

**UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**PLATFORMA PRE KOLABORATÍVNE VYUČOVANIE
MATEMATIKY HEJNÉHO METÓDOU**

Diplomová práca

Bratislava, 2018

Bc. Daniel Linhart

**UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**PLATFORMA PRE KOLABORATÍVNE VYUČOVANIE
MATEMATIKY HEJNÉHO METÓDOU**

Diplomová práca

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: RNDr. Peter Borovanský, PhD.

Bratislava, 2018

Bc. Daniel Linhart



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Daniel Linhart

Študijný program: aplikovaná informatika (Jednoodborové štúdium,
magisterský II. st., denná forma)

Študijný odbor: aplikovaná informatika

Typ záverečnej práce: diplomová

Jazyk záverečnej práce: slovenský

Sekundárny jazyk: anglický

Názov: Platforma pre kolaboratívne vyučovanie matematiky Hejného metódou
Collaborative platform for Hejny's math teaching method

Ciel: Hejného metóda vyučovania matematiky sa čoraz viac rozširuje do výuky základných škôl. V rámci bakalárskej práce sme ukázali, že použitie IKT (mobilných zariadení) je vitaným obohatením tejto metodiky a vyučovania. Umožňuje jednotlivým žiakom prispôsobiť tempo práce a zadávať primerané úlohy podľa individuálnych potrieb konkrétneho žiaka. Taktiež má svoju silnú motivačnú stránku. Najväčšou nevýhodou tohto spôsobu výuky, ako sme sa aj presvedčili na testovanej aplikácii je, že ked' v triede každý žiak sleduje svoj tablet, prestáva komunikácia v triede a kooperatívne riešenie žiakov, čo je v priamom rozpore s Hejného metódou, aj s modernými trendmi vo vyučovaní. Cieľom práce je preto navrhnúť prostredie pre kolaboratívne riešenie matematických úloh v rámci triedy, ilustrované na jednom matematickom prostredí Hejného metódy. Základná funkcionality prostredia by mala umožniť zdieľať jednu úlohu viacerými žiakmi, jej spoločné riešenie pomocou aplikácie, interakciu žiak-žiak a učiteľ-žiak. Klient-server architektúra riešenia umožní učiteľovi hromadne zadávať úlohy, tiež štatisticky diagnostikovať výsledky, najčastejšie chyby, či postupy žiakov. Rola učiteľa umožní sledovať progres jednotlivých žiakov, skupín žiakov, aj celej triedy. Týmto učiteľ získa zaznamenávací a diagnostický nástroj.

Vedúci: RNDr. Peter Borovanský, PhD.

Konzultant: RNDr. Dagmar Môťovská

Katedra: FMFI.KAI - Katedra aplikovej informatiky

Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 09.10.2016

Dátum schválenia: 13.10.2016

prof. RNDr. Roman Ďuríkovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie

Čestne vyhlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením vedúceho diplomovej práce, s použitím uvedenej literatúry a zdrojov dostupných na internete.

V Bratislave, dňa xx.xx.20xx

Meno a priezvisko

Pod'akovanie

Abstrakt

LINHART, Daniel: *Platforma pre kolaboratívne vyučovanie matematiky Hejného metódou (Diplomová práca)* – Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. – Školiteľ: RNDr. Peter Borovanský, PhD.: FMFI UK, 2018

Kľúčové slová:

Abstract

LINHART, Daniel: *The Hejny method educational software – Adding triangles (Diplomová práca)* - Comenius University in Bratislava, Faculty of mathematics, physics and informatics; Department of applied informatics - Adviser: RNDr. Peter Borovanský, PhD.: FMFI UK, 2016

Key words:

Obsah

1	Úvod.....	12
1.1	Motivácia a analýza problematiky.....	12
1.2	Cieľ práce.....	12
1.3	Existujúce riešenia	13
1.4	Štruktúra práce.....	13
2	Východiská.....	15
2.1	Hejného metóda.....	15
2.2	Komunikácia	15
2.3	Podobné riešenia	15
2.4	Cieľová skupina používateľov.....	15
3	Technológie.....	16
3.1	Čo je to Firebase	16
3.1.1	Ako to funguje ?.....	16
3.1.2	Firebase Database.....	17
3.1.3	Firebase Auth.....	17
3.2	Node.js	17
3.2.1	Neblokujúci cyklus udalostí	18
3.2.2	Asynchrónne programovanie	19
3.3	Porovnanie SQL a NoSQL databáz.....	20

3.3.1	SQL databázy	21
3.3.2	NoSQL databázy.....	21
3.3.3	Zhrnutie.....	22
3.4	Zhrnutie.....	22
3.4.1	Unity 3D.....	23
4	Návrh riešenia.....	24
4.1	Cieľ projektu	24
4.2	Funkčné požiadavky	24
4.2.1	Zdieľanie	24
4.2.2	Virtuálna tabuľa	24
4.2.3	Zaznamenávanie štatistik	24
4.2.4	Zrkadlenie obrazovky žiaka	25
4.2.5	Webové rozhranie pre učiteľa	25
4.3	Návrh používateľského prostredia – mobilná aplikácia	26
4.3.1	Prihlasovanie a registrácia do aplikácie	27
4.3.2	Výber triedy	29
4.3.3	Návrh zdieľať s	30
4.3.4	Návrh zdieľať – Tabuľa	31
4.4	Návrh používateľského prostredia – webová aplikácia	32
4.4.1	Návrh vytvorenie/editovanie triedy	33

4.4.2	Vytvorenie úlohy.....	33
4.4.3	Pridanie úloh na tabuľu.....	34
4.4.4	Zobrazenie obrazovky žiaka	34
4.4.5	Zobrazenie štatistik.....	34
4.5	Návrh databázy	35
4.5.1	Triedy	35
4.5.2	Úlohy	35
4.5.3	Úlohy pripnuté na tabuľu a zdieľané obrazovky.....	36
4.5.4	Štatistiky	36
4.5.5	Tabule.....	36
4.5.6	Používatelia	36
5	Implementácia	38
5.1	Integrácia s Firebase	38
5.2	Vytvorenie C# modulu pre mobilnú aplikáciu.....	39
5.3	Úlohy na tabuľi	39
5.3.1	Pridanie príkladu na tabuľu	39
5.3.2	Zobrazenie úloh na tabuľi.....	39
5.4	Zdieľanie úloh.....	40
5.4.1	Odoslanie požiadavky	40
5.4.2	Prijatie požiadavky	40

5.4.3	Zaznamenávanie ľahov	41
5.5	Webové rozhranie	41
5.5.1	Prihlásenie a registrácia	43
5.5.2	Vytvoriť úlohu.....	43
5.5.3	Zoznam tried a mojich úloh.....	44
5.5.4	Priradenie úloh ku triedam.....	44
6	Testovanie.....	45
6.1	Prvé testovanie.....	45
6.1.1	Priebeh testovania.....	45
6.1.2	Zhrnutie testovania.....	46
6.2	Druhé testovanie	46
Záver	47	
Literatúra a internetové zdroje	48	
Prílohy.....	50	

Zoznam obrázkov

Obrázok 1 Príklad blokujúceho kódu	18
Obrázok 2 Príklad neblokujúceho kódu v Node.js.....	19
Obrázok 3 Synchrónne čítanie textového súboru.....	19
Obrázok 4 Nesprávne asynchrónne čítanie súboru v Node.js.....	20
Obrázok 5 Správne asynchrónne čítanie súboru v Node.js.....	20
Obrázok 6 Návrh komunikácie	27
Obrázok 7 Prihlásovanie	28
Obrázok 8 Registračný formulár s ukážkou chybného vstupu.....	29
Obrázok 9 Scéna kde si žiak volí triedu do ktorej chce vstúpiť. V spodnej časti môžeme vidieť možnosť pridania triedy	30
Obrázok 10 Sekvenčný diagram nadviazania spojenia	30

1 Úvod

1.1 Motivácia a analýza problematiky

Dovoľte mi aby som nadviazal na úvod z mojej bakalárskej práce[1], v ktorom spomínam ako je čím ďalej náročnejšie udržať pozornosť dieťaťa štandardnými metódami výuky. Preto som sa rozhodol vytvoriť mobilnú aplikáciu na podporu výuky matematiky Hejného metódou[2]. Pri testovaní aplikácie sme sa presvedčili, že tento spôsob výuky je pre žiakov zaujímavý a oplatí sa v ňom pokračovať.

Pri testovaní sme sa ale mohli presvedčiť, že technológie zabíjajú spoluprácu medzi žiakmi. Žiaci pracovali na svojich zariadeniach a plnili generované úlohy. Vďaka čomu úplne vypadla spolupráca a komunikácia medzi žiakmi. Tento efekt je nežiadúcim prvkom nakoľko je v priamom rozpore s Hejného metódou[1][2]. No nie je to len Hejného metóda ktorá si zakladá na dôležitosti spolupráce. O tejto téme je viacero prác a kníh za mienku stojí publikácia Nové formy skupinového vyučovania.

Tradičný spôsob učenia je zameraný rozumovo a vyžaduje si značné úsilie. Utlmuje pozornosť a radosť z nových vedomostí, spomaľuje schopnosť učiť sa a potláča nutkanie klásť otázky. Vyvoláva odpor a rýchlo sa pri tomto spôsobne učenia zabúda. Dôvodom je využívanie len kognitívnej časti mozgu. Pričom pri učení v skupinách zapájame obe polovice mozgu. Výhodami tohto typu výuky sú rýchle vnímanie, spontánne aha-efekty a rýchlo vybaviteľné spomienky[3].

Z tohto dôvodu sme sa rozhodli vytvoriť platformu pre vyučovanie matematiky Hejného metódou. Cieľom platformy bude obnoviť spoluprácu medzi žiakmi a zároveň kontrolovať žiakov ako sa im darí plniť učiteľom zadané pokyny.

1.2 Ciel' práce

Našou prácou chceme prispieť k používaniu tabletov, na vyučovacích hodinách. Vďaka našej práci si žiaci prvého stupňa základných škôl budú môcť precvičiť matematiku a zároveň sa podeliť o zaujímavé úlohy zo svojimi spolužiakmi. Zároveň do platformy pribudne aj rola učiteľa, ktorý môže žiakom zadáť rôzne úlohy a zároveň kontrolovať ich

aktivitu. Našou úlohou bude vytvoriť platformu, ktorá bude pre žiaka ale aj pre učiteľa zaujímavá a uľahčí im vzdelávací proces.

1.3 Existujúce riešenia

V 21. storočí je vskutku nevýdané natrafiť na jedinečnú aplikáciu. I v našom prípade existuje množstvo podobných riešení. Tieto riešenia by sa dali rozdeliť do dvoch kategórií.

Do prvej kategórie by som zaradil úlohy, ktoré sa podobajú spracovaním a majú v sebe implementovanú real-time komunikáciu. Týchto riešení je nespočetné množstvo. Pri týchto aplikáciách je na škodu, že nemajú edukatívny charakter.

Druhým typom ani tak neboli aplikácie ako využívanie hlavnej myšlienky našej práce. Touto myšlienkovou je komunikácia. Išlo o rôzne publikácie, v ktorých sa vysvetľovala dôležitosť spolupráce a akými doteraz dostupnými technológiami sa ju snažili udržiavať.

Je málo aplikácií, ktoré sa tieto dve kategórie snažia prepojiť. Hlavnou ideou našej práce je tieto dve kategórie prepojiť. Platforma pre kolaboratívne vyučovanie matematiky Hejného metódou bude mať štandardné vlastnosti edukačnej hry s možnosťou kooperácie pri riešení úloh.

1.4 Štruktúra práce

V prvej kapitole sa zaoberáme analýzou a opisom problému. Ďalej sa pozrieme na existujúce riešenia a porovnáme ich.

V druhej kapitole si podrobne analyzujeme technológie ako Firebase, Node.js, Angular2, SQL databázy. Vysvetlíme si ich výhody a nevýhody. V závere kapitoly si rozoberieme dôvody prečo sme sa rozhodli pre Firebase a Angular2.

Tretia kapitola sa zaoberá funkčnými požiadavkami na systém, minimálnym systémovým požiadavkám, cieľovej skupine a inštalácii.

Vo štvrtej kapitole si podrobnejšie zanalyzujeme podobné riešenia.

Piata kapitola sa zaoberá návrhom samotnej platformy, ktorá je zložená z troch častí. Prvou časťou je návrh komunikácie v mobilnej aplikácii. Druhou časťou je návrh webového rozhrania, ktoré je určené pre rolu učiteľa. Posledná, tretia časť obsahu návrh databázy.

V šiestej kapitole sa zaoberáme implementáciou systému. Kapitola je zameraná na vysvetlenie základnej funkcionality, priblíženie štruktúry kódu a použitie jednotlivých technológií.

V poslednej kapitole sa zameriavame na testovanie aplikácie v praxi. Kapitola obsahuje postrehy, ktoré sme pri testovaní spozorovali. Pripomienky, ktoré sme dostali od používateľov. Kapitola je zakončená celkovým zhrnutím ako sa aplikácií darilo pri testovaní.

2 Východiská

2.1 Hejného metóda

2.2 Komunikácia

2.3 Podobné riešenia

2.4 Cieľová skupina používateľov

Cieľovou skupinou sú všetci žiaci prvého stupňa základných škôl a ich učitelia nezávisle od toho akým spôsobom sa na školách vyučuje.

3 Technológie

Základom pre realtime platformu bolo vhodne zvoliť technológie. Z dôvodu multiplatformovosti musela technológia na prenos dát mať podporu tak ako webového rozhrania, tak aj mobilných rozhraní (android, iOS). Z tohto dôvodu sa nám výber technológií značne zúžil. Rozhodovali sme sa medzi technológiou Firebase od firmy Google a variantom Node.js ako server a SQL Databázou. Kde by naša aplikácia komunikovala so serverom pomocou webových servisov a následne by server vykonával zmeny v databáze. V nasledujúcej časti si priblížime tieto technológie.

3.1 Čo je to Firebase

Firebase[4] je vývojárska platforma vyvinutá firmou Firebase, Inc. Firebase je Backend-as-a.Service (BaaS), vďaka čomu je firebase našim serverom, API aj databázou. Firebase ma tieto komponenty.

- *Firebase Analytics*
- *Firebase Cloud Messaging*
- *Firebase Auth*
- *Firebase Database*
- *Firebase Storage*
- *Firebase Hosting*
- *Firebase Test Lab for Android*
- *Firebase Crash Reporting*

Tieto komponenty môžeme využívať jednotlivo ale aj ako jeden celok. V našej práci využívame Firebase Auth a Firebase Database.

3.1.1 Ako to funguje ?

Databáza umožňuje vytvárať spolupracujúce aplikácie tým, že umožní zabezpečený prístup do databázy. Údaje zotravajú na lokálnej úrovni aj v prípade, že je zariadenie offline. Po obnovení spojenia zariadenie synchronizuje zmeny lokálnych údajov s údajmi

v databáze, ktoré sa vyskytli počas nedostupnosti zariadenia a tým sa automaticky pospájajú konflikty.

Rozhranie API je navrhnuté tak Realtime Database je navrhnuté tak, aby umožnilo len rýchlo dostupné operácie. Čím nám umožní vytvoriť aplikáciu bez kompromisov v reakcii aj pri vysokom počte používateľov.

3.1.2 Firebase Database

Firebase databáza[4] je databáza hostovaná v cloude. Dáta sú ukladané ako JSON. Všetky dátá sú synchronizované v reálnom čase s každým pripojeným klientom. Pri vytvorení multiplatformovej aplikácie, všetci klienti zdieľajú jednu inštanciu realtime databázy. Vďaka tomu automaticky dostávajú aktualizácie s najnovšími dátami.

Služba poskytuje rozhranie API umožňujúce synchronizáciu dát medzi klientmi a ukladaním dát na cloudových úložiskách spoločnosti Firebase. Databáza je dostupná pomocou rozhrania REST API a väzieb na pár javascriptových frameworkov (AngularJS, React). Spoločnosť dodáva aj klientske knižnice, ktoré umožňujú integráciu s aplikáciami Java, Objective-C, JavaScript, Android, iOS, Node.js, swift, Unity. Vývojári používajúci databázu v reálnom čase, majú možnosť zabezpečiť si svoje dátá radou bezpečnostných nastavení na strane servera.

3.1.3 Firebase Auth

Poskytuje backendové služby, ľahko použiteľné sústavy SDK a ready-made knižníc na autentifikáciu používateľov. Firebase Authentication[6] využíva priemyselné štandardy ako OAuth 2.0 a OpenID Connect. Vďaka týmto štandardom je ich integrovať do vlastných aplikácií. Podporovaná je autentifikácia pomocou hesiel, telefónnych čísel ale aj poskytovateľov federálnej identity ako Google či Facebook.

3.2 Node.js

Node.js je open source serverový framework. Je dostupný na všetky platformy (Windows, Linux, Mac OS, atď.). Node.js využíva JavaScript na strane serveru. Od ostatných sa odlišuje udalosťami riadenou architektúrou (event-driven architecture). Bežné webové servery pri

paralelných volaniach vytvárajú nové vlákna, čím dochádza k zahľteniu pamäte. Node.js dokáže spracovať tisícky pripojení s jediným vláknom a obmedzenou pamäťou.

Architektúru Node.js a mechanizmy ako pracuje si lepšie vysvetlíme v podkapitolách[11].

3.2.1 Neblokujúci cyklus udalostí

Node.js dokáže obsluhovať viacero požiadaviek na server a súčasne neplytvá prázdnymi cyklami v I/O úlohách. Z tohto dôvodu je označovaný ako neblokujúci. Bežné servery blokujú nasledujúce požiadavky, pokiaľ sa vykonáva operácia. Node.js je neblokujúci vďaka využitiu cyklu udalostí. Cyklus udalostí je softvérový model, v ktorom sú skombinované neblkoujúce I/O a event-driven I/O. Každá registrovaná udalosť je zavolaná vtedy keď sa niečo udeje v programe.

```
1  <?php
2
3  //Príklad blokujúceho kódu
4  print("Ahoj");
5  sleep(5);
6  print("Druhy prikaz");
7  print("Koniec");
8
9  // Vypíše sa Ahoj
10 // Program čaká 5 sek
11 // Vypíše sa Druhy prikaz
12 // Vypíše sa Koniec
```

Obrázok 1 Príklad blokujúceho kódu

Na obrázku Obrázok 1 môžeme vidieť blokujúci kód. Funkcia *sleep()* nám blokuje hlavné vlátko a tým pádom nemôžu byť vykonané ďalšie inštrukcie. Na obrázku Obrázok 2 máme príklad neblokujúceho kódu. Node.js využíva cyklus riadených udalostí. Čiže aj napriek použitiu blokovacej funkcie *setTimeout()* je funkcia neblokovaná. SetTimeout si zaregistrouje svoju udalosť a umožní programu pokračovať ďalej [11].

```
1 //Príklad neblokujúceho kódu v Node.js
2 console.log("Ahoj");
3 setTimeout(function{
4     console.log("Druhy prikaz");
5 },5000);
6 console.log("Koniec");
7
8 // Vypíše sa Ahoj
9 // Vypíše sa Koniec
10 // Vypíše sa Druhy prikaz
11
```

Obrázok 2 Príklad neblokujúceho kódu v Node.js

3.2.2 Asynchrónne programovanie

Pokým asynchrónna časť Node.js dovoľuje prijať a spracovať všetky žiadosti, asynchrónne programovanie sa stará o to aby boli žiadosti vybavené efektívne. Využíva pritom dostupnú pamäť a obmedzený počet časových cyklov. Node.js dosahuje vysokú mieru paralelnosti vďaka asynchrónnym volaniam cez funkciu callback.

```
1 <?php
2 //synchrónne čítanie súboru
3 $file = fopen('text.txt', 'r');
4 // čaká na otvorenie súboru
5 $content = fread($file, 10000);
6 // čaká na načítanie súboru
7 print($content);
8 // vypíše obsah
```

Obrázok 3 Synchrónne čítanie textového súboru

Pre lepšie vysvetlenie asynchrónnych volaní použijeme tri príklady v ktorých budeme čítať textový súbor. Na prvom obrázku môžeme vidieť synchrónne čítanie. Tento spôsob je neefektívny. Plytvá našim časom, kým čaká na súborový systém. Na obrázku Obrázok 4 je prepísaná verzia kódu z obrázku Obrázok 3 do JavaScriptu. Tu môžeme vidieť že kód spadne na chybe. Dôvodom je že ide o asynchrónne volanie. Čiže sa spustí funkcia *fs.open* a pokračuje vykonaním funkcie *fs.read*. Premenná *file* sa nastaví až po prijatí callback funkcie.

```

1 //Nesprávne čítanie súboru v Node.js
2 var fs = require('fs');
3 var file;
4 var buf = new Buffer(10000);
5 // posledny argument je callback funkcia
6 ▼ fs.open('text.txt','r',
7         function(handle){
8             file = handle;
9         });
10
11 ▼ fs.read(file,0,10000,null,
12         function(){
13             console.log(buf.toString());
14             file.close(file,function(){}));
15         });

```

Obrázok 4 Nesprávne asynchrónne čítanie súboru v Node.js

Na poslednom obrázku môžeme vidieť správne naprogramované asynchrónne volanie. Dôležitý je pre nás tretí parameter funkcie *fs.open*. Ide o parameter callback. V prvom parametri funkcie callback mame informáciu o úspešnosti, druhý označuje medzi výsledky z poslednej operácie[11].

```

1 //Správne asynchrónne čítanie súboru v Node.js
2 var fs = require('fs');
3 var file;
4 // posledny argument je callback funkcia
5 fs.open('text.txt','r',
6         function(handle){
7             var buf = new Buffer(10000);
8             fs.read(file,0,10000,null,
9                 function(length){
10                  console.log(buf.toString('utf8',0,length));
11                  file.close(file,function(){}));
12             });
13         });
14

```

Obrázok 5 Správne asynchrónne čítanie súboru v Node.js

3.3 Porovnanie SQL a NoSQL databáz

Pri výbere vhodných technológií serverovej časti bolo potrebné vhodne k serveru doplniť aj databázu. Ukladanie dát je pre nás dôležité. V tomto prípade nejde len o dátu, ktoré si žiaci medzi sebou vymieňajú ale aj o uchovávanie výsledkov. Na výber sme mali medzi SQL

a NoSQL databázami. Na to aby sme sa dokázali správne rozhodnúť sme si ale museli zanalyzovať, ktorý typ bude pre našu aplikáciu vhodnejší.

