Eduard Krivánek, Tvorba Internetových aplikácii, Špecifikácia
Malý moodle

1. Popis aplikácie

Malý moodle je aplikácia do edukačného prostredia, slúži na uľahčenie testovania žiakov z rôznych predmetov a následne ich známkovanie. Cieľovou skupinou sú teda učitelia a žiaci nejakého školského ústredia.
V aplikácii si učitelia môžu vytvoriť kurzy, nastaviť minimálny počet bodov na známku a odoslať pozvánku cvičiacim či žiakom, ktorú je potrebné potvrdiť. Taktiež žiak môže požiadať stať sa členom kurzu, ktorú musia cvičiaci potvrdiť. Následne cvičiaci daného kurzu vytvárajú testy žiakom priamo v aplikácii, pomocou dynamických formulárov, ktorej výsledná forma musí byť potvrdená učiteľom, aby sa to žiakovi zobrazilo.

Žiak sa môže zúčastniť testu len v danom časovom intervale, kedy je test otvorený a má preddefinovaný čas na jeho ukončenie. Aplikácia pri vypracovávaní testu zaznamenáva koľko času študentovi trvalo zodpovedanie jednotlivých otázok (rozdiel času medzi kliknutou odpoveďou na danú otázkou a predošlou otázkou) a taktiež čas, ktorý pri vypracovaní testu strávi na inej stránke. Pri online testoch je častým zvykom podvádzať, preto budeme merať ako dlho sa žiak nachádza mimo aplikácie počas vypracovávania testu.

Test žiakovi sa uzavrie v momente keď ho dokončí, alebo ak jeho čas prekričí dovolený čas vypracovania testu. Následne učiteľ môže prideliť cvičiaceho, alebo cvičiaci samého seba na obodovanie odpovedí žiaka, s možnosťou napísania komentára k odpovediam.

Žiaci a učitelia vidia dokopy koľko bodov účastníci kurzu dosiahli s rozdielom, že žiak si môže pozrieť len vlastný ohodnotený test, pričom cvičiaci si môže pozrieť test hociktorého žiaka. Systém priebežne generuje známku žiakovi podľa preddefinovaných nastavení učiteľom, s tou možnosťou, že učiteľ môže opraviť výslednú známku žiakovi.

Na záver aplikácia bude generovať logy pre užívateľom, ako napríklad, že test žiaka bolo ohodnotené a získal X bodov, či test bol pridelený na istého cvičiaceho a musí ho skontrolovať.

2. Typy používateľov a právomoci

Typy používateľov v aplikácii sú nasledovne

- Anonymous
- Authorized
- Participant
- Marker
- Teacher
- Admin

Anonymous
Väčšina funkcionalít v aplikácii bude požadovať autentifikáciu užívateľa, ale nie je to prvoradou nutnosťou. Užívateľ bez prihlásenia má rolu anonymous, ktorá mu umožňuje len vylisovať existujúce kurzy v aplikácii. Neuvidí ale žiadne testy či členov daného kurzu. Ak sa chce stať žiakom vybraného kurzu, musí sa prihlásiť.

Authorized
Túto rolu má užívateľ v prípade, keď je prihlásený do aplikácie. Rola mu umožňuje nasledujúce funkcionality:

- Prihlásenie sa na kurz
- Vyhľadávať členov v aplikácii

Participant
Táto rola je priradená k príslušnej skupine. Užívateľ ju teda získa ak je členov (študentov) nejakej skupiny. Táto rola mu umožňuje:

- Vidieť ďalších členov kurzu
- Získať notifikáciu o blížiacom sa teste
- Vypracovať test vytvorený v danom kurze
- Byť známkovaný v danom kurze

Marker
Táto rola je priradená k príslušnej skupine. Užívateľ ju teda získa ak je hodnotiacim (cvičiacim) nejakej skupiny. Táto rola mu umožňuje

- Vytvoriť test pre žiakov skupiny
- Opraviť a oznámkovať vypracované testy študentov
- Pozvať či vyhodiť študenta z kurzu

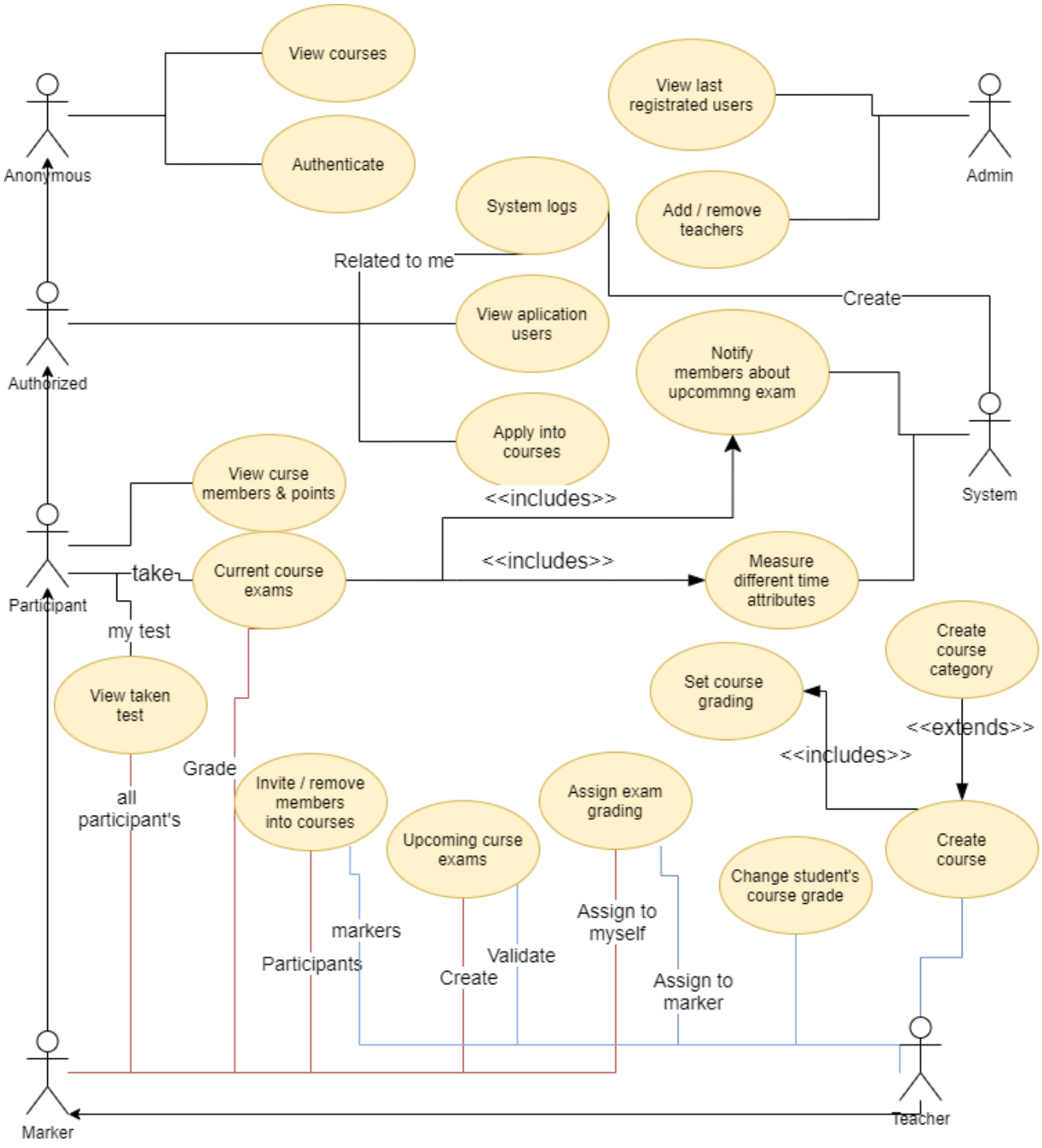
Teacher
Táto rola užívateľovi môže byť pridelená len administrátorom. Umožňuje nasledujúce veci

- Vytvoriť kurz či kategóriu kurzu
- Schváliť v kurze test vytvorený cvičiacim
- Pozvať či vyhodiť cvičiaceho zo svojho kurzu
- Editovať výslednú známku študenta vo svojom kurze

Admin
V aplikácii bude existovať len jeden administrátor. Administrátor má nasledujúce vlastnosti

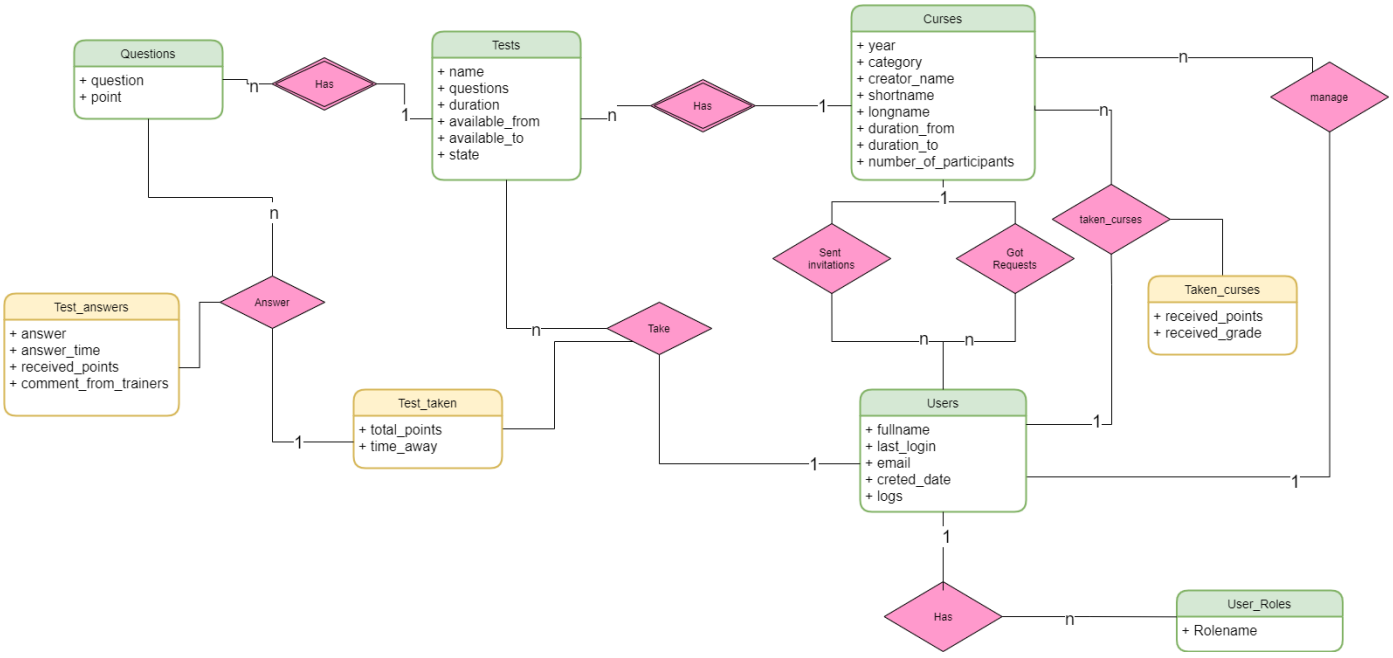
- Môže užívateľom pridať či odobrať rolu teacher
- Vidí niekoľko posledných registrovaných užívateľov do aplikácie

3. User case diagram

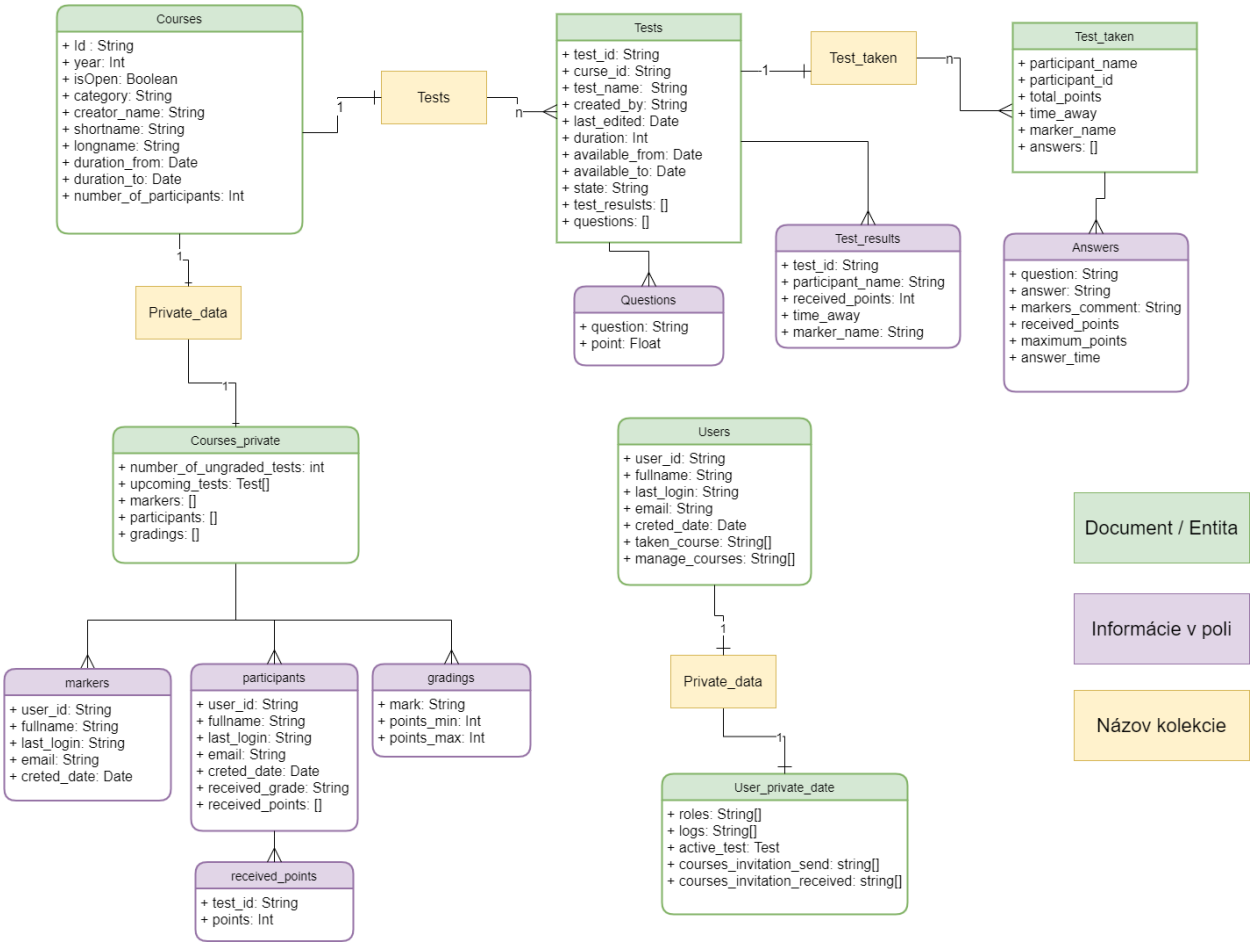


4. Databáza

Ako databáza sa bude používať cloudová databáza firebase, ktorá patrí do typu NoSql databáz. To znamená, že nenasleduje žiadnu preddefinovanú štruktúru a dáta môžu byť nenormalizované, tj. jedna entita (napríklad užívateľ) sa môže vyskytovať na viacerých miestach.
Toto riešenie je bližšie ku klientovi, pretože všetky potrebné údaje môžeme držať v jednom dokumente, tým pádom zaniká potreba vytvárania zložitých joinových operácii. Entitno relačný diagram môžeme vidieť na nasledujúcom obrázku.



Ak by sme šli cestou tradičnej sql databázy, všetky typy relácii (ružové rámčeky) by boli vlastné tabuľky. Keďže používame firebase a máme denný limitovaný počet read / write operácii do našej databázy, je oveľa múdrejšie donormalizovať entity, ktorých atribúty sa nebudú meniť a dať ich do jedného dokumentu. Pripájam taktiež návrh NoSql databázy, ktorú budem implementovať.



5. Popis tabuliek

Krátky popis ku návrhu databázy. Firebase používa logiku, kde si vytvoríme jednu kolekciu, ktorá môže obsahovať viacero dokumentov s unikátnym ID-čkom, do ktorého sa ukladá štruktúra dát v json tvare. To znamená, že do jedného dokumentu môžeme uložiť okrem základných typov ako number, string, boolean, aj objekty typu map a list. Taktiež jeden dokument môžeme obsahovať viacero pod kolekcií s viacerými dokumentami v každej kolekcii a takto do nekonečna. My keď robíme query do databázy, tak stále získame všetky údaje, ktoré sú v jednom dokumente.

