EKONOMICKÁ UNIVERZITA V BRATISLAVE

Fakulta hospodárskej informatiky

Evidenčné číslo: 103004/I/2020/421000013782

Globálny distribuovaný systém rýchlej spätnej väzby adaptívneho riadenia prednášok

Inžinierska práca

Akademický rok 2019/20

Bc. Zuzana Demčáková

Globálny distribuovaný systém rýchlej spätnej väzby adaptívneho riadenia prednášok

Inžinierska práca

Študijný program: Informačný manažment

Študijný odbor: Ekonómia a manažment

Školiace pracovisko: Ekonomická univerzita v Bratislave

Vedúci práce: Ing. Jaroslav Kultan, PhD.

Máj 2020 **Bc. Zuzana Demčáková**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Zuzana Demčáková
Študijný program: informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor: ekonómia a manažment
Typ záverečnej práce: Inžinierska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Globálny distribuovaný systém rýchlej spätnej väzby adaptívneho riadenia prednášok

Anotácia: Súčasný stav prípravy študentov na prednášky a cvičenia je veľmi nízky. Tento problém by mohol pomôcť riešiť systém, ktorý by umožňoval viacnásobnú kontrolu vedomostí všetkých študentov počas prednášky a zároveň by uchovával údaje o úspešnosti odpovedí každého študenta. Takýto systém by umožnil adaptívne riadenie prednášky, zo stany učiteľa, na základe odpovedí študentov.

Vedúci: Ing. Jaroslav Kultan, PhD.
Katedra: KAI FHI - Katedra aplikovanej informatiky FHI
Vedúci katedry: Ing. Mgr. Peter Schmidt, PhD.

Dátum zadania: 23.10.2018

Dátum schválenia: 25.10.2018

Ing. Mgr. Peter Schmidt, PhD.
vedúci katedry

Ekonomická univerzite v Bratislave
FAKULTA HOSPODÁRSKEJ INFORMATIKY

Anotácia

Študijný program: Informačný manažment

Inžinierska práca: Globálny distribuovaný systém rýchlej spätnej väzby adaptívneho riadenia prednášok

Autor: Bc. Zuzana Demčáková

Vedúci práce: Ing. Jaroslav Kultan, PhD.

Jazyk práce: Slovenský jazyk

Máj 2020

Súčasný stav prípravy študentov na prednášky a cvičenia je veľmi nízky. Tento problém by mohol pomôcť riešiť systém, ktorý by umožňoval viacnásobnú kontrolu vedomostí všetkých študentov počas prednášky a zároveň by uchovával údaje o úspešnosti odpovedí každého študenta. Takýto systém by umožnil adaptívne riadenie prednášky, zo strany učiteľa, na základe odpovedí študentov s cieľom urobiť prednášku zaujímavejšou pre študentov. Učiteľ na základe získaných odpovedí vynechá tie časti témy, o ktorých sa pomocou testu presvedčil, že ich študenti ovládajú s cieľom, podrobnejšie vysvetľuje tie časti témy, ktoré študenti ovládajú menej, o čom svedčí menší počet správnych odpovedí zo strany študentov.

Kľúčové slová: distribuovaný systém, riadenie prednášok, dizajn aplikácie, Java, programovanie, databáza, nasadzovanie

University of Economics in Bratislava

FACULTY OF ECONOMIC INFORMATICS

Anotation

Degree Course: Informatic Management

Engineering thesis: Globally distributed rapid feedback system for adaptive lecture management

Author: Bc. Zuzana Demčáková

Supervisor: Ing. Jaroslav Kultan, PhD.

Language: Slovak

May 2020

The current state of preparation of students for lectures and exercises is very low. This problem could help solve a system that would allow for multiple control of the knowledge of all students during a lecture while retaining data on the success of each student's answers. Such a system would allow the adaptive management of lectures, from the teacher's tuition, based on student responses to make the lecture more interesting for students. Based on the answers received, the teacher omits those parts of the topic that the test has proven to be student-controlled, explains in more detail those parts of the topic that students have less control, as evidenced by fewer correct answers from students.

Keywords: distributed system, lecture management, application design, Java, programming, database, deployment

POĎAKOVANIE

Na začiatku práce by som sa rada poďakovala svojmu garantovi za užitočné rady, rodine za morálnu podporu počas celého štúdia.

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že vypracovanie tejto inžinierskej práce bolo spracované na základe poznatkov z uvedenej literatúry a vlastných poznatkov získaných z praxe.

.....

Zuzana Demčáková

Obsah

SLOVNÍK POJMOV

Pojem	Vysvetlenie
BATCH	Dávkový súbor a jeho spracovanie
BROADCAST	Správa a princíp prenosu správ v sieťach
CPU	Centrálna procesorová jednotka
DEPLOYMENT	Nasadenie systému do prevádzky
E-LEARNING	Vyučovací proces s použitím informačných technológií
HARDWER	Technické vybavenie počítača
LOGIN	Prihlásenie
OPEN SOURCE	Softvér, ktorého zdrojový kód je dostupný
PUSH	Funkcia pre posunutie zdrojového kódu z úrovne lokálnej na globálnu
RADIO BUTTON	Ovládací prvok, kde je možné označiť len jednu možnosť
REDUNDACIA	Nadbytočnosť dát
REPOSITORY	Informácia o mieste zdrojového kódu a softvérových balíčkov. Miesto odkiaľ systémy inštalujú
SMARTPHON	Inteligentné telefóny
TASK	Úloha

Úvod

Život v digitálnom svete nám prináša viac výhod, akými je napríklad používanie informačných systémov (IS). Digitalizácia sveta má v prvom rade prinášať prínos do každodenného života – v práci, škole ale aj v domácom prostredí. Práve táto myšlienka sa stala hlavným podnetom na vytvorenie tejto diplomovej práce. Vytvoriť systém, ktorý prinesie úžitok vyučujúcim, študentom a vytvoriť tak zábavnejšiu a efektívnejšiu formu výučby.

V sfére informačných technológií (IT) je veľké množstvo už vytvorených systémov, ktorých cieľová skupina je školstvo. Problémom však je, že tieto systémy nemusia spĺňať požiadavky aké konkrétna inštitúcia požaduje. Prispôsobenie požiadaviek je síce jedna z možností, no tento spôsob modifikácie je nákladný, pretože na vytvorenie zmien potrebujeme častokrát externé zdroje. Preto sme sa rozhodli vytvoriť vlastný systém na základe našich vlastných požiadaviek. Zadaním, ktoré zároveň hovorí o hlavných požiadavkách, sa stala téma ***„Globálny distribuovaný systém rýchlej spätnej väzby adaptívneho riadenia prednášok“***.

Hlavným pilierom tohto projektu sa stali tri základné požiadavky. Vzniknutý systém má byť taký systém, ktorý by bol nápomocný pre adaptívne riadenie prednášok a to takým spôsobom, aby učiteľ mal v reálnom čase informáciu o kvalitách študentov ale aj o tom, čomu nerozumejú. Tento spôsob výučby by mal priniesť študentom adaptívnu a zábavnejšiu formu prednášky, pretože im bude vysvetlené primárne to, čo nevedia. Zároveň chceme aby bol systém globálny, práve preto aby študenti aj vyučujúci mali prístup k systému kdekoľvek, čo môže vyučujúci využiť napríklad pri online výučbe. Ako poslednú požiadavku sme zadali - vytvorenie distribuovaného systému, pre zber konzistentných dát s vytvorením dvoch serverov s rôznymi prístupovými právami.

Prvá časť sa zameriava hlavne na objasnenie problematiky distribuovaného systému. Čitateľ práce sa na začiatku dostane do danej problematiky, kde je teoreticky vysvetlené fungovanie takéhoto systému. Pre samotnú realizáciu sme sa aj my potrebovali oboznámiť so spracovaním dát v danom type softvéru, ukladanie dát na distribuované úložiska a replikáciu údajov. Teoretická časť zahŕňa aj existujúce systémy na podporu vzdelávania. Na základe už existujúcich systémov sa vieme inšpirovať v rámci výhod a nevýhod,

a následne môže byť možná implementácia funkcií aj do nášho systému. Taktiež sa budeme venovať výhodám a nevýhodám používania takýchto systémov vo výučbe.

Po teoretickom získaní teoretických informácií o systéme, ktorý chceme vytvoriť, sme začali s teoretickým návrhom nášho systému. Kapitola zahŕňa ciele práce, konkrétne požiadavky na systém z pohľadu používateľov. Tieto požiadavky hovoria o funkciách, ktoré bude softvér vykonávať. V rámci teoretických návrhov systému sme zaradili aj dizajn systému a rôzne diagramy, ktoré nám pomôžu pri skonštruovaní systému.

Pokračovaním predošlej kapitoly je už popis vytvorenia praktickej časti. Na začiatku sme čitateľa oboznámili s rôznymi formami a nástrojmi, ktorými sa softvér dá vytvoriť a následne je detailný opis nami použitých nástrojov a spôsob, ako sme náš distribuovaný systém vytvorili. Systém je rozdelený na podkapitoly, ktoré predstavujú jeho komponenty. Práca sa zaoberá popisom riešenia pomocou programovacieho jazyka cez jeho dátovú základňu a vysvetlená je tiež aj časť deploymentu. Po vytvorení systému, ktorý bol najprv otestovaný nami sme vytvorili väčšie testovanie na študentoch našej univerzity. Výsledky testovania sa nachádzajú v experimentálne časti práce.

Záver práce patrí diskusii, možným vylepšeniam práce.

1 Súčasný stav

Každý človek musí na základe legislatívy absolvovať povinné školské vzdelávanie. To sa od jeho počiatku až po dnes rapídne zmenilo. Vzdelávací program bol nútený prispôsobiť sa vývoju doby nielen v sociálnych oblastiach ale aj v oblastiach vedy. Jedným z bežných predmetov sa tiež stala informatika. Práve tento predmet začal pôsobiť vo výučbe nie len ako samostatný predmet ale stáva sa súčasťou iných predmetov ako doplnok vyučovania. Takýto spôsob výučby, ktoré pozostáva z kombinácie klasického vyučovacieho systému s kombináciou informačno-komunikačných technológií, prináša lepšie dosiahnuté výsledky v rámci vyučovania pre študentov ale aj pre učiteľov. Takýto systém vyučovania pozostáva z 3 fáz: (M. Fikar, 2020), (Babinská, 2010)

- Prvá fáza – vytvorenie „živých prednášok“
- Druhá fáza – využívanie počítačov na cvičeniach a seminároch
- Tretia fáza – používanie elektronických materiálov pri samoštúdií

Takýto systém vyučovania však nesmie nijako nahradiť kontakt medzi vyučujúcim a študentmi, pretože takýto systém by mohol mať na študentov negatívny účinok. Následok zlého vyučovania u študentov narúša sociálne vzťahy a vytvára virtuálny svet. (M. Fikar, 2020)

Jednou z možností vzdelávania je tzv. alternatívna forma, ktorá poskytuje študentom možnosti vzdelávania a na základe jeho preferencií si môže vybrať sám. Avšak nie len na Slovensku ale aj v celom svete je takáto forma vzdelávania nevyužíva. Podstatou je, že klasická výučba sa úplne nedá nahradiť e-learningom (vyučovaním pomocou počítača). Rovnako to platí aj naopak. Ak by sa zaviedla iba jedna z foriem (napríklad ak by sa vyučovalo iba pomocou e-learning), druhá forma by úplne vymizla. Viacero expertov tak zastáva názor, že najlepšou formou výučby je kombinácia oboch foriem vyučovania. (M. Fikar, 2020)

Dôležité je aj spomenúť, že zavedenie počítačov do vyučovacích procesov je efektívne iba vtedy, ak vyučujúci ma kladný postoj a dostatočné vedomosti nie len vo vyučovacích oblasti, ale aj vo výpočtovej oblasti. Tieto techniky sa rovnako ako informatika stále posúvajú a vytvárajú tak nové zaujímavé systémy, ktoré prinášajú pozitívne výsledky od študentov. (M. Fikar, 2020)

Zamerajme sa na vytvorenie Informačných systémov. Takýto systém môžeme nazvať aj súhrnom prvkov, ktoré vykazujú určité vlastnosti pre dosiahnutie zhromažďovania, spracovania, prenosu dát a ich skladovanie. Takýto systém sa musí skladať z 3 hlavných prvkov (G. Reynolds, 2010):

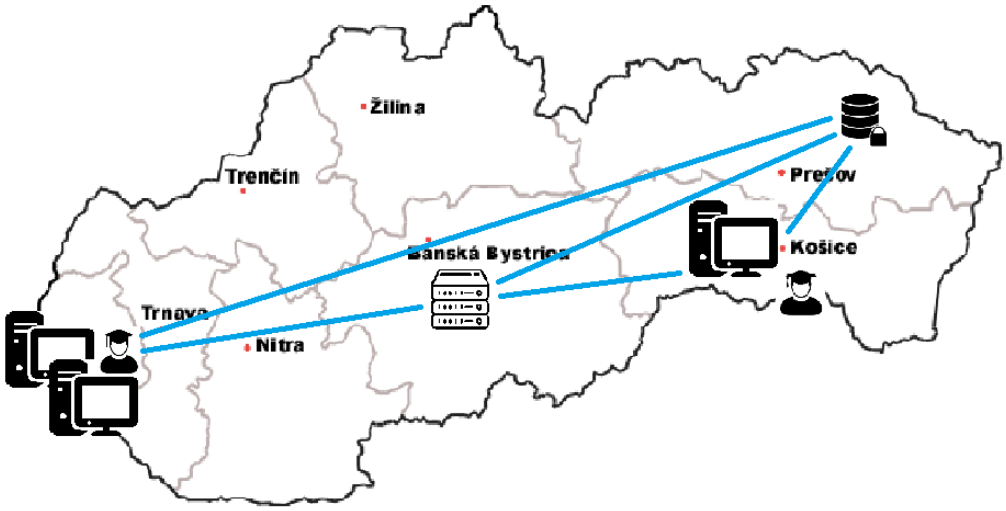
- Inputy - sú to prvky, ktoré do systému vstupujú aby boli ďalej spracované
- Processing – sú to také systémové prvky, ktoré transformujú vstupné prvky na výstupné
- Outputy – sú to koncové prvky, ktoré sú poskytované používateľovi.

Tieto tri prvky sú základnými komponentmi. Po ich vytvorení je možné systém rozšíriť o prvky pre riadenie, či kontroling alebo rozšírenie komponentu na vyhodnocovanie výsledkov, takzvané feedbacky. Takéto informačné systémy sú navrhnuté dobre vtedy ako používateľovi slúžia ako nástroj na rozhodovanie na troch riadiacich úrovniach (strategickej taktickej a operatívnej úrovni). (J. Vymětal, 2005)

1.1 Distribuovaný systém

Pre vysvetlenie pojmu distribuovaný systém a jeho fungovanie je potrebné začať pri distribuovanom výpočte. Ide teda o riešenie úlohy, na ktorej sa podieľa viacero počítačov. Výpočet môže prebiehať aj súbežne. Prečo boli ľudia nútení rozmýšľať nad rozdelením úloh? Prvú vec, ktorú bolo nutné riešiť bola geografická distribúcia dát. Problém bol v tom, že dáta sa nachádzali na rôznych serveroch v rozdielnych častiach krajiny, no spracovanie týchto dát mal prebiehať v rámci jedného procesu. Tento dôvod vznikol na základe práce veľkých firiem hlavne bankový systém. Vzhľadom na stále rozširovanie bankovej sféry bolo potrebné vyriešiť nie len distribúciu dát, ale aj vybaviť distribuované systémy špecifickými distribuovanými funkciami. To znamená, že niektoré z funkcií sú pridelené iba špecifickým procesorom na ktorom môžu byť vykonávané a ich výsledok je posielaný zadávateľovi požiadavky. (Janeček, 2020)

Druhým dôvodom sa stala zložitosť výpočtu a kapacita procesorov. Už dávno aj pred príchodom BigData sa procesy stávali čoraz náročnejšie, čo sa prejavuje na výpočtovej kapacite. Ide o rozloženie výkonu a viacero procesorov a dynamické (Mandík & Vrba, 2007) prideľovanie procesorov, čo urýchli dosiahnutie výsledkov a zároveň je možné do systému dostať viac požiadaviek naraz s väčšími kapacitami. (Janeček, 2020)



Obrázok 1 – Geografické znázornenie distribúcie dát [autor]

Distribuovaným systémom môžeme nazvať komplexný systém, ktorý pozostáva z menších autonómnych informačných systémov, ktoré sú navzájom prepojené, no ich fyzické umiestnenie je na rôznych miestach. Tieto čiastkové systémy môžu pracovať samostatne a spracovávať rozdielne druhy dát. Podstatnou informáciou je, samostatné fungovanie menších systémov, ktoré v spojený vytvárajú distribuovaný systém, so spoločnými a neustále aktuálnymi dátami. Dosiahnutie spoločných dát je častokrát vytvárané na báze zdieľania dát. Najčastejším dôvodom prečo sa vytvárajú distribuované systémy je tvorba komplexnejších systémov, ktoré poskytujú rôzne užívateľské právomoci za podmienky, že tieto čiastkové systémy budú využívať rovnakú dátovú základňu (Klimeš, 2020).

Medzi základné vlastnosti distribuovaného systému patrí:

1. TRANSPARENTNOSŤ – na základe poznatku, že systém pracuje na viacerých počítačoch, je potrebné aby tento jav nezasahoval do práce používateľa. Za transparentný systém považuje taký, ktorý sa používateľovi javí ako jednotný systém prístupný z jedného miesta. Táto vlastnosť však môže byť chápaná z rozdielnych hľadísk, ktoré vysvetľuje nasledujúca tabuľka (Klimeš, 2020):

Tabuľka 1 - Pohľady na vlastnosť „transparentnosť“ (Klimeš, 2020)

Pohľad Transparentosti	Popis
Access	Prístup používateľa k zdrojom má byť neobmedzený rovnako ako pri lokálom umiestnení.
Location	Používateľ nepotrebuje vedieť presné umiestnenie, kde práve proces pracuje.
Migration	Používateľ nevie, že sa zdroje informácie premiestnili.
Relocation	Procesy sú vykonávané na rôznych procesoroch, bez ohľadu na to aby ich premiestnenie používateľa obmedzovalo.
Replication	Používateľ nevie koľko kópií informácií distribuovaný systém obsahuje
Concurrency	Používateľa neovplyvňujú ďalší používatelia a systém
Failure	Používateľ nezaregistruje zaregistrovať, ak jeden zo serverov spadne.

2. PRISPÔSOBIVOSŤ – táto vlastnosť hovorí o závislosti medzi podsystémami. Prvým spôsobom je vytvorenie Autonómie. Systémy sú na sebe úplne nezávislé, dokonca môžu mať odlišný hardware, no v konečnom dôsledku vytvárajú jednotný systém. (Klimeš, 2020)

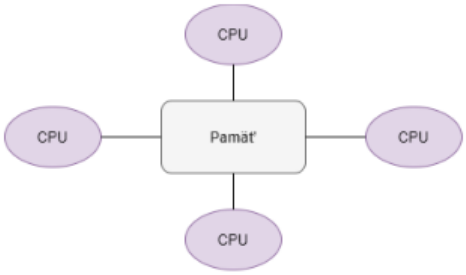
   Druhým spôsobom je decentralizované riadenie, kde procesory robia navzájom nezávislé rozhodnutia. Ich výsledky sú nakoniec reprezentované pomocou synchronizácie pomocou centralizovaného riadenia.
3. ŠKÁLOVATEĽNOSŤ – ide o možnosť prispôsobenia sa aj väčším požiadavkám. Ide o ľahšie pripájanie systémov a tvorbu nových požiadaviek. (Klimeš, 2020)
4. VÝKONNOSŤ – Distribuovaný systém poskytuje rýchlejšie odozvy, pretože spracované požiadavky si systém vie ľahšie manažovať ako jednoprocesorový systém. Ak je to potrebné, systém si požiadavky rozloží podľa aktuálnej potreby. Táto vlastnosť je dôležitá v dnešnej dobe, pretože objem požiadaviek na systémy a počet používateľov sa stále zvyšuje. (Klimeš, 2020)

1.1.1 Distribuované úložisko pre údaje

V rámci tvorby distribuovaného systému je potrebné vybrať si formu úložiska ktorú budeme používať. V tejto časti sa nachádza viacero spôsobov ako riešiť problém s úložiskom. Pre správny výber úložiska je potrebné oboznámiť sa so základnými typmi.

Zdieľaná pamäť

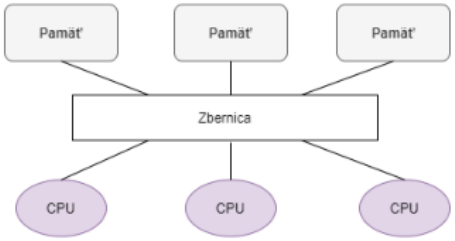
Hlavná myšlienka v tomto prístupe dátového úložiska je podmienka, že všetky CPU jednotky sa pripájajú na jednu pamäť. Tá slúži pre všetkých rovnako s rovnakými podmienkami a prístupmi pre každú jednotku. Zároveň platí, že ak jedna CPU zmení údaje v pamäti, druhá CPU jednotka môže pracovať iba s tými údajmi, ktoré zmenila predchádzajúca jednotka. Avšak problémom nie je ak sú dve rozdielne jednotky pripojené na pamäťové úložisko v rovnaký čas. Hlavnou výhodou je rýchle poskytovanie informácií procesorom. (Čerňanský, 2020)



Obrázok 2 - Zdieľaná pamäť [autor]

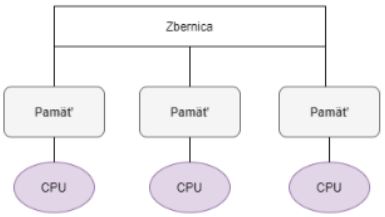
Pre toto zdieľanie pamäte sa využívajú dva prístupy, ktorými sú (Čerňanský, 2020):

- *Uniform Memory Acces (UMA)* – ide o pripájanie rovnakých procesorov, pripojených k pamäti v rovnakom čase.



Obrázok 3 - Zdieľaná pamäť typu UMA [autor]

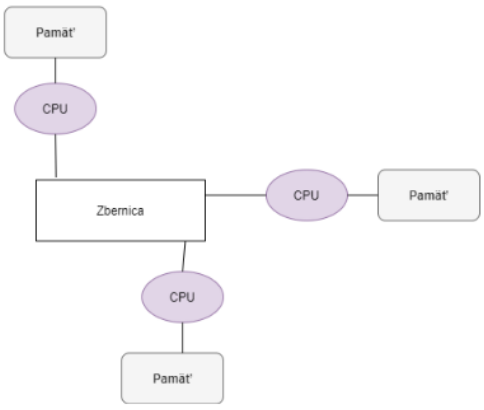
- Non – Uniform Memory Acces (NUMA) – ide o pripájanie viacerých procesorov, kde každý z nich má vlastnú pamäť a pripojené sú na sieť. To znamená, že pamäte procesorov, môžu byť navzájom adresované.



Obrázok 4 - Zdieľaná pamäť typu NUMA [autor]

Distribuovaná pamäť

Každý z procesorov pracuje nezávisle a má svoju vlastnú pamäť. Z tohto vyplýva, že ich pamäte nemôžu byť adresované navzájom ako to bolo pri NUMA. Globálny priestor pamäte, medzi týmito CPU neexistuje. Výhodou je, že zmeny v pamäti jedného procesora nemajú žiadny vplyv na druhú jednotku CPU. Aby si procesory mohli navzájom vymieňať informácie musí byť vytvorená takzvaná komunikačná sieť, ktorá je pravidelne synchronizovaná tak, aby mali procesory vždy korektné dáta pre ich samostatnú prácu. Túto sieť je potrebné naprogramovať pre každý distribuovaný systém osobitne, podľa jednotlivých požiadaviek. (Čerňanský, 2020)



Obrázok 5 - Zdieľaná distribuovaná pamäť [autor]

1.1.2 Distribuovane spracovanie

Ako sme už na začiatku kapitoly spomenuli, že podstatou distribuovaného spracovania je, že procesy bežia na rozdielnych serveroch, je potrebné vysvetliť ako tieto

procesy navzájom komunikujú a aké je volanie podprocesov. Na túto komunikáciu slúžia dva typy prostriedkov:

1. Systémy posielajúce správy
2. Volanie vzdialených podprogramov

Systém posielajúci správy

Dôležitou úlohou systému je tiež posielanie viacerým príjemcom správne správy. Tento prenos zabezpečuje prenosová infraštruktúra. Takýto spôsob, ktorý obsahuje redundanciu dát, sa nazýva „multicast comunincation“. Starý spôsob tejto komunikácie fungoval iba v rámci jednej podsiete cez broadcasty. Súčasný stav je realizovaný pomocou špeciálnych multicastových protokolov a je možný prenos cez internet. Takýto pokrok spôsobil, že môžu byť ľahšie realizovateľné systémy bez toho aby procesy boli rozdelené do určitých skupín. Teda ide o prepojenie procesov bez ohľadu na ich umiestnenie v sieti.

Jednou z možnosti ako môžu procesy medzi sebou efektívne komunikovať je cez IP adresy. Takáto komunikácia je nazývaná IP multicast. Pri takejto skupinovej komunikácií ide o vymieňanie správ pomocou adries. Systémy navzájom poznajú svoje IP adresy a tak zdrojový systém odošle správu a danú IP adresu. Distribúcia je potom zabezpečená pomocou sieťovej infraštruktúry pre všetkých príjemcom skupiny. Výhodou tohto spôsobu je, že systémy môžu byť navzájom pripájané a odpájané kedykoľvek. Potrebné však je zabezpečiť, aby systémy tieto zmeny vedeli zaregistrovať. (Šec, 2020)

Volanie vzdialených podprogramov

Prvá z technológií sa nazýva „Remote Procedure Call“ (RPC). Ide o procedúru, ktorá umožňuje spustenie procedúr pomocou vzdialeného prístupu na iných počítačoch, bez ohľadu na špeciálne definovanie komunikácie na vzdialenom počítači. Ide predovšetkým o typ aplikácie klient/server. Server má za úlohu poskytovanie procedúr, ktoré sú následne volané klientom. Prvotný podnet dáva klient zadaním požiadavky RPC na server, ten prevedie požadovanú procedúru a pošle ju späť klientovi spolu s hodnotami výsledku. (Valjašek, 2020)

Druhou technológiou je „Common Object Request Broker Architecture“ (COBRA). Je to objektový model pre špecifikáciu distribuovaných aplikácií. Je jazykovo nezávislý s podporou transparentného použitia distribuovaných objektov. Klientovi sú tak prístupné objekty bez ohľadu na to v akom jazyku boli implementované. Princíp spočíva v tom, že

klient k objektom pristupuje iba na úrovni generovania požiadaviek, pre ktoré závisí v akom jazyku sú napísané. (Valjašek, 2020)

1.1.3 Replikácia údajov

Keďže sa v teoretickej časti zaoberáme distribuovanými systémami je potrebné oboznámiť sa aj s takzvanými distribuovanými databázovými systémami a akú úlohu v nich zastáva replikácia údajov.

Význam distribuovaného riešenia pre databázy vznikol na základe zvyšovania počtu používateľov, ktorí sa pripájajú na databázy. V tomto prípade je centralizovaný databázový systém nepostačujúci. (K. Matiaško, 2009)

Hlavné charakteristiky pre tieto systémy sú: (K. Matiaško, 2009)

- Jednotlivé uzly medzi sebou komunikujú. Uzly majú medzi sebou sprístupnené informácie, no prezentujú sa tak akoby informácie boli uložené na jednotlivých uzloch.
- Distribuovaná databáza je reprezentovaná pomocou prepojených databáz, no používateľ s nimi pracuje ako s jednou komplexnou databázou.

Distribuovaná databáza sa často spája s pojmom replikácia dát, pre ktoré platia tieto vlastnosti: (K. Matiaško, 2009)

1. *Load Balancing* – hlavnou úlohou je aby sa zvýšila výkonnosť daného systému. Princípom je replikácia časti databázy alebo jedného jej uzla, pričom dochádza k ich synchronizácií s ostatnými uzlami. Tento proces vedie k vzniku kópií do viacerých uzlov. Pri väčšom počte týchto dátových priestorov, je tak možné rozdeliť požiadavky na viacero uzlov.
2. *High Availability* – hlavnou úlohou je zvýšenie dostupnosti na systém. Znova musíme mať viacero uzlov, ktoré budú používateľovi poskytovať informácie za podmienky, že ak je nedostupný jeden uzol, ostatné uzly vedia poskytnúť všetky informácie.

Pri replikácií dát je potrebné brať do úvahy, že sa môžu vyskytnúť rôzne problémy spojené s konzistenciou dát alebo s fungovaním ostatných aplikácií. Tieto problémy sú častokrát vyriešené samotným riešením vývojára, no chyby a ich riešenie môže súvisieť aj

so samotným typom replikácie, ktoré si tvorca systému vyberie. Aby sa predišlo chybovosti nasledujúca časť bude venovaná typom replikácií.

Prvým typom je synchrónna replikačná stratégia je založená na princípe, okamžitom prejave zmeny na všetkých uzloch. Ide teda o princíp že zmenu, ktorú používateľ vykoná sa automaticky ukáže aj na ostatných systémoch. Tieto zmeny sú úspešne vykonané až vtedy, ak sú zmenené údaje na všetkých uzloch. Hlavnou úlohou je zobrazenie vždy aktuálnych dát. Medzi nevýhody patrí dlhšia doba odozvy, keďže zmeny sa musia prejaviť na viacerých miestach. (Wong, 2009)

Druhým typom je asynchrónna replikácia, ktorá je opakom synchrónnej replikácie. Zmeny nie sú zobrazené na všetkých uzloch, no zmena je považovaná za úspešne vykonanú, pretože jej zmena sa prejavila na jednom uzle. Zmeny na ostatných systémových miestach sa prejavia až neskôr napríklad po dávkovom spracovaní batchom. Nevýhodou častokrát býva nekonzistentnosť dát všetkých uzlov. (Wong, 2009)

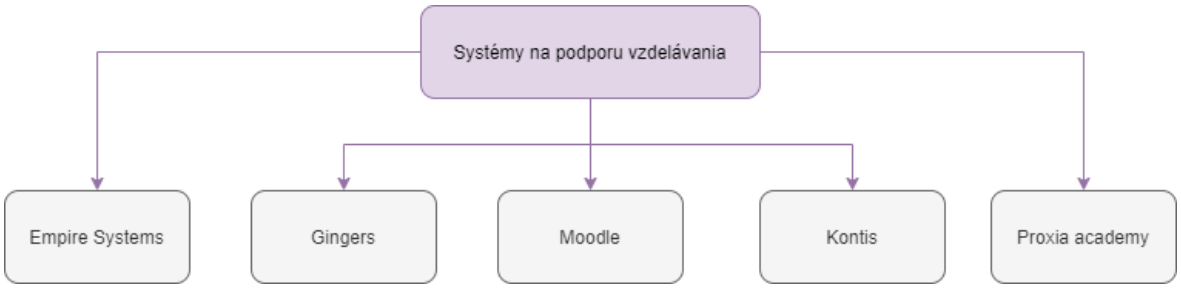
1.2 Systémy na podporu vzdelávania

Vzhľadom na stále viac rozvíjajúci sa stav technologického vybavenia sa tomuto trendu prispôsobujú aj vzdelávacie inštitúcie, ktorými sú aj školy. Aktuálne používanie elektronickej formy vo výučbe je reprezentované najmä ako doplnok výučby. Najčastejšie formy sú: nahrávanie prezentácií, odovzdávanie zadaní, či tvorba kurzov. (Prucha, 2009)

Hlavným problémom, prečo školy nechceli zavádzať systémy pre vzdelávanie do svojej formy štúdia boli najmä finančné náklady, ktoré boli spojené s nákupom softvérov a techniky, pretože klasická forma výučby bola výrazne lacnejšia. V dnešnej dobe je to však ale inak. Prístup k technike je jednoduchý, veľa systémov je tiež dostupných v smartphonoch a softvéry sú častokrát dostupné vo forme open source, takže sa dá povedať, že náklady sú takmer nulové. (Burgerová, 2008)

1.2.1 Rozdelenie systémov na podporu vzdelávania

Táto kapitola je zameraná na stručné definovanie existujúcich systémov na podporu vzdelávania. Kapitola má slúžiť na oboznámenie s nimi.



Obrázok 6 - Prehľad exitujúcich systémov na podporu vzdelávania [autor]

Obrázok 7 - Systém na podporu e-learning Moodle (MoodleNet, 2020)

Jedným z najčastejšie používaných systémov pre podporu vzdelávania v rámci stredných a vysokých škôl je systém Moodle. Je to systémový balíček, ktorý zahŕňa množstvo funkcionalít na podporu výuky. Ide o systém, ktorý sa prispôsobuje každodennému štýlu vyučovania. Systém je dostupný vo forme open soucre verzie, ku ktorej stačí verejná a ľahko dostupná GNU licencia. Používatelia tak majú plné právo na prístup k systému, no zároveň je Moodle chránený autorskými právami, zahrnutými v licencií. Na jeho používanie je postačujúci prístup k internetu, pretože systém je dostupný aj pomocou http path na mobilnom prehliadači, či v aplikácií, ktorá bola dostupná od roku 2012 pre operačný systém iOS. (MoodleNet, 2020)

Jednou z výhod systému je orientácia na používateľa. Vzdelávací systém Moodle obsahuje viacero svetových jazykov. Aktuálne systém podporuje až 40 variácií jazykov. Problém však môže nastať pri novších verziách, ktoré neobsahujú tieto jazykové mutácie. Vzhľadom na to, že je systém open source a update systému je častý, môže sa stať, že novšie verzie systému nebudú prispôsobené pre niektoré jazyky. (Ambruš, 2004)

Hlavnými funkcionalitami, ktoré tento systém poskytuje sú: (I. Porumbi, 2020)

- Vytváranie zoznamov študentov, ktorý sa v danom kurze nachádzajú
- Zaznamenanie činností používateľa v prehľadnom stave (pomocou kalendára)
- Prístup k materiálom v danom kurze (zobrazenie multimediálnych vyučovacích materiálov)
- Prístup k informáciám o kurze.
- Možnosť vytvorenia voľne prístupných alebo súkromných kurzov
- Možnosť odovzdávania výukových materiálov aj zo strany študentov
- Možnosť vytvárania textových poznámok

- Vytváranie a vyhodnocovanie testov

Moodle sa stal veľmi populárnym nástrojom pre podporu vzdelávanie hlavne preto že ponúka veľa funkcionalít, s ktorými sa jednoducho pracuje, takže je vhodný pre všetky typy používateľov. (I. Porumbi, 2020)

Obrázok 8 - Systém na podporu e-learning Empire (Systems, 2020)

Jedným z najväčších poskytovateľov v rámci nášho územia je spoločnosť Empire Systems s.r.o. Systém je vytvorený pre podporu e-learnigu nie len pre študentov ale aj pre firmy. Tento systém je založený na vytváraní profilov, vypĺňanie elektronických testov a možnosť prezerania si výsledkov testov. Tento systém sa využíva menej, pretože všetky funkcie, ktoré poskytuje, rovnako ale bezplatne zahŕňa vyučovací systém Moodle. (M. Fikar, 2020) (Systems, 2020)

Obrázok 9 - Systém na podporu e-learning Gingers (Gingers, 2020)

Alternatívou pre vyučovanie vytvorila aj spoločnosť Gingers s.r.o. Táto spoločnosť sa snaží klientovi poskytnúť správne systémové riešenie elektronického vzdelávania. Nejde o tvorbu ani poskytovania nového systému. Firma nevyvíja systém, ale poskytuje poradenstvo pri správnom výbere softvérového riešenia. (M. Fikar, 2020) (Gingers, 2020)

Obrázok 10 - Systém na podporu e-learning Kontis (Kontis, 2020)

V rámci územia Slevenska a Česka bol taktiež vytvorený systém pre e-learnig s názvom Kontis. Ide o systém s podobnými funkcionalitami ako Moodle. Spoločnosť poskytuje modulové riešenie, kde si zákazník môže vytvoriť multimediálne kurzy s použitím obrázkov, prezentácií, č vo forme videa. Systém od tejto firmy je najčastejšie využívaný malými či veľkými firmami na území Slovenskej a Českej republiky. (M. Fikar, 2020) (Kontis, 2020)

Obrázok 11 - Systém na podporu e-learning Proxia (Academy, 2020)

Na trhu v rámci e-learningu sa nachádzajú aj firma ako je napríklad PROXIA. Tento systém neslúži ako systém vyučovania ale spoločnosť poskytuje úpravu už existujúcich systémov na výučbu, ktoré prispôsobí zákazníkovi podľa jeho požiadaviek. Podľa nadefinovaných požiadaviek od zadávateľa sú tak vytvorené lekcie, prednášky či slovníky s pojmami. Ide o komplexnú administratívu vzdelávacieho systému takmer pre kohokoľvek. (M. Fikar, 2020) (Academy, 2020)

1.2.2 Výhody a nevýhody používania systémov na podporu vzdelávania

Po tom, čo si každá škola vyberie systém, ktorý najviac vyhovuje jej potrebám je potrebné zvážiť, či sa jeho používanie považuje za úspešné. V prípade malých inštitúcií je niekedy zavádzanie systémov menej užitočné a preto je pred zavedením e-learningu potrebné sa zamyslieť aké sú výhody a nevýhody pre učiteľov ale aj pre študentov.

Najčastejší dôvod prečo školy nezavádzajú tieto systémy sú finančné prostriedky pre vývoj vlastného systému, alebo poskytovaného systému, ktorý je potrebné spravovať. Výber systému závisí aj od vybavenosti školy. Za neúspechom pre zavedenie e-learningu stoja aj študenti a ich vybavenosť informačnými technológiami. Ak študenti doma nemajú prístup k internetu alebo počítačovým zariadeniam, je pre školu zbytočné zavádzať tieto systémy, ak nebudú mať spätnú väzbu. (Sudický, 2012)

Ak berieme do úvahy že prvá podmienka, že študenti aj škola sú dostatočne vybavení IKT a zároveň škola má systém, ktorý by používala je potrebné si uvedomiť, aké sú výhody elektronickej výučby.

K výhodám pre učiteľa môžeme zaradiť (Burgerová, 2008):

1. Materiály na výučbu môže učiteľ poslať študentom kedykoľvek, študenti sa tak môžu na hodinu pripraviť vopred ale zároveň môžu mať materiál k dispozícii aj po jeho odprednášaní (napríklad pri štúdiu k záverečným skúškam)
2. Materiál môže byť editovaný v čase, vzhľadom na meniaci sa stav a technológie tak učiteľ študentom môže kedykoľvek poslať prednášky v aktuálnej verzii

3. Možnosť testovania. Namiesto ručnej kontroly testov systém kontroluje a vyhodnocuje testy študentov automaticky. Ide o rýchlejší prístup k výsledkom a spoľahlivú spätnú väzbu pre vyučujúceho.
4. S použitím systémov je pre učiteľa jednoduchšie prideľovanie práce na doma. Zadania tak študenti jednoducho odovzdajú na základe pokynov k úlohe a času do ktorého bolo možné jeho vypracovanie.

K výhodám pre študenta parí (Burgerová, 2008):

1. Stály prístup k materiálom, ktoré sa nachádzajú v danom kurze
2. Prístup k materiálom odkiaľkoľvek, poskytuje študentom aj väčšiu možnosť individuálneho štúdia.
3. Pri používaní systémov na testovanie majú aj študenti (rovnako ako učitelia) rýchlejší prístup k výsledkom.
4. Študenti vďaka systémom na podporu vzdelávania si vedia predávať informácie a ľahšie komunikovať s učiteľom aj z domu.

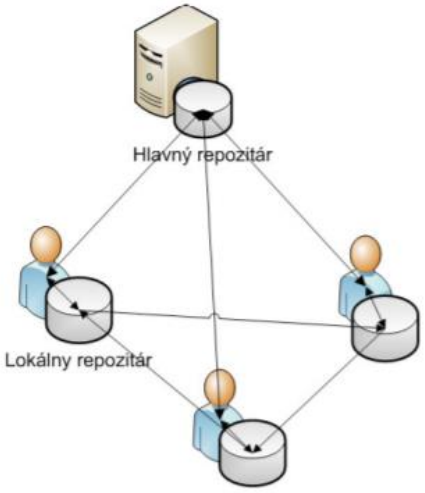
1.3 Prečo je dobré využívať distribuované systémy

Je dobré zavádzanie distribuovaných systémov do organizácií? Túto otázku by si mali na začiatku položiť všetky organizácie ešte pred ich zavedením do prevádzky. Vysvetlenie je možné rozdeliť z pohľadu používateľa ale aj z pohľadu vývojára. Vzhľadom na to že používateľský pohľadu používateľa bola venovaná predošlá kapitola tentokrát bude kapitola viac zameraná na výhody pre vývojára.

Prínos systémov z pohľadu používateľa do úvahy je braná hlavne celková efektívnosť a úžitok. Tento fakt je veľmi subjektívny, pretože systém by mal byť dobrý pre používateľa. Ak je tento subjekt spokojný systém môže byť považovaný za efektívny. (Molnár, 2001)

Využívanie distribuovaného systému z pohľadu vývojára je programovanie aplikácie jednoduchšie, pretože pri centrálnom systéme práce musí posúvať čiastkové kódy do centrálneho repozitára. Pri distribuovanom systéme to tak nie je a neexistuje žiaden centrálny repozitár. Programátor tak svoje zmeny len publikuje, pretože všetky repozitári sú rovnocenné. Ďalej je to výhoda testovania si kódu lokálne. Takýmto spôsobom sa programátor vyhne chybám, ktoré nájde v lokálnom prostredí. Ešte pred samotným

nasadením si programátor implementovanú novinku otestuje lokálne, zistí či systém funguje tak ako má a až následne tieto zmeny posunie na server. (Holák, 2020)



Obrázok 12 - Prístup vývojára k DS (Holák, 2020)

Distribuované systémy so sebou prinášajú aj flexibilný prístup. To znamená, že sa vytvára priestor pre širší prístup a intenzívnejší spôsob zdieľania kódu v rámci tímov, ktorý v konečnom dôsledku vytvorí používateľovi sa javiaci jeden systém na báze distribuovaného spôsobu riadenia. (Holák, 2020)

Na záver je potrebné poznamenať aj fakt, že použitie distribuovaných systémov prináša viac výhod najmä pri väčších projektoch. Avšak to neznamená, že malé projekty sa nedajú vytvárať distribuovane, no ich prístup a práca s nimi môže spôsobiť komplikovanejšie zdrojové kódy, vyššie náklady na správu systému a zaučenie ľudí ako systém správne používať. (Holák, 2020)

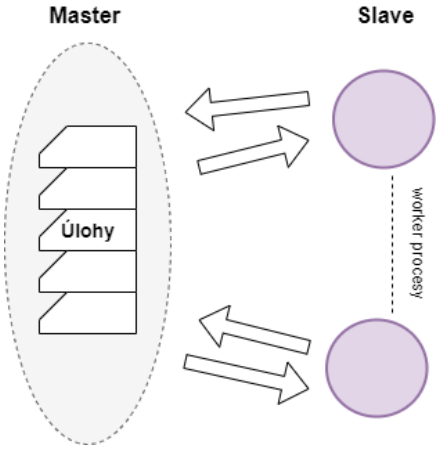
1.4 Adaptivita distribuovaných systémov na iné systémy

Ako bolo spomenuté v prvej kapitole, ktorá bola zameraná na distribuované systémy spomenutý centralizovaný a decentralizovaný spôsob nahrávania údajov. V tejto časti si vysvetlíme oba tieto prípady.

Centralizovaný spôsob riadenia úloh

Do tejto kategórie vstupujú 2 typy procesov. Ako prvý je zadávateľ/vykonávateľ (Master a Slave) procesov. Tieto typy berieme ako celok, pretože navzájom fungujú. Na začiatku sú udržiavané kolekcie úloh, toto zabezpečuje Master, ktorý jednotlivé úlohy (tasky) prideľuje Slave procesom. Ak už daná úloha prešla oboma procesmi, Slave si vypýta ďalšiu úlohu až kým nie sú dokončené úlohy ako celok, ktorý nazývame „Work pool“. Tento

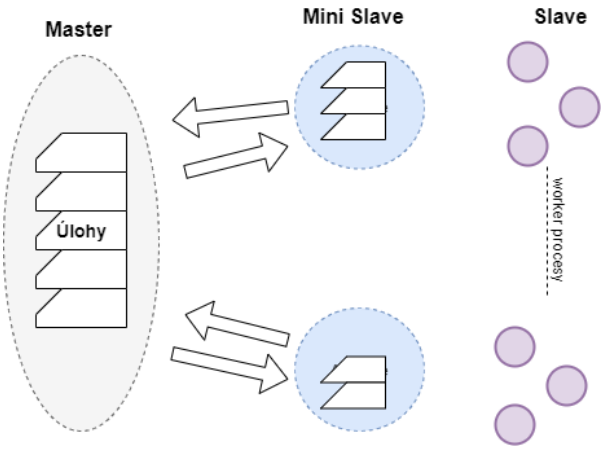
jednoduchý algoritmus zabezpečuje dosiahnutie splnenia úlohy tak, aby sme dostali aj požadované výsledky. Tento systém však nie je vhodný v rámci časovej výkonnosti. Najväčším problémom tohto mechanizmu je, chýbajúci multitasking. Každý Master-Slave proces môže v jednom momente vykonávať iba jednu úlohu. No v rámci malého počtu Slave procesov sú jednotlivé výpočty rýchle a jednoduché. (Y. Belabbas, 2020)



Obrázok 13 - Centralizovaný spôsob riadenia úloh [autor]

Decentralizovaná spôsob riadenia úloh

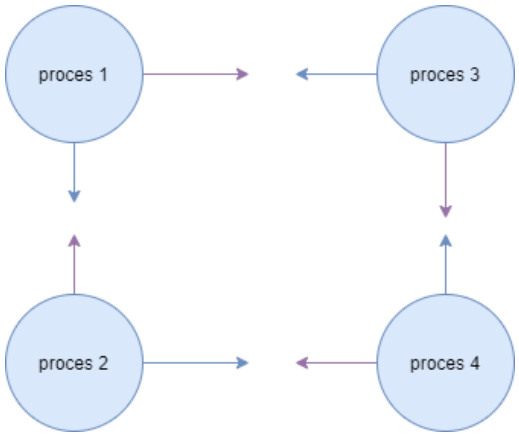
Na základe problému, ktorý bol spomenutý pri centralizovanom spôsobe bol vytvorený decentralizovaný spôsob. Práve ten má riešiť problém zaťaženia procesov. Princíp fungovania je ten, že celkový „work pool“ sa v prvej fáze rozdelí na menšie časti, ktoré sú nazvané „mini master procesess“. Takýmto spôsobom je vytvorený stále jeden Master proces a skupina menších samostatných slave procesov. Výsledky týchto menších skupín sa následne zoskupuje. Takýto mechanizmus je založený s myšlienkou, že pri tomto algoritme je malá pravdepodobnosť vyžiadania jednej a tej istej úlohy od Master pre 2 mini Slave procesy. (Y. Belabbas, 2020)



Obrázok 14 - Decentralizovaná spôsob riadenia úloh [autor]

Distribuovaný spôsob riadenia úloh

Decentralizovaný model distribuovaného spracovania nemá presnú štruktúru práce jednotlivých procesov. Každý z týchto procesov si zabezpečuje a riadi vlastnú kolekciu dát, pričom funguje komunikácia medzi procesmi, kde si môžu vymieňať výsledky svojich procesov. Dôležité je ale poskytovanie informácií. V modeli musí byť zapísané, ktorý proces komu poskytne informácie. Častokrát platí, že jeden proces neposkytuje svoje informácie všetkým ostatným. (Y. Belabbas, 2020)



Obrázok 15 - Distribuovaný spôsob riadenia úloh [autor]

1.5 Spôsoby tvorby distribuovaných systémov ako nástroj na vyučovaní študentov

Po teoretickej časti, v ktorej bol teoreticky opísaný distribuovaný systém a jeho časti sa dostávame k ich tvorbe. Aj v tomto prípade sa vyskytujú tri spôsoby ako na to. Výber spôsobu môže byť ovplyvnení aj typom distribuovaného systému. Dôležitou skutočnosťou je, že najzložitejšiu časťou je zvyčajne ladenie programu. (Ledvina, 2020)

Medzi základné typy budovania systému patria (Ledvina, 2020):

- Vytváranie od ZÁKLADOV – tento spôsob znamená, že systém bude vytváraný od začiatku, takže programátor začína na „čistom papieri“. Výhodou je, že systém je vytvorený používateľom na mieru, no práca programátorov je o to zložitejšia. Z toho vyplýva aj fakt, že vytváranie systému je aj časovo náročnejšie. Takýto spôsob budovania nedovoľuje preberať fungujúce a zabehnuté oblasti aplikácií, čo považujeme za najväčšiu nevýhodu tohto spôsobu.
- Vytváranie MODIFIKÁCIOU – takáto technika minimalizuje množstvo programátorského úsilia, pretože využívajú jadro už existujúceho systému.

