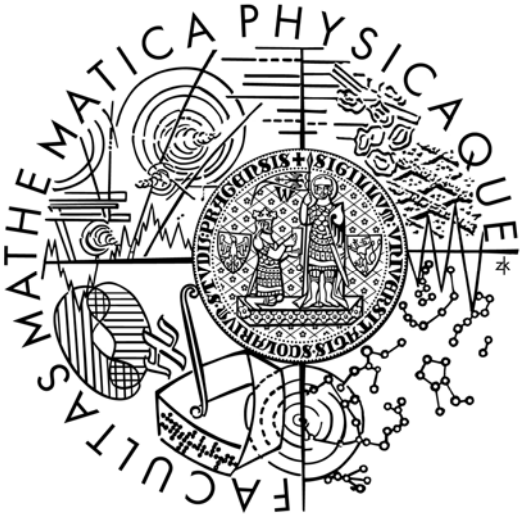
UNIVERZITA KARLOVA V PRAZE
MATEMATICKO-FYZIKÁLNÍ FAKULTA

BAKALÁŘSKÁ PRÁCE




Jaroslav Pastorek

# Informační systém pro obchodníka s cennými papíry

Katedra softwarového inženýrství

VEDOUCÍ BAKALÁŘSKÉ PRÁCE:
RNDr. Tomáš Tichý
Katedra aplikované matematiky

Studijní program: informatika, obor programování

2007

***Poďakovanie***
Rád by som poďakoval všetkým, ktorí sa ma podporovali pri vývoji projektu a písaní samotnej práce. Ďakujem môjmu vedúcemu za usmerňovanie môjho snaženia.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce

V Prahe dňa 9.8.2007

Jaroslav Pastorek

# Obsah

**Názov práce:** Informační systém pro obchodníka s cennými papíry
**Autor:** Jaroslav Pastorek
**Katedra (ústav):** Katedra softwarového inženýrství
**Vedúci bakalárskej práce:** RNDr. Tomáš Tichý
**E-mail vedúceho:** tichy@math.cas.cz

**Abstrakt:**
Bakalárska práca sa zaoberá návrhom a implementáciou informačného systému typu klient - server pre obchodníka s cennými papiermi. Systém umožňuje uchovávanie a sledovanie najdôležitejších evidencií - klientov, zmlúv, platieb, pokynov na obchodovanie a príslušných číselníkov. Taktiež je možné jeho prostredníctvom vykonávať vybrané služby Centrálneho depozitára cenných papierov SR a obchody prostredníctvom Burzy cenných papierov Bratislava v súlade s legislatívnymi podmienkami Slovenskej Republiky.

**Kľúčové slová:** cenné papiere, obchodník, služby CDCP, BCPB, databáza, C#

**Title:** Information system for capital brokers
**Author:** Jaroslav Pastorek
**Department:** Department of Software Engineering
**Supervisor:** RNDr. Tomáš Tichý
**Supervisor's email address:** tichy@math.cas.cz

**Abstract:**
Bachelor thesis considers design and implementation of information system for capital broker based on client – server architecture. System provides storage for most important evidencies - clients, contracts, payments, trade orders and other respective evidencies. System also provides functionality for creating sevices of Centrálny depozitár cenných papierov SR and operating trades through Burza cenných papierov Bratislava. All operations accord to legal conditions of Slovenská republika.
**Keywords:** stocks, CDCP services, BCPB, database, C#

# 1. Úvod

Rastúci dopyt po investičných službách spolu so zvýšeným tlakom kontrolných orgánov, ako sú Národná banka Slovenska[1], Centrálny depozitár cenných papierov[2] alebo Burza cenných papierov Bratislava[3], vytvárajú aj zvýšené nároky na obchodníka s cennými papiermi, ktorý musí zvládnuť veľký objem vykonávaných obchodov pri zachovaní všetkých zákonných podmienok. Tento stav si vyžaduje vysoko špecifické programové vybavenie, ktoré však nie je bežne dostupné, keďže sa nejedná o bežný účtovný alebo skladový systém. Na trhu je viacero riešení, avšak väčšina z nich sú rozsiahle a drahé bankové systémy, v ktorých je samotný obchod s cennými papiermi[4] okrajovou záležitosťou a ponúkajú funkcionalitu, ktorú nie každý obchodník potrebuje, alebo sú to staršie systémy, ktoré nevyhovujú novým požiadavkám trhu.

Cieľom tejto práce je preto navrhnúť a implementovať systém, ktorý by pokrýval práve najdôležitejšie nároky a najbežnejšie operácie z každodennej praxe obchodníka a zároveň zodpovedal zákonným normám.

Problematika cenných papierov a obchodov s nimi je natoľko rozsiahla, že presahuje rozsah bakalárskej práce, možno dokonca aj softwarového projektu, avšak keďže ma táto problematika zaujala, rozhodol som sa vyvinúť systém, ktorý pokrýva určitú ucelenú časť práce obchodníka s cennými papiermi.

V ďalšom texte nasleduje vytýčenie rozsahu a kontextu práce spolu s úvodom do problematiky obchodovania s cennými papiermi, analýza a návrh riešenia a popis samotnej implementácie.

---

1 – ďalej len NBS
2 – ďalej len CDCP
3 – ďalej len BCPB
4 – ďalej aj CP

# 2. Špecifikácia problému a rozsah práce

Základnou potrebou každého obchodníka je evidovať informácie o obchodoch, vykonávať samotné obchody a podávať o nich informácie a hlásenia kontrolným orgánom. Toto hrubé definovanie a rozdelenie jeho činností nám poslúži pri návrhu systému, je potrebné však podrobnejšie popísať časti a činnosti, z ktorých sa každá táto potreba skladá. Je však zároveň nutné vymedziť, čo bude predmetom samotnej práce, identifikovať a vybrať funkcie, ktoré musí spĺňať každý systém pre obchodníka s cennými papiermi a definovať oblasti, ktoré budú ponechané na implementáciu v prípadných rozšíreniach projektu.

## *2.1. Evidencie*

Nie je možné, aby obchodník obchodoval cenné papiere a neviedol o tom žiadne záznamy, jednak je to v priamom rozpore so zákonom, dokonca by takýto obchodník nebol schopný žiadne obchody vykonávať. V procese získavania požiadaviek a špecifikácie systému boli nájdené základné entity, ktoré systém nutne musí uchovávať- klienti, zmluvy, platby, pokyny na obchodovanie a jednotlivé obchody. Pri bližšom pohľade samozrejme vystúpia do popredia aj iné, preto bola nutné každej z nich venovať náležitú pozornosť.

### *Klienti*

Zásadná zložka celej evidencie, aj keď na prvý pohľad nemusí byť jasná jej dôležitosť. Obchody, ktoré obchodník vykonáva však nemôžu byť anonymné, nie je možné cenné papiere nakupovať takpovediac do vzduchu. Pri každom klientovi je nutné uchovávať jeho identifikačné a kontaktné údaje v rozsahu, ktorý vyžaduje zákon.

### *Zmluvy*

S každým klientom musí byť riadne uzatvorená a podpísaná zmluva na obchody s cennými papiermi (zmluva môže obsahovať viacero obchodov s rôznymi cennými papiermi – tu vystupuje do popredia evidovať zoznam cenných papierov). Systém preto eviduje elektronický ekvivalent zmluvy. Existencia zmluvy je dôležitá aj pre správne peňažné vysporiadanie s klientom, keďže pri niektorých typoch obchodov kupec musí najprv poukázať finančné prostriedky na klientský účet obchodníka, prípadne obchodník musí vyplatiť klientovi peniaze získané pri predaji jeho cenných papierov. Každá zmluva musí byť finančne vyrovnaná, čiže na nej musí byť nulový zostatok peňažných prostriedkov. Každá zmluva je zviazaná s maklérom, ktorý ju s klientom uzatvoril. Preto je potrebné viesť aj evidenciu obchodníkových maklérov.

### *Makléri*

Evidencia maklérov slúži hlavne pre potreby identifikovania makléra priradeného k zmluve a neskôr je potrebné pripojiť ho k výstupným hláseniam a správam o obchodoch.

### *Cenné papiere*

Pri analýze požiadavku evidovať zmluvy sa objavila potreba uchovávať zoznam cenných papierov. Každý cenný papier je označený identifikátorom ISIN (International Securities Identifying Number), čo je dvanásťmiestny alfanumerický kód pozostávajúci z dvoch písmen kódu krajinu, deväť miestneho národného identifikátora a kontrolnej číslice (avšak nie všetky cenné papier tento formát spĺňajú). Cenný papier je vydaný emitentom, ktorý je registrovaný na CDCP a je charakterizovaný počtom emitovaných CP vo všetkých emisiách, počtom vrátených CP, stavom emisie a dátumom registrácie emisie. Avšak toto sú len povinné položky, CP má veľké množstvo atribútov, ktorých existencia a hodnota závisia od druhu cenného papiera.

Zoznam cenných papierov sa získava z CDCP vo forme xml súboru, avšak systém musí umožňovať aj vytváranie CP (takzvané listinné CP).

### *Emitenti*

Keďže cenný papier je vydávaný emitentom, musí byť uchovávaná aj evidencia emitentov. Každý emitent je identifikovaný číslom IČO alebo rodným číslom, názvom(menom, priezviskom) a typom osoby.

Evidencia emitentov sa napĺňa z rovnakého xml súboru ako evidencia CP. Z dôvodu vytvárania listinných CP je dôležité, aby bolo možné do evidencie pridávať a editovať nových emitentov.

### *Platby*

Potreba uchovávať platby plynie jednak z nutnosti vykonania finančného vyrovnania a zároveň pri kúpe cenných papierov nie je možné vykonať obchod skôr, ako klient poukáže peniaze na účet obchodníka. Každá platba je preto viazaná na zmluvu, aby bolo možné vykonať uvedené operácie.

Uchováva sa preto číslo účtu na ktorý alebo z ktorého platba prišla, typ platby (príjem, výdaj, obchodník potrebuje možnosť vytvárať ďalšie odvodené typy platieb od týchto základných) a jej finančný objem.

### *Pokyny na obchodovanie*

Každá zmluva obsahuje záväzok zobchodovať určitý počet cenných papierov. Ako bolo spomenuté, zmluva môže obsahovať rôzne cenné papiere, dokonca rôzne typy obchodov. Preto sa ku každému typu obchodu a cennému papieru (ktorý je identifikovaný číslom ISIN) vytvorí pokyn na obchodovanie.

Existujú pokyny rôznych druhov, podľa typov obchodov, ktoré reprezentujú a trhov, na ktorých sa obchody vykonávajú. V tejto práci sú uvažované anonymné obchody (označované aj ako fixing) a nahlasované priame obchody (NPO) prostredníctvom BCPB a obchody s listinnými cennými papiermi. Podrobnejší popis je uvedený v kapitole Obchody a služby

Okrem už spomenutých atribútov ako sú druh pokynu a trh musí byť vymedzená aj doba platnosti. Pokyn sa stáva platným zadaním do systému, dátum konca platnosti vyplní obchodník pri jeho vytváraní, avšak nemôže existovať pokyn s neobmedzenou dobou platnosti.

Z pokynu musí byť jasné, koľko a akých cenných papierov sa má obchodovať. Taktiež je dôležité, aby bola uvedená cena, za ktorú sa má obchodovať daný CP.

Pokyn po jeho vytvorení nie je možné editovať, a to ani v prípade jeho chybného zadania. Takýto pokyn sa označí ako chybne zadaný a bude tak aj uvádzaný vo výpise z evidencie.

### *Obchody*

Pokyn reprezentuje príkaz na obchod, avšak obchod nemusí byť vykonaný naraz v celom objeme a za jednotnú cenu (aj keď pri niektorých typoch obchodov je to typická situácia). Preto je nutné evidovať samotný vykonaný obchod.

## *2.2. Obchody a služby*

Z rôznych kombinácií druhov obchodov, služieb a trhov, na ktorých ich je možné realizovať, boli vybrané tie, ktoré obchodník využíva najčastejšie. Preto sa práca zameriava na obchody realizované prostredníctvom BCPB (NPO a anonymné obchody) a na obchod s listinnými CP.

Dôležitým detailom, ktorý je nutné spomenúť pred samotnou analýzou jednotlivých druhov obchodov a služieb je priebeh finančného vysporiadania. Pri niektorých druhoch obchodov je totiž možné zvoliť, či peniaze za zobchodované CP pošlú kupec na účet obchodníka a ten potom pošle peniaze predávajúcemu (ako je popísané pri evidencii zmlúv), alebo majú klienti dohodnutý transfer finančných prostriedkov iným spôsobom (toto nie je možné pri anonymných obchodoch, vtedy peniaze putujú vždy prostredníctvom obchodníka). Do finančného vysporiadania treba zahrnúť aj poplatky, ktoré obchodník vyberá od klientov za vykonané obchody.

Pred vykonaním niektorých obchodov je nutné zriadiť špecifické služby (tie sú konkrétne popísané pri jednotlivých obchodoch a potom aj samostatne) či už pre klienta samotného alebo obchodované cenné papiere.

Obchodovať sa môže iba taký pokyn, pre ktorý ešte neuplynula doba platnosti, má zriadené všetky potrebné služby a v prípade že sa jedná o obchod s finančným vyrovnaním je nutné, aby kupec odoslal na klientsky účet obchodníka dostatok finančných prostriedkov.

### *Anonymné obchody*

Ak má klient záujem obchodovať svoje cenné papiere, avšak nie je ešte známa protistrana obchodu, ide o takzvaný anonymný obchod. Klient vlastne ponúka svoje CP na predaj, prípadne predkladá na trh ponuku na ich kúpu. Špecifikuje pri tom limitnú cenu, za ktorú je ochotný cenné papiere obchodovať, čiže v prípade nákupu maximum, pri predaji minimum. Skutočná cena potom musí dodržať tieto limity.

