být možnost nalézt jakoukoliv účtenku. Dalším kritériem pro vyhledávání by měl být časový interval vystavení účtenek, interval celkové ceny a celkového počtu položek. Informace o nalezených účtenkách stejně tak jako jejich pdf vizualizace by mělo být snadné zobrazit či opětovně vytisknout.

2.2.4 Stornování účtenek

Kromě zobrazení informací o nalezených účtenkách je potřeba vytvořit nástroj pro stornování účtenek, neboť v reálném světě zahradnictví je někdy potřeba z různých důvodů účtenku stornovat. Stornovaná účtenka se ale nesmí z evidence vymazat, na místo toho se jí přiřadí pouze příznak stornováno, na jehož základě se pak k účtenkám přistupuje.

2.2.5 Automatické generování účetních sestav za zvolené období

K hlavním přínosům celého sytému patří zejména úspora finančních nákladů, které je nutné vynakládat za využití účetních služeb. Tvorba účetních sestav a přehledů, které jsou nezbytné pro případnou kontrolu, je značně časově náročná je-li realizována výhradně lidskými silami a možnostmi. S vysokou časovou náročností se pak úměrně zvyšuje i cena poskytnutých účetních služeb. Je tedy zřejmé, že úspora finančních prostředků je v tomto místě výhodná a nepříliš složitá.

Nástroj pro generování účetních sestav by měl být schopen na základě nastaveného časové intervalu sestavit tabulku, kde sloupce budou obsahovat kódy prodaných rostlin, celkové součty za den a celkové součty DPH kategorií. Řádky tabulky pak budou obsahovat zkonsolidovanou tržbu jednotlivých rostlin za daný den. Poslední řádek pak bude obsahovat celkové součty sloupců.

datum	10001	Celkem	Celkem 9%	DPH 9%	Základ 9%	Celkem 19%	DPH 19%	Základ 19%
1.9.08		6974	6485	535,5	5949,5	489	78,1	410,9
2.9.08	100	0	0	0	0	0	0	0
3.9.08		5164	4727	390,3	4336,7	437	69,8	367,2
4.9.08		4578	4393	362,7	4030,3	185	29,5	155,5
≈	≈	≈	≈	≈	≈	≈	≈	≈
27.9.08		3939	3665	302,6	3362,4	274	43,7	230,3
28.9.08	50	0	0	0	0	0	0	0
29.9.08		2043	1995	164,7	1830,3	48	7,7	40,3
30.9.08		2075	2075	171,3	1903,7	0	0	0
Součet	**250**	**28943**	**26940**	**2224,3**	**24715,7**	**2003**	**319,8**	**1683,2**

Tabulka 2.1 - Přehled prodaných roslin za dané období

Zároveň s tabulkou přehledů prodaných rostlin se musí vygenerovat tabulka celkový souhrn DPH, která obsahuje celkovou DPH rekapitulaci za zvolené období.

Sazba v %	Základ daně v Kč	DPH v Kč
9	24 715,60	2 224,40
19	1 683,20	319,80

Tabulka 2.2 - Celkový souhrn DPH

Další tabulkou by měl být souhrn účtenek za dané období, ve kterém budou jednotlivé položky účtenek rozepsané a zkonsolidované dle kategorií DPH a jejich celkových součtů. Musí být možnost automaticky vygenerované tabulky vyexportovat do programu Microsoft Excel. Dále je nezbytné, aby se účtenky s příznakem storno do celkových sestav nezapočítávaly.

datum	č.ú.	DPH 9%	DPH 19 %	celkem	DPH 9%	základ 9%	DPH 19%	základ 19%	základy celkem	DPH celkem
1.9.08	64528	90		90	7,43	82,57			82,57	7,43
1.9.08	64628	500		500	41,28	458,72			458,72	41,28
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
29.9.08	82428	400	30	430	35,5	394,5			394,5	35,5
celkem	180	9440	7070	85510	676,65	963,35	28,72	941,28	7904,63	7605,37

Tabulka 2.3 - Přehledy účtenek za zvolené období

2.2.6 Automatické generování grafů prodeje rostlin

Majiteli zahradnictví je potřeba zpřístupnit nejlépe formou sloupcových grafů stav prodeje za jednotlivé dny v aktuálním měsíci a celkový prodej za jednotlivé měsíce.

3. Nástroje a technologie

3.1 Databázový systém a jazyk SQL

Databázový systém (DBS – database system) je složen ze samotné databáze obsahující konkrétní data a systému řízení báze dat SŘBD (DBMS – database managment system), který představuje skupinu programů spravujících databázi. Sklad informací nebo-li databáze je množina strukturovaných a uspořádaných souborů, které jsou uložené nejčastěji na pevných discích počítačů. Tyto soubory mohou obsahovat velká množství dat, s nimiž lze podle potřeby manipulovat nebo v nich efektivně vyhledávat. Jakákoliv manipulace s daty v databázi probíhá prostřednictvím jazyka SQL nebo pomocí již zmíněné skupiny programů, které přímo určují možnosti příslušného databázového systému.

Databázové systémy se začaly výrazněji rozvíjet v 60. letech, kdy se používaly zejména síťové datové modely, které byly založeny na souborech a vztazích mezi nimi. Koncem 60. let a začátkem let 70. vyvinula firma IBM první verzi hierarchického modelu dat, ve kterém spojitosti mezi daty bylo možné reprezentovat pomocí stromové struktury. Nevýhody obou datových modelů byly vyřešeny až s nástupem relačního modelu dat, který je založen na pevném matematickém základu. [5]

3.1.1 Relační databáze

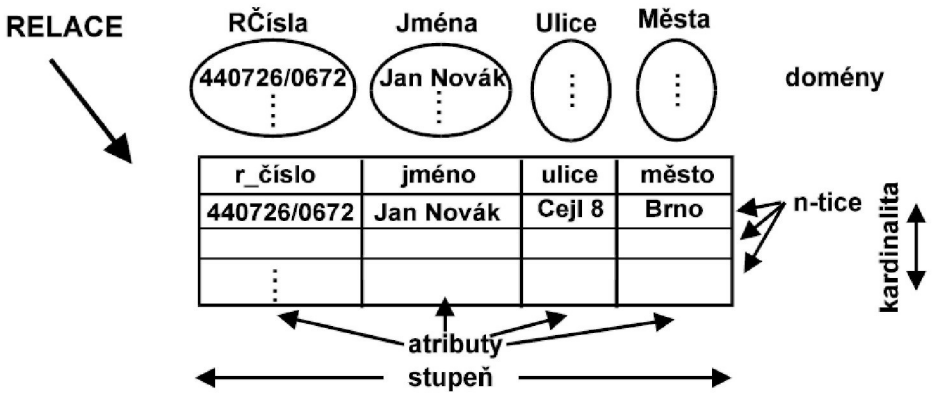
Relační databáze jsou založeny na relačním modelu dat (RMD), který vychází z teorie množin a predikátové logiky. Relační model dat navrhl a publikoval v roce 1970 Dr. E. F. Cood v článku: *„A relational model of data for large shared databanks“*. Článek mimo jiné obsahuje i následující ideje:

- Relační model dat odděluje data od jejich implementace.
- Přístup k datům je symetrický. Při manipulaci s daty nás nezajímají přístupové metody k datům.
- Pro manipulaci s daty je k dispozici relační kalkul a relační algebra.
- Pro omezení redundance jsou k dispozici pojmy umožňující normalizovat relace. (Vhodně navrhovat databázové struktury.)

Relace ***R*** v RMD lze podle definice [6] chápat jako uspořádanou trojici < ***A, D, T***>, kde

- ***A*** je konečná množina jmen atributů
- ***D*** je zobrazení, které každému jménu atributu $a \in A$ přiřadí neprázdnou množinu (obor hodnot atributu), která se nazývá doména atributu *D(a)*
- ***T*** je konečná podmnožina kartézského součinu domén *D(a)* příslušejících jednotlivým atributům $a \in A$

Samotná datová struktura databáze je reprezentována pomocí tabulek, kde každá tabulka je sestavena z řady sloupců (atributů a příslušných domén) a řádků nebo-li n-tic, které představují n rozměrný vektor neboli záznam.



Obrázek 3.1 - Relační struktura dat [7]

Relační databáze sestávající z řady tabulek často obsahuje sloupce, které jsou vázány na sloupce v jiných tabulkách. Takto propojená datová pole jsou na sobě určitým způsobem závislá. Jejich vztahy jsou založeny na klíčových hodnotách uložených v příslušných sloupcích. Tabulky jsou mezi sebou ve vzájemném vztahu (relaci). Relační databáze napodobuje způsob myšlení lidí. Lidé mají totiž sklon seskupovat podobné objekty do systematických celků a rozkládat složité objekty na jednodušší. Povaha relační databáze je ve skutečnosti úplně stejná.

Logickou strukturu informací uložených v databázi popisuje „Logické schéma relační databáze" (zkráceně „schéma"). Schéma relační databáze se skládá z definic „objektů schématu", to jsou jednak definice tabulek a integritních omezení, dále definice dalších (do relačního modelu přímo nepatřících) objektů jako jsou indexy, uložené procedury a podobně. Definice objektu schématu znamená vytvoření vazby mezi jedinečným jménem v rámci schématu a definicí objektu. V rámci definice je možné jednotlivým objektům schématu nastavovat parametry určené aktuální

implementací DB systému. Z objektů schémata jsou nejdůležitější tabulky (table), omezení (constraint), indexy (index), triggery (trigger) a uložené procedury (stored procedure). Tabulky slouží jako logické struktury pro uložení dat, constraint a trigger slouží k deklarativnímu (constraint) a procedurálnímu (trigger) vyjádření integritních omezení, indexy slouží k ovlivnění výkonu některých operací (například vyhledávání a řazení) a uložené procedury slouží k implementaci algoritmů pro manipulaci s daty v rámci databázového serveru.

Kromě logického schématu je možné ovlivnit také fyzickou strukturu databáze, například uložení databáze na disku počítače. Parametrizace a ovlivnění fyzické struktury databáze úzce souvisí s operačním systémem a hardwarovým vybavením použitým k provozu databázového serveru. Formát uložené databáze v souborech operačního systému obvykle není možné vůbec ovlivnit. Také není možné přenášet databázi (s výjimkou nejjednodušších DB systémů jako MS Access) z jednoho počítače na druhý pouhým zkopírováním souborů databáze. Pro přenos dat je vytvořený „přenosný“ formát, který se musí ze zdrojové databáze vytvořit a z kterého se musí cílová databáze zrekonstruovat (export a import dat).

Úložiště metadat v databázi se obvykle nazývá systémový katalog. Nejdůležitější složkou metadat je logické schéma a fyzická struktura databáze. Systémový katalog navíc slouží k ukládání provozních informací o stavu a výkonu databáze, přidělených prostředcích operačního systému pro výkon dané databáze a podobně. Informace ze systémového katalogu jsou pro klienty většinou k dispozici jen pro čtení. Logická struktura systémového katalogu je opět nejčastěji tabulková.

Libovolný klient přistupuje k datům v databázi podle logické struktury definované schématem. Má k dispozici jedině operace pracující s jednotlivými objekty schématu a jejich datovým obsahem. Jakou operaci nebo jakou informaci klient konkrétně po databázi požaduje, se vyjádří formou věty jazyka SQL (dotazu). Pro pojmenování operandů (objektů schématu) je v rámci databázového serveru vytvořený jmenný systém. Kompletní jméno objektu je obvykle složené ze jména databáze, jména schématu a jména objektu v rámci schématu. Jednotlivé části jména jsou oddělené vhodným oddělovačem, nejčastěji tečkou. V případě, že jméno operandu není úplně celé specifikované, databázový server si „domysli“ celé jméno podlé kontextu aktuální komunikace s klientem. [8]

3.1.2 Historie a popis jazyka SQL

Jazyk *SQL (Structured Query Language)* byl vyvinut zaměstnanci společnosti IBM někdy v polovině 70. let s úmyslem poskytnout účinný nástroj pro manipulaci s daty uloženými v relačních databázích. Práce s daty v databázích do té doby nebyla snadná. Prvořadým cílem nového jazyka bylo jeho snadné použití a syntaxe blízká anglickému jazyku. Vývojáři v laboratořích IBM vycházeli z předpokladu, že obchodníci, podnikatelé a zaměstnanci budou s databází "hovořit" srozumitelným jazykem a jen tak, jakoby mezi řečí, ji požádají o příslušné informace. V začátcích neměl jazyk SQL na růžích ustláno. Vše se ovšem rázem změnilo, když byl poprvé použit jako jazyk pro přístup k datům v databázích Oracle. Prakticky od té doby je standardem pro přístup k datům uloženým v relačních databázích. [8]

Jazyk SQL se více blíží ke komunikačnímu než k programovacímu jazyku. Kód většiny programovacích jazyků musí být nejprve přeložen příslušným překladačem a teprve pak může být spouštěn. Kód jazyka SQL však zpravidla překládán není. Lidé si při vzájemné komunikaci kladou otázky a pokud znají odpovědi, odpovídají na ně. SQL funguje naprosto stejně. Aby bylo možno s databází komunikovat, je nutné nejprve otevřít komunikační kanál. Systém MS SQL 2000 k tomu poskytuje program SQL Server Service Manager.

Jazyk SQL lze rozdělit na dvě základní skupiny příkazů. Jedna definuje syntaxi tvorby databázových objektů a druhá obsahuje příkazy pro manipulaci s daty. První část9 se říká jazyk DCL *(Data Creation Language)* neboli syntaxe tvorby dat někdy známa jako DDL *(Data Definition Language)*. Je to stále jazyk SQL, ale příkazy této skupiny se vztahují pouze ke tvorbě objektů. Druhá skupina obsahuje příkazy, které jsou naopak určeny pouze pro manipulaci s daty. Tato skupina se označuje jako jazyk DML *(Data Manipulation Language)* neboli syntaxe manipulace s daty.

Vzhledem k tomu, že SQL je jazyk, má také určitá pravopisná pravidla. Existuje také slovní zásoba slov používaných v jazyce SQL. Tato slova se označují jako vyhrazená slova (reserved words), neboť mají speciální význam. Prvním slovem v každém příkazu jazyka SQL musí být vyhrazené slovo. Obvykle jde o sloveso, které sděluje databázi, co je po ní požadováno. Slova, která následují za vyhrazeným slovesem, jsou seznamem argumentů nebo instrukcí upřesňujících informace, které se mají zobrazit nebo změnit atd. V seznamu argumentů příkazu SQL jsou názvy tabulek, sloupců atd. [9]

SQL je velmi výkonný a užitečný nástroj pro vyhledávání a získávání dat. Jeho použití je velmi snadné, vložíme příkazy SQL do kódu aplikace. Jejich spouštění z prostředí koncové aplikace se ničím neliší od spouštění na příkazovém řádku. Přesná a úplná syntaxe jednotlivých příkazů je k dispozici v definici jazyka SQL92. [9]

Příkaz	Význam
CREATE TABLE	Vytvoření nové tabulky
ALTER TABLE	Modifikace existující tabulky
INSERT INTO <jméno tabulky> (<seznam sloupců>) values (seznam hodnot)	Vložení nového záznamu do tabulky
UPDATE <jméno tabulky> SET <seznam přiřazení> where <podmínka>	Aktualizace jednoho nebo více řádků tabulky
SELECT <seznam sloupců> FROM <jméno tabulky> WHERE <podmínka> ORDER BY <seznam sloupců>	Výběr záznamů z databázové tabulky

Tabulka 3.1 – Příkazy jazyka SQL použité při realizaci aplikace [9]

Firma Microsoft implementovala jazyk SQL do Transact-SQL (T-SQL), které umožňuje vytvářet dodatečné programovací konstrukce. Programy napsané v T-SQL se skládají z příkazů SQL a standardních programovacích objektů:

- **Proměnné**
  Umožňují ukládat hodnoty v paměti počítače. Jméno proměnné musí na svém začátku obsahovat znak zavináč *@promenna*. Před použitím se musí proměnná deklarovat pomocí příkazu **DECLARE**. Do proměnné lze přiřazovat pomocí příkazů **SET** a **SELECT.**
- **Podmíněná logika**
  Logické větvení kódu pomocí klíčového slova **IF** případně **ELSE** nebo pomocí příkazu **CASE, WHEN, THEN** pro porovnání jedné hodnosty se seznamem.
- **Cyklus WHILE**
  V cyklu **WHILE** lze využít i příkaz **CONTINUE,** který způsobí skok na začátek cyklu nebo příkaz **BREAK**, který okamžitě cyklus ukončí.
- **Další funkce a příkazy GOTO, RETURN, WAITFOR, RAISERROR**
  Bližší popis syntaxe jazyka T-SQL a další možnosti, lze nalézt ve specifikaci.

3.1.3 Databázový systém MS SQL Server 2000

V posledních letech je stále více kladen důraz na získávání dat a jejich efektivní zpracování. Společnost Microsoft před několika lety uvedla na trh software Microsoft SQL Server 2000, který je vhodným nástrojem pro práci s daty. Dnes je již dostupná verze Microsoft SQL Server 2008. Microsoft SQL Server 2000 (dále jen SQL Server) je databázový a analytický systém určený pro elektronické obchodování a zpracování dat. Patří mezi jedny z nejvýkonnějších relačních databázových systémů, které dnes velmi často tvoří základní databázovou vrstvu podnikových informačních systémů.

SQL Server 2000 obsahuje několik grafických nástrojů pro správu. Pomocí většiny těchto nástrojů můžeme spravovat lokální i vzdálené zdroje. Například v SQL Server Enterprise Manager lze zaregistrovat nový server a pak se k němu připojit. Potom je možné server a všechny jeho databáze spravovat na dálku ze svého počítače. Přehled hlavních grafických správcovských nástrojů používaných při realizaci aplikace a účel jejich využití je v tabulce. [10]

Správcovský nástroj	**Účel**
SQL Server Query Analyzer	Vizuální utilita pro vytváření dotazů a skriptů v SQL. Vhodná pro spouštění jakýchkoli příkazů SQL, kontrolu dotazů nebo analýzu indexů.
SQL Server Enterprise Manager	Hlavní správcovský nástroj SQL Serveru 2000. V něm lze spravovat servery SQL, databáze, zabezpečení a jiné.
SQL Server Service Manager	Slouží ke správě a konfiguraci služeb SQL.
SQL Server Network Utility	Slouží ke konfiguraci síťových knihoven.

Tabulka 3.2 – Použité nástroje pro správu SQL serveru

3.2 Microsoft .NET

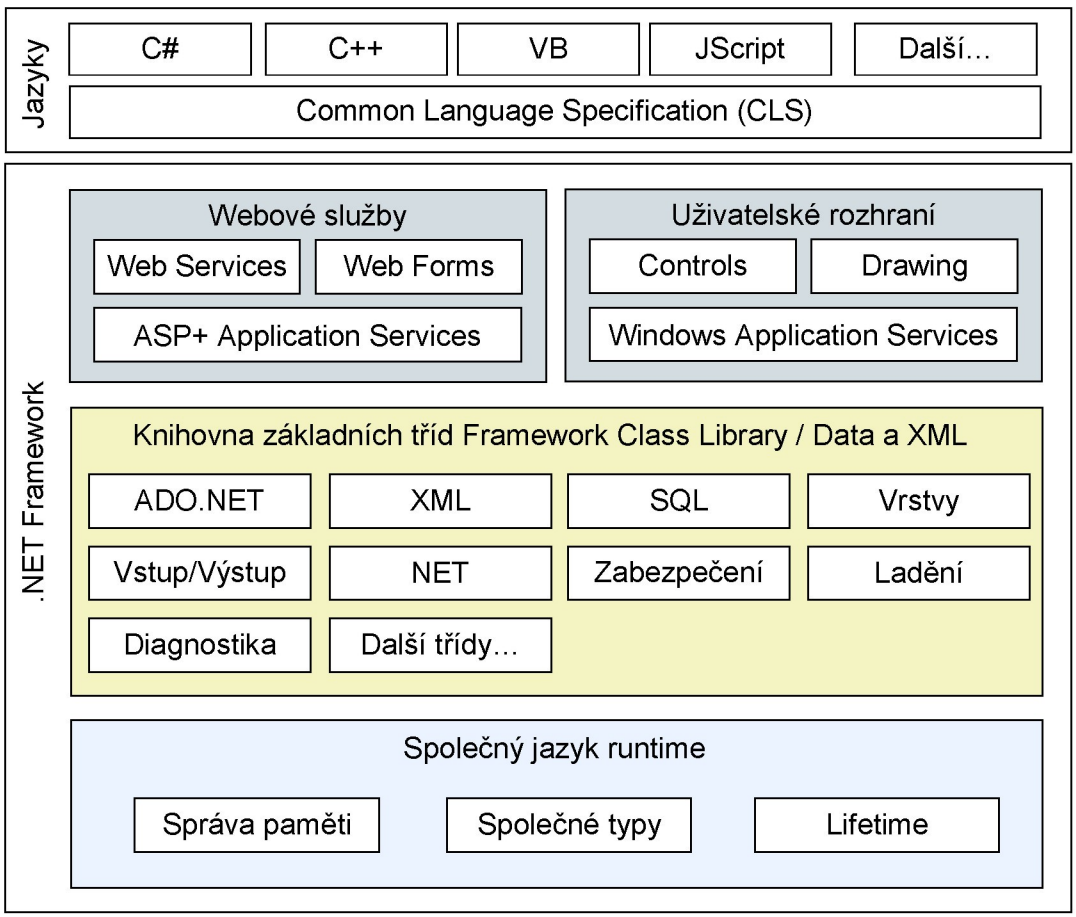
Microsoft .NET představuje iniciativu společnosti Microsoft, jejímž výsledkem je platforma .NET, která byla oficiálně uvedena firmou Microsoft v roce 2000 jako sada softwarových technologií, jejichž cílem je propojení světa informací, lidí, systémů a zařízení, kde hlavním pojivem má být XML. Tím Microsoft .NET představuje novou generaci vývoje aplikací pro operační systémy Windows založeném na rozsáhlé sadě základních tříd s názvem .NET Framework.

Vývoj trval přibližně čtyři roky a hlavními důvody vzniku .NET byla nekompatibilita jednotlivých programovacích jazyků a s tím související obtížná spolupráce mezi programy a knihovnami napsanými v odlišných jazycích (např. C++ a Visual Basic), vysoká chybovost aplikací, problémy s verzemi knihoven, zastaralý a nepřehledný způsob vývoje dosavadních webových aplikací. [11]

Platforma .NET obsahuje vše, co je potřebné pro vývoj a provoz aplikací – sadu vývojových nástrojů, programovacích rozhraní a stavebních prvků, odpovídajících serverových (jak operačních, tak aplikačních) prostředí a podporu celé řady zařízení.

3.2.1 Microsoft .NET Framework

.NET Framework (rámec .Net) je platforma pro vytváření a provozování aplikací .NET. .NET Framework je tvořen několika vrstvami. Na nejnižší úrovni se nachází společný jazykový modul *Common Language Runtime* (CLR) realizující základní infrastrukturu, na které je celý .NET Framework vybudován. Nad CLR se nachází knihovna tříd rámce .Net *Framework Class Library* (FCL) a knihovna pro podporu přístupu k datům a práci s XML. Na nejvyšší úrovni .NET Framework jsou dvě knihovny usnadňující tvorbu webových aplikací a tvorbu klasických aplikací s uživatelským rozhraním.



Obrázek 3.2 – Struktura .NET Framework a navazující jazyky

Platforma .NET se snaží o jazykovou nezávislost a proto .NET Framework také obsahuje část nazvanou *Common Language Specification* (CLS), která popisuje základní vlastnosti očekávané od všech programovacích jazyků na platformě .NET. V současné době jsou podporované jazyky Visual Basic, C#, C++ a JScript.

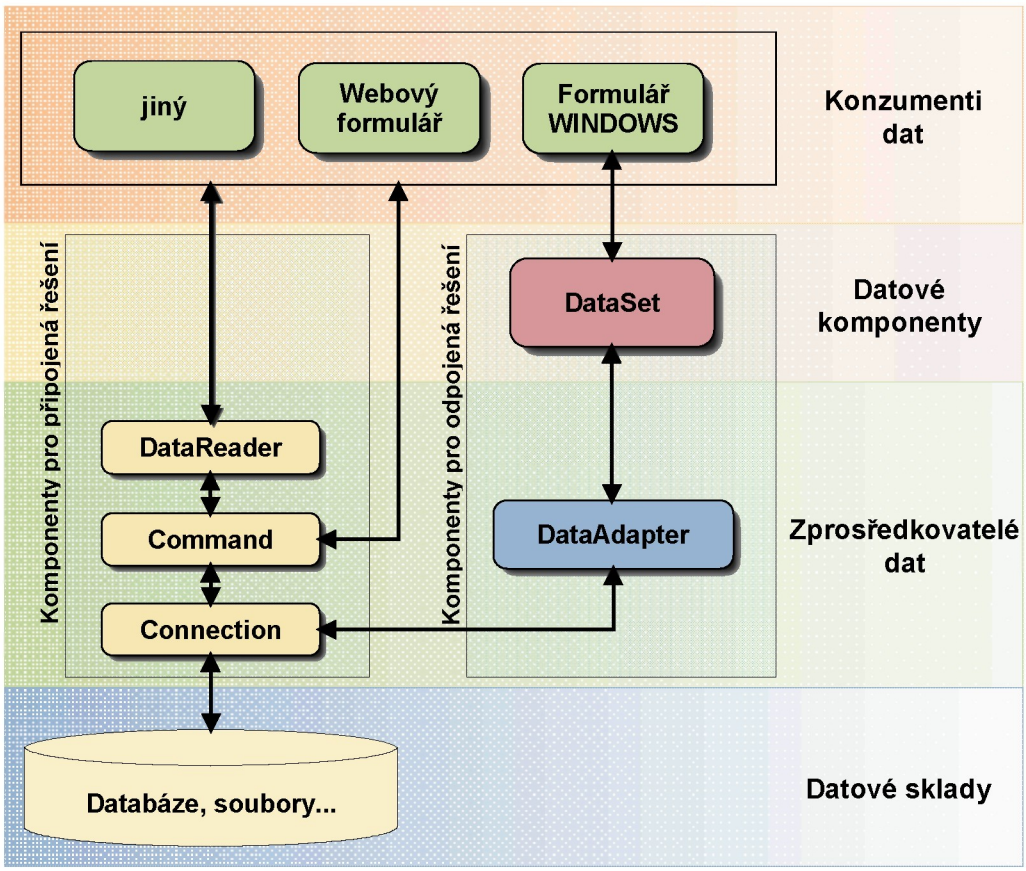
.NET Framework tvoří spolu s MS Visual Studiem ucelené vývojové prostředí pro tvorbu webových XML služeb. Jádrem celé platformy je právě sada programovacích rozhraní .NET Framework – vedle celé řady tříd obsahuje také obecný jazykový runtime, který řídí provádění kódu napsaného v libovolném podporovaném programovacím jazyce.

První verze platformy .NET Framework byla vydána v roce 2002 a nese označení .NET Framework 1.0. Kromě rozsáhlé knihovny tříd obsahovala i rozsáhlou podporu jazyků a Visual Studio .Net 2002. V průběhu vývoje získávala technologie. NET Framework nové třídy a značně se rozšířilo portfolio podporovaných služeb. V současné době je nejnovější verzí Microsoft .NET Framework s označením 3.5, která navazuje na předchozí verzi s označením 3.0. Tato verze podporuje nové Microsoft technologie: Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), Windows Workflow Foundation (WF), Windows CardSpace a další. Nejnovější verze .NET Framework 3.5 už byla obohacena i o Sevis Pack 1, který přináší mnoho funkcí a celou řadu vylepšení. [12]

3.2.2 Objektový model ADO.NET

Jak je patrné ze struktury technologie .NET Framework ADO.NET představuje skupinu tříd, která umožňuje pracovat s daty. Následující diagram zobrazuje zjednodušený pohled na primární objekty objektového modelu ADO.NET. Třídy ADO.NET se dělí na dvě hlavní skupiny:

- Zprostředkovatelé dat (Data Providers, Manager Providers)
- Sada dat (DataSet) a z ní čerpající konzumenti dat



Obrázek 3.3 – Diagram primárních objektů ADO.NET

Zprostředkovatelé dat [9,12]

Součásti zprostředkovatelů dat jsou specifické vzhledem ke zdroji dat. .NET Framework obsahuje dva zprostředkovatele dat. Obecného zprostředkovatele, který umí komunikovat s jakýmkoli zdrojem dat OLE DB, a zprostředkovatele SQL Serveru. Oba zprostředkovatelé obsahují stejné objekty, i když se jejich názvy i některé vlastnosti a metody liší. Pro svou aplikaci jsem využíval objekty zprostředkovatele SQL Serveru.

- SqlConnection
- SqlCommand

3.2.3 Microsoft Chart Controls

Společnost Microsoft v roce 2008 volně zpřístupnila nový ovládací prvek **Microsoft Chart Controls**, který umožňuje snadnou a efektivní tvorbu grafů v ASP.NET a WinForms aplikacích. Knihovna je určena pro rozhraní .NET Framework 3.5 SP1. Samotné vizualizační prvky vycházejí z technologie, kterou firma Microsoft získala zhruba před rokem akvizicí části firmy Dundas. Microsoft Chart Controls bude součástí nového .NET Frameworku s označením 4.0. Kromě snadné tvorby grafů nabízejí nové prvky i podporu cachování, ukládání vygenerovaných grafů ve formě obrázků na disk, různé kategorie grafů (sloupcový, pruhový, prostorový, finanční, koláčový, atd.) a mnoho dalšího.

Společnost Dundas kromě knihovny *Dundas Chart for .Net*, která byla využita při realizaci webové aplikace IS Eurogarden, nabízí další zajímavé nástroje pro řešení některých vizualizací WinForms nebo webových aplikací. Pro vizualizaci dynamických dat společnost Dundas nabízí knihovnu *Dundas Guage for .NET* a pro digitalizaci geografických dat knihovnu *Dundas Map for .NET*. [13]

3.2.4 ASP.NET

ASP (Active Server Page) je skriptovací platforma od společnosti Microsoft a je určená pro dynamické zpracování webových stránek na straně serveru. První verze ASP měla označení ASP 1.0 a byla distribuována jako součást IIS 3.0 v prosinci roku 1996. Do současnosti bylo vydáno 6 hlavních verzí, z nichž ta poslední nese označení ASP.NET 3.0 a byla vydána v únoru roku 2006.

Ve své diplomové práci jsem použil k realizaci webové aplikace ASP.NET 2.0, které vyšlo v listopadu roku 2005. Obecně verze s označením ASP.NET jsou nástupci původního ASP a jsou integrovány s .NET Framework a svými možnostmi se od původního ASP fundamentálně liší. Vývojáři webový aplikací používajících některou z verzí ASP.NET mohou využívat stejné třídy .NET Framework jako při tvorbě jakékoliv jiné aplikace .NET. Mohou tedy snadno využívat stejné nástroje jako využívají vývojáři při tvorbě klientských aplikací. Proto byla tato technologie programátory přijata s nadšením a stala se zdatným konkurentem ostatním technologiím pro tvorbu webových aplikací. [14]

Webová technologie ASP.NET nabízí širokou řadu nástrojů a disponuje některými vlastnostmi, které ji zvýhodňují oproti jiným webovým technologiím. [14]

ASP.NET se neinterpretuje, ale kompiluje

ASP.NET používá interpretované skriptovací jazyky jako C# nebo VB.NET, což programátorům usnadňuje práci. Na druhou stranu zde ale vzniká potřeba kompilace skriptů, která zapříčiňuje výraznou degradaci výkonu celé ASP aplikace. Aplikace ASP.NET se tedy kompilují vždy a není možné spouštět kód napsaný v C# bez předchozí kompilace. Kompilace probíhá ve dvou etapách.

V první etapě se kód napsaný v C# zkompiluje do přechodného kódu, který se nazývá Microsoft Intermediate Language (MSIL) nebo zkráceně IL. Druhá etapa nastává těsně předtím, než se stránka skutečně vykoná. V tomto okamžiku se kód IL zkompiluje do nativního nízkoúrovňového strojového kódu.

ASP.NET je vícejazyčné

Jak již bylo zmíněno výše pro samotnou tvorbu aplikaci má programátor možnost volby z několika programovacích jazyků, přičemž žádný z nabízených jazyků .NET nepodává lepší výsledky než jiný jazyk z .NET, protože sdílejí stejnou infrastrukturu, která je definována ve specifikaci společného jazyka CLS (Common Language Specification).

ASP.NET běží uvnitř společného runtime jazyků

Nejdůležitějším aspektem ASP.NET je skutečnost, že ASP.NET běží uvnitř runtime enginy CLR. Z toho plyne řada výhod:

- Automatická správa paměti (garbage collection)
- Typová bezpečnost
- Rozšiřitelná metadata
- Strukturované zpracování chyb
- Multithreading

ASP.NET je objektově orientované

Samotné ASP nemá k dispozici příliš rozsáhlý objektový model. Disponuje pouze malou sadou objektů, která tvoří vrstvu nad zdrojovými detaily HTTP a HTML. Oproti tomu ASP.NET je plně objektově orientované a kód má úplný přístup ke všem objektům v .NET Framework.

ASP.NET podporuje různá zařízení a různé prohlížeče

Kvůli různorodosti webových prohlížečů podporujících různé verze HTML (HTML 3.2, HTML 4.0, XHTML 1.0 nebo WML – Wireless Markup Language) se realizace webových aplikací optimalizovaných pro co nejširší skupinu prohlížečů značně komplikuje. ASP.NET řeší tento problém díky bohatě vybavené skupině ovládacích prvků, které realizují svůj kód přizpůsobivě, což znamená, že berou v úvahu schopnost klienta.

ASP.NET se snadno rozmísťuje a konfiguruje

Každá instalace .NET Frameworku poskytuje stejné jádro tříd a proto se ASP.NET aplikace rozmísťují relativně snadno. Stačí jen do správného adresáře zkopírovat příslušné soubory na server podporující ASP.NET. Webovou aplikaci lze snadno konfigurovat pomocí souboru web.config, který obsahuje hierarchická seskupení nastavení aplikace uložená ve formátu XML.

