Probatur – online testovanie študentov

FRANTIŠEK ŠMITALA

2007

Probatur – online testovanie študentov

BAKALÁRSKA PRÁCA

František Šmitala

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Študijný odbor: INFORMATIKA

Školiteľ bakalárskej práce:
RNDr. Richard Ostertág

BRATISLAVA 2007

Abstrakt

ŠMITALA, František: Probatur – online testovanie študentov. [bakalárska práca] – Fakulta matematiky, fyziky a informatiky Univerzity Komenského v Bratislave; Katedra informatiky. – Školiteľ: RNDr. Richard Ostertág – Bratislava 2007. (TODO) strán.

Bakalárska práca sa zaoberá návrhom, analýzou a vývojom webovskej aplikácie pre online testovanie študentov. Dôraz je kladený na časť študentského rozhrania a najmä na samotné testovanie. Zaoberá sa podrobným popisom programu a spôsobom výberu prostriedkov na jeho vývoj. K bakalárskej práci je priložená výsledná aplikácia pre online testovanie študentov.

Kľúčové slová: online testovanie študentov, PHP, SQL

Predslov

Dôležitá súčasť výuky je skúšanie. Má za úlohu overiť či si študent osvojil učivo a či ho vie aplikovať v praxi. Skúšanie má mnoho podôb. Od preukázania zručností v praxi, cez ústny pohovor až po písomné testovanie. Hlavne posledné spomenuté prežíva v poslednej dobe nevídaný rozmach. Je to hlavne kvôli tomu, že testovanie dokáže veľmi efektívne zistiť či študent danému učivu naozaj rozumie a takisto je veľmi jednoduché na organizáciu. Učiteľ šetrí čas tým, že nemusí vyskúšať každého študenta osobitne, ale stačí vykonať jedno hromadné testovanie. Problémom ale zostáva príprava testových otázok a vyhodnotenie vypracovaných testov. Hlavne druhý spomínaný problém môže trvať veľmi dlho, a preto študenti niekedy musia čakať na výsledky neprimerane dlhý čas. Tu sa vynára myšlienka zefektívniť celý tento proces a využiť informačné technológie vytvorením aplikácie, ktorá ho zautomatizuje.

Probatur – online testovanie študentov je aplikácia, ktorá umožňuje učiteľom prehľadnú správu otázok a k nim priradených odpovedí, dáva možnosť automatického generovania testov a taktiež vykoná ich opravu a vyhodnotenie. Pre študentov prináša nevídaný komfort pri vypracovaní testu, okamžité výsledky zo skúšky spolu s prehľadne opraveným testom, prehľad doterajších termínov na ktorých sa zúčastnil a termíny na ktoré je prihlásený.

Aplikácia je voľne dostupná pre každého a vzhľadom na štýl programovania sa v nej dajú robiť zmeny veľmi jednoducho a preto si ju každý môže upraviť pre svoju potrebu. Výberom použitých technológií sme zjednodušili rozšírenie, keďže všetky technológie sú voľne dostupné.

Ďakujem svojmu vedúcemu bakalárskej práce, RNDr. Richardovi Ostertágovi za cenné rady a pripomienky pri písaní tejto práce. Zároveň by som chcel poďakovať kolegom Jurajovi Porubskému a Martinovi Roškovi za bezproblémovú spoluprácu počas vývoja tejto bakalárskej práce a všetkým, ktorí majú zásluhu na jej úspešnom a včasnom ukončení.

Čestne prehlasujem, že túto bakalársku prácu som vypracoval samostatne len s použitím uvedenej literatúry.

V Bratislave 31. mája 2007 _____

Obsah

Úvod

Súčasťou tejto práce je okrem samotnej aplikácie aj opis návrhu, analýzy a implementácie tohto projektu, teda vytvorenie špecifikácie, návrhu databázy, popis zdrojového kódu a tiež sa zaoberá bezpečnosťou.

Prvá kapitola sa venuje jednotlivým etapám pri tvorbe softvéru a každú z nich rozoberá. Keďže sme na tomto projekte pracovali viacerí, tak obsahuje rozdelenie projektu na menšie celky. Obsahuje tiež špecifikáciu celého projektu a opis výberu použitých rôznych technológií, postupov a štandardov. Stručne opisuje spôsob implementácie a testovania projektu.

Druhá kapitola popisuje vzhľad a funkčnosť naprogramovaného projektu. Vzhľadom na fakt, že tento opis je z časti obsiahnutý v špecifikácii, tak sa táto kapitola venuje hlavne funkčnej časti. Najväčší priestor je venovaný najdôležitejšej časti projektu – samotnému testovaniu a vyhodnoteniu.

Tretia kapitola sa zaoberá vnútornou štruktúrou projektu, teda zdrojovým kódom. Okrem súborovej štruktúry programu sa zaoberá jednotlivými triedami a funkciami, ktoré sú implementované v tomto projekte.

V poslednej kapitole je venovaný priestor pre rozobratie fenoménu dnešnej doby – bezpečnosti systému. Opisuje základné typy útokov a spôsob ako je pred nimi aplikácia chránená.

Kapitola 1

Etapy tvorby softvéru

Pri tvorbe robustnejších systémov je dôsledné plánovanie práce neoddeliteľnou súčasťou. O to viac to platí, keď sa na tvorbe softvéru podieľajú viacerí ľudia. Tento projekt nebol výnimkou a podrobné plánovanie práce bolo jeho súčasťou. Etapy tvorby softvéru by sme mohli rozdeliť do nasledujúcich bodov:

1. zadanie úlohy
2. etapa analýzy a špecifikácie požiadaviek
3. etapa návrhu
4. etapa implementácie
5. etapa testovania
6. etapa prevádzky a údržby systému

V nasledujúcej kapitole bude podrobne opísaná každá z týchto etáp a tiež bude opísané akým spôsobom sme danú etapu vykonávali.
Pri tvorbe softvéru sú tiež dôležité role, ktoré jednotlivci zaangažovaný do projektu zastávajú. V našom projekte sa vyskytujú nasledujúce role:

1. **zadávateľ** – organizácia alebo jednotlivec, ktorý má záujem o vývoj softvérového produktu
2. **riešitelia** – samotní vývojári
3. **používatelia** – užívatelia, ktorý budú vyvinutý softvér používať

Pri väčších projektoch je týchto rôl viac, ale vzhľadom na rozsah a charakter toho nášeho by boli zbytoční (sponzor, prevádzkovateľ...).

1.1 Zadanie úlohy

Pri tvorbe softvéru má veľmi dôležité miesto zadávateľ úlohy. V našom prípade je zadávateľom jednotlivec a je ním RNDr. Richard Ostertág. Z vlastných skúseností vie, že tvorba testov pre študentov je pre učiteľa zbytočne zložitá procedúra, a preto prišiel s nasledovným zadaním projektu:

Vytvoriť web aplikáciu, ktorá umožní študentom on-line odpovedať na písomnú časť skúšky, pokiaľ je realizovateľná formou testu.

Aplikácia by okrem zozbieranie odpovedí od študentov, mala samotné testy aj generovať a hodnotiť. Generovanie by prebiehalo výberom z väčšej množiny otázok. Vytvorené testy by mali byť rovnako ťažké. Z toho vyplýva potreba systému hodnotenia obtiažnosti otázok, ktorý by mohol byť založený aj na predošlej úspešnosti študentov na tej ktorej otázke.

Výhodou pre skúšajúceho bude, že nebude musieť písomky ručne mechanicky opravovať. Výhodou pre študentov bude, že svoju úspešnosť sa dozvedia okamžite. Výhodou pre školu, že nebude treba tlačiť testy na papier.

Podrobnejšia špecifikácia na základe osobnej konzultácie (systém by mal riešiť aj generovanie opravných testov, korektnú archiváciu všetkých vypracovaných testov aj po zmene aktuálnej množiny otázok, zobrazovanie určitých štatistík, ...).

Toto zadanie sa potom stalo východiskom pre ďalšie etapy.

1.2 Etapa analýzy a špecifikácie požiadaviek

Táto jedna z časovo najnáročnejších etáp tvorby softvéru je kľúčová pre celý projekt. Ak sa totižto nepodarí pripraviť kvalitnú špecifikáciu projektu, tak je dopredu odsúdený

na neúspech. Špecifikácia musí byť dostatočne podrobná na to, aby nevznikali pri samotnej implementácii softvéru nejasnosti. Etapa analýzy a špecifikácie požiadaviek je vlastne transformácia neformálneho zadania úlohy a predstavy zadávateľa do komplexného a štruktúrovaného popisu vytváraného softvéru.
Vo všeobecnosti nemusí platiť, že zadávateľ je odborník z oblasti informačných technológií, a preto pri tejto fáze je veľmi dôležitá veľmi úzka spolupráca zadávateľa a vývojárov. V našom prípade je ale zadávateľ zároveň aj odborník v informačných technológiách, a preto mal veľmi jasnú predstavu o tom, ako má projekt vyzerať. Táto etapa prebiehala formou stretnutí všetkých zainteresovaných.
Projekt sme si rozdelili na niekoľko logických častí a každý napísal špecifikáciu o tej svojej. Juraj Porubský spracoval správu otázok a generovanie testov, Martin Roško si pripravil administrátorskú časť a ja som spracoval testovanie, hodnotenie testov, archiváciu a štatistiky. Ako sa neskôr ukázalo, uvedené rozdelenie nebolo práve najlepším riešením a pri samotnej implementácii sme si rozdelenie zmenili. Po prezentácii jednotlivých častí vždy nasledovala diskusia a upresňovali sa detaily či iné nezrovnalosti, prípadne sa dosť zásadne celá špecifikácia zmenila.

Niektoré zmeny, ktoré zásadnejším spôsobom zmenili špecifikáciu:

- V pôvodnej verzii sme chceli prihlasovanie na skúšky a vytváranie predmetov riešiť v rámci našej aplikácie. Neskôr sa to ukázalo ako zbytočné a rozhodli sme sa použiť existujúci systém na prihlasovanie na skúšky (http://felix.fmph.uniba.sk) a z neho tieto údaje importovať vopred dohodnutým protokolom.
- Oproti pôvodne zvažovanej myšlienke pribudla možnosť testy aj tlačiť, a potom po naskenovaní a automatickom vyhodnotení bolo nutné zabezpečiť import týchto dát.
- Pre každú otázku sme zamýšľali počítať po ukončení testovania zložitosť na základe odpovedí študentov, napríklad na základe toho, koľkí odpovedali správne, ale do úvahy sme chceli brať aj to, aký dobrý je študent (nesprávna odpoveď od študenta so známkou A má väčšiu váhu ako od študenta s Fx). Táto

časť bola nakoniec zjednodušená, pretože nemala také dôležité opodstatnenie (viď nasledujúci bod).

- V pôvodnej verzii sme chceli pri generovaní testov brať do úvahy aj zložitosť otázky, čo ale viedlo k exponenciálne zložitým algoritmom generovania testov.
- Zvažovali sme nad myšlienkou vizualizovať učebňu, aby mal proktor presný prehľad o tom, kto kde sedí. Ukázalo sa to ako zbytočne zložité, a nakoniec sme od tejto myšlienky upustili.

V nasledujúcej časti sa nachádza kompletná finálna špecifikácia, z ktorej sme vychádzali v ďalších fázach projektu.

1.2.1 Špecifikácia rozhrania pre študentov

Rozhranie pre študentov bude vytvorené web rozhraním cez stránku z časti prístupnej len z lokálnej siete Fakulty matematiky, fyziky a informatiky Univerzity Komenského.

1.2.1.1 Prihlásenie

Prvá stránka bude slúžiť na prihlasovanie do systému. Po prihlásení korektným menom a heslom, ktoré budú také isté ako na stránke na prihlasovanie na skúšky, sa študentovi zobrazí hlavná stránka.

1.2.1.2 Hlavná stránka

Hlavná stránka bude slúžiť na zobrazovanie informácií ako sú práve prebiehajúce skúšky, výsledky zo skúšok atď. Štruktúra bude nasledovná. Na stránke najvyššej úrovne bude výpis všetkých predmetov aktuálnych pre daného študenta. To znamená predmety, na ktoré je zapísaný. Tieto predmety budú klikacie a po kliknutí na niektorý z nich sa zobrazia bližšie informácie (časť 1.2.1.3.).

1.2.1.3 Informácie o predmete

Stránka s informáciami o skúške bude mať tieto položky. Od prednášajúceho sa tu bude nachádzať slovná poznámka k nemu, to znamená napríklad odporúčaná literatúra. Ďalej tu bude počet otázok v teste, bodovanie (za úplnú správnu odpoveď, za podmnožinu správnej odpovede, za nesprávnu odpoveď a za nezodpovedanú otázku) a samozrejme hodnotenie (stupnica) a výsledná známka (teda posledná). Bude tu tiež zoznam termínov (minulých aj budúcich). V prípade, že sa študent niektorého z tohto termínu zúčastnil, tak bude tento termín klikací a po kliknutí naň sa zobrazí stránka s bližšími informáciami o termíne (časť 1.2.1.4.). V prípade, že práve prebieha skúška z tohto predmetu, tak po kliknutí na zodpovedajúci termín sa spustí samotné testovanie. Toto však bude možné iba z lokálnej siete Fakulty matematiky, fyziky a informatiky Univerzity Komenského. Špeciálnym typom skúšky bude demoskúška, ktorá bude demonštrovať aké asi otázky z daného predmetu môžu študenti dostať a tiež bude slúžiť na oboznámenie sa s prostredím tejto webovskej aplikácie. Vytvorenie demotestu je na učiteľovi (vytvára sa iba jeden) a nie je povinný.

1.2.1.4 Informácie o termíne

Na tejto stránke si študent bude môcť prezrieť svoje výsledky, nie však podrobne aj s testom, ale iba počet bodov z každej otázky a celkový počet bodov spolu s pridelenou známkou. Okrem toho tu budú poznámky učiteľa k tomuto termínu a tiež poznámky k tomuto konkrétnemu testu študenta.

1.2.1.5 Vypracovávanie testu

Po kliknutí na odkaz pre spustenie testu sa vyberie test pre tohto študenta z dopredu vygenerovaných testov. Test bude pozostávať z dvoch častí. Z prvej časti so zoznamom otázok a druhej časti, kde sa bude zobrazovať otázka.
Aby sa zabránilo zneužitiu systému a niekto nevypracovával test pod iným menom ako svojím, tak sa po usadení študentov skontroluje ich identita na základe ISIC karty. Hneď

po usadení budú študenti vyzvaní, aby sa prihlásili. Potom bude možnosť prihlasovania zablokovaná a bude možná jedine za prítomnosti proktora. Študenti, ktorí by skončili skôr, než je skontrolovaná ich identita, musia zavolať najprv proktora a ten ju skontroluje. Pri prerušení testovania (WC...) sa študenti neodhlasujú, ale budú mať možnosť zablokovať si vypracovávaný test. V prípade väčšieho výpadku sa bude môcť prihlasovanie opäť povoliť a bude opäť skontrolovaná identita študentov.

1.2.1.5.1 Zoznam otázok

V prvej časti sa bude nachádzať zoznam všetkých otázok farebne rozlíšených (význam farebného rozlíšenia v časti 1.2.1.5.2). Kliknutím na niektorú z otázok (budú zobrazené ako číslo otázky a začiatok z otázky kvôli úspore miesta, ale zároveň aby si študent vedel spomenúť čo ktorá otázka aspoň zhruba obsahuje) sa v druhej časti táto otázka zobrazí celá. Kliknutím na tlačidlo "odoslať test" sa test odošle a už nebude možná jeho ďalšia úprava. Kliknutím na tlačidlo "locknúť test" sa test zablokuje a bude sa čakať na meno heslo prihláseného používateľa.

1.2.1.5.2 Zobrazenie otázky

V druhej časti sa nachádza plné znenie testovej otázky spolu s možnými zaklikávacími odpoveďami typu a,b,c... Ďalej tu bude textové pole určené na poznámku, ktoré sa bude môcť využiť napríklad na komentár k nejednoznačnej otázke a podobne. Tlačidlom "ulož" sa čiastkové odpovede spolu s komentárom uložia na server.

V tejto časti bude aj základná navigácia (predchádzajúca a nasledujúca otázka) a možnosť "vyfarbiť" otázku, teda prideliť jej akýsi status. Tento bude napomáhať študentovi označiť si nejakou farbou otázky, ktoré už má vypracované, alebo tie, ktoré má vypracované len z časti, tie ktoré vôbec nevie a podobne. Toto bude čisto interná záležitosť študenta, pretože tieto statusy sa nebudú brať do úvahy ani pri hodnotení, ani nikde inde. Samozrejmosťou bude časomiera, ktorá bude ukazovať zostávajúci čas a čas strávený nad danou otázkou. Periodicky bude kontrolované či nevypršal čas alebo či test už nebol ukončený. V prípade, že bude nutné rozpracovaný test znova načítať, bude sa to robiť tým istým spôsobom ako spúšťanie testu, len na serveri už budú uložené medzivýsledky, ktoré sa vezmú do úvahy.

1.2.1.6 Hodnotenie testu

Po kliknutí na odoslať v sekcii vypracovávania testu sa test začne okamžite hodnotiť. Všetky kritéria hodnotenia nastavuje skúšajúci pri generovaní testu.To znamená, že program už len prečíta parametre uložené v databáze a na základe nich test ohodnotí. Každú otázku zaradí do kategórie (správne zodpovedaná otázka, podmnožina správnych odpovedí, nesprávna odpoveď, nezodpovedaná otázka) a na základe tejto kategórie určí počet bodov. Suma týchto bodov určuje celkový dosiahnutý počet bodov.
Po odoslaní testu sa študentom zobrazí hodnotenie a opravený test, teda budú vidieť celý test len na čítanie s vyznačenými študentovými odpoveďami a s označenými správnymi odpoveďami.
Študent bude potom presmerovaný späť na stránku s informáciami o predmete (časť 1.2.1.3.), kde pribudne už výsledok so známkou. Pre neho bude daný termín už uzavretý, a preto tam možnosť na spustenie testu už nebude. Týmto pre študenta vlastne termín skončil, prípadne môže prísť na ďalší termín. Dodatočné informácie o tejto skúške, ako napríklad priradenie známky, rôzne úpravy zo strany skúšajúceho (penalizácia za podvádzanie, uznanie viacerých možných správnych odpovedí a podobne) nájde na už spomínanej stránke s informáciami o predmete.

Poznámky

Prihlasovanie na termín bude prebiehať externou aplikáciou systému http://felix.fmph.uniba.sk. Z nej sa potom budú sťahovať dáta dohodnutým protokolom (napríklad vo formáte XML).
Test je možné vypracovávať aj v papierovej podobe. V tom prípade, je rozhranie pre študentov viac-menej zbytočné. Študenti si tam môžu len pozrieť výsledky.

1.2.2 Špecifikácia rozhrania pre učiteľov

Rozhranie pre skúšajúceho učiteľa je tiež riešené web aplikáciou. Prístup k nemu je z ľubovoľného miesta pripojeného na sieť internet. Hlavným cieľom rozhrania je ponúknuť učiteľovi vhodné prostredie, v ktorom je schopný pripraviť pre študentov

testy, otestovať ich a následne oznámkovať. Rozhranie je však komplexnejšie, učiteľovi ponúka aj rôzne nadštandardné možnosti.
V rozhraní sa rozlišujú 2 role: učiteľ a proktor. Keďže proktori sú osoby dozerajúce na správny priebeh skúšky, prístup je im umožnený iba do sekcie testovania.

1.2.2.1 Prihlasovanie

Prvá stránka slúži na prihlasovanie do systému. Prihlasovacie údaje sú odosielané systému, ktorý overí ich správnosť. Po úspešnom prihlásení sa skúšajúci dostane na stránku informácie, kde sa mu zobrazia všeobecné informácie o predmetoch.

1.2.2.2 Stránky nezávislé od predmetu

1.2.2.2.1 Stránka informácie

Informačná stránka má ponúknuť prihlásenému užívateľovi informácie o urgentných udalostiach v predmetoch, ktoré sa ho týkajú. Do urgentných udalostí spadá zoznam práve prebiehajúcich skúšok a zoznam budúcich skúšok. Ku každej práve prebiehajúcej skúške sa zobrazí odkaz na pokračovanie testovania, ktoré prepne užívateľa do testovacieho režimu. V zozname budúcich skúšok je to podobné. Každá budúca skúška obsahuje možnosť spustenia testovania.

1.2.2.2.2 Stránka predmety

Táto sekcia zobrazuje učiteľom zoznam predmetov, v ktorých vyučujú. Ku každému predmetu sa zobrazí odkaz [detaily] na detailnú správu predmetu.

1.2.2.2.3 Stránka proktori

Stránka proktori sa delí na dve časti:

- *zoznam proktorov* obsahujúci mená všetkých osôb, ktoré môžu na predmetoch, ktorých je prihlásený učiteľ učiteľom, vykonávať funkciu proktora.
- *pridávanie proktorov* obsahuje zoznam osôb v systéme, ktoré môžu byť na funkciu proktora dosadené. V tomto zozname sa zobrazia mená osôb, ktoré

vystupujú v systéme ako učitelia alebo ako proktori, ktorých si prihlásený učiteľ ešte do zoznamu proktorov nepridal.

1.2.2.2.4 Stránka zmena hesla

Formulár na zmenu hesla obsahuje tri vyplňovacie položky:

- *staré heslo*, potrebné na overenie totožnosti užívateľa
- *nové heslo*
- *zopakovanie nového hesla* znižujúce pravdepodobnosť preklepu pri zadávaní nového hesla

Pokiaľ sa totožnosť užívateľa overí a nové heslo sa zhoduje so zopakovaným heslom, nové heslo sa pre užívateľa uloží v databáze.

1.2.2.3 Stránky závislé od predmetu

1.2.2.3.1 Stránka info o predmete

Všeobecné nastavenia predmetu sa uskutočňujú práve v tejto sekcii. Dovoľujú učiteľovi nastaviť popis k predmetu a dĺžku trvania skúšok. Popis k predmetu slúži na to, aby študenti získali bližšie informácie o predmete, ako napr. o cieli predmetu, spôsobe výučby, či podmienkach úspešného absolvovania predmetu. Dĺžka trvania skúšok sa nastavuje pre celý predmet jednotne, lebo sa predpokladá objektívnosť podmienok pre študentov.

1.2.2.3.2 Časť testov

Najdôležitejšou časťou učiteľského rozhrania je práve časť týkajúca sa testov, ktorá má učiteľovi umožniť testy generovať, spravovať ich a nastavovať parametre potrebné pre časť testovania. Táto časť zahŕňa v sebe štyri stránky: h*odnotenie, generovanie, demo test* a *importovanie*.

1.2.2.3.2.1 Stránka hodnotenie

Hlavným cieľom nášho systému ponúknuť učiteľovi aj študentom rýchle spracovanie testov. Aby program po vypracovaní testu vedel určiť akú známku má dať študentovi,

musí učiteľ zadať stupnicu známok a systém bodovania testových otázok. Bodovanie otázok sa rozdeľuje do štyroch skupín:

- *celá správna odpoveď* – študent označil všetky správne odpovede a žiadne iné
- *čiastočná odpoveď* – študent označil neprázdnu podmnožinu správnych odpovedí
- *nezodpovedaná otázka* – študent v otázke neoznačil ani jednu odpoveď
- *zlá odpoveď* – študent označil aspoň jednu odpoveď nepatriacu do správnych odpovedí

Učiteľ jednotlivým skupinám nastavuje počet bodov, ktoré študent získa za každú odpoveď patriacu do danej skupiny. Poradie vymenovania skupín určuje aj prioritu zaradenia odpovede do skupiny.

1.2.2.3.2.2 Stránka generovanie

V stránke generovanie prebieha samotné generovanie testov a skladá sa z dvoch častí: *generovanie* a *nastavenie generovania*.

Nastavenie generovania:

Dovoľuje učiteľovi nastaviť spôsob generovania testov. Základnými parametrami sú: počet otázok a hodnoty *min* a *max* pre každý okruh tém z predmetu. Hodnota *min* pri okruhu tém hovorí, koľko otázok minimálne sa má z daného okruhu alebo z jeho podokruhov vybrať. Hodnota *max*, naopak, určuje maximálny počet otázok. Okruhy tém sú vypísané hierarchicky, aby učiteľovi ponúkli prehľad o ich štruktúre.

Generovanie:

Táto časť má za úlohu umožniť učiteľovi mazať už vygenerované testy a dovoliť mu nové testy generovať. Keďže testy môžu byť viacerých typov – papierové a vypracovávané pri počítači – správa testov je rozdelená do dvoch samostatných častí. Prvá týkajúca sa testov pri počítači, druhá papierových testov. Ku každému typu testov je uvedený počet už vygenerovaných testov, tieto testy má učiteľ možnosť všetky vymazať. Ďalej nasleduje možnosť nového generovania. Učiteľ zadá počet testov, ktoré sa majú vygenerovať a kliknutím na tlačidlo *generuj* sa samotné generovanie na základe nastavení spustí. Pri testoch vypracovávaných pri počítači je v tejto podsekcii zobrazená

ešte jedna voľba, a to generovať demo test. Bližšie informácie o demo teste sa nachádzajú v časti 3.2.3.

1.2.2.3.2.3 Stránka demo test

Aby mal študent pred skúškou možnosť zoznámiť sa so štýlom a prostredím testovania, učiteľ má možnosť vytvoriť jednoduchý demo test, ktorý má na tento účel slúžiť. Výsledky demo testov sa v databáze dlhodobo neukladajú, preto ich výsledok nijako nemôže ovplyvniť študentovu známku. Generovanie demo testu prebieha v stránke generovanie, v tejto stránke sa vygenerovaný demo test učiteľovi iba zobrazí.

1.2.2.3.2.4 Stránka importovanie

V stránke generovanie pri voľbe generovania papierových testov sa vytvorené testy zapíšu v PDF dokumente, ktorý si učiteľ neskôr môže vytlačiť. V hlavičke testu sa nachádza názov predmetu, ku ktorému patrí a jednoznačný identifikátor testu. V tele strany sa nachádza zoznam otázok a odpovedí k nim. Študent si pri vypracovávaní testu opíše identifikátor testu, zaznačí svoje ISIC číslo slúžiace na jeho identifikáciu a ďalej zaznačí odpovede, ktoré si myslí, že sú správne. Učiteľ, na základe študentami vypracovaných testov, vytvorí XML dokument reprezentujúci zoznam testov. Štruktúra XML dokumentu je daná DTD dokumentom '/dtd/tests.dtd'. Pri importe údajov sa zadáva termín skúšky, na ktorom sa import prevádza, a kontroluje sa, či identifikátor testu určuje test, ktorý bol vytvorený pre predmet patriaceho učiteľovi. Pokiaľ sú všetky údaje platné, vypracovanie testov sa zaznačí do databázy a ku každému testu sa prepočítajú body, ktoré študent získal. Nakoniec sa určí výsledná známka.

1.2.2.3.3 Stránka termíny skúšok

Testovanie prebieha na skúškach, ktorých termíny sa importujú zo systému Félix používaného fakultou matematiky, fyziky a informatiky Univerzity Komenského. Keďže bol náš systém navrhovaný na spoluprácu so spomínaným systémom, vytváranie termínov skúšok nie je v našom systéme obsiahnuté. Stránka termíny skúšok obsahuje tri časti: *zoznam termínov skúšok, zoznam študentov na termíne, pridávanie študentov na termín*.

Zoznam termínov skúšok:

Obsahuje zoznam termínov skúšok z aktuálne zvoleného predmetu. Tento zoznam sa dá jednoduchým filtrom upravovať. Filter dovoľuje učiteľovi nastaviť dátum, od ktorého sa majú termíny skúšok zobraziť.

Zoznam študentov na termíne:

Zobrazí sa zoznam prihlásených študentov na zvolenom termíne, utriedený vzostupne podľa mena. Pokiaľ je skúška typu budúca, je učiteľovi dovolené upravovať zoznam študentov. Pridávanie študentov na termín sa deje v nasledujúcej časti.

Pridávanie študentov na termín:

Učiteľovi sa zobrazí podľa mena utriedený zoznam študentov, ktorí majú daný predmet zapísaný a zároveň nie sú zapísaní na zvolenom termíne. Kliknutím na odkaz [pridať] nachádzajúci sa vedľa mena študenta sa príslušný študent pridá na termín.

1.2.2.3.4 Stránka databáza otázok

Databáza otázok dovoľuje učiteľovi vytvárať, editovať a mazať otázky, z ktorých sa neskôr bude test generovať. Aby sa učiteľovi uľahčilo a sprehľadnilo spravovanie otázok, bolo potrebné zaviesť ďalšie štruktúry, na základe ktorých sú otázky členené, a to: *okruhy tém*, *otázky* a *sady odpovedí*. S každou sa dajú vykonávať tri akcie: *vytvoriť, editovať a vymazať*. Pri spravovaní databázy otázok je dôležité dávať si pozor na archiváciu. To znamená, že systém nedovolí učiteľovi vymazať z databázy otázku, ktorá už bola v teste použitá. Vtedy ju iba označí za zmazanú.

1.2.2.3.5 Stránka vyhľadávanie

Umožňuje vyhľadávanie objektov na základe určených kritérií. Vyhľadávací formulár sa skladá z troch položiek. Prvá určuje hľadaný text. Pokiaľ je prázdna, zobrazí všetky hľadané objekty, inak zobrazí iba tie, ktoré obsahujú hľadané slovo. Druhá položka udáva typ objektu, v ktorých sa má vyhľadávať. Na výber sú štyri možnosti: *sady odpovedí, otázky, okruhy tém, študenti*. Posledná, tretia, položka definuje typ triedenia. Triedenie môže byť: *vzostupné* a *zostupné*.

1.2.2.3.6 Stránka štatistika

Sekcia štatistika má za úlohu informovať učiteľa o rôznych štatistických údajoch vzťahujúcich sa k predmetu. Tieto údaje majú pomôcť učiteľovi odhaliť zlé nastavenia predmetu, akými napr. môžu byť príliš ľahké alebo príliš ťažké otázky v testoch. Štatistické údaje sa dajú získavať z rôznych uhľov pohľadu. My sme sa však snažili zahrnúť sem tie, ktoré sa nám zdali byť pre učiteľa najužitočnejšie. Stránka štatistika sa delí na dve časti: š*tatistika podľa úspešnosti, štatistika zastúpenia známok*.

Štatistika podľa úspešnosti:

Zobrazuje objekty a k nim prislúchajúci priemerný počet bodov získaný v testoch. Pred zobrazením tohto typu štatistiky je ešte potrebné upresniť svoje požiadavky vypľňovacím formulárom, ktorý obsahuje štyri položky:

- Aký objekt je predmetom štatistického skúmania. Možnosťami sú: *sady odpovedí, otázky, okruhy tém, študenti, termíny.*
- Dátum určujúci obdobie, od ktorého sa zohľadňujú štatistické údaje.
- Dátum určujúci obdobie, do ktorého sa zohľadňujú štatistické údaje.
- Spôsob triedenia, ktorý výstup štatistiky triedi podľa priemerného počtu bodov. Na výber sú dve možnosti: *vzostupne* a *zostupne*.

Štatistika zastúpenia známok:

Zobrazuje tabuľku známok a k nim priradený počet testov, ktoré boli príslušnou známkou ohodnotené. Výber testov, ktoré sa majú brať do úvahy, je nastaviteľný podľa termínu skúšky. Učiteľ má aj možnosť zobraziť zastúpenie známok všetkých testov, tzn. bez ohľadu na termín. Zoznam všetkých termínov skúšok, patriacich predmetu, je zobrazený vo vyplňovacom formulári.

1.2.2.3.6 Testovanie

Táto časť bude obsahovať zoznam všetkých prihlásených študentov spolu s časom stráveným vypracovávaním testu. Bude tu možnosť pridať čas (resp. ubrať v prípade zápornej hodnoty) a pri každom mene bude checkbox, ktorý určí, či sa akcia prevedie aj na danom študentovi. Taktiež tu bude možnosť uzavrieť termín, čo sa prejaví tak, že už žiadny študent nebude môcť spustiť test v tomto termíne (toto sa dá potom ešte

odblokovať – ošetrenie prípadov ako neskorý príchod... a tak ďalej) a "ukončenie testovania", čo má za následok ukončenie termínu a vynulovanie časomier všetkých prihlásených. To spôsobí, že študenti už nebudú môcť upravovať test. Po ukončení testovania bude učiteľ (proktor) presmerovaný na hlavnú stránku.

1.2.3 Špecifikácia rozhrania pre administrátora

Vzhľadom na predpokladané veľké množstvo užívateľov, musí byť rozhranie pre správcu systému prehľadné a navigácia by mala byť intuitívna. Jeho súčasťou musia byť aj nástroje pre zmenu nastavení systému, napríklad možnosť editovať prístupové údaje k vzdialenému serveru, poskytujúcemu dáta na import alebo zmena hesiel jednotlivých užívateľov, vrátane hesla samotného administrátora.

1.2.3.1 Prihlasovacia stránka

Administrátorské rozhranie je určené výlučne pre správcu systému, preto musí byť prístup k nemu vybavený prihlasovacím formulárom, ktorý bude od žiadateľa pre vstup do tejto časti systému žiadať prihlasovacie meno a heslo. Vstup bude umožnený iba užívateľom, ktorí správne vyplnia tieto údaje a následne odošlú požiadavku stlačením tlačítka pre prihlásenie, po ktorom systém overí správnosť údajov. Ak boli vyplnené správne, presmeruje administrátora na hlavnú stránku administrátorského rozhrania, v opačnom prípade zobrazí opäť prázdny prihlasovací formulár.

1.2.3.2 Študenti

Študenti a učitelia sa prihlasujú do systému pre online testovanie cez študentské, respektíve učiteľské rozhranie. Pre správne fungovanie však musia byť vytvorené a každému užívateľovi pridelené kontá, tak aby bolo možné, po prihlásení sa študenta alebo učiteľa, ho jednoznačne identifikovať v systéme. Každý užívateľ musí mať pridelené jednoznačné prihlasovacie meno a heslo. Študent má pridelené predmety, ktoré má v danom roku zapísané z ktorých môže byť skúšaný. Úlohou administrátora je zabezpečiť, aby sa každý užívateľ dokázal do systému prihlásiť a aby mal pridelené všetky predmety, ktoré pri svojej práci potrebuje. Aby bola práca administrátora

efektívna, musí byť sekcia so študentami a učiteľmi prehľadná, jednoduchá na ovládanie a intuitívna.

V sekciách študentov a učiteľov, ktoré budú od seba oddelené, bude hlavným navigačným prvkom zoznam užívateľov s akciami, ktoré sa na nich dajú vykonať. Pod akciami sa ruzmiepridávanie, editovanie, zmena hesla a mazanie užívateľov. Súčasťou týchto sekcií bude vyhľadávací formulár a formulár pre pridávanie užívateľov do systému.

1.2.3.3 Odbory

Štúdijné odbory budú v systéme vytvorené pre zvýšenie prehľadnosti a zefektívnenie práce so študentskými kontami. Každý študent môže patriť pod maximálne jeden štúdijný odbor. Odbory majú pridelených svojich garantov, ktorými môžu byť iba učitelia, nachádzajúci sa v systéme. Je potrebné vytvoriť formulár pre pridávanie a editovanie štúdijných odborov.

1.2.3.4 Predmety

V študentskom rozhraní systému sa študentom po prihlásení zobrazí zoznam predmetov, z ktorých môžu byť testovaní v priebehu roka. Učiteľské rozhranie poskytuje učiteľom možnosti vytvárania sád otázok a generovanie testov pre jednotlivé predmety. Preto spolu so študentskými a učiteľskými kontami v systéme, je veľmi dôležitá aj správa predmetov. Znamená to, že každý predmet, ktorý má byť využívaný v systéme pre online testovanie, musí mať prideleného učiteľa a zoznam študentov, ktorý z neho môžu byť skúšaný.

1.2.3.5 Import

Pridávanie všetkých dát do systému manuálne by zabralo obrovské množstvo času a zvýšila by sa pravdepodobnosť chýb pri zadávaní údajov. Preto musí byť systém pre online testovanie študentov navrhnutý tak, aby dokázal komunikovať a využívať služby už existujúceho systému pre prihlasovanie sa študentov na skúšky alebo ľubovoľného

iného systému, ktorý dokáže komunikovať so systémom pre testovanie študentov pomocou jednoduchého protokolu.

1.2.3.6 Záloha

Pri návrhu systému, je dôležité myslieť dopredu, teda, ak sa bude využívať dlhšiu dobu, postupne sa naplní údajmi, ktoré sa stanú medzičasom nepotrebné, alebo budú dôležité iba kvôli štatistikám. Sem patria údaje o študentoch a skúškach a údaje, ktoré sú na ne priamo naviazané, napríklad testy a odpovede na ne. V tejto sekcií bude možné vykonať akcie, ktoré zabezpečia zálohu dát, načítavanie zálohy do databázy a jej odstránenie.

1.2.3.7 Nastavenia

Kvôli bezpečnosti musí byť v systéme možnosť zmeny prístupového hesla do administrátorského rozhrania. Používanie rovnakého hesla počas dlhšej doby by dalo potenciálnemu útočníkovi väčšiu šancu pre jeho získanie a zároveň, ak by nebola v systéme možnosť zmeny hesla, už raz útočníkom získané heslo by nebolo možné zmeniť. V sekcii s nastaveniami sa preto okrem nastavení údajov potrebných pre komunikáciu s externým systémom poskytujúcim službu importu dát, musí nachádzať aj formulár pre zmenu prístupového hesla do administrátorskej časti.

1.3 Etapa návrhu

V predchádzajúcej etape sme určovali, že čo má softvér robiť. V tejto etape sme určovali ako budú stanované ciele realizované. Súčasťou je už finálne rozdelenie projektu na 3 logické celky a každý z nás jeden takýto celok bude vypracovávať. Rozdelili sme si to nasledovne. Martinovi Roškovi zostala jeho pôvodná administrátorská časť, Juraj Porubský mal na starosti rozhranie pre študentov a ja som spracovával rozhranie pre študentov. Z veľkej časti sa táto etapa kryla s vytváraním špecifikácie, pretože už počas tejto etapy sme mysleli aj na implementačné detaily. V tejto etape sme vyberali aj vývojové prostredie, programovací jazyk a vyberali sme, že ktoré technológie budeme využívať a samozrejme sem patrí aj návrh databázy.

1.3.1 Vývojové prostredie

Keďže všetci traja vývojári pracujeme prevažne pod operačným systémom Microsoft Windows, tak voľba operačného systému bola jednoznačná. Textový editor si už každý vyberal sám za seba. Ja som sa rozhodol pre editor PSPad, s ktorým mám už bohaté skúsenosti. Jedná sa o jednoduchý, ale efektívny editor s prehľadným a intuitívnym ovládaním. Je to program dostupný voľne k stiahnutiu na http://www.pspad.com a je zadarmo.

1.3.2 Programovací jazyk

Vzhľadom na to, že náš program je webovskou aplikáciou, núkali sa nám 2 najschodnejšie riešenia výberu programovacieho jazyka. Prvou alternatívou bolo PHP a druhou ASP.NET. Vzhľadom na fakt, že nikto z nás nemal s ASP.NET vážnejšie skúsenosti, tak sme sa rozhodli pre PHP.

PHP je jeden z najrozšírenejších skriptovacích programovacích jazykov používaných na tvorbu webovských aplikácii. Sú v ňom vytvorené projekty typu MediaWiki, fórum phpBB a mnoho iných. Podporuje objektovo orientované programovanie, čo je pre nás kľúčové, vzhľadom na to, že celý projekt chceme robiť pomocou objektov.

Na prezentačnú vrstvu sme sa rozhodli použiť valídny striktný štandard XHTML 1.0 s použitím valídnych kaskádových štýlov CSS. Výhody používania valídneho striktného módu sú predovšetkým v tom, že stránka sa zobrazí v každom prehliadači dodržiavajúceho štandardy (W3C – World Wide Web Consortium, ktoré produkuje slobodné štandardy pre World Wide Web) rovnako.

Aj na prezentačnej úrovni sme sa snažili oddeliť layout od dizajnu, a preto sú všetky formátovania robené cez externé CSS. Toto prináša radu výhod od jednoduchšieho vykonávania zmien v dizajne až po prehľadnejší HTML kód.

1.3.3 Výber ostatných technológií a postupov

Okrem už spomenutých prostriedkov sme sa rozhodovali aké technológie ešte využijeme. Kvôli prehľadnosti a oddeleniu prezentačnej vrstvy od logickej sme sa rozhodli použiť šablónovací systém Smarty. V niektorých častiach programu bolo nutné použitie JavaScriptu a AJAXu. Na tvorbu dizajnu sme sa rozhodovali medzi tabuľkovým layoutom a layoutom vytvoreným pomocou HTML tagov "*div*". Aj keď som z viacerých dôvodov zástanca druhej alternatívy, rozhodli sme sa pre čiastočne tabuľkový dizajn, hlavne z dôvodu už predprogramovaného layoutu z už existujúcej stránky systému na prihlasovanie na skúšky.

1.3.3.1 Smarty

Výhody oddelenia HTML kódu od PHP kódu sú nesporné. Od prehľadnosti, cez rýchlosť vykonávania zmien až po množstvo užitočných funkcií naimplementovaných do Smarty. Šablóny fungujú tak, že sa z PHP kódu vyvolá šablóna, ktorá sa má zobraziť, pred tým sa pripravia hodnoty (napríklad z databázy), ktoré budú tejto šablóne odoslané a Smarty parser nahradí výskyty všetkých premenných v šablóne zodpovedajúcou hodnotou. V Smarty je okrem toho naimplementovaných množstvo užitočných funkcií, napríklad na prácu s regulárnymi výrazmi, *while* či *foreach* cykly (umožňujú veľmi elegantne spracovávať premenné obsahujúce polia), vetvenie (*if / elseif / else*), vlastné funkcie či takzvané modifikátory, ktoré umožňujú formátovať premenné (napríklad výraz *{$premenna|escape}* znamená, že na toto miesto vypíš obsah premennej "*premenna*" a preveď všetky špeciálne znaky na HTML entity).

1.3.3.2 JavaScript

JavaScript je multiplatformový objektovo orientovaný skriptovací jazyk veľmi rozšírený najmä ako skriptovací jazyk vkladaný do HTML kódu a vykonávaný na strane klienta. V našom projekte sme JavaScript využili pri dynamicky sa meniacich súčastiach stránky (odpočítavadlo), či ako kontrola nechceného kliknutia na dôležité tlačidlo.

1.3.3.3 AJAX

AJAX (*Asynchronous JavaScript and XML*) je označenie technológie vývoja interaktívnych webovských aplikácií, ktoré menia svoj obsah bez nutnosti znovunačítania celej stránky. Využíva nasledovné technológie:

- **HTML** a **CSS** ako prezentačná vrstva
- **DOM** a **JavaScript** na zobrazenie a dynamické zmeny prezentovaných informácií
- **XMLHttpRequest** na výmenu dát s webovým serverom (vo formáte XML, plain text, ...)

V našom projekte sme AJAX využili napríklad pri posielaní pravidelných pingov (s hodnotou timestampu – časovej pečiatky) na server, aby sa v prípade havárie systému zachovala informácia o tom, že kedy výpadok nastal a tým pádom mohlo prerušené testovanie plynule pokračovať.

1.3.4 Návrh databázy

Návrh databázy mal na starosti Martin Roško a počas našich stretnutí sme tento návrh konzultovali a upravovali. Výsledkom návrhu bol UML model databázy, na ktorom bolo prehľadne vidieť jednotlivé prepojenia (primárne a cudzie kľúče) medzi jednotlivými tabuľkami. Výsledná štruktúra zahŕňa tabuľky týkajúce sa predmetov, užívateľov (študentov, učiteľov, proktorov a administrátorov) a nakoniec tabuľky vzťahujúce sa na skúšky a testy. V nasledujúcej časti stručne opíšem štruktúru databázy, jednotlivé tabuľky a ich vzájomné prepojenia.

1.3.4.1 Tabuľky

Nasledujúca časť obsahuje výpis všetkých tabuliek v databáze. Pred samotným výpisom treba ešte povedať, že v databáze sa nachádza niekoľko tabuliek, ktoré obsahujú špeciálne pole *obsolete.* Toto určuje či je daný záznam aktuálny. Na vysvetlenie, napríklad v prípade, že učiteľ zmaže nejakú otázku kvôli tomu, že ju už nechce do testov dávať, nie je možné tento záznam len tak vymazať, pretože potom by už nebolo

možné dostať sa k presnému zadaniu testu študenta, ktorý takúto otázku už dostal. Práve toto pole určuje aktuálnosť záznamu. Záznam nebude vymazaný, len označený ako vymazaný.

Výpis tabuliek:

tUser

V tejto tabuľke sú uložení všetci užívatelia systému. Obsahuje informácie o mene, akademickom titule, prihlasovacom mene a hesle a čísle ISIC/ITIC karty.

tStudents, tTeachers, tAdmins, tProctors

Tieto tabuľky umožňujú diferenciáciu jednotlivých užívateľov. Obsahujú prepojenie na užívateľa (*tUser*) a tStudents obsahuje ešte prepojenie na špecializáciu (*tSpecialization*) a informáciu o roku štúdia.

tCourseToUser

Ide vlastne o prepojenie many-to-many medzi tabuľkami predmetu (*tCourse*) a užívateľom (*tUser*).

tProctorToTeacher

Počas testovania ako dozor nemusí byť učiteľ osobne, ale osoba ním poverená. Aby sa táto mohla prihlásiť do systému a riadiť priebeh skúšky, na toto slúži táto tabuľka. Ide vlastne o prepojenie many-to-many medzi učiteľmi a proktormi (obe *tUser*).

tSpecialization

Študenti sú zaradený do študijných odborov. Táto tabuľka obsahuje názov študijného odboru a tiež prepojenie na svojho garanta (*tUser*).

tCourse

Táto jedna z najdôležitejších tabuliek obsahuje informácie o predmete ako napríklad názov, popis, počet otázok, dĺžka trvania testov tohto predmetu (kvôli objektívnosti sa táto vlastnosť viaže celkovo na predmet a nie na jednotlivé skúšky) a tiež prepojenie na stupnicu hodnotenia (*tScale*) a prepojenie na template demotestu (k tomuto sa dostaneme neskôr).

tScale

Táto tabuľka okrem spodných hraníc pre každú známku obsahuje a spôsob hodnotenia pre každý typ odpovede. Poznáme 4 základné typy odpovede: správne odpovede, podmnožinu správnych odpovedí, nesprávne odpovede a otázky nezodpovedané.

tDomain

Každá otázka je zaradená do okruhu tém. Práve táto tabuľka si tieto okruhy pamätá a vzhľadom na to, že obsahuje cudzí kľúč sama na seba, tak sú tieto okruhy štruktúrovateľné. Takisto obsahuje prepojenie na predmet, keďže každý okruh je viazaný na práve jeden predmet. Ďalej obsahuje informácie typu názov okruhu a popis okruhu.

tCourseToDomain

Táto tabuľka prepája predmet (*tCourse*) a okruh tém (*tDomain*) a pamätá si nastavenia dôležité pri generovaní testu a to minimálny a maximálny počet otázok, ktoré sa majú z daného okruhu v teste vygenerovať.

tQuestion

Každá otázka má v tejto tabuľke svoje znenie a prepojenie na okruh tém (*tDomain*).

tAnswerSet

Každá otázka môže obsahovať viacero sád odpovedí a práve táto tabuľka formou one-to-many preväzuje otázku (*tQuestion*) a jednotlivé odpovede (*tAnswer*).

tAnswer

Odpovede (*tAnswer*) sa prepájajú so sadou odpovedí (*tAnswerSet*). Táto tabuľka obsahuje ešte samotné znenie odpovede a informáciu o tom, či je táto odpoveď správna.

tExam

Táto tabuľka by sa dala nazvať ako zoznam termínov. Obsahuje prepojenie na predmet (*tCourse*), ďalej obsahuje dátum konania skúšky, stav skúšky (pripravovaná, práve prebiehajúca a ukončená) a nakoniec komentár, ktorý môže napísať učiteľ k danému termínu.

tExamToUser

Jedná sa o prepojenie typu many-to-many študentov (*tUser*) prihlásených na daný termín (*tExam*).

tTestTemplate

Táto tabuľka je vlastne šablóna testu. Nie fyzický test, ale iba informácia o tom aké otázky a aké sady odpovedí sú k tejto šablóne pripojené. Obsahuje prepojenie na predmet (*tCourse*), pre ktorý bola vygenerovaná a informáciu o tom, či sa jedná o test v papierovej alebo elektronickej podobe.

tTestItems

Šablóna testu (*tTestTemplate*) obsahuje sady odpovedí (*tAnswerSet*) a k nim prislúchajúce otázky a práve táto tabuľka tieto prepojenia umožňuje spraviť formou many-to-many.

tTest

Samotný "fyzický" test, je uložený v tejto tabuľke. Obsahuje prepojenia na termín (*tExam*), šablónu testu (*tTestTemplate*) a študenta (*tUser*). Okrem toho obsahuje pridelenú známku, celkový čas vypracovávania, komentár od učiteľa/proktora a počet získaných bodov.

tTestAnswer

Táto tabuľka spolu s *tTestAnswerToAnswer* obsahujú vypracovaný samotný test prepojený na test (*tTest*). Nachádza sa v nej vyhodnotenie otázky, teda typ odpovede (správna odpoveď, podmnožina správnych odpovedí, nesprávna odpoveď a nezodpovedaná otázka), pridelené body za túto otázku, informáciu o farbe otázky (podrobnejšie vysvetlené v časti 1.2.1.5.2), komentár od študenta k danej otázke (nejednoznačnosť...) a index určujúci poradie otázky v rámci testu.

tTestAnswerToAnswer

Samotné odpovede študenta sa ukladajú do tejto tabuľky. Obsahuje prepojenia s odpoveďami (*tAnswer*) a vypracovanými otázkami (*tTestAnswer*). Okrem hodnoty odpovede (0,1 – áno/nie) obsahuje aj poradie odpovede v rámci sady.

tSession

Do tejto tabuľky sa ukladajú informácie o aktuálnych sedeniach (session) v systéme. Okrem samotnej Session Id sa zaznamenáva aj čas posledného prístupu.

tTimeStamps

Do tejto pomocnej tabuľky sa zaznamenávajú "pingy" formou časových pečiatok počas testovania, aby sa v prípade výpadku systému dalo zistiť, kedy výpadok nastal. Obsahuje prepojenia na študenta (*tUser*) a na aktuálny test (*tTest*).

1.3.4.2 Prepojenia medzi tabuľkami

V predchádzajúcom texte boli spomínané 2 najrozšírenejšie prepojenia medzi tabuľkami. Prepojenia sa vytvárajú cudzími kľúčmi a môžu byť typu:

1. *many-to-many* prepojenie medzi *A* a *B*

2. *one-to-many* prepojenie medzi *A* a *B*

Prvé spomínané prepojenie označuje vlastnosť, že každému záznamu v *A* môžu byť priradené viaceré záznamy v *B* a opačne, každému záznamu v *B* môžu byť priradené viaceré záznamy v *A*. Typickým príkladom je napríklad prepojenie medzi študentami a predmetmi, kde jeden študent môže byť zapísaný na viacero predmetov a naopak, jeden predmet môže mať zapísaných veľa študentov. Toto prepojenie sa rieši pomocou ďalšej tabuľky, v ktorej si pamätáme jednoznačné identifikátory oboch tabuliek.

Rozšírenejšie prepojenie je druhý spomínaný typ a označuje fakt, že každý záznam v tabuľke A môže mať priradený väčší počet záznamov v tabuľke B. Príkladom môže byť prepojenie predmetu a termínu. Jeden predmet môže obsahovať viac termínov ale opačne to neplatí. Toto prepojenie sa rieši stĺpcom v tabuľke *B* obsahujúcim jednoznačný identifikátor záznamu v tabuľke *A*.

1.4 Etapa implementácie

Cieľom tejto etapy je programová realizácia jednotlivých častí. Táto takisto veľmi dôležitá etapa už bola robená prakticky priamočiaro s využitím poznatkov získaných počas uplynulých etáp. Priamo počas programovania dochádzalo aj k priebežnému testovaniu jednotlivých častí, ale v menšom rozsahu. Výsledok tejto etapy je podrobnejšie vysvetlený v samostatnej kapitole venujúcej sa zdrojovému kódu (3. kapitola). Výsledkom tejto etapy je ale aj podrobný popis prostredia a ovládania, ktorý bol už z časti uvedený v časti špecifikácie, ale podrobnejšie sa mu venujem v samostatnej kapitole venujúcej sa popisu naprogramovaného projektu (2. kapitola).

1.5 Etapa testovania

Táto etapa v našom prípade bola relatívne jednoduchá. Jej cieľom je overiť funkčnosť a úspešnosť kooperácie jednotlivých programových častí ako celku. Pri tomto testovaní sa samozrejme objavilo niekoľko chýb. Najčastejšie chyby vznikali v dôsledku toho, že keďže všetky 3 moduly na seba nadväzujú, tak sa niekto spoliehal na nejakú vlastnosť

systému toho druhého, ktorá tam v skutočnosti nebola alebo bola zabudnutá alebo vyhodnotená ako zbytočná. Po odstránení týchto nedostatkov bol systém pripravený na odovzdanie. Vyvrcholením tejto etapy bolo takzvané akceptačné testovanie, ktoré vykonáva zadávateľ. V našom prípade to bol náš školiteľ a táto časť bola obsiahnutá vlastne v záverečnej prezentácii tohoto projektu.

1.6 Etapa prevádzky a údržby systému

Od okamihu, keď je projekt akceptovaný zadávateľom, sú všetky zmeny klasifikované ako údržba. V našom prípade sme sa ale do tejto fázy nedostali vzhľadom na fakt, že náš projekt bude mať opodstatnenie až v najbližšom skúškovom období.

Kapitola 2
Popis naprogramovaného projektu

V nasledujúcej kapitole bude venovaný priestor pre podrobné vysvetlenie správania časti projektu naprogramovanú mnou. Okrem kompletného študentského rozhrania som mal ale na starosti aj celé testovanie, teda aj z pohľadu učiteľa, respektíve proktora, a preto pozornosť bude upriamená aj na túto časť. Vzhľadom na fakt, že štruktúra je už popísaná v špecifikácii projektu, v tejto časti sa budem venovať skôr funkčnosti ako štruktúre.

2.1 Hlavná stránka

Hlavná stránka podáva základné informácie o systéme (v budúcnosti sem bude môcť pribudnúť napríklad krátka príručka ako pracovať s týmto systémom) a tiež ponúka na výber z niekoľkých možností ako pokračovať ďalej. Je to voľba zmeniť heslo a výpis predmetov zapísaných aktuálnym študentom.
V celom rozhraní pre študentov sa nachádza menu s tromi hlavnými položkami a to s už spomínanými voľbami spolu s touto hlavnou stránkou.
Ďalej sa v celom projekte nachádzajú takzvané *breadcrumbs*, ktoré významne uľahčujú orientáciu na stránkach, ktorých štruktúra nie je orientovaná do šírky ale hlavne do hĺbky, ako napríklad môj projekt. Jedná sa o techniku zobrazenia navigácie na internetových stránkach, kde má užívateľ v každom okamihu prehľad o tom, na akej úrovni sa v systéme nachádza. V tomto prípade, keďže sa jedná o systém so štruktúrou takmer výhradne orientovanou do hĺbky, menu takmer úplne stratilo význam, ale pre konzistentnosť s ostatnými časťami systému som ho naprogramoval.

Hneď pod horným bannerom s logom univerzity sa okrem nadpisu nachádza aj informácia informácia o prihlásenej osobe s možnosťou sa odhlásiť. Táto pozícia sa okrem časti testovania nikde nemení, a preto užívateľ nebude mať problém túto voľbu nájsť. Dodržiavanie takýchto štandardov má za následok to, že sa užívateľ v systéme rýchlo udomácni a nebude mať problém s orientáciou na stránke. Táto vlastnosť internetových stránok sa nazýva vonkajšia konzistencia a je veľmi dôležitou súčasťou tvorby stránok, ktoré sú určené pre široké publikum.

2.2 Zmena hesla

V tejto sekcii je študent vyzvaný aby zadal svoje staré heslo a dvakrát nové heslo. Po overení hesla starého, je overená aj zhodnosť nových hesiel a v prípade úspešnosti sú zmeny uložené v databáze.

2.3 Výpis predmetov

Táto sekcia obsahuje výpis všetkých predmetov, ktoré má aktuálny študent zapísané. Každý záznam tohto listu je odkazom na detail o tom ktorom predmete.

2.4 Detail predmetu

V tejto časti sa nachádzajú dôležité informácie o aktuálnom predmete. V popise sa študent môže dozvedieť napríklad sylaby predmetu alebo tu môže nájsť odkaz na domovskú stránku predmetu. Ďalšie informácie, ktoré sa študent z tejto stránky môže dozvedieť je počet otázok v teste, bodovanie, hodnotenie, výsledná známka a tiež zoznam termínov. V prípade, že predmet má priradený demotest, tak ten je v tomto zozname zobrazený a pripravený na spustenie. Ostatné termíny môžme rozdeliť do 3 častí.

- **termíny, ktoré ešte neprebehli –** na mieste týchto je zobrazený iba dátum

- **termíny, ktoré sú už uzavreté** – na mieste týchto je zobrazený dátum spolu so známkou v zátvorke a celý tento text bude link na zobrazenie bližších informácií o termíne
- **termíny, ktoré práve prebiehajú –** takže na mieste týchto je zobrazený dátum s informáciou o tom, že tento termín práve prebieha a kliknutím na tento link je spustený test

Výsledná známka na tejto stránke je vypočítaná ako známka z posledného termínu, na ktorom sa daný študent zúčastnil. V prípade, že takýto termín ešte nebol, bude namiesto známky zobrazený text N/A, teda známka nie je dostupná.

2.5 Detail termínu

Táto stránka zobrazuje výsledky študenta na danom termíne. Nachádza sa tu tabuľka s bodmi, kde pre každú otázku je priradený počet získaných bodov. Zároveň je táto hodnota farebne odlíšená na základe toho o aký typ odpovede sa jedná.

- **správna odpoveď** – body sú zvýraznené zelenou farbou
- **podmnožina správnych odpovedí** – body sú zvýraznené žltou farbou
- **nesprávna odpoveď –** body sú zvýraznené červenou farbou
- **nezodpovedaná otázka** – body sú zvýraznené oranžovou farbou

Ďalej je tu výsledná známka z tohto predmetu a polia pre poznámky od učiteľa len pre tohto konkrétneho študenta, alebo pre všetkých zúčastnených na danom termíne.

2.6 Testovanie

Táto jedna z najdôležitejších kľúčových súčastí celého projektu by sa dala rozdeliť na také dve časti a to demotestovanie a klasické testovanie. Aj keď majú mnoho spoločného, budú vysvetlené oddelene.

2.6.1 Demotestovanie

V tejto časti budú vysvetlené len rozdiely oproti klasickému testovaniu a zvyšok bude vysvetlený v časti o klasickom testovaní. Prvý markantný rozdiel spočíva v tom, že demotestovanie prebieha len zo študentského rozhrania a teda nie je potrebná kooperácia s učiteľským rozhraním. Ďalší veľký rozdiel je v tom, že kdežto pri klasickom testovaní je základom ID termínu, z ktorého sa potom už dá všetko zistiť, v prípade demotestovania je to ID šablóny testu, ktorá je súčasťou vlastností predmetu. Totižto demotest nie je viazaný na žiadny termín, pretože je spustiteľný kedykoľvek a aj keď žiaden termín neexistuje, tak demotest existovať môže. Posledným veľkým rozdielom je fakt, že po skončení demotestovania sa zmažú všetky informácie o tomto teste vrátane záznamu v tabuľke *tTest* a tiež záznamov s vypracovanými odpoveďami a podobne v tabuľkách *tTestAnswer* a *tTestAnswerToAnswer*.

2.6.2 Klasické testovanie

Klasické testovanie môže byť rozdelené na dve časti pretože prebieha z učiteľského aj študentského rozhrania. Každej časti sa budem venovať zvlášť, pretože každá má na starosti trocha iné úlohy.

2.6.2.1 Testovanie z učiteľského rozhrania

Učiteľské rozhranie má v súvislosti s testovaním za úlohu zobraziť učiteľovi/proktorovi zoznam študentov prihlásených a vypracuvávajúcich test. Pri kliknutí na link na spustenie testovania sa v databáze upraví hodnota stavu tohto termínu z "plánovanej" na "prebiehajúcu". Na základe tohto sa v študentskom rozhraní zobrazí tento termín ako spustiteľný.

V zozname prihlásených študentov je pri každom mene checkbox. Tento má význam pri pridávaní času, pretože čas bude pridaný len tým, ktorý majú tento checkbox zaškrtnutý. Presné fungovanie časomiery bude podrobnejšie vysvetlené v časti zaoberajúcej sa testovaním z pohľadu študentského rozhrania.

Ďalšie možnosti tohto rozhrania sú uzavretie termínu a ukončenie testovania. Prvý z nich má za následok označenie termínu ako ukončeného (s možnosťou voľby termín

opäť otvoriť) a druhý spôsobí nastavenie časomiery všetkých študentov na hodnotu aktuálnej časovej pečiatky. Toto spôsobí ukončenie testovania pre všetkých študentov. Zároveň sa termín označí ako ukončený bez možnosti opätovného spustenia.

2.6.2.1 Testovanie zo študentského rozhrania

2.6.2.1.1 Prípravná fáza

Po kliknutí na link na spustenie testu sa najprv overí, či má študent právo na spustenie tohto testu. Teda, či je zapísaný na daný termín, či je prihlásený a podobne. Potom sa v databáze vyberie náhodná testová šablóna prislúcha danému predmetu. Keď sa nájde, tak sa vytvorí záznam v tabuľke *tTest* a tiež v tabuľkách *tTestAnswer* a *tTestAnswerToAnswer*. Do tTest sa pridá informácia o vybratej šablóne, o užívateľovi a o termíne. Zároveň sa pole s otázkami náhodne premieša a na základe nového poradia sa uložia indexy do tabuľky *tTestAnswer*. Potom sa pre každú otázku zoberie množina odpovedí a tiež sa premieša a toto poradie sa uloží do tabuľky *tTestAnswerToAnswer*. V session sa uloží informácia o tom, že test bol už spustený.

Dôležitou súčasťou je testovanie toho, či test už nebol spustený. Ak bol, tak sa na základe časovej pečiatky zistí čas výpadku a na základe toho sa nastaví nová hodnota časomiery pre tento test. Časomiera funguje tým spôsobom, že v tabuľke *tTest* sa nachádza informácia o tom, kedy by mal test skončiť. Ak aktuálna časová pečiatka prevyšuje túto hodnotu, tak test sa ukončí. Takže v prípade výpadku stačí zistiť, že koľko testu zostávalo do konca (uložená hodnota časomiery mínus posledná časová pečiatka) a k tomu prirátať aktuálnu časovú pečiatku.

2.6.2.1.1 Samotné testovanie

Keď už test beží, tak interakciu so serverom môžeme rozdeliť na automatickú komunikáciu pomocou AJAXu a na komunikáciu študenta.

V prvom zmieňovanom prípade sa na server odosiela časová pečiatka a identifikácia testu. Na serveri sa potom tieto hodnoty zapíšu do tabuľky *tTimeStamps* a vráti sa hodnota času do konca testovania. Túto hodnotu spracujeme a zobrazíme v stránke určenej na testovanie. V prípade, že je táto hodnota nulová alebo záporná, tak sa úprava testu zablokuje a zobrazí sa informácia o tom, že časový limit vypršal.

K druhému typu komunikácie so serverom dochádza pri uložení hodnôt testu, teda po kliknutí na tlačidlo uložiť, ktoré odošle formulár na server, kde bude spracovaný. Súčasťou tejto časti je aj možnosť označiť farebne otázky, aby sa študentovi v nich dobre orientovalo. Táto čiste interná záležitosť umožňuje napríklad červenou farbou zvýrazniť otázku, na ktorú študent nevie odpovedať. Tieto hodnoty sa ukladajú do tabuľky *tTestAnswer*. Okrem navigácie (nasledujúca otázka, predchádzajúca otázka) sú tu ešte 2 dôležité tlačidlá a to možnosť locknúť test a možnosť test odoslať.

Lock testu je dôležitá napríklad vtedy, keď si študent potrebuje len na chvíľu odbehnúť ale test ešte neukončil. Zamknutím testu zabezpečí, aby mu niekto jeho test neupravil, bez jeho vedomia. Táto funkcionalita funguje tak, že keď je test locknutý, tak je nastavená session s informáciou o tomto fakte. Vtedy sa čaká na zadanie mena a hesla študenta.

Možnosť odoslať test je už záverečným aktom, ktorý daný test vyhodnotí a výsledky zapíše do databázy. Samotnému hodnoteniu sa venuje samostatná časť.

Ešte je treba spomenúť spôsob akým sa ráta čas strávený nad jednotlivými otázkami. Takže GETovským parametrom sa odosiela informácia o poslednej časovej pečiatke a tiež o indexe otázky. Pri najbližšom odoslaní údajov na server sa zistí aktuálna časová pečiatka, zistí sa teda aký dlhý čas bol strávený nad poslednou otázkou a z indexu otázky sa zistí o ktorú konkrétne otázku išlo. V tabuľke *tTestAnswerToAnswer* sa potom táto informácia uloží. Vzhľadom na čiste informatívny charakter tohto údaju sa nemá zmysel zaoberať bezpečnosťou tohto riešenia (keďže zmeniť GETovký parameter je veľmi jednoduché), ale do budúcna by sa tieto informácie mohli vytvárať session a týmto spôsobom sa odosielať.

2.7. Vyhodnotenie

Prvú časť vyhodnotenia majú demotesty aj klasické testy rovnakú. V prvom rade sa zrušia všetky sessiony vytvorené počas testovania (informácia o spustení, id testu) a skontroluje sa, či test už nebol hodnotený. Ak nie, tak sa vyhodnotí.

Naplní sa pole stupnicou a informáciou o bodovaní a ďalšie pole sa naplní odpoveďami študenta. Odpovede sa zoradia podľa príslušnosti k otázke, a preto pri postupnom

prechádzaní spracovávame postupne kompletné otázky akoby po blokoch. V rámci jednej otázky si pamätáme nasledovné informácie:

- či bola aspoň jedna odpoveď označená (*$answered*)
- či bola zaškrtnutá odpoveď, ktorá by nemala (*$bad*)
- či nebola zaškrtnutá odpoveď, ktorá by mala (*$partial*)

Po prejdení všetkých odpovedí v rámci otázky na základe vyššie uvedených premenných zistíme o aký typ odpovede ide. Ak je premenná *$bad* pravdivá, je jasné, že odpoveď je nesprávna. Ďalej testujeme, či vôbec bola aspoň nejaká odpoveď zaškrtnutá a keď nie *($answered* je nepravdivá), tak je jasné, že na otázku nebolo odpovedané. Ak je pravdivá premenná *$partial*, tak je otázka zodpovedaná čiastočne (určite je aspoň jedna odpoveď správna, pretože ináč by sme už otázku vyhodnotili ako prázdnu alebo nesprávnu). No a ak ani jedna z týchto možností neplatí, tak je jasné, že otázka je zodpovedaná správne. Na základe tohto sa priradí otázke patričný počet bodov a tiež typ odpovede (do tabuľky *tTestAnswer*). Na konci tohto spracovania sa na základe sumy získaných bodov zistí, aká známka má byť priradená a zapíše sa to do patričnej tabuľky v databáze (*tTest*).

Ďalej sa v tejto časti zmaže dočasný záznam v *tTimeStamps* a ak sa jedná o demotest, tak sa po výpise výsledkov zmažú všetky informácie o tomto teste vrátane samotného testu a jeho vypracovania (*tTest*, *tTestAnswer*, *tTestAnswerToAnswer*).

Kapitola 3

Popis zdrojového kódu

V nasledujúcej kapitole budú stručne opísané kľúčové konštrukcie používané v projekte. Najprv stručne opíšem programovú štruktúru projektu, potom bude opísaný význam všeobecných tried, ktoré naprogramoval Martin Roško a budú rozobrané triedy týkajúce sa študentského rozhrania. V ďalšej časti bude venovaná pozornosť JavaScriptu, kde sa nachádza tiež niekoľko kľúčových funkcií.

3.1 Programová štruktúra

Celý projekt je rozdelený do už spomínaných troch častí a to časti študentskej, učiteľskej a administrátorskej. Každá z týchto častí má vlastný priečinok. Zo zdielaných priečinkov je najdôležitejší priečinok s knižnicami, kde s okrem triedy Smarty nachádzajú aj triedy zdieľané v rámci celého projektu a to sú triedy na prácu s databázou, na prácu s menu, so samotnou aplikáciou a tak ďalej. Tieto triedy budú podrobnejšie popísané neskôr.

Štruktúra študentskej časti je nasledovná. V hlavnom priečinku sa nachádza hlavný súbor *index.php*, kde sa prilinkujú všetky knižnice z priečinka knižníc *classes*. V hlavnom priečinku sa nachádza aj externý javascriptovský súbor a súbor *ajax.php*, ktorý ma za úlohu odpovedať na požiadavky vyslané pomocou AJAXu. Okrem už spomínaného priečinka s knižnicami sa tu nachádzajú priečinky s obrázkami (*images*) a definovanie kaskádových štýlov v externom CSS súbore (*styles*). Ďalším významným je priečinok *smarty*, kde sa nachádzajú samotné šablóny, skompilované šablóny, nastavenia a cache. Skompilovaná šablóna je vlastne šablóna preložená Smarty parserom do php kódu, ktorý sa potom už len priamočiaro prekladá. Posledné dva spomínané naše Smarty nevyužíva.

3.2 Triedy

Používanie tried vo väčších projektoch prináša množstvo výhod. Medzi ne patrí napríklad fakt, že kód písaný objektovo je oveľa prehľadnejší, ľahšie sa do neho robia zmeny a je naprogramovaný rýchlejšie (pri väčších projektoch). Vzhľadom na fakt, že PHP je objektovo orientovaný programovací jazyk, tak výnimkou nebol ani náš projekt, a preto sú triedy významnou súčasťou.

3.2.1 Všeobecné triedy

3.2.1.1 cAbstractDB, cMySqlDB, cPostgreSqlDB

Celá práca s databázou je zaobalená do tried cMySqlDB a do cPostgreSqlDB na základe toho, aký databázový systém bude používaný. Štruktúra týchto tried je definovaná v abstraktnej triede cAbstractDB. Poskytované metódy týchto tried sú pripojenie na databázový server, vykonanie dotazu, uloženie výsledku dotazu do poľa a v prípade, že bude vrátených viac riadkov, tak aj uloženie do viacrozmerného poľa.

3.2.1.2 cApplication

Táto trieda má za úlohu vybrať, spracovať a zobraziť požadovanú stránku. Túto stránku určuje parameter *site*. Metódy tejto triedy sú:

- abstraktná metóda *GetPageById($pageId)* - ktorá na základe Id vráti objekt typu cPage
- metóda *Run()* - spustí metódy na spracovanie a vykreslenie požadovanej stránky, na základe úspešnosti metódy *Process()* spustí buď vykreslenie stránky (metóda *Draw()*), alebo spustí metódu *SayError()*, ktorá vypíše chybu

Od tejto triedy sú potom odvodené triedy *cStudentApplication*, *cTeacherApplication* a *cAdminApplication*, pre každý celok zvlášť.

3.2.1.3 cPage

Základom tejto triedy je definovať abstrakciu nad konkrétnymi triedami stránok. Obsahuje tieto hlavné metódy:

- abstraktná metóda *Process()* - jej potomkovia definujú spracovanie údajov
- abstraktná metóda *Draw()* - jej potomkovia pripravujú dáta potrebné na zobrazenie, volá samotné šablóny
- metóda *SayError()* - zobrazí šablónu s chybovou správou

3.2.1.4 cMenu, cMenuItem

Tieto triedy slúžia na prácu s menu. Metódy oboch tried umožňujú pridávanie, vyhľadávanie a zobrazovanie menu v systéme. Vzhľadom na jednoduchosť menu v študentskej časti sú tieto triedy v tejto časti neni využívané.

3.2.2 Triedy študentského rozhrania

Všetky triedy študentského rozhrania sú potomkami všeobecných abstraktných tried. Trieda *cStudentApplication* je potomkom triedy *cApplication* a trieda *cStudentPage* zase triedy *cPage*. Potom ďalšie triedy *cCourseDetailPage*, *cCourseListPage*, *cDemoTestPage*, *cEvaluateTestPage*, *cExamDetailPage*, *cChangePasswordPage*, *cMainPage* a *cTestPage* sú potomkovia triedy *cStudentPage*.

3.2.2.1 cStudentApplication

Táto trieda implementuje metódu *GetPageById($pageId)*. Najprv prejde všetky POSTovské a GETovské parametre a ošetrí ich na výskyt nežiadúcich znakov (bližšie informácie v časti týkajúcej sa bezpečnosti). Potom na základe *$pageId* vráti nejakého potomka triedy *cStudentPage* a ktorého najprv vytvorí.

3.2.2.2 cStudentPage

Táto trieda implementuje metódy *Process()*, *Draw()* a *AssignQueryResult($query, $name, $type)*. Prvá spomínaná iba vráti true alebo false na základe testu, či je užívateľ prihlásený. Metóda *Draw()* zase na základe toho istého testu zobrazí buď šablónu

prihlasovacej stránky alebo hlavnú stránku. Metóda *AssignQueryResult($query, $name, $type)* sa využíva vo viacerých častiach projektu, tak som sa rozhodol ju dať sem. Jej úlohou je poslanie dotazu na MySql server a na základe typu výsledku (pole, viacrozmerné pole) ho poslať ako premennú s menom *$name* šablónovaciemu systému. Súčasťou je ošetrenie správnosti výsledku.

3.2.2.3 cMainPage

V časti *Process()* na základe zvolenej akcie (GETovský parameter) sa buď pokúsi užívateľa prihlásiť alebo odhlásiť. Pri prihlásení sa vytvoria potrebné sessiony ako napríklad id prihláseného užívateľa či jeho plné meno. Tieto sa potom zrušia pri odhlásení. V časti prípravy dát pre šablónovací systém sa nepripravujú žiadne hodnoty.

3.2.2.4 cCourseDetailPage

Táto trieda nespracováva žiadne údaje, len ich vypisuje, preto je všetko dôležité naimplementované v metóde *Draw()*. Vyberá z databázy informácie o skúške, zoznam termínov pre daného študenta (jeho ID je poslané GET parametrom) a výslednú známku a tieto hodnoty odošle šablónovaciemu systému.

3.2.2.5 cCourseListPage

V metóde *Draw()* vyberie z databázy všetky predmety, na ktoré je študent zapísaný.

3.2.2.6 cExamDetailPage

Táto trieda v metóde *Draw()* pripraví hodnoty potrebné pri vypísaní informácií o termíne a to názov predmetu, dátum, známku, komentár, počet bodov a tiež pripraví pole dosiahnutých bodov za jednotlivé otázky a typ odpovede (správnosť).

3.2.2.7 cChangePasswordPage

V metóde *Process()* sa zistí či bol odoslaný formulár a či sú odoslané údaje korektné. V prípade, že áno, tak sa zistí, či je staré heslo správne, a potom upraví záznam v databáze s novým heslom. Metóda *Draw()* iba zavolá príslušnú šablónu.

3.2.2.8 cTestPage a cDemoTestPage

Tieto triedy sú veľmi podobné, ale nie vždy narábajú s tými istými údajmi. Pôvodne boli obe v jednej triede, ale kvôli prehľadnosti som ich nakoniec rozdelil. Základný rozdiel je spomenutý už v časti 2.5, preto sa budem venovať hlavne spoločným črtám. Takže navyše tu je metóda *PrepareParams()*. Má za úlohu načítať potrebné parametre ako napríklad z indexu otázky zistiť ID sady odpovedí a ID vypracovanej odpovede. Keďže sa táto časť vyskytovala v kóde viac krát, tak som jej vyčlenil samostatnú metódu.

Metóda *Process()* spracúva niekoľko možných akcií. Okrem toho, že najprv zisťuje oprávnenosť požiadavky (je test spustený?, nevypršal čas?, je prihlásený užívateľ oprávnený test spustiť?) tak spracúvava požiadavku na spustenie testu, uloženie odpovede, uzamknutie / odomknutie testu alebo označenie otázky nejakou farbou.

Metóda *Draw()* má zase na starosti prípravu všetkých potrebných údajov, ktoré sa zobrazia v teste. To znamená znenie otázok, ich farby, odpovede aktuálnej otázky, informácie o časomiere, o študentovi a tak ďalej.

3.2.2.9 cEvaluateTestPage

V metóde *Process()* sa po teste, či už daný test nebol vyhodnotený spracujú uložené hodnoty a výsledky vyhodnotenia sa uložia. Bližšie informácie o funkčnosti tejto triedy sú k dispozícii v časti 2.6.

V metóde *Draw()* sa okrem klasickej prípravy dát pre šablónovací systém (výsledky z testu) mažú dočasné záznamy v databáze. Je to z toho dôvodu, že sme to nemohli dať do metódy *Process()*, lebo s týmito údajmi pracujeme aj v tejto metóde.

3.3 JavaScript funkcie

3.3.1 Všeobecné funkcie

Funkcie, ktoré sa netýkajú AJAXu sú iba dve a to funkcia na formátovanie číselného údaju s počtom sekúnd na tvar "HH:MM:SS". Názov tejto funkcie je *formatTime(ts)*, kde *ts* je počet sekúnd. Posledná funkcia je timer(), ktorá je periodicky volaná každú sekundu a má na starosti časovač v sekcii testovania.

3.3.2 AJAX

AJAXovské funkcie máme tri. Prvá z nich *createRequestObject()* vráti inštanciu triedy *XMLHttpRequest*, ktorá je pre AJAX kľúčová. Funkcia *sendRequest()* odosiela požiadavky na server s požadovanými parametrami a nakoniec funkcia *handleResponse()* spracúvava vrátenú odpoveď zo servera. Vzhľadom na fakt, že táto odpoveď je jedno číslo (počet sekúnd do konca testovania), tak je výsledok posielaný nie ako XML dokument ale ako čistý text. V prípade, že hodnota prijatej odpovede je nulová alebo záporná, tak sa zmení obsah testovej stránky len na informáciu o tom, že časový limit pre test už vypršal.

Kapitola 4

Bezpečnosť systému

Zabezpečenie bezpečnosti systému je neoddeliteľnou súčasťou každého systému, ktorý narába s citlivými dátami. V našom prípade sme museli okrem útočníkov zvonka myslieť aj na možnosť zneužitia systému zvnútra a to študentmi, ktorých vynaliezavosť je niekedy obdivuhodná. V tejto kapitole budú rozobraté 2 najznámejšie útoky z prostredia webovských aplikácií, a potom sa budem venovať menej formálnym problémom a spôsobom, ako sa voči nim chráni náš program.

4.1 HTML injection

Tento typ útoku sa najčastejšie vyskytuje hlavne na fórach, rôznych stránkach, kde sa dajú pridávať komentáre a podobne. Podstatou je pokus vložiť na stránku svoj vlastný HTML kód. Napríklad, keď sa neošetrí vstup na nejakom fóre a útočník ako príspevok napíše jednoduchý JavaScipt na presmerovanie na svoju stránku, tak každý návštevník, ktorý si tento príspevok prezrie, bude presmerovaný.

Ochrana pred týmto typom útoku je veľmi jednoduchá. Stačí každý inkriminovaný vstup prefiltrovať cez funkciu *htmlspecialchars*, ktorá všetky HTML tagy prekonvertuje do bezpečného formátu, takzvaných escape characters. Napríklad znak začiatku tagu "<" prekonvertuje na "<".

V mojej časti som to riešil globálne a v triede *cStudentApplication* som prešiel všetky GETovské a POSTovské parametre a ošetril som ich uvedenou funkciou. Okrem toho Smarty ponúka modifikátor *escape*, ktorý tiež skonvertuje nebezpečné znaky do escape characters.

4.2 SQL injection

Tento snáď najjednoduchší a najbežnejší spôsob útoku na databázu má veľmi jednoduchú myšlienku. Typickým spôsobom testovania správnosti mena a hesla je nasledujúcim spôsobom:

```
$query = “SELECT * FROM Users WHERE login=’$_POST[‘login’]’ AND pass=’$_POST[‘pass’]’”;
$result = mysql_query( $query );
if (mysql_num_rows($result)>0)
{
        // uzivatel je prihlaseny
        // jeho udaje su v prvom riadku vrateneho pola
}
```

Uvažujme prípad, že útočník zadá ako parameter login nasledujúci text:

```
' OR 1 OR '
```

Potom bude mať vykonávaný príkaz nasledovnú syntax:

```
SELECT * FROM Users WHERE login='' OR 1 OR '' AND pass=''
```

Je zrejmé, že *WHERE* podmienka sa vyhodnotí vždy ako pravdivá a preto záznam vráti všetkých užívateľov a prvý z nich štandardne býva administrátor. Týmto sa to ale nekončí, pretože útočník môže úplne získať kontrolu nad databázou tým, že tam vloží ďalší príkaz (napríklad DROP TABLE...).

Ochrana pred uvedenou vážnou bezpečnostnou chybou je tiež veľmi jednoduchá. Stačí na chúlostivé vstupy aplikovať funkciu *mysql_real_escape_string*, ktorá sa postará o to, aby sa napríklad pred úvodzovky dali ešte spätné lomítka.

V mojej časti som to opäť riešil globálne v triede cStudentApplication podobne ako pri HTML injection.

4.3 MD5 "dekryptovanie"

Hašovacie algoritmy používané na ukladanie hesiel v databáze prinášajú určité zvýšenie bezpečnosti, ale zároveň majú aj nedostatky. Jedným z nich je napríklad kolíznosť

hašovacích algoritmov, ale tejto téme nebude venovaný priestor vzhľadom na neaktuálnosť pre danú problematiku.

Problémom je skôr fakt, že užívatelia si často vyberajú veľmi jednoduché heslá a na svete je množstvo databáz obsahujúcich obrovské množstvo potenciálnych hesiel s ich hašom (väčšinou MD5 hašom, ktorý je najpoužívanejší). Preto ak sa niekto dostane k heslám v našej databáze, tak by až 80% hesiel mohol odchytiť práve vďaka takýmto stránkam.

Ďalším problémom je fakt, že mnoho užívateľov má rovnaké heslá. Potom ak niekto chce získať heslo užívateľa A, ktorý má také isté heslo ako užívateľ B (a tým pádom aj rovnaký haš), stačí mu presvedčiť užívateľa B o tom, aby mu to heslo povedal.

Aby sme zabránili jednoduchosti hesiel, tak využívame takzvané "osolenie" hesiel. Pre každé heslo si budeme pamätať ešte náhodný reťazec znakov rozumnej dĺžky a do databázy nebudeme ukladať priamo haš hesla, ale haš zreťazenia tohto náhodného reťazca a samotného hesla. Týmto spôsobom budeme aj testovať správnosť hašu pri autentifikácii.

4.4 Neoprávnené požiadavky

Väčšina parametrov je odovzdávaná metódou GET, takže užívateľ veľmi rýchlo pochopí napríklad význam zápisu url adresy: "http://.../index.php?site=examDetail&itemId=1".

Táto linka spustí bližšie informácie o termíne s ID rovným 1. Preto keby užívateľ zadal namiesto itemId=1 napríklad 4, tak by sa mohol dostať k termínu, na ktorom by nemal čo hľadať. Preto je na každej stránke test oprávnenosti na danú akciu. Napríklad v tomto prípade je na základe ID prihláseného užívateľa zistené, či užívateľ má daný termín absolvovaný.

4.5 Zamedzenie podvádzania študentov

Posledná časť kapitoly o bezpečnosti je zameraná na nie chyby v systéme, ale na to, ako zabrániť veľmi častému javu na skúškach a to podvádzanie študentov.

Jeden z najrozšírenejších spôsobov je vypracovanie testu za niekoho iného. Tento problém je nutné riešiť za kooperácie proktora, ktorý po usadení študentov, spustení testovania a uzavretí testovania musí ručne skontrolovať či každý študent vypracováva test pod svojím menom a pod svojím ISIC číslom (ktoré je zobrazené počas testovania). Potom už len stačí, aby proktor dával pozor na to, aby sa niekto nepresadil. Odhlásenie a opätovné prihlásenie iným menom už nie je možné.

Ďalším problémom je zlomyseľnosť študenta. Napríklad si "omylom" vypne počítač, alebo sa niekto postará o vypadnutie sieťového pripojenia počas testovania. Riešením sú permanentné ukladanie údajov na server a posielanie pingov s časovými pečiatkami na server, kde sa tieto údaje ukladajú a na základe nich sa potom dá zistiť, že kedy výpadok nastal a test obnoviť do stavu v akom bol, keď výpadok nastal.

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť aplikáciu na online testovanie študentov formou testov a tým testovanie zjednodušiť a urýchliť. Vytvorená aplikácia vznikala v tíme so 4 členmi, konkrétne náš školiteľ RNDr. Richard Ostertág a vývojári Juraj Porubský, Martin Roško a ja. Mojou úlohou v tomto projekte bolo hlavne vytvorenie študentského rozhrania a samotného testovania aj pre učiteľské rozhranie. Po sérii stretnutí aj s ostatnými kolegami sa nám najprv podarilo projekt detailne vyšpecifikovať a nakoniec aj doviesť do úspešného konca. Myslíme si, že sa nám stanovené ciele podarilo dosiahnuť a aplikácia je teda pripravená na použitie v reálnych podmienkach.

Aplikácia ale ešte má potenciál na ďalší vývoj. Napríklad by sa mohli doimplementovať programové časti spomínané už v časti 1.2, ktoré boli z rôznych dôvodov z projektu vynechané. Pri generovaní testov by sa mohlo napríklad rátať aj so zložitosťou jednotlivých otázok a tak generovať testy s približne rovnakou obtiažnosťou. Ďalej by sa mohla vytvoriť aplikácia, ktorá by spracovala naskenované vytlačené testy a v rozumnej forme (XML) by bola schopná zistené údaje exportovať. Priestor pre ďalší vývoj je aj v bezpečnosti, pretože každý väčší projekt má svoje neduhy, ktoré sa dajú opraviť až po nejakom čase v ostrej prevádzke.

Zoznam použitej literatúry

[1] ŠEŠERA, Ľ., MIČOVSKÝ, A. *Objektovo-orientovaná tvorba systémov a jazyk C++*. Perfekt, 1994. ISBN 80-85261-66-9

[2] KATUŠČÁK, D.: *Ako písať vysokoškolské a kvalifikačné práce.* 2. vyd. Bratislava. 1998. 119 s. ISBN 80-85697-82-3

[3] Wikipédia – slobodná encyklopédia. http://www.wikipedia.org/ (2007-05-30)

[4] JANOVSKÝ, D.: Jak psát web. http://www.jakpsatweb.cz/ (2007-05-27)

[5] JavaScript Tutorial. http://www.tizag.com/javascriptT/ (2007-05-27)

[6] MySQL 5.0 Reference Manual. http://dev.mysql.com/doc/refman/5.0/en/ (2007-05-30)

[7] PHP: Hypertext Preprocessor. http://www.php.net/ (2007-05-27)

[8] Ajax Tutorial : Ajax Help and Tutorials. http://www.ajaxtutorial.net/ (2007-05-30)

[9] Kouba, Š.: SMARTY - chytré šablony pro PHP. http://interval.cz/clanky/smarty-chytre-sablony-pro-php/ (2007-05-27)

A Prílohy

K práci je priložená CD príloha obsahujúca kompletný projekt *Probatur – online testovanie študentov.* Na stránke http://www.sprite.edi.fmph.uniba.sk/~probatur/ sa dá celý projekt otestovať.