Na vyššie spomínanom obrázku zelenou farbou sú označené dokumenty. Atribút v dokumente, ktorý je typu list, som rozpísal do vlastných tabuliek označenou fialovou farbou. Treba si ale pamätať, že to nie je ďalšia tabuľka, ale objekt, ktorý sa bude ukladať do tohto listu. Ak istý dokument (napr. Courses) obsahuje ďalšiu kolekciu dokumentov, ktoré patria k tejto entite (napr. Tests), tak túto kolekciu som označil oranžovou farbou. Nižšie rozpisuje význam kolekcii a atribútov, ktoré sa budú nachádzať v jednom dokumente. Občas zamieňam slová dokument a tabuľka, ale v kontexte sa jedná o to isté.

5.1) Kolekcia **courses**

Keďže vytvárame aplikáciu, kde študenti sa môžu zúčastniť ľubovoľných kurzov , tak jedna z najdôležitejších tabuliek budú courses, ktorá obsahuje všetky informácie daného kurzu. Entity do tejto tabuľky môže vytvoriť len užívateľ, ktorý má rolu učiteľa.
Entita v tomto dokumente má nasledujúce atribúty:

- course_id – id kurzu
- year – rok v ktorom bol vytvorený
- category – všeobecná kategória kurzov (napríklad INF / CHEM ..)
- shortname – skratka kurzu
- longname – celé meno kurzu
- duration_from – dátum začiatku kurzu
- duration_to – dátum konca kurzu
- number_of_ungraded_tests – počet neohodnotených testov
- upcoming_tests – potvrdené testy, ktoré musia žiaci vypracovať
- markers – cvičiaci kurzu
- participants – žiaci kurzu
- gradings – počet bodov potrebných na danú známku

5.2) Kolekcia **Courses_search**

V aplikácii chceme implementovať vyhľadávanie kurzov. Prvý myšlienka by mohla byť, že vyhľadávanie kurzov budeme robiť z kolekcie courses. To je ale neefektívny postup, pretože ako som vyššie spomínal, firebase vracia všetky fieldy v danom dokumente. Čiže v tomto prípade pri vyhľadávaní kurzu by sa na stranu klienta posielali aj dáta ohľadne žiakoch či cvičiacich v danom kurze.
Preto musíme vytvoriť ďalšiu kolekciu, v ktorom dokumenty budú rozdelené podľa kalendárneho roku a všetky kurzy, ktoré sa vytvorili v danom roku, budú uložené do poľa kurzov. Týmto spôsobom jedným čítaním vrátime klientovi všetky kurzy, ktoré sa vytvorili v jednom roku.
Dokument má atribúty

- year – rok v ktorom boli kurzy vytvorené
- categories – existujúce kategórie kurzu (napr. AIN, INF, atď.)
- courses – pole kurzov, ktoré sa v danom roku vytvorili.

5.3) Kolekcia **Tests**

Kolekcia obsahuje všetky vytvorené testy daného kurzu. Napr. testy, ktoré cvičiaci vytvorili, ale ešte nie sú schválené učiteľom, tým pádom ich študenti nevidia, až po testy, ktoré študenti absolvovali a boli z nich ohodnotený.

- test_id – id testu
- curse_id – id kurzu ku ktorému patrí test

- test_name – meno testu
- created_by – meno výtvorcu testu
- last_edited – dátum poslednej úpravy
- duration – čas za koľko je potrebné odovzdať test od otvorenia
- available_from – čas od ktorého je prístupný test pre žiakov
- available_to – čas do ktorého je prístupný test pre žiakov
- state – stav v akom je test (vytváraný / schválený ..)
- test_results – pole celkových bodov, ktoré jednotlivý študenti získali
- questions – pole otázok na test

5.4) Kolekcia **Test_taken**

Kolekcia obsahuje vypracovaný test študenta, jeho odpovede na jednotlivé otázka, dosiahnuté body a možné komentáre cvičiaceho ku odpovediam študenta.

- participant_name – užívateľ ktorý vypracoval test
- participant_id – id študenta, ktorý vypracoval test
- total_points – získané body
- time_away – čas, ktorý strávil mimo obrazovky aplikácie
- marker_name – meno cvičiaceho, ktorý hodnotí test
- answers – pole odpovedí žiaka na otázky testu

5.5) Kolekcia **users**

Okrem základných atribútov ako meno, email a pod. by som vyzdvihol atribúty taken_curses a manager_curses, ktoré sú typu pole stringov.

- manage_curses – pole ID kurzov, ktoré učiteľ vytvoril
- taken_curses – pole ID kurzov, ktorých sa užívateľ zúčastnil

5.6) Dokument **user_private_data**

V aplikácii taktiež máme nielen vyhľadávanie kurzov, ale aj vyhľadávanie užívateľov. Čo ale nechceme, je vracať na stranu klienta súkromné informácie užívateľa, ako napríklad: role užívateľa, logy, kurzy do ktorých sa chce prihlásiť, preto tieto informácie musíme taktiež rozdeliť do samotnej kolekcie, ktorý bude obsahovať len jeden dokument s názvom user_private_data s nasledujúcimi atribútmi.

- roles – pole rolí, ktoré má užívateľ v aplikácii
- logs – logy vygenerované pre užívateľa (napr. že jeho test bol obodovaný)
- active_test – test, ktorý aktuálne užívateľ vypracováva
- curses_invitation_received – pole prijatých žiadostí do kurzov
- curses_invitation_sent – pole odoslaných žiadostí do kurzov

6. Technologické požiadavky

Aplikácie bude vyznačená architektúrou hrubého klienta, čo znamená, že všetky dáta sa budú modifikovať na strane klienta. Pre zložitejšie webové aplikácie je dobré použiť istý framework. V mojom prípade na strane klienta budem používať angular.
Ako UI knižnica v projekte bude použitý IONIC.
Keďže aplikácia má byť verejné prístupná, bude ju treba zavesiť na istého cloudového poskytovateľa. V mojom prípade využijem Firabase, ktorý funguje ako Backend as a service (BAAS) a umožní nám zavesiť naň našu aplikáciu. Taktiež ponúka NoSql databázu s názvom firestore, ktorá posiela updaty klientom v reálnom čase, ktorí počúvajú na zmeny istých dokumentov.

7. Časový plán

Týždeň 5. (8.03 - 14.03)
- vytvorenie kostry aplikácie (komponenty a tabuľky) – 12h
- navigácie medzi komponentami – 2h

Týždeň 6. . (15.03 - 21.03)
- vytvorenie nového kurzu – 8h
- vytvorenie novej kategórie kurzu – 1h
- pozvať žiakov a cvičiacich do kurzu – 2h
- poslať pozvánku (žiaci) do kurzu – 1h
- zobrazenie pozvánok užívateľom do kurzu – 2h

Týždeň 7. + Týždeň 8. (22.03 - 04.04)
- vytvorenie dynamických formulárov (testy) – 10h
- notifikovanie žiakov o blížiacom sa teste – 6h
- vypracovanie testu len v časovom intervale odomknutia testu (available_from) – 2h
- automatické submitnutie testu po prekročení času (duration / available_to) – 2h
- meranie času koľko užívateľ strávil mimo obrazovky aplikácie – 5h
- meranie času koľko užívateľ strávil medzi odpoveďami na jednotlivé otázky – 5h

Týždeň 9. (05.04 - 11.04)
- ohodnotenie vypracovaných testov žiakom – 6h
- oprava výslednej známky žiakom (len učiteľ) – 1h
- zabezpečenie rút a funkcionalít v aplikácie len pre oprávnených – 6h
- vygenerovanie testovacích dát do databázy. – 6h

Týždeň 10. (12.04 – 18.04)

- doplynutie na firestore – 1h
- generovanie logov pre užívateľov – 4h
- vyhľadávanie kurzov v aplikácii – 2h
- vyhľadávanie užívateľov v aplikácii – 1h

Týždeň 11. (19.04 – 25.04)

- dorábanie nedokončených vecí
- testovanie – 3h

Optimistický odhad na dokončenie funkcionalít vylisovaných v časovom pláne je zhruba 88 hodín.

Eduard Kriváneк