Probatur – online testovanie študentov

BAKALÁRSKA PRÁCA

Martin Roško **2007**

PROBATUR – ONLINE TESTOVANIE ŠTUDENTOV

BAKALÁRSKA PRÁCA

Martin Roško

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KATEDRA INFORMATIKY

Študijný odbor: 9.2.1 Informatika

Vedúci bakalárskej práce: RNDr. Richard Ostertág

Bratislava 2007

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

V Bratislave 8.júna 2007 _____

Ďakujem vedúcemu mojej bakalárskej práce RNDr. Richardovi Ostertágovi za pomoc pri vypracovávaní špecifikácie a cenné podnety pri jej implementovaní.

Ďakujem mojim kamarátom a spolužiakom Jurajovi Porubskému a Františkovi Šmitalovi za spoluprácu na projekte Probatur.

Abstrakt

ROŠKO, Martin: Probatur – online testovanie študentov. [bakalárska práca] – Fakulta matematiky, fyziky a informatiky Univerzity Komenského v Bratislave; Katedra informatiky. – Školiteľ: RNDr. Richard Ostertág – Bratislava 2007. 49 strán.

Bakalárska práca sa zaoberá návrhom a analýzou webovskej aplikácie pre online testovanie študentov. Popisuje postup práce pri vývoji aplikácie a rozoberá jej bezpečnosť. V práci sa kladie väčší dôraz na návrh a analýzu administrátorského rozhrania aplikácie. Obsahuje popis databázového modelu aplikácie. Definovaný je v nej protokol pre výmenu dát medzi aplikáciou a externým systémom, poskytujúcim údaje o študentoch, učiteľoch, predmetoch a skúškach. K bakalárskej práci je priložená výsledná aplikácia pre online testovanie študentov.

Kľúčové slová: online testovanie študentov, PHP, SQL

Predslov

Testovanie vedomostí študentov môže mať rôzne formy. Ústna odpoveď, písomné skúšanie formou riešenia príkladov a úloh na papier, alebo môže byť vo forme testu. Pri písomnej skúške musia študenti na svoje výsledky čakať často až niekoľko dní a učitelia musia pracne opraviť každú odovzdanú písomku, čo je v prípade písomnej skúšky vo forme testu neefektívne, keďže odpovede majú vopred definovanú štruktúru vo forme označenej alebo neoznačenej odpovede. Navyše musia učitelia v snahe zabrániť opakovaným zadaniam testov, vymýšľať a formulovať otázky vždy nanovo. Práve tu sa ponúka možnosť využitia webovskej aplikácie, ktorá by uľahčila prácu učiteľov a urýchlila vyhodnocovanie testov pre študentov.

Systém pre online testovanie študentov je aplikácia, ktorá dáva učiteľom možnosť automatického generovania testov, na základe obtiažnosti, zo sád otázok, ktoré si samy vytvoria, vykoná ich automatickú opravu a ponúka im prehľadnú štatistiku doterajších výsledkov zo skúšky.

Študentom poskytuje pohodlnú formu skúšky, prakticky okamžité poskytnutie výsledku odovzdaného testu, prehľad doterajších výsledkov z testov, ktoré študent absolvoval a skúšok, ktoré ho čakajú.

Použité technológie sú voľne dostupné a systém môže byť použitý v ľubovoľnej sieti, ktorá ich podporuje. Zároveň aplikácia nie je viazaná na žiadnu konkrétnu databázu údajov. Tie sú do systému importované prostredníctvom jednoduchého, v tejto práci definovaného, protokolu pre výmenu informácií o študentoch a termínoch skúšok.

Obsah

Úvod

Obsahom tejto práce je, okrem samotnej aplikácie, postup pri jej implementácií počnúc návrhom a špecifikáciou požiadaviek, návrhom databázového modelu, analýzou a popisom tried, na ktorých je vystavaná samotná aplikácia až po úvahy a riešenia bezpečenosti samotnej aplikácie.

V prvej kapitole sa venujem návrhu a špecifikácií požiadaviek na aplikáciu. Opisujem v nej rozdelenie práce na aplikácií, na ktorej okrem mňa pracovali spolužiaci Juraj Porubský a František Šmitala. Rozoberám v nej funkcionalitu a štruktúru administrátorského rozhrania. Detailne v nej bude opísaný databázový model.

V druhej kapitole sa venujem protokolu pre výmenu dát medzi aplikáciou a externým systémom, ktorý poskytuje aplikácií dáta o študentoch, učiteľoch. predmetoch a skúškach vo formáte *xml.*

Tretia kapitola sa venuje samotnej implementácií aplikácie. V úvode sa v nej venujem výberu prostredia, v ktorom bola aplikácia písaná a výberu technológií, ktoré potrebuje pre svoj beh. Zdôvodním výber šablónového systému pre zvýšenie prehľadnosti kódu a opíšem jednotlivé triedy so stručným popisom ich metód.

Štvrtá kapitola je zameraná na úvahy o bezpečnosti. Popíšem v nej algoritmus, ktorý som využil pri šifrovaní prenosu dát medzi aplikáciou a externým systémom.

V ďaľšom texte sa bude pod pojmom systém rozumieť samotná aplikácia, teda systém pre online testovanie študentov. Pojmom administrátor alebo správca budem označovať osobu, ktorá má prihlasovacie práva do administrátorského rozhrania aplikácie.

1. Návrh a špecifikácia požiadaviek

Pred samotnou implementáciou aplikácie, bolo dôležité špecifikovať požiadavky a navrhnúť databázový model s ktorým sa bude pracovať. Spolu s mojími kolegami a vedúcim bakalárskej práce sme absolvovali radu stretnutí, pri ktorých sa diskutovalo o požiadavkach, ktoré sú kladené na aplikáciu, problémoch, ktoré by mohli ich implementáciou alebo používaním aplikácie vzniknúť a hľadali sme najlepšie riešenia a podobu, v akej by mohli byť zahrnuté do výslednej špecifikácie.
Keďže výslednú aplikáciu budú využívať študenti a učitelia, bolo dobré že sa na stretnutiach okrem nás, ako študentov, pravidelne zúčastňoval aj vedúci našej bakalárskej práce, ktorý nám formuloval požiadavky z učiteľskej strany.
Pre zvýšenie efektivity pri programovaní sme si vopred rozdelili úlohy a prácu na celej aplikácií. Systém pozostáva z troch väčších celkov: učiteľského, študentského a administrátorského rozhrania, ktoré sú od seba nezávislé, zdieľajú ale tú istú databázu údajov. V mojej práci sa venujem časti, na ktorej som pracoval ja, administrátorskému rozhraniu a protokolu pre výmenu dát s externým systémom.

1.1 Rozhranie pre administrátora

Systém pre online testovanie študentov, by nemohol správne fungovať bez rozhrania pre správcu systému, v ktorom administrátor vykonáva základné akcie pre správne fungovanie celého systému pre testovanie študentov. Znamená to, že sa v tejto časti musia nachádzať nástroje pre vytváranie užívateľov a predmetov. Pojmom užívateľ označujeme študenta alebo učiteľa. Administrátor navyše musí mať možnosť meniť alebo mazať nesprávne zadané, alebo zle naimportované údaje.

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.

1.1.1 Prihlasovacia stránka

Administátorské rozhranie je určené výlučne pre správcu sytému, 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.

Súčasťou prihlasovacej stránky bude stručná poznámka o rozhraní pre administrátora, ktorá poskytne užívateľom, ktorí nie sú administrátormi, informáciu o kontakte na administrátora a odkáže ich na rozhranie pre študenta respektíve na rozhranie pre učiteľa.

1.1.2 Hlavná stránka

Hlavná stránka rozhrania je rozdelená na dve časti. Naľavo je zobrazené menu, ktoré uľahčuje navigáciu a poskytuje prehľad o tom, kde sa práve nachádza prihlásený administrátor. Položky menu sú rozbaľovacie, znamená to, že po kliknutí na niektorú z nich sa môže zobraziť zoznam ich podpoložiek. Po kliknutí na ľubovoľnú z nich sa vykoná príslušná akcia, ktorá vo väčšine prípadov znamená presmerovanie na niektorú podstránku administrátorského rozhrania. Výnimkou je položka *Logout*, ktorou sa administrátor odhlasuje zo systému.

Položka v menu môže obsahovať ľubovoľný počet podpoložiek. Pre väčšiu prehľadnosť je menu navrhnuté tak, že maximálny počet podpoložiek je 3. Menu administrátorského rozhrania obsahuje tieto položky:

1. *Študenti* – kliknutím na túto položku sa administrátor dostane na stránku so zoznamom študentov. Táto položka je zároveň rozbalovacia, čo znamená, že obsahuje ďalšie podpoložky:
    1.1. *Zoznam* – je ekvivalentná položke *Študenti*, po kliknutí na ktorú, sa automaticky označí práve táto položka.

1.2. *Pridať* – presmeruje administrátora na stránku pre pridávanie nových študentov.

1.3. *Vyhľadať* – zobrazí v hlavnom okne formulár s filtrom pre vyhľadávanie v databáze študentov.

2. *Učitelia* – zobrazí zoznam učiteľov a obsahuje podpoložky slúžiace na vykonávanie akcií spojených s učiteľmi.

   2.1. *Zoznam* – podobne ako v časti menu pod položkou *Študenti* je aj tu položka *Zoznam* ekvivalentná svojej nadpoložke a kliknutím na ňu sa administrátorovi zobrazí zoznam učiteľov v databáze.

   2.2. *Pridať* – zobrazí formulár pre pridávanie učiteľov do databázy systému.

   2.3. *Vyhľadať* – ponúkne administrátorovi možnosť vyhľadávať v databáze učiteľov.

3. *Odbory* – obsahuje položky, ktoré umožňujú prácu so štúdijnými odbormi, do ktorých sú v systéme zaraďovaný študenti.

   3.1. *Zoznam* – presmeruje stránku na zoznam všetkých štúdijných odborov, ktoré sa nachádzajú v databáze systému.

   3.2. *Pridať* – ak chce administrátor pridať nový štúdijný odbor, k formuláru na jeho pridanie sa dostane cez túto položku v menu

4. *Predmety* – táto časť systémového menu obsahuje navigáciu k stránkam pre správu predmetov.

   4.1. *Zoznam* – vybraním tejto položky sa užívateľovi zobrazí stránka so zoznamom predmetov.

   4.2. *Pridať* - zobrazí formulár pre pridávanie predmetov do databázy

   4.3. *Vyhľadať* – presmeruje administrátora na stránku obsahujúcu filter pre vyhľadávanie predmetov v systéme.