Väčšinou ide o modifikáciu sekundárnych funkcií. Možnosť modifikácie môže vzniknúť aj zmenou centralizovaného systému na decentralizovaný.

- Vytváranie NADSTAVBOU – v takomto budovaní sa snažíme nemeniť existujúci systém ale vytvoriť jeho nadstavbu. Z toho vyplýva že pôvodný systém nebude zásadne modifikovaný. Systémové jadro je modifikované iba o funkcie riadiace nové funkcionality. Realizácia nastáva pridaním jednej alebo viacerých vrstiev distribuovaného systému. Príkladom tohto systému je MS-DOS[1].

[1] MS-DOS – operačný systém od firmy Microsoft.

2 Cieľ

Základným cieľom práce je analýza existujúcich metód pre riadenie prednášky, s vytvorením systému na skvalitnenie výučby a úsporu času. Na splnenie cieľov bolo vytvorené niekoľko úloh. Po ich splnení, môžeme povedať, že systém je správny a splnil zadanie našej práce. Hlavnými úlohami bolo:

- Analyzovať požiadavky na systém
- Vytvoriť schému správania sa systému
- Graficky znázorniť a vysvetliť dizajn aplikácie
- Vybrať vhodné nástroje na tvorbu systému
- Vybrať prostredie pre nasadenie aplikácie
- Pripraviť si testovacie dáta do systému
- Pripraviť návrh pre rast aplikácie

Podnet pre vytvorenie tejto diplomovej práce vznikol za účelom zefektívnenia prednáškového systému. Spôsob výučby, aký doteraz poznáme sa stále zakladá na systéme, kedy učiteľ vysvetľuje učivo, no od študentov nedostáva žiadnu spätnú väzbu počas priebehu prednášky. Zavedením tohto systému tak učiteľ, nie len že dostane odozvu na prednesenú tému, ale taktiež získa prehľad o správne zodpovedaných otázkach a nesprávnych odpovedí. V danom prípade učiteľ získa prehľad o stave vedomostí študentov v reálnom čase počas prednášky. Otázkam, ktorým študenti rozumejú, sa venuje menej a tým, kde odpovede študentov boli nejednoznačné, sa následne môže viac venovať. Papierové formy výučby sú nahradzované elektronickým online spôsobom. Táto forma riadenia prednášok prináša rýchlejšiu odozvu, ako použitie papierovej formy, pretože systém sám porovná odpovede a učiteľ tak dostane iba prehľad. Ďalšou výhodou pri použití tohto distribuovaného systému je uchovávanie dát. Na rozdiel od skladovania papiera, je dostupná databáza, ktorá poskytuje aktuálne aj historické dáta. Takýmto riešením vieme využiť vlastnú databázu, ktorá zabezpečí, že k nej má prístup iba kompetentná osoba. Systém môžeme následne integrovať do viacerých predmetov, na základe generovaných otázok. Ako poslednú požiadavku môžeme definovať distribuovanosť systému, ktorý zabezpečí prístup k systému odkiaľkoľvek, no iba určitým osobám s obmedzením prístupu k dátam.

2.1 Plánovanie a požiadavky na aplikáciu

Pred samotnou fázou programovania je potrebné vytvoriť analýzu, ktorou získame požiadavky. Do úvahy je potrebné brať rôzne kombinácie požiadaviek, pre nami vytvorené časti, a tak vytvoriť požiadavky. Pre riešenie nami zadaného cieľa boli požiadavky rozdelene na dve skupiny: pre používateľa aj pre systém. Téma „Globálny distribuovaný systém rýchlej spätnej väzby adaptívneho riadenia prednášok“, je zameraná hlavne na používateľa, pre vytvorenie jednoduchého a zrozumiteľného systému, tak aby vykonával všetky potrebné funkcie. Distribuovaný systém na riadenie prednášky z pohľadu používateľa je ešte členený na kategórie, ktorými je učiteľ ako používateľ a študent ako používateľ.

Požiadavky Učiteľa ako používateľa

Používateľ Učiteľ, je entita, ktorá riadi priebeh prednášky a následne ju riadi. Aby to bolo efektívnejšie, zadá do systému testovacie otázky:

- Počet zadaných otázok do systému je neobmedzený
- Počet odpovedí na jednu otázku je maximálne 5, no minimálny počet nie je určený (systém povoľuje zadať napríklad iba 2 odpovede)
- Na každú otázku je možná jedna správna odpoveď

Po tom, čo študenti zodpovedajú na otázky chce používateľ vidieť iba prehľady odpovedí na základe ktorých bude vznikať priebeh prednášky:

- Prvý pohľad bude znázorňovať otázky s najmenším počtom správnych odpovedí, čo vypovedá o témach, ktoré študenti nepochopili, a potrebujú sa im venovať najviac, témy týchto otázok by mali byť považované za základ prednášky.
- Ďalšou podstatnou tabuľkou bude Zoznam najčastejších odpovedí, kedy bude tabuľka usporiadaná podľa poradia testu. Ku každej otázke budú zobrazené jej odpovede a číselný údaj, koľko študentov vybralo danú možnosť.

Jednou z posledných požiadaviek používateľa je nedostupnosť výsledkov testov pre študentov. Keďže entita Učiteľ má inú rolu ako študenti, výsledky systému by taktiež mali byť iné. Zároveň je potrebné ochrániť tieto údaje, aby nenastal prípad, že študenti na základe výsledkov boli ovplyvnení pri vypĺňaní svojich testov.

Požiadavky Študenta ako používateľa

Používateľom Študent, môže byť akýkoľvek človek, ktorý sa zúčastní prednášky. Touto entitou je myslená entita, ktorá sa zúčastňuje priamo prednášky ale môže byť na prednáške pripojený aj z externého prostredia.

- Vzhľadom na skutočnosť, že študent nemusí mať žiadne vedomosti o systéme je prvou požiadavkou aby bol systém sprístupnený čo najjednoduchšie pomocou internetu.
- Študent chce zachovať svoju anonymitu počas vypĺňania testu
- Ďalšia funkcionalita, ktorá je pre entitu prístupná je prehľadnosť testu. Niekedy sa môže stať, že sa niektoré dáta stratia alebo ich nevidno. Preto študent potrebuje mať grafické rozdelenie jednotlivých otázok pre lepšie pochopenie.

Požiadavky na systém

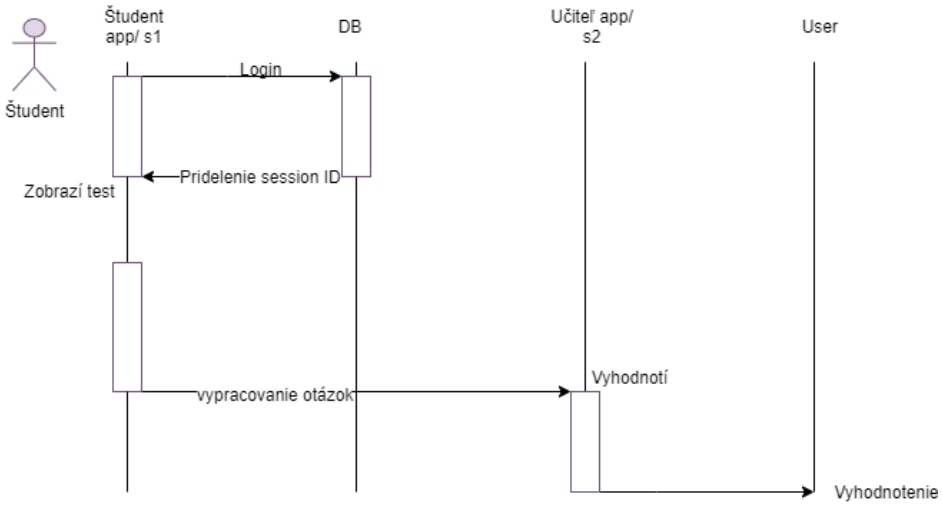
Systémové požiadavky budú hovoriť o základných funkcionalitách, ktoré musia byť splnené aby sa vykonali všetky úlohy správne:

- Aby sa používatelia mohli pripojiť na systém je potrebný prístup na internet. Pre systém je relevantné zariadenie, na ktorom sa bude front-end vykonávať.
- Systém potrebuje poznať IP adresy databázy z ktorej bude čerpať dáta pre zobrazenie obrazoviek používateľom. Zároveň tieto databázy musia byť naplnené informáciami.
- Dôležitou požiadavkou je tiež, že jeho používatelia sú oboznámený s prácou so systémom a pripojením sa na konkrétnu IP adresu.

Splnením všetkých požiadaviek by mal systém všetkým entitám poskytovať správny priebeh prednášky. Rovnako tak môžeme systém považovať za správny, pretože jeho využitie je tzv. „user friendly“, čo znamená, že systém je jednoduchý pre používanie a zároveň sú s ním ľudia spokojný, pretože spĺňa všetky ich požiadavky. Niekedy je potrebné použiť detaily ako farebné prevedenie, alebo rozloženie komponentov tak, že prezentácia a funkcionality systému používateľovi zapáčia natoľko, že systém začne používať častejšie.

2.2 *Sekvenčný diagram*

Pre pochopenie fungovania našej analýzy je potrebné zadefinovať modely, na základe ktorých bol celý koncept vytvorený. Riešenie bolo definované tak, aby bolo čo najviac jednoduché pre používateľa, učiteľa aj študentov. Tým je myslené, že systém rozlišuje sekciu pre študenta a sekciu pre učiteľa. Následný diagram vyjadruje interakcií. Z tohto pohľadu vidíme ako systém komunikuje s danými servermi a akú odozvu im poskytuje v priebehu času.



Obrázok 16 - Sekvenčný diagram [autor]

Obrázok – sekvenčný diagram, zobrazuje fungovanie a jednotlivé časti systému, ktoré sú v tejto časti vysvetlené a jednotlivo popísané. Návrh tohto spracovania bol založený na vytvorení dvoch nesúvisle od seba bežiacich serverov a osobitne pripojenej databázy.

Server s1 je vytvorený pre prístup študenta. Aplikácia je sprístupnená na určenej doméne, na ktorú sa študent pripojí. Prístup k nej majú všetci študenti, ktorí poznajú zadanú linku. Po pripojení sa na sever systém zobrazí klientovi základnú obrazovku pre prihlásenie. Kliknutie na prihlasovacie tlačidlo systém zachytáva a vytvára v databáze nový riadok s údajmi, ktoré študent zadáva. Pri vstupe je zároveň vygenerované session_id[2], pod ktorým je študent zapísaný pri ďalšom spracovaní. Toto označenie používateľa nám zabezpečuje čiastočnú anonymitu, na základe toho do akých tabuliek má používateľ databázy prístup. No zabezpečuje aj spätné dohľadanie a priradenie odpovedí danému študentovi. Ďalším krokom servera 1 je zobrazenie testu, pre každého študenta sa zobrazí totožný test pre vyplnenie.

[2] Unikátne číslo – JavaServer TM – Online : http://www.itu.dk/~sestoft/javaservlets/session_track/SessionTr.html, navštívené 2.1.2020

Každá otázka obsahuje jednu správnu odpoveď (check box). Systém povoľuje aj vynechanie odpovedí, čo znamená, že otázku, ktorá nebola zaznačená považuje za neoznačenú, no nutne sa nevyžaduje jej vyplnenie. Na konci testu je zobrazené tlačidlo pre odoslanie odpovedí na vyhodnotenie. V tomto bode sa študentovi zobrazí obrazovka s úspešným výsledkom. S1 ďalej nemá prístup k spracovaným výsledkom.

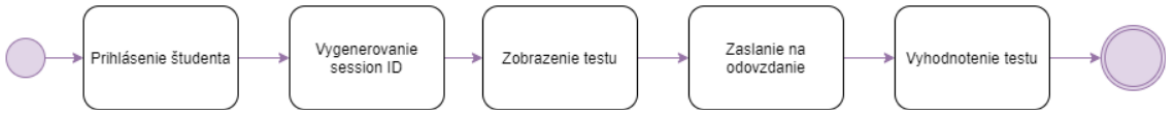
Server S2 je výhradne určený pre učiteľa. Na rozdiel od serveru 1 učiteľ je používateľ, ktorý má prístup iba k výsledkom spracovania testu. Dôvodom takéhoto riešenie je myšlienka, že učiteľ je zadávateľ testu a v jeho záujme sú iba výsledky v prehľadnom tvare. Prístup do aplikácie je na začiatku rovnaký ako pre študenta a to pomocou zadaného linku. Po zobrazení je na spodnej obrazovke možnosť prekliknúť sa na „s2“. Pre tohto klienta je zobrazená obrazovka pre zadanie „MASTER_KEY“, ktorý pozná iba učiteľ, aby študenti nemali prístup pre jednotlivé prehľady. Ak je zadaný kľúč nesprávny, zobrazená je obrazovka s informáciou, že zadaná hodnota nebola správna. No pri správnom zadaní kľúča sa používateľovi zobrazí obrazovka obsahujúca vyhodnotené otázky v prehľadoch. Samozrejme táto jednoduchá autorizácia pomocou „MASTER_KEY“ môže byť nahradená povolením konkrétnej IP adresy stroja, VPN prístupu alebo silnejším prihlasovaním pomocou mena a hesla.

Databáza je prepojená s oboma servermi a aplikácia s ňou pracuje počas jej celého používania. Jej úložisko je oddelené so serverovou časťou, na ktorú sa pripája pomocou IP adresy. Prepojenie jednotlivých tabuliek databázy je prehľadne zobrazené v časti Riešenie - Databázová schéma. Sekvenčný diagram nám len zobrazuje, odpoveď pre server 1 so session ID, ktoré sa zobrazuje v hypertextovom linku pri vypĺňaní testu a zároveň poskytuje vyhodnotené výsledky pre server 2.

Ako **User** je v sekvenčnom diagrame označený Používateľ, ktorý výsledné dáta bude ďalej využívať na riadenie prednášky a na základe nich sa vytvorí jej priebeh .

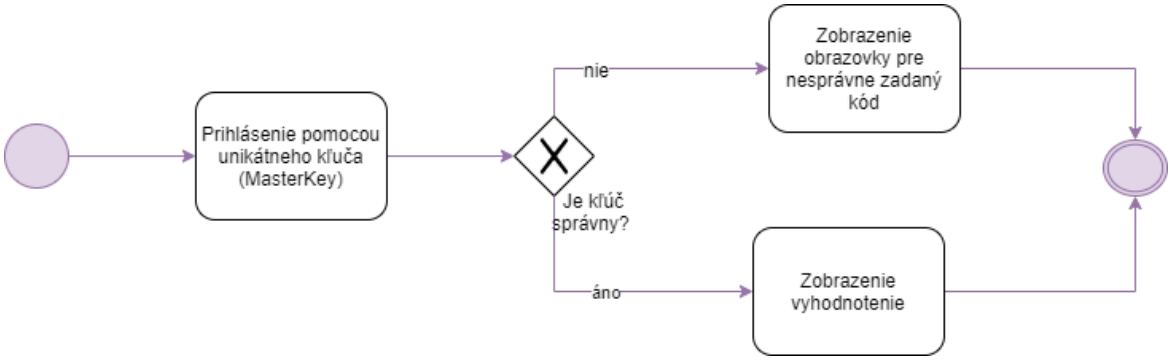
2.3 Activity diagram

UML diagramy boli zostavené na lepšie pochopenie stavov v ktorých sa servery nachádzajú. Pomocou piktogramov sú zobrazené toky týkajúce sa krokov systému. Aktivity diagramy sú zložené vždy zo začiatočného a koncového bodu. Stavy v ktorých sa entity nachádzajú medzi nimi popisujú reálny stav, ktorý systém kontroluje. (GeeksforGeeks, 2020)



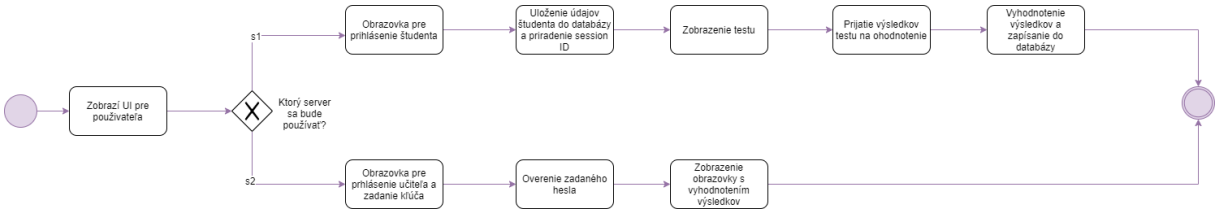
Obrázok 17 - Activity diagram – študent [autor]

Vyplnenie testu pre študenta je znázornené v obrázku Activity diagram – študent. Proces tak vysvetľuje celý proces od začiatku do konca. Na úvod dostane študent obrazovku s prihlásením. Každý zo študentov je považovaný za novo zapísaného študenta. Proces je nastavený tak, že pri jeho opätovnom spustení, študent nemá vytvorené konto ale je považovaný za nového klienta. Je potrebné aby študent zaevidoval svoje meno a priezvisko. Tieto údaje sú ďalej ukladané spolu s prideleným identifikačným číslom (session ID). Zároveň s pridelením identifikátora sa študentovi zobrazuje test. Študent zodpovie otázky a zašle ich na odovzdanie. Po zhodnotení správnych a nesprávnych odpovedí dochádza k ich vyhodnoteniu.



Obrázok 18 - Activity diagram – učiteľ [autor]

Učiteľ začína so systémom pracovať ak sa mu zobrazí obrazovka pre zadanie kľúča, ktorý študenti nepoznajú, a tak nemôžu pristupovať k vyhodnoteniam a tiež k správnym odpovediam na otázky. Po zadaní tohto kľúča vstupuje údaj do rozhodovacej brány, kde sa vyhodnotí správnosť zadaného kódu. Ak je kód vyhodnotený ako správny, učiteľ má prístup k vyhodnoteniu a prehľadom. Pri nesprávnej kombinácií, je prístup k dátam zakázaný a zobrazuje sa obrazovka s informáciou, že zadaný kľúč nie je správny.



Obrázok 19 - Activity diagram – systém [autor]

Obrázok Activity diagram – systém zachytáva globálny zjednodušený návrh systému v ktorom je vidieť, že v prvom kroku je potrebné zistiť, ktorý zo serverov bude použitý.

2.4 Návrh vonkajšej formy aplikácie

Pri vytváraní nových aplikáciách, je vždy potrebné sa zamyslieť, nad ich prezentáciou a grafickým znázornením. K systému sa zvyčajne dostávajú používatelia, ktorí ho nepoužívajú každodenne a tak je potrebné aby funkcionality boli jednoduché, zrozumiteľné a použiteľné pre kohokoľvek. Pri tvorbe by sa nemalo zabúdať aj na vizuálne prvky (vonkajší dizajn), ktoré tak používateľa navádzajú aby sa k systému vrátil.

Vzhľadom na zameranie systému, je jeho vizuálna stránka ladená do farieb fakulty. Použitá farba bola v odtieni fialovej „Dark moderate violet“ . Jej označenie podľa HEX schémy je nasledovné číslo #633580. Všetok zobrazený text bude v štýle „Verdana“. Výber textu vznikol na základe subjektívnych preferencií. Každá zo zobrazených stránok bude obsahovať tri časti: (Color HEX, 2020)

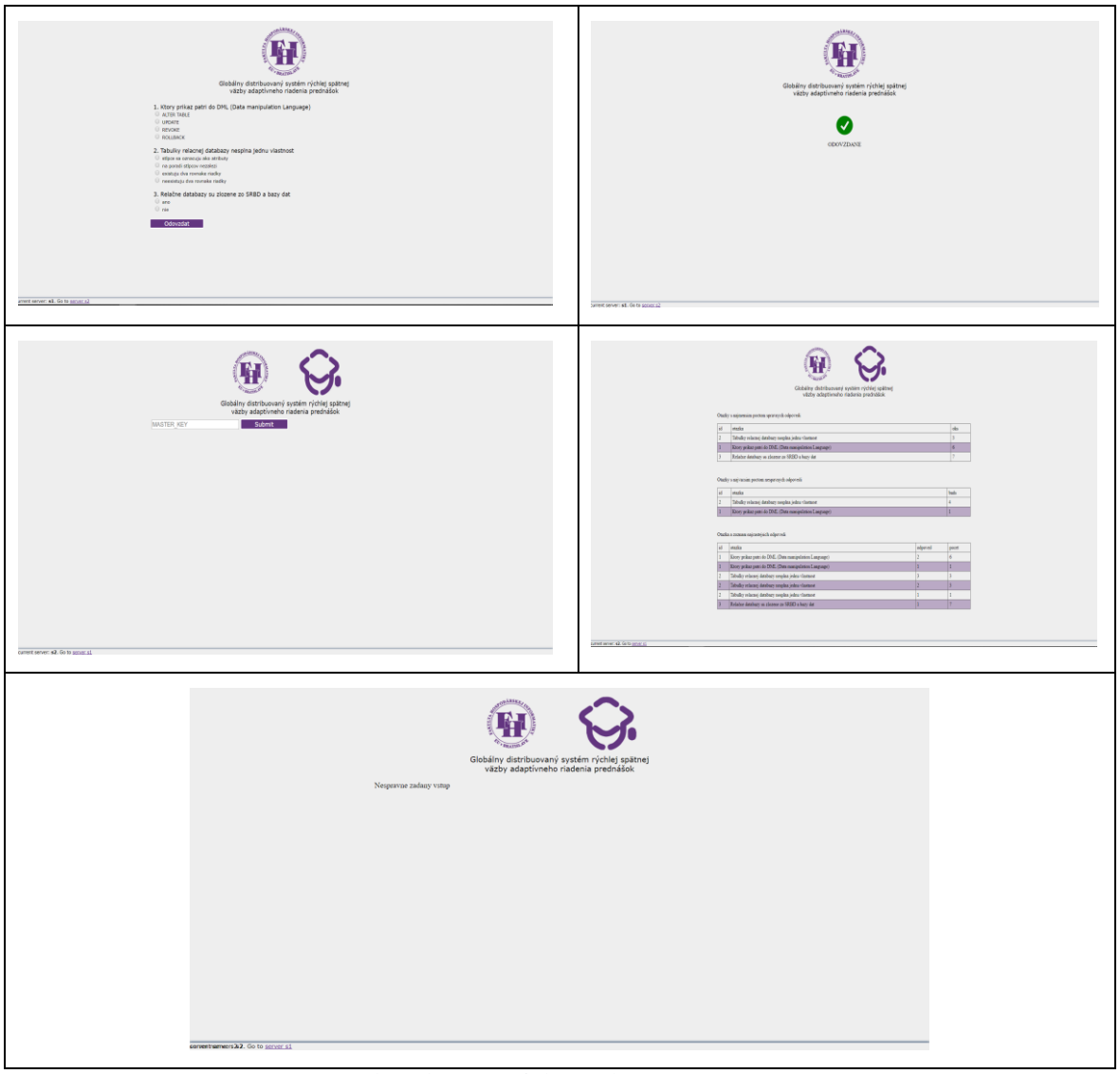
- Hlavičku
- Telo
- Pätu

V hlavnej vrchnej časti sa nachádza logo školy spolu s názvom tejto práce. Telo sa podľa zobrazenej stránky bude odlišovať. Päta strany slúži ako informácia pre používateľa o tom, na ktorom zo serverov sa nachádza poprípade ak sa chce prepnúť na iný server, táto časť obsahuje odkaz pomocou ktorej bude používateľ presmerovaný na iný server.

Obrázok 20 - Grafický návrh stránky [autor]

Zameriame sa ale na hlavnú obsahovú časť systému. Telo systému by malo v každej svojej časti niesť podstatu toho, čo systém zachytáva. Na Obrázku Grafický návrh stránky, je vyobrazenie *s1*- login (prihlásenie študenta). Na tejto obrazovke sa budú nachádzať dve ikony, do ktorých študent zadá svoje identifikačné údaje. Na obrazovke nebolo potrebné doplniť ďalšie polia, pretože systém nepotrebuje zachytávať nové informácie o študentovi. Nad poliami sa nachádza informácia, Prihlásenie študenta, čo študentovi slúži ako oznámenie o tom, čo bude na tejto obrazovke vykonávať. Polia sú zobrazené bielou farbou. V každom poli sa nachádza text s informáciou na čo jednotlivé pole slúži. Po tom, čo klient klikne do poľa sa text z poľa zmaže a prepíše sa textom ktorý chceme do danej sekcie zapísať.

Doplnkom polí je tlačidlo vstúpiť. Tlačidlo taktiež slúži na potvrdenie zadaných informácií do polí. Zobrazenie tlačidla je fialovej farby, rovnako ako logo v hlavičke obkolesené bielou čiarou. Zobrazenie textu v tlačidle je bielej farby. Po prejdení kurzora na tlačidlo sa farba tlačidla zmení z fialovej na bielu a keďže by biely text nebolo na bielom podklade vidieť, text sa zmení na čiernu farbu. Zmena tlačidla je zavedená do grafickej úpravy aby používateľ mal lepší prehľad o tom, či kurzorom prešiel po tlačidle a že tlačidlo je aktívne.



Obrázok 21 - Grafický návrh stránok [autor]

Zobrazenie obrazovky Testu sa bude znova líšiť len v časti Telo. V tejto časti sa zobrazujú otázky na ktoré má študent odpovedať. Element je naformátovaný na stred obrazovky. Každá z otázok je vyznačená poradovým číslom a obsahom otázky. Pri dlhších otázkach sa vytvorí zalomenie textu a veta zostane iba v blokovom elemente. Otázka tiež obsahuje odpovede, ktorých môžu byť najmenej jedna a maximálny počet je určený na 4. Odpovede sú zobrazované pod sebou a každá z nich obsahuje možnosť označiť iba jednu správnu odpoveď, pre označenie správnej odpovede. Po kliknutí na jednu z odpovedí sa biely krúžok označí. Túto odpoveď je možné meniť kliknutím na inú z odpovedí.

Do grafického dizajnu je zaradené tiež vyznačenie aktuálnej otázky. Ak študent prejde myšou na konkrétnu otázku, jej obsah spolu s odpoveďami, ktorej k nej prislúchajú, sa podfarbia bielou farbou. Ide o funkciu aby boli lepšie vyznačené, ktoré z odpovedí sú určené ktorej otázke. Otázky sú zároveň medzí sebou oddelené medzerou.

Na konci stránky sa nachádza tlačidlo „Odovzdať“, ktoré sa správa rovnako ako na prihlasovacej obrazovke. Toto tlačidlo teraz slúži na odovzdanie testu, ak už študent odpovedal na všetky otázky alebo už v teste ďalej nechce pokračovať. Po kliknutí na tlačidlo sa používateľ dostane na ďalšiu z obrazoviek.

Posledná z obrazoviek slúži ako informačná obrazovka, a jej telo obsahuje informáciu o úspešnom odovzdaní testu. Na začiatku je použité zelené logo. Pod logom sa nachádza text „ODOVZDANÉ“ ako doplnok loga.

Pre hlavičku *servera 2* bude doplnená ikona, s akademickým motívom ako doplnok informácie, že sa používateľ nachádza na serveri určeného pre učiteľa. Obrazovka prihlásenia pre tento server sa bude zhodovať s login obrazovkou ktorá je určená študentovi s rozdielom, že tu sa nachádza iba jedno pole, do ktorého je možné vpísať text a tým je miesto určené pre vloženie kľúča, ktorým učiteľ pristupuje k výsledkom. Logika tlačidla zostáva rovnaká ako na ostatných obrazovkách.

Obrazovka s výsledkami pre server dva sa líši od všetkých ostatných. V mandatórnej časti je zobrazených niekoľko tabuliek s prehľadmi výsledkov. Po vyhodnotení testov, sú výsledky vkladané do troch tabuliek. Nad každou z tabuliek sa nachádza popis, čo je jej obsahom.

Pre testovanie sú vytvorené tri typy prehľadov:

- Otázky s najmenším počtom správnych odpovedí – tabuľka zobrazuje otázku a počet správnych odpovedí na každú z nich
- Otázky s najväčším počtom nesprávnych odpovedí – tabuľka obsahuje otázku a počet zlých odpovedí na ňu
- Otázky a zoznam najčastejších odpovedí – pre túto tabuľku sa zobrazuje otázka, číslo odpovede a počet, koľkokrát študenti klikli na túto odpoveď.

Prvé dve tabuľky budú zobrazené pomocou troch stĺpcov. V prvom stĺpci sa nachádza číslo otázky, v druhom obsah otázky a v poslednom stĺpci sa nachádza počet odpovedí na konkrétnu otázku (pre prvú z tabuliek sa zobrazujú správne odpovede a pre druhú nesprávne odpovede). Riadky sú farebne odlíšené (bez výplne/ fialovou) pre lepšiu orientáciu v tabuľke.

V poslednej z tabuliek je pridaný štvrtý stĺpec. Obsah prvých dvoch stĺpcov je rovnaký ako v predošlom obsahu. Tretí stĺpec označuje číslo odpovede a posledný stĺpec

zaznamená počet klikov na danú odpoveď. Pokiaľ niektorú z odpovedí neoznačil ani jeden zo študentov, jej číslo sa v tabuľke nezobrazuje. Ak učiteľ zadá nesprávne prihlasovacie údaje bude mu zobrazená obrazovka, na ktorej sa zobrazuje iba hláška „Nesprávne zadaný vstup“.

3 Nástroje na tvorbu konceptu

V tejto kapitole práce sme sa tiež zaoberali existujúcimi možnosťami na tvorbu systému. V predchádzajúcej časti boli teoreticky rozpracované požiadavky na systém spolu s návrhmi na grafický dizajn. Táto časť bude venovaná možnostiam, ktoré môžeme použiť na tvorbu systému, a na základe charakteristických vlastností nástrojov na tvorbu budú vybrané konkrétne programovacie jazyky, pomocou ktorých sa bude realizovať navrhnutý distribuovaný systém pre riadenie prednášky.

Tabuľka 2 - Prehľad nástrojov na tvorbu systému

Skupina nástrojov	Typ nástroja	Nástroje na tvorbu aplikácie
Programovací jazyk pre tvorbu webovej stránky	značkovací jazyk	HTML a CSS
	skriptovací jazyk	Javascript
	skriptovací jazyk	PHP
Systém riadenia bázy dát	SQL	Oracle, MySQL
	NoSQL	ArangoDB, Cassandra, MongoDB, DynamoDB
Tvorba internetovej aplikácie	programovací jazyk	Java
	programovací jazyk	C#
	programovací jazyk	C++
	programovací jazyk	Python
Prehľad výsledkov testov		Microsoft Forms

3.1 Prehľad možných programovacích jazykov pre tvorbu webové stránky

V tejto časti sa nachádzajú základné jazyky pre tvorbu front-endu webovej aplikácie. Opísané sú ich základné vlastnosti a princíp používania. Tieto nástroje budú v systéme slúžiť na vytvorenie časti systému, ktorý sa zobrazuje používateľovi.

Obrázok 22 - Tvorba webovej stránky (Fiverr, 2020)

HTML a CSS

Jedným z najpoužívanejších značkovacích jazykov je HyperText Markup Language (HTML). Tento jazyk sa používa na vytvorenie jednoduchých webových stránok. Pre jeho používanie je potrebné aby programátor poznal už definované tagy, ktorými je možné vytvárať a formátovať dokumenty na samotnej webovej stránke. Takýto dokument ďalej môže obsahovať text, hypertextové odkazy na iné stránky, obrázky. Tento jazyk nás ale obmedzuje vo vytváraní výpočtov. HTML jazyk vytvára iba vonkajšiu formu stránky, no pre dosiahnutie interaktivity je potrebné použiť iný jazyk (napr. JavaScript) a doplniť kód o skripty, ktoré vyvolavajú akciu, po kliknutí na objekt stránky. (Hogan, 2011) (Holzschlag, 2006)

Výhodou HTML stránky je jej jednoduchosť a vysoká dostupnosť učenia sa tohto jazyka aj pre začiatočníkov. Najznámejším online „učiteľom“ sa stala stránka W3C. Na tejto stránke sa nachádzajú ukážky použitia spolu s krátkymi možnosťami vytvorenia si vlastného kódu. Avšak prichádzajú nové metódy, ktorými je tento jazyk potlačovaný. (Hogan, 2011), (Holzschlag, 2006)

CSS skratka vznikla od pojmu Cascading Style Sheet a je častokrát spojovaná s HTML. Dôvodom je, že sa často používajú spolu. CSS môžeme považovať ako zápis HTML dokumentu. Sám o sebe využitie nemá. No v spojení s HTML už áno, pretože dodáva celému dokumentu štýlovanie a formátovanie stránky. Pod pojmom formátovanie môžeme mať na mysli vlastnosti textu, zobrazujúce sa objekty, farby a vizuálne umiestnenie hypertextových elementov. CSS môže byť napísaný priamo v kóde stránky ale častokrát sa dáva do priečinka s kódom stránky a v HTML sa napája iba v kóde. Tento spôsob je prehľadnejší a ak je potrebné zmeniť iba vizuál, nie je potrebné zasahovať do kódu stránky ale len do CSS časti a vytvoriť požadovanú zmenu. Po uložení a znovu načítaní stránky sa zmenia všetky prepísané časti (napr. zmena veľkosti názvu, zmena textu, zmena farby tlačidiel). (Mandík & Vrba, 2007), (Hogan, 2011), (Schafer, 2009)

JavaScript

Patrí medzi jeden zo základných jazykov, ktorými je možná tvorba webových aplikácií. Veľký úspech tento jazyk priniesol hlavne vo vytváraní dynamických časti webu. Pomocou vytvorenia vlastných skriptov je možné vytvoriť pohyblivé a interaktívne časti. Nebýva celá aplikácia zhotovená iba v jazyku JavaScript, prevažne slúži na dotvorenie špecifických prvkov. Častokrát je doplnená HTML kódom, ktorý obsahuje skripty uložené v priečinku. Veľmi jednoduchý je prístup k písaniu, keďže je možné vytvoriť si akýkoľvek skript už v jednoduchom textovom editore aký je aj Notepad. Pre lepšie grafické znázornenie sa častokrát na jeho písanie používajú programy, ktoré sú určené na písanie HTML alebo CSS kódu. V takomto programe sa programátor lepšie vyzná v kóde, keďže štýlovanie textu je farebné alebo odsadenie textov je znázornené v súvislých odsekoch. Asi najviac integrovaným prostredím pre tento jazyk je Visual Studio alebo aj Notepad++. (Suehring, 2008)

PHP

Hypertext Preprocessor (PHP) je skriptovací jazyk, pracujúci v dokumente napísanom v HTML, a dodáva mu tak možnosť generovania obsahu. Návrh tohto jazyka je vytvorený tak, aby pracoval na rovnakej oblasti a akcie je možné napísať pomocou krátkeho kódu. Najlepším príkladom je pripojenie na databázu, ktoré je možné zapísať pomocou troch riadkov. Odozva PHP jazyka na webovú aplikáciu je veľmi rýchla a efektívna, čo zlepšuje priepustnosť webového servera. (Castagnetto & kolektív, 2007)

Výhodami PHP sú jednoduchosť pre triviálne projekty, pre nekomerčné používanie je jazyk možné použiť zadarmo, hardvérové nároky na používanie sú minimálne. (Mandík & Vrba, 2007)

3.2 Prehľad možných Systémov riadenia bázy dát systémov

Databázové systémy nám slúžia ako úložiská dát, do ktorých môžeme dáta vkladať ale zároveň z nich aj ďalej čerpať. Keďže tento typ skladovania informácií je v dnešnej dobe používaný vzniklo viacero možností a typov databáz, ktoré možno použiť. Niektoré sú prístupné ako free verzie, za niektoré treba platiť. No vďaka rozmanitosti sa ale používateľ vie rozhodnúť čo si vyberie na základe toho aké požiadavky na databázu má. Základné rozdelenie databázových systémov je podľa veľkosti ukladania dát na Relačné databázy (inak nazývané SQL databázy) a NoSQL databázy. (Pokorný & Valenta, 2013)

Obrázok 23 - Databázové systémy (WNX.com, 2020)

SQL databázy

Tento spôsob ukladania sa zakladá na tvorbe tabuliek, ktoré sa spájajú do schém. Táto štruktúra však musí byť vopred určená a zároveň musia byť aj jasne určené vzťahy medzi jednotlivými tabuľkami, čo je najčastejšie zabezpečené pomocou primárnych a cudzích kľúčov. Tabuľky nazývame entitami, ktoré sú logicky vytvorené tak, aby každá z tabuliek bola nositeľom istej vlastnosti, ktorou vieme definovať celú skupinu dát uloženú v tabuľke. Tento logický súvis nie je vytvorený náhodne. Ďalej ho vieme využiť na prácu s dátami. Jednou z najčastejších funkcií je filtrovanie dát. Pri veľkom množstve dát sa na základe logického výberu vieme dostať k požadovaným dátam napríklad pomocou selektovania. (Pokorný & Valenta, 2013)

Ako je spomínané v odstavci vyššie po naplnení databázy vieme s dátami ďalej pracovať. Pre ľahšiu a prehľadnejšiu prácu s dátami boli vytvorené systémy na tvorbu príkazov. Medzi najznámejšími poskytovateľmi sú (MySQL, 2020):

- Oracle – bol založený v roku 1977 J. Ellisom. Systém pracuje ako multiplatformový systém a poskytuje používateľovi aj pokročilú prácu s dátami. Práve pre jeho komplexnosť a spoľahlivosť je dodnes obľúbený aj vo viacerých veľkých firmách. Oracle je podporovateľom PL/SQL jazyka, takže je možné implementovať customizované procedúry, a tak si prispôsobiť databázu pre vlastné účely. (Oracle, 2020)
- MySQL – bol založený vo švédsku a to firmou MySQL AB. Dnes patrí pod Oracle, pretože sa MySQL stala dcérskou spoločnosťou Oracle Corporation. Na rozdiel od Oracle je táto platforma bezplatná a voľne stiahnuteľná pre súkromné účely. Prístup k dátam je pomocou SQL jazyka. Jej inštalácia je kompatibilná s operačnými systémami Linux a MS Windows. Aktuálne je MySQL vylepšená natoľko, že bola vylepšená pre používanie triggerov a pohľadov (view). (MySQL, 2020)

NoSQL

Nakoľko prístup k dátam pomocou SQL príkazov nebol vždy efektívny, bolo potrebné vynájsť iný spôsob a tak došlo k vzniku NoSQL databáz. Tento pojem bol použitý Carlom Strozzihm. NoSQL sa stali populárnymi aj vďaka tomu, že s nimi začali pracovať veľké spoločnosti ako Amazon, LinkedIN a Google. Tieto firmy začali využívať tak obrovské štruktúry dát, že klasické relačné databázy, ktoré boli doteraz definované nestačili. NoSQL s ich vytvorením priniesli so sebou výhody akými sú flexibilný dátový model, efektívnosť čítania v dátovom modely a tiež ekonomické výhody v rámci ukladania na pamäť. (Secretariat, 2020) (Pokorný & Valenta, 2013) Modelovanie týchto databáz sa zvyčajne vytvára 4 spôsobmi: (I. Holubova, 2015)

- Grafové databázy – ukladá dáta do pohľadov, ktoré sú zvyčajne ukladané ako grafické rozhrania pospájané väzbami. Tie reprezentujú vzťahy medzi entitami. Vyhodnocovanie týchto dát sa najčastejšie robí grafovými úlohami. Jedným z najznámejších poskytovateľov tohto typu databáz je ArangoDB (I. Holubova, 2015)
- Stĺpcové databázy – sú tvorené tabuľkou, do ktorých je možné vkladať ľubovoľné dáta ako nové stĺpce tabuľky. Pridávanie riadkov nie je potrebné. Medzi tento typ databáz patrí projekt od spoločnosti Facebook, ktorá sa volá Cassandra (DataStax, 2020)
- Dokumentové databázy – pracuje s takzvanými dokumentami, rôznych druhov, so štruktúrovanou formou ako je JSON alebo XML, a tým vzniká možnosť jednoduchého vyhľadávania dát podľa obsahu . Zástupcom dokumentových NoSQL databáz je MongoDB (I. Holubova, 2015)
- Databáza kľúč-hodnota –vznikla pre objekty, ktoré sú navzájom prepojené unikátnymi kľúčmi. Spoločnosť Amazon prišla s najznámejším projektom, ktorý využíva databázy typu kľúč- hodnota a tým je DynamoDB databáza. (G. Decandia, 2007)

3.3 Prehľad nástrojov na tvorbu internetovej aplikácie

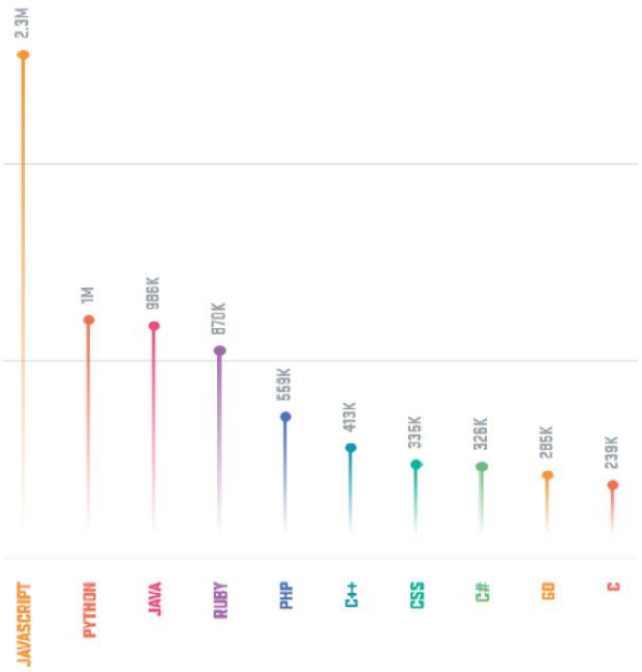
Pri tvorbe aplikácií, či sa jedná o webové aplikácie, či počítačové hry, pre ich vytvorenie je potrebné vopred si určiť programovací jazyk, v ktorom sa bude daný softvér využívať. Programovací jazyk môžeme lepšie opísať ako jazyk, ktorým hovorí počítač. Obsahuje špeciálne znaky a algoritmy, na základe ktorých počítač vykonáva určité úlohy. Tieto jazyky sa pravidelne obnovujú respektíve vyvíjajú a vylepšujú, preto aj počítače sú

v dnešnej dobe schopné vykonávať skoro všetky úlohy, ktoré človek navrhne. Programátor je človek, ktorý takýto kód vytvára a spolu s kódom sú vytvárané aj aplikácie. Podľa typu aplikácie, boli vytvorené rôzne programovacie jazyky, ktoré je možné používať. (Guzan, 2015)

Základné typy programovacích jazykov sú delené na: (Guzan, 2015)

- Procedurálne jazyky
    - štruktúrované
    - objektovo orientované
- Neprocedurálne jazyky
    - funkcionálne
    - Logické

Pre stručný prehľad bolo vybraných niekoľko programovacích jazykov spolu s ich stručným popisom pre lepšiu orientáciu.



Obrázok 24 - Najpopulárnejšie programovacia jazyky (House of Bots, 2020)

Java

Patrí medzi najpoužívanejší programovací jazyk pre podnikové webové aplikácie. Dôvodom je hlavne stabilnosť tohto jazyka. Jeho široké využitie sa vzťahuje aj na vývoj mobilných aplikácií pre Android, pretože tento samotný operačný systém je založený na Java. Veľkou výhodou využívania jazyka je nezávislosť od platformy, čo programátorom

poskytuje možnosť písania kódu z akéhokoľvek zariadenia, či miesta. Aj keď je Java veľmi populárny a flexibilný jazyk, jeho použitie je väčšinou na tvorbu back-endu aplikácií a front-end aplikácií je viazaný na Java no písaný prevažne v iných jazykoch ako napríklad HTML alebo PHP. (House of Bots, 2020)

C#

Je jeden z najjednoduchších objektovo-orientovaných programovacích jazykov. Jeho syntax sa odvíja od jazyka C a zároveň je podobný s jazykmi C++ alebo čiastočne aj s Java. Princíp programovania sa zakladá na orientáciu programovania komponentov, čo znamená, že návrh softvéru je napojený na komponenty, ktoré sú vo forme integrovaných balíčkov funkcií. Program často pracuje s objektmi, čo sa nazýva jednotný systém typov. Znamená to, že všetky typy dát (aj primitívne typu int, sa nachádzajú v type object. Takýto typ má výhodu, že môže byť definovaný aj používateľom, čo programátora neobmedzuje pri programovaní logicky náročnejších aplikácii. C Sharp vznikol najme pre vývoj služieb spoločnosti Microsoft. (Microsoft, 2020)

C++

Tento jazyk je jedným z najznámejších programovacích jazykov. Dôvodom je, že je veľmi podobný programovaciemu jazyku C, ktorý bol jedným z prvých jazykov na programovanie. C++ je dynamickým jazykom, pretože kontrola kódu sa vykonáva ešte pred jeho spustením. Hlavný rozdiel medzi C a C++ jazykom je, že C++ jazyk je vyšším jazykom, čiže je vylepšený o funkcie ako je podpora zapuzdrovania, polymorfizmus a dedičnosť. (House of Bots, 2020)

Python

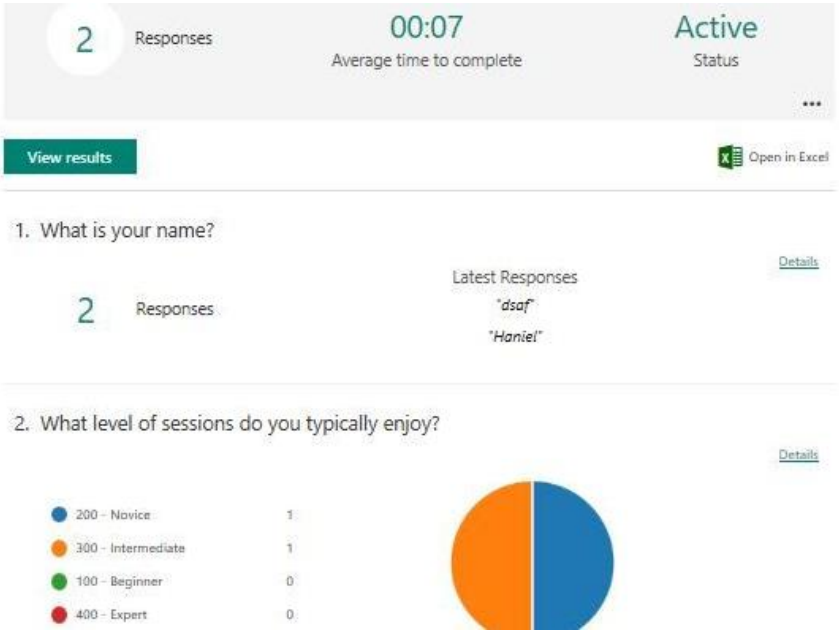
Programovací jazyk Python sa dočkal veľkého úspechu najmä v posledných rokoch. Jeho popularita vzrástla viac ako ktorémukoľvek inému programovaciemu jazyku. Ide o ľahko naučiteľný jazyk, a jeho funkcie sa dajú programovať v krátkych kódoch, čo zabezpečuje prehľadnejší zdrojový kód. Phyton sa zaraďuje medzi najpoužívanejší jazyk v oblasti dátovej vedy. Jeho využitie sa častokrát spája s oblasťou umelej inteligencie, Data Sience a oblasťou strojového učenia. Jedná sa o viacúčelový jazyk, ktorý spoločnosti zaoberajúce sa vývojom softvérov používajú ako data science a zároveň sa často využíva pre vývoj webových aplikácií. (House of Bots, 2020)

3.4 Prehľad foriem zobrazovania výsledkov testov

Poslednou kapitolou v rámci kapitoly nástroje na tvorbu, je venovaný niektorým formám poskytovania výsledkov, a ich analyzovanie. Po vyhodnotení dát je potrebné aby boli dáta zobrazené používateľovi, lebo iba tak s výsledkami môže ďalej pracovať. Týchto foriem zobrazovania výsledkov je viac no potrebné je vybrať si takú formu, ktorá je pre analytika dát najprijateľnejšia a najzrozumiteľnejšia.

Microsoft Forms

Služba Microsoft Forms bola vytvorená pre online vyplnenie formulárov. Ide o jednoduchú formu dotazníkov pre kohokoľvek. Služba je jednoducho použiteľná. Pomocou obrazovky si sami viete napísať otázky a priradiť im odpovede a pomocou pozvánky je možné tento dotazník rozposlať ďalej. Dotazníky bývajú často anonymizované, čo sa častokrát využíva pri tvorbe recenzií, či vyhodnocovaní odpovedí na bakalárske a diplomové práce. Na základe zodpovedaných otázok si tvorca dotazníka môže pozrieť stručný prehľad odpovedí, počet respondentov, priemerný čas na zodpovedanie celého testu a zoznam odpovedí na otázky. Prehľad obsahuje aj koláčový graf zostrojený na základe vyhodnotenia údajov. Celý prehľad je dostupný na stránke Microsoft Forms. Výhodou je taktiež zálohovanie dát. Túto zálohu si používateľ vie stiahnuť pomocou ďalšej office služby a tou je Excel. (Microsoft, 2020)



Obrázok 25 – Microsoft Forms (Microsoft, 2020)

Tabuľkové zobrazenie

Jedno zo základných zobrazení je pomocou tabuliek. Systém Microsoft Forms poskytuje možnosť stiahnuť si tabuľku vo formáte *.xls v ktorej sa nachádzajú výsledky testu. Túto formu môžeme považovať ako backup alebo zálohu, lebo prístup k stiahnutému súboru máme kedykoľvek. Uloženú verziu v počítači, môžeme využívať kedykoľvek a nemusíme pritom pristupovať k systému Micrsoft forms.

Zobrazenie pomocou grafu

Druhým užitočným spôsobom vytvorenia výsledkov je zobrazovanie pomocou grafu. Výsledky zobrazované týmto spôsobom, sa môžu odlišovať v type grafu. Na našom modelovom príklade, je zobrazený koláčový graf, ktorý hovorí percentuálnej úspešnosti zodpovedaných otázok. Typy grafu sú zvyčajne zvolené na základe toho, aké sú spracovávané typy dát. Najčastejšími používanými grafmi sú:

- spojnicový, ktorý sa používa na zobrazenie trendov v čase
- stĺpcový, používaný na porovnávanie hodnôt rozdelených v kategóriách
- koláčový graf, percentuálne porovnanie časti celku (súčet častí v percentách sa musí rovnať 100% inak je graf nesprávny)

3.5 Metódy tvorby informačného systému

Kapitola je zameraná na oboznámenie sa s existujúcimi metodikami pre tvorbu softvéru. Cieľom je aby sme sa pochopili zmysel a postup tvorby systému na základe už vytvorených postupov. Výber správnej metodiky sa bude ďalej vyvíjať na základe jednotlivých vlastností metodik.

Výber správnej metodiky pre vývoj systémov je závislý najmä od zložitosti softvéru. Pri väčších systémoch je potrebné aby sa na projekte podieľalo viacero tímov – vývojári, analytici, testeri a koordinátori. Takéto veľké projekty je potrebné vytvárať postupne a systematicky aby sa predišlo prípadným nedorozumeniam v rámci komunikácie. Naopak pri menších projektoch sa softvér môže dynamicky meniť. Dôvodom je potreba menších kapacít ľudí, a jednoduchšia implementácia, pretože častokrát zadávateľ kladie podmienky počas vytvárania systému. Metodiky sú častokrát vytvárané tak aby vyhovovali vlastnostiam projektu. Hlavnými dôvodmi, prečo je dobré využívať existujúce metodiky informačných systémov sú: (Thomas, 2007)

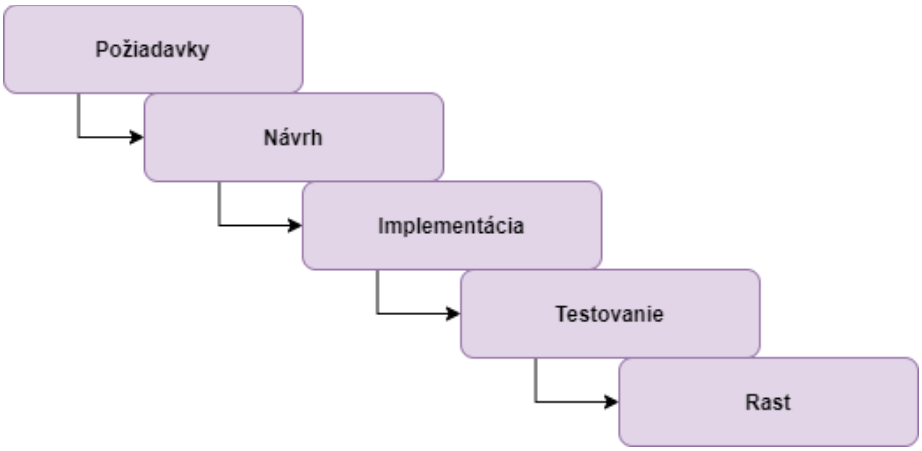
- Nastaviť si postupné fázy projektu, ktorými sa bude vykonávať vývoj, aby boli splnené ciele projektu
- Efektívna tvorba analýzy softvérového procesu
- Jasná definícia vstupov a výstupov procesu
- Metodiky častokrát odpovedajú na otázku, či je potrebná konzultácia projektu s expertmi alebo zákazníkmi.
- Najdôležitejším dôvodom je: metodika odpovedá na otázku ako bude prebiehať koordinácia jednotlivých činností systému (činnosťami sa myslí analýza, vývoj, testovanie, nasadenie systému do prevádzky).

V práci budú vysvetlené 4 metodiky: (Thomas, 2007)

- Vodopádový prístup k tvorbe softvéru
- Metodika Rational unified proces (metodika RUP)
- Metodika Unified process (metodika UP)
- Agilný prístup tvorby softvéru

Vodopádový prístup k tvorbe sotfvéru

Princíp založený na modely vodopád je vytvorenie takej postupnosti fáz, ktoré sa cyklicky neopakujú. Je to idealizovaný stav, ktorý je v praxi ťažké dosiahnuť. Model pozostáva z 5 etáp: požiadavky, návrh, implementácia, testovanie a údržba (či rast). Tieto fázy tvorby systému sú realizované samostatne, no musia byť vykonávané v takom poradí, v akom boli vymenované. Dôvodom striktnej postupnosti je nadväznosť etáp. Až po ukončení jednej etapy, je možné začať ďalšiu. Avšak povolené je prekrývanie fáz. (Buchalcevová, 2005)



Obrázok 26 - Model vodopád [autor]

Výhodami používania tohto modelu je: (Buchalcevová, 2005)

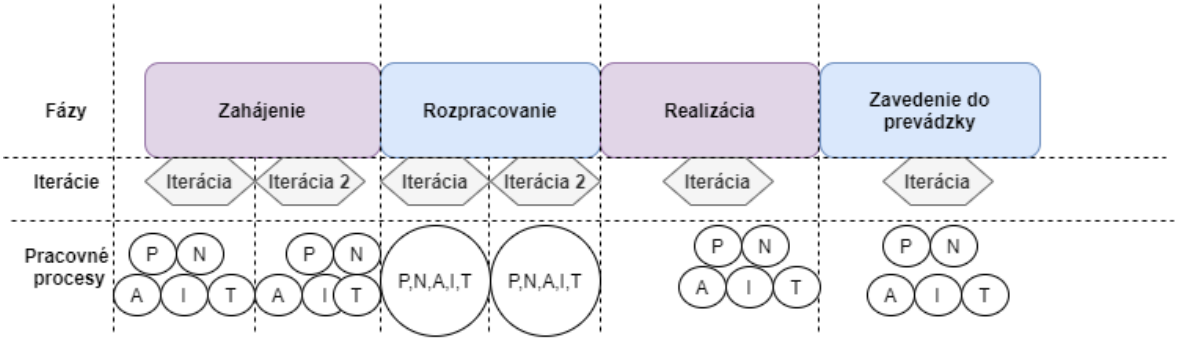
- Možnosť jednoduchého riadenia tvorby systému, pretože postupnosť krokov je jasne určená.
- Pri nemeniacich sa požiadavkách na systém je výsledný produkt taký, že spĺňa všetky požiadavky od používateľa.

Medzi nevýhody používania modelu zaraďujeme: (Buchalcevová, 2005)

- Častokrát používateľ nevie dopredu určiť všetky požiadavky, čo vytvára problém, pretože model nie je prispôsobený často meniacim sa požiadavkám.
- Požadované zmeny sú náročné na čas. Nové zmeny musia opätovne prejsť etapami modelu.
- Zákazník nedostáva beta verzie, kde by si mohol vyskúšať systém, poprípade ho zmeniť. Zákazník vidí finálnu verziu. Pri prípadných zmenách sa zvyšujú náklady na čas, peniaze.

Metodika Unified process (metodika UP)

Táto metodika je založená na štandardoch Softvérového inžinierstva. Vytvorená je tak, aby mohla byť použiteľná, pre akýkoľvek projekt, bez ohľadu na jeho rozsah. Najpoužívanejšia forma prezentácie procesov je pomocou unifikovaného modelovacieho jazyka (UML). Princíp fungovania metodiky UP je založený na iteráciách a pracovných procesoch. Iterácie predstavujú samostatné malé projekty a každý z iterácií ma samostatné činnosti. Vzhľadom na to, že iterácie na seba nenadväzujú, môžu prebiehať paralelne. (Buchalcevová, 2005)



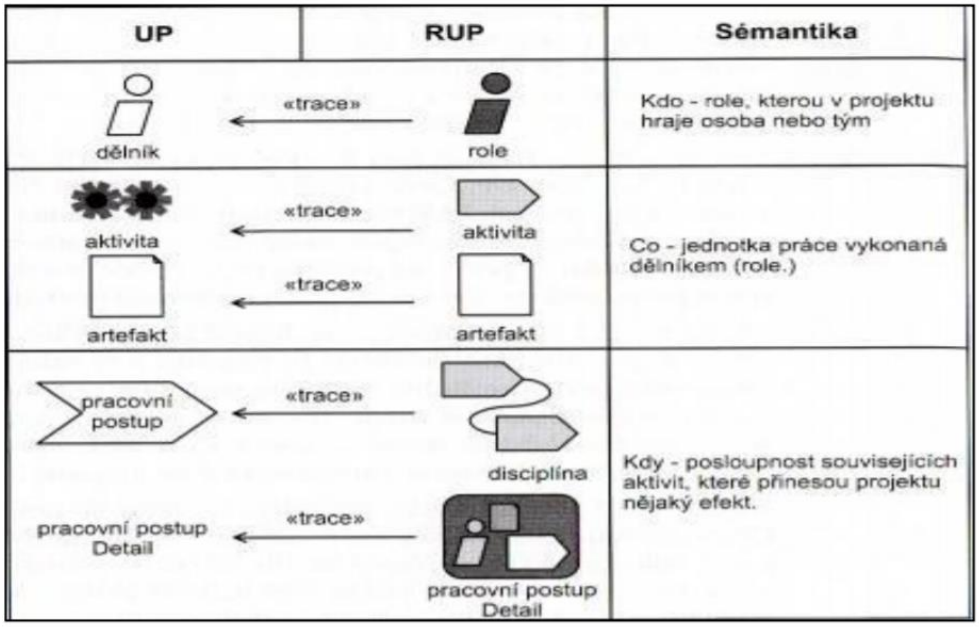
Obrázok 27 - Metodika UP [autor]

Medzi fázy patria: Zahájenie, Rozpracovanie, Realizácia, Zavedenie do prevádzky. Pričom každá z iterácií obsahuje samostatne päť pracovných procesov: požiadavky, analýza, návrh, implementácia, testovanie. V rámci metodiky UP je potrebné do tvorby softvéru zahrnúť riadenie rizík, riadenie požiadaviek. (Buchalcevová, 2005)

Pri modelovaní procesov sa využíva už spomínaný UML diagram. Diagramy môžu byť reprezentované diagramom tried, diagramom komponentov, diagramom nasadenia alebo profilovania a diagramami správania a iterácií. Hlavnou nevýhodou používania tejto metodiky je technická zručnosť vytvárania a čítania UML diagramov. (Vrana, 2008)

Metodika Rational unified proces (metodika RUP)

Metodika RUP je nadstavba metodiky UP. Ide o podobný princíp využívania UML jazyka. Rozdiely sa vyskytujú iba vo významnosti pre niektoré značky. (J. Arlow, 2007)



Obrázok 28 - Rozdiel medzi metodikami RUP a UP (J. Arlow, 2007)

V metodike RUP sa využívajú už doteraz overené postupy ktorými sú: (Buchalcevová, 2005)

- Vývoj prebieha na základe iterácií. Po každom šprinte sa do systému pridávajú nové vlastnosti, pričom dochádza k postupnému zjednocovaniu.
- Vytvára sa správa požiadaviek od používateľa. Ide o ich doplňovanie a hodnotenie.
- Využíva sa architektúra komponentov a vytvárajú sa čiastočne riešenia.
- Vytvára sa vizualizácia modelu systému.

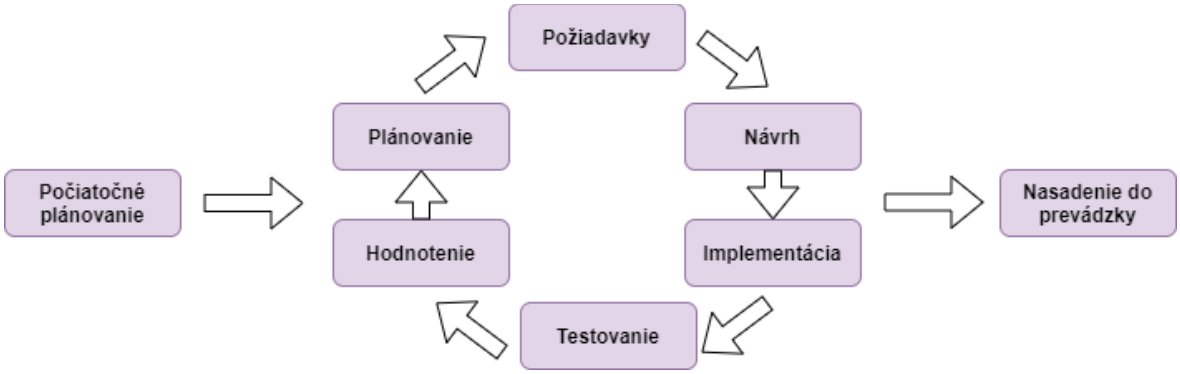
Na vývoji systému sa podieľajú 4 elementy: pracovníci, činnosti, medziprodukty a pracovné procesy. (Buchalcevová, 2005)

Tabuľka 3 - Základné elementy metodiky RUP

Element	Otázka	Význam
Pracovníci	Kto?	Ide o určenie zodpovednosti za konkrétne aktivity. Pracovníkom nemusí byť konkrétna osoba, môže to byť rola, ktorú daná osoba zastupuje.
Činnosti	Ako?	Reprezentuje cieľ, ktorý má byť splnený. Jedna činnosť je častokrát pripisovaná iba jednej osobe.
Medziprodukty	Čo?	Ide o výsledok iterácie projektu. Týmto projektom môžu byť napríklad: návrh modelu, špecifikácia, zdrojový kód alebo výsledná aplikácia.
Pracovné procesy	Kedy?	Ide o postupnosť činností aby bol správne dosiahnutý cieľ.

Agilný prístup tvorby softvéru

Vznik agilného prístupu vznikol na základe odklonenia sa od klasického prístupu, ktorý je náročný z pohľadu administratívy ale najmä je náročný na čas. Klasické metodiky sú častokrát zložité a striktne určené. Agilný prístup sa zakladá na myšlienke rýchlo prispôsobujúceho sa systému na akúkoľvek zmenu, pričom čas na vytvorenie systému zostáva približne rovnaký. (Z. Šochová, 2014)



Obrázok 29 - Metodika agilný prístup [autor]

Základnými princípmi pre agilnú metodiku sú: (Larman, 2004), (Z. Šochová, 2014)

- Tvorba krátkych iterácií, pričom sa ako prvé vytvárajú najdôležitejšie funkcionality systému. V ďalších iteráciách sa postupne dopĺňajú požiadavky zamestnávateľa.
- Stála komunikácia je kľúčová pre vytvorenie systému. Zákazník a celý tím podieľajúci sa na tvorbe systému je v neustálom kontakte a tak je možné vykonať zmeny počas tvorby systému alebo implementovať nové požiadavky od zákazníka.
- Automatizované testy sú vítané. Vzhľadom na možnosť dynamicky sa meniaceho systému je výhodné využívať automatizované testy. Ide o rýchly a účinný test, ktorý je potrebné vykonávať po každej iterácií. Ide tak o časovo nenáročné a presné odhalenie možných nedostatkov systémov.

Agilná metóda testovania sa zakladá na troch pojmoch, „sprint“, „release“, „backlog“. (Larman, 2004)

Sprint je iterácia, ktorá rozdeľuje cyklus tvorby systému na menšie časové úseky. Častokrát ide o krátke časové úseky, ktoré obsahujú zoznam činností, ktoré je potrebné v danom čase splniť. Trvanie jednotlivých šprintov nemusí byť rovnaké. (Larman, 2004)

Release je samotná nasadenie produktu do prostredia po každom šprinte. (Larman, 2004)

Backlog predstavuje sadu činností, ktoré je potrebné splniť. Je možné ich meniť aj počas životného cyklu softvéru. Tento celok činností sa postupne rozdeľuje do jednotlivých šprintov podľa priority. (Larman, 2004)

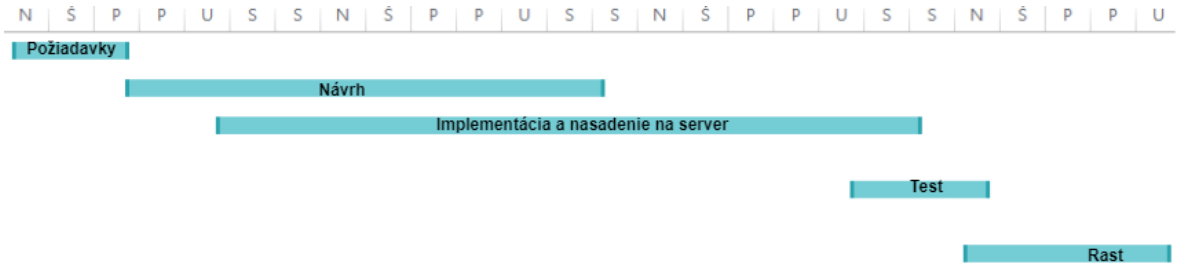
4 Výsledky práce

Kapitola je zameraná na realizáciu systému a jej testovanie. V jej prvej časti je popísaná tvorba systému, ktorá bola uskutočnená na základe vybraných nástrojov na tvorbu a pomocou nich vznikol funkčný systém s požiadavkami, ktoré boli na neho kladené v cieľoch práce. Po nasadení aplikácie je potrebné jej otestovanie a analýza výsledkov. Posledná podkapitola tejto časti je venovaná testovaniu a vyhodnoteniu výsledkov spolu s reálnymi údajmi zozbieranými od študentov.

4.1 Algoritmus tvorby informačného systému

Na začiatku vytvárania nášho systému sme si stanovili algoritmus jeho tvorby. Pre projekt sme si stanovili základné vlastnosti, aby sme si neskôr mohli vybrať metodiku, ktorou sa vyvíja realizácia systému. Projekt je stredne veľkého rozsahu. Na začiatku projektu si zadávateľ určí požiadavky na systém a na základe nich sa vytvorí požadovaný produkt. V požiadavkách od používateľa, nie sú poskytnuté technické parametre na systém.

Vzhľadom na popis projektu sme použili model vodopád. Dôvodom výberu tejto metodiky, je jasne určený postup tvorby informačného systému. Priebeh tvorby systému sme zaznamenali pomocou Gantovho diagramu spolu so slovným vysvetlením pod obrázkom.



Obrázok 30 - Gantov diagram vývoja systému [autor]

V prvom kroku sme si určili požiadavky na systém. Na základe zadanej témy sme spolu s vedúcim práce určili požiadavky od používateľov. Následne sa vytvoril návrh v podobe diagramov, ktorý vysvetľuje fungovanie systému v teoretickej rovine. Beta verzia systému v tejto metodike nebola potrebná. Už počas vytvárania návrhu sa začalo s implementáciou systému, kde sa začali vytvárať základné funkcionality, ktorými je zobrazovanie obrazoviek pre jednotlivé entity. Po odsúhlasení kompletného návrhu boli zostrojené všetky funkcionality v súlade s návrhom. Počas etapy implementácia je možné systém testovať so študentmi, aplikácia bola nasadená na server a prebehlo aj prvotné

testovanie. Poslednou a dôležitou fázou je rast aplikácie. Aj po hodnotení vytvoreného systému nepovažujeme projekt za ukončený. Pre pokračovanie a vytvorenie lepšieho systému sme pripravili fázu rast, v ktorej sú popísané možné vylepšenia v nasledujúcom životnom cykle systému.

Vzhľadom na výber vodopádového prístupu je potrebné vykonať všetky etapy v poradí v akom sme ich opísali. Po ukončení poslednej etapy je vytvorené zhodnotenie splnenia cieľov pre daný životný cyklus.

4.2 Popis riešenia

Táto časť práce je venovaná vytvoreniu systému. V kapitole ciele sme sa venovali teoretickej analýze návrhu systému, definovali sme si postupne jednotlivé časti a ich funkcionality, ktoré boli navrhnuté tak, aby spĺňali všetky požiadavky používateľov. V tejto časti je popis fyzického prepojenia a vytvorenia komponentov aplikácie aby, pracovala tak ako je popísané v kapitole 2. Zároveň vysvetlíme, prečo sme vybrali konkrétne nástroje na tvorbu z kapitoly 3 pre náš softvér.

Logika aplikácie bola vytvorená v jazyku Java. Pri vylepšení aplikácie je potrebné zvážiť, či je nie je potrebné zmeniť nejaký z nástrojov, poprípade či netreba zakúpiť vyššiu verziu softvérov. Použité vývojové prostredie pre tvorbu aplikácie bolo IntelliJ IDEA. Ďalším komponentom ktorý bolo potrebné vybrať bol server, vhodný na implementáciu aplikácie pracujúcej na distribuovanom spôsobe spracovania. Princíp fungovania programu je bližšie opísaný v nasledujúcich častiach tejto kapitoly. Databáza systému je pripájaná v rámci nástroja JetBrains.

