Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Jakub Švorc
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Webový e-learningový portál pre výuku programovania
Web e-learning portal for programming education

Anotácia: Cieľom bakalárskej práce je navrhnúť a implementovať webový e-learningový portál pre výuku programovania. Tento portál integruje prezentačné a cvičebné zložky výuky. Umožní poskytovať učebné materiály, streaming prednášok a ich ukladanie s možnosťou prehrávania. Umožní tvorbu programovacích cvičení, editovanie kódu, ladenie a zbiehanie vo vybranom programovacom jazyku. Poskytne možnosť vytvárania kurzov s prednáškami, cvičeniami a ich základný menežment. Systém bude realizovaný pomocou technológii/nástrojov: PostgreSQL, ASP.NET / C#, JavaScript a iné.

Annotation: The aim of the bachelor thesis is to design and implement e-learning web application for education of programming. This application provides presenting and exercising aspects of classes. Portal also provides learning resources, streaming lectures which can be recorded and replayed later. It allows producing programming exercises in certain programming language, editing code, debugging and compilation functions. Also it let users to make courses with lectures, exercises and their overall basic management. System will be build on top of technology like PHP (framework), PostgreSQL, ASP.NET, C# and others.

Vedúci: Ing. František Gyárfáš, CSc./
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.
Dátum zadania: XX.10.2020
Dátum schválenia: doc. RNDr. Damas Gruska, PhD.
garant študijného program

.....
študent

.....
vedúci práce

Poďakovanie

Týmto ďakujem svojím spolužiakom a školiteľovi, Ing. Františkovi Gyarfášovi, CSc., za podporu, usmerňovanie, užitočné rady a cenné pripomienky počas celého obdobia tvorby tejto práce.

Abstrakt

Švorc, Jakub: *Webový e-learningový portál pre výuku programovania*. Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky. Vedúci práce: Ing. František Gyárfáš, CSc.

Cieľom tejto bakalárskej práce je navrhnúť a vytvoriť webovú aplikáciu pre výuku programovania, podrobnejšie rozdeliť vyučovací proces do menších celkov a podrobnejšie pokryť tému problematiky, ktorej sa bude lekcia venovať. Práca analyzuje niektoré už existujúce aplikácie tohto zamerania, špecifikuje požiadavky a popisuje implemetnáciu jej kľúčových častí. V záverečnej kapitole je práca zosumarizovaná a popisuje možné rozšírenia v budúcom vývoji.

Kľúčové slová: elarning, webová aplikácia, vyučovanie, programovanie, testovanie

Abstract

Švorc, Jakub: *Web e-learning portal for programming education*. Comenius University in Bratislava, Faculty of mathematics, physics and informatics. Supervisor: Ing. František Gyárfáš, CSc.

Purpose of this bachelor thesis is to implement and design web application for programming education, divide educational process into smaller units and cover whole topic of lesson in more detail. Thesis analyzes some existing applications of similar purpose, specifies requirements and describes implementation of its key elements. Last chapter sumarizes thesis and present possibilities for extending the application in future development.

Key words: elearning, web application, education, programming, testing

Obsah

Úvod

Od roku 2019, kedy vypukla pandémia vírusu Covid-19, sme boli nútení sa vzdelávať dištančnou formou. Najčastejšie vo forme video hovorov so zdieľaným obrazom prednášajúceho alebo cvičiaceho spolu s ústnym výkladom.

Cieľom tejto bakalárskej práce je vyvinúť webový vzdelávací portál, ktorý zjednotí rôzne používané nástroje a prostredia do jedného. Umožní testovanie riešení programovacích úloh, poskytne základnú administratívu a menežment hodiny, umožní zdieľať materiály z hodín, prednášok a cvičení, prehrávanie nahratého videa z hodín.

Úlohy budú zadané vo forme textu na stránke lekcie. K úlohám testy na servery, ktoré budú zverejnené pre žiakov.

Pred spustením testu a otestovaním riešenia sa potrebné súbory uložia na server do dočasného priečinku, odkiaľ sa pošlú do prostredia sandboxu, kvôli bezpečnosti, a následne v ňom sa spustia.

Od vypuknutia pandémie sa téma e-learningu a online vyučovania stala veľmi živou a často zmieňovanou. Práve táto situácia bola motiváciou k voľbe tejto témy. Vytvoriť projekt, ktorý by bol v tomto čase užitočný a vítaný je teda hlavný faktor, ktorý ovplyvnil výber a zameranie témy.

Pre výber technológií som prešiel niekoľko prieskumov najpopulárnejších a najrozšírenejších technológií na vývoj webových aplikácií, z ktorých som vyberal. Okrem popularity som vzal do úvahy aj podporu pre vývojárov a spracovanie dokumentácie pre danú technológiu.

Text práce je rozdelený do nasledovných celkov: oboznámenie s problematikou, popis, návrh aplikácie a implementácia, a záver. V úvode sa taktiež dozvieme, ktoré technológie sú použité pri implementácií aplikácie, podobné existujúce aplikácie a portály a ešte niečo bližšie k problematike dištančného vzdelávania.

1. Oboznámenie s problematikou

V tejto časti uvediem problematiku, ktorou sa táto bakalárska práca zaoberá, vysvetlím niektoré kľúčové pojmy a dodatočnú motiváciu k výberu tejto témy

1.1. Úvod do problematiky

Ako už bolo v úvode spomenuté, situácia spojená s pandémiou ma motivovala k výberu tejto témy. Jeden z problémov, ktoré som postrehol je ten, že v rámci jednej fakulty alebo aj katedry sa na vysokých školách a univerzitách používa viacero komunikačných prostriedkov na výuku dištančnou formou. Voľba viacerých prostriedkov môže miasť študentov a viesť k nedorozumeniam.

Riešiť problém online vyučovania sa začal bez prípravy a mnohí boli odkázaní na to, aby sa učili zaobchádzať s rôznymi nástrojmi a organizáciou „za pochodu“, bez prípravy. Mnoho nástrojov, s ktorými som prišiel do kontaktu poskytovalo mnoho funkcií, z ktorých bola využitá veľmi malá časť, aplikácia teda bola náročnejšia na počítače a prostriedky napriek tomu, že sa jej veľká časť nevyužila.

1.2. Existujúce funkčné aplikácie a prostriedky

- Programiz: Learn to code for free [1]

Online python interpreter nástroj, okrem pythonu má aj C, C++ a Java kompilátory. Obsahuje aj študíjny materiál pre viacero jazykov ako C#, Kotlin, Swift a podobne. Materiál na študovanie je vo forme videí a textových poznámok ku kódu.

Neumožňuje testovanie riešení. Jej meno nie je veľmi rozšírené napriek jej účelu. Má však aj mobilnú aplikáciu, pomocou ktorej je možné aj v telefóne programovať keď nie je možné používať počítač alebo väčšie zariadenie.

- Python Tutor - Visualize Python, Java, JavaScript, C, C++, Ruby code execution [2]

Online nástroj, používaný a odporúčaný RNDr. Andrejom Blahom, PhD. na cvičeniach programovania, ktorý umožňuje programovať online, bez potreby sťahovania

prostredí, kompilátorov, interpreterov alebo iných nástrojov, na spúšťanie kódov a programovanie.

Ponúka širšiu škálu programovacích jazykov a má možnosť graficky zobrazovať to, čo sa deje na pozadí programu, ako napríklad stav pamäte, referencie na objekty v pamäti, uľahčuje vykreslovanie binárnych stromov, spájaných zoznamov alebo iných dátových štruktúr.

Tento nástroj neposkytuje možnosť zdieľania vzdelávacieho materiálu, ani nemá možnosť testovania programov alebo riešení. Slúži výhrane ako online prostredie na programovanie pre jedinca. Pracovať v skupine alebo chatovať v týme nie je s týmto nástrojom možné. Menšia ponuka programovacích jazykov je oproti iným online nástrojom nevýhoda.

Vizualizácia programov, krokovanie behu programu je veľmi dobre spracovaná, užitočná najmä, keď sa používateľ učí o referenciách, pamäti a objektoch.

- W3Schools [3]

Webový portál zameraný na vyučovanie programovania od webového konzorcia, kde vyučovanie prebieha iba pomocou textového výkladu a ukážkových kódov, ktoré je možné spustiť a otestovať.

Okrem toho, že na stránke je ponúkaný spúšťateľný vzorový kód, môže tento kód používateľ meniť, skúšať a spúšťať svoje kódy, ktoré však bežia v obmedzenom režime, kvôli bezpečnosti.

Veľká výhoda tohto portálu je, že ponúka širokú škálu tutoriálov pre veľké množstvo programovacích jazykov. Ku každému z týchto jazykov je možnosť programovať online v online editoroch v obmedzenom režime.

Dizajn tohto porátlu je tiež kvalitne a dobre vyhotovený, ľahká orientácia, prehľadná navigácia a rozdelený obsah do logických celkov je používateľky priateľký a prehľadný.

• Osobné stránky a portály profesorov a predmetov na univerzite

Mnoho profesorov má vlastné portály a stránky, kde zvereňujú študijný materiál, prevažne textovou formou. (napr. osobná stránka profesora Gyarfáša [4], profesora Blaha [5, 6],

profesora Šimka [7], a pod.)

Každá stránka je zameraná na výučbu konkrétnej jednej problematiky. Nakoľko každý predmet a/alebo profesor má svoju vlastnú stránku, stránky nie sú jednotné, nemajú jednotný dizajn a možnosti.

Niektoré stránky slúžia iba na zdieľanie študijného materiálu, niektoré ponúkajú aj možnosť testovania alebo online kódovania a posielanie rešení na zadania a úlohy.

• LIST [8, 9]

Univerzitný systém, ktorý okrem možnosti zdieľania zadaní, materiálov a úloh poskytuje aj možnosť zbiehania testov, základný menežment a ukladanie predošlých verzií riešení. Zadania sú väčšinou ručne napísané alebo v priloženom textovom súbore. Využíva sa najmä jeho testovacia časť a materiál alebo zadania sa zverejňujú na iných webových stránkach.

Má jednoduchú a prehľadnú vizuálnu stránku, ponúka testovanie riešení vo viacerých jazykoch a taktiež zdieľanie materiálu na študovanie, či vo forme súborov alebo priamo na stránke zobraziť ako textový obsah.

- Moodle [10]

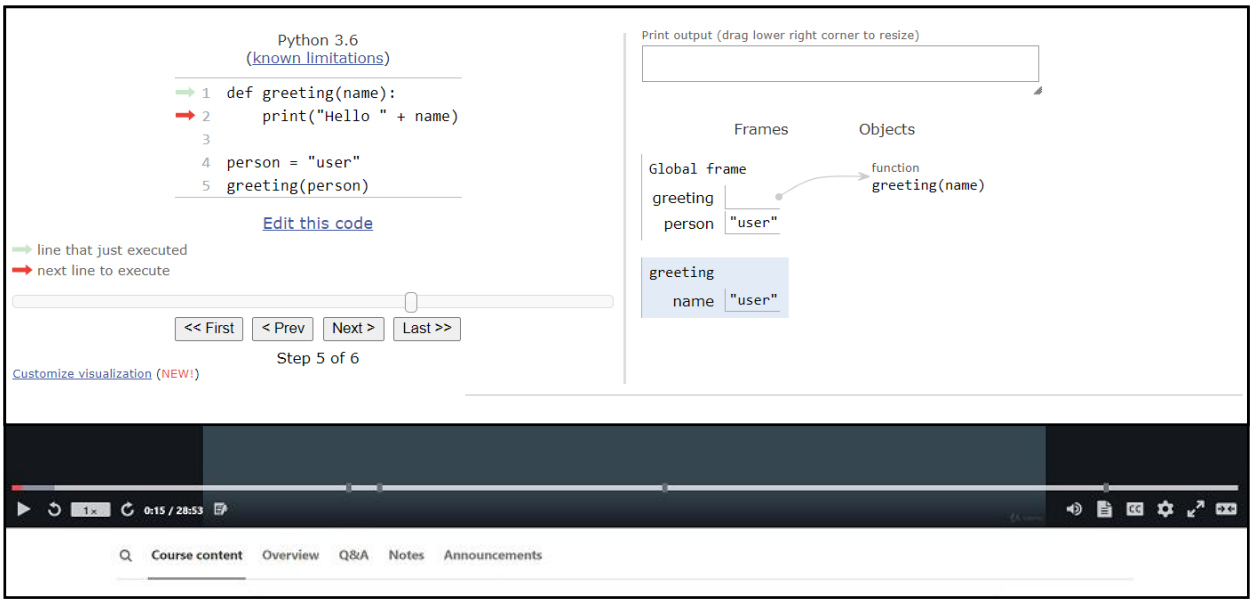
Podobne ako LIST, je aj Moodle univerzitný systém, pomocou ktoré je možné komunikovať vďaka chatu a posielaním správ, je možné na ňom zdieľať zadania, súbory a odovzdávať riešenia, ktoré však nie je možné spúšťať, teda sa väčšinou jedná o textové riešenia, nie o programovacie.

- Udemy [11]

Online škola programovania, kde je možné materiál na výučbu zdieľať najmä video formou, teda celý výklad a ukážky sú vo videu, alebo pomocou textových súborov (napr. PDF).

Má príjemný farebne zladený a dizajn, obsah rozdelený do logický celkov a intuitívnu navigáciu a orientáciu po stránke. Ponuka vzdelávacieho materiálu je široká, okrem množstva programovacích jazykov a kurzov ponúka aj umelecké, finančné alebo iné kurzy.

Nevýhodou tejto školy je, že videá musí používateľ ručne spracovať, upraviť a nahrať do kurzu, sú zvyčajne dlhé a zadanie ktoré vedúci kurzu zadá nie je možné skontrolovať softwareom a teda správnosť riešenia si musí účastník skontrolovať ručne alebo na fórach.



Jednou z úloh aplikácie k tejto bakalárskej práci je zjednotenie týchto a podobných nástrojov a aplikácií, poskytnúť na jednom mieste možnosti ktoré každá z nich ponúka osobitne. Teda spojiť možnosť komunikovať a zdieľať študijny materiál. Tým, že aplikácia je vo webovom prostedí, oprostíme používateľov od inštalovania softwareu.

- Free Code Camp akadémia [12]

Online nezisková organizácia, ktorá poskytuje video tutoriály programovania zadarmo pre širokú škálu programovacích jazykov. Všobecne má dobré renomé a je to populárna platforma pre začiatočníkov.

Tutoriály sú vo forme videí alebo textovej podobe s obrázkami blokmi kódu, ktorý je vysvetlený. Vyučujúci sú dobrovoľníci, ktorý majú pracujú ako programátori, a však ich predagogické skúsenosti môžu byť nedostatočné, žiadne alebo aj výborné.

1.3. Podobné bakalárske práce

Témou e-learningu, možnostiam programovania v online priestore, testovania programov a online menežmentom úloh a zadaní sa v minulosti zaoberali viaceré práce, ktoré som použil pri tvorbe aplikácie.

- VIRTUÁLNE PROSTREDIE PRE TVORBU INTERAKTÍVNYCH MULTIMEDIÁLNYCH KURZOV PROGRAMOVANIA

Aplikácia vyvinutá študentom aplikovanej informatiky Marekom Letavayom [13] v 2013. Slúžila na podobné účely – teda online výuku a fungovala podobným štýlom, teda lekcie a vyučovanie prebiehalo pomocou videa. V krátkom videu sa študent oboznámil s problematikou. Následne vypracoval riešenie, ktoré prešlom naprogramovaným testom a študentovi sa zobrazil výsledok s vyhodnotením testu.

Používateľ má možnosť pracovať aj v režime, kedy si píše svoje vlastné kódy mimo zadania, má možnosť také svoje kódy spúšťať, pozorovať a skúmať.

Práca ma motivovala jej funkciami a možnoťami ktoré ponúka – zbiehanie testov, použitiu videí ako primárny študíjny materiál a aj jej dizajnom a jednoduchým a intuitívnym ovládaním a jednoduchou administráciou a menežmentom.

Nevýhodou tejto práce je, že neposkytuje možnosť komunikácie ako s vyučujúcim tak ani s inými študentami a taktiež nie je úplne jasné ako tieto videá musel vyučujúci spracovať ručne a akým spôsobom boli vložené do databázy.

Okrem videí táto aplikácia neposkytuje inú možnosť ako zdieľať materiál na študovanie.

- NOVÝ DLHODOBÝ VIACÚČELOVÝ SKLAD ÚLOH NA CIČENIA

Bakalárska práca z roku 2013 študenta aplikovanej informatiky Andreja Jursu [9]. Táto webová aplikácia je využívaná najmä ako skladisko úloh, riadenia ich verzií a testovanie riešení pomocou naprogramovaných testov.

Táto aplikácia umožňuje aj zdieľanie súborov, ktoré si môžu študenti stiahnuť, zadávať úlohy v textovej podobe priamo na stránke na ktorej sa riešenie odovzdáva a možnosť poskytnúť jednoduchú spätnú väzbu od vyučujúceho v podobe poznámky ku riešeniam, ktoré boli nahraté do systému.

Táto práca ma inšpirovala k využitiu podobného prístupu k tvorbe verzií riešení, k navrhnutiu databázového systému a využívaniu a prepojeniu študentských elektronických účtov.

Možnosť posielania správ alebo možnosť chatu, namiesto jednoduchej poznámky k riešeinu táto aplikácia neposkytuje a to je nedostatok. Napriek všetkým možnostiam, ktoré aplikácia ponúka je bežne využívaná len tá časť, ktorá spravuje úlohy, spúšťa testy a vyhodnocuje ich.

1.4. Použité technológie

Pred začatím tvorby portálu k tejto bakalárskej práci som najprv spravil prieskum. Vedel som, že budem potrebovať nástroj na tvorbu webu, takže prvé ktoré som zvažoval boli JavaScript, HTML, CSS a PHP.

Po dokončení prieskumu som sa nakoniec rozhodol vymeniť PHP za technológiu ASP.NET a jazyk C#, kvôli ich objektovo – orientovanému prístupu, dobrou podporou, širokou dostupnosťou rôznych frameworkov a kvalitným vývojovým prostredím.

Pre potreby práce s databázou som zvolil PostgreSQL systém, ktorý je veľmi populárny a k jeho výberu ma motivovalo, že na prácu s ním sú vedené predmety na univerzite, ktoré som absolvoval a taktiež kvôli tomu, že framework ASP.NET už disponuje spôsobom, ktorým riadi a ovláda databázu bez potreby inštalovania dodatočných knižníc.

1.4.1. JavaScript, jQuerry, CSS, HTML

Štandardné technológie vo webovom prostredí. CSS – kaskádové štýly spravujú vyzuálnu časť webovej stránky, HTML – značkovací jazyk riadi obsah stránky a JavaScript môže riadiť oboje, ako aj vytvárať dynamický obsah stránky bez potreby opätovného načítania celej stránky ale iba niektorej jej časti, interakciu s používateľom a iné, jQuerry – najpopulárnejšia JavaScript knižnica, ktorá zjednodušuje používanie JavaScriptu na webových stránkach, zaobaľuje zložité skripty a kódy do jednoduchých volaní funkcií alebo procedúr a taktiež pomocou jQuerry je vytvorených množsto pluginov jednoduchých na implemetovanie do webovej aplikácie.