5. *Import* – táto položka neobsahuje žiadne podpoložky, po kliknutí na ňu sa zobrazí stránka s linkami na importovanie dát do systému, zo vzdialeného servera.

6. *Záloha* – podobne ako predošlá, ani táto položka nie je rozbaľovacie, neobsahuje teda žiadne podpoložky. Presmeruje administrátora na zálohovaciu stránku.

7. *Nastavenia* – položka menu, ktorou sa otvárajú systémové nastavenia. Obsahuje dve podpoložky:
    7.1. *Zmena hesla* – zobrazí formulár, ktorým sa mení heslo pre prístup do administrátorskeho rozhrania systému.
    7.2. *Importovanie* – načíta stránku s nastaveniami pre pripájanie sa k vzdialenému serveru pre import dát.
8. *Logout* – kliknutím na túto položku sa administrátor odhlási zo stránky.

V pravej časti hlavnej stránky sa nachádza okno, do ktorého sa načítavajú jednotlivé podstránky administrátorského rozhrania. Nad týmto oknom je umiestnený informačný pás, názov aktuálne načítanej sekcie a odkaz na odhlásenie sa zo systému.

1.1.3 Študenti

Študenti sa prihlasujú do systému pre online testovanie cez študentské rozhranie. Pre správne fungovanie však musia byť vytvorené a každému študenovi pridelené kontá, tak aby bolo možné, po prihlásení sa študenta, ho jednoznačne identifikovať v systéme. Každý študent musí mať pridelené jednoznačné prihlasovacie meno a heslo, má pridelené predmety, ktoré má v danom roku zapísané a z ktorých môže byť skúšaný. Úlohou administrátora je zabezpečiť, aby sa každý študent dokázal do systému prihlásiť a aby mal pridelené všetky predmety, ktoré potrebuje. Aby bola práca administrátora efektívna, musí byť sekcia so študentami prehľadná, jednoduchá na ovládanie a intuitívna.

Hlavným prvkom tejto časti je zoznam študentov, na ktorý sa dá dostať priamo z hlavného menu, kliknutím na položku *Študenti*. V zozname sa zobrazuje meno a priezvisko študentov, študijný odbor, ktorý navštevujú, rok štúdia a základné akcie pre prácu s jednotlivými študentskými kontami. Pod základnými akciami chápeme tieto:

1. *detaily* – zobrazí informácie o študentskom konte
2. *editovať* – zobrazí formulár pre úpravu informácií o študentskom konte
3. *zmazať* – zruší príslušné študentské konto

Predpokladom je, že v systéme bude veľký počet študentov, preto musí byť zoznam rozdelený na menšie časti. Listovať v ňom sa dá pomocou odkazov *ďalej* a *naspäť* a jednotlivé stránky zoznamu sa dajú načítať po kliknutí na *index* stránky. Dôležitou vlastnosťou zoznamu je to, že je utriediteľný podľa stĺpcov ktoré obsahuje a to vzostupne aj zostupne. Kvôli prehľadnosti majú susedné riadky iné farebné pozadie a riadok, nad ktorým je kurzor je farebne zvýraznený. Informácia o študijnom odbore študenta je hyperlink, kliknutím na ktorý sa užívateľ presunie na stránku s detailnými informáciami o konkrétnom odbore.

Vybratím akcie *detaily* na konkrétnom študentovi, sa zobrazí stránka s detailnejšími informáciami. Obsahuje, základné informácie ako je meno, priezvisko, odbor a ročník, navyše je tu informácia o predmetoch, ktoré má študent v systéme zapísané s možnosťou pridelovania nových predmetov študentovi alebo odoberania už pridelených predmetov. Z tejto stránky sa dá na jeden krok dostať do všetkých ostatných podstránok, ktoré súvisia s editáciou vybratého študentského konta, kliknutím na jeden z odkazov: *editovať, zmeniť heslo, vymazať.*

Akciou e*ditovať* alebo vybratím položky *pridať* z hlavného menu, sa zobrazí formulár pre prácu s údajmi o študentskom konte. Formulár pre pridávanie a editovanie študentského konta, je rovnaký. Rozdiel medzi nimi je v tom, že pri editovaní, sú informácie vo formulári vyplnené aktuálnymi údajmi o študentovi. Formulár obsahuje tieto položky:

1. *Titul(pred)* – titul študenta, ktorý sa uvádza pred menom, môže ostať nevyplnený
2. *Meno* – povinný údaj, meno študenta
3. *Priezvisko* – povinný údaj, priezvisko študenta
4. *Titul(za)* – titul študenta, ktorý sa uvádza za jeho menom, môže ostať nevyplnený
5. *ISIC* – povinný údaj, identifikačné číslo študenta nachádzajúce sa na jeho ISIC karte
6. *Rok štúdia* – povinný údaj, môže byť v rozsahu 1 až 8
7. *Odbor* – vyberá sa z listu odborov, nemusí byť vyplnený

V prípade nesprávneho vyplnenia formulára alebo v prípade, že zadané *ISIC* číslo koliduje s iným ISIC číslom v databáze, je administrátor upozornený, ale údaje ktoré už vyplnil ostávajú vo formulári. Po akceptovaní požiadavky na vytvorenie alebo editovanie študentského konta, je správca presmerovaný na stránku s informáciami o študentovi.

Akcia *zmeniť heslo* presmeruje administrátora na stránku, na ktorej môže zmeniť prihlasovacie heslo vybratého študenta. Administrátor nepotrebuje poznať predchádzajúce heslo, ale pre overenie správnosti zadaného hesla, musí nové heslo vyplniť dvakrát, pre prípad, že by došlo k preklepu.

Akciou *vymazať* sa odoberajú študentské kontá z databázy. Mohlo by sa stať, že pri vyberaní akcie administrátor omylom klikne na akciu *vymazať*, preto musí byť každá žiadosť o vymazanie z databázy potvrdená ešte raz.

Ak administrátor hľadá konkrétneho študenta alebo skupinu študentov, môže využiť možnosť vyhľadávania v databáze študentov. Filter pre vyhľadávanie sa zobrazí po vybratí položky *Vyhľadať* pod položkou *Študenti*. Vyhľadávať môže správca podľa mena, priezviska, prihlasovacieho mena, ISIC čísla, roku štúdia a štúdijného odboru. Údaje, ktoré ostanú nevyplnené nebudú vo výsledkoch vyhľadávania zohľadnené. Stačí vypísať prefix hodnoty, napríklad mena alebo priezviska, ktoré má byť vyhľadané. Po stlačení tlačítka *Hľadať* sa zobrazí zoznam študentov, ktorí vyhovujú zadanému filtru. Zoznam je podobný hlavnému zoznamu študentov a dajú sa priamo z neho vykonávať akcie na jednotlivých študentských účtoch.

1.1.4 Učitelia

Tak ako študenti, musia mať aj učitelia pridelené prihlasovacie údaje a musia byť jednoznačne identifikovateľný so svojim kontom v systéme. Štruktúra sekcia s učiteľmi je podobná štruktúre študentskej sekcie. Hlavným navigačným prvkom je zoznam učiteľov, ktorý obsahuje informáciu o mene a priezvisku učiteľa, jeho *ITIC* číslo a rovnaké základné akcie pre prácu s učiteľským kontom ako zoznam študentov: *detaily, editovať, vymazať*.

Podstránka systému o infromáciach o učiteľskom konte poskytuje administrátorovi detailný prehľad o učiteľovi, obsahuje celé meno a priezvisko učiteľa, jeho prihlasovacie meno, ITIC číslo a zoznam predmetov, ktoré má v systéme učiteľ pridelené. Na podstránke sú umiestnené navigačné prvky, pre zobrazenie editovacieho formulára, zmenu prihlasovacieho hesla a vymazávanie konta z databázy.

Formulár pre editovanie a pridávanie učiteľského konta obsahuje tieto prvky:

1. *Titul(pred)* – titul učiteľa, ktorý sa uvádza pred menom, môže ostať nevyplnený
2. *Meno* – povinný údaj, meno učiteľa
3. *Priezvisko* – povinný údaj, priezvisko učiteľa
4. *Titul(za)* – titul učiteľa, ktorý sa uvádza za jeho menom, môže ostať nevyplnený
5. *ITIC* – povinný údaj, identifikačné číslo učiteľa nachádzajúce sa na jeho ITIC karte

Pre zmenu hesla a vymazávanie platia rovnaké pravidlá, aké boli definované v študentskej sekcií. Vyhľadávanie je prístupné po vybratí položky *Vyhľadať* z učiteľského menu, filtrovať sa dá podľa prihlasovacieho mena, mena, priezviska a ITIC čísla učiteľa. Po stlačení tlačítka *Hľadať* sa zobrazí zoznam učiteľov, ktorý vyhovujú filtru.

1.1.5 Odbory

Štúdijné odbory boli 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.

Zoznam odborov sa zobrazí po vybratí položky *Odbory* z hlavného menu, môže byť utriedený podľa názvu, alebo podľa mena garanta odboru. Každý riadok v zozname obsahuje názov štúdijného odboru, meno granta a základné akcie pre prácu s odbormi: *detaily, editovať, zmazať*.

Na detailnej stránke odboru je vypísaný okrem názvu študijného odboru a jeho garanta aj zoznam študentov daného odboru, rozdelených podľa ročníka štúdia. Kliknutím na meno študenta v zozname sa stránka presmeruje na informácie o vybratom študentovi.

Pri pridávaní a editovaní odborov sa používa rovnaký formulár, v prípade editovania, sú hodnoty v ňom vopred vyplnené. Každý odbor musí mať jedinečné meno, preto bude v prípade kolízie mien administrátor upozornený na chybu pri zadávaní údajov. Meno je povinný údaj, garant študijného odboru nemusí byť nutne vybratý zo zoznamu potenciálnych garantov.

1.1.6 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.

Aj v sekcií predmetov, administrátorského rozhrania, je hlavným prvkom zoznam už vytvorených a v databáze sa nachádzajúcich predmetov. Zoznam je triediteľný podľa názvu, podľa kódu predmetu alebo podľa mena profesora, ktorý ho má pridelený. Každý riadok obsahuje základnú informáciu o predmete, kliknutím na učiteľa sa dá dostať do sekcie učiteľov na stránku s detailnými informáciami o konkrétnom učiteľovi. Tak ako zoznamy v predchádzajúcich sekciách, aj zoznam predmetov obsahuje pri každom predmete akcie *detaily, editovať* a *zmazať*.

Detailná stránka o predmete poskytuje, okrem informácií uvedených v zozname, navyše prehľadný zoznam študentov, s možnosťou odpísania študenta z predmetu. Kliknutím na meno študenta v zozname sa administrátor dostane do sekcie študentov, priamo do informačnej stránky o zvolenom študentovi. Priamo z tejto stránky sa dá dostať k formuláru pre editovanie predmetu a je tu možnosť vymazania predmetu z databázy.

Pridávanie a editovanie predmetu prebieha prostredníctvom toho istého formuláru, ktorý obsahuje tieto položky:

1. *Názov* – predmetu, ktorý musí byť jedinečný a musí byť vyplnený
2. *Kód* – predmetu, nie je povinný údaj, pomáha pri lepšej orientácií a vyhľadávaní v systéme
3. *Popis* – informácie o predmete, ktoré sa zobrazia študentom, učiteľ ich môže vo svojom rozhraní meniť, je to nepovinný údaj a môže ostať nevyplnený
4. *Učiteľ* – zo zoznamu učiteľov sa vyberie jeden, ktorému bude predmet v systéme pridelený, učiteľ sa nemusí vybrať, ale pokiaľ nebude pridelený učiteľ, nebude možné využívať predmet pri skúšaní študentov

V prípade nesprávneho vyplnenia formuláru, teda ak nie je zadaný názov predmetu, alebo názov predmetu koliduje s iným názvom predmetu, ktorý sa nachádza v databáze, je administrátor upozornený na chybu, už vyplnené údaje ale nezmiznú.

Vymazanie predmetu z databázy vyžaduje potvrdenie, pre prípad mylného kliknutia na akciu *zmazať*.

Tak ako u študentov a učiteľov, aj predmetov môže byť väčšie množstvo, preto je dobrým nástrojom pre rýchlejšie získanie informácií o požadovanom predmete formulár s filtrom pre vyhľadávanie v databáze predmetov. V ňom sa dá špecifikovať názov predmetu, kód predmetu alebo učiteľ predmetu alebo skupiny predmetov, ktoré správca požaduje. Po stlačení tlačítka *Hľadať* sa zobrazí zoznam predmetov, ktoré vyhovujú zadanému filtru.

1.1.7 Import

Pridávanie všetkých dát do systému manuálne by zabralo obrovské množstvo času a zvýšila by sa pravdepodobnosť chýb pri zadávaní údajov. Preto je 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 na výmenu dát, ktorý bude popísaný v ďalšej stati.

Sekcia administrátorského rozhrania, pre import, obsahuje jednoduchý návod ako správne importovať dáta a obsahuje odkaz na stránku s nastaveniami pre správnu komunikáciu s externým systémom. Na tejto stránke sa nachádzajú aj samotné akcie pre vykonanie importu.

Po kliknutí na akciu *Importovať užívateľov* alebo *Importovať predmety,* sa pokúsi systém nadviazať kontakt s externým serverom, ktorý poskytuje službu pre import dát, ak autentifikácia prebehne v poriadku, získajú sa automaticky dáta a uložia sa do databázy. O výsledku systém oboznámi administrátora buď oznamom o úspešnom importovaní, alebo chybovou hláškou, v prípade, že sa nepodarilo získať dáta. Žiadne dáta sa neukladajú duplicitne, znamená to, že po viacnásobnom importe, tých istých dát sa nezvýši množstvo dát v databáze.

1.1.8 Záloha

Pri návrhu systému, bolo 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 su na ne priamo naviazané, napríklad testy a odpovede na ne. V tejto sekcií je možné vykonať tri akcie, podstránka obsahuje aj informácie o tom, ako správne postupovať pri zálohe dát.

Akcia *Zahájenie nového ročníka* sa vykonáva na začiatku nového školského roka, jej vykonaním sa zrušia staré väzby medzi predmetmi a študentami, zvýši sa rok štúdia každého študenta a vymažú sa z databázy dáta, ktoré sú príliš staré. Dôležité je, aby sa táto akcia nemohla omylom vykonať niekoľko krát po sebe, napríklad opakovaným načítaním stránky. Preto pred vykonaním tejto akcie je potrebné opísať náhodne vygenerovaný reťazec šiestich znakov. Po zahájení nového ročníka by mal nasledovať import dát, čím sa vytvoria nové väzby medzi študentami a predmetmi.

Záloha databázy je akcia, ktorá uloží do súboru dáta z databázy, ktoré môžu byť neskôr opätovne načítané do systému. Zálohujú sa iba údaje, ktoré sú ovplyvnené akciou *Zahájenie nového ročníka.* Na stránke je upozornenie, že je vhodné vykonať zálohu dát pred každým zahájením nového ročníka.

Načítanie zo zálohy zoberie zálohu označenú zo zoznamu doteraz urobených záloh a načíta ju do databázy. Tieto údaje nemôžu byť aktívne využívané, využívajú sa pri prehľadoch a štatistikách o skúškach a testoch. Keďže načítanie väčšieho množstva dát, môže databázu zahltiť, musí byť v systéme možnosť ich opätovného odstránenia. To sa vykoná po stlačení tlačítka *Odstrániť zálohu z db.*

1.1.9 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, nachádza aj formulár pre zmenu prístupového hesla do administrátorskej časti.

Pri menení hesla, musí užívateľ vyplniť staré heslo a zadať nové, ktoré následne zopakuje. Ak staré heslo nie je správne zadané, alebo je nové heslo zle zopakované, systém užívateľa upozorní na chybu pri pokuse o zmenu hesla. V opačnom prípade prebehne zmena hesla.

Nastavenie pre import dát sa vykonávajú na samostatnej podstránke. Administrátor musí vyplniť URL systému, ktorý poskytuje službu, identifikátor, ktorý mu bol externým systémom pridelený a musí zadať verejný kľúč a modulus, ktorým bude šifrovaná komunikácia medzi oboma systémami.

1.2 Rozhranie pre učiteľa

Rozhranie pre učiteľa má formu web aplikácie, ktorá je prístupná z ľubovoľného miesta na internete. Učiteľovi sa po prihlásení a autentifikovaní do tejto časti systému zobrazí úvodná stránka, na ktorej je výpis aktuálnych udalostí. Udalosťou môže byť oznam o budúcej alebo odkaz na práve prebiehajúcu skúšku. Pri zozname budúcich skúšok je odkaz, ktorým sa skúška spúšťa.

Dôležitou sekciou učiteľského rozhrania je sekcia *predmety*. Hlavná stránka tejto sekcie obsahuje zoznam predmetov, ktoré má v systéme učiteľ pridelené. Kliknutím na *detaily* pri názve predmetu sa učiteľ dostane do časti, v ktorej môže vytvárať sady otázok a odpovedí k vybratému predmetu, zaraďovať ich do okruhov tém a následne z nich generovať testy. Každému predmetu môže priradiť stupnicu hodnotenia a vytvoriť *demo test*. Prácu učiteľovi uľahčujú nástroje pre vyhľadávanie v sadách odpovedí, otázkach, okruhoch tém alebo študentoch. Je možné zobraziť štatistiku o predmete, ktorá ponúka prehľad o doterajších výsledkoch.

Z hlavnej stránky rozhrania sa dá dostať do sekcie *proktori,* kde si učiteľ vytvára proktorov, teda zoznam osôb, ktoré môžu dohliadať na priebeh skúšky. Samozrejmosťou je možnosť zmeny prihlasovacieho hesla.

1.3 Rozhranie pre študenta

Študenti do systému vstupujú cez rozhranie pre študenta. Tak ako predchádzjúce dve rozhrania, aj toto je riešené formou web aplikácie. Po prihlásení sa, sa zobrazí študentovi zoznam predmetov, kliknutím na ktoré sa načíta stránka s informáciami o vybratom predmete. Na stránke s detailami o predmete sa študentovi zobrazujú informácie o predmete, ktoré vyplnil učiteľ vo svojom rozhraní. Ukazuje sa tu popis predmetu, stupnica, spôsob hodnotenia a termíny skúšok. Kliknutím na práve prebiehajúcu skúšku, sa spustí testovanie študenta. Počas skúšky sa pre zamedzenie podvádzania, študent nemôže odhlásiť zo systému, pre dočasné zablokovanie ale môže využiť akciu *lock-out* ktorá jeho testovanie dočasne pozastaví a obnovené bude prostredníctvom proktora, ktorý si môže

overiť, či študent, ktorý test začal vypracovávať je ten, ktorý sa pokúša odblokovať prebiehajúci test. Pri vypracovávaní testu, sa študentovi zobrazuje zoznam otázok, každú otázku si môže pre svoju potrebu označiť ako vyriešenú, prečítanú, zodpovedanú alebo rozpracovanú. Klikaním na otázky, sa študentovi zobrazuje zoznam možných odpovedí na konkrétnu otázku. Študent môže ku každej otázke pridať vlastný komentár, ktorý sa zobrazí učiteľovi po vyhodnotení testu. Súčasťou testu je aj časomiera, ktorá zobrazuje čas strávený vypĺňaním testu. Dáta sa v pravidelných intervaloch synchronizujú so serverom, čím sa predchádza ich možnej strate pre prípad výpadku siete alebo iných neočakávaných okolností. Po odoslaní testu, sa skúška pre študenta končí. Test sa vyhodnotí a študentovi sa zobrazia správne odpovede, ak to učiteľ povolil v nastaveniach testu.

1.4 Databázový model

Po spísaní špecifikácie, bolo ďalším krokom pri vývoji aplikácie navrhnutie databázového modelu, podľa ktorého sa pri implementovaní aplikácie vytvorila databáza. Databázový model bol vytvorený v UML a bol nezávislý od akéhokoľvek konkrétneho databázového systému. Vizualizácia modelu predstavovala graf, kde vrcholmi boli jednotlivé tabuľky systému, ktoré obsahovali zoznam stĺpcov, pričom zvýraznené boli *primary keys* a *foreign keys*. Vzťah medzi tabuľkami reprezentovala orientovaná hrana grafu.

Výsledný model zahŕňal 22 tabuliek, pričom do modelu neboli zahrnuté tabuľky, ktoré boli od tabuliek v modeli nezávislé: *tSession* a *tSettings*. Model by sa dal rozdeliť na tri väčšie celky, ktoré sú navzájom poprepájané. Tabuľky viažúce sa na predmety, užívateľov a na testy a skúšky.

1.4.1 Štruktúra predmetov

Tabuľka predmetov je jedným z centrálnych bodov databázového modelu, sú na ňu naviazané tabuľky s okruhom tém, otázkami a odpoveďami, z ktorých sa pre daný predmet generujú testy a každý predmet má priradenú stupnicu hodnotenia. Tabuľka predmetov je zároveň bodom, kde sa prepájajú všetky tri väčšie celky databázového modelu.

Tabuľka tCourses

Predmet musí mať svoje jednoznačné *id.* Každému predmetu je pridelená stupnica hodnotenia, učiteľ môže vytvárať demo test, užívateľom musí byť poskytnutý prehľad o názve a popise predmetu, pre generovanie testov je dôležitá informácia o počte otázok, ktoré sa majú v teste pre daný predmet vygenerovať a dĺžka trvania testu. Všetky tieto vlastnosti predmetu musia byť zohľadnené pri vytváraní tabuľky *tCourses*, ktorá obsahuje 8 stĺpcov:

1. *id* – id predmetu, primárny kľúč
2. *id_scale* – foreign key do tabuľky *tScale*, viaže predmet so stupnicou hodnotenia
3. *id_demoTest* – foreign key do tabuľky *tTest*
4. *name* – názov predmetu
5. *description* – popis predmetu
6. *code* – kód predmetu
7. *questionCount* – počet otázok generovaných v teste
8. *duration* – trvanie testu v minútach

Tabuľka tScale

Pri vyhodnocovaní testu, musí byť študentovi priradená výsledná známka. Každý predmet môže mať inú stupnicu hodnotenia. V tabuľke *tScale* je, okrem požadovaného počtu bodov na dosiahnutie určitej známky, uvedený aj spôsob hodnotenia testu, čiže počet bodov získaných za jednu odpoveď. Pri vyhodnocovaní sa odpovede delia do štyroch typov. Správna odpoveď je vtedy, keď študent vyberie všetky správne možnosti a žiadnu nesprávnu, čiastočne správna odpoveď je vybratie nie všetkých správnych, ale žiadnej nesprávnej. Zlá odpoveď je vtedy, keď sa v odpovedi nachádza chybná možnosť a prázdna odpoveď je odpoveď bez označenia možných odpovedí. Tabuľka *tScale* má 10 stĺpcov:

1. *id* – id stupnice, primárny kľúč
2. *ptsFull* – počet bodov za správnu odpoveď

3. *ptsEmpty* – počet bodov za prázdnu odpoveď
4. *ptsPartial* – počet bodov za čiastočne zodpovedanú odpoveď
5. *ptsBad* - počet bodov za nesprávne zodpovedanú odpoveď
6. *A* – počet bodov potrebných na získanie známky A
7. *B* – počet bodov potrebných na získanie známky B
8. *C* – počet bodov potrebných na získanie známky C
9. *D* – počet bodov potrebných na získanie známky D
10. *E* – počet bodov potrebných na získanie známky E

Tabuľka tDomain

Učiteľ vytvára otázky, ktoré sú zaraďované do okruhov tém. Tabuľka *tDomain* uchováva informácie o okruhoch tém pre jednotlivé predmety, okruhy tém majú hierarchickú štruktúru, čo znamená, že okruh tém môže mať väčší počet podokruhov. Relácia medzi tabuľkami *tDomain* a *tCourse* vyjadruje prepojenie na základe foreign key *id_course* formou many-to-one. Otázky, odpovede, okruhy tém a testy sa môžu stať časom *obsolete*, čiže zastarané, takýto prípad nastane vtedy, keď ich učiteľ nechce viac používať pre generovanie testov, ale musia ostať v databáze, kvôli predchádzajúcim skúškam. Tabuľka *tDomain* má 6 stĺpcov:

1. *id* – id okruhu tém, primárny kľúč
2. *id_parent* – id nadokruhu tém, pod ktorý daný okruh tém patrí
3. *id_course* – id predmetu, pre ktorý je okruh tém vytvorený
4. *subject* – názov okruhu tém
5. *description* – popis okruhu tém
6. *obsolete* – označuje, či je okruh tém zastaraný

Tabuľka tCourseToDomain

Pri generovaní testov sa vyberajú otázky zo sád odpovedí. Učiteľ pred samotným generovaním nastaví, minimálny a maximálny počet otázok, ktoré sa vo vygenerovanom teste majú z konkrétneho okruhu tém objaviť. Tieto informácie sú uložené v tabuľke *tCourseToDomain,* ktorá ma päť stĺpcov:

1. *id_course* – id predmetu z tabuľky *tCourse*
2. *id_domain* – identifikátor okruhu tém, pre ktorý budú platiť kvantifikačné obmedzenia
3. *min* – minimálny počet otázok, ktoré sa majú z daného okruhu tém vybrať do testu
4. *max* – maximálny počet otázok
5. *is_NULL* – ak je označený, indikuje, že sa z okruhu tém, nemajú vyberať žiadne otázky

Tabuľka tQuestion

Testy pozostávajú z otázok, ktoré sú zaraďované do okruhov tém, z ktorých sa testy generujú. Každá otázka musí byť formulovaná, teda musí obsahovať svoje znenie. Tabuľka *tQuestion* má 4 stĺpce:

1. *id* - id otázky, primárny kľúč
2. *id_domain* – foreign key, id domény pod ktorú otázka patrí
3. *question* – znenie otázky
4. *obsolete* – označuje, či je otázka zastaraná

Tabuľka tAnswerSet

Každej otázke môže prislúchať viac množín odpovedí, ktoré majú nastavenú rôznu obtiažnosť. Tabuľka *tAnswerSet* prepája otázku a sady odpovedí, formou one-to-many. Tabuľka má 4 stĺpce:

1. *id* – id sady odpovedí, primárny kľúč
2. *id_question* – foreign key, prepája sadu odpovedí s otázkou
3. *difficulty* – obtiažnosť sady odpovedí
4. *obsolete* – určuje zastaranosť sady odpovedí

Tabuľka tAnswer

Sadu odpovedí tvoria odpovede, tie sú uložené v tabuľke *tAnswer*. Každá odpoveď pozostáva zo svojho znenia a z pravdivosti odpovede na danú otázku. *tAnswer* pozostáva zo 4 stĺpcov:

1. *id* – id otázky, primárny kľúč
2. *id_answerSet* – id sady odpovedí, do ktorej patrí odpoveď
3. *text* – znenie odpovede
4. *correct* – určuje správnosť odpovede

1.4.2 Štruktúra užívateľov

Podobne ako tabuľka predmetov, je jedným z hlavných uzlov databázového modelu aj tabuľka užívateľov. Pod užívateľmi sa myslia študenti, učitelia, administrátori alebo proktori. Študenti potrebujú mať na rozdiel od ostatných typov užívateľov v databáze uchované údaje navyše. Ide o rok štúdia a študijný odbor. Každý učiteľ môže mať vytvorený vlastný zoznam proktorov, ktorí môžu dohliadať na priebeh skúšky. Potrebné sú aj tabuľky pre prepojenie študentov a učiteľov s predmetmi a tabuľkou so skúškami.

Tabuľka tUser

Spoločné údaje všetkých štyroch typov užívateľov sú uložené v tabuľke *tUser*. Každý užívateľ, ktorý využíva systém, musí mať prihlasovacie meno a heslo, ktorým sa autentifikuje pri prihlasovaní. V tejto tabuľke je uložené aj meno užívateľov, ktoré je rozdelené na štyri časti, meno, priezvisko a tituly pred a za menom. Toto rozdelenie vzniklo kvôli lepšej práci so záznamami v tabuľke, hlavne pri triedení a vyhľadávaní užívateľov. Študenti a učitelia majú pridelené ISIC respektíve ITIC číslo, ktoré je pre každého z nich

jedinečné a slúži ako jednoznačný identifikátor pri importovaní dát. Výsledná tabuľka *tUser* má 9 stĺpcov:

1. *id* – id užívateľa v databáze, primárny kľúč
2. *namePrefix* – prefix, pred menom užívateľa, akademické tituly, píšuce sa pred menom
3. *name* – meno užívateľa
4. *surname* – priezvisko užívateľa
5. *nameSuffix* – akademické tituly, ktoré sa píšu za menom
6. *login* – prihlasovacie meno
7. *password* – 32 znakový *md5 hash* prihlasovacieho hesla s náhodne zvoleným prefixom
8. *passPrefix* – postupnosť šiestich náhodných znakov, náhodne vygenerovaná pri vytváraní užívateľa
9. *ISIC* – číslo z identifikačnej karty študenta alebo učiteľa

Tabuľky tStudents, tTeachers, tAdmins, tProctors

Pre identifikovanie typu užívateľa, boli vytvorené tabuľky *tStudents, tTeachers, tAdmins, tProctors* ktoré, až na tabuľku *tStudents,* obsahujú jediný stĺpec a to *id* z tabuľky *tUser.*

Tabuľka *tStudents* navyše obsahuje stĺpce *id_specialization* id študijného odboru z tabuľky *tSpecialization,* ktorá bude popísaná nižšie. Stĺpec *year* uchovávajúci informáciu o roku štúdia študenta a pomocný stĺpec *bkp,* ktorý sa využíva pri nahrávaní a vymazávaní zálohy z databázy.

Tabuľka tCourseToUser

Každý študent môže mať v systéme priradených viac predmetov a každý predmet má viac študentov. Podobne každý učiteľ môže mať v systéme viac predmetov. Tabuľka *tCourseToUser* predstavuje reláciu many-to-many medzi tabuľkami *tUser* a *tCourse*. Obsahuje dva stĺpce:

1. *id_course* – id predmetu z tabuľky *tCourse*

2. *id_user* – id užívateľa z tabuľky *tUser*

Tabuľka tProctorToTeacher

Učitelia si môžu vytvárať proktorov, ktorý budú mať právomoc riadiť priebeh skúšky. Každý učiteľ môže mať vytvorených niekoľko proktorov. Tabuľka *tProctorToTeacher* má dva stĺpce:

1. *id_proctor* – id proktora z tabuľky *tUser*
2. *id_teacher* – id učiteľa z tabuľky *tUser*

Tabuľka tSpecialization

Študenti sú pre lepšiu navigáciu v administrátorskom rozhraní zaradený do študijných odborov, každý odbor má svoj názov a môže mať prideleného garanta. *tSpecialization* má tri stĺpce:

1. *id* – id študijného odboru, primárny kľúč
2. *name* – názov študijného odboru
3. *id_garant* – id garanta študijného odboru z tabuľky *tUser*

1.4.3 Štruktúra testov a skúšok

Testovanie študentov a priebeh skúšky vyžadujú, aby sa v databáze uchovávali údaje o skúškach a testoch. Každá skúška má zoznam študentov, ktorí sa na skúšku prihlásili a môžu ju absolvovať. Študentom sú pridelené testy, na ktoré odpovedajú. Každý test je definovaný sadou otázok a odpovedí, tvorí šablónu, podľa ktorej sa vytvárajú záznamy v databáze pre jednotlivé študentmi vypracované testy. Učiteľ alebo študent má možnosť pridať osobnú poznámku k testu, odpovedi alebo celému termínu. Toto všetko muselo byť zohľadnené pri navrhovaní časti databázového modelu, týkajúceho sa časti o testoch a skúškach.

Tabuľka tExam

Testovanie študentov začína tým, že sa študenti prihlásia na skúšku, následne pri spustení skúšky, sú im pridelené testy. Tabuľka *tExam* je prepojovací uzol databázového modelu, medzi užívateľmi, predmetmi a vypracovanými testami. Každá skúška má pridelený jednoznačný identifikátor a je priradená zvolenému predmetu. Vypracované testy sú potom naviazané na skúšku, počas ktorej boli vypracovávané. Tabuľka *tExam* má päť stĺpcov:

1. *id* – identifikátor skúšky, ktorý je primárnym kľúčom tabuľky
2. *id_course* – id predmetu z tabuľky *tCourse*
3. *date* – dátum konania skúšky, využíva sa najmä pri importovaní zoznamu skúšok pre zvolený predmet
4. *state* – skúška môže byť v troch stavoch: *scheduled* – skúška čaká na spustenie, *progress* – skúška práve prebieha, *finished* – skúška bola ukončená
5. *comment* – učiteľ môže po skončení skúšky napísať všeobecný komentár o výsledkoch skúšky

Tabuľka tExamToUser

Počet študentov, ktorí môžu absolvovať skúšku je variabilný, preto musí byť ich zoznam nezávislý od tabuľky *tExam.* Informácia o študentoch sa uchováva v tabuľke *tExamToUser,* ktorá ma dva stĺpce:

1. *id_exam* – identifikátor skúšky z tabuľky *tExam*
2. *id_user* – identifikátor užívateľa z tabuľky *tUser*

Tabuľka tTestTemplate

Bolo by neefektívne z hľadiska uloženia dát, každý vypracovávaný test, ukladať do databázy spolu s jeho štruktúrou. Vygenerované testy sa ukladajú do tabuliek *tTestTempalte* a *tTestItems*. Tieto tabuľky tvoria šablónu, podľa ktorej sa pri riešení testu študentom zobrazujú otázky, na ktoré odpovedajú. Podobne ako v časti s otázkami

a odpoveďami, aj test môže byť označený učiteľom ako *obsolete.* V tabuľke sa zároveň uchováva informácia o tom, či bol test vytlačený. *tTestTemplate* má štyri stĺpce:

1. *id* – identifikátor testu, primárny kľúč
2. *id_course* – identifikátor predmetu, pre ktorý bol test vygenerovaný
3. *printed* – určuje, či bol test použitý v papierovej forme
4. *obsolete* – ak hodnota nie je 0, označuje test ako zastaraný

Tabuľka tTestItems

Rôzne testy pozostávaju z rôzneho počtu otázok, vytvorenie dostatočne veľkého počtu stĺpcov v tabuľke *tTestTemplate,* ktoré by obsahovali id otázok by bolo neefektívne a zároveň by bol každý test limitovaný maximálnym počtom otázok, ktorý by nemohol byť v systéme prekročený. Uchovávanie identifikátorov v jednom stĺpci tabuľky *tTestTemplate* v podobe poľa alebo vopred definovanej štruktúre, by zase malo negatívny dopad na výpočtovú efektivitu. Riešením tohto problému je tabuľka *tTestItems,* ktorá reprezentuje reláciu many-to-one a uchováva informácie o otázkach s konkrétnou sadou odpovedí, ktoré boli vygenerované pre konkrétny test. Tabuľka *tTestItems* má dva stĺpce:

1. *id_testTemplate* – identifikátor testu z tabuľky *tTestTemplate*
2. *id_answerSet* – identifikátor sady odpovedí, ktorý zároveň jednoznačne určuje otázku

Tabuľka tTest

Konkrétny test musí byť identifikovateľný tým, kto ho vypĺňal, na akej skúške bol riešený, a ku ktorému z vygenerovaných testov patrí. Učiteľ by mal mať možnosť po vyhodnotení pridať poznámku k testu, ktorá by dala dodatočnú informáciu študentovi o výsledku. Vyhodnotenému testu sa pridelí známka a uchová sa pre prípad že by sa časom zmenila stupnica, táto hodnota musí byť statická. *tTest* obsahuje osem stĺpcov:

1. *id* – identifikátor testu, primárny kľúč
2. *id_exam* – id skúšky, počas ktorej bol test riešený

3. *id_testTemplate* – id vzoru testu z tabuľky *tTestTemplate*
4. *id_user* – id študenta, ktorý test vypracovával
5. *comment* – poznámka k testu
6. *totalTime* – celková doba riešenia testu
7. *mark* – výsledná známka
8. *points* – počet bodov, získaných za test

Tabuľka tTestAnswer

Pri vypĺňaní testu študent postupne odpovedá na otázky, ktoré sa mu v jeho rozhraní zobrazia. Po odoslaní sa každá odpoveď automaticky vyhodnotí a oboduje príslušným počtom bodov, na základe aktuálnej stupnice predmetu. Dôležitá vlastnosť testu je, že poradie otázok je každému študentovi pridelené náhodne, preto je v tabuľke dôležitý stĺpec *index,* ktorý určuje, na ktorú otázku študent odpovedal. K jednotlivým odpovediam môže študent pridať svoju poznámku pre učiteľa. Systém zároveň automaticky počíta čas, ktorý študent strávil pri danej otázke. Tabuľka *tTestAnswer* pozostáva z ôsmich stĺpcov:

1. *id* – identifikátor študentovej odpovede
2. *id_test* – test ku ktorému odpoveď patrí
3. *comment* – poznámka od študenta
4. *totalTime* – celkový čas strávený riešením otázky
5. *index* – poradie otázky v teste
6. *points* – počet bodov získaných za odpoveď
7. *answerType* – typ odpovede (full, missing, partial, bad)
8. *answerMark* – pomocná hodnota, pre zvýšenie prehľadu študenta o danom teste

Tabuľka tTestAnswerToAnswer

Táto tabuľka prepája jednotlivé odpovede študenta s konkrétnymi odpoveďami v tabuľke *tAnswer*. Zároveň určuje, či študent možnú odpoveď označil za správnu alebo nie. *tTestAnswerToAnswer* má štyri stĺpce:

1. *id_testAnswer* – identifikátor odpovede v konkrétnom teste
2. *id_answer* – identifikátor odpovede vytvorenej učiteľom
3. *value* – určuje, či študent odpoveď označil ako pravdivú
4. *index* – poradie otázky v teste

1.4.4 Ostatné tabuľky

V databáze systému sú vytvorené ďalšie dve tabuľky, ktoré neboli zahrnuté v pôvodnom databázovom modely, pretože s tabuľkami v ňom nie sú v žiadnej relácií. Prvou z nich je tabuľka *tSession* do ktorej sa ukladajú dáta z jednotlivých session, ktoré práve prebiehajú v systéme. Druhou tabuľkou je tabuľka *tImportData,* kde sa uchovávajú niektoré systémové nastavenia.

2. Protokol pre výmenu dát

Systém pre online testovanie študentov potrebuje pre svoj beh údaje o študentoch, učiteľoch, predmetoch a skúškach. Požiadavkou bolo, aby bol systém nezávislý na už existujúcich databázach, ktoré obsahujú tieto dáta a mohol byť inštalovaný do ľubovoľného systému, ktorý podporuje technológie, ktoré aplikácia využíva pre svoj beh. Bolo sa preto potrebné zaoberať otázkou, ako tieto údaje preniesť do databázy aplikácie. Tá podporuje manuálne pridávanie dát do databázy, ale v praxi ide o veľké množstvo informácií, ktoré by bolo nutné prácne zadávať v administrátorskom rozhraní systému. Navyše by sa takýmto manuálnym pridávaním zvýšila pravdepodobnosť nesprávne zadaných údajov. Riešením je vytvorenie protokolu, ktorým by aplikácia komunikovala s externým systémom, ktorý tieto údaje má uložené vo svojej databáze. Protokol by nemal byť zložitý, aby bolo jednoduché implementovať ho v už existujúcich systémoch, akým je napríklad systém pre prihlasovanie na skúšky, bežiaci na webe Fakulty Matematiky, Fyziky a Informatiky Univerzity Komenského. Údaje sa v takom prípade budú prenášať sieťou, je preto potrebné zabezpečiť, aby boli informácie získané z externého systému, prístupné iba pre autentifikovaného užívateľa, ktorým je v tomto prípade systém Probatur.

2.1 Štruktúra dát

Prenášané dáta musia byť vo formáte, ktorému by rozumeli obidve komunikujúce strany. Najvýhodnejšou formou je prenos dát vo formáte *XML* (Extensible Markup Language) a to z niekoľkých dôvodov. XML je štandardizovaný formát, vyvinutý organizáciou World Wide Web Consortium, pre zdieľanie dát medzi rozdielnymi informačnými systémami. Je veľmi rozšírený a existuje podpora pre parsovanie a vytváranie xml súborov v takmer všetkých programovacích jazykoch. XML je *markup language* a štruktúra dát je syntakticky a môže byť sémanticky definovaná napríklad pomocu *DTD* (Document Type Definiton). Pozitívom je aj fakt, že XML podporuje kódovanie UTF-8. Berúc do úvahy všetky výhody spojené s XML som sa rozhodol, že protokol bude využívať práve tento formát.

2.2 Autentifikácia

Aplikácia, žiadajúca si dáta, sa musí systému, ktorý ich poskytuje autentifikovať. Autentifikácia prebieha počas zaslania každej požiadavky na systém a pozostáva v overení jednoznačného identifikátora žiadateľa, ktorý mu systém poskytujúci službu pridelil. Otázkou je, prečo sa musí systém pre testovanie študentov autentifikovať pomocou identifikátora. Predstavme si centrálnu databázu študentov a učiteľov veľkej univerzity. Niektoré fakulty môžu využívať systém pre online testovanie študentov. Pri importe musí systém nad centrálnou databázou študentov vedieť rozlíšiť, ktoré dáta posielať.

Pre vyššiu bezpečnosť sú prenášané dáta šifrované pomocou symetrickej blokovej šifry *AES* (Advanced Encryption Standard), použitím 256bitov dlhého kľúča. Súčasťou autentifikácie musí byť aj výmena kľúča medzi oboma systémami, tento kľúč ale musí byť prenášaný v šifrovanej podobe, inak by šifrovanie dát stratilo zmysel, keďže by ich mohol hocikto ľahko dešiforvať.

2.3 Formát žiadosti

Protokol pre výmenu dát je definovaný nad protokolom HTTP (Hyper Text Transfer Protocol). Požiadavky na server sa posielajú formou príkazu POST a parametre musia mať nasledujúcu štruktúru.

data – verejným kľúčom servera zašifrované zreťazenie md5 hash-u náhodného šesť znakového prefixu s identifikátorom žiadateľa s 256bitovým náhodne vygenerovaným kľúčom pre následnú komunikáciu.

action – príkaz špecifikujúci o ktoré dáta má žiadajúca strana záujem. Podporované príkazy sú *users, courses, exams*, ktoré budú popísané nižšie

param1 – parameter príkazu, ktorý nemusí byť použitý pri každom type príkazu

iv – náhodný 256bitový inicializačný vektor pre algoritmus AES

idp – náhodne zvolený 6 znakový reťazec, ktorý sa používa ako prefix v hash-i identifikátora v parametri *data*

2.4 Formát odpovede

Ak server úspešne autentifikuje žiadateľa o dáta, pošle mu naspäť dáta vo forme *xml,* o ktoré mala žiadajúca strana záujem. V opačnom prípade pošle odpoveď, kde do HTTP hlavičky vloží kód *403 Forbidden.*

2.5 Príkaz users

Príkazom *users* sa od servera žiada zoznam študentov a učiteľov. Príkaz nepotrebuje žiaden parameter navyše. Odpoveď vo formáte xml musí sémanticky spĺňať kritériá definované v *users.dtd* a ktorého štruktúra je nasledovná:

```
<!ELEMENT users              (teachers,specialization*)>
<!ELEMENT teachers           (teacher*)>
<!ELEMENT teacher            (namePrefix?,name,surname,nameSuffix?,ISIC)>
<!ELEMENT specialization     (name,garant,students)>
<!ELEMENT students           (student*)>
<!ELEMENT student            (namePrefix?,name,surname,nameSuffix?,ISIC,year)>
<!ELEMENT namePrefix         (#PCDATA)>
<!ELEMENT name               (#PCDATA)>
<!ELEMENT surname            (#PCDATA)>
<!ELEMENT nameSuffix         (#PCDATA)>
<!ELEMENT ISIC               (#PCDATA)>
<!ELEMENT year               (#PCDATA)>
<!ELEMENT garant             (#PCDATA)>
```

Tým je jednoznačne určená štruktúra dát. V obdržanom xml súbore je najskôr uvedený zoznam učiteľov, ktorý môže byť prázdny. Každý učiteľ musí byť popísaný prefixom mena, teda titulmi, ktoré sa píšu pred menom, *namePrefix* sa v elemente *teacher* môže vyskytovať práve raz alebo ani raz. Nasledujú elementy *name* a *surname,* ktoré sú povinné a musia sa vyskytovať v elemente *teacher* práve raz, nasleduje *nameSuffix,* tituly, ktoré ss píšu za menom a povinný údaj ISIC.

Za zoznamom učiteľov nasleduje zoznam študentov, ktorý sú rozdelený podľa študijného odboru, ktorý študujú. Každý element *specialization*, ktorý sa v xml súbore nachádza musí obsahovať práve jeden element *name, garant* a *students*. Kde *name* je názov študijného

odboru, *garant* je identifikátor garanta štúdia a *students* je zoznam študentov, ktorí študujú daný študijný odbor. Element *students* môže obsahovať ľubovoľný, aj nulový, počet elementov *student,* ktorý je definovaný rovnako ako element *teacher* a navyše musí povinne obsahovať element *year* ktorý definuje rok štúdia.

Ako identifikátori sa volia pri študentoch a učiteľoch ISIC respektíve ITIC čísla, ktoré má každý jedinečné. Študijný odbor je definovaný svojím názvom.

2.6 Príkaz courses

Pre vyžiadanie zoznamu predmetov spolu so zoznamom študentov, ktorí majú jednotlivé predmety zapísané, bol definovaný príkaz *courses*. Štruktúra odpovede musí byť validná vzhľadom na obmedzenia definované v *courses.dtd*:

```
<!ELEMENT courses          (course*)>
<!ELEMENT course           (label,code,description,teacher,students)>
<!ELEMENT teacher          (id)>
<!ELEMENT students         (id*)>
<!ELEMENT id               (#PCDATA)>
<!ELEMENT label            (#PCDATA)>
<!ELEMENT code             (#PCDATA)>
<!ELEMENT description      (#PCDATA)>
```

Odpoveď na príkaz *courses* teda pozostáva zo zoznamu predmetov, ktorých môže byť ľubovoľne veľa. Každý predmet, definovaný elementom *course,* musí povinne obsahovať práve jeden element *label, code, description, teacher, students*. Kde element *label* jednoznačne určuje predmet na základe jeho názvu, element *code* je kód predmetu, *description* je stručná charakteristika predmetu, *teacher* je identifikátor učiteľa a element *students* obsahuje zoznam študentov, ktorý môže byť aj prázdny. Zoznam tvoria elementy *id,* čo sú identifikátori študentov vo forme ich ISIC čísiel.

2.7 Príkaz exams

Prihlasovanie na skúšky prebieha nezávisle od aplikácie, preto je potrebné do protokolu o výmene dát zahrnúť aj príkaz *exams,* ktorý vráti učiteľovi zoznam skúšok spolu so

zoznamom študentov, ktorí sú na dané skúšky zapísaný. Štruktúra obdržaných dát musí spĺňať nasledujúce obmedzenia:

```
<!ELEMENT exams                 (exam*)>
<!ELEMENT exam                  (courseName,date,students)>
<!ELEMENT courseName            (#PCDATA)>
<!ELEMENT date                  (#PCDATA)>
<!ELEMENT students              (id*)>
<!ELEMENT id                    (#PCDATA)>
```

Prijaté dáta majú teda podobu zoznamu skúšok, reprezentovaných elementom *exam.* Zoznam skúšok môže byť aj prázdny alebo môže obsahovať ľubovoľný počet elementov. Jednotlivé skúšky pozostávajú z elementov *courseName, date* a *students.* Kde *courseName* je názov predmetu ktorého sa skúška týka, *date* je dátum vo formáte *unix timestamp*, ktorý reprezentuje dátum ako počet sekúnd uplynutých od polnoci 1.januára 1970. Element *students* obsahuje zoznam identifikátorov študentov, ktorí sú prihlásený na skúšku, v podobe ich ISIC čísla.

3. Vývoj aplikácie

Vytvorením špecifikácie, databázového modelu a definovaním protokolu pre výmenu dát sa skončila fáza návrhu a analýzy aplikácie. Pred jej samotným implementovaním bolo potrebné zvoliť technológie, ktoré sa pri vývoji použijú a ktoré bude aplikácia používať. Rovnako dôležité bolo vytvorenie frameworku, nad ktorým by sa jednotlivé časti systému implementovali.

3.1 Výber technológií

Aplikácia by mala byť prístupná z internetu. Ako najvýhodnejšia sa ukázala voľba, spraviť aplikáciu v podobe HTML stránky, ktorej jednotlivé rozhrania by sa správne zobrazovali v ľubovoľnom internetovom prehliadači. Stránky preto musia byť validné. Pri definovaní výsledného vzhľadu sa použil *stylesheet language CSS* (Cascading Style Sheet). Samotná stránka spĺňa štandardy *XHTML 1.0.*

Ako skriptovací jazyk bol zvolený jazyk *PHP,* ktorý sme uprednostnili pred *ASP.NET* pre naše lepšie skúsenosti s týmto skriptovacím jazykom. Zároveň PHP poskytuje všetku funkcionalitu, ktorú sme pri programovaní aplikácie potrebovali. PHP je prístupné vo viacerých verziách, pre našu aplikáciu sme zvolili PHP verzie 5.0, ktoré v porovnaní so staršími verziami poskytuje lepšiu prácu s triedami. Samotný beh aplikácie si vyžaduje, aby boli na serveri, kde beží, nainštalované aj externé moduly PHP pre prácu s databázou *MySql* alebo *PostgreeSQL,* modul pre prácu so šifrovacími funkciami (*mcrypt*) a pre prácu s reťazcami v kódovaní *UTF-8.* (*mbstring*).

Niektoré časti systému používajú okrem server-side skriptov aj skripty, ktoré bežia na užívateľovej strane, ktoré sú implementované pomocou *Javascritp-u.*

Pri vývoji webovskej aplikácie je praktické, písať programovú výkonnú časť kódu tak, aby nezávisela od výsledného vzhľadu aplikácie. Táto vlastnosť sa dá dosiahnuť použitím šablónového systému. Znamená to, že dizajn a záverečný vzhľad stránky sa dá meniť bez toho, aby sa museli robiť zásahy do programovej časti kódu. Zvažovali sme pri výbere šablónového systému dve možnosti. Prvou bolo vytvorenie vlastného systému, čo by malo

tú nevýhodu, že by sa znížil čas práce na samotnom programe implementovaním nového šablónového systému a neskoršie zmeny by si od dizajnéra vyžadovali naštudovanie, pre neho nového systému. Druhou možnosťou, pre ktorú sme sa, po prihliadnutí na nedostatky prvej, rozhodli, je široko rozšírený šablónový systém *Smarty*.

Poslednou, ale nie vzhľadom na dôležitosť, voľbou bola voľba databázy pre uchovávanie dát systému. Tu sme sa priklonili k riešeniu vo forme databázovej medzivrstvy, ktorá oddelila prístup k fyzickej databáze od volaní z aplikácie. Medzivrstva dokáže pracovať s databázami *MySql* a *PostgreeSQL.*

3.2 Základné triedy

Aplikácia je rozdelená na tri väčšie celky. Implementáciu každého z nich, mal na starosti niekto iný. Každý celok je tvorený súborom niekoľkých menších celkov, stránok. Vytvoreniu spoločného rozhrania, na ktorom by boli jednotlivé celky vybudované sa musela venovať veľká pozornosť. Hlavná trieda, ktorá zabezpečuje beh aplikácie je *cApplication,* ktorú implementujú triedy *cAdminApplication, cTeacherApplication* a *cStudentApplication.* Tieto triedy zabezpečujú vykonávanie a zobrazovanie objektov triedy *cPage,* reprezentujúcich jednotlivé stránky aplikácie. Dôležitou triedou je aj trieda *cAbstractDB,* ktorá slúži ako medzivrstva medzi aplikáciou a databázou. Jej inštancia je vytvorená v súbore *index.php* a ako parameter sa predáva inštancií triedy *cApplication,* ktorá ju ďalej posúva svojim objektom.

K základným triedam patrí aj trieda *cExchange,* ktorá má implementované metódy pre komunikáciu s externým systémom pre import dát.

Triedy cAbstractDB, cMySqlDB, cPostgreeSqlDB

Volania funkcií konkrétneho databázového systému sú zabalené do tried *cMySqlDB* a *cPostgreeSglDB,* ktorých štruktúru definuje abstraktná trieda *cAbstractDB.* Táto má definované abstraktné metódy pre pripájanie sa k databázovému serveru, metódu pre posielanie príkazu databázovému systému a metódy vracajúce výsledok databázovej operácie ako jedno alebo viacrozmerné pole.

Trieda cApplication

Trieda *cApplication* je definovaná ako abstraktná, riadi výber a následné spracovanie a zobrazenie stránky. Stránku, ktorá je určená pre spracovanie, určuje parameter *site.* Ako parameter, dostáva pri inicializácií objekt triedy *cDB* a cestu k súborom potrebných pre beh jej potomkov. Konštruktor triedy inicializuje inštanciu objektu *Smarty* a zaháji session. Trieda má definovanú abstraktnú metódu *GetPageById,* ktorá vracia objekt typu *cPage* a následne v metóde *Run* volá jeho metódy, ktoré zabezpečia spracovanie a vykreslenie požadovanej stránky. Konkrétne sa zavolá metóda *Process,* ktorá vracia hodnotu *true* alebo *false*. Na základe výsledku volania metódy *Process* sa vykoná buď vykreslenie stránky volaním metódy *Draw* alebo sa vypíše chybová hláška volaním metódy *SayError,* triedy *cPage.*

Trieda cPage

Konkrétne stránky sú odvodené od rodičovskej triedy *cPage.* Tá obsahuje inštanciu triedy *cMenu*. Konštruktor triedy dostáva ako parametre inštancie tried *Smarty* a *cDB,* parameter *action*, na základe ktorého sa potomkovia triedy rozhodujú, akú akciu majú vo volaní metódy *Process* vykonať a polia premenných posielaných http metódou *GET* a *POST.* Trieda *cPage* je abstraktná a potomkovia musia mať implementované metódy *Process* a *Draw*, ktoré sú v tejto triede definované ako abstraktné.

Trieda cMenu

Každá časť systému má svoje menu. Trieda *cMenu* definuje štruktúru, od ktorej sú odvodení potomkovia tejto triedy pre jednotlivé rozhrania. Trieda *cMenu* obsahuje pole objektov typu *cMenuItem,* ktoré reprezentujú jednotlivé položky celého menu. Trieda definuje metódy pre pridávanie nových položiek do menu a pre získanie položiek podľa mena alebo indexu v poli.

Trieda cMenuItem

Položky v menu sú reprezentované triedou *cMenuItem*. Trieda obsahuje pole objektov *cMenuItem,* ktoré reprezentujú jednotlivé podpoložky, parametre *bSelected* a *bExpanded* určujú, či je položka označená a či sa zobrazujú jej podpoložky. Trieda obsahuje metódy

AddItem – pre pridávanie podpoložiek, *GetItemByLabel* a *GetItemByIndex,* ktoré vracajú podpoložky na základe ich mena, alebo indexu v poli.

Trieda cExchange

Medzi základné triedy, zdieľané administrátorským aj učiteľským rozhraním patrí aj trieda *cExchange.* V tejto triede sú implementované metódy pre pripojenie sa k externému systému, poskytujúcemu dáta pre import, metóda pre verifikáciu prijatých dát vo formáte *xml,* ktorá vráti hodnotu *true* alebo *false* podľa toho, či je štruktúra prijatých dát validná vzhľadom na ich *document type definiton*. Okrem toho obsahuje trieda verejné metódy *ImportCourses, ImportUsers* a *ImportExams*, ktoré naimportujú získané valídne dáta do databázy systému.

3.3 Triedy administrátorského rozhrania

Triedy opísané v tejto časti, sú potomkami niektorých zo základných tried. Patrí sem trieda *cAdminAppliation,* ktorá je potomkom triedy *cApplication,* implementuje jej abstraktnú metódu *GetPageById,* ktorá na základe paramtera *action* vytvorí inštanciu triedy, ktorá je potomkom základnej triedy *cPage* s ktorou potom v metóde *Run* pracuje. Tieto inštancie môžu byť typu *cStudentPage, cTeacherPage, cCoursePage, cImportPage, cBackupPage, cSettingsPage* alebo *cSpecializationPage.*

Triedy cUserPage, cStudentPage, cTeacherPage

Triedy *cStudentPage* a *cTeacherPage* majú veľa spoločných metód. Preto bola vytvorená rodičovská trieda *cUserPage* ktorá je potomkom triedy *cPage.* Táto trieda má implementované metódy, ktoré sa volajú zo študentskej a učiteľskej sekcie administrátorského rozhrania. Zahŕňa metódy pre kontrolu údajov zadaných užívateľom pri pridávaní alebo editovaní užívateľa v databáze, metódu *ChangePassword* ktorá mení heslo užívateľa a metódu *GetUserInfo,* ktorá vráti hodnoty z databázy, ktoré majú študenti aj učitelia rovnaké.

Trieda *cStudentPage* okrem metód pre pridávanie, editovanie a mazanie študenta v databáze, implementuje metódu *SearchResult,* ktorá vracia zoznam študentov,

vyhovujúcich zadanému filtru a preťažuje metódu rodičovskej triedy *VerifyUserValues,* ktorá kontroluje správnosť zadaných údajov o študentovi. Konštruktor triedy sa na základe *id* pokúsi vytvoriť inštanciu triedy *cStudent*, ak sa inštancia nepodarí vytvoriť, napríklad kvôli nesprávnemu *id,* niektoré metódy nebudú vykonané a vykoná sa predvolená akcia *list,* ktorá zobrazí zoznam študentov.

Podobne ako trieda *cStudentPage,* aj trieda *cTeacherPage* implementuje metódy pre pridávanie, editovanie, mazanie a vyhľadávanie učiteľov v databáze a vytvára inštanciu triedy *cTeacher*.

Trieda cCoursePage

Trieda *cCoursePage* zahŕňa všetky potrebné metódy pre sekciu predmetov administrátorského rozhrania. Pri inicializácia vytvára inštanciu triedy *cCourse* na základe *id*, ktoré získa ako *get* parameter. Ak sa inštancia nepodarí vytvoriť, podobne ako v triedach *cStudentPage* a *cTeacherPage* sa niektoré akcie nebudú môcť vykonať a nastaví sa prednastavená akcia *list.*

Trieda cSpecializationPage

Implementuje metódy pre spracovávanie a zobrazovanie informácií o štúdijných odoboroch. Vytvára inštanciu triedy *cSpecialization* na základe *id,* na ktorej v prípade úspešnej inicializácie vykonáva akcie volaním metód tejto triedy, podľa parametra *action.*

Triedy cImportPage, cBackupPage, cSettingsPage

Tieto triedy majú implementované metódy pre vykonávanie systémových funkcií. Trieda *cImportPage* na základe parametra *action* zavolá príslušnú metódu pre import užívateľov alebo predmetov. Trieda *cBackupPage* obsahuje metódy pre zálohu systému a pre prácu so zálohovanými súbormi. V triede *cSettingsPage* sú vytvorené metódy, ktoré menia systémové nastavenia.

Triedy cUser, cStudent, cTeacher

V metódach tried *cStudentPage* a *cTeacherPage,* sa modifikujú údaje o študentoch a učiteľoch v databáze, ale údaje z databázy o užívateľoch využívajú aj niektoré iné triedy, napríklad *cExchange, cCoursePage.* Vytvorili sa preto triedy, ktoré zaobaľujú prácu nad databázov študentov a učiteľov. Základná trieda je *cUser,* od ktorej sú odvodené triedy *cStudent* a *cTeacher,* ktoré ju rozširujú o niektoré špecifické vlastnosti študentov alebo učiteľov. *cUser* má metódy, ktoré vytvárajú nové záznamy v databáze, alebo ich modifikujú a mažú. Konštruktor týchto tried sa na základe parametra *id* pokúsi nájsť v databáze príslušný záznam, ak je výsledok operácie pozitívny, naplní premenné tejto triedy hodnotami z databázy. Pri získavaní týchto hodnôt, potom nie je potrebné pristupovať do databázy. Samozrejme pri menení alebo vymazávaní záznamu v databáze sa upravia aj tieto premenné. Odvodené triedy, *cStudent, cTeacher* navyše kontrolujú, či je *id* aj v tabuľke *tStudents* respektíve *tTeachers.* Navyše obsahujú premenné, ktoré nie sú pre všetky typy užívateľov rovnaké.

Triedy cCourse, cSpecialization

Tak ako triedy opísané v predchádzajúcom odseku, aj triedy *cCourse* a *cSpecialization* vytvárajú medzivrstvu medzi databázou a triedami, odvodenými od triedy *cPage.* Tieto triedy majú okrem základných metód modifikujúcich záznamy v databáze aj metódy, ktoré vracajú polia objektov typu *cStudent,* študentov, ktorý sú v databáze v relácií s predmetom alebo študijnym odborom.

4. Bezpečnosť

Pri vývoji aplikácie je potrebné venovať zvýšenú pozornosť bezpečnosti. Najmä ak ide o webovskú aplikáciu s citlivými dátami, ktoré by mohli byť útočníkom zneužité v jeho prospech. V systéme pre online testovanie študentov musia platiť obmedzenia pre užívateľov, ktorí by chceli mať prístup do časti programu alebo k dátam, pre ktoré nemajú dostatočné privilégiá.

