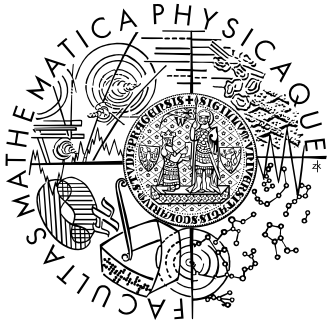
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Juraj Moško

Systém chovu králíků

Katedra softwarového inženýrství

Vedoucí bakalářské práce:
RNDr. Tomáš Bureš, Ph.D.

Studijní program: informatika, obor programování

2008

Poďakovanie: Ďakujem vedúcemu mojej bakalárskej práce za usmerňovanie pri vypracovávaní práce a za odbornú pomoc. Taktiež ďakujem všetkým, ktorí ma podporovali v tom, aby som prácu úspešne dokončil.

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

V Prahe dňa 8.8.2008

Juraj Moško

Obsah

Názov práce: Systém chovu králikov
Autor: Juraj Moško
Katedra (ústav): Katedra softwarového inženýrství
Vedúci bakalárskej práce: RNDr. Tomáš Bureš, Ph.D.
E-mail vedúceho: bures@dsrg.mff.cuni.cz

Abstrakt:
Bakalárska práca sa zaoberá návrhom a implementáciou informačného systému pre chovateľov králikov. Systém umožňuje vedenie evidenčných záznamov o chove králikov jednotlivých chovateľov. Tieto záznamy sú združované do záznamov príslušného chovateľského klubu, kde môžu byť zdieľané s ostanými členmi klubu. Záznamy jednotlivých klubov ďalej vyhodnocuje a prezentuje webová aplikácia zväzu. Systém taktiež umožňuje tlač príslušných dokumentov podľa štandardných tlačív Slovenského zväzu chovateľov. Všetky štandardy týkajúce sa evidencie králikov sú v súlade s registračným poriadkom Slovenského zväzu chovateľov.

Kľúčové slová: králik, chovateľ, evidencia, databáza, SZCH

Title: System for rabbits breeders
Author: Juraj Moško
Department: Department of Software Engineering
Supervisor: RNDr. Tomáš Bureš, Ph.D.
Supervisor's email address: bures@dsrg.mff.cuni.cz

Abstract:
Bachelor thesis considers design and implementation of information system for rabbit breeders. System provides administration of evidence's records about rearing of rabbits of individual breeders. These records are merged to records of particular club of rabbit breeders, where they can be shared with others members of club. Records of individual clubs are evaluated and presented by web application of breeders association. System also provides printing of competent documents according to forms of Slovenský zväz chovateľov. All standards related evidence of rabbits is compliant with registration order of Slovenský zväz chovateľov.

Keywords: rabbit, breeder, evidence, database, SZCH

1 Úvod

V chovateľskom svete je vedenie evidencie elektronickou formou stále veľkou neznámou. Väčšina chovateľov je zo staršej generácie, ktorá uprednostňuje klasický spôsob písanou formou. Pri potrebe vypísania toho istého rodokmeňa pre 8 potomkov jedných rodičov sa premárni veľa času, ktorý by sa dal využiť efektívnejšie. Druhá skupina chovateľov si zase nevedie evidenciu vôbec a tým stráca prehľad o vlastnom chove.

Cieľom projektu je teda navrhnúť a naimplementovať systém umožňujúci vedenie chovnej evidencie, zostavenie a tlač rodokmeňov a publikovanie informácií o svojom chove v rámci klubu a v konečnom dôsledku aj v rámci zväzu.

Treba postupovať od dôkladnej teoretickej analýzy problému a nájsť odpoveď na otázky typu: Čo evidovať? Ako to evidovať? Ktoré záznamy naopak nie je potrebné viesť detailne?

Po zodpovedaní týchto otázok prikročíme k návrhu celého systému. Zamyslíme sa nad zvolenými technológiami a odôvodníme ich výber.

Nakoniec po analýze a návrhu sa dostaneme k samotnej implementácii nášho systému. Tu si musíme dať pozor, aby sme neskĺzli do implementácii malicherností a obmedzili si tak priestor na vytvorenie komplexného systému.

Táto problematika ďaleko presahuje rozsah tejto práce. Keďže som však sám chovateľom a tento problém ma zaujal, chcel by som vyvinúť systém, ktorý implementuje základnú časť problému.

2 Špecifikácia problému a rozsah práce

Hlavný problém vedenia chovateľskej evidencie spočíva v redundancii záznamov medzi chovateľmi, čo zvyšuje výskyt chybných údajov. Úplná centralizácia údajov má tiež svoje nevýhody. Preto potrebujeme nájsť vhodný stred medzi centralizovanými a distribuovanými záznamami. Pre jasné uvedomenie si tohto problému potrebujeme bližšie definovať hierarchiu a evidenciu chovu.

2.1 Hierarchia

Organizovaný chov králikov na Slovensku, tak ako iné odvetvia podlieha istej hierarchii. Jednotliví chovatelia sú združovaní do Základných organizácii (ZO), tie sú zasa podriadené príslušnému Oblastnému výboru (OV) a tie podliehajú chovateľskému zväzu[1]. Takáto štruktúra nevyhovuje našej problematike, pretože spomenuté organizačné celky reprezentujú regióny a nie chovateľov s podobným zameraním. Preto budeme uvažovať nad inou hierarchiou.

Na samom spodku tejto hierarchie figurujú chovy jednotlivých chovateľov, tí sú združovaní do chovateľských klubov a nad tým všetkým sa nachádza zväz chovateľov[2]. Táto hierarchická štruktúra nie je jedinou možnou, avšak s ohľadom na rôznorodosť chovateľských zväzov v Európe, nie je možné uvažovať nad nejakou všeobecnou hierarchiou. Práve kvôli týmto rozdielom v zväzoch sa obmedzíme len na problematiku v rámci Slovenského zväzu chovateľov[3].

2.2 Evidencia

Racionálnu a efektívnu šľachtiteľskú prácu v chove králikov nie je možné vykonávať bez účelnej evidencie zvierat. Teda chov králikov, ktorý sa podrobuje istému šľachtiteľskému zámeru musí mať správne vedené podklady, ktoré vypovedajú o tom, či sa chov uberá správnym smerom. Povinnosťou člena SZCH je registrovať králikov v príslušnom klube[4], označiť ich priradeným tetovaním a vydávať rodokmene. V neposlednom rade dôkladne vedená evidencia dáva chovateľovi prehľad o vlastnom chove.

Problematika evidencie nespočíva len v tom, v akej forme viesť chovné záznamy, ale aj čo presne evidovať. Skúsime preto definovať, čo všetko by naše chovné záznamy mali obsahovať.

[1] Táto regionálna štruktúra zahrňuje aj chovateľov hydiny, holubov, atď.
[2] V skutočnosti len časť zväzu, a to Ústredná odborná komisia (ÚOK) pre chov králikov a kožušinových zvierat
[3] Ďalej len SZCH
[4] Chovateľ ktorý nie je členom žiadneho klubu, registruje králiky u oblastného registrátora

Králik

Informácie o samotnom králikovi tvoria najdôležitejšiu jednotku evidenčných záznamov. Každého králika sprevádza v chovateľskom svete jeho rodokmeň, ktorý sám o sebe má veľkú evidenčnú hodnotu. Napriek tomu vedenie ďalších údajov o králikovi, ktoré sa v rodokmeni nenachádzajú, nie je na škodu. Napríklad ocenenie králika na výstavách, nie je v rodokmeni králika dostatočne zdokumentované.

Chovateľ

Ďalšou časťou bez ktorej by sa evidencia chovu nezaobišla sú záznamy o chovateľoch. Každý králik bol vychovaný nejakým chovateľom, tento chovateľ sa uvádza v rodokmeni králika. Po následnom predaji králika by sa do rodokmeňa králika mala zapísať zmena majiteľa, v praxi sa to však nevykonáva. Budeme teda rozlišovať chovateľa a majiteľa králika

Výstavy a ocenenia

Každý organizovaný chov by mal byť prezentovaný verejnosti. Na to slúžia výstavy odchovaných králikov, ktoré sa zvyčajne konajú na konci sezóny. Králiky sú ocenené bodovým hodnotením[5] a niektoré aj špeciálnym hodnotením („Víťaz výstavy“, „Čestná cena“, atď.). Každý králik na výstave dostane oceňovací lístok, kde sa nachádza slovné hodnotenie k jednotlivým pozíciám[6].

Je nereálne, ukladať slovné hodnotenie do databázy, preto sa pri ocenení králika obmedzíme len na bodové a špeciálne hodnotenie. Budeme ukladať ocenenia králika z každej výstavy. V dnešnej dobe vydáva každá výstava svoj vlastný katalóg. Problematika ohľadom prihlasovania zvierat na výstavu, zostavovanie a tlač katalógu sa rieši v samostatných aplikáciách, preto sa ňou nebudeme ďalej zaoberať.

Vrh

Evidencia vrhov nám dáva predpoklad pre budúce zostavenie rodokmeňa pre konkrétneho králika. K tomu je potrebné viesť záznamy o rodičoch a dátume vrhu. Pre registráciu králika je potrebné vyplniť aj ďalší doklad, a to „Pripúšťacie potvrdenie králika“. Preto budeme tiež ukladať počty narodených, odchovaných a registrovaných králikov.

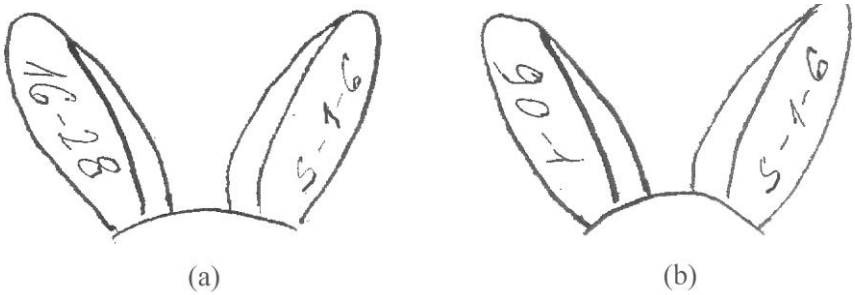
Registrácia a tetovanie

Každý vystavovaný králik musí byť zaregistrovaný u klubového (oblastného) registrátora a musí byť označený pridelenými tetovacími značkami. Chovateľ zasiela

[5] 0 – 100 bodov
[6] Hmotnosť, tvar, typ, srsť

registrátorovi vytlačené rodokmene a pripúšťacie potvrdenie, ten mu ich potvrdí a pridelí tetovacie značky. Tetovacie značky pravého ucha u králikov jedného vrhu nasledujú za sebou, pričom samci sú v tejto postupnosti pred samicami. Prideľovanie tetovacích značiek sa riadi Registračným poriadkom SZCH.

Nasledujú pravidlá a formát tetovacích značiek.[7]



Obrázok 2.1

Ľavá ušnica[8]

V tejto ušnici sa nachádza znak „S“ reprezentujúci krajinu (Slovensko), v ktorej bol králik registrovaný, nasleduje číslo „1“ vyjadrujúce mesiac (január) narodenia králika a číslica „6“[9] reprezentujúca rok[10] narodenia králika. (Obrázok 2.1 a)

Pravá ušnica

Tetovacie značky tejto ušnice sú komplikovanejšie, pretože vyjadrujú špecifický druh a stupeň chovu. Z tetovania rozoznávame či ide o klubovú, oblastnú, alebo o ústrednú registráciu a stupeň chovu – kmeňový, plemenný, špeciálny. Pri klubovej registrácii sa ďalej definuje číslo línie potomkov konkrétneho samca (zakladateľa línie), číslo línie sa dedí po otcovi.

