

**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



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 navrhnuť 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 viacerým 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 aplikovanej 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 04.05.2018

Meno a priezvisko

Pod'akovanie

Tento cestou by som rád pod'akoval môjmu školiteľovi, RNDr. Petrovi Borovanskému, PhD., za jeho čas, cenné rady a prípomienky. Rovnako rád by som pod'akoval aj Cirkevnej základnej škole Narnia Bratislava za pomoc pri testovaní diplomovej práce.

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

Spolupráca a komunikácia sú základné pravidlá Hejného metódy, avšak väčšina edukačných aplikácií tieto pravidlá nedodržiava. Preto sme sa rozhodli vytvoriť aplikáciu na výučbu matematiky s využitím Hejného metódy. V našej aplikácii kladieme dôraz na komunikáciu medzi žiakmi a ich spoluprácu s učiteľmi, pričom výučba prebieha na tabletoch. Naša práca poukazuje na to, aká je dôležitá spolupráca a na čo sa je potrebné zamerať pri vývoji podobnej aplikácie. Výsledkom našej práce je aplikácia umožňujúca zdieľať úlohy medzi žiakmi, pridávať úlohy na virtuálnu tabuľu a kontrolovať aktivitu žiakov učiteľom. Aplikácia umožňuje zaraďovať žiakov do tried, sledovať štatistiky, či pridávať úlohy triede. Platforma je založená na komunikácii v reálnom čase s použitím technológie Firebase.

Kľúčové slová: Hejného metóda, platforma, vyučovanie matematiky, kolaboratívne vyučovanie

Abstract

LINHART, Daniel: *Collaborative platform for Hejny's math teaching method (Diploma thesis)* - Comenius University in Bratislava, Faculty of mathematics, physics and informatics; Department of applied informatics - Adviser: RNDr. Peter Borovanský, PhD.: FMFI UK, 2018

Collaboration and communication are fundamental principles of Hejny's method, however most of existing educational applications don't follow these principles. Therefore we have decided to create application for teaching math with using Hejny's method. In our application we focus on communication between pupils and their collaboration with teachers, while tablets are used for teaching. Our work points to importance of collaboration and what we need to focus on developing a similar application. Result of our work is an application which allow sharing exercises, adding exercises to the virtual board and control of pupil activity by teachers. Application allows assign pupils to the classes, observe statistic or add exercises to class. Platform is based on real-time communication with using Firebase technology.

Key words: Hejny method, platform, learning mathematics, teaching mathematics, collaborative teaching

Obsah

1	Úvod.....	13
1.1	Motivácia a analýza problematiky.....	13
1.2	Cieľ práce.....	13
1.3	Existujúce riešenia	14
1.4	Štruktúra práce.....	14
2	Východiská.....	16
2.1	Hejného metóda.....	16
2.2	Zdieľanie, spolupráca a podpora	19
2.3	Podobné riešenia.....	20
2.3.1	Webcasting.....	21
2.3.2	TicTacToe a Word Search Online.....	22
2.3.3	Schoolwork.....	22
2.4	Cieľová skupina používateľov.....	24
3	Technológie.....	25
3.1	Čo je to Firebase	25
3.1.1	Ako to funguje ?.....	25
3.1.2	Firebase Database.....	26
3.1.3	Firebase Auth.....	26
3.2	Node.js	26

3.2.1	Neblokujúci cyklus udalostí	27
3.2.2	Asynchrónne programovanie	28
3.3	Porovnanie SQL a NoSQL databáz.....	29
3.3.1	SQL databázy	30
3.3.2	NoSQL databázy.....	30
3.3.3	Zhrnutie.....	31
3.4	Angular 2	31
3.5	Unity 3D.....	32
4	Návrh riešenia.....	33
4.1	Cieľ projektu	33
4.2	Funkčné požiadavky	33
4.2.1	Zdieľanie.....	33
4.2.2	Virtuálna tabuľa	33
4.2.3	Zaznamenávanie štatistik.....	33
4.2.4	Zrkadlenie obrazovky žiaka	34
4.2.5	Webové rozhranie pre učiteľa	34
4.3	Návrh používateľského prostredia – mobilná aplikácia	35
4.3.1	Prihlásovanie a registrácia do aplikácie	36
4.3.2	Výber triedy	37
4.3.3	Návrh zdieľať s	39

4.3.4	Návrh zdieľať – Tabuľa	41
4.4	Návrh používateľského prostredia – webová aplikácia	42
4.4.1	Návrh vytvorenie/editovanie triedy	43
4.4.2	Vytvorenie úlohy.....	43
4.4.3	Pridanie úloh na tabuľu.....	44
4.4.4	Zobrazenie obrazovky žiaka	44
4.4.5	Zobrazenie štatistík.....	45
4.5	Návrh databázy	45
4.5.1	Triedy	46
4.5.2	Úlohy	46
4.5.3	Úlohy pripnuté na tabuľu a zdieľané obrazovky.....	47
4.5.4	Štatistiky.....	47
4.5.5	Tabule.....	48
4.5.6	Používatelia	48
5	Implementácia	49
5.1	Integrácia s Firebase	49
5.2	Vytvorenie C# modulu pre mobilnú aplikáciu.....	50
5.3	Úlohy na tabuľi	50
5.3.1	Pridanie príkladu na tabuľu	50
5.3.2	Zobrazenie úloh na tabuľi.....	51

5.4	Zdielanie úloh.....	51
5.4.1	Odoslanie požiadavky	51
5.4.2	Prijatie požiadavky	52
5.4.3	Zaznamenávanie ľahov	52
5.5	Webové rozhranie	53
5.5.1	Prihlásenie a registrácia	55
5.5.2	Vytvoriť úlohu.....	55
5.5.3	Zoznam tried a mojich úloh.....	56
5.5.4	Priradenie úloh ku triedam.....	56
6	Testovanie.....	57
6.1	Prvé testovanie.....	57
6.1.1	Priebeh testovania	57
6.1.2	Zhrnutie testovania.....	58
6.2	Druhé testovanie	59
Záver		60
Literatúra a internetové zdroje		62
Prílohy.....		64

Zoznam obrázkov

Obrázok 1 Opis fungovania systému Webcasting.....	21
Obrázok 2 ukážka systému Schoolwork od spoločnosti Apple	23
Obrázok 3 ukážka zadávania úloh pre žiakov v systéme Schoolwork	23
Obrázok 4 Príklad blokujúceho kódu	27
Obrázok 5 Príklad neblokujúceho kódu v Node.js.....	28
Obrázok 6 Synchrónne čítanie textového súboru.....	28
Obrázok 7 Nesprávne asynchrónne čítanie súboru v Node.js.....	29
Obrázok 8 Správne asynchrónne čítanie súboru v Node.js.....	29
Obrázok 9 Návrh komunikácie	35
Obrázok 10 Prihlásovanie	36
Obrázok 11 Registračný formulár s ukážkou chybného vstupu.....	37
Obrázok 12 Scéna kde si žiak volí triedu do ktorej chce vstúpiť. V spodnej časti môžeme vidieť možnosť pridania triedy	38
Obrázok 13 Sekvenčný diagram nadviazania spojenia	39
Obrázok 14 zoznam žiakov, s ktorými môžeme zdieľať úlohu	40
Obrázok 15 návrh tlačidla zdieľať s.....	40
Obrázok 16 návrh tlačidla pripnúť na tabuľu.....	40
Obrázok 17 notifikácia žiadosti o spoluprácu.....	41
Obrázok 18 zoznam úloh na tabuli	42

Obrázok 19 ukážka vytvárania úlohy v učiteľskom rozhraní	44
Obrázok 20 zobrazenie štatistik celej triedy	45
Obrázok 21 detail objektu Trieda	46
Obrázok 22 detail objektu SHARED_SCREEN	47
Obrázok 23 detail objektu používateľ	48
Obrázok 24 konfiguračný súbor pre JavaScript	49
Obrázok 25 Handler kontrolujúci žiadosti o spoluprácu	50
Obrázok 26 prijatie požiadavky o zdieľanie	52
Obrázok 27 ukážka generovaného html kódu vo frameworku Angular	54
Obrázok 28 zdrojový kód pridávania hodnoty do prázdnego poľa v šablóne	56
Obrázok 29 žiaci v zápale riešenia úloh	58

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 zmienku 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 nevidané natrafiť na jedinečnú aplikáciu. I v našom prípade existuje množstvo čiastkovo podobných riešení. Počas celého vývoja sme sa nestretli priamo s didaktickou aplikáciou, ktorá by umožňovala komunikáciu medzi používateľmi. Preto sme sa rozhodli poučiť sa z jednotlivých podobných riešení a výsledky zúročiť v našej aplikácii.

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 tieto atribúty spájajú, a preto je hlavnou ideou našej práce 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 opisuje prečo sme sa pre danú tému rozhodli, s akými typmi existujúcich riešení sme sa stretli a aký je cieľ našej práce.

V druhej kapitole nazvanej Východiská si podrobnejšie analyzujeme čo je to Hejného metódou a aké má princípy, načo je dobrá kolaborácia a aké podobné riešenia už existujú.

Tretia kapitola sa zaoberá technológiami, pre ktoré sme sa rozhodli, čitateľovi priblíži ich fungovanie aj dôvod prečo sme sa pre nerozhodli.

Štvrtá kapitola sa zaoberá funkčnými požiadavkami a 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 obsahuje návrh databázy.

V piatej 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 aplikácii darilo pri testovaní.

2 Východiská

2.1 Hejného metóda

Vít Hejný analyzoval príčinu, prečo sa žiaci nesnažia porozumieť problémom a miesto toho si radšej pamätajú vzorce, ktoré sú vhodné iba na riešenie štandardných úloh – hľadal preto neštandardné úlohy, ktoré testoval na žiakoch a svojom synovi. Kvôli politickej situácii sa jeho poznatky nemali možnosť viac rozšíriť.

V roku 1974 sa matematik Milan Hejný po konflikte s učiteľkou svojho syna rozhodol svojho syna učiť sám. Spoločne s niekoľkými kolegami začal rozpracovať poznatky svojho otca. V roku 1987 boli nové myšlienky ucelene publikované. Na rozdiel od tradičných metód výuky matematiky boli zamerané na budovanie siedte mentálnych matematických schém, ktoré si žiak vytvára riešením vhodných úloh a diskusií o svojich riešeniach.

Hejného metóda je založená na 12 princípoch, ktoré sa skladajú do uceleného konceptu tak, aby dieťa objavovalo matematiku samo. Vychádza zo 40 rokov experimentov a prakticky využíva historické poznatky[1] [2].

1. Budovanie schém

Dieťa vie aj to, čo sme ho neučili. Každý človek si pamätá rôzne schémy (domu, záhrady, ...), za pomoci týchto schém je schopný odpovedať na otázky typu: Kol'ko sa nachádza vo vašom dome okien? Spamäti na túto otázku dokáže odpovedať len málokto, no ak si predstavíme schému domu, odpovedať dokážeme. Hejného metóda sa snaží tieto schémy budovať a posilňovať. Schémy na seba napája a vyvádzajú z nich konkrétné úsudky, vďaka čomu si dieťa rýchlo uvedomí, že polovica je číslo 0.5.

2. Práca v prostrediach

Učíme sa opakovanej návštevou. Ak dieťa pozná prostredie a cíti sa v ňom dobre, nerozptyľujú ho neznáme objekty. Dieťa sa dokáže sústredit' na úlohu a neprekáža mu neznámy kontext. Každé prostredie funguje inak. Systém prostredí je nastavený tak, aby zachytával čo najširšie spektrum fungovania detskej mysele.

a všetky štýly učenia. Výsledkom je motivovaná myseľ dieťaťa k novým experimentom.

3. Prelínanie tém

Matematické zákonitosti neizolujeme. Je dôležité, aby sme poznatky dieťaťu nepredávali samostatne, ale aby sme ich zviazali so známou schémou, ktorú si dieťa dokáže kedykoľvek vybaviť. Netrháme od seba matematické výrazy a javy, no zapájame pri nich rôzne stratégie riešenia. Dieťa si tak môže samo vybrať, ktorý spôsob mu najviac vyhovuje. Na hodinách tak nezapočujeme to staré otrepané „*ale ved' to sme sa učili už dávno*“.

4. Rozvoj osobnosti

Podporujeme samostatné uvažovanie detí. Pri vytváraní novej metódy kládol profesor Hejný dôraz na to, aby sa deti nenechali manipulovať. Učiteľ preto nepredáva hotové poznatky, no vedie deti predovšetkým k diskusii, argumentácii a vyhodnocovaniu. Deti sa potom vedia samé rozhodnúť, čo je správne a dokážu rešpektovať druhého. Dokonca statočne nesú aj dôsledky svojho konania. Popri matematike sa prirodzene učia aj základom sociálneho chovania.

5. Skutočná motivácia

Ked' „neviem“ a „chcem vedieť“. Matematické úlohy sú v Hejného metóde postavené tak, aby ich riešenie deti bavilo. Správna motivácia je tá, ktorá prichádza z vnútra a nie je nútensá zvonku. Deti vďaka vlastnej snahe prichádzajú na riešenia úloh. Deti neoberáme o radost' z úspechu a vďaka atmosfére v triede sa kolegálne tlieska aj tým, ktorý na riešenie prídu pozdejšie.

6. Reálne skúsenosti

Stavíame na vlastných zážitkoch dieťaťa. Využívame skúsenosti dieťaťa, ktoré nadobudlo od svojho narodenia. Stavíame na konkrétnych skúsenostiah, z ktorých si dieťa dokáže urobiť vlastný úsudok. Deti napríklad balia kocky do papiera, pričom sa naučia kol'ko má kocka strán a vrcholov i ako sa vypočítava povrch.

7. Radosť z matematiky

Výrazne pomáha pri ďalšej výuke. Skúsenosti hovoria, že najúčinnejšou motiváciou pre deti je ich pocit úspechu - radosti z dobre vyriešenej náročnej úlohy. Táto radosť pramení z vlastného pokroku a uznania okolia. Deti tak nemajú voči matematike averziu.

8. Vlastné poznatky

Majú väčšiu váhu ako tie prevzaté. Úlohou dieťaťa je poskladať štvorec z drievok. Vezme si tri drievka a príde na to, že mu to nestací, tak si vezme aj štvrté. Po dokončení štvorca sa rozhodne ho zväčsiť. V tomto momente začína tušiť, že na zväčšenie bude potrebovať ďalšie štyri drievka. Takýmto spôsobom dieťa objavuje vzorec na výpočet obvodu štvorca.

9. Rola učiteľa

Sprievodca a moderátor diskusií. Spoločenská predstava učiteľa je obraz niekoho, kto vie a učí. Učiteľ matematiky ovláda matematiku, preto o nej môže učiť. V množstve prípadov sa tak aj deje. V našom chápaní výuky je rola učiteľa celkom iná. Učiteľ je ten, kto žiakov organizuje a podnecuje žiakov k práci. Ak niekto niečo vysvetľuje, tak je to žiak. Učiteľ len citlivu reaguje na situácie v triede a znižuje alebo zvyšuje náročnosť úloh - je len tichým sprievodcom hodín a drží sa v úzadí.

10. Práca s chybou

Predchádzame u detí zbytočnému strachu. Ak by malo dieťa zakázané padať, nikdy by sa nenaučilo chodiť. Analyzovanie chyby vedie k hlbšej skúsenosti, vďaka ktorej si lepšie zapamätá poznatky. Deti podporujeme, aby si chybu našli samy a učíme ich vysvetliť, prečo chybu urobili. Dôvera medzi žiakom a učiteľom následne vyvoláva radosť žiakov z dobre odvedenej práce.

11. Primerané výzvy

Pre každé dieťa zvlášť podľa jeho úrovne. Učiteľ zadeľuje úlohy podľa toho, čo ktoré dieťa potrebuje - u slabších žiakov tým predchádza pocitom úzkosti a strachu, lepším predkladá nové výzvy.

12. Podpora spolupráce

Poznatky sa rodia vďaka diskusii. Deti nečakajú, kým sa výsledok objaví na tabuli. Pracujú v skupinách alebo samostatne. Žiak je potom schopný vysvetliť, ako k výsledku došiel a môže to vysvetliť aj ďalším. Výsledok sa tak rodí vďaka spolupráci a žiaci si odnášajú plnohodnotný poznatok[4].

2.2 Zdieľanie, spolupráca a podpora

Teoreticky hviezda – v praxi šedá myš; aj tak by sme mohli označiť postavenie spolupráce na vyučovacích hodinách v našich školách.

Kooperatívne vyučovanie nie je zázračnou metódou, ktorú by sme mohli uplatniť, keď už všetko ostatné zlyhalo. Je však východiskom prirodzenému učeniu, pri ktorom dochádza k zmenám správania, ktoré je podmienené skúsenosťou na základe chýb alebo veľkého záujmu. Účinky kooperatívneho vyučovania sú spontánne, efektívne a dlhodobé[3].

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 klášť 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].

Pomocou kooperatívneho vyučovania sa žiaci učia, komunikovať, spolupracovať a navzájom sa hodnotiť. Pri výskume bolo analyzovaných 27 štúdií o kooperatívnom vyučovaní. Robert E. Slavin zistil, že až v devätnásťich prípadoch bolo kooperatívne vyučovanie značným pozitívom v porovnaní s tradičným spôsobom. Tradičný spôsob bol lepší len v jednom prípade a vo zvyšných siedmych sa rozdiely nezistili. Slavin zistil, že kooperatívne vyučovanie má pozitívny efekt pri motivácii žiakov aj učiteľov dosahovať vysoký učebný výkon v porovnaní s individualizovaným a súťaživým typom vyučovania[13].

Cieľom kooperatívnych aktivít je rozvíjať schopnosť pozerať sa na problém očami druhých, akceptovať iné názory, rozlišovať problémy, s ktorými si dokážeme poradíť samostatne a ktoré si vyžadujú spoluprácu. Naučia žiakov konáť tak, aby bol dosiahnutý spoločný cieľ. Pri kooperatívnej výuke žiaci pracujú v malých skupinách, riešia úlohy a osvojujú si vedomosti. V skupinkách je navodená atmosféra rovnoprávnosti[13].

Skupinová výučba sa však nerovná kooperatívnej výuke. Skupinová výuka je organizačnou podobou výučby, pri ktorej je nahradené bežné usporiadanie, či menej bežná individuálna forma. Chápanie kooperatívnej výučby je založené na spolupráci pri dosahovaní cieľa. Výsledky jednotlivca sú podporované celou skupinou a skupina má prospech z aktivity jednotlivca. Zdieľanie, spolupráca a podpora sú tak základnými pojмami kooperatívneho vyučovania[13].

Podľa Piageta (1970) je „kooperatívnosť - spolupráca - rozhodujúcim prostriedkom formovania racionálneho myслenia, je rozhodujúca pre rozumový vývoj detí. Detia najlogickejšie myslí v diskusii s inými detmi.“

Skupina príde oveľa častejšie na viaceré myšlienky ako človek, ktorý sám premýšľa nad problémom. Pretože jednotlivec nemá možnosť vymeniť si myšlienkové postupy, pospájať ich, prípadne rozvinúť alebo vyvrátiť. Skupina ponúka možnosť spoznať nové uhly pohľadu, tak, že získa z vedomostí iného člena skupiny. Každý člen má iné poznatky a názory, ktoré sú výhodou skupiny hlavne vo vzťahu ku kreativite a kvalite riešenia. Naviac ak sa členovia skupiny budú podporovať, môžu si navzájom rozšíriť svoje schopnosti[3].

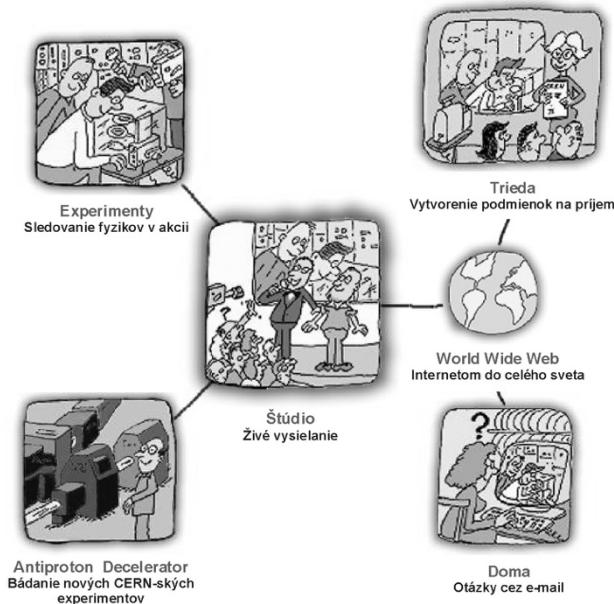
Ak konáme v skupine s porozumením, naučíme sa diskutovať a argumentovať, vďaka čomu môžeme odhaliť vlastné nedostatky v znalostiach a spoznať iné uhly pohľadu[3].

2.3 Podobné riešenia

V nasledujúcej podkapitole si podrobnejšie analyzujeme podobné riešenia, povieme si o ich výhodách aj nevýhodách. Ako bolo už v úvodnej kapitole spomenuté, existujú 2 typy podobných riešení. Prvým typom sú riešenia, ktoré podporujú spoluprácu, ale len na úrovni video konferencie - ide o systém Webcasting. Druhým typom sú aplikácie, podporujúce spoluprácu, no nejde o didaktický softvér.

2.3.1 Webcasting

Ide o jednoduchý a finančne nenáročný spôsob prenosu a archivácie digitálnych záznamov pomocou internetu. Tento spôsob prezentovania si získal veľký obdiv.



Obrázok 1 Opis fungovania systému Webcasting

Hlavne vo vzdelávacích inštitútoch sa stal nevyhnutnou pomôckou v procese dištančného vzdelávania a sprostredkovania najnovších výsledkov.

Prvé využitie systému našli v CERNe a systém slúžil na pracovné porady, semináre a konferencie. V roku 2000 založili tým LIVEfromCERN, ktorého úlohou bolo pripraviť cyklus prednášok pre základné a stredné školy. Počas živého prenosu bolo možné nahliadnuť nielen do štúdia v CERNe, ale hlavne do laboratórií, v ktorých v tom čase prebiehal skutočný výskum. Účastníci mohli klášť otázky prostredníctvom emailu vedcom podieľajúcim sa na výskume. Niekoľko dní pred uskutočnením projektu boli prostredníctvom Virtuálnej kolaborácie stredné školy v Bratislave a Košiciach oboznámené s možnosťou účasti.

Medzi výhody patrí ľahká dostupnosť pre používateľov a na tú dobu bezproblémový prenos obrazu. Používatelia sa pomocou videokonferencie mohli bezprostredne podeliť o svoje zistenia. Ako nevýhodu systému pokladáme, že používatelia nedokázali medzi sebou komunikovať prostredníctvom systému, ale museli používať ďalšiu komunikačnú vrstvu[14].

2.3.2 TicTacToe a Word Search Online

Obe z hier sme sa rozhodli označiť ako podobné našej budúcej aplikácií z jednoduchého dôvodu. Aplikácia, ktorú máme a budeme do nej vytvárať komunikačnú platformu, má veľmi podobnú štruktúru ako spomínané aplikácie. Komunikačná platforma TicTacToe ako aj Word Search Online pracuje v reálnom čase. To pre bežného používateľa znamená, že ľah protihráč alebo spoluhráča sa na jeho zariadení vykreslí takmer ihned. Jediné oneskorenie spôsobuje internetová sieť, no s týmto problémom sa musí počítať. Čo sme si mohli všimnúť u aplikácie Word Search Online je, že komunikácia prebieha štýlom FIFO a tento princíp komunikácie by sme chceli využiť aj v našej platforme.

Pri analyzovaní podobných riešení sme sa zamerali aj na user interface aplikácií. Presnejšie povedané na spôsob akým sa v aplikáciách nadväzuje spojenie, respektíve čo všetko musí používateľ urobiť, aby sa mu to podarilo.

TicTacToe[15] má dve možnosti nadviazania spojenia. Prvou je rýchla hra, kde nám aplikácia nadviaže spojenie s náhodným protihráčom a spojenie udržuje po celý čas hry. Druhým spôsobom, ktorý je pre nás zaujímavejší, bolo pozvanie hráča. Po zvolení tejto možnosti sme mali na výber dva zo zoznamy. V jednom bol zoznam hráčov, s ktorými sme hrali naposledy a v druhom bol zoznam hráčov, ktorých máme v priateľoch. Poslednou možnosťou bolo vyhľadanie spoluhráča cez vyhľadávacie okno podľa jeho mena.

Word Search Online[16] má rovnako ako TicTacToe dve možnosti. Prvou je Quick Match, ktorá súpera vyberie náhodne. Druhou možnosťou je pozvať priateľa avšak táto možnosť je v tejto aplikácii riešená veľmi nešťastne, nakoľko pre prístup k tejto možnosti sa musíme do aplikácie prihlásiť za pomocí nášho Facebook účtu.

2.3.3 Schoolwork

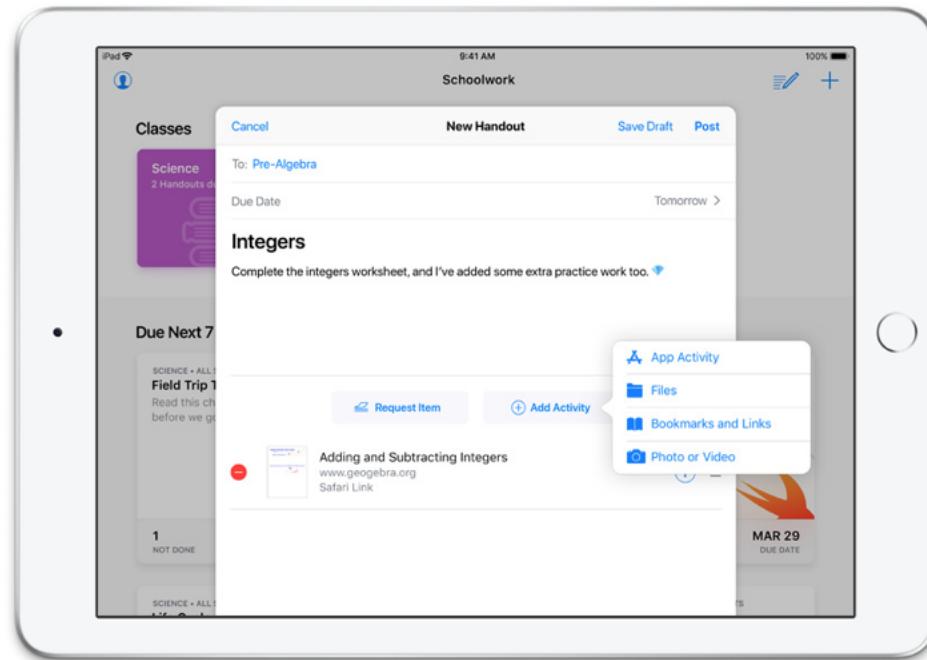
Aplikácia Schoolwork[17][18] je novinkou od spoločnosti Apple, ktorá bola predstavená koncom marca 2018. Aplikácia Schoolwork je pripravovaná pre zariadenie iPad a má

uľahčovať interakciu medzi učiteľom a žiakmi v triede. Funguje ako Hub pre iné didaktické aplikácie.



Obrázok 2 ukážka systému Schoolwork od spoločnosti Apple

Učiteľ môže pomocou aplikácie zadávať žiakom úlohy, sledovať, ako ich žiaci postupne riešia a v prípade potreby aj osobitne pracovať s vybranými žiakmi. Úlohy môže organizovať na základe priority, tried alebo termínov. Učiteľ zadáva žiakom úlohy pomocou správ podobným e-mailom, ktoré obsahujú dátum dokončenia, sprievodný text a samotnú úlohu.



Obrázok 3 ukážka zadávania úloh pre žiakov v systéme Schoolwork

Apple považuje za prínos práve odkazovanie na konkrétné funkcie či sekcie iných aplikácií. Cez Schoolwork môže učiteľ odkázať žiakov priamo na tú časť aplikácie, s ktorou chce pracovať. Spoločnosť Apple vylepšila aj existujúcu aplikáciu Classroom.

Cez aplikáciu Classroom môže učiteľ pomocou svojho zariadenia ovládať iPady žiakov, vidieť čo na nich robia a zobraziť ich obsah na projektore. Ktoréhokoľvek študenta alebo celú triedu môže učiteľ cez aplikáciu navigovať na knihu, aplikáciu alebo stránku. V aplikácii je tiež možné na iPadoch uzamknúť aplikáciu, aby žiaci nemohli používať nič iné, pokial' im to učiteľ nedovolí.

Služba obsahuje aj nástroj Apple School Manager, ktorý slúži učiteľom a administrátorom na spravovanie účtov študentov, ich zariadení a studijných materiálov. So službou možno inštalovať aj aplikácie a stáhovať obsah priamo do zariadení žiakov, a to jednotlivo aj naraz.

2.4 Cieľová skupina používateľov

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

3 Technológie

Základom pre realtime platformu bolo vhodne zvoliť technológie. Z dôvodu multiplatformovosti musela mať technológia na prenos dát 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 tieto technológie priblížime.

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 naším serverom, API aj databázou. Firebase má 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 Realtime Database je navrhnuté tak, aby umožnilo len rýchlo dostupné operácie. Umožní nám tak 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, zdieľajú všetci klienti 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 klientské 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 možné 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 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 neblokujúce 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 4 Príklad blokujúceho kódu

Na obrázku Obrázok 4 môžeme vidieť blokujúci kód. Funkcia `sleep()` nám blokuje hlavné vlákno a tým pádom nemôžu byť vykonané ďalšie inštrukcie. Na obrázku Obrázok 5 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 5 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 6 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 a plytvá našim časom, kým čaká na súborový systém. Na obrázku Obrázok 7 je prepísaná verzia kódu z obrázku Obrázok 6 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 7 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 máme informáciu o úspešnosti, druhý označuje 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 8 Správne asynchrónne čítanie súboru v Node.js

3.3 Porovnanie SQL a NoSQL databáz

Pri výbere vhodných technológií serverovej časti bolo potrebné k serveru vhodne 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

SQL databázy alebo relačné databázy sú postavené na relačnom modeli. 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 reláciách 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 namiesto 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 štruktúrované, neštruktúrované, prípadne semi-štruktúrované. NoSQL databázy využívajú 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 zavážila hlavne možnosť pristupovať k hodnotám na základe kľúča. Ďalším plusom pre NoSQL bol prístup k dátam pomocou REST alebo SOAP requestov. Túto kapitolu sme spracovali s použitím [7] [8].

3.4 Angular 2

Angular 2 je nasledovníkom populárneho JavaScript frameworku AngularJS, ktorý je primárne určený na vývoj jednostránkových webových aplikácií. Angular je využívaný spoločnosťou Google. Je naprogramovaný v jazyku TypeScript, ktorý je kompilovaný do JavaScript kódu. Vďaka tomu aplikácia naprogramovaná v Angular 2 pobeží v ľubovoľnom prehliadači. Súčasťou Angularu je oficiálne API na komunikáciu s real time databázou Firebase, ktoré využívam vo svojej aplikácii. Angular môže dosiahnuť vysokú rýchlosť aj pri veľkých aplikáciach vďaka Web Workerom a renderovaniu na strane servera. Vyznačuje

sa dobrou škálovateľnosťou a vďaka dátovým modelom splňa veľké požiadavky na prácu s dátami. Vďaka použitiu deklaratívnych šablón Angular umožňuje rýchle vytváranie funkcií, či definovanie vlastných komponentov. Tiež poskytuje routovanie pre navigáciu medzi stránkami[10][12].

3.5 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 medzi sebou 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 chce s nimi 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 môže na tabuľu pridať sady úloh. Žiaci môžu príklady z tabule riešiť, ale aj pridávať nové úlohy. Každá trieda by mala mať svoju vlastnú tabuľu, čím sa zabezpečí, že sa úlohy dostanú k správnym adresátom. Žiaci budú o nových úlohách na tabuli notifikovaní 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

počtu úloh a úspešnosť ich riešenia. Druhým typom záznamu budú konkrétné 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 podozrenia, že žiak nepracuje, alebo si nevie dať rady s úlohou, si učiteľ bude môcť otvoriť obrazovku žiaka vo svojom systéme. Uvidí tak presnú kópiu 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 heslo. O tom, či je heslo jedinečné, bude informovať systém. Po vytvorení triedy aplikácia presmeruje používateľa na domovskú stránku, kde triedu už bude vidieť.

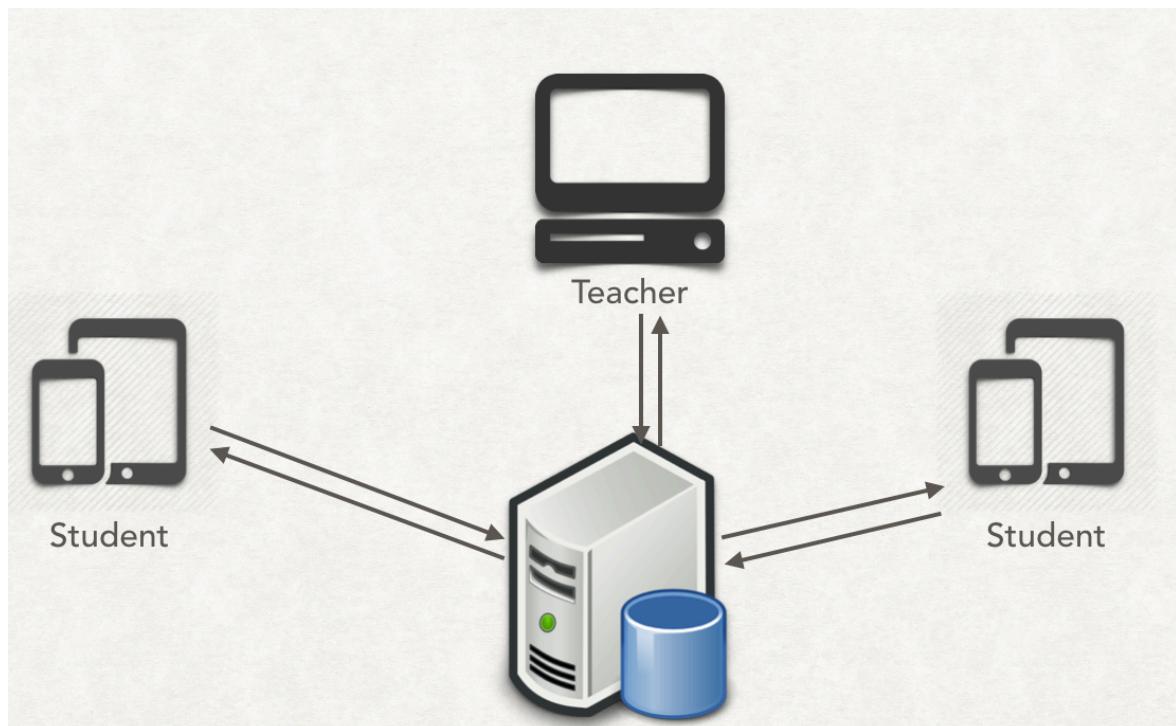
Vytvorenie úloh – aplikácia by mala používateľovi umožniť vytvoriť úlohu. Táto úloha sa zaznamená do databázy. Používateľ bude môcť následne úlohu priradiť jednej alebo viacerým triedam, prípadne bude môcť úlohu editovať alebo zmazať.

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, a to počet správnych a počet nesprávnych úloh. Štatistiky žiaka sa budú zobrazovať ako zoznam príkladov vyriešených z virtuálnej tabule a celkový počet príkladov.

Zobrazenie obrazovky žiaka – používateľ webovej aplikácie bude mať možnosť zobraziť si obrazovku žiaka, na ktorej uvidí posledné riešenú úlohu v prípade, že práve nerieši nové zadanie. 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 spustí overenie prihlásenia/neprihlásenia užívateľa. Po tejto kontrole môžu nastať 2 scenáre. V prvom je žiak už prihlásený, dátu prihláseného používateľa sa stiahnu 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 vnejakej z tried. Ak sa žiak nenachádza v žiadnej triede, má možnosť zadat heslo triedy a vstúpiť 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.

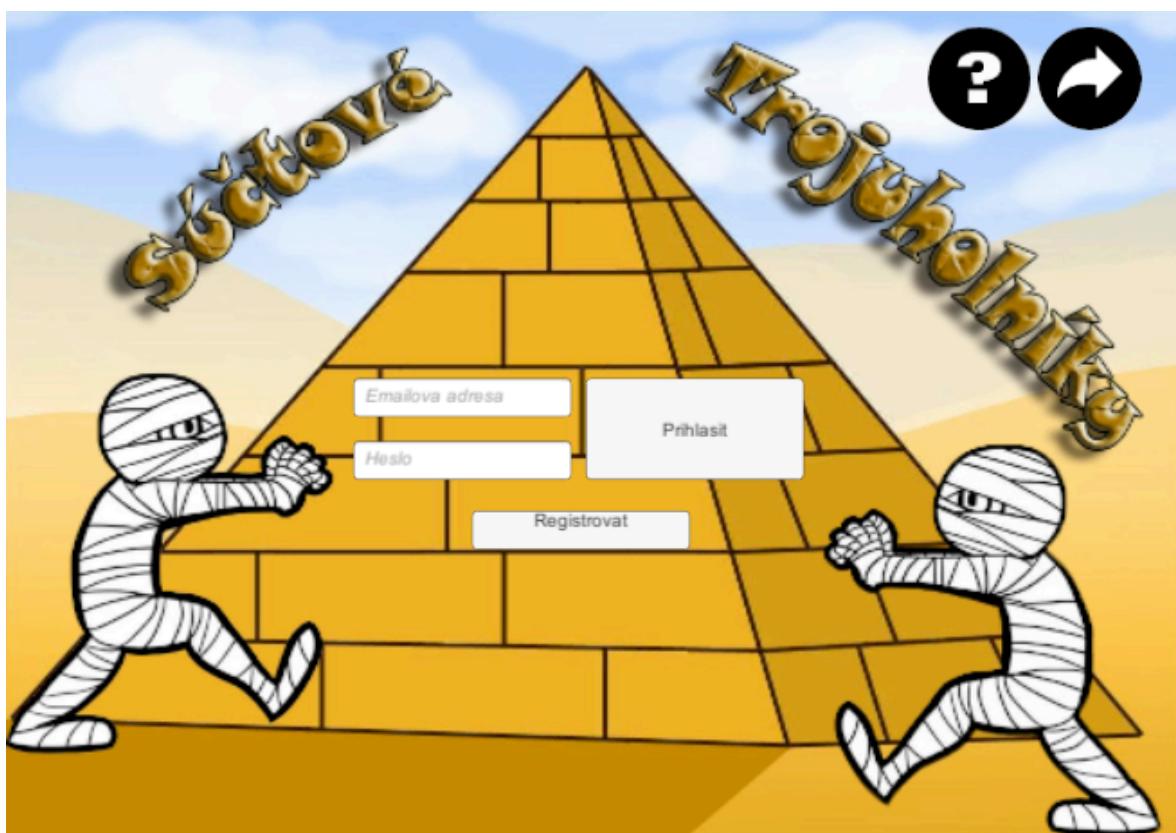


Obrázok 9 Návrh komunikácie

Druhou, hlavnou časťou našej práce je funkcia Shared with... (Zdieľať s...) a cieľom tejto funkcionality je, aby žiaci svoje úlohy riešili spolu, tzn. jedna úloha, ktorú rieši niekoľko žiakov na viacerých tabletach. Hlavným cieľom pri tejto funkcií bolo navrhnuť ju tak, aby

sme neprenášali 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 sú obaja žiaci z triedy 1.A a môžu medzi sebou zdieľať úlohy, ale aj prezerať tabuľu, ktorú majú spoločnú. Druhým prípadom je rozdielnosť tried oboch žiakov. Tu sa žiakom medzi sebou spojenie nadviazať nepodarí, keďže cieľom je nadviazať komunikáciu v rámci tried a nie mimo nich.

4.3.1 Prihlásование a registrácia do aplikácie



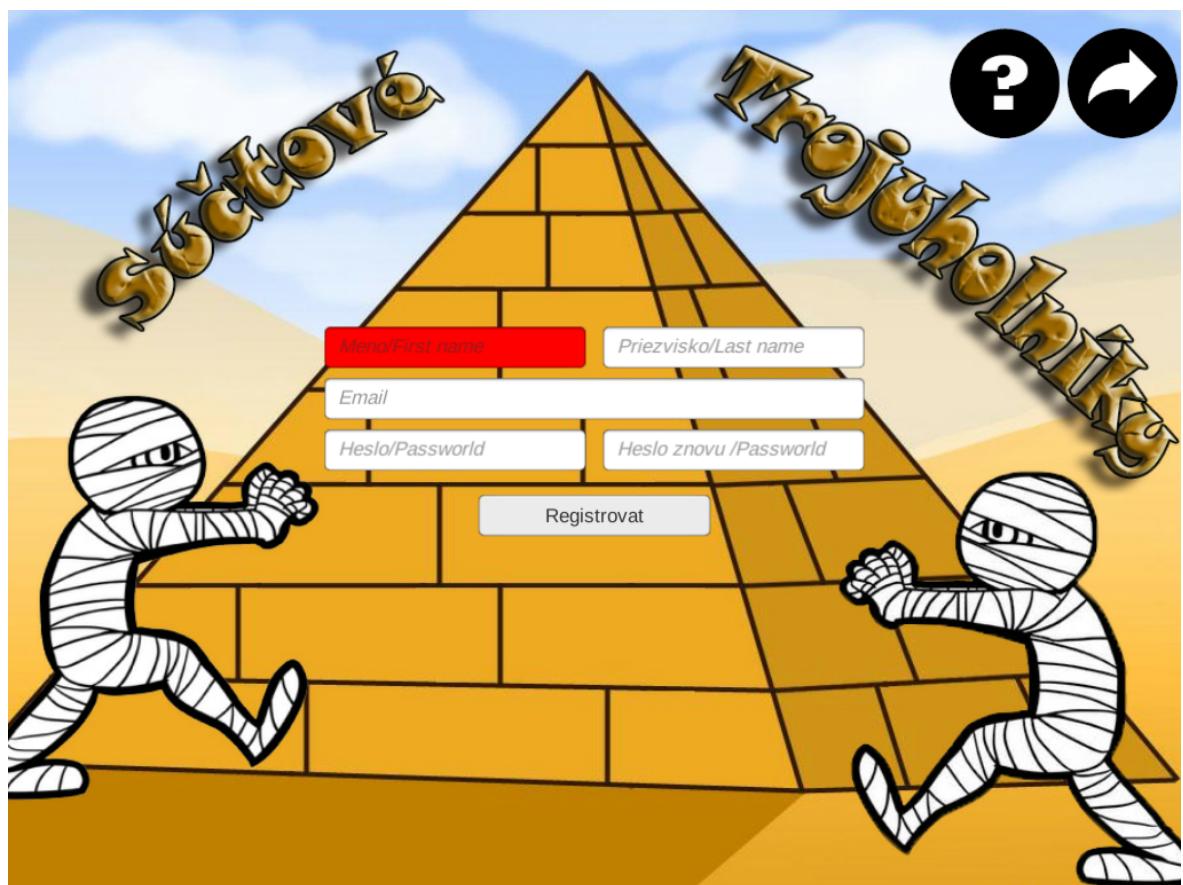
Obrázok 10 Prihlásanie

Používateľ sa prihlasuje pomocou e-mailu a hesla, ktoré si sám zvolí. V prípade, že aplikáciu používa prvýkrát, žiak sa pred prihlásením musí zaregistrovať. Po úspešnom prihlásení bude používateľ presmerovaný na scénu s výberom triedy. Prihlásenie na zariadení trvá dovtedy, pokiaľ sa žiak z aplikácie sám neodhlási, alebo neprebehne aktualizácia aplikácie. V prípade zlyhania prihlásenia bude príčina oznamená formou informatívneho výpisu.

Nezaregistrovanému používateľovi sa zobrazí krátky formulár, ktorý obsahuje :

- E-mail
- Meno
- Priezvisko
- Heslo

Po vyplnení formulára a jeho odoslaní prebehne registrácia a po úspešnom zaregistrovaní sa, 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 polička.



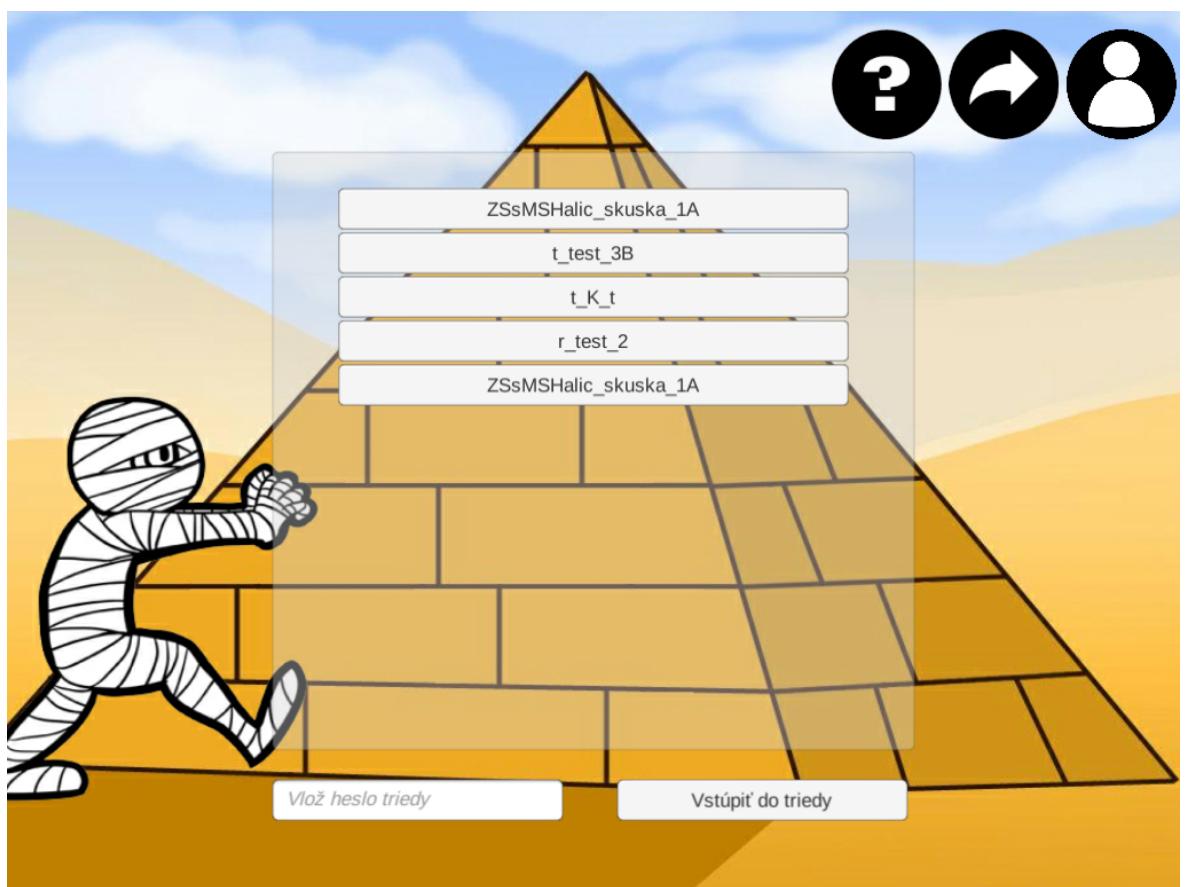
Obrázok 11 Registračný formulár s ukážkou chybného vstupu

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 k tomu došlo, systém automaticky doplní triedu do nášho scrollbaru. Tento handler počúva aj na zmenu názvu triedy, príp. zmazanie. V prípade, že by učiteľ triedu premenoval práve v momente, keď si žiak volí triedu, automaticky by sa mu názov triedy zmenil, v prípade odstránenia by trieda okamžite zmizla z ponuky.

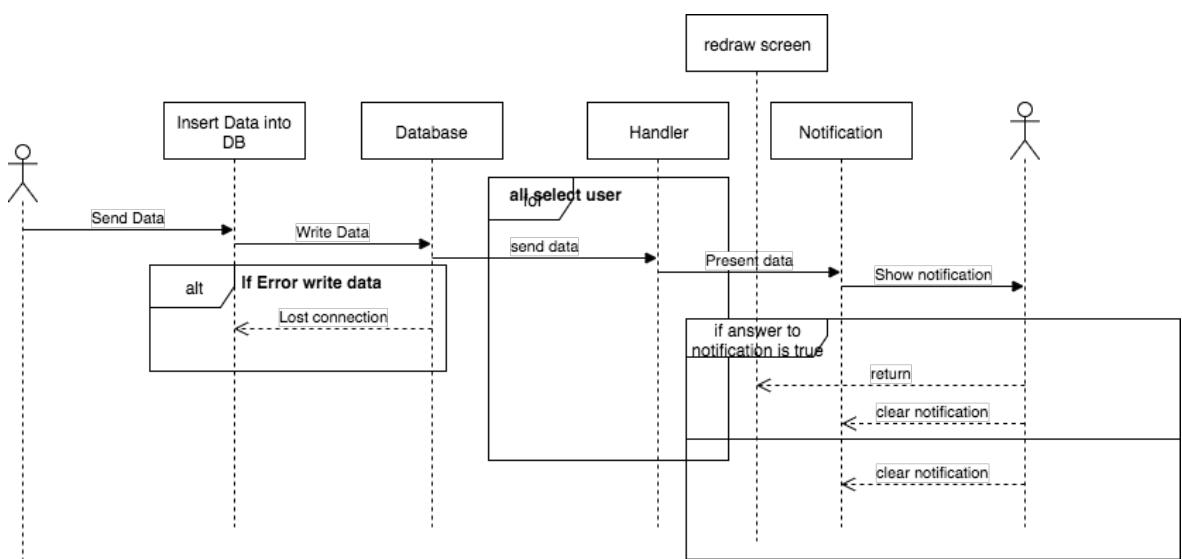
Druhou funkciou tejto scény je ukladanie tried. Funkcia funguje na základe hesla triedy. Žiak zadá heslo triedy, ktoré určí učiteľ. Po zadaní hesla prebehne kontrola, či na základe hesla existuje takáto trieda. Ak systém danú triedu v databáze nájde, automaticky žiaka do tejto triedy priradí a zobrazí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



Obrázok 12 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 funkcionálít platformy a aj najkomplikovanejšia na návrh. Pri jej vývoji bolo potrebné dodržať určité dôležité podmienky, ako sú real time, malý obsah prenášaných dát, nadviazanie spojenia a princíp lock/unlock obrazovky. Poslednú spomenutú funkciu bolo najproblematickejšie vyriešiť, nakoľko sa nám ponúkalo niekoľko možností. Prvou možnosťou bolo uzamknutie obrazoviek všetkým neaktívnym používateľom. Čiže, používateľ s označeným objektom by zablokoval prístup ostatným. V tomto spôsobe by mohol nastať problém v tom, že vždy sa môže nájsť užívateľ, ktorý by uzamkýnal obrazovku ostatným a znemožňoval výuku. Druhou metódou bola rozšírenie prvej metódy 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 do šablóny, bude objekt vrátený na pôvodné miesto a obrazovka odblokovaná. Táto metóda bola lepšia, avšak vysvetliť žiakom, prečo zmizlo zvýraznenie objektu, by bolo zdlhavé. Metóda úplne neriešila blokovanie ostatných užívateľov. Ďalšou metódou, ktorá prichádzala do úvahy, bola metóda FIFO. Princíp spočíval vtom, že všetci užívatelia mohli označiť rovnaký objekt, avšak zvýraznenie tohto objektu zmizlo všetkým ihneď po tom, ako ho najrýchlejší užívateľ umiestnil na novú pozíciu. Tento spôsob sa ukázal ako najvhodnejší. Správnosť rozhodnutia si overíme pri prvom testovaní. Nadviazanie spojenia je zabezpečené pomocou handlera,

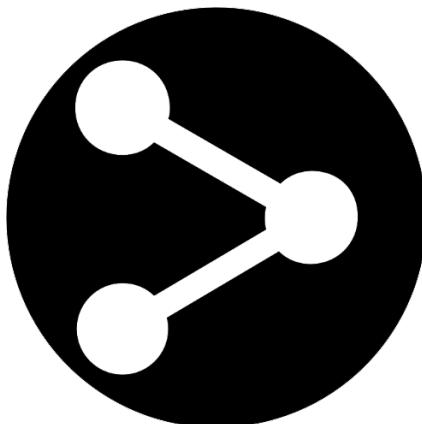


Obrázok 13 Sekvenčný diagram nadviazania spojenia



Obrázok 14 zoznam žiakov, s ktorými môžeme zdieľať úlohu

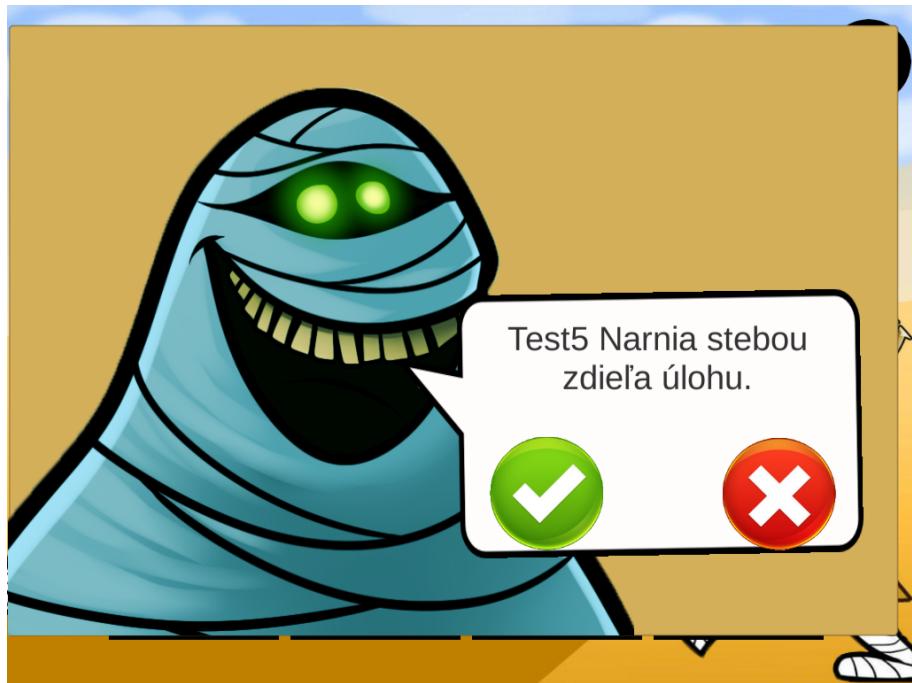
ktorý počúva na objekte „waitForShare“ v databáze. V momente zmeny v tomto objekte sa zobrazí žiadost o nadviazanie spojenia. V momente potvrdenia žiadosti sa zosynchronizuje obrazovka s navrhovateľom. Od tohto okamihu je vytvorené spojenie medzi používateľmi až do odpojenia sa, pripadne vyriešenia úlohy.



Obrázok 15 návrh tlačidla zdieľať s...



Obrázok 16 návrh tlačidla pripnúť na tabuľu

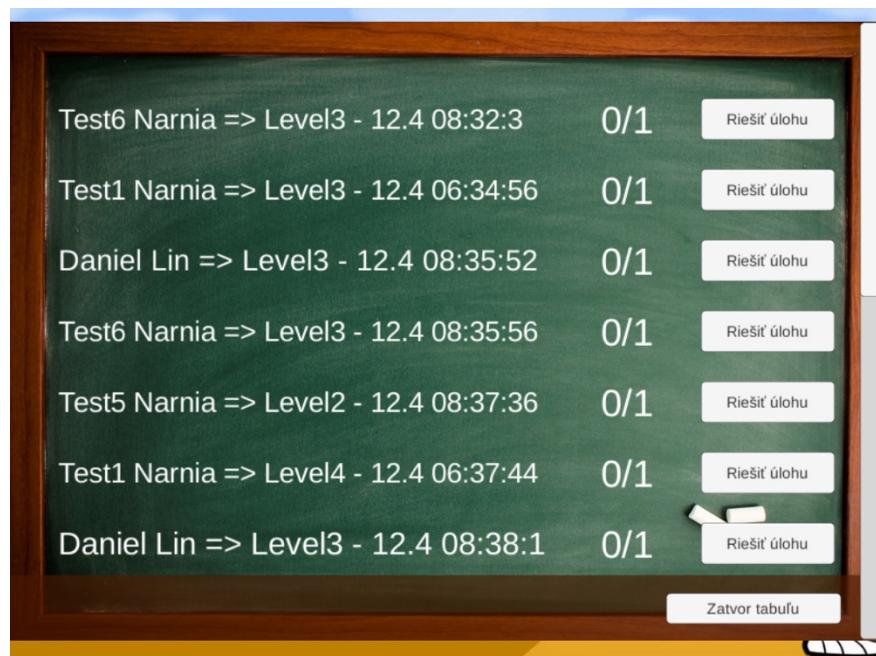


Obrázok 17 notifikácia žiadosti o spoluprácu

4.3.4 Návrh zdieľať – Tabuľa

Časť Tabuľa je ďalšou dôležitou funkciou našej platformy. Funkcia „Zdieľať s“ 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 oň podeliť so svojimi spolužiakmi. V tejto časti bolo dôležité vhodne navrhnuť databázu, nakoľko sa úlohy môžu opakovať. V rámci databázy, ale aj tabule by priamo v programe tým pádom vznikali zbytočné duplicity, a tak by sme žiakov nezaujali rovnakým opakovaním sa úloh/zadaní. Druhým dôležitým krokom je 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 tabuľi všetkým žiakom v rámci triedy.

O pridávaní úloh na tabuľu nebudú žiakom chodiť notifikácie, ktoré by v tomto prípade mohli byť príliš časté a rušivé. Používateľ je o počte pribudnutých úloh informovaný formou výpisu . Je na zvažení používateľa, ako často si skontroluje tabuľu.



Obrázok 18 zoznam úloh na tabuľi

Tabuľa je navrhnutá tak, aby neslúžila len žiakom. Na tabuľu môže pridávať úlohy aj učiteľ zo svojho webového rozhrania. Príklady pridané na tabuľu sú logované a učiteľ si na základe toho dokáže skontrolovať riešiteľa i úspešnosť.

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. Uvedenú technológiu sme zvolili z dôvodu, že mobilná aplikácia je prispôsobená žiakom. Preto sme sa rozhodli oddeliť detskú hradú 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 a 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 detaľy všetkých záznamov.

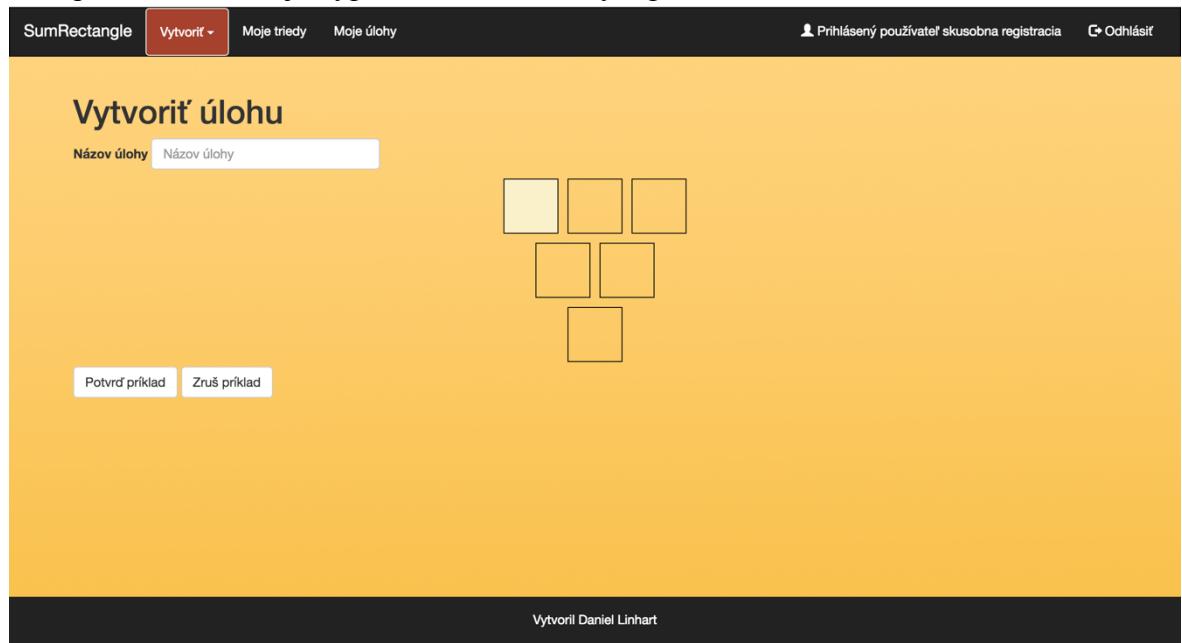
4.4.1 Návrh vytvorenie/editovanie triedy

Z dôvodu, že aplikáciu môžu používať žiaci z rôznych tried a škôl, sme sa rozhodli vytvoriť množinu tried. Tieto triedy slúžia učiteľom na začlenenie si žiakov do tried. Na vytvorenie triedy má každý učiteľ vo svojom rozhraní pripravený krátky formulár, ktorý obsahuje názov triedy a heslo. Heslo triedy musí byť jedinečné, aby zariadenia žiakov vedeli na základe hesla triedu identifikovať. Po vyplnení formulára a odoslaní sa trieda automaticky zobrazí v DB a je prístupná žiakom. Žiaci potrebujú vedieť na vstup do danej vstupný kľúč, 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 a je rozdelená 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 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* zruší len 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 je úloha hotová. Po použití tlačidla uložiť úlohu sa úloha zapíše do databázy a o úspešnosti nás aplikácia informuje výpisom a automatickým presmerovaním medzi zoznam úloh.



Obrázok 19 ukážka vytvárania úlohy v učiteľskom rozhraní

4.4.3 Pridanie úloh na tabuľu

Na to, aby sa žiaci dostali k úlohám, ktoré sme si pre nich vytvorili, je potrebné ich priradenie 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 triede priradiť. Modálne okná sa zobrazia po zvolení možnosti *Priradiť 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 funkcionality. 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.

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 žiakovi o tom, koľko príkladov vyrátal správne a koľko nesprávne. Nad týmto zoznamom sa ešte nachádzajú celkové súhrny vypočítaných úloh a údaje o počte správnych a nesprávnych riešení.

Druhá podstránka, ku ktorej sa dostaneme po kliknutí možnosti zobraz štatistiky nad žiakom, zobrazuje štatistiky žiaka a v hornej časti sa opäť nachádza súhrn o počte celovo vyriešených úloh i o počte správnych a nesprávnych riešení. Pod týmto súhrnom sa nachádza zoznam príkladov, ktoré učiteľ pridal na tabuľu aj s výsledkami, koľko z nich vypočítať a koľko chýb pri ich počítaní urobil.

Štatistiky triedy Narnia1	
Počet všetkých riešených príkladov: 164	
Počet správne vyriešených príkladov: 123	
Počet nesprávne vyriešených príkladov: 41	
Zoznam štatistik žiakov:	
Test1 Narnia	Správne: 31 Nesprávne: 7
Daniel Lin	Správne: 21 Nesprávne: 6
Test5 Narnia	Správne: 15 Nesprávne: 9
Test2 Narnia	Správne: 30 Nesprávne: 2
Test3 Narnia	Správne: 4 Nesprávne: 4
Test7 Narnia	Správne: 1 Nesprávne: 4
Test4 Narnia	Správne: 1 Nesprávne: 3
Test6 Narnia	Správne: 20 Nesprávne: 6

Obrázok 20 zobrazenie štatistik celej triedy

4.5 Návrh databázy

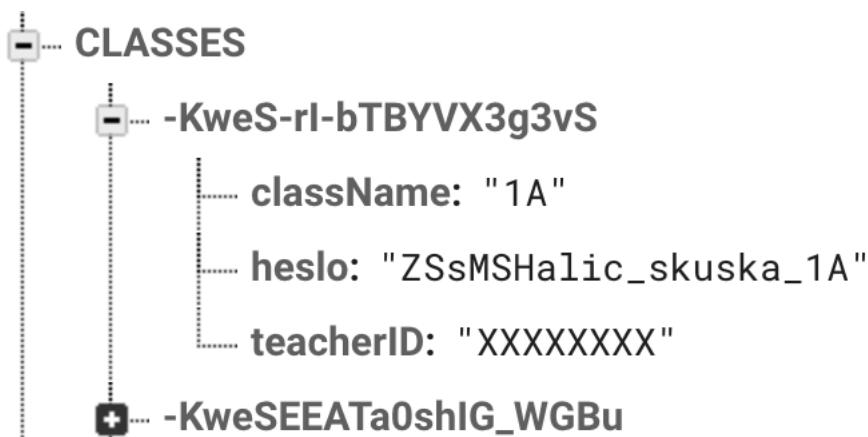
Pri návrhu databázy sme museli klášť dôraz na jednoduchý prístup k dátam. Z dôvodu nepoužitia SQL databázy sme tým pádom nemáli k dispozícii operácie ako join, group by, atď. Bolo potrebné premysliť si, ku ktorým dátam budeme pristupovať často, a teda bude rozumné nevnárať ich hlboko do JSON štruktúry.

Databázu sme rozdelili do siedmich základných objektov:

- *CLASSES* (*tryedy*)
- *EXAMS* (*úlohy*)
- *EXAMS_PINNED_TO_TABLE* (*úlohy pripnuté na tabuľu*)
- *SHARED_SCREEN* (*zdieľané obrazovky*)
- *STATISTICS* (*štatistiky*)
- *TABLES* (*tabuľky*)
- *USERS* (*používateľia*)

4.5.1 Triedy

V objekte triedy sa nachádzajú 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 priradení 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.

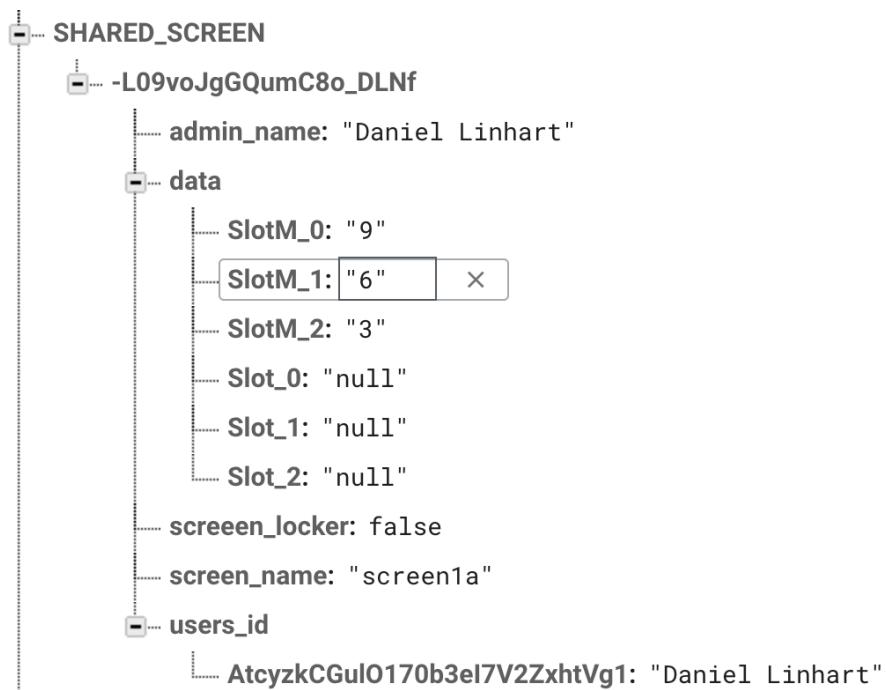


Obrázok 21 detail objektu Trieda

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



Obrázok 22 detail objektu SHARED_SCREEN

Návrh tohto objektu bol náročnejší, nakoľko sme si museli zapamätať celé nastavenie danej obrazovky. Ide o objekty, ktoré nám slúžia k zstrojeniu identickej scény ako u predchodcu. 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, ktorí 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. Keďže pri dátach uložených do objektov trieda alebo používateľ by sme sa k dátam dostali náročnejšie. Museli by sme použiť viac cyklov pre získanie špecifických dát. 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 kolko príkladov daná úloha obsahuje.

4.5.6 Používateľia

Návrh tohto objektu bol veľmi náročný, nakoľko sme museli vhodne rozhodnúť, ktoré dátá 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, e-mail 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, pričom sa jedná o úlohy nachádzajúce sa na tabuli. V poslednom objekte sa nachádza informácia o tom kolko úloh pribudlo na tabuli od poslednej kontroly tabule. Kedže pre každú triedu máme inú tabuľu, tento objekt obsahuje zoznam tried, v ktorom je nasledujúca informácia uložená.



Obrázok 23 detail objektu používateľ

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 pri implementácii aplikácie bola integrácia technológie firebase do našich aplikácií. Možnosti integrácie firebase sme si mohli pozrieť v dokumentácii služby[5].

V prvom kroku sme museli vytvoriť projekt v službe firebase. Na vytvorenie projektu bolo potrebné uviesť názov projektu a krajinu. Na základe názvu projektu nám bolo 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 platoform. Ďalším krokom pri integrácii bolo stiahnutie API rozhrania pre Unity a nakopírovať ho spolu aj s konfiguračnými súbormi do projektu. Posledným krokom bolo nainštalovať API do nášho Unity projektu.

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.

```
// Set the configuration for your app
// TODO: Replace with your project's config object
var config = {
  apiKey: "apiKey",
  authDomain: "projectId.firebaseio.com",
  databaseURL: "https://databaseName.firebaseio.com",
  storageBucket: "bucket.appspot.com"
};
firebase.initializeApp(config);

// Get a reference to the database service
var database = firebase.database();
```

Obrázok 24 konfiguračný súbor pre JavaScript

5.2 Vytvorenie C# modulu pre mobilnú aplikáciu

V 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 nainšportovať 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 kóde. Do modulu sme preto implementovali funkcie na zápis a čítanie z databázy.

```
public void HandleControlRequestSharedScreen(object sender, ChildChangedEventArgs args)
{
    if (args.DatabaseError != null)
    {
        Debug.LogError(args.DatabaseError.Message);
        return;
    }
    string key = args.Snapshot.Key;
    string adminName = args.Snapshot.Child("admin_name").Value.ToString();
    string screenKey = args.Snapshot.Child("share_object").Value.ToString();

    infoAboutShare.enabled = true;
    Text text = GameObject.Find("txtRequestFrom").GetComponent<Text>();
    text.text = adminName + " stebou zdieľa úlohu.";
    Button btnSuhlas = GameObject.Find("btnSuhlas").GetComponent<Button>();
    btnSuhlas.onClick.AddListener(delegate
    {
        AcceptShareScreen(screenKey, key);
    });
    Button btnNesuhlas = GameObject.Find("btnNesuhlas").GetComponent<Button>();
    btnNesuhlas.onClick.AddListener(delegate
    {
        MissedShareScreen(key);
    });
}
```

Obrázok 25 Handler kontrolujúci žiadosti o spoluprácu

5.3 Úlohy na tabuľi

Z dôvodu, že v predchádzajúcej verzii aplikácie sme tabuľu nemali, museli sme 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 stlačení tlačidla *pripnúť* sa zavolá metóda *PripniPríkladNaTabulu*. Táto metóda vytvorí novú inštanciu dátového modelu a

naplní ju. V ďalšom kroku vytvorí identifikačný klúč a úlohu vloží do databázy. Po zapísaní do databázy zavolá metódu *InfoAboutPin* – táto metóda informuje používateľa o úspešnosti pridania úlohy a názve úlohy. Opäťovné volanie metódy *PripniPrikladNaTabulu* 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 tabuli nám slúži metóda *showExamsOnBoardAdd*. Metóda v sebe obsahuje viacero vnorených asynchronných volaní a je implementovaná nasledovne. V prvom volaní si z databázy vyžiadame dátu 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 prekresľujú hned' pri pridaní a nie je potrebné tabuľu obnoviť.

5.4 Zdieľanie úloh

Túto časť implementácie sme sa rozhodli rozdeliť do troch podkapitol. V nich si postupne rozoberieme ako sú implementované jednotlivé časti, ktorými sú odoslanie požiadavky, prijatie požiadavky a zaznamenávanie ľahov.

5.4.1 Odoslanie požiadavky

Implementácia zdieľania úloh sa skladala z viacerých častí. Začína sa 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, klúč a hodnotu a vykreslí riadok s používateľom. Metóda *HandleShowAllUserInClassAdd* je implementovaná tak, že pri prvom spustení sa vykoná toľkokrát, koľko 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 inšanciu 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 spravuje metóda *MissedShareScreen*. Táto metóda má za úlohu odstrániť žiadosť z databázy a zatvoriť informáciu o zdieľaní.

```
public void AcceptShareScreen(string screenKey, string requestKey)
{
    GlobalData.playerData.cestaKZdielanimDatam = "/SHARED_SCREEN/" + screenKey + "/data/";
    _fbc.inserMyIdToSharedScreen(_playerData.UserId, _playerData.Name, screenKey);
    FirebaseDatabase.DefaultInstance.GetReference("/USERS/" + _playerData.UserId + "/waitForShare/")
        .Child(requestKey).RemoveValueAsync();
    FirebaseDatabase.DefaultInstance.GetReference("/SHARED_SCREEN/" + screenKey + "/")
        .GetValueAsync().ContinueWith(task =>
    {
        if (task.IsCompleted)
        {
            DataSnapshot snap = task.Result;
            GlobalData.playerData.zdielaneDataAll = snap;
            string nameScene = snap.Child("screen_name").Value.ToString();
            UnbindAllHandler();
            SceneManager.LoadScene(nameScene);
        }
    });
}
```

Obrázok 26 prijatie požiadavky o zdieľanie

5.4.3 Zaznamenávanie ľahov

V predchádzajúcich podkapitolách sme si rozobrali implementáciu Odoslania požiadavky aj Prijatie, no ešte stále sa nám zmeny neprevádzajú. Na zaznamenávanie ľahov nám slúžia metódy *HandleChildChanged* a *UpdateResult*. *HandleChildChanged* slúži na kontrolu, či niektorí zo žiakov nespravili ľah. 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 ľahu. Metóda

zaznamená násťah do databázy. Po vykonaní metódy *UpdateResult* sa následne na všetkých tabletoch, s ktorými máme spojenie, spustí metóda *HandleChildChanged*.