4.1 Generovanie neoprávnených požiadaviek

Ak sa užívateľ prihlási do systému a ten mu povolí vstup, uloží sa do *session* typ užívateľa, ktorý môže mať tieto hodnoty: *admin, teacher, student, proctor*. Na základe typu, má potom užívateľ prístup iba k údajom a častiam systému, pre ktoré má privilégiá. V triede *cApplication* je definovaná abstraktná metóda *GetPageById*, ktorá vráti inštanciu triedy *cPage* a až následne sa v metóde *Run* triedy *cApplication* volá metóda process tohto objektu. Triedy vytvorené dedením z triedy *cApplication* implementujú metódu *GetPageById,* kde na základe parametra *action* vrátia príslušnú inštanciu triedy *cPage.* Ak ale užívateľ nemá v *session* nastavený typ, ktorý ho oprávňuje k vstupu alebo vykonávaniu akcií v príslušnom rozhraní, táto metóda vráti inštanciu triedy *cPage,* ktorá obsluhuje prihlasovanie sa do rozhrania. Znamená to, že ak by chcel potenciálny útočník naslepo vytvoriť príkaz, ktorý by sa vykonal v metóde *Process* nejakej inštancie triedy *cPage*, neuspeje, pretože sa zavolá metóda *Process* prednastavenej triedy. Uspieť by mohol jedine v tom prípade, keby dokázal modifikovať hodnoty premenných v *session.* Tomuto typu útoku sme ale zabránili tým, že *session* ukladá svoje informácie do databázy systému do tabuľky *tSession.* Ak by teda útočník dokázal modifikovať tieto hodnoty, musel by mať prístup do databázy. O túto ochranu sa ale musí postarať poskytovateľ služby a administrátor databázy.

Príklad útoku:
Útočník odpozoroval, že pri vymazávaní užívateľov z databázy sa posiela nasledovná http požiadavka *GET*:

```
http://url_systemu/index.php?site=student&action=do_delete&id=668
```

Mohol vytušiť, že *id* označuje id študenta a pokúsiť sa opakovaným útokom vymazať všetkých študentov z databázy. Spôsob popísaný vyššie, akým sme takéto situácie ošetrili, zabraňuje podobnému typu útoku.

4.2 Bezpečné uloženie hesiel v databáze

Ak by sa útočník dostal k heslám v databáze, vedel by sa pod cudzím prihlasovacím menom prihlásiť so systému a spôsobiť užívateľovi škody. Heslo v databáze preto nesmie byť uložené ako *plain text.* Zvyčajne sa pri ukladaní hesiel do databázy používa ich hash, na overenie hesla sa potom porovná hash z databázy s hashom vstupu, ktorý ako heslo zadal užívateľ. Tento prístup má slabinu v tom, že na internete sa dá veľmi ľahko nájsť program, ktorý používa slovníkový útok na hash, využívajúc pri tom rozsiahlu databázu často používaných hesiel. Spôsob ako zabrániť spätnému získaniu hesla z jeho hash-u, je pridávanie náhodného prefixu k heslu a namiesto ukladania samotného hash-u hesla, sa do databázy uloží hash zreťazeného náhodného prefixu a hesla. Do databázy sa uloží aj zvolený náhodný prefix, čo nepredstavuje slabinu tohto prístupu, pretože jednosmernosť hashovacej funkcie zabraňuje spätnému získaniu vzoru. Náhodný prefix ale nemalým podielom znižuje pravdepodobnosť úspechu slovníkového útoku.

4.3 Ochrana pred HTML Injection

HTML Injection je typ útoku, kde sa útočník snaží do html kódu stránky vsunúť svoj vlastný kód a zmeniť tým vzhľad stránky. Ako príklad môžme uvažovať nasledujúcu situáciu. Útočník ako študent, pridá ako poznámku k odpovedi v teste text v tvare:

```
Na <a href=”utocnikova url”>tejto</a> stranke je ina odpoved
```

Ak by systém nebol chránený pred HTML Injection, učiteľovi by sa pri prezeraní testu zobrazila poznámka s hyperlinkom na útočníkovu stránku, ktorá môže byť nebezpečná. Riešenie tohto typu útoku je v PHP veľmi jednoduché, stačí každý vstup od užívateľa ošetriť volaním funkcie *htmlspecialchars,* ktorá prekonvertuje nebezpečné znaky do bezpečného formátu, *escape characters.*

4.4 Ochrana pred SQL Injection

Na podobnom princípe ako HTML injection je založený útok SQL Injection. V tomto prípade sa útočník nesnaží vložiť na stránku kód navyše, ale pokúša sa upravením vstupných premenných modifikovať *SQL* príkaz, ktorý systém z týchto premenných vyskladá. Ak napríklad php skript pre autentifikáciu užívateľa vyzerá nasledovne:

```
$query = "SELECT * FROM tUser WHERE login='$_POST['login']' AND password='$_POST['pass']'";

mysql_query( $query );
```

Útočník môže poslať také hodnoty premenných *login* a *pass*, ktoré by mu zaručili prístup do systému aj bez znalosti hesla:

```
$_POST['login'] = "user";  //za user dosadi login niektoreho uzivatela

$_POST['pass'] = "' OR 1=1;
```

Bez ošetrenia vstupu, by požiadavka na databázu vyzerala nasledovne:

```
SELECT * FROM tUser WHERE login='user' AND password='' OR 1=1
```

Tento príkaz by bol sql serverom vyhodnotený vždy ako pravdivý a útočník by sa tak bez znalosti hesla vedel prihlásiť do systému. V PHP je vytvorená funkcia pre ošetrenie vstupu pred možným SQL Injection útokom. Táto funkcia sa volá *mysql_real_escape_string* a v našom systéme sa volá na každom vstupe, ktorý je súčasťou *sql query* posielanej SQL serveru.

4.5 Bezpečnosť protokolu pre výmenu dát

V protokole pre výmenu dát medzi aplikáciou a externým systémom požiadavky obsahujú autentifikačné údaje a kľúč, ktorým je následne šifrovaná komunikácia. Formát požiadavky sa spomína v 2.kapitole tejto práce. V tejto časti by som chcel rozobrať spôsob akým sa požiadavky šifrujú, pomocou asymetrickej šifry *RSA*.

RSA je jeden z najpoužívanejších asymetrických šifrovacích systémov. Pri šifrovaní sa predpokladá, že existuje verejný a súkromný kľúč účastníka komunikácie. V našom prípade pozná súkromný kľúč iba systém, ktorý poskytuje dáta. Jeho verejný kľúč je prístupný komukoľvek. Inicializácia inštancie RSA prebieha nasledovne:

1. Zvolia sa dve veľké prvočísla p a q $(p \neq q)$ a určí sa $n = pq$
2. Zvolí sa e tak, aby platilo: $1 < e < \varphi(n)$ a $\text{nsd}(e,\varphi(n)) = 1$
3. Dopočíta sa d, ktoré spĺňa: $ed \equiv 1 \pmod{\varphi(n)}$

Verejným kľúčom je dvojica $<e,n>$ súkromným kľúčom je číslo $<d>$. Šifrovanie a dešifrovanie prebieha nasledovne, nech m je číslo z množiny {0,1,...,n-1}

šifrovanie: $E(m) = m^e \bmod n$

dešifrovanie: $D(c) = c^d \bmod n$

Bezpečnosť RSA je založená na predpoklade, že problém faktorizácie dostatočne veľkých čísiel je ťažký problém. Ak by vedel útočník ľahko faktorizovať n vedel by získať $\varphi(n)$ a dopočítať súkromný kľúč d.

Preto som musel naprogramovať kód, ktorý dokáže efektívne modulárne umocňovať veľké čísla. Algoritmus, ktorý som použil, je zapísaný v pseudokóde:

```
Nech bmbm-1...b0 je binárny zápis čísla b

function Mod-Exp( a, b, n )
d <- 1
for i <- m to 0 do
   begin
     d <- (d.d) mod n
     if bi = 1then d <- (d.a) mod n
   end
return d
```

Záver

Cieľom tejto práce bolo vytvorenie aplikácie, ktorá by umožňovala online testovanie študentov formou testov, čím by sa sprehľadnilo a urýchlilo ich vyhodnocovanie. Požiadavkou bolo, aby bola aplikácia nezávislá od platformy na ktorej beží a aby bola prístupná aj z internetu. Výsledkom série sedení s kolegami Jurajom Porubským, Františkom Šmitalom a vedúcim mojej bakalárskej práce RNDr. Richardom Ostertágom, bola detailná špecifikácia aplikácie. Mojou úlohou bolo vytvorenie protokolu pre komunikáciu s externým systémom, ktorý našej aplikácií dodáva dáta, navrhnutie databázového modelu a naprogramovanie rozhrania pre administrátora.

Myslíme si, že cieľ práce sa nám podarilo naplniť. Výsledná aplikácia sa môže začať používať v praxi a umožniť tak učiteľom ľahko a prehľadne generovať testy na skúšky, minimalizovať čas strávený ich opravovaním. Aplikácia im poskytne dlhodobý prehľad o výsledkoch skúšok, pričom štatistiky môžu využiť pri vylaďovaní zložitosti úloh pre študentov. Študenti získajú vďaka našej aplikácií, ktorá im poskytuje pohodlný spôsob vypĺňania testov, prehľad o ich doterajších výsledkoch a dozvedia sa výsledky skúšok, ktoré prebehnú v systéme pre online testovanie, okamžite po ich skončení.

Pri tvorení špecifikácie sme rozoberali viacero možností, ktoré sa nakoniec vo výslednej aplikácií neobjavili, buď pre veľkú časovú náročnosť pri ich implementovaní alebo pre technické obmedzenia, ktoré sa vyskytli. Jednou z takých možností bola zjednodušená vizualizácia miestnosti, v ktorej prebieha skúška, pre dozerajúceho učiteľa alebo proktora. Inou možnosťou, ktorá sa nakoniec vo výslednej špecifikácií neobjavila bolo hodnotenie skúšky na základe celkovej úspešnosti študentov v testoch počas všetkých termínov.

V tejto práci by sa dalo pokračovať implementovaním spomenutých návrhov, ktoré vo výslednej aplikácií neboli zahrnuté. Pri webovských aplikáciách je vždy možné zvyšovanie ich bezpečnosti pred novými typmi útokov. Práca by sa dala rozšíriť o detailný popis kľúčových algoritmov a kapitolu s popisom dizajnu aplikácie.

Zoznam bibliografických odkazov

[1] ELMASRI, R., NAVATHE, S. *Fundamentals of Database Systems.* Addison-Wesley, 1994. ISBN 0-8053-1753-8

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