1. 16 – 28: králik je registrovaný v oblastnom výbore Trenčín („16“) a je dvadsiaty ôsmy („28“) v poradí svojho plemena v danom roku. (Obrázok 2.1 a)

2. 90 – 1: králik je registrovaný ústredným registrátorom[11] („90“) a je prvý („1“) v poradí svojho plemena v danom roku. (Obrázok 2.1 b)

[7] Ako je uvedené v Haspra a kol. (1996) [1]
[8] Obrázok znázorňuje králika pri pohľade spredu, teda ľavá ušnica je na obrázku tá vpravo
[9] V čase vydania registračného poriadku to znamenalo rok 1996, dnes to znamená rok 2006
[10] Ide o poslednú číslicu roka, nepredpokladá sa, že králik bude žiť viac ako 10 rokov
[11] Ústredný registrátor registruje králiky chovateľov, ktorý patria do ZO, ktoré nie sú podriadené žiadnemu OV

Obrázok 2.2

3. Špeciálny chov

   a. S – 154: králik je registrovaný v chovateľskom klube[12] („S“) a je stopäťdesiaty štvrtý v poradí svojho plemena v danom roku. (Obrázok 2.2 a)

   b. S – 1504: králik je registrovaný v chovateľskom klube („S“), pochádza z pätnástej línie („15“) a v tejto línii je štvrtý („04“) v poradí svojho plemena v danom roku. (Obrázok 2.2 b)

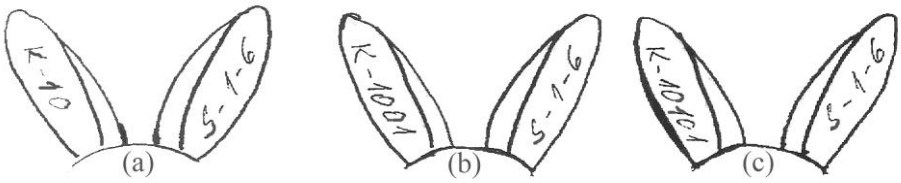
   c. S – 1544: králik je registrovaný v chovateľskom klube („S“), pochádza z pätnástej línie („15“) a v tejto línii je štyridsiaty štvrtý („44“) v poradí svojho plemena v danom roku. (Obrázok 2.2 c)

4. Kmeňový chov

   a. K – 10: králik pochádza z kmeňového chovu („K“) a je desiaty („10“) v poradí svojho plemena v danom roku. Registrovaný je u ústredného registrátora, alebo v klube, ak klub pre dané plemeno existuje. (Obrázok 2.3 a)

   b. K – 1001: králik pochádza z kmeňového chovu („K“), z desiatej línie („10“) a v tejto línii je prvý („01“) v poradí svojho plemena v danom roku. (Obrázok 2.3 b)

   c. K – 10101: králik pochádza z kmeňového chovu („K“), z desiatej línie („10“) a v tejto línii je stoprvý („101“) v poradí svojho plemena v danom roku. (Obrázok 2.3 c)



Obrázok 2.3

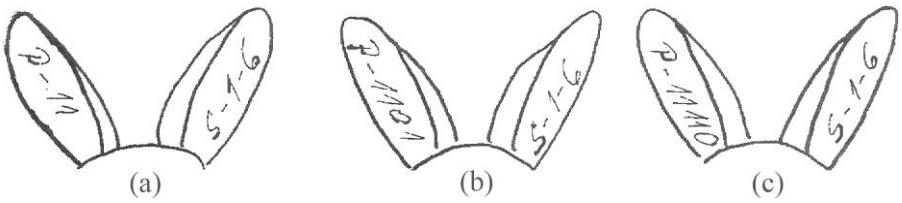
[12] Každý špeciálny chov je klubový, avšak klubový chov môže byť aj kmeňový, alebo plemenný

5. Plemenný chov

    a. P – 11: králik pochádza z plemenného chovu („P“) a je jedenásty („11“) v poradí svojho plemena v danom roku. Registrovaný je u ústredného registrátora, alebo v klube, ak klub pre dané plemeno existuje. (Obrázok 2.4 a)

    b. P – 1101: králik pochádza z plemenného chovu („P“), z jedenástej línie („11“) a v tejto línii je prvý („01“) v poradí svojho plemena v danom roku. (Obrázok 24. b)

    c. P – 11110: králik pochádza z plemenného chovu („P“), z jedenástej línie („11“) a v tejto línii je stodesiaty („110“) v poradí svojho plemena v danom roku. (Obrázok 24. c)



Obrázok 2.4

Pomlčky v tetovaní sú v praxi nahradené medzerami, alebo úplne zrušené, preto budeme ďalej predpokladať len tetovanie bez pomlčiek a medzier.

2.3 Postup riešenia

V každej hierarchii, ktorá nemá záznamy centrálne uložené a spravované, dochádza k redundancii záznamov, čo vedie k rozdielnym a chybným údajom. Preto je potrebné údaje chovateľov centralizovať a na to nám poslúži uvažovaná hierarchia chovateľ – klub – zväz.

Táto hierarchia nám ponúka 2 možnosti centralizácie údajov, na úrovni klubu a na úrovni zväzu. Centralizovať záznamy chovateľov na úrovni zväzu má viaceré nevýhody. Združenie dát na jednom mieste, by bez replikácie predstavovalo dosť veľké riziko straty všetkých údajov. Záťaž by spočívala na jednom bode a tento bod by sa stal úzkym hrdlom nášho systému. Na druhej strane stojí fakt jednotnosti údajov v celom systéme. Keďže však zväz nepotrebuje informácie o všetkých údajoch chovateľov, rozhodneme sa pre centralizáciu záznamov na úrovni klubu. Klub následne poskytne zväzu len potrebnú časť údajov.

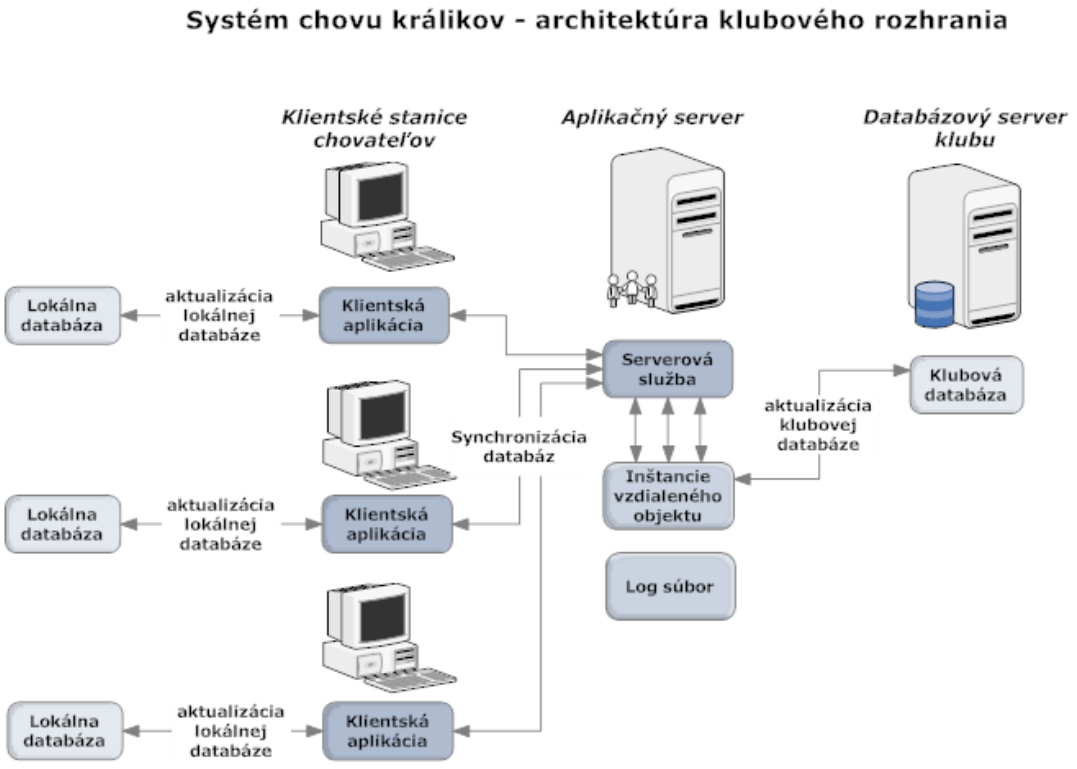
3 Návrh

3.1 Architektúra systému

Z predchádzajúcej analýzy vyplýva, že celý systém bude pracovať s databázami na rôznych úrovniach uvažovanej hierarchie. Tieto databázy musia byť sprístupnené nejakým užívateľským rozhraním, kvôli zobrazovaniu a aktualizácii záznamov. Požiadavka centralizácie a distribúcie záznamov nám zase kladie podmienku prepojenia databáz. Nami uvažovaná hierarchia má tri úrovne. Tieto tri úrovne sa v architektúre dajú pokryť dvoma časťami, keďže centralizáciou záznamov na úrovni klubu, sme spojili úroveň chovateľa s úrovňou klubu.

Architektúra klubového systému

Predchádzajúce tvrdenia, ktoré spájajú chovateľa s klubom musíme brať s nadhľadom. Jednou zo základných požiadaviek na systém je, aby bol prístupný každému chovateľovi, teda aj takému, čo nie je členom žiadneho klubu. Taktiež v praxi nie každý počítač je pripojený k internetu, preto je nutnosť mať u každého chovateľa vlastnú databázu, ktorá umožní offline pripojenie k záznamom chovateľa.



Obrázok 3.1

Na prvý pohľad je z obrázku[13] zjavné, že architektúra klubového systému nie je založená na klasickej klient – server architektúre, ale že je použitá o niečo zložitejšia 3-vrstvová architektúra[14]. Táto architektúra sa skladá z týchto vrstiev:

- **Dátová vrstva**
  Tu sú dáta ukladané a získavané z databázy, v našom modeli túto vrstvu reprezentuje klubová databáza.

- **Logická (aplikačná) vrstva**
  Na tejto vrstve sú vykonávané logické operácie a výpočty, taktiež táto vrstva tvorí prepojenie medzi dátovou a prezentačnou vrstvou, v našom modeli ju reprezentuje aplikačný server.

- **Prezentačná vrstva**
  Túto vrstvu reprezentuje užívateľské rozhranie, v našom modeli je to klientska aplikácia

Tento model bol zvolený preto, že všetci chovatelia sa ku klubovej databáze pripájajú cez jeden aplikačný server, teda implementácia aplikačnej vrstvy v rámci klubu je na jednom mieste. Je preto ľahšie spravovateľná, ako keby bola implementovaná v každej klientskej aplikácii samostatne. To nám dáva napríklad výhodu pri prechode na inú klubovú databázu[15], stačí prepísať distribúciu dát v aplikačnej vrstve, presunúť dáta z jednej databázy do druhej a klientskú aplikáciu zmena nijako nezasiahne.