Pred samotným programovaním aplikácie bola vytvorená nasledovná logika fungovania softvéru. Systém bude simulovať distribuovanú prácu s dátami pomocou dvoch serverov. Oba softvéry budú rozdelené na základe požiadaviek, serveru 1 a serveru 2. Takéto rozdelenie sa neskôr prejaví v UI používateľa, na základe HTTP PATH. Používateľ na základe toho, na aký server sa chce pripojiť klikne na daný odkaz. Na základe používateľa pripojeného na server sa odvíjajú aj ostatné funkcionality systému a tie nie je možné preskočiť ani vynechať. To zabezpečuje, že každý z používateľov má obmedzené práva na zapisovanie aj čítanie údajov.

Program hlavnej triedy AppClass

Program je spúšťaný pomocou hlavnej triedy AppClass, zjednodušene povedané v tejto triede sa nachádza základná logika pre správne fungovanie programu. Ide o hlavné príkazy *pippo.POST* a *pippo.GET*, ktoré na základe HTTP PATH zobrazia požadovanú stránku a vykonajú danú akciu. Tieto základné operácie fungujú na princípe zobrazenia webových stránok alebo v našom prípade konkrétnych URL adries. Na prvom mieste v kóde je *pippo.GET*, ktorý sa pripája na ftl template a zobrazuje jeho obsah. *Pippo.POST* popisuje akcie danej obrazovky. Ako príklad je uvedená časť kódu - prihlásenie študenta. (Pippo, 2020)

```
pippo.GET("/s1/login", routeContext -> {
    routeContext.render("s1_login");
});

pippo.POST("/s1/test", routeContext -> {
    // submit student
    String meno = routeContext.getParameter("formularMeno").toString();
    String priezvisko =
routeContext.getParameter("formularPriezvisko").toString();
    String session = UUID.randomUUID().toString();

    Student student = new Student(0, meno, priezvisko, session);
    questionsDao.addSessionForStudent(student);

    // redirect the student to test page
    HashMap<String, Object> params = new HashMap<>();
    params.put("session", student.getSession());
    routeContext.redirect("/s1/test", params);
});
```

V uvedenom príklade je vidieť, že prvá časť kódu zobrazuje UI pre používateľa pomocou *„/s1/login“*, a okrem zobrazenia nie je implementovaná žiadna funkcionalita. Po prihlásení študenta sa pomocou funkcie *POST* načítavajú základné údaje o študentovi. Zároveň prebieha prideľovanie unikátneho session ID pomocou funkcie. Následne je študent presmerovaný na test. Podobná logika na princípe *POST* a *GET* je vytvorená aj pre nasledujúce obrazovky:

Tabuľka 4 - Rozdelenie systému na dva servery

Server 1 - študent	Server 2 - učiteľ
/s1	/s2
/s1/login	/s2/login
/s1/test	/s2/results
/s1/results	

Poradie krokov serverov nie je možné zmeniť ani vynechať, pretože pri každom kroku sa vykoná určitá akcia a iba takto postupne vyplnený test sa považuje za správny. Server 1 a 2 pracujú nezavisle od seba. Pristupovanie na oba je možné kedykoľvek a odkiaľkoľvek, kde je možnosť prístupu na internet.

Program pripojenia aplikácie na databázu

Vytvorenie systému, ktorý pracuje na báze distribuovaného spracovania si vyžadovalo pripojiť softvér na databázu. Konektivita databázy a programu je zabezpečená pomocou triedy ConnectionManager. V tomto čiastkovom kóde, zobrazenom nižšie je potrebné zadať vlastné údaje k prístupu do databázy, ktorými sú URL adresa databázy, „user name“, a heslo. Po pripojení, je program schopný načítať údaje z databázy ale ich aj zapisovať. Na databázu sa môže pozerať aj programátor, ktorý si môže overiť správnosť naprogramovanej časti. Pripojenie je však úspešné až po tom, čo databázový server povolí programátorovu adresu (pripojením IP adresy). Takýmto spôsobom sa ošetrí prístup k databáze komukoľvek a odkiaľkoľvek. (Oracle, 2020)

```
public Connection createConnection() {
   try {
      Class.forName( "com.mysql.jdbc.Driver" );
   } catch (ClassNotFoundException e) {
      logger.error( "Where is your MySQL JDBC Driver?" );
   }
   logger.info( "MySQL JDBC Driver Registered!" );
   Connection connection = null;
   try {
      connection = DriverManager.getConnection(
"jdbc:mysql://wh39.farma.gigaserver.cz:3306/demcak_net_zuzana?serverTimezone=UTC"
, "31675_zuzana", "***********" );
   } catch (SQLException e) {
      logger.error( "Connection Failed! Check output console", e );
   }
   return connection;
}
```

Program spracovania výsledkov v systéme

Poslednou dôležitou funkciou pre objasnenie programu je spracovanie získaných výsledkov od študentov. Všetky získané odpovede sú zapísané v konkrétnej databázovej tabuľke. Údaje sú následne spracované programom do takej zrozumiteľnej formy, ktorá je zobrazená učiteľovi (pre server 2). Vyhodnotenie výsledkov závisí od subjektívneho uváženia na základe ktorého sa napíše SQL príkaz. V našom prípade sme vytvorili triedu ServiceDaoImpl.java.

```
@Override
public List<QuestionView> detailedQuestions() {
    List<QuestionView> questions = new ArrayList<>();
    Connection connection = new ConnectionManager().createConnection();
    if (connection != null) {
        try {
            String sql = "select B.id, B.content, A.answer, COUNT(*) as responses
                         from answers A " +
                         "JOIN questions B ON A.question = B.id " +
                         "GROUP BY A.question, A.answer " +
                         "ORDER BY A.question, responses DESC";
            PreparedStatement ps = connection.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                int id = rs.getInt("id");
                String content = rs.getString("content");
                int answer = rs.getInt("answer");
                int responses = rs.getInt("responses");

                QuestionView questionView = new QuestionView(id, content, answer,
responses);
                questions.add(questionView);
            }
        } catch (SQLException ex) {
            logger.error("Nepodarilo sa ziskat questions z databazky", ex);
        }
    }
    return questions;
}
```

Časť kódu zachytáva anotácia Override, ktorá vytvára prehľad pre učiteľov. Začiatok kódu vytvára nové pole, do ktorého budú vkladané spracované údaje. Vo bloku TRY-CATCH sa nachádza dôležitá časť - SQL dopyt. Práve ten spracováva výsledky, podľa toho, čo chceme zobrazovať (viď. Zvýraznená časť kód). Na základe tohto sa vytvorí tabuľka v prehľadnej forme, ktorý slúži učiteľovi v rámci výučby.

Po vytvorení všetkých funkcionalít je potrebné aby bola aplikácia nasadená webový server. Pre naše riešenie sme sa rozhodli použiť server od poskytovateľa OpenShift, ktorý pracuje s distribuovaným spracovaním[3]. Server je poskytovaný firmou Red Hat. Pre správne nasadenie projektu bolo potrebné upraviť projekt tak, aby obsahoval podporou softvéru

[3] OpenShift online – Red Hat – Online : https://www.openshift.com/products/online/, navštívené 5.1.2020

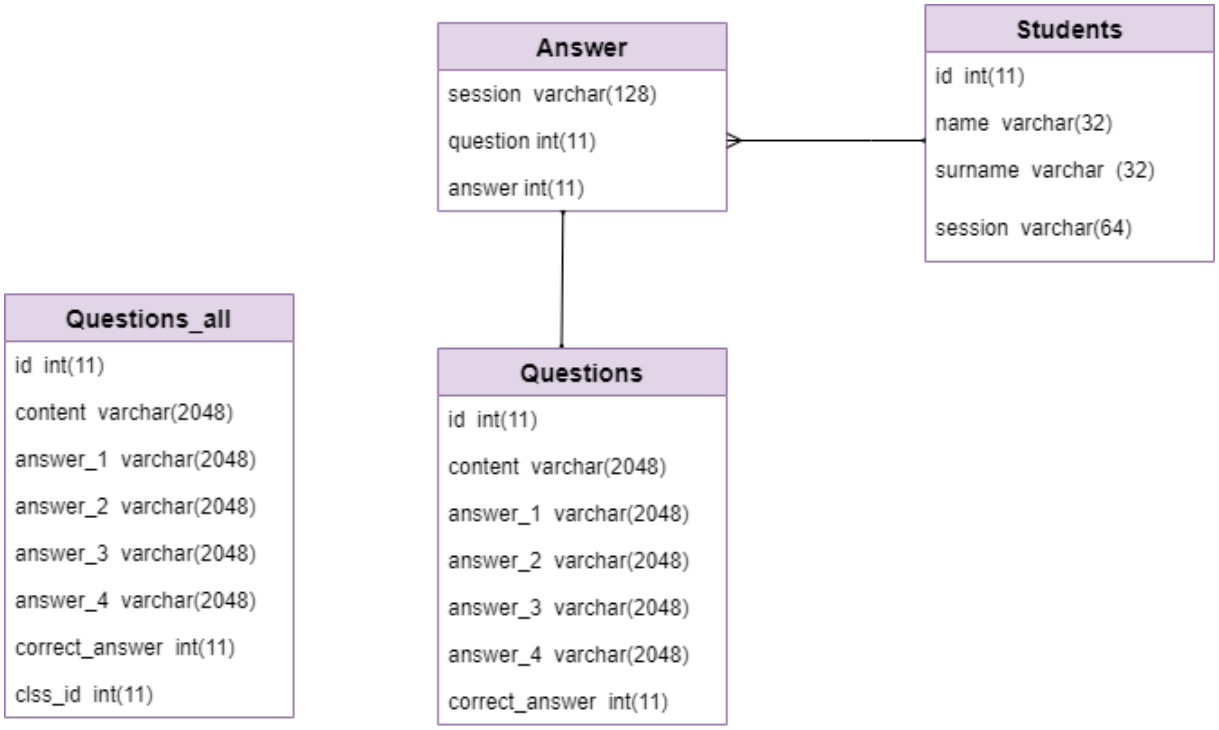
Maven. Takáto úprava bola nutná pre využívanie vopred pripravených knižníc. (Maven, 2020).

4.3 Dátové modelovanie

Definovanie úložiska pre dáta je jednou z kľúčových podmienok pre väčšinu aplikácií. Pre náš systém bol integrovaný nasledujúci databázový model na zapisovanie a čítanie dát.

Vzhľadom na rozšírené používanie jazyku SQL bola použitá MySQL databáza pripojená pomocou triedy ConnectionManager v zdrojovom kóde. Prístup na túto databázu je možný iba po splnení podmienky, že databázový server povolil prístup pre danú IP adresu na ktorej systém pracuje. To zabezpečí korektnosť dát, pretože cudzie zdroje sa na databázu nepripoja, takže do nej nevpíšu dáta ani s nimi nebudú manipulovať.

Databázová schéma systému je v podstate jednoduchá, no dostačujúca na prácu systému. Znázornená je na nasledujúcom obrázku.



Obrázok 31 - Databázová schéma [autor]

Jednotlivé tabuľky boli na platforme vytvorené pomocou SQL príkazu CREATE TABLE s jednotlivými atribútmi, a takými hodnotami, aby boli postačujúcej veľkosti pre zápis dlhších polí.

Tabuľka **Students** obsahuje menovitý zoznam študentov, ktorý absolvovali test. Zároveň sa tu nachádza priradený primárny kľúč ID, ktorý sa agreguje s postupným pribúdaním riadkov v tabuľke. Hodnota ID stúpa každým študentom o jednu jednotku. Atribúty meno a priezvisko sú hlavnými kľúčmi pre identifikovanie prihláseného študenta. Ďalej sa v tabuľke nachádza priradené session_ID pre každého prihláseného používateľa. Tento atribút zabezpečuje anonymitu študenta pri vypĺňaní. Takýto spôsob označovania je často využívaný v systémoch, aby sa klient prihlasoval pod jedinečným číslom a nie konkrétnym menom a priezviskom, ako jedna z foriem šifrovania. Zároveň sa takáto možnosť využívania unikátnych čísel používa z legislatívnych dôvodov ako je napr. GDPR. Tabuľka Students je prepojená s tabuľkou Answer, do ktorej sa vkladá už spomínané sessionID, s jednotlivými otázkami a odpoveďami.

Tabuľka **Questions_ALL** je kľúčovým úložiskom, v ktorom sa nachádzajú všetky testové otázky pre daný rok alebo semester. Táto tabuľka obsahuje rovnako ako aj ostatné jeden primárny atribút, ktorým je číslo otázky. Ďalej sa v nej nachádzajú atribúty ako obsah otázky (content) a 4 odpovede. Každý z atribútov bol vytvorený ako textové pole s dĺžkou 2^{11} znakov. Nastavenie veľkosti polí sa odvíjalo od úvahy, že otázka aj odpovede môžu byť aj rozsiahlejšieho charakteru. Dôležitým atribútom tabuľky je aj „correct_answer“, ktorá označuje správnu odpoveď. Zapísaná je v podobe čísla od 1-4. Toto označenie je totožné s číslom v názve jednotlivých atribútov Answer_?, ktoré vytvárajú možné odpovede testu. Posledným atribútom je class_id. Ten rozdeľuje jednotlivé otázky podľa toho, na ktorú hodinu sú určené, čo zabezpečuje tak jednoduché označenie otázky pre daný okruh tém.

Tabuľka **Questions** je označovaná ako pracovná tabuľka, ktorej obsah sa zobrazuje pre študentov vo forme testu. Atribúty su totožné s tabuľkou Questions_All s rozdielom, že neobsahuje atribút class_id. Obsah tejto tabuľky sa mení a na základe okruhu otázok, ktorý chceme zobrazovať (definovaný class_id). Otázky sa do tabuľky Questions importujú na základe selektu:

```
INSERT INTO questions (id, content, answer_1, answer_2, answer_3, answer_4, correct_answer)

SELECT id, content, answer_1, answer_2, answer_3, answer_4, correct_answer

FROM questions_all

WHERE class_id = 1;
```

Podmienka správneho naplnenia tabuľky je, že Question je prázdna tabuľka a select obsahuje správne číslo class_id. Preto bola táto tabuľka definovaná ako pracovná, pretože jej obsah je premenlivý. Každá z otázok sa zobrazuje v databáze ako jeden záznam, čo reprezentuje jeden riadok v databáze.

Tabuľka **Answer** zobrazuje tabuľku výsledkov, ktorá sa následne používa pre vytvorenie prehľadov učiteľovi. Tieto dáta reprezentujú odpovede testov vyplnené študentmi. Riadky reprezentujú všetky zodpovedané otázky, správne aj nesprávne. Preto je možné s tabuľkou naďalej pracovať a vytvárať rôzne pohľady.

Celá databázová schéma tvorí základňu pre využívanie systému, pretože obsahuje hlavnú časť, ktorou sú dáta.

4.4 Nasadenie na server

Nasadenie aplikácie na server je jedna z najdôležitejších krokov k tomu, aby sme vytvorili plne funkčnú aplikáciu. Z vedomostí ktoré sme nadobudli pri vytváraní systému môžeme povedať, že táto časť bola jedna z najzložitejších. Problémové bolo samotné doladenie systému aby ho bolo možné spustiť.

Obrázok 32 - Prostredie Red Hat (Hat, 2020)

Prvým krokom k správnemu spusteniu aplikácie je výber vhodného prostredia pre nasadenie. Vzhľadom na naše zadanie bolo vybrané prostredie OpenShift. Výber bol ohodnotený ako správny na základe vedomosti o veľkosti priestoru servera, jazykov ktoré server podporuje a podobne.

Po prihlásení do služby Red Hat, bolo potrebné naimportovať náš projekt, ktorý bol doteraz vytvorený iba na lokálnej úrovni. Dôvodom vytvorenia globálnej úrovne nášho systému je potreba vytvoriť distribuovaný systém. Požadujeme aby so systémom mohli pracovať viacerí študenti naraz pomocou internetu bez akejkoľvek znalosti o jazyku Java alebo bez znalosti teórie o distribuovaných systémoch. Práve spoločnosť Red Hat zabezpečuje všetky podmienky vhodné pre použitie. Import kódu na server musí prebiehať

pomocou inej webovej služby pre podporu vývoja aplikácií. Pre naše potreby bola využitá služba Git Hub. (Brown, 2020)

Obrázok 33 - Webová služba GitHub (GitHub, 2020)

Obrázok 34 - Aplikácia GitBash (GitBash, 2020)

Zdrojový kód aplikácie bolo potrebné nahrať do nami vytvoreného repozitára. Vykonanie inportu súboru do GitHubu je možné cez GitBash (aplikácia, ktorá pomocou príkazov v Commands line vykoná akciu), alebo pomocou JetBrain cez príkaz „Push“. Po nahratí aplikácie sa dostaneme k hypertextovému odkazu, ktorý určí odkiaľ server zavolá zdrojový kód, ktorý bude slúžiť ako aplikácia. V našom prípade je kód dostupný na webovej lokalite: (GitHub, 2020)

https://github.com/Zuzana-Demcakova/fhidemdiplomovka.git

Následne je potrebné nahrať projekt na server OpenShift. Po zadaní všetkých potrebných údajov je potrebné spustiť „Build“, ktorý nám skontroluje kód a vykoná všetky potrebné kompilácie nahraného súboru. Ak táto verzia skončí s chybou, je potrebné opraviť chyby a spustiť „Build“ znova. Práve tento krok je náročný pre validáciu dát a spustenie projektu.

Po bezchybnom ukončení server zobrazí úspešnú hlášku v konzole a zobrazí aplikáciu v stave „aktívna“. V detaile je aplikácia doplnená o ďalšie stavy a tými sú „Running“ alebo „Completed“. Detail obsahuje aj adresu na ktorej sa aplikácia nachádza vo webovom prehliadači.

http://fhidemdiplomovka-fhidemdiplomovka.apps.ca-central-1.starter.openshift-online.com/

Výstupná adresa je pomerne dlhá, pretože nie je naviazaná na žiadnu platenú doménu, ktorú by sme si vybrali. Preto bolo potrebné využiť ďalšiu službu, ktorá nám vytvorí kratšiu URL adresu. Tento krok skrátenia webovej stránky bol vytvorený iba na základe potreby lepšieho prístupu k aplikácie pre študentov. Namiesto prepisovania dlhej linky, je jednoduchšie ak bude prístupová adresa aplikácii kratšia a študenti sa na aplikáciu

pripoja rýchlejšie. Skrátenie adresy je možné vytvoriť napríklad pomocou webovej stránky Bitly.

Obrázok 35 - Logo spoločnosti Bitly (Bitly, 2020)

Táto stránka slúži ako platforma na správu prepojení. Hlavnou funkciou je už spomínané skrátenie odkazov. Výhodou tejto služby je, že je možné používať ju bezplatne. Jej jednoduché používanie častokrát využívajú malé firmy. Bezplatná verzia zabezpečuje iba funkcie skrátenia linky a analyzovanie prepojenia. Platené balíčky služby Bitly poskytujú rozšírenejšie funkcie. Služba je uznávaná aj veľkými medzinárodnými spoločnosťami, ktoré ju využívajú ako sú napríklad ESPN (najväčší poskytovateľ športových kanálov), odevná spoločnosť NIKE, Amazon alebo časopis The New York Times. (Bitly, 2020) (Bitly, 2020)

Pre porovnanie ako veľmi je výhodné využiť spomínanú službu bola vytvorená tabuľka s URL linkami pred a po použití webovej stránky Bitly. V rámci tabuľky je vidieť, že skrátenie linky bolo účinnejšie až o 75%, spolu s http prefixom.

Tabuľka 5 - Skrátenie URL linky

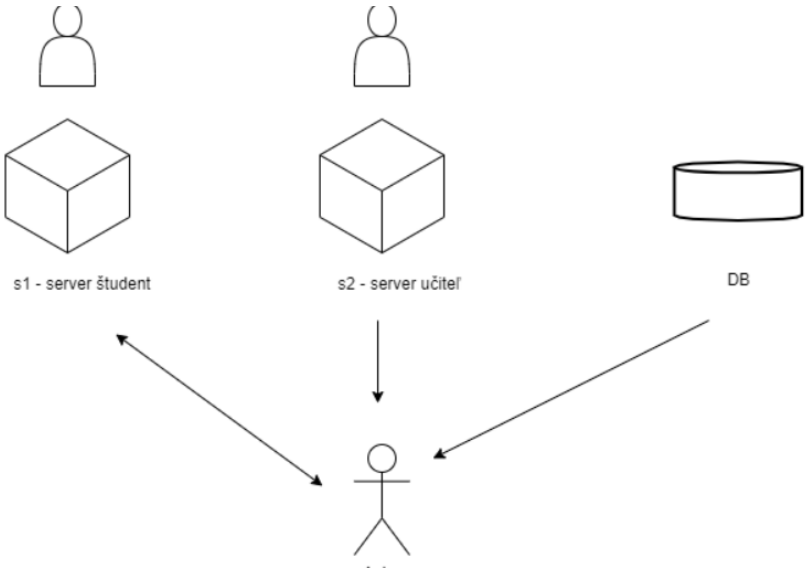
	URL link	**Počet znakov**
OpenShift	http://fhidemdiplomovka-fhidemdiplomovka.apps.ca-central-1.starter.openshift-online.com/	88
Bitly	https://bit.ly/2vU8m5V	22

4.5 Zabezpečenie distribuovaného systému

Distribuované spracovanie je v oblastiach programovania a informatiky pomerne široký pojem. Práve z toho dôvodu je potrebné uviesť dôkaz, že náš systém spĺňa jeden z cieľov zadania, ktorým je vytvorenie distribuovaného systému. V rámci dokazovania môžeme systému pripisovať názov distribuovaný na základe dvoch vlastností.

Prvou vlastnosťou distribuovanosti systému je vytvorenie širiteľného systému ako celku. Aj napriek tomu, že jeho komponenty sa nenachádzajú na jednom servery, systém pracuje ako celok a ukladá všetky dáta správne bez ohľadu na lokáciu komponentov. Druhá

vlastnosť sa bude zaoberať logikou serveru 1, ktorý bol ešte samostatne vytvorený ako distribuovaný. Bližší opis systému sa nachádza pod obrázkami nižšie.



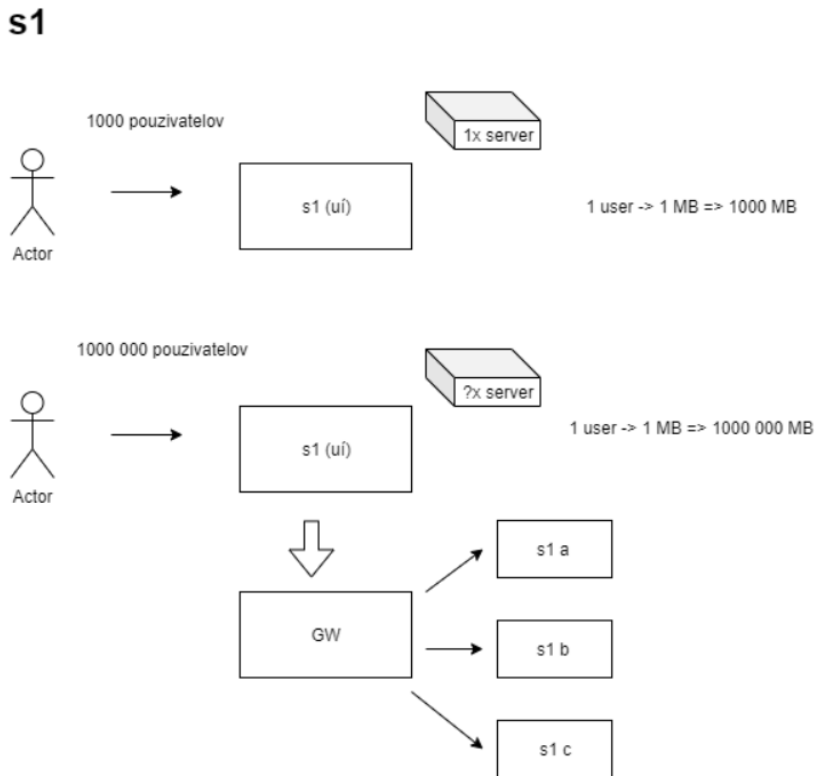
Obrázok 36 - Logika distribuovaného systému ako celok [autor]

Na obrázku je zobrazený systém a jeho používateľ, čo môže byť akákoľvek osoba, ktorá pracuje so systémom - je reprezentovaný panáčikom, ktorý sa pripája na server odkiaľkoľvek. Šípky znamenajú, že používateľovi sú zo systému poskytované údaje ale zároveň môže do systému zadávať dáta aj on a to vyplnením testu. Podobnú schému by mal aj učiteľ, s výnimkou, že pri ňom systém poskytuje spracované údaje po vyplnení testu študentmi. Tieto dáta sú vždy správne, aj keď sú títo dvaja používatelia (študent aj učiteľ) na rôznych miestach. Práve táto vlastnosť zabezpečuje distribuovaný systém.

Systém ako celok je rozdelený na 3 časti, ktoré sa nachádzajú na rôznych miestach, no navzájom medzi sebou komunikujú. Server 1 sa nachádza na Americkom serveri, Server 2 sa nachádza na školských serveroch, aby sa zabezpečila bezpečnosť dát a aby prístup k dátam mohli mať iba kompetentné osoby. Posledným komponentom je databáza. Na jej umiestnenie sme využili súkromnú databázu so serverom v Českej republike. Z tohto popisu je vidno, že časti sú rozdistribuované po celom svete, no ich prepojenie je zabezpečené na základe IP adries a tak je aplikácia schopná pracovať ako celok.

Ďalším vhodným príkladom distribuovaného systému je nezávisle používanie serverov. V tomto prípade nie je potrebné aby boli používané oba servery naraz. Študent si test môže vyplniť v čase A a učiteľ sa k výsledkom dostane v čase B. Napriek odlišným časom majú obaja používatelia správne údaje, nehľadiac na fakt, že ak študent vyplnil test,

učiteľ nemusel mať prístup k internetu a naopak. Systém si aj napriek týmto stavom dokáže zachovať konzistentné dáta a ďalej ich distribuovať.



Obrázok 37 - Distribuovaný systém server 1 [autor]

Druhou už spomínanou vlastnosťou distribuovaného systému je využitie služby OpenShift, ktorého servery fungujú na distribuovanom spôsobe spracovania, ktoré vyplýva z obrázka. Táto časť práce sa zaoberá iba serverom 1, na ktorý sa pripájajú študenti. Pre jednoduché vysvetlenie sme si pripravili modelovú situáciu. Predpokladajme, že 1 používateľ na používanie systému potrebuje 1 MB priestoru na serveri, na ktorom aplikácie pracuje. Systém práve používa 1000 používateľov. Dokopy tak využívajú 1000 MB pamäte servera. Takýto malý počet dát zvládne bez problémov spracovať jeden počítač. V prípade ak sa náš počet používateľov navýši na 1000 000 (napr. systém nepoužívajú všetci študenti školy pre dotazník spokojnosti alebo občania republiky na predvolebný prieskum), tak server potrebuje naraz spracovať 1000 000 MB, čo už žiaden super počítač v dnešnej dobe nie je schopný zvládnuť. Preto servery spoločnosti OpenShift tieto úlohy distribuujú na viacero serverov a tak sú dáta spracované v rovnakom čase. Princíp spočíva v tom, že každá požiadavka od používateľa vchádza do takzvanej brány, kde sa úloha pošle na server s voľnou kapacitou na spracovanie. Ak je plne zaneprázdnený úloha sa pošle na druhý

počítač. Výsledné dáta sú aj posielané na našu databázu v správnom tvare, aj napriek tomu, že neboli vyhodnotené na jednom zariadení.

4.6 Výsledky experimentu

V tejto kapitole sa budeme zaoberať testovaním systému a analýze výsledkov získaných z experimentu. Pre analýzu bolo potrebné zozbierať testovacie dáta. Pre experiment boli vybraní študenti Ekonomickej univerzity 2. ročníka, v rámci predmetu „Pokročilé využívanie databáz“. Pre študentov boli zostavené otázky z prednášanej témy a tak sme získali dáta, ktorých výsledky sú zaznamenané v nasledujúcich kapitolách.

4.6.1 Príprava experimentu

Testovacia fáza obsahuje dve sady otázok pre študentov, ktoré budú použité počas prednášky. Vybrané témy, na ktoré sme sa zamerali vznikli na základe poskytnutých materiálov od vedúceho prednášok. Kniha, ktorá slúžila ako podklad pre otázky sa nazýva „Pokročilé využitie databáz pre ekonomické školy„ a autormi sú Jaroslav Kultan a Peter Schmidt. Zameranie otázok sa týkalo prevažne kapitol 1-3 zo spomínaného zdroja. Následne sa naplnila databázová tabuľka Questions_all pripravenými otázkami.

V príprave poskytujeme aj zoznam otázok a vyznačených správnych odpovedí:

Tabuľka 6 – Zoznam testových otázok

Otázka	***Odpovede***
1.Vytvorenie databázy na základe multidimenzionálnej analýzy nesleduje podnik z pohľadu	• Časového hľadiska • Priestorového hľadiska • Z pohľadu materiálu • Z pohľadu výrobkov • Veľkosti podniku
2. Čo tvorí štruktúru Dátového skladu typu Hviezda	○ 2 tabuľky faktov ○ 2 tabuľky dimenzií ○ 1 tabuľka faktov a 2 tabuľky dimenzií ○ 2 tabuľky faktov a 1 dimenzia ○ 1 tabuľka faktov a 1 dimenzia
3. Na čo slúži ETL	• Tvorbu reportov z databázy • Čistenie citlivých dát v databáze • Transfomáciu dát do databázy • Transformáciu dát do tabuliek • Čistenie citlivých dát v dimenziách

4.Analytický nástroj pre analýzu OLAP nemusí spĺňa jedno z nasledujúcich pravidiel	○ Dostupnosť ○ Architektúru klient/server ○ Intuitívnu manipuláciu s dátami ○ Flexibilný reporting ○ Obmedzený počet dimenzie a úroven agregácii
5. Priamy data mining je	• Klasifikácia • Zhlukovanie • Algoritmus rozhodovací strom • Asociačné pravidlá
6.Triggery definujú udalosti, ktoré sa majú vykonať v prípade situácie ktorá je definovaná nad databázou	○ Áno ○ Nie
7. Procedural Language extension of Structured Query Language slúži na	• Definovanie vstupných dát pre naplnenie tabuliek • Definovanie a spúšťanie programovej jednotky • Sledovanie zmien dát v DB • Sledovanie konštrukcie vetvenia
8.Čo je to procedúra	○ Nástroj objektového programovania ○ Nástroj grafického programovania ○ Nástroj logického programovania ○ Nástroj štruktúrovaného programovania -
9.Parametre procedúr sú	• Add, return • In, out, in out • Get, set
10.Pre jednoduchý cyklus platí	○ Príkaz exit hovorí o počte opakovaní ○ Príkaz loop označuje postačujúcu podmienku pre spustenie cyklu ○ Slučka sa vykonáva aspoň raz ○ Pre PLSQL poznáme iba jednoduchý cyklus WHILE

Otázka č.1 až 5 sa nachádzajú v prvom kole testovania a preto sú zaradené do sekcie označenej classID = 1.Druhá časť otázok je považovaná za druhú sadu a označená je rovnakým spôsobom ako už spomínaná prvá polovica, no s číslom 2 (classID = 2). Na základe toho, ktorá zo skupín je aktuálne testovaná, tá bude vložená do pracovnej tabuľky **Questions**. Testovanie oboch častí sa zakladá na rovnakej logike, no s iným obsahom dát.

Podstatné je nájsť výsledok pre otázku, či je vedenie prednášok adaptívnym spôsobom efektívnejšie ako klasický spôsob výučby.

Testovacia vzorka obsahuje 22 študentov druhého ročníka pre predmet Pokročilé využitie databáz. Študenti boli na začiatku oboznámený s podmienkami testovania a následne im bola poskytnutá URL adresa, kde je možné pristúpiť k testu. Študenti mali na vyplnenie testu približne 10 minút.

Výsledkom experimentu chceme dosiahnuť jednoduchšie a efektívnejšie vedenie prednášok. V časti príprava sme sa venovali vytvoreniu systému, pre vyhodnocovanie otázok zodpovedaných študentmi.

Pre zber informácií na analýzu výsledkov bol zostrojený testovací scenár, ktorý hovorí o tom, ako bude experiment prebiehať. Po jeho vykonaní sa budú zozbierané výsledky analyzovať a budú vyhodnotené na základe očakávaných výsledkov pre splnenie cieľa a pre splnenie Testovacieho scenára.

Testovací scenár je pripravený v nasledovnej forme:

- V prvom kroku študenti dostanú študijný materiál (vopred), ktorý slúži ako príprava na prednášku (skriptá).
- Na začiatku prednášky sa študenti oboznámia s testovacím systémom
- Študenti dostanú prístupovú URL adresu k systému a vyplnia test
- Systém vyhodnotí otázky a vytvorí prehľady
- Učiteľ dostane v reálnom čase prehľady, na základe ktorých môže byť vytvorený plán prednášky, resp. väčší dôraz bude kladený na otázky, ktoré boli zodpovedané najhoršie.
- Študenti vyplnia dotazník spokojnosti, vytvorený pomocou Microsoft Forms, ktorý bude slúžiť na ohodnotenie systému a jeho funkčnosti z hľadiska študenta ako používateľa.

Očakávaný výsledok testovania:

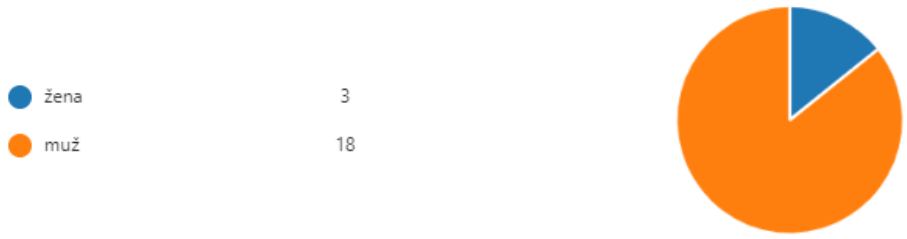
- Systém bude plne funkčný a zaujímavý pre študentov ako nová forma výučby.
- Študenti viac porozumejú vysvetľovanej téme, pretože výklad je zameraný na veci, ktoré sú nejasné.
- Dotazník spokojnosti poskytne spätnú odozvu od študentov a tak môže systém rásť a zlepšovať sa.

4.6.2 Celkové porovnanie

Po vykonaní testu sme dostali nasledovné výsledky popísané v tejto časti. Výsledky sú definované grafmi a ich slovným opisom.

Testovacia vzorka študentov

Testu sa zúčastnilo 21 študentov, z toho približne 85 percent boli muži a zvyšok prieskumu pozostáva z troch žien. Výsledky o študentoch sme získali z dotazníka Micrsoft forms. Na testovaní systému sa však zúčastnil ešte jeden študent, ktorý odmietol vyplniť dotazník o spokojnosti, no do systému je zapísaný a jeho odpovede budeme považovať za relevantné vo výsledkoch otázok.

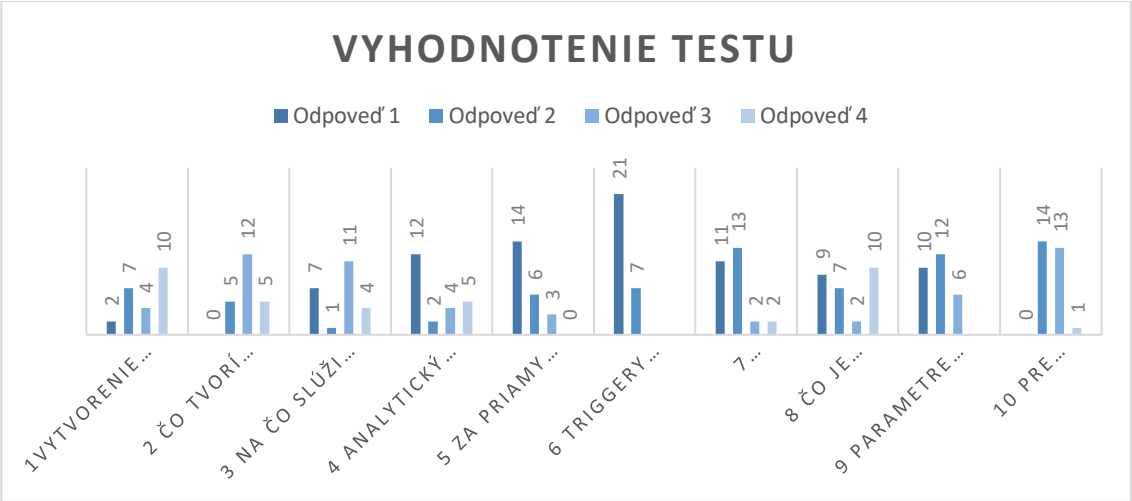


Obrázok 38 - Vzorka študentov [autor]

Taktiež je potrebné spomenúť, že jeden študent odpovedal iba na prvé kolo otázok, čo je reprezentované otázkami 1 – 5. Preto môžu mať nasledujúce prehľady rozdielny počet odpovedí. A jeden zo študentov odovzdal druhú časť testu 8-krát. Tento fakt ovplyvňuje počet odpovedí pre otázky 6 - 10.

Porovnávanie odpovedí – celý test

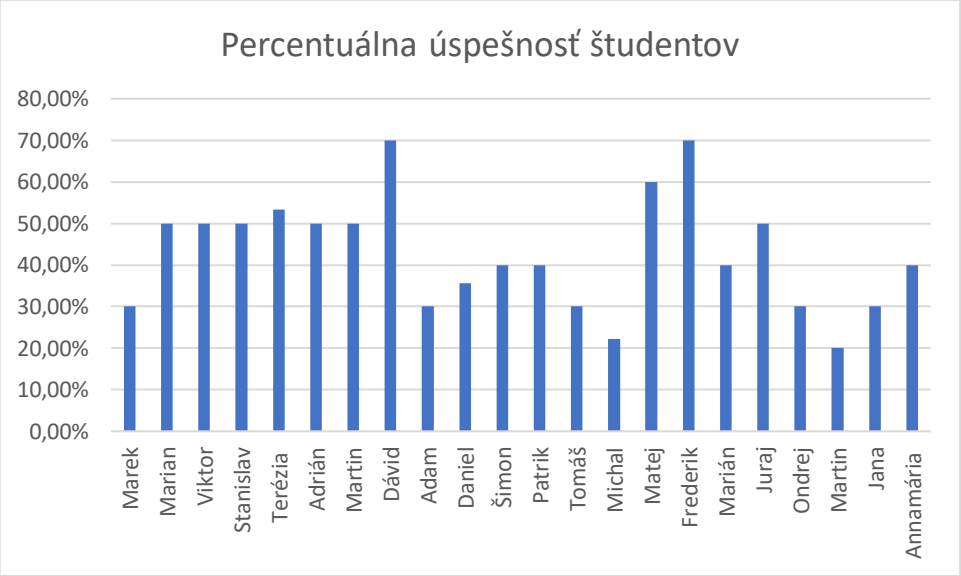
Nasledujúci graf zobrazuje porovnanie odpovedí pre jednotlivé otázky.



Obrázok 39 - Vyhodnotenie jednotlivých otázok [autor]

Aktuálny graf sa nezameriava na správnosť jednotlivých odpovedí, ide iba o grafické znázornenie výsledkov. Z grafy je vidieť že tri otázky obsahujú jednu možnosť, označenú hodnotou 0. Ostatné otázky majú v každej odpovedi svojho zástupcu a odpovede sú rozdielne.

Pre celkové vyhodnotenie testu sme pripravili aj úspešnosť študentov. Aj na základe vedomosti, že študenti vypĺňajú test pod session ID, je systém pripravený na vylepšenie a vieme ho využiť aj na finálne hodnotenie študentov. Aktuálne sme nasimulovali situáciu pomocou SQL selectu, kde sme študentom priradili ich odpovede a na základe toho bol vytvorený graf úspešnosti študentov.



Obrázok 40 - Graf úspešnosti študentov [autor]

Žiaden zo študentov nevyplnil test so stopercentnou úspešnosťou. Najvyššie hodnotenie dostali 2 študenti, pričom dosiahli 80% správnych odpovedí. Na 7 otázok odpovedali správne. Naopak najhoršie hodnotenie dostal študent s 20% úspešnosti.

Porovnanie odpovedí - systém (vylepšenie adaptívneho riadenia prednášky)

Nasledujúci obrázok reprezentuje výsledky testovania systému. Tieto prehľady slúžia na lepšiu orientáciu učiteľa na silné aj slabé stránky študentov ako celku. Na rozdiel od vyhodnotenia testu z predošlej časti je táto časť zameraná na vytvorenie takého prehľadu, ktorý by vyhodnocoval jednotlivé otázky na základe správne zodpovedaných otázok. Obrázok s názvom „Otázky s najväčším počtom nesprávnych odpovedí“ reprezentuje zoradenie od najhoršie zodpovedaných po najlepšie zodpovedané.

Otazky s najvacsim poctom nespavnych odpovedi

id	otazka	bads
9	Parametre procedúr sú	28
8	Čo je procedúra	18
5	Za priamy dataminig považujeme	17
10	Pre jednoduchý cyklus platí	15
7	Procedural Language extension of Structured Query Language slúži na	15
1	Vytvorenie databázy na základe "multidimenzionálnej" analýzy podnik nesleduje z pohľadu	13
3	Na čo slúži ETL	12
4	Analytický nastroj Olap nemusí spĺňať jednu z následujúcich pravidiel	11
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	10
6	Triggery definujú udalosti, ktoré sa majú vykonať v prípae situácií, ktoré sú definované nad databázov	7

Obrázok 41 - Výstup zo systému [autor]

Obrázok je odrazom systému ktorý sa nachádza na obrazovke serveru 2. Na prvý pohľad je vidieť, že prehľad zaznamenáva iba otázky (spolu s ich číslovaním), bez ohľadu na obsah odpovedí. V prvom prípade nie je potrebné vidieť obsah odpovedí, podstatné je, že na otázku odpovedal študent nesprávne. Práve počet študentov, ktorý označili inú ako správnu odpoveď, zachytáva stĺpec s názvom „bads“. Toto číslo je potrebné zobrazovať, aby si vedúci prednášky vedel vytvoriť prehľad o tom, koľko študentov nepozná správny výsledok pre danú tému.

Na poradí otázok prehľadu záleží. Ako prvá sa zobrazuje otázka s najväčším počtom nesprávnych odpovedí. Táto otázka (respektíve jej obsah) má byť hlavnou časťou prednášky. Vysoký počet nesprávnych otázok indikuje, že študenti potrebujú odborný výklad v rámci danej témy. Dôležitým faktorom je už aj spomínaný počet nesprávnych odpovedí. V nasledujúcej tabuľke sa nachádzajú percentuálne výsledky zodpovedaných otázok.

Tabuľka 7 - Úspešnosť otázok

Číslo otázky	Počet študentov	Počet nesprávnych odpovedí	Neúspešnosť otázky
1	22	13	**59%**
2	22	10	**45%**
3	22	12	**55%**
4	22	11	**50%**
5	22	17	**77%**
6	28	7	**25%**
7	28	15	**54%**
8	28	18	**64%**
9	28	28	**100%**
10	28	15	**54%**

Na základe tabuľky úspešnosti vidíme, že otázka 9 má až 100% nesprávnych odpovedí. Systém tak správne vyhodnotil otázku ako najhoršie zodpovedanú a preto ju zapísal ako prvú. Opačne to funguje rovnako. Posledná otázka, ktorá je zobrazená na učiteľovej obrazovke je otázka o triggeroch a tej prislúcha číslo 6, na ktorú nesprávne zodpovedalo iba 25%, čo predstavuje iba 7 študentov. Na základe tohto vysvetlenia môžeme vidieť, že systém pracuje správne a môže tak napomôcť učiteľovi, pri adaptívnom riadení prednášky.

V rámci vyhodnotenia otázok od najhorších po najlepšie je potrebné analyzovať aj jednotlivé odpovede v rámci odpovede. Aby učiteľ vedel reagovať na rôznorodosť toho ako študenti odpovedali, bol vytvorený nasledujúci prehľad, ktorý zachytáva presný počet klikov na jednotlivé odpovede.

id	otazka	odpoved	pocet
1	Vytvorenie databázy na základe "multidimenzionálnej" analýzy podnik nesleduje z pohľadu	4	10
1	Vytvorenie databázy na základe "multidimenzionálnej" analýzy podnik nesleduje z pohľadu	2	7
1	Vytvorenie databázy na základe "multidimenzionálnej" analýzy podnik nesleduje z pohľadu	3	4
1	Vytvorenie databázy na základe "multidimenzionálnej" analýzy podnik nesleduje z pohľadu	1	2
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	3	12
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	2	5
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	4	5
3	Na čo slúži ETL	3	11
3	Na čo slúži ETL	1	7
3	Na čo slúži ETL	4	4
3	Na čo slúži ETL	2	1
4	Analytický nastroj Olap nemusí spĺňať jednu z následujúcich pravidiel	1	12
4	Analytický nastroj Olap nemusí spĺňať jednu z následujúcich pravidiel	4	5
4	Analytický nastroj Olap nemusí spĺňať jednu z následujúcich pravidiel	3	4
4	Analytický nastroj Olap nemusí spĺňať jednu z následujúcich pravidiel	2	2
5	Za priamy dataminig považujeme	1	14
5	Za priamy dataminig považujeme	2	6
5	Za priamy dataminig považujeme	3	3

Obrázok 42 - Výsledky prvej časti testovania [autor]

Otázky sú v tomto prípade zoradené od prvej až po poslednú s rozdielom, že ku každej otázke je zapísané číslo odpovede a počet študentov, ktorý ju považovali za správnu. Vyhodnotenie každej z otázok je založené na nasledovnej logike. Výsledky pre otázku číslo dva sú zaznamenané v nasledujúcej tabuľke.

Tabuľka 8 -Vyhodnotenie otázky č.2

Číslo otázky	Obsah otázky	Číslo odpovede	Odpoveď	Počet odpovedí	Počet percent
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	3	1 tabuľka faktov a 2 tabuľky dimenzií	12	55%
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	2	2 tabuľky dimenzií	5	22,5%
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	4	2 tabuľky faktov a 1 dimenzia	5	22,5%
2	Čo tvorí štruktúru Dátového skladu typu HVIEZDA	1	2 tabuľky faktov	-	-

Z tejto otázky vidíme, že jej prislúchajú 3 odpovede. Odpoveď „2 tabuľky faktov“ sa v prehľade nenachádza, čo indikuje, že túto možnosť študenti úplne vylúčili. Tento fakt sa považuje za pozitívny, pretože odpoveď nie je správna. Najlepšie ohodnotená odpoveď je pri čísle answer_3, ktorá je správna. To znamená, že až 55% študentov odpovedalo na otázku správne. Avšak toto percento je iba tesne nad polovicou, a preto sledujeme ďalšie odpovede. Tými sú „2 tabuľky dimenzií“ a „2 tabuľky faktov a 1 dimenzia“. Na každú z nich odpovedalo presne 5 študentov. Potrebné je teda vysvetliť študentom danú oblasť učiva.

