Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÝ PROJEKT

2019/20

Vladimír Vávra

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

ROČNÍKOVÝ PROJEKT

Předmět: Programování

Téma: Digitalizace výuky programování

Autor: Vladimír Vávra

Třída: 2.E

Školní rok: 2019/20

Vedoucí práce: Ing. Daniel Kahoun

Třídní učitel: Mgr. Lenka Veselá

Rád bych na tomto místě poděkoval svému učiteli programování, ing. Danielu Kahounovi, který mi vnukl nápad na tuto práci a po celou dobu ročníkové práce odpovídal na mé dotazy ohledně práce. Zároveň bych také chtěl poděkovat svým spolužákům Vítku Lisnerovi za vymyšlení jména projektu a Janu Žůrkovi za přípravu loga a jeho variant.

Prohlašuji, že jsem ročníkovou práci vypracoval/a samostatně pod vedením Ing. Daniela Kahouna a že jsem poctivě citoval/a všechny použité zdroje a literaturu.

V Praze dne _____ _____

vlastnoruční podpis autora

Anotace

Program navazuje na ročníkovou práci „Digitalizace výuky programování“ a poskytuje tak úlohám sepsaným v ročníkové práci možnost testování. Testování probíhá buď formou kvízových otázek, nebo pomocí bezpečného spuštění a vyhodnocení odeslaných zdrojových kódů řešících zadanou úlohu. Hlavní důraz při vytváření i vyhodnocování úloh je kladen na automatizaci. Veškeré úlohy jsou tedy ohodnoceny programem okamžitě po odeslání. Součástí programu je i funkční a bezpečná autentizace a autorizace pro předem vytvořené účty.

Abstract

The program is a follow-up to the course "Digitalization of Programming Teaching" and thus provides the possibility of testing for tasks created in term paper. Testing is performed either in the form of quiz questions or through safe execution and evaluation of the submitted source code solving the given task. The main emphasis in creating and evaluating tasks is put on automation. All tasks are evaluated by the program immediately after sending. The program also includes functional and secure authentication and authorization for pre-created accounts.

Die Annotation

Das Programm ist eine Fortsetzung des Kurses "Digitalisierung des Programmierunterrichts" und bietet somit die Möglichkeit, Aufgaben zu testen, die in der Hausarbeit erstellt wurden. Das Testen erfolgt entweder in Form von Quizfragen oder durch sichere Ausführung und Auswertung des eingereichten Quellcodes zur Lösung der jeweiligen Aufgabe. Der Schwerpunkt bei der Erstellung und Bewertung von Aufgaben liegt auf der Automatisierung. Alle Aufgaben werden unmittelbar nach dem Senden vom Programm ausgewertet. Das Programm umfasst auch die funktionale und sichere Authentifizierung und Autorisierung für vorab erstellte Konten.

Obsah

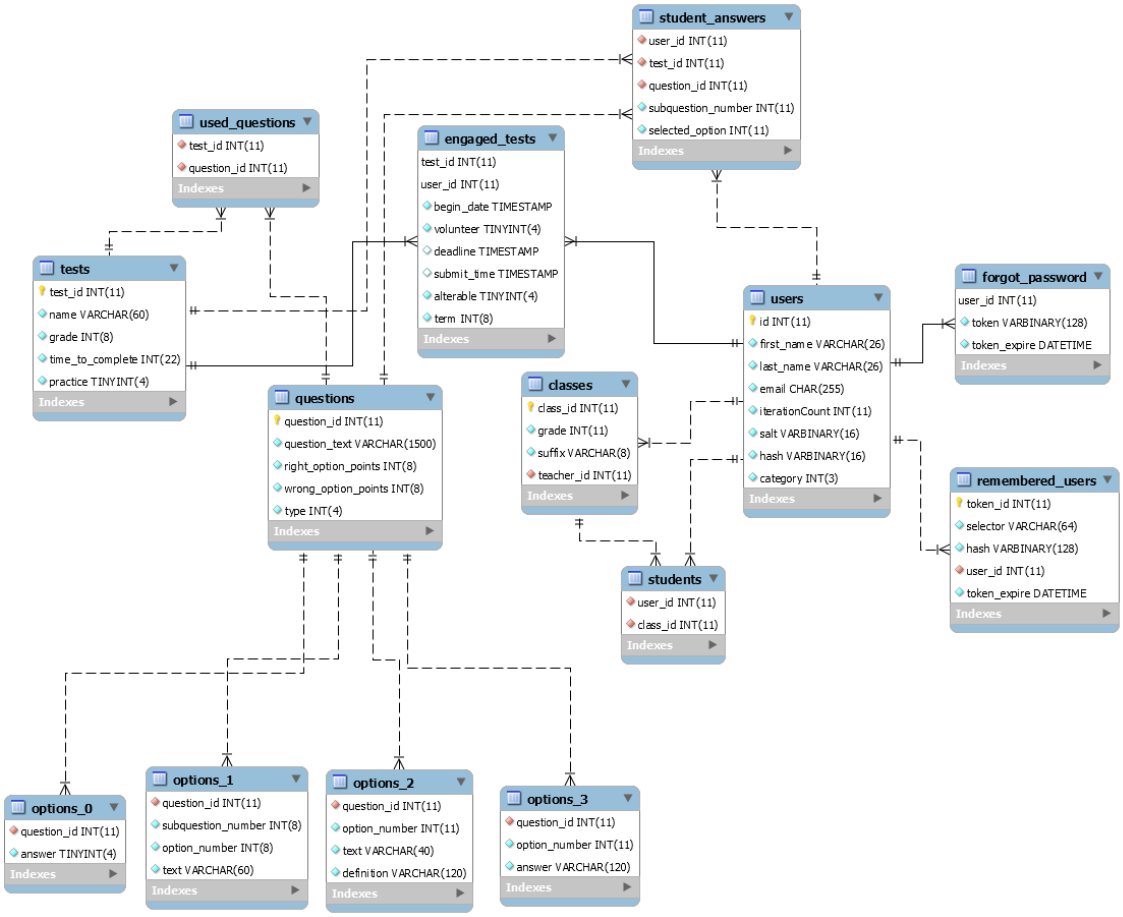
Úvod

Tento projekt navazuje na ročníkovou práci z předmětu programování z roku 2019/2020 jménem Digitalizace výuky programování. Účelem projektu je vytvořit systém, díky kterému je možné testovat kvízové otázky vytvořené v práci. Jelikož je webovou aplikací, běží na nějakém serveru, v našem případě Tomcat 9 (možnost změnit). Vzhledově si bere inspiraci ze systému na známkování Ing. Daniela Kahouna, každopádně funguje odlišně. Program je určen pro učitele a studenty, kde jsou pro každou uživatelskou třídu dostupné jiné možnosti. Učitel může vytvářet, přidělovat a spouštět testy studentům. Student může test spouštět a vyplňovat, kdy jsou mu po testu zobrazeny jeho výsledky. Původně bylo zamýšleno k současným 4 typům úloh přidat ještě pátou, ve které by se automaticky a bezpečně otestovaly zdrojové kódy řešící danou úlohu, od tohoto záměru však bylo kvůli nedostatku času upuštěno. Důležitou součástí projektu je také vlastní přihlašovací systém, který zajišťuje přihlašování a správu účtů včetně funkcí jako zapomenuté heslo nebo pamatuj si mě.

1. Databáze

Na uchovávání dat programu je použita databáze MySQL 8.0, viz. příloha. Ta běží na portu 3306. K její snadné údržbě a manipulaci s účty před zavedením uživatelské třídy administrátora je použit program MySQL Workbench 8.0.

1.1. Architektura



Obrázek 1 – architektura databáze

Architektura databáze je vidět na obrázku 1. Obdélníky značí jednotlivé tabulky, kde jsou u každé z nich popsány všechny sloupce, které obsahuje. Bleskem jsou označeny primární klíče (unikátní identifikátory pro tabulku), červeně sloupce odkazující na klíče z jiné tabulky, modře hodnoty, které nesmí být null a bíle jakékoliv hodnoty datového typu napsaného vpravo.

Databáze je navržena tak, aby obsahovala co nejvíce dat v co nejmenším objemu a zároveň a je velice dobře škálovatelná.

1.1.1. Uživatelé

Pro reprezentaci uživatelů je klíčová tabulky users. V ní má každý uživatel záznam o svém jméně, e-mailu a uživatelskou třídu a informace o heslech (viz. kapitola 2) uložené pod unikátním identifikátorem.

Spolu s tabulkou users tvoří jádro reprezentace uživatelů tabulky classes a students. Tabulka classes uchovává data o třídách, přesněji její stupeň, sufix (například pro 2.E je sufix „E“) a id učitele, který má třídu na starost. Dále musíme nějakým způsobem zařadit studenty do tříd. Jelikož není každý uživatel zároveň studentem, vkládat tuto informaci přímo do tabulky users by bylo proti principům databází a zabíralo by to zbytečný prostor. Proto existuje třída students, ve které se párují čísla tříd s čísly studentů.

Pro další funkce spojené s přihlašováním existují také tabulky forgot_password (viz. kapitola 2.2.5) a remembered_users (kapitola 2.2.3). Funkce sloupců jsou po přečtení kapitol zřejmé.

1.1.2. Testy

Informace o testech jsou uloženy v tabulce tests. Jsou jimi jméno testu, stupeň tříd, pro které je test určen, doba na vypracování testu v sekundách a proměnná practice říkající, zda se jedná o procvičování. Procvičovací testy jsou dostupné všem, kteří mají vyšší ročník než je požadováno a nezapočítávají se do hodnocení mezi ostatní testy.

Studentovi je možné test přidělit. V takovém informace o novém testu zobrazí při přihlášení. Datum tohoto přiřazení reprezentuje proměnná begin_date. Při přidělování může učitel nastavit, zda je test pro studenta dobrovolný, nebo ne. V případě, že student tento test nespustí, nebude součástí hodnocení. V případě, že ho spustí, do hodnocení se započítá. To označuje proměnná volunteer. Takto přiřazený test může student spustit. Jakmile k tomu dojde, tak se do sloupce deadline zapíše čas, do kterého musí student odevzdat test. Jakmile test odevzdá, že zapsána proměnná submit_time s časem odevzdání. Proměnná alterable označuje, zda je možné stále odpovídat na otázky. False bude tehdy, dojde-li čas nebo pokud odevzdá uživatel test dřív. Proměnná term označuje časové období, kdy byl test napsán. Jelikož je možné napsat například test pro první stupeň na stupni pátém, bude se započítávat do hodnocení mezi testy pátého stupně. Hodnota tohoto políčka je stejná jako hodnota grade v tabulce classes, ovšem pouze v době inicializace políčka, poté může být jiná.

Takovýto test je ovšem k ničemu, protože v sobě nemá otázky. Na to je potřeba další tabulka, kterou je questions. Ta obsahuje samotné znění otázky a informace o tom, jak má probíhat bodování. Jednotlivé otázky mají svoje vlastní pod-otázky a v tabulce questions je informace, kolik bodů přidat, jestli na pod-otázku odpoví uživatel správně, resp. odebrat, odpoví-li špatně. Současný systém podporuje 4 typy otázek. Informace o tom, o jakou otázku jde nese proměnná type, která nabývá hodnot od 0 do 3.

Pro každý typ otázky je připravena tabulka se jménem ve tvaru options_(číslo typu otázky).

V options_0 se vyskytují řešení otázek typu pravda, nebo lež. Otázkou je v tomto případě výrok a odpovědí je pravda, nebo lež podle toho, zda je výrok pravdivý.

V options_1 se nacházejí řešení otázek typu doplň do textu. Otázkou je nyní text, který obsahuje nespecifikovaný počet tří podtržítek za sebou („___“). Na tomto místě se poté v programu nachází select box, do kterého student vybírá ze několika možností text, který nejlépe pasuje do kontextu. Platí, že je mezi odpověďmi právě jedna správná odpověď. Sloupec subquestion_number obsahuje číslo, na kolikátý znak tří podtržítek se má select box vložit. Option_number značí číslo textu, kde 0tý prvek označuje správnou možnost. K těmto záznamům je přidělen text označující samotnou možnost.

Například tabulka pro následující text: „Abychom mohli psát kódy v Javě, potřebujeme kromě JRE, ve které je obsažen virtuální stroj ___ (***JVM***, *JDK*, *JVS*), stáhnout také ___ (***JDK***, *JVB*, *JKD*)“, kde v závorce jsou možnosti na doplnění na prázdné místo a tučně označená možnost je správně vypadá takto:

question_id	subquestion_number	option_number	text
55	0	0	JVM
55	0	1	JDK
55	0	2	JVS
55	1	0	JDK
55	1	1	JVB
55	1	2	JKD

Obrázek 2 – options_1 ukázka

V options_2 se nachází řešení otázek typu spoj pojem s definicí. K jednotlivým pod-otázkám se nachází definice a pojem. Tabulka options_3 obsahuje otázky typu seřaď pojmy podle kritéria. Velice se podobá options_2, pouze nám z tabulky vypadla nutnost uchovávat definici, stačí nám jen pojem.

Databáze je navržena tak, aby šlo otázky využívat ve více testech najednou, například pokud je v testu nalezena chyba, je vytvořen nový test se stejnými otázkami a mírnými úpravami. Toho je docíleno pomocí tabulky used_questions, která vytváří testy. Spojuje totiž otázky s testy, ve kterých jsou užité.

Odpovědi studentů jsou uchovány v tabulce student_answers. Ta pomocí tří cizích klíčů spojuje test, uživatele a otázku a uchovává informace, kterou jak odpověděl na jakou pod-otázku. Tato tabulka je použitelná pro všechny typy otázek.

1.2. Komunikace

1.2.1. Spojení

Spojení databáze a aplikace zaštiťuje MySQL Connector 8.019. Tento jar je vložen do projektu jako knihovna a pomocí třídy DbManager využívající JDBC poskytuje aplikaci přístup k databázi. DbManager je databázový wrapper, který v sobě obsahuje instanci třídy Connection, na které se dají připravovat dotazy do databáze. Tato instance se inicializuje pomocí metody newConnection, který vytvoří zcela nové spojení s databází. Na vytvoření potřebuje ovšem data k tomu nutná, jako jméno databáze, účtu, odkaz s portem a heslo. Tyto údaje jsou napevno napsané ve zdrojovém kódu a v případě změny databáze musí být přepsány. V budoucnosti s implementací třídy Admin bude přidána možnost databáze měnit za chodu aplikace. Pomocí metody getConnection je poté toto spojení dostupné ostatním objektům. Spojení se nakonec uzavře pomocí zavolání metody closeConnection.

1.2.2. Databázové objekty

Data z databáze je nejpraktičtější reprezentovat pomocí objektů obsahující vlastnosti odpovídající sloupcům v databázi. Veškeré důležité metody jsou popsány pomocí javadoců. Data jsou z databáze získávána pomocí PreparedStatementů, aby se zabránilo SQL injekci, kdy se uživatel může pokusit zadat do pole, ze kterého se přímo zadává do databáze vlastní SQL dotaz, který může zpřístupnit útočníkovi databázi. Některé objekty obsahují metody využité při tvorbě HTML stránky na backendu. Takové metody mají v názvu HTML.

Jako nejdůležitější třídu odpovídající databázové tabulce na reprezentaci uživatelů můžeme nepochybně označit třídu User. Ta odpovídá tabulce users. Z User dědí třídy

Student a Teacher (třída Admin není implementována). Třída Student obsahuje referenci na další databázovou třídu Classs a List všech EngagedTestů. Tato proměnná není při vytváření objektu naplněna testy. Musí se to udělat až později pomocí metody loadEngagedTests. Třída Teacher obsahuje List objektů Classs reprezentující třídy. Stejně jako u Studenta a testů musí být i tento list naplněn zvlášť, a to pomocí metody loadClasses.

Třída Classs označuje třídu žáků. Jako u předchozích případů musí být list žáků naplněn až externě pomocí metody loadStudents.

K uživatelům je nutné už doplnit jen třídu Token a její potomky. Token reprezentuje společné vlastnosti tabulek remembered_users a forgot_password. Obsahuje proměnné na uchování ID uživatele, hashe a časové známky. Obsahuje také proměnnou na řetězcovou podobu tokenu. Z této třídy dědí 2 třídy. První z nich je ResetToken. Nepřidává žádné instanční proměnné navíc, jen specifikuje hodnoty zděděných. Přidává ovšem metody na uložení tokenu do databáze a jejich získání. Třída RememberMeToken přidává navíc dvě proměnné, id pro ID tokenu a selector. O tokenech více v kapitole 2.

Co se týče testových tabulek, tak v programové reprezentaci hraje významnou roli třída Test. Ta obsahuje všechny proměnné z tabulky tests a k tomu pomocné proměnné na usnadnění procesu bodování testu. Otázky uchovává v listu a jejich objekty je nutné externě přidat pomocí zavolání metody loadQuestions. Ze Testu dědí třída EngagedTest. Ta reprezentuje test, který byl studentovi přidělen. Obsahuje důležitou metodu saveAnswers, která uloží všechny aktuální odpovědi do databáze. Jestliže se v databázi odpověď na tuto otázku již vyskytuje, tak ji přepíše. Jestliže ne, tak ji přidá.

Jak už je psáno v kapitole o tabulkách v databázi. Test je sám o sobě bez otázek k ničemu. Ty jsou reprezentovány pomocí třídy Question. Pro všechny otázky je společná jedna třída Question. Možnosti je opět nutné externě do listu možností načíst pomocí metody loadOptions. Tato metoda je ovšem na každé otázce zavolána již v metodě loadQuestions. V případě, že učitel vytváří otázku na frontendu, musí se server ujistit, zda zadal validní data. O většinu se stejně jako u třídy Test stará konstruktor, který při jakékoliv chybě okamžitě vyhodí výjimku, ovšem zde musíme zjistit, zda se do listu questions nepokouší uživatel vložit možnost jiného typu než je typ otázky. To řeší metoda addOption.

Tyto objekty, které jsou přidávány do listu v otázce jsou všechno potomci třídy Option. Tyto třídy jsou celkem 4. První je BooleanOption s jednou proměnnou typu boolean

obsahující správnou odpověď na otázku. Druhou je FillInTextOption, která reprezentuje jednu pod-otázku. Obsahuje proměnnou držící správné slovo a ArrayList špatných slov. Třetí třídou je OrderOption, která obsahuje text přiřazený k pod-otázce. Poslední třídou je ConnectOption, která nedědí přímo z Option, nýbrž z OrderOption. Přidává další proměnnou obsahující definici pojmu. Tyto možnosti se získávají z databáze pomocí statické metody createOptions na třídě Options. Vrací ArrayList všech možností, které jsou spárovány s otázkou předanou jako argument.

V samotné třídě Option se nachází číslo pod-otázky a číslo pod-otázky, kterou student označil při vypracovávání testu. Každá z možností dědících z Option si určuje vlastní pravidla na vyhodnocování správnosti studentova rohodnutí (viz. kód)

2. Autentizace a autorizace

2.1. Uživatelské třídy

Program podporuje tři typy uživatelů. Jsou jimi student (Student), učitel (Teacher) a administrátor (Admin). Každý uživatel aplikace je svázaný právě s jedním z nich a každý z nich má odlišné pravomoci v systému. Uživatelé typu student mohou spouštět vypracovávat jim přidělené testy a zobrazit stav své klasifikace. Učitelé mohou vytvářet, přidělovat a spouštět testy pro studenty, které jim jsou přiděleny, na studenty jiných učitelů nemají vliv. Mohou také zobrazovat klasifikaci studentů, které učí. Administrátor zatím nebyl implementován, nicméně konceptem této třídy je vytváření nových uživatelů a přidělování práv. Nevidí však do klasifikace studentů a nemůže jim přidělovat ani spouštět testy.

2.2. Přihlášení

Samotné přihlášení probíhá přes webovou stránku, jejíž design vytvořil (Colorlib, 2018). Kód stránky je uložen v souboru login.jsp doplněn CSS a JS souborem. K přihlášení jsou potřeba znát e-mail uživatele a jeho heslo, jelikož e-mail slouží jako unikátní identifikátor účtů. Před odesláním údajů je skriptem zkontrolována validita zadaných údajů (přítomnost zavináče, tečky a nějakých znaků při vyplňování hesla). Po dokončení validace se data odesílají na server, konkrétně na servlet LoginController pomocí HTTPS.

O to, aby se zachovalo přihlášení i při přecházení mezi stránkami se stará session (sezení), jelikož HTTP je stateless protokol. Ten automaticky ukládá objekt uživatele (User), který je přihlášen. Session je identifikován pomocí JSESSIONID, které je uchováno v Cookie souboru na straně klienta.

V plánu bylo také začlenit do přihlašování OAuth II protokol a tedy i přihlašování přes google a jiné sociální sítě. Nicméně to se nepodařilo kvůli neznalosti Mavenu a podobných automatických build nástrojů. Rozhodně je v plánu je začlenit v budoucnosti.

2.2.1. Teorie

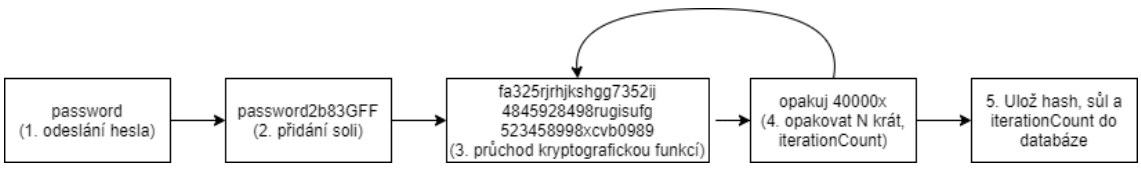
Ukládat hesla v čisté podobě v databázi je z hlediska bezpečnosti naprosto hrubá chyba. Místo toho jsou hesla uložena pomocí takzvaných hashů. Hash je řetězec znaků pevně stanovené délky, který získáme, vložíme-li naše heslo do nějaké hashovací funkce,

v programu SHA512. Délka takového hashe je programem stanovená na 128 znaků. Zásadami těchto kryptografických funkcí je, že jsou pomalé, aby lépe čelili brute-force útokům, kdy hackeři zkouší různá hesla, a že každá byť jen sebemenší změna v hesle vyprodukuje naprosto odlišný hash. Teoreticky tedy ověření platnosti hesla probíhá tak, že v databázi jsou uložena jednotlivá hesla v podobě hashů, jakékoliv nové příchozí heslo se nechá zpracovat stejnou hashovací funkcí jako na heslo v databázi a tyto hashe se porovnají.

Tento koncept je ovšem stále zranitelný. Jestliže hacker získá přístup k databázi, může pomocí speciální tabulky obsahující velké množství možných hashů porovnávat přímo hashe a obejít tak pomalou kryptografickou funkci. Jakmile najde hashe, které se shodují, vyhledá pomocí tabulky i původní heslo. Těmto tabulkám se říká rainbow tables, duhové tabulky

Tomuto lze zabránit pomocí takzvané soli. Sůl je řetězec znaků, které se přidávají na konec hesla a spolu s ním se nechají zpracovat hashovací funkcí. Tento hash je poté uložen v databázi. Spolu s ním je nutné do databáze uložit i sůl. Ta může na rozdíl od hesla být nezašifrovaná. Útočník může její hodnotu klidně znát. Jejím úkolem totiž není ztížení prolomení hesla před brutálními útoky, nýbrž vyřazení možnosti používat duhové tabulky. Jejich síla totiž tkví v tom, že dovolí útočníkovi obejít nutnost zpracovávat heslo pomocí pomalé kryptografické funkce. Pokud ovšem použijeme sůl, tak je tabulka k ničemu, jelikož všechna hesla v tabulce po přidání soli úplně jiný hash.

Třešničkou na dortu při ztěžování prolomení hesla je poté počet iterací hashovací funkce. Můžeme tak ještě více zpomalit hashovací funkce, kdy hashujeme hash z minulé iterace. Tento počet iterací je opět nutné uložit spolu s hashem a solí.



Obrázek 3 – Postup hashování

2.2.2. Implementace

Servlet nejprve vytvoří nový objekt User pomocí konstruktoru přebírající email uživatele. Následně na něm zavolá metodu authenticate, které předá jako argument heslo. Metoda vrací true, jestliže se heslo v databázi shoduje s předaným heslem. Její logiku obstarává

hlavně třída Hasher a její statická metoda hash. Ta jako argumenty přebírá počet iterací hashovací funkce, sůl jako pole bytů, odeslané heslo a délku hashe. Kromě hesla a délky hashe se tyto údaje nacházely v databázi a v době volání metody jsou uloženy v instanci třídy User, na které je volána metoda authenticate. Metoda hash na tento účet vytvoří instanci Hashovací funkce, která poté vrací pole bytů reprezentující hash hesla. Návod získán z (Millington, 2020).

Pomocí metody slowEquals je poté hash uložený v databází porovnán s hashem vyprodukovaným metodou hash. Porovnávat pole bytů klasickým způsobem pomocí Arrays.equals by nebylo dobré řešení. Systém by se totiž stal zranitelný vůči takzvanému timing útoku. Metoda equals by vrátila false okamžitě, když by zjistila, že se nějaké dva prvky neshodují. Pokud tedy například nesouhlasí hned první znak hashe, je informace, že je heslo špatně dřív, než kdyby byl tento odlišný znak na konci. Jelikož ovšem timing útok můžeme použít jen na porovnání hashů a jak již víme, s každou sebemenší změnou hesla se dramaticky mění hash, o hesle nám timing útok neřekne téměř nic. Útočník ovšem na dálku může zjistit přibližný hash hesla, aniž by se dostal do databáze. I tento minimální detail je ale lepší zabezpečit pomocí slowEquals. Ten funguje tak, že pokud narazí na špatný znak nebo se liší délka hashů, neukončuje okamžitě svou činnost, nýbrž nastaví nějakou pomocnou proměnnou na false a na konci metody tuto proměnnou vrací.

Jestliže je autentizace úspěšná, je na HttpSession objekt uložen objekt třídy User tohoto aktuálně přihlášeného uživatele a uživatel je přesměrován na hlavní stránku, index.jsp. Jestliže úspěšná nebyla, je uživatel přesměrován zpět na přihlašovací stánku, kde je v URL předán parametr LoginFail. Na stránce se poté zobrazí chybová hláška ohlašující neúspěšné přihlášení.

