FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Trainer - Kvízový a testový modul pre webový portál na podporu výučby
Student:	Matej Pašek
Vedoucí:	Ing. Jan Matoušek
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 14. února 2024 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Pokyny pro vypracování

Jedným z hlavných problémov klasickej formy výučby na cvičeniach, alebo obecne akýchkoľvek skupinových vyučovacích hodinách býva aktivita zo strany študentov. Cieľom projektu Trainer je, poskytnúť študentom nástroj – webovú aplikáciu, ktorá umožní všetkým študentom sa zapojiť do priebehu cvičenia, poprípade v prípade nejakých nejasností sa anonymne prihlásiť o pomoc. Hlavným cieľom tejto aplikácie zo strany učiteľa je zjednodušenie alebo úplné zautomatizovanie niektorých manuálnych procesov, napríklad nutnosť si zapamätať všetkých študentov ktorým má zapísať bonusové body. Cieľom tejto práce je vytvoriť moduly spoločných kvízov a precvičovacích testov do tohto systému.

Pokyny k vypracovaniu:

1) Analyzujte už existujúce konkurenčné riešenia v oblasti skupinových kvízov a hodnotených testov.
2) Analyzujte požiadavky pre uvažované moduly.
3) Identifikujte a navrhnite základné procesy z požiadavkou a analýzy konkurenčných riešení, navrhnite aplikačnú architektúru.
4) Zvolte optimálne technológie pre vývoj projektu.
5) Na základe návrhu vyviňte prototyp, ktorý bude splňovať business požiadavky.
6) Otestujte prototyp rôznymi druhmi testov.
7) Spätne analýzujte vývoj a navrhnite nové budúce funkcionality a navrhnite vylepšenia tých existujúcich.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 14. února 2024 v Praze.

Bakalárska práca

TRAINER - KVÍZOVÝ A TESTOVÝ MODUL PRE WEBOVÝ PORTÁL NA PODPORU VÝUČBY

Matej Pašek

Fakulta informačních technologií
Katedra softwarového inžinierstva
Vedúci: Ing. Jan Matoušek
16. mája 2024

Odkaz na túto prácu: Pašek Matej. *Trainer - Kvízový a testový modul pre webový portál na podporu výučby.* Bakalárska práca. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov kódu

Rád by som poďakoval môjmu vedúcemu Ing. Janovi Matouškovi, za jeho pomoc, trpezlivosť a spätnú väzbu pri tvorbe tejto práce. Ďalej by som rád poďakoval svojej rodine a partnerke za ich dlhoročnú podporu pri mojom štúdiu a obecne v mojom živote. V neposlednej rade by som rád poďakoval všetkým, ktorí sa podieľali v nejakej forme na tejto práci, konkrétne tímu študentov z predmetu BI-SP1 a všetkým študentom a učiteľom, ktorí mi počas posledných mesiacov poskytovali spätnú väzbu.

Vyhlásenie

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

V Praze dňa 16. mája 2024

Abstrakt

Cieľom tejto práce je popísať proces vývoja kvízového a testového modulu pre webový portál Trainer, ktorý slúži na podporu výučby vybraných predmetov na FIT ČVUT. Tento proces prebieha v niekoľkých fázach - analýza, návrh, implementácia a následné testovanie. Pre vývoj serverovej časti aplikácie, ktorá vystavuje REST API, je použitý framework Spring, pri vývoji webového rozhrania je použitý framework VueJS. Po vyvinutí sú tieto moduly následne integrované do systému Trainer a to ako serverová časť tak webové rozhranie, čím je dosiahnutý základný cieľ tejto práce - vytvorenie a integrácia týchto modulov do systému, aby mohli byť následne používané na cvičeniach (resp. obecne pri výučbe), alebo na individuálne štúdium. Aplikácia Trainer, spolu s týmito modulmi, je priebežne vyvíjaná a je dostupná na webovej stránke, po prihlásení ČVUT identitou. V prílohách tak isto je možné vidieť dokumentáciu API pre tieto moduly.

Klíčová slova Kvízová aplikácia, Webový portál, Výučbový portál, Spring framework, Kotlin, Vue.js, Samostatný test

Abstract

The aim of this thesis is to describe the process of developing a quiz and test module for the Trainer web portal, which is used to support the teaching of selected courses at the CTU FIT. This process is carried out in several phases - analysis, design, implementation and subsequent testing. For the development of the server part of the application, which exposes the REST API, the Spring framework is used, for the development of the web interface the VueJS framework is used. Once developed, these modules are then integrated into the Trainer system, both the server part and the web interface, thus achieving the main goal of this thesis - to create and integrate these modules into the system so that they can be subsequently used in tutorials (or teaching in general) or for individual study. The Trainer application, along with these modules, is continuously being developed and is available on a website, after logging in with a CTU identity. The API documentation for these modules can also be seen in the attachments.

Keywords Quiz application, Web portal, Educational portal, Spring framework, Kotlin, Vue.js, Selftest

Seznam zkratek

API	Application programming interface
UI	User Interface
UX	User Experience
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
CRUD	Crate Read Update Delete
JSON	JavaScript Object Notation
CSV	Comma-Separated Values
XML	eXtensible Markup Language
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
IOC	Inversion Of Control
WSGI	Web Server Gateway Interface
JVM	Java Virtual Machine
JRE	Java Runtime Environment
GC	Garbage Collector
SQL	Structured Query Language
JPQL	Java Persistence Query Language
JPA	Java Persistence API
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SFC	Single File Component
DTO	Data Transfer Object
MVC	Model View Controller
ORM	Object-Relational Mapping

Úvod

Vykonávanie rôznych aktivít v spoločnosti neznámych ľudí, je pre nemalé množstvo ľudí nepríjemné a štúdium nie je výnimkou. Jedným z hlavných problémov na vyučovacích hodinách, či už sa jedná o základnú, strednú, alebo vysokú školu, je aktivita zo strany študentov. Toto sa dá pripísať viacerým faktorom, jedným z ktorých je práve diskomfort zapríčinený štúdiom v skupine s neznámymi ľudmi .

Len v málo prípadoch sú študenti aktívni sami od seba, napríklad keď ich preberaná tématika zaujíma, chcú sa dozvedieť viac, alebo niečím potrebujú pomôcť a nehanbia sa opýtať. Väčšinou sú študenti motivovaný k aktivite vyučujúcimi, napr. bonusovými bodmi, prípadne inými malými výhodami. Toto veľa krát nie je dostatočné a aj keby študenti chceli a aj mohli vo veľa prípadoch bonusový bod získať radšej ostanú z akéhokoľvek dôvodu ticho.

Každý vyučujúci sa na hodiny pripravuje inak, no prakticky každý si potrebuje pripraviť podklady, na ktorých hodinu bude stavať. V niektorých prípadoch to môže byť niečo jednoduché, ako napríklad pár odrážok textu, obsahujúce čo chce na danej hodine prebrať, inokedy to môžu byť úlohy, spoločné aktivity atď., na ktoré je vyučujúci nútený využiť rôzne externé aplikácie, obzvlášť, ak tieto aktivity chce mať v elektronickej forme. To so sebou prináša problém - ak vyučujúci chce vytvoriť viac úloh, rôznych typov, musí použiť viac aplikácií.

Projekt Trainer, má za úlohu uľahčiť proces výučby ako pre študentov, tak pre vyučujúcich. Prínos pre vyučujúcich je hlavne centralizovanie prípravy materiálov na hodiny - problém, ktorý som popísal vyššie a automatizácia niektorých činností ktoré musia vyučujúci aktuálne vykonávať manuálne. Študentom zas poskytne nástroj, pomocou ktorého sa budú môcť zapájať do priebehu cvičenia bez toho, aby sa museli prihlásiť, alebo vykonávať iné aktivity pred skupinou neznámych ľudí.

V tejto práci sa venujem 2 menším častiam - modulom, vrámci tohto projektu. Kvízový modul umožní vytvárať, spúšťať a hrať kvízy priamo v aplikácií Trainer. Testový modul má za úlohu pomôcť študentom s prípravou na výučbu, zápočet, skúšku, atď, pomocou testov, generovaných z preddefinovanej množiny otázok. Vytvorenie kvízového a testového modulu eliminuje potrebu využívať externú aplikáciu na tieto aktivity.

Najprv zanalyzujem už existujúce aplikácie zaoberajúce sa touto problematikou a požiadavky na mnou implementované riešenie. Následne na základe tejto analýzy namodelujem základné procesy v tejto aplikácií a navrhnem uzívateľské rozhranie. Na základe návrhu potom prejