VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

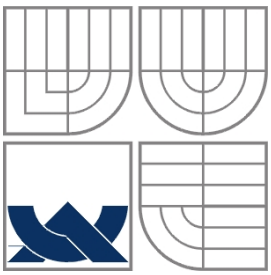
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EVIDENČNÍ SYSTÉM CESA VUT V BRNĚ

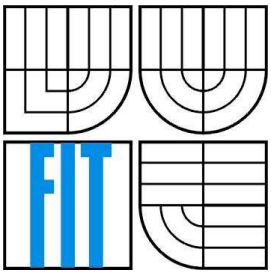
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PETR KADLEC
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EVIDENČNÍ SYSTÉM CESA VUT V BRNĚ

ATTENDANCE RECORDING SYSTEM CESA VUT BRNO

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — PETR KADLEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAEL KUNC
SUPERVISOR

BRNO 2007

Abstrakt

Cílem této práce je navrhnout a implementovat evidenční systém. Výsledným produktem je webová aplikace zaznamenávající vstupy pomocí čtečky čárových kódů. Při implementaci byl kladen důraz na rozsáhlé používání Ajaxu. Implementace systému proběhla s použitím jazyků PHP, MySQL, XHTML, CSS a Javascript.

Klíčová slova

PHP, AJAX, Mootools, MySQL, čárový kód, evidenční systém, čtečka čárových kódů

Abstract

The purpose of this bachelor thesis was to design and implement attendance recording system. The final product is web application recording entrance using barcode reader. Special attention was given on huge utilization of Ajax. The system was implemented in languages PHP, MySQL, XHTML, CSS a Javascript.

Keywords

PHP, AJAX, Mootools, MySQL, barcode, attendance recording system, barcode reader

Citace

Příjmení Jméno: Název práce v jazyce práce. Brno, rok, bakalářská práce, FIT VUT v Brně.

Evidenční systém CESA VUT v Brně

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michaela Kunce. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

…… ……
Petr Kadlec
9. května 2008

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Michaelu Kuncovi a Ing. Petru Dubovi za poskytnutí cenných rad a ochoty při řešení této práce. Dále bych rád poděkoval Ing. Františku Kreslíkovi za půjčení čtečky čárových kódů.

Obsah

Úvod

Podnětem pro vznik této bakalářské práce se stala potřeba evidovat a monitorovat využívání sportovních zařízení CESA VUT v Brně. Stát přispívá škole na návštěvu až 2 sportovních hodin do týdne každému studentovi. O tom, kolik ve skutečnosti kdo sportovišť využívá, ale není žádný přehled. Evidenční systém popsaný v této práci představuje řešení tohoto problému.

Práce je rozdělena do šesti kapitol. Technologie pro vývoj informačních systému jsou popsány v *první kapitole*. Nejdříve je popsána obecná struktura systému a poté jsou postupně představeny jednotlivé technologie MySQL, PHP, UML,XHTML, CSS, Javascript a Ajax.

Technologie čárových kódů a technologie s ní spjaté jsou představeny v *kapitole druhé*. Kapitola vysvětluje funkci čtečky čárových kódů, protokol HID a princip činnosti snímače CCD. V závěru je uveden způsob, jakým se čtečka konfiguruje.

Ve *třetí kapitole* jsou analyzovány požadavky na evidenční systém.

Kapitola čtvrtá představuje na základně požadavků návrh systému. Dynamická struktura systému je namodelována pomocí diagramu use-case, statická pomocí diagramu ER. Rovněž jsou zde stanoveny podmínky pro funkčnost evidenčního systému.

Vlastní implementaci popisuje *kapitola pátá*. Podrobně je zde rozvedeno zobrazování docházkových seznamů a vkládání nových záznamů. Stručněji je zde zmíněna funkce našeptávače, tvorba grafů a řada dalších.

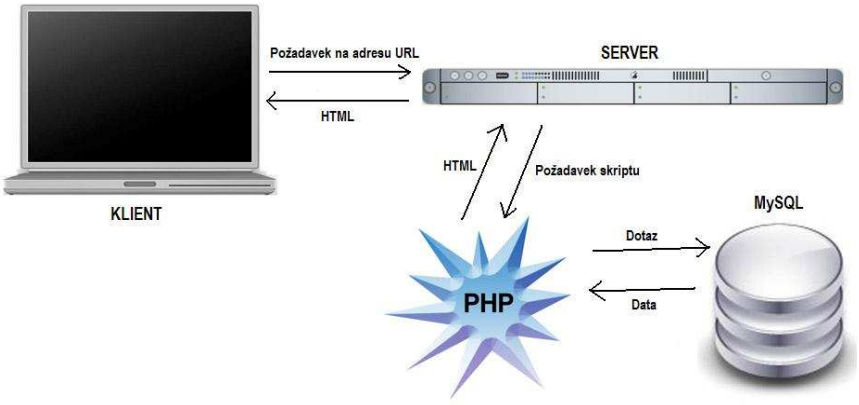
Závěrečná *šestá kapitola* shrnuje výsledky, upozorňuje na úskalí při používání systému a navrhuje další možnosti rozšíření.

1 Technologie pro vývoj informačních systémů

Při tvorbě informačního systému vyvstane jako jedna z prvních otázek volba programovacího jazyka. Z trojice nejpoužívanějších technologií PHP, ASP/.NET, Java2EE padla volba právě na první zmíněnou. Mezi hlavní důvody této volby patří :

- Integrace se stávajícím systémem psaným taktéž v prostředí PHP+MYSQL
- Jednoduchost, obrovská flexibilita jazyka, rychlost vývoje
- Množství volně dostupných knihoven a hotových řešení
- Rozsáhlá komunita vývojářů

Obrázek 1.1 představuje typickou strukturu dynamické webové aplikace. V následujících podkapitolách postupně představím všechny použité technologie s jejich využitím v informačním systému.



Obrázek 1.1: Architektura informačního systému

1.1 UML

UML (Unified Modeling Language) je jazyk s bohatou sémantikou a syntaxí, který usnadňuje návrh a vizualizaci různých typů aplikací. Model v UML se skládá z různých diagramů, jež představují pohledy na různé části sémantického základu navrhované aplikace. Sémantickým základem je souhrn specifikací aplikace, který vymezuje oblast, v němž se může návrh pohybovat. Diagram ve vizuální formě zobrazuje právě jeden konkrétní obraz o aplikaci. Žádný dvourozměrný diagram nemůže zachytit komplexní aplikaci v celku, ale soustředí se vždy právě na jeden důležitý aspekt. Mezi nejčastěji používané diagramy patří Use-case diagram a ER diagram.

1.1.1 Use-case diagram

Diagram Use-case (diagram případu užití) zachycuje hlavní funkční požadavky na systém. Případy užití neboli *Use Case* jsou psány z pohledu zákazníka a podávají první představu o rozsahu projektu. Při formulaci diagramu řešíme, které procesy má systém podporovat a jací uživatelé budou systém využívat. Diagram zde zobrazuje dynamické chování systému.

1.1.2 ER diagram

ER (Entity-relationship) diagram představuje model reprezentující strukturu dat. Při návrhu systému se využívá tohoto diagramu k popisu informací, jež mají být uloženy v databázi.

Model obsahuje 2 základní elementy – entity a vztahy. Entita reprezentuje diskrétní objekt (za entity lze považovat podstatná jména) zatímco vztah (relationship) zachycuje, jak 2 nebo více entit spolu souvisí navzájem(vztahy představují především slovesa).

1.1.3 EER diagram

EER diagram (Extended entity-relationship) obsahuje veškeré modelovací koncepty ER diagramu a přidává nové:

- Podtřídy/supertřídy
- Specializace/generalizace
- Dědění atributů a vztahů

Díky těmto dodatečným EER konceptům lze modelovat aplikace přesněji a kompletněji. EER diagram také zahrnuje i objektově orientované koncepty, jako třeba dědictví. Převzato z [10].

1.2 PHP

PHP je v současné době asi nejrozšířenější technologie umožňující snadné programování na straně serveru (server-side programming). Výhoda takovéhoto přístupu je možnost tvorby různých interaktivních webových stránek. Skript napsaný v PHP je nejprve zpracován na serveru a výsledek je poté odeslán volajícímu počítači jako běžná statická HTML (XHTML) stránka. Poté co je již stránka načtena klientem, není možné ji již dále pomocí PHP měnit.

Mezi základní vlastnosti jazyka patří, že je interpretovaný, procedurální, typ proměnné se určí v okamžiku přiřazení hodnoty (PHP je dynamicky typový), pole jsou heterogenní, jazyk je case-sensitive a v současnosti už podporuje i objektový model.

PHP je nástupcem staršího produktu, nazvaného PHP/FI. PHP/FI vytvořil Rasmus Lerdorf v roce 1995 jako jednoduchou sadu skriptů v jazyce Perl pro zpracování záznamů o přístupech k jeho webu. Protože byla potřeba větší funkčnost, napsal Rasmus mnohem rozsáhlejší implementaci v C, která byla schopna komunikovat s databázemi a umožňovala uživatelům vyvíjet jednoduché

dynamické aplikace pro web. K značnému rozšíření PHP přispěla až druhá verze PHP/FI 2.0 vydaná v roce 1997.

PHP 3.0 byla první verze, která se velmi blížila takovému PHP, jak ho známe dnes. Vytvořili ho Andi Gutmans a Zeev Suraski v roce 1997. Jednou z nejsilnějších zbraní PHP 3.0 byly jeho obrovské možnosti rozšíření. Kromě pevné infrastruktury pro mnoho různých databází, protokolů a API koncovým uživatelům, přilákaly především možnosti rozšíření PHP 3.0 také tucty vývojářů, kteří se připojili a vytvořili nové rozšiřující moduly. Nový jazyk byl uvolněn pod novým názvem. Zůstalo jen PHP, což je rekurzívní akronym - PHP: Hypertext Preprocessor.

Roku 2000 vychází PHP 4.0 s novým jádrem nazvaným Zend Engine. Nové PHP zvyšuje především výkon a přidává další klíčové prvky, jako je podpora pro mnoho různých WWW serverů, HTTP sessions, buffering výstupu, bezpečnější způsoby zpracování vstupů uživatele a mnoho nových jazykových konstruktů.

Roku 2004 byla uvolněna nejnovější verze PHP 5.0 s přepracovaným Zend Engine 2. V nové verzi jazyka byly především posíleny bezpečnostní mechanismy a uveden nový, podstatně kvalitnější objektový model, umožňující používat PHP jako skutečný objektově orientovaný jazyk.

V současné době jsou podporovány již jen 2 poslední verze PHP – PHP 4.4.7 a PHP 5.2.2. Částečně převzato z [1] a [5].

1.3 MySQL

Pro potřebu trvalého uchovávání dat na serveru je vhodným řešením využití databázového serveru. Jedním z nejrozšířenějších je databázový systém MySQL vytvořený a udržovaný švédskou firmou AB. Jedná se systém správy databází (DBMS, databaze management system) určený pro relační databáze. Relační databáze je kolekcí vzájemně provázaných dat uložených v podobě textu, čísel nebo binárních souborů řízenou právě systémem DBMS. MySQL je multiplatformní databáze. Komunikace s ní probíhá pomocí jazyka SQL.

Relační databáze používá k ukládání informací více tabulek, čímž je informace dělena na nejmenší součásti. Do sedmdesátých let připomínaly databáze spíše listy tabulkového kalkulátoru s jedinou ohromnou tabulkou obsahující všechny informace. Přestože relační databáze vyžadují při návrhu a implementaci mnohem více úsilí oproti běžným databázím, nabízejí větší míru spolehlivosti a integrity dat. Kromě jiného také podporují rozšířené možnosti vyhledávání a také možnost sdíleného přístupu.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení. Od verze 5.0 však již podporuje i uložené procedury, triggery a pohledy.

Maximální velikost databáze je daná maximální velikostí souborů podporovaná operačním systémem, nikoliv MySQL vnitřním limitem. Operační systém Windows 32bit se systémem souborů NTFS má limit pro maximální velikost databáze okolo 2TB. Informace jsem čerpal z [2].

1.4 HTML, XHTML, CSS

HTML (HyperText Markup Language) je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML. Jazyk HTML se z něj vyčlenil specifikací konkrétních značek.

Od první verze HTML 0.9 vydané v roce 1991, která nepodporovala ani grafický režim, prošel jazyk HTML značným vývojem. Poslední funkční verze je 4.01 z roku 1999. Podle původního předpokladu se mělo jednat o poslední verzi, po které by se přešlo na XHTML. Roku 2007 však byla založena nová pracovní skupina HTML, jejímž cílem je vývoj nové verze HTML 5.0 a její uvolnění do roku 2010. Vývoj HTML a XHTML nyní probíhá odděleně.

XHTML (***ex**tensible **hyper**t**ext** **m**arkup **l**anguage*) je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý W3C. K základním rozdílům oproti HTML patří:

- Všechny tagy musí být ukončené a to včetně nepárových
- Všechny tagy a jejich atributy musí být zapsány malými písmeny
- Všechny hodnoty atributů musí být uzavřeny do uvozovek
- Dokument musí začínat XML deklarací je-li dokument kódován jinak než v UTF-8

Myšlenka oddělení vzhledu dokumentu od jeho struktury a obsahu dala vzniku CSS (*Cascading Style Sheets*). Původně to měl umožnit už jazyk HTML. Jazyk byl navržen standardizační organizací W3C, autorem prvotního návrhu byl Håkon Wium Lie.

K hlavním výhodám patří rozsáhlé možnosti formátování, konzistentní styl, oddělení struktury a stylu, dynamická práce se styly, formátování XML dokumentů, větší kompatibilita alternativních webových prohlížečů, kratší doba načítání stránky, změna v souboru .css se projeví na celém webu, podpora různých stylů pro různá výstupní zařízení. Jedinou větší nevýhodou je ne vždy dostatečná podpora v majoritních prohlížečích a tím daná složitost optimalizace pro všechny webové prohlížeče. Více v [3] a [9].

1.5 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich z tehdejší společnosti Netscape. Syntaxe vychází z rodiny jazyků C/C++/Java. Ranná verze JavaScriptu se jmenovala LiveScript. Microsoft jako konkurenční společnost zareagovala svou vlastní verzí nazvanou JScript a zapříčinila tím boj prohlížečů Netscape Navigator vs. Internet Explorer, který trvá dodnes.

Standardizací prochází až v roce 1997, kdy se obě společnosti obrátily na třetí stranu – organizaci ECMA(European Computer Manufactures Association). Výsledkem je, že se podpora javascriptu mezi prohlížeči sjednocuje, i když určité rozdíly stále existují. Více na [3]. Mezi jeho základní rysy patří:

- Interpretován na straně klienta (možná ale i interpretace na straně serveru - LiveWare či Rhinola)
- Case-sensitive
- Nelze se na něj spoléhat – asi 5% klientů má vypnutý JavaScript
- Objektově orientovaný
- Přístup k prvkům HTML pomocí modelu DOM (Document Object Model)

1.5.1 AJAX

Autorem první zmínky o AJAXu je Jesse James Garrett, který tuto technologie poprvé uveřejnil. AJAX je zkratka pro Asynchronous JavaScript and XML. Slovo asynchronous v názvu značí, že prohlížeč nečeká na odpověď od serveru, nýbrž je schopen odpověď zpracovat už v okamžiku, kdy k ní skutečně dojde. Přenosy dat se ve skutečnosti odehrávají na pozadí, aniž by musel prohlížeč pozastavit prováděné operace a na něco čekat.

JavaScript hraje v technologii AJAXu klíčovou roli, protože právě JavaScript řídí úkony odehrávající se v prohlížeči. Javascript vytváří spojení se serverem při odesílání i příjmu požadavků od serveru.

Posledním součástí AJAXu je jazyk XML, jež představuje jeden ze způsobů, jak zpracovat odpověď od serveru. Server může také odpovědět čistým textem. Klíčovou částí JavaScriptu, jež umožňuje využití AJAXu, je objekt XMLHttpRequest, díky němuž je možné se připojit k serveru a zpracovat odpověď na pozadí.

Výhoda použití AJAXu spočívá v odstranění nutnosti načtení celého dokumentu a tím spojené nižší zátěže webových serverů. Nevýhodou může být absence uživatelsky oblíbených tlačítek zpět a dopředu. Mezi nejpopulárnější funkce na webu využívající Ajax je automatické dokončování, interaktivní mapy, táhni a pusť a vyskakovaní okna. Více na [4].

2 Čtečka čárového kódu

Čtečka čárového kódu nebo-li také scanner je hardwarové vstupní zařízení umožňující převedení fyzické 2D nebo 3D předlohy do digitální podoby pro další využití, většinou pomocí počítače. Dělí se při tom na 1D a 2D podle typu čárového kódu či podle toho jestli využívají laserový paprsek nebo diodu. Pro samotné snímání obrazové informace je využívána elektronická součástka CCD. Většina moderních ručních čteček podporuje několik komunikačních rozhraní od RS232, KBW až po USB implementující protokol HID, kde se pak čtečka chová jako další klávesnice.

