Univerzita Pardubice

Fakulta ekonomicko-správní

Informační systém pro Českou abilympijskou asociaci

Martin Ibl

Bakalářská práce

2009

Univerzita Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Martin IBL**

Studijní program: **B6209 Systémové inženýrství a informatika**

Studijní obor: **Informatika ve veřejné správě**

Název tématu: **Informační systém pro Českou abilympijskou asociaci**

Zásady pro vypracování:

1. Popis prostředí České abilympijské asociace
2. Současný stav, jeho možnosti a potřeby
3. Návrh informačního systému
4. Implementace provedeného návrhu

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Česká abilympijská asociace [online]. 2008 [cit. 2008-06-22]. Dostupný z WWW: <http://www.abilympics.cz/>.**

[2] **Interní zdroje České abilympijské asociace**

[3] **ŠPÍRKOVÁ, Petra. Moderní metody řízení projektů. [s.l.], 2008. 76 s. Bakalářská práce.**

[4] **ŠIMONOVÁ, Stanislava, MYŠKOVÁ, Renata, JIRAVA, Pavel. Projektování informačních systémů - UML, procesní řízení. [s.l.] : [s.n.], 2006. 114 s.**

[5] **KOFLER, Michael, ÖGGL, Bernd . PHP 5 a MySQL 5. [s.l.] : [s.n.], 2007. 680 s.**

Vedoucí bakalářské práce: **Ing. Milan Tomeš**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **6. října 2008**

Termín odevzdání bakalářské práce: **1. května 2009**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 6. října 2008

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.
Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 25. 4. 2009

Martin Ibl

PODĚKOVÁNÍ

Rád bych poděkoval panu Ing. Milanu Tomešovi, za poskytnuté cenné rady a připomínky, vstřícný přístup při konzultačních hodinách a odborném vedení při zpracování mé bakalářské práce.

ANOTACE

Práce se zabývá řešením problémů České abilympijské asociace, které souvisí s pořádáním národní Abilympiády. Problém je řešen návrhem a následnou implementací informačního systému, který umožní lépe spravovat všechny problémové činnosti spojené s Abilympiádou. Tato práce postihuje celou šíři životního cyklu vývoje informačního systému a posléze hodnotí jeho úspěšné zavedení do provozu.

KLÍČOVÁ SLOVA

informační systém, implementace, MVC model, back-end, front-end, CMS Joomla!, PHP, programování, JavaScript, MySQL, datový model, datové modelování

TITLE

Information System of the Czech Abilympic Association

ANNOTATION

The work deals with solving problems of Czech abilympic association, which are related with hosting of national Abilympic games. Problem is solved by design and resulting implementation of information system, which will provide better administration of all issue activities joined with Abilympic games. This work focuses total life cycle of developing information system and evaluate its successful implementation.

KEYWORDS

Information System, Implementation, MVC model, Back-end, Front-end, CMS Joomla!, PHP, Programming, JavaScript, MySql, Data model, Data modeling

Obsah

Seznam obrázků

Seznam tabulek

Seznam příloh

Úvod

V současnosti stále ještě existuje spousta firem, které spravují velké množství dat pomocí různých „zastaralých metod“. Jako nejhorší případ těchto „zastaralých metod“, lze uvést papírová správa dat, tj. firma nevyužívá žádné elektronické pomůcky, které by ji ulehčily práci. Další velice rozšířenou metodou je elektronická forma správy dat, kde však chybí vzájemný kontext. Může jít například o správu dat v různých tabulkových editorech, kde vzniká vysoká redundance záznamů a zároveň relativní pracnost s jejich manipulací a případným prováděním různých analýz. Pro firmy, které spravují velké množství dat, jsou tyto formy náročné na správu a dlouhodobě neudržitelné. Proto je vhodné navrhnout vhodný informační systém, který usnadní a částečně zautomatizuje časté procesy ve firmě.

Cílem této práce je navrhnout a implementovat bezpečný a plně funkční informační systém (IS) pro Českou abilympijskou asociaci, který usnadní správu dat spojených s pořádáním národní Abilympiády.

První kapitola se zabývá teoretickými základy tvorby informačního systému, jeho životním cyklem a metodikami. Druhá kapitola je zaměřena na základní představení České abilympijské asociace a její současný stav a potřeby. Dále je zde formulována řada požadavků, které by měl budoucí informační systém splňovat. Poslední součástí této kapitoly je stručný přehled nástrojů použitých pro modelování a implementaci informačního systému. Další dvě kapitoly obsahují analýzu problému (tvorba ERD) a posléze návrh řešení ve formě relačního schématu dat. V páté kapitole je stručně představena architektura a funkčnost CMS Joomla!, na němž je informační systém posléze implementován. Šestá kapitola je zaměřena na samotnou implementaci informačního systému. Sedmá kapitola obsahuje souhrnné představení ošetřených výjimek pomocí JavaScriptu. Předposlední kapitola se zabývá závěrečnými pracemi na systému, tj. jeho zavedení do provozu a následné testování. Poslední kapitolou je závěrečné shrnutí této práce a vyhodnocení definovaných cílů.

1 Základní postup vývoje IS

Tato kapitola se zaměřuje na vysvětlení základních východisek, principů a přístupů k tvorbě IS a modelování vůbec.

1.1 Informační systém

Informační systém lze definovat jako soubor lidí, metod a technických prostředků zajišťujících sběr, přenos, uchování, zpracování a prezentaci dat s cílem tvorby a poskytování informací dle potřeb příjemců informací, činných v systémech řízení. [1]

1.2 Základní východiska a principy

Celkový pohled na proces vzniku a existence IS, který pokrývá celý životní cyklus systému, se běžně nazývá **metodika tvorby informačních systémů**. Na druhou stranu metodika rozpracovává jednotlivé dimenze všech fází životního cyklu, příslušné metody, techniky a nástroje. S touto problematikou úzce souvisí vývoj tzv. CASE nástrojů. [2]

Metodika tvorby IS je souhrn [2]:

- etap,
- přístupů,
- zásad,
- postupů,
- pravidel,
- dokumentů,
- řízení,
- metod,
- technik
- a nástrojů,

který pokrývá celý životní cyklus informačních systémů. Metodiky určují **kdo**, **kdy**, **co** a **proč** má dělat v průběhu vývoje a provozu IS. Metodika by se měla vztahovat na všechny prvky IS (viz kapitola 1.1). [1]

Jak již bylo zmíněno, metodiky jsou obsahově naplňovány jednotlivými **metodami**, s nimi souvisejícími **technikami** a k tomu potřebnými **nástroji**. [1]

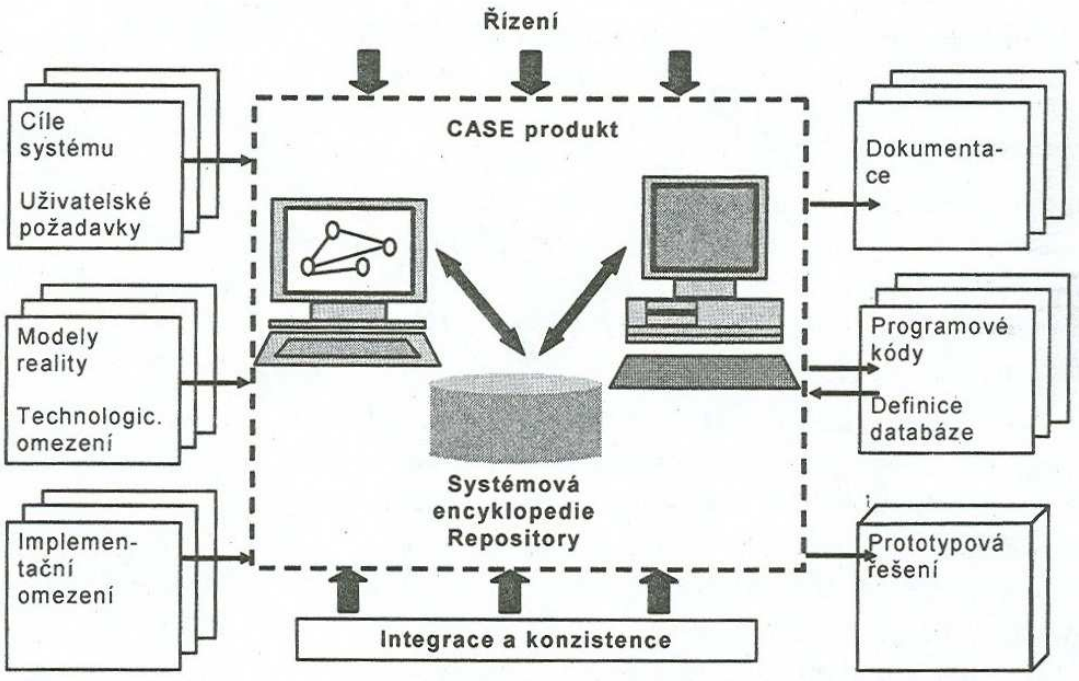
- **Metoda** určuje, **co** je třeba udělat v určité fázi nebo činnosti vývoje či provozu IS. Metoda je vždy spojena s určitým přístupem (objektový nebo strukturovaný).
- **Technika** určuje, **jak** se dobrat požadovaného výsledku. Zpravidla určuje přesný postup jednotlivých činností, způsob použití nástrojů, varianty rozhodnutí v určitých situacích a co z nich vyplývá, vymezuje obor působnosti apod. Na rozdíl od metody je přesnější v závěrech.
- **Nástroj** je prostředek k uskutečnění určité činnosti v procesu vývoje a provozu IS a prostředek k vyjádření výsledku této činnosti.

Není možné jednoduše prohlásit, že jednotlivé metody jednoznačně patří určitým metodikám, nebo že každá technika je tu výhradně k podpoře nějaké své metody apod.

Metodiky jsou různou měrou uplatňovány v rámci prostředků automatizované podpory projektování označené zkratkou CASE (Computer Aided Software/System Engeneering). [1]

Obecné vlastnosti nástrojů CASE (viz Obrázek 1) [1]:

- Jádrem prostředku case je centrální depozitář.
- Grafická podpora nástrojů popisu IS.
- K výstupům patří standardní dokumentace, často i generované programové kódy a popisy databáze, případně také prototypová řešení systému.
- Pro generování programových kódů a popisů databáze je důležitá možnost zpětného generování abstraktních popisů z konkrétního kódu (reverse engineering).



Obrázek 1 - Obecné schéma CASE. Zdroj [1]

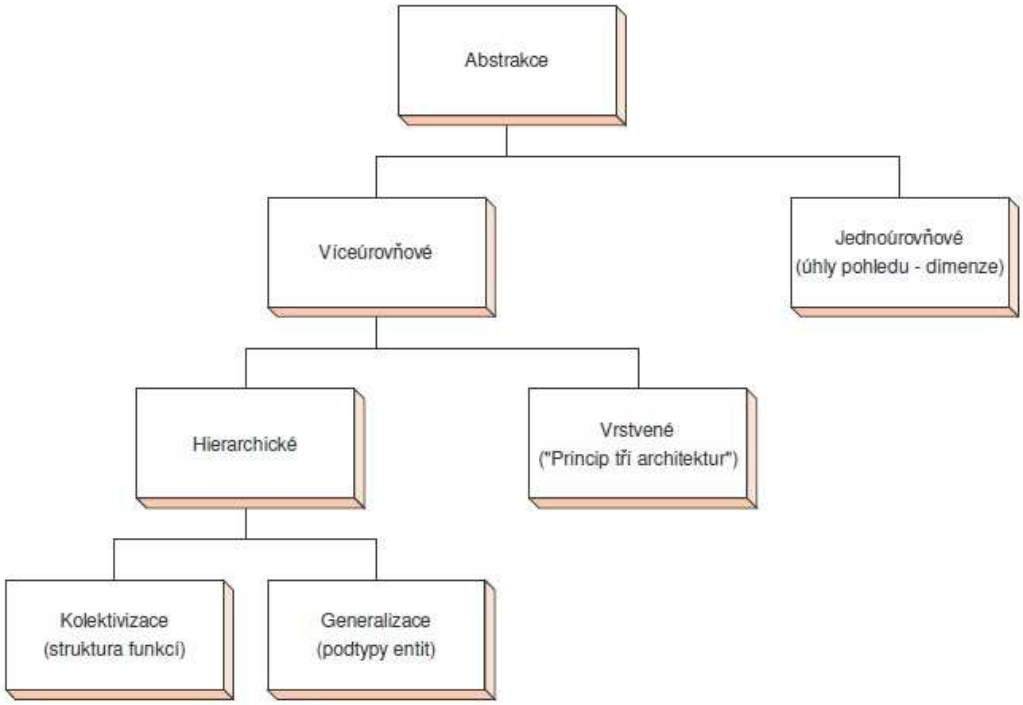
1.2.1 Princip abstrakce

Abstrakce je definována jako myšlenkový proces, který záměrně odlučuje odlišnosti a zvláštnosti zkoumané reality. Zároveň zjišťuje obecné a podstatné vlastnosti předmětů a jevů okolní skutečnosti a vztahy mezi nimi. Abstrakce tedy znamená, že se k některým nepodstatným skutečnostem záměrně nepřihlíží. Hlavním důvodem existence principu abstrakce v metodách analýzy a návrhu IS je **snaha o rozdělení zkoumané problematiky na mentálně zvládnutelné části**. Typickým rysem problematiky, zkoumané při návrhu IS je totiž vždy její značná **rozsáhlost** a **složitost**. [2]

Abstrakce lze podle jejich způsobu rozdělit do základních typů [2]:

- **Víceúrovňové**, kde každý abstraktní prvek může být tvořen buď konkrétními (dále nedělitelnými), nebo také abstraktními prvky.
    - **Hierarchické** abstrakce jsou prostředek rozkladu prvků vyvíjeného systému do detailní úrovně pohledu. Obecně existují dva základní typy hierarchické abstrakce:
        - **Kolektivizace** (část – celek) – se typicky používá ve funkčním modelu, kde se dělí systém na podsystémy, části podsystémů atd. Vyšší celek je zcela definován jako souhrn svých částí (nemá jiný význam). [1]
        - **Generalizace** (podtyp – obecný typ) – se typicky používá v datovém modelu, kde je možné uvažovat o jednotlivých specifických variantách nadřízeného pojmu. Na rozdíl od kolektivizace není nadřazený celek definován jakou souhrn podřízených částí, ale jako nositel jejich společných vlastností (atributů). [1]
    - **Vrstvené** abstrakce definují takovou hierarchickou strukturu prvků, kde každý prvek je detailním rozpracováním svého nadřízeného prvku z určitého hlediska. Nejznámějším příkladem vrstvené abstrakce je **princip tří architektur**.
- **Jednoúrovňové** abstrakce, kde celá struktura konkrétních prvků je jednorázově plošně rozdělena na jednotlivé abstraktní oblasti substruktury (úhly pohledu). Konkrétně se v metodách analýzy a návrhu IS jedná o **princip různých pohledů (modelů)** na vytvářený systém datový versus funkční model atd.

Výše uvedenou klasifikaci abstrakcí shrnuje Obrázek 2.