Pri anonymnom obchodovaní prostredníctvom BCPB je potrebné, aby bol obchodník registrovaný na účte klienta, čo je možné dosiahnuť zriadením služby E01 (podrobne bude popísaná neskôr). Ďalej je pri predaji nutné zabezpečiť, aby bol obchodník priradený k cenným papierom klienta, na čo slúži služba O65. Odporúča sa taktiež blokovať cenné papiere na účte klienta prostredníctvom služby O23, avšak táto operácia na rozdiel od služby O65 nie je pri anonymnom predaji povinná.

Keďže protistrana nie je pri zadávaní obchodu známa, prebieha finančné vysporiadanie s účasťou obchodníka. Pri kúpe cenných papierov preto musí klient zabezpečiť dostatočný objem finančných prostriedkov na klientskom účte obchodníka.

Samotný obchod prebieha v podobe objednávky na obchod prostredníctvom súboru exportovaného systémom, ktorý sa následne importuje do burzového systému (SPOZUS).

### *Nahlasované priame obchody*

V prípade, že kupec aj predajca cenných papierov sú známi, je možné vykonať prostredníctvom BCPB takzvaný priamy obchod. Aj v tomto prípade je potrebné, aby obaja účastníci obchodu boli registrovaní na BCPB a aby predávajúci mal zriadené službu O65, prípadne O23. Obchod samotný opäť prebieha prostredníctvom exportného súboru.

Priame obchody je možné vykonávať aj medzi klientmi rôznych obchodníkov, projekt sa však zaoberá len situáciou, kedy sú obe strany klientmi jedného a toho istého obchodníka. Prípadné rozšírenie funkcionality by však nemalo byť náročné.

### *Listinné cenné papiere*

Pri listinných CP nie je nutné obchod vykonávať prostredníctvom BCPB či CDCP. Všetky kompentencie sú v rukách emitenta (ten už však musí byť registrovaný na CDCP).

Typickou situáciou pri obchode s listinnými CP je jeden kúpny/ predajný pokyn na ktorý sa párujú protipokyny. Je dôležité, aby cena za jeden CP bola pri oboch pokynoch rovnaká.

Tento typ obchodu môže prebiehať s finančným vyrovnaním prostredníctvom obchodníka ale aj bez neho. Nie je potrebné vykonávať žiadne špeciálne hlásenie o vykonaní obchodu, je ale nutné uchovať informácie o ňom pre potreby kontroly.

### *Vysporiadanie*

Po vykonaní obchodov prostredníctvom BCPB je nutné spracovať obchody, ktoré boli vykonané. Spracovanie spočíva v odpočítaní cenných papierov z pokynu, ku ktorému sa obchod vzťahuje, pripísaní alebo odpísaní finančných prostriedkov k zmluve klienta (a neskôr aj v ich poukázaní na účet klienta, prípadne ich vyžiadaní) – to v prípade, že finančného vysporiadania sa zúčastňoval obchodník. Taktiež je nutné určiť poplatky, ktoré bude obchodník vyžadovať za vykonané služby.

Vysporiadanie prebieha v dvoch fázach. Počas prvej fázy je na konci obchodného dňa z BCPB k dispozícii takzvaný **A súbor** s obchodmi, ktoré obchodník v ten deň vykonal. Tento slúži na priradenie klientov k obchodom, keďže v prípade anonymných obchodov nie je nutné vykonávať všetky obchody prostredníctvom exportovaného súboru s objednávkou, ale maklér môže zobchodovať istý počet cenných papierov, o ktorých vie, že pre ne sú vytvorené pokyny, samozrejme za cenu, ktorá vyhovuje cenovým limitom. Tieto cenné papiere je potom nutné rozdeliť jednotlivým klientom s vyhovujúcimi pokynmi, pričom sa uspokojujú pokyny v poradí v akom boli prijaté.

Keď sú všetky cenné papiere priradené k pokynom (respektíve klientom), exportuje sa **B súbor** na BCPB, v ktorom sú údaje o priradených obchodoch a klientoch.

Do troch dní je na BCPB k dispozícii súbor konečného vysporiadania – **C súbor**. V ňom sú uvedené všetky obchody už priradené k pokynom a stav ich majetkového vysporiadania. V prípade, že obchod a prevod cenných papierov prebehol korektne, obsahuje položka s obchodom návratový kód 100. Vtedy je obchod definitívne uzatvorený a môže sa pristúpiť k určeniu poplatkov a finančnému vysporiadaniu (ak je potrebné).

Ako už z predošlého textu plynie, vysporiadanie sa týka iba NPO a anonymných obchodov, pri listinných CP nie je nutné sa ním zaoberať.

### *Registrácia na CDCP – služba E01*

Pred vykonaním obchodov prostredníctvom BCPB je nutné najprv zaregistrovať obchodníka na účte klienta na CDCP. Na to slúži služba E01. Jej vytvorenie prebieha prostredníctvom súboru, v ktorom sú údaje o klientoch, pre ktorých obchodník žiada zriadiť službu. Odpoveďou na žiadosť je súbor, v ktorom je indikátorom úspešného zriadenia služby návratový kód 100.

Registráciu na CDCP je nutné zriadiť len pred prvým obchodom prostredníctvom BCPB. Pri ďalších obchodoch je už služba zriadená.

### *Záznam osoby oprávnenej nakladať – služba O65*

Pri predaji cenných papierov je nutné priradiť obchodníka k určitému počtu cenných papierov jednej emisie na účte majiteľa. Na tento účel slúži služba O65. Jej vytvorenie je podobné ako vytvorenie služby E01, exportuje sa súbor s údajmi o klientovi, dátume, do ktorého má byť služba platná, identifikáciou emisie cenného papieru a jeho počte. Odpoveďou je opäť súbor, v ktorom je ku každej žiadosti pripojený návratový kód, ktorého hodnota 100 indikuje úspešné vytvorenie služby.

Zriadenie služby O65 je povinné iba pri predaji cenných papierov prostredníctvom BCPB, inak jej existencia nie je nutná.

### *Pozastavenie práva nakladať – služba O23*

Služba O23 zabezpečuje pozastavenie práv nakladať s určitým počtom cenných papierov jednej emisie na účte majiteľa v prospech obchodníka. Spôsob jej vytvorenia je identický s tvorbou služby O65.

Odporúča sa zriadiť túto službu pre anonymné predaje, aby sa zabránilo iným obchodníkom manipulovať s cennými papiermi klienta a predišlo sa tak ďalším problémom, ktoré by mohli nastať napríklad pri vysporiadaní.

## *2.3. Hlásenia a správy*

Neoddeliteľnou súčasťou práce obchodníka sú aj kontrolné hlásenia a správy. Hlavnými sú denník prijatých pokynov a denník obchodov obchodníka.

Systém preto musí umožňovať generovanie a tlač zoznamu všetkých pokynov, ktoré boli zadané do evidencie a všetkých obchodov, ktoré boli prostredníctvom obchodníka vykonané za dané časové obdobie.

## *2.4. Ďalšie požiadavky na systém a súhrn*

Vyššie uvedený text slúži ako úvod do problematiky obchodovania s cennými papiermi a pre jednoduchšie pochopenie jednotlivých funkcií systému a u ich účelu. Zároveň je to aj ako funkčná špecifikácia celého systému.

Systém bude umožňovať súčasný prístup k evidenciám a prácu s nimi viacerým užívateľom primárne v rámci podnikovej siete LAN s ohľadom na možné konflikty pri práci. Platformou, na ktorej bude systém nasadený bude Microsoft Windows XP. Taktiež bude vytvorený systém autentifikácie a autorizácie užívateľov.

Systém bude umožňovať:

i. Evidovať klientov, to znamená vytvoriť klienta, editovať jeho atribúty prípadne zmazať klienta.

ii. Evidovať zmluvy

iii. Evidovať maklérov

iv. Evidovať platby

v. Evidovať pokyny na obchody

vi. Evidovať emitované cenné papiere, pričom je možné načítať ich zoznam zo výstupného xml súboru služby CDCP d089

vii. Evidovať emitentov

viii. Vykonávať NPO a anonymné obchody prostredníctvom BCPB a obchody s listinnými cennými papiermi, vykonané obchody budú evidované pre potreby kontroly a vysporiadania

ix. Zriaďovať služby E01, O65 a O23 pre potreby obchodovania

x. Vysporiadanie obchodov, či už majetkové alebo finančné (ak je potrebné)

xi. Vytváranie denníka obchodov a prijatých pokynov

xii. Súčasnú prácu viacerých užívateľov

xiii. Riadenie prístupu, správu užívateľov a užívateľských práv

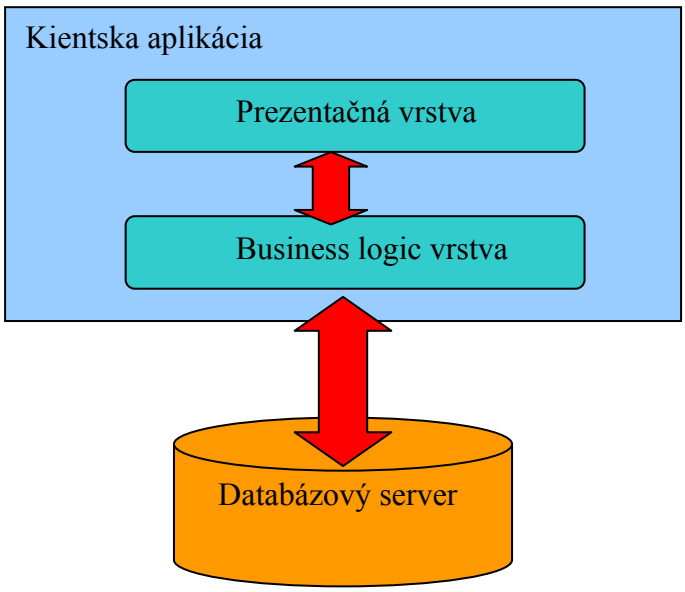
# 3. Návrh

## *3.1. Architektúra systému*

Z analýzy požiadaviek na funkcionalitu a používanie systému plynie jeho databázový charakter. Keďže musí byť zabezpečený prístup viacerých užívateľov k dátam, ponúka sa využitie databázového servera, ku ktorému sa budú jednotliví užívatelia pripájať a budú im poskytnuté žiadané dáta, ktoré budú centrálne uložené, čiže architektúra typu klasický klient – server.

Návrh teda predpokladá databázový server, ktorý bude prístupný na niektorom zo serverov podnikovej siete a klientskú aplikáciu, ktorá zaobstaráva komunikáciu so serverom, získanie žiadaných dát z databáze a ich prezentáciu užívateľovi.

Samotná klientská aplikácia však oddeľuje prezentačnú vrstvu, ktorú predstavuje užívateľské rozhranie systému, a takzvanú business logic vrstvu, ktorá sa stará o získanie dát z databázy a rôzne operácie nad nimi.

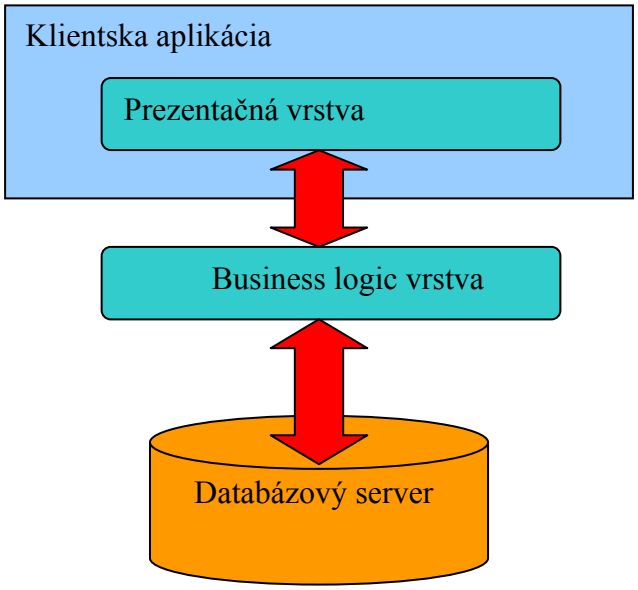



**Obrázok: Model architektúry systému**

Na tomto mieste treba podotknúť, že ďalším možným riešením je takzvaná multi-tier (viacvrstvová) architektúra, ktorá umožňuje úplnú nezávislosť dátovej, business logic a prezentačnej vrstvy. Business logic vrstvu tvorí typicky aplikačný server. Jednotlivé vrstvy pri takomto riešení môžu fungovať na rôznych platformách a môžu byť aj fyzicky umiestnené na rôznych strojoch.

Po dôkladnom zvážení však neboli identifikované žiadne zásadné prínosy pre prácu a údržbu daného systému, pre ktoré by bolo výhodnejšie zvoliť pre navrhovaný

systém túto architektúru. Nie je nutné platformové ani fyzické oddelenie jednotlivých častí, bola uprednostnená kompaktnosť a jednoduchosť klasickej klient server architektúry.




**Obrázok: Model multi tier architektúry**

Na implementáciu zvolenej architektúry bola vybraná platforma .NET, keďže má systém bežať pod Windows XP a .NET je nástroj priamo určený pre vývoj aplikácií pre Windows, a celá klientská aplikácia bude písaná v jazyku C#. Ako databázový stroj bol zvolený MS SQL Server 2005, hlavne pre dobrú spoluprácu s platformou .NET. Dôvody výberu relačnej databáze sú uvedené v nasledujúcej kapitole.

## *3.2. Návrh databázy*

### *Databázový model*

Ako už bolo spomenuté, pre uchovanie dát bude vytvorený databázový server. Je však dôležité rozhodnúť, akú databázu bude systém používať. Pre uvažovaný systém prichádzajú do úvahy tri možnosti.

1) Relačná databáza – v súčasnej dobe najrozšírenejšie riešenie s dobre definovanou metodikou návrhu. Dáta sú uložené v reláciách (tabuľkách).
2) Objektová databáza – základom databáze nie je tabuľka, ale objekt, ktorý ma vlastnosti a metódy, ktoré s nimi manipulujú. Záznamy sú jednotlivé inštancie objektu. Tento model lepšie vyhovuje návrhu zložitých systémov, pri ktorých sa modelovanie relačných databáz stáva neprehľadným, avšak nie je presne definovaná samotná metodika návrhu.