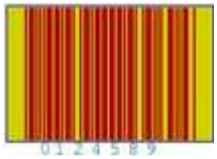
2.1 Čárové kódy

Čárový kód je prostředek pro automatizovaný sběr dat. Je tvořen černo-tiskem vytištěnými pruhy (v některých novějších verzích kódu mozaikou) definované šířky, umožňující přečtení pomocí technických prostředků - čteček či skenerů. Patent na čárový kód byl poprvé udělen v roce 1949. Podle způsobu, jakým se konkrétní znak kóduje do skupiny pruhů, se kódy dělí do skupin. Nejpoužívanější skupiny kódů jsou:

- **Code 2/5** 

- **EAN** 

- **Codabar** 

- **Code 11** 
- **UPC** 

- **Code 128** 

- **Code 3/9** 

- **Code 93** 

Více na [6].

2.2 Protokol HID

HID (Human interface device) je typ počítačového zařízení kterými člověk zadává počítači příkazy. Termín HID se nejčastěji odkazuje na USB-HID specifikaci. Primárním cílem pro HID bylo umožnění inovace PC vstupních zařízení spojené se zjednodušením jejich instalace. Před vznikem

HID každé vstupní zařízení používalo svůj vlastní komunikační protokol. Jakákoliv změna v hardwaru vyžadovala změnu existujícího protokolu nebo vytvoření nového. Oproti tomu všechna HID zařízení využívají samo-popisný balík ovladačů obsahující nekonečné množství různých typů dat a formátů. Typickými hardwarovými zástupci využívající protokol HID jsou klávesnice, touchpad, myš, gamepad ale také Wii ovladač a další. Více na [8].

2.3 CCD snímač – princip činnosti

CCD (Charge-Coupled Device- zařízení s vázanými náboji) využívá podobně jako všechny ostatní světlocitlivé součástky fyzikálního jevu známého jako fotoefekt. Tento jev spočívá v tom, že částice světla foton při nárazu do atomu dokáže přemístit některý z jeho elektronů ze základního do tzv. excitovaného stavu a odevzdá mu přitom energii

$$E=\nu.h$$ (kde **ν** je kmitočet fotonu a **h** je tzv. Planckova konstanta.)

V polovodiči se takto uvolněný elektron může podílet na elektrické vodivosti respektive je možno ho z polovodiče odvést pomocí přiložených elektrod, tak jak se to děje například u běžné fotodiody. Ta po dopadu světla vyrábí elektrický proud.

U CCD je ovšem elektroda od polovodiče izolována tenoučkou vrstvičkou oxidu křemičitého $\mathbf{SiO_2}$, který se chová jako dokonalý izolant, takže fotoefektem uvolněné elektrony nemohou být odvedeny pryč. Více na [7].

2.4 Konfigurace čtečky

Změna nastavení čtečky probíhá přes konfigurační čárové kódy. Ty se dají v příslušném programu nachystat a vytisknout a postupným přikládáním čtečky k jednotlivým kódům čtečku konfigurujeme. Čtečka po každém úspěšném přečtení čárového kódu odpoví patřičným zvukovým signálem.

2.4.1 Nastavení čtečky

Abychom zabránili vepsání čárového kódu do libovolné položky formuláře na stránce zvolili jsme pro potřebu evidenčního systémů klávesu F12 jako prefix každého přečteného kódu a jako sufix klávesu Enter. Postup nastavení prefixu na klávesu F12 by byl následující:

1)  Start konfiguračního módu (3* pípnutí)

2)  Nastavení prvního prefixu (1* pípnutí)

3)  kód 1. bytu (1* pípnutí)

4)  kód 2. bytu (2* pípnutí)

5)  kód 3. bytu (3* pípnutí)

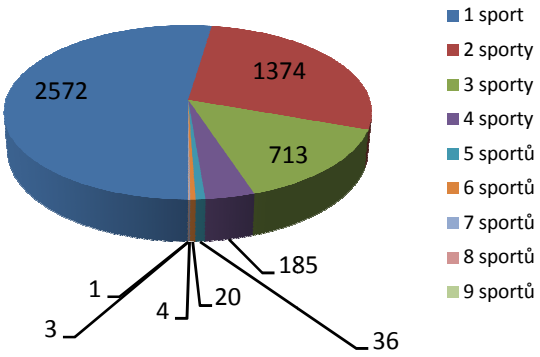
6)  Konec konfiguračního módu (3* pípnutí)

3 Analýza požadavku na evidenční systém

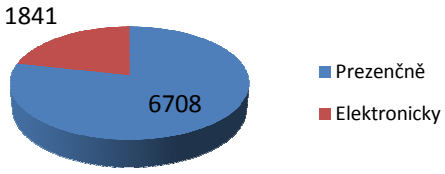
3.1 Současný stav

V letním semestru roku 2008 se zapsalo 4908 sportovců do 8553 hodin. Stát přispívá škole na návštěvu až 2 sportovních hodin do týdne každému studentovi. Z důvodů nenaplněné kapacity je však dovoleno studentům dovoleno zaregistrovat si 3 hodiny týdně. 3 hodiny týdně je i maximální počet hodin, které dovolí zapsat elektronická registrace v prvním týdnu semestru. Po uplynutí doby pro elektronickou registraci je však ještě možné se připsat do nějaké hodiny „prezenčně“ – na místě. O zapsaní zájemce do hodiny potom rozhoduje instruktor vyučující příslušnou hodinu. S tímto prezenčním do-registrování by však maximální počet hodin zapsaných jedním studentem neměl přesahovat 3. Realita však ukazuje, že přibližně 5% studentů tuto hranici překračuje. Více graf 3.1. Z grafu je také patrné, že stát hradí přibližně pouze 80% hodin.

Druhým problémem je již dříve zmíněné „prezenční“ do-registrování. Každý student, jež je zapsán prezenčně, musí být také dopsán do systému. A s tím je samozřejmě spjata další administrativní činnost. Co je při elektronické registraci uděláno automaticky, musí zde být uděláno ručně. Více graf 3.2.



Graf 3.1: rozdělení počtu hodin/týden



Graf 3.2: rozdělení podle typu registrace

3.2 Úvod do problematiky

Organizace CESA v současné době vyučuje zhruba padesát sportů provozovaných na 44 sportovištích. Aby se dosáhlo integrace se všemi sporty, musely by být všechny tyto sportoviště vybaveny čtečkou, počítačem a přípojkou na internet. V řadě sportovišť tak můžeme narazit

na problém. Některé sportoviště (bowlingové centrum, všechny bazény, golfové centrum a další) CESA přímo nevlastní, ale pouze využívá společně s ostatními subjekty. Těžko lze zde tak zasahovat do místního vybavení. U takových sportů jako nordic walking nebo horská kola nelze splnit podmínku žádnou, protože tyto sporty ve skutečnosti ani žádné sportoviště nevyužívají.

Implementovaný evidenční systém by se tedy soustředil pouze na ty sporty a sportoviště, které výše zmíněné podmínky splňují.

3.3 Požadavky na evidenční systém

Formou konzultací s tvůrcem současného informačního systému a návrháře databáze a pracovníky CESA VUT byly formulovány následující požadavky:

- Minimální zásahy do stávajících tabulek databáze (z důvodu funkčnosti stávajícího IS)
- Intuitivní a jednoduché ovládání
- Zobrazení statistik a grafů
- Možnost vyhledávání osob
- Sjednocení účtů jednotlivých sportů
- Zaznamenání vstupu přes kartu s čárovým kódem s možností rozšíření na čipovou kartu
- Zabezpečení heslem
- Variabilní nastavení

4 Návrh

4.1 Neformální popis systému

Vytvářený evidenční systém je určen pro neziskovou organizaci CESA VUT Brno. Vstupy jsou do systému přidávány pomocí čtecího zařízení (čtečka čárových kódů) po projetí kartičkou s čárovým kódem, kterou by každý student vlastnil.

Základní zobrazovanou jednotku představuje jedna hodina, která se dělí na zapsané do dané hodiny a ti se dále dělí na jednotlivé vstupy. Systém by měl bez nutnosti přihlášení zobrazovat právě probíhající hodinu na daném sportovišti a 2 hodiny před a po. Po přihlášení je možnost:

- vidět a procházet výpisy ze všech hodin, vyhledávat podle vedoucích a sportovišť
- nastavit místo sportoviště
- vyhledávat v osobách majících zapsaný nějaký sport a zobrazovat informace o jejich hodinách a stavu účtu

Při samotném zaznamenání vstupu do hodiny je k aktuálnímu času registrace přičtena určitá doba, aby byl zápis proveden do správné hodiny, protože do hodiny se chodí zpravidla dříve než začne. Systém by měl zobrazovat nějaké statistiky, především o docházce do jednotlivých hodin.

Do systému mají přístup pouze Instruktoři. Administrátor zde má pouze roli zodpovědné osoby za správně nastavený konfigurační soubor. Systém by měl být jednoduše konfigurovatelný pomocí základních systémových proměnných:

- Počet týdnů semestru
- Počet týdnů zkouškového období
- Začátek semestru
- Začátek zkouškového období
- Použití čipových či čárových kódů

4.2 Návrh řešení

Při formalizaci návrhu evidenčního systému bylo použito modelovacích technik vycházejících z jazyka UML.

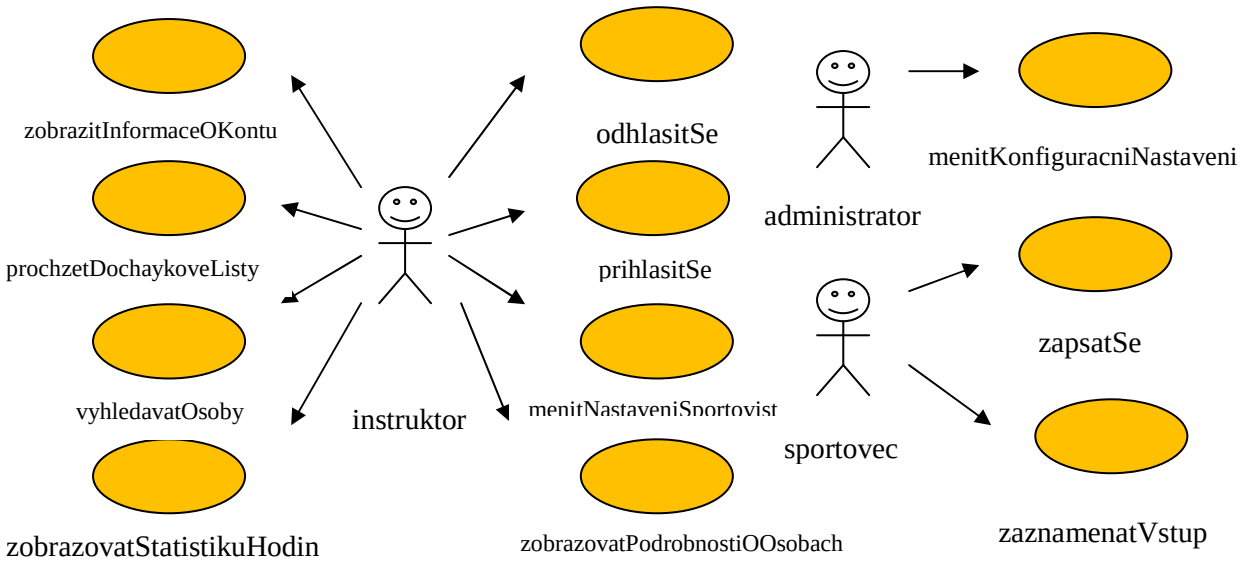
4.2.1 Dynamická struktura

Z diagramu 4.1 vyplívá, že uživatel systému bude zastávat jednu ze tří rolí – instruktor, administrátor nebo sportovec.

Instruktor – v systému má téměř všechny pravomoci (přihlášení, prohlížení, vyhledávání)

Sportovec – jedinou pravomocí sportovce je zapsat se do hodiny a zaznamenat vstup

Administrátor – má přístup ke konfiguračním souborům systému



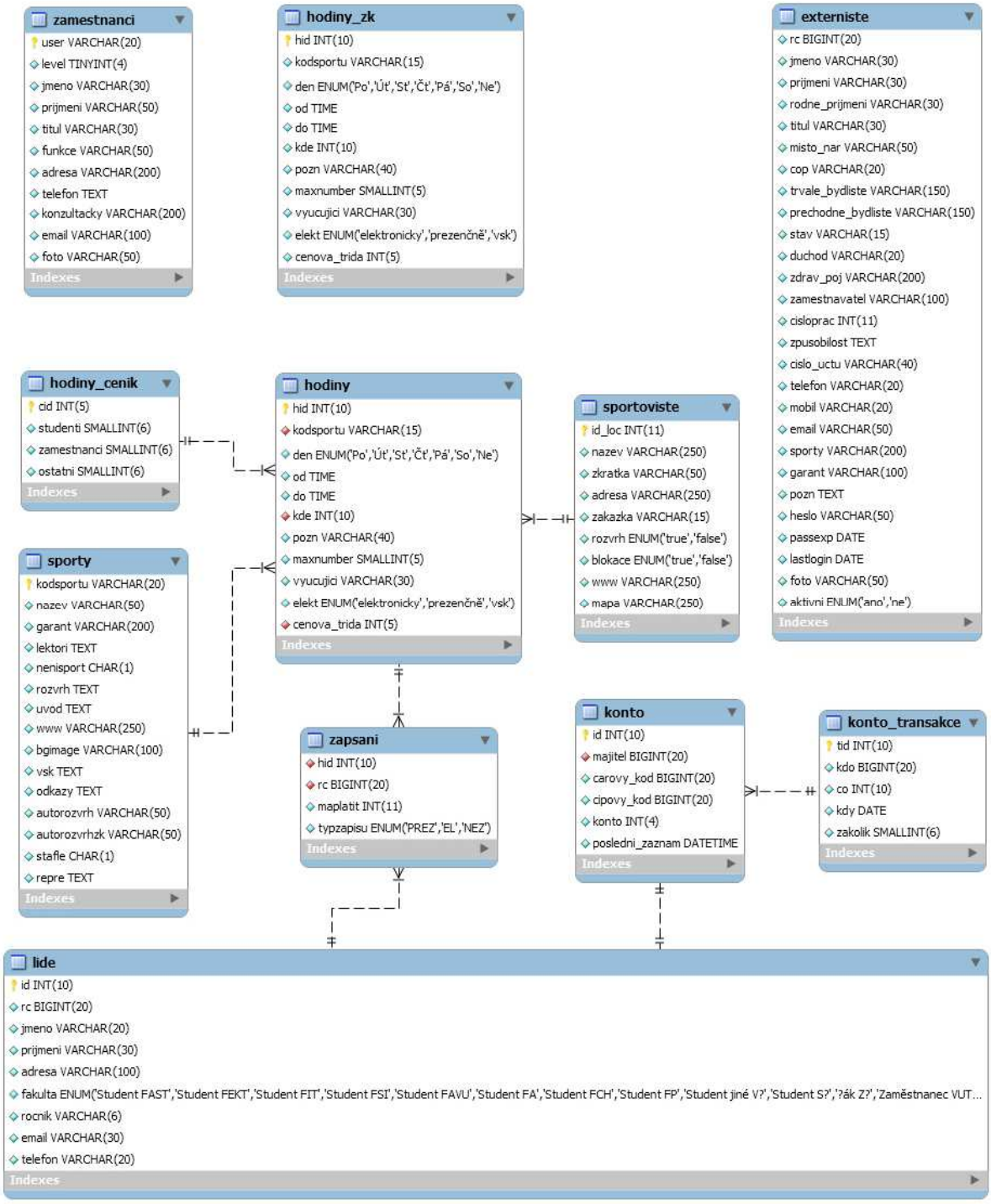
Obrázek 4.1: Use-case diagram evidenčního systému

4.2.2 Statická struktura

Vyvíjený evidenční systém využívá stávající databáze CESA VUT Brno. Pro vlastní potřebu přidává pouze 3 tabulky:

- tabulku pro uchování informace o kontu každého sportovce
- tabulku s jednotlivými vstupy
- tabulku s cenami hodin

Téměř všechny ostatní tabulky jsou ponechány beze změny v originální podobě. Změna nastala pouze v nezbytně nutných případech. Dotkla se pouze 3 tabulek a to tabulek *hodiny_zk* a *hodiny*, kde přibyl shodný atribut *cenova_trida* a tabulky *zapsani*, kde ve výčtovém typu atributu *typzapisu* přibyla třetí možnost NEZ. Celková podoba je zachycena na obrázku 4.1



Obrázek 4.1: EER digram databáze vytvořený pomocí nástroje MySQL Workbench 5.0

4.2.3 Návrh databáze

Ve stávajícím IS figuruje jako cizí klíč v mnoha tabulkách rodné číslo daného člověka. Jelikož tabulka *lide* obsahuje jako primární klíč číslo *id*, dalo by se toto číslo použít namísto rodného čísla. V zájmů zachování kompatibility jsem se rozhodl používat jako cizí klíč taktéž rodné číslo.

Klíčovou tabulkou evidenčního systému je tabulka *konto_transakce*. Z ní se generuje výsledná docházka. Obsahuje atributy *kdo* (rodné číslo účastníka), *co* (evidenční číslo hodiny), *kdy* (aktuální datum) , *zakolik* (cena za hodinu) a autoinkrementovaný primární klíč *id*. Hodnota atributu *zakolik* je zde redundantně uložena, přestože by se dala cena dané hodiny dala dohledat. Cena hodiny je zde uložena záměrně, protože cena jednotlivých hodin se může v průběhu času měnit. Očekávaný počet řádků tabulky pro jeden semestr je 40 000 – 50 000. Při vkládání do tabulky bude jediným výrazným problémem získání čísla *hid* aktuální hodiny. Jízdní řád výukových hodin je naplánován na jeden týden pondělí až neděle, takže k jednoznačnému určení hodiny, bude stačit kde, datum a čas. Kde vybere místo sportoviště, z datumu získáme název dne a čas nám přesně určí kterou hodinu.

Tabulka *lide* obsahuje seznam všech lidi na VUT od doby vzniku systému CESA. Prozatím má 63 990 záznamů. Primárním klíčem je zde inkrementované číslo *id*. Atribut *rc* (rodné číslo) je unikátním klíčem, proto se může používat také jako cizí klíč. Protože je do tabulky *lidé* vkládáno z různých míst, nemohl jsem integrovat tabulku *konto* do tabulky lide, přestože by vztah 1:1 k tomu mohl nabádat. Kromě atributů *ročník* (počet let studia na VŠ při vložení sportovce do databáze, zpravidla u všech 0, nebo NULL), *adresa* (adresa bydliště), *telefon*, *email*, *telefon* obsahuje ještě atribut *fakulta*, jež je klíčový pro stanovení ceny za sportovní hodinu.

V tabulce *konto* jsou uloženy potřebné kódy pro přihlášení pomocí čipových/čárových karet, datum poslední změny na účtu (atribut *posledniI_zaznam*), *majitel* (rodné číslo) , *konto* (stav účtu) a atribut *id.*

Tabulka *hodiny* představuje soubor vypsaných hodin, do kterých se může student zapsat. Tato tabulka je využívána pouze v období semestru bez zkouškového období. Ve zkouškovém období se využívá identické tabulky *hodiny_zk* s jiným rozvrhem hodin. Před začátkem každého semestru jsou v těchto tabulkách prováděny určitě změny. Jednoznačným identifikátorem je číslo *hid* stejně tak jako kombinace atributů *den,od,do,kde*. Jeden řádek této tabulky představuje ve výsledném evidenčním systému celou prezenční listinu pro danou hodinu. Atribut *vyucujici* představuje instruktora přiřazeného k dané hodině. Zpravidla je v každé hodině uveden 1, 2 nebo 3 instruktoři.

Tabulka *zapsani* představuje zápis studenta do zvolené hodiny. V evidenčním systému je to jeden řádek tabulky. Atribut *maplatit* představuje v současném IS částku, kterou má student zaplatit dopředu, jedná-li se o sportovní hodiny hrazené jen z části (plavání, golf, squash,…). Atribut *typzapisu* představuje způsob zapsaní do hodiny. V současném IS je to možnost PREZ-prezenčně,

EL-elektronicky. Pro potřebu evidenčního systému jsem zavedl ještě třetí možnost NEZ-nezapsán. Průměrná velikost tabulky na 1 semestr je 7000-9000 záznamů. Po každém semestru dochází k vyčištění tabulky.

Z tabulky *sportoviste* získává evidenční systém pouze název sportoviště sloužící k pojmenování sportoviště a číslo *id_loc* pro jeho jednoznačnou identifikaci.

Tabulka sport slouží v současném IS jako místo pro ukládání webové podoby titulní strany jednotlivých sportů. Evidenční systém si z této tabulky pouze vybírá atribut *nazev* (celý název sportu).

Z tabulek externiste a zamestnanci sice nejsou vybírány žádné informace a se systémem nejsou nijak spojeny, spojení by zde však správně být mnělo. Díky špatnému původnímu návrhu těchto tabulek dochází na některých místech k duplicitě informací.

4.3 Předpoklady pro zavedení evidenčního systému

- každé sportoviště by mělo své vlastní čtecí zařízení (čipové či čárové), které by bylo zpravidla umístěno po cestě mezi hlavním vchodem do sportoviště a šatnou. Sdílení 2 a více sportovišť jednoho čtecího zařízení není možné z důvodu, že by nešlo určit, kam že se to daný sportovec vlastně zapsal
- každé sportoviště bude vlastnit alespoň 1 počítač. Potřeba alespoň jednoho počítače je dána nutností prohlížet docházkový seznam a také musí být na každém počítači uložena v cookies informace o daném sportovišti
- na daném počítači je nainstalován Internet Explorer 6 nebo 7 nebo Mozzila Firefox 2.0, není vypnutý javascript a cookies
- trvalé připojení sportoviště k internetu

5 Implementace

5.1 Technologie použité při implementaci

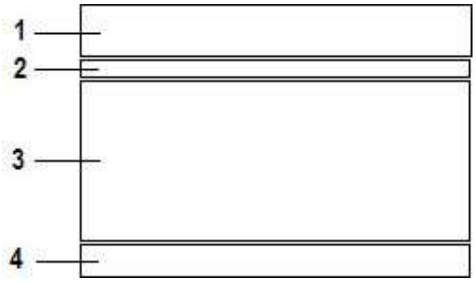
Evidenční systém byl implementován pomocí těchto technologií

- PHP4 + framework JPgrapf
- MySQL 5
- XHTML 1.1
- Javascript + knihovna Mootools + AJAX
- CSS 2

Jelikož server wasp.ro.vutbr.cz podporuje pouze php4, byly v implementaci objektové rysy použity jen minimálně. Klientská část aplikace je optimalizovaná pro Internet Explorer 6,7 a Mozzillu Firefox 2.0. Kódování webových stránek je UTF-8, kódování databáze bylo ponecháno latin2.

5.2 Grafické rozhraní

Přestože se jedná o interní aplikaci, určitý grafický standart by zde však přesto neměl chybět. Pro potřebu aplikace byl vybrán jedno-sloupcový layout s hlavičkou a patičkou, který bude společným všem stránkám.



Obrázek 5.2.1: Návrh layoutu

Obrázek 5.2.2: Návrh designu

V hlavičce(1) bude zobrazována název aktuálního sportoviště a našeptávač. Pod hlavičkou se nachází lišta nastavení(2) s vysunovacím menu sloužící k navigaci a vyhledávání v evidenčním systému. Největší střední oblast(3) bude zobrazovat vybrané informace. Nejspodnější vrstva slouží jako patička(5).

5.3 Nastavení systému

Základní nastavení lze měnit v souboru konfigurace.php. K základním systémovým proměnným patří $zacatek_semestru, $pocet_tydnu_v_semestru a $hodiny pomocí kterých lze jednoduše přepínat mezi zkouškovým obdobím a semestrem. Změnu používání čipových/čárových karet lze změnit parametrem $kody. K vedlejším proměnným patří $posun_zacatku_hodiny, ošetřující předčasný příchod do hodiny, a $odhlaseni, jež stanovuje časový limit pro automatické odhlášení.

5.3.1 Nastavení sportoviště

Jediné nastavení, které se dá měnit přímo v systému, je nastavení sportoviště. Informace o číslu sportoviště je nezbytná při záznamu vstupů a zobrazení docházkového seznamu. Využít databázi pro uložení a načítání této informace není možné, protože každé sportoviště potřebuje mít jinou hodnotu. Jediným řešením je tedy tuto informaci ukládat do klientského počítače v podobě cookies. Do proměnné $_COOKIE['sportoviste'] je tedy uloženo číslo sportoviště s délkou platnosti nastavenou na jeden rok. Platnost cookie je při každém úspěšném přihlášení do systému prodloužena.

5.4 Implementace docházkového seznamu

Nejdůležitějším a nejnáročnějším úkolem při tvorbě evidenčního systému je zobrazení docházkové tabulky. O konečné podobě se rozhodlo již při samotném návrhu. Jednodušší ale výkonově slabší možností by bylo přidat do databázové tabulky *zapsani* dvanáct sloupečků, kde by každý reprezentoval jeden týden. Při projetí kartou by se pak pomyslná čárka zaznamenala do správného chlívečku a do tabulky *konto_transakce* by se přidal záznam kvůli podrobným výpisům účtu. Toto řešení by bylo nesmírně jednouché na implementaci a dotazy na databázi. K získání evidenční tabulky by se pak stačilo dotázat databázového serveru SELECT t1., t2, t3, … t11, t12 … FROM … Velkou nevýhodou takovéhoto přístupu je však značné plýtvání. Databáze by v takovém případě udržovala velké množství prázdných polí. Další nevýhodou by byla komplikovaná změna a nucený zásah do databáze v případě, když by se rozšiřoval počet týdnů v semestru. V opačném případě - snížení počtu týdnů, by zde zůstaly prázdné sloupce.

Implementačně složitějším, ale efektivnějším řešením, pro které jsem se také rozhodl, je generovat záznamovou tabulku pouze ze záznamů ukládaných do tabulky *konto_transakce*. Hlavním problémem při tomto postupu bylo, jak vygenerovat z několika sloupců vícerozměrnou tabulku. Řešení více-rozměrnosti tabulky spočívá v použití SQL příkazu SUM. Vyplnění tabulky hodnotami zajistíme pomocí SQL příkazu CASE. Nejprve pomocí PHP funkcí *spravny_den()* vygenerujeme pole pondělků a nedělí a poté průchodem cyklem vytváříme část databázového dotazu. Při dotazu vybíráme řádky s odpovídajícím rodným číslem, číslem hodiny a datem *between* pondělí neděle. Na konci dotazu ještě nesmíme zapomenout řádky sloučit pomocí klauzule GROUP BY.

```
$sloupce.=",SUM(case when ((KT.kdy between '".$pole_pondeli[$i]."' and '".$pole_nedele[$i]."' )
                        AND KT.kdo=Z.rc AND KT.co=Z.hid)
                then 1 END) as 't".$i."'" ;
```

Ve výsledku toto řešení přináší nesporně více výhod - nulové plýtvání databázovým prostorem, pružná reakce na změny v systému. Ukázka celého dotazu viz příloha.

5.5 Záznam vstupů

Primárním úkolem evidenčního systému je zaznamenávat jednotlivé vstupy. Záznamu vstupu docílíme vytvořením skrytého formuláře umístěného mimo stránku. Tento formulář je vložen do každé stránky. Pomocí javascriptu kontrolujeme událostí zmáčknutí klávesy a čekáme na smluvený signál. Za dohodnutý signál bylo zvoleno zmáčknutí klávesy F12. Díky vestavěné funkci javascriptu *keyCode* tak kontrolujeme každou zmáčknutou klávesu, jestli se nerovná kódu 123 (kód klávesy F12). Při zmáčknutí klávesy F12 se přesměřuje fokus na skrytý textový input a zmáčknutí klávesy ENTER způsobí odeslání formuláře.

Odesílání formuláře opět využívá javascriptu a Ajaxu, aby mohlo samotné zaregistrování probíhat na pozadí a nerušilo znovunačtením celé stránky. Odesláním formuláře se zavolá javascriptová funkce *zaznam()* nebo *zaznam2()*, která odešle serverové části skriptu (zapsat2.php) obsah textového pole a číslo lokace uložené v cookies. Na základě těchto 2 hodnot, aktuální času a aktuálního datumu se ve skriptu uloženém v souboru zapsat2.php rozhoduje o zaznamenání vstupu.

Skript nejdříve funkcí *test_hodiny()* otestuje, jestli v daný datum, čas a místo existuje nějaká hodina, do které by se dalo zapsat. V případě úspěchu poté porovnáme obsah z textového pole s čárovými kódy přiřazených jednotlivým lidem, jestli vůbec někdo s takovým číslem existuje. Jestli existuje, zavolá se funkce *vloz_zaznam()*. V této funkci se ještě provede kontrola na dvojí projetí funkcí *kontrola_dvoj_projeti()* a jestliže funkce nic nevrátí, může se konečně vložit záznam do databáze. Jestliže se jedná o první záznam do dané výukové hodiny a funkce *test_zapsani*() vrátí null vložíme nejprve záznam do tabulky *zapsani*. Tento záznam se pak ve výsledném docházkovém listu zobrazí jako řádek tabulky. Poté přidáme záznam i do tabulky *konto_transakce*, kde se tento záznam zobrazí ve správném okénku jako ona pověstná „čárka“. Nakonec ještě upravíme stav konta majitele. Ukázka netriviálního dotazu na databázi pro zisk informací o studentovy a hodině viz. příloha [2].

V jakékoliv případě nesplnění podmínky se zobrazí jako odpověď ze serveru chybová hláška, informující uživatele o neúspěchu při evidování. Forma odpovědi se liší podle toho, ze které stránky byl formulář odeslán. Ve většině případů se odpověď zobrazí jako popup okénko v pravém horním rohu. Použitím efektu *transition* z knihovny Mootools zajístíme, že popup okénko zůstane viditelné jen na pár vteřin a po chvíli se vytratí.

5.6 Obecné vlastnosti systému

Nedílnou součástí kvalitních informačních systému je možnost vyhledávání, filtrování a řazení zobrazených záznamů. Prvním náznakem filtrování je přihlašovací strana systému. Kromě přihlašovacího okna také zobrazuje právě probíhající hodiny na daném sportovišti v rozmezí 5-ti hodin. Uživateli zde není dána žádna možnost filtrování, protože filtrování hodin je prováděno automaticky podle času a data.

5.6.1 Stránkování

Samozřejmostí je u rozsáhlých záznamů také prezence stránkování. Výhodou je kompaktnější a rychlejší přenos dat ze serveru, rychlejší zobrazení v prohlížeči a také hezčí vzhled. Stránkovač je pro lepší navigaci zobrazen na začátku a konci seznamu. Zobrazeno je vždy 8 odkazů na sousední stránky z obou stran aktuální strany a odkazy na první, poslední, předchozí a následující stranu.

5.6.2 Filtrování hodin

Pro vyhledávání v hodinách je nutné se do systému nejprve přihlásit. Systém vyhledávání byl implementován jako sada rolet, checkboxů a textový polí, kde výběrem jednotlivým filtrů upravujete výsledný dotaz na databázi zobrazující docházkový seznam. Vyhledávat hodiny lze podle místa, dne a instruktora. Je zde také možnost ovlivnit počet zobrazených týdnů (aktuální týden je zaškrtnut automaticky) stejně jako počet záznamů na straně.

5.6.3 Vyhledávání lidí a podrobnosti

Pro možnost vyhledávání lidí a zobrazení podrobností, je nejprve nutné kliknout na kteroukoliv osobu. Kliknutím na někoho vytvoříme nové okno a pomocí GET metody odešleme proměnnou *id*. V případě nastavení pouze proměnné $_GET['id'] se zobrazí podrobnosti o zvolené osobě a zapsaných sportech. Při použití hledání skript nejprve porovná hledaný výraz s databází a pokud vrátí pouze jeden řádek, dojde k přesměřování na tutéž stránku s proměnnou id vybrané osoby. Pokud je vybraných osob více, zobrazí se stránkovaný seznam vybraných osob. Zvolením podrobnosti u vybrané osoby zobrazíme seznam všech vstupů.

5.6.4 Našeptávač

Jedním z nejznámějším a nejužitečnějším využití Ajaxu je našeptávač. Byla to právě funkce Google Suggest, která masové použití Ajaxu zahájila. V evidenčním systému je našeptávání použito při vyhledávání lidí.

První implementovanou verzí v systému byl našeptávač, který Ajax ve skutečnosti ani nevyužíval. Princip spočíval v tom, že se při načtení stránky nahrály všechny možnosti pomocí

javascriptu do pole, a napsaný výraz se tak nemusel porovnávat s jednotlivými údaji v databázi, ale porovnávání probíhalo s jednotlivými položkami pole.

Pozitivními vlastnostmi tohoto postupu byla okamžitá reakce na změnu hledaného výrazu. Je daleko rychlejší vyhledávat v javascriptovém poli uloženém v paměti, než při každé změně výrazu posílat dotaz na server a databázi.

Toto řešení však má dva i velké nedostatky. Při prvním načtení vždy trvalo javascriptu dlouhou dobu vyhledávací pole vytvořit. Prodleva dosahovala při počtu 5 tisíc záznamů i několik vteřin. S počtem záznamů v řádu statisíců by se tak prodleva mohla značně prodloužit. Druhým nedostatkem je, že vyhledávací pole je po celou dobu drženo v pamětí počítače. S enormním množství možností by se tak mohlo množství volné paměti značně zredukovat. Při malém množství napovídaných slov je toto řešení však velmi efektivní. Kvůli velké prodlevě při načítání není tato verze v evidenčním systému využívána.

Druhá implementovaná verze již plně využívá objektu XMLHttpRequest. S každou změnou hledaného výrazu se na pozadí vytvoří funkcí *createObject*() nový objekt třídy XMLHttpRequest, funkcí *autosuggest*() se pošle na server, kde proběhne dotaz na databázi, a výsledek je poslán zpět.