5.5 Webové rozhranie

Na implementáciu webového rozhrania sme zvolili technológiu Nette[9]. Ide o PHP framework.

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. Šablóny v template Latte sme pripravovali v jazyku HTML a CSS. Príprava šablón spolu s dizajnom stránky (CSS) nenasvedčovali tomu, že by sme mohli mať v ďalších krokoch problémy.

Druhým krokom bolo vytvorenie registrácie a prihlásovanie na backendovej strane stránky, kde sme po prvýkrát zaznamenali problematicosť. 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 funkcia okrem routovania stránok bola naprogramovaná v JavaScripte a za pomoci technológie JQuery doplnaná 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 a z tohto dôvodu sme sa nakoniec rozhodli pre 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.

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.

```
<div class="row">
  <div class="col-xs-12 col-sm-12 col-md-12 col-lg-12">
    <div class="all_classes">
      <div class="MainBox" *ngFor="let d of zoznamTried">
        <a class="ImageBox" [routerLink]="/openclass", d.$key">
          <div class="c_name">
            {{d.className}}
          </div>
        </a>
        <div class="ButtonBox1">
          <a class="pridaj_ulohu_pajpa" id="pridaj{{d.$key}}" name="" data-toggle="modal" data-target="#pridajUlohu" (cli
          <a class="zobraz_ulohy_pajpa" id="zobraz{{d.$key}}" name="" data-toggle="modal" data-target="#zobrazUlohy" (cli
          <a [routerLink]="/classstatistics", d.$key">Zobraz štatistiky</a>
        </div>
        <div class="ButtonBox2">
          <a class='edit' [routerLink]="/updateclass", d.$key" ><i class="fa fa-pencil-square-o" aria-hidden="true"></i>
          <a class='close' (click)="deleteClass(d.$key)"><i class='fa fa-window-close-o' aria-hidden=true></i></a>
        </div>
      </div>
    </div>
  </div>
```

Obrázok 27 ukážka generovaného html kódu vo frameworku Angular

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

Na to, aby sme do projektu mohli vložiť šablóny zo starého projektu, sme museli vytvoriť komponentu ku každej šablóne.

Komponenta 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 prázdný, 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 e-mail 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 e-mail, heslo, meno a priezvisko. Najskôr sa skontrolujú všetky povinné hodnoty na strane serveru a v prípade, že kontrola prebehla úspešne, zavolá sa metóda *registerUserWithEmailAndPassword*. Táto metóda vykoná samotnú registráciu a zapíše dátu do databázy. O úspešnosti nás následne servis informuje rovnakým spôsobom ako pri prihlásovaní.

5.5.2 Vytvoriť úlohu

Vytváranie úloh zastrešuje komponenta *CreateExamPageComponent* a funguje nasledovne. V konštruktore sa vygeneruje zoznam možností, ktoré môžeme dosadzať do prázdných polí. Po stlačení tlačidla pridať príklad sa nastaví prázdna šablóna. Do šablóny 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 potvrdiť príklad, prebehne metóda *submitTask*. Metóda skontroluje, či sú všetky hodnoty vyplnené a následne vloží príklad do dátového modelu. Pridanie príkladov je možné opakovať. Po stlačení tlačidla uložiť sa zavolá metóda *saveExam*.

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

```
getBoxId(boxId: string) {
    /* uchovava cislo otvorenego boxu */
    this.box = boxId;
}
appendNumberIntoBox(number) {
    // prida cisla do boxov v trojuholniku
    $(`#${this.box}.empty());
    `(`<img style='width: 59px;height: 59px; left: -1px;top: -1px;' src=....../assets/img/Number/"${number}.jpg alt=Cislo_"${number}"></a>`)
    .appendTo(`#${this.box}`);
    this.priklad[this.box] = number;
    this.prikladJson = JSON.stringify(this.priklad);
}
```

Obrázok 28 zdrojový kód pridávania hodnoty do prázdnego pola v šablóne

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 vracia zoznam všetkých tried, ktoré obsahujú *TeacherID* rovnaké ako prihlásený používateľ.

5.5.4 Priradenie úloh ku triedam

Po vygenerovaní tried môžeme pri každej vidieť 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 boli pri testovaní sledované, miesto testovania, samotný priebeh i zhodnotenie testovania.

6.1 Prvé testovanie

Prvé testovanie prebehlo 12. apríla 2018 v Cirkevnej základnej škole Narnia. Cieľom testovania bolo preveriť funkčnosť aplikácie v praxi, ale aj reakcie žiakov na interface aplikácie.

Počas testovania 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

Na začiatku testovania sme žiakov oboznámili s aplikáciou. Nakoľko žiaci už prostredie súčtových trojuholníkov poznali, nebolo potrebné im toto prostredie opäť 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 zahrať. Neskôr sa už začali vzájomne slovne informovať o tom, kto má akú úlohu. Toto bol podnet pre nás, nabádať ich objavovať ďalšie možnosti aplikácie. V prvej polovici testovania však naše podnetы niekoľkokrát stroskotali. Zhruba v polovici testovania sa nám naskytla modelová situácia využitia zdieľania úloh. Jedna zo žiačok 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 funkcionalite a následne ju vo veľkom využívať. Tu vznikla chvíľková hystéria. Žiaci sa začali prekrikovať, kto s kým rieši úlohy,

a tak to vyzeralo do konca testovania. Na konci testovania sme sa každého žiaka spýtali na dojmy z testovania aplikácie, rovnako i na to, čo by sme na aplikácii mali zmeniť a naopak, čo by malo ostať zachované.



Obrázok 29 žiaci v zápale riešenia úloh

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 funkcionalitou ju prijali a začali využívať. 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 niekoľkokrát 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. Podstatnými

boli pre nás i názory žiakov a pedagóga vykonávajúceho dozor. Počas testovania žiaci vypočítali spolu 164 príkladov, z ktorých 123 bolo vyriešených správne.

6.2 Druhé testovanie

Druhé testovanie je naplánované na druhú polovicu mesiaca máj a uskutoční sa opäť na základnej škole Narnia. Druhý test bude zameraný na využívanie aplikácie pri kooperatívnej práci v skupinách. Pri teste budú žiaci rozdelení do skupín a každá skupina dostane sériu úloh z učiteľského rozhrania. Počas testovania budeme sledovať ako žiaci dokážu vďaka aplikácii spolupracovať a riešiť sady úloh.

Záver

Cieľom našej práce bolo vytvoriť aplikáciu, ktorá podporí spoluprácu a komunikáciu medzi žiakmi. Počas vývoja aplikácie sme si museli spraviť podrobnejšiu analýzu Hejného metódy a princípov kooperatívneho vyučovania, aby sme si mohli spraviť lepší obraz pri následnom výbere technológií a návrhu aplikácie.

Nakoľko sme mali danú technológiu Unity 3D, museli sme vhodne doplniť technológiu na samotnú komunikáciu medzi zariadeniami a webovým rozhraním a technológiu na vývoj webového rozhrania. Po dlhšej štúdii možných frameworkov a komunikačných rozhraní sme sa rozhodli pre využitie technológie Firebase doplnenú o framework Angular 2 pre webové rozhranie.

Pri tvorbe návrhu sme vychádzali z funkčných požiadaviek, ktoré sme si predom zadefinovali. Počas tvorby návrhu sa dbalo aj na to, aby boli ovládacie prvky intuitívne pre dieťa a neodvádzali pozornosť tým, že musí samo skúmať funkčnosť jednotlivých prvkov.

Výsledkom predchádzajúceho výskumu je mobilná aplikácia podporujúca spoluprácu medzi žiakmi a webové rozhranie pre učiteľa, ktorý môže žiakom zadávať rôzne úlohy. Pomocou aplikácie môžu žiaci riešiť úlohy spoločne, alebo ich zdieľať na virtuálnej tabuli. Učiteľ zase môže sledovať štatistiky žiakov, pridávať im úlohy, aplikácia mu umožní rozdeliť žiakov do skupiniek, ale aj skontrolovať ktoréhokoľvek žiaka.

Počas testovania aplikácie s deťmi sme sa zameriavali na to, ako žiaci reagujú na aplikáciu, ako chápu ovládacie prvky a nakoniec samotné využívanie aplikácie. Výsledok testovania môžeme označiť ako pozitívny. Žiaci na aplikáciu reagovali kladne a pri ovládacích prvkoch im nebolo potrebné nič vysvetľovať. Dokonca pri odhalení funkcie zdieľania vznikol menší ošial. Aplikácia sa počas testovania správala podľa špecifikácie a nenarazili sme na žiadnu nepredvídanú situáciu, ktorá by znemožnila ďalšie testovanie.

Do budúcnosti plánujeme vyladiť komunikáciu a synchronizáciu medzi zariadeniami, rýchlosť samotnej aplikácie. Tiež plánujeme doplniť animované prvky do aplikácie, čím by sme zvýšili atraktivitu aplikácie. Ďalším krokom bude testovanie aplikácie na zariadeniach

od spoločnosti Apple a následné uverejnenie aplikácie na Google Play a App Store. Veríme, že naša práca by si mohla nájsť uplatnenie vo vyučovacom procese.

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] Google, Angular, [cit. 14.4.2018],
Dostupné na <https://angular.io/>
- [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>
- [13] PhDr. Božena Petrášová, (2010) ORGANIZAČNÉ SÚSTAVY VYUČOVANIA,
Univerzita sv. Cyrila a Metoda v Trnave, Filozofická fakulta

- [14] Dirner, A., et al., Virtuálna kolaborácia. Využitie nových informačno-komunikačných technológií vo výučbe fyziky
- [15] BYRIL, TicTacToe Online, [cit. 28.4.2018]
Dostupné na <https://play.google.com/store/apps/details?id=com.byril.tictactoe>
- [16] KKZAP Word Online Games, Word Search Online, [cit. 28.4.2018]
Dostupné na
<https://play.google.com/store/apps/details?id=com.best.word.search.online.multiplayer>
- [17] Apple, Education - Tools for Teaching - Apple, [cit. 28.4.2018]
Dostupné na <https://www.apple.com/education/teaching-tools/>
- [18] Maroš Žofčin, Apple staval na iPady v školách. Ukázal balík aplikácií pre učiteľov, [cit. 28.4.2018]
Dostupné na <https://www.zive.sk/clanok/131285/apple-stavil-na-ipady-v-skolach-ukazal-balik-aplikacii-pre-ucitelov/>

Prílohy

Prílohou práce je CD, ktoré obsahuje aplikáciu aj so zdrojovými kódmi a diplomovú prácu vo formáte PDF.