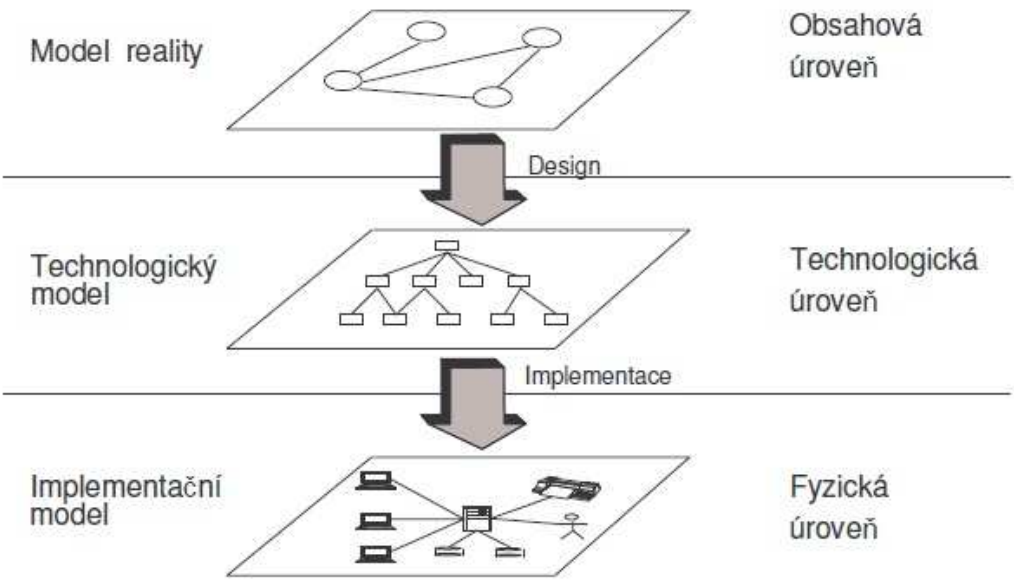
Obrázek 2 - Klasifikace abstrakcí. Zdroj [2]

Princip tří architektur definuje použití abstrakce pro vývoj informačního systému po jednotlivých vrstvách (vrstvená abstrakce). Jednotlivé vrstvy se zaměřují na 3 hlavní aspekty vyvíjeného systému. Tyto hlavní aspekty tvoří přirozenou posloupnost úkolů, a to od specifikace obsahu systému, přes vyplývající možnosti technologického řešení, až ke konkrétní použité technologie určující implementační možnosti.

Návrh IS potom podle principu tří architektur probíhá ve třech po sobě následujících úrovních (viz Obrázek 3) [1]:

- **Konceptuální úroveň** – zde je vytvořen zcela obecný, čistě obsahový model systému, nezatížený ani technologickou koncepcí řešení, ani jeho implementačními specifiky.
- **Technologická úroveň** – zde je vytvořen model systému, zohledňující technologickou koncepci řešení, tj. ve strukturovaném pojetí koncepci organizace dat (technologie souborová, stromová, síťová, i relační databázová atd.) a technologickou koncepci jejich zpracování. Technologický model stále nesmí být zatížen implementačními specifiky řešení. Je zde tedy abstrahováno od implementačních specifik řešení, obsahové náležitosti jsou dány obsahovým modelem a zde se neřeší. Technologický návrh určuje, jak bude obsah systému v dané technologii realizován (zohledňuje „logiku“ použití zvolené technologie).
- **Implementační úroveň** – zde je vytvořen model systému, zohledňující implementační specifika použitého vývojového prostředí (konkrétního databázového systému,

programovacího jazyka a dalších prostředků, jako například vývojové prostředí GUI atd.). Není zde abstrahováno od žádných specifik řešení, obsahové náležitosti jsou dány konceptuálním řešením, technologie je dána technologickým řešením, implementační návrh se tedy týká pouze implementačně specifických rysů systému. Implementační návrh určuje, **čím** je technologické řešení realizováno.



Obrázek 3 - Princip "tří architektur". Zdroj [1]

Každá ze tří úrovní definuje specifickou architekturu. Každá architektura má svou specifickou logiku a specifický předmět zájmu (obsah, technologii a implementační specifika). Pro metody to znamená:

- pro každou úroveň návrhu mít specifický **jazyk** (viz Tabulka 1) a **techniky návrhu**,
- pro každý přechod z jedné úrovně do následující mít specifické **techniky přechodu** z jedné úrovně do druhé.

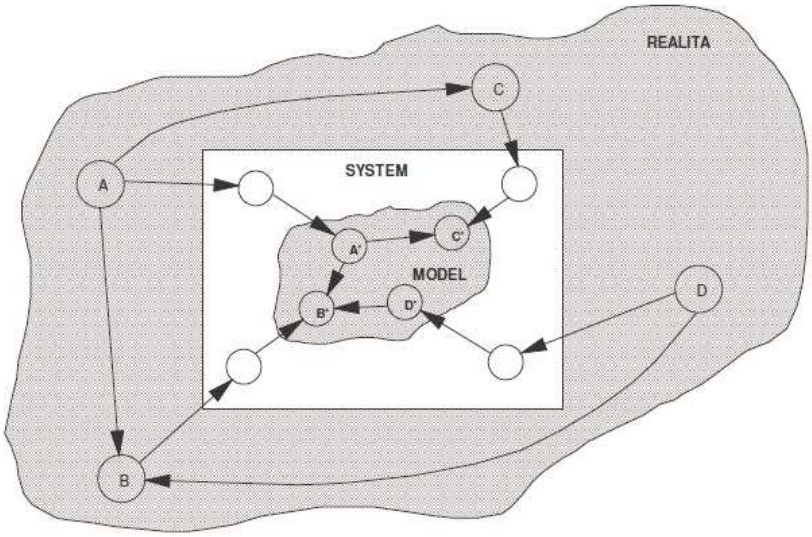
Tabulka 1 - Specifické jazyky. Zdroj vlastní

	Jazyk úrovně
Modelování reality	Diagram entit a jejich vztahů – ERD (Entity Relationship Diagram) UML – Diagram tříd, Stavový diagram, Use Case, aj.
Technologický návrh	Relační model dat - RMD UML – Diagram komponent, aj.
Implementační prostředí	Konkrétní programovací jazyk

1.2.2 Princip modelování

Ústřední myšlenkou modelování je fakt, že všechny modely jsou špatně (neodpovídají realitě), ale některé jsou užitečné. Model je vždy zjednodušením, abstrakcí reality (jinak by to nebyl model). Model tedy nikdy nemůže být úplně správně - vždy se něčím od reality liší. [3]

Princip modelování je základním principem metod konceptuálního návrhu informačního systému. Metodické postupy a vlastnosti jejich nástrojů a technik vycházejí z myšlenky, že informační systém je modelem reálného systému (reálného světa). Obecně princip modelování ilustruje Obrázek 4. [2]



Obrázek 4 – Princip modelování. Zdroj [2]

Funkční přístup k modelování

Funkční přístup chápe model reálného světa jako **souhrn stavů reálného světa a změn těchto stavů**. Smyslem modelování ve funkčním pojetí je [2]:

- Použití abstrakce, která umožňuje ohlížet od nepodstatných rysů reality (implementačních charakteristik, organizačních celků a činností, které se netýkají přímo informačního systému atd.) a tím zjednodušit úlohu analytika systému.
- Formalizací pojmů vytvořit prostředek dorozumění mezi odborníky rozdílných profesí - analytikem systému a odborníkem v dané oblasti reálného světa.
- Možnost provádět relativně lacino a bez následků změny v modelu, které by přímo v reálném světě byly příliš nákladné, nebo neuskutečnitelné. Potřeba neustálých změn modelu vyplývá jednak z neustálých změn reality a jednak ze samotného procesu zkoumání reálného světa, jehož se účastní řada lidí, z nichž každý má jiný úhel pohledu a tím nutně vidí stejné věci různě a vždy mají problémy si vzájemně porozumět.

Datový přístup (datové modelování)

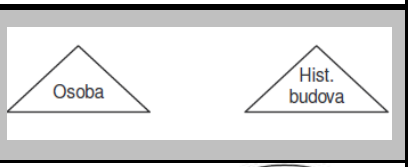
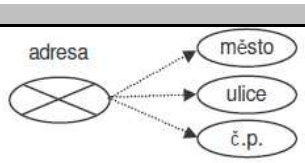
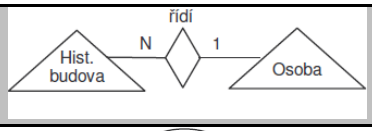
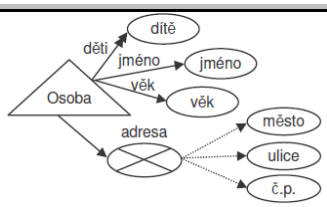
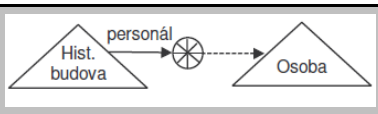
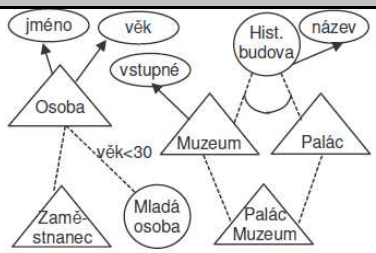
Z hlediska datového přístupu je podstata modelu jiná. Zatímco funkční přístup se zabývá především událostmi, stavy, atd., datová analýza se zaměřuje na **vlastnosti (atributy)** reálného světa. Základní skutečností datového modelování je, že takováto podoba datové základny informačního systému zaručuje [2]:

- jednoznačnost datových položek (každá položka má jasný význam, vyjadřuje buď atribut entity, nebo vztahu),
- konsistenci dat v informačním systému, tím, že omezuje redundance dat na technologické minimum.

Způsob modelování prostřednictvím abstrakce

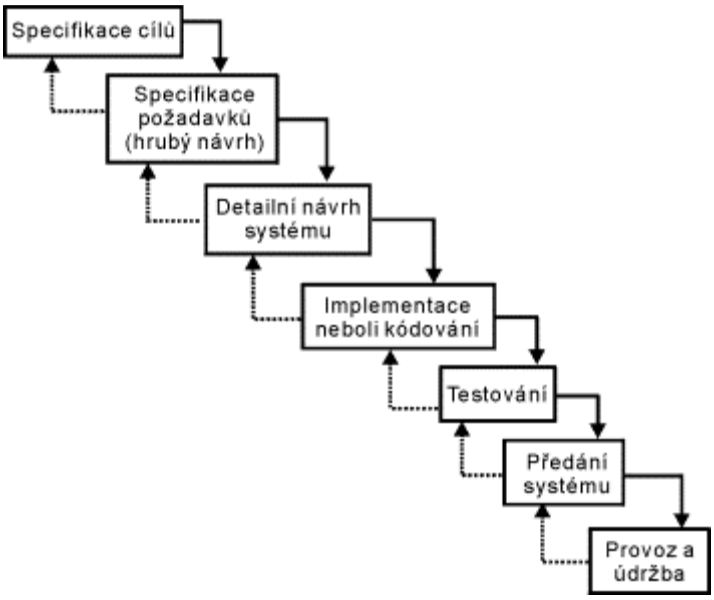
Při modelování reality se mnoho metodik uchyluje k využití různých výrazových prostředků. Tyto modely se nazývají **sémantické modely**. Základním prostředkem modelování sémantiky je již zmiňovaná **abstrakce**. Přehled abstrakcí používaných v těchto modelech shrnuje následující tabulka. [2]

Tabulka 2- Sémantické výrazové prostředky. Zdroj [2]

Prvek	Charakteristika	Příklad
Třídy a entity	Entity sdílející společné charakteristiky jsou sdruženy do tříd. Existence entity je závislá na její příslušnosti k nějaké třídě. Bez této příslušnosti nemůže entita existovat.	
Agregace	Agregace je proces spojování více entit do entity vyšší úrovně.	
Asociace	Asociace nevytváří novou entitu, nýbrž jen spojuje více entit dohromady.	
Atributy	Formálně je atribut dvousměrný vztah mezi dvěma třídami entit. Intuitivně se atributy využívají pro reprezentaci charakteristik tříd entit.	
Seskupení	Využívá se pro vytváření entity z množiny entit s danou strukturou. Na rozdíl od třídy není entitou na metaúrovni.	
Specializace a generalizace	Specializace je zjemnění třídy do podtřídy (Zaměstananec/Mladá osoba). Generalize je sjednocení více tříd do jedné. (Hist. budova)	

1.3 Životní cyklus vývoje IS

Životním cyklem vývoje IS se rozumí souhrn dílčích procesů, které pokrývají celý časový úsek od zahájení až po dokončení informačního systému. Životní cyklus IS je relativní a je každou konkrétní metodikou chápan trochu jinak. Příkladem životního cyklu může být vodopádový model, který ilustruje Obrázek 5.



Obrázek 5 - Vodopádový model životního cyklu. Zdroj [4]

1.3.1 Předběžná analýza (specifikace cílů)

Základem jakékoli práce na vývoji či jakékoli úpravě stávajícího systému jsou požadavky uživatelů a cíle organizace. Cílem předběžné analýzy je tedy dané požadavky shromáždit, v hrubých rysech rozebrat a odhadnout dobu realizace a celkové náklady. Hlavní je tedy pouze sestavit základní rámec požadavků, cílů a funkcí, ne je podrobně rozebírat.

Celkový rámcový projekt by měl obsahovat zhruba následující věci:

- časový plán projektu,
- zdroje nutné k řešení (finance, personál, SW, HW a podobně),
- odhad funkčnosti, rozsahu systému, ekonomické efektivnosti a návratnosti investice.

Nástroje pro vytvoření specifikačního projektu:

- analýza současného stavu - cílem je zjistit současný stav, nedostatky a navrhnout změny,
- získání požadavků uživatelů a zjištění požadovaných vstupních a výstupních informací,
- seznam problémů, které jsou známy, popis jejich důsledků a nástin řešení.

Konečným výstupem této části je dokument, který specifikuje účel systému, identifikuje jeho uživatele a jejich zásadní požadavky, definuje části systému a navrhuje jejich řešení, obsahuje seznamy událostí a odhady datové základny, technického a softwarového zajištění. [4]

1.3.2 Analýza systému (specifikace požadavků)

Tato část cyklu je detailnější verze předchozí části. Důležitost této fáze je klíčová, neboť veškeré chyby ve struktuře dat i systému, které se zde neodhalí, jsou později velice obtížně odstranitelné. [4]

V této fázi se specifikují všechny prvky systému (entity) a jejich vazby (asociace).

1.3.3 Návrh

Tato část je výsledkem analýzy systému. Výsledkem je dokument, který je podkladem pro obsah smlouvy s externí firmou o návrhu a realizaci IS, časový harmonogram, cena vyvíjeného projektu, konkrétní implementace systému (patří sem logický datový model a fyzický datový model), podmínky zavádění v organizaci, záruční servis a podmínky celkového předání IS.

Prvky studie, které musí obsahovat:

- Základní informace o tvůrcích systému, v případě externí firmy její specifikace, dále pokud jde o systém složený z několika podsystémů také informace o jejich dodavatelích.
- Základní informace o organizaci, pro kterou je systém vyvíjen, včetně uvedení týmu zaměstnanců, kteří popřípadě budou spolupracovat s externí firmou.
- Popis současného stavu organizace.
- Globální návrh IS, neboli logický datový model, který je návrhem funkcí a dat systému bez ohledu na technologické prostředí.
- Detailní návrh IS, neboli fyzický datový model, který obsahuje funkční analýzu systému, datovou analýzu, popis veškerých datových toků v organizaci a popis funkcí řízených událostmi. Celkovým výstupem je návrh funkcí a dat budoucího systému, které jsou definovány na základě prostředí, ve kterém bude systém implementován.
- Detailní popis nasazení IS v praxi, SW a HW studie související s nasazením nového IS.
- Detailní popis testovacího provozu systému, včetně poskytování záručního servisu.
- Celkový harmonogram spolupráce, do něhož patří časový harmonogram dodávky, platby, celková cena, podmínky dodání, ceny pozáručního servisu a podobně.

Při tvorbě studie se nesmí zapomínat na to, že je nutné veškerá fakta uvést v dostatečně detailním provedení a v podobě, která bude pochopitelná všem členům vedení, kteří provádí závěrečná rozhodnutí (tím je míněn zejména logický datový model). Celá studie by měla být vytvářena s vědomím, že je to poslední dokument, se kterým se management setká před konečným rozhodnutím o realizaci systému. V případě dohody mezi firmou a tvůrci systému tato studie slouží jako podklad realizace systému a podklad pro podmínky předání a testování. [4]

1.3.4 Implementace

Tato část životního cyklu IS je vlastním programováním, kterého se účastní vybraní experti v programování a analytik nesoucí zodpovědnost za správnost řešení. Jako podklady pro jejich práci slouží veškeré informace shromážděné předchozími etapami a fyzický návrh systému.

Postup práce je následující. Na základě získaných faktů z fyzického návrhu se definují vstupy a výstupy jednotlivých operací a určí způsob jejich modifikace. Naprogramují se veškeré funkce a doladí se jejich vzájemné propojení. Dále se jednotlivé realizované funkce ověří a připraví se testovací data, která musí obsahovat maximální procento konečných reálných dat. [4]

1.3.5 Testování

V této etapě se provádí připravené testy na hotovém IS. Je nutné vyzkoušet veškeré možné reakce systému na zadávaná data a zjištěné nedostatky opravit. Testování se často provádí na systému, který ještě není v reálném prostředí, neboť případné selhání by mohlo mít rozsáhlé následky. Příkladem jsou systémy ve zdravotnictví, letectví, jaderném průmyslu a podobně. [4]

1.3.6 Zavádění systému

Zaváděním systému je míněna především jeho instalace, zavedení do provozu organizace, transformace původní datové základny tak, aby byla přístupná novému systému, poskytnutí manuálů a školení uživatelům. Při školení je nejlepším postupem nejprve školit vedoucí pracovníky a pokračovat zaměstnanci v provozu.

Tato etapa se nesmí v žádném případě podcenit, neboť jejím zanedbáním by mohla u budoucích uživatelů vzniknout averze vůči novému systému a tím neúspěch celého projektu.

Zavedení systému může být provedeno jedním z následujících způsobů [4]:

- **Souběžná strategie** – je založena na pokračujícím provozu původního systému + současný provoz nového systému. Provoz obou systémů trvá, dokud nový systém nepracuje spolehlivě a uživatelé s ním nejsou dostatečně seznámeni. Tato metoda je bezpečná, ale velice náročná pro zaměstnance, neboť musí provádět dvakrát totéž, což by mohlo vést k averzi vůči novému IS. Proto se na toto období najímají externí pracovníci.
- **Pilotní strategie** – je založena na zavedení nového systému jen ve vybrané části podniku a po jeho ověření se systém zavede do celé organizace. Jako pilotní část se vybere taková, která je poměrně náročná a je možné na ní ověřit co nejvíce problémových oblastí.

- **Postupná strategie** – využívá se zejména u velice složitých systémů, kde jsou složité vnitřní vazby. Nejprve se zavádějí primární části IS, na kterých ostatní části závisí, po jejich ověření se podobným postupem zavádí ostatní části až po zavedení celého systému.
- **Nárazová strategie** – spočívá v odstranění původního systému a zavedení kompletního nového systému. Tato strategie je velice riskantní, ale ušetří se při ní čas i pracovní síly.

1.3.7 Zkušební provoz

Zkušební provoz je celková realizace projektu, ve které je poskytovatel povinen zajistit okamžitý servis, odstranit chyby zjištěné během provozu, nebo dořešit dodatečné požadavky uživatelů v rámci původního návrhu. [4]

1.3.8 Rutinní provoz a údržba

Tato etapa je závěrečnou fází projektu, ve které je systém provozován a používán. Do této etapy také spadá údržba systému, tedy zajištění správného provozu, úprava parametrů aplikací nebo změny některých programů tak, aby splňovaly nové požadavky uživatelů. Mezi základní povinnosti zajištění provozu IS patří organizace prací na počítačích a v síti tak, aby byl zajištěn soulad s původním projektem a dokumentací, zajištění přístupových práv k jednotlivým aplikacím, sledování činnosti počítačů a síťových prostředků z hlediska výkonu a poruchovosti, zajištění optimálního provozu systému, zabezpečení systému a ochrana dat před neoprávněným přístupem, nebo minimalizace škod vzniklých výpadkem systému např. záložními systémy nebo archivací dat. V neposlední řadě do této etapy také patří i opětovné školení uživatelů. [4]

1.4 Požadavky na IS a jejich role v procesu vývoje IS

Požadavkem se obecně rozumí jednotka potřebnosti funkcionality nebo jiné vlastnosti systému.

Pojem požadavek může mít širší, nebo užší význam [2]:

- uživatelský (užší) – znamená subjektivní požadavek, kladený uživatelem na systém,
- obsahový (širší) – znamená požadavek na obsah systému, obecně jakýkoliv důvod k potřebě jisté funkcionality či vlastnosti systému. Kromě fyzického požadavku uživatele může požadavek vzniknout na základě změny strategie společnosti nebo jako výsledek analýzy podle pravidel a technik metodiky, který odhalil potřebu modelovat systémem jisté reálné zákonitosti, nebo formální důvod kladený normou, či standardem apod.

Obsahové požadavky, na rozdíl od uživatelských, vznikají už před zavedením systému do provozu. Tyto požadavky mohou být hierarchicky členěny.

Pro potřebu analýzy je třeba požadavky dělit na:

- **Funkční** požadavky reprezentují požadavky na funkční obsah informačního systému. Vznikají už při tvorbě strategie společnosti a postupně jsou detailizovány až na úroveň požadavky na elementární funkce systému.
- **Nefunkční** požadavky reprezentují všechny ostatní požadavky na systém. Dělí se na:
    - **Technologické** požadavky vznikají po určení technologií, které budou na realizaci informačního systému použity, např. určení operačního systému, typu databázového systému, apod.
    - **Designové** požadavky se řeší na úrovni realizace systému. Sem patří například požadavky na vzhled uživatelského rozhraní, rychlost odezvy, apod.

2 Základní informace, požadavky a nástroje

Tato kapitola stručně představuje Českou abilympijskou asociaci, její současný stav, potřeby a návrh řešení problémů spojených s tímto současným stavem. Dále jsou zde formulovány základní požadavky na informační systém, který nahradí současnou administrativu této asociace.

2.1 Česká abilympijská asociace

Sdružení Česká abilympijská asociace (CAA) vzniklo v roce 1997 a sídlí v Pardubicích. V počátcích svého působení se zaměřila na prosazování abilympijského hnutí v České republice a na přípravu páté světové Abilympiády, která se konala v roce 2000 v Praze. Jedná se zpravidla o dvoudenní soutěžní klání, kterého se mohou zúčastnit pouze zdravotně hendikepovaní soutěžící, a to v celé řadě disciplín. Nyní asociace působí také v oblasti zaměstnávání zdravotně postižených lidí a v oblasti prevence a odstraňování stavebních bariér. [5]

Mezi významné aktivity tohoto občanského sdružení patří [5]:

- sociální rehabilitace (metodou podporovaného zaměstnávání),
- sociálně aktivizační služby (volnočasové aktivity),
- osobní asistence,
- konzultační a poradenská střediska bezbariérovosti,
- Abilympijský zpravodaj,
- národní Abilympiáda,
- zážitkový seminář „Společnou cestou“,
- příprava k práci v rámci pracovní rehabilitace.

2.1.1 Současný stav a potřeby v CAA

Hlavní činností CAA je pořádání národní Abilympiády, na kterou se v průběhu roku mohou hlásit soutěžící na celou řadu disciplín. Mimo soutěžící se na Abilympiádu hlásí i rozhodčí, pořadatelé, dobrovolníci a doprovody soutěžících. V současné době se všichni tito „klienti“ mohou přihlásit pouze pomocí klasických papírových přihlášek, které musí poslat poštou. Průměrně CAA příjme ročně kolem 300 přihlášek od všech klientů. Každá přihláška obsahuje řadu osobních údajů, informace o způsobu dopravy na Abilympiádu a hlavně informace o stravování a ubytování v průběhu soutěží. S touto skutečností vyvstává řada problémů [6]:

- termín ukončení přijímaní přihlášek a kontrola zaplacení poplatků,
- neudržitelná správa velkého množství záznamů v MS Excel a vysoká redundance,
- velice náročné přidělování ubytování klientům,

- možnost odebírat Abilympijský zpravodaj.

Podávání přihlášek poštou sebou nese velké problémy spojené s dodržením termínu ukončení přijmutí přihlášek, jelikož CAA brala v potaz i přihlášky přijaté po tomto termínu. Tímto způsobem se stalo přijímání přihlášek chaotické a kontrola zaplacení poplatků se stala nepřehledná. Dalším velkým problémem je správa velkého množství špatně strukturalizovaných dat, jelikož zaměstnanci CAA přepisují všechny údaje z přihlášek do aplikace MS Excel. Toto vedlo k nutnosti zařadit na tuto práci více zaměstnanců, kteří by všechny záznamy přepsali a zkontrolovali. Nejedná se zde pouze o podané přihlášky, které znamenali pouze několik set záznamů v „excelovském“ sešitě, ale i seznam všech lidí, kteří chtějí odebírat Abilympijský zpravodaj. Databáze Abilympijského zpravodaje čítala tisíce záznamů, které se museli aktualizovat podle aktuálního čísla Abilympijského zpravodaje (měsíčník). Posledním nejzávažnějším problémem je správa ubytování. Jelikož klient může v přihlášce vyplnit jména upřednostňovaných spolubydlících, je následné přidělování pokojů na základě „excelovských“ tabulek velice náročné. [7]

2.1.2 Návrh řešení problémů

S těchto důvodů je pro potřeby CAA navrhnut informační systém, který by dokázal zaregistrovat „klienty“ jako uživatelé systému, a poté jim dovolil podat libovolnou přihlášku či potvrdit možnost odebírání Abilympijského zpravodaje. Informační systém by dále obsahoval správcovské rozhraní, kam budou mít přístup pouze uživatelé s potřebnými právy. Bližší možnosti systému jsou specifikovány v kapitole 2.2.

2.2 Požadavky na systém

Česká abilympijská asociace měla jasnou vizi, jaké by měl mít informační systém funkcionality. Na základě této vize a další vnějších vlivů byl sestaven souhrn požadavků na systém.

2.2.1 Funkční požadavky

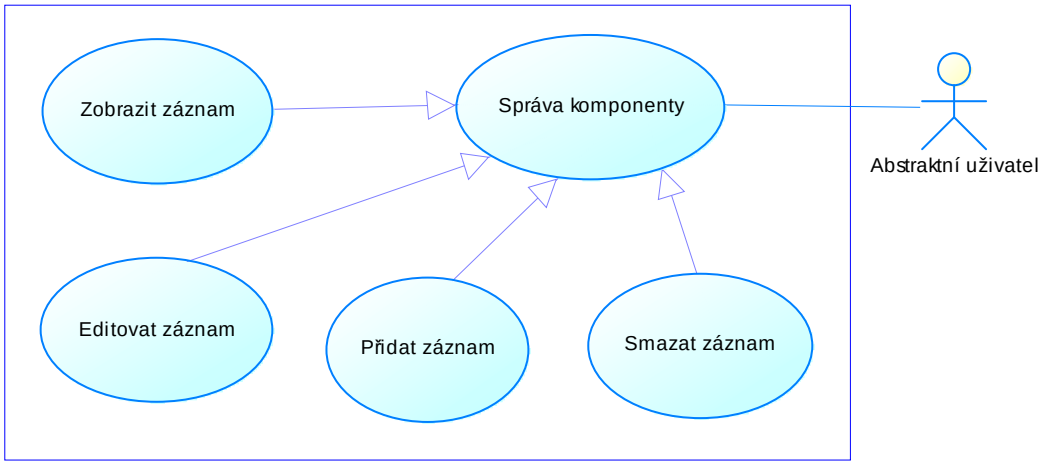
Funkční požadavky na systém jsou následující:

- **FRONT-END (veřejné uživatelské rozhraní)**
    - možnost **zaregistrovat se** s pevně stanovenou strukturu požadovaných atributů,
    - možnost **přihlásit se** do systému po registraci,
    - možnost **soutěžících** přihlásit se na Abilympiádu,
    - možnost **doprovodů** soutěžících přihlásit se na Abilympiádu,
    - možnost **rozhodčích** přihlásit se na Abilympiádu,
    - možnost **dobrovolníků** přihlásit se na Abilympiádu,

- možnost **pořadatelů** přihlásit se na Abilympiádu,
- možnost odebírat **Abilympijský zpravodaj** jakýmkoli registrovaným uživatelem.

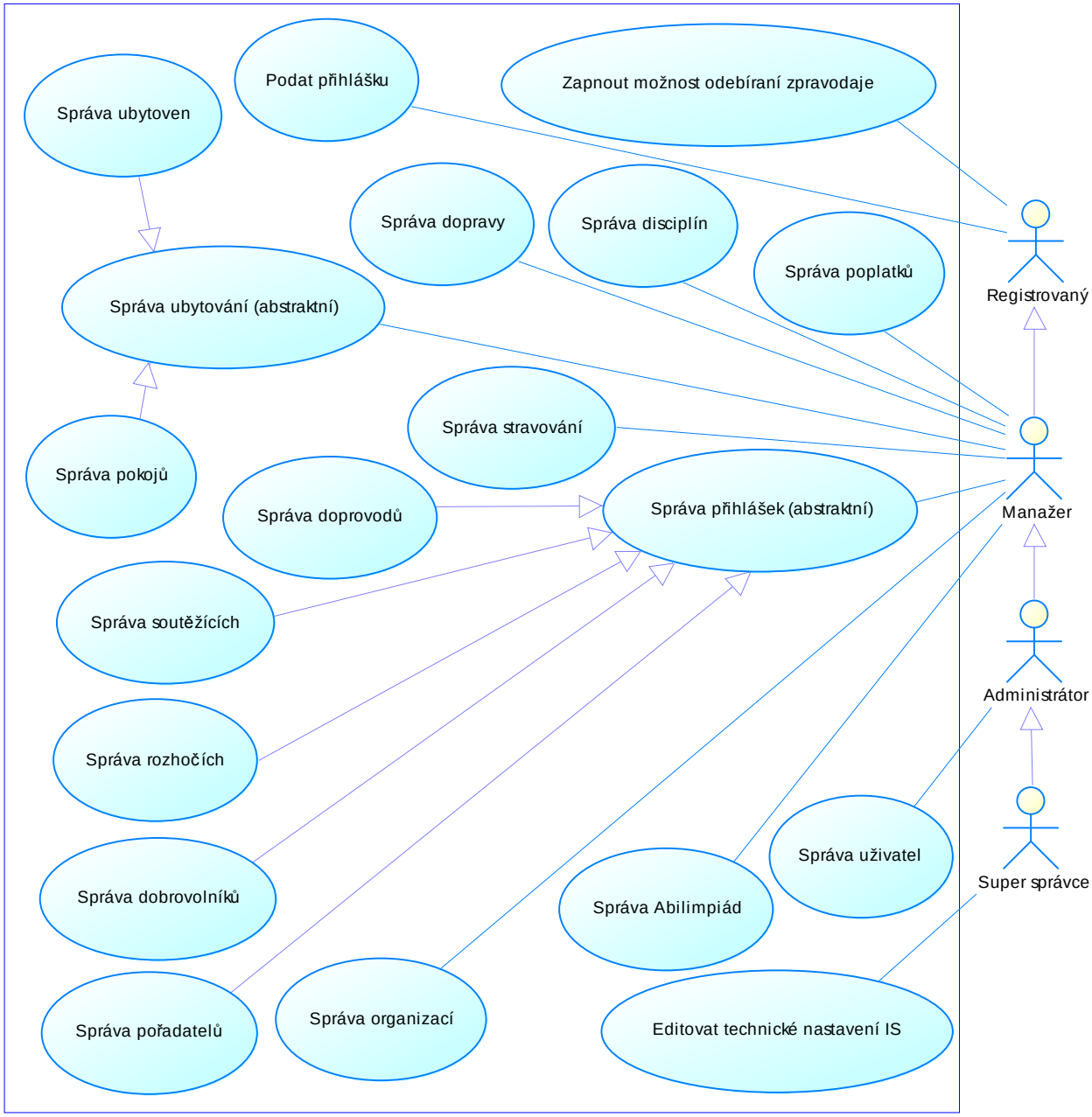
- **BACK-END (správcovské uživatelské rozhraní)**
  - **uživatel musí být vytvořen jinou osobou**, která mu přidělí práva (popř. může zvýšit práva registrovaného uživatele z front-endu pro přístup do back-endu),
  - možnost **přihlásit se** do systému po vytvořený účtu s potřebnými právy pro přístup,
  - možnost spravovat **uživatele** (mazat, přidávat, editovat, zobrazovat),
  - možnost spravovat **přihlášky soutěžící** (mazat, přidávat, editovat, zobrazovat),
  - možnost spravovat **přihlášky doprovodů** (mazat, přidávat, editovat, zobrazovat),
  - možnost spravovat **přihlášky rozhodčích** (mazat, přidávat, editovat, zobrazovat),
  - možnost spravovat **přihlášky dobrovolníků** (mazat, přidávat, editovat, zobrazovat),
  - možnost spravovat **přihlášky pořadatelů** (mazat, přidávat, editovat, zobrazovat),
  - možnost spravovat **poplatky**, které musí platit soutěžící (zobrazovat),
  - možnost spravovat **Abilympijský zpravodaj** (zobrazovat),
  - možnost spravovat **stravování** klientů (mazat, zobrazovat),
  - možnost spravovat **dopravu** klientů (mazat, zobrazovat),
  - možnost spravovat **ubytování** klientů (spravovat ubytovny, spravovat pokoje),
  - možnost spravovat **disciplíny** soutěžících a rozhodčích (mazat, zobrazovat, přidávat),
  - možnost spravovat **Abilympiády** (mazat, přidávat, editovat, zobrazovat).

Všechny tyto funkční požadavky strukturalizovaně ilustrují následující Use Case diagramy. Obrázek 6 znázorňuje abstraktního uživatele, který interaguje s komponentou. Komponentou se rozumí všeobecná datová struktura (viz kapitola 5.3), která nabízí standardně 4 operace (zobrazit záznam, editovat záznam, přidat záznam a smazat záznam).



Obrázek 6 - Use Case diagram - Abstraktní využívání komponenty. Zdroj vlastní

Tyto operace jsou univerzální pro všechny komponenty, a tudíž je možné je zaštítit pojmem „Správa“. Tohoto zevšeobecnění využívá následující Use Case diagram (viz Obrázek 7), a tím přehledně ilustruje využívání informačního systému uživateli. Pro ještě lepší přehlednost je využito zásad generalizace/specializace pro aktéry i samotné use case. Pro úplnost lze konstatovat, že Use Case „Správa dopravy“ a „Správa stravování“ nebude obsahovat možnost „Přidat záznam“ a „Editovat záznam“, jelikož tyto úkony jsou zajišťovány na úrovni „Správy přihlášek“.



Obrázek 7 - Use Case diagram - Využívání IS. Zdroj vlastní

2.2.2 Nefunkční požadavky

Ze strany asociace nebyly pevně stanoveny žádné nefunkční požadavky avšak na základě funkčních požadavků a celkového oboru působnosti CAA byli stanoveny tyto požadavky:

- Technické požadavky:
  - internetový prohlížeč (Firefox, Internet Explorer, aj.),
  - libovolný operační systém, který podporuje internetové prohlížeče a grafický mód.
- Designové požadavky:
  - jednoduché uživatelské rozhraní (jednoduchá šablona),
  - možnost změny velikosti písma.

Designové požadavky vyplývají ze skutečnosti, že většina uživatelů IS (soutěžící) jsou nějakou formou postižení lidé.

2.3 Nástroje

Pro vývoj informačního systému byly zvoleny dva nástroje, které usnadní datové modelování a vytvoří základ pro samotnou implementaci IS.

2.3.1 Datové modelování - PowerDesigner

Pro samotné datové modelování byl na základně osobní zkušenosti zvolen CASE nástroj PowerDesigner od společnosti Sybase. Jedná se o nástroj, který nabízí plně integrované prostředí pro datovou a objektovou analýzu informačních systémů. Přitom plně podporuje zavedené přístupy a metodiky jako je Unified Modeling Language (UML) nebo tříúrovňový návrh databáze. [8]

2.3.2 Implementace – CMS Joomla!

Pro implementaci informačního systému byl na základě osobní zkušenosti zvolen CMS Joomla!. Jedná se o redakční systém pro správu článků, uživatelů, atd. Díky tomu, že se jedná o OpenSource systém, výborně se hodí pro účely této práce. Bude totiž potřeba analyzovat celou strukturu API a následně upravit základní funkce jádra podle výsledků analýzy. Není však stěžejní měnit všechny funkce, protože Joomla! podporuje rozšiřování pomocí komponent, modulů a zásuvných modulů (plug-ins). Jedná se tedy o velice vhodný modulární systém, ke kterému existuje velké množství freeware šablon. Detailně je tento CMS probrán až před samotnou implementací v kapitole 5.

3 Datové modelování – konceptuální úroveň

Cílem datového modelování je navrhnout kvalitní datovou strukturu pro konkrétní aplikaci a databázový systém, který bude tato aplikace využívat k uložení dat.

3.1 Entity

Entity je libovolný objekt reálného světa. Entita musí být rozlišitelná od ostatní entit. Jednotlivé entity specifikované na základě požadavků uvádí Tabulka 3.

Tabulka 3 - Entity. Zdroj vlastní

Číslo	Název entity	Klíčový atribut, atributy
1	Uživatel (jos_users)	id, username, password, jmeno, adresa, datum_narozeni, cislo_op, telefon_pevna, telefon_mobil, email, pohlavi, usertype, block, sendEmail, registerDate, lastvisitDate, activation, params, poznamka, poznamka_int, zprav, nazev_org, aktive_org, isneworg, pozn_org, fotka, foto_type, foto_size, foto_name
2	Přihláška soutěžícího (jos_soutez)	id, slozenka_ano_ne, prukaz_typ, prukaz_cislo, postizeni, kod_sou, vozik_ele, vozik_mech, berle, jina, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
3	Přihláška rozhodčího (jos_rozhod)	id, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
4	Přihláška doprovod (jos_doprovod)	id, slozenka_ano_ne, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
5	Přihláška dobrovolníka (jos_dobrov)	id, slozenka_ano_ne, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
6	Přihláška pořadatele (jos_porad)	id, slozenka_ano_ne, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
7	Disciplíny (jos_disicp)	id, nazev_disc, aktive, poznamka_disc, date_disc, od_disc, do_disc
8	Doprava (jos_doprava)	id, druh_dopravy, datum_prijezdu, spz
9	Stravování (jos_strava)	id, datum_stravy, snidane, obed, vecere
10	Ubytování (jos_ubytovani)	id, datum_ubytovani, ubyt_bool, nazev_uby, adresa, bezb, contact_rec, poznamka_uby, pocet_luzek, cislo_pokoje, poznamka_pok
11	Abilympiáda (jos_abi)	id, aktive, rok, rocnik, misto, mesto, date_first, date_second, poznamka, str_fr, str_se, str_th, str_fo, str_fi, uby_fr, uby_se, uby_th, uby_fo, uby_fi

3.2 Výsledný ER diagram

Na základě požadavků specifikovaných v kapitole 2.2 a entit z přechozí podkapitoly byl vytvořen entitně relační model dat (viz Příloha č. 1). Tento model tvoří základní datovou strukturu budoucího IS na dané úrovni abstrakce (konceptuální úroveň).

4 Návrh – logická úroveň

Návrh v procesu modelování datové základny IS se skládá z rady dílčích kroků. Prvním krokem je transformace vstupního ERD na logický model (relační model dat). Dalším krokem je postupná normalizace logického modelu podle normálních forem (1NF – 3NF).

4.1 Transformace

Transformace ER diagram do relačního modelu dat je proces, který mapuje entity do relací a přiděluje jim vzájemnou referenční integritu.

4.1.1 Referenční integrita

Referenční integrita je definována pomocí cizího klíče, který odkazuje na podřízenou relaci. Tabulka 4 obsahuje seznam relací doplněných o referenční integritu.

Tabulka 4 - Souhrn relací s referenční integritou. Zdroj vlastní

Číslo	Název relace	Primární klíč, Cizí klíče (*_id), Atributy
1	jos_users	id, username, password, jmeno, adresa, datum_narozeni, cislo_op, telefon_pevna, telefon_mobil, email, pohlavi, usertype, block, sendEmail, registerDate, lastvisitDate, activation, params, poznamka, poznamka_int, zprav, nazev_org, aktive_org, isneworg, pozn_org, fotka, foto_type, foto_size, foto_name
2	jos_soutez	id, user_id, abi_id, slozenka_ano_ne, prukaz_typ, prukaz_cislo, postizeni, kod_sou, vozik_ele, vozik_mech, berle, jina, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
3	jos_rozhod	id, user_id, abi_id, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
4	jos_doprovod	id, user_id, abi_id, slozenka_ano_ne, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
5	jos_dobrov	id, user_id, abi_id, slozenka_ano_ne, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
6	jos_porad	id, user_id, abi_id, slozenka_ano_ne, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
7	jos_discip	id, for_id, typ_id, abi_id, nazev_disc, aktive, poznamka_disc, date_disc, od_disc, do_disc
8	jos_doprava	id, for_id, typ_id, abi_id, druh_dopravy, datum_prijezdu, spz
9	jos_strava	id, for_id, typ_id, abi_id, datum_stravy, snidane, obed, vecere
10	jos_ubytovani	id, for_id, typ_id, abi_id, datum_ubytovani, ubyt_bool, nazev_uby, adresa, bezb, contact_rec, poznamka_uby, pocet_luzek, cislo_pokoje, poznamka_pok
11	jos_abi	id, aktive, rok, rocnik, misto, mesto, date_first, date_second, poznamka, str_fr, str_se, str_th, str_fo, str_fi, uby_fr, uby_se, uby_th, uby_fo, uby_fi

4.2 Normalizace

Aby byl relační model dat navržen bez redundancí a tím i bez dalších špatných vlastností z redundance plynoucích se využívá proces **normalizace**. Normalizace je uplatňována pomocí funkčních závislostí nazývaných **normální formy**. [9]

4.2.1 První normální forma (1NF)

Relace je v první normální formě tehdy, když obsahuje pouze atomické atributy. Pokud tomu tak není, musí se relace upravit dekompozicí neatomických atributů. [9]

Relace, které nesplňovaly 1NF jsou opraveny v Tabulka 5.

Tabulka 5 - Úprava relací dle 1NF. Zdroj vlastní

Číslo	Název relace	Primární klíč, Cizí klíče (*_id), Atributy
1	jos_users (atribut adresa)	id, username, password, jmeno, ulice, mesto, psc, datum_narozeni, cislo_op, telefon_pevna, telefon_mobil, email, pohlavi, usertype, block, sendEmail, registerDate, lastvisitDate, activation, params, poznamka, poznamka_int, zprav, nazev_org, aktive_org, isneworg, pozn_org, fotka, foto_type, foto_size, foto_name
2	jos_ubytovani (atribut adresa)	id, for_id, typ_id, abi_id, datum_ubytovani, ubyt_bool, nazev_uby, ulice_uby, mesto, psc, bezb, contact_rec, poznamky_uby, pocet_luzek, cislo_pokoje, poznamka_pok

4.2.2 Druhá normální forma (2NF)

Relace je **ve druhé normální formě** (2NF), jestliže je v první normální formě a každý sekundární atribut je úplně závislý na složeném klíči relace. [9]

Všechny relace jsou v 2NF, jelikož žádná neobsahuje složený klíč.

4.2.3 Třetí normální forma

Relace je ve **třetí normální formě** (3NF), jestliže je ve 2NF a žádný sekundární atribut není tranzitivně závislý na žádném klíči relace. [9]

Všechny relace, které nesplňovali tuto normu, jsou upraveny dekompozicí na větší počet relací (viz Tabulka 6).

Tabulka 6 - Úprava relací dle 3NF. Zdroj vlastní

Číslo	Název relace	Primární klíč, Cizí klíče (*_id), Atributy
1	jos_users	id, fot_id, org_id, psc_id, gid, username, password, jmeno, ulice, datum_narozeni, cislo_op, telefon_pevna, telefon_mobil, email, pohlavi, usertype, block, sendEmail, registerDate, lastvisitDate, activation, params, poznamka, poznamka_int, zprav
1a	jos_foto	id, fotka, foto_type, foto_size, foto_name
1b	jos_organizace	id, nazev_org, aktive_org, isneworg, pozn_org
1c	jos_city	id, psc, mesto
2	jos_soutez	id, abi_id, user_id, slozenka_ano_ne, prukaz_typ, prukaz_cislo, postizeni, datum_podani, changedate, changewho, poznamka, poznamka_int, storno, uby_name
2a	jos_soudop	id, sou_id, dop_id, doprovod_pass
3	jos_discip	id, abi_id, nazev_disc, aktive, poznamka_disc
3a	jos_disc_time	id, dis_id, date_disc, od_disc, do_disc
3b	jos_disc	id, dis_id, for_id, typ_id, order
4	jos_ubytovna	id, cit_id, nazev_uby, ulice_uby, bezb, contact_rec, poznamky_uby
4a	jos_pokoj	id, abi_id, uby_id, pocet_luzek, cislo_pokoje, poznamka_pok
4b	jos_ubyt	id, pok_id, for_id, abi_id, typ_id, datum_ubytovani, ubyt_bool

4.2.4 Výsledný normalizovaný relační model dat

Proces normalizace dekomponoval relační datový model, tudíž je tento model nyní konzistentní a odolný vůči anomáliím způsobeným redundancí. Výsledný relační model dat není v této práci uveden, avšak může být zaměněn fyzickým datovým modelem, který je obsažen v příloze č. 2. Rozdíly mezi relačním a fyzickým datovým modelem jsou minimální a v podstatě se jedná pouze o rozdílnou definici datových typů jednotlivých atributů (fyzický datový model je zatížen na MySql).

.

5 Základní funkčnost a vlastnosti CMS Joomla!

Základní funkčnost CMS Joomla! je dána řadou datových struktur, které tvoří základní architekturu tohoto nástroje. Tyto struktury jsou důležité pro samotnou implementaci, jelikož se podle této architektury musí celý IS implementovat.

5.1 Datové struktury CMS Joomla!

Jelikož je CMS Joomla! redakčním systémem (popř. portálový systém) má dobrý základ pro zveřejňování obsahu. S tím souvisí i spousta nástrojů pro user-friendly prostředí. Základní datové struktury využívané CMS Joomla! shrnuje Tabulka 7.

Tabulka 7 - Datové struktury. Zdroj vlastní

Název	Popis
Komponenty	Komponenty v systému Joomla! jsou v podstatě datové struktury, které interagují s databází a zpracovávají, zobrazují či ukládají data.
Moduly	Moduly jsou určeny k tvorbě struktur, které se zobrazují uživateli v prohlížeči. Modul je např. menu, okno s možností přihlášení, bannery, atd.
Plug-iny	Plug-in neboli zásuvný modul, je určen pro rozšíření funkcionalit systému. Jedná se např. o možnosti různých forem autentizace (LDAP, GMail, atd.), funkcí na posílání hromadných emailů, editory textu, atd.