3.3.1 SQL databázy

Alebo relačné databázy sú postavené na relačnom modely. Základom relačných databáz sú jasne štruktúrované tabuľky. Optimálnu štruktúru databáz dosahujeme normalizáciou. Normalizácia je odstránenie alebo aspoň zníženie prebytočných položiek. V SQL databázach sú riadky chápane ako záznamy v tabuľke a stĺpce poskytujú informácie o relaciach medzi záznamami. Štandardným dotazovacím jazykom pre relačné databázy je SQL.

Vlastnosti relačných databáz

- Tabuľky majú jedinečné názvy
- Bunka tabuľky môže obsahovať práve jednu hodnotu
- Každý stĺpec má jedinečný názov
- Každá hodnota v jednom stĺpci je z rovnakej domény
- Poradie stĺpcov nemá význam, stĺpce môžeme medzi sebou vymieňať
- Neexistujú duplicitné záznamy

3.3.2 NoSQL databázy

V nerelačných databázach sa miesto štruktúrovaných tabuľiek na ukladanie dát používa pojem kľúč-hodnota. Dáta sa ukladajú do databázy bez schémy pre databázu. Dáta nie sú povinné dodržiavať štandardné schémy databáz, len sa ukladajú hodnoty pod poskytnuté kľúče. Tieto systémy sú využívané pri práci s veľkými množstvami dát. V týchto typoch databáz môžu byť dáta štruktúrovane, neštruktúrovane prípadne semi-štrukturované. Využíva schopnosť ukladať a načítať veľké množstvo dát bez závislostí na vzťahoch. Zvyčajne sú operácie obmedzované len na jednotlivé položky. NoSQL pracuje s distribuovanou architektúrou, a s dátami, ktoré sú redundantne uložené na viacerých serveroch. Vďaka tomuto spôsobu je systém ľahko škálovateľný a možné zlyhanie servera akceptované.

3.3.3 Zhrnutie

Relačné databázy pozostávajú z tabuľiek, ktoré musia mať jasne definovaný dátový model. Schéma je jasne daná a obsahuje vzťahy a obmedzenia, ktoré si vyžaduje dátová integrita. Dátový model musí byť normalizovaný a je založený na reprezentácii dát a nie na aplikačnej funkcionalite. K dátam sa pristupuje pomocou jazyku SQL. Jazyk má možnosť pristupovať k viacerým tabuľkám, vykonávať filtrovanie a agregáciu. SQL obsahuje aj funkcie pre logické spracovanie dát. Relačné databázy majú vlastné API, prípadne využívajú všeobecné API. Z dôvodu ukladania dát v ich prirodzenej štruktúre musia byť dátá mapované medzi aplikačnou a relačnou štruktúrou.

Kľúč-Hodnota databázy (NoSQL) nemajú pevne definovanú schému pre doménu. Doména je priestor do ktorého sú vkladané prvky. Medzi doménami nie sú explicitne definované vzťahy. V rámci jednej domény môžu mať prvky rôzne schémy. Prvky sú kľúče, ktoré môžu mať k sebe pripojenú dynamickú sadu kľúčov. Atribúty sú zvyčajne typu string, no v niektorých implementáciách majú typy. K dátam sa pristupuje pomocou API metód a integrálna logika je obsiahnutá v aplikačnom kóde. Cieľom NoSQL databáz je poskytovať SOAP a REST APIs, ktoré umožňujú dátové volania.

Oba typy databáz majú svoje výhody a nevýhody. Rozhodnúť sa medzi nimi nebolo jednoduché. Nakoniec sme sa rozhodli využiť NoSQL. Pri rozhodovaní zavážilo hlavne možnosť pristupovať k hodnotám na základe kľúča. Ďalším plusom pre NoSQL bol prístup k dátam pomocou REST alebo SOQP requestov. Túto kapitolu sme spracovali s použitím [7] [8]

3.4 Zhrnutie

Technológia Firebase aj technológia Node.js majú svoje výhody aj nevýhody a vybrať si medzi nimi nebolo jednoduché. Obe technológie pracujú s asynchronnými volaniami. Medzi výhody technológie Firebase sa dá zaradiť dostupnosť a podpora. Firebase má na svojom serveri aj clouдовú NoSQL databázu čo považujeme ako ďalšiu výhodu oproti technológií Node.js. Firebase ale má aj svoje nevýhody. Budúcnosť serveru nemáme v našich rukách,

pribúdajúcimi používateľmi stúpa cena za používanie. Medzi výhody Node.js považujeme plnú kontrolu nad serverom a bezproblémové presunutie na iný server. Node.js neobsahuje databázu je to len server tým by nám vznikli náklady na prevádzku databázy. Node.js nemá API na komunikáciu s klientom oproti od Firebase. Z tohoto porovania nám vyšiel ako víťaz Firebase. Na vývoj webového rozhrania sme si zvolili PHP framework Nette.

3.4.1 Unity 3D

Unity 3D je multiplatformový herný engine, ktorý bol vytvorený firmou Unity Technologies. Táto platforma sa používa na vývoj hier pre počítače, konzoly, VR/AR, mobilné zariadenia. Prvoplánovo bola platforma vyvinutá pre Apple, no časom sa rozšírila o ďalšie platformy. Unity 3D je zaujímavé hlavne vďaka podpore viacerých plaforiem. Platformy, ktoré Unity podporuje sú Windows, OSX, Android, iOS, BlackBerry, Windows Phone 8 a platformy herných konzol.

4 Návrh riešenia

4.1 Ciel projektu

Cieľom práce je vytvoriť platformu pre vyučovanie matematiky Hejného metódou, pomocou ktorej budú môcť žiaci spolupracovať. Platforma používateľom umožní zdieľať úlohy, ktoré im vygeneruje aplikácia ale aj tie, ktoré si sami vytvoria. V našej aplikácii sa nezabudlo ani na učiteľov, pre ktorých sú pripravené funkcie ako sledovanie aktivity a zadávanie úloh žiakom.

4.2 Funkčné požiadavky

V nasledujúcej podkapitole vykonáme podrobnejší rozbor požiadaviek na náš systém.

4.2.1 Zdieľanie

Žiakovi sa vygeneruje úloha, ktorá sa mu zdá náročná a rozhodne sa ju riešiť so svojimi spolužiakmi. Vyberie si funkciu zdieľať. Zobrazí sa mu zoznam používateľov, ktorý sa nachádzajú práve v triede a odošle im požiadavku riešiť s nimi príklad. Príjemcom sa zobrazí notifikácia, o tom že niekto s nimi chce spoločne riešiť príklad. Po prijatí požiadavky sa nadviaže spojenie medzi žiakmi. Spojenie sa ukončí po úspešnom vyriešení úlohy alebo v prípade že jeden zo žiakov dá podnet na ukončenie spojenia.

4.2.2 Virtuálna tabuľa

Virtuálna tabuľa slúži všetkým používateľom. Učiteľ na tabuľu môže pridať sady úloh. Žiaci môžu príklady z tabule riešiť ale aj pridať na tabuľu nové úlohy. Každá trieda by mala mať svoju vlastnú tabuľu, čím sa zabezpečí že úlohy sa dostanú k správnym adresátom. Žiaci budú o nových úlohách na tabuli notifikovaný po vstúpení do úvodného menu. Vyriešené úlohy sa žiakovi na tabuli zobrazovať nebudú.

4.2.3 Zaznamenávanie štatistik

Zaznamenávať sa budú dva typy dát. Do databázy sa bude zaznamenávať výsledok úlohy. Teda či žiak vyriešil úlohu správne alebo nesprávne z čoho sa bude dať vyčísliť štatistika

koľko úloh žiak vyriešil a s akou úspešnosťou. Druhý typ záznamu budú konkrétnie úlohy z virtuálnej tabule. Tu bude pre nás dôležité aj ako úloha vyzerala čiže sa bude logovať nie len výsledok ale aj zadanie.

4.2.4 Zrkadlenie obrazovky žiaka

Bude slúžiť učiteľovi na kontrolu žiakov v reálnom čase. V prípade podezrenia, že žiak nepracuje alebo si nevie dať rady s úlohou. Si učiteľ bude môcť vo svojom systéme otvoriť obrazovku žiaka, kde uvidí presnú kopiu jeho obrazovky aj s časom poslednej aktualizácie. Z informácií zobrazených v systéme bude vedieť identifikovať príčinu nečinnosti.

4.2.5 Webové rozhranie pre učiteľa

Pôjde o webovú aplikáciu, ktorá bude slúžiť učiteľom na kontrolu žiakov, roztriedenie žiakov do skupín (tried), zadávanie úloh, zobrazovanie štatistik.

Vytvorenie tried – používateľ webovej aplikácie bude môcť po prihlásení vytvoriť triedu. Na vytvorenie bude potrebné zadať meno triedy a nejaké heslo. Heslo bude musieť byť jedinečné. O tom či je heslo jedinečné bude informovať systém. Po vytvorení triedy aplikácia presmeruje používateľa na domovskú stránku kde triedu už bude vidieť.

Vytvorenie úloh – aplikácia by mala používateľovi umožniť vytvoriť úlohu. Táto úloha sa zaznamená do databázy. Používateľ bude môcť následne úlohu priradiť jednej alebo viacerým triedam. Prípadne bude môcť úlohu editovať alebo zmazať.

