Zápočtová úloha z předmětu KIV/ZSWI

OBJEKTOVÝ NÁVRH APLIKACE

pro Evidence výskytu UFO

18. 4. 2006

Tým: BFU

Členové:

Ladislav	Ašenbrener	troomar@students.zcu.cz
Martin	Sloup	arcao@arcao.com
Václav	Purchart	lipop@students.zcu.cz
Jan	Tichava	jtichava@students.zcu.cz
Miroslav	Hesoun	mhesoun@students.zcu.cz
Martin	Žibrický	matesz@students.zcu.cz

Obsah

1. ÚVOD 3

1.1 ÚČEL SYSTÉMU 3
1.2 SLOVNÍČEK DEFINIC, POJMŮ A ZKRATEK 3
1.3 ODKAZY NA DALŠÍ DOKUMENTY 3

2. KONTEXT A ARCHITEKTURA SYSTÉMU 4

2.1 KONTEXT SYSTÉMU 4
2.2 ARCHITEKTURA SYSTÉMU 4
2.3 SEKVENČNÍ DIAGRAMY VYBRANÝCH DŮLEŽITÝCH ČÁSTÍ 5
2.3.1 Přidat filtr 5
2.3.2 Zobraz graf 5
2.3.3 Editace položky 5
2.4 PŘEHLED PODSYSTÉMŮ 6
2.5 ZVOLENÝ PROGRAMOVACÍ JAZYK A DATABÁZE 6

3. TYPY INFORMACÍ ZPRACOVÁVANÉ SYSTÉMEM 6

3.1 DATABÁZE 6

4. PODSYSTÉMY 7

4.1 SERVER 7
4.2 JAVA KLIENT 7
4.2.1 Popis synchronizace databází 8
4.2.2 Třída SpravaDatabaze 9
4.2.3 Třída Graf 10
4.2.4 Třída DataProGrafKlient 10
4.2.5 Třída DataProGraf 11
4.2.6 Třída PrezentacniVrstva 11
4.2.7 Třída DocasnaDatabaze 11
4.2.8 Třída Main 11
4.3 WEB KLIENT 11

5. GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ A POPIS FUNKCE 12

5.1 SPRÁVA 12
5.1.1 Prohlížení 12
5.1.2 Editování 13
5.2 GRAFY 15
5.3 SYNCHRONIZACE 15
5.3.1 Synchronizace databáze 15
5.3.2 Nastavení 16

6. PŘIŘAZENÍ TŘÍD/MODULŮ PROGRAMÁTORŮM 16

1. Úvod

Tento dokument popisu návrh systému pro evidenci výskytu UFO. Stručně nastiňuje základní principy, na jakých funguje celý systém a zároveň obsahuje poměrně podrobné části, které jsou určené pro následnou implementaci.

1.1 Účel systému

Systém bude sloužit k zaznamenávání výskytu UFO nejen v České republice. Bude existovat centrální databáze úkazů, ve které bude možné vyhledávat podle mnoha kritérií, zobrazovat různé statistiky a ukázat polohu na jednoduché interní mapě.