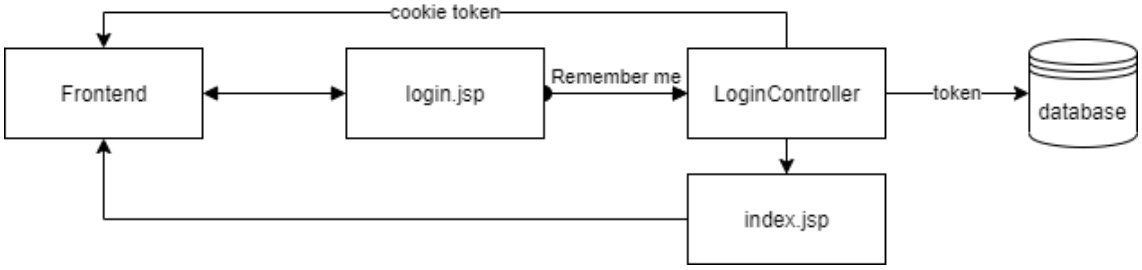
Abychom zabránili tomu, aby se někdo dostal na ostatní stránky nebo servlety bez přihlášení, použijeme Filter, v implementaci AuthenticationFilter. Ten povolí spuštění nějakého servletu. Pokud patří do skupiny v kódu vyjmenovaných servletů, tak je servlet spuštěn pouze tehdy, je-li user jiný od null a zároveň session není zaniklá.

2.2.3. Remember me

V případě, že se uživatel nechce stále přihlašovat, může při přihlášení zaškrtnout tlačítko remember me. Při příštím spuštění stránky na stejném počítači již nebude vyžadováno přihlášení a uživatel bude automaticky přesměrován na hlavní stránku.

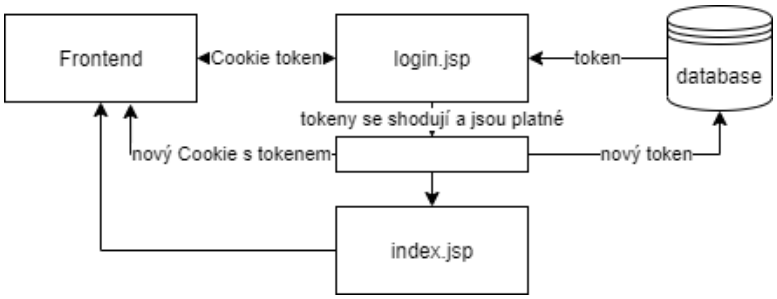
Tento proces je možný díky takzvaným tokenům. Token tvoří dva řetězce. Selector a hodnota tokenu. Selector je náhodný řetězec, který je v čisté podobě uložen v databázi. Hodnota tokenu je v databázi uložena v podobě hashe. Tento token je poslán i ke klientovy ve formě Cookie. Ta obsahuje obě hodnoty v čisté podobě, ani hodnota tokenu tedy není hashovaná. Při příštím přihlášení poté prohlížeč odešle na server tento token. Pomocí selectoru najde server v databázi hodnotu tokenu a porovná ji s hodnotou tokenu, který přišel od klienta prohnaným hashovací funkcí. Pokud se shodují a token ještě nevypršel, dojde k automatickému přihlášení.

Jestliže bylo při úspěšné autentizaci toto tlačítko škrtnuté, vytvoří LoginController nový RememberMeToken svázaný s uživatelem. Ten dědí ze třídy Token, která váže token k uživateli a zároveň dovoluje kontrolovat, zda token nevypršel díky Timestamp objektu. RememberMeToken rozšiřuje token o vlastnost selectoru. Token vygeneruje nový selector a také jeho hodnotu. Okamžitě tuto hodnotu uloží do databáze s datem vypršení za týden. Aby nedošlo k přetěžujícímu útoku na databázi, je počet tokenů na jednoho uživatele omezen na 20. Selector a hodnota tokenu je poté přidána do Cookie odeslaného pomocí HttpServletResponse objektu.



Obrázek 4 – Uložení RememberMeTokenu

Při každém spuštění login.jsp je spuštěn scriplet zkoumající, zda není přítomen soubor Cookie obsahující RememberMeToken. Toho je docíleno pomocí statické metody isLogged na servletu LoginController. Jestliže se takový Cookie v prohlížeči nachází, vytvoří se pomocí statické metody retrieveRememberToken, která jako argument dostane selector. Jestliže se hodnoty tokenů shodují, aktuální soubor Cookie se vymaže a vytvoří se zcela nový token s novým selectorem i hodnotou, které nahradí aktuální token v Cookie i v databázi. Zároveň se také obnoví platnost tokenu na týden. Automatické přihlašování tedy zůstane na tomto zařízení zachování i nadále, ovšem bezpečnost se díky obměně tokenu zvýšila.



Při odhlášení je soubor Cookie obsahující token vymazán a současně s ním i záznam v databázi o tokenu.

Obrázek 5 – Získání tokenu

2.2.4. Změna hesla

Po přihlášení je možné na hlavní stránce v navigaci kliknout na tlačítko změnit heslo. To uživatele přesměruje na upravenou stránku podobnou té přihlašovací, kde zadá své staré a nové heslo. Po verifikaci je formulář odeslán ChangePasswordController sevletu. Jestliže se staré heslo shoduje s heslem v databázi a zároveň zadáno nové heslo, tak se pomocí metody changePassword na instanci User změní heslo v databázi na nové heslo. Uživatel je poté přesměrován opět na hlavní stránku.

2.2.5. Zapomenuté heslo.

V případě, že uživatel zapomněl své heslo, může si ho pomocí svého přístupového e-mailu změnit. Toho docílí tak, že na přihlašovací stránce klikne na tlačítko zapomenuté heslo. Když uživatel tak učiní, bude přesunut na stránku podobnou přihlašovací, kde vyplní e-mail svého účtu. Po odeslání tohoto formuláři bude zavolán ForgotPasswordController servlet, jež na tento e-mail odešle odkaz (Pankaj, 2014), na který když uživatel v mailu klikne, bude mu dovoleno změnit heslo. Současně se do databáze uloží ResetToken obsahující hash tohoto tokenu. Odkaz v e-mailu bude obsahovat jako parametr tento token v čisté podobě. Uživatel bude poté mít 30 minut na to, aby na odkaz klikl a heslo změnil. Ten ho přesune na stránku podobné přihlašovací, kde může vyplnit pouze nové heslo. Ve skutečnosti se jedná o stránku changePassword.jsp. Po odeslání se probudí NewPasswordController, který se pokusí změnit heslo na nově zadané. Podle toho, zda se mu to podaří odkáže zpět na login.jsp s odpovídajícím parametrem.

2.2.6. Odhlášení

Odhlášení se provádí na hlavní stránce. Při kliknutí na tlačítko odhlásit se se aktivuje LogoutController servlet. Ten vymaže z databáze RememberMeToken pro aktuální

zařízení společně s přiděleným Cookie souborem. Zároveň také anuluje aktuální session a tím znemožní další přihlašování. Pomocí nastavení hlavičky na hlavní stránce také zabráníme tomu, abychom se i po odhlášení mohli pomocí tlačítka zpět dostat na hlavní stránku bez přihlášení (viz. kód).

2.3. Autorizace

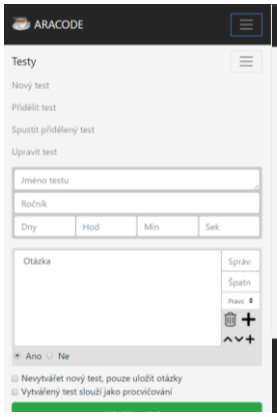
Autorizací rozumíme přidělování pravomocím různým typům uživatelům a zobrazování jim jiného obsahu. Aby se předešlo redundantnímu kódu, je pro všechny typy tříd vytvořena jen jedna hlavní stránka, která se generuje v závislosti na typu třídy pomocí JSTL tagu choose. Pro aplikaci platí, že si autorizaci jednotlivé funkce řeší po svém. Zjistit, zda je User učitelem nebo studentem je celkem jednoduché, protože si tuto vlastnost udržuje jako proměnnou. Problém ovšem je, pokud se například učitel jedné třídy chce vměšovat do klasifikace studentů třídy jiné. Oficiálně na to nemá právo a musí si to tedy tento servlet sám zařídit, aby takovéto kroky nemohl učitel podniknout.

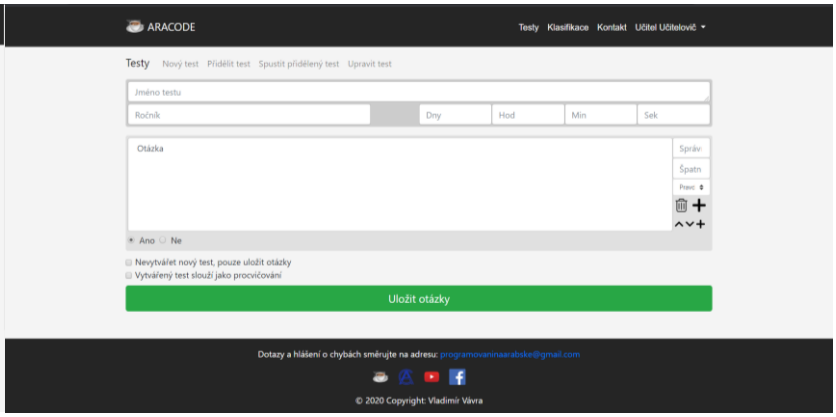
3. Testy

Jak je již zmíněno v kapitole 2.3, hlavní stránkou frontendu je index.jsp. Ta se vykreslí každému uživateli jinak. Učitelská stránka vypadá jinak než studentská. Je tomu tak díky JSTL tagům. V této kapitole probereme jednotlivé funkce učitelů a studentům, které stránka poskytuje. Celý web je plně responzivní a veškeré bloky jsou uzpůsobeny tak, aby při zmenšením zachovaly hezký vzhled i přehlednost. Všechny frontendové stránky využívají framework bootstrap a knihovnu do javascriptu JQuery. O hladké scrollování a správné umístění hlavní navigace se stará skript MainJS.js.

3.1. Učitel

Hlavním úkolem učitelů je v této aplikaci vytváření a administrace testů. Administrací rozumíme přidělování, spouštění a úpravy otázek. Tyto jednotlivé úkony jsou rozděleny do bloků, ke kterým se dá dostat pomocí druhé navigace. Vlevo ukázka zmenšené verze. Při kliknutí na položku z druhé navigace se pomocí skriptu zviditelní jiný blok. Všechny bloky jsou tedy již částečně sestaveny při vykreslování stránky, jen nejsou vidět. Blok upravit test není zatím implementován a úpravy testů je prozatím možné provádět pouze manipulací s databází.





Obrázek 6 – pro mobily *Obrázek 7 – Pro počítač*

3.1.1. Vytváření

Při spuštění stránky je učiteli automaticky zobrazen blok, kde se vytváří testy. Horní políčka slouží na vyplnění informací o testu. Jméno testu (max 60 znaků), ročník (1 -255), počet dní (0 - 365), počet hodin (0 - 23), počet minut (0 - 59), počet sekund (0 - 59). Jde o logicky zvolené rozsahy, které zároveň brání přetečení databáze,

například co se jména testu týče. Kontrola správnosti zadání rozsahu probíhá jak na frontendu, tak i backendu v případě, že někdo s frontendem manipuloval.

Hlavně však jde o tvorbu otázek. K tomu slouží dolní pole. Do velkého textového pole se píše zadání otázky, které nesmí přesáhnout velikost 1500 znaků. Napravo se poté napíše počet bodů získaný za správnou v pod-otázce a ztracený počet bodů za špatnou odpověď. Pomocí selectu je možné vybrat typ otázky.

U otázky typu pravda, nebo lež se vlevo dole označí správná odpověď.

U otázky typu doplňte do textu je třeba do textu psát znaky tří podtržítek (___), aby bylo možné vytvořit novou pod-otázku toho je možné docílit pomocí malého plusu v pravém panelu u otázky. U otázky typu pravda, nebo lež pochopitelně tlačítko nic nedělá. Možnosti se párují se znaky v textu v pořadí, v jakém jdou seshora dolů a zleva doprava. V případě, že chce uživatel přidat špatnou odpověď, klikne na velké tlačítko plus vedle pod-otázky. Pro odebrání pod-otázky klikne na velký koš u pod-otázky, přičemž platí, že minimálně jedna pod-otázka musí zůstat. Jestliže se počet pod-otázek neshoduje s počet tří podtržítek, zobrazí se chybová zpráva, která zmití až při nápravě Pod-otázky se dají posouvat pomocí šipek vlevo od nich. Ty posunou možnost buď o 1 nahoru, nebo o 1 dolů.

U otázek typu přiřaď pojem k definici záleží na pořadí, v jakém jsou shora dolů umístěny bloky se zadáním pod-otázek. Otázky se totiž na server posílají v pořadí shora dolů a tedy se i v tomto pořadí ukládají do databáze. Kvůli způsobu sestavení otázky tohoto typu při testu, jak uvidíte v následující kapitole na pořadí záleží, a tedy jsou i zde přítomny šipky na posouvání pod-otázky. U otázek typu seřaď pojmy je přítomnost šipek samozřejmostí.

To stejné, co s pod-otázkami můžeme dělat taktéž s otázkami. Můžeme je mazat pomocí ikony koše. Stále ovšem platí, že musí být přítomna alespoň jedna otázka, tedy poslední přítomná nejde smazat, aby bylo možné vybudovat nový test. Otázky se také mohou posouvat pomocí šipek. Otázky v testu budou sestaveny ve stejném pořadí, jako otázky zadané zde, jelikož jsou data na server odeslána s explicitním číslem otázky vygenerovaným při odesíláním a přidělaným na konec každého inputu. Server tak do databáze uloží otázky v pořadí, v jakém jsme je odeslali. Test je poté sestaven ve stejném pořadí, jaké definuje pořadí z databáze.

Před odesláním můžeme zaškrtnout dva checkboxy. Pokud zaškrtneme první, uloží se pouze otázky a na horní blok s informacemi o testu nebude brán žádný zřetel. Pokud zaškrtneme druhý, bude test brán jen jako procvičování a bude při hodnocení zahrnut do jiné kategorie.

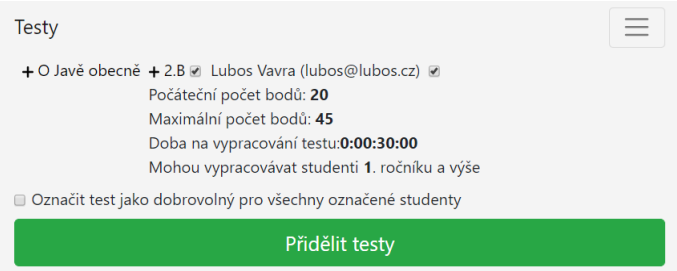
O funkčnost tohoto procesu se stará javascriptový soubor TestCreatingManager.js. Ten obsahuje funkce na přidávání otázek, jejich mazání, zviditelňování bloků z navigace a také o správné pojmenování dat, které při odeslání zamíří na server. Technické detaily souboru jsou dostupné přímo v souboru a není třeba ho zde rozebírat.

Po odeslání formuláře se data dostanou na servlet TestCreator. Zde se data parsují a validují. Pomocí metody getParameter na objektu HttpRequest lze získat data z frontendu, znáte-li správné jméno. Jak již bylo řečeno, o správné pojmenování se při odesílání formuláře postala skript TestCreatingManager.js. Například text otázky je pojmenován question-text(questionNumber). Na začátku je vytvořen objekt třídy Test, kterému jsou postupně předávány objekty třídu Question, jak se postupuje parsováním dat. Jakmile již nejsou žádné další otázky, je test uložen pomocí metody addNewTest do databáze. Poté je učitel odkázán zpět na index.jsp.

3.1.2. Přidělování

Na rozdíl od vytváření testů je přidělování proces, pro nějž se dá rozhraní plně vytvořit na backendu. Nejprve se pomocí statické metody retrieveTests na třídě Test vytvoří List testů, které jsou postupně procházeny. Každá třída, která má vyšší nebo stejný stupeň jako ten, který je minimálně požadován na započetí testu je i se svými studenty začleněna stránky.

Tento blok má formát, kdy se učiteli ukáže seznam všech testů. Pokud na nějaký klikne, zobrazí se mu informace o testu a všechny třídy, kterým může test zadat. Na třídu může opět kliknout a poté se zobrazí všichni studenti, kterým je test možné přidělit. To je možné tehdy, pokud jim test již nebyl přidělen. Jestliže chce student někomu přidělit test, klikne na checkbox u jeho jména. V případě kliku na checkbox třídy jsou označeni všichni studenti. V případě kliknutí na checkbox testu jsou označeni všechny třídy společně s jejich studenty.



Například takto může vypadat jeden prvek bloku přiděl testy. To, jak má vypadat HTML je uloženo ve třídách Test a Student, viz. javadoc ve zdrojovém kódu.

Obrázek 8 – Přidělení testů

Po stisknutí tlačítka přidělit testy se spustí servlet EngageTest. Ten uloží do databáze do tabulky engaged_tests informace o přidělení testu. Zároveň musí servlet zkontrolovat, že učitel zadává testy jen studentům, které učí. Toho je docíleno pomocí metody teachesStudent volané na objektu učitele.

3.1.3. Spuštění

Spuštění testů má podobný design, jako přidělování testů tím rozdílem, že tentokrát se nejvýše v hierarchii bloků nevyskytuje test, nýbrž třída. Po rozbalení třídy je možné vidět všechny studenty dané třídy. Ty je možné dále rozbalit a označit tak všechny testy, které je možné u nějakého studenta spustit.

Po odeslání formuláře jsou data odeslána na servlet RunTest. Ten k aktuálnímu času přičte čas na vypracování testu a určí tak čas, kdy musí být test odevzdán. Tento čas je poté vložen do databáze do sloupce deadline. Při příštím načtení stránky již tento test nepůjde u daného člověka spustit, protože se na stránku vypisují pouze ty testy, které mají deadline null.

3.2. Student

Na rozdíl od učitele student nemá druhou navigaci a veškeré informace se vyskytují na hlavní stránce. V první sekci se vyskytují dvě tabulky obsahující hodnocení testů. První tabulka obsahuje data povinných testů, zatímco druhá obsahuje data procvičování. U každého testu je možné zobrazit informace pomocí kliknutí na šipku, která zobrazí blok s informacemi.

3.2.1. Test

Test je skládán na backendu pomocí souboru test.jsp. Při odkazu na tuto stránku je vytvořen objekt EngagedTest, jehož ID je předáno jako parametr. Okamžitě se zavolá metoda loadQuestions. Stránka poté vykresluje otázky uložené v tomto obejtku.

Grafickou podobu určují metody ve třídě Question. Aby bylo vypracovávání testu přehledné, je odstraněna klasická navigace. Není tedy možné se odtud odhlásit ani měnit heslo. Místo toho se zde nachází navigace, která obsahuje čísla jednotlivých otázek, kde při kliknutí na nějaké z těchto čísel je automaticky uživatel přesunut na místo, kde se otázka nachází. Zároveň má tato navigace za účel upozornit, které otázky ještě student nezodpověděl. V případě, že student nezaškrtl žádnou možnost u otázky, tak zůstává barva čísla v navigaci bílá. Jestliže zaškrtl nějakou odpověď u všech pod-otázek, rozsvítí se zeleně, aby bylo jasné, že na ni již odpověděl.

Otázky jsou očíslované od 1 až do N. Zadání otázky je napsáno nadpisem 6. třídy, kde je tučně zvýrazněn startovní a maximální počet bodů v otázce. Při najetí kurzorem na tyto informace o bodech je zobrazen tooltip s podrobnými informacemi o bodování.

Každý typ otázky opět vypadá zcela jinak. Otázka typu pravda, nebo lež obsahuje 3 tlačítka typu radio. Pravda, lež, nevím. V případě, že uživatel klikne na nevím, získá startovní počet bodů. Neodpověděl špatně, ale ani dobře, proto nezíská body navíc.

Otázka typu doplňte do textu na zadání nahrazuje všechny tři podtržítka select boxem, který obsahuje všechny možnosti včetně správné. To, v jakém pořadí jsou zobrazeny možnosti je určeno náhodně a při každém spuštění testu je to jiné. Je také možností neoznačit žádnou možnost. Hodnocení je stejné jako při označení nevím u otázky typu pravda, nebo lež. V případě, že student zaškrtl nějakou možnost kromě prázdné u všech select boxů, změní se barva otázky v navigaci na zeleno.

Otázka typu spojte pojem s definicí je řešena tak, že se definice v pořadí, v jakém jsou v databázi vypíší na obrazovku jako číslovaný seznam. Pod ně jsou vypsány definice, kde u každé z nich je select box s čísly od 1 do počtu možností. V případě, že jsou všechny select boxy zaplněné nějakým číslem, rozsvítí se zelená. Jestliže jsou ovšem ve dvou select boxech v otázce stejná čísla, rozsvítí se číslo červeně, protože situace, kdy na jeden pojem připadají dvě definice nikdy nemůže nastat a jedná se o chybu uživatele.

Podobně je zobrazena i otázka typu seřaď pojmy. Tam je u každého pojmu jeden select box a o rozsvěcení čísel v navigaci platí to samé, co u předchozího typu otázek.

3.2.2. Odeslání odpovědí

Test může být odeslán dvěma způsoby. Buď pomocí tlačítka vlevo nahoře nebo pomocí tlačítka odeslat test. Pokud stisknete tlačítko vlevo nahoře, uloží se vám odpovědi na

server a v případě, že ztratíte spojení se serverem vám po skončení času vyhodnotí server tyto možnosti. Server si pamatuje jen poslední uložené odpovědi. V případě kliknutí na tlačítko odeslat test již student nebude moci test upravovat, tedy proměnná alterable se nastaví na false. Jak v testu, tak i v databázi. O odeslání se stará servlet SaveTestResults, který parsuje data z frontendu a vytvoří nový EngagedTest s těmito daty. Na tomto engagedTestu je poté zavolána metoda saveAnswers, která uloží všechny odpovědi do databáze. Platí přitom, že pokud na nějakou otázku student odpověděl nevím, tak se do databáze vůbec neukládá. V případě, že se odpověď na tuto otázku v databázi již nachází, je přepsána. V případě, že ne, tak je přidána. Tohoto je docíleno pomocí volání uložené SQL procedury přímo v databázi.

Jak servlet, tak i test.jsp jsou chráněny proti pokusům o odeslání odpovědí po termínu nebo odeslání odpovědí na test, který má proměnnou alterable nastavenou na false. O toto se starají metoda isAlterable.

V průběhu testu se může student na vypracovávaný test dostat přes index.jsp. Tato možnost mu bude přístupná až do vypršení časového limitu. Jakmile se tak stane, tak okamžitě po kliknutí na tlačítko pokračovat v testu bude proměnná alterable v tomto testu nastavena na false a nebude ho možné dále vypracovávat.

3.2.3. Vyhodnocení odpovědí

V databázi se nenachází žádná tabulka na zapisování výsledku odpovědí. Je to z důvodu šetření místa. Ve skutečnosti totiž všechny informace potřebné k ohodnocení již máme k dispozici. Máme odpovědi na otázky a instrukce na dávání bodů. O ohodnocení se stará metoda loadReachedPoints. Ta načte do proměnné reachedPoints v EngagedTestu. Na vyhodnocení odpovědí je nejprve potřeba zavolat metodu loadAnswers, která načte odpovědi z databáze. Poté je pomocí metody isRight na objektu Option zjišťováno, zda jsou odpovědi správné, nebo ne. Jestliže odpověď je správná, je na danou pod-otázku přidán počet bodů za správnou odpověď. Jestliže je odpověď špatná, je odebrán počet za špatnou odpověď. Jestliže je selectedOption na objektu Option -1, poté student odpověděl, že neví a nic mu není přičteno, ani odečteno.

Každá možnost si svoje odpovědi ohodnocuje jinak, proto je metoda isRight abstraktní. U otázek pravda, nebo lež je nutné zkontrolovat hodnotu v databázi a tu porovnat s odpovědí studenta. U otázek typu doplňte do textu je odpověď správná, pokud

selectedOption nabývá hodnoty 0. U zbylých dvou typů otázek je odpověď správná, jestliže selectedOption nabývá stejné hodnoty jako subquestionNumber. Pomocí těchto údajů a metod je na stránce index.jsp sestavena tabulka povinných a nepovinných testů. Pro povinné testy platí, že si může student vybrat, jaké časové období chce zobrazit. K tomu slouží select tag, který při změně odesílá formulář, jenž znovu-načte stránku a při tom předá parametr, ve kterém je obsažena informace o novém časovém období.

Student se může po skončení testu podívat na své odpovědi, kde je u každé otázky zobrazen počet bodů, které za odpověď na otázku dostal. Není přitom ovšem vidět, která odpověď je správná, což bude přidáno v budoucnu.

Závěr

Nepodařilo se splnit veškeré cíle, které byly stanoveny. Chybí automatické testování zdrojových kódů, celá administrátorská třída, možnost učitele nahlížet do klasifikace a skládání testů z již hotových otázek. Tyto. Jelikož chci práci přihlásit do soutěže SOČ, všechny tyto cíle budou do té doby splněny. Celý program čeká masivní předělávání, protože toto je má první zkušenost s webovými aplikacemi a to si vybralo značnou daň na kvalitě zdrojových kódů. Pro toto předělání bude využit Spring framework.

Citace

Colorlib. (21. 4 2018). *login-form-v14.* Načteno z colorlib.com: https://colorlib.com/wp/template/login-form-v14/

Cresnar, G. (nedatováno). *garbage icon.* Načteno z flaticon.com: https://www.flaticon.com/free-icon/garbage_126468

Google. (nedatováno). *chevron down icon.* Načteno z flaticon.com: https://www.flaticon.com/free-icon/down-chevron_566015?term=chevron%20up&page=1&position=2

Google. (nedatováno). *chevron up icon.* Načteno z flaticon.com: https://www.flaticon.com/free-icon/up-chevron_566014?term=chevron%20up&page=1&position=8

Market, V. (nedatováno). *upload icon.* Načteno z flaticon.com: https://www.flaticon.com/free-icon/cloud-computing_428535?term=upload&page=1&position=77

Millington, S. (3. 4 2020). *Hashing a Password in Java.* Načteno z baeldung.com: https://www.baeldung.com/java-password-hashing

Pankaj. (2014). *Send Mail in Java using SMTP.* Načteno z journaldev.com: https://www.journaldev.com/2532/javamail-example-send-mail-in-java-smtp

srip. (nedatováno). *Plus icon.* Načteno z flaticon.com: https://www.flaticon.com/free-icon/add_1237946?term=plus&page=1&position=1

Seznam obrázků

Použité programy a technologie

- Microsoft Word
- JDK 8u231
- Netbeans 8.2 IDE
- Tomcat 9
- MySQL Workbench
- MySQL Connector 8.019
- SASS, HTML, CSS, bootstrap 4.1
- Javascript, jquery 3.2.1

Přílohy:

Database.sql - databáze