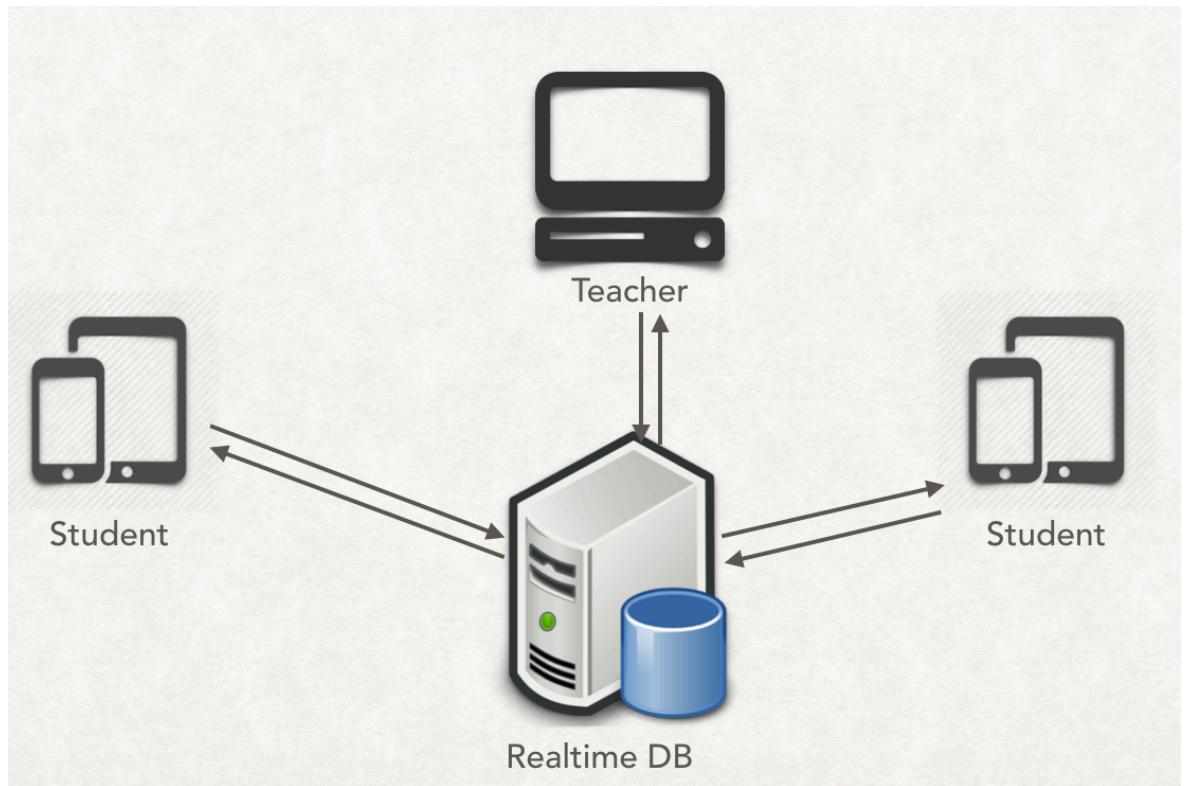
Zobrazenie štatistik – aplikácia umožní používateľovi zobraziť štatistiky triedy alebo jednotlivého žiaka. Štatistiky triedy sa budú zobrazovať formou zoznamu žiakov, kde pri každom žiakovi budú dve hodnoty počet správnych a počet nesprávnych úloh. Štatistiky žiaka sa budú zobrazovať ako zoznam príkladom vyriešených z virtuálnej tabule a celkový počet príkladov.

Zobrazenie obrazovky žiaka – používateľ webovej aplikácie bude mať možnosť si zobraziť obrazovku žiaka na ktorej sa mu zobrazí posledná úloha ktorú riešil v prípade, že práve nerieši úlohu. V opačnom prípade sa zobrazí aktuálna obrazovka aj s následnými zmenami. V obidvoch prípadoch sa zobrazí čas aktualizácie obrazovky.

4.3 Návrh používateľského prostredia – mobilná aplikácia

Mobilná aplikácia je určená pre žiakov základných škôl. Po spustení budú mať žiaci na výber medzi online a offline verziou aplikácie. Po zvolení online verzie sa spraví overenie či je používateľ prihlásený alebo nie. Po tejto kontrole môžu nastať 2 scenáre. Prvým scenárom je, že žiak už je prihlásený. Vtedy sa stiahnu dáta prihláseného používateľa z DB a načíta sa nám scéna výberu triedy. V prípade, ak žiak nie je prihlásený zobrazí sa nám scéna prihlásiť. V prípade, že žiak nie je do aplikácie zaregistrovaný zvolí si možnosť registrácie, ktorá sa nachádza na scéne prihlásiť. Po registrácii sa užívateľovi (žiakovi) zobrazí možnosť vstúpiť do triedy v prípade že sa už nachádza v nejakej z tried, ak sa žiak nenachádza v žiadnej triede má možnosť zadať heslo triedy a automaticky vstúpi do triedy. Každé heslo triedy sa automaticky žiakovi uloží do DB medzi jeho triedy. Po vstúpení do triedy bude mať používateľ k dispozícii tú istú funkcionality ako neprihlásený, ktorá je rozšírená o blackboard (tabuľu) a zdieľanie úloh. Po otvorení blackboard sa žiakovi zobrazia všetky úlohy, ktoré zadal učiteľ ale aj úlohy, ktoré si zadávajú žiaci medzi sebou.

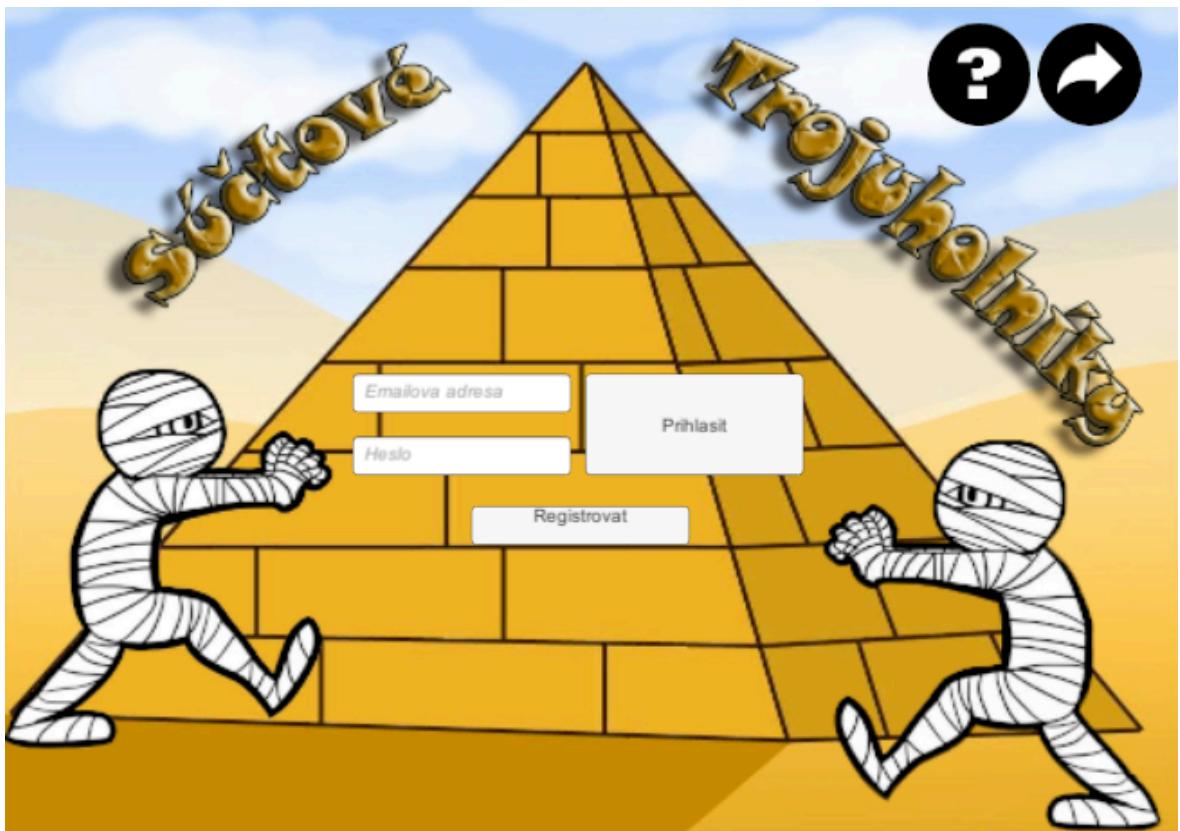
Druhou hlavnou časťou našej práce je funkcia Shared with ... (Zdieľať s ...), cieľom tejto funkcionality je aby žiaci svoje úlohy riešili spolu (Jedna úloha, ktorú rieši niekoľko žiakov na viacerých tabletoch). Hlavným cieľom u tejto funkcie bolo ju navrhnúť tak aby sme neprenásali obrovské dátá, nakoľko táto funkcia musí mať podporu real time. Funkcie Shared with ... ako aj blackboard sú funkčné pre žiakov len z rovnakých tried. V modelovom prípade Janko je v triede 1.A a Marienka v 1.A tito žiaci môžu medzi sebou zdieľať úlohy ale aj si pozrieť tabuľu, ktorú majú spoločnú. Druhým prípadom je že Janko bude v inej triede ako Marienka. V tomto prípade sa žiakom medzi sebou spojenie nadviazať nepodarí, keďže cieľom je naviazať komunikáciu v rámci tried a nie mimo nich.



Obrázok 6 Návrh komunikácie

4.3.1 Prihlásovanie a registrácia do aplikácie

Používateľ sa prihlásuje pomocou emailu a hesla, ktoré si sám zvolí v prípade že aplikáciu používa prvý krát, žiak sa pred prihlásením musí zaregistrovať. Po úspešnom prihlásení používateľ bude presmerovaný na scénu s výberom triedy. Prihlásenie na zariadení trvá dovtedy pokiaľ sa žiak z aplikácie sám neodhlási alebo neprebehne aktualizácia aplikácie. V prípade, ak prihlasovanie z nejakej príčiny zlyhá bude mu príčina oznamená formou informatívneho výpisu.

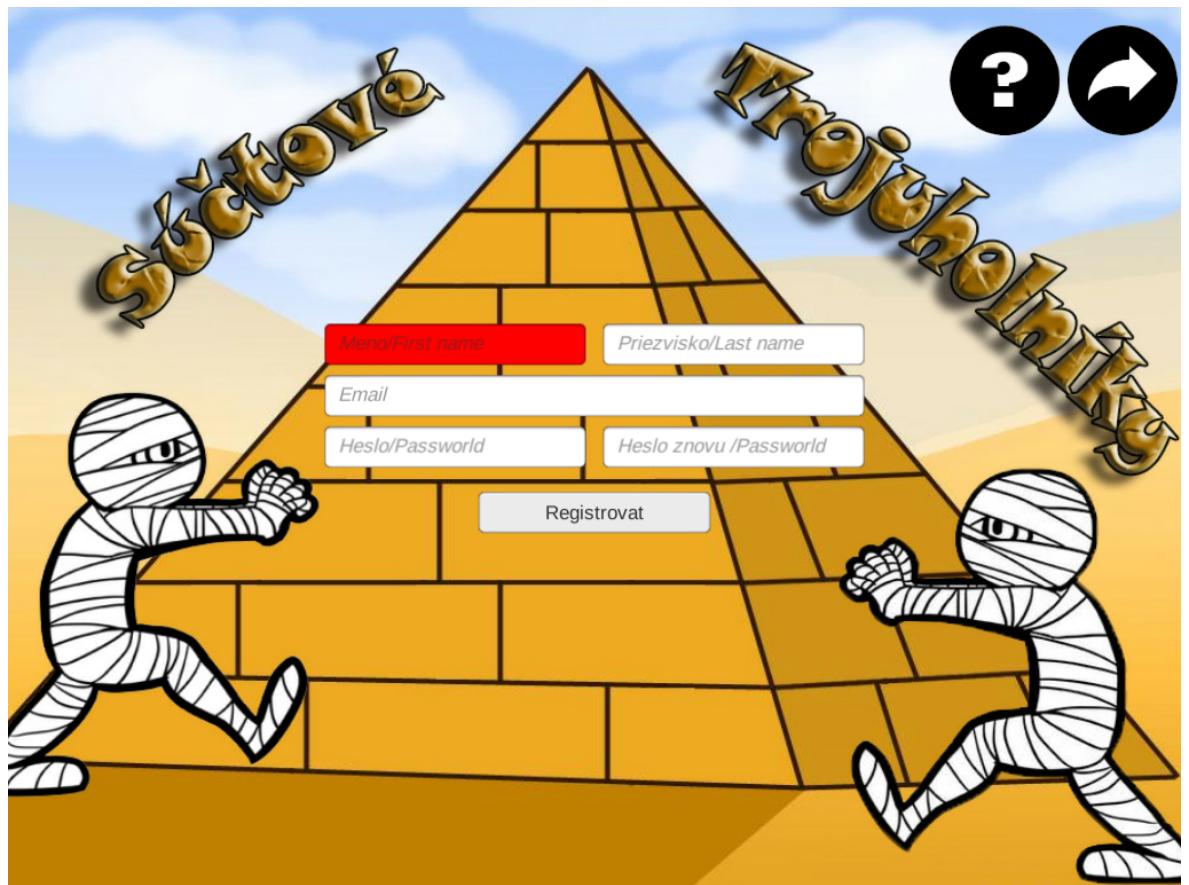


Obrázok 7 Prihlásenie

Nezaregistrovanému používateľovi sa zobrazí krátky formulár ktorý obsahuje :