6	Triggery definujú udalosti, ktoré sa majú vykonať v prípae situácií, ktoré sú definované nad databázov	1	21
6	Triggery definujú udalosti, ktoré sa majú vykonať v prípae situácií, ktoré sú definované nad databázov	2	7
7	Procedural Language extension of Structured Query Language slúži na	2	13
7	Procedural Language extension of Structured Query Language slúži na	1	11
7	Procedural Language extension of Structured Query Language slúži na	4	2
7	Procedural Language extension of Structured Query Language slúži na	3	2
8	Čo je procedúra	4	10
8	Čo je procedúra	1	9
8	Čo je procedúra	2	7
8	Čo je procedúra	3	2
9	Parametre procedúr sú	2	12
9	Parametre procedúr sú	1	10
9	Parametre procedúr sú	3	6
10	Pre jednoduchý cyklus platí	2	14
10	Pre jednoduchý cyklus platí	3	13
10	Pre jednoduchý cyklus platí	4	1

Obrázok 43 - Výsledky druhej časti testovania [autor]

Z druhej časti testovania vyplýva ďalší zaujímavý fakt. Analýza je zameraná na otázku číslo 6. Tá bola vyhodnotená systémom ako najlepšie zodpovedaná.

Tabuľka 9 - Vyhodnotenie otázky číslo 6

Číslo otázky	Obsah otázky	Číslo odpovede	Odpoveď	Počet odpovedí	Počet percent
6	Triggery definujú udalosti, ...	1	áno	21	75%
6	Triggery definujú udalosti, ...	2	nie	7	25%

Tabuľka bližšie definuje počet odpovedí. Na túto otázku boli len 2 možnosti odpovedí (áno, nie), ktoré boli vyhodnotené v pomere 3:1 študentov. Aj napriek tomu, že úspešnosť otázky je až 75 percent, netreba zabúdať, že stále sa nachádza 7 študentov, pre ktorých je obsah otázky neznámi a je potrebné aspoň trochu sa venovať téme triggery.

Celkové výsledky získané zo systému

V tejto časti sa nachádzajú zovšeobecnené postrehy, ktoré sú získané zo systému. Táto časť slúži aj ako celkové zosumarizovanie funkcií aplikácie na základe testovania.

- Ani jedna z otázok nezískala stopercentnú úspešnosť pre správne odpovede. To znamená, že je potrebné sa vyjadriť ku všetkým oblastiam z testu aby študenti pochopili tému prednášky. Výsledky vznikli a základe toho, že systém vyhodnotil jednotlivé odpovede.
- Systém vyučujúcemu zobrazuje otázky s najväčším počtom zlých odpovedí ako prvé, čo slúži na vytvorenie študijného plánu. Dôvodom takéhoto zobrazenia je aby z výsledkov bolo ako prvé vidieť najhoršie zodpovedané otázky aby sa týmto témam venoval dostatočne dlhý čas prednášky. Výsledkom sa stala téma Procedúry.

Vyhodnotenie dotazníka spokojnosti.

Študenti si v prvej fáze testovania vyskúšali prácu so systémom. Následne im bol poskytnutý dotazník spokojnosti. Test pozostával z dotazníka vytvoreného pomocou Microsoft forms. Prvé otázky boli zamerané na základné údaje o študentoch, ich pohlavie a ročník. Ďalšie otázky opisovali systémové funkcionality. Dotazník bol vytvorený tak, že

každú z otázok bolo potrebné ohodnotiť známkou 1-5, kde číslo 1 bolo najhoršie ohodnotenie a číslo 5 je najlepšie hodnotenie.

Študentom bolo položených týchto 5 otázok:

1. ***Tento systém by som rád používal častejšie***
   Vyhodnotenie: otázka bola ohodnotená na známku 3. To znamená, že študenti nijako nepociťujú zmenu či už majú klasickú výučbu alebo používajú systém. Hodnotenie pre celú túto prácu je považované za kladné. Pretože systém má mať väčšiu výpovednú hodnotu pre učiteľa. A fakt, že študentov používanie systému nezaťažuje sa považuje za dobré.
2. ***Myslím si, že systém je jednoduchý na používanie***
   Vyhodnotenie: hodnotenie je 4,19. Hlavnou úlohou pre vytvorenie systému bola myšlienka takého návrhu, ktorý by bol zrozumiteľný pre kohokoľvek, kto doposiaľ aplikáciu nepoužíval. Práve toto hodnotenie potvrdilo, že cieľ bol splnený. Takto vysoká známka to potvrdzuje, ale zároveň, je priestor pre jeho vylepšenie.
3. ***Myslím si, že funkcie v systéme sú dobre integrované***
   Vyhodnotenie: hodnotenie funkcionalít je číslo 4. Pred prvým kontaktom študentov so systémom, boli respondenti v krátkosti oboznámení s funkcionalitami a cieľom systému. V závislosti na rýchlu odozvu a správne vyhodnotenie bol naplnený aj ďalší z cieľov pre splnenie diplomovej práce.
4. ***Myslím si, že systém je zložitý na používanie***
   Vyhodnotenie: hodnotenie zložitosti systému od študentov dostalo číslo 2. Zdanlivo nízke hodnotenie potvrdzuje otázku číslo jedna. Táto otázka bola zaradená na potvrdenie pravdivosti zo strany študentov.
5. ***Návrhy na vylepšenie systému***
   Vyhodnotenie: odpovedalo 8 študentov. Posledná otázka bola v dotazníku nepovinná. Táto otázka slúži na vyhodnotenie systému a jeho zlepšenie. Odpovede študentov budú zaradené v časti diskusia, ktorá je zameraná na vylepšenie systému a jeho rast.

Celkové hodnotenie systému dostalo veľmi dobrý výsledok o ktorom hovoria aj vysoké známky od študentov. Systém pracoval správne nevykazoval žiadne známky preťaženia. Celkové testovanie trvalo približne 25 minút (testovanie prebiehalo v 2 fázach), čo predstavuje iba 27% času prednášky. Zvyšok času prednášky tak učiteľ môže efektívne

využiť na prednášanie učiva. Očakávania a ciele, ktoré sme si na začiatku tvorby systému určili, môžeme vďaka časti výsledkov práce považovať na splnené. Systém vyhodnocuje otázky v rámci real-time decision a učiteľovi tak vhodné podmienky pre skracuje čas spracovania výsledkov a vytvára tak adaptívne riadenie prednášky.

5 Diskusia

Vytvorený softvér bol navrhnutý pre distribuované spracovanie dát na zlepšenie vedenia prednášok. Systém bol navrhnutý tak, aby spĺňal podmienky pre dosiahnutie cieľa práce. Takto funkčná aplikácia je pripravená na ďalší možný rast. V diskusii sa budeme zaoberať vylepšeniami, ktoré má zmysel do budúcna považovať za akceptovateľné a má zmysel pokračovať vo vývoji systému. Oblasť rozvoja aplikácií je pevne previazaná aj s programátorovými znalosťami v tejto oblasti.

Vylepšenia v našom prípade môžeme rozdeliť do dvoch hlavných skupín ktorým sa budeme venovať:

- Technické
- Praktické

Technické vylepšenia

V tejto časti sa nachádzajú technické zlepšenia existujúceho systému z pohľadu tvorby aplikácie pomocou nových metód, ktoré sa každoročne obmieňajú na základe pokrokov v informatike.

1. Zmena Hesla pre učiteľa – aktuálne postačujúce bolo vytvorenie jedného hesla, ktoré bolo používané pri každom teste. Overenie konkrétneho hesla sa nachádza v kóde programu. V rámci bezpečnosti prístupu k dátam, sa tento spôsob nepoužíva. Preto jeden z návrhov aktualizácie systému, je vytvorenie funkcie generovania unikátneho hesla pre učiteľa.
2. Zmena nahrávania otázok – databázová schéma s 1 tabuľkou všetkých otázok a pracovnou tabuľkou otázok zabezpečuje nahrávanie otázok iba kompetentným osobám, ktoré majú prístup na vpisovanie do tabuliek databázy, no v rámci jednoduchosti by bolo dobré vytvoriť novú funkciu s metódou pre vpisovanie a update otázok a to v takej forme, aby s ňou vedel pracovať človek, ktorý nemá skúsenosť s SQL príkazmi.
3. Vytvorenie tlačidla na stiahnutie reportu – osoby, ktoré nemajú prístup do databázy nevedia fyzicky pristupovať k report dát v ktorýkoľvek deň. Vylepšením aplikácie by preto bolo vytvorenie tlačidla (buttonu) na stiahnutie reportu, ale zároveň by človek nemusel pristupovať k dátam priamo v databáze, čo by obmedzilo možnosť poškodenia dát v dátovom sklade ľudským zlyhaním.

4. Pridelenie časovej stopy (timestamp) – aktuálna logika systému je nastavená tak, že databázu si musí spravovať používateľ. Databáza sa musí naplniť pri každej novej prednáške a zároveň aj po prednáške je potrebné urobiť si report. Vylepšením by bolo priradiť časové pečiatky, čo by používateľovi vytvorilo komfort, kedy by nemusel zakaždým vymazať databázu pri opakovanom používaní.

Praktické vylepšenia

Do tejto kategórie zaraďujeme individuálne preferencie človeka. Niektoré z vylepšení plynú na základe výsledkov vyplnenia dotazníka spokojnosti, ktorý bol vyplnený študentmi z testovacej vzorky.

1. Prvou požiadavkou je registrácia študenta – ak by sa systém ďalej používal častejšie a zároveň by slúžil na sledovanie účasti študentov, je potrebné vytvoriť registráciu študenta, aby sa študenti neevidovali pod unikátnym session ID ale boli by zapísaný do istej kategórie napríklad podľa ročníka alebo krúžkov.
2. Vytvorenie možnosti zaznačiť viac odpovedí pre jednu otázku
3. Vytvorenie prehľadu výsledkov. Jednou z odoziev študentov bola aj požiadavka na zobrazenie ich výsledkov. Vzhľadom na fakt, že systém bol zameraný na uľahčenie vedenia prednášky hlavne pre vyučujúceho, táto funkcionalita nebola zapracovaná.
4. Poslednou požiadavkou je user design. Vytvorenie customizovaného systému, kde si klient môže vybrať vlastné pozadie je dobrým vylepšením a posilnením systému.

V rámci práce sme sa prioritne zamerali na hlavné požiadavky, ktoré boli vytvorené používateľmi, no vylepšenia v podobe nových tlačidiel, či funkcií môžu podporiť jednoduchšiu prácu so systémom. V aktuálnom stave je na používanie systému potrebná aj mierna znalosť s používaním databáz.

6 Záver

Práca zahrnula teoretickú časť, na základe ktorej bol vytvorený a zrealizovaný experiment. Pred samotným testovaním bolo potrebné vytvoriť systém, ktorý by bol schopný splniť všetky požiadavky, ktoré sú v experimente. Záver práce opisuje aj zhodnotenie práce, jej prínos z pohľadu tvorby systému ale aj prínos pre používateľov.

Prvá časť priniesla teoretické priblíženie pojmov distribuovaného systému. Po tejto časti môžeme povedať, že v súčasnosti sa viacero aplikácií vytvára práve týmto distribuovaným spôsobom. V dnešnej rýchlo rozvíjajúcej sa informačnej sfére potrebujeme rýchle procesy dostupné odkiaľkoľvek a práve takéto riešenie pomocou distribuovaných systémov je aktuálne jeden z najvhodnejších prístupov na tvorbu. Môžeme však predpokladať, že aj tento spôsob vývoja aplikácií sa bude v budúcnosti naďalej vyvíjať.

V druhej časti sme si stanovili ciele na základe ktorých sa začalo s tvorbou systému a plnením zadaných cieľov. Funkčnosť systému bola následne overená experimentom na prednáške, ktorej sa zúčastnili študenti Ekonomickej univerzity. Vytvorenie systému sa považuje za kľúčovú časť práce, pretože program bolo potrebné skonštruovať tak, aby vyhovoval všetkým požiadavkám. Prvotné testovanie sa vykonávalo na lokálnom počítači, pravidelne počas programovania. Počas vyhotovenia systému sa vyskytli 2 kritické body. Za prvý z nich považujeme naprogramovanie správnych funkcií programu tak, aby vykonávali správne procesy a po spustení a kompilácii nenastali fatálne chyby. Druhým kritickým miestom bolo nasadzovanie. Samotné nasadenie si vyžaduje veľké množstvo prispôsobení ako napríklad zmena nástrojov z nástroja gradlu na maven, prepojenie IP adries naprieč všetky podsystémy programu alebo získanie http path. Všetky tieto prispôsobenia programu, pri prechode z lokálneho spúšťania na globálne, sú závislé od servera na ktorý sa systém nasadzuje. Po nasadení na server bolo možné začať s testovaním.

Výsledky experimentu boli získané z dvoch testovaní. Funkcionality systému fungovali presne podľa požiadaviek. Na systém bolo naraz pripojených viac študentov. Každý z nich sa na systém pripájal pomocou svojho mobilu alebo počítača. Spolu s vedúcim práce sme priebežne sledovali server učiteľa a kontrolovali pribúdajúce údaje od študentov. Celkové testovanie spolu s vyplnením dotazníka trvalo približne 25 minút. Odpovede na otázky sú zaznamenané v kapitole 4.5.2. Rozptyl správnych a nesprávnych odpovedí bol veľký a žiadna z otázok nemala 100% úspešnosť.

Dotazník spokojnosti od študentov slúžil ako podklad pre 5. kapitolu Diskusia. V tejto časti sa nachádzajú možné vylepšenia pre rast systému, ktoré môžu slúžiť ako materiál pre tvorbu budúcej práce.

Po otestovaní môžeme zhodnotiť ciele práce za splnené. Za hlavný prínos systému považujem jednoduchý spôsob ako vyhodnocovať otázky v reálnom čase pomocou vytvoreného systému. Túto skutočnosť sme zachytili už počas testovania, kedy sme na učiteľskom serveri po obnovení stránky videli, že počet zodpovedaných otázok bol po každom novom načítaním väčší, čo vyvodzovalo záver, že študenti postupne ukončujú vyplnenie testov a dáta sú hneď k dispozícii. Kladný názor na zaradenie takejto vyučovacej metódy prišiel aj zo strany študentov v nepovinnej časti dotazníka ktorou bolo pole hodnotenie návrhy.

Zoznam literatúry

Academy, P. (15. 3 2020). *Proxia Academy Kreativita na prvom mieste*. Dostupné na Internete: Proxia Academy - webová lokalita: http://www.proxia.org/sk

Ambruš, R. (2004). *Multimediálna podpora predmetu Logické systémy: tímový projekt.* Bratislava: Slovenská technická univerzita.

Babinská, M. (2010). Súčastný stav a možnosti e-learningu podpory vzdelávania na všetkých typocho škôl na Slovensku. *Súčastný stav a možnosti e-learningu podpory vzdelávania na všetkých typocho škôl na Slovensku* (s. 358-369). Bratislava: Študentská vedecká konferencia FMFI UK.

Bitly. (22. 2 2020). Dostupné na Internete: Bitly - webová lokalita: https://bitly.com/

Bitly. (22. 2 2020). *What is Bitly?* Dostupné na Internete: What is Bitly? - webová lokalita: https://support.bitly.com/hc/en-us/articles/230895688-What-is-Bitly-

Brown, K. (19. 1 2020). *How-To-Geek*. Dostupné na Internete: How-To-Geek - webová lokalita: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

Buchalcevová, A. (2005). *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* Praha: Grada.

Burgerová, J. (2008). *Tvorba a aplikácia e-learningových kurzov vo vysokoškolskej výučbe.* Praha: Soukromná vysoká škola ekonomických studií, s.r.o.

Castagnetto, J., & kolektív. (2007). *PHP programujeme profesionálne.* Brno: Computer Press.

Color HEX. (8. 2 2020). Dostupné na Internete: Color Hex - webová lokalita: https://www.color-hex.com/color/633580

Čerňanský, M. (19. 3 2020). *Paralelné programovanie*. Dostupné na Internete: Paralelné programovanie - webová lokalita: http://www2.fiit.stuba.sk/~cernans/pp/pp_texts/pp_prednaska_2_uvod.pdf

DataStax. (9. 2 2020). *CQL for Cassandra2.0 & 2.1 DataStax, Inc. Documentation*. Dostupné na Internete: CQL for Cassandra - webová lokalita: http://doc.datastax.com/en/cql/3.1

Fiverr. (9. 2 2020). Dostupné na Internete: Fiverr - webová lokalita: https://www.fiverr.com/riffibadr/teach-u-php-mysql-html-css-javascript-jquery-programming-basics-with-practicing

G. Decandia, D. H. (2007). *Dynamo: Amazon's Highly Available Key-value Store.* Operating Systems Review.

G. Reynolds, G. S. (2010). *Principles of Information Systems.* Boston: Course Technology.

GeeksforGeeks. (5. 1 2020). *geeksforgeeks.org*. Dostupné na Internete: internet: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

Gingers. (15. 3 2020). *Gingers©*. Dostupné na Internete: Gingers© - webová lokalita: https://www.gingers.sk/about_sk.html

GitBash. (19. 1 2020). *Git for Windows*. Dostupné na Internete: Git for Windows - webová lokalita: https://gitforwindows.org/

GitHub. (19. 1 2020). *GitHub Inc.* Dostupné na Internete: GitHub, Inc. - webová lokalita: https://github.com/github

Guzan, M. (2015). Variations of Boundary Surface in Chua's Circuit. *Radioengineering*, 814-823.

Hat, R. (19. 1 2020). *OpenShift*. Dostupné na Internete: inetrnet: https://www.openshift.com/

Hogan, P. B. (2011). *HTML5 a CSS3.* Praha: Computer Press.

Holák, P. (29. 3 2020). *DISTRIBUOVANÉ SYSTÉMY NA SPRÁVU VERZIÍ – STOJÍ TO ZA TO?* Dostupné na Internete: DISTRIBUOVANÉ SYSTÉMY NA SPRÁVU VERZIÍ – STOJÍ TO ZA TO? - webová lokalita: http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/kniha/2012/Resources/Essays/Essay_35.pdf

Holzschlag, E. M. (2006). *HTML a CSS jdi do toho.* Praha: Grada Publishing.

House of Bots. (15. 2 2020). Dostupné na Internete: House of Bots / webová lokalita: https://www.houseofbots.com/news-detail/12146-4-8-most-popular-programming-languages-&-frameworks-of-2019---all-programmer-should-have-knowledge

I. Holubova, J. K. (2015). *Big Data a NoSQL databáze.* Praha: Grada.

I. Porumbi, S. T. (16. 3 2020). *Moodle*. Dostupné na Internete: Bringing the technical and didactial perspective together in the design and development of a Moodle App within the Frame model: http://research.moodle.net/mod/resource/view.php?id=100

J. Arlow, I. N. (2007). *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh.* Brno: Computer Press.

J. Vymětal, A. D. (2005). *Informačný a znalostní management v praxi.* Praha: LexisNexis.

Janeček, J. (31. 3 2020). *Distribuované systémy* . Dostupné na Internete: Distribuované systémy - webová lokalita: https://dsn.felk.cvut.cz/wiki/_media/vyuka/x36dsv/prednasky/x36dsv_skripta_2000.pdf

K. Matiaško, M. V. (2009). *Databázové systémy a technológie.* Bratislava: STU.

Klimeš, C. (25. 3 2020). *Distribuovaé systémy*. Dostupné na Internete: Distribuovaé systémy - webová lokalita: https://www1.osu.cz/~prochazka/ds/SkriptaKlimes.pdf

Kontis. (15. 3 2020). *Kontis elearning*. Dostupné na Internete: Kontis elearning - webová lokalita: https://www.kontis.sk/home.html

Larman, C. (2004). *Agile and Iterative Developmentt: A Manager's Guide.* Boston: Addison-Wesley.

Ledvina, J. (29. 3 2020). *Prednášky z distribuovaných systému*. Dostupné na Internete: Prednášky z distribuovaných systému - webová lokaita: https://zcu.arcao.com/kiv/ds/ds.pdf

M. Fikar, Ľ. M. (15. 3 2020). Výskum potrieb a možností online vzdelávania verejnej správy v stredoeurópskom kontexte a príručka pre lektorov. Bratislave, Slovensko. Dostupné na Internete: https://www.uiam.sk/~fikar/books/fsev/index.htm

Mandík, O., & Vrba, L. (2007). *Stručný úvod do jazyka PHP.* Praha: skriptá.

Maven. (5. 1 2020). *Apache Maven Project*. Dostupné na Internete: chrome: https://maven.apache.org/

Microsoft. (15. 2 2020). Dostupné na Internete: Microsoft - webová lokalita: https://docs.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/

Microsoft. (14. 2 2020). Dostupné na Internete: Microsoft - webová lokalita: https://support.office.com/sk-sk/article/zobrazenie-v%C3%BDsledkov-v-slu%C5%BEbe-microsoft-forms-31c4b0cd-a89b-42e7-9429-1c14a1d8ef67

Molnár, Z. (2001). *Efektívnost informačných systému 2.* Praha: Grada.

MoodleNet. (16. 3 2020). *Moodle*. Dostupné na Internete: Moodle - webová lokalita: https://docs.moodle.org/38/en/Main_page

MySQL. (9. 2 2020). Dostupné na Internete: MySQL - webová lokalita: https://www.mysql.com

Oracle. (9. 2 2020). Dostupné na Internete: Oracle - webová lokalita: https://www.oracle.com/database/

Oracle. (16. 2 2020). Dostupné na Internete: Oracle - webová lokalita: https://docs.oracle.com/javaee/7/api/javax/resource/spi/ConnectionManager.html

Pippo. (16. 2 2020). *Pippo Micro Java Web Framework*. Dostupné na Internete: Pippo Micro Java Web Framework -webová lokalita: http://www.pippo.ro/doc/routes.html

Pokorný, J., & Valenta, M. (2013). *Databázové systémy.* Praha: ČVUT.

Prucha, J. (2009). *Pedagogický slovník.* Praha: Portál.

Secretariat, I. C. (9. 2 2020). *Information Technology - Abstract Syntax Notation One : Specification of Basic Notation*. Dostupné na Internete: Specification of Basic Notation - webová lokalita: https://www.iso.org/standard/38641.html

Schafer, S. M. (2009). *HTML, YHTML a CSS.* Praha: Grada Publishing.

Sudický, J. Z. (2012). *E-learning: učení (se) s online technologiemi.* Praha: Wolters Kluwer.

Suehring, S. (2008). JavaScript krok za krokem. In S. Suehring, *JavaScript krok za krokem* (s. 336). Brno: Computer Press.

Systems, E. (15. 3 2020). *Empire Systems*. Dostupné na Internete: Empire Systems - webová lokalita: https://www.empiresystems.sk/

Šec, D. (22. 3 2020). *Distribuované systémy*. Dostupné na Internete: Distribuované systémy - webová lokalita: https://theses.cz/id/wow3pb/STAG84353.pdf

Thomas, A. H. (2007). *Programátor pragmatik: Jak se stát lepším programátorem a vytvářet kvalitní software.* Brno: Computer Press.

Valjašek, P. (22. 3 2020). *Systém pre distribuované spracovanie výpočtovo náročnych úloh*. Dostupné na Internete: Systém pre distribuované spracovanie výpočtovo náročnych úloh - webová lokalita: https://diplomovka.sme.sk/zdroj/3239.pdf

Vrana, I. (2008). *Projektování informačních systémů s UML.* Praha: Česká zemědělská univerzita.

WNX.com. (9. 2 2020). Dostupné na Internete: WNX.com - webová lokalita: http://blog.wnx.com/2016/12/11/converting-an-application-from-mysql-to-oracle-in-grails/

Wong, k. a. (2009). *A High Perfomance Implementation for Near Real Time Asynchronous Replication.* USA: Los Alamitos.

Y. Belabbas, S. S. (19. 3 2020). *Dynamic load Balancing Strategy for Grid Comuting*. Dostupné na Internete: Dynamic load Balancing Strategy for Grid Comuting - webová lokalita: https://www.researchgate.net/publication/250852207_Dynamic_Load_Balancing_Strategy_for_Grid_Computing

Z. Šochová, E. K. (2014). *Agilní metody řízení projektů.* Praha: Computer Press.

Zoznam Obrázkov

Zoznam Tabuliek

Zoznam Príloh

V prílohe sa nachádza CD so zdrojovým kódom programu a videonahrávka z praktického testovanie systému.