1.2 Slovníček definic, pojmů a zkratek

databáze v paměti	zkopírovaná databáze ze souboru, nemusí být úplná

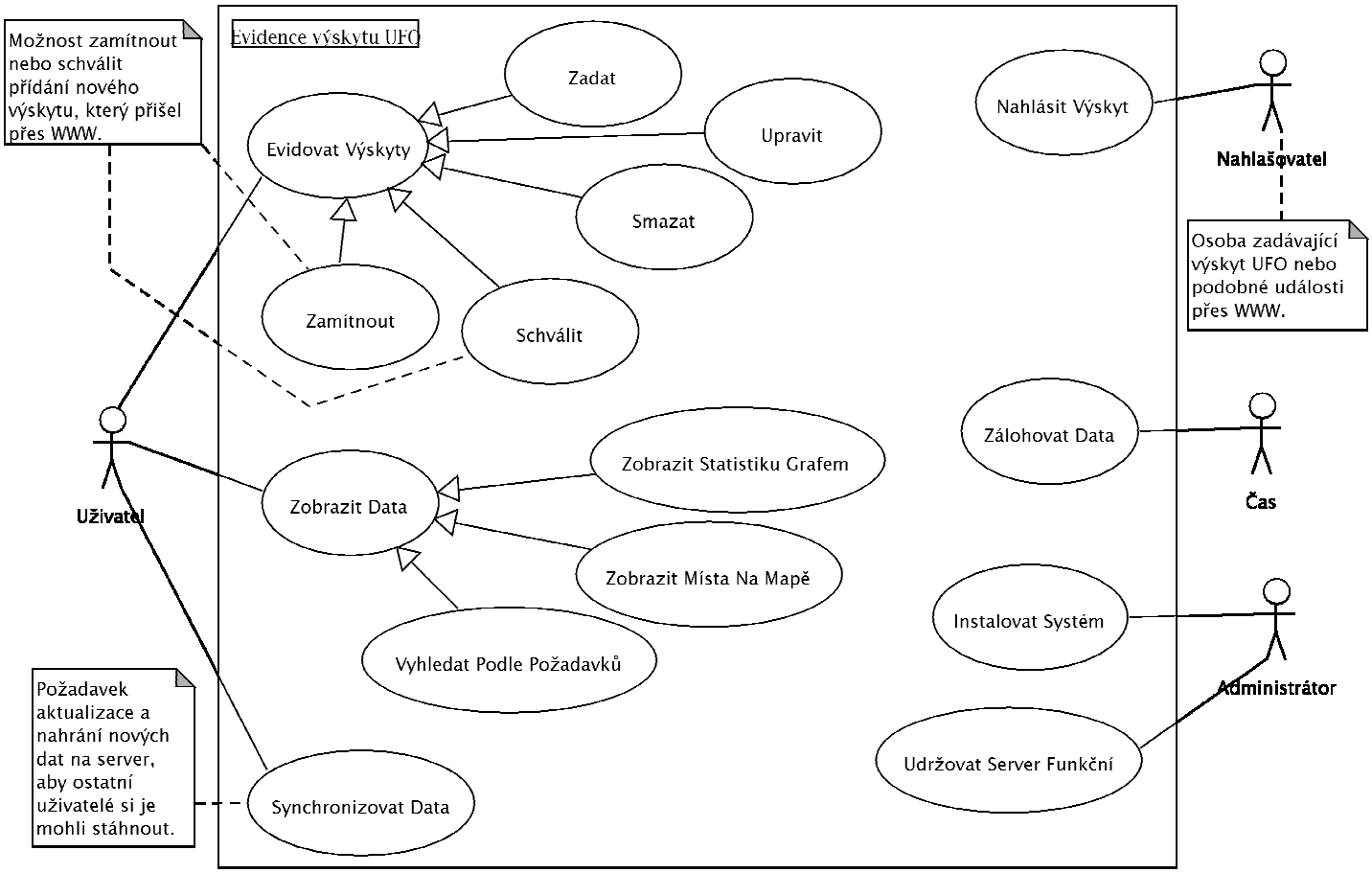
1.3 Odkazy na další dokumenty

Podrobný návrh databáze	http://zcu-zswi.arcao.com/materialy/databaze/db_report/
Náhled na GUI	http://home.zcu.cz/~jtichava/zswi/

Přílohy
Návrh databáze

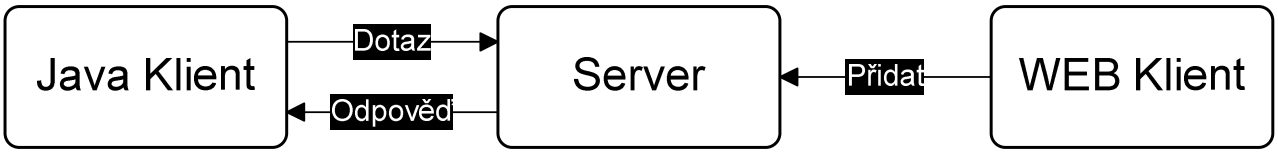
2. Kontext a architektura systému

2.1 Kontext systému



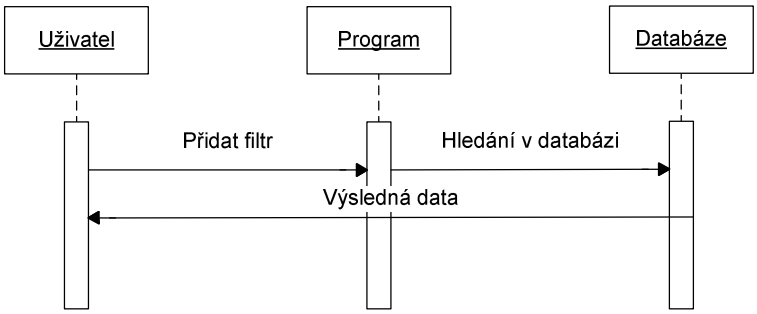
2.2 Architektura systému

Celý systém evidence bude rozdělen na tři základní podsystémy. Jedna část bude na straně serveru, která bude spravovat databáze od všech uživatelů. Druhá část je spuštěna na straně klienta, bude mít k dispozici celou databázi pro prohlížení bez připojení k síti. Třetí část slouží pouze pro přidávání dalších položek do databáze pomocí webového prohlížeče.

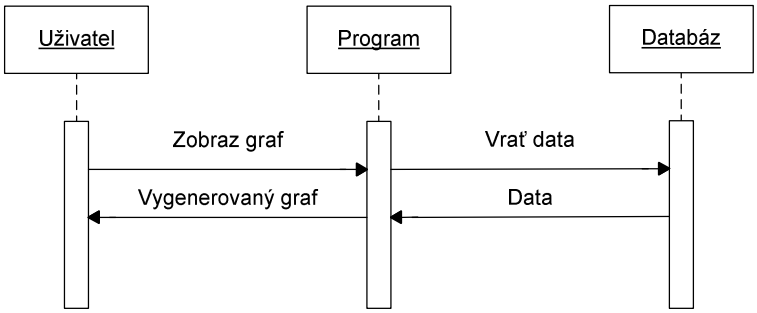


2.3 Sekvenční diagramy vybraných důležitých částí

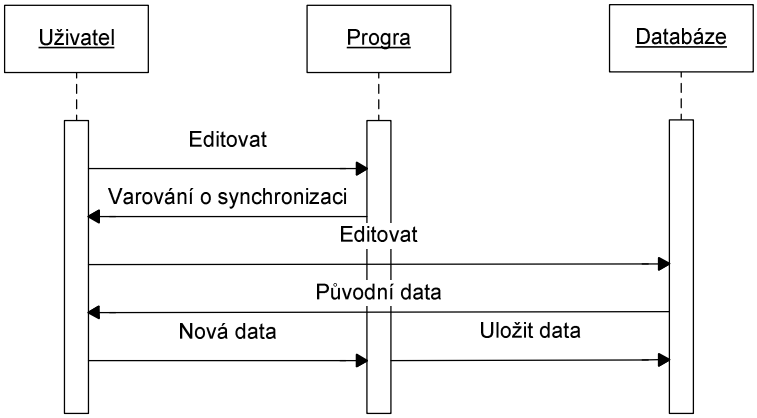
2.3.1 Přidat filtr



2.3.2 Zobraz graf



2.3.3 Editace položky



2.4 Přehled podsystémů

Na straně serveru bude pouze databáze, ve které budou obsažena data všech uživatelů. Databáze bude mít ř tabulek, které budou obsahovat záznamy, konkrétní podrobnosti v sekci 4.
Klient bude mít k dispozici celou databázi, která je uložena na serveru, lokálně. Bude možné její prohlížení bez připojení k sítí a celá databáze tím bude chráněna proti selhání serveru. Uživatel může požádat o synchronizaci dat v databázi.
Přes webový prohlížeč bude moci návštěvník webových stránek přidávat záznamy o vlastním pozorování, ale administrátor bude muset přidání záznamu schválit.