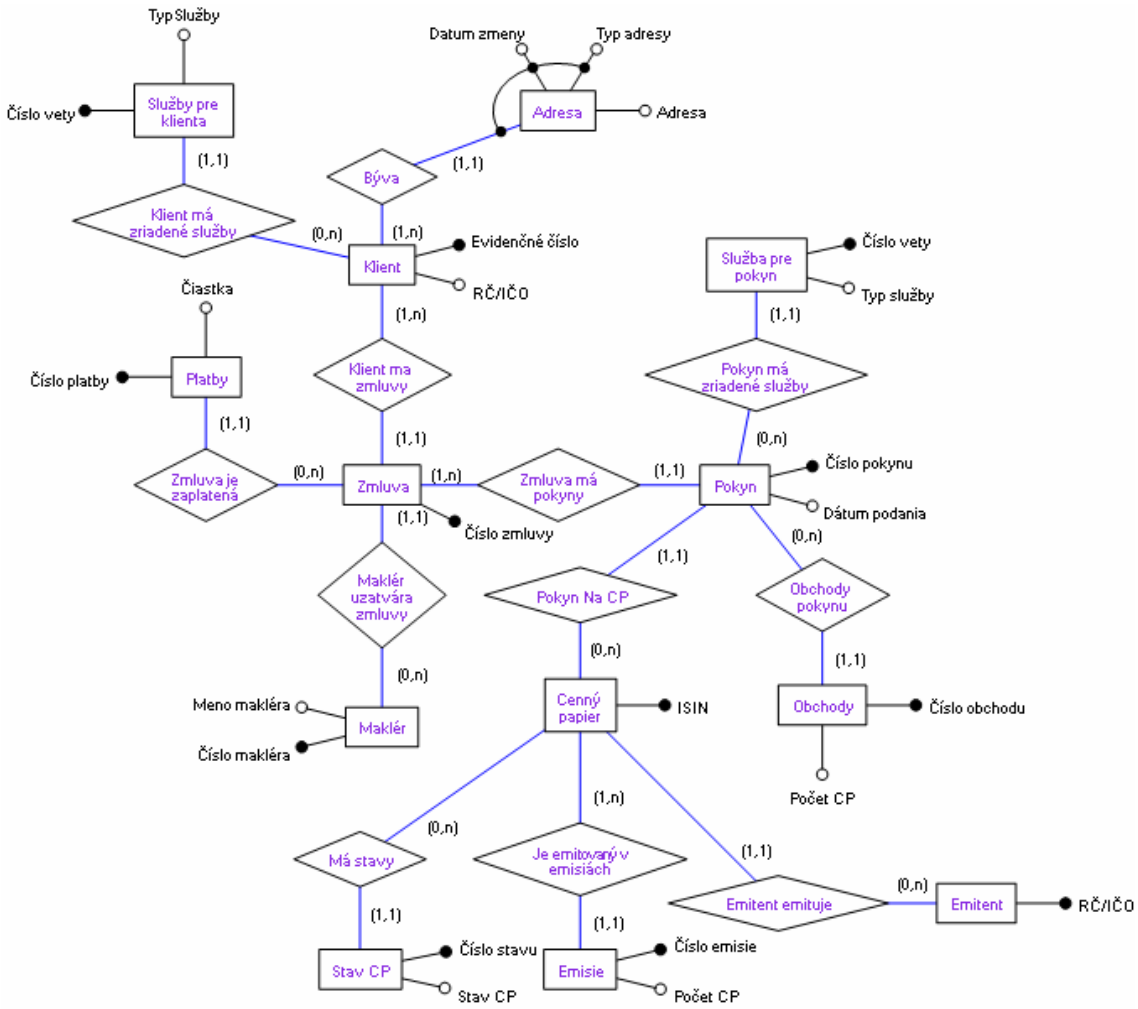
3) Objektovo – relačná databáza – snaha o prepojenie relačného a objektového modelu, keďže sú však tieto v priamom rozpore, je aj úspešnosť tohto spojenia rozporuplná

Z uvedených možností bola nakoniec zvolená relačná databáza, a to hlavne vďaka prepracovanej a relatívne jednoduchej metodike návrhu, ktorá je všeobecne známa. Objektová databáza je výhodná, pokiaľ by objekty boli držané po celý čas v pamäti (čo pri danom systéme nemožno zaručiť, dokonca to nie je pravdepodobné), avšak pri ich ukladaní na pevný disk pri serializácii by došlo k strate získaného výkonu. Celkovo sa názory na objektové a objektovo – relačné databázy rôznia, majú horlivých prívržencov aj odporcov.

Aj z tohto dôvodu bola zvolená osvedčená relačná databáza.

## *Návrh relačného modelu*

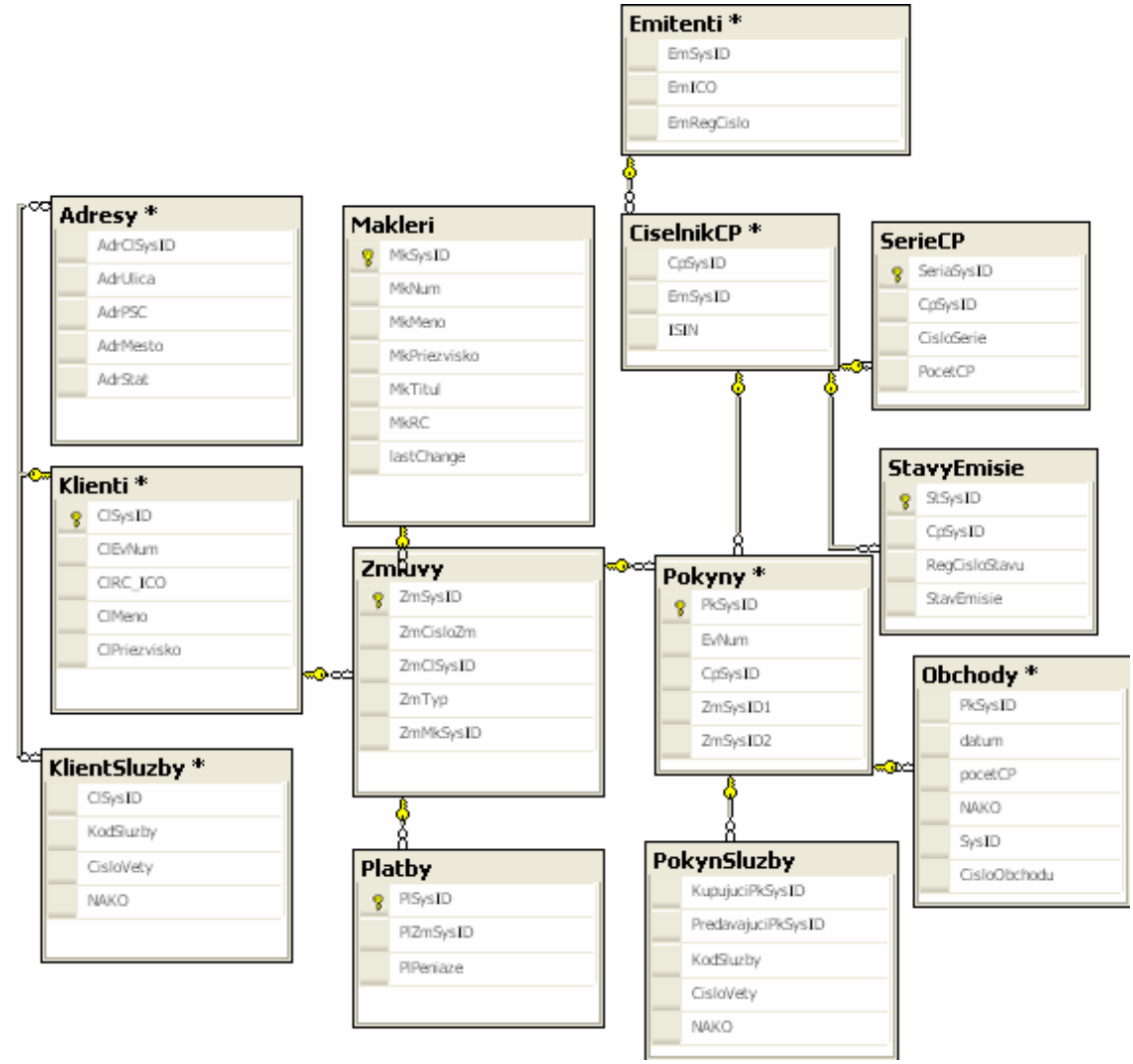
Na návrh relačného modelu databázy sa používajú dve techniky. Jednou je entitno – relačné (ER) modelovanie, pri ktorej sa vytvorí ER diagram zložený z entít, vzťahov a atribútov, ktorý sa následne prevedie na relačný model. Je to vlastne spôsob návrhu zhora nadol, najprv sa identifikujú entity a následne sa k nim priraďujú atribúty. Opačným procesom je normalizácia, pri ktorej sa primárne určia




**Obrázok: ER diagram**

atribúty, ktoré sa priradia do relačných tabuliek. Dáta v takto navrhnutej tabuľke však obsahujú redundantné informácie, čo môže spomaliť vkladanie alebo upravovanie záznamov, preto sa procese normalizácie upravuje model do normálových foriem, ktoré by tomu mali zabrániť. Zároveň však môže dôjsť k spomaleniu niektorých operácií, keďže dáta sa musia hľadať vo viacerých tabuľkách. Je preto dôležité nájsť rovnováhu medzi redundanciou a dekompozíciou do rôznych tabuliek. V praxi je pomerne často používaná kombinácia oboch techník.

Pre potreby systému pre obchodníka s cennými papiermi bol vytvorený ER – diagram, ktorý slúži pre predstavu o existencii a vzťahu jednotlivých entít. Z vyššie uvedených požiadaviek tieto pomerne jasne plynú. Jediné entity, ktoré neboli explicitne spomenuté sú adresa, stavy cenného papiera a emisie cenného papiera. Do diagramu však boli zahrnuté, pretože pri analýze požiadaviek na evidenciu klienta vyšlo najavo, že je potrebné uchovávať históriu zmien jeho adresy a cenný papier je vydávaný vo viacerých emisiách a môže mat definovaný určitý počet stavov, pričom je opäť potrebné všetky uchovávať. Vytvorený model bol




**Obrázok: Relačná schéma databázy**

prevedený na relácie a tieto boli upravené tak, aby odzrkadľovali aj funkčné požiadavky na systém. Na obrázku sú pre prehľadnosť zobrazené len základné entity.

Po prevedení ER – diagramu na relácie vznikne 13 tabuliek, ktoré sú poprepájanécudzími kľúčmi. Pre každú tabuľku je definovaný umelý primárny kľúč typu SQL bigint s identity(1,1), čo znamená., že pre každý pridaná záznam sa automaticky vloží hodnota o jedna vyššia ako pri predchádzajúcom vložení.

Pre potreby zaistenia súbežnej práce viacerých užívateľov majú tabuľky stĺpec typu timestamp, ktorý sa automaticky zmení pri každej zmene záznamu. Táto vlastnoť sa neskôr využije pri aktualizácii záznamu.

Pre potreby vykonania jednej služby pre dva pokyny sú v tabuľke PokynSluzby dva cudzie kľúče do tabuľky pokynov – jeden pre kupca, druhý pre predajcu . Obe hodnoty sa momentálne napĺňajú iba pri obchode NPO, v ostatných prípadoch je vyplnený iba príslušný stĺpec tabuľky.

Stav služieb a obchodov pre pokyny a klientov sa typicky zisťuje vyhľadaním v príslušnej tabuľke, toto však neplatí pre službu E01. Táto je natoľko typická pre klienta, že jej stav je uvedený okrem tabuľky KlientSlužby aj v tabuľke Klienti.

Okrem tabuliek, ktoré sú zobrazené na priloženom obrázku, obsahuje databáza aj množstvo ďalších pomocných tabuliek – číselníkov, v ktorých sú definované hodnoty niektorých atribútov hlavných entít. Okrem týchto tabuliek ešte databáza uchováva tabuľku užívateľov systému pre potreby autentifikácie a autorizácie a tabuľku s údajmi o obchodníkovi (jeho číslom IČO, číslom na CDCP a pod.). Všetkých tabuliek v databáze je dohromady 37.

### *Manipulácia s dátami v databáze*

Pri manipulácii s dátami je potrebné dbať na bezpečnosť a efektivitu, preto interakcia s databázou prebieha primárne prostredníctvom uložených procedúr servera, ktoré poskytujú aj zlepšenie v rýchlosti vykonania aj ochranu pred základnými útokmi typu Sql injection. Pre každú tabuľku je preto vytvorená sada funkcií, ktoré umožňujú pridať, aktualizovať a zmazať záznam (add, update, delete a get, ktorá vyberá údaje z databázy aj s príslušnými spojeniami viacerých tabuliek).

Update procedúry zohrávajú dôležitú úlohu pri zabezpečení súbežnej práce užívateľov a na určenie, či dáta, s ktorými užívateľ pracoval a chce ich uložiť do databáze, medzitým neboli pozmené. Dosiahne sa to pridaním položky do klauzule where príkazu update. Ten potom vyzerá nasledovne:

```
UPDATE {názov tabuľky} SET {nastavenie hodnôt stĺpcov}
WHERE {podmienka vyhľadania} AND lastChange = @lastChange
```

Kde lastChange je stĺpec tabuľky s hodnotou timestamp a @lastChange premenná s pôvndou hodnotou stĺpca.
Ďalej budú popísané niektoré zložitejšie procedúry, ktoré vykonávajú niektoré špeciálne operácie.

**add/updateKlient** – táto dvojica procedúr sa síce na prvý pohľad nelíši od bežných add/update procedúr, avšak pri evidovaní klientov je potrebné urobiť určité kontroly vkladaných hodnôt, keďže vkladané (alebo aktualizované) dáta sa už v databáze môžu nachádzať. Dalo by sa to dosiahnuť aj integritnými obmedzeniami na úrovni tabuľky (cez príznak unique). Unikátnosť niektorých atribútov (ako napríklad rodné číslo) sa síce predpokladá, ale v realite môže byť situácia trochu iná. Preto táto procedúra skontroluje, či už sa záznam s podobnými hodnotami v databáze nenachádza a pridá ho iba vtedy, ak je nastavený príznak override.

**procedúry vysporiadanie_A_Kupa/Predaj/NPO** – sú využívané pri načítaní A súboru vysporiadania a priradení klientov k obchodom. Znížia počet cenných papierov, ktoré je ešte potrebné zobchodovať a predbežne zmenia aj peňažný zostatok na zmluvách. Zároveň pridajú záznam o novom obchode do príslušnej tabuľky a vrátia údaje o obchode a klientovi pre potreby vytvorenia súboru B.

**procedúry vysporiadanie_A_Makler** – sú volané, keď je potrebné rozdeliť určité množstvo cenných papierov (predaných alebo kúpených) medzi jednotlivých klientov. Na tento účel používajú cursor, ktorý je naplnený vhodnými pokynmi (sú zaplatené, majú vytvorené príslušné služby), ktoré sú zoradené podľa dátumu podania. Jednotlivé záznamy sú potom v cykle prechádzané a sú k nim vytvárané obchody až kým počet cenných papierov, ktoré bolo potrebné rozdeliť, nie je nulový. Procedúra opäť vracia podklady pre vytvorenie B súboru.

**procedúra vysporiadanie_C** – využívaná pri načítaní posledného súboru vysporiadania. Vypĺňa návratový kód obchodu, v prípade, že je iný ako 100, vracia pokyn aj zmluvu do pôvodného stavu pred načtaním súboru A (čiže zmení počet cenných papierov, ktoré je ešte potrebné zobchodovať a pripíše/odpíše finančné prostriedky na zmluve).

**procedúra vysporiadajListCP** – pri párovaní pokynov na obchod s listinnými CP vytvor príslušnú dvojicu obchodov, aktualizuje počty CP, ktoré treba zobchodovať a v prípade potreby vykoná aj úpravu finančných zostatkov na zmluvách.

**procedúra fillCiselnikCP** – slúži na napĺňanie číselníka cenných papierov. Skontroluje, či sa emitent už nenachádza v evidencii a prípadne do evidencie pridá nového emitenta. Následne pridá alebo aktualizuje záznam o cennom papieri (v závislosti na tom, či sa v evidencii už nachádza CP s rovnakým číslom ISIN).

**procedúra createVyuctovanie** – vyúčtuje všetky vhodné (to znamená tie, ktorých všetky pokyny sú ukončené) zmluvy, nastaví im príznak vyúčtovaná a dátum vyúčtovania. Vráti zoznam vyúčtovaných zmlúv.

**procedúra getObchodyVyuctovanie** – používa sa pri tlači vyúčtovania, vyberie z databázy všetky obchody ku všetkým pokynom danej zmluvy.

**procedúra getDennik** – vytvorí denník obchodníka v rozsahu špecifikovanom hraničnými dátumami. Vyberie všetky realizované obchody a pripojí k nim údaje o pokyne, ku ktorému sa vzťahujú. Ak pokyn nebol alebo ešte nie je realizovaný, vráti iba údaje o pokyne.

## *3.3. Návrh klientskej aplikácie*

Ako bolo uvedené v časti o architektúre systému, okrem databázového servera bude systém zahŕňať aj klientsku aplikáciu, ktorá pozostáva z dvoch vrstiev – business logic a GUI. Návrh a implementácia oboch vrstiev sa bude riadiť zásadami objektovo orientovaného programovania.

### *Filozoia vrstvy business logic*

Táto vrstva zaobstaráva získavanie dát z databáze a rôzne operácie nad nimi. Slúži aj na oddelenie užívateľského rozhrania od dát. Pri jej návrhu je dôležité brať do úvahy možnosti, ktoré ponúka .NET. Na pripojenie k databáze, získanie dát z nej a dokonca aj na ich uchovanie je preto možné použiť komponenty ADO.NET, v tomto prípade (keďže aplikácia je pripojená k SQL Serveru) SqlDataAdapter, SqlDataReader, DataSet.

V ADO.NET je presadzovaný takzvaný disconnected (odpojený) model získavania dát a práce s nimi. Je založený na predpoklade, že udržiavanie trvalého spojenia je obojstranne náročné a obmedzujúce pre obe strany, klienta aj server, preto sa klient pripojí k databáze, získa dáta, ktoré potrebuje a hneď sa odpojí. Keď užívateľ skončí prácu s dátami, klient otvorí pripojenie k databáze a aktualizuje dáta v nej. Otváranie stále nového pripojenia je síce pomerne náročná operácia, ale tento problém sa snaží eliminovať takzvaný connection polling.

Tento odpojený model však vo viacužívateľskom prostredí predpokladá využitie optimistic concurrency control (OCC) – to znamená, že na dátach, ktoré užívateľ získal, nie sú žiadne zámky, neobmedzený prístup k nim majú všetci ostatní užívatelia, predpokladá sa teda, že väčšina databázových operácií medzi sebou nekoliduje. Ak sa tak stane, systém musí tento konflikt detekovať. Takýto spôsob práce dobre funguje ak sa s dátami nepracuje príliš intenzívne, čo je možné predpokladať aj pri systéme pre obchodníka s cennými papiermi. Zo skúseností z prevádzky podobných systémov vyplýva, že jednotliví užívatelia väčšinou pracujú s rôznymi evidenciami, prípadne tendencia aktualizovať rovnaké záznamy v databáze je malá. Preto tento model je možné použiť v uvažovanom projekte.

Z uvedeného priamo vyplýva požiadavka na systém určiť, či dáta, s ktorými užívateľ pracoval, neboli zmenené iným užívateľom. Tu sa úplne vyjasňuje použitie automaticky aktualizovaného stĺpca typu timestamp v tabuľkách a následne aj v príkaze UPDATE (nepriamo z toho vyplýva, že objekty vrstvy business logic musia uchovávať túto hodnotu pre neskoršiu aktualizáciu).

So znalosťou vyššie uvedených skutočností je možné pristúpiť k vlastnému návrhu business logic vrstvy. Z požiadaviek plynie, že systém bude musieť pracovať aj s tabuľkovými dátami (rôzne výsledky vyhľadávaní, zoznamy exportovaných služieb, zobrazenie číselníkov pre potreby ich napĺňania a editácie) ako aj s jednotlivými entitami. ADO.NET ponúka pre reprezentáciu tabuliek dát v pamäti triedu DataSet, prípadne DataTable. Preto pre dáta v podobe skupín záznamov, ako sú už spomenuté výsledky vyhľadávaní, číselníky a podobné dáta, s ktorými nie sú vykonávané žiadne zložitejšie operácie, bude použitá práve táto trieda. Pre zložitejšie entity budú vytvorené samostatné objekty, ktoré budú zabezpečovať všetky potrebné operácie.

Triedy DataSet a DataTable je možné vytvárať dvomi spôsobmi. Pri prvom sa schéma týchto tried (definície tabuliek a ich stĺpcov) vytvorí pri ich napĺňaní dátami z databázy. Pri druhom spôsobe sa táto schéma vytvorí vopred (či už písaním kódu ale použitím na to určeného editora) a následne sú do nej vkladané dáta, pričom sa kontroluje ich správnosť. Tento prístup sa označuje ako typed-DataSet.

### *Špecifické business objekty*

Pre hlavné entity budú teda vytvorené špeciálne objekty, ktoré ich budú reprezentovať v systéme. Je pomerne jasné, že tieto objekty si budú navzájom pomerne podobné, preto je v duchu objektovo orientovaného programovania vhodné definovať spoločného predka, ktorý bude definovať štruktúru všetkých podobných tried a pridanie novej entity do systému bude vlastne vyžadovať iba oddedenie novej triedy od tohto objektu, a zároveň bude implementovať metódy, ktoré potom budú použité v potomkoch (tí si samozrejme môžu definovať vlastné implementácie). Bude preto definovaná trieda

```
abstract public class BusinessClass : INotifyPropertyChanged
```

ktorá deklarovaná ako abstraktná, čiže nebude možné vytvárať jej inštancie, keďže je len spoločným predkom pre odvodené triedy. Trieda implementuje rozhranie (interface) INotifyPropertyChanged, ktorý definuje metódy potrebné pre správne prepojenie s GUI prostredníctvom konceptu DataBindingu (ten bude podrobnejšie rozobratý pri návrhu samotného GUI) . Budú v nej ďalej definované základné dátové položky

```
protected string _baseAddProcedure;
protected string _baseUpdateProcedure;
protected string _baseGetProcedure;
protected byte[] _lastChange;

protected long _ID;
```

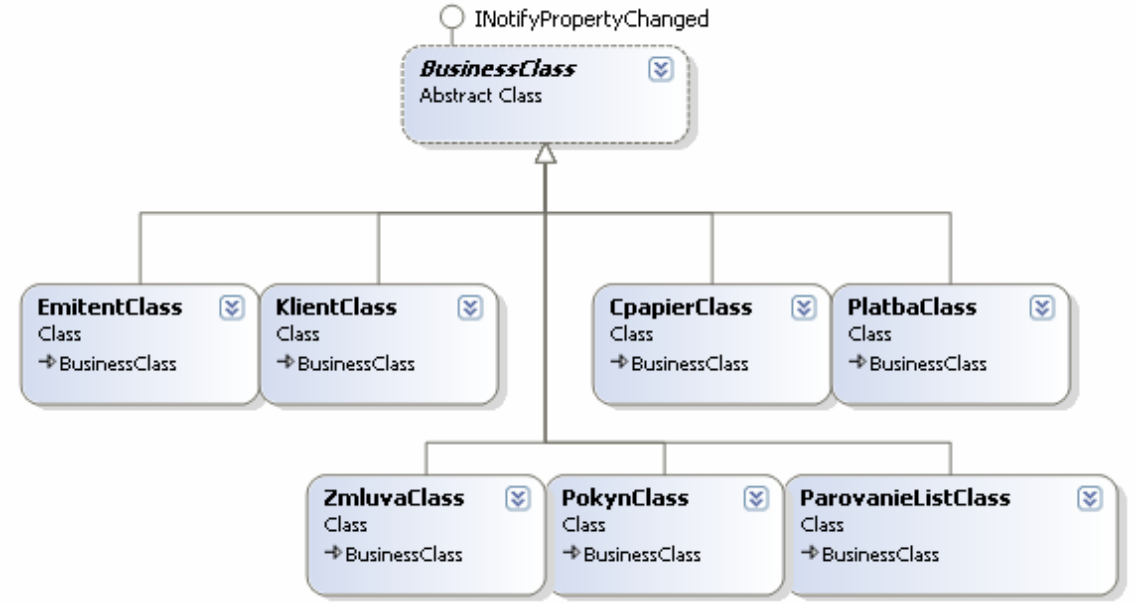
kde _baseAddProcedure je názov uloženej procedúry SQL Servera, ktorá pridá objekt do databázy, _baseUpdateProcedure je meno procedúry pre aktualizáciu a _baseGetProcedure zas pre získanie dát z databázy. Premenná _lastChange uchováva hodnotu stĺpca timestamp daného záznamu pre potrebu OCC. Položka _ID reprezentuje databázový primárny kľúč (ako bolo spomenuté pri návrhu databázy, je to umelá hodnota). K nej je definovaná vlastnosť

```
public long ID
```

pre potrebu prístupu k privátnej položke triedy. V triede sú ďalej definované tri základné metódy

```
public virtual bool update()
public virtual void get(long zmID)
public virtual bool add()
```

ktoré ako už názvy hovoria, vykonávajú pridanie, aktualizáciu a získanie dát z databáze. V tejto triede sú definované aj ďalšie metódy, ich popis však patrí skôr




**Obrázok: Hierarchia tried BusinessClass**

do programátorskej dokumentácie. Je možné si všimnúť, že položky triedy nie sú definované ako private, ale ako protected, aby boli prístupné z potomkov.
Od tejto základnej triedy sú potom odvodené už konkrétne business triedy, ich hierarchiu možno vidieť na obrázku.
Z názvov tried je jasné, aké entity reprezentujú, možno až na ParovanieListClass – tá slúži na párovanie pokynov na predaj listinných CP. Konkrétny popis jednotlivých tried je opäť v programátorskej dokumentácii.

Business logic vrstva obsahuje aj ďalšie triedy, ktoré nereprezentujú žiadne entity ani služby, ale majú skôr podporý charakter.

**Trieda Cache** - slúži ako dátový zdroj pre GUI ktoré má prezentovať výsledky vyhľadávania. Keďže záznamov, ktoré vyhovujú vyhľadávacím kritériám môže byť veľké množstvo, táto trieda vykonáva stránkovanie výsledkov. Načítavajú sa vždy len tie dáta, ktoré je potrebné zobraziť, čo sa prejavuje v rýchlejšej odozve (užívateľ nemusí čakať, kým sa načítajú všetky dáta) a aj menšej pamäťovej náročnosti.

**Trieda CiselnikCPFiller** – používa sa na načítanie číselníka cenných papierov z xml súboru do databázy. Na načítanie XML dát do relačnej databázy by bolo možné použiť balík SQLXML a jeho triedu SQLXMLBulkLoad.SQLXMLBulkload.4.0, avšak tá nie celkom vyhovuje požiadavkám, keďže súbor je potrebné mapovať do viacerých tabuliek a popritom ešte vyhľadávať cudzie kľúče do číselníkov (a kontrolovať, či nenadobúdajú nedefinovanú hodnotu) podľa hodnôt zo súboru. Preto bola vytvorená táto trieda, ktorá sa stará o všetky uvedené činnosti.