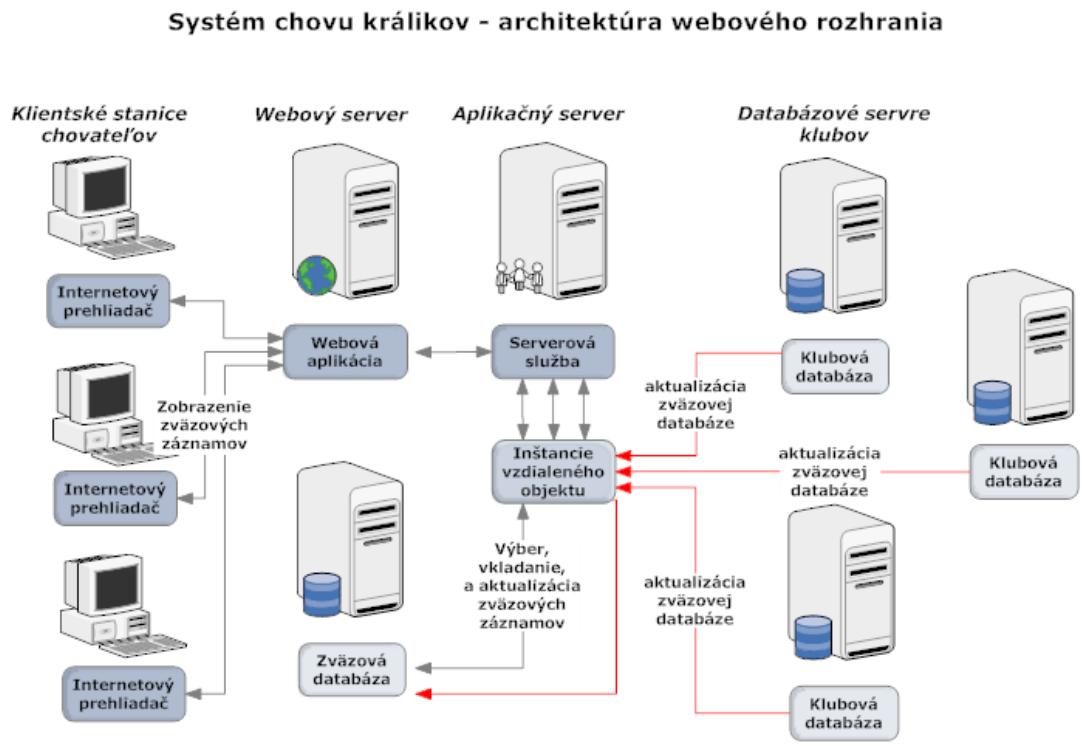
Výmena klientskej databázy by samozrejme nešla tak jednoducho. Po predchádzajúcich argumentoch sa ponúka možnosť implementovať medzivrstvu aj pre pripojenie k lokálnej databáze chovateľa. Treba si však uvedomiť, že aplikačná vrstva by musela byť tiež inštalovaná na každú klientskú stanicu, čím by sme prišli o spomenuté výhody aplikačnej vrstvy. Pripojenie klientskej aplikácie k lokálnej databáze je natoľko jednoznačné, že nevyžaduje medzivrstu. Preto je toto pripojenie založené na klient – server architektúre.

Architektúra zväzového systému

Druhú časť celkovej architektúry, tvorí systém pre zväz chovateľov. Nakoľko centralizáciu údajov sme preniesli na jednotlivé kluby, zväzová aplikácia bude mať za úlohu zosumarizovať a prezentovať klubové záznamy. Na to najlepšie slúži webové rozhranie, preto sme pre náš systém vybrali túto možnosť.

[13] Obrázok 3.1
[14] Z anglického three-tier architecture
[15] Napríklad pri prechode z Oracle na MS SQL

Webový prehliadač u klienta sa pripojí k webovému serveru s našou webovou aplikáciou. Táto komunikácia je už implementovaná vo webovom prehliadači, takže pri pohľade na obe architektúry môžeme vidieť paralelu medzi webovou aplikáciou a klientskou aplikáciou chovateľa. Nedajme sa však pomýliť, webová aplikácia „žije“ na serveri len jedna, na rozdiel od klientskej aplikácie, ktorú má každý chovateľ nainštalovanú na svojom počítači.



Obrázok 3.2

Pri pohľade na obrázok[16] môžeme opäť vypozorovať 3-vrstvovú architektúru. Webová aplikácia sa prostredníctvom aplikačného servera pripojí ku klubovým databázam a stiahne si odtiaľ prístupné dáta klubu. Tieto dáta si ukladá vo svojej databáze, ktorá je tiež prístupná cez aplikačný server. Práve preto, že webová aplikácia je v celom systéme len jedna, môžeme využiť výhody 3-vrstvovej architektúry aj pre pripojenie k „lokálnej“ databáze zväzu. V skutočnosti táto databáza nemusí byť ani fyzicky prítomná na rovnakom serveri ako webová aplikácia.

[16] Obrázok 3.2

3.2 Voľba platformy

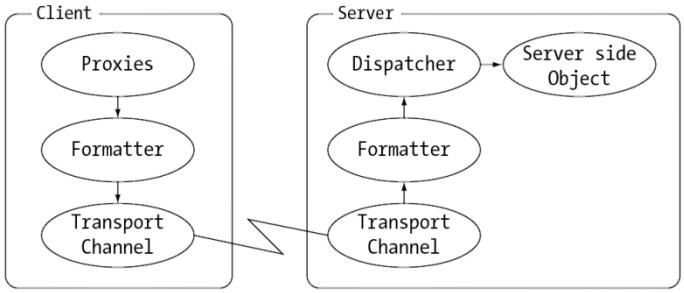
Vzhľadom k aktuálnemu rozšíreniu medzi chovateľmi bol zvolený operačný systém Microsoft Windows XP s podporou .NET Framework v2.0. Vývojová platforma bola vybratá platforma .NET, ktorá je priamo od Microsoftu, čo dáva predpoklad na bezproblémovú integráciu vyvíjaného projektu do operačného systému. Za programovací jazyk bol zvolený jazyk C#.

3.3 Použité technológie

Okrem štandardných technológii ako je práca so súbormi, tvorba GUI, atď., boli použité komplexnejšie technológie .NET remotingu a ADO.NET pre prácu s dátami. Na tieto dve technológie sa bližšie pozrieme.

.NET remoting[17]

Remoting ako proces je súčasťou programov, ktoré potrebujú komunikovať skrz určité hranice. Vzhľadom k tomu je používaný práve pre komunikáciu medzi rôznymi aplikáciami[18], typicky na rôznych počítačoch. V .NET Framework táto technológia vytvára základ pre distribuované systémy a nahrádza DCOM[19].



Obrázok 3.3 .NET Remoting - architektúra podľa Rammera (zjednodušená)

Výhody .NET remotingu boli už načrtnuté v popise architektúry nášho systému. Skúsme spomenúť vlastnosti tejto technológie ako ich popisuje Rammer (2002) [2]:

- **Jednoduchý na implementáciu**
  Implementovanie systému využívajúceho remoting je jednoduchší ako v iných architektúrach. Nepotrebujete vytvárať rozhranie v abstraktnom jazyku ako napríklad pri použití technológii CORBA a DCOM.

[17] Ako je uvedené v Rammer (2002) [2]
[18] Technológia .NET remotingu je navrhnutá aj pre komunikáciu v rámci jednej aplikácie
[19] Distrbuted Component Object Model - Proprietárne riešenie od firmy Microsoft na komunikáciu medzi distribuovanými softvérovými komponentami, po nástupe technológie .NET je vytláčaný práve .NET remotingom

- **Rozšíriteľná architektúra**
  V kontraste s inými architektúrami remotingu väčšina vsrtiev .NET remotingu môže byť rozšírená, alebo úplne nahradená. To nám dáva napríklad možnosť prepísaním pár riadkov v konfiguračnom súbore zmeniť transportný kanál z Tcp na Http.

- **Serializácia dát**
  Objekt, ktorý je posielaný cez .NET remoting je na jednej strane serializovaný a na druhej deserializovaný. Serializačný mechanizmus nám umožňuje posielať nie len jednoduché dátové typy, ale aj komplexné objekty.

- **Manažment životného cyklu**
  Každému vzdialenému objektu sa po vytvorení pridelí hodnota lifetime. Po vyčerpaní tejto hodnoty objekt na serveri zanikne. Obnova lifetime hodnoty, môže byť vykonávaná rôznymi spôsobmi, najčastejšia možnosť je obnova po každom zavolaní funkcie objektu.

Systém využívajúci remoting sa skladá väčšinou z 3 častí, zo serverovej aplikácie, klientskej aplikácie a vzdialeného objektu, ktorý je na strane servera. Klientská aplikácia volá metódy vzdialeného objektu, ktoré sa vykonávajú na serveri.

Vzdialený objekt môže byť typu *MarshallByValue*, keď je pri volaní celý objekt serializovaný a na strane klienta sa vytvorí kópia objektu, alebo typu *MarshallByReferenece*, keď na strane klienta sa vytvorí špeciálny proxy objekt, ktorý sprístupní objekt vytvorený na serveri.

Ďalším parametrom konkrétneho riešenia .NET remotingu je spôsob aktivácie vzdialeného objektu. Rozlišujú sa 2 základné typy: Objekt aktivovaný serverom[20] a objekt aktivovaný klientom[21], SAO ďalej rozlišuje aktivačný mód *Singleton* a *Single call*.

- SAO
  - *Singleton*
    Objekt je vytvorený pri prvej požiadavke prvého klienta a na serveri existuje táto inštancia pre všetkých klientov. Preto pri aktivácii tohto typu musíme zabezpečiť zdieľanie zdrojov pre viacerých klientov (vlákien).

  - *Single call*

[20] Z anglického server activated object (SAO)
[21] Z anglického client activated object (CAO)

Pre každú požiadavku je vytvorený nový objekt. Z toho vyplýva, že tento systém je bezstavový a neumožňuje zdieľanie dát medzi klientmi.

- CAO:
  - Pre každého klienta sa vytvorí jedna inštancia objektu a ten si na serveri udržuje stav. Zdieľanie dát medzi klientmi nie je možné.

Popis zvoleného riešenia pre náš systém je podrobne rozobratý v časti popisujúcej implementáciu aplikačného servera.

ADO.NET

ADO.NET je .NET framework pre prácu s dátami. Zabezpečuje jednak získanie týchto dát z databázy a na druhej strane uloženie dát v operačnej pamäti pre ďalšiu prácu s nimi. Podľa týchto úloh by sa koncepcia objektového modelu technológie ADO.NET dala rozdeliť na 2 základné časti: Data providery a DataSety.

Data providery

Data provider slúži na sprostredkovanie dát uložených v databáze. .NET framework ponúka 4 základné typy providerov. Avšak Data provider ku vlastnej databáze typicky implementuje tvorca databázy. Ak by každý výrobca implementoval Data providera nezávisle na ostatných, celá koncepcia Data providera by bola zmätená a pri prechode na novú databázu by sa pravdepodobne musel celý systém naimplementovať odznova. Preto ADO.NET definuje určité rozhrania a bázové triedy, ktoré musí každý data provider implementovať. Vysvetlíme význam tých najdôležitejších.

- *DbConnection*
  Táto trieda reprezentuje pripojenie k databáze a deklaruje metódy pre otvorenie a uzatvorenie pripojenia. Taktiež ponúka funkciu na začatie transakcie. Implementácie ostatných tried potrebujú *DbConnection* ako svoju vlastnosť.

- *DbCommand*
  Reprezentuje SQL dotaz, ktorý chceme predložiť databáze. Poskytuje metódy *ExecuteNonQuery()* – vykoná SQL dotaz a vráti počet ovplyvnených riadkov, používa sa pre SQL príkazy INSERT, UPDATE a DELETE, *ExecuteScalar()* – vykoná SQL dotaz (SELECT) a vráti prvý záznam v prvom riadku z výsledku dotazu, využíva sa pri použití agregačných funkcií a *ExecuteReader()* – vykoná SQL dotaz (SELECT) a vráti výsledok vo forme inštancie objektu *DbDataReader*. Implementácia obsahuje zaujimavú vlastnosť *CommandType*, pomocou ktorej môžeme určiť typ príkazu ako volanie uloženej procedúry.

- *DbParameter*
  DbCommand obsahuje aj vlastnosť *Parameters*, čo je kolekcia objektov typu *DbParameter*. Toto rozhranie slúži na definovanie parametrov príkazu. Príkaz, ktorý má hodnoty parametrov priamo v texte príkazu, je náchylný na útoky typu SQL injection, obzvlášť ak sú hodnoty získané priamo od užívateľa. Preto sa do textu príkazu píše len názov parametru, ktorý sa potom k *DbCommand* pripojí. *DbParameter* má základné vlastnosti *DbType*, *ParameterName*, *Value* a vlastnosť *Direction*, ktorá určuje, či parameter je vstupný, výstupný, alebo návratová hodnota uloženej funkcie.