2.5 Zvolený programovací jazyk a databáze

Jako databáze se použije MySQL.
Klientská část bude napsána celá v programovacím jazyku Java. Jako výhodu je možné vyzdvihnout přenositelnost mezi různými platformami a nezávislost na operačním systému, cenou za tuto vlastnost je pomalejší běh aplikace.
Webové rozhraní pro vkládání údajů do databáze bude realizováno pomocí skriptovacího jazyka PHP, který bude spuštěn na webovém serveru Apache.

3. Typy informací zpracovávané systémem

3.1 Databáze

Spolu s programem je i soubor lokální databáze, který obsahuje kopii databáze na serveru.

Databáze obsahuje 5 tabulek:

- adresar – osoby, které vedly pozorování
- kruhy – databáze kruhů v obilí
- mista – seznam míst
- ufo_duch – různé výskyty UFO
- uzivatele – seznam uživatelů

Podrobnější popis tabulek včetně grafického znázornění viz:
http://zcu-zswi.arcao.com/databaze/navrh/

Na straně serveru bude jako databáze použito MySQL.
Na straně klienta

4. Podsystémy

4.1 Server

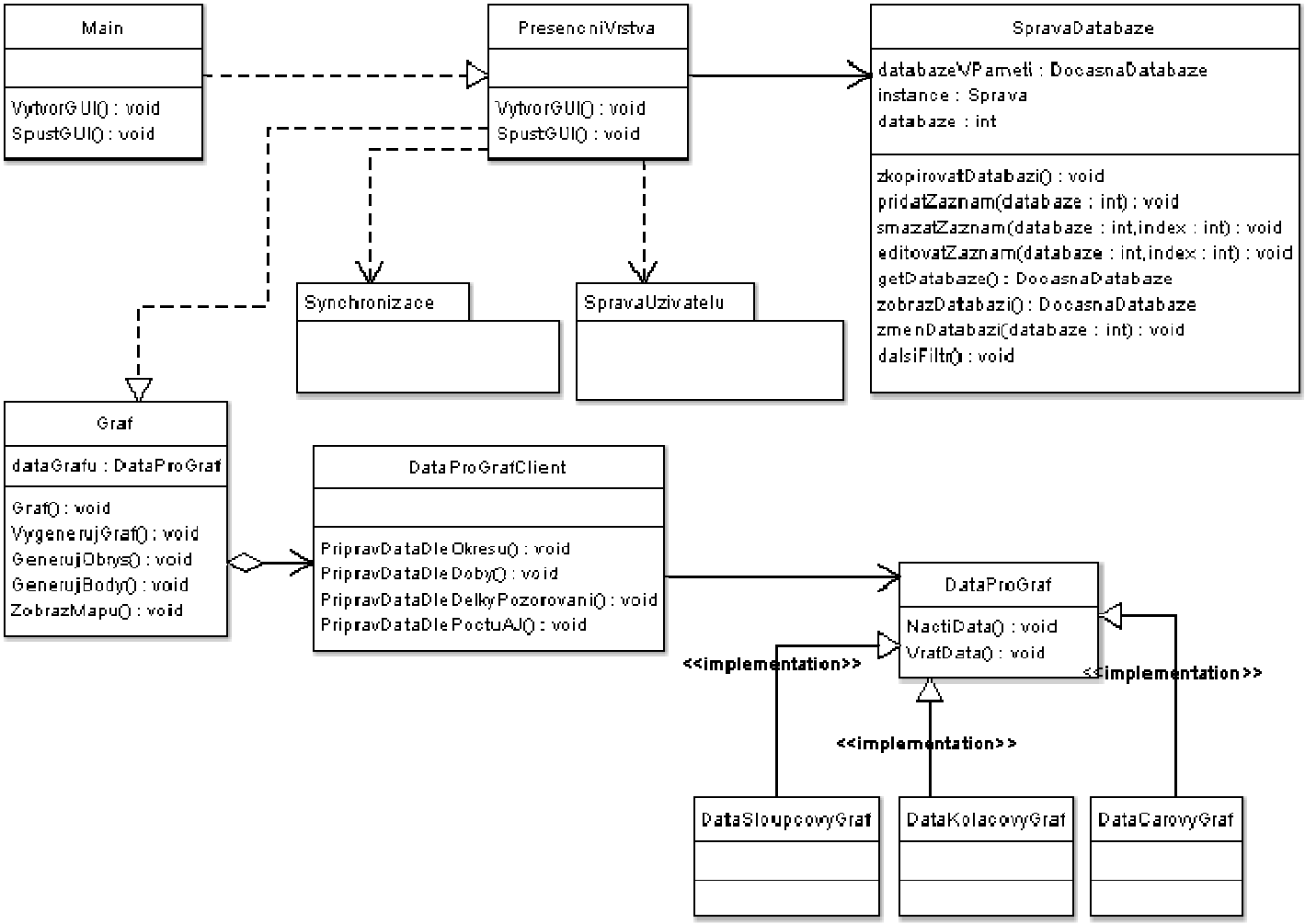
Na serveru poběží PHP 4 (nejlépe poslední existující verze této řady, ale není to podmínka) a MySQL 4.1 a novější, kde budou uchovávána data. Pro komunikace mezi Klientem a Serverem budou použity WebServices použitím protokolu Hessian[1], který je daleko šetrnější na přenesená data než, samotný xml formát a jeho parsování zabírá daleko méně času než parsování XML (všeobecně známý problém).

Samotný Hessian nemá interně vestavěnou gzip kompresi přenosu, ale myslíme si, že by nebyl problém rozšířit Hessiana o tuto vlastnost.

Ještě před implementací jsme zjistili pár problémů:

- Samotný MySQL má problémy s uložením roku u datumu pod 1970. Celé jsme to vyřešili fyzickém uložení datumu přidáním 2000 let.
- I samotná implementace Hessiana pod PHP má problémy s rokama menších než 1970. Řešení je psát veškerý datum textově.

4.2 Java Klient



[1] http://www.caucho.com/hessian/

4.2.1 Popis synchronizace databází

Celou operaci rozdělíme na 5 částí.

4.2.1.1 Synchronizace adresáře

Nejprve vybereme z tabulky adresář v HSQLDB všechny záznamy, které mají nastaven sloupec Lokalni_kopie na true. Postupně tyto záznamy přidáme do MySQLdb do tabulky adresář a zapamatujeme si u každého záznamu jeho nové ID.

Následně smažeme (fyzicky) z HSQLDB z tabulky adresář všechny záznamy, kde sloupec Lokalni_kopie = true.

Nyní si stáhneme všechny záznamy z MySQLdb které mají nastaven datum u položky Aktualizace větší než je datum v HSQLDB u položky Aktualizace. Nové položky přidáme přes INSERT, již existující modifikujeme přes UPDATE (případně smažeme a znovu přidáme) (případně to celé zastřešíme použitím INSERT ... ON DUPLICATE KEY UPDATE ...).

4.2.1.2 Synchronizace místa

Nejprve vybereme z tabulky místa v HSQLDB všechny záznamy, které mají nastaven sloupec Lokalni_kopie na true. Postupně tyto záznamy přidáme do MySQLdb do tabulky místa a zapamatujeme si u každého záznamu jeho nové ID.

Následně smažeme (fyzicky) z HSQLDB z tabulky místa všechny záznamy, kde sloupec Lokalni_kopie = true.

Nyní si stáhneme všechny záznamy z MySQLdb které mají nastaven datum u položky Aktualizace větší než je datum v HSQLDB u položky Aktualizace. Nové položky přidáme přes INSERT, již existující modifikujeme přes UPDATE (případně smažeme a znovu přidáme) (případně to celé zastřešíme použitím INSERT ... ON DUPLICATE KEY UPDATE ...).

4.2.1.3 Synchronizace ufo_duch

Nejprve vybereme z tabulky ufo_duch v HSQLDB všechny záznamy, které mají nastaven sloupec Lokalni_kopie na true. Postupne tyto zaznamy pridame do MySQLdb do tabulky ufo_duch. Pokud je ve sloupci Svedek_id použit starý ID již vloženého záznamu v MySQLdb (co jsem provedli v 1. části), použijeme jeho nové ID, které jsme si v 1. části zapamatovali. To samé provedeme jako u Svedek_id u sloupce Misto_id.

Následně smažeme (fyzicky) z HSQLDB z tabulky ufo_duch všechny záznamy, kde sloupec Lokalni_kopie = true.

Nyní si stáhneme všechny záznamy z MySQLdb které mají nastaven datum u položky Aktualizace větší než je datum v HSQLDB u položky Aktualizace. Nové položky přidáme přes INSERT, již existující modifikujeme přes UPDATE (případně smažeme a znovu přidáme) (případně to celé zastřešíme použitím INSERT ... ON DUPLICATE KEY UPDATE ..., pokud to HSQLDB podporuje).

4.2.1.4 Synchronizace kruhy

Nejprve vybereme z tabulky kruhy v HSQLDB všechny záznamy, které mají nastaven sloupec Lokalni_kopie na true. Postupne tyto zaznamy pridame do MySQLdb do tabulky ufo_duch. Pokud je ve sloupci Misto_id použit starý ID již vloženého záznamu v MySQLdb (co jsem provedli v 2. části), použijeme jeho nové ID, které jsme si v 2. části zapamatovali.

Následně smažeme (fyzicky) z HSQLDB z tabulky kruhy všechny záznamy, kde sloupec Lokalni_kopie = true.

Nyní si stáhneme všechny záznamy z MySQLdb které mají nastaven datum u položky Aktualizace větší než je datum v HSQLDB u položky Aktualizace. Nové položky přidáme přes INSERT, již existující modifikujeme přes UPDATE (případně smažeme a znovu přidáme) (případně to celé zastřešíme použitím INSERT ... ON DUPLICATE KEY UPDATE ..., pokud to HSQLDB podporuje).

4.2.1.5 Synchronizace uzivatele

Tabulka uzivatele se vůbec nesynchonizuje, pouze se stáhne její nová podoba a uloží se jen určité sloupce, tak aby se na klientské databazi (HSQLDB) neuchovávali hesla, jen potřebné informace, které jsou nutné pro zobrazení informace o člověku, který poslední upravoval záznam.

4.2.2 Třída SpravaDatabaze

Třída obstarává celou interakci uživatele s databází. Obsahuje metody pro přidávání, editování a mazání záznamu. Umožňuje přepínání zobrazení na databázi, kterou chce uživatel prohlížet nebo prohledávat.

4.2.1.2 Konstruktory

Defaultní konstruktor zavolá metodu `zkopirovatDatabazi()`

4.2.1.3 Metody

`public void zkopirovatDatabazi()`
Zkopíruje celou databázi z pevného disku do paměti

`public void pridatZaznam(int databaze)`
Metoda při zavolání zobrazí dialog pro přidávání nového záznamu do databáze. Parametr `databaze` určuje, do které tabulky má být nový záznam uložen a zároveň také určuje, který dialog se má zobrazit.

`public void smazatZaznam(int databaze, int index)`
Nastaví příslušnému indexu v databázi příznak na smazáno. Nedochází k uplnému odstranění záznamu z databáze.

`public void editovatZaznam(int databaze, int index)`
Podobně jako `pridatZaznam` zobrazí stejný dialog, který má vyplněná všechna pole podle záznamu v databázi. Uživatel má možnost údaje změnit

`public DocasnaDatabaze getDatabaze()`
Vrátí celou databázi, která je v paměti. Metoda je určena hlavně pro následné vykreslování grafů a map.

`public DocasnaDatabaze zobrazDatabázi(int databazaze)`
Poskytuje data pro `JTable`, která jsou zobrazena pomocí `prezentacniVrstva` uživateli. Jako parametr se předává databáze, kterou chce uživatel zobrazit

`public void zmenDatabazi()`
Přepínání mezi databázemi pro zobrazení.

`public void dalsiFiltr()`
Zobrazuje dialog pro přidání dalšího filtru. Na základě uživatelovo vstupu dojde k vyhledaní v databázi v paměti.

4.2.3 Třída Graf

Třída obstarává celou interakci uživatele s databází. Obsahuje metody pro přidávání, editování a mazání záznamu. Umožňuje přepínání zobrazení na databázi, kterou chce uživatel prohlížet nebo prohledávat.

4.2.1.2 Konstruktory

Konstruktor pouze vynuluje atributy třídy

4.2.1.3 Metody

`public void vygenerujGraf()`
Vygeneruje a zobrazí graf s využitím třídy `DataProGrafKlient`

`public void zobrazMapu()`
Zobrazí vygenerovanou mapu. Nejprve si vygeneruje hrubý obrys České republiky a následně vygeneruje body výskytu příslušného úkazu.

4.2.4 Třída DataProGrafKlient

Třída připravuje data, která jsou potřebná pro vykreslení grafu

4.2.1.3 Metody

`public void pripravDataDleOkresu()`
Z databáze, která je v paměti vezme příslušná data a předá je novému objektu `DataProGraf`

`public void pripravDataDleDoby()`
Z databáze, která je v paměti vezme příslušná data a předá je novému objektu `DataProGraf`

`public void pripravDataDleDobyPozorovani()`
Z databáze, která je v paměti vezme příslušná data a předá je novému objektu `DataProGraf`

`public void pripravDataDlePoctuAJ()`
Z databáze, která je v paměti vezme příslušná data a předá je novému objektu `DataProGraf`

4.2.5 Třída DataProGraf

Ze zpracovaných dat vytváří požadovaný graf

4.2.1.3 Metody

`public void nactiData()`
Načte data, která předpřipraví `DataProGrafKlient a provede konverzi na příslušný typ grafu`

`public JPane vratData()`
Vrací data v podobě grafu, který se vykreslí

4.2.6 Třída PrezentacniVrstva

Obstarává celé GUI. Generuje jej z grafické stránky a zároveň obsluhuje události, které uživatel vyvolá.

4.2.1.3 Metody

`public void vytvorGUI()`
Vytvoří celé grafické prostředí

`public void spustGUI()`
Zobrazí vytvořené prostředí

4.2.7 Třída DocasnaDatabaze

Třída slouží k uchování databáze v paměti, její čtení a zápis do ní, vyhledávání a všechny ostatní funkce, které poskytuje databáze, která je uložená na disku.

4.2.8 Třída Main

Hlavní třída celé aplikace, která spouští `PrezentacniVrstva`

4.2.1.3 Metody

`public void main()`
Zajistí zavolání metod na vytvoření a spuštění grafického prostředí

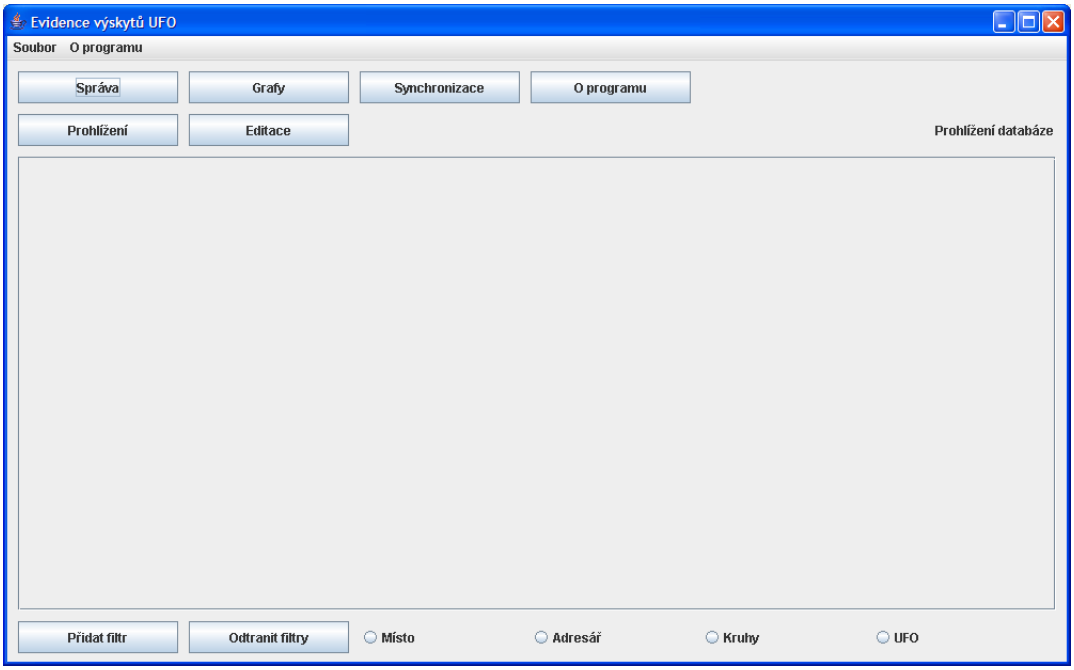
4.3 WEB Klient

Webový klient bude proveden přes rozhraní běžného internetového prohlížeče s využitém PHP skriptu. Uživatel vyplní tento dotazník http://www.kpufo.cz/codelat/ufo.htm. Data budou odeslána na server a uložena do speciální tabulky v MySQL databázi. V okamžiku, kdy uživatel klienta v Javě provede synchronizaci, dojde ke stažení také této položky a bude mu nabídnuta ke kontrole. Uživatel následně buď uloží nový záznam do databáze nebo ho zamítne.

5. Grafické uživatelské rozhraní a popis funkce

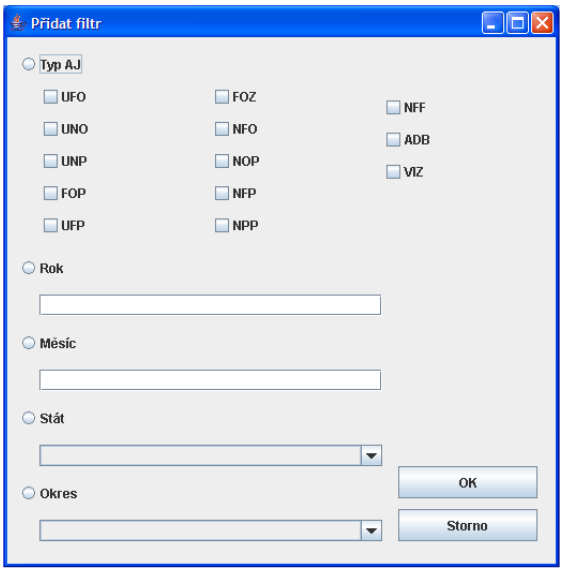
5.1 Správa

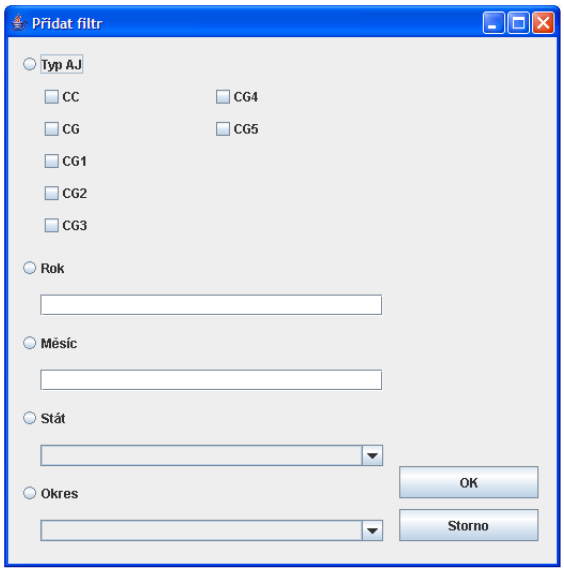
5.1.1 Prohlížení



Při spuštění programu dojde k zobrazení `Správy databáze` v režimu `Prohlížení`, nebude aplikován žádný filtr a jako výchozí bude vybraná databáze UFO. Při spuštění také dojde k překopírování lokální databáze do paměti pro další zpracování.

- tlačítko `Přidat filtr`





V závislosti na vybrané databázi bude zobrazen jeden těchto dialogů. Pokud bude vybraná databáze `Kruhy v obilí`, bude zobrazen dialog napravo. V případě vybrané databáze `UFO` bude zobrazen dialog nalevo.

Vybraný filtr bude předán databázi, která najde relevantní data v databázi v paměti a přepíše jimi stávající databázi v paměti.

- tlačítko `Odstranit filtry`

Budou zrušeny všechny filtr tím, že dojde ke znovunačtení databáze z disku.

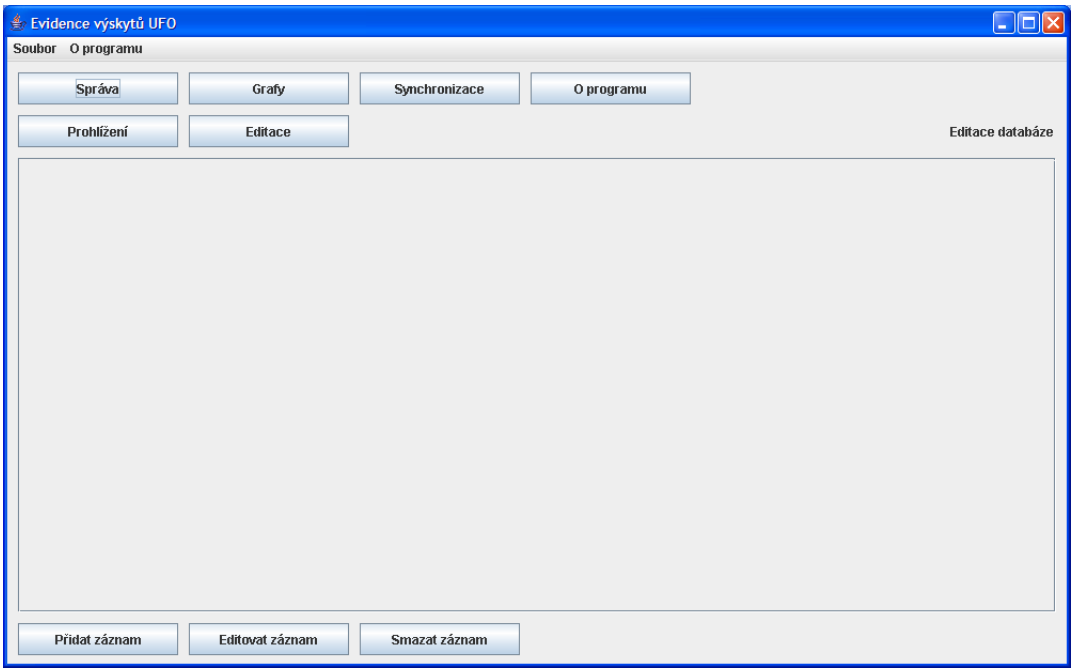
- přepínač `Místo`, `Adresář`

Zobrazí se příslušná databáze a dojde k deaktivaci tlačítek pro přidávání a odebírání filtrů.

- přepínač `Kruhy`, `UFO`

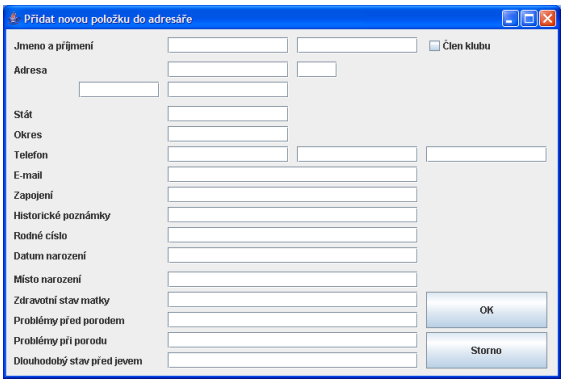
Zobrazí se příslušná databáze a dojde k aktivaci tlačítek pro přidávání a odebírání filtrů. Budou odstraněny všechny filtry, které byly použity

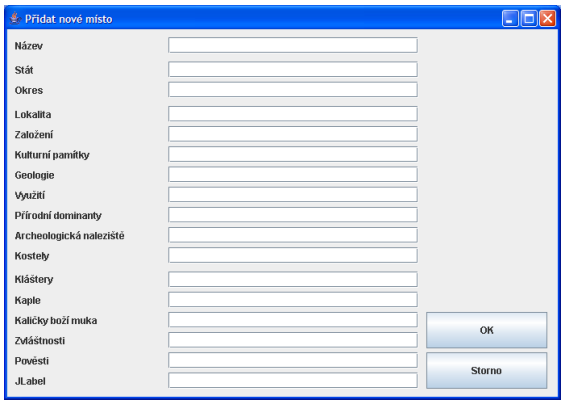
5.1.2 Editování

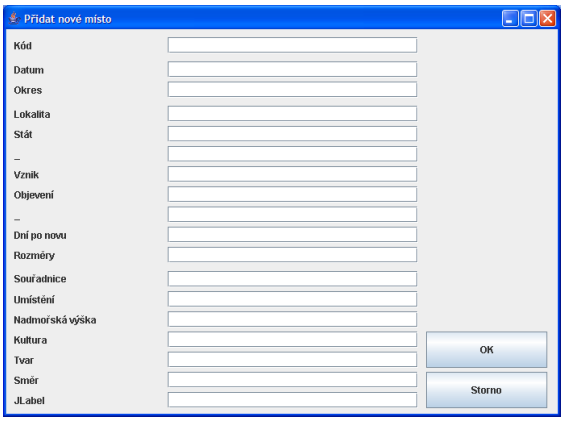


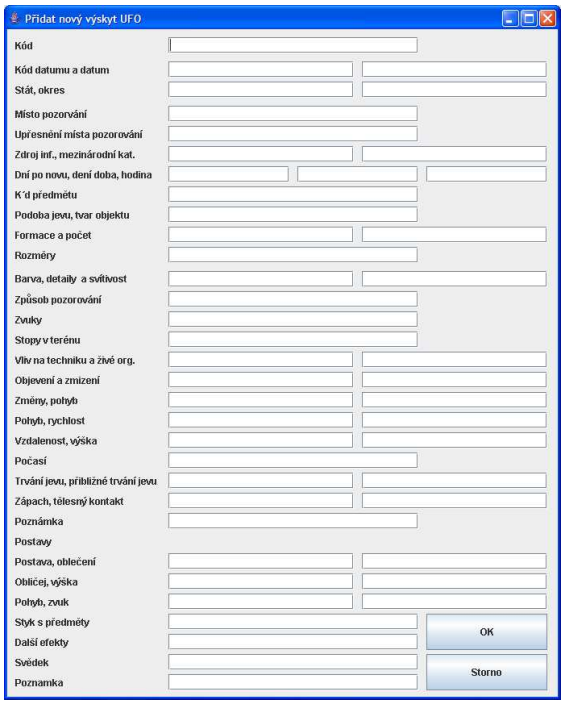
- tlačítko `Přidat záznam`

V závislosti na otevřené databázi bude zobrazen jeden z následujících dialogů. Před jeho zobrazením bude zobrazeno varování o doporučené synchronizaci s databází na serveru (pouze 1x za běhu programu). Po vyplnění formuláře uživatelem dojde k uložení do databáze na disku a do databáze v paměti. Záznam bude také zobrazen mezi vyhledanými záznamy bez ohledu na to, zda odpovídá kritériím pro vyhledávání nebo ne.









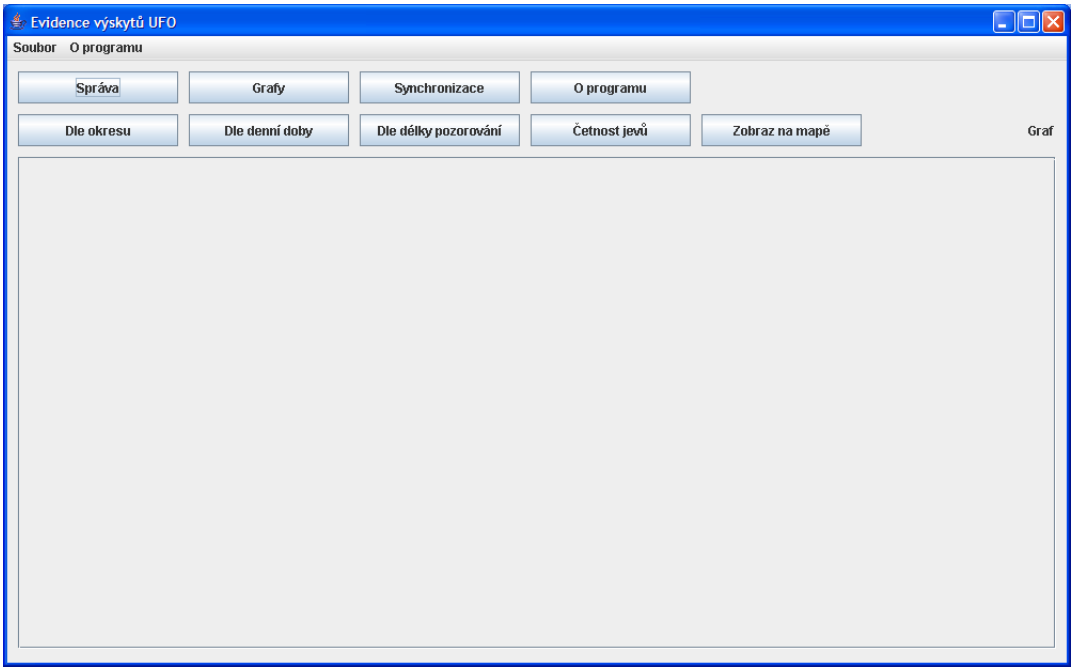
- tlačítko `Editovat záznam`

Obdobně jako u přidávání záznamu bude uživatel varován o doporučené synchronizaci, následně bude zobrazen shodný dialog jako u přidávání záznamu. Dialog bude mít předvyplněné hodnoty, které může uživatel změnit. Změna se zapíše do databáze na disku i v paměti.