- Email
- Meno
- Priezvisko
- Heslo

Po vyplnení formulára a odoslaní prebehne registrácia a po úspešnom zaregistrovaní bude žiak automaticky presmerovaný na scénu s výberom tried. V prípade, že žiak nesplní niektorú z požiadaviek políčko sa rozsvieti na červeno spolu s textom, v ktorom sú vypísané podmienky políčka.

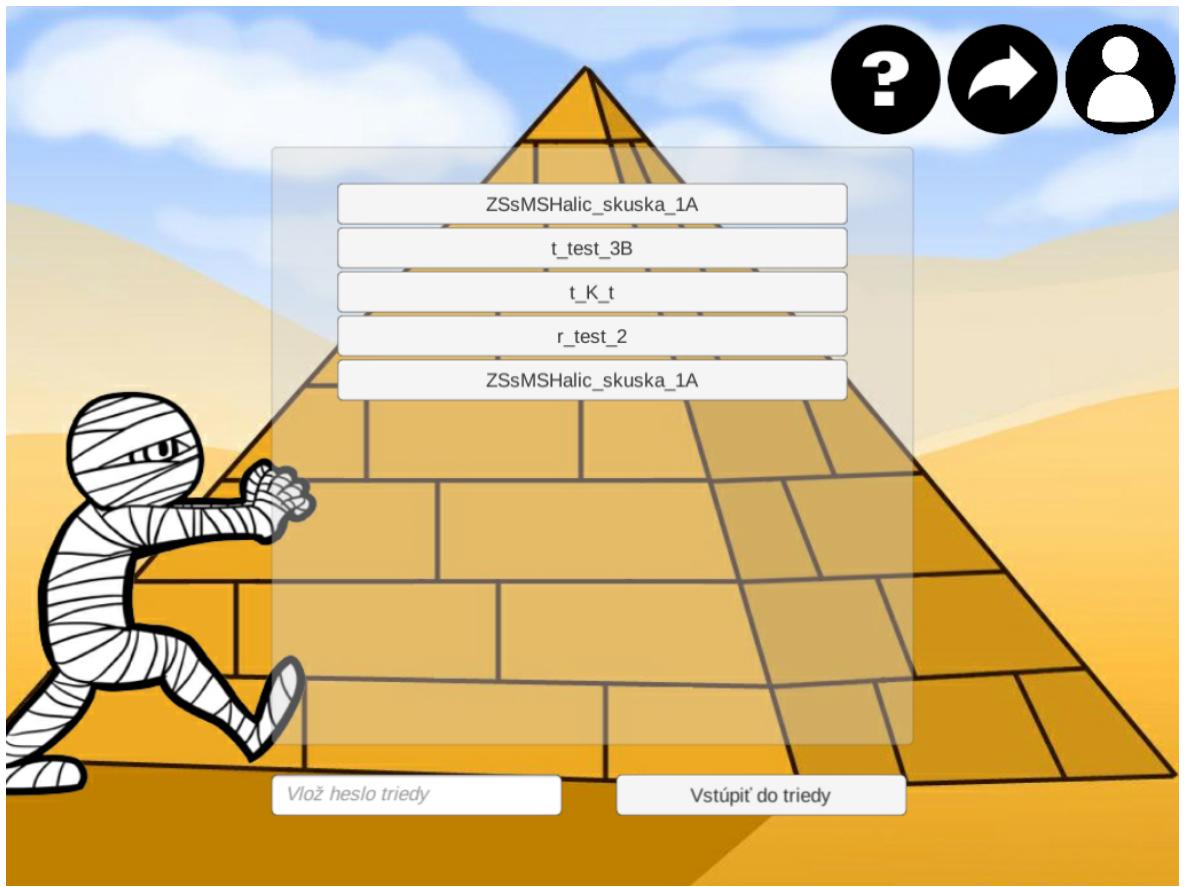


Obrázok 8 Registračný formulár s ukážkou chybného vstupu

4.3.2 Výber triedy

Scéna slúži na zvolenie triedy kliknutím na názov triedy v scrollbare, v ktorom bude zoznam tried vypísaný. Objekt obsahuje handler eventov , ktorý kontroluje či si žiak nepridal ďalšiu triedu v prípade, že áno doplní nám triedu do nášho scrollbaru. Tento handler počúva aj na zmenu názvu triedy prípadne zmazaniu. V prípade, že by učiteľ triedu premenoval zrovna v momente, keď si žiak volí triedu automaticky by sa mu názov triedy zmenil, v prípade odstránenie by trieda zmizla z ponuky.

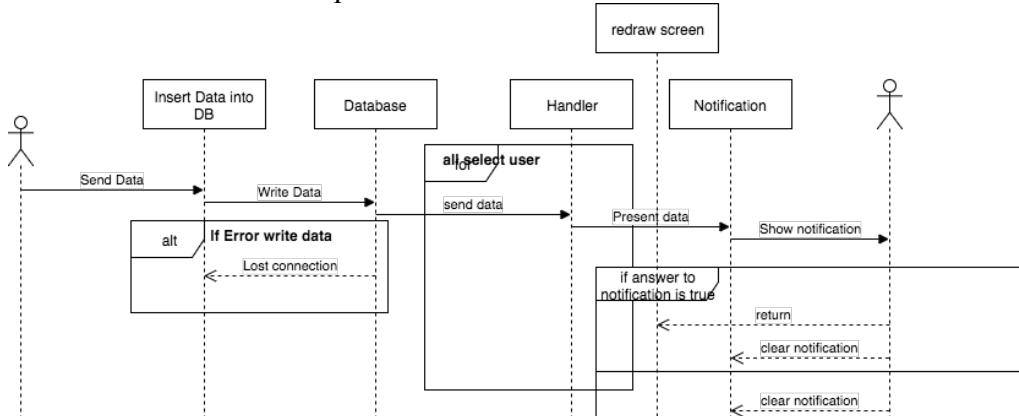
Druhou funkciou tejto scény je ukladanie tried. Funkcia funguje na základe hesla triedy. Žiak zadá heslo triedy, ktoré im učiteľ určí. Po zadaní hesla prebehne kontrola, či na základe hesla existuje takáto trieda, ak systém nájde danú triedu v databáze automaticky žiaka do tejto triedy priradí a zobrazí sa mu v zozname tried. V opačnom prípade notifikuje žiaka, že trieda s daným heslom neexistuje.



Obrázok 9 Scéna kde si žiak volí triedu do ktorej chce vstúpiť. V spodnej časti môžeme vidieť možnosť pridania triedy

4.3.3 Návrh zdieľať s ...

Jedna z hlavných funkcia platformy a aj najčažším orieškom na návrh. Ako som už spomíнал bolo potrebné dodržať určité dôležité podmienky ako sú real time, malý obsah prenášaných dát, nadviazanie spojenia a princíp lock/unlock obrazovky. Posledný spomenutý problém bolo najproblematickejšie vyriešiť nakoľko sa nám naskytalo niekoľko možností ako ísť na tento problém. Prvou možnosťou bolo lockovanie obrazoviek všetkým



Obrázok 10 Sekvenčný diagram nadviazania spojenia

neaktívnym používateľom. Čiže používateľ s označeným objektom by zablokoval prístup ostatným. V tomto spôsobe bol problém ten, že vždy by sa našiel žiak, ktorý by lockoval screen ostatným a znechutil im výuku. Druhou metódou bola rozšírená prvá metóda o časovač. Tento návrh bol založený na princípe klientskych časovačov, kde by sa spustil časovač pri označení objektu. V prípade, že by žiak do určitého času neumiestnil objekt bol by objekt od značený a obrazovka odblokovaná. Táto metóda bola lepšia, avšak vysvetliť deťom prečo sa im od značil objekt by bolo problematické. Metóda úplne neriešila blokovanie ostatných. Ďalšou metódou, ktorá prichádzala do úvahy bola metóda FIFO. Princíp bol vtom, že všetci mohli označiť ten rovnaký objekt, avšak všetkým sa objekt od značil po tom ako ho ten najrýchlejší umiestnil na novú pozíciu. Tento spôsob sa ukázal ako najvhodnejší na použitie. Či to bolo správne rozhodnutie sa uvidí pri prvom testovaní. Nadviazanie spojenia je zabezpečené pomocou handllera, ktorý počúva na objekte "waitForShare" v databáze. V tom momente ako nastane zmena v tomto objekte zobrazí sa žiadosť o nadviazanie spojenia. V momente potvrdenia žiadosti sa zosynchronizuje obrazovka s navrhovateľom. Týmto momentom je vytvorené spojenie medzi používateľmi až do odpojenia sa používateľa pripadne vyriešenia úlohy.

(scéna poslania requestu)

(scéna informovani o zdielani)

4.3.4 Návrh zdieľať – Tabuľa

(tabuľa)

Časť Tabuľa je ďalšou dôležitou funkciou našej platformy. Funkcia share with priamo spájala žiakov a nútla ich spolupracovať. Táto funkcia umožňuje podeliť sa so zaujímavými príkladmi v rámci triedy. Takže, ak žiak vymyslí náročný alebo zaujímavý príklad, prípadne mu ho vygeneruje generátor úloh bude sa môcť snim jednoducho podeliť zo svojimi spolužiakmi. V tejto časti bolo dôležité vhodne navrhnuť databázu nakoľko sa úlohy môžu opakovať. V rámci databázy ale aj tabule už priamo v programe by tým pádom vznikali zbytočné duplicity a tak by sme žiakov otravovali rovnakými úlohami viackrát. Druhým dôležitým krokom je vyriešiť označovanie úloh za vyriešené. Po vyriešení úlohy sa žiakovi príklad už nebude zobrazovať. Po vyriešení týchto úvodných funkcií bude tabuľa fungovať

takýmto scenárom. Žiak vytvorí príklad a pridá ho na tabuľu. V tom momente sa zobrazí na tabuli všetkým žiakom v rámci triedy.

(notifikacia)

O pridaní úloh na tabuľu nebudú žiakom chodiť notifikácie. Notifikácie by v tomto prípade mohli byť príliš časté a rušili by používateľov. Takto je na zvážení používateľa ako často si skontroluje tabuľu. Používateľ je informovaný formou výpisu koľko úloh pribudlo od posledného kontrolovania tabule.

Tabuľa je navrhnutá tak aby neslúžila len žiakom. Na tabuľu môže pridať úlohy aj učiteľ zo svojho webového rozhrania. Príklady pridané na tabuľu sú logované a učiteľ si na základe toho dokáže skontrolovať kto jeho úlohy riešil a s akou úspešnosťou ich riešil.

(diagram)

4.4 Návrh používateľského prostredia – webová aplikácia

Webová aplikácia je platforma navrhnutá výhradne pre učiteľov. Je to z dôvodu množstva funkcionality, ktorú bude tento web obsahovať. Pre túto technológiu sme sa rozhodli z dôvodu, že mobilná aplikácia je prispôsobená žiakom. Preto sme sa rozhodli oddeliť detskú hrajú časť platformy od tej pedagogickej časti avšak tieto 2 aplikácie sú prepojené spoločnou databázou. Webová časť obsahuje tieto funkcie:

- *Vytvorenie/editovanie triedy*
- *Pridanie úlohy na tabuľu*
- *Zobrazenie všetkých tried*
- *Zobrazenie žiakov v triedach*
- *Odobratie žiakov z tried*
- *Vytvorenie úlohy*
- *Zobrazenie štatistik žiaka*
- *Zobrazenie obrazovky žiaka*

Dizajn webovej aplikácie je rozdelený na 2 časti prvou časťou je navigácia. Navigácia je navrhnutá tak aby sme sa dostali na všetky podstránky ktoré obsahujú všetky vyššie uvedené funkcie. Po prihlásení sa používateľovi našej aplikácie zobrazí stránka obsahujúca zoznam všetkých tried, ktoré používateľ doposiaľ vytvoril. Táto stránka je zároveň nastavená ako domovská stránka. Druhou časťou je stredový panel. Zobrazujú sa v ňom detaile všetkých záznamov.

4.4.1 Návrh vytvorenie/editovanie triedy

Z dôvodu, že aplikáciu môžu používať žiaci z rôznych tried a škôl rozhodli sme sa vytvoriť množinu tried. Tieto triedy slúžia učiteľom na začlenenie si žiakov do tried. Každý z učiteľov si vytvorí triedu. Na vytvorenie triedy má v svojom rozhraní pripravený krátky formulár. Ktorý obsahuje len názov triedy a heslo. Heslo triedy musí byť jedinečné aby zariadenia žiakov vedeli na základe hesla identifikovať triedu. Po vyplnení formulára a odoslaní sa trieda automaticky zobrazí v DB a je prístupná žiakom. Žiaci na vstup do danej triedy potrebujú vedieť vstupný klúč, ktorý je špecifický. Učiteľ môže názov a heslo triedy kedykoľvek editovať. Editovanie hesla a názvu žiakov neobmedzí. Žiaci nachádzajúci sa v triede nové heslo zadávať nemusia. Týkať sa to bude len nových žiakov.

4.4.2 Vytvorenie úlohy

Funkcia vytvorenie úlohy patrí medzi základné úlohy webovej aplikácie. Vytváranie sme navrhli nasledovne. Vytváranie úlohy je rozdelené na dve časti v prvej časti je potrebné pomenovať úlohu. Bez zadaného názvu nie je možné úlohu uložiť. V ďalšom kroku si používateľ zvolí možnosť pridať príklad a dostane ponuku troch šablón. Po zvolení vhodnej šablóny sa mu zvolená šablóna vykreslí. Do prázdnych políčok sa hodnoty dopĺňajú nasledovne. Používateľ klikne na políčko ktorému chce zadať hodnotu. Po kliknutí sa mu zobrazí zoznam hodnôt, ktoré môže doplniť. Tlačidlo *Potvrdiť príklad* slúži na pridanie príkladu do pripravovanej úlohy a tlačilo *zrušiť* len zruší aktuálny príklad. Po potvrdení príkladu sa nám príklad zobrazí v danej úlohe vo formáte JSON odkiaľ je možné vyčítať aká hodnota sa na akom mieste nachádza prípadne je ešte možné príklad odstrániť. Funkciu pridať príklad môžeme opakovať neobmedzene pokial' sa nerozhodneme, že úloha je hotová. Po použití tlačidla uložiť úlohu sa úloha zapíše do databázy a o úspešnosti nás aplikácia informuje výpisom. A presmerovaním medzi zoznam úloh.

(obrazok vytváranie úlohy)

4.4.3 Pridanie úloh na tabuľu

Na to aby sa žiaci dostali k úlohám, ktoré sme si pre nich vytvorili je potrebné ich priradiť žiakom na tabuľu. Túto procedúru sme navrhli pomocou dvoch modálnych okien. V jednom z okien sa zobrazujú úlohy priradené danej triede a v druhom okne sa nachádzajú úlohy ktoré je možné pridieť. Modálne okná sa zobrazia po zvolení možnosti *Priradit' úlohu* alebo *Priradené úlohy*. Tieto tlačidlá sú umiestnené na domovskej obrazovke a každá trieda ma svoju dvojicu týchto tlačidiel.

4.4.4 Zobrazenie obrazovky žiaka

Podstránka je navrhnutá len ako textová stránka, stránka neobsahuje žiadnu funkciu. Do hornej časti stránky sme umiestnili informáciu o tom koho obrazovku si prezeráme. Pod menom je umiestnený dátum a čas poslednej zmeny obrazovky. Pod týmto informatívnym blokom sa nachádza samotná obrazovka v takom stave ako ju žiak má. Prípadne sa tam nachádza posledný stav jeho obrazovky. Zmeny na tejto obrazovke sa prejavujú bez použitia možnosti obnovenia stránky.

(diagram pripojenie k databaze vytvorenie even hadlera)

4.4.5 Zobrazenie štatistik

Vo webovej aplikácii pracujeme s dvoma typmi štatistik. Prvým typom sú štatistiky žiaka a tým druhým sú štatistiky triedy. Zobrazovanie štatistik sme preto navrhli ako dve podstránky našej aplikácie. Prvá obsahuje zoznam všetkých žiakov v triede a informatívny výpis pri každom žiакovi o tom koľko príkladov vyráhal správne a koľko nesprávne. Nad týmto zoznamom sa ešte nachádzajú celkové súhrny koľko úloh žiaci vypočítaných celkovo, koľko z toho bolo vypočítaných správne a koľko nesprávne.

Druhá podstránka ku ktorej sa dostaneme po kliknutí možnosti zobraz štatistiky ale nie nad triedou ale nad žiakom zobrazuje štatistiky žiaka v hornej časti sa opäť nachádza súhrn

o tom koľko úloh vypočítal správne nesprávne a celkovo. Pod týmto súhrnom sa nachádza zoznam príkladov, ktoré učiteľ pridal na tabuľu aj s výsledkami koľko z nich vypočítal a koľko chýb pri ich počítaní urobil.

(obrázok štatistik)

4.5 Návrh databázy

Pri návrhu databázy sme museli klásiť dôraz na jednoduchý prístup k dátam. Z dôvodu že sme nepoužili SQL databázu sme a tým pádom nemáme k dispozícii operácie ako join,group by atď. Z tohoto dôvodu sme si museli premysliť, ku ktorým dátam budeme pristupovať často a teda bude rozumné nevnárať ich hlboko do JSON štruktúry.

Databázu sme rozdelili do siedmich základných objektov. Sú to:

- *CLASSES (triedy)*
- *EXAMS (úlohy)*
- *EXAMS_PINNED_TO_TABLE (úlohy pripnuté na tabuľu)*
- *SHARED_SCREEN (zdieľané obrazovky)*
- *STATISTICS (štatistiky)*
- *TABLES (tabule)*
- *USERS (používateelia)*

4.5.1 Triedy

V objekte triedy sa nachádzajú všetky triedy všetkých používateľov. Každá trieda má svoje jedinečné ID. Podľa tohto ID vieme jednoznačne identifikovať objekt v databáze. Už konkrétny objekt Trieda obsahuje dva zoznamy. V prvom zozname sa nachádzajú všetci žiaci priradený k triede. V druhom zozname máme žiakov, ktorí sú aktuálne online. Objekt trieda ešte obsahuje meno a heslo triedy a ID používateľa ktorý objekt vytvoril.

4.5.2 Úlohy

Objekt úlohy obsahuje zoznam všetkých úloh. V objekte sa nachádza názov úlohy, zoznam všetkých príkladov, ktoré úloha obsahuje a id používateľa, ktorý úlohu vytvoril.

4.5.3 Úlohy pripnuté na tabuľu a zdieľané obrazovky

Návrh tohto objektu bol náročnejší nakoľko sme si do tohto objektu museli zapamätať celé nastavenie danej obrazovky. Ide o objekty, ktoré nám slúžia k zostrojeniu identickej scény ako mal predchodca. Ďalej sa v objekte nachádza názov úlohy a meno používateľa ktorý úlohu vytvoril.

Objekt zdieľané obrazovky je navrhnutý podobne ako objekt úlohy pripnuté na tabuľu. Zdieľané obrazovky majú naviac zoznam id používateľov, ktorý medzi sebou zdieľajú obrazovku.

4.5.4 Štatistiky

Objekt je navrhnutý ako zoznam ID tried, ktorý obsahuje zoznam ID žiakov a hodnoty správne a nesprávne. Týmto spôsobom sa vyvarujeme duplicit dát v databáze. Výhodou je aj prístup k dátam nakoľko pri vnorení dát do objektu triedy alebo používateľ by sme sa k dátam dostávali náročnejšie a museli by sme použiť viac cyklov pre konkrétné dátu. Medzi výhody by som zaradil aj šetrenie dátami a tým pádom aj rýchlejšej odpovede zo strany serveru.

4.5.5 Tabule

Tento objekt je podobný ako objekt štatistiky obsahuje len zoznam ID tried a pod každým ID triedy sa nachádza zoznam ID príkladov pripnutých na tabuľu a číselnej hodnoty. Táto hodnota nám prezrádza koľko príkladov daná úloha obsahuje.

4.5.6 Používatelia

Návrh tohto objektu bol veľmi náročný nakoľko sme museli vhodne rozhodnúť, ktoré dátu ponecháme ešte v rámci objektu a ktoré už vytiahneme do samotného objektu. Nakoniec sme objekt zostrojili ako objekt, ktorý obsahuje meno, priezvisko, aktuálne zvolenú triedu, email a potom štyri objekty. Prvým je aktuálna obrazovka – tento objekt zaznamenáva zmeny pri riešení úloh. Druhým je objekt Moje triedy. V objekte sa nachádza zoznam pridaných tried do mobilnej aplikácie. Tretím objektom sú vyriešené úlohy. Ide o úlohy nachádzajúce sa na tabuli. V poslednom objekte sa nachádza

informácia o tom koľko úloh pribudlo na tabuli od poslednej kontroly tabule. Keďže pre každú triedu máme inú tabuľu tento objekt obsahuje zoznam tried v ktorom je nasledujúca informácia uložená.

5 Implementácia

V kapitole implementácia si popíšeme spôsob integrácie technológií do našej aplikácie. Následný vývoj potrebných modulov a funkcií. Základnú štruktúru programu a problémy, ktoré nás pri implementácii postretli.

5.1 Integrácia s Firebase

Prvým krokom, ktorý sme museli pri implementácii našej aplikácie urobiť bolo na integrovať technológiu firebase do našich aplikácií. Možnosti integrácie firebase sme si mohli pozrieť v dokumentácii služby[].

(webove prostredie firebase)

V prvom kroku sme si museli vytvoriť projekt v službe firebase. Na vytvorenie projektu bolo potrebné uviesť názov projektu a krajinu. Na základe názvu projektu nám je vytvorené špecifické Id. Po vytvorení projektu je potrebné otvoriť nastavenia. V nastaveniach v záložke general si zvolíme platformu pre ktorú chceme stiahnuť konfiguračný JSON súbor. V našom prípade sme si stiahli konfiguračné súbory oboch mobilných platform. Ďalším krokom pri integrácii bolo stiahnutie API rozhrania pre Unity a nakopírovať ho spolu aj s konfiguračnými súbormi do projektu. Posledným krokom bolo naimportovanie API do nášho Unity projektu.

(ukazka konfiguracnych suborov)

Integrácia do webového rozhrania bola jednoduchšia. Pomocou príkazu `$ npm install angularfire2 firebase --save` sa nám do webovej aplikácie stiahlo a nainštalovalo API a už bolo potrebné len vyplniť hodnoty ako apiKey a databaseURL v konfiguračnom súbore.

V prípade integrácie pre JavaScript stačí do kódu nakopírovať konfiguráciu, ktorá sa nachádza v dokumentácii služby.

(obrazok kodu integrácie pre JS)

5.2 Vytvorenie C# modulu pre mobilnú aplikáciu

Nasledujúcej kapitole si rozoberieme ako sme postupovali pri implementácii C# modulu. Modul nám slúži na komunikáciu medzi frontendovou časťou aplikácie a databázou. Takže môžeme povedať, že modul tvorí časť backendu aplikácie. Do modulu bolo potrebné nainštalovať firebase knižnice. Knižnice vytvárajú spojenie medzi databázou a našou aplikáciou. Následne sme si implementovali funkcie, ktoré sú spoločné pre celú aplikáciu aby sme zabránili vzniku duplicitných častí v kódu. Do modulu sme preto implementovali funkcie na zápis a čítanie z databázy.

5.3 Úlohy na tabuli

Z dôvodu, že v predchádzajúcej verzii aplikácie sme tabuľu nemali. Museli sme si pripraviť do scény template, na ktorom sa budú zobrazovať dátá z databázy. Po upravení scény pre naše potreby sme sa mohli pustiť do programovania backendu aplikácie.

5.3.1 Pridanie príkladu na tabuľu

Pridávanie úlohy na tabuľu prebieha nasledovne. Po stlacení tlačidla pripnúť sa zavolá metóda *PripniPríkladNaTabulu*. Táto metóda vytvorí novú inštanciu dátového modelu a naplní ju. V ďalšom kroku vytvorí identifikačný kľúč a úlohu vloží do databázy. Po zapísaní do databázy zavolá metódu *InfoAboutPin*. Táto metóda informuje používateľa o uspešnosti pridania úlohy a pod akým názvom bola pridaná. Opäťovné volanie metódy *PripniPríkladNaTabulu* úlohu nepridáva duplicitne ale len aktualizuje predošlý stav a o úspešnosti nás opäť informuje metóda *InfoAboutPin*.

5.3.2 Zobrazenie úloh na tabuli

Na zobrazenie úloh na tabuľu nám slúži metóda *showExamsOnBoardAdd*. Metóda v sebe obsahuje viacero vnorených asynchronných volaní a je implementovaná nasledovne. V prvom volaní si z databázy vyžiadame dátá používateľa, ktorý úlohu vytvoril. V prípade, že request skončí úspešne vyžiadame si všetky príklady pripnuté na tabuľu. Po úspešnom druhom volaní vytriedime príklady, ktoré sa zobrazí nemajú a zavolá sa tretie volanie. Toto volanie nám vráti v akom stave riešenia je úloha a zavolá metódu *generateExamToogleList*,

ktorá následne príklad vykreslí na tabuľu. Celá metóda je vytvorená tak aby sa mohla vykonávať v reálnom čase. To znamená že príklady na tabuli sa prekreslujú hned' pri pridaní a nie je potrebné tabuľu obnoviť.

5.4 Zdieľanie úloh

5.4.1 Odoslanie požiadavky

Implementácia zdieľania úloh sa skladala z viacerých častí. Celé sa to začína zobrazením zoznamu používateľov v triede. Na zobrazenie používateľov sme museli naprogramovať metódu *HandleShowAllUserInClassAdd*, ktorej úlohou je získať všetkých prihlásených používateľov v triede. Metóda po prijatí výsledku spracuje dátu a spustí metódu *generateStudentToogleList*. Táto metóda dostáva ako vstupné parametre, kľúč a hodnotu a vykreslí riadok s používateľom. Metóda *HandleShowAllUserInClassAdd* je implementovaná tak, že pri prvom spustení sa vykoná toľko krát kolko záznamov je v databáze a následne už len prepisuje zmeny. Ďalším krokom je odoslanie požiadavky používateľom. Na túto úlohu sme implementovali metódu *ShareScreenWith*. Metóda vytvorí inštanciu dátového modelu a naplní ju potrebnými hodnotami a zapíše ich do databázy. Následne vytvorí zoznam používateľov, s ktorými chceme dátu zdieľať a zavolá metódu *SendRequest* so vstupnými parametrami zoznam používateľov ktorým úlohu posielame a id zdieľanej úlohy. Metóda *SendRequest* rozpošle používateľom požiadavky na zdieľanie.

5.4.2 Prijatie požiadavky

Prijatie požiadavky nám zastrešuje metóda *HandleControlRequestSharedScreen*. Metóda má za úlohu kontrolovať či sa v databáze na objekte *waitForShare* nenastala zmena. Z našej implementácií ide o pribudnutie hodnoty. V prípade že metóda zaznamená zmenu, prijaté dátu spracuje a zobrazí používateľovi informáciu o žiadosti, s možnosťou výberu prijatia žiadosti alebo odmietnutia. Prijatie žiadosti spracováva metóda *AcceptShareScreen*. Metóda prijme dátu, naplní ich do dátového modelu a otvorí vhodnú scénu. Odmietnutie žiadosti má nestarosť metóda *MissedShareScreen*. Táto metóda má za úlohu odstrániť žiadosť z databázy a zatvoriť informáciu o zdieľaní.

5.4.3 Zaznamenávanie t'ahov

V nasledujúcich podkapitolách sme si rozobrali implementáciu Odoslania požiadavky aj Prijatie no ešte stále sa nám zmeny neprejavujú. Na zaznamenávanie t'ahov nám slúžia metódy *HandleChildChanged* a *UpdateResult*. *HandleChildChanged* slúži na kontrolu či niektorí zo žiakov nespravil t'ah. Metóda dostáva hodnoty ako kľúč slotu, v ktorom nastala zmena a hodnota aká sa tam nachádza. Metóda hodnoty vloží do modelu a prekreslí obrazovku používateľa. Metóda *UpdateResult* sa vykonáva vždy po našom t'ahu. Metóda zaznamená násťah do databázy. Po vykonaní metódy *UpdateResult* sa následne na všetkých tabletcoch s ktorými máme spojenie spustí metóda *HandleChildChanged*.

5.5 Webové rozhranie

Na implementáciu webového rozhrania sme zvolili technológiu Nette. Ide o PHP framework.

(obrázok štruktúry projektu po naistalovaní)

V prvom kroku sme si pripravili šablóny, ktoré nám slúžia na vykreslovanie získaných dát z DB. Na prípravu šablón nám framework poskytuje vlastný šablónovací template nazývaný Latte. Šablony v template Latte sme pripravovali v jazyku HTML a CSS. Príprava šablón spolu aj s dizajnom stránky (CSS) nenasvedčovali tomu, že by sme mohli mať v ďalších krokoch problémy.

(obrázok ukazka kodu prepojenia sablon s backendom).

Druhým krokom bolo vytvorenie registrácie a prihlásovanie na backendovej strane stránky a tu sme začali narážať na problémy. Pri implementácii sa nám nepodarilo integrovať do projektu PHP API na podporu Firebase. Z tohto dôvodu sme museli integrovať JavaScriptové API. JavaScriptové API nám v projekte bezchybne fungovalo avšak sme prišli o potenciál framewroku Nette nakoľko sme úplne obchádzali backend frameworku. Všetka funkciaľita okrem routovania stránok, bola naprogramovaná v JavaScripte a za pomocí technológie JQuery dopĺňaná do Latte šablón. Z tohto dôvodu sme sa rozhodli prehodnotiť výber technológií. Na začiatku sme sa rozhodovali medzi technológiami Nette, Laravel,

Spring MVC, Angular 2. Nakoľko PHP API pre Firebase nebolo funkčné, prvé dva frameworky z výberu vypadli. Angular 2 mal menšie serverové požiadavky a podobnú štruktúru ako predchádzajúci framework Nette aj z tohoto dôvodu sme sa nakoniec rozhodli Angular.

Na nainštalovanie frameworku sme si museli ako prvé nainštalovať program node.js. Node.js sme nainštalovali na macOS za pomoci príkazu *brew install node*. Po nainštalovaní node.js sme mohli pristúpiť k samotnej inštalácii Angularu. Angular sme nainštalovali pomocou príkazu *npm install -g @angular/cli*. Po nainštalovaní všetkých potrebných technológií sme mohli pristúpiť k vytvoreniu projektu.

(obrazok seria prikazov na vytvorenie projektu)

Po vytvorení projektu bolo potrebné integrovať API pre Firebase. V prvom kroku sme museli pomocou príkazu *npm install angularfire2 firebase --save* stiahnuť a nainštalovať knižnicu.

Štruktúra projektu bola veľmi podobná projektu vytvorenému v Nette. Z tohoto dôvodu sme mohli použiť šablóny zo starého projektu. V šablónach sme museli vykonať drobné zmeny. Zmeny sa týkali prevažne zobrazovania dát. CSS štýly sme automaticky mohli zachovať tie zo starého projektu.

(obrázok štruktúry projektu angular)

Na to aby sme do projektu mohli vložiť šablony zo starého projektu museli sme vytvoriť ku každej šablone komponentu. Komponent sa v angulari skladá zo štyroch súborov.

