Managersko-ekonomická olympiáda - elektronické zpracování testů

Electronic tests processing - MEO

Roman Ryba

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Roman RYBA**
Osobní číslo:	**A07088**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Managersko-ekonomická olympiáda – elektronické zpracování testů**

Zásady pro vypracování:

1. **Analýza současného stavu problematiky.**
2. **Návrh způsobu testování v rámci vytvářeného informačního systému.**
3. **Naplnění testové části databáze.**
4. **Navržení a popis průbehu vyhodnocování testů.**
5. **Zajištění správy a zabezpečení testových otázek.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **LACKO, L. PHP a MySQL – hotová řešení. Computer Press, 2006, ISBN: 80-251-1249-7.**
2. **PROKOPOVÁ, Z.: Databázové systémy MySQL+PHP. FAI UTB Zlín, s. 126, 2006, Vysokoškolská skripta. ISBN 80-7318-486-9.**
3. **RIORDAN, Rebecca M. Vytváříme relační databázové aplikace. Praha : Computer Press, 2000. 280 s. ISBN 80-7226-360-9.**
4. **SCHNEIDER, R.,D. MySQL – Oficiální průvodce tvorbou, správou a laděním databází. Grada, ISBN: 80-247-1516-3.**
5. **ULLMAN, L. PHP a MySQL, Computer Press, Brno, 2004,534 s. ISBN 80-251-0063-4.**
6. **WELLING, L., THOMSON, L. MySQL Průvodce základy databázového systému. Computer Press, Brno, 2005, ISBN: 80-251-0671-3.**

Vedoucí bakalářské práce: **doc. Ing. Zdenka Prokopová, CSc.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.

doc. Ing. Ivan Zelinka, Ph.D.

ABSTRAKT

Cílem předkládané bakalářské práce bylo navrhnout a naprogramovat aplikační rozhraní pro testovou část Managersko-ekonomické olympiády a vytvořit tak náhradu za doposud rozesílanou papírovou formu testu. Konkrétně jde o vytvoření rozhraní jak pro vyplňování testu, tak pro správu otázek a přístupu k testům. Testové otázky jsou zařazeny do více skupin a ke každé otázce muže být dle skupiny různý počet odpovědí. Vytvořený program každému uchazeči vygeneruje "unikátní" test z definovaných otázek a odpovědí.

Klíčová slova: PHP, XHTML, MySQL, www, JavaScript, test, generátor

ABSTRACT

The aim of this bachelor project is to design and program a filling form for the test part of the MEO to create a substitute to paper version of the test. In the concrete to create a filling form for the test and form for modification of the questions and to set time when the test can be generated and filed. Test questions are in several categories and in each category can be a different number of answers. And generator that creates for each person “unique“ test from inserted questions and answers.

Keywords: PHP, XHTML, MySQL, www, JavaScript, test, generator

Rád bych poděkoval své vedoucí bakalářské práce doc. Ing. Zdeně Prokopové, CSc., za odborné vedení, cenné rady a hlavně za čas věnovaný mé práci.

Motto:

> *Jestliže jste postavil vzdušný zámek, vaše práce nemusí být zbytečná. Nyní k němu dodělejte základy.*
>
> Henry David Thoreau

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….……

podpis diplomanta

OBSAH

ÚVOD

Managersko-ekonomická olympiáda je test určen pro třetí ročníky středních škol a pořádá jej Fakulta managementu a ekonomiky Univerzity Tomáše Bati ve Zlíně. Managersko-ekonomickou olympiádu má na starosti realizační tým, který se skládá ze studentů Fakulty managementu a ekonomiky Univerzity Tomáše Bati ve Zlíně. Měli požadavek na letošní ročník olympiády, aby byl modernizovaný z papírové formy pro dnešní věk počítačů a internetu. Avšak tento rok ještě měly školy na výběr, zdali chtějí psát olympiádu papírovou formou nebo vytvořeným webovým rozhraním. Od realizačního týmu jsem převzal seznam otázek papírové verze předkola testu a podle něj vytvořil typy a druhy otázek pro elektronickou formu testu. Hlavním záměrem je zjednodušit práci realizačního týmu s testem Managersko-ekonomické olympiády. V dnešní době je internet na všech školách a není více multiplatformní jazyk než HTML nebo v dnešní době modernější XHTML. U programování v PHP, které pomáhá dynamicky generovat webové stránky, se nemusíme starat o to, jaký operační systém nebo jaký prohlížeč používá, protože část naprogramovaná v PHP se vykonává na serveru a uživatel ve výsledku vidí stránku, jako by byla napsána v HTML. Dříve, když byly webové stránky statické pouze v HTML, by nebylo možné podobnou aplikaci vytvořit, dnes však existuje více než jedna možnost jak generovat dynamicky webové stránky a dodat tím tak dynamičnost danému projektu. Nejdůležitější podstata této práce tedy je dynamické webové rozhraní pro generování, vyplňování a vyhodnocení testů jednotlivým uchazečům Managersko-ekonomické olympiády a vytvoření rozhraní pro správu otázek a odpovědí a administrace okolo testové části tak, aby byl použitelný i pro další ročníky. Managersko-ekonomická olympiáda je umístěna na adrese < http://olympiada.fame.utb.cz/ >. Předkolo proběhlo dne 16. 4. 2010, finále proběhlo 12. 6. 2010 a očekávám, že pro další ročníky vzniknou i další typy otázek podle požadavků realizačního týmu na formu testu.

I. TEORETICKÁ ČÁST

1 INFORMACE K OLYMPIÁDĚ

1.1 Papírová forma

Při papírové formě probíhala veškerá komunikace mezi školou, zúčastňující se olympiády, a realizačním týmem prostřednictvím České pošty. Vyhodnocování papírové formy testu bylo prováděno inteligentním skenováním každého výsledkového archu. V letošním roce bylo přihlášeno celkem 54 škol a 1587 studentů celkem. Z pohledu ceny a času je to odeslaní dopisů školám, v letošním roce bylo osloveno 324 škol, jestli se chtějí zúčastnit Managersko-ekonomické olympiády. Pak se musí čekat na doručení a odpověď. Potom se musí odeslat testy a výsledkové archy studentům na školy, které souhlasily s účastí na olympiádě. Po vyplnění výsledkových archů předkola se musí odeslat zpět ke zpracování realizačnímu týmu. Tyhle testy se pak strojově vyhodnotí a odešlou se výsledky a pozvánky na finále. To pro nás znamená zaplatit odeslání přibližně 320 dopisů a také potištění spousty papíru. Když budeme počítat s tří denní lhůtou doručení, tak od odeslání prvního dotazu na účast na předkole až po odeslaní výsledků testu a pozvánky na finále uplyne alespoň 15 dní (nepočítá se s tím, že přes víkend pošta nedoručuje a čas, který zabere vyhodnotit test). Hlavním cílem téhle práce je minimalizovat náklady a časovou náročnost pro zpracování testu Managersko-ekonomické olympiády

1.2 Možnosti realizace

V dnešní moderní době počítačů a internetu neexistuje snadnější, levnější a méně časově náročnější řešení, než je pomocí programu odesílat data přes internet, která buď obsahují data ke zpracování, nebo rovnou výsledná pouze k setřídění.

Jedna možnost je vytvořit aplikaci jako vlastní program v některém jazyce, například C, C++,C#,F#, Java, .NET, Pascal a další. Jedna ze dvou věcí, která by se dala vytknout takovéhle realizaci, je vyšší složitost při vytvoření aplikace, která bude bezchybně fungovat na všech operačních systémech. Druhá věc je distribuce takového softwaru ke koncovým uživatelům.

Druhá možnost je vytvořit aplikaci pomocí webového rozhraní s přístupem pomocí internetového prohlížeče. Taková aplikace napsaná podle standardů pro HTML nebo XHTML by neměla mít problém s funkčnosti na jakémkoliv operačním systému, jenž má

kterýkoli webový prohlížeč. Výhodou je, že není potřeba distribuce ani instalace. Všechno je na serveru, který je přístupný z internetu pomocí individuálního prohlížeče.

1.3 Vybraná realizace

Pro realizaci testu byla vybrána forma webové aplikace. Také nám vyhověli s žádostí a byla nám přidělena subdoména na serveru utb.cz a diskový prostor na školním serveru. Server je Apache s MySQL databází, což pro nás znamená, že využijeme jazyk PHP a MySQL databázi pro dynamické zobrazování stránek. Při téhle možnosti realizace, jestliže budeme dodržovat standardy pro XHTML, si zajistíme i kompatibilitu s jakýmkoli webovým prohlížečem na jakémkoli operačním systému. Pouze například u rozšíření jako je javascript, si musíme dávat pozor, protože jejich podpora nemusí být povolena. V takovémto případě by mohly nastat potíže s funkčností, proto na to nesmíme zapomínat.

2 WEBOVÉ TECHNOLOGIE

2.1 Značkovací jazyky

Značkovací jazyk je jazyk, který kromě vlastního textu obsahuje také instrukce pro jeho zpracování definované ve značkách. Značky musí být jednoznačně odlišeny od textu. Ty slouží k definování, jak bude zpracován text uvnitř anebo co daný text uvnitř znamená pro aplikaci, která jej zpracuje.

2.1.1 Rozdělení značkovacích jazyků

Značkovací jazyky lze rozdělit do dvou skupin.

2.1.1.1 Jazyky popisné (deskriptivní)

Jejich konstrukce slouží k popisu informací obsažených v dokumentu. Typickými představiteli jsou XML a HTML- jejich prostřednictvím jdou například vymezit na stránce rámce, bloky informací, styly, odkud kam sahají jednotlivé odstavce či popsat odkaz na jinou stránku. Je ponecháno na zpracovávajícím programu, jak s těmito informacemi naloží a jak je promítne kupříkladu do zobrazení dokumentu.

2.1.1.2 Jazyky výkonné (procedurální)

Obsahují i instrukce na úrovni programovacího jazyka. Využití proměnných + nástroje pro přiřazování a využívání jejich hodnot. Výkonné jazyky zpravidla také umožňují popisovat vizuální charakteristiky výstupu. Uživatel může tedy velmi přesně řídit vzhled výsledného dokumentu. Mezi procedurální jazyky patří TeX či PostScript.

2.1.2 HTML

Jazyk vychází z dříve vyvinutého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn rozvíjejícím se vývojem webových prohlížečů a potřebami uživatelů. Od verze 2.0 je HTML aplikací SGML(Standard Generalized Markup Language - je univerzální značkovací metajazyk, který umožňuje definovat značkovací jazyky jako své vlastní podmnožiny), kromě HTML5, ten již nebude vycházet z SGML.

HTML používá párové a nepárové značky (tagy). Značku definují znaky < a > uvnitř se nachází text, to také znamená, že se znaky < a > nesmí objevit v textu uvnitř značek, protože by byl jejich obsah vyhodnocen právě jako značka. Jako náhrada za znaky < a > se používá &lt a > které se ve výsledném textu zobrazí jako < a >. Párové značky mají například formu:

```
1 <a href="olympiada.fame.utb.cz">Naše E-M Olympiada</a>
```

Každá párová značka musí být zakončena stejnojmennou značkou, jež má před jménem /.

Nepárové značky nemají párovou ukončovací značku, ale ukončují se samy na konci s /.

```
2 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

2.1.3 XML

Extensible Markup Language (XML) je velmi jednoduchý, velmi flexibilní textově formátovaný značkovací jazyk. Byl vyvinut a standardizován konsorciem W3C zjednodušením staršího jazyka SGML. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů, u kterých popisuje strukturu z hlediska věcného obsahu.

2.1.4 XHTML

Extensible hypertext markup language – rozšiřitelný hypertextový značkovací jazyk je novější forma jazyka HTML. Podpora jazyka XHTML je v současných prohlížečích naprosto stejná jako podpora HTML.

XHTML se v praxi vyskytuje ve třech verzích:

- XHTML 1.0 přechodové (transitional)
- XHTML 1.0 striktní (strict)
- XHTML 1.1

Zatímco XHTML 1.0 transitional je nejjednodušší a nejvíce podobná HTML, XHTML 1.0 striktní je velmi pracné v ohledu na dodržování syntaxe, v XHTML 1.1 je spousta věcí zakázána a jeho používání je ještě pracnější.

2.2 Webové programovací jazyky

2.2.1 AJAX

AJAX (Asynchronous JavaScript and XML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Tyto aplikace jsou vyvíjeny s využitím technologií:

- HTML (nebo XHTML) a CSS pro prezentaci informací
- DOM a JavaScript pro zobrazování a dynamické změny prezentovaných informací
- XMLHttpRequest pro asynchronní výměnu dat s webovým serverem

Ajax ve skutečnosti není konkrétní jednotlivá technologie, ale pojem označující použití několika technologií dohromady s určitým cílem.[10]

2.2.2 PHP

PHP - Hypertext Preprocessor (Hypertextový preprocesor) původní název byl Personal Home Page (osobní domácí stránky) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek.[5] Informace o nejnovějších verzích nalezneme na http://php.net/, kde je také nejrozsáhlejší dokumentace k PHP v mnoha světových jazycích.

2.2.2.1 Pohled do historie

Verze 1.0.0 vydaná 8. června 1995 Oficiální název - Personal Home Page Tools

Verze 2.0.0 nastroj pro tvorbu dynamických webových stránek

Verze 3.0.0 přepsán celý základ PHP

Verze 4.x.x Přidán Zend engine, přidány proměnné ($_GET, $_POST, $_SESSION, etc.),

Verze 5.x.x Nový Zend engine II s novým objektovým modelováním. [9]

V začátcích PHP by pro nás bylo velmi těžké a možná i nemožné vytvořit takovou aplikaci jakou máme možnost vytvořit dnes v PHP verzi 5. Dnes si nedokážu představit vytvořit aplikaci bez proměnných typu $_GET, $_POST, $_SESSION, jejich využití je v PHP mnohostranné ale hlavně usnadňují práci s předáváním informací a parametrů, to se nejvíce využívá k zjišťování oprávnění přístupu uživatele na danou stránku.

2.2.3 JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich. JavaScript je závislý na prohlížeči (uživatel si může vypnout). V různých verzích prohlížečů nemusí skript vždy korektně fungovat. Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej s názvem spojuje jen podobná syntaxe. JavaScript umožňuje například vytvořit hodiny, hodnotit data ve formuláři, počítat, dynamizovat data, také umožňuje tvorbu všemožných prvků k oživení webu, přes blikající texty po jednoduché hry. Základem dynamického webu je právě JavaScript. K práci s JavaScriptem byste měli znát základy HTML.

2.2.4 JSP

Java Server Pages je nástroj pro psaní dynamických HTML stránek, založený na jazyce Java, funkčností velmi podobný ASP nebo PHP. Jedná se vlastně o HTML stránky, do nichž je pomocí speciálních značek vložen kód v Javě, který se provádí při vyřizování dotazu na straně serveru. Hlavní odlišností je kompilace stránek do tzv. *servletů*, což jsou speciální třídy v jazyce Java, které pak komunikují s webovým serverem. JSP by se tedy daly také charakterizovat jako nástroj na psaní servletů.

2.2.5 ASP

ASP (Active Server Pages) je skriptovací platforma společnosti Microsoft, primárně určená pro dynamické zpracování webových stránek na straně serveru. Její nástupce, ASP.NET, lze chápat jako širší a komplexnější technologii, která se od ASP v mnoha ohledech liší. ASP.NET je založen na CLR (Common Language Runtime), který je sdílen všemi aplikacemi postavenými na .NET Frameworku. Díky tomu lze realizovat projekty v jakémkoliv jazyce podporujícím CLR, např. Visual Basic.NET, JScript.NET, C#, Managed C++, ale i mutace Perlu, Pythonu a další. Aplikace založené na ASP.NET jsou také rychlejší, neboť jsou předkompilovány do jednoho či několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu generovány.

2.3 Databázové systémy

Databázový systém slouží k uchování databází (tj. tabulek údajů o stejné struktuře). Tyto údaje můžeme vkládat, opravovat, zobrazovat, třídit podle různých hledisek, spojovat s jinými databázemi, kopírovat pro použití v jiných systémech (ve vlastních programech), vytvářet křížové tabulky, údaje zobrazovat v různých typech grafů.

2.3.1 SQL

SQL (Structured Query Language – strukturovaný dotazovací jazyk)

Specializovaný programovací jazyk určený k práci s daty umožňuje realizovat všechny operace s daty. Používá se především v relačních databázích. SQL lze použít k založení databáze, definování struktury tabulek, stanovení integritních omezení, vkládání dat do databáze, výběr dat z databáze, zobrazení dat. Neobsahuje prostředky pro formátování výstupů, pouze pro definici jejich obsahu. Komunikace v aplikacích probíhá jako klient/server. SQL server zajišťuje správu a provoz databáze, vykonávání programů v jazyce SQL a vracení jejich výsledků klientskému programu. SQL klient formuluje a odesílá dotaz nebo příkaz pro manipulaci s daty v jazyce SQL pro zpracování na straně SQL serveru.[2]

2.3.1.1 MySQL

MySQL je multiplatformní databáze. Komunikace s ní probíhá pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o rozšíření tohoto jazyka. MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení - má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery a uložené procedury. Díky tomu, že je multiplatformní, rychlá, výkonná a navíc volně šiřitelná, má vysoký podíl mezi v současné době používanými databázemi. Velmi oblíbená je kombinace MySQL s PHP a Apache jako základ webového serveru.[3]

2.3.1.2 MicrosoftSQL

Databázová platforma MS SQL je dostupná jen pro operační systémy Microsoft Windows. Transact-SQL (T-SQL) je rozšíření SQL jazyka o další možnosti od společností Microsoft a Sybase. Nejvíce se využívá pro stránky napsané v jazyku ASP, který je taktéž od společnosti Microsoft.

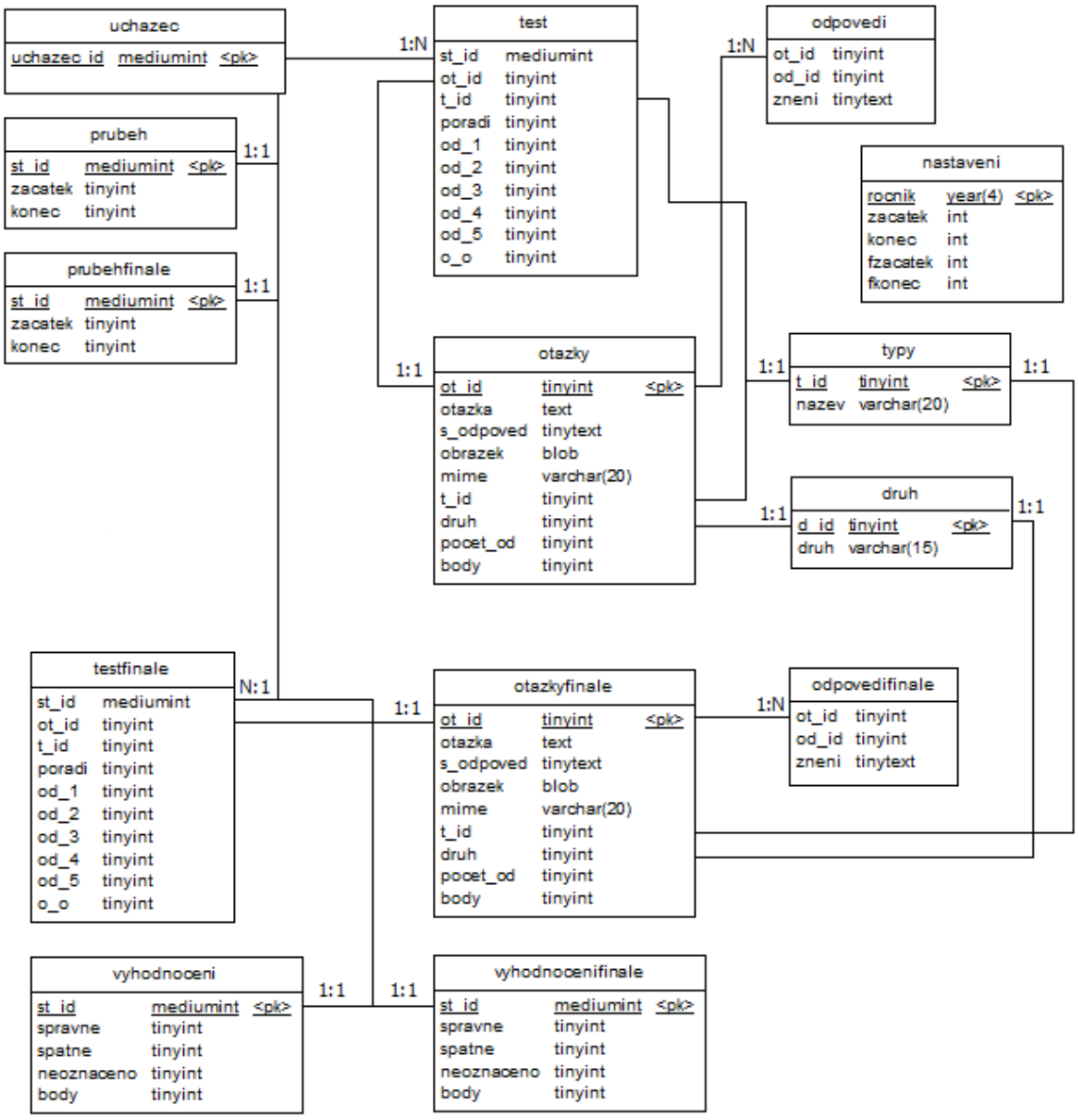
2.3.2 Oracle

Oracle je systém řízení báze dat (Oracle database management system – DBMS), moderní multiplatformní databázový systém s velice pokročilými možnostmi zpracování dat, vysokým výkonem a snadnou škálovatelností. Správa databáze Oracle je prováděna na fyzické a logické úrovni. Logická struktura databáze je množina objektů a vztahů mezi nimi. Základem logické struktury systému Oracle jsou tabulky, do kterých Oracle ukládá všechna data. Fyzická struktura databáze je množina souborů operačního systému, uložených na pevném disku. Aktuální verzí je Oracle Database 11g. Tento systém podporuje nejen standardní relační dotazovací jazyk SQL, ale také jazyk PL/SQL od firmy Oracle, rozšiřující možnosti vlastního SQL.

II. PRAKTICKÁ ČÁST

3 STRUKTURA TABULEK V MYSQL

Při návrhu databáze je velice důležité zvolit správnou strukturu tabulek. V tomto konkrétním případě jsem vybral strukturu databáze trochu rozsáhlejší. Překládané schéma by bylo možné minimalizovat přidáním indexu do tabulek, které jsou rozděleny na předkolo a finále. Z hlediska častého přístupu do těchto tabulek během testu jsem chtěl snížit počet jejich řádků, aby byla rychlejší odpověď na MySQL dotaz z php.



Obr. 1. Struktura tabulek v MySQL

3.1 Tabulka uchazec

Tato tabulka je vytvořena mým kolegou, který se stará o správu uživatelských účtů.

Je velmi důležité tuhle tabulku svázat s tabulkami testu a průběhu, abychom mohli jednoznačně určit, který test psal který uchazeč.

3.2 Tabulka test

Tabulka obsahuje informace o testech jednotlivých studentů.

Položka st_id reprezentuje číslo typu unsigned smallint, které dokáže jednoznačně identifikovat až 65535 různých uživatelů, pro které je test určen. Položka od_id je číselný identifikátor typu unsigned tinyint, který určuje, o jakou otázku se z tabulky otázek jedná. Položka t_id typu unsigned tinyint určuje typ otázky, do této tabulky jsem jej přidal z důvodu, abych minimalizoval složitost SQL dotazu a snížil tak počet přístupů do databáze, protože přes tento parametr se provádí základní dělení otázek. Položka poradi typu unsigned tinyint určuje pořadí, v jakém se otázky zobrazují v testu. Položka od_1, od_2, od_3, od_4, od_5 typu unsigned tinyint obsahuje číslo určující, která z odpovědí pro danou otázku se bude zobrazovat. Položka o_o typu unsigned tinyint určuje odpověď, kterou student u otázky označil.

3.3 Tabulka otazky

Položka od_id typu unsigned tinyint je primárním klíčem, díky tomu se nám nemůže stát, aby některá z otázek měla stejné id jako otázka jiná. To by způsobilo velké problémy hlavně při vyhodnocování testu. Položka otazka obsahuje znění otázky typu text, jež může obsahovat až 65535 znaků. To by mělo stačit pro jakoukoliv otázku, zřejmě by stačilo i míň znaků, ale nejbližší nižší datový typ má maximální hodnotu 255 znaků, což mi připadá zase málo pro znění otázky. Položka s_odpoved obsahuje znění správné odpovědi, je typu tinytext, která může mít maximální délku řetězce 255 znaků, což by mělo stačit pro znění odpovědi. Položka obrazek je typu blob, v téhle položce je uložen obrázek jako binarní posloupnost o maximální délce 65535 znaků, neboli obrázek o maximální velikosti 64KiB. Položka mime typu varchar(20) je řetězec o maximální délce 20 znaků, ve němž je uložen mime typ souboru. Díky tomuhle záznamu je možno reprodukovat více než jen jeden typ obrázku. Položka t_id typu unsigned tinyint určuje, jakého je otázka typu. Položka d_id

určuje, jakého je otázka druhu. Položka pocet_od typu unsigned tinyint určuje, kolik má mít otázka zobrazených odpovědí, v tomhle provedení je možné zobrazit maximálně 5 odpovědí k jedné otázce. Položka body typu unsigned tinyint určuje, kolik bodů dostane uchazeč za zodpovězení dané otázky správně.

3.4 Tabulka odpovedi

Tabulka obsahuje pouze špatné odpovědi k otázkám, jednoznačnost odpovědí je dána kombinací ot_id a od_id. Položka ot_id typu unsigned tinyint určuje, ke které otázce patří daná odpověď. Položka od_id typu unsigned tinyint udává index dané odpovědi k otázce, s velikostí unsigned tinyint je možné k jedné otázce přiřadit až 255 odpovědí. Položka znění typu tinytext obsahuje znění odpovědi o maximální délce 255 znaků.

3.5 Tabulka typy

Účel tabulky je přiřadit číslu typu jeho název. Položka t_id typu unsigned tinyint je primárním klíčem a určuje číslo, podle kterého se sjednocují otázky daného typu. Položka nazev typu varchar(15) obsahuje název daného typu otázky.

3.6 Tabulky druh

Stejně jako u typu je účelem této tabulky přiřadit číslu druhu otázky jeho název. Položka d_id typu unsigned tinyint je primárním klíčem a určuje číslo, podle kterého se stanoví, jakého je otázka druhu. Položka druh typu varchar(15) obsahuje název daného druhu otázky.

3.7 Tabulka prubeh

V této tabulce jsou informace o tom, kdy který student započal a ukončil test. Položka st_id typu unsigned mediumint je primárním klíčem a jednoznačně určuje, kterému uchazeči paří daný řádek v tabulce. Položka zacatek typu unsigned int obsahuje čas, kdy byl test úspěšně vygenerován a započat. Datový typ byl zvolen int, protože čas je uložen v unixovém formátu. Položka konec typu unsigned int obsahuje čas konce testu v unixovém formátu.

3.8 Tabulka vyhodnoceni

Tabulka obsahuje výsledky jednotlivých studentů v testu. Položka st_id typu unsigned mediumint je primárním klíčem a jednoznačně určuje, kterému uchazeči patří daný řádek v tabulce. Položka spravne typu unsigned tinyint obsahuje počet odpovědí vyhodnocených jako správné. Položka spatne typu unsigned tinyint obsahuje počet odpovědí vyhodnocených jako špatné. Položka neoznačeno typu unsigned tinyint obsahuje počet neoznačených odpovědí. Položka body typu unsigned tinyint obsahuje počet bodů za otázky vyhodnocené jako správné.

3.9 Tabulky pro finále

Tabulky mají stejnou strukturu, jako tabulky pro předkolo, pouze jejich názvy jsou jiné. Pro tohle rozdělení jsem se rozhodl z hlediska rychlosti přístupu k datům v MySQL databázi.

3.10 Tabulka nastaveni

Tabulka obsahuje informace o tom, kdy je přístupné generování testu v daném ročníku.

Položka rocnik typu year(4) je primárním klíčem, je důležité, aby neexistovalo více záznamů pro jeden ročník, to by mohlo způsobit, že by se negenerovaly testy v daném ročníku. Položka zacatek typu unsigned int určuje, odkdy je přístupné generování testu předkola a položka konec stejného typu určuje, dokdy je přístupné generování. Položky fzacatek a fkonec jsou stejné jako zacatek a konec jen s tím rozdílem, že omezují generování testu finále.

4 PROVEDENÍ V PHP

4.1 Generování testu

U generátoru testu - náhodného míchání jsem se rozhodoval, jestli mám generovat test uchazečům předem, anebo jestli se jim má vygenerovat test až po tom, co kliknou na odkaz na stranách, který je bude na test odkazovat. Nejdříve při testovaní na localhostu na mém počítači mi trvalo vygenerování jednoho testu přibližně 0,9 sekundy. Po odstranění veškerých pomocných výpisu, které mi pomohly daný skript doladit, se čas vykonání snížil na hodnotu přibližně 0,56 sekundy. Takový čas se mi zdá vysoký, když uvažujeme, že by si v jeden okamžik zažádali až tisíce lidi o generování testu. V tom případě by mohl nastat problém a server by nemusel zvládnout vyřídit všechny požadavky. Při generování testu předem by se problém se zatížením neprojevil. Skript by se vykonal sekvenčně. V takovém případě se mi ale vyskytl problém - při generování testu pro velký počet uživatelů mi server oznámil „Fatal error: Maximum execution time of 60 seconds exceeded“, což znamená, že server věnuje zpracování jednoho php souboru maximálně 60 sekund. Tahle hodnota se dá změnit buď v php.ini na serveru, nebo přidat do skriptu příkaz „set_time_limit(0)“, kde číslo v závorce udává, kolik sekund se maximálně server bude věnovat zpracování php souboru. Pokud nastavíme 0, je maximální čas nastaven na nekonečno. Když jsem skript testoval na serveru, kde bude uložen, jeho vykonání bylo mnohem rychlejší než na mém počítači na localhostu. Můj počítač generoval 1600 testů přibližně 185 sekund a server to zvládl za 45sekund. Z toho jsem usoudil, že nebude žádný problém, když se testy budou generovat při prvním přístupu k testu. Ve finální verzi se jeden test jednomu uchazeči generoval přibližně 0,02 sekundy, a i kdyby nastalo velké zatížení z hlediska vysokého počtu uživatelů, neměl by se vyskytnout problém. Maximálně se prodlouží čas, než se test vygeneruje. Čas k odpočítávání se ukládá až na konci cyklu, takže by ani neměli být uživatelé „okradeni“ o čas jim přiřazený z hlediska delšího času vygenerování.

4.1.1 Zabezpečení přístupu

Generátor náhodného míchání otázek pracuje tak, že se připojíme k databázi MySQL, kde jsou uloženy otázky, se kterými budeme pracovat. Nejdříve se ale provede kontrola, zdali je administrátorem zvolený interval, v němž si uživatel může nechat vygenerovat test.

Kontrola se provádí tak, že se dotážeme serveru pomocí php na aktuální rok a podle něj se vyberou hodnoty z tabulky v MySQL databázi. Dále se pomocí php serveru dotážeme na aktuální čas v unixovém formátu, který se pak vyhodnotí, jestli je jeho hodna větší než minimální čas a menší než maximální čas, který jsme získali z databáze. Pokud jsou tyto podmínky splněny, přejde se k další kontrole, která zjisti, zda byl test už vygenerován, aby se nedal vygenerovat vícekrát. To by mohlo způsobit různé problémy. Kontroluje se výběrem z databáze, jestli existuje informace o tom, kdy má test započít a čas konce testu, protože tyhle informace se vkládají do databáze až jako poslední. Nakonec se provede kontrola, jestli se vložily všechny otázky. Pokud je splněna i podmínka, že neexistuje záznam o začátku a konci testu, přejde se k vygenerování testu.

4.1.2 Míchání otázek

Nejprve se naplní pole otázkami tak, že se vyberou postupně podle toho, do kterého typu patří. V mém řešení pro M-EO je generování omezeno pouze na 4 typy a pokud si bude přát realizační tým rozšíření pro více typů, nejde o nic složitého jej připsat do zdrojového kódu. Po uložení všech otázek a jejich informací potřebných pro vygenerování do pole se začne provádět míchání v poli podle skupin, jak si přál realizační tým. Míchání se provádí tak, že se spočítá maximální hodnota náhodného čísla, které se vygeneruje, vygenerujeme náhodné číslo se zvolenou maximální hodnotou, které použijeme jako index při výběru z pole, ve kterém jsou uloženy otázky, a uložíme tento prvek na první pozici v novém poli výsledných otázek. Poté vezmeme poslední otázku daného typu z pole a uložíme ji na pozici prvku, který jsme uložili do nového pole, a cyklus se opakuje s tím rozdílem, že se pokaždé sníží maximum náhodného čísla o 1, protože máme o jednu otázku v poli míň. Až se takhle „přehází“ všechny otázky jednoho typu, přesune se míchání na otázky dalšího typu, dokud nezamícháme i část pole se čtvrtým typem otázek. Když je tohle hotové, přejde se k zamíchání odpovědí k jednotlivým otázkám.

4.1.3 Míchání odpovědí

Postupně se prochází celé pole otázek a nejdříve u každé zjistíme, jaký je její druh, abychom vybrali správný způsob zamíchání. Zatím jsem podle materiálu od realizačního týmu vymyslel 2 typy otázek. První je obyčejná, kde se vyberou všechny špatné odpovědi z databáze a zjistí se jejich počet a podle toho, zdali je jejich počet menší než nastavený počet odpovědí, se postupně přiřadí do pole a na konec pole se připojí i správná odpověď.

Toto pole se posléze zamíchá stejným principem jako pole otázek a přejde se na další otázku. Pokud je jejich počet stejný nebo větší, vybere se náhodně takový počet otázek, jaký je nastavený pro danou otázku a nakonec se náhodně nahradí jedna z těchto odpovědí správnou odpovědí a přejde se na další otázku. Druhý typ je „abcde“, kde není potřeba míchání. Prostě se dosadí postupně odpovědi a přejde se na další otázku. Potom co jsou všechny otázky i odpovědi zamíchané, přejde se k uložení vygenerovaného testu do databáze. Postupně se projdou pole, která se během generování naplnila, a informace z těchto polí se uloží do SQL dotazu, který se provede. Po uložení otázek do databáze se spočítají v databázi všechny úspěšně uložené otázky patřící danému studentovi, a pokud neodpovídá počet otázek zadanému počtu otázek, smaže se všechno v databázi v tabulce testu pro daného studenta a vypíše se: generování se nezdařilo. V tom případě se musí generovat znovu po 3 sekundách. Pomoci meta tagu refresh se znovu načte stránka generátoru a tím se zahájí nové generování. Pokud ale počet otázek odpovídá zadanému číslu, tak se do databáze zapíše čas vygenerování testu a čas konce testu.

4.1.4 Přiklad

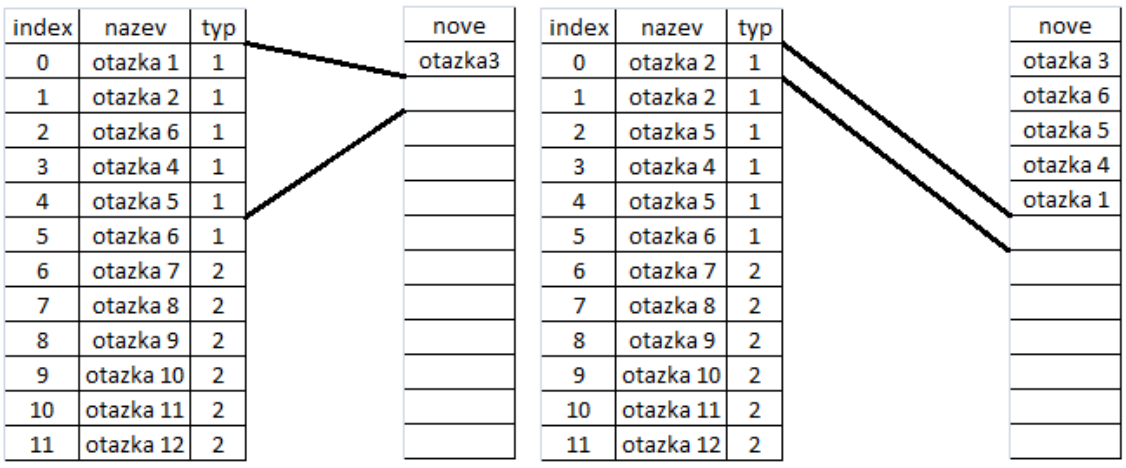
Jak to funguje, prakticky vysvětlím na následujícím přikladu, kde budeme mít 12 otázek dvou různých typů. V prvním kroku se vygeneruje číslo od 0 do celkového počtu otázek prvního typu. Vygeneruje-li se například číslo 2, znamená to, že na první pozici se dosadí otázka s číslem 3, protože se pole otázek indexuje od 0. Dále se sníží maximální hodnota generovaného náhodného čísla pro výběr otázek a pokračuje se dalším generováním.

index	nazev	typ	nove
0	otazka 1	1	
1	otazka 2	1	
2	otazka 3	1	
3	otazka 4	1	
4	otazka 5	1	
5	otazka 6	1	
6	otazka 7	2	
7	otazka 8	2	
8	otazka 9	2	
9	otazka 10	2	
10	otazka 11	2	
11	otazka 12	2	

index	nazev	typ	nove
0	otazka 1	1	otazka3
1	otazka 2	1	
2	otazka 3	1	
3	otazka 4	1	
4	otazka 5	1	
5	otazka 6	1	
6	otazka 7	2	
7	otazka 8	2	
8	otazka 9	2	
9	otazka 10	2	
10	otazka 11	2	
11	otazka 12	2	

Obr. 2. První kroky míchání otázek

Kdyby se jako další vygenerovalo znovu číslo 2, to znamená, že se dosadí na další pozici otázka 6, která byla v minulém kroku dosazena na pozici 2, protože otázka 6 byla na posledním místě otázek prvního typu. Takhle se pokračuje dále, dokud nezůstane poslední otázka a v tom případě se vygeneruje vždy 0 a přejde se na část pole dalšího typu.



index	nazev	typ
0	otazka 1	1
1	otazka 2	1
2	otazka 6	1
3	otazka 4	1
4	otazka 5	1
5	otazka 6	1
6	otazka 7	2
7	otazka 8	2
8	otazka 9	2
9	otazka 10	2
10	otazka 11	2
11	otazka 12	2

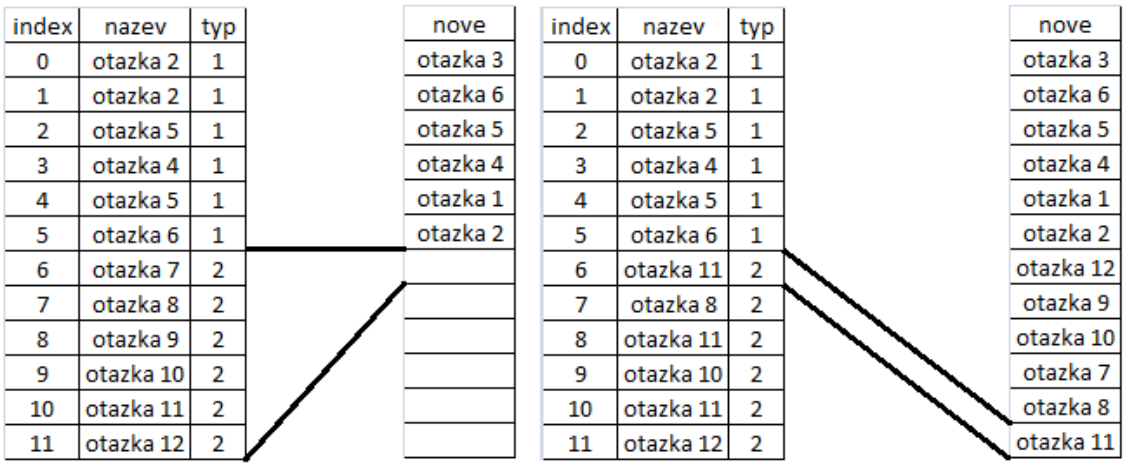
nove
otazka3

index	nazev	typ
0	otazka 2	1
1	otazka 2	1
2	otazka 5	1
3	otazka 4	1
4	otazka 5	1
5	otazka 6	1
6	otazka 7	2
7	otazka 8	2
8	otazka 9	2
9	otazka 10	2
10	otazka 11	2
11	otazka 12	2

nove
otazka 3
otazka 6
otazka 5
otazka 4
otazka 1

Obr. 3. Další kroky míchání otázek

U dalšího typu otázek se postupuje stejně s náhodným výběrem otázek jako u otázek prvního typu. Tak se postupuje až do vložení všech otázek do nového pole.



index	nazev	typ
0	otazka 2	1
1	otazka 2	1
2	otazka 5	1
3	otazka 4	1
4	otazka 5	1
5	otazka 6	1
6	otazka 7	2
7	otazka 8	2
8	otazka 9	2
9	otazka 10	2
10	otazka 11	2
11	otazka 12	2

nove
otazka 3
otazka 6
otazka 5
otazka 4
otazka 1
otazka 2

index	nazev	typ
0	otazka 2	1
1	otazka 2	1
2	otazka 5	1
3	otazka 4	1
4	otazka 5	1
5	otazka 6	1
6	otazka 11	2
7	otazka 8	2
8	otazka 11	2
9	otazka 10	2
10	otazka 11	2
11	otazka 12	2

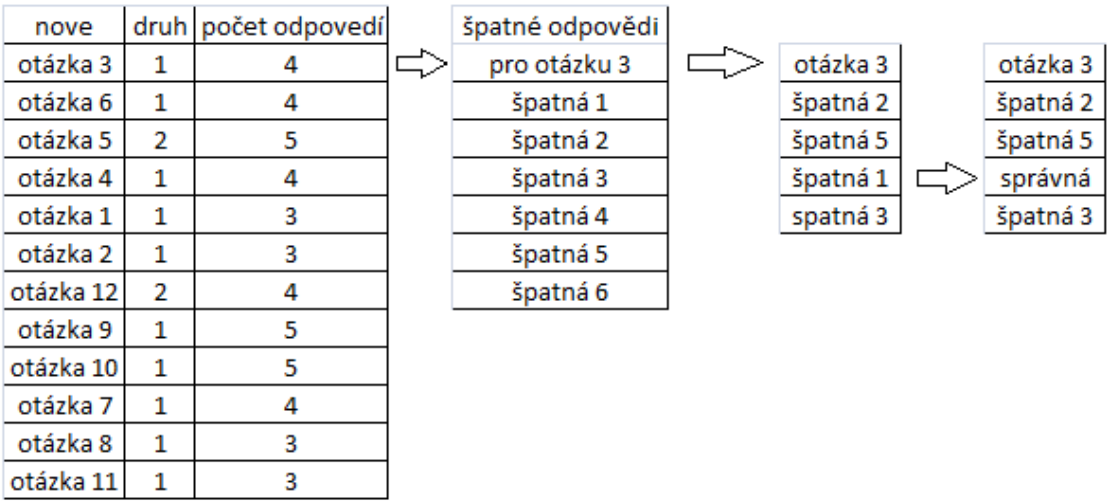
nove
otazka 3
otazka 6
otazka 5
otazka 4
otazka 1
otazka 2
otazka 12
otazka 9
otazka 10
otazka 7
otazka 8
otazka 11

Obr. 4. Finální kroky míchání otázek

Po naplnění tohoto pole jsme získali náhodně seřazené otázky podle typu. Když máme tohle pole, můžeme se přesunout na zamíchání odpovědí k jednotlivým otázkám.

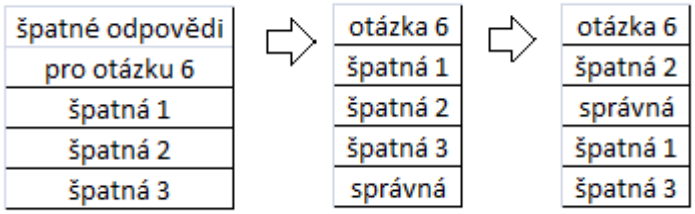
Postupuje se tím způsobem, že se projde nové pole, v němž jsou zamíchány otázky a podle toho, jaký má otázka druh, takovým se postupuje způsobem při zamíchání odpovědí.

Například otázka 3, která je na prvním místě v novém poli, je druh 1, což znamená, že to je obyčejná otázka. Dále se postupuje tak, že se vyberou z databáze všechny možné špatné odpovědi a spočítají se a jejich počet se porovná s počtem odpovědí, které mají být zobrazené u otázky. Například otázka 3 bude mít nastavený počet odpovědí na 4. Z databáze se vybere 6 špatných odpovědí, to znamená, že se z těch šesti špatných odpovědí vyberou náhodně 4 z nich a dosadí se do pole odpovědí pro danou otázku, po tomhle se vygeneruje náhodné číslo, které reprezentuje pozici, na kterou se dosadí správná odpověď a přejde se na další otázku v pořadí.



Obr. 5. Otázka druh 1 (obyčejná otázka) případ 1

Další otázka je otázka 6, ta má stejný druh jako předešlá otázka, ale po vybrání špatných odpovědí se zjistí, že jsou jen 3 špatné odpovědi, to znamená, že se k nim přiřadí jako čtvrtá správná odpověď, aby seděl počet odpovědí se zadaným počtem odpovědí, a pole odpovědí se náhodně zamíchá. A přejde se na další otázku v pořadí.



Obr. 6. Otázka druh 1 (obyčejná otázka) případ 2

Další otázka je otázka 5, u ní je nastavený druh 2, což je abcde druh otázky, v tomhle případě není nutné odpovědi vůbec míchat, prostě se postupně naskládají odpovědi za sebe, což ve výsledku znamená, že se zobrazí postupně odpovědi „a b c d e“ a můžeme se přesunout na další otázku.

nove	druh	počet odpovedí
otázka 3	1	4
otázka 6	1	4
otázka 5	2	5
otázka 4	1	4
otázka 1	1	3
otázka 2	1	3
otázka 12	2	4
otázka 9	1	5
otázka 10	1	5
otázka 7	1	4
otázka 8	1	3
otázka 11	1	3

otázka 5
a
b
c
d
e

Obr. 7. Otázka druh 2 (abcde otázka)

Takovýmto způsobem se postupuje u všech dalších otázek, dokud se nepřiřadí odpovědi všem otázkám. Když je tahle část hotová, je test připravený na zobrazení a vyplnění uchazečem.

4.2 Zobrazení testu

Obr. 8. Zobrazený test uchazečem

Po úspěšném vygenerování testu je uchazeč přesměrován na zobrazení testu, aby ho vyplnil v daném časovém limitu. Pokud by se sem uživatel snažil dostat před vygenerováním testu nebo po jeho ukončení, zobrazí se prázdná stránka.

V levé části obrazovky je plovoucí menu, posouvá se na stránce podle toho, kde se zrovna uživatel nachází. Je to, myslím, velmi užitečné tím, že uživatel nemusí vždy přejít nahoru nebo dolů na stránce, aby si test uložil nebo ukončil. V horní části menu jsou odkazy, které posouvají na začátek daných okruhů testu. Pod tím je zobrazený čas (pomoci javascriptu), pokud tento čas vyprší, test se uloží a ukončí. Ve spodní části menu jsou dvě tlačítka ULOŽIT a ULOŽIT A UKONČIT.

V hlavním těle testu jsou zobrazeny otázky podle pořadí a podle skupin. Každá z nich má své ohraničení pomocí tagu <fieldset> </fieldset>, kde je pro číslování otázek použité disabled tlačítko <input type="button"> s upraveným stylem v css souboru, v tagu <legend></legend> uvnitř fieldsetu. Znění otázky je zvýrazněné větším písmem definovaným jako <h3>. Pokud existuje obrázek, zobrazí se pod zněním otázky. Aby se pro zobrazení nemusely z databáze načítat dvakrát data obrázku, existence dat určitého obrázku se kontroluje podle existence záznamu o MIME typu obrázku v databázi.

Odpovědi jsou realizované jako radio button <input type="radio">, každá otázka má odpovědi svázané pomocí name vlastnosti tagu input do RadioGroup s číslem pořadí otázky. To zajisti, že půjde označit jenom jedna odpověď a rozliší to vzájemné odpovědi jednotlivých otázek. Pokud již byla otázka označena a uložena, při načtení testu se zobrazí i označení u odpovědí, které byly označeny.

4.2.1 Při nepovolení scriptů prohlížečem

Na stránce jsou také využité tagy <noscript>, které zobrazí na stránce jejich obsah, pouze když je v internetovém prohlížeči zakázán javascript. Pokud by měl uživatel zakázán javascript, nezobrazí se mu čas do konce testu, proto je v <noscript> náhradní řešení, kterým je iframe, v němž se vypisuje čas do konce s automatickým obnovováním v 5 sekundovém intervalu. Dále se vypíše nahoře v oblasti testu upozornění, že nedojde k automatickému uložení a ukončení, když vyprší čas. Také menu v levé části se nebude automaticky posouvat, proto se pod každou otázkou vypíše odkaz na posun nahoru k tlačítkům pro uložení a ukončení testu.

4.3 Uložení testu

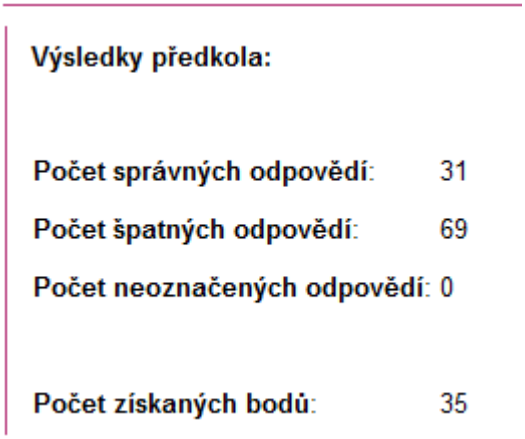
Uložení se provádí tak, že se odešlou informace z formuláře testu o tom, která odpověď u které otázky je označena. Je to jednoznačně určeno názvem a hodnotou proměnné, jež se odešle z testu. Společně s aktuální hodnotou se posílá i předešlá hodnota označení uložená do proměnné při načtení testu. Omezí se tak počet zápisů do databáze, než se zapíše informace o tom, která odpověď je označena, u které otázky se tak provede porovnání aktuálního označení s označením při načtení testu z databáze. Zbytečně tak nepřepisujeme v databázi číslo označení stejným číslem. Po uložení hodnot se zjišťuje, zdali bylo kliknuto na tlačítko uložit, pokud ano, automaticky se vrátí uživatel na stránku s testem. Pokud nebylo stlačeno tlačítko uložit, vyhodnotí se to jako konec testu. Je to bez porovnání proto, aby se na konci testu, když javascript provede submit, vyhodnotilo jako konec testu.

4.4 Vyhodnocení testu

Ještě před samotným vyhodnocením se testuje, jestli má uchazeč už dokončený test. Kontroluje se čas konce testu s aktuálním časem na serveru, aby se nemohl uchazeč podívat na výsledky dříve, než ukončí test. U obyčejné otázky se vyhodnocení provádí tak,

že se podle označené odpovědi převezme hodnota označené odpovědi (její id v databázi). Pokud nebyla žádná odpověď označena, dosadí se jako id číslo 9999, jelikož takové id v databázi u odpovědi není možné. Podle id odpovědi se provede hodnocení, zdali je otázka zodpovězena správně nebo špatně. Je-li hodnota id = 9999, otázka se vyhodnotí jako neoznačená. U otázky druhu abcde se převezme znění označené otázky, jestliže je neoznačená, dosadí se zde jako slovo "7FAS8". Problém by mohl nastat, jedině pokud by uchazeč neznačil otázku a slovo "7FAS8" by byla správná odpověď. Myslím si, že taková pravděpodobnost je velice malá, protože tento druh je přednostně určen pro otázky k obrázkům, kde nemá smysl míchat odpovědi. Vyhodnocení jako správná se provádí pomocí porovnávání řetězců funkcí "strcasecmp(String_1,String_2)". Pokud funkce strcasecmp vrátí číslo 0, znamená to, že porovnávané řetězce jsou úplně stejné a otázka je vyhodnocena jako správná. Pokud funkce strcasecmp vrátí jiné číslo než nula, porovná se odpověď se slovem "7FAS8". Pakliže funkce strcasecmp vrátí 0, vyhodnotí se otázka jako neoznačená, jinak se vyhodnotí jako špatná odpověď.

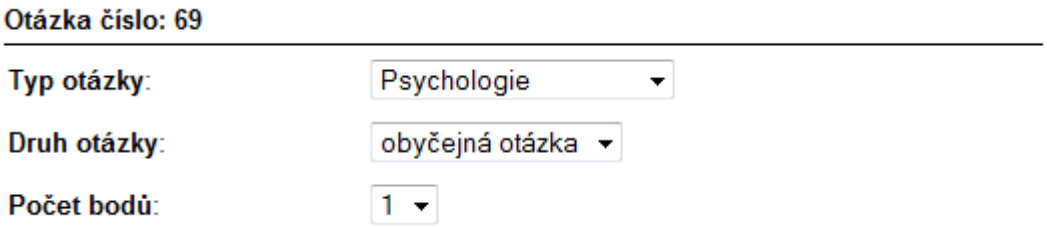
Po dokončení a vyhodnocení testu uchazeč uvidí počet otázek, které označil správně, počet otázek, které označil špatně, počet neoznačených otázek a počet bodů celkem získaných za správně zodpovězené otázky.



Obr. 9. Vyhodnocení testu

4.5 Úprava otázek

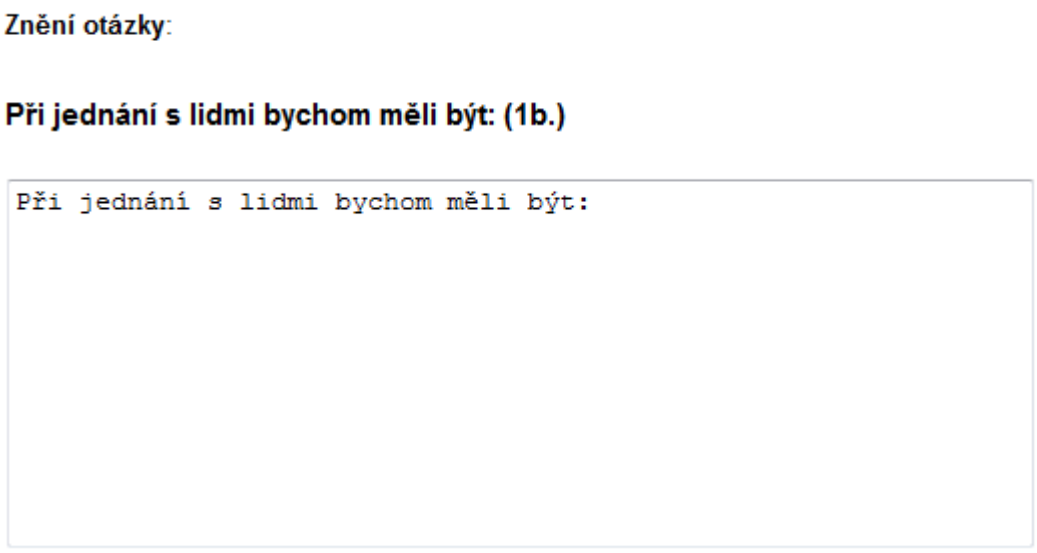
V horní části se nachází číselný seznam otázek, který funguje jako seznam odkazů na úpravu jednotlivých otázek. Pod seznamem je zobrazena vybraná otázka pro úpravu.



Obr. 10. Horní část úprav

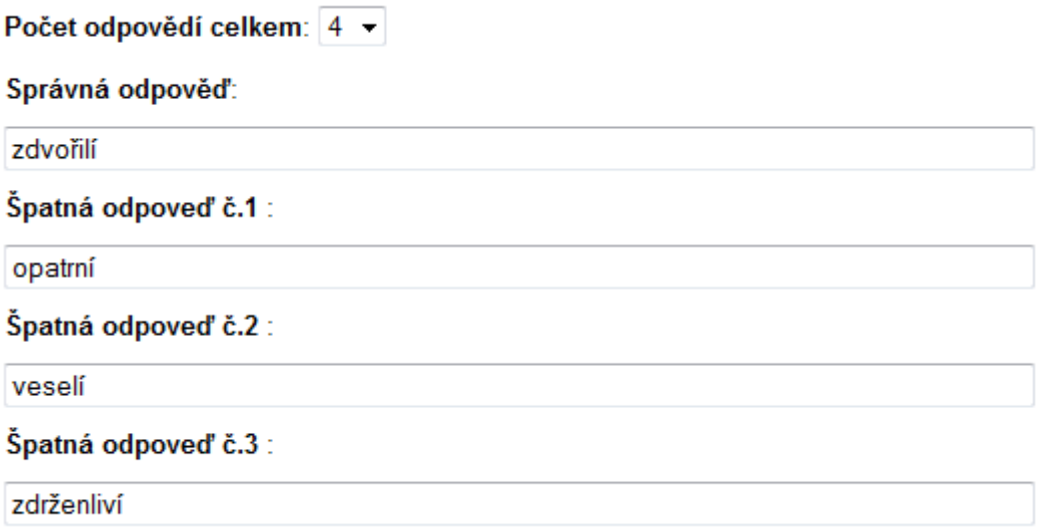
Nahoře je číslo vybrané otázky, abychom si byli jistí, kterou otázku upravujeme.

Pod čarou se nachází část s úpravou otázky. Jako první můžeme nastavit, jakého je otázka typu, výběr je pomocí seznamu, který je načtený podle typů, jaké se nachází v databázi. Další výběr je výběr druhu otázky řešený stejně jako výběr typu. Další nastavení je počet bodů, které jsou za otázku, to je realizování statickým výběrovým polem od hodnoty 1 do 5 bodů. Výběrové seznamy pomohou snížit možnou chybu uživatele i je to pohodlnější, než kdyby musel psát každou z možností ručně.



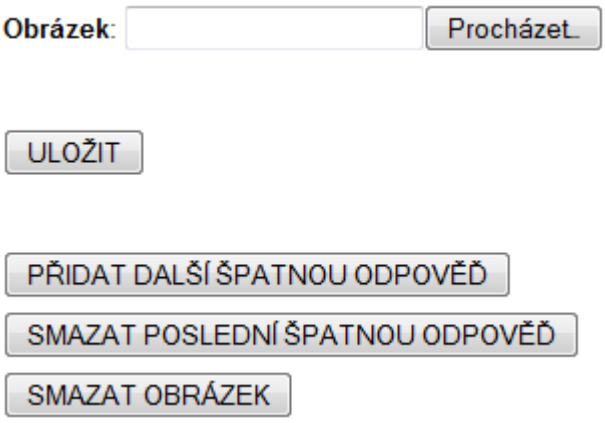
Obr. 11. Úprava znění otázky

Nahoře je zobrazena otázka tak, jak bude vypadat v testu. Je mimo textové pole, aby se daly sledovat změny textu, které můžeme provést vložením html tagů, jako je například vložení tabulky, změna barvy a velikosti písma a tak dále. Tyhle tagy by se vypsaly jenom jako text a uživatel by neviděl výsledek vytvořený pomocí html tagů. Pod vypsaným zněním otázky je textové pole pro úpravu znění otázky.



Obr. 12. Úprava odpovědí

Pod úpravou znění otázky je nastavení počtu odpovědí. Je možno nastavit 2 až 5 možných odpovědí, které se zobrazí u otázky po zamíchání ze všech odpovědí. Dále následuje řádek pro úpravu znění správné odpovědi. A dále se zobrazí řádky pro úpravu všech špatných odpovědí, seřazeny podle id odpovědi v databázi.

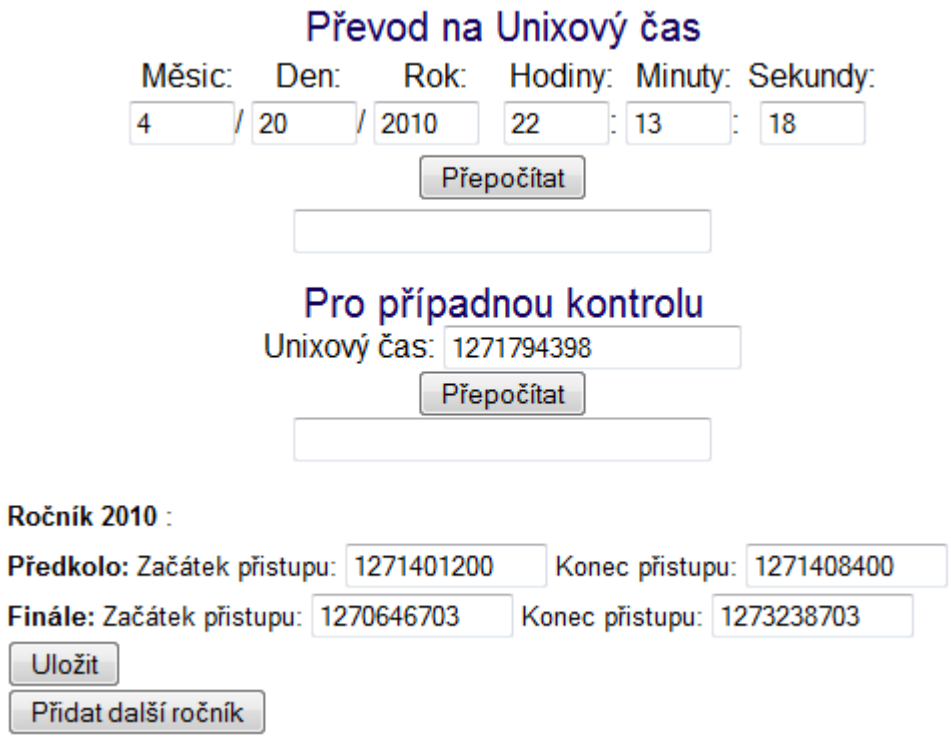


Obr. 13. Spodní část úpravy otázek

Ve spodní části úprav se nachází tlačítka pro úpravy. Jako první tlačítko je vybrání obrázku k otázce. Tímhle způsobem se obrázek odešle na server, uloží se jako dočasný soubor. Ten se zpracovává scriptem pro úpravu otázek, kde se tento soubor, který se uložil na serveru, načte binárně a doplní o potřebná lomítka, aby došlo k uložení. Například znak " by způsobil neuložení do databáze a chybu MySQL dotazu, proto je nutné znaky, které by mohly způsobit problémy, doplnit lomítkem \" pomocí funkce „mysql_escape_string()“.

Tlačítko uložit odešle všechny informace scriptu pro zpracování. Odesílají se aktuální informace a informace před úpravou, ty se spolu porovnávají, aby se zbytečně nepřepisovaly informace v databázi stejnou informací. Tlačítko přidat další špatnou odpověď odešle pouze číslo id poslední špatné odpovědi zvýšené o 1, což bude reprezentovat id odpovědi, která bude přidána. Tlačítko smazat poslední špatnou odpověď smaže odpověď s nejvyšším id ze špatných odpovědí. Poslední tlačítko smazat obrázek smaže z databáze data obrázku a informaci o jeho mime typu.

4.6 Nastavení času přístupu k testu



Obr. 14. Nastaveni začátku a konce generování

Pro určení času používám unixovou formu zápisu času. V horní časti je iframe okno, ve kterém je zobrazený převodník unixového času na normální a zase naopak. Pro funkčnost převodníku je nutné mít zapnutý javascript, bez něj se neprovede přepočet.

Pod převodníkem se nachází položky rozdělené do ročníků a ty jsou dále rozdělené na předkolo a finále. Ten nastavuje čas, po který bude generátor testu generovat testy. Test ale můžeme vyplnit i po vypršení času přístupu ke generování testu.

5 PRUBĚH TESTU PŘEDKOLA M-E O

Dalo by se říct, že test předkola selhal. Test zvládlo dokončit110 uchazečů z 364 uchazečů, kteří jej psali. Hlavní příčinou jeho nezvládnutí bylo, že garbage collector smazal 24 minut staré (neobnovené) dočasné soubory s daty sessionu na straně serveru a tím pádem nedošlo k uložení odeslaných dat formulářem testu. Informace o tom, komu daná data patří, se předávají v sessionu a pokud ten skončí, je tím pádem neznámý uživatel přesměrován na úvodní stránku. Kolega si myslel, že kdybychom jim tam napsali, ať si test ukládají každých 10-15 minut, nikdo by to nedodržel, ať raději přijdu na jiné řešení. Bohužel jsem nestihl v čas vymyslet aspoň substituci k prodloužení sessionu.

5.1 Řešení session problému

Na stránkách manuálu k php jsem našel důvod, proč nejde použít v našem případě standardní session zabudovaný v php, ani jeho jednoduchá modifikace:

"WARNING for Debian users. Just to drive you completely crazy Debian does its own form of session management and will completely ignore all alterations to the values who do within your PHP script.

Debian sets up a <crontab /etc/cron.d/php5> which deletes all files, including those in subdirectories, which exceed the gc_maxlifetime specified in the <php.ini> file only. That is, on Debian (and likely variants like Ubuntu) modifying the session expiration settings (like gc_maxlifetime) does *NOTHING*. You *HAVE* to modify the global <php.ini>. Not even a <.htaccess> file will help you." [8]

Česky zkráceně to znamená, že garbage colector pro defaultní složku, kde se ukládaji data session na serveru s debianem, se řídí pouze a jedině podle hodnoty u session.gc_maxlifetime v souboru php.ini a žádná lokální nastavení session.gc_maxlifetime nebudou fungovat, protože garbage collector promaže všechny složky a podsložky v /etc/cron.d/php5, kde jsou uloženy soubory sessionů.

Server, na kterém je umístěna M-E O je Apache/2.2.3 (Debian) a to znamená, že pro nás platí, co je napsáno výše. Bohužel nemám přístup k php.ini. Nastavovat session.gc_maxlifetime na hodinu a více pro všechny stránky, které jsou na daném serveru hostované, by asi nebylo nejlepší řešení.

Proto jsem vytvořil .htaccess soubor, který přesměroval ukládání souborů session do jiné složky, než je defaultní a nastavil jsem garbage collector, aby mazal soubory až po třech hodinách od jejich posledního načtení pomocí funkce "session_start()". Taky jsem nastavil, aby se garbage collector spouštěl s šancí 0,5% při spuštění "session_start()", podle nastavení proměnných podle vzorce :

$$\frac{gc\_propability}{gc\_divisor} * 100 = \frac{1}{200} * 100 = 0{,}5\%$$

Zdrojový kód v .htacces:

```
1 php_value session.save_path /var/www/...
```

Určení nového umístění, kam bude PHP ukládat soubory session, je však nutné mít pro tuhle složku na serveru nastavená práva pro zápis, aby zde mohlo PHP ukládat soubory sessionů.

```
2 php_value session.gc_divisor 200
```

```
3 php_value session.gc_probability 1
```

Nové nastavení pro spouštění garbage colletoru.

```
4 php_value session.gc_maxlifetime 10800
```

Nastavení maximální hodnoty času neaktivity v sekundach, po kterou si můžeme být jistí, že nám garbage collector daný session nesmaže.

5.2 Odpočet do konce

U některých uchazečů se objevil problém s tím, že jim odpočet neukazoval přesný čas do konce. To bylo způsobeno tím, že javascript získával aktuální čas z počítače, na kterém byl test psán. Divil jsem se, že při aktualizacím času z internetu se čas lišil v některých případech až o několik minut.

5.2.1 Řešení

Byla odebrána kontrola času podle času na straně uživatele, aby se předešlo nepřesnostem při posunutém času na straně uživatele. Například pokud by měl uživatel náhodou o celou hodinu více, než je čas serveru, došlo by k okamžitému ukončení testu. Funkce na odpočet času byla upravena. Při načtení stránky php dosadí do scriptu javascriptu hodnotu času konce a aktuálního času na serveru do proměnných v javascriptu. Funkce uka_cas(x) s

argumentem x přijímajícím celé číslo, který se po načení stránky každou jednu sekundu zvýší o 1 a přičte k času načtení stránky. Ale taktéž to není úplně přesné, protože se script vykonává taky určitý čas řádově v tisícinách a desetitisícinách sekundy. Tak za 50 minut, když by vykonání trvalo 0,02 sekundy, by měl na konci testu uchazeč celou minutu k dobru. Naštěstí tenhle čas je menší. Ztráta těchto nepřesnosti se koriguje pokaždé při uložení, které provede znovunačtení stránky.

6 ÚPRAVY PRO FINÁLE

Základní úprava je prodloužení času pro vyplnění testu na 120 minut a změna jednotlivých částí okruhů otázek. Dále byly přidány 2 nové druhy otázek a snížen celkový počet otázek.

6.1 Nové druhy otázek

6.1.1 Vícerozměrná otázka

Je to otázka, u níž se odpovídá v každém řádku odpovědi výběrem z více možností. Podle původního schématu by bylo možné takovou otázku zobrazit, ale už by nebylo možné ji správně vyhodnotit. Aby šlo takovou otázku jednoznačně uložit do databáze, bylo by potřeba rozšířit tabulku otázek o položku od_id2, která definuje druhý rozměr odpovědi, a o index určující, jestli je daná odpověď správná. Pro uchování jednoznačné odpovědi by bylo potřeba také vytvořit další tabulku, aby se dalo určit konkrétně, že u téhle otázky na tomhle řádku je označena tahle odpověď. Při míchání se mění pouze pořadí řádku, jednotlivé položky v řádku mají danou posloupnost podle zadání otázky.

6.1.2 Textová otázka

Je to otázka, u které se očekává odpověď v podobě textu napsaného řešitelem testu. To znamenalo, že jsme museli přidat do databáze položku, ve které se uchová řetězec, který uživatel napíše, a řetězec, se kterým se bude porovnávat. Pokud bude zadaný řetězec stejný, jako řetězec v databázi odpověď je vyhodnocena jako správná.

ZÁVĚR

Ve své bakalářské práci jsem se zaměřil na webové aplikace a jejich možnosti realizace za pomoci nejpoužívanějších webových technologií, kterými jsou XHTML, PHP, ASP, Javascript. Jako vybraná byla realizace v PHP, ale věřím, že by byla i velice zajímavá realizace v ASP, jedinou nevýhodou by však byla potřeba Windows serveru.

V praktické části jsem se zaměřil na objasnění struktury uložení dat v databázi a významu jednotlivých položek. Podrobnému vysvětlení principu generování testu a významu jeho jedinečnosti a možnostem administrátorského rozhraní pro úpravu otázek a nastavení.

Vytvořená aplikace umožňuje vygenerovat uchazeči olympiády „unikátní“ test z uložených otázek a odpovědí v databázi a vyhodnotit jej. Je možné ho také zpětně reprodukovat z dat uložených v databázi. Pro administrátory je zde rozhraní pro úpravu otázek pro test, dále změny v nastavení povolení přístupu k testu. Test je ve verzi jak pro předkolo, tak pro finále.

Na konec bych chtěl jen dodat, že jsem velice rád, že jsem se mohl podílet na vytvoření elektronické podoby Managersko-ekonomické olympiády. Naučil jsem se opravdu mnoho ohledně možností a o problémech webových technologií.

ZÁVĚR V ANGLIČTINĚ

In my bachelor, I focused on web applications and their potential using the most used Web technologies, which are XHTML, PHP, ASP, Javascript. As implementation was chosen PHP, but I believe it would also be very interesting implementation of the ASP, the only disadvantage would be a need for Windows server.

In the practical part I focused on aimed the data structure in a database and on the importance of each item. And detail explanation of the principle of generating the test and the significance of its "uniqueness". And the possibilities of admin interface for settings and possibilities of questions modification.

Created application allows candidate of the test to generate "unique" test from questions and answers stored in the database and evaluate it. It can by also re-reproduced from data stored in the database. For administrators, there is an interface to edit questions for the test, and section for changes in configuration to permit access to the test. Test is in a version for qualifying round and the finals.

At the end I just wanted to add that I am very glad that I could participate in the creation of computerized version of Managersko-ekonomická olympiáda. I learned a really lot about the possibilities and difficulties of web technologies.

SEZNAM POUŽITÉ LITERATURY

[1] PROKOPOVÁ, Z.: Databázové systémy MySQL+PHP. FAI UTB Zlín, s. 126, 2006, Vysokoškolská skripta. ISBN 80-7318-486-9.

[2] LACKO, Luboslav. SQL: Hotová řešení. Brno : Computer Press, a.s., 2007. 296 s. ISBN 80-7226-975-5.

[3] SCHNEIDER, R.,D. MySQL - Oficiální průvodce tvorbou, správou a laděním databází. Grada, ISBN: 80-247-1516-3.

[4] GUTMANS, Andi; BAKKEN, Stig Sæther; RETHANS, Derick. *PHP 5 : Power Programming*. New Jersey:Prentice Hall PTR, 2004. 720 s. ISBN 0-13-147149-X.

[5] ULLMAN, Larry. PHP a MySQL : Názorný průvodce tvorbou dynamických WWW stránek. Brno : Computer Press, a.s., 2004. 536 s. ISBN 80-251-0063-4.

[6] WELLING, L., THOMSON, L. MySQL Průvodce základy databázového systému. Computer Press, Brno, 2005, ISBN: 80-251-0671-3.

[7] HOPE, Paco; WALTHER, Ben. Web Security Testing Cookbook. 1st Edition. Sebastopol : OReilly Media, Inc., 2008. 300 s. ISBN 978-0-596-51483-9.

[8] EdA-qa at disemia dot com : WARNING for Debian users. In *User Contributed Notes : Session Functions*. 2009-05-08. [s.l.] : [s.n.], 2009 [cit. 2010-04-18]. Dostupné z WWW: <http://php.net/manual/en/ref.session.php>

[9] Wikipedia – Internetová encyklopedie [online]. [cit. 2010-04-24]. Dostupný z WWW: <http://wikipedia.org>

[10] Interval – Webdesign a e-komerce denně [online]. [cit. 2010-04-24]. Dostupný z WWW: <http://interval.cz/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	Hypertext Preprocessor (Personal Home Page)
SQL	Structured Query Language
ASP	Active Server Pages
HTML	HyperText Markup Language
XHTML	eXtensible HyperText Markup Language
JSP	Java Server Pages
WWW	World Wide Web
XML	eXtensible Markup Language

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH