UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Systém na prípravu učebných materiálov a testových zadaní

Bakalárska práca

2018 **Tomáš Bočinec**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Systém na prípravu učebných materiálov a testových zadaní

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. František Gyarfaš, CSc.

Bratislava, 2018 **Tomáš Bočinec**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Tomáš Bočinec
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: Webová výuka programovania pomocou metodológie TDD
Web system for learning programming using TDD method

Anotácia: Cieľom bakalárskej práce je vytvoriť interaktívne webové prostredie pre výuku programovania v rôznych programovacích jazykoch formou testmi riadeného programovania (TDD). Aplikácia umožní vytvárať úlohy, ktoré budú študenti riešiť vo webovom prehliadači pomocou cyklu: nesplnený test, program, refaktorizácia. Študenti budú postupne pridávať v cykle jednotkové testy a programový kód, pokiaľ nenaprogramujú celé zadanie. Kód testov aj programov sa bude ukladať na serveri a spúšťať vo virtuálnom prostredí. Aplikácia bude vedieť spracovať programy vo viacerých programovacích jazykoch. Bude obsahovať webové vývojové prostredie pre písanie programového kódu vo vybranom jazyku a na serveri zbiehanie a hodnotenie odovzdaných riešení. Bude obsahovať nástroje na administráciu študentov. Technológie/nástroje: PHP (framework), MySQL, HTML 5, CSS, JavaScript (jQuery, Bootstrap), kompilátory vybraných jazykov, knižnice pre unit testing a virtuálny server.

Vedúci: Ing. František Gyarfaš, CSc.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 09.10.2017

Dátum schválenia: 16.10.2017 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie

Čestne vyhlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením vedúceho bakalárskej práce, s použitím uvedenej literatúry a zdrojov dostupných na internete.

V Bratislave dňa _____

Meno Priezvisko

Poďakovanie

Abstrakt

PRIEZVISKO, Meno: *Názov práce(Diplomová práca)* – Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. – Školiteľmeno školiteľa.: FMFI UK, RRRR, XXstrán

.....

Kľúčové slová:.....

Abstract

PRIEZVISKO, Meno: *Anglický názov (Diploma thesis)* – Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics – Supervisor meno .: FMFI UK, 2016, XX pages

…….

Keywords:……

Predhovor

.....

Obsah

Úvod

1. Teoretické východiská

V prvej kapitole sa zameriavam na analyzovanie existujúcich riešení podobných tejto práci a popísanie nevyhnutnej teórie potrebnej k pochopeniu danej problematiky.

1.1. Elearning – edukačný softvér

Elearning predstavuje najmodernejší spôsob výučby s využitím informačných technológií. Je to implementácia informačných technológií do vývoja, distribúcie a riadenia vzdelávania alebo výučby. Elearning je efektívnym prostriedkom, ako sa spojiť so študentmi alebo zamestnancami a odovzdať im cenné informácie na diaľku. Toto odovzdávanie informácií sa najčastejšie realizuje prostredníctvom on-line dištančných kurzov. Dištančné vzdelávanie ako také, teda samoštúdium za pomoci tútora, ktorý je fyzicky oddelený od študujúcich a vzdelávanie iba koordinuje, existuje už veľa rokov. Rozvoj informačných technológií zasiahol aj do vzdelávania a výrazne vylepšil vyučovací proces. [10]

Edukačný softvér je softvér vytvorený špeciálne pre vyučovanie ako nástroj pre učiteľov na učenie alebo pre študentov na učenie sa [11]

1.2. Virtualizácia

Virtualizácia je proces, pomocou ktorého prevádzame fyzicky prostriedok na softwarovú vrstvu. Napríklad môžeme prevádzkovať niekoľko operačných systémov súčasne na jednom serveri.

Virtualizačný software tzv. hypervisor vytvára štandardizovaný hardvér. Poskytuje ho pre všetky virtuálne stroje, ktoré nad ním bežia. Tento hardvér zahŕňa všetko potrebné k behu počítača, teda CPU, operačnú pamäť, vstupno-výstupné zariadenia, komunikačné prostriedky a podobne. [7]

Takýto abstrahovaný hardvér je dostupný operačnému systému, ktorý nepotrebuje byť špeciálne upravený na beh vo virtuálnom prostredí. Virtualizácia je teda transparentná, hardvér poskytnutý hypervisorom je natoľko štandardný, že každý podporovaný operačný systém s ním vie bez problémov pracovať.[7]

1.2.1. Virtuálny sandbox

Medzi výhody virtualizácie patrí rýchla prenesiteľnosť dát, rýchle spustenie a nasadenie systému, oddelenie fyzického hardvéru, čo znej robí dobrý základ na sandbox. Programátorský sandbox je priestor, kde môžeme bezpečne spúšťať softvér bez toho, aby mal dosah na nežiaduce miesta. Môžeme ho rôzne obmedzovať, napríklad nastavením

dostupného výpočtového výkonu alebo prístupu na internet. Výhodou je, že virtuálny sandbox vieme v prípade poruchy rýchlo obnoviť.

2. Extrémne programovanie

Extrémne programovanie patrí medzi jednu z metodík agilného vývoja softvéru. Jej hlavným cieľom je včasne reagovať na požiadavky zákazníka, zlepšiť kvalitu produktu a zlepšiť výkon programátorov. Názov vznikol podľa toho, že v extrémnom programovaní sa využívajú v maximálnej miere všetky metódy, ktoré sa uplatnili pri vývoji. Napríklad keď sa osvedčilo písanie testov tak v extrémnom programovaní sa testuje každá metóda, keď sa osvedčila jednoduchosť tak v extrémnom programovaní sa píše kód najjednoduchšie ako sa dá. Medzi základné programátorské praktiky patrí testovanie, refaktorizácia kódu, jednoduchý návrh, priebežná integrácia. Medzi tímové praktiky patrí párové programovanie, spoločné vlastníctvo kódu a štandardizácia kódu. Medzi obchodné praktiky sa radí úzka spolupráca zo zákazníkom, napríklad spôsobom zákazník na pracovisku.

2.1. Testovanie softvéru

Testovanie je proces, ktoré poskytuje zainteresovaným stranám informácie o kvalite testovaného softvérového produktu alebo služby[2]. Cieľom je nájsť možné softvérové chyby a tým overiť, že je softvér vhodný na používanie prípadne vývoj ďalších funkcionalít. Samotné testy by mali overovať, či softvér spĺňa samotné požiadavky zo zadania a dosahuje výsledky, ktoré zainteresované strany očakávajú, či správne reaguje na všetky možné vstupy a vykonáva svoje funkcie v prijateľnom čase.

2.1.1. Unit testing

Unit testing je vývojový postup, v ktorom programátori vytvárajú testy súbežne s vývojom softvéru. Testy majú byť krátke kusy kódu, s cieľom otestovať funkcionalitu čo najmenšej programovej jednotky vlastného kódu aplikácie, ako je funkcia či metóda triedy. Testy používajú objekty aplikácie, avšak existujú vnútri unit testing frameworku. Takýto prístup má niekoľko výhod. Produkčný kód nie je poprepletaný testami. Veľkosť skompilovanej aplikácie nenarastá. Testy sa môžu spúšťať oddelene od aplikácie, a teda objekty sa môžu testovať izolovane [3]. Vo všeobecnosti môžeme testovanie aplikácií z hľadiska vnútorného prístupu rozdeliť na dve hlavné skupiny. Je to funkcionálne (black-box) a štrukturálne (white-box) testovanie. Pri prístupe black-box testovania je tvorcovi testov

neznáma vnútorná štruktúra testovaného kódu aplikácie. Znamená to, že nevie čo sa po spustení deje v jeho vnútri, len zadá testovací vstup a prekontroluje výstup. Pri unit testingu sa používa hlavne druhý white-box prístup, pretože testerovi je známa vnútorná štruktúra kódu a reprezentácia dátových štruktúr. Unit testy sú schopné testovať len public objekty. Táto skutočnosť núti programátora dobre prehodnotiť dizajn tried. Musí sa rozhodnúť, ktoré objekty ešte majú byť skryté pred vonkajším svetom a naopak, ktoré musia byť viditeľné. Tvorba softvéru pomocou unit testov tak nepriamo zabezpečuje dobrý programátorský štýl objektovo orientovaného programovania.

2.1.2. Testovacie frameworky

Testovací framework alebo unit testing framework je softvérový nástroj na zjednodušenie tvorby a obsluhy testov. Testovacie frameworky sa pre komplexnosť svojej podpory pri vytváraní, spúšťaní a vyhodnocovaní testov stali neoddeliteľnou súčasťou spektra nástrojov nevyhnutných pri tvorbe a testovaní softvéru. Pre vývojárov riadiacich sa metodológiou Test-Driven Development hrajú kľúčovú rolu vo všetkých fázach vývoja softvéru. Zasahujú do architektúry softvéru, upravujú jeho dizajn a využívajú sa aj pri optimalizácii jeho výkonu. V dnešnej dobe existuje veľké množstvo testovacích frameworkov takmer na každý jazyk, napríklad jazyk C má viac ako 50 testovacích frameworkov.

2.1.3. Použitie frameworkov

V nasledujúcej kapitole sú opísané testovacie frameworky pre jazyk C ++ a jazyk python, ktoré sú využívané aj v samotnej aplikácií.

2.1.3.1. PyUnit

PyUnit je oficiálny unit test framework pre programovací jazyk python. Nachádza sa v základnom balíku knižníc pythonu. Framework je veľmi intuitívny, obsahuje základné funkcie na kontrolu ako assertEqual, assertTrue, assertIn, assertIsInstance. Výhodou tohto frameworku je, že jednotlivé testy sa dajú púšťať priamo z konzoly.

2.1.3.2. Google Test

Google Test je knižnica na unit testovanie pre programovací jazyk C++ založená na architektúre XUnit[5]. Knižnica spadá pod licenciu BSD 3-clause a bola oficiálne vydaná v auguste 2016. Samotné testy môžu byť spúšťané jednotlivo alebo všetky naraz. Výhodou

je, že po každom zbehnutí testov vám framework ponúkne aj XML report. Spoločnosť Google zaviedla vlastnú klasifikáciu testov.

Testy sa radia do troch kategórií:

a) Malé testy (Unit testy) - Väčšinou malé automatické testy, ktoré vykonávajú test nad jednou funkciou alebo metódou. Zameriavajú sa na typické problémy, ako napríklad testy rôznych vstupov alebo chybné podmienky. Tieto testy by mali trvať krátko, spravidla do pár sekúnd.
b) Stredné testy (Integračné testy) - Stredné testy testujú jednu alebo viacero ucelených funkcionalít aplikácie. Dôraz sa kladie na testovanie medzi rôznymi funkciami a metódami, ktoré medzi sebou komunikujú.
c) Veľke testy (Akceptačne testy) - Akceptačne testy pokrývajú viacej funkcionalít a vytvárajú reálne používateľské scenáre. Často kontrolujú, či softvér spĺňa požiadavky zákazníka.

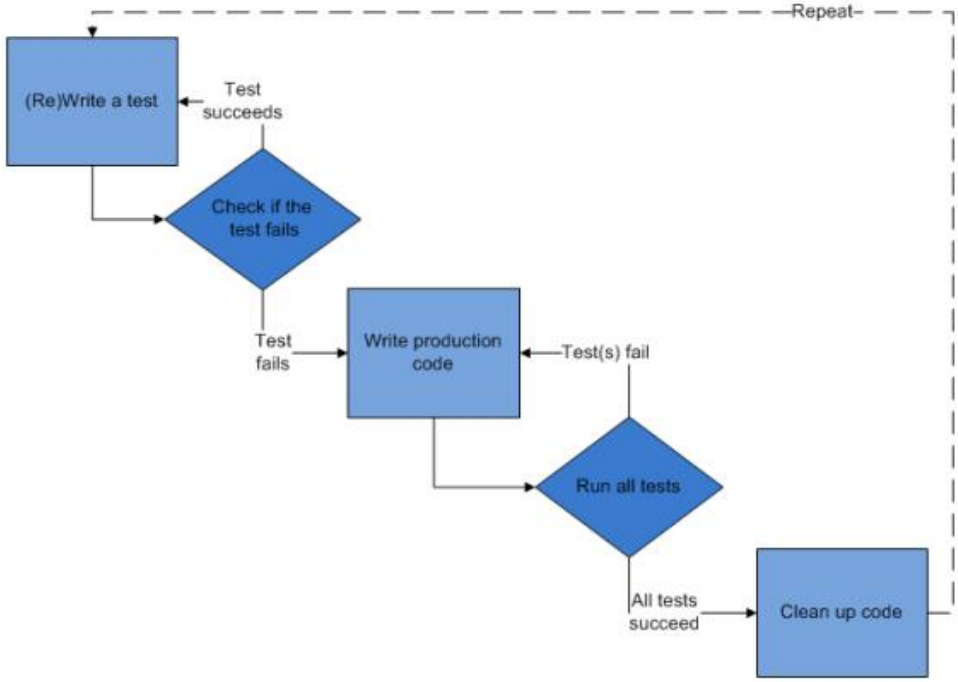
2.2.Refaktorizácia kódu

Refaktorizácia je proces reštrukturalizácie naprogramovaného kódu, ktorý nemení jeho funkčnosť ale vnútornú štruktúru s ohľadom na minimálne vnášanie nových chýb. Refaktoringom zlepšujeme kvalitu kódu, znižujeme počet chýb, zvyšujeme čitateľnosť a jednoduchosť kódu, a tým zvyšujeme aj rýchlosť vývoja. Najlepšie sa refaktoruje kód, ktorý je dobre testovaný. Odporúča sa zmeny robiť po malých častiach a po každej zmene pustiť testy, aby sme vedeli, že sa zachovala funkčnosť. Medzi najčastejšie dôvody refaktorizácie patria: Duplicitný kód, dlhé metódy, veľké triedy, zložité štruktúry podmienok. Refaktorovať sa odporúča po každom dopísaní nového kódu.

2.3.Test Driven Development

Test-Driven Developmnet (vývoj riadený testami), často nazývaný aj test-first programming, patrí medzi tzv. agilné metodológie vývoja softvéru, ktoré sa v posledných rokoch v kruhoch softvérového inžinierstva stávajú čoraz populárnejšie. Pre každý z týchto postupov vývoja softvéru platí, že sa snaží skrátiť dobu vývoja, poskytnúť čo najkvalitnejší produkt z hľadiska špecifikovanej funkčnosti a chybovosti riešenia. Jednou z priorít je tiež zmenšiť riziko „stroskotania“ celého projektu na zmene alebo vzniku nových požiadaviek, zavádzaním pružného postoja k požiadavkám pomocou iteratívneho prístupu vývoja

projektu. Hoci vývojári používali Test-Driven Developmnet niekoľko desaťročí v rôznych formách, táto stratégia vývoja získala väčšiu pozornosť až ako jeden zo základných postupov extrémneho programovania [1]. Stratégia Test-Driven Developmnet vyžaduje písanie automatizovaných testov, ešte pred vytvorením vlastného funkčného kódu v krátkych a rýchlych iteráciách [1]. Testy sa následne vyhodnocujú prostredníctvom simulačných programov alebo testovacích frameworkov. Výsledky testovania odhalia, ktoré testy neuspeli a tým poukážu na chyby v implementácii. Nesprávne kusy programového kódu je potrebné opraviť a znovu spustiť všetky testy. Programový kód je nutné upravovať do tej doby, pokiaľ všetky 3 testy nebudú úspešné. Nikdy sa nesmie vynechať žiadny test, aj keď mnohokrát táto možnosť vývojára láka ako najjednoduchšie riešenie. Napísať test pred vytvorením vlastného programového kódu, vždy spustiť každý doposiaľ napísaný test a iterovať kód, kým sa objavuje čo i len jeden test skončený neúspechom, sú základnými dogmami vývoja riadeného testami.



Obrázok č. 1 : Test-Driven Development cyklus

Proces vývoja softvéru pomocou tejto metodológie sa dá opísať pomocou nasledovných krokov, ktoré sú znázornené aj na obrázku č.1.

1. Nový test pre ešte neimplementovaný kód

V prvom kroku je za úlohu naprogramovať test, ktorý pokrýva ešte nenaprogramovanú časť alebo funkcionalitu, čiže pri prvom spustení by nemal prejsť. Toto donúti programátora premýšľať nad tým, akú rôznu funkcionalitu musí naprogramovať.

2. Spustiť testy

   V druhom kroku spustíme všetky testy a nami nový test by nemal prejsť, čo je skúška toho, či nie je nový test zbytočný. Taktiež overíme si, či testovací framework pracuje správne.

3. Napísať a upraviť kód

   Musíme napísať alebo upraviť kód tak, aby vyhovel novému testu. Implementácia by mala byť čo najjednoduchšia. V tomto prípade sa sústredíme iba nato, aby nový test prešiel.

4. Spustiť všetky testy

   Spustíme všetky testy a overíme, že nový kód prejde úspešne cez všetky testy. Ak by niektorý test neprešiel, musíme vyhľadať chybu a odstrániť ju.

5. Refaktorovať

   Keďže vieme, že nový kód je funkčný , musíme ho upraviť po štrukturálnej stránke a tým zlepšiť jeho kvalitu a čitateľnosť. Na konci refaktorizácie musíme overiť, či prešli všetky testy. Čiže skontroluje, že sme nezmenili funkcionalitu.

Test-Driven Development je metodológia, ktorá vývojárom dáva pokyn písať nový kód len vtedy, ak zlyhal niektorý z testov alebo pri pridávaní funkcionality [1]. Zmeny a úpravy programového kódu sú akceptovateľné len pri refaktorizácii alebo ladení chýb. V procese ladenia je dôležité zamerať sa na predvídateľné chyby a vytvárať testy, ktoré s veľkou pravdepodobnosťou na danej chybe zlyhajú. Akonáhle je takýto test implementovaný a skončí s neúspechom, je nevyhnutné príslušný testovaný kód opraviť a znova spustiť všetky testy pre overenie odstránenia zistenej chyby. Ak sa táto chyba opäť vyskytne v inej časti implementácie, tento test ju určite znova odhalí. Neustále opakovanie Test-Driven Development cyklu tvorí základ procesu vývoja softvéru prostredníctvom tejto metodológie. Jeho cieľom je vyvinúť „čistý kód, ktorý funguje“ [2]. Najskôr je zabezpečená funkčnosť, pomocou testovania, a následnou refaktorizáciou aj čistota vyvíjaného kódu. Táto metodológia poskytuje viac než len overenie správnosti implementovaného kódu. Tvorba testov núti programátora vopred premýšľať o funkcionalite a ideálnom dizajne

nového programového kódu. Test-Driven Development teda nepriamo robí celý vývoj časťou metodického, nízkoúrovňového procesu dizajnu softvéru. Akonáhle sú nové testy nasadené, slúžia ako rozhodujúci pracovný príklad toho, ako má byť nový kód použitý. Z týchto dôvodov je čas strávený písaním testov nielen testovacím úsilím. Investície do testovania sú rovnako investíciami do dizajnu [6].

2.4. Existujúce riešenia

V súčasnej dobe si už máloktorý študent siahne pri výučbe programovania po klasickej knihe. Je to z dôvodu, že je dostupné veľké množstvo nástrojov na výučbu programovania. Väčšina z nich je dostupná online. Tieto riešenia často umožňujú výučbu rôznych programovacích techník a jazykov. Vo väčšine prípadov sa v nich ako dorozumievací jazyk využíva anglický jazyk ,ale existujú aj systémy s rôznymi jazykovými mutáciami. K niektorým aplikáciám je možné pristúpiť zdarma a iné sú spoplatnené, napríklad mesačným členstvom alebo platbou za konkrétny kurz. Tieto aplikácie najčastejšie využívajú na výučby video tutoriály a testovacie kvízy. V niektorých aplikáciách sa nachádzajú kurzy na výučby TDD, ale žiadne riešenie neposkytuje čisto výučbu pomocou tejto techniky. V nasledujúcich podkapitolách sú popísané najpoužívanejšie riešenia.

2.4.1. Treehouse

Treehouse je internetová online škola, ktorá ponúka kurzy pre začiatočníkov aj pokročilých. Spoločnosť Treehouse bola založená 22. marca 2011 a momentálne ju využíva viac ako 180,000 aktívnych užívateľov. Vzdelávanie je založené na online video tutoriálov, ktoré sú doplnené kvízovými otázkami a programátorskými výzvami. Aktuálne je možné sa dostať k viac ako 220-tim kurzom, ktoré spolu obsahujú viac ako 27 500 minút video tutoriálov. Kurzy pokrývajú oblasti ako web dizajn, vývoj webových aplikácií, vývoj mobilných aplikácií a vývoj hier. Na základe absolvovania jednotlivých kurzov sú študentom udelené certifikáty, ktoré uznávajú aj mnohé veľké firmy pri prijímacích pohovoroch. Aplikáciu je možné si vyskúšať na sedem dní zadarmo a následne si spoločnosť účtuje 25 dolárov za mesiac.

2.4.2. Codeacademy

Codeacademy je internetová online škola, ktorá ponúka kurzy v dvanástich najpoužívanejších programovacích jazykoch. Je založená na obchodnom modeli Freemium.

Tým pádom je časť základnej funkcionality používateľom k dispozícií zdarma, avšak využitie kompletných služieb je spoplatnené. Medzi spoplatnené služby patrí napríklad online profesor, prístup k zdrojovým kódom z reálnych projektov alebo pokročilé kvízy. Stránku online školy, od jej založenia, využilo viac ako 25 miliónov používateľov, ktorí absolvovali viac ako 100 miliónov cvičení.

2.4.3. Code Avangers

Code Avangers je vyvíjaná vývojmi z Nového Zélandu od roku 2015. Služba zameraná na výučbu programovacích jazykov Python, Javascript, HTML a CSS. Výhodou je integrovaný textový editor, pomocou ktorého počas celej výučby študent píše kód. Aplikácia sa snaží čo najviac kontrolovať riešenia formou rôznych testov. V prípade, že sa žiak zasekne a nevie sa ďalej pohnúť, aplikácia mu napovie ako má pokračovať. Za sumu 120 dolárov na polroka dostanete prístup k viac ako 500 kurzom a lekciám a k viac ako 100 rôznym projektom. Službu je možné si vyskúšať na týždeň zadarmo.

2.4.4. Learn 2 Code

Learn 2 Code je slovenská spoločnosť, ktorá okrem online kurzov poskytuje aj klasické prezenčné služby. Pôsobí na trhu od roku 2012. Väčšinu kurzov poskytuje za poplatok od 30 € do 250 € alebo formou ročnej platby k prístupu na všetky online kurzy za 250€. Kurzy absolvovalo viac ako 15 000 študentov. Online kurzy sú dostupné pre technológie a programovacie jazyky ako napríklad Java, Python, Swift, PHP. Kurzy sú podávané formou video prednášok a programátorských úloh, ktorých riešenia je možné konzultovať na fóre.

2.5.Podobné bakalárske práce

2.5.1. Webová výuka programovania v C++ pomocou jednotkového testovania Viliam Vakerman

Cieľom tejto bakalárskej práce bolo vytvoriť internetovú aplikáciu na výučbu programovacieho jazyka C++ formou TDD. Študent má možnosť písať testy spolu s programátorským kódom priamo vo webovom prostredí, ktoré ho donucuje používať TDD.

3. Návrh aplikácie

3.1. Prehľad využívaných technológií a postupov

Nasledujúce podkapitoly sa venujú analyzovaniu využitia technológie. Výber technológií bol smerovaný tak, aby aplikácia spĺňala všetky kritéria kvality a zároveň zostala čo najjednoduchšia.

3.1.1. FrontEnd technológie

Užívateľské prostredie bude napísané v najnovšom štandarde značkovacieho jazyka HTML 5, a štylovacieho jazyka CSS 3, ktorý bude predkompilovaný pomocou technológie Sass.

Sass je kompilovaný jazyk, ktorý rozširuje syntax CSS o premenné, cykly, podmienky a funkcie.

Prostredie integruje webovú tému SwiftUniverzity. Webová stránka je založená na webovej sade nástrojov bootstrap, javascriptovskej knižnici Jquery a techniky AJAX.

3.1.2. BackEnd technológie

Správa webovej stránky bude spustená na webovom serveri apache, na ktorom bude nainštalovaná najnovšia verzia PHP 7.2.2. Budem využívať framework laravel 5.5.

Laravel je momentálne najpoužívanejší opensource PHP framework, ktorý využíva softwarovu architektúru MVC.

Využijem aj python http Server, ktorý bude spúšťať užívateľský kód vo virtuálnom sandboxe.

3.1.3. Nástroje na testovanie

Na testovanie python kódu som použil Python unit framework PyUnit. Na testovanie C++ kódu som zvolil Google Test FrameWork.

3.1.4. Databázové technológie

Laravel zabaľuje prácu z databáz a umožňuje využívať ľubovoľnú relačnú databázu, ktorú je možné pri nasadení zmeniť. Pri vývoji využijem relačnú databázu MYSQL.

3.1.5. Nástroje na virtualizáciu

.....

3.1.6. Ostatné nástroje

Aplikáciu som verzioval pomocou nástroja GIT a hosťoval som ju na stránke GITHUB.

Ako balíkový systém na PHP knižnice budem využívať composser s oficiálnym repozitárom.

(dopísať kompilátori k neskôr zvoleným prog. jazykom)

3.1.7. Vývojové prostredie

Počas programovania som využil vývojové prostredia od spoločnosti InteliJ. Na písanie PHP som využil aplikáciu PHP STORM a na vývoj časti v pythone aplikáciu PYCHARM. Na správu databázy DATAGRID. Výhodou je, že prostredia sú takmer identické a veľmi intuitívne. Na prostrediach ocenujem najme možnosť prepnúť textový editor do režimu VIM, kedy sa samotne pohybovanie v kóde vykonáva totožne ako v editore VIM.

4. Implementáciaaplikácie

5. Výsledky

Záver

Literatúra a internetové zdroje

[1] JANZEN, D. - SAIEDIAN, H. 2005. Test-Driven Development: Concepts, Taxonomy, and Future Direction. In Computer. 2005, roč. 38, č. 9, s. 43- 50.

[2] Kaner, Cem (November 17, 2006). *Exploratory Testing* (PDF). *Quality Assurance Institute Worldwide Annual Software Testing Conference*. Orlando, FL. Retrieved November 22, 2014.

[3] KRAJČ, D. 2008. Testovanie správnosti programového kódu v C++ pomocou unit testing framevorkov : bakalárska práca. Bratislava : Univerzita Komenského, 2008

[4] dostupne ná https://docs.python.org/2/library/unittest.html

[5] A quick introduction to the Google C++ Testing Framework, Arpan Sen, IBM DeveloperWorks, 2010-05-11, retrieved 2016-04-12

[6] BECK, K. 2005. Test-Driven Development: By Example. 7. vyd. Boston : Addison-Wesley Professional, 2005. ISBN 0-321-14653-0.

[7] dostupne ná https://en.wikipedia.org/wiki/Virtualization

[8] dostupne ná http://pnw.sk/virtualizacia-jej-vyhody/

[10] dostupne ná http://www.elearning.sk/co-je-elearning.html

[11] dostupne ná fttp://files.virtual-lab.sk/APVV%20VV-11/Clanky%20a%20publikacie/Edukacny%20softver_Daniela_Lehotska_Bezakova.pdf

[12] dostupne ná https://www.itnetwork.cz/html-css/webove-portfolio/tutorial-moderni-webove-portfolio-sass

Prílohy

Digitálna príloha –