3.3 XML – úvod, historie, co nabízí

Zkratka XML představuje rozšířitelný značkovací jazyk (z anglického eXtensible Markup Language), který není závislý na zvoleném operačním systému. Je standardem schváleným konsorciem W3C pro značkování dokumentů. Definuje obecnou syntaxi, která se používá pro označování dat jednoduchými, člověku srozumitelnými značkami. Ustanovuje standardní formát pro počítačové dokumenty. Tento formát je natolik flexibilní, že jej lze dále upravit pro tak různorodé oblasti, jako jsou webové stránky, elektronická výměna dat, vektorová grafika, aplikace katastrů nemovitostí, serializace objektů, vzdálená volání procedur, či systémy hlasové pošty. [15]

Historie XML

Předchůdcem XML a zároveň jedním z prvních značkovacích jazyků byl SGML (Standard Generalized Markup Language) přijatý jako ISO norma v roce 1986. Byl ale natolik složitý a obecný, se spoustou volitelných vlastností, že nenašel příliš velké uplatnění. Dal ovšem základ pro vývoj dalších, lépe uplatnitelných značkovacích jazyků kromě XML například HTML (HyperText Markup Language). [15]

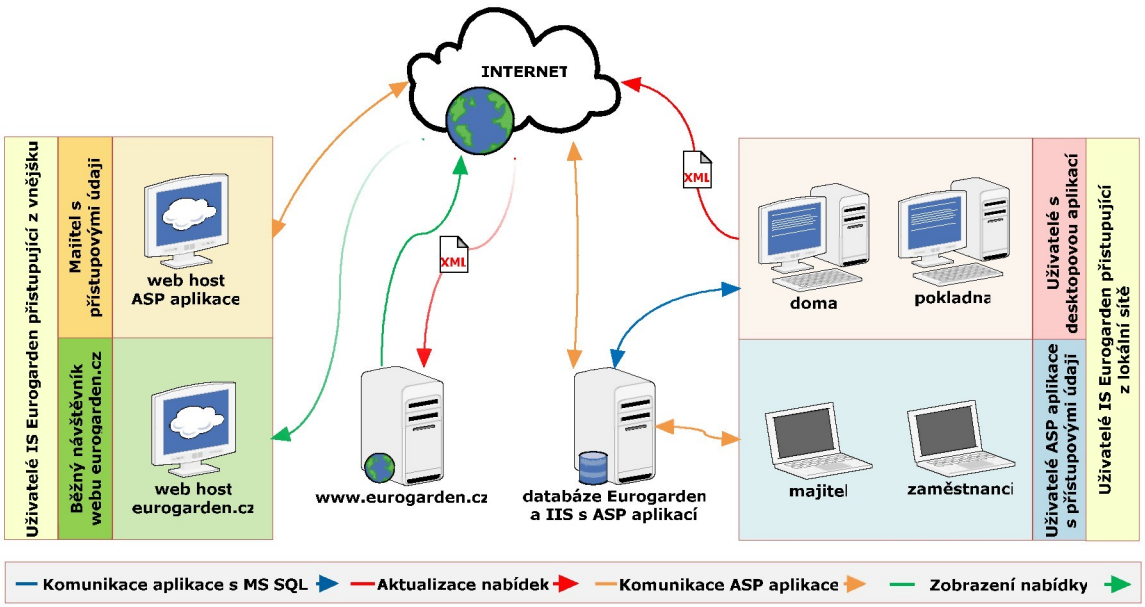
Co XML nabízí

XML je meta-značkový jazyk pro textové dokumenty. Data jsou v dokumentech XML obsažena ve formě textových řetězců, jež jsou uzavřeny do textových značek, které tato data popisují. Konkrétní textový řetězec se spolu s příslušnou značkou označuje jako element. XML nedefinuje pevně danou množinu značek a elementů, u nichž by se předpokládalo, že budou fungovat komukoliv ve všech oblastech nasazení, jako tomu je například u HTML. XML nabízí vývojářům a autorům dokumentů možnost definovat si takové elementy, jaké právě oni potřebují. V ostatních ohledech je však XML velice striktní. Specifikace XML přesně definuje, jakou syntaxi musí značka dodržet, jak se jednotlivé elementy značkami oddělují, jak značka vypadá, jaké jsou pro elementy přípustné názvy, kam se umísťují atributy a tak dále. [15]

4. Návrh informačního systému

Samotný návrh informačního systému vycházel z podrobného porozumění uživatelských požadavků popsaných v předchozí kapitole a specifikací používaného identifikačního systému. Z požadavků zadavatele vyplynulo, že bude nutné vytvořit vícevrstvý softwarový nástroj, který bude schopen nad datovým úložištěm vykonávat požadované funkce.

Zadavatel dále požadoval, aby jako majitel mohl mít přístup do informačního systému odkudkoliv přes celosvětovou síť internet. Aplikační vrstva informačního systému se tedy rozdělila na dvě části, kde první částí je desktopová aplikace a druhou webová aplikace. Obě aplikace musí pracovat na stejném datovém zdroji, kterým je databázový server MS SQL. Následující obrázek zobrazuje nasazení jednotlivých aplikací informačního systému se znázorněním komunikačních toků.



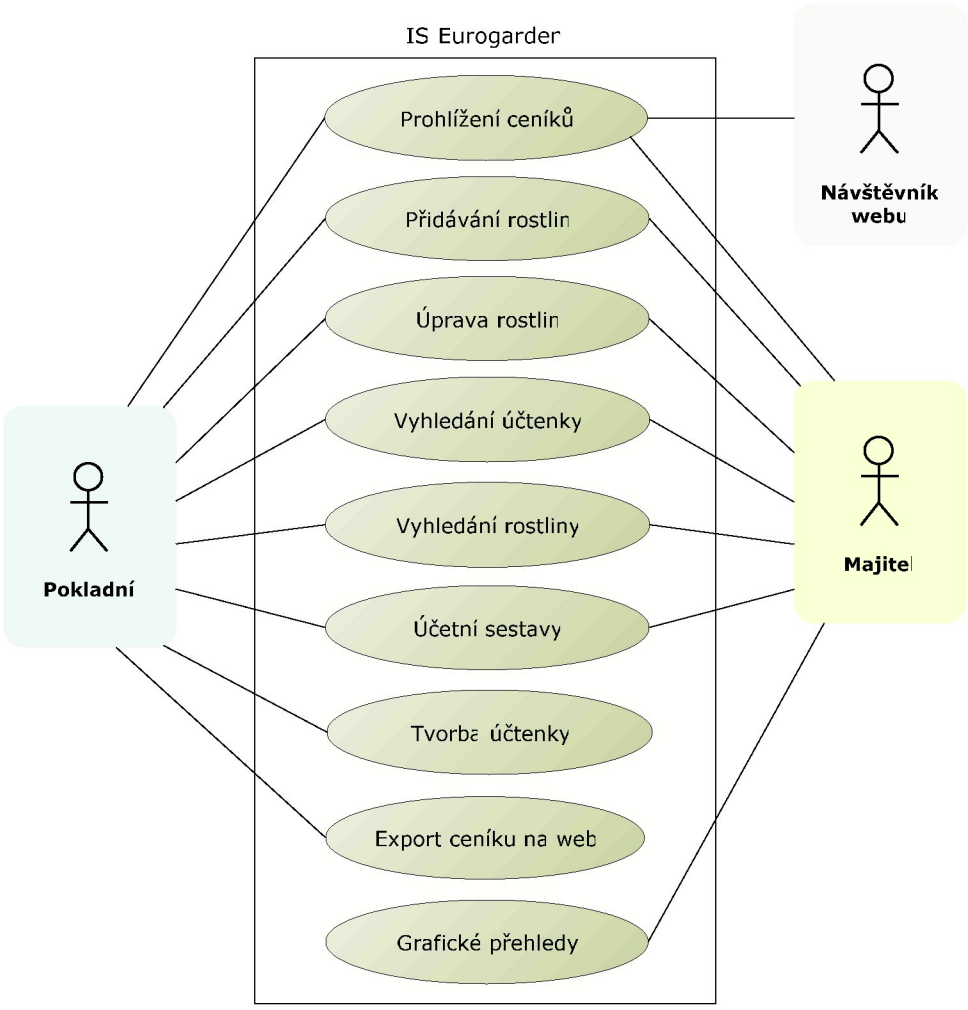
Obrázek 4.1 – Nasazení aplikací IS Eurogarden

Jak je z obrázku patrné s nasazením desktopové aplikace se počítá na stanicích doma a pokladna. Předpokládá se, že aplikace na stanici pokladna bude využívána nejvíce, neboť vlastní periférie pro automatické čtení čárových kódů a tiskárnu účtenek. Webová aplikace je nasazena na IIS (Internet Information Services) a lze k ní přistupovat jak z místní sítě zahradnictví tak z vnějšku prostřednictvím sítě internet. Dále je na obrázku patrný datový tok aktualizace nabídek rostlin, který je realizován prostřednictvím xml souboru.

4.1 Diagram případů použití

Výsledné softwarové řešení bude využíváno zejména třemi skupinami uživatelů, kteří se liší v množině přístupných funkcí. Na obrázku níže je zobrazen diagram případů použití tzv. *Use case diagram*, který vyjadřuje základní vztahy mezi *Actors* vně informačního systému a *Use Case* uvnitř informačního systému.

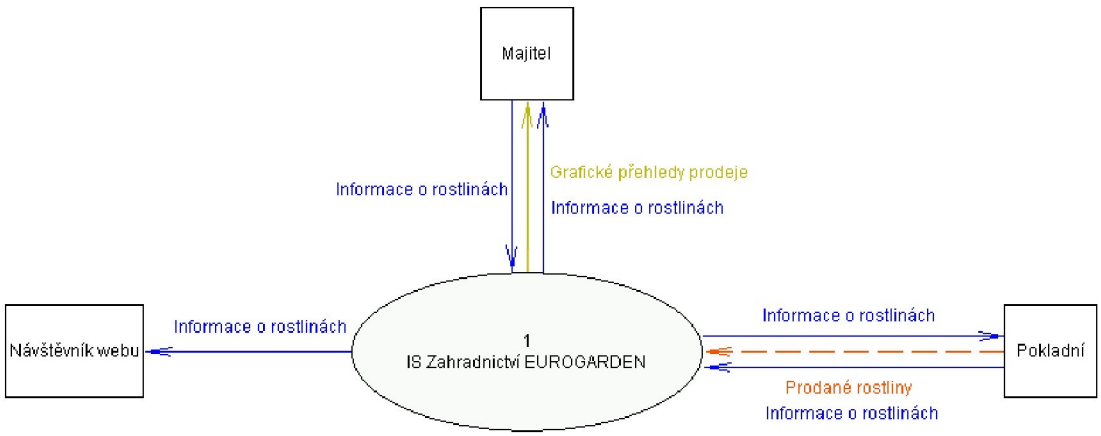
Uživatelům skupiny pokladní musí být zpřístupněny všechny funkce potřebné k uskladnění i správě rostlin a dále také nástroje k tvorbě a správě účtenek. Uživatelé skupiny majitel mají k dispozici takřka všechny nástroje, kterými disponují uživatelé skupiny pokladní a však pro jednoznačné určení zodpovědnosti za tvorbu účtenek a aktualizaci ceníků na webu mají uživatelé skupiny majitel tyto nástroje znepřístupněné. Oproti skupině pokladní mají uživatelé skupiny majitel navíc automaticky vygenerované grafické přehledy prodeje. Uživatelé skupiny návštěvníků webu mohou pouze nahlížet do ceníků.



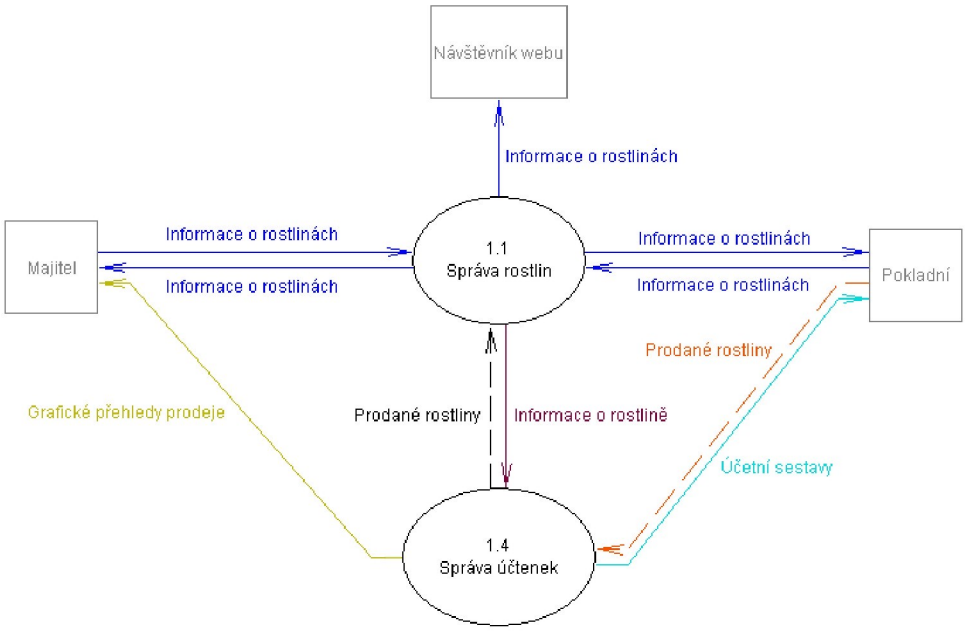
Obrázek 4.29 - Diagram případů použití

4.2 DF diagramy

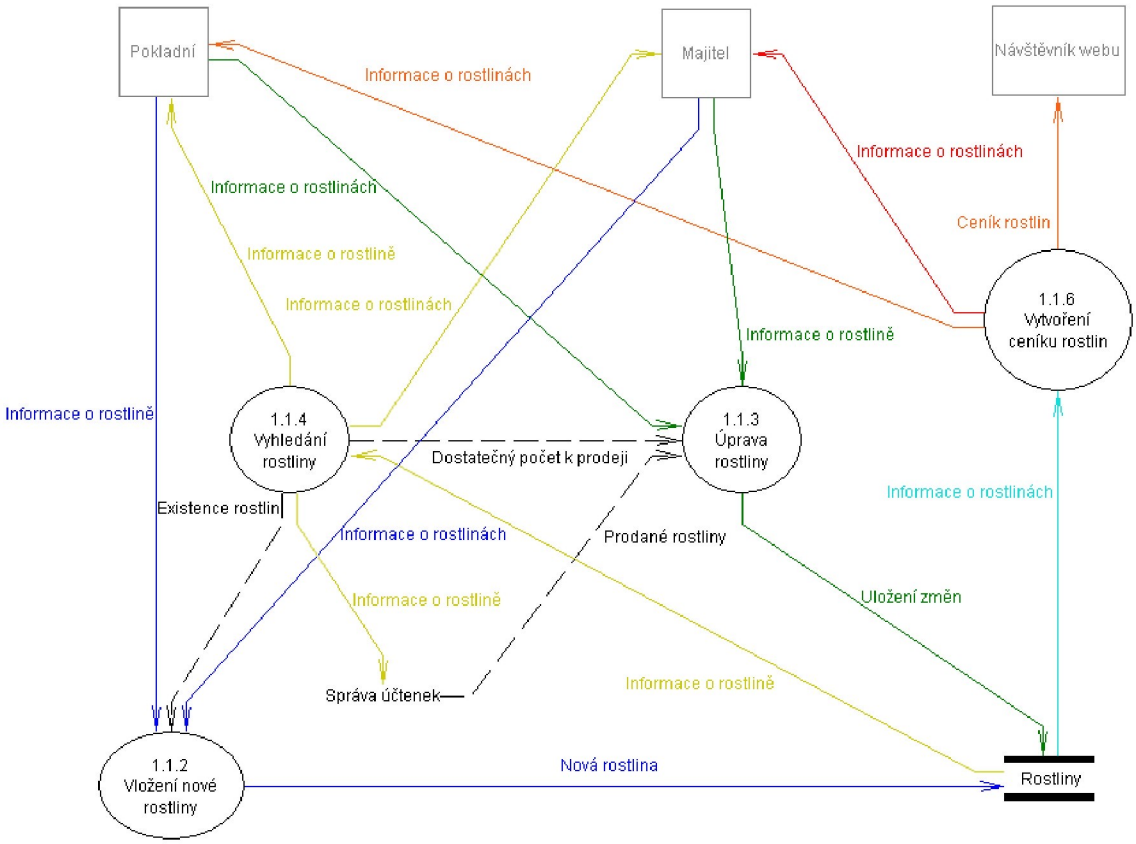
Pro lepší pochopení jednotlivých funkcí informačního systému bylo použito diagramů datových toků (DFD – Data Flow Diagram), pomocí nichž lze snadno graficky znázornit složité struktury, které by jinak byly obtížně pochopitelné. DFD vychází z oblasti teorie grafů a jejich tvorba se řídí určitými pravidly. Níže jsou zobrazeny pouze některé zajímavé data-flow diagramy zachycující datové toky IS zahradnictví. Diagramy jsou rozkresleny pouze do určité úrovně.