5.2 Bezpečnost CMS Joomla!

CMS Joomla! má širokou komunitu uživatelů, kteří ho testují z hlediska funkčnosti i bezpečnosti. Tato kapitola úzce souvisí s cílem této práce, a to s bezpečnostní výsledného systému.

5.2.1 Autentizace

K autentizaci neboli ověření identity uživatele se využívá unikátní **uživatelské jméno** a **heslo**. Dále musí každý uživatel mít svoji **emailovou adresu**, pomocí níž může dokončit registraci.

5.2.2 Autorizace

Autorizace (schválení, povolení) kontroluje práva konkrétního ověřeného uživatele dle **ACL** (viz kapitola 6.3.1). Pokud uživatel nemá přístup k dané části IS, je mu přístup zamítnut.

5.2.3 Ověřovací token

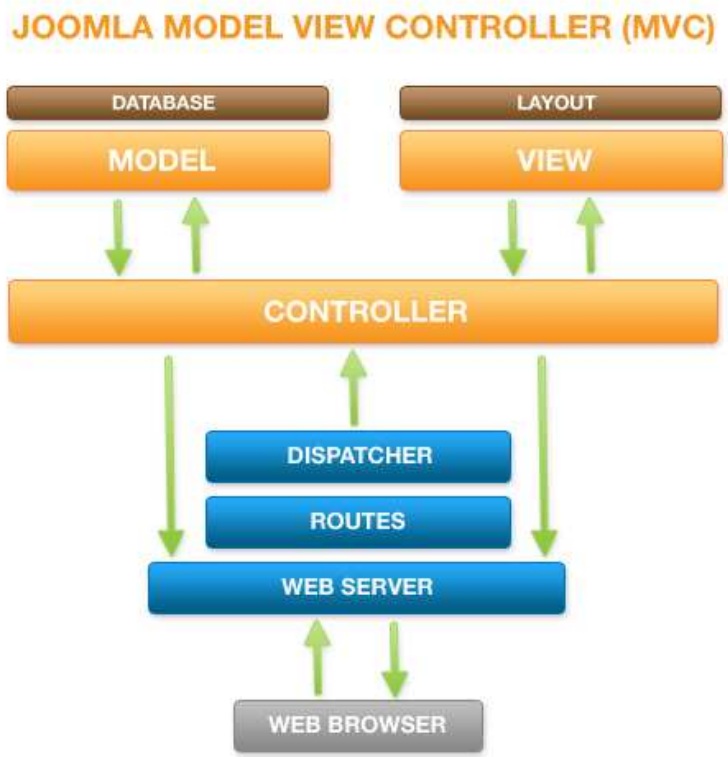
Ověřovací token s určitou dobou platnosti slouží k zajištění jednoznačnosti přístupu. Přístup k jakékoli části CMS Joomla! je realizován přes **index.php**. Každému uživateli přihlášenému do IS se

vygeneruje **token**, který je ověřován při vstupu na jakýkoli soubor IS. Tím je zajištěna vysoká bezpečnost a znemožněn přístup na jakoukoli stránku IS bez přihlášení.

Ověřovací token slouží také při registraci nových uživatelů. Náhodný **token** je vygenerován systémem a poslán uživateli na email. Při prvním přihlášení musí uživatel zadat tento **token** pro ověření.

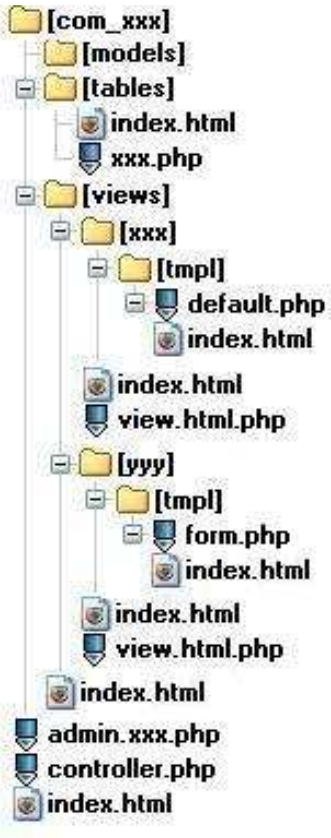
5.3 Architektura MVC (Model-View-Controller)

Celý Framework CMS Joomla! je napsaný v objektově orientovaném PHP. K tvorbě komponent využívá architekturu MVC. MVC je softwarová architektura, která odděluje datový model, uživatelské rozhraní a řídící logiku do tří nezávislých celků s vlastním rozhraním (viz Obrázek 8).



Obrázek 8 - Architektura MVC. Zdroj [10]

V prostředí souborového systému (file system) je komponenta nejčastěji uspořádaná viz Obrázek 9. Pro tuto práci je toto schéma universální. Po nahrazení *xxx* názvem libovolné komponenty (např. soutěžící) a *yyy* názvem libovolného formuláře (např. přihláška) je možné zkonstruovat libovolnou komponentu. Soubor **admin.xxx.php** je v tomto stromu hlavním přístupovým bodem ke komponentě.

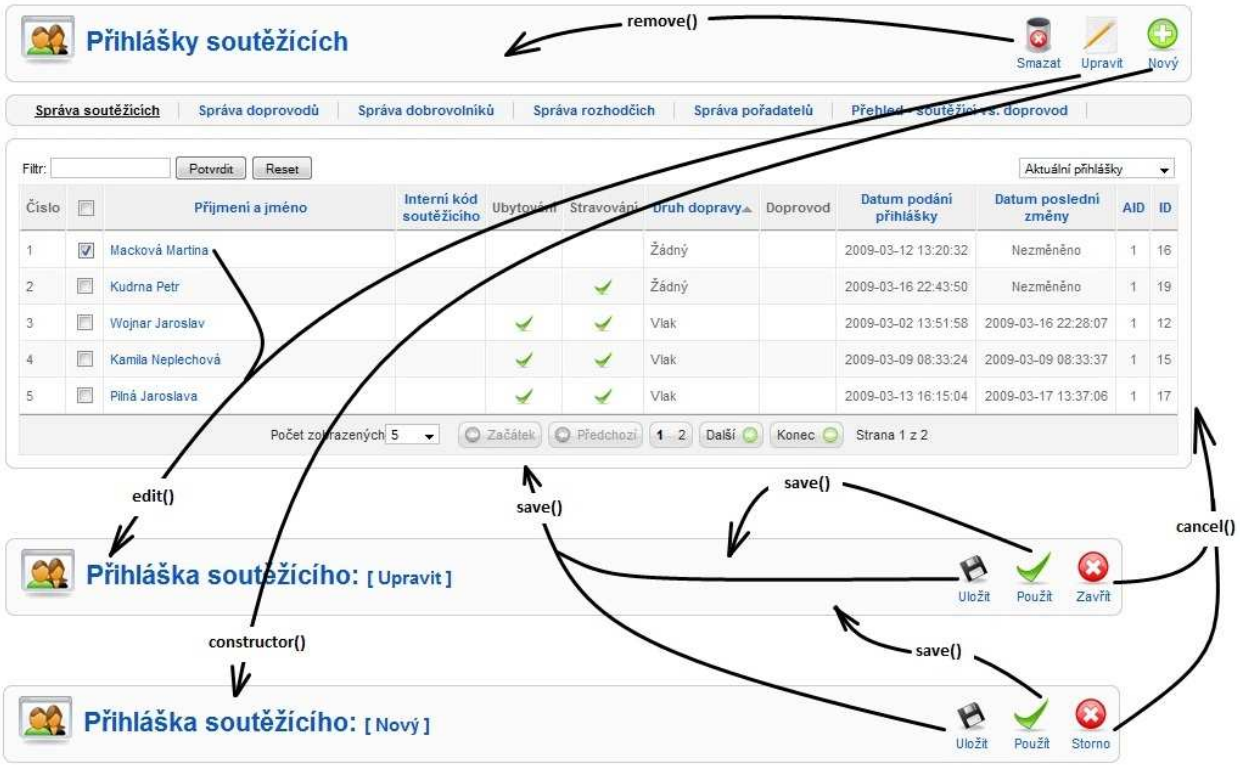


Obrázek 9 - Všeobecná struktura komponenty. Zdroj vlastní

- **Model (model)** je v podstatě připravený objekt (objekty), se kterým později pracuje **Controller**. Objekt se vytvoří pomoci **SQL** dotazů do **MySql** databáze.

- **View (pohled)** pracuje s **modelem**. Vhodným způsobem ho zpracuje a zobrazí uživateli na obrazovce. **View** se dále rozděluje na **šablonu**, která formátuje výstup na obrazovce a na část, která pracuje s **modelem**.

- **Controller (řadič)** zajišťuje reakce na podněty od uživatele a přizpůsobuje **model** a **pohled** požadavkům. Těmito podněty může být např. mazání, ukládání, editace, atd. Každý tento podnět je definován jako funkce ve třídě řadiče (viz Obrázek 10).



Obrázek 10 - Příklad fungování řadiče. Zdroj vlastní

6 Implementace návrhu

Tato kapitola je pojata jako případová studie komponenty pro správu soutěžících. Další komponenty jsou pouze analogií na tuto komponentu a jsou zde zmíněny pouze okrajově. Konkrétní syntax jazyka není v této práci stěžejní, a proto zde není uváděna. Všechen zdrojový kód je však možné nalézt na přiloženém DVD (viz Příloha č. 3).

6.1 Fyzický datový model

Na základě požadavků a datového modelování byl navrhnut fyzický datový model, který zaručí všechny funkční potřeby kladené na IS. (viz Příloha č. 2).

Na základě fyzického datového modelu je vygenerována sada SQL dotazů, které vytvoří jednotlivé tabulky v databázi. Základní instalace CMS Joomla! obsahuje 36 tabulek. Pro potřeby přestavby bylo vytvořeno dalších 18 tabulek a jedna struktura tabulky byla přetvořena (JOS_USERS). Vnitřní struktura tabulek a jejich vzájemné propojení jasně vyplývá z datového modelování (viz kapitoly 3 a 4) a proto již v dalším textu nebude zmiňována.

6.2 Základní struktura budoucího IS

Struktura budoucího IS je určena komplexní strukturou komponent, které pokrývají všechny definované požadavky. Komponenty se dělí na 2 skupiny. První skupinou jsou komponenty pro správcovskou část IS, které poskytují komplexní funkčnost IS (viz Tabulka 8).

Tabulka 8 - Komponenty správcovské části IS. Zdroj vlastní

Název	Komponenta	Charakteristika
Správa soutěžících	COM_SOUTEZ	Smyslem komponenty je správa přihlášek od soutěžících. Správce bude mít zobrazeny všechny podané přihlášky. Každou přihlášku bude moci zobrazit a upravit většinu údajů (kromě jména soutěžícího). Také bude mít možnost napsat novou přihlášku a vybrat ze seznamu zaregistrovaných uživatelů, kteří ještě nemají podanou žádnou přihlášku na soutěžení pro konkrétní rok.
Správa doprovodů	COM_DOPROVOD	Komponenta se stará o přihlášky doprovodů, kteří doprovázejí soutěžící. Správce má zobrazen seznam všech přihlášek. Každou přihlášku bude moci zobrazit a upravit většinu údajů, dle potřeby (kromě jména doprovodu). Také bude mít možnost napsat novou přihlášku a vybrat ze seznamu zaregistrovaných uživatelů, kteří již nejsou v konkrétním roce doprovodem nebo soutěžícím. Na úvodní obrazovce komponenty budou zobrazeny nejdůležitější údaje z přihlášek. Nechybí ani filtry pro potřeby organizace (stornované přihlášky, hledání).
Správa rozhodčích	COM_ROZHOD	Komponenta umožňuje spravovat přihlášky pro rozhodčí. Správce bude mít zobrazeny všechny podané přihlášky. Každou přihlášku bude

		moci zobrazit a upravit většinu údajů (kromě jména rozhodčího). Také bude mít možnost napsat novou přihlášku a vybrat ze seznamu zaregistrovaných uživatelů, kteří již nejsou rozhodčím v daném roce. Na úvodní obrazovce komponenty budou zobrazeny nejdůležitější údaje z přihlášek. Nechybí ani filtry pro potřeby organizace (stornované přihlášky, hledání).
Správa dobrovolníků	COM_DOBROV	Komponenta umožňuje spravovat přihlášky pro dobrovolníky. Správce bude mít zobrazeny všechny podané přihlášky. Každou přihlášku bude moci zobrazit a upravit většinu údajů (kromě jména dobrovolníka). Také bude mít možnost napsat novou přihlášku a vybrat ze seznamu zaregistrovaných uživatelů, kteří již nejsou dobrovolníky v daném roce. Na úvodní obrazovce komponenty budou zobrazeny nejdůležitější údaje z přihlášek. Nechybí ani filtry pro potřeby organizace (stornované přihlášky, hledání).
Správa pořadatelů	COM_PORAD	Komponenta umožňuje spravovat přihlášky pro pořadatele. Správce bude mít zobrazeny všechny podané přihlášky. Každou přihlášku bude moci zobrazit a upravit většinu údajů (kromě jména pořadatele). Také bude mít možnost napsat novou přihlášku a vybrat ze seznamu zaregistrovaných uživatelů, kteří již nejsou pořadateli v daném roce. Na úvodní obrazovce komponenty budou zobrazeny nejdůležitější údaje z přihlášek. Nechybí ani filtry pro potřeby organizace (stornované přihlášky, hledání).
Správa interakce soutěžící vs. doprovod	COM_SOUDOP	Komponenta spravuje interakci mezi soutěžícími a doprovody. Správa spočívá v přehledném zobrazení soutěžících včetně jejich doprovodů. Tato komponenta má zásadní význam hlavně z důvodů toho, že soutěžící může mít více doprovodů, nebo naopak doprovod může mít více soutěžících.
Správa disciplín	COM_DISCIP	Komponenta umožňuje spravovat disciplíny. Správce bude mít možnost přidávat nové disciplíny a zobrazovat si seznam všech disciplín. Hlavním účelem této komponenty je přehled všech vybraných disciplín. Na základě tohoto přehledu může CAA zajistit materiál i dostatečné prostory pro jednotlivé disciplíny.
Správa dopravy	COM_DOPRAVA	Komponenta spravuje všechny záznamy o dopravě, které byly získány od klientů (soutěžící, dobrovolník, atd.). Správa spočívá ve filtrování jednotlivých druhů dopravy. Smysl této komponenty spočívá v zjištění počtu klientů, kteří přijedou na Abilympiádu v určitý čas a určitým druhem dopravy. Na základě těchto zjištění může CAA zajistit např. dopravu od vlakového nádraží k místu konání Abilympiády pro potřebný počet lidí.
Správa stravování	COM_STRAVA	Komponenta spravuje všechny záznamy o stravě, které byly získány od klientů (soutěžící, dobrovolník, atd.). Správa spočívá ve filtrování jednotlivých druhů stravy (snídaně, oběd, večeře), data stravování a typu klienta. Je zde tedy možné zjistit, jaké množství různých druhů jídel bude potřeba zajistit v určitý den pro klienty.
Správa ubytování	COM_UBYTOVANI	Komponenta spravuje všechny události spojené s ubytováním. Jedná se o velice komplexní komponentu, která zaštituje správu ubytoven, pokojů, přehled obsazenosti pokojů a přehled klientů požadujících ubytování.
Správa	COM_POPLATKY	Komponenta spravuje všechny záznamy o poplatcích, které jsou

poplatků		uloženy v tabulkách JOS_SOUTEZ a JOS_DOPROVOD. Správa spočívá v zobrazení seznamu soutěžících a doprovodů, kteří mají buď zaplacený, nebo nezaplacený poplatek.
Správa organizací	COM_ORGANIZ	Komponenta umožňuje spravovat organizace, které může uživatel systému vyplnit při registraci.
Správa zpravodaje	COM_ZPRAVODAJ	Komponenta spravuje všechny záznamy o zpravodaji, které jsou uloženy v tabulce JOS_USERS. Správa spočívá v zobrazení seznamu všech klientů (vč. adresy), kteří si přejí odebírat Abilympijský zpravodaj.
Správa Abilympiád	COM_ABI	Komponenta je v podstatě řadič celého IS. Umožňuje vybrat jednu aktivní Abilympiádu, podle které se budou dále diferencovat všechny komponenty.