- *page.component.css* – v tomto súbore sú umiestnené CSS štýly špecifické iba pre túto jednu komponentu v našom prípade je súbor prázdna nakoľko máme globálne CSS štýly pre celý projekt a nie len pre jednu komponentu.
- *page.component.html* – obsahuje šablónu stránky, ktorá sa renderuje spolu s CSS.
- *page.component.spec.ts* – slúži na automatizované testovanie.
- *page.component.ts* – je controller našej komponenty.

5.5.1 Prihlásenie a registrácia

Pri implementácii sme si vytvorili komponentu *login-page* a *registration-page*. Na prihlásenie nám slúži komponenta *login-page*, ktorá obsahuje triedu *LoginPageComponent*. Po načítaní stránky sa nám spustí konštruktor triedy *LoginPageComponent* v ktorom sa vykoná kontrola či nie sme v systéme prihlásený. V prípade úspešnej kontroly sa nám načíta stránka prihlásenia, ak kontrola skončí neúspešne znamená to, že používateľ je prihlásený a presmeruje ho na domovskú obrazovku.

Prihlásenie zabezpečuje metóda *login*. Metóda sa zavolá použitím tlačidla prihlásiť a na vstup dostane prihlasovací email a heslo. Metóda odošle prihlasovacie údaje na server, ktorý údaje spracuje a následne nám vráti používateľa alebo chybu. Na základe chyby je používateľ informovaný o probléme, ktorý nastal pri prihlásovaní. V prípade úspešného prihlásenia je používateľ presmerovaný na domovskú stránku.

Podobne ako pri prihlásovaní aj pri registrácii sa vykoná kontrola na zistenie či používateľ nie je prihlásený a následne sa mu zobrazí regisračný formulár. Samotnú registráciu zastrešuje metóda *register*. Vstupnými parametrami do metódy sú email,heslo,meno a priezvisko. Najskôr sa skontrolujú všetky povinné hodnoty na strane serveru a v prípade ,že kontrola prebehla úspešne sa zavolá metóda *registerUserWithEmailAndPassword*. Táto metóda vykoná samotnú registráciu a zapíše dátá do databázy. O úspešnosti nás následne service informuje rovnakým spôsobom ako pri prihlásovaní.

5.5.2 Vytvoriť úlohu

Vytváranie úloh zastrešuje komponenta *CreateExamPageComponent* a funguje nasledovne. V konštruktore sa nám vygeneruje zoznam možností ktoré môžeme dosádzať do prázdných polí. Po stlačení tlačidla pridať príklad sa nám nastaví prázdna šablóna. Do šablony sa dosadzujú hodnoty pomocou metódy *appendNumberIntoBox*. Táto metóda nám pridáva hodnoty z modálneho okna do prázdnych boxov v šablóne príkladu. Po vyplnení všetkých políčok v príklade a stlačení tlačidla potvrď príklad prebehne metóda *submitTask*. Metóda skontroluje či sú všetky hodnoty vyplnené a následne vloží príklad do dátového modelu. Pridanie príkladov je možne opakovať. Po stlačení tlačila uložiť sa zavolá metóda

saveExam. Metóda úlohu odošle na server, ktorý úlohu zapíše do databázy a vráti request o úspešnosti akcie.

(obrazok backendu privavanie hodnoty do sablony)

5.5.3 Zoznam tried a mojich úloh

Implementácia zoznamu tried a mojich úloh je veľmi podobná. V konštruktore triedy sa vykoná metóda *zobrazTriedy()*. Metóda nám vracia zoznam všetkých tried, ktoré obsahujú *TeacherID* rovnaké ako prihlásený používateľ.

5.5.4 Priradenie úloh ku triedam

Po vygenerovaní tried môžeme vidieť pri každej z tried tlačidlá ako *Prirad' úlohu* alebo *Priradené úlohy*. Po stlačení prvého spomenutého tlačidla sa spustí metóda *zobrazZoznamUlohNaPridanie*. Metóda vygeneruje do modálneho okna všetky úlohy, ktoré prihlásený používateľ vytvoril a ešte nie sú priradené triede. Druhé tlačilo zavolá metódu *zobrazZoznamPriradenychUloh*. Táto metóda otvorí modálne okno a vygeneruje zoznam všetkých priradených úloh triede. Metódy sú medzi sebou prepojené a tak ak úlohu odoberieme z jedného zoznamu automaticky sa nám bude zobrazovať v tom druhom a naopak.

6 Testovanie

V kapitole testovanie sa budeme zaoberať testovaním aplikácie v praxi. V prvej časti testovania si prejdeme, ktoré parametre sme sledovali pri testovaní. Kde testovanie prebiehalo, ako testovanie prebiehalo a zhodnotenie testovania.

6.1 Prvé testovanie

Prvé testovanie prebehlo 12.4.2018 v Cirkevnej základnej škole Narnia. Cieľom testovania bolo preveriť aplikáciu v praxi jej chovanie ale aj reakcie žiakov na interface aplikácie.

Počas testovaní sme sledovali:

- reakcie žiakov na aplikáciu
- pochopenie ovládacích prvkov
- reakcie aplikácie na rôzne podnety
- odozvu aplikácie pri pripojení viacerých zariadení
- využívanie možnosti spolupracovať na úlohách

6.1.1 Priebeh testovania

Testovanie prebiehalo za prítomnosti školského psychológa, ktorý priebeh testovania aplikácie sledoval spolu s nami.

Na začiatku testovania sme žiakov oboznámili s aplikáciou. Nakol'ko žiaci prostredie súčtových trojuholníkov poznali nebolo potrebné im toto prostredie vysvetľovať a tak sa s radosťou pustili do testovania. Prvá štvrtina testovania pôsobila opatrným dojom, kde si každý riešil úlohy sám. Bolo to spôsobené hlavne spoznávaním sa s aplikáciou a túžbou žiakov sa zahráť. Neskôr sa už začali informovať o tom kto má akú úlohu. Toto bol podnet pre nás nabádať ich objavovať ďalšie možnosti aplikácie. No v prvej polovici testovania naše podnete niekol'ko krát stroskotali. Zhruba v polovici testovania sa nám naskytla modelová situácia využitia zdieľania úloh. Žiačka si nevedela poradiť s vyriešením vygenerovanej úlohy a tak sa na nás obrátila. My sme jej navrhli skúsiť použiť funkciu zdieľania úlohy

s ostatnými spolužiakmi. Tu nastal zlom. Po zobrazení prvej žiadosti o spoluprácu sa žiaci začali informovať o tejto funkciionalite a následne ju vo veľkom využívať. Tu vznikla chvíľková hystéria. Žiaci sa začali prekrikovať ktori s kym rieši úlohy a tak to vyzeralo do konca testovania. Na konci testovania sme sa každého žiaka spýtali ako sa im aplikácia páčila. Čo by sme na aplikácii mali zmeniť a naopak čo by malo ostať zachované.

(Fotka z testovania)

6.1.2 Zhrnutie testovania

Prvé testovanie môžeme označiť za úspešné. Žiaci veľmi rýchlo pochopili základné ovládacie prvky aplikácie. K použitiu ovládacích prvkov na zdieľanie alebo pripnutie úloh na tabuľu ich bolo treba nabádať, no po zoznámení sa s funkciunalitou ju s radosťou využívali. Reakcie aplikácie boli presne podľa špecifikácie a ničím nás neprekvapila. Odozva aplikácie bola rýchla a nijakým spôsobom neznemožňovala používanie. Jediný problém, na ktorý sme pri testovaní narazili bolo využívanie mobilnej siete. Sieť sa behom testovania niekoľko krát preťažila a niektorých žiakov z aplikácie odpojila. Druhým problémom, ktorý spôsobila sieť bolo občasné nestiahnutie zdieľanej obrazovky. Testovanie nám ale ukázalo problémy, ktoré sa nám v laboratórnych podmienkach nepodarilo nasimulovať. Ďalej sme dostali názory žiakov a pedagóga vykonávajúceho dozor. Počas testovania žiaci vypočítali 164 príkladov a z toho 123 úloh vypočítali správne.

6.2 Druhé testovanie

Záver

Literatúra a internetové zdroje

- [1] Daniel Linhart, (2016) Softvérová podpora vyučovania matematiky Hejného metódou - prostredie Súčtové trojuholníky, Bakalárská práca, Univerzita Komenského v Bratislave, Fakulta Matematiky, Fyziky a Informatiky
- [2] H-mat, 12 klíčových principů, [cit. 13.4.2018]
Dostupné na <https://www.h-mat.cz/principy>
- [3] Ing. Edita Schima, Mgr. Jaroslava Urbánková, (2014) Nové formy skupinového vyučovania, Metodicko-pedagogické centrum v Bratislave,
ISBN 978-80-565-0206-8
- [4] Ondřej Žára, Firebase: krátké seznámení – Zdroják, [cit. 19.5.2017],
Dostupné na <https://www.zdrojak.cz/clanky/firebase-kratke-seznameni>
- [5] Google, Firebase Realtime Database, [cit. 19.5.2017]
Dostupné na <https://firebase.google.com/docs/database/>
- [6] Google, Firebase Authentication, [cit. 19.5.2017]
Dostupné na <https://firebase.google.com/docs/auth>
- [7] Pete Warden, (2011) Big Data Glossary, O'Reilly Media, Inc.,
ISBN 978-1- 449-31459-0
- [8] Jana Vyháliková, (2013) Informačné systémy s využitím NoSQL databáz,
Bakalárská práca, Bankovní institut vysoká škola Praha zahraničná vysoká škola
Banská Bystrica
- [9] Nette Foundation, Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework, [cit. 14.5.2018]
Dostupné na <https://nette.org/cs>
- [10] Aleš Dostál, Vývoj webových aplikací: React a Angular 2, [cit. 14.4.2018],
Dostupné na <http://www.aspectworks.com/2016/09/vyvoj-aplikaci-react-angular-2/>
- [11] Nimesh Chhetri, (2016) A Comparative Analysis of Node.js (Server Side JavaScript), Saint Cloud State University.
- [12] Sebastian Eschweiler, Angular 2 + Firebase Introduction – CodingTheSmartWay.com Blog – Medium , [cit. 20.4.2018], Dostupné na <https://medium.com/codingthesmartway-com-blog/angular-2-firebase-introduction-b4f32e844db2>

- [1] Marek, DIRNER Alexander–DEMKO Jozef–DOMARACKÝ, FRANKO František–HLAVÁČOVÁ Júlia–MARTINSKÁ Gabriela, and MURÍN Pavol–Šerý Michal. "VIRTUÁLNA KOLABORÁCIA. HĽADANIE NOVÝCH FORIEM VZDELÁVANIA S VYUŽITÍM NOVÝCH IKT V ROZVOJI FYZIKÁLNEHO POVEDOMIA MLÁDEŽE."
- [2] Dirner, A., et al. "Virtuálna kolaborácia. Využitie nových informačno-komunikačných technológií vo výučbe fyziky."
- [3] DEPEŠOVÁ, Jana. "Virtual communication in educational system." Journal of technology and information (2013).
- [4] Dirner, Alexander, et al. "2. Virtuálna kolaborácia."

Prílohy