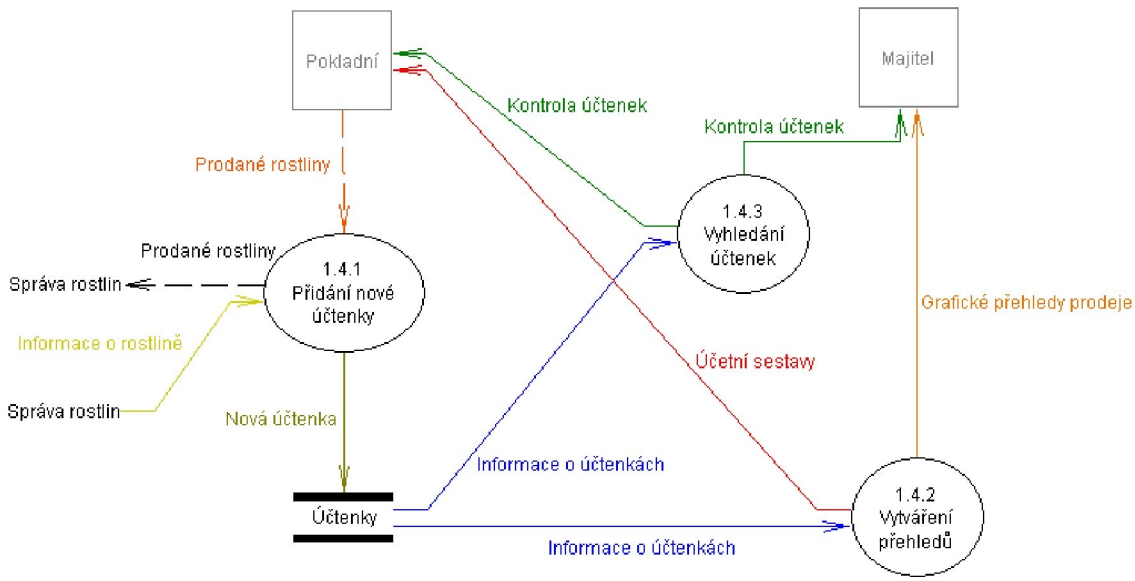
Obrázek 4.310 - Obecný pohled na rozhraní IS



Obrázek 4.411 - Pohled na toky dat v IS zahradnictví



Obrázek 4.5 - Proces správa rostlin



Obrázek 4.6 - Proces správa účtenek

Jak je z diagramů patrné jednotlivé procesy DFD odpovídají nebo přímo rozšiřují funkce uvedené v Use case diagramu. Další zajímavé DFD diagramy, jako např. popis procesu *Přidání nové účtenky*, lze nalézt v příloze.

4.3 Návrh databázové aplikace

Před návrhem algoritmů následoval hierarchický rozklad zadání na jednotlivé dílčí části, v rámci nichž se realizoval konečný návrh strukturovaných algoritmů. Samotný rozklad vycházel z diagramu případů použití a z diagramů datových toků DFD. Jednotlivé části hierarchického rozkladu mají určité vazby, které je nutné respektovat, neboť některé procesy jsou na sobě přímo závislé. Například uložení nebo tisk účtenky je závislé na generování čísla účtenky, protože nelze vytvořit účtenku bez unikátního čísla nebo účtenku s číslem, které by neodpovídalo účetnímu formátu nebo by jinak narušovalo posloupnost čísel účtenek.

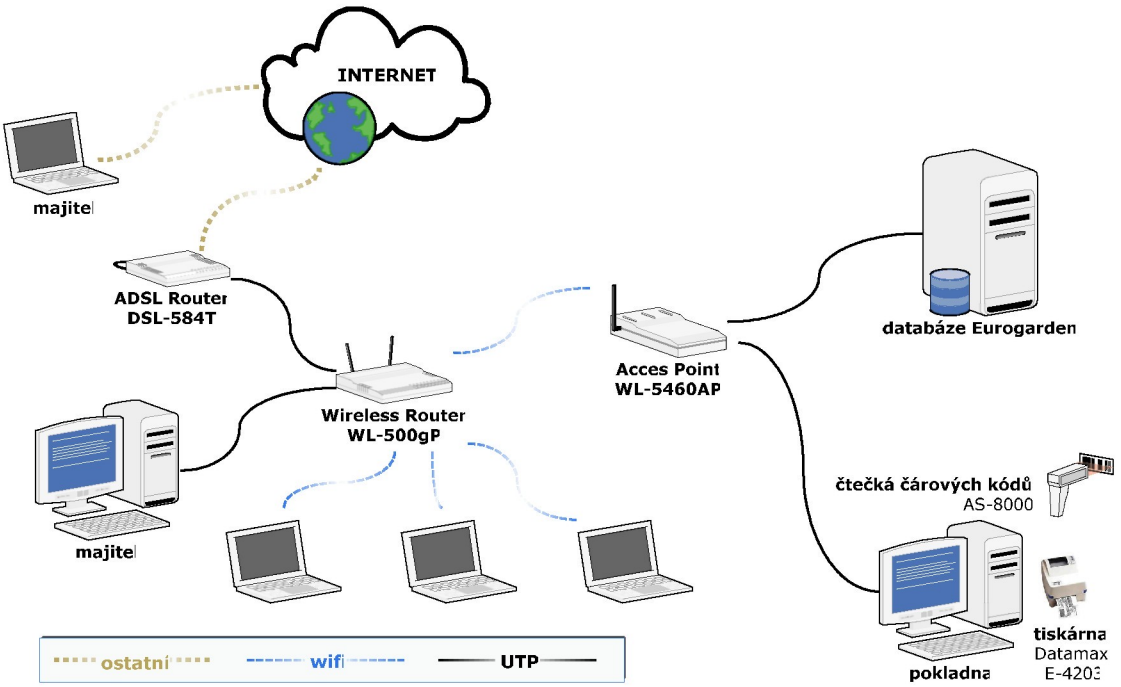
Samotné uživatelské rozhraní aplikace bylo navrženo s ohledem na schopnosti cílových uživatelů. V případě WinForm aplikace na stanici pokladna budou pracovat uživatelé nejčastěji středoškolsky vzděláni v ekonomickém nebo humanitním směru se základními znalostmi z oblasti botaniky. Lze proto předpokládat, že uživatelé budou mít určité povědomí o základech práce s počítačem. Na druhou stranu nelze předpokládat, že by většina uživatelů byla schopná využit rozhraní založené na bázi interpretací příkazů některého z formálních jazyků, např. SQL. Je proto zřejmé, že uživatelské rozhraní nelze realizovat formou konzole, ale musí být grafické a pro uživatele dostatečně intuitivní, s bohatou zpětnou vazbou. Díky předchozí testovací verzi informačního systému a díky programů kancelářského balíku Microsoft Office 2003 mají zaměstnanci zahradnictví již určité zkušenosti s grafickým prostředím, z čehož vyhází i návrh grafického rozhraní WinForm, které je založené na vizuálních komponentách.

Desktopová aplikace byla navržena zejména pro počítačovou stanici pokladna, ke které je připojena tiskárna účtenek DATAMAX E-4203 a čtečka čárových kódů AveScan AS-8000. Většina funkcí a nástrojů je proto uzpůsobena zejména potřebám skladové evidence a maloobchodnímu prodeji. Samozřejmě lze aplikaci nainstalovat i na jakýkoliv jiný počítač, který se nachází v místní síti zahradnictví Eurogarden. Některé funkce aplikace ale budou z části omezené, neboť není žádoucí vytvářet účtenku bez možnosti okamžitého tisku a bez možnosti uložení účtenky ve formátu PDF do odpovídající složky.

4.4 Návrh webové aplikace

Vzhledem k tomu, že webová aplikace obsahuje některé základní funkce desktopové aplikace, její návrh a rozklad zadání se lišil zejména v požadavku na generování grafických přehledů. Návrh grafického uživatelského rozhraní vycházel z rozhraní desktopové aplikace a z možností, které nabízejí třídy technologie .NET Framework.

Aby byla webová aplikace kdykoliv přístupná musí být umístěna na počítači s nepřetržitým a spolehlivým provozem. Ve firemní síti zahradnictví Eurogarden se nachází počítačová stanice, která je neustále v chodu, neboť je k ní připojen kamerový systém zahradnictví. Na této stanici je implementovaný i databázový systém s výše popsanou databází Eurogarden, která je pro aplikace informačního systému zdrojem dat.



Obrázek 4.7 - Struktura sítě

Z obrázku je patrné, že bude-li chtít majitel přistupovat k webové aplikaci implementované na stanici s databází Eurogaren odjinud než z místní sítě, musí mít připojení k internetu, pevnou IP adresu ADSL Routeru DSL-584T a správně nastavené směrování v síti. Před samotnou realizací aplikace bylo nutné nejprve ověřit, zda lze pomocí aktivních prvků počítačové sítě zahradnictví směrovat komunikaci potřebným způsobem. Všechna zařízení směrování podporují a tak pomocí směrovacích pravidel byl nastaven vzdálený přístup k webové aplikaci a administrátorským rozhraním aktivních prvků v síti, aby byla umožněna vzdálená správa.

5. Realizace návrhů

Softwarové řešení informačního systému zahradnictví je realizováno na základě získaných znalostí z výše uvedených kapitol a použité literatury, prostřednictvím programovacího jazyka C# a skriptovacího jazyka ASP 2.0 s využitím vývojového prostředí Microsoft Visual Studio 2008 a databázového systému MS SQL 2000. Realizaci návrhů aplikací předcházela tvorba databáze, jejíž návrh vyházel z popisu reálného světa. Proces tvorby samotných aplikací je založen na návrzích jednotlivých procesů popsaných pomocí data flow diagramů a diagramu případů použití s ohledem na společný datový zdroj, databázového systému.

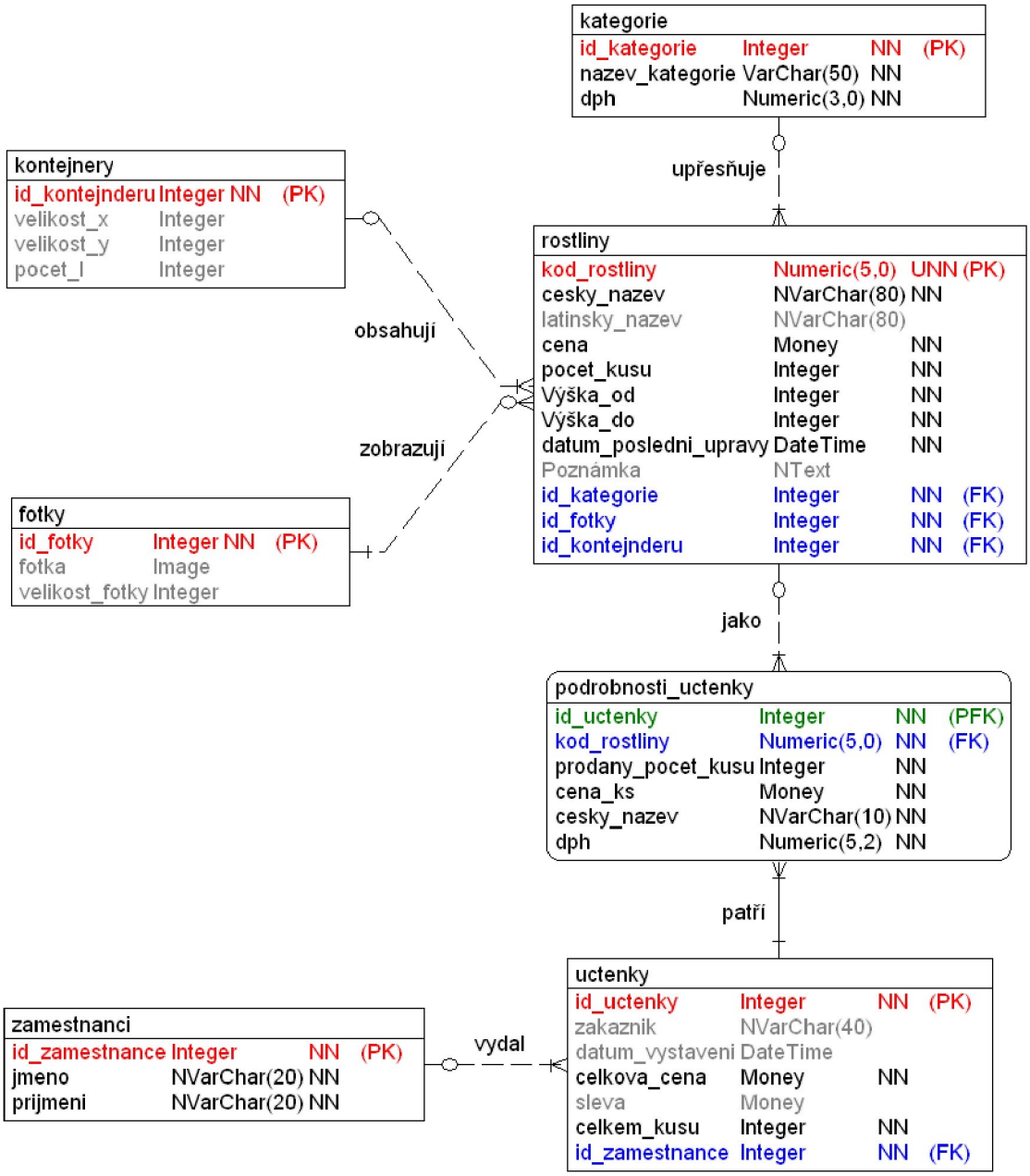
5.1 Řešení databázového systému

Pro realizaci databázového serveru byla vybrána počítačová stanice se standardními hardwarovými parametry a s operačním systémem Windows XP Profesional SP3 od společnosti Microsoft. Stanice je součástí místní sítě zahradnictví a má nastavenou pevnou IP adresu, aby bylo možné k ní přistupovat i odjinud. Na stanici je implementován databázový systém MS SQL Server 2000, který kromě báze dat obsahuje i nástroje potřebné pro správu.

5.1.1 Návrh schématu databáze

Návrhu schématu databáze byl rozdělen do několika částí. V první části byly navrženy požadované entity, které jsou reprezentovány pomocí tabulek. Entitám bylo nutné přiřadit atributy a nadefinovat tak záhlaví tabulek. V poslední část byla realizována integritní omezení. Je možné navrhnout libovolné množství databázových schémat, které splňují shodné podmínky pro ukládání informací o popisovaném úseku reality. Jednotlivá schémata se však budou lišit. Největší odlišnosti by bylo možné nalézt v množství redundantních informací a ve výkonu jejich implementací na databázovém systému. Logickým cílem návrhu je tedy návrh schématu s minimální redundací dat a s maximálním výkonem. Ne vždy to je ale možné, proto se často musí hledat kompromis, ale obecně platí snaha soustředit se více na výkon.

Samotné schéma databáze Eurogarden bylo realizované pomocí nástrojů pro tvorbu entitně relačního diagramu programu pro návrh databázových struktur CASE Studio 2 ver. 2.25.0. Návrh byl realizován s ohledem na zásady návrhu relačních databází a s ohledem na možnosti databázového systému MS SQL 2000.



Obrázek 4.8 - ER diagram - fyzický pohled

V entitně relačním diagramu je zachycena fyzická struktura relační databáze, která vychází z popisu reálného světa zahradnictví. Databázové schéma obsahuje 7 uživatelských tabulek, jejichž atributy mají datové typy nejvíce odpovídající realitě. Atributy u jednotlivých entit dostatečně popisují daný objekt a jejich množství je pro realizaci požadavků optimální. Nemělo by tak docházet ke zbytečné redundaci dat a naopak k jejich nedostatku. Vazby mezi entitami a příslušná kardinalita vycházejí z logické interpretace popisu světa zahradnictví. Je zřejmé, že každá rostlina se může vyskytovat na více položkách účtenky a že každá rostlina nutně spadá do nějaké kategorie, kvůli výpočtu dph na účtence atp.

Realizace integritních omezení spočívá v deklarativním nebo procedurálním vyjádření. V rámci databázového schématu zahradnictví Eurogarden jsou integritní omezení vyjádřena převážně deklarativním způsobem.

Vyjádření integritních omezení deklarativním způsobem

- „NOTNULL“ pro vybrané atributy entit
- Primární klíče
- Kontrola hodnot (CHECK)

  Příkaz CHECK byl využit zejména při kontrole hodnot atributů kód rostliny, kdy každý kód musí mít přesně 5 znaků.

Vyjádření integritních omezení procedurálním způsobem

- uživatelské procedury a funkce
- triggery reagující na události vložení (before insert), modifikace (before update) a odstranění (before delete) nad tabulkami fotky a kontejnery

5.1.2 Realizace databáze

Z navrženého databázového schématu program Case Studio vygeneroval SQL skript, který se následně spustil v databázovém nástroji Query Analyzer databázového systému MS SQL 2000. Po odladění několika nepřesností se vytvořila na databázovém serveru databáze odpovídající specifikám navrženého schématu.

Prvotní databázové schéma bylo navrženo s ohledem na minimalizaci redundace dat, což ale při realizaci aplikace vykazovalo jisté nedostatky související s aktualizačními anomáliemi příkazů INSERT, UPDATE, DELETE. Zejména při vytváření účtenek, kdy je nutné uchovávat informace o prodaných rostlinách. Nebylo optimální využívat výhradně funkčních závislostí. Kdyby položky účtenek obsahovaly pouze referenci na kódy rostlin, tak ve chvíli kdy by došlo ke změně ceny rostliny nebo dph kategorie, do které rostlina spadá, by se databáze ocitla v nekonzistentním stavu, neboť účetní sestavy, které by aplikace, pracující nad databází Eurograden, generovala by neodpovídaly skutečnosti. Obdobné problémy by nastaly při odstranění záznamu o rostlině nebo kategorii dph. Bylo tedy nezbytné databázové schéma optimalizovat nejen s ohledem na minimalizaci redundace dat, ale také s ohledem na aktualizační anomálie. Do tabulky *Podrobnosti_uctenky* tak byly přidány atributy popisující český název rostliny, cenu, za kterou byla rostlina na dané účtence prodaná a sazbu dph, do které daná rostlina aktuálně spadala při prodeji.