Druhou skupinou jsou komponenty veřejné uživatelské části (viz Tabulka 9).

Tabulka 9 - Komponenty veřejné části IS. Zdroj vlastní

Název	Komponenta	Charakteristika
Správa soutěžících	COM_SOUTEZ	Komponenta, která zobrazí přihlášku soutěžícího a umožní klientovi ji uložit, nebo stornovat již podanou přihlášku
Správa doprovodů	COM_DOPROVOD	Komponenta, která zobrazí přihlášku doprovodu a umožní klientovi ji uložit, nebo stornovat již podanou přihlášku
Správa rozhodčích	COM_ROZHOD	Komponenta, která zobrazí přihlášku rozhodčího a umožní klientovi ji uložit, nebo stornovat již podanou přihlášku
Správa dobrovolníků	COM_DOBROV	Komponenta, která zobrazí přihlášku dobrovolníka a umožní klientovi ji uložit, nebo stornovat již podanou přihlášku
Správa pořadatelů	COM_PORAD	Komponenta, která zobrazí přihlášku pořadatele a umožní klientovi ji uložit, nebo stornovat již podanou přihlášku
Správa zpravodaje	COM_ZPRAVODAJ	Jednoduchá komponenta, která obsluhuje možnost klientů odebírat Abilympijský zpravodaj

6.3 Uživatelé informačního systému

Nejdůležitější součástí každého informačního systému jsou jeho uživatelé. V souvislosti s uživateli však nastává otázka bezpečnosti jejich údajů a samozřejmě ochrana proti neoprávněnému přístupu. Nedílnou součástí je samozřejmě diferenciace uživatelů do skupin, které určí práva uživatelů.

6.3.1 Přístupová práva

CMS Joomla! využívá pro správu přístupových práv knihovnu phpGACL (generic access control list). Tato knihovna využívá třístupňový pohled na realitu (ACO, ARO, AXO). Všechny objekty, které požadují přístup k čemukoli, se nazývají **ARO** (Access Request Objects). Těmito objekty jsou v tomto případě uživatelé, kteří požadují, či je jim přidělena určitá forma přístupových práv. **ACO** (Access Control Objects) představují aktivity, které mohou být uživatelům (ARO) přiděleny. Příkladem aktivity

může být editace, pouze pro čtení, atd. Posledním objektem je **AXO** (Access eXtension Objects), který zaštituje všechny objekty, ke kterým mají být práva přidělena. V podstatě daná trojice v phpGALC nám říká – **kdo** má právo, na **co** a v **jakém** objektu toto právo má. [11]

6.3.2 Uživatelské skupiny

Pokud jsou nastaveny mechanismy, které zajistí strukturu přístupových práv pro jakékoli objekty, je na řadě utvořit skupiny uživatelů s podobnými právy.

Hlavním rozdělením uživatelů je na **uživatele s přístupem do správcovské části IS** a **uživatele s přístupem do veřejné části IS**. Z hlediska analýzy a požadavků ze strany asociace byla skupina správcovské části dále rozdělena na skupiny **Super správců, Administrátorů a Manažerů** jejich prává shrnuje Tabulka 10.

Tabulka 10 - Uživatelské skupiny. Zdroj vlastní

Skupina	Práva
Super správce	Všechna práva
Administrátor	Má všechna práva na správu obsahu, ale nemůže měnit technické nastavení CMS Joomla!
Manažer	Práva jako Administrátor, pouze nemůže spravovat uživatele systému

6.3.3 Přihlašování a registrace uživatelů

Standardní možnosti CMS Joomla! mají pevně danou strukturu údajů, které potřebuje uživatel k registraci. Potřeba CAA je však, dle požadavků poněkud jiná, proto je potřeba změnit standardní API. Konkrétní realizace změn spočívá v doplnění několika atributů do datového základu komponenty pro správu uživatelů a posléze tyto změny promítnout do funkčního modelu.

Přihlašování již registrovaných klientů probíhá klasickým způsobem pomocí **uživatelského jména** a **hesla**.

6.4 Komponenta COM_SOUTEZ (Správa soutěžících)

Komponenta COM_SOUTEZ je jednou z mnoha komponent, které informační systém obsahuje.

Veřejná komponenta COM_SOUTEZ umožňuje pouze obsluhu jedné konkrétní přihlášky, a to přihlášky přihlášeného uživatele (soutěžícího). Na základě této veřejné komponenty je přihláška zobrazena, poté uživatelem vyplněna a její obsah je dále uložen do proměnlivého počtu databázových tabulek. Počet tabulek se liší podle toho, zda soutěžící požaduje určitou „službu“

v přihlášce (např. ubytování, stravování, apod.). Podrobně je tato problematika znázorněna na Obrázek 11 v kapitole 6.5.2. Komponenta dále umožňuje stornovat již podanou přihlášku, popř. znovuobnovit stornovanou přihlášku.

Komponenta COM_SOUTEZ ve správcovské části umožňuje celkovou správu všech podaných přihlášek z veřejné části. Tato správa spočívá v zobrazování seznamu všech již podaných přihlášek, možnost jejich mazání, editace či případné podání nové přihlášky. Nad seznamem přihlášek lze provádět řadu dalších operací, které jsou podrobně probrány v kapitole 6.6.

6.5 Vynucování referenční integrity

Jak již bylo zmíněno v kapitole 4, referenční integrita se stará o obsahovou správnost dat a tudíž o konzistenci celé databáze. Jelikož v základní verzi databázového stroje MySql není žádný nástroj, který zajišťuje dodržování referenční integrity dat, musí být tato integrita vynucena programově. Referenční integrita je programově ošetřena v řadiči (controller) každé komponenty.

6.5.1 Vkládání nových záznamů

Při vkládání nových záznamů se využívá referenční integrity pro správné ukládání dat do databáze. Jedná se o přiřazování cizích klíčů do podřízených tabulek. V komponentě COM_SOUTEZ se jedná o:

- přidělení ID aktuální Abilympiády jako cizí klíč do přihlášky a tím jasně přiřadit přihlášku konkrétní Abilympiádě,
- přidělení ID konkrétního uživatele jako cizí klič do přihlášky a tím jasně definovat, který uživatel přihlášku podal,
- přidělení ID přihlášky jako cizí klíč do záznamů o disciplínách,
- přidělení ID přihlášky jako cizí klíč do záznamů o doprovodech (pokud bylo v přihlášce vybráno),
- přidělení ID přihlášky jako cizí klíč do záznamů o stravování, ubytování a dopravy (pokud bylo v přihlášce vybráno).

6.5.2 Změna existujících záznamů

Pokud je změněn některý záznam, musí být zaručeno dodržení referenční integrity mezi tabulkami. V komponentě COM_SOUTEZ jde především o správné uložení doprovodů soutěžícího, dopravy, stravování a ubytování. Univerzální schéma pro změnu, či přidání záznamů znázorňuje Obrázek 11.

U **doprovodu** mohou nastat 3 základní problémy:

- soutěžící původně neměl doprovod, ale nyní chce,

- soutěžící měl doprovod, ale nyní chce více doprovodů,
- soutěžící měl doprovod, ale nyní chce méně doprovodů.

U **stravování** mohou nastat následující situace:

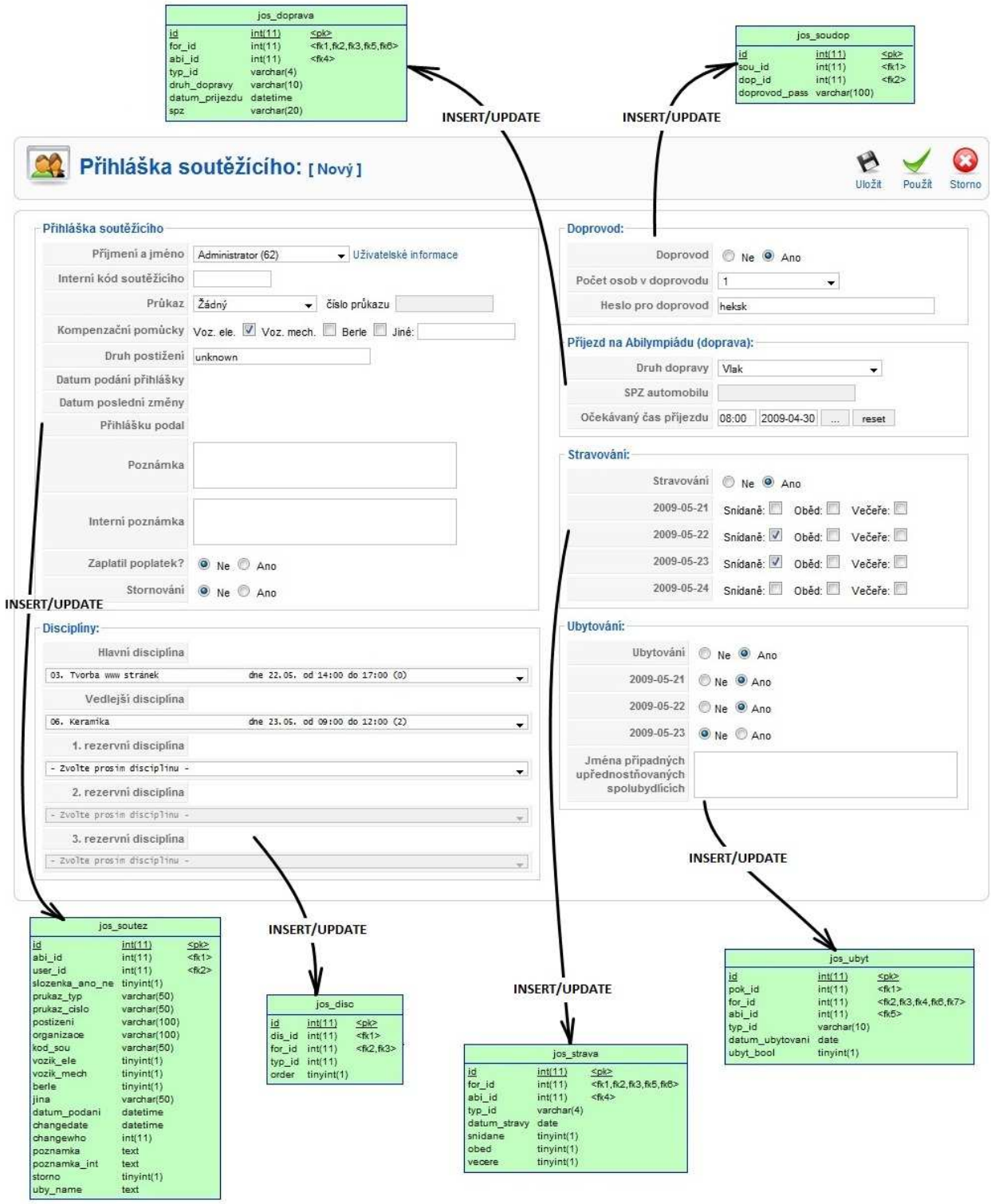
- soutěžící chtěl stravování, ale nyní nechce,
- soutěžící nechtěl stravování, ale nyní chce.

Jelikož u stravování je rezervováno v tabulce JOS_STRAVA tolik řádků, kolik je celkem dnů stravování, není zde potřebné řešit, které dny chce soutěžící stravu. Pokud nechtěl stravování a nyní chce, vytvoří se v tabulce JOS_STRAVA tolik záznamů kolik je celkem dní. Naopak, pokud stravování již nechce, všechny záznamy tak zůstávají, pouze se vynulují hodnoty stravování (snídaně, oběd, večeře).

Ubytování funguje na stejném principu jako stravování.

V případě **disciplín** je dodržení referenční integrity následující:

- při každé editaci soutěžícího se exitující záznamy v tabulce JOS_DISC upraví (odstraní se odkaz na konkrétního soutěžícího),
- při uložení se vyhledají v tabulce „volné disciplíny“ (disciplíny bez soutěžícího) a přidělí se odkaz na konkrétního soutěžícího,
- pokud neexistují žádné „volné disciplíny“ vytvoří se nový záznam v tabulce JOS_DISC.

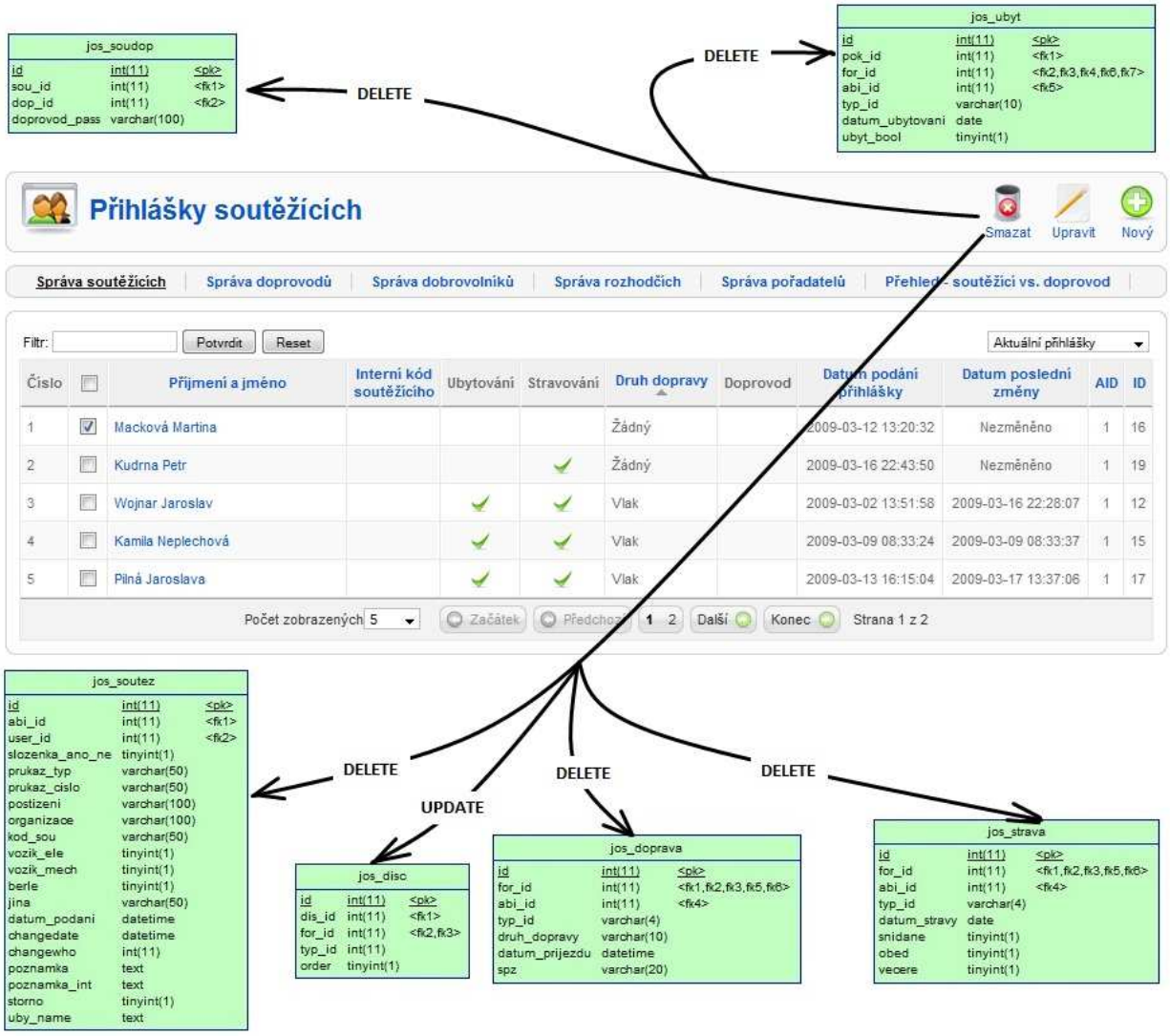


Obrázek 11 - Správa soutěžících – Přidávání/Editace záznamů. Zdroj vlastní

6.5.3 Mazaní existujících záznamů

Při mazání celých záznamů z tabulky JOS_SOUTEZ jsou z důvodu dodržení referenční integrity smazány data i s tabulek JOS_UBYT (správa ubytování), JOS_STRAVA (správa stravování), JOS_DOPRAVA (správa dopravy), JOS_SOUDOP (správa doprovodu soutěžících) a JOS_DISC (přidělené

disciplíny). V tabulce JOS_DISC se záznam nemaže, pouze se upraví smazáním odkazu na soutěžícího. Schéma celé této operace znázorňuje Obrázek 12.