Tato verze umožňuje našeptávat z množiny záznamů omezených pouze možnostmi jazyka php a databáze. Drobnou nevýhodou se může zdát jisté zpožděné napovídání oproti první verzi.

5.6.5 Přihlašování a zabezpečení

Protože není v evidenčním systému po přihlášení možnost ručního vkládání záznamů, čí jakékoliv modifikování obsahu databáze, dílčí informace lze prakticky pouze prohlížet a uživatel tak nemůže způsobit žádnou školu, rozhodl jsem se v tomto směru pro jednoduché zabezpečení společným heslem. Toto bezpečností opatření se však může změnit, podle přání zadavatele.

Samotné přihlašování bylo opět implementováno s využitím Ajaxu a knihovny Mootools. Odeslání formuláře způsobí dotaz na pozadí. Odeslaná formulářová data jsou porovnávána se vzorem na serveru. V případě shody je do proměnné $_SESSION['prihlasen'] nahrána hodnota 11 a uživatel je přesměrován na index.php. V opačném případě server na XMLHttpRequest odpoví chybovou hláškou informující uživatele o neúspěšném přihlášení. Knihovna mootools je zde využívána pro skrytí chybové hlášky po uplynutí určité doby a zobrazení přihlašovacího formuláře.

5.7 Knihovna JPgraph

Knihovna JPgraph je v systému využívána ke tvorbě grafů. Tato knihovna umožňuje vytvářet nejrůznější typy grafů, logaritmické osy, dovoluje měnit vykreslení (osy, popisky,pozadí), vybírat z formátů (JPG,PNG,GIF), generovat klikací mapy, vyhlazovat čáry pomocí anti-aliasingu, a další . V systému jsou grafy využívány na dvou místech:

- Statistika docházky ke každé hodině (sloupcový)
- Pokročilé statistiky (koláčový)

Vytvoření požadovaného grafu je díky tomuto frameworku značně usnadněno. Pro tvorbu sloupcového nejprve pomocí konstruktoru Graph() vytvoříme okno grafu a pomocí BarPlot() vytvoříme samotný graf. Sloučením grafů do jednoho obrázku zajistíme voláním AccBarPlot(). U všech těchto objektů lze modifikovat jejich vlastnosti. Vytvoření grafů zajistí volání Stroke(). Princip tvorby koláčového grafu je podobný tvorbě sloupcového. Rozdíl je jen v názvech konstruktorů jednotlivých objektů a možnostech modifikace jejich vlastností.

6 Závěr

Přestože nebyl systém testován v reálných podmínkách, věřím, že by byl schopen úspěšně plnit svoji funkci. Při opravdovém nasazení by však bylo potřeba počítat s některými riziky. Na lokálním počítači musí být zapnutý internetový prohlížeč a v něm tato webová aplikace. K zapsání vstupu dojde jenom tehdy, je-li aktivní kterékoliv okno evidenčního systému. Prakticky to znamená, že se na daném počítači nemůže dělat nic jiného. Nutností je také zapnutá anglická klávesnice. Pokud bude zapnutá česká klávesnice, bude čtečka namísto čísel zobrazovat písmena. Při nesplnění kterékoliv z těchto podmínek, nebude evidence fungovat.

Možných rozšíření a úprav systému se nabízí hned několik. Ajaxu je v evidenčním systému využíváno v hojné míře, míst pro jeho využití by se ale našlo daleko více. Mohlo by se jej využít ve všech seznamech pro procházení a řazení (náznak byl implementován v seznamu statistiky lidí s nejvíce zaregistrovanými hodinami), na defaultní straně by se mohl při každém vstupu načíst aktualizovaný docházkový list.

Jednotlivé čtečky by se daly naprogramovat, aby každá přidávala před přečtené číslo ještě číslo sportoviště, kde je umístněna. Díky tomu by pak každá čtečka nevyžadovala svůj vlastní počítač. Opodstatněné by bylo použití tohoto řešení v místech, kde spolu jednotlivé sportoviště přímo sousedí.

Největším přínosem by však bylo použití čipových čteček a čipových karet namísto těch čárových. V dnešní době je podle mého soudu technologie čárových kódů již překonaná. Použitím čipových čteček by se dalo využít stávajících čipových karet studentů a zaměstnanců VUT. Odpadla by tak nutnost tisknout karty s čárovým kódem a jejich distribuce. Nezbytné by však v takovém případě bylo poskytnutí čísel jednotlivých karet ze strany VUT. Dalším logickým krokem se nabízí sjednocení VUT účtu, ze kterého čerpají peníze menzy a účtu používaný evidenčním systémem. Takovýto přístup by však vyžadoval podstatné zásahy do implementace a pravděpodobně i změnu programovacího jazyka na ASP/.NET.

Literatura

[1] BAKKEN, S. *Manuál PHP*. 2007. [online]. [cit. 2008-05-01]. URL: <http://www.php.net/manual/cs/>.

[2] ULLMAN, L. *PHP a MySQL: Názorný průvodce tvorbou dynamických WWW stránek.* Computer Press. 1. vydání. 2004. ISBN 80–251–0063–4.

[3] POWELL, T. A. *Web Design:Kompletní průvodce.* Computer Press. 1. vydání. 2004. ISBN: 80-722-6949-6.

[4] HOLZER, S. *Mistrovství v AJAXu*. 1. vydání. 2007. ISBN 978-80-251-1850-4.

[5] GUTMANS, A. - BAKKEN S. S. - RETHANS D. *Mistrovství v PHP 5*. Computer Press. 1. vydání 2007. ISBN: 978-80-251-1519-0.

[6] Wikipedie. Otevřená encyklopedie. [online]. [cit. 2008-05-04]. URL: <http://cs.wikipedia.org/wiki/%C4%8C%C3%A1rov%C3%BD_k%C3%B3d>.

[7] Wikipedie. Otevřená encyklopedie. [online]. [cit. 2008-05-04]. URL: <http://cs.wikipedia.org/wiki/CCD>.

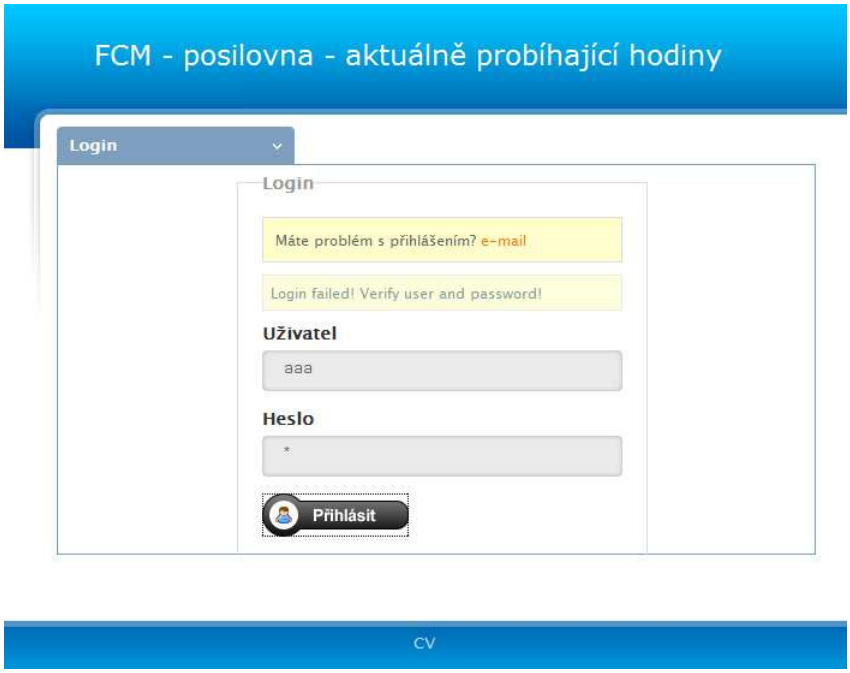
[8] Wikipedie. Otevřená encyklopedie. [online]. [cit. 2008-05-04]. URL: <http://en.wikipedia.org/wiki/Human_interface_device>.

[9] CYROŇ, M. *CSS – kaskádové styly*. Praha: Grada, 1. vydání. 2006. ISBN 80-247-1420-5.