5.1.3 Popis tabulek

SQL skript vygenerovaný CASE Studiem obsahoval výhradně příkazy z kategorie DDL, pomocí kterých se na databázovém serveru vytvořila databáze Eurogarden obsahující několik tabulek a skupinu integritních omezení.

Název tabulky	Ukládané informace
Fotky	Fotografie rostlin
Uctenky	Základní informace o účtenkách
Podrobnosti_uctenky	Informace o jednotlivých položkách účtenek
Rostliny	Rostliny v zahradnictví
Zamestnanci	Základní informace o zaměstnancích zahradnictví
Kontejnery	Typy kontejnerů
Kategorie	Kategorie rostlin

Tabulka 5.1 - Popis použitých tabulek

5.1.4 Vlastní SQL procedury a funkce

Pro realizaci komunikace aplikace s databázovým serverem se používá jazyk SQL. Příkazy jazyka SQL lze specifikovat na úrovni zdrojového kódu aplikace, kde se dotazy snadno přiřadí příslušným objektům, které za účasti dalších objektů realizují komunikaci se serverem SQL. Kromě tohoto způsobu lze SQL dotazy specifikovat v rámci SQL procedur a funkcí, které jsou uložené na serveru. V aplikaci se požadovaná uložená procedura nebo funkce zavolá pomocí příkazu EXECUTE zkráceně EXEC a název procedury nebo funkce s případnými parametry. Poslední zmíněná možnost byla při realizaci informačního systému použita nejčastěji, protože případné problémy lze snadno řešit na SQL serveru a není nutné zasahovat do zdrojových kódů aplikací.

Při vytváření uživatelských procedur a skalárních funkcí uložených na serveru bylo kromě standardních příkazů jazyka SQL a konstrukcí T-SQL využito i několik standardních funkcí poskytovaných serverem.

- agregační funkce (COUNT, MAX, SUM)
- řetězcové funkce (RIGHT, LEFT, LOWER)
- funkce pro práci s datem a časem (YEAR,GETDATE)
- systémové funkce (CAST, CONVERT, @@ERROR)

Použité vlastní SQL procedury a funkce

- zmena_poctu_rostlin — přičte kusy zpět do skladu při stornování účtenky
- update_ks — změna počtu kusů při úpravě účtenky
- pnpu — přidá položky účtenky do podrobnosti účtenky
- pnu — vytvoří novou účtenku
- update_polozky — pro úpravu záznamů o rostlinách
- smaz_zaznam — pro smazání rostlin
- pnp — přidá novu rostlinu do evidence
- pnpWeb — přidá základní informace o rostlině do evidence
- vratKody — vrátí informace o položkách účtenky
- vyberRostliny — vrátí základní vlastnosti rostliny dle ID
- dejNejKod — vrátí nejvyšší kód rostliny v dané kategorii
- dejID_Kontejneru — vrátí ID kontejneru dle rozměrů
- dejID_Velikosti — vrátí ID kontejneru dle hodnoty velikosti
- searchWeb — vrátí rostliny dle zadaných kritérií
- sp_kategorie — vrátí informace o kategoriích rostlin
- sp_rostliny — vrátí informace o rostlinách dle čísla kategorie
- sp_soucet_pres_id — vrátí součty prodeje rostlin ve zvoleném období
- sp_soucty_uctenek — vrátí podklady pro souhrny účtenek
- sp_soucty_dph_sazby — vrátí součty sazeb dph prodeje za období
- vratCU — funkce vrátí poslední šestičíslí čísla účtenky

5.1.5 Použité pohledy

Pro získání konkrétních podmnožin dat z jedné nebo více tabulek databáze bylo použito pohledů, které obsahují příkazy jazyka SQL. Využívání pohledů ušetří programátorům zbytečné opakování rozsáhlých výběrů a zvyšuje přehlednost.

- mujPohled — vrátí kompletní přehledy o rostlinách
- prvnich10 — vrátí posledních 10 upravených rostlin
- uctenky_search — záznamy účtenek vyhovující kritériím
- uctenky_view — záznamy pro výpis informací o účtenkách
- data_uctenky — záznamy pro souhrny účtenek
- SouhrnUctenek — záznamy pro účetní sestavy

5.2 Realizace desktopové aplikace

Desktopová aplikace byla vytvořena ve vývojovém prostředí Microsoft Visual Studio 2008 a samotný zdrojový kód byl napsán v programovacím jazyku C#. Při realizaci uživatelského rozhraní aplikace se využívalo zejména vizuálních komponent nabízených technologií .NET Framework 3.5 ze třídy Form. Komunikace s databázovým serverem probíhá prostřednictvím nevizuálních komponent z knihovny ADO.NET. Pro chod některých procesů bylo dále nezbytné využít dalších nevizuálních komponent jakými byly např. timer a backgroundWorker. Samotný zdrojový kód aplikace tak využívá většinou základní komponenty. Výjimkou je knihovna pdfSharp, která slouží k tisku a ukládání účtenek ve formátu pdf.

Vývojové prostředí nabízí díky nativní podpoře Microsoft Windows Installer (MSI) nástroje pro tvorbu windows instalátorů. Pro snadnou implementaci byl instalátor vytvořen i u deskotpové aplikace.

5.2.1 Realizace požadovaných funkcionalit

Konečný projekt se skládá ze 17 WinFormů, kde hlavním WinFormem je Form1. Na tomto WinFormu jsou jednotlivé nástroje a funkce seskupeny do 5 kategorií, mezi kterými lze klepnutím snadno přepínat pomocí vizuální komponenty *tabControl*. Pro snazší orientaci v záložkách *tabControlu* bylo použito grafických symbolů s následujícími popisy.

- POKLADNA — nástroje a funkce potřebné pro tvorbu účtenek
- PŘIDAT — jednoduchý formulář pro přidání nové rostliny
- SKLAD — kompletní zobrazení obsahu evidence rostlin
- SESTAVY — nástroje a funkce pro generování účetních sestav
- ÚČTENKY — celkový přehled vystavených účtenek

Funkce a nástroje, které jsou zahrnuty ve výše popsaných skupinách jsou uživateli nabízeny na jednotlivých záložkách *tabPage* vizuální komponenty *tabControl*. K rozmístění ostatních vizuálních komponent bylo využito s*plitContainerů, GroupBoxů* a *panelů*. Nastavením specifických vlastností (zejména vlastnosti Dock) u těchto seskupovacích komponent bylo možné dosáhnout i stabilního zobrazování aplikace pro různá rozlišení monitorů. Aplikaci tak lze bez problémů se zobrazením používat při rozlišeních vyšších než 1280x800.

Pro zvýšení přehlednosti a ovladatelnosti aplikace byly univerzální nástroje a funkce nevhodné pro začlenění do záložek *tabControlu* přidány do menu v horní části okna. Menu je reprezentované vizuální komponentou *menuStrip* s položkami odpovídajícími dalším nástrojům nebo jejich skupinám.

Při spuštění aplikace bylo nezbytné ošetřit v hlavní metodě *Main()* nastavení základních parametrů aplikace, kterými jsou cesta k programu pro čtení pdf souborů, cesta k adresáře pro ukládání účtenek ve formátu pdf a připojovací řetězec k databázi. Pakliže jsou všechny tyto parametry otestované s kladným výsledkem, spustí se hlavní Form1. Pokud ale nastane v testování nastavení chyba zobrazí se na místo Form1 Form14, který obsahuje vstupní pole a dialogy pro vyplněné hodnot nastavení. Zadané hodnoty se ukládají do XML souboru, ze kterého se při každém dalším spuštění načítají hodnoty nastavení.

5.2.2 Tvorba a uložení účtenek

Nástroje určené pro vytváření účtenek jsou umístěné na první záložce *tabControlu* s názvem pokladna. Do procesu vytvoření účtenky je nutné uživatelem vložit základní informace o položkách účtenky, kterými jsou nejčastěji právě rostliny jednoznačně identifikované čárovými kódy. Uživatel může zadat čárový kód rostliny dvěma způsoby. První způsob je zadání kódu pomocí čtečky čárových kódů. Aplikace rozpozná čtení kódu pomocí čtečky díky pomocnému symbolu # před každým čárovým kódem na nosičích. Na základě rozpoznání správného formátu čárového kódu se rostlina vyhledá. Druhý způsob je zadávání čárových kódů ručně do *maskedTextBoxu*, kde je správný formát vstupního čísla určen maskou. Po kliknutí na tlačítko přidat se opět rostlina dle čárového vyhledá. Pro oba způsoby se po vyhledání rostliny dle čárového kódu nejprve ověří jejich počet na skladě a ve chvíli, kdy jich bude k dispozici nulový počet zobrazí se *MessageBox* s oznámením, že nelze přidat rostlinu na účtenku, neboť není na skladě. Pakliže je jejich počet nenulový automaticky se rostlina přidá jako položka účtenky do tabulky, která je reprezentovaná vizuální komponentou *dataGridView.* Při zadání jednoho kódu vícekrát dojde ke konsolidaci položek a k navýšení jejich počtů na účtence a k adekvátní změně ceny. Přidat položku účtenky lze i pomocí nástroje pro vyhledávání, který se aktivuje klepnutím v menu na rostliny. Po zobrazení *Formu8* uživatel vyplní příslušná formulářová pole a po klepnutí na tlačítko vyhledat se mu na základě SQL dotazu obsahujícího konstrukt LIKE zobrazí

výsledky. Po klepnutí pravým tlačítkem myši na vybranou rostlinu a vybráním příkazu přidat na účet ze zobrazeného *contextMenuStrip*, se vybraná rostlina přes proces ověření počtu kusů přidá do tabulky *dataGridView.*

Pro výše zmíněné postupy přidání rostliny na účtenku se využívá uložené SQL procedury *vyberRostliny* s parametrem rozpoznaného kódu rostliny. Na základě vráceného záznamu je možné ověřit dostatečný počet kusů. Aby uživatel získal maximální přehled o rostlinách, které právě prodává, zobrazí se po přidání položky do *dataGridView* v levé části okna v *groupBoxu* popis charakteristických vlastností rostliny včetně její fotografie, je-li k dispozici. *ContextMenuStrip* nabízí kromě úpravy počtu kusů také možnost odstranit položku z účtenky nebo zobrazit detail položky.

Dále aplikace nabízí u položek přidaných v *dataGridView* nástroje pro změnu počtů kusů, odstranění položky z účtenky a zobrazení podrobných informací o položce. Pro použití některé z těchto funkcí musí uživatel klepnout pravým tlačítkem na upravovanou položku a z místní nabídky *contextMenuStrip* vybrat požadovanou funkci. Pro změnu počtu kusů se zobrazí nové formulářové okno *Form7,* kterému se předá z hlavního *Formu1* informace o maximálním počtu kusů, jenž je možné zvolit. Tato hodnota se nastaví jako maximum vstupního číselného pole *numericUpDown*. Po změně a potvrzení tlačítkem změnit dojde k předání nové hodnoty počtu kusů do *dataGridView* a dojde k aktualizaci všech doposud vypočtených hodnot vytvářené účtenky. Aby bylo možné předávat hodnoty v rámci více formů nebo přímo přistupovat k vlastnostem a metodám, musel se u ostatních formů nastavit jejich rodič *Form1*, což se provedlo před samotným zobrazením okna.

Po naplnění *dataGridView* položkami může uživatel dále využít *numericUpDown* pro nastavení případné slevy. Při nastavování procentuální hodnoty slevy se automatiky přepočítávají pole celkové ceny a částky k vrácení. Chce-li uživatel účtenku uzavřít a vytisknout, klepne na tlačítko uzavřít a stisk, kterým se aktivuje proces vygenerování účtenky. Tento proces nejprve pomocí SQL funkce *vratCU* získá číslo účtenky pak pomocí uložených procedur *pnu* přidá do tabulky *uctenky* záznam o nové účtence a pomocí procedury *pnpu* přidá do tabulky *podrobnosti_uctenky* záznamy o položkách účtenky. Jakmile dojde k vložení nových položek aktivuje se *trigger*, který zmenší počty kusů u příslušných rostlin v tabulce *rostliny.* Po úspěšném vložení záznamů do databáze se postupně aktivuje procedura *savePDF(),* která pomocí knihovny *PdfSharp vytvoří* a uloží vizuální podobu účtenku včetně záhlaví, položek a zápatí s rekapitulací dph. Procedura *printPDF()* pak účtenku ve formátu pdf vytiskne.

5.2.3 Export účetních sestav

Na záložce sestavy lze nalézt nástroje pro generování účetních sestav a přehledů účtenek. Pomocí komponent *dateTimePicker* uživatel vymezí období, za které se mají souhrny vytvořit. Tento časový interval se předá jako parametr uloženým SQL procedurám *sp_soucet_pres_id, sp_soucty_uctenek* a *sp_soucty_dph_sazby*. Procedury vrátí kolekce dat, nad kterými se provedou dodatečné úpravy a zobrazí se v příslušných *dataGridView*, jenž jsou reprezentovány přehlednými tabulkami. Tabulky pro zobrazení přehledů jsou umístěny na samostatných záložkách *tabPage* komponenty *tabControl*. Pro zvýšení přehlednosti jsou řádkům tabulky nastaveny příslušné vlastnosti textu, pozadí a formátu zobrazovaných dat.

Hlavním cílem tvorby účetních přehledů je úspora finančních prostředků vynakládaných za účetní služby za tvorbu účetních sestav pro případnou kontrolu finančního úřadu. Automaticky vygenerované sestavy, kterou jsou obsažené v tabulkách komponent *dataGridView*, se po klepnutí na tlačítko vyexportují do sešitů aplikace Microsoft Excel. Pro automatický export bylo nutné do projektu přidat referenci na knihovnu objektů Microsoft Excel *Microsoft.Office.Interop.Excel*. Pomocí této knihovny se při exportu sestav nejprve spustí aplikace Microsoft Excel a získá se jako objekt, ve kterém se načte nový sešit. K jednotlivým sešitům se přistupuje jako k objektům a pomocí vlastnosti *Cells* lze snadno adresovat buňky a vkládat do nich data. Prostřednictvím dalších vlastností a metod se pak obsahy buněk a buňky samotné naformátují, aby výsledná tabulka byla přehledná.

Pro úplnost se ještě přidá do záhlaví časový údaj vygenerování sešitu a interval za jaké období je souhrn vytvořen. Údaje se nastaví vlastností *PageSetup.LeftHeader* a *PageSetup.RightHeader*. O stavu vytváření informuje uživatele *progressBar*, který využívá třídu *backgroundWorker* pro realizaci asynchronních procesů. Jakmile je soubor vygenerován, zobrazí se aplikace Microsoft Excel, která doposud běžela skrytě na pozadí a uživatel má možnost výsledná data dále upravovat nebo rovnou přistoupit k jejich tisku.

Export účetních sestav byl otestován na verzích Microsoft Office 2003 a 2007. Primárně je předpokládáno využití na verzi Microsoft Office 2007, proto ani nižší verze sady Microsoft Office testovány nebyly.

5.2.4 Export ceníků na webový server

Aby uživatel nemusel stejnou rostlinu vkládat zvlášť do MS SQL databáze přes aplikační rozhraní desktopové aplikace a zvlášť přes webové rozhraní do MySQL databáze, ze které se sestavují ceníky pro návštěvníky webových stránek zahradnictví, bylo požadováno vytvořit nástroj pro synchronizaci databází. Při návrhu bylo však zjištěno, že synchronizovat obě databáze respektive jejich rozdílné záznamy by bylo možné, ale nepříliš efektivní, neboť je zbytečné a nepříliš bezpečné na webový server umísťovat celou databázi Eurogarden včetně informací o prodeji. Proto byl pro řešení problému navržen nástroj, který vyexportuje pouze požadované informace o rostlinách.

Pomocí komponent ADO.NET a SQL příkazu, ve kterém příkaz SELECT vybere požadované sloupce ze spojení INNER JOIN tabulek Rostliny a Kategorie. Výsledné záznamy naplní *dataSet*, který je datovým zdrojem pro grafickou reprezentaci tabulky *dataGridView*. Při úpravě nebo vložení nové rostliny uchovává desktopová aplikace jejich případné ilustrace ve speciální složce, neboť právě přenos obrázků je na celém exportu nejnáročnější. Obrázky připravené k exportu se uživateli zobrazují v přehledném seznamu *listBoxu*.

Po klepnutí na tlačítko aktualizovat se z *dataSetu* pomocí funkce *GetXml()* vygeneruje XML soubor, který bude společně s obrázky přenesen na webový server. Způsob přenosu byl zvolen přes ftp. Pro přenos je využíván objekt *FtpWebRequest*, který je schopen navázat spojení se severem a přenést požadované soubory. Pro navázání spojení s webovým serverem bylo nutné zadat název serveru, přístupové jméno a heslo. Jakmile se aplikace připojí zapíše *fileStreamy* exportovaných souborů na server pomocí metody *Write*. O průběhu exportování je uživatel informován prostřednictvím *toolStripProgressBaru*, který je spjat se třídou pro zpracování asynchronních procesů *backgroundWorker*.