1.4.2. ASP.NET

Moderný populárny framework spoločnosti Microsoft. Tento framework je navrhnutný práve na tvorbu webových aplikácií. Umožňuje jednoducho riadiť komunikáciu

medzi používateľom a serverom, zjednodušuje prácu s databázou a štandardne rozdeľuje aplikáciu do troch logických celkov, podľa návrhového vzoru MVC. [14]

Podľa tohto vzoru je aplikácia rozdelená do Model – View – Controller častí.

1.4.2.1. Model

Vo vzore MVC model predstavuje dáta. Model môže byť zložitý objekt alebo aj niektorý z primitívnych dátových typov (celé číslo, znak, pravdivostná hodnota *true* alebo *false* a podobne). Slúži ako kontajner dát, metód, funkcií, stavov a komponentov (napr. databázové spojenie, súbory, a pod.). Pomocou modelu vieme prenášať objekty (prípadne dáta) medzi controller a view. Taktiež môže byť jeden model zdieľaný naprieč celou aplikáciou alebo byť použitý iba na jedinej stránke.

1.4.2.2. View

View je časť aplikácie, ktoré sa skladá iba z html (prípadne cshtml alebo iných hypertextových) súborov. Zastrešuje vizuálnu časť aplikácie, teda to, čo je používateľovi zobrazené v prehliadači. Technicky, do časti view patrí front – end časť, teda výstup dát po spracovaní, ich formátovanie, grafické prezentovanie a štruktúra.

1.4.2.3. Controller

Pomocou controller je možné spravovať models a views. Controller zabezpečujú akési triedenie, po obstaraní požiadavky je zavolaný controller. Ten podľa URL adresy zavolá svoju príslušnú funkciu, spracuje požiadavku, vyhodnotí ju a rozhodne ktorý view a model pošle ako odpoveď. Predstavuje jedro systému, jeho logickú časť a celkovo správanie aplikácie.

1.4.2.4. Route

Route zaobstaráva posielanie požiadaviek správnemu controllers a volanie správnej akcie. Sú to vzory znakových reťazcov, ktoré popisujú ktorý controller bude spracovávať požiadavku na základe toho, do ktorého vzoru reťazca spadá konkrétna URL adresa.

Napr. adresa so vzorom *https://server.sk/c/m/d* znamená, že požiadavku na tento server bude spracovávať controller *c*, ktorý ju spracuje v metóde *m* a do tejto metódy mu budú namapované dáta *d*.
Je bežnejšie, že dáta a hodnoty sú posielané priradením do premenných s nejakým názvom, takže vzorová adresa by mohla mať aj tvar *https://server.sk/c/m?d=1*. Mapovanie bude fungovať podobe ako v prvom prípade, ale do metódy *m* sa do parametra s názvom *d* predá hodnota *1*.

Pri väčšine zdrojov sa časť route neuvádza, je nepovinná. Pre úplnosť však túto časť uvádzam.

1.4.3. C#

Populárny objektovo – orientovaný staticky typový jazyk spoločnosti Microsoft z rodiny jazykov C. Je možné pomocou neho ovládať ASP.NET framework, .NET framework a všetky jeho Core verzie.

Core označenie znamená, že výsledná aplikácia môže byť nasadená do behu bez ohľadu na počítač a jeho konfiguráciu, na ktorom bola zostrojená a môže rovnako bežať na počítačoch bez ohľadu na ich vlastnosti a nastavenia, jedná sa o tzv. cross-platform.

Verzie bez označenia Core sú optimalizované pre prácu a budovanie aplikácií prevažne pre systémy Windows. Okrem toho, verzie Core ponúkajú aj podporu pre širšie spektrum aplikácií (formulárové aplikácie, webové aplikácie s použitím ASP.NET Core, desktopové aplikácie, atď.).

Podobný je aj vzťah medzi ASP.NET a ASP.NET Core.

1.4.4. PostgreSQL [15]

Open – source databázový objektovo – relačný systém. Poskytuje vysokú flexibiltu čo sa týka možností prispôsobiť si ho pre svoje potreby. Má veľké množsto už zabudovných bezpečnostných prvkov, typovú kontrolu, dobrú dokumentáciu a existuje veľa nástrojov, ktoré prácu s PostgreSQL uľahčujú.

Má jednoduchšiu syntax v porovnaní s inými databázovými systémami, obsahuje viaceré mechanizmy na zlepšenie výkonu (napr. indexovanie), širšiu škálu dátových typov, jednoduchú tvorbu vlastných funkcií, možnosť ich preťažovania, rôzne zabudovné aj užívateľom definované jazykové konštrukcie (napr. konštrukcia *with, serial*) a je užívateľsky priateľskejší.

Je možné takýto systém aj rozširovať o rôzne funkcie, používať v ňom procedulárne jazyky, ako Python, C, Perl, PL/PGSQL a iné. Taktiež poskytuje aj rôzne množinové operácie a teda možnosť pracovať s tabuľkami a výsledkami dopytov ako s množinami.

1.4.5. Sandbox

Keďže zdrojové súbory pre testy budú pochádzať z rôznych zdrojov, je treba zabezpečiť, aby program, ktorý test spustí nemohol zahltiť miesto na disku, prepisovať alebo čítať systémové súbory, prehladávať miesto na disku, ktoré nie je pre používateľa určené a iné nezvrátiteľné škody alebo utajené informácie.

Existuje viacero metód, spôsobov alebo krokov, ktorými sa dá zabezpečiť istý stupeň integrity a bezpečnosti [17], napríklad zavedním whitelistov alebo blacklistov. Avšak väčšina z nich sa dá skúseným programátorom obísť, nakoľko python neposkytuje modifikátory prístupu. Je teda možné nízkoúrovňové objekty sprístupniť, meniť, nastavovať alebo zamaskovať a tak obísť mnohé obmedzenia whitelistu alebo blacklistu.

Riešením je použitie sandboxingu [16] – z angličtiny pieskovisko. Sandbox je miesto určené na testovanie a skúšanie programov bez toho, aby došlo ku škodám, ak program pochádza od neznámeho zdroja. (V prenesenom význame je to pieskovisko, kde sa „hrajú“ programátori a vývojári a testujú rôzne programy.)

Sandbox je podobný virtuálnym systémom, ale je v nich značný rozdiel. Narozdiel od virtuálnych systémov je sandbox menší z hľadiska kapacity na disku, jednoduchší a menej rozbustný. K tejto bakalárskej práci používam sandbox Sandboxie. [18]

Pred spustením programu sa do prostredia sandboxu pošlú súbory, ktoré majú byť spustené alebo otestované. Tie sa v sandboxe spustia. Ak by program mal bežať neobmedzene dlho (napr. obsahuje nekonečný cyklus) po určitom časovom limite sa sandbox vypne. Všetky

súbory, ktoré by boli programom vytvorené sa po ukoneční sandboxu zmažú, pokiaľ pri spustení nie je nastavené inak.

Prostredie v sandboxe je skopírované z priečinku, z ktorého je spúštaný – teda sandbox je kópia akutálneho používateľa, jeho súborov, priečinkov a iného obsahu. V sandboxe tak nie je možné čítať súbory, ktoré nie sú pre používateľa určené a ktoré sú mimo aktuálneho priečinku. Takto Sandboxie zamieta prístup mimo akutálny priečinok, tým že súbory, ktoré nie sú v pracovnom priečinku neexistujú. Napríklad ak je súbor, ktorý chceme otestovať v priečinku *D:\priecinok*, je program obmedzený iba na tento priečinok a zapisovanie alebo čítanie mimo neho skončí chybou neexistujúceho priečinku alebo súboru.

Sandbox a všetky jeho programy bežia vo vlastnej reží a sú izolované. Z toho dôvodu nie je možné priamo získavať výstupy zo sandboxu klasickými metódami ako je predávanie výstupu. Najjednoduchším riešením, ktoré rieši problém získavania výstupu zo sandboxu je presmerovanie výstupu do súboru a následne obsah tohto súboru vrátiť používateľovi.

Ďalší problém, ktorý vyplýva z toho, že všetko čo sandbox spúšťa izolovane je v tom, že ak sa program nepodarí spustiť, informácia o neúspešnom pokuse a dôvode zlyhania je nedostupná. Jedným z dôvodov, kedy program zlyhá pred spustením, je syntaktická chyba alebo pri zlom odsadení riadkov. To sú chyby, ktoré pôvodne detekuje vývojové prostredie a zabraňuje spusteniu pred odstránením týchto chýb. Je však možné chybový výstup presmerovať do súboru, rovnako ako štandardný.

1.4.6. JQuery

Najpopulárnejšia JavaScript knižnica, ktorá uľahčuje a urýchľuje prácu s javascriptom, manipuláciu s udalosťami a pristupovanie k HTML DOM elementom (Document Object Model). Narozdiel od čistého javascriptu, jQuery šetrí kód a miesto, zlepšuje čitateľnosť a je veľmi užívateľsky priateľský a umožnuje jednoduché pridávanie pluginov a iných doplnkov.

Spoločnosť Microsoft zabudovala jQuery do frameworku .NET, ASP.NET a jeho Core verzií, teda nie je potrebné túto knižnicu inštalovať, nachádza sa v nich automaticky.

1.4.7. Rich Text Editor

JQuery WYZIWYG plugin, ktorý umožňuje úpravu textu bez používania HTML DOM tagov a atribútov. Formát a obsah textu, ktorý je zobrazený v editore je na pozadí zapisovaný a formátovaný pomocou tagov a atribútov. Zapísaný text pomocou HTML je potom možné jednoducho vkladať do stránky a upravovať.

2. Špecifikácia a návrh aplikácie

V nasledujúcej kapitole budem popisovať návrh aplikácie, jej funkcionálne požiadavky a špecifikácie.

2.1. Priebeh výučby

Štandardný vyučovací proces pozostáva z dvoch primárnych častí, podľa toho, ktorým smerom sú predávané informácie:

1. Od učiteľa smerom ku študentom, najčastejšie v podobe prednášky, výkladu, poznámok, skrípt alebo iného študijného materiálu.

2. Od študentov smerom k učiteľovi v podobe domácej úlohy, skúšky, projektu, referátu a podobne.

Problémom takýchto štandardných postupov je náročnosť a objem informácií s obmedzeným časom, čo má za následok to, že nie každá téma spojená s problematikou môže byť rovnako podaná a následne otestovaná, teda niektoré témy budú podané a naštudované menej podrobne alebo vôbec.

Úlohou tejto aplikácie je poskytnúť možnosť obidva tieto smery rozdeliť na menšie a následne tieto menšie celky otestovať, čím sa problematika rozloží do menších častí, zložitejšia problematika sa rozdelí do menších celkov, ktoré je jednoduchšie spracovať.

Aplikácia musí pracovať aj s videom, ako so študijným materiálom, v ktorom bude vysvetlená problematika. Pretože obsah videa pojednáva o viacerých témach, je potrebné, aby bolo možné video rozdeliť pomocou časových značiek a vyhradiť tak čas videa, ktorá sa venuje jednej problematike. Okrem toho, musí byť možné k videu priložiť aj textové poznámky, komentáre, vzorové kódy alebo zadanie, ktoré treba spliť a vypracovať a následne toto zadanie otesovať pomocou unit testov.

2.2. Unit testing

Unit testing (jednotkové testovanie) je metóda, pomocou ktorej sa testujú jednotlivé časti kódu. Kód je rozdelený do tzv. jednotiek. Jednotka kódu je pomyselná najmenšia časť kódu ktorá sa dá a ktorú má zmysel otestovať. Najčastejšie to bývajú funkcie, procedúry alebo metódy objektov a tried. V niektorých prípadoch môže byť jednotkou aj celý program, v závislosti od zadania.

Na testovanie sa najčastejšie využívajú moduly, knižnice, frameworky alebo iné nástroje mimo hlavného programu. Unit test pozostáva z test casov (testovacích prípadov), ktoré predstavujú rôzne scenáre a manipulujú s hlavným programom tak, že volajú jednotlivé jednotky a podsúvajú im testovacie dáta, ktoré sú vopred nastavené. Môžu byť aj falošné alebo zámerne nesprávne pre beh programu. Tým sa otestuje či program správne pracuje a ako reaguje na nesprávne dáta.

Najčastejším scenárom je, že jednotky vracajú nejaké hodnoty alebo nastavujú stavy, ktoré sú potom porovnávané s očakávanými hodnotami a stavmy a na základe tohto porovnania je určený výsledok testu. Okrem hodnôt a stavov je možné kontrolovať aj to, či program spôsobí výnimku, či typ výnimky zodpovedá očakávanej výnimke a podobne.

Niekedy je možné testy využiť aj inak ako na testovanie správnosti existujúceho kódu. V prípade testami riadeného vývoja (*test driven development*) sú najprv zostrojené testy a až následne k nim je dopisovaný hlavný program.

Správnym rozdelením a zostrojením scenárov je možné presne určiť chybu, jej dôvod a iné dôležité informácie. Preto je dôležité správne navrhnutie a realizovanie testovacích scenárov.

2.3. Version – control

Aby mal používateľ prístup k svojím predošlím súborom, je potrebné ich ukladať a sprístupniť. Preto aplikácia disponuje jednoduchým version-controlingom (riadením verzií).

Do databázy je poslaný každý súbor, ktorý používateľ prostredníctvom aplikácie spustí alebo odovzdá ako riešenie. Tieto súbory sú zviazané s používateľom, čo umožňuje jednoduché filtrovanie ktoré patria tomu-ktorému používateľovi.

Súbory v databáze sú rozlišované pomocou vnútorných mechanizov(riadi ich sama databáza pomocnými štruktúrami) – čísla ID, ktoré každému riadku priradí jedinečné číslo, ktoré slúži aj ako primárny kľúč tabuľky, čím je možné jednoznačne určiť súbor.

2.4.Spúšťanie a zobrazovanie testov a výsledkov

Aplikácia umožňuje testovať odovzdaný kód pomocou unit testov. Tie sú spúšťané podobným spôsobom ako programy bez unit testov. Rozdiel spočíva v tom, aký súbor má byť spustený.

Testovací súbor, ktorý vedúci kurzu nahrá do databázy je pred samotným testom stiahnutý z databázy, do pracovného priečinka používateľa. Spolu s testom je tam uložený aj odovzdaný zdrojový súbor.

Pred spustením testovania je potrebné sledovať istú štruktúru nastavení: súbor, ktorý je určený na otestovanie sa musí nazývať *solution*, musia sa nachádzať v tom istom priečinku, testovací súbor musí importovať modul *solution*, musí obsahovať funkciu *main*, v ktorej je volaná funkcia *main* unittest modulu (*unittest.main*). a tzv. *main block* (hlavný blok). Hlavný blok je špecialitou jazyka python.

```
def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

Pretože je potrebné, aby testovací súbor nasledoval takúto prísnu štruktúru, je dostupná používateľovi, ktorý má administrátorské práva, jednoduchá šablóna s inštrukciami na vytváranie ľubovoľných testov.

2.5.Spúšťanie programu bez testov

Pred spustením programu v sandboxe, je potrebné program stiahnuť na disk. Program zapísaný v editovacom okne sa stiahne a zapíše do súboru, ktorý je následne uložený do používateľovho pracovného priečinku aj do databázy.

Absolútna cesta k tomuto súboru je potom predaná pri spúšťaní sandboxu, podobne ako pri spúšťaní testov.

Počas behu programu aj unittestu je kontrolovaný čas, ktorý zaznamenáva ako dlho program beží. Po prekonaní 30 sekundového limitu je program natvrdo ukončený a používateľovi je podaná správa o uplynutí časového limitu.

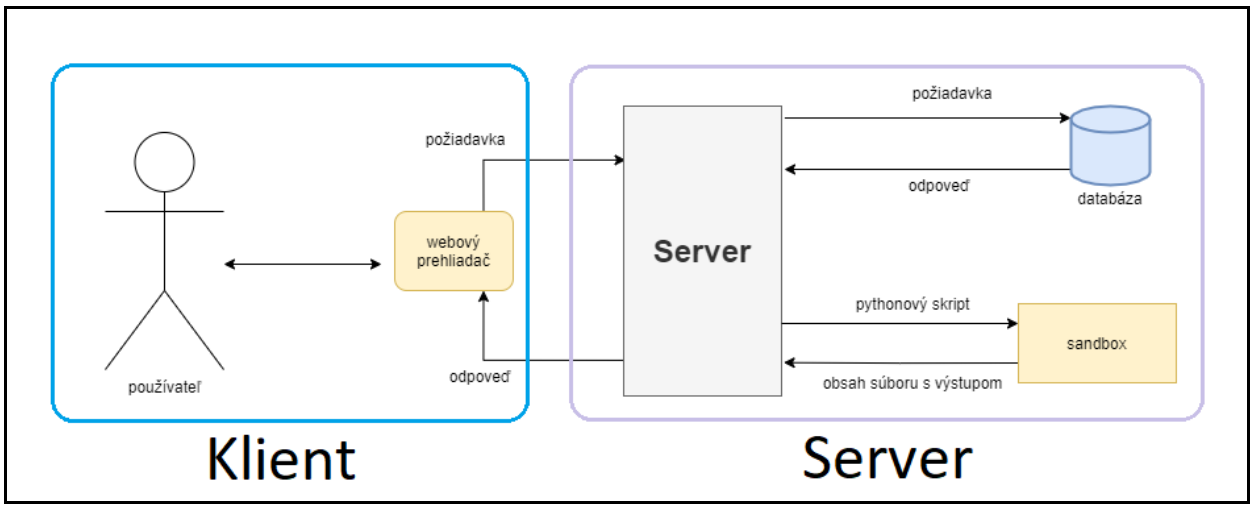
2.6. Klient – server architektúra

Pretože aplikácia je určená pre webové prostredie, je dôležité, aby všetky zmeny, ktoré nastanú v aplikácií, boli odzrkadlené aj u požívateľov. Bolo by nepraktické, aby každý používateľ musel po vykonaní nejakej zmeny sťahovať a prípadne inštalovať celú aplikáciu u seba na počítači. Takýto problém rieši architektúra klient – server [21].

Celá aplikácia je rozdelená do dvoch veľkých častí: klient – predstavuje používateľa, konkrétne jeho prehliadač. Ten komunikuje so serverom, prijíma odpovede a posiela požiadavky. Získané odpovede sú potom klientovi prehľadne a zrozumiteľne zobrazené.

Časť server je tá, na ktorej prebieha celá funkcionalita, kde je uložená logika aplikácie a ktorá vyhodnocuje požiadavky a vytvára odpovede. Na serveri sú úložené aj všetky zdrojové súbory, čo je dôležité z pohľadu bezpečnosti.

Po vyslaní požiadavky od klienta smerom ku serveru je požiadavka spracovaná a vyhodnotená. Požiadavky môžu byť rôzne – spustenie programu, spustenie unit testov, zmena údajov v databáze a podobne. Všetky takéto výpočty a zmeny prebiehajú na serveri a používateľovi je potom poslaná odpoveď v závislosti od požiadavky.



2.7. Vzor Row data gateway

Databázu je možné si predstaviť ako tabuľku, ktorá môže byť menšia aj väčšia. V tejto tabuľke sa nachádzajú hodnoty, ktoré častokrát odkazujú na iné tabuľky a tak podobne. Aby bolo možné, takýto jeden riadok jednoducho získať z databázy a používať v aplikácií je zavedený vzor Row Data Gateway, ktorý tabuľky reprezentuje pomocou tried a jedna inštancia takejto triedy potom reprezentuje jeden riadok.

Okrem toho, príkazy ktoré zahŕňaju jednu tabuľku sú zabalené do jednej triedy, takže zmeny ktoré bude potrebné vykonať budú vykonané na jednom mieste.

Tento vzor je využitý v každej triede models.

2.8. Roly používateľov

Každý používateľ aplikácie má isté práva v závislosti od toho, akú rolu zastupuje. V aplikácií sú využívané práve dve roly a s nimi spojené práva:

- Rola bežného používateľa (študent, user): predstavuje skupiny s obmedzenejšími právami, ich úlohou je riešiť zadania, pozeranie videií, prihlasovanie sa.

- Rola administrátora (učiteľ, admin): má vyššie privilégia, má právo vytvárať kurzy, lekcie a kapitoly lekcií, má právo meniť a upravovať obsah kapitol, pridávať študentov na kurzy.

V tejto bakalárskej práci je rozdiel medzi userom a adminom zatiaľ iba v obsahu stránky – to čo vidí, k čomu má prístup. Neskôr je možné tieto roly a ich práva ešte viac rozšíriť, prípadne pridať viaceré roly.

2.9. Bezpečnosť aplikácie

Nakoľko je aplikácia webového charakteru a spúšťa programy od rôznych používateľov, je potrebné myslieť na bezpečnosť a minimalizovať riziká, ktorými môže byť aplikácia vystavená. Kvoli týmto dvom faktom je potrebné zaviesť bezpečnosť, ktorá chráni systém počasť spúšťania programov a počas práce vo webovom prostredí.

Bezpečnosť spúšťaných programov, ich obmedzenia a správu pamäti s ktorou program používateľa pracuje zastrešuje prostredie sandboxu (viď 1.4.5., 2.4., 2.5.)

2.9.1. Bezpečnosť aplikácie vo webovom prostredí.

Aplikácia príjíma od používateľa rôzne vstupy z formulárov pri registrácií, prihlasovaní, spúšťaní programov, pri zmenách obsahov a podobne. Do týchto formulárov má používateľ možnosť zadať ľubovoľné reťazce, ktoré môžu byť správne – mená, heslá, riadne súbory s riešením, email, môžu byť chybné – používateľ nedopatrením zadá nesprávny vstup, hodnotu, súbor, alebo môžu byť podvrhnuté – používateľ sa pokúsi vkladať kúsy kódov do formulárov, čím môže narušiť štruktúru stránky, odhaliť skryté informácie alebo všeobecne narušiť správny chod aplikácie.

Je preto potrebné, aby aplikácia rozlišovala medzi chybou a podvrhom a aby správne reagovala na takéto situácie.

Jednou z možností ako zabezpečiť aby prípadne podvrhy aplikácia zachytávala a neposúvala ďalej je kontrolovať každý vstup, ktorú používateľ zadá. Toto sa dá čiastočne dokázať, avšak nie je to vždy úplne možné, nakoľko niektoré dobré spôsoby kážu, aby používateľ pracoval s rôznymi vstupmi, ktoré môžu vyzerať ako kúsky kódu. Napríklad silné heslá musia obsahovať špeciálne znaky, teda takýmto heslom môže byť aj niečo, čo vyzerá ako kód, alebo rôzne používateľské mená, emaily a podobne.

Ďaľšou možnosťou je, aby boli takéto vstupy brané ako postupnosť znakov bez významu (takzvaný *raw string*). Teda ak by používateľ zadal do vstupu kód, takýto vstup nebude braný ako kód, ale iba ako postupnosť znakov.

Používaním raw stringov je možné zabrániť viacerým webovým útokom – HTML injection, SQL injection a vkladaniu skriptov.

HTML injecetion je spôsob útoku, pri ktorom je ako vstup poslaný HTML kód, ktorý naruší štruktúru stránky, odhladí skryté prvky alebo zobrazí dáta, ktoré majú byť skryté. Tým, že všetky HTML tagy nebudú parsované ale budú brané iba ako postupnosť znakov, znaku „väčší ako“, „menší ako“, znak rovnosti, úvodzoviek a apostrofov.

SQL injection je podobný spôsob útoku ako pri HTML, ale s tým rozdielom, že namiesto HTML kódu je vkladaný SQL kód, ktorý po parsovaní oddelí hlavný kód od vloženého, najčastejšie je zakomentovaný a teda databázový systém ho nevyhodnocuje, a je nahradený kódom, ktorý vložil útočník, čo môže mať za následok odhladenie, zneužitie, zmazanie alebo sprístupnenie citlivých údajov, častí aplikácie ktoré nemajú byť používateľovi bez správnych oprávnení sprístupnené, môže zmazať a zmeniť dáta uložené v databáze a podobne.

Okrem používania raw stringov je vysoko odporúčané v databázových systémoch používať *prepared statement* – dopyt do databázového systému, do ktorého je najskôr poslaný samotný dopyt a následné dáta sú poslané zvlášť, čo spôsobí, že textové dáta sú spracovávané bez vyhodnocovania.

3. Implementácia

V tejto kapitole popíšem implenetáciu databázového systému, jednotlivých funkčných častí aplikácie a vysvetlím niektorú problematiku, ktorá nemusí byť úplne zrejmá.

3.1. Implementácia databázy

Na uchovávanie dát používam systém PostgreSQL. Na komunkáciu s databázovým systém je potrebné mať nainštalovanú knižnicu pre prácu s databázou. VisualStudio obsahu štandardne knižnicu ODBC na tento účel[19]. Tento systém môže vytvárať rôzne pomocné tabuľky, ale primárnych tabuliek je týchto sedem:

1. users – tabuľka uchováva údaje o používateľoch: ich jedinečné ID, používateľské meno, email, typ prístupových práv (rola) a heslo. Je dobrým programátorským zvyok neukladať heslá v čistom textovom formáte, preto heslá sú pred vložením zašifrované a do tabuľky vložené v zašifrovanej podobe

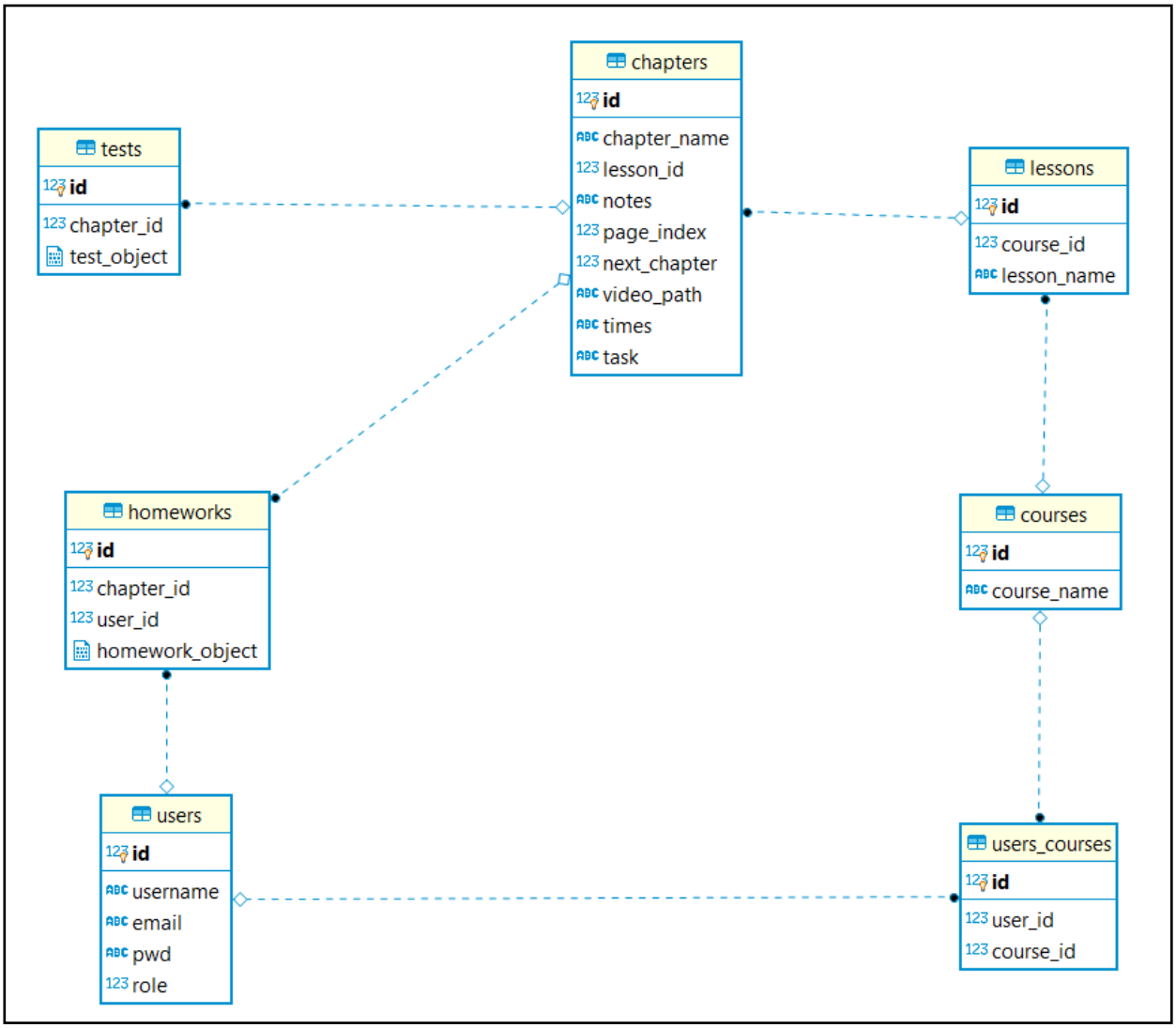
2. courses – v tejto tabuľke sú údaje, ktoré popisujú kurz: jeho ID a názov

3. lessons – obsahuje id hodiny, id kurzu ku ktorému hodina patrí a názov hodiny

4. chapter – tabuľka pre jednotlivé kapitoly. Obsahuje informácie o kapitolách: id kapitoly, názov kapitoly, id hodiny ku ktorej kapitola patrí, poznámky ku kapitole, poradové číslo kapitoly, číslo id ako odkaz na ďalšiu kapitolu, cestu ku videu ktoré patrí do danej kapitoly, časy ktoré ohraničujú začiatok a koniec videa a zadanie úlohy.

5. users_courses – väzobná tabuľka, ktorá vyjadruje vzťah „študent má zapísaný predmet“, pozostáva z id študenta a id kurzu.

6. homeworks – obsahuje všetky programy, ktoré študent sputil alebo odovzdal. Skladá sa z kapitoly, ku ktorej je odovzdané riešenie, používateľa a súboru, ktorý bol spustený alebo nahratý



7. tests – tabuľka je podobná ako homeworks – obsahuje súbor v ktorom je zapísaný test a kapitolu, ku ktorej test patrí.

3.2. Stavba hodiny, kapitoly

Priebeh hodiny a teda jej stavba pozostáva z nasledujúcich častí:

- Práca s video materiálom – študent má k dispozícii video s výkladom ku danej problematike, kde je časovo ohraničená časť videa, ktorá je pre daný podproblém určená. Skrytie video prvku a následne zorazenie texu zabezpečuje javascript. Video počas prehrávania vo webovom prehliadači pravidelne vo veľmi vysokej frekvencií štartuje udalosť *onchange*. Táto udalosť je vyvolaná

pri každej zmene času v elemente video, rádovo na niekoľko desatín milisekundy. Čas, ktorý používateľ vyhradil tomuto videu je reprezentovaný ako dvojprvkové pole s desatinnými číslami predstavujúcimi začiatok a koniec videa. Reagovanie na udalosť *onchange* a kontrola času tak prebieha na presnosť desatín milisekundy. Po dosiahnutí konečného času je video zastavené a nahradené textom zadania.

- Poznámky k výkladu – okrem videa je k dispozícii aj možnosť doplnenia poznámok. Nie sú obmedzované na dĺžke, takže je možné do poznámok prepísať aj celú lekciu, ak to bude potrebné.

- Zadanie – text zadania s inštrukciami, ktorý sa po skončení videa zobrazí.

- Práca s programom a odovzdávanie riešení – po ukončení prehrávania videa, je časť s videom nahradená textom zadania. K tomuto zadaniu používatelia vypracujú riešenie a to odovzdajú na server, kde je následne možnosť ho otestovať. Odovzdať riešenie je možné dvoma spôsobmi: buď odovzdaním súboru cez formulár, alebo je možné riešenie napísať do programovacieho okna, spustiť a následne otestovať.

3.2.1. Zmena obsahu

Meniteľný obsah kapitoly pozostáva z videa a poznámok alebo zadania. Jej obsah je zapísaný do databázy. Pri načítavaní obsahu kapitoly sa tento obsah stiahne a pri potvrdení zmien sa opätovne zapíše.

V kontexte videa, je možné meniť samotné video – video súbor, ktorý je uložený na serveri a ktorý k danej kapitole patrí, alebo je možné meniť časové ohraničenie videa, teda jeho začiatok a koniec.

Textové poznámky alebo zadanie je možné meniť pomocou WYSIWYG textového editora, ktorý umožňuje okamžité zobrazenie formátovania textu. Po potvrdení zmien sa tento text zakóduje nahradením otváracích a uzatváracích znakov tagov to reprezentácie HTML entity (otvaraci tag ‚<‘ sa prepíše na ‚*<*‘ a uzatvárací znak ‚>‘ na ‚*>*‘). Bez nahradenia, ASP.NET

vyhodí bezpečnostnú výniku a zamietne spracovanie takého textu. Podobný proces je využívaný v jazyku PHP použitím funkcie *real_espace_string*, ktorého účel je podobný.

3.3. Práca s videom

Pri upravovaní obsahu kapitoly je možnosť zadať časové ohraničenie, začiatok odkedy má byť video pustené a koniec, kedy má skončiť. Tieto časy sú následne uložené v databáze spolu s obsahom kapitoly.

Videá sú uložené na serveri, čím zaniká potreba ich z databázy sťahovať, pretože môžu zaberať veľa miesta a neustálym sťahovaním videa by bola plynulosť a výkonnosť aplikácie veľmi obmedzená. V databáze na miestach, kde je potrebná práca s videom je použitá cesta k videu namiesto celého súboru. Po skončení videa je namiesto video bloku zobrazený text zadania.

3.4. Presun na nasledujúcu kapitolu

Presun na nasledujúcu kapitolu (ak existuje) je realizovaný pomocou hypertextových odkazov. Odkaz je nastavený na základe atribútu NextChapter v triede ChapterModel a v tabuľke chapters. Je to poradové číslo kapitoly, ktorá spadá pod danú lekciu. Číslo lekcie je predávané ako hodnota v URL adrese a nextchapter je číslo o jedna väčšie ako akutálne číslo kapitoly alebo -1 v prípade, že nasledujúca kapitola už nie je.

V princípe to funguje ako spájaný zoznam, kde na nasledujúci prvkom (*next*) označuje číslo nasledujúcej kapitoly.

3.5. Pridávanie novej kapitoly

Pridávanie novej kapitoly funguje tak, že sa vytvorí nová kapitola a tá sa nastaví ako *next* aktuálne otvorenej. Problém môže vzniknúť, ak nasledujúca kapitola za touto aktuálnou existuje. Podobne ako pri používateľoch aj v tejto tabuľke musia byť dvojice (id lekcie, poradové číslo) unikátne. Nie je teda možné iba vkladať nové kapitoly a zistiť či už nasledujúca kapitola existuje a vetviť chod aplikácie podľa toho, by bolo menej prehľadné, je preto využitá funkcionalita PostreSQL systému – *UPSERT*. Táto funkcionalita v sebe kombinuje metódy *UPDATE*, ktoré aktualizujú položky v databáze a metódu *INSERT*, ktorá do databázy vkladá nové riadky.

Princíp je taký, že sa najprv systém pokúsi o vloženie riadkov. V prípade, že by malo dojsť k porušeniu integritného obmedzenia *UNIQUE*, ktoré zabezpečuje jedinečnosť, má programátor na výber ako postupovať. Môže nepokračovať a dopyt do databázy ukončiť alebo sa v prípade zlyhania pokúsiť o aktualizovanie existujúcich riadkov.

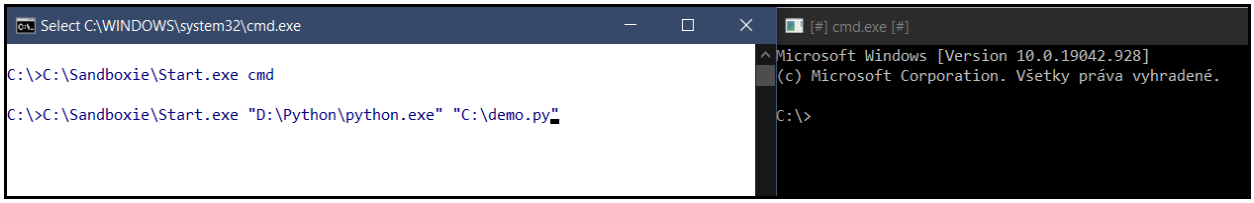
Upsert nie je jedna funkcia, ale je to funkcia *INSERT* ktorá obsahuje klauzulu *ON CONFLICT DO UPDATE*. V programovacích jazykoch je to podobné ako blok *try – catch*.

Formulár na pridávanie kapitoly a úpravu obsahu je rovnaký. Líšia sa iba v menších častiach ako hlavička, titulok a spôsom akým spracovaný formulár. Ale väčšia časť je rovnaká. Aby nenastal problém pri práci s duplicitným kódom a všetky zmeny bolo potrebné vykonať v jednom súbore na jednom mieste.

Táto veľká spoločná časť je uložená v takzvanom *partial view*. Je to HTML súbor, ktorý nie je kompletný podľa štandardov (napríklad nemusí mať definovaný typ *<!DOCTYPE>*, hlavičku *<head>*, ozačenie HTML dokumentu *<html>* a pod.). Jeho obsah je potom počas renderovania stránky vložený, takže celý obsah stránky sa vyskladá a skompletizuje. Slúži na to volanie funkcie frameworku *Html.RenderPartial*, ktorému je do argumentov poslaný názov tohto partial view. Používaním partial views sa zabraní duplicitnému kódu, sprehľadní sa

3.6. Ovládanie sandboxu

Výhodou sandboxu Sandboxie je to, že je možné ho spúšťať pomocou príkazového riadku, takže je jednoduché ho ovládať. Príkazy musia nasledovať nasledovnú syntax: *<Priečinok Sandboxie\Start.exe> <názov programu alebo cesta k nemu> <parametre programu>*



Spustenie sandboxu prebieha v novom vlákne procesora. TToto vlákno inicializuje proces sandboxu s potrebnými cestami a čakanie na výsledok prebieha taktiež v ňom, čím funkčnosť aplikácie nie je obmedzená, zatiaľ čo sa čaká na skončenie sandboxu a vytvorenie potrebných výstupných súborov.

Ovládanie sandboxu pomocou .NET frameworku je citlivé na úvodzovky, medzery, zadávanie parametrov, presmerovanie alebo prepínače a ich umiestnenie alebo vynechanie môže spôsobiť nesputenie programu v sandboxe. Preto je jednoduchšie, istejšie a prehladnejšie spúšťať programy v ňom pomocou batch skriptov, ktore ako argument dostanú cestu k súboru, ktorý má byť sputený a samotné spúšťanie je prenechané na skript. Tým sa syntax správne rozparsuje a problémy spomenuté vyššie rieši samotný skript.

```
int timeout = 30;
using (Process process = new Process())
{
    string args = $"\"{PathToTest}\"";
    ProcessStartInfo info = new ProcessStartInfo()
    {
        FileName = "C:\\abc.bat",
        UseShellExecute = true,
        Arguments = args
    };
    process.StartInfo = info;
    process.Start();
    process.WaitForExit(timeout);

    int elapsedSecs = 0;
    while (!File.Exists(resultPath) && elapsedSecs < timeout)
    {
        try
        {
            System.Threading.Thread.Sleep(1_000);
            elapsedSecs++;
        }
        catch (System.ArgumentOutOfRangeException e) { result = e.Message; }
        catch (System.Threading.ThreadInterruptedException e) { result = e.Message; }
        catch (System.Threading.ThreadAbortException e) { result = e.Message; }
        catch (System.Exception e) { result = e.Message; }
    }
```

Napríklad, použitie presmerovania výstupu ako argumenty v príkazovom riadku nefungujú:

```
C:\Sandboxie\Start.exe C:\demo.py > output.txt
```

Parsovanie spôsobí to, že v sandboxie je spustený program *demo.py* a výstup má byť presmerovaný do *output.txt* súboru. V skutočnosti bude tento súbor prázdny, pretože do neho bude presmerovaný výstup zo sandboxu, teda nič, nakoľko presmerovanie sa nedeje v sandboxe, nemá sandbox čo vrátiť.

Použitie úvodzoviek taktiež problém nerieši, pretože ak príkaz upravíme ako: *C:\Sandboxie\Start.exe „C:\demo.py > output.txt“*

v skutočnosti sandbox vyhlási chybu, že súbor „C:\demo.py > output.txt“ neexistuje. Iná kombinácia úvodzoviek bude spôsobovať podobné problémy. Preto program ktorý sa má spustiť a presmerovanie rieši batch skript, ktorý je v sandboxe pustený.

(OBRAZOK - DIAGRAM PREPOSIELANA ARGUMENTOV !!)

Samotný nový proces je spustený pomocou triedy *Process* z frameworku .NET. Ten simuluje spúšťanie programu z príkazového riadku.

FileName obsahuje názov, prípade cestu k hlavnému programu, ktorý má byť spustený, Arguments predstavuje argumenty, ktoré sú priložené ku programu, sú zapísané do príkazového riadku ako argumenty volania. Okrem ciest to môžu byť rôzne prepínače, modifikátory alebo iné argumenty. V tomto prípade je argumentom cesta k súboru, ktorý má byť spustený v pythone.

3.7. Životný cyklus spúšťania programu

Pre sprehľadnenie a zhrnutie je v tejto podkapitole uvedený celý proces spúšťania programu.

Najprv sa musí program nachádzať v systéme. To je možné buď odovzdaním súboru cez formulár alebo je možné program napísať priamo v aplikácií.

Pred spustením aplikácie sa skontroluje, či programovacie okno obsahuje nejaký text. Ak áno, je tento text zapísaný do súboru a uložený do databázy.

Z databázy sa stiahne posledné riešenie, ktoré používateľ spustil alebo nahral (teda ak napísal do programovacieho okna nejaký kód, tento kód sa stáva posledným riešením). Je stiahnutý do pracovného priečinku používateľa. V závislosti od toho, čo chce používateľ spustiť nastanú dva prípady: chce spustiť samotný program alebo chce spustiť test.

V prípade programu je vytvorená inštancia triedy *Process*, ktorá spustí sandbox a v ňom program.

V prípade testu je stiahnutý posledný nahratý test z databázy, ktorý patrí ku kapitole. Tento test je stiahnutý do pracovného priečinku používateľa, kde sa nachádza aj program, ktorý má byť otestovaný. Následne je pomocou *Process* triedy spustený sandbox, ktorému je predaná cesta k testu.

V sandboxe je spustený batch súbor, ktorý spustí program a presmeruje výstup, štandardný aj chybový, do súboru. A začne cyklus hľadania. Počas neho v pravidelných časových intervaloch aplikácia kontroluje, či výstupný súbor sandboxu existuje a či je možné ho čítať.

Na vytovrenie súboru a jeho sprístupnenie je pevne určený čas tridastich sekúnd. Do tohto limitu, ak súbor neexistuje alebo nie je prístupný na čítanie, je zavolaná funkcia, ktorá ukončí beh daného sandboxu a používateľovi poskytne výstup o tom, že ubehol časový limit a program neskončil. Tým sa predíde nekonečným cyklom alebo programom, ktoré by mohli zahltiť pamäť, ak by bežali príliš dlho.

V opačnom prípade, teda po úspešnom nájdení a sprístupnení výstupého súboru, je jeho obsah predaný používateľovi a teda aj všetko výstup, ktorý program vytvorí vyprodukuje. Tento súbor sa vždy nachádza v určitom priečinku, ktorý vytvorí Sandboxie prostredie, teda hľadanie súboru je vždy sústredené na jedno miesto.

Pretože chybový výstup je presmerovaný do rovnakého súboru ako štandardný výstup, aj v prípade zlyhania programu pred spustením (kvôli syntaktickej chybe alebo nesprávnom odsadení) je k dispozícií táto informácia a je posnutá používateľovi, ako klasický výstup.

Pred spustením programu je vymazaný obsah existujúceho sandboxu. Tým je program spsutený vždy v novom a prázdnom sandboxe.

3.8. Základný menežment

Používatelia, ktorí zastupujú rolu administrátora alebo učiteľa, potrebujú aspoň základný menežment na správu obsahu a používaľov.

(ZMENIŤ ROLE : ADMIN, UČITEĽ, ŠTUDENT, POPISY UPDATOVAŤ)

3.8.1. Registrácia

Každý má možnosť zaregistrovať sa a si svoje konto. Každé konto musí mať jedinečný email a používateľské meno, čím je zaručená jedinečnosť používateľov a možnosť je jednoduchého a jednoznačného vyhladávania V opačnomprípade sa registácia nepodarí a aplikácia vráti chybu o tom, že v databáze sa aspoň jeden z údajov nachádza.

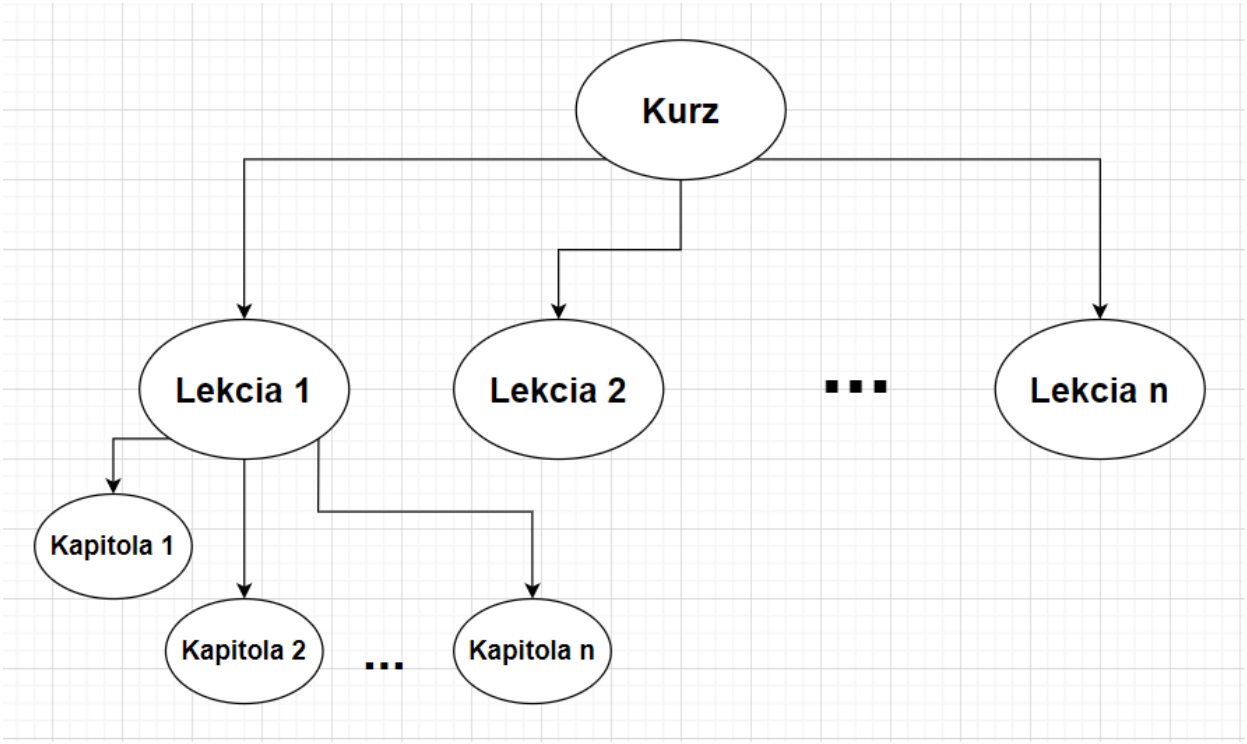
Okrem emailu a používateľského mena je potrebné vyplniť heslo. Heslo sa v aplikácií nikde nepoužíva v čistej textovej forme – vo forme plain text, ale je už od začiatku zašifrované a uložené do databázy. V databáze existuje šifrovacia funkcia *MD5,* ktorou sa pred vložením heslo zašifruje a až následne je v databáze uložené.

3.8.2. Dostpuné kurzy

Vďaka tomu, ako sú kurzy navrhnuté je možné jednoduchým krátkym dopytom do databázy poskytnúť používateľovi prehľad o tom, ku ktorým kurzom má prístup a jednoduchá sa dá overiť, či k danému kurzu má používateľ prístup. Nakoľko aplikácia využíva predávanie niektorých hodnôt pomocou URL adresy (tzv. *query string*), používateľ sa môže pokúsiť, či už nedopatrením alebo zámerne, získať prístup aj ku kurzom, ku ktorým prístup mať nemá.

Na toto nám slúži jednoduchá väzobná tabuľka *users_courses*, ktorá uchováva informácie o tom, ktorý používateľ má ku ktorým kurzom prístup. Informácie sú uložené ako dvojice (id kurzu, id používateľa).

Hierarchia kurzu potom vo výsledku určuje aj to, ku ktorým lekciám a ku ktorým kapitolám má používateľ prístup : kurzu pozostáva z lekcií (lekcia zastrešuje jednu tému). Následne jedna lekcia pozostáva z kapitol (menšie celky jednej témy). A v týchto kapitolách sa nachádza samotný obsah.



3.8.3. Sprístupňovanie kurzov používateľom

Použitie väzobnej tabuľky z 3.5.3 zohráva rolu aj pri sprístupňovaní kurzu niektorému používateľovi. To môže spraviť iba ten, ktorý má administrátorské práva.

Pomocou používateľského mena, je v tabuľke používateľov nájdený ten, ktorý má byť pridaný do kurzu. Jeho id sa spoločne s id kurzu vložia do tejto väzobnej tabuľky a tým je kurz sprístupnený pre používateľa. Na zapísanie je využité jednoduché vkladanie s klauzulou *ON CONFLICT*. V prípade že sa takáto dvojica už v tabuľke nachdáza, databázový systém nespôsobí výnimku. Ak používatľské meno nie je správne a teda sa v tabuľke nenachádza používateľovi je predaná správa o chybe.

(ZOBRAZIŤ VŠETKY KURZY – NA POŽIADANIE PRIDÁVAŤ ŠTUDENTA)

3.9. Zobrazovanie lekcií pre konkrétneho používateľa

Každý používateľ má po príhlásení k dispozícií zoznam kurzov. Po kliknutí na kurz je používateľovi zobrazený zoznam hodín, ktoré sú v danom kurze.

Každá lekcia je zviazaná s niektorým kurzom a tak má odkaz na tento kurz ako atribú tabuľky. Pri načítavaní lekcií je využitý jednoduchý filter – číslo kurzu, bez potreby používania *JOIN* spánia tabuliek alebo iných zložitejších konštrukcií.

(OBRAZOK ?!?! - PREROBIŤ)

3.10. Obmedzenia a potrebný software

Aplikácia je primárne vyvíjaná a testovaná na *localhost*. Vývoj je ešte obmedzený, pretože spustenie aplikácie cez vývojové štúdio vytvorí dočasný veľmi jednoduchý a obmedzený server určený na jedno spustenie. Všetky cesty a oprávnenia teda môžu fungovať pri spúšťaní cez štúdio, ale pri nasadení na reálny localhost alebo server sa aplikácia pravdepodobne bude správať inak.

3.10.1.Python interpreter

Aby bolo možné spúšťať pythonovské programy a testy je potrebné mať nainštalovaný pythonový interpreter. Je to v podstate program, ktorý zabehu prekladá pythonový program riadok za riadkom do inštrukcií, ktoré procesor vykonáva.

Interpreter pythonu je možné ovládať aj pomocou príkazového riadka. Túto vlastnosť využívam pri spúšťaní testov a programov používateľov.

Na správny chod aplikácie je potrebné, aby bol pythonový interpreter uložený v premennej prostredí *%PATH%*. V prípade, že nie je interpreter v premennej, je potrebné vytvoriť batch súbor, ktorý obsahuje cestu k nemu. Program, ktorý má interpreter spustiť, je cez príkazový riadok poslaný ako argument v podobe absolútnej cesty.

Príkaz je citlivý na syntax, zvlášť ak niektorá z ciest obsahuje medzery. V tedy je potrebné správne použiť úvodzovky. O ich správne usporiadanie, umiestnenie a použitie sa postará samotný batch skript.

3.10.2.IIS

Internet Information Service je serverová služba – program, ktorý vytvorí jednoduchý server, na ktorom bude na pozadí bežať aplikácia. Pre spúšťanie webových aplikácií frameworku .NET je nevyhutný.

Jeho funkcionalita je podobná ako má pre jazyk PHP program Apache. Bez toho by funkcionalita servera zahŕňala iba komunikáciu cez statické HTML súbory. Teda žiadna komunikácia s databázou, žiaden dynamicky generovaný obsah stránky a podobne.

IIS služba po nastavení a inicializovaní servera a jeho nastavení na pozadí spustí aplikáciu a všetky.

Problémom po spustení serverovej služby je, že všetky cesty, s ktorými aplikácia pracuje je dôležité skontrolovať a prípadne prestaviť, nakoľko istá časť aplikácie je premiestnená do priečinku, ktorý serverová služba vytvorí a potrebuje v ňom mať niektoré súbory aplikácie – html súbory, súbory objektov, skripty, css súbory, kofiguračné súbory, DLL knižnice aplikácie a podobne.

3.10.3.Databázový driver

Aby databázový systém mohol riadne fungovať, je potrebné, aby vedel pracovať na najnižšej úrovni počítača – s diskom, procesorom a pamäťou [20]. Aplikácia využíva PostgreSQL a ODBC. Preto driver musí byť taký, aby vedel spracovať príkazy zo systému PostgreSQL. Pretože ale opužívateľ nebude ovládať databázu priamym písaním dopyt, musí driver dostávať dopyty inak – cez knižnicu ODBC, ktorá umožňuje komunikáciu medzi aplikáciou a driverom.

Na tento účel je použitý driver PSQLODBC, vyvinutý a udržovaný spoločnosťou, ktorá vyvíja PosgtreSQL.

3.11. Ukážky

(Obrázky : hlavné menu, obsah kapitoly, úprava obsahu kapitoly, spúšťanie programov, spúšťanie testov, výsledky, obsah pracovného priečinku)

Záver

Testovanie

Použitá literatúra, zdroje

[1] Programiz, *Programiz: Learn to code for free*, dostupné online, dňa 10.5.2021, URL: https://www.programiz.com/

[2] Python Tutor, *Python Tutor - Visualize Python, Java, JavaScript, C, C++, Ruby code execution*, dostupné online, dňa: 15.2.2021, URL: http://www.pythontutor.com/visualize.html#mode=edit

[3] W3Schools, *W3Schools Online Web Tutorials,* dostupné online, dňa 5.2.2021, URL: w3schools.com

[4] Ing. Fratišk Gyarfáš, Csc., *Osobná stránka Ing. Františka Gyárfáša, PhD.*, dostupné online, dňa: 5.2.2021, URL: https://dai.fmph.uniba.sk/~gyarfas/

[5] RNDr. Andrej Blaho, PhD., *Materiály k predmetom Programovanie (1) 1-AIN-130/16 a Programovanie (2) 1-AIN-170/13 na FMFI UK*, dostupné online, dňa 5.2.2021, URL: http://python.input.sk/

[6] RNDr. Andrej Blaho, PhD., *Materiály k predmet Algoritmy a dátové štruktúry 1-AIN-210/15 na FMFI UK.*, dostupné online, dňa 5.2.2021, URL: http://struct.input.sk/

[7] Ing. Alexander Šimko PhD., *Teaching - Alexander Šimko*, dostupné online, dňa: 5.2.2021, URL: https://db.dai.fmph.uniba.sk/teaching/

[8] *L.I.S.T.*, dostupné online, dňa: 5.2.2021, URL: https://list.fmph.uniba.sk/

[9] JURSA, Andrej., *NOVÝ DLHODOBÝ VIACÚČELOVÝ SKLAD ÚLOH NA CIČENIA*, bakalárska práca, 2013, FMFI UK Bratislava

[10] Moodle, *E-learning Univerzity Komenského*, dostupné online, dňa: 5.2.2021, URL: https://moodle.uniba.sk/

[11] Udemy, *Online Courses - Learn Anything, On Your Schedule*, dostupné online, dňa: 5.2.2021, URL: udemy.com

[12] Free Code Camp, *Learn to Code — Coding Courses for Busy People*, dostupné online, dňa 10.5.2021, URL: https://www.freecodecamp.org/

[13] LETAVAY, Marek., *VIRTUÁLNE PROSTREDIE PRE TVORBU INTERAKTÍVNYCH MULTIMEDIÁLNYCH KURZOV PROGRAMOVANIA*, bakalárska práca, 2013, FMFI UK Bratislava

[14] Microsoft, *Developer tools, technical documentation and coding examples Microsoft Docs*, dostupné online, dňa: 5.2.2021, URL: https://docs.microsoft.com/en-us/

[15] PostgreSQL, *PostgreSQL: The world's most advanced open source database*, dostupné online, dňa: 5.2.2021, URL: https://www.postgresql.org/

[16] Avast, *What is sandboxing and how does it work?*, dostupné online, dňa: 15.2.2021, URL: https://www.avast.com/business/resources/what-is-sandboxing

[17] MCKELLAR, Jessica, online konferencia *PyCon 2014*, dňa: 15.2.2021, URL: https://www.youtube.com/watch?v=sL_syMmRkoU

[18] Sandboxie, *Sandboxie - Sandbox software for application isolation and secure Web browsing*, dostupné online, dňa: 5.2.2021, URL: https://www.sandboxie.com/

[19] Microsoft, *What is ODBC?*, dostupné online, dňa: 5.2.2021, URL: https://docs.microsoft.com/en-us/sql/odbc/reference/what-is-odbc?view=sql-server-ver15

[20] PostgreSQL, *psqlodbc - PostgreSQL ODBC driver*, dostupné online, dňa: 5.2.2021, URL: https://odbc.postgresql.org

[21] W3schools, *What is Client-Server Architecture?*, dostupné online, dňa: 10.5.2021, URL: https://www.w3schools.in/what-is-client-server-architecture/