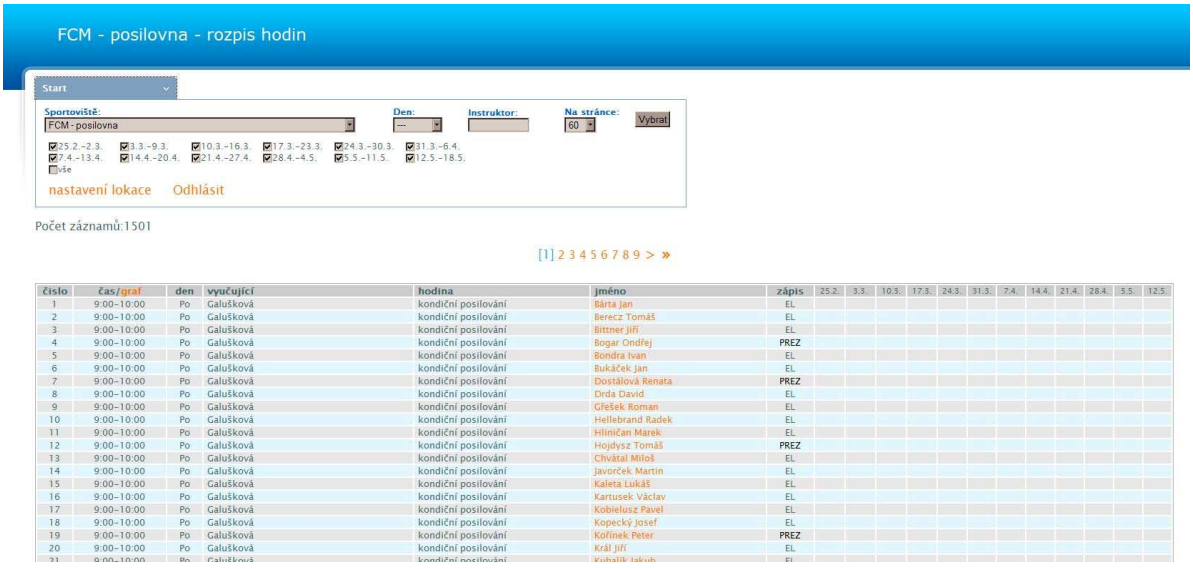
[10] RAMEZ, E. – SHAMKANT, B. N. *Chapter 4: Enhanced Entity-Relationship (ERR) modeling.* [online]. [cit. 2008-5-14].
URL: <http://www.cs.clemson.edu/~artpell/classes/462_662/classnotes/Ch04.ppt>.

Seznam příloh

1 Ukázky uživatelského rozhraní

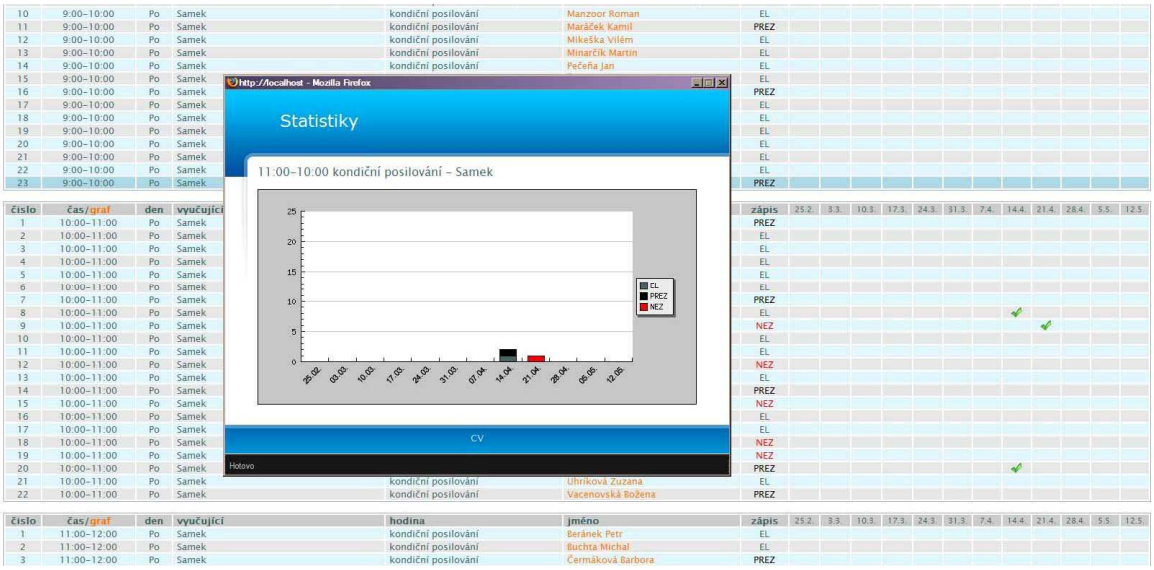


Obrázek 1: Detail přihlašovacího formuláře



Obrázek 2: Ukázka docházkového seznamu a možnosti filtrování

Obrázek 3: Detail vyhledávání osob a podrobných informací



Obrázek 4 : Detail statistiky

2 Příklady dotazů na databázi

```
$dotaz2="SELECT (CASE WHEN (('Student FAST'=L.fakulta OR
                             'Student FSI'=L.fakulta  OR
                             'Student FEKT'=L.fakulta OR
                             'Student FIT'=L.fakulta  OR
                             'Student FCH'=L.fakulta  OR
                             'Student FP'=L.fakulta   OR
                             'Student FA'=L.fakulta   OR
                             'Student FP'=L.fakulta) AND
                                L.rc=".$kdo.") THEN studenti
                        WHEN ('Zaměstnanec VUT'=L.fakulta AND L.rc=".$kdo.") THEN zamestnanci
                        WHEN (('Student jiné VŠ'=L.fakulta OR
                               'Žák ZŠ'=L.fakulta          OR
                               'Zaměstnanec VUT'=L.fakulta OR
                               'Zaměstnanec jiné V'=L.fakulta OR
                               'Absolvent VUT'=L.fakulta   OR
                               'Ostatní'=L.fakulta) AND
                                  L.rc=".$kdo.") THEN ostatni
                    END) AS cena,H.hid,L.rc,K.konto
       FROM hodiny_cenik HC,".$hodiny." H,lide L,konto K
       WHERE H.cenova_trida=HC.cid
       AND L.rc=".$kdo."
       AND L.rc=K.majitel
       AND H.hid=(SELECT H.hid
                  FROM ".$hodiny." H
                  WHERE H.kde=".$kde."
                  AND H.den=(case DATE_FORMAT(CURDATE(),'%w') WHEN 1 THEN 'Po'
                                                              WHEN 2 THEN 'Út'
                                                              WHEN 3 THEN 'St'
                                                              WHEN 4 THEN 'Čt'
                                                              WHEN 5 THEN 'Pá'
                                                              WHEN 6 THEN 'So'
                                                              WHEN 7 THEN 'Ne'
                             END)
                  AND ( ADDTIME(CURRENT_TIME(),'0:".$posun.":0') BETWEEN H.od AND H.do))";
```

Dotaz 1: Získání informací o studentovi a hodině

```
SELECT SP.nazev,L.prijmeni,L.jmeno,TIME_FORMAT(H.do, '%k:%i') AS do,
       TIME_FORMAT(H.od, '%k:%i') AS od,H.den,Z.typzapisu,
       SUM(case when ((KT.kdy between '2008-02-25' and '2008-03-02' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't1',
       SUM(case when ((KT.kdy between '2008-03-03' and '2008-03-09' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't2',
       SUM(case when ((KT.kdy between '2008-03-10' and '2008-03-16' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't3',
       SUM(case when ((KT.kdy between '2008-03-17' and '2008-03-23' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't4',
       SUM(case when ((KT.kdy between '2008-03-24' and '2008-03-30' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't5',
       SUM(case when ((KT.kdy between '2008-03-31' and '2008-04-06' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't6',
       SUM(case when ((KT.kdy between '2008-04-07' and '2008-04-13' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't7',
       SUM(case when ((KT.kdy between '2008-04-14' and '2008-04-20' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't8',
       SUM(case when ((KT.kdy between '2008-04-21' and '2008-04-27' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't9',
       SUM(case when ((KT.kdy between '2008-04-28' and '2008-05-04' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't10',
       SUM(case when ((KT.kdy between '2008-05-05' and '2008-05-11' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't11',
       SUM(case when ((KT.kdy between '2008-05-12' and '2008-05-18' )
                      AND KT.kdo=Z.rc AND KT.co=Z.hid) then 1 END) as 't12',
       H.vyucujici,L.rc,H.hid
FROM zapsani Z,hodiny H,sporty SP,sportoviste S,lide L,konto_transakce KT
WHERE Z.hid=H.hid
AND H.kodsportu=SP.kodsportu
AND H.kde=S.id_loc
AND L.rc=Z.rc
AND S.id_loc='36'
GROUP BY Z.rc,Z.hid
ORDER BY H.den,H.od,L.prijmeni
LIMIT 0,60
```

Dotaz 2: Výpis docházkového seznamu