Vyexportovaný XML soubor slouží jako datový zdroj pro vygenerování webové stránky s ceníky. Pomocí XSLT transformace se převedou zdrojová data ze souboru XML do souboru HTML, ve kterém je specifikována i grafická podoba zobrazení dat. Samotná XSLT transformace probíhá v souboru *ceniky.php*, kde se nejprve načtou data z XML souboru, následně se vytvoří objekt *domDocument*, do kterého se vloží soubor XSL popisující specifika převodu. Po té se vytvoří objekt *XSLTProcesor*, kterému se předají vstupní data z XML a specifikace převodu XSL. Výsledný soubor HTML se funkcí *include* vloží mezi odpovídající html tagy.

5.3 Realizace webové aplikace

Webová aplikace stejně jako desktopová aplikace byla realizována ve vývojovém prostředí Microsoft Visual Studio 2008 a pracuje nad stejný datovým zdrojem databázovým serverem MS SQL. U skriptovací platformy ASP.NET byl použit programovací jazyk C#. K realizaci aplikace bylo použito standardních tříd ASP.NET 2.0, uložených SQL procedur a pohledů a dále několika rozšiřujících tříd ovládacího prvku Microsoft Char Controls, které bylo nutné stáhnout z webu společnosti Microsoft.

Webová aplikace obsahuje všechny základní funkce pro správu rostlin a evidence účtenek, které vycházejí z nástrojů desktopové aplikace. Projekt webové aplikace obsahuje následující *webFormy*.

- default.aspx — výchozí skript s web komponentou *login*
- uvod.aspx — obsahuje grafické přehledy, vyhledávání rostlin a účtenek
- sprava.aspx — obsahuje nástroje pro správu rostlin
- uctenky.aspx — obsahuje nástroje pro vyhledávání účtenek
- search.aspx — vrací kolekci vyhledávaných rostlin
- add_new.aspx — formulář pro založení nové rostliny
- detailView.aspx — zobrazení detailních informací o rostlině

5.3.1 Řešení oprávněného přístupu

Na výchozím web formu default.aspx je umístěn ovládací prvek login, který poskytuje hotové uživatelské rozhraní pro zadání jména a hesla. Jakmile uživatel vyplní údaje a klepne na tlačítko přihlásit odstartuje se metoda *Authenticate* ze třídy *FromsAuthentication.* Metoda provádí automatickou autentizaci na základě dvou parametrů uživatelského jména a hesla a porovnává je s údaji v souboru *Web.config*. Pakliže není autentizace úspěšná zobrazí se stránka s oznámením zamítnutého přístupu nebo je uživatel přesměrován zpět na původní stránku. Je-li autentizace úspěšná spustí se metoda *RedicertFromLoginPage* ze třídy *FromsAuthentication.* Metoda očekává jméno uživatele a booleovskou hodnotu. Předá-li se hodnota *true* bude vytvořeno cookie, které bude platné stále i při restartu prohlížeče. Hodnota *false* byla předána v aplikace a vytvořené cookie bude platné po dobu stanovenou v souboru *Web.config* nebo do okamžiku restartu prohlížeče. Je-li uživatel přihlášen, zobrazí se informace o stavu přihlášení v šabloně *LoginView*, kde se může uživatel klepnutím na tlačítko kdykoliv odhlásit metodou *SignOut.*

5.3.2 Grafické výstupy prodeje

Grafické přehledy prodeje jsou v aplikaci realizovány pomocí tříd ovládacího prvku Microsoft Char Controls. Na *webFormu* uvod.aspx jsou umístěny dvě vizuální komponenty *Chart*, které mají nastavenou vlastnost *charType* na *Column*, takže výsledné grafy jsou sloupcové. Nastavením dalších vlastností bylo docíleno prostorového efektu, barevných gradientů a průhlednosti sloupců. Vlastnosti komponenty *Chart* umožňují specifikovat nejen výsledný vzhled sloupců, ale také vlastnosti os, pozadí, natočení, popisek a celou řadu dalších vlastností.

První grafická komponenta reprezentuje tržby za jednotlivé dny v aktuálním měsíci. Datovým zdrojem je SQL dotaz vracející kolekci dvou sloupců. Hodnoty sloupce číslo dne jsou u grafu použity jako hodnoty osy x. Hodnoty druhého sloupce součty za den jsou použity jako hodnoty osy y respektive jako hodnoty jednotlivých sloupců. Druhá grafická komponenta reprezentuje tržby za posledních 12 měsíců. Oproti předchozímu se tedy liší pouze v SQL dotazu, který vrací jinou kolekci dat (měsíc a součet za měsíc), a v barevných gradientech.

Závěr

Na začátku práce bylo nezbytné se seznámit s aktuálním stavem identifikačního systému, který jsem společně s testovací verzí databázové aplikace vytvořil v loňském roce v rámci magisterského projektu. Na základě zkušeností s testovací verzí aplikace zadavatel specifikoval nové požadavky, které bylo nutné řádně prostudovat a vyhodnotit jejich přínos a realizovatelnost. Během řešení diplomové práce jsem se dále seznámil s možnostmi a omezeními při tvorbě databázových aplikací. Dále jsem nastudoval možnosti práce s databázovým systémem MS SQL 2000 a s dotazovacím jazykem SQL a T-SQL. Pro samotnou realizaci hlavních cílů práce bylo nezbytné se seznámit s možnostmi tvorby desktopové a webové aplikace prostřednictvím jazyka C# a technologie .NET Framework ve vývojovém prostředí Microsoft Visual Studia 2008.

Na základě získaných znalostí a zkušeností jsem navrhl databázové schéma relační databáze eurogarden, vytvořil jsem algoritmy aplikačních funkcí klientské i databázové komponenty a navrhl jsem grafická rozhraní desktopové i webové aplikace. Po schválení návrhů jsem implementoval databázové schéma na databázový server MS SQL 2000 a s využitím jazyka SQL a T-SQL jsem na serveru realizoval databázovou část správy rostlin a účtenek. S využitím vývojového prostředí Microsoft Visual Studio 20008 a komponent techlogie .NET Framework jsem vytvořil grafické uživatelské rozhraní pro desktopovou a následně i pro webovou aplikaci. Pomocí jazyka C# jsem po té implementoval navržené algoritmy. Z konečné verze desktopové aplikace byl pomocí instalačních nástrojů vývojové prostředí vytvořen instalační balík, pomocí něhož byla aplikace implementována na stanici pokladna. Webová aplikace byla implementována na lokální server se službou IIS.

V dosavadním průběhu testování se aplikace jeví jako stabilní a rychlé. Zadavatel je tak zatím s řešením spokojen, neboť vytvořené aplikace splňují všechny zadané požadavky. Aplikace by se ve spojení s identifikačním systémem měly stát stěžejním softwarovým nástrojem při správě rostlin a účtenek. I když současný stav řešení je z pohledu zadavatele konečný, není vyloučené, že by v budoucnu mohly být aplikace dále rozšířeny. Možnosti dalšího rozvoje mnou vytvořených aplikací vidím zejména v rozšíření desktopové aplikace o nástroje pro kalkulace cen při realizaci zahrad s tvorbu faktur nebo v rozšíření webové aplikace o tvorbu účetních souhrnů.

Použitá literatura a zdroje

[1] SIXTA, J. - MAČÁT, V. *Logistika - teorie a praxe*, Brno: CP Books, 2005. Kapitola 11, s. 204-210. ISBN 80-251-0573-3

[2] IKOS Liberec, s.r.o. *Datamax E-4204* [online]. [cit. 10. prosince 2008]. URL: <http://www.datamax.cz/pages/cz/cary/tiskarna-012.html>

[3] CODEWARE, s.r.o. *AS-8000 AveScan* [online]. s. 9. [cit. 10. prosince 2008]. URL: <http://www.codeware.cz/resources/manuals/argox/as-8000_user_cz.pdf>

[4] PETRLÍK, L. *Přednášky k předmětu Základy softwarového inženýrství* [online]. Katedra informatiky a výpočetní techniky na ZČU, s. 2-22. [cit. 9. února 2009]. URL: <http://www.kiv.zcu.cz/~luki/vyuka/zswi/predn/zswi2005.pdf.>

[5] POKORNÝ, J., HALAŠKA, I. *Databázové systémy*, Praha: Vydavatelství ČVUT, 2003. ISBN 80-01-02789-9

[6] ŠTULLER, J., *Relational Data Model* [online]. Přednáška k předmětu Databázové řídící systémy na TU v Liberci, s. 29. [cit. 10. února 2009]. URL: <http://www.cs.cas.cz/stuller/php/uploaded/Databazove systemy 2.pdf>

[7] ZENDULKA, J., *Databázové systémy – 4. Relační model dat*, Přednáška k předmětu DSI na VUT v Brně, s. 2-3. [cit. 2. ledna 2009]. URL: < http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/4_relmod.pdf>

[8] CAPEKOVÁ, Z., *Databázový systém určený k podpore ekonomických úvah v oblasti pol'nohospodárskej techniky.* Liberec, 2005. 60 s. Diplomová práce. Technická univerzita v Liberci. Vedoucí diplomové práce Pavel Rydlo, 29-35 s.

[9] PRICE, J., *C# - programování databází*, Praha: Grada, 2005. ISBN 80-247-0982-1

[10] STANEK, W. R.: *Microsoft SQL Server 2000*, Brno: Computer Press, 2003. ISBN 80-7226-880-5

[11] PROSISE, J. *Programování v Microsoft .NET*, Brno: Computer Press, 2003. ISBN 80-7226-879-1

[12] PETZOLD, CH. *Programování Microsoft Windows v jazyce C#. Svazek,* Praha: SoftPress, 2006. ISBN 80-251-1058-3

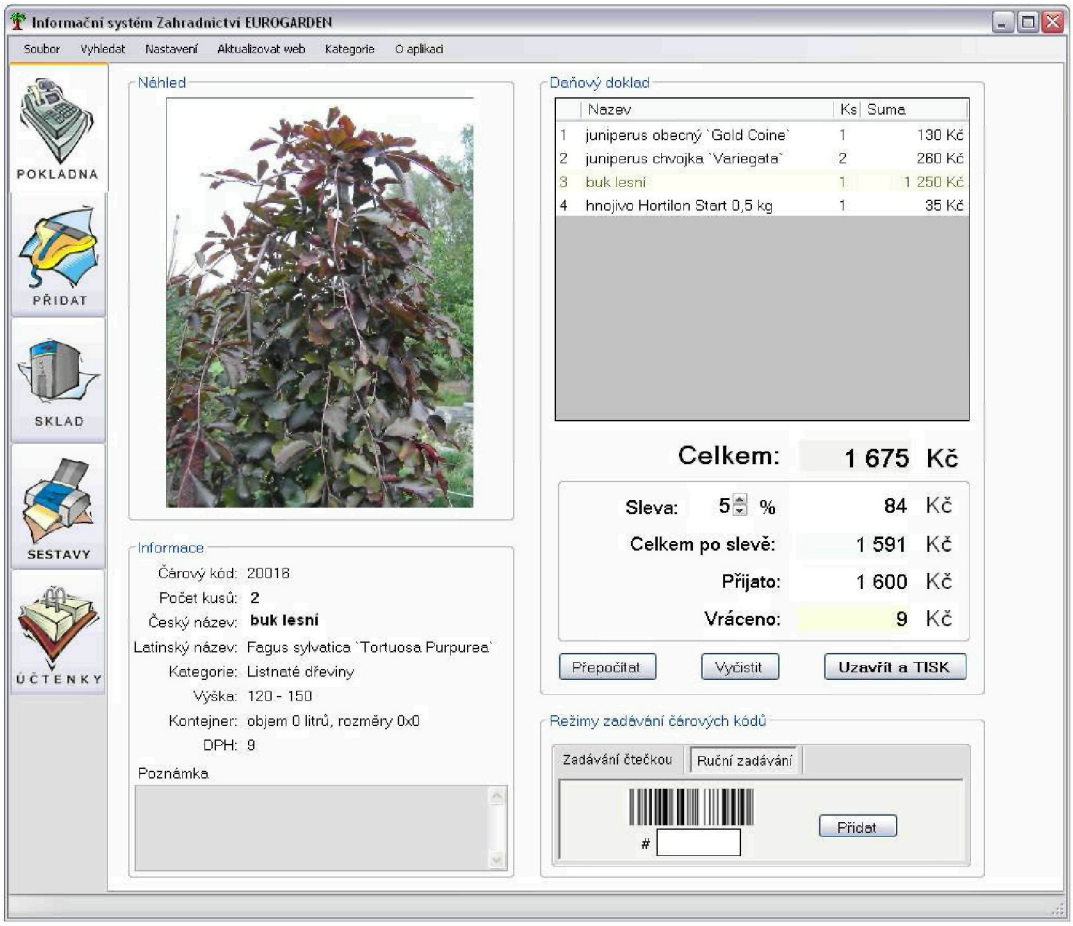
[13] DUNDAS, *Dundas Chart for .NET,* [online]. [cit. 20. února 2009]. URL: <http://www.dundas.com/Products/Chart/Net/index.aspx>

[14] MACDONALD, M. – SZPUSZTA, M. *ASP.NET 2.0 a C#*, Brno: Zoner Press, 2006. ISBN 80-86815-38-2

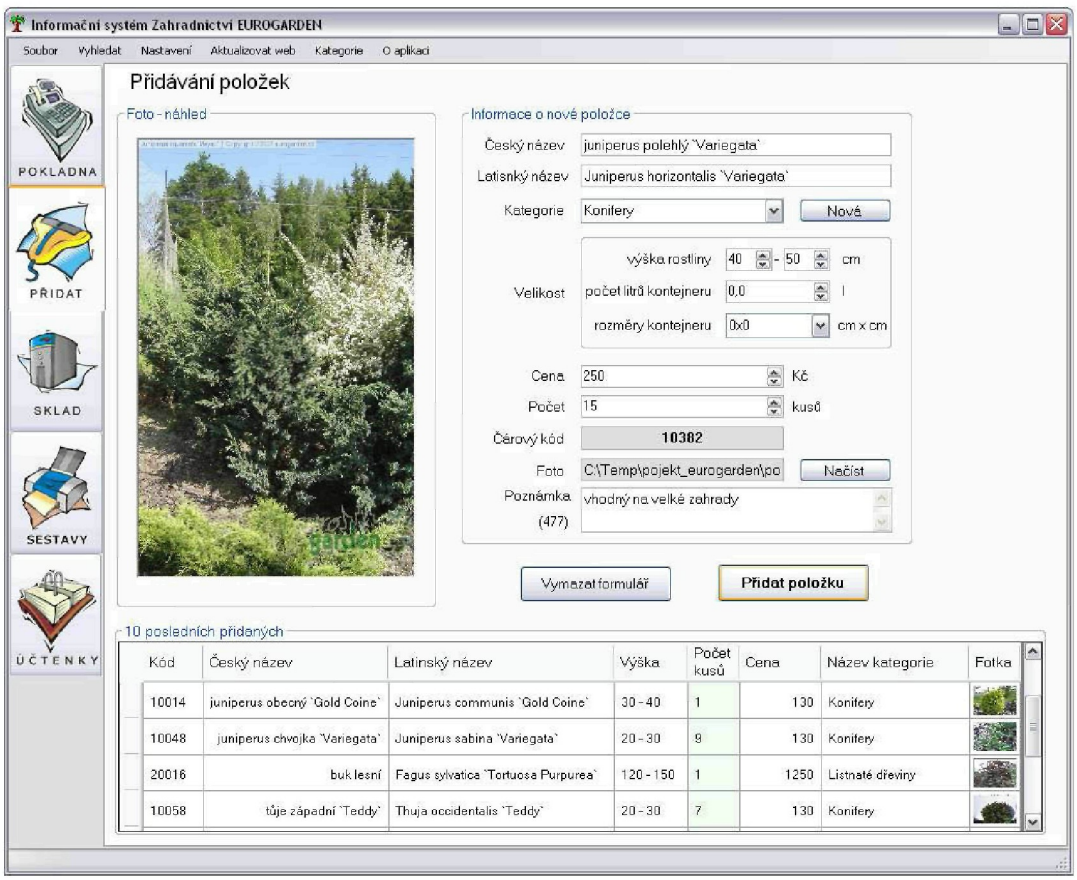
[15] ESPOSITO, D. *XML - efektivní programování pro .NET*, Praha: Grada Publishing a.s., 2004. ISBN 80-247-0775-6

Přílohy

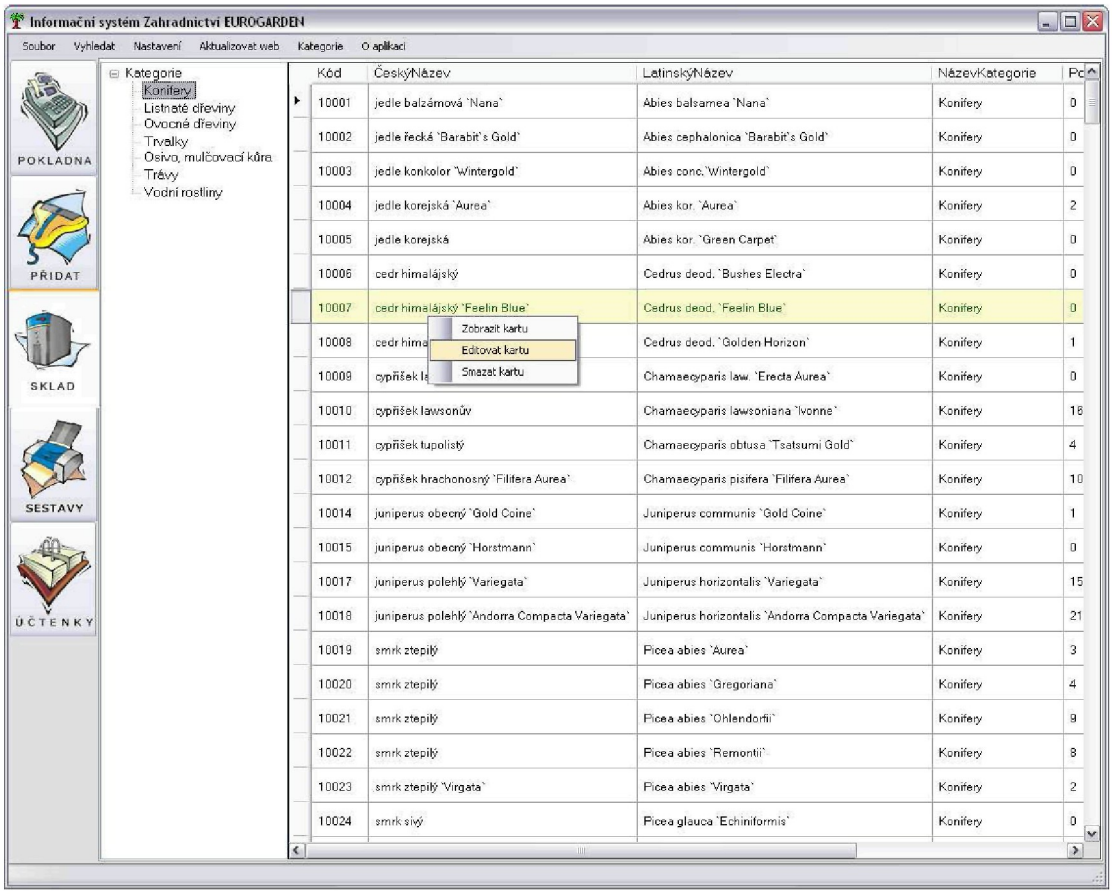
1. Print-screeny desktopové aplikace
2. Print-screeny webové aplikace
3. Některé zajímavé SQL funkce a procedury
4. Zajímavé funkce a procedury v C#
5. Grafický výstup procesu tvorby účtenek
6. Grafický výstup souhrnu účtenek
7. DFD diagram procesu přidání nové účtenky
8. CD



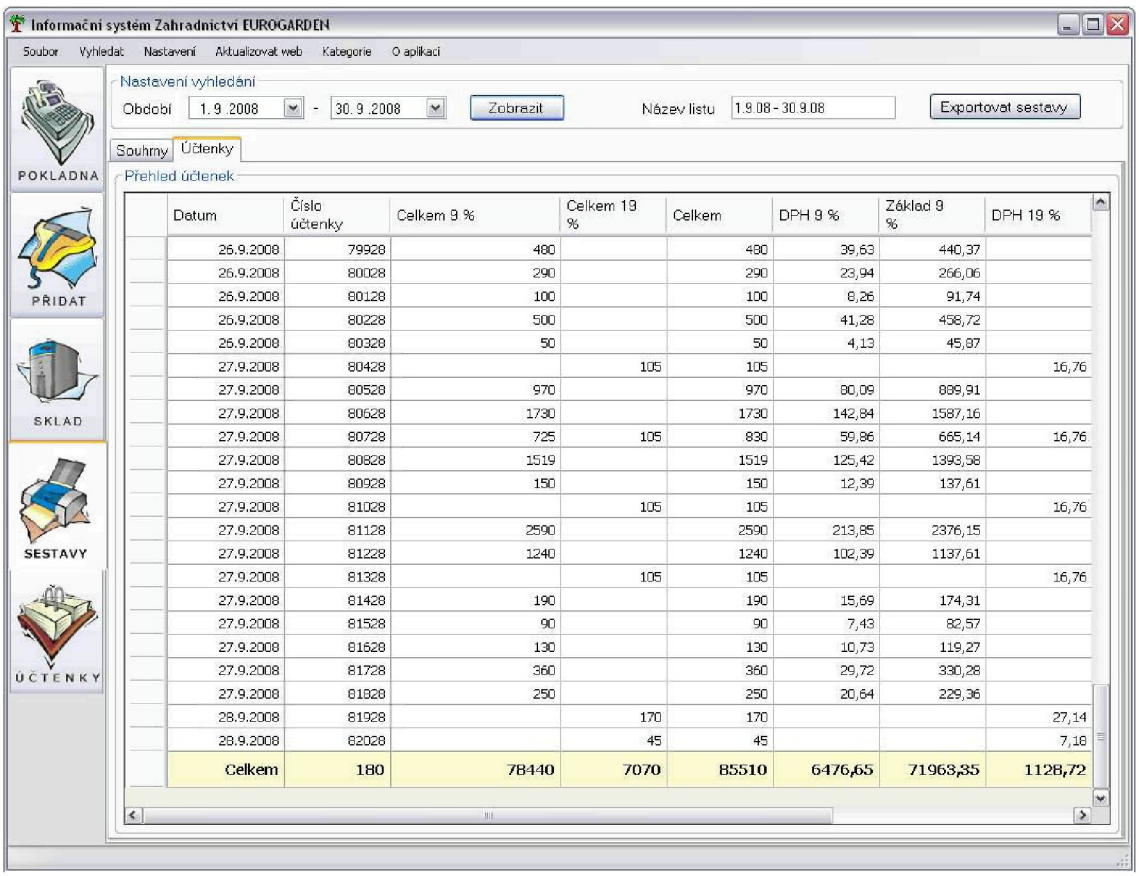
Obrázek 1.1 - Nástroje záložky pokladna



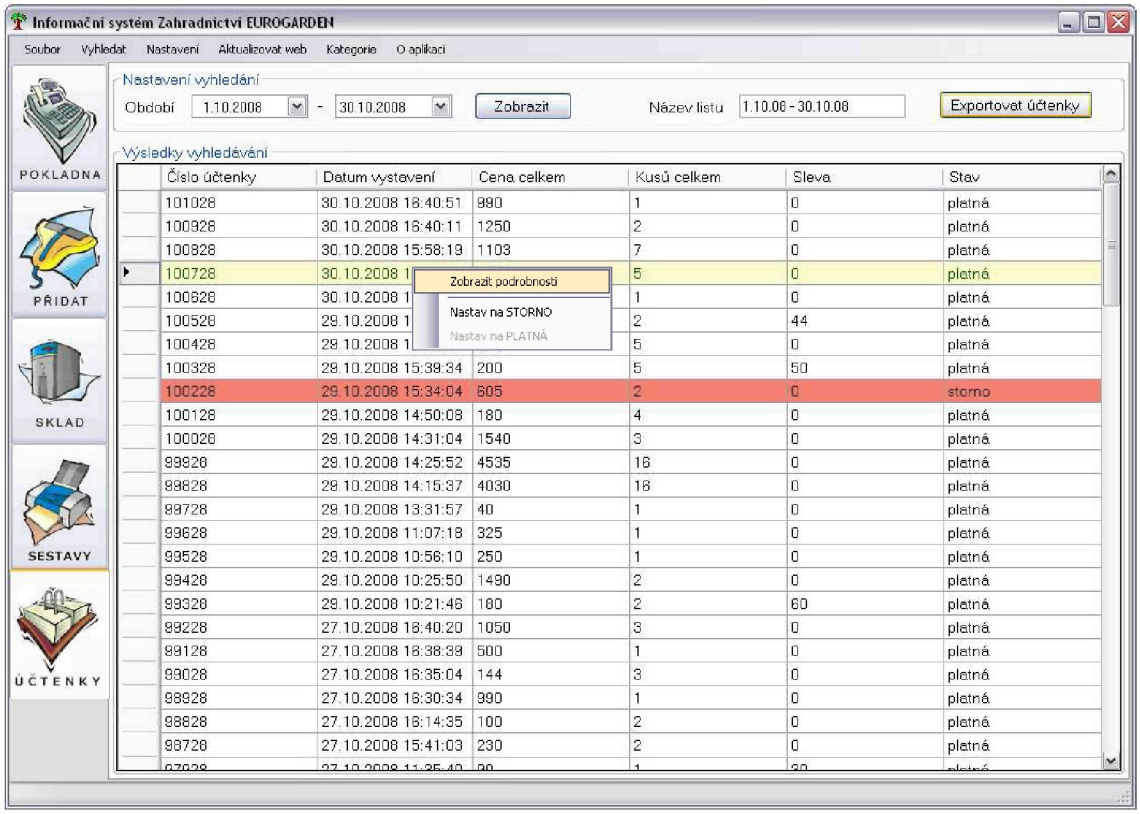
Obrázek 1.2 - Formulář pro přidání nové rostliny



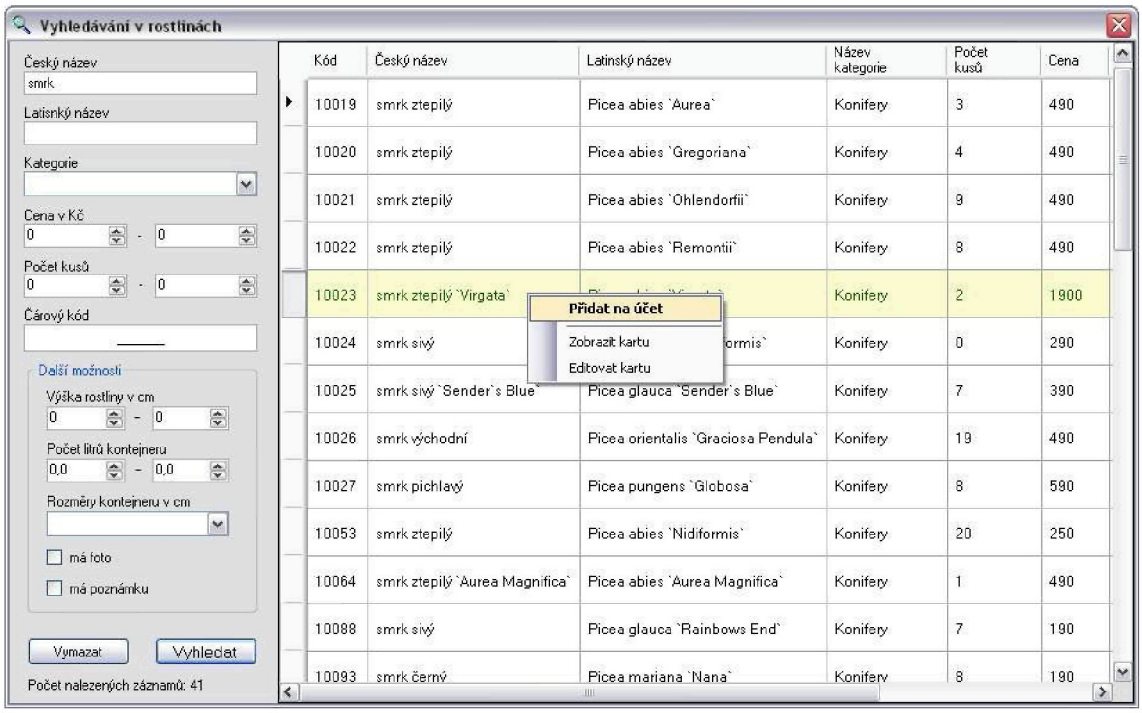
Obrázek 1.3- Zobrazení skladu rostlin



Obrázek 1.4 - Nástroje pro tvorbu účetních sestav

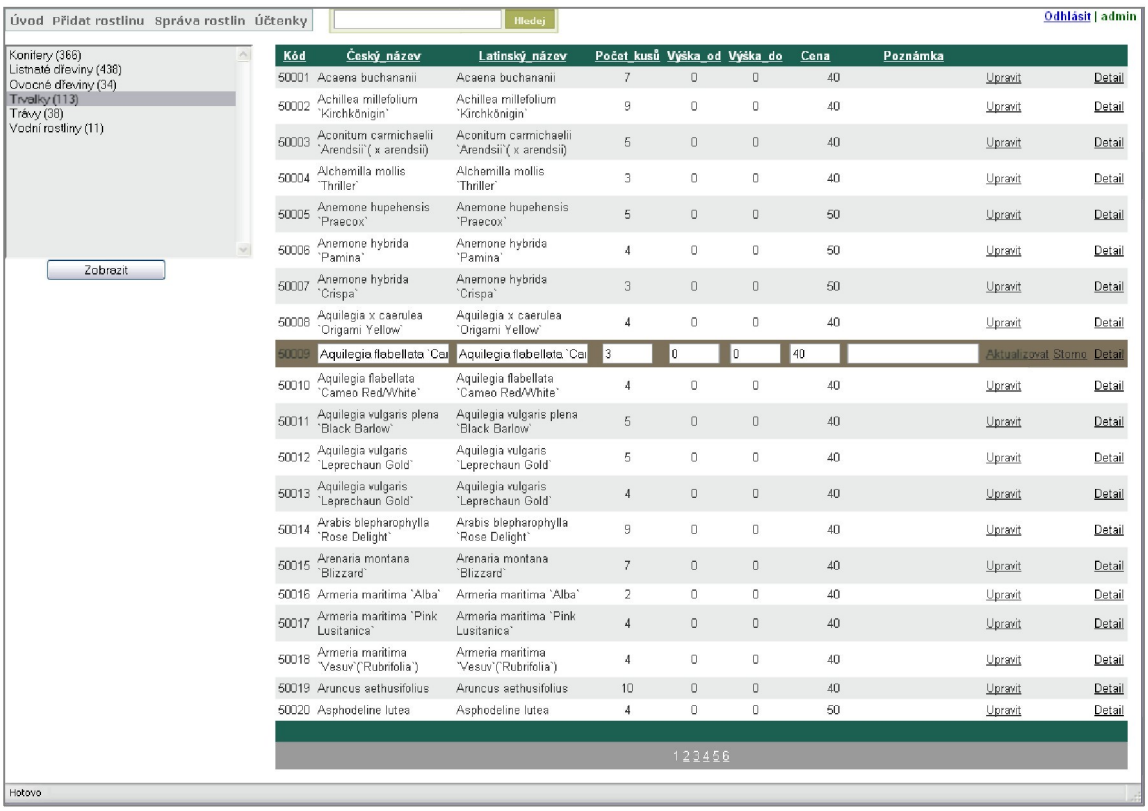


Obrázek 1.5 - Přehled vystavených účtenek



Obrázek 2 - Nástroj pro vyhledání rostlin

Obrázek 2.1 - Úvodní stránka webové aplikace s grafy



Obrázek 2.1 - Správa rostlin ve webové aplikaci

Některé zajímavé SQL procedury a funkce

3.1 Procedura pro vytvoření podkladů pro účetní souhrny

```
CREATE PROCEDURE sp_soucty_den
@od varchar(10),
@do varchar(10)
AS
SELECT CONVERT(smalldatetime, CONVERT(varchar(10),
      dbo.Uctenky.Datum_vystavení, 104), 104) AS Datum,
      dbo.Podrobnosti_uctenky.Kód,
      ROUND(SUM(dbo.Podrobnosti_uctenky.Cena_kusu -
      ((dbo.Podrobnosti_uctenky.Cena_kusu * dbo.Uctenky.Sleva + 0.0)/
      (dbo.Uctenky.Cena_celkem + dbo.Uctenky.Sleva))), 1)  AS Soucet
FROM dbo.Podrobnosti_uctenky RIGHT OUTER JOIN dbo.Uctenky ON
      dbo.Podrobnosti_uctenky.ID_Účtenky = dbo.Uctenky.ID_Účtenky
WHERE (dbo.Uctenky.Datum_vystavení BETWEEN @od AND @do) and
      dbo.Uctenky.Stav=1
GROUP BY dbo.Podrobnosti_uctenky.Kód, CONVERT(smalldatetime,
      CONVERT(VARCHAR(10), dbo.Uctenky.Datum_vystavení, 104), 104)
ORDER BY dbo.Podrobnosti_uctenky.Kód, Datum
GO
```

3.2 Funkce vracející číslo účtenky

```
CREATE  FUNCTION vrat_cu (@datum datetime)
RETURNS int
AS
BEGIN
DECLARE @cu int
IF(SELECT TOP 1 ID_Účtenky FROM Uctenky WHERE
     right(ID_Účtenky,2)=left(year(@datum),1)+''+right(year(@datum),1)
     ORDER BY ID_Účtenky DESC) IS null
begin
IF(right(year(@datum),2))>=10
   begin
     set @cu = cast(right(year(@datum),2) as int)
   end
else
   begin
   set @cu = cast(left(year(@datum),1)+''+right(year(@datum),1)as int)
   end
end
else
begin
SELECT TOP 1 @cu = ID_Účtenky FROM Uctenky WHERE
right(ID_Účtenky,2)=left(year(@datum),1)+''+right(year(@datum),1)
ORDER BY ID_Účtenky DESC
end

RETURN @cu

END
```

Zajímavé funkce a procedury v C#

4.1 Přidání nové rostliny

```
private void button3_Click_1(object sender, EventArgs e)
  {
   if(checkBox1.Checked)
       {
       jeImg = false;
       id_fotky = 0;
       }
    bool overeni = true;
    if (textBox1.Text.ToString() == "")
       {
System.Windows.Forms.MessageBox.Show("Chybí Česý název.", "Varování",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
overeni = false;
       }
if (textBox2.Text.ToString() == "")
       {
System.Windows.Forms.MessageBox.Show("Chybí Latinský název.",
"Varování", MessageBoxButtons.OK, MessageBoxIcon.Warning);
overeni = false;
       }
if (comboBox1.Text == "zvolte kategorii")
       {
System.Windows.Forms.MessageBox.Show("Zvolte kategorii", "Varování",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
overeni = false;
       }
if (comboBox2.Text == "vybrat")
       {
System.Windows.Forms.MessageBox.Show("Zvolte rozměr kontejneru",
"Varování", MessageBoxButtons.OK, MessageBoxIcon.Warning);
overeni = false;
       }
if (numericUpDown5.Value == 0)
       {
System.Windows.Forms.MessageBox.Show("Chybí cena.", "Varování",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
overeni = false;
        }
if (maskedTextBox1.Text == "")
        {
System.Windows.Forms.MessageBox.Show("Chybí čárový kód.", "Varování",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
overeni = false;
        }
this.Text = overeni.ToString();
if (overeni == true)
        {
  if (jeImg == true & textBox9.TextLength > 0)
          {
           Image image;
           image = Image.FromFile(textBox9.Text.ToString());
           String cesta = Application.StartupPath.ToString() +
"/images_ceniky/" + maskedTextBox1.Text.ToString() + ".jpg";
```

```
        image.Save(cesta);
        byte[] photo = GetPhoto(strImgName);
        if (id_puv_fotky == 0)
             {
             this.Text = jeImg.ToString() +
             maskedTextBox1.TextLength.ToString();
             sqlCommand2.CommandText = "SELECT COUNT(*) FROM Fotky";
             sqlConnection1.Open();
             mySqlDataReader2 = sqlCommand2.ExecuteReader();
                while (mySqlDataReader2.Read())
                    id_fotky = (int)mySqlDataReader2[0];
                    id_fotky = id_fotky + 1;
                    mySqlDataReader2.Close();
                    sqlConnection1.Close();
System.Data.SqlClient.SqlCommand addEmp = new
System.Data.SqlClient.SqlCommand(
"INSERT INTO Fotky (ID_Fotky, Fotka, Velikost) VALUES (@ID_Fotky,
@Fotka, @Velikost)", sqlConnection1);

addEmp.Parameters.Add("@ID_Fotky", SqlDbType.Int).Value = id_fotky;
addEmp.Parameters.Add("@Fotka", SqlDbType.Image, photo.Length).Value =
photo;
addEmp.Parameters.Add("@Velikost", SqlDbType.Int).Value =
photo.Length;

     sqlConnection1.Open();
     addEmp.ExecuteNonQuery();
     sqlConnection1.Close();
             }
             else
             {
             id_fotky = id_puv_fotky;
             System.Data.SqlClient.SqlCommand addEmp = new
             System.Data.SqlClient.SqlCommand(
             "UPDATE Fotky SET Fotka=@Fotka, Velikost=@Velikost where
ID_Fotky=@ID_Fotky", sqlConnection1);

addEmp.Parameters.Add("@ID_Fotky", SqlDbType.Int).Value = id_fotky;
addEmp.Parameters.Add("@Fotka", SqlDbType.Image, photo.Length).Value =
photo;
addEmp.Parameters.Add("@Velikost", SqlDbType.Int).Value =
photo.Length;
          sqlConnection1.Open();
          addEmp.ExecuteNonQuery();
          sqlConnection1.Close();
                    }
               }
               else
                    if (checkBox1.Checked)
                         id_fotky = 0;
                    else
                         id_fotky = id_puv_fotky;
string[] rozmery = comboBox2.Text.Split('x');
sqlConnection1.Open();
sqlCommand3.CommandText = "exec update_polozky " + rozmery[0] + "," +
rozmery[1] + "," + numericUpDown3.Value.ToString().Replace(',', '.') +
",'" + comboBox1.Text + "'," + maskedTextBox1.Text + ",'" +
apos(textBox1.Text) + "','" + apos(textBox2.Text) + "'," +
numericUpDown1.Value + "," + numericUpDown5.Value + "," +
numericUpDown2.Value + "," + numericUpDown4.Value + "," +
id_fotky+",'"+apos(textBox3.Text.ToString())+"'";
```

```
if (sqlCommand3.ExecuteNonQuery() > 0)
      {
System.Windows.Forms.MessageBox.Show("Editace položky proběhla
úspěšně.", "Potvrzení", MessageBoxButtons.OK,
MessageBoxIcon.Information);
formClear();
       if (mujParentForm12 == null)
          {
           mujParentForm8.akt_grid();
           }
           else
           {
           mujParentForm12.akt_grid();
           }
           this.Close();
           }
           else
System.Windows.Forms.MessageBox.Show("Položku se nepodařilo
zeditovat.", "Chyba", MessageBoxButtons.OK, MessageBoxIcon.Warning);
sqlConnection1.Close();
         }
}
```

4.2 Vygenerování dokumentu aplikace MS Excel

```
private void generujEXCEL2(DataGridView dataGrid)
  {
  Excel.Application oXL;
  Excel._Workbook oWB;
  Excel._Worksheet oSheet;
  Excel._Worksheet oSheet1;
  Excel.Range oRng;
   try
      {
      //Spustí aplikaci Excel a získá aplikaci jako objekt.
      oXL = new Excel.Application();
      //Načte nový sešit.
      oWB = (Excel._Workbook)(oXL.Workbooks.Add(Missing.Value));
      oSheet = (Excel._Worksheet)oWB.ActiveSheet;
oSheet.PageSetup.Orientation = Excel.XlPageOrientation.xlPortrait;
oSheet.Name = "Kategorie " + mujParentForm17.textBox5.Text;
oSheet.PageSetup.LeftHeader = "Vygenerováno " + DateTime.Now;
oSheet.PageSetup.RightHeader = "Období " +
mujParentForm17.dateTimePicker1.Value.Day + "." +
mujParentForm17.dateTimePicker1.Value.Month + "." +
mujParentForm17.dateTimePicker1.Value.Year + " - " +
mujParentForm17.dateTimePicker2.Value.Day + "." +
mujParentForm17.dateTimePicker2.Value.Month + "." +
mujParentForm17.dateTimePicker2.Value.Year;

//Přidá záhlaví tabulky buňku po buňce.
int sloupce = mujParentForm17.pocetKatDph * 3 + 2;
     for (int h = 1; h < dataGrid.Rows.Count + 1; h++)
         {
          for (int i = 0; i < sloupce; i++)
            {
             if (h == 1)
                {
                 switch (i)
```

```
        {
        case 0:
        oSheet.Cells[1, 1] = dataGrid.Columns[0].Name.ToString();
        oSheet.get_Range("A1", Convert.ToString(preved(sloupce - 1) +
        "1")).Borders.LineStyle = 1;
        oSheet.get_Range("A2", Convert.ToString(preved(sloupce - 1) +
        Convert.ToString(dataGrid.Rows.Count))).Borders.LineStyle = 1;
        oSheet.get_Range("A2", Convert.ToString(preved(sloupce - 1) +
        Convert.ToString(dataGrid.Rows.Count))).Borders.Weight = 2;
        break;
        default:
        oSheet.Cells[1, i + 1] = dataGrid.Columns[dataGrid.Columns.Count
        - sloupce + i].Name.ToString();
        oSheet.get_Range("A1", Convert.ToString(preved(sloupce - 1) +
        "1")).Borders.LineStyle = 1;
        oSheet.get_Range("A2", Convert.ToString(preved(sloupce - 1) +
        Convert.ToString(dataGrid.Rows.Count))).Borders.LineStyle = 1;
        oSheet.get_Range("A2", Convert.ToString(preved(sloupce - 1) +
        Convert.ToString(dataGrid.Rows.Count))).Borders.Weight = 2;
        break;
        }
    }
    else
    {
     switch (i)
      {
        case 0:
        oSheet.Cells[h, i + 1] = dataGrid[i, h - 2].Value.ToString();
        break;
        default:
        oSheet.Cells[h, i + 1] = dataGrid[dataGrid.Columns.Count -
        sloupce + i, h - 2].Value.ToString();
        break;
        }
     }
 }
backgroundWorker1.ReportProgress(1);
}

oSheet.get_Range(Convert.ToString(preved(0) +
Convert.ToString(dataGrid.Rows.Count)),
Convert.ToString(preved(mujParentForm17.pocetKatDph * 3 + 1) +
Convert.ToString(dataGrid.Rows.Count))).Columns.Font.Bold = true;

for (int h = 0; h <= mujParentForm17.pocetKatDph; h++)
   {
     if (h == 0)
        {
        oSheet.get_Range(Convert.ToString(preved(1) +
        Convert.ToString(dataGrid.Rows.Count + 3)),
        Convert.ToString(preved(3) +
        Convert.ToString(dataGrid.Rows.Count + 4 +
        mujParentForm17.pocetKatDph))).Borders.LineStyle = 1;
        oSheet.get_Range(Convert.ToString(preved(1) +
        Convert.ToString(dataGrid.Rows.Count + 3)),
        Convert.ToString(preved(3) +
        Convert.ToString(dataGrid.Rows.Count + 4 +
        mujParentForm17.pocetKatDph))).Borders.Weight = 2;
        oSheet.Cells[dataGrid.Rows.Count + 3, 2] = "DPH REKAPITULACE";
        oSheet.get_Range(Convert.ToString(preved(1) +
        Convert.ToString(dataGrid.Rows.Count + 3)),
```

```
        Convert.ToString(preved(3) +
        Convert.ToString(dataGrid.Rows.Count + 4))).Columns.Font.Bold =
        true;
        oSheet.get_Range(Convert.ToString(preved(1) +
        Convert.ToString(dataGrid.Rows.Count + 3)),
        Convert.ToString(preved(3) +
        Convert.ToString(dataGrid.Rows.Count + 3))).MergeCells = true;
        oSheet.get_Range(Convert.ToString(preved(1) +
        Convert.ToString(dataGrid.Rows.Count + 3)),
        Convert.ToString(preved(3) +
        Convert.ToString(dataGrid.Rows.Count + 3))).HorizontalAlignment
        = Excel.XlHAlign.xlHAlignCenter;

        oSheet.Cells[dataGrid.Rows.Count + 4, 2] = "Sazba %";
        oSheet.Cells[dataGrid.Rows.Count + 4, 3] = "Základ Kč";
        oSheet.Cells[dataGrid.Rows.Count + 4, 4] = "Daň Kč";
        oSheet.get_Range(Convert.ToString(preved(1) +
        Convert.ToString(dataGrid.Rows.Count + 4)),
        Convert.ToString(preved(3) +
        Convert.ToString(dataGrid.Rows.Count + 4))).HorizontalAlignment
        = Excel.XlHAlign.xlHAlignRight;

      }
      else
      {
        for (int i = 0; i < 3; i++)
         {
         oSheet.Cells[dataGrid.Rows.Count + h + 4, i + 2] =
         mujParentForm17.dataGridView6[i, h - 1].Value.ToString();
         }
      }
      backgroundWorker1.ReportProgress(1);
  }

oSheet.get_Range("A1", "IV1").Font.Bold = true;
oSheet.get_Range("A1", "IV1").VerticalAlignment =
Excel.XlVAlign.xlVAlignCenter;
oSheet.get_Range("A1", "A255").Font.Bold = true;
oSheet.get_Range("A1", "A255").VerticalAlignment =
Excel.XlVAlign.xlVAlignCenter;
oSheet.get_Range("B2", "IV255").NumberFormat = "0,00";
oRng = oSheet.get_Range("A1", "IV250");
oRng.EntireColumn.AutoFit();
oXL.UserControl = true;
oSheet1 = (Excel._Worksheet)oWB.Sheets[2];
generujEXCEL3(mujParentForm17.dataGridView7, oSheet1);
oXL.Visible = true;
}
 catch (Exception theException)
  {
   String errorMessage;
   errorMessage = "Chyba: ";
   errorMessage = String.Concat(errorMessage, theException.Message);
   errorMessage = String.Concat(errorMessage, " Řádek: ");
   errorMessage = String.Concat(errorMessage, theException.Source);
   MessageBox.Show(errorMessage, "Chyba");
  }
}
```

4.3 Nahrání souborů na server pomocí FTP

```
private void UploadFile(string filename)
  {
  string[] konc = filename.Split('.');
  if (konc[1] != "xml")
  filename = "/images_ceniky/" + filename;
  else
  filename = "/" + filename;
  FileInfo fileInf = new FileInfo(Application.StartupPath.ToString() +
  filename);
  string uri = "ftp://" + ftpServerIP + "/" + filename;
  FtpWebRequest reqFTP;
  reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri(
  "ftp://" + ftpServerIP + "/" + filename));
   toolStripStatusLabel1.Text = "ftp://" + ftpServerIP + "/" +
   filename;
   reqFTP.Credentials = new NetworkCredential("eurogarden.cz",
   "heslo_pro_pristup_pres_ftp");
   reqFTP.KeepAlive = false;
   reqFTP.Method = WebRequestMethods.Ftp.UploadFile;
   reqFTP.UseBinary = true;
   reqFTP.ContentLength = fileInf.Length;
   int buffLength = 2048;
   byte[] buff = new byte[buffLength];
   int contentLen;
   FileStream fs = fileInf.OpenRead();
      try
        {
          Stream strm = reqFTP.GetRequestStream();
          contentLen = fs.Read(buff, 0, buffLength);
         while (contentLen != 0)
             {
             strm.Write(buff, 0, contentLen);
             contentLen = fs.Read(buff, 0, buffLength);
              }
             strm.Close();
             fs.Close();
             toolStripStatusLabel2.Text = "Přeneseno";
             }
             catch (Exception ex)
             {
               MessageBox.Show(ex.Message, "Upload Error");
               toolStripStatusLabel2.Text = "Nastala chyba přenosu";
             }
   }
```

Grafický výstup procesu tvorby účtenek

Datum	Číslo účtenky	Celkem 9%	Celkem 19%	Celkem	DPH 9%	Základ 9%	DPH 19%	Základ 19%	Základy celkem	DPH celkem
2.10.2008	829	180,00	80,00	260,00	14,86	165,14	12,77	67,23	232,37	27,63
2.10.2008	830		525,00	525,00			83,82	441,18	441,18	83,82
2.10.2008	831	3 645,00		3 645,00	300,96	3 344,04			3 344,04	300,96
2.10.2008	832	440,00	374,00	814,00	36,33	403,67	59,71	314,29	717,96	96,04
2.10.2008	833		115,00	115,00			18,36	96,64	96,64	18,36
3.10.2008	834	670,00		670,00	55,32	614,68			614,68	55,32
3.10.2008	835	440,00		440,00	36,33	403,67			403,67	36,33
3.10.2008	836	50,00		50,00	4,13	45,87			45,87	4,13
3.10.2008	837	410,00		410,00	33,85	376,15			376,15	33,85
3.10.2008	838	250,00		250,00	20,64	229,36			229,36	20,64
3.10.2008	839	150,00		150,00	12,39	137,61			137,61	12,39
3.10.2008	840	590,00		590,00	48,72	541,28			541,28	48,72
3.10.2008	841	420,00		420,00	34,68	385,32			385,32	34,68
3.10.2008	842	250,00		250,00	20,64	229,36			229,36	20,64
3.10.2008	843	200,00		200,00	16,51	183,49			183,49	16,51
3.10.2008	844	425,00		425,00	35,09	389,91			389,91	35,09
4.10.2008	845	350,00		350,00	28,90	321,10			321,10	28,90
4.10.2008	846	250,00		250,00	20,64	229,36			229,36	20,64
4.10.2008	847	1 880,00		1 880,00	155,23	1 724,77			1 724,77	155,23
4.10.2008	848	290,00		290,00	23,94	266,06			266,06	23,94
4.10.2008	849	450,00		450,00	37,16	412,84			412,84	37,16
4.10.2008	850	250,00		250,00	20,64	229,36			229,36	20,64
4.10.2008	851		175,00	175,00			27,94	147,06	147,06	27,94
5.10.2008	852	905,00	70,00	975,00	74,72	830,28	11,18	58,82	889,10	85,90
5.10.2008	853	150,00		150,00	12,39	137,61			137,61	12,39
6.10.2008	854	110,00		110,00	9,08	100,92			100,92	9,08
6.10.2008	855	90,00		90,00	7,43	82,57			82,57	7,43
6.10.2008	856	900,00		900,00	74,31	825,69			825,69	74,31
8.10.2008	857		230,00	230,00			36,72	193,28	193,28	36,72
8.10.2008	858	6 250,00		6 250,00	516,06	5 733,94			5 733,94	516,06
8.10.2008	859	750,00		750,00	61,93	688,07			688,07	61,93
8.10.2008	860	290,00		290,00	23,94	266,06			266,06	23,94
8.10.2008	861	650,00		650,00	53,67	596,33			596,33	53,67
8.10.2008	862	600,00		600,00	49,54	550,46			550,46	49,54
8.10.2008	863	460,00	115,00	575,00	37,98	422,02	18,36	96,64	518,66	56,34
8.10.2008	864	250,00		250,00	20,64	229,36			229,36	20,64
8.10.2008	865	420,00		420,00	34,68	385,32			385,32	34,68
8.10.2008	866	960,00		960,00	79,27	880,73			880,73	79,27
Celkem	**kusů 38**	**24 375,00 Kč**	**1 684,00 Kč**	**26 059,00 Kč**	**2 012,60 Kč**	**22 362,40 Kč**	**268,86 Kč**	**1 415,14 Kč**	**23 777,54 Kč**	**2 281,46 Kč**

DFD diagram procesu přidání nové účtenky