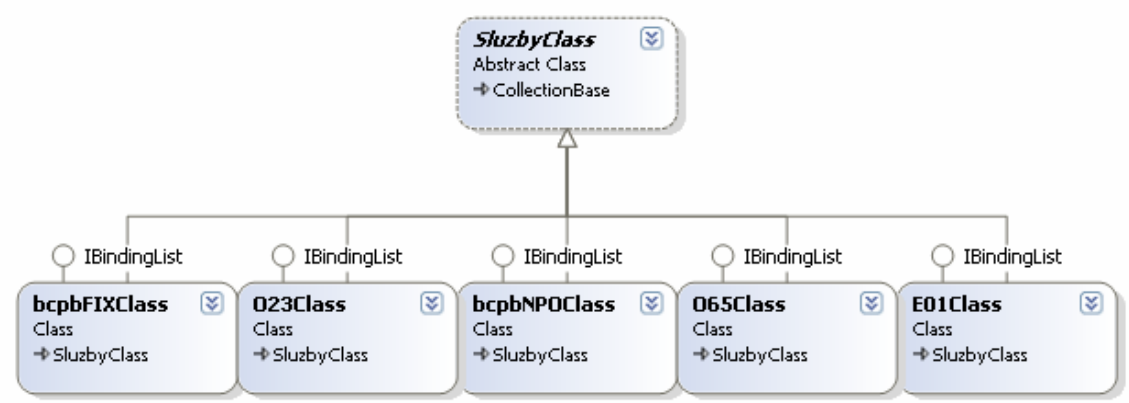
**Trieda DataProvider** – ponúka funkcie na načítavanie tabuľkových dát (typicky číselníkov) pre potreby GUI.

**Trieda DataProvider** – obsahuje ako statickú položku objekt na pripojenie k databáze a funkcie na odstránenie SQL injection útokov z príkazov

**Trieda User** – reprezentuje užívateľa systému, obsahuje metódy na jeho vytvorenie, autentifikáciu a autorizáciu.

**Trieda Security** – obsahuje funkcie na šifrovanie a hashovanie.

**Trieda SluzbaClass** a triedy od nej odvodené zabezpečujú vykonávanie služieb E01, O23, O65, obchodov NPO a anonymných obchodov. Obsahujú zoznamy údajov o entitách, pre ktorú sa má služba vykonať v podobe potomkov triedy **VetaClass,** ktorí majú definované metódy na svoj export do súborov a zápis




**Obrázok: Hierarchia tried SluzbyClass**

o svojom stave do databázy. Triedy sú navrhnuté tak, aby bolo možné zobraziť ich obsah a prípadne niektoré služby ešte pred ich exportom zrušiť. Potomkovia triedy SluzbaClass taktiež umožňujú importovať súbory s odpoveďami.

Podobne vyzerá aj hierarchia tried VetaClass.

**Triedy VysporiadanieAClass a VysporiadanieCClass** slúžia na načítanie súborov vysporiadania a ich spracovanie v databáze.

**Trieda VyuctovanieClass** uspokojuje potreby vyúčtovania. Obsahuje statickú funkciu, ktorá vyúčtuje všetky vhodné zmluvy v evidencii (prostredníctvom uloženej procedúry createVyuctovanie). Samotné inštancie tejto triedy slúžia ako zdroj dát pre tlač vyúčtovania prostredníctvom **VyuctovanieCrystalReport** pre konkrétnu zmluvu. Obsahuje inštanciu typed-datasetu **VyuctovanieDataSet**, ktorý obsahuje zoznam všetkých platieb a obchodov k zmluve.

**typed-Dataset DennikDataSet** – uchováva informácie o pokynoch a obchodoch pre potreby vytvorenia správy denník obchodníka prostredníctvom **DennikCrystalReport**. DataSet je napĺňaný uloženou pocedúrou getDennik.

### *Prenzentačná vrstva – GUI*

Prezentačnú vrstvu v tomto projekte tvoria okná Windows Forms. GUI bolo navrhnuté ako jedno hlavné MDI okno s menu, v ktorom sa zobrazujú dcérske okná. Konkrétnejš popis aj s obrázkami je možné nájsť v ďalšej kapitole.

Previazanie medzi business logic vrstvou a zobrazovacími komponentmi zabezpečuje koncept DataBinding, v ktorom sa určitá vlastnosť objektu na formulári zviaže s vlastnosťou príslušného business objektu a zmena vlastnosti v jednej vrstve sa odrazí v zmene príslušnej vlastnosti v druhej vrstve.

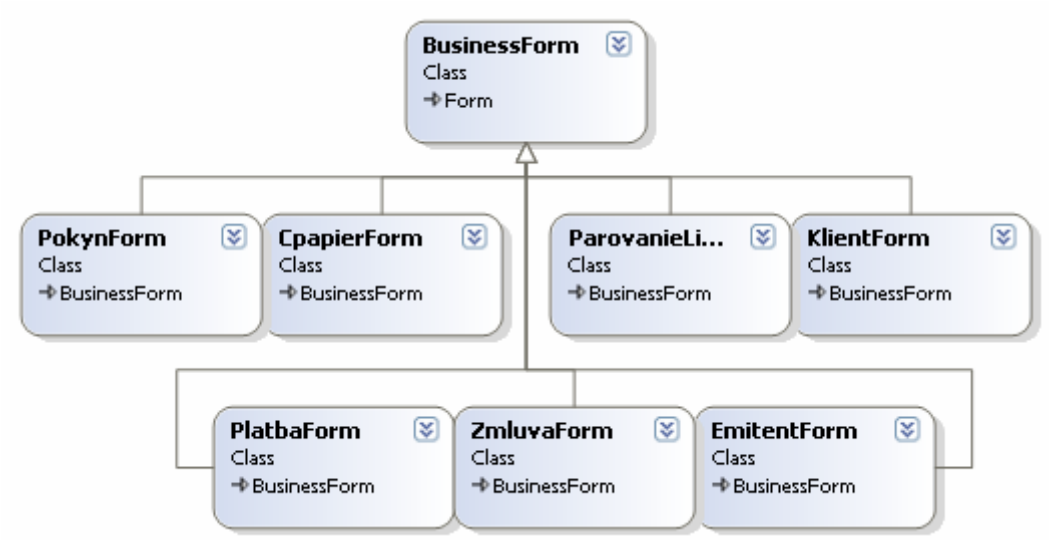
Podobne ako pri business objektoch je výhodné definovať určité bázové triedy, od ktorých budú oddedené konkrétne formuláre.
Hlavnými formulármi sú tie, ktoré zabezpečujú zobrazovanie hlavných business objektov (čiže hlavných entít). Pre ne slúži ako predok trieda

```
abstract public class BusinessForm : Form
```

ktorá definuje niektoré dátové položky ako napríklad

```
public BusinessClass businessObject
```

čo je odkaz na business objekt, ktorý uchováva dáta zobrazované formulárom. Trieda ďalej definuje niektoré funkcie, ktoré nastavujú vlastnosti kontrolných prvkov formuláru a pod. Podrobnosti je možné opäť nájsť v programátorskej dokumentácii.



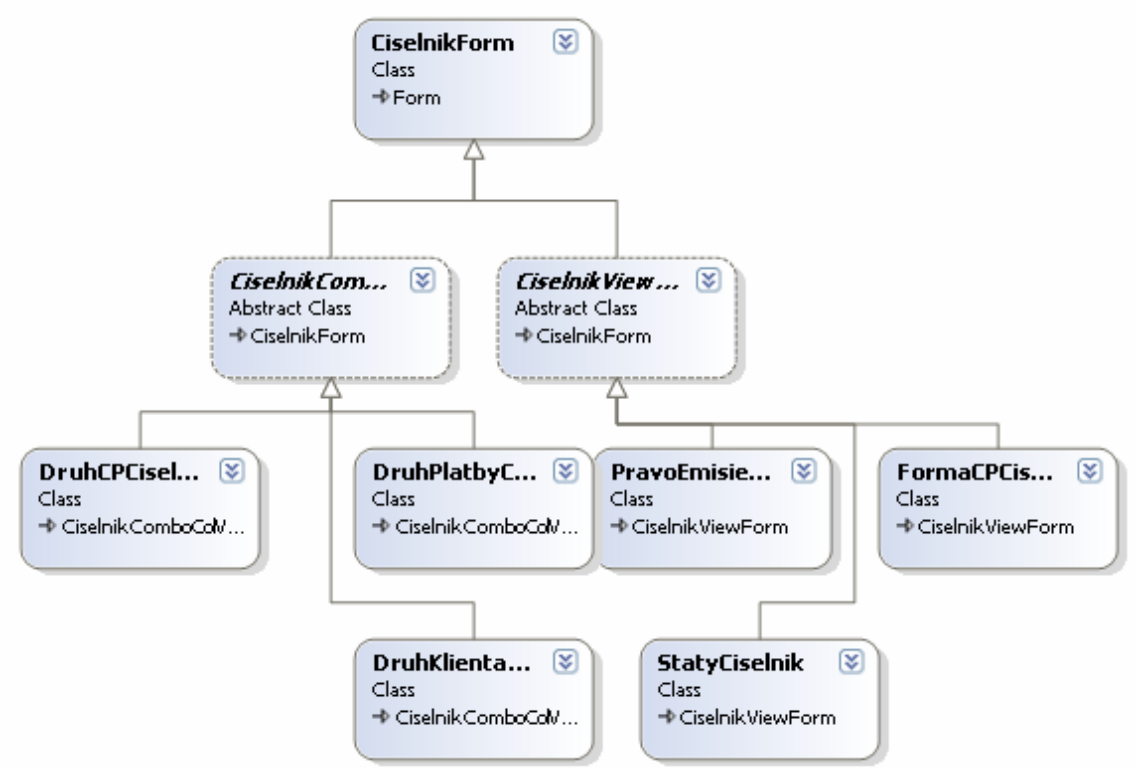

**Obrázok: Hierarchia BusinessForm**

Okrem formulárov, ktoré reprezentujú navrhnuté business objekty sa v aplikácii nachádzajú aj formuláre, ktoré zobrazujú tabuľkové dáta, na čo využívajú komponentu DataGridView. Takto sú zobrazované naprklad niektoré číselníky. Tiež majú jedného spoločného predka

```
abstract public partial class CiselnikForm : Form
```

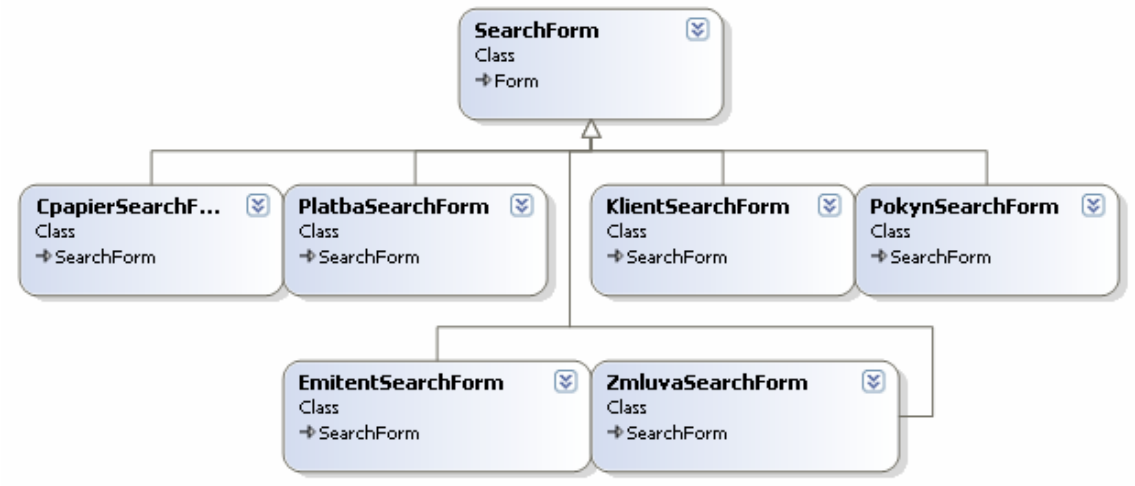
Od neho sú odvodené ďalšie dve abstraktné triedy, jedna pre číselníky ktoré zobrazujú čisto textové dáta a druhá pre číselníky, ktorých DataGridView obsahuje stĺpec typu ComboBox. Na ďalších obrázkoch je vidieť hierarchiu odvodenú od

tried BusinessForm a CiselnikForm, pre veľký počet odvodených tried je ich zobrazených len niekoľko.




**Obrázok: Hierarchia CiselnikForm**

Pre potreby vyhľadávania je navrhnutá trieda **SearchForm**, ktorá definuje funkcie používané pri vyhodnocovaní podmienok vyhľadávania. Jej potomkami sú potom formuláre na zadávanie vyhľadávacích podmienok. Po zostavení vyhľadávacej podmienky je táto predaná triede **SearchResultForm** ktorá slúži na zobrazenie výsledkov vyhľadávania, na čo opäť používa komponentu DataGridView. Pri jej vytvorení je potrebné predať jej štruktúru, v ktorej sú uvedené názvy jednotlivých stľpcov zobrazovaných dát, a takisto vytvoriť delegáta (pointer na funkciu), ktorá




**Obrázok: Hierarchia tried SearchForm**

sa bude starať o zobrazenie správneho formulára s detailom vyžiadanej položky ak užívateľ dvakrát klikne na vybraný riadok komponenty DataGridView. Ako zdroj dát bude slúžiť už skôr spomenutá trieda Cache, ktorá vykonáva stránkovanie získaných dát.

### *Zabezpečenie*

Systém pre obchodníka s cennými papiermi musí poskytovať aj istú mieru ochrany a bezpečnosti.

**Zabezpečenie connection stringu** – údaje slúžiace na pripojenie k databáze je možné zneužiť, preto je nutné ich chrániť. Na tento účel je celý reťazec zašifrovaný šifrou Rijndael, ktorá je k dispozícii vo .NET frameworku. Zašifrovaný reťazec je potom uložený do registrov Windows. Nie je to síce neprekonateľná prekážka, avšak je to o niečo bezpečnejšie ako uloženie hesla v plain texte.

**Systém práv** – ako bolo povedané v požiadavkách na systém, je potrebné riadiť prístup jednotlivých užívateľov k dátam. Preto bol navrhnutý systém práv, kde ku každej z hlavných entít v systéme existuje sada práv – pridávať, prezerať, aktualizovať a odstraňovať a tieto práva je možné ľubovoľne rozdeľovať medzi užívateľov. Touto úlohou je poverený jeden administrátor, ktorý môže vytvárať nových užívateľov a takisto spravuje databázu (napĺňa jednotlivé číselníky). Na implementáciu systému práv slúži tabuľka užívateľov a trieda User, ktorá uchováva všetky práva užívateľa.

**Autentifikácia** – aj kvôli dodržiavaniu systému práv je potrebné identifikovať užívateľa, ktorý so systémom pracuje a zamedziť prístupu nežiaducim osobám. Preto sa pred začiatkom práce s aplikáciou musí každý užívateľ prihlásiť zadaním prihlasovacieho mena a hesla (heslo je zadané pri vytváraní užívateľa administrátorom, avšak užívateľ si ho môže ľubovoľne meniť. Heslo je uchovávané v databáze ako zahashovaná hodnota pôvodného hesla s pridaním náhodného reťazca, takzvaného „saltu“. Heslo sa tak stáva odolnejším voči slovníkovým útokom.

# 4. Implementácia

V tejto časti sú uvedené problémy, ktoré sa vyskytli pri implementácii a charakteristika finálneho systému spolu so stručným popisom jeho funkcií a používania.

## *4.1. Problémy pri implementácii*

Pri implementácii sa nevyskytli žiadne závažnejšie problémy. Okolo .NETu je sústredená veľká skupina vývojárov a preto boli všetky zažehnané už v počiatkoch. Za zmienku stoja azda iba dva.

### *Chyby v komponente DataGridView*

Pri príprave na návrh projektu a získavaní znalostí o vývoji aplikácii pre .NET boli nájdené v rôznych zdrojoch pri komponente DataGrid prívlastky ako „buggy“, čiže chybová. Tá je síce nahradená novým prvkom DataGridView, i tak však boli objavené dve chyby.

**Nevyžiadané pridanie prázdneho riadku** – za istých okolností a pri istej kombinácii operácií s riadkami komponenty táto samovoľne zobrazí jeden prázdny riadok. Zatiaľ nie je známa príčina tohto problému ani jeho uspokojivé riešnie. Táto chyba sa naštastie prejaví iba zriedka a nemá vplyv na správny chod aplikácie.

**Problém s DataGridViewComboBoxColumn** – pri komponente DataGrid mnohým vývojárom chýbala jednoduchá možnosť pridať stĺpec, ktorý by obsahoval ComboBox. Preto DataGridView môže obsahovať stĺpec DataGridViewComboBoxColumn, avšak ani ten sa nevyhol chybám. V prípade, že hodnoty tohto ComboBoxu je potrebné naplniť hodnotami z tabuľky z databázy, a previazať hodnotu cudzieho kľúča v tabuľke, ktorá je zobrazená v DataGridView so stĺpcom tabuľky, ktorá napĺňa ComboBox a typy týchto stĺpcov sú rôzne ako string, dochádza k problémom. Pri použití v aplikácii je našťastie možné previesť hodnotu cudzieho kľúča (int) na reťazec a tak je chyba eliminovaná.

**ComboBox schopný zobraziť viacero stĺpcov** – takzvaný multicolumn combobox. Nachádza využitie pri nastavovaní vlastností entity, ktoré sú uchovávané v číselníkoch prostredníctvom cudzieho kľúča, často je totiž potrebné zobraziť nielen samotný cudzí kľúč ale aj popis, ktorý je uvedený tabuľke, ktorá definuje číselník. Toto ešte umožnuje klasický ComboBox, kde sa dá nastaviť zobrazovaný stĺpec a stĺpec, ktorý bude uchovávať hodnotu cudzieho kľúča. Ak je však potrebné zobraziť viac stĺpcov, nie je už možné použiť klasický ComboBox. Z internetových vývojárskych fór je zrejmé, že podobný komponent by našiel využitie vo veľkom množstve aplikácií, je preto zarážajúce, že Windows Forms nič podobné neobsahujú.

Preto bola pre potreby aplikácie vybratý komponent ColumnComboBox, ktorá je voľne dostupná na adrese

http://www.codeproject.com/cs/combobox/MultiColComboSugtionBox.asp.

Nie je síce ideálna, avšak po drobných úpravách v oblasti DataBindingu spĺňa všetky požiadavky, ktoré na ňu aplikácia kladie.

Samostatnou kapitolou by bolo použitie multicolumn comboboxu v komponente DataGridView, avšak po prvých pokusoch implementovať niečo podobné bola táto možnosť zavrhnutá, keďže úžitok by bol menší ako vynaložené úsilie.

## *4.2. Popis aplikácie a jej ovládania*

Základné požiadavky, ktoré musí spĺňať prostredie, v ktorom má systém pracovať sú Windows XP s nainštalovaným frameworkom .NET 2.0. Ďalej musí byť k dispozícii databázový server MS SQL Server, ktorý musí byť dostupný.

### *Vytvorenie databáze*

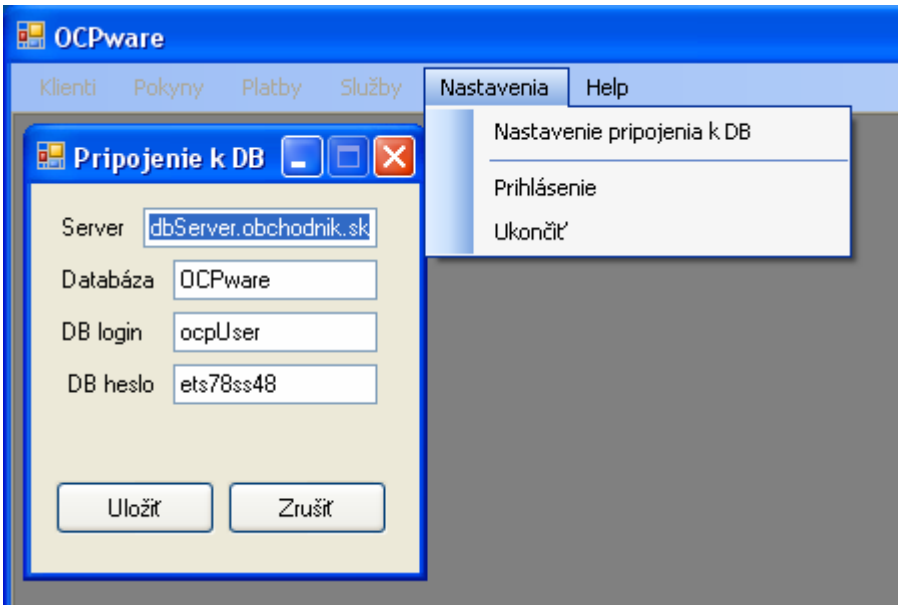
Spolu s SQL Serverom sa odporúča nainštalovať aj SQL Server Management Studio, prostredníctvom ktorého potom bude prebiehať vytvorenie databázy. Pri inštalácii je dôležité nastaviť spôsob autentifikácie na SQL Server Authentication (toto nastavenie je možné zmeniť aj neskôr prostredníctvom Management Studia cez menu „Properties“ daného servera na stránke „Security“). Potom je potrebné vytvoriť databázu a v nej spustiť skripty s definíciami tabuliek, uložených procedúr a iniciačným naplnením databáze (hlavne číselníkov). Taktiež sa odporúča pre potreby aplikácie vytvoriť samostatný databázový login. Najprv je nutné pomocou príkazov

```
CREATE ROLE db_executor
GRANT EXECUTE TO db_executor
```

vytvoriť rolu, ktorá umožňuje spúšťanie uložených procedúr a následne vytvoriť nový login cez položku servera Security – Logins – New Login, nastaviť SQL Server authentication a na strane User Mapping vytvoriť mapovanie do príslušnej databázy s rolami db_datareader a db_executor. Server je teraz pripravený odpovedať na požiadavky klientskych aplikácií.

### *Spustenie klientskej aplikácie*

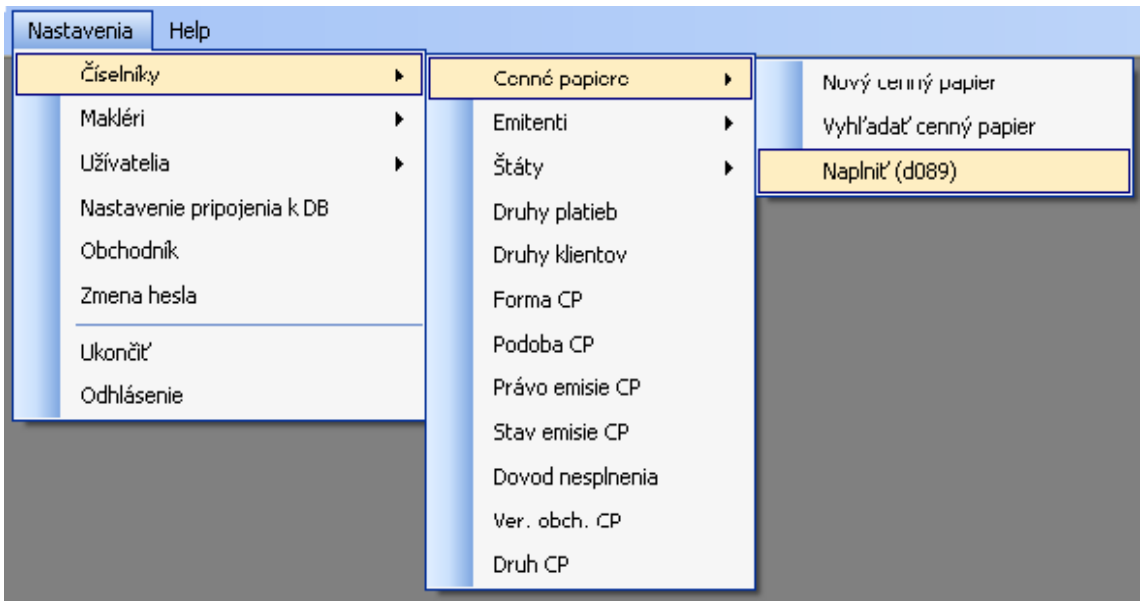
Pri prvom spustení je užívateľ upozornený, aby nastavil parametre databázového pripojenia, keďže bez neho je aplikácia prakticky nefunkčná. Potom sa zobrazí hlavné okno aplikácie, v prípade, že boli zadané zlé parametre databázového pripojenia, môže nabiehanie aplikácie trvať dlhšie, keďže pri štarte sú z databázy načítavané niektoré údaje (podobá situácia môže nastať aj pri výpadku spojenia). V jeho hornej časti sa nachádza menu, v ktorom je možné nastaviť parametre pripojenia a následne sa prihlásiť do aplikácie.
Pri inicializácii databázy bol vytvorený užívateľský účet pre administrátora systému. Ten sa do aplikácie prihlasuje s loginom „Admin“, pri prvom pihlásení je nutné, aby mu bolo nastavené nové heslo.




**Obrázok: Nastavenie databázy**

## *Práca administrátora*

Administrátorovou úlohou je správa užívateľov a správa pomocných údajov v databáze – číselníkov. Túto prácu mu umožňuje vykonávať menu „Nastavenia“.

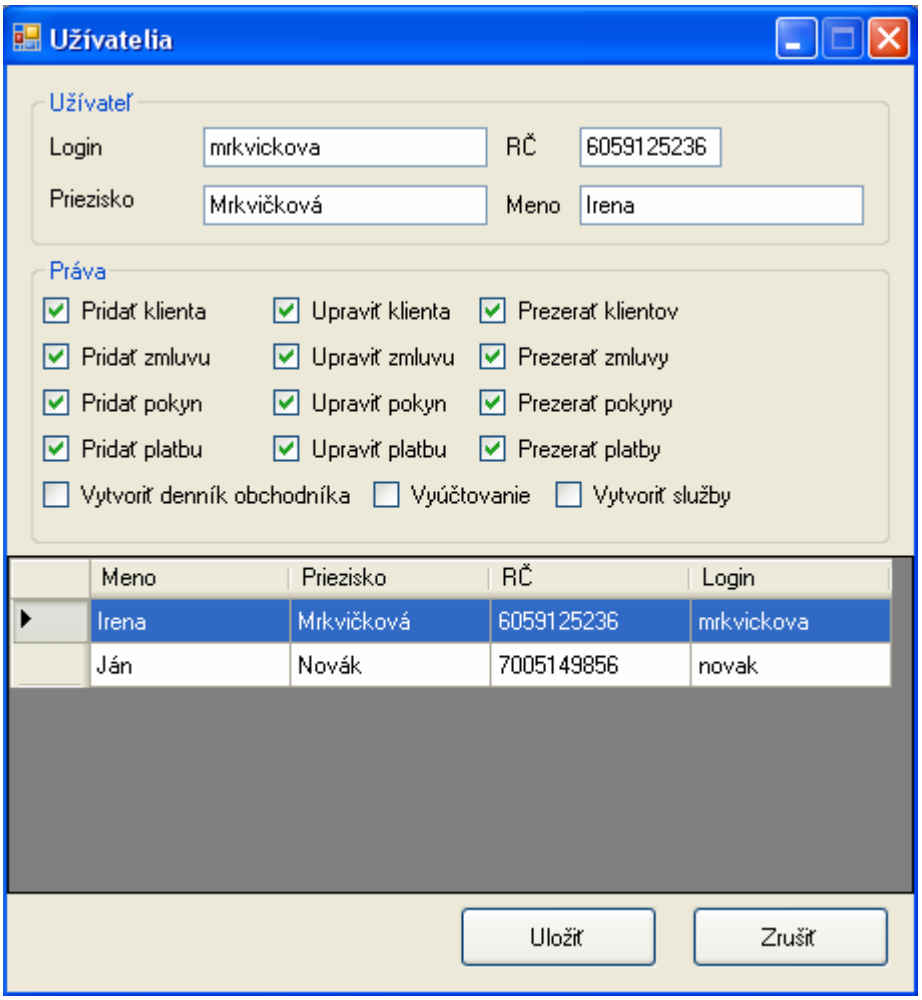



**Obrázok: Menu „Nastavenia“ administrátora**

V tomto menu si môže administrátor nechať zobrazovať jednotlivé číselníky, ktoré môže následne upravovať. Na obrázku je vybraná položka naplniť číselník CP. Pri jej výbere musí administrátor špecifikovať umiestnenia načítavaného xml súboru so službou CDCP d089 a súboru, do ktorého sa budú zapisovať prípadné chybové

hlásenia (tie nie sú vypisované na obrazovku, keďže ich môže byť veľké množstvo a aplikácia by nerobila nič iné, len vypisovala chyby). Keďže načítavaných cenných papierov je pomerne veľké množstvo je nutné skontrolovať, či sú pre hodnoty ich atribútov definované príslušné hodnoty číselníkoch a každú položku je potrebné pridať do databázy, trvá celá operácia pomerne dlho. S aplikáciou počas pridávania nie je možné pracovať, je len zobrazovaný celkový počet spracovaných položiek a počet chybných. Pridávanie je možné prerušiť stlačením tlačidla „Prerušiť“.

Jednou z hlavných úloh administrátora je správa užívateľov. Je možné vytvoriť užívateľa, nastaviť mu heslo, zmeniť heslo v prípade, že ho užívateľ zabudne, a nastavovať práva.



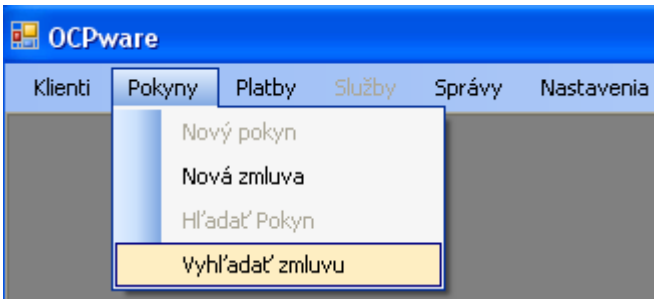

**Obrázok: Užívatelia systému**

Administrátorovou úlohou je taktiež nastaviť údaje o obchodníkovi, ako sú jeho IČO, skratka na BCPB a registračné číslo na CDCP.

Správne vyplnenie jednotlivých číselníkov má zásadný dopad na fungovanie systému, pretože podľa ich hodnôt potom aplikácia určuje chod niektorých svojich funkcií. Hodnoty číselníkov by mali spĺňať predpisy dané CDCP.

### *Práca užívateľa*

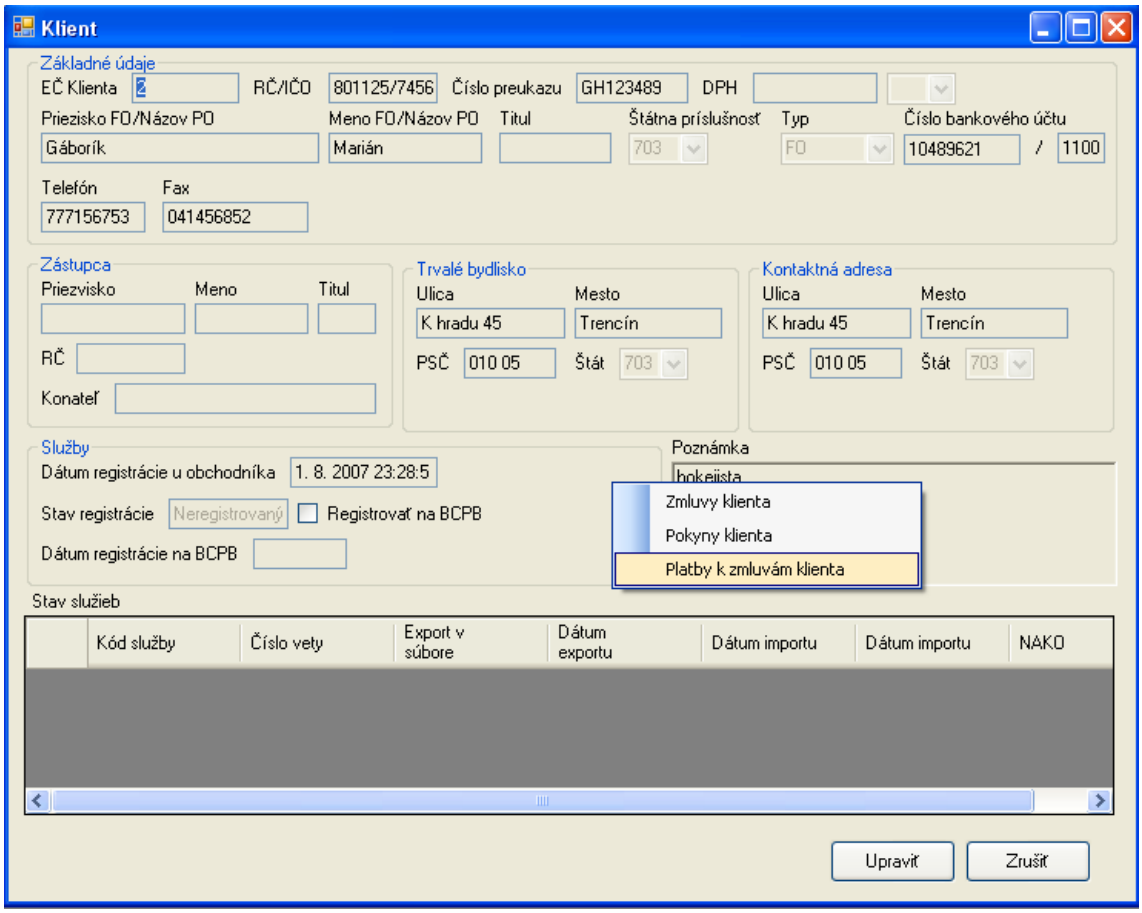
Užívateľ pracuje s aplikáciou opäť pomocou menu v hornej časti hlavného okna. Jednotlivé položky môžu byť pre daného užívateľa neaktívne, čo záleží na nastavení jeho práv.




**Obrázok: Užívateľ nemá práva na vytváranie a vyhľadávanie pokynov a na vytváranie služieb**

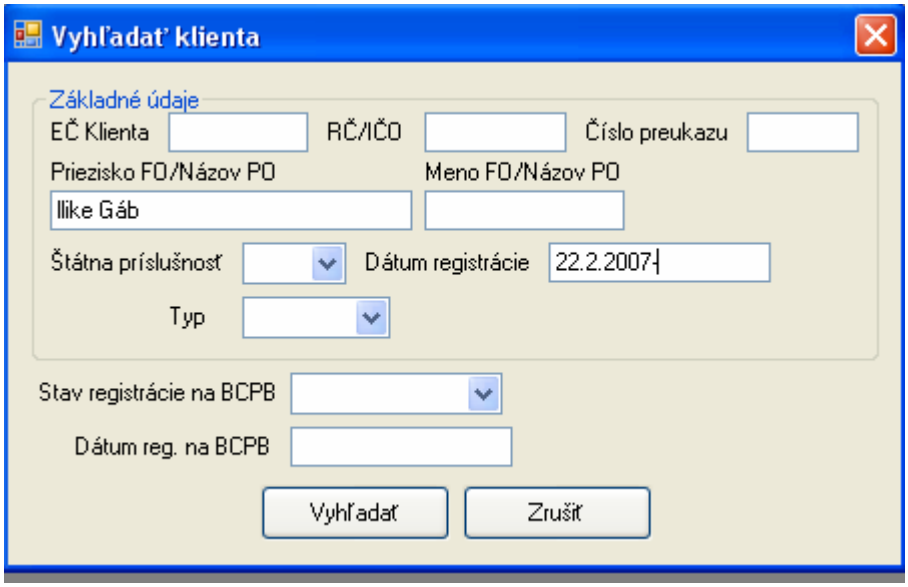
### Evidencia klientov

Evidencia klientov je spravovaná prostredníctvom položky „Klienti“ hlavného menu. Je možné vytvoriť nového klienta alebo vyhľadávať klientov v evidencii. Na obrázku je zobrazený formulár, ktorý obsahuje detaily o klientovi. Po stlačení tlačidla „Upraviť“ je možné zmeniť niektoré atribúty vybraného klienta. Formulár takisto obsahuje niekoľko kontextových menu. Pri stlačení pravého tlačidla myši nad oblasťami „Trvalé bydlisko“ a „Kontaktná adresa“ sa objaví menu s možnosťou zobrazenia histórie adries (či už trvalej alebo kontaktnej), v ostatných oblastiach sa vyvolá menu umožňujúce získať údaje o klientových zmluvách, pokynoch a platbách. V položke „Stav služieb“ sa zobrazujú informácie o službách, ktoré majú byť pre klienta zriadené. Pri pridávaní alebo editácii je dôležité vyplniť všetky povinné atribúty, ak sa tak nestane, pri pokuse o uloženie dát aplikácia užívateľa upozorní na tento stav.




**Obrázok: Detail klienta spolu s kontextovým menu**

Vyhľadávanie klientov prebieha prostredníctvom ďalšieho formulára.




**Obrázok: Formulár na vyhľadávanie klientov**

Pri vyhľadávaní je možné zadávať podmienky, ktoré majú spĺňať vyhľadané záznamy. Pri textových hodnotách je možné zadať buď presný reťazec, ktorému sa má rovnať daný atribút, napríklad keď je v kolónke „Meno“ formulára na obrázku hodnota „Jaroslav“, budú z databázy získaní všetci klient s menom Jaroslav. Existuje však aj ďalšia možnosť, kedy sa pred vyhľadávací reťazec umiestni operátor „like“. Potom pri zadaní hodnoty „like Jar“ budú vo výsledkoch zahrnutí užívatelia s menom Jaroslav, Jaromír, Jarmila a podobne.

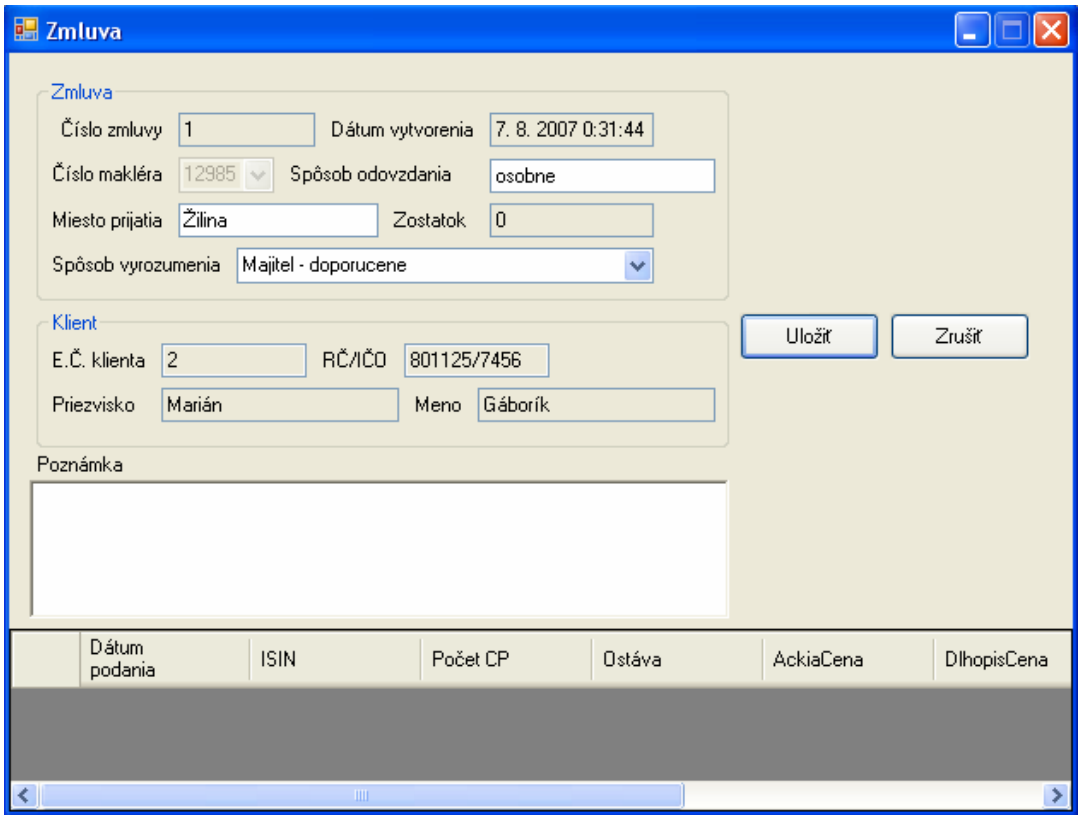
Ďalej je možné zadať intervalovú podmienku na atribúty typu dátum. Tá sa zadáva prostredníctvom znaku pomlčky záleží od jej umiestnenia, aké výsledky budú vrátené. Ak je pomlčka pred dátumom ( - 25.4.2007), znamená to, že budú vyhľadané záznamy, ktoré majú v danom atribúte menšiu alebo rovnakú hodnotu, pomlčka za dátumom zas špecifikuje tie, majú hodnotu vyššiu, ak je pomlčka medzi dvomi dátumami, znamená to interval, v ktorom sa má hodnota atribútu vyskytovať a ak v reťazci pomlčka nie je, musí mať atribút rovnakú hodnotu. Výsledky vyhľadávania sú zobrazené pomocou ďalšieho formulára, ktorý je možné vidieť na obrázku. Ak užívateľ dvakrát klikne na niektorý záznam, zobrazí sa jeho detail.


**Obrázok: Výsledky vyhľadávania**

## Evidencia zmlúv

Pred vytváraním pokynov je nutné vložiť do evidencie zmluvu. To je možné prostredníctvom ďalšieho formulára, ktorý opäť slúži aj na zobrazenie a editáciu. Do položky klient je potrebné vyplniť evidenčné alebo rodné číslo klienta, pre ktorého je zmluva vytváraná. Pri zadaní správneho údaju sa automaticky doplnia ostatné, ak sa zadané hodnoty nenachádzajú v databáze, užívateľ je upozornený. Opäť je cez kontextové menu možné vykonávať rôzne operácie. Menu nad komponentom DataGridView v spodnej časti umožňuje pridávať pokyny (ak je pokyn v editovacom móde, do ktorého sa dostane stlačením tlačidla upraviť) alebo zobraziť detaily už existujúcich pokynov. Menu vyvolané nad ostanými časťami okna ponúka možnosť zobraziť detail klienta alebo platby náležiace k zmluve. Ak je zmluva vyúčtovaná, prostredníctvom tohto menu je možné nechať vytvoriť a zobraziť správu o vyúčtovaní, ktorá sa zobrazí v novom okne, Prostredníctvom ktorého je ju možné vytlačiť.

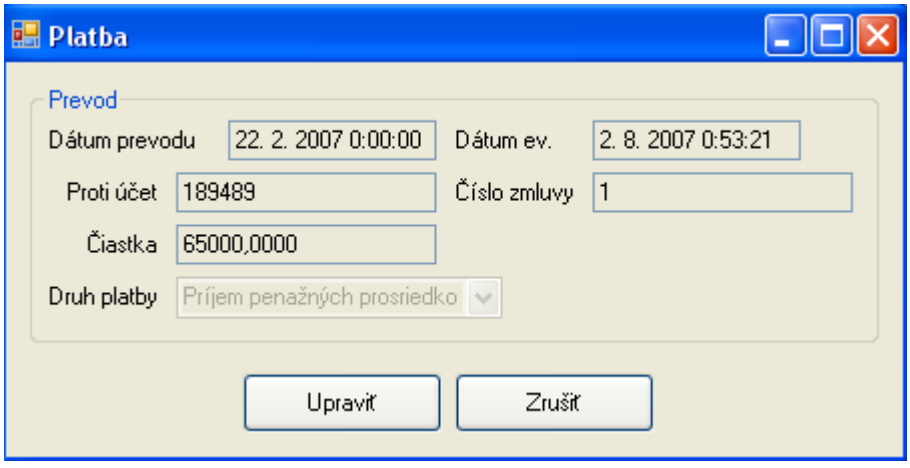
Obsluha formulára pre vyhľadávanie zmlúv sa v zásade nelíši od formulára na vyhľadanie klientov.




**Obrázok: Formulár zmluvy**

## Evidencia platieb

Každá platba musí byť priradená niektorej z vytvorených zmlúv, preto je nutné zadať číslo zmluvy. Ak sa dané číslo v evidencii nenachádza, chyba je oznámená užívateľovi. Pomocou kontextového menu je možné zobraziť zmluvu, ku ktorej platba náleží. Vyhľadávanie platieb sa nelíši od vyhľadávania v ostatných evidenciách.




**Obrázok: Platba**

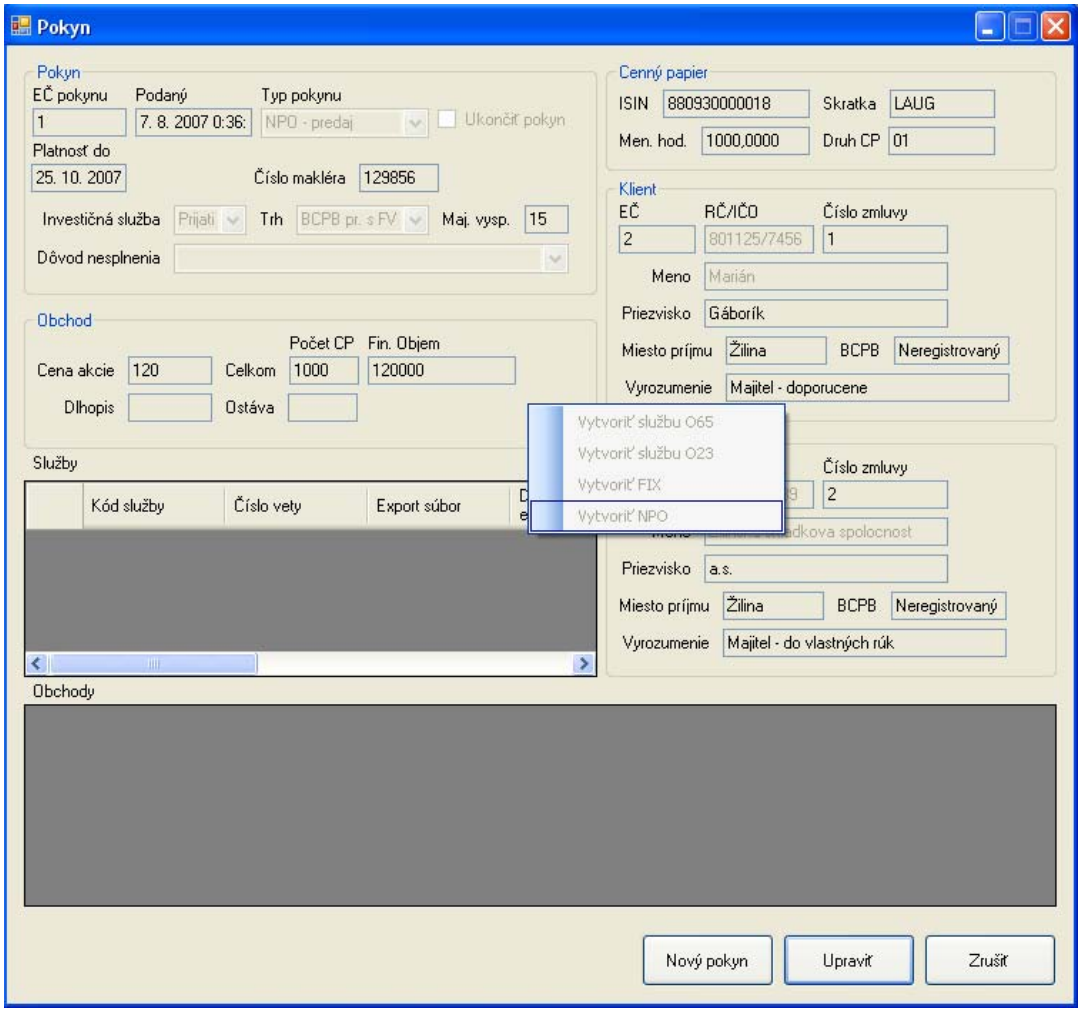
### Evidencia pokynov

Ako už bolo spomenuté, pokyn možno vytvoriť priamo z formulára zmluvy, pričom sa automaticky vyplní číslo zmluvy, ku ktorej pokyn náleží alebo z menu výberom položky „Nový pokyn“.

Pri výbere typu pokynu „NPO“ je možné vložiť číslo protizmluvy, opäť ak nebude nájdená v evidencii, tak užívateľ o tom bude informovaný. V prípade obchodu typu NPO bude pri ukladaní pokynu vytvorený protipokyn, ktorý bude tiež vložený do evidencie.

Pomocou kontextových menu je možné otvoriť vyhľadávanie v číselníku cenných papierov a čo je dôležitejšie, vytvárať služby a obchody (to je však možné len pokiaľ sú splnené všetky ich predpoklady).

Po načítaní C súboru vysporiadania je nutné doplniť k jednotlivým obchodom poplatky.




**Obrázok: Formulár pokynu s kontextovým menu služieb**

Ak je pokyn naplnený, vypršala doba platnosti, alebo je z nejakého dôvodu nesplnený, je potrebné pokyn ukončiť. To sa prevedie zaškrtnutím položky

„Ukončiť pokyn“ na formulári pokynu. Keď bude takýto pokyn uložený, už ho nebude možné spätne editovať.

### Export služieb a obchodov

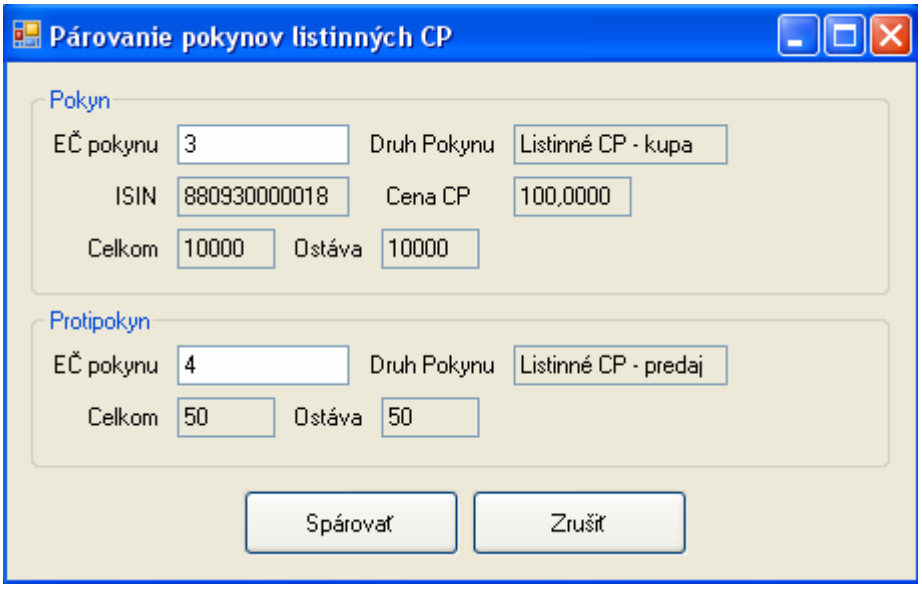
Export služieb a obchodov prebieha prostredníctvom menu „Služby“. Pre každú službu je možné zobraziť zoznam viet, ktoré majú byť exportované a jednotlivé položky z neho odoberať. Pri exportovaní služby je zároveň do evidencie pridaný záznam o jej vytvorení. Ak už bola daná služba vytvorená, užívateľ je o tom informovaný.

### Vysporiadanie

V položke „Služby“ hlavného menu je možné nájsť aj ponuku vysporiadať vykonané obchody importom súborov A a C. Pri importe súboru A je však naviac potrebné vybrať umiestnenie súboru B, do ktorého budú zapisované obchody priradené klientom.

### Párovanie listinných cenných papierov

Na párovanie listinných cenných papierov je vytvorený samostatný formulár, do ktorého sa zadávajú čísla pokynov na spárovanie, pričom systém skontroluje, či sú kompatibilné. Keďže v typickom prípade sa pri obchodoch tohto typu postupuje tak, že na jeden hlavný pokyn sú párované protipokyny, ktoré ho majú naplniť, v tomto formulári do častí „Pokyn“ vyplní hlavný pokyn a do časti „Proti pokyn“ sa zadávajú protipokyny, ktoré ho majú naplniť. Po úspešnom spárovaní preto časť „Pokyn“ ostane vyplnená a časť „Protipokyn“ sa vynuluje, aby bolo možné spárovať novú dvojicu.




**Obrázok: Formulár na párovanie pokynov**

### Vyúčtovanie

V menu „Správy“ sa nachádza položka „Vyúčtovať k dnešnému dňu“. Po jej vybratí prebehne vyúčtovanie vhodných zmlúv, ktoré sú následne zobrazené prostredníctvom nového formulára.

### Denník obchodníka

Prostredníctvom menu „Správy“ cez položku „Denník obchodníka“ je možné vytvoriť denník pokynov a obchodov obchodníka. Je nutné zadať časový interval, pre ktorý má byť denník vytvorený. Následne sa zobrazí nové okno so žiadanými údajmi.

V denníku sa nachádzajú všetky obchody, ktoré obchodník vykonal, spolu s údajmi o k nim náležiacich pokynoch. Ak pokyn nebol obchodovaný sú v denníku vyplnené len údaje o pokyne.

### Ukončenie práce s aplikáciou

Užívateľ po skončení práce s programom sa môže buď odhlásiť, alebo ho úplne vypnúť. V oboch prípadoch ak sa v systéme nachádzajú nejaké služby, ktoré neboli exportované, aplikácia na to užívateľa upozorní a neskončí, pokiaľ o to užívateľ explicitne nepožiada.

# 5. Záver

Cieľom bakalárskej práce bolo navrhnúť a implementovať informačný systém pre obchodníka s cennými papiermi, ktorý by spĺňal zároveň požiadavky legislatívy aj obchodníka. Najnáročnejšou časťou bolo pochopenie princípu obchodu s cennými papiermi a následne jeho premietnutie do návrhu systému.

Implementovaný systém umožňuje uchovávať všetky pre obchodníka dôležité evidencie ako sú klienti, zmluvy, platby a pokyny a rôzne podporné evidencie a číselníky, zriaďovať služby a vykonávať obchody spolu s finančným a majetkovým vysporiadaním.

Systém implementuje vybranú časť problematiky, preto existujú možnosti jeho rozšírenia:

- Doplnenie o definície cenníkov pre automatický výpočet komisionálnych alebo trhových poplatkov
- Doplnenie o ďalšie trhy a druhy obchodov
- Spolupráca s elektronickým bankovníctvom

## *5.1. Použité nástroje a knižnice*

Pre vytvorenie ER - diagramu bol použitý program ERTOS od Hany Kozelkovej, ktorý vznikol ako bakalárska práca na MFF UK v roku 2006

Pre potreby GUI, konkrétne multicolumn comboboxu, bola použitý komponent

http://www.codeproject.com/cs/combobox/MultiColComboSugtionBox.asp

Vytváranie správ a tlač je realizovaná prostredníctvom CrystalReports, ktoré sú súčasťou Microsoft Visual Studia 2005, ktoré bolo použité na vývoj celej klientskej aplikácie.

Na vývoj databáze bolo použité Microsot SQL Server Management Studio.

# 6. Literatúra

Gunnerson, Eric: *Začínáme programovat v C#.* Computer Press, Praha, 2001

Microsoft: *Microsoft Developer Network.* http://msdn.microsoft.com/

Kozelková, Hana: *Editor ER diagramů s podporou převodu do relačního Modelu,* bakalárska práca MFF UK, Praha, 2006

Price, Jason: *C# - programování databází.* Grada, Praha, 2005

Sceppa D.: *ADO.NET*, Microsoft Press, Redmond, 2002

Zákon č. 566/2001 Z.z. O cenných papieroch a investičných službách a o zmene a doplnení niektorých zákonov (Zákon o cenných papieroch) v znení zákona č. 336/2005 Z.z.