- *DbDataReader*
  Používa sa spolu s funkciou *ExecuteReader()* triedy *DbCommand*. Táto trieda vracia sekvenčne riadky, ktoré vyhoveli SQL dotazu. Slúži na to metóda *Read()*. Ďalšie využitie tejto triedy je použitie metód na priamy prístup k hodnotám aktuálneho riadku, slúžia nám na to metódy *GetInt32()*, *GetString()*, *GetDateTime()*, atď., pričom po zavolaní takejto metódy sa indexer posunie na ďalší stĺpec. Túto techniku využijeme ak poznáme presné poradie a typ stĺpcov danej tabuľky.

- *DbDataAdapter*
  Táto trieda slúži na prepojenie databázových dát s dátami uloženými v pamäti teda s *DataSet*om. Implemetuje triedu *Fill()*, ktorá naplní *DataSet*, alebo *DataTable*. Obsahuje vlastnosti, *InsertCommand*, *UpdateCommand* a *DeleteCommand*, ktoré potrebujú byť definované ak chceme použiť metódu *Update()*. Táto metóda updatuje dáta v databáze, podľa toho ako sa zmenili dáta v *DataSet*e, ktorý naplnil príslušný *DbDataAdapter*.

DataSety

DataSet v .NET architektúre je pomenovanie pre jednoduchú relačnú databázu, ktorá je uložená v pamäti. *DataSet* sa naplní pomocou triedy *DbDataAdapter* a cez túto triedu dokáže updatovať databázu. Implementuje metódy GetChanges(), AcceptChanges() a RejectChanges(), ktoré pracujú so zmenami dát, ktoré nastali v *DataSet*e od jeho naplnenia. Taktiež dokáže zapisovať schému a dáta do Xml súboru a opätovne tieto dáta načítať. Rozlišujeme *DataSet* silne typovaný a netypovaný.

Silne typovaný *DataSet* má svoju schému popísanú v Xml súbore s touto schémou spolupracuje aj kompilátor. To má výhodu, že väčšina chýb sa objaví už pri preklade. Ďalšou výhodou je využitie Intellisense vývojového prostredia. Značnou nevýhodou tohto typu je, že pri akejkoľvek zmene štruktúry databázy sa musí upraviť aj štruktúra typovaného *DataSet*u, čo väčšinou vedie k zmene v kóde aplikácie.

Schéma netypovaného *DataSet*u je vytváraná buď dynamicky v kóde aplikácie, alebo, čo je častejšie riešenie, sa schéma automaticky vytvorí pri naplnení *DataSet*u triedou *DbDataAdapter*. Výhody typovaného *DataSet*u sa preto strácajú, no na druhej strane sa práca s netypovaným *DataSet*om dá naimplementovať tak, že zmena v databáze neovplyvní fungovanie aplikácie.

S triedou *DataSet* úzko spolupracujú nasledujúce triedy.

- *DataTable*
  DataSet obsahuje vlastnosť *Tables*, čo je kolekcia tried *DataTable*, ktoré reprezentujú tabuľky v databáze. Priame operácie s dátami v *DataSet*e sú vykonávané cez príslušnú *DataTable*. Trieda obsahuje vlastnosti *Rows* a *Columns*, čo sú kolekcie tried *DataRow* a *DataColumn*. Zaujímavá je metóda *Select()*, ktorá umožňuje pomocou špecifického výrazu vybrať len určité riadky z tabuľky.

- *DataView*
  Táto trieda umožňuje filtrovanie a triedenie záznamov objektu *DataTable*, ktorý je nastavený vo vlastnosti *Table*. Na filtrovanie záznamov slúži vlastnosť *RowFilter*, kde sa zadáva potrebný výraz (veľmi podobný WHERE klauzule v SQL). Pre zoradenie záznamov potrebujeme zadať názvy stĺpcov, podľa ktorých sa má radiť a na to slúži vlastnosť *Sort* (syntax podobná ORDER BY klauzule v SQL).

- *DataColumn*
  Reprezentuje stĺpec v tabuľke.

- *DataRow*
  Reprezentuje riadok v tabuľke.

- *DataRelation*
  DataRelation definuje relačný vzťah medzi dvoma tabuľkami, kde jedna je závislá na druhej. Typicky reprezentuje FOREIGN KEY CONSTRAINT v databáze. Pri práci so záznamom v jednej tabuľke dokáže sprístupniť záznam, ktorý je na ňom závislý v druhej tabuľke.

- *Constraint*
  Reprezentuje obmedzenie definované v databáze.

Popis zvoleného riešenia pre náš systém je podrobne rozobratý v časti popisujúcej implementáciu aplikačnej a prezentačnej vrstvy.

4 Implementácia

4.1 Databázová vrstva

Databáza slúži na ukladanie chovateľských záznamov. Pre svoju rozšírenosť, prirodzené uchovávanie dát a rokmi overenú použiteľnosť dotazovacieho jazyka SQL bol uprednostnený model relačnej databázy pred modelom objektovej databázy. Ako vyplýva z architektúry, systém bude pracovať s tromi typmi databáz:

- Lokálna databáza chovateľa
- Centrálna databáza klubu
- Zväzová databáza

Lokálna databáza chovateľa

Jednou zo základných požiadaviek kladených na implementovaný systém je sprostredkovanie funkčnosti vedenia si svojej chovateľskej evidencie každému chovateľovi. Preto bol od začiatku uvažovaný aj model práce bez pripojenia k internetu. Táto požiadavka kladie značné obmedzenie na umiestnenie databázy. Lokálna databáza bude preto nainštalovaná na počítači chovateľa spolu s klientskou aplikáciou. Obmedzenia na lokálnu databázu klientskej aplikácie boli prirodzene najväčšie zo všetkých troch typov. Databáza musí byť kompaktná, aby sa dala ľahko zálohovať a nezaberala veľa miesta na disku. Musí ovládať základnú syntax jazyka SQL a v neposlednom rade musí existovať Data provider pre ADO.NET. Po zvážení týchto podmienok bola zvolená databáza SQLite.

SQLite databáza sa na pevnom disku nachádza vo forme jedného súboru, čím sa vyriešil problém zálohovania a obnovy databázy, keď stačí zálohovať práve tento jeden súbor. SQLite podporuje syntax jazyka SQL92, okrem nasledujúcich funkcií[22].

- **FOREIGN KEY CONSTRAINT**
  SQLite pozná ich syntax, ale ignoruje ich. Cudzie kľúče boli v našej databáze nahradené BEFORE a AFTER triggermi.

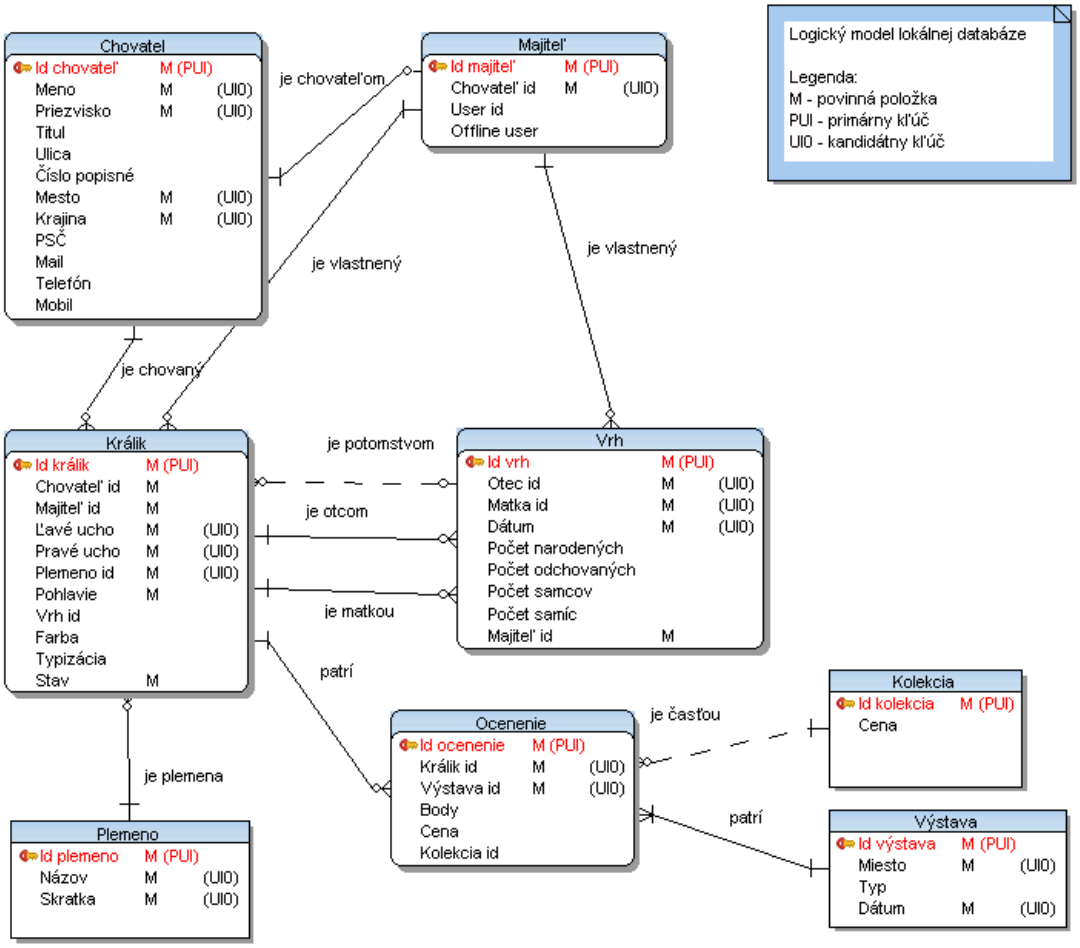
- **RIGHT a FULL OUTER JOIN**
  Spájanie tabuliek bolo použité pri vytváraní pohľadov, syntax bola prispôsobená tomuto obmedzeniu a použité sú len podporované príkazy LEFT OUTER JOIN a JOIN.

- **Zapisovanie do pohľadov**

[22] Spomenuté sú len tie, ktoré boli potrebné a boli nahradené iným riešením

SQLite navrhuje riešenie, vytvorením BEFORE triggerov pre daný pohľad a pre príkazy UPDATE, INSERT, DELETE. Po zadaní niektorého z týchto príkazov sa zavolá príslušný trigger, pri neexistencii triggera sa vyvolá výnimka. Toto riešenie bolo použité aj v našej databáze.

Náhrada obmedzení databáze triggermi má nepriaznivý vplyv na výkon, keďže však databáza pracuje len s dátami jedného užívateľa, táto nevýhoda je akceptovateľná.



Obrázok 4.1 Logický model lokálnej databázy

Pred implementáciou databázy bol zostavený logický model[23], pomocou ktorého sa neskôr vytvorila schéma[24]. Entity logického modelu boli prevedené na tabuľky a podľa vzťahov medzi nimi boli definované integritné obmedzenia. Z toho nám v lokálnej databáze vzniklo 8 tabuliek. Ku všetkým kľúčom, či už primárnym, kandidátnym, alebo cudzím boli vytvorené indexy, kvôli optimalizácii vyhľadávania.

V správnom návrhu databázy by primárne kľúče nemali byť vytvorené s pôvodných stĺpcov tabuľky, ale mal by sa pridať nový stĺpec s jednoznačným identifikátorom.

[23] Obrázok 4.1
[24] Schéma spĺňa podmienky tretej normálnej formy

Prvotný návrh preto počítal s auto - inkrementálnymi primárnymi kľúčmi. Neskôr sa však zistilo, že pre distribuované databázové systémy nie je takýto typ kľúča vhodný, pretože je stavový. Pre synchronizáciu databáz potrebujeme nestavové kľúče, ktoré by sa dali relatívne jednoducho synchronizovať. Preto v celom systéme sú použité takzvané GUID[25] kľúče, ktoré sú tvorené 16 bytovým číslom a sú nestavové. Prirodzene tento typ bol použitý aj pre cudzie kľúče.

Pri implementácii synchronizácie databáz bol ku každej tabuľke pridaný stĺpec *deleted,* ktorý vypovedá o tom, či je daný riadok zmazaný a stĺpec *version*, ktorý určuje verziu daného záznamu v tabuľke. Bližší význam týchto stĺpcov je vysvetlený v implementácii aplikačného servera.

Databáza sa snaží kontrolovať integritu dát obmedzeniami definovanými k príslušným tabuľkám. Na čo nám obmedzenia nestačili, bolo implementované integritnými triggermi. Napríklad problematika formátu registračných značiek je implementovaná BEFORE triggerom.

Keďže SQLite nepodporuje uložené procedúry, operácie pracujúce s dátami boli implementované až na prezentačnej vrstve.

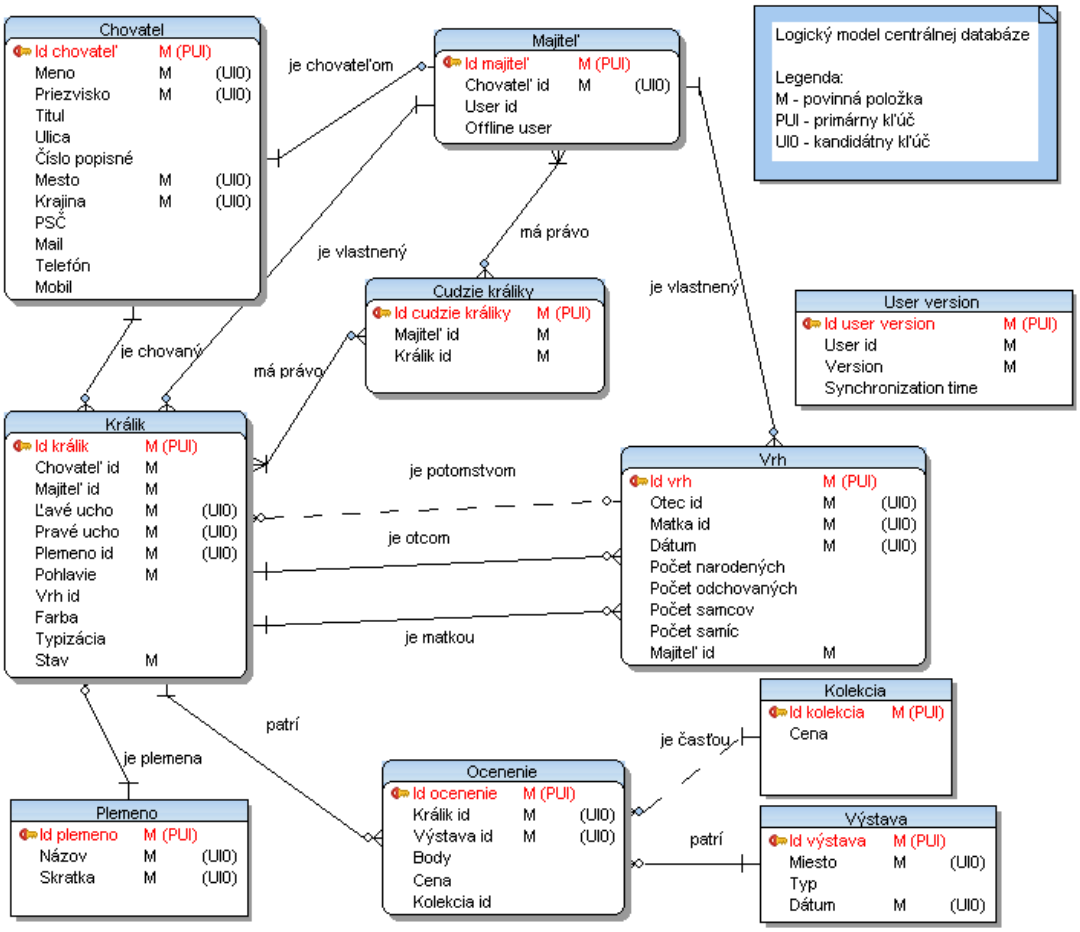
Ako posledné boli do lokálnej databázy implementované pohľady, ktoré boli pridávané dodatočne, keď si to prezentačná vrstva vyžadovala.

Centrálna databáza klubu

V klubovej databáze sú uložené dáta všetkých chovateľov klubu. Prvá požiadavka na centrálnu databázu je multi - užívateľský prístup. Keďže databáza bude umiestnená na samostatnom serveri, nie je nutné aby bola kompaktná. Nášmu systému a požiadavkám vyhovujú databázy MS SQL a Oracle. Zvolená bola databáza Oracle, hlavne kvôli predchádzajúcim praktickým skúsenostiam s týmto systémom.

Pri multi – užívateľskom prístupe do databázy vychádza na povrch otázka ako spravovať užívateľov. Ponúkajú sa dve možnosti. Buď správu užívateľov nechať na databázovej vrstve, alebo ju posunúť do vrstvy aplikačnej. Z bezpečnostných dôvodov, bola zvolená prvá možnosť. Model správy užívateľov v Oracle je jednoduchý na používanie a zároveň dostatočne robustný. Administrátor klubovej databázy teda vytvorí databázovú schému a užívateľov, ktorým pridelí práva na objekty, s ktorými budú môcť pracovať. Práva v Oracle sú typu SELECT, INSERT, UPDATE, DELETE a EXECUTE a objekt, pre ktorý sú práva definované je najčastejšie tabuľka alebo uložená procedúra[26].

[25] Global unique identifier
[26] Uložené procedúry sú zvyčajne obalené do balíčkov funkcií



Obrázok 4.2 Logický model centrálnej databázy

Z logického modelu možno vypozorovať, že oproti lokálnej databáze nám pribudli dve tabuľky.

Užívateľ môže pracovať s králikom, ktorého on v databáze vytvoril, alebo mu ho iný užívateľ predal, teda je majiteľom tohto králika. Tabuľka c*udzie_králiky* definuje ďalších králikov, s ktorými môže príslušný užívateľ pracovať. Táto tabuľka bola vytvorená preto, aby chovateľ mohol zostaviť rodokmeň každého králika, čo vlastní. Ak jeden užívateľ predá králika druhému užívateľovi, okrem zmeny položky majiteľ u predávaného králika sa do tabuľky c*udzie_králiky* zapíše vzťah predok predávaného králika[27] – kupujúci a vzťah predávaný králik – predávajúci. Užívateľ teda nemá prístup k tabuľke *králik*, ale je definovaný pohľad *moje_králiky*, ktorý vyfiltruje len prístupné záznamy. Obdobne sú pre tabuľky *vrh* a *ocenenie*, definované pohľady *moje_vrhy* a *moje_ocenenia*, ktoré tiež pracujú s tabuľkou *cudzie_králiky*. Ostatné tabuľky sú chovateľom prístupné celé.

[27] Pre každého predka jeden vzťah

Druhou zmenou oproti lokálnej databáze je tabuľka *user_version*. V nej sú ku každému užívateľovi, uložené informácie ohľadom poslednej synchronizácie. Bližšie informácie k správe verzií dát je v popise implementácie aplikačnej vrstvy.

Oracle na rozdiel od SQLite podporuje uložené procedúry. Ku každej tabuľke v databáze bol vytvorený balíček s procedúrami *vloz()*[28], *zmen()* a *zmaz()*, ktoré implementujú príslušné operácie s tabuľkami. Z bezpečnostných dôvodov namiesto pridelenia priamych práv INSERT, UPDATE a DELETE na príslušnú tabuľku, bolo pridelené právo EXECUTE na tento balíček funkcií. Okrem týchto procedúr boli vytvorené ďalšie, pre jednoduchšiu prácu s databázou, k tabuľke králik je to napríklad funkcia *predaj_králika()*. Taktiež bol vytvorený balíček *PKG_admin* s funkciami pre správu užívateľov.

Zväzová databáza

Databáza zväzu je veľmi podobná databáze klubu. Sú na ňu kladené rovnaké požiadavky, preto bola opäť zvolená databáza Oracle. K zväzovej databáze nie je implementovaná správa užívateľov, v podobe administrátorského balíčka funkcií, pretože sa predpokladá len prezentačný charakter webovej aplikácie. Správa užívateľov môže byť neskôr pridaná, keď to bude potrebné.

Zväzová databáza bola odvodená od klubovej. Stĺpce *deleted* a *version* potrebné pre synchronizáciu boli odstránené, taktiež aj tabuľka *user_version*. Na druhej strane do každej tabuľky bol pridaný stĺpec *klub_id* a bola vytvorená tabuľka *klub*, kde sú uložené informácie o klube, prihlasovacie údaje a adresa klubovej databázy. Pre prezentáciu údajov vo webovej aplikácii boli vytvorené pohľady. Ďalšie pohľady a tabuľky môžu byť dodefinované.

4.2 Aplikačná vrstva

Aplikačný server

Aplikačnú vrstvu v našom systéme reprezentuje aplikačný server. Bola uprednostnená forma služby operačného systému, pred konzolovou aplikáciou, pretože nie je potrebná žiadna interakcia s užívateľom. Úlohou aplikačného servera je distribuovať dáta z databázy prezentačnej vrstvy. Komunikácia medzi serverom a prezentačnou vrstvou je zabezpečená .NET remotingom a komunikácia medzi serverom a databázovou vrstvou je zabepečená Data providerom.

Pre .NET remoting bol zvolený Http kanál, pretože sa bude využívať aj v prostredí Internetu v spojení s IIS (webová aplikácia). Pre serializáciu dát bol vybraný *Binary formatter*. Aktivácia objektu je typu CAO. Toto riešenie sme zvolili preto, že volanie

[28] Názvy funkcií sú uvádzané bez parametrov

zo strany klienta môže byť v rámci nejakej databázovej transakcie a v tom prípade potrebujeme mať otvorené pripojenie k databáze po celú dobu trvania transakcie. Nestavový *Single call* teda nepripadá vôbec do úvahy. A keďže každý užívateľ pristupuje ku svojej schéme, musí existovať pripojenie k databáze pre každého užívateľa zvlášť, tým sa vylučuje možnosť použiť *Singleton*. Zmeny konfigurácie .NET remotingu sú možné prostredníctvom konfiguračného súboru.

CommunicationObject reprezentuje vzdialený objekt .NET remotingu. Ide o objekt typu *MarshallByReference*. Tento objekt implementuje funkcie pre prístup k databáze a zápis do log súboru. Na komunikáciu s databázou je použitý Oracle Data Provider, ktorého implementovala spoločnosť Oracle.

Keďže hlavne pri synchronizácii databáz je pravdepodobná dlhšia prestávka medzi jednotlivými volaniami *CommunicationObject*u, obnova lifetime hodnoty týmto spôsobom je nevyhovujúca. Preto je lifetime hodnota obnovovaná na strane klienta pomocou časovača, tento časovač je zapnutý po dobu práce s objektom a následne je vypnutý. *CommunicationObject* tiež implementuje časovač, aby po prípadnom prerušení spojenia mohol bezpečne zatvoriť pripojenie k databáze a zavolať funkciu *Garbage collectora*. Interval serverového časovača je menší ako interval u klienta a taktiež hranica pre obnovu lifetime hodnoty je podstatne vyššia ako hranica pre zánik objektu.

Synchronizácia databáz

Ako bolo spomínané skôr, dáta z lokálnych databáz chovateľov klubu sú centralizované v klubovej databáze. Toto by nebolo možné bez synchronizácie týchto databáz. Pri implementácii klientskej aplikácie[29] sa zvažovalo kedy a ako synchronizovať. Ponúkali sa nasledujúce možnosti.

- Pracovať sa bude nad jednou databázou. Synchronizácia prebehne pred začiatkom a po skončení práce. Primárna práca nad klubovou databázou je podmienená existenciou aktívneho pripojenia k internetu. Inak by sa pracovalo nad lokálnou databázou.

- Pracovať sa bude nad oboma databázami naraz. Synchronizácia prebehne pred začiatkom práce a čiastočne odpadne potreba synchronizovať po skončení práce[30].

Keďže práca nad oboma databázami naraz nám nevyrieši problém vzniknutia konfliktov, rozhodli sme sa primárne pracovať nad lokálnou databázou

[29] Synchronizácia je implementovaná na strane klienta, keďže server nepozná lokálnu databázu
[30] V skutočnosti nad klubovou databázou pracujú viacerí užívatelia naraz, preto stále bude potrebná synchronizácia smerom z klubovej databázy do lokálnej.

a synchronizácia prebehne výlučne na požiadanie užívateľa. Konflikty sa budú riešiť až pri synchronizácii.

Pre potreby synchronizácie bol každej tabuľke pridaný stĺpec *deleted*. Lokálna klubová databáza používa logický delete namiesto fyzického zmazania záznamov. Tento stĺpec zabezpečuje odlíšenie zmazaných záznamov od ostatných. Pri práci s lokálnou a zväzovou databázou sú zmazané riadky ignorované[31].

Aby mohol synchronizačný mechanizmus efektívne pracovať, potrebuje vedieť, ktoré záznamy boli od poslednej synchronizácie zmenené. Pre tento problém bol implementovaný verzovací systém dát. Každý riadok v každej synchronizovanej tabuľke obsahuje záznam v stĺpci *version*. V klubovej databáze existuje sekvencia - číslo, ktoré sa pred každou synchronizáciou inkrementuje. Zmenené a pridané záznamy dostanú potom toto číslo verzie. V tabuľke *user_version* je ku každému chovateľovi zaznamenané číslo verzie poslednej synchronizácie. Pred začiatkom synchronizácie sa mechanizmus synchronizácie pozrie do tejto tabuľky a pre synchronizáciu vyberie len záznamy ktoré majú vyššie číslo verzie ako číslo verzie poslednej synchronizácie príslušného užívateľa. Pri práci s lokálnou databázou mimo synchronizáciu sa zmeneným a pridaným záznamom nastaví číslo verzie na *null*. Pri synchronizácii z lokálnej databázy do centrálnej sa vyberú len záznamy, ktoré nemajú žiadne číslo verzie.

Pri práci na našom systéme môžu vzniknúť konflikty. Tak ako celá synchronizácia aj riešenie konfliktov prebieha automaticky na pozadí aplikácie, bez zásahu užívateľa. Konflikty sú riešené nasledujúcim algoritmom.

1. Ak sú primárne kľúče rôzne, zmení sa primárny kľúč v lokálnej databáze podľa kľúča v centrálnej databáze.

2. Primárne kľúče sú už rovnaké.
   I. Ak je záznam z tabuľky "králik", "vrh", alebo "ocenenie", replikujú sa údaje z lokálnej databázy.
   II. Ak je záznam v centrálnej databáze zmazaný a v lokálnej nie, replikujú sa údaje z lokálnej databázy.
   III. Inak sa replikujú údaje z centrálnej databázy.

Synchronizácia databáz prebieha v samostatnom vlákne aplikácie.

Synchronizácia prebieha automaticky pred predajom a po predaji králika. Predaj králika je jediná funkcia, ktorá nie je prístupná z offline režimu.

[31] Okrem synchronizácie databáz.

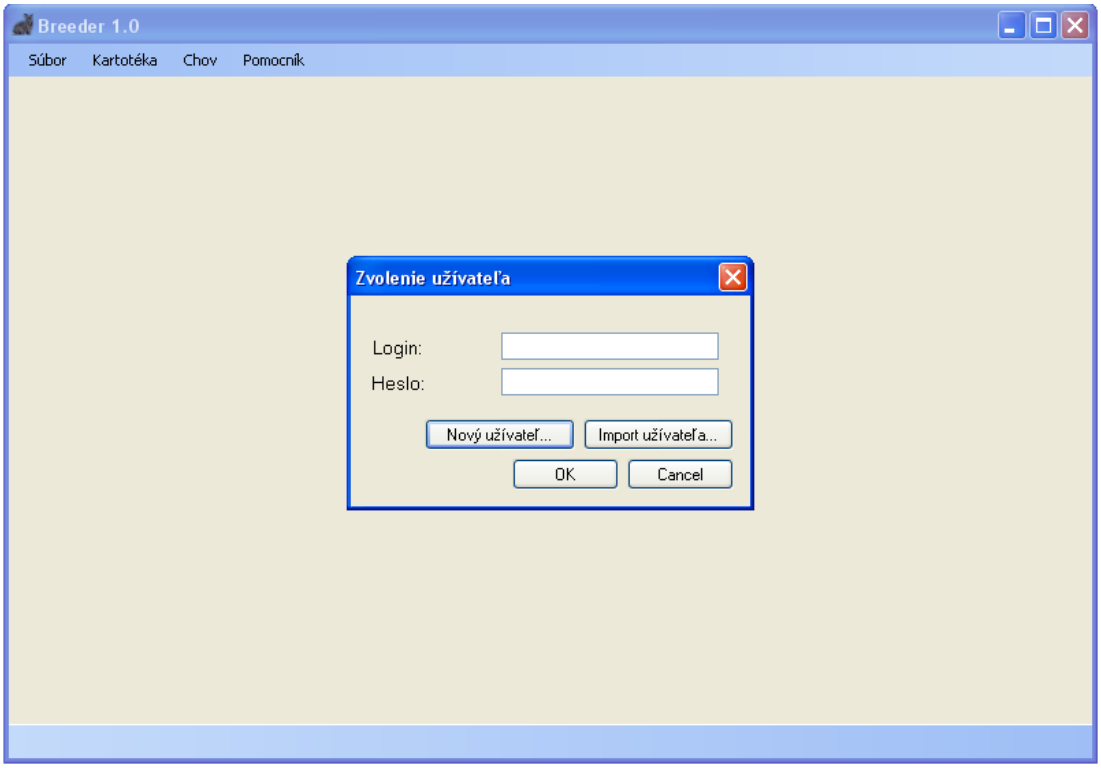
Celá synchronizácia je vykonávaná v rámci jednej transakcie na lokálnej databáze a v rámci druhej na centrálnej databáze. Pri výnimke nastane rollback oboch transakcií.

4.3 Prezentačná vrstva

Prezentačnú vrstvu reprezentuje užívateľské rozhranie klientskej aplikácie chovateľa a webovej aplikácie zväzu. Pri implementácii celého systému bol kladený dôraz na vytvorenie korektnej architektúry a databázového základu. Týmto utrpela funkcionalita prezentačnej vrstvy, ktorej implementácia nedostala dostatok priestoru. To sa týka najmä zväzovej aplikácie. Keďže však ide o webové rozhranie, vďaka dobrému základu, dodatočná implementácia funkcionality by nemala byť veľkým problémom.

Klientská aplikácia

Aplikácia pre svoju prácu potrebuje určiť, s ktorým databázovým súborom sa bude pracovať. Výber tohto súboru je riešený prihlásením užívateľa[32], ktorý je asociovaný s týmto súborom. Táto asociácia je po vytvorení užívateľa zapísaná v Xml súbore.



Obrázok 4.3 Klientská aplikácia - zvolenie užívateľa

[32] Obrázok 4.3

Klientská aplikácia, okrem prezentačných vlastností, tiež preberá úlohu aplikačnej vrstvy pre prácu s lokálnou databázou. SQLite databázový systém je od výrobcu dostupný vo forme konzolovej aplikácie a tiež sú dostupné rôzne knižnice, pre prácu s databázou. My sme použili knižnicu System.Data.SQLite[33], ktorá implementuje Data provider pre ADO.NET. Pre vytvorenie schémy je použitá pôvodná konzolová aplikácia, keďže Data provider nepodporuje vytvorenie schémy z pripraveného súboru. Pre tabuľky, ktoré na zobrazenie dát nepotrebovali SQL operáciu JOIN bol vytvorený typovaný *DataSetBreeder*, pre ostatné musel byť použitý netypovaný *DataSet*. Typovaný *DataSet* nedokáže uložiť dáta spojených tabuliek.

Užívateľské rozhranie je vytvorené pomocou *Windows Forms*. Zobrazenie dát z databáze je implementované pomocou komponenty *DataGridView*[34], ktorá má formu tabuľky s podporou radenia riadkov. Táto komponenta podporuje aj priamu editáciu zobrazovaných riadkov, táto funkcia však nebola využitá a bola nahradená modálnymi dialógmi na vkladanie, zobrazovanie a editáciu dát.

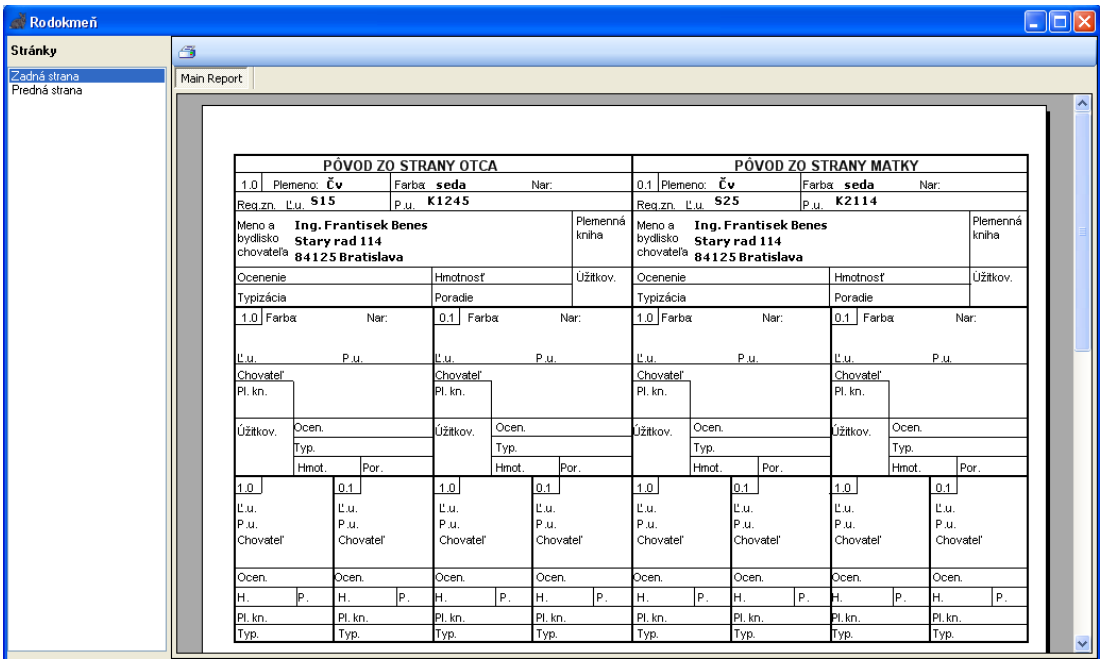
Obrázok 4.4 Klientská aplikácia - zobrazenie chovateľov

Nastavenia aplikácie boli uložené v konfiguračnom Xml súbore. Pre správu konfigurácie bola vytvorená trieda *ConfigurationClass*.

Špecifickou funkciou aplikácie je podpora tlače rodokmeňa[35] a pripúšťacieho potvrdenia[36] králika. Dokument je zostavený z dát prítomných v databáze a je doplnený do prednastavenej šablóny. Šablóna bola vytvorená nástrojom *Crystal Reports*, ktorého obmedzená verzia je súčasťou vývojového prostredia *Microsoft Visual Studio 2005*. Táto šablóna je pevne daná a užívateľ ju nemôže meniť. Toto obmedzenie môžeme obhájiť tým, že šablóna Slovenského zväzu chovateľov nepodlieha častým zmenám. *Crystal Reports* podporujú typovaný *DataSet* ako zdroj dát odkiaľ dokážu čerpať. Preto bol vytvorený typovaný *DataSetPrint* pre tento účel. Na zobrazovanie dokumentov bola použitá komponenta *CrystalReportsViewer*. Táto

[33] Knižnica je dostupná na http://sqlite.phxsoftware.com
[34] Obrázok 4.4
[35] Príloha č. 2, Obrázok 3 a Obrázok 4
[36] Príloha č. 2, Obrázok 1 a Obrázok 2

komponenta je vytvorená v samostatnom okne aplikácie[37], pretože zobrazenie prvého dokumentu trvá istú dobu. Takto po zobrazení ďalšieho dokumentu nemusíme znova čakať, kým sa vytvorí inštancia *CrystalReportsViewer*a.



Obrázok 4.5 Klientská aplikácia - zobrazenie rodokmeňa

Webová aplikácia

Na implementáciu webovej aplikácie bola použitá technológia ASP.NET a jazyk C#.

Aplikácia si každý deň v stanovenú dobu stiahne dáta z klubových databáz do svojej databázy. Toto je zabezpečené skriptom, ktorý je pri umiestnení na webový server, zavedený do plánovača úloh. Skript pre každý záznam v tabuľke klub vytvorí pripojenie, stiahne potrebné dáta a uvoľní pripojenie pre ďalší záznam. Zväzová databáza nedefinuje kandidátne kľúče. Tabuľka plemeno sa nesťahuje z klubových databáz, práve naopak, pred sťahovaním dát z klubu sa upravia primárne kľúče v klubovej tabuľke plemeno podľa zväzovej tabuľky. Konflikty teda môžu nastať len v prípade rovnakých GUIDov, čo je však veľmi nízka pravdepodobnosť, preto sa taký konflikt ignoruje.

Užívateľské rozhranie je implementované pomocou *Web Forms*. Na zobrazenie dát je použitá komponenta *GridView*, ktorá zobrazuje dáta pohľadov definovaných v databáze[38]. Zobrazené dáta je možné filtrovať podľa prednastavených filtrov[39].

[37] Obrázok 4.5
[38] Obrázok 4.6
[39] Obrázok 4.7

Priezvisko	Titul	Ulica	Číslo popisné	Mesto	PSČ	Krajina	Mail	Telefón	Mobil	Klub
Benes	Ing.	Stary rad	114	Bratislava	84125	Slovensko				Klub cincily velkej
Novak		Stefanikova	45	Bratislava	84545	Slovensko	ja@ty.my		0544454545	Klub cincily velkej
Novy	Mgr.	Dolna	12	Martin	03601	Slovensko	novy@my.jj	44577881	0547823145	Klub cincily velkej
Klub1	DUMMY	DUMMY	11	Kosice		Slovensko	DUMMY@dummy.dummy			Klub velkych svetlych striebornych
Klub1	DUMMY	DUMMY	12	Presov		Slovensko				Klub velkych svetlych striebornych
Klub2	DUMMY	DUMMY	21	Martin		Slovensko	DUMMY@dummy.dummy			Klub KANINO
Klub2	DUMMY	DUMMY	22	Nitra		Slovensko				Klub KANINO
Klub3	DUMMY	DUMMY	31	Secovce		Slovensko	DUMMY@dummy.dummy			Klub belgickych obrov
Klub3	DUMMY	DUMMY	32	Dolny Kubin		Slovensko				Klub belgickych obrov

Obrázok 4.6 Webová aplikácia - zobrazenie chovateľov

Priezvisko	Titul	Ulica	Číslo popisné	Mesto	PSČ	Krajina	Mail	Telefón	Mobil	Klub
Benes	Ing.	Stary rad	114	Bratislava	84125	Slovensko				Klub cincily velkej
Novak		Stefanikova	45	Bratislava	84545	Slovensko	ja@ty.my		0544454545	Klub cincily velkej
Novy	Mgr.	Dolna	12	Martin	03601	Slovensko	novy@my.jj	44577881	0547823145	Klub cincily velkej

Obrázok 4.7 Webová aplikácia - vyfiltrovaný chovatelia Klubu činčily veľkej

Treba podotknúť, že záznamy chovateľov, ktorí pracujú len v offline režime klientskej aplikácie a nepublikujú svoje záznamy, zväzová aplikácia nemá šancu získať a vyhodnotiť. Preto prezentované dáta budú v istej miere skreslené. V praxi však tiež nie všetky kluby a chovatelia poskytujú zväzu dostatočné údaje.

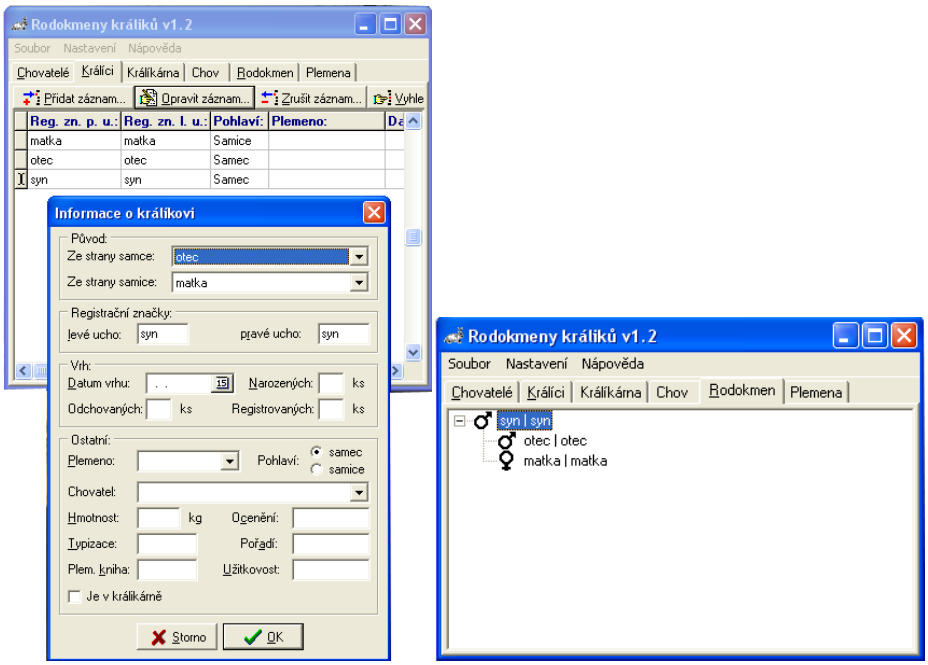
5 Porovnanie s podobnými produktmi

Problematika evidencie bola špecifikovaná na Slovenský zväz chovateľov, nemá teda veľký význam porovnávať systém s produktmi iných krajín. Keďže však podobný systém na Slovensku neexistuje, pokúsim sa o porovnanie s podobným systémom v Českej republike.

Rodokmeny králíků[40]

Aplikácia nepodporuje viac užívateľov, hneď po spustení sme vpustení do databázy. Ukladať si môžeme dáta o králikoch, chovateľoch a vrhoch. Plemeno si užívateľ zadefinuje, program neposkytuje preddefinované plemená. Rozsah vedenej evidencie je presne prispôsobený formátu českého rodokmeňa, nič viac, nič menej. Databáza je rozložená do viacerých súborov, program umožňuje export a import údajov do súboru, avšak len po jednom zázname. Registračné značky nie sú formátované, teda užívateľ si tam môže zadať prakticky čokoľvek. Výhodou je jednoduchý náhľad na zostavený rodokmeň. Tlač rodokmeňa je možný po zakúpení licencie, program tlačí vyplnené rodokmene a pripúšťacie potvrdenia podľa vzoru tlačív Českého svazu chovatelů.

Aplikácia Rodokmeny králíků je databázovou aplikáciou určenou pre chovateľov k evidencii chovaných jedincov a k tlači vyplnených formulárov pripúšťacích potvrdení a rodokmeňov[41]. Z uvedeného vyplýva, že aplikácia nedosahuje parametre nášho systému, preto porovnaniu netreba prikladať veľký význam.



[40] Aplikácia je dostupná na adrese http://www.sweb.cz/ro_kr/.
[41] Ako je uvedené na http://www.sweb.cz/ro_kr/.

6 Záver

Cieľom bakalárskej práce bolo navrhnúť riešenie pre vedenie evidencie chovu králikov. Bola vypracovaná analýza a stanovenie základných hraníc problému[42]. Boli sme si vedomí, že tieto hranice nemusia byť definitívne. Po preniknutí do hĺbky a vypracovaní projektu môžeme zhodnotiť, že sme sa od celkovej predstavy trochu vychýlili. V prvotnom návrhu sa nepočítalo s problematikou synchronizácie databáz, zostavenia tlačových zostáv, či zamedzenia prístupu k záznamom ostatných chovateľov klubu. Vyriešenie týchto problémov bolo časovo náročné, čo sa odrazilo na rozhodnutí neimplementovať úlohy registrátora a poradcu chovu. Je samozrejmé, že sme naším projektom nemohli pokryť celú problematiku, preto tieto úlohy a ďalšie funkcie odporúčame zahrnúť do najbližšieho rozšírenia systému.

Treba však poznamenať, že táto aplikácia by mala byť prínosom pre zväz, kluby ako aj pre samotných chovateľov, keďže podobný projekt na Slovensku ešte neexistuje. Systém umožňuje chovateľom viesť si evidenciu o svojom chove a zdieľať niektoré informácie s ostatnými chovateľmi klubu. Zväz má možnosť z dát dodaných od klubov, zostaviť štatistiky a prehľady, z ktorých nielen slovenský chovateľ získa prehľad o chove králikov na našom území.

[42] Príloha č. 1 : *Prvotné stanovenie hraníc systému – hlbšia analýza*

Literatúra

[1] Haspra P. a kol. (1996): *Registračný poriadok pre registráciu a tetovanie králikov v Slovenskej republike*. Ústredná odborná komisia pre chov králikov a kožušinových zvierat, Bratislava.

[2] Rammer I. (2002): *Advanced .NET Remoting*, Apress, Berkeley.

[3] Microsoft: *Microsoft Developer Network*, http://msdn.microsoft.com/

[4] Oracle: *Oracle documentation*,http://www.oracle.com/technology/documentation

Prílohy

Príloha č. 1: Prvotné stanovenie hraníc systému – hlbšia analýza

Program: Chovateľ králikov (aplikácia na úrovni informačného systému)

Na klubovej úrovni:

Chovateľ:

-- kartotéka chovaných králikov(jednotlivé králiky sú na seba naviazané)

-- pridanie záznamu – plemeno (skratka plemena – jednoznačne určuje plemeno)

- farba (u niektorých plemien bezvýznamná)
- registračné značky
  - pravé ucho (typ chovu – (línia) - číslo)
  - ľavé ucho (označenie štátu – mesiac – rok narodenia)
- pohlavie (samec - 1.0, samica - 0.1)
- dátum narodenia (dátum)
- údaje o vrhu – narodených (počet)
  - odchovaných (počet)
  - registrovaných (počet)
- meno a adresa chovateľa
- registračné značky otca
- registračné značky matky
- hmotnosť (u dospelých, reálne číslo)
- ocenenie (u dospelých vystavovaných, číslo 0 - 100)
- typizácia (u dospelých typizovaných, znakový kód určujúci parametre králika)

-- oprava záznamu - oprava jednotlivých atribútov

-- zrušenie záznamu – zrušenie (archivovanie) záznamu so všetkými atribútmi

- archivácia je dôležitá, aby sa nestratila väzba v rodokmeňoch
- z archívu je králik vyradený po určitom čase (niekoľkých rokoch)

-- zobrazuje chov králikov v aktuálnej sezóne (príp. archívne záznamy)

- filtre

-- zostavenie rodokmeňa

- vytvorenie rodokmeňa pomocou odkazov na matku a otca

-- tlač rodokmeňa

- vytvorenie tlačovej zostavy
- tlač

-- odoslanie informácii klubovému Registrátorovi
- odoslanie žiadosti o pridelenie registračných značiek
- dátum vrhu
- dátum pripustenia
- registračné značky otca
- registračné značky matky
- pohlavie vrhnutých mláďat
- údaje o vrhu – narodených (počet)
- odchovaných (počet)
- registrovaných (počet)
- plemeno
- farba
- chovateľ

-- odoslanie informácii klubovému Poradcovi chovu
- odoslanie údajov o chovných samcoch
- pôvodný majiteľ
- registračné značky
-registračné značky otca
-registračné značky matky

-- náhľad na ostatné chovy s klubu (bez možnosti zapisovania)
- filtre

-- **...**

Registrátor:

-- prístup k celej databáze s možnosťou prideliť (zapísať) registračné číslo

-- vedenie registračných hárkov
- uložené údaje o registrovaných králikoch získane od chovateľov
- poradie vrhu
- poradie králika
- chovateľ
- dátum registrácie
- dátum pripustenia
- dátum vrhu
- registračné značky
- registračné značky otca
- registračné značky matky

Poradca chovu:

-- rozhoduje o jednotlivých líniách (definuje nové línie podľa aktuálnych samcoch) a tieto informácie zasiela Registrátorovi
- určí líniu podľa otca daného králika, alebo priradí novú líniu (otec je bez línie, alebo daný králik je zo zahraničia), špeciálna línia -„bezlínie”, línie (čísla)

-- na konci sezóny zhodnotí aktuálnu sezónu, kvalitu jednotlivých línii
    - slovné ohodnotenie z pohľadu posudzovateľa (v jednotlivých pozíciách oceňovacieho lístka)
-- stanovuje termín klubovej výstavy

Na zväzovej úrovni: (webové rozhranie)

UV (Slovenského) zväzu chovateľov
-- prístup k informáciám jednotlivých klubov a ich zosumarizovanie
    - poskytuje výpis údajov vybraných (najlepších) králikov

Administrátor
-- pridanie chovateľa
    - registrácia osobných údajov
        - meno a priezvisko
        - adresa
        - login
        - heslo

-- odobratie chovateľa
-- prideľovanie práv
    - definovanie Registrátora, Poradcu chovu

Autorizačný systém
-- vpustenie do databáze
    - požaduje login a heslo
-- zamietnutie prístupu

Príloha č. 2: Šablóny dokumentov

Prihlasujem k zápisu do registračnej knihy mláďatá uvedených rodičov a žiadam o pridelenie tetovacích značiek

Samica párená dňa: .....

Nezostala kotná – párená dodatočne dňa: .....

Mláďatá vrhnuté dňa: .....

Vypĺňuje chovateľ			Vypĺňuje oblastný registrátor	
			Tetovacie značky	
Č. bež.	* Pohlavie mláďat	Farba	ľavé ucho	pravé ucho
1.				
2.				
3.				
4.				
5.				
6.				
7.				
8.				

* Pohlavie označte u samca 1.0, u samice 0.1. Samca píšte vopred

Podpis svedka pripustenia*** .....

ZO SZCH obdržala od chovateľa dňa .....

*) Chovateľ je ten, u koho sa zviera narodilo.
**) Adresa musí byť presná s udaním PSČ.
***) Vlastnoručný podpis

SLOVENSKÝ ZVÄZ CHOVATEĽOV
ODBORNÁ KOMISIA
PRE CHOV KRÁLIKOV

PRIPÚŠŤACIE POTVRDENIE KRÁLIKA

Prihlasujem k registrácii:

Plemeno: .....

Farba: .....

Narodených .. ks, odchov ..... ks, registrovaných ..... ks

Chovateľ* .....

Adresa ** .....

Číslo členskej legitimácie .....

Pečiatka OV SZCH „Registrácia"

Pečiatka ZO SZCH „Registrácia"

.....
Dátum zápisu do registr. knihy
(Vyplní oblastný alebo klubový registrátor)

Obrázok 1 Pripúšťacie potvrdenie králika - predná strana

PÔVOD ZO STRANY OTCA				PÔVOD ZO STRANY MATKY			
1.0 Plemeno: Farba: Nar.:				0.1 Plemeno: Farba: Nar.:			
Reg. zn. Ľ. u. P. u.				Reg. zn. Ľ. u. P. u.			
Meno a bydlisko chovateľa			Plemenná kniha	Meno a bydlisko chovateľa			Plemenná kniha
Ocenenie	Hmotnosť	Úžitkovosť		Ocenenie	Hmotnosť	Úžitkovosť	
Typizácia	Poradie			Typizácia	Poradie		
1.0 Farba: Nar.:	0.1 Farba: Nar.:			1.0 Farba: Nar.:	0.1 Farba: Nar.:		
Ľ. u. P. u.	Ľ. u. P. u.			Ľ. u. P. u.	Ľ. u. P. u.		
Chovateľ Pl.kn.	Chovateľ Pl.kn.			Chovateľ Pl.kn.	Chovateľ Pl.kn.		
Úžitkovosť Ocen. Typ. Hmot. Por.	Úžitkovosť Ocen. Typ. Hmot. Por.			Úžitkovosť Ocen. Typ. Hmot. Por.	Úžitkovosť Ocen. Typ. Hmot. Por.		
1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ	1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ	1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ	1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ
Ocen.	Ocen.	Ocen.	Ocen.	Ocen.	Ocen.	Ocen.	Ocen.
H. P.	H. P.	H. P.	H. P.	H. P.	H. P.	H. P.	H. P.
Pl. kn.	Pl. kn.	Pl. kn.	Pl. kn.	Pl. kn.	Pl. kn.	Pl. kn.	Pl. kn.
Typ.	Typ.	Typ.	Typ.	Typ.	Typ.	Typ.	Typ.

Obrázok 2 Pripúšťacie potvrdenie králika - zadná strana

TYPIZÁCIA:

Do rodokmeňa sa bude uvádzať iba podľa propozícií vydaných klubom a schválených odbornou komisiou pre chov králikov ÚV SZCH.

PORADIE:

Sa uvádza arabskými číslami, lomenými rokom. Udáva v príslušnom roku poradie 10 najlepších králikov určitého plemena, pre ktoré je klub založený.

PLEMENNÁ KNIHA CHOVATEĽSKÝCH KLUBOV

PK - plemenná kniha do ktorej sa budú môcť zapísať zvieratá, ktorých súrodenecké línie budú vyhovovať požiadavkám klubu
E - elitná plemenná kniha je pre zvieratá, ktorých rodičia sú už zapísaný v PK. Ďalšou podmienkou je predvedenie najmenej dvoch súrodeneckých línií potomkov, zodpovedajúcich najmenej 93 bodom.

ÚŽITKOVOSŤ:

Sa bude uvádzať dvomi číslami, a to: prvé číslo znamená počet vrhov od samice, alebo počet pripustených samíc príslušným samcom. Druhé číslo znamená počet narodených mláďat od príslušného samca.
Uvádzame príklad: v rodokmeni samice zapísané číslo 1/8 znamená, že ide o prvý vrh s 8 narodenými mláďatmi. Označenie číslo 3/20 u samca znamená že po príslušnom samcovi sa narodilo v troch vrhoch 20 mláďat.

UPOZORNENIE:

Je bezpodmienečne potrebné v rodokmeni králika uvádzať ocenenie a jeho hmotnosť. Podmienka ocenenia sa uvádza už v registračnom poriadku a preto je neprípustné registrovať a tetovať mláďatá od neocenených rodičov. Správnosť a pravdivosť údajov v tomto rodokmeni potvrdzuje registrátor základnej organizácie. Rodokmeň králika je úradná listina a každé zneužitie je trestné a znamená ukončenie členstva vo Zväze chovateľov

ZMENA MAJITEĽA KRÁLIKA:

.....
.....
.....
.....

sídlo ZO dátum pečiatka a podpis registrátora ZO

SLOVENSKÝ ZVÄZ CHOVATEĽOV
ODBORNÁ KOMISIA
PRE CHOV KRÁLIKOV

RODOKMEŇ KRÁLIKA

Samec - samica Farba .....
Plemeno .....
Registr. značky:
Ľavé ucho: .....
Pravé ucho: .....
Narodený (á): .....
narodených: ..... ks, odchov ..... ks, registr. ..... ks

Meno a bydlisko chovateľa: .....
.....
.....

Ocenenie	Hmotnosť
Typizácia	
Poradie	

TETOVACIE ZNAČKY ľavého ucha

Bez pečiatky neplatné

Dňa: .....

..... Podpis registrátora OV

TETOVACIE ZNAČKY pravého ucha

Obrázok 3 Rodokmeň králika - predná strana

PÔVOD ZO STRANY OTCA				PÔVOD ZO STRANY MATKY			
1.0 Plemeno: Farba: Nar.:				0.1 Plemeno: Farba: Nar.:			
Reg. zn. Ľ. u. P. u.				Reg. zn. Ľ. u. P. u.			
Meno a bydlisko chovateľa			Plemenná kniha	Meno a bydlisko chovateľa			Plemenná kniha
Ocenenie	Hmotnosť		Úžitkovosť	Ocenenie	Hmotnosť		Úžitkovosť
Typizácia	Poradie			Typizácia	Poradie		
1.0 Farba: Nar.:	0.1 Farba: Nar.:			1.0 Farba: Nar.:	0.1 Farba: Nar.:		
Ľ. u. P. u.	Ľ. u. P. u.			Ľ. u. P. u.	Ľ. u. P. u.		
Chovateľ Pl.kn.	Chovateľ Pl.kn.			Chovateľ Pl.kn.	Chovateľ Pl.kn.		
Úžitkovosť Ocen. Typ. Hmot. Por.	Úžitkovosť Ocen. Typ. Hmot. Por.			Úžitkovosť Ocen. Typ. Hmot. Por.	Úžitkovosť Ocen. Typ. Hmot. Por.		
1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ	1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ	1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ	1.0 Ľ. u. P. u. Chovateľ	0.1 Ľ. u. P. u. Chovateľ
Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.	Ocen. H. P. Pl. kn. Typ.

Obrázok 4 Rodokmeň - zadná strana

Príloha č. 3: Obsah priloženého média

- **Aplikačný server** (priečinok Aplikacny_server)
    - Zdrojové kódy (priečinok src)
    - Archív s binárnymi súbormi a inštalátor serverovej služby (priečinok bin)
    - Programátorská dokumentácia
    - Oracle Data Provider (priečinok Prerekvizity)
- **Klientská aplikácia** (priečinok Chovatel_kralikov)
    - Zdrojové kódy (priečinok src)
    - Inštalátor (priečinok setup)
    - Testovacia verzia aplikácie s vloženými demonštračnými dátami (priečinok bin)
    - Programátorská dokumentácia
- **Schéma centrálnej databázy** (priečinok Centralna_databaza)
    - Skripty na vytvorenie a zmazanie schémy + popis vytvárania užívateľov
- **Webová aplikácia** (priečinok Webova_aplikacia)
    - Zdrojové kódy (priečinok src)
    - Programátorská dokumentácia
- **Schéma zväzovej databázy** (priečinok Zvazova_databaza)
    - Skripty na vytvorenie a zmazanie schémy
    - Skript na vloženie testovacích dát