- tlačítko `Smazat záznam`

Po varování o synchronizaci a po dotazu, zda opravdu smazat bude u záznamu změněn příznak na smazaný. Změna se opět zapíše do obou databází.

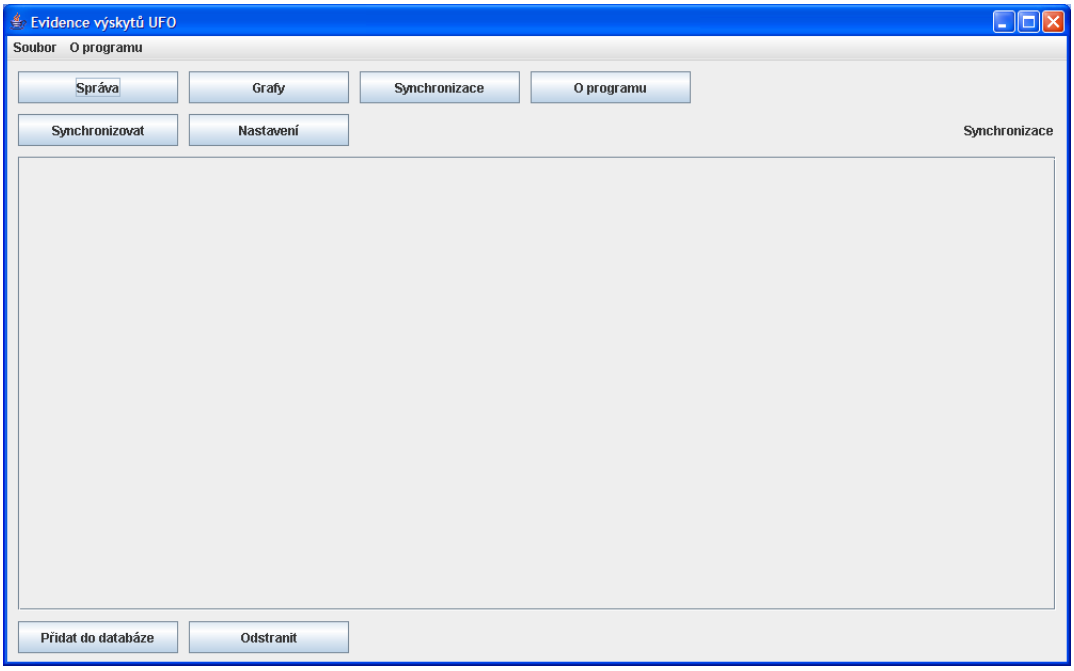
5.2 Grafy



Po stisku příslušného tlačítka bude zobrazen vybraný graf nebo mapa.

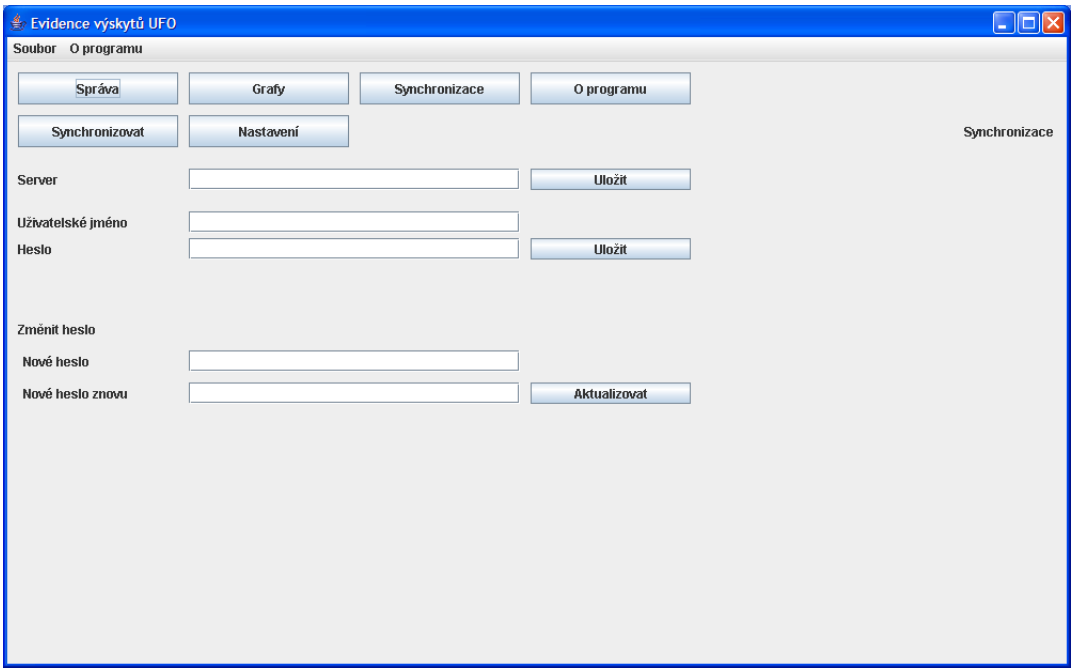
5.3 Synchronizace

5.3.1 Synchronizace databáze



V okamžiku zobrazení dojde ke spojení se serverem a synchronizaci lokální databáze a databáze na serveru. Současně se stáhnou i položky „speciální“ databáze, ve které jsou uloženy záznamy od cizích pozorovatelů, po stažení dojde k odstranění z databáze na serveru. Uživatel je uvidí v tabulce a příslušným tlačítkem záznam uloží nebo zahodí.

5.3.2 Nastavení



`Server`	Nastavení serveru s centrální databází
`Uživatelské jméno`	Jméno uživatele, který používá tuto kopii
`Heslo`	Heslo stejného uživatele
`Změnit heslo`	Změna hesla za nové – pouze v online přímo na serveru

6. Přiřazení tříd/modulů programátorům

Ladislav Ašenbrener	uživatelská dokumentace
Martin Sloup	databáze
Václav Purchart	mapa, grafy
Jan Tichava	GUI
Miroslav Hesoun	PHP formulář, skript na uložení do databáze
Martin Žibrický	správa databáze