Obrázek 12 - Správa soutěžících - Mazání záznamů. Zdroj vlastní

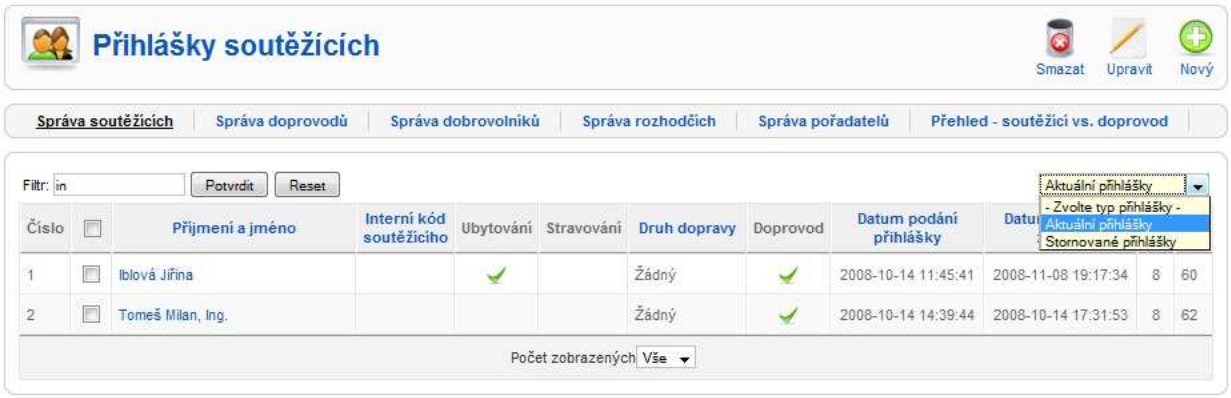
6.6 Registry

V CMS Joomla! je implementován registr, který ukládá pro jednotlivá sezení (session) změny filtrů, řazení a navigace. Každá komponenta má jeden registr.

6.6.1 Filtry

Filtry slouží k upřesnění výběru záznamů z databáze dle uživatelských požadavků. Jedná se o pevně nadefinované filtry, které vycházejí z analytického modelování. Dále každá komponenta obsahuje filtr „Hledat“, kde může uživatel zadat hledaný výraz (liší se dle komponent). V komponentě COM_SOUTEZ je napravo filtr dle typu přihlášky. Je zde možné filtrovat aktuální přihlášky a zvlášť

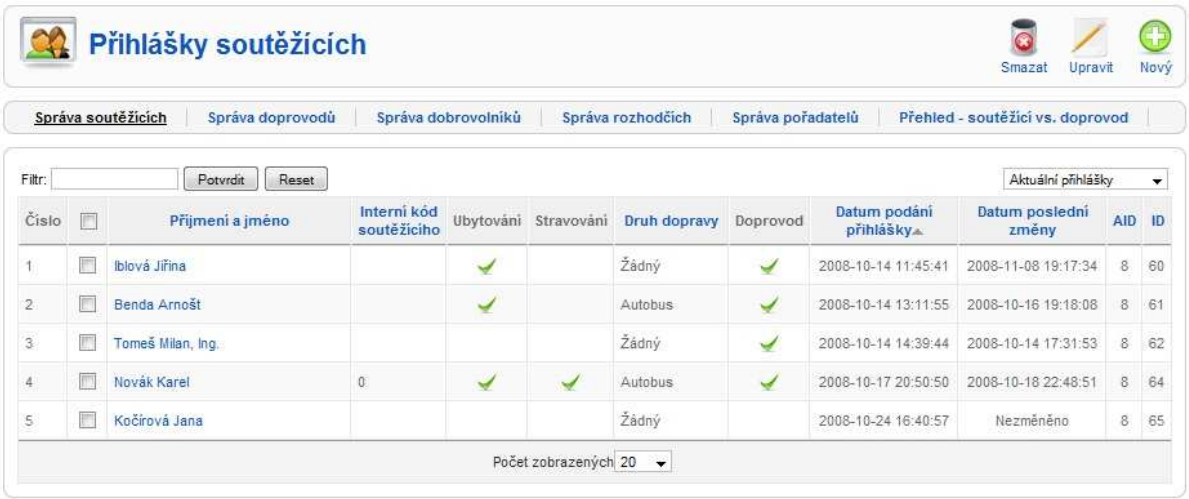
stornované přihlášky (implicitně je filtr nastaven na aktuální přihlášky). V levé části je filtr, který vypíše záznamy, které se shodují ve jménu soutěžícího s hledaným řetězcem (viz Obrázek 13).



Obrázek 13 - Správa soutěžících - Filtry. Zdroj vlastní

6.6.2 Řazení

Řazení zobrazených záznamů dle jednotlivých sloupců slouží pro lepší administraci. Pro konkrétní sezení (session) je nastavení řazení uloženo do registru a až do odhlášení uživatele zůstává. V komponentě COM_SOUTEZ je možné řadit dle sloupců, které mají nadpis modrou barvou (příjmení a jméno, interní kód soutěžícího, aj.). Na Obrázek 14 jsou záznamy řazeny dle data podání přihlášky vzestupně.

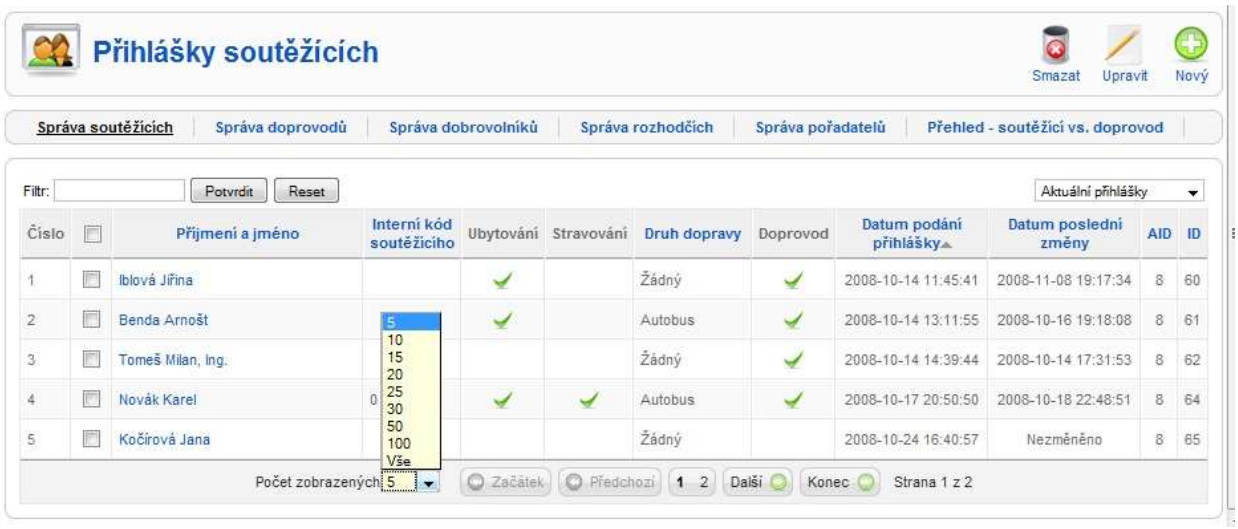


Obrázek 14 - Správa soutěžících - Řazení. Zdroj vlastní

6.6.3 Navigace

V dolní části každého seznamů záznamů je navigace, která slouží pro výběr počtu zobrazovaných záznamů na stránku a pro pohyb mezi jednotlivými stránkami (viz Obrázek 15). Navigace je na rozdíl od filtrů a řazení uložena v globálním registru a tudíž uplatněna na všechny komponenty, tzn., pokud

uživatel změní počet záznamů na stránce, projeví se tato změna ve všech komponentách (pro konkrétní sezení).



Obrázek 15 - Správa soutěžících - Navigace. Zdroj vlastní

6.7 Ošetření uživatelský interakcí

Ošetření uživatelských interakcí je dosaženo pomocí jazyka JavaScript, který kontroluje definované podmínky a popř. uživatele upozorní na výjimku či předem stanovenou událost.

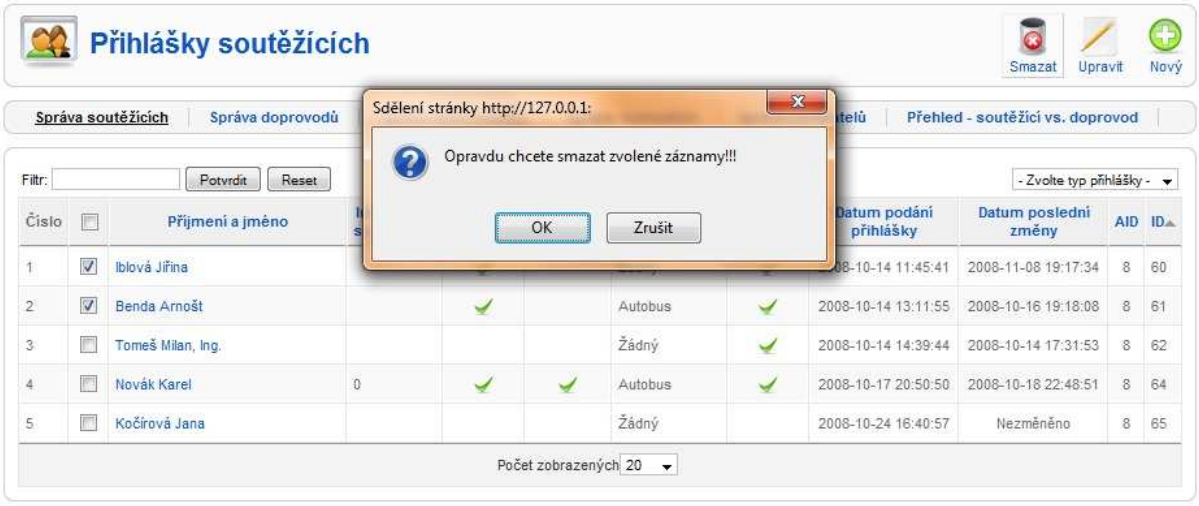
6.7.1 Formuláře

Každý formulář v IS je určitým způsobem ošetřen jazykem JavaScript. První a v zásadě nejdůležitější interakcí je ověření vyplnění všech požadovaných políček ve formuláři. Pokud uživatel nevyplnil (popř. neoznačil, nevybral) nějaké políčko, je posléze upozorněn oknem s upozorněním o problému. Souhrn všech nutných ošetřených výjimek obsahuje kapitola 7.

Druhou stránkou ošetření formulářů je jejich dynamické přizpůsobování aktuálnímu kroku uživatele. V přihlášce komponenty COM_SOUTEZ se jedná o dynamickou kontrolu disciplín, tj. uživatel si nemůže vybrat 2 stejné disciplíny a zároveň disciplíny, které se kryjí v časech. V souvislosti s disciplínami se i dynamický zpřístupní další disciplína, pokud je první vybrána. Stejně je tomu i u doprovodu, stravování a ubytování. Pokud uživatel zvolí, že chce doprovod, stravování nebo ubytování automaticky se mu zpřístupní následné volby.

6.7.2 Mazání

V administrátorské části IS jsou všechny záznamy v databázi chráněny proti smazáním pomocí přidělených práv uživatelů. Ti uživatelé, kteří mají právo mazat záznamy, jsou pomocí JavaScriptu upozorněni, zda opravdu chtějí dané záznamy smazat (viz Obrázek 16).



Obrázek 16 - Správa soutěžících - Ošetřené mazání záznamů. Zdroj vlastní

7 Ošetření všech komponent pomocí JavaScriptu

Aby byla zajištěna referenční integrita databáze, je nutné všechny uživatelské interakce se systémem ošetřit proti výjimkám.

7.1 Komponenta COM_USERS

Formulář uživatele upravené komponenty COM_USERS je nutné ošetřit z důvodu přidání dalších atributů (viz Tabulka 11).

Tabulka 11 - Ošetřené výjimky komponenty COM_USERS. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vyplněno jméno uživatele • musí být vyplněno uživatelské jméno • musí být vyplněno heslo (vč. potvrzení hesla) • musí být vyplněn email • musí být vybráno pohlaví • musí být vyplněna ulice, město a PSČ • musí být zadáno datum narození uživatele • musí být vyplněno číslo občanského průkazu • nesmí být vybrán cizí jazyk bez úrovně znalostí a opačně
Front-end	• musí být vyplněno jméno uživatele • musí být vyplněno uživatelské jméno • musí být vyplněno heslo (vč. potvrzení hesla) • musí být vyplněn email • musí být vybráno pohlaví • musí být vyplněna ulice, město a PSČ • musí být zadáno datum narození uživatele • musí být vyplněno číslo občanského průkazu • nesmí být vybrán cizí jazyk bez úrovně znalostí a opačně

7.2 Komponenta COM_SOUTEZ

Formulářové ošetření komponenty COM_SOUTEZ proti výjimkám spočívá v podobě výstražných JavaScriptových oken, které informují uživatele o tom co je špatně (viz Tabulka 12).

Tabulka 12 - Ošetřené výjimky komponenty COM_SOUTEZ. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vybráno jméno soutěžícího • musí být vybrán typ průkazu (vč. čísla průkazu) • musí být vyplněn typ postižení soutěžícího • musí být vybrána min. jedna disciplína • pokud soutěžící chce doprovod, musí vyplnit počet doprovodu (vč. ověřovacího klíče) • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky
Front-end	• musí být vybrán typ průkazu (vč. čísla průkazu) • musí být vyplněn typ postižení soutěžícího • musí být vybrána min. jedna disciplína • pokud soutěžící chce doprovod, musí vyplnit počet doprovodu (vč. ověřovacího klíče) • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky

7.3 Komponenta COM_DOPROVOD

Ošetření výjimek v komponentě COM_DOPROVOD je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 13).

Tabulka 13 - Ošetřené výjimky komponenty COM_DOPROVOD. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vybráno jméno doprovodu • musí být vybrán min. 1 soutěžící pro doprovod (vč. zadání správného ověřovacího klíče) • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky
Front-end	• musí být vybrán min. 1 soutěžící pro doprovod (vč. zadání správného ověřovacího klíče) • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky

7.4 Komponenta COM_ROZHOD

Ošetření výjimek v komponentě COM_ROZHOD je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 14).

Tabulka 14 - Ošetřené výjimky komponenty COM_ROZHOD. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vybráno jméno rozhodčí • musí být zvolena min. 1 disciplína • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky
Front-end	• musí být zvolena min. 1 disciplína • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky

7.5 Komponenta COM_DOBROV

Ošetření výjimek v komponentě COM_DOBROV je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 15).

Tabulka 15 - Ošetřené výjimky komponenty COM_DOBROV. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vybráno jméno dobrovolníka • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky
Front-end	• musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky

7.6 Komponenta COM_PORAD

Ošetření výjimek v komponentě COM_PORAD je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 16).

Tabulka 16 - Ošetřené výjimky komponenty COM_PORAD. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vybráno jméno pořadatele • musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky
Front-end	• musí být zvolen typ dopravy (vč. data příjezdu a SPZ pokud je o automobil) • pokud chce soutěžící stravování, musí vybrat min. 1 jídlo z nabídky • pokud chce soutěžící ubytování, musí vybrat min. 1 den z nabídky

7.7 Komponenta COM_UBYTOVANI

Ošetření výjimek v komponentě COM_UBYTOVANI je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 17).

Tabulka 17 - Ošetřené výjimky komponenty COM_UBYTOVANI. Zdroj vlastní

	Ošetřené výjimky
Back-end	Správa ubytoven: • musí být vyplněn název ubytovacího zařízení • musí být vyplněna ulice • musí být vyplněno město • musí být vyplněno PSČ Správa pokojů: • musí být vyplněno číslo pokoje • musí být vyplněno ubytovací zařízení, do kterého pokoj patří • musí být vyplněn počet lůžek

7.8 Komponenta COM_DISCIP

Ošetření výjimek v komponentě COM_DISCIP je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 18).

Tabulka 18 - Ošetřené výjimky komponenty COM_DISCIP. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vyplněn název disciplíny

7.9 Komponenta COM_ORGANIZACE

Ošetření výjimek v komponentě COM_ORGANIZACE je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 19).

Tabulka 19 - Ošetřené výjimky komponenty COM_ORGANIZACE. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vyplněn název organizace

7.10 Komponenta COM_ABI

Ošetření výjimek v komponentě COM_ABI je zajištěno pomocí JavaScriptových dialogových oken (viz Tabulka 20).

Tabulka 20 - Ošetřené výjimky komponenty COM_ABI. Zdroj vlastní

	Ošetřené výjimky
Back-end	• musí být vyplněn ročník Abilympiády • musí být vyplněn rok konání Abilympiády • musí být vyplněn první den soutěží • musí být vyplněn druhý den soutěží • musí být vyplněn min. 1 den pro stravování • musí být vyplněn min. 1 den pro ubytování

8 Uvedení do provozu a testování

Po vytvoření informačního systém je nutné jej umístit na odpovídající webhosting a započít s jeho intenzivním testováním ze strany CAA.

8.1 Webhosting

Na základě požadavků a technických parametrů IS byli specifikovány následující skutečnosti, které musí webhosting splňovat:

- technologie PHP 5 a vyšší,
- databázový stroj MySql,
- nepoužívat Safe Mode,
- kapacita 100 MB a více,
- relativně vysoká rychlost odezvy.

Webhosting zajistila CAA pro doménu **http://db.abilympics.cz**. Administrátorské rozhraní má tedy adresu **http://db.abilympics.cz/administrator/**. Webhosting se jeví jako odpovídající jak ve smyslu technologické základny, tak i rychlosti.

8.2 Testování

Po úspěšném založení webhostignu mohl být IS nahrán a připraven k testování. Testování se zaměřilo převážně na:

- funkční komplexnost,
- funkční správnost,
- validitu zdrojového kódu,
- sémantickou správnost textu,
- použitelnost celého systému.

Testovací provoz byl zahájen začátkem ledna 2009 a plynule přešel do „ostrého“ provozu koncem února 2009. Testování se ujala CAA a v jeho průběhu se nevyskytl žádný závažný problém, který by zásadním způsobem degradoval celkovou použitelnost systému. Byly nalezeny pouze drobné chyby s validací javascriptových kódů v různých prohlížečích. V průběhu dokončování této práce již byly úspěšně podány všechny přihlášky soutěžících a jejich následná správa ze strany CAA se jeví jako bezproblémová.

Závěr

Tvorba informačních systémů je velice komplexní činnost a v současné době je nedílnou součástí informační společnosti. Využití webového IS je velkou výhodou jak už ve formě dostupnosti, tak v multiplatformosti jeho využívání.

Tato práce byla zaměřena na konkrétní realizaci informačního systému pro potřeby České abilympijské asociace. Hlavním cílem bylo vytvořit funkční informační systém, který poskytne všem uživatelům ulehčení práce. Tj. klientům poskytne možnost podat elektronickou přihlášku namísto papírové a zaměstnancům asociace zásadním způsobem usnadní správu všech těchto přihlášek. Systém je v době dokončování této práce v bezproblémovém provozu a nejeví žádné známky nedostatků. Jelikož byl úspěšně navržen a implementován informační systém, který zlepšil správu dat spojenou s pořádáním Abilympiád, je možné konstatovat, že cíl této práce byl splněn.

Použitá literatura

[1] ŠIMONOVÁ, Stanislava, MYŠKOVÁ, Renata, JIRAVA, Pavel. *Projektování informačních systémů - UML, procesní řízení.* [s.l.] : [s.n.], 2006. 114 s.

[2] ŘEPA, Václav. *Metodika vývoje informačního systému s pomocí nástroje Power Designer* [online]. 2009 [cit. 2009-04-02]. Dostupný z WWW: <http://www.sybase.cz/buxus/docs/Metodika_vyvoje_IS_06_2006.pdf>.

[3] PELÁNEK, Radek. *Modelování: Obecné principy* [online]. 2009 [cit. 2009-04-02]. Dostupný z WWW: <http://www.fi.muni.cz/~xpelanek/IV109/slidy/modelovani.pdf>.

[4] ŠMÍD, Vladimír. *Životní cyklus informačního systému* [online]. 2009 [cit. 2009-04-02]. Dostupný z WWW: <http://www.fi.muni.cz/~smid/mis-zivcyk.htm>.

[5] *Česká abilympijská asociace* [online]. 2008 [cit. 2008-06-22]. Dostupný z WWW: <http://www.abilympics.cz/>.

[6] ŠPÍRKOVÁ, Petra. *Moderní metody řízení projektů*. [s.l.], 2008. 76 s. Bakalářská práce.

[7] Interní zdroje České abilympijské asociace

[8] SYBASE SOFTWARE, S.R.O.. *PowerDesigner* [online]. 2009 [cit. 2009-04-02]. Dostupný z WWW: <http://www.sybase.cz/index.php?option=com_content&view=article&id=3&mid=24>.

[9] ŠARMANOVÁ, Jana. *Teorie zpracování dat* [online]. 2003 [cit. 2009-04-04]. Dostupný z WWW: <http://www.miroslavkrupa.cz/download/TZD_dist_2.pdf>.

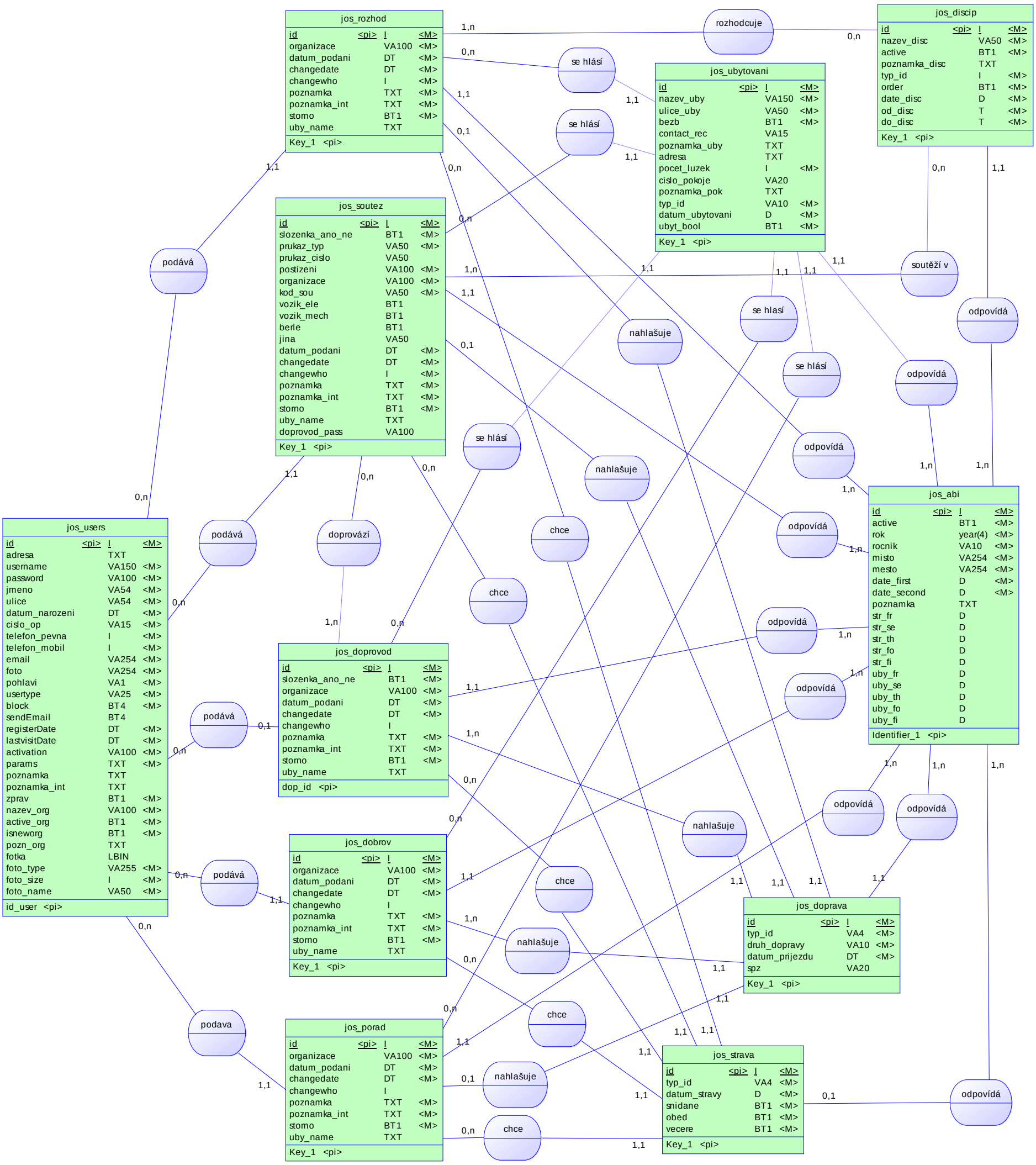
[10] *What is Joomla?* [online]. 2009 [cit. 2009-04-11]. Dostupný z WWW: <http://www.joomla.org/about-joomla.html>.

[11] *PHP Generic Access Control Lists* [online]. 2008 [cit. 2008-12-10]. Dostupný z WWW: <http://phpgacl.sourceforge.net/>.

Seznam zkratek

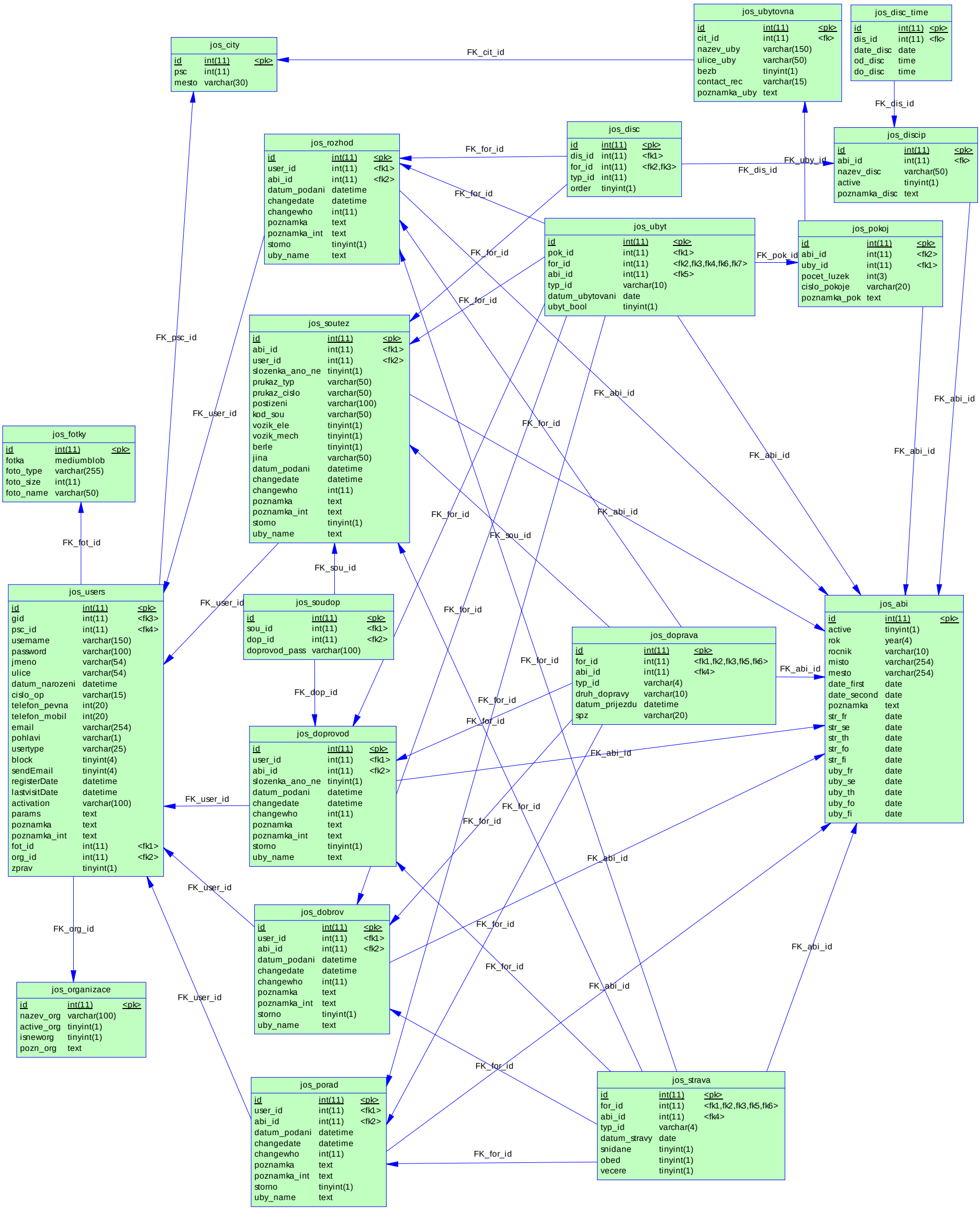
ACL	seznam pro řízení přístupu (Access Control List)
ACO	objetky kontrolující přístup (Access Control Objects)
API	aplikační programové rozhraní (aplication program interface)
ARO	objekty požadující přístup (Access Request Objects)
AXO	objekty rozšířeného přístupu (Access eXtension Objects)
CAA	Česká abilympijská asociace (Czech Abilympic Association)
CMS	systém pro správu obsahu (content manage system)
HW	hardware
IS	informační systém (information system)
MVC	model-pohled-řadič (model-view-controller)
PHP	hypertextový preprocesor (Hypertext Preprocessor)
SQL	dotazovací strukturovaný jazyk (Structured Query Language)
SW	software

Příloha č. 1 – ERD



Zdroj vlastní

Příloha č. 2 – Fyzický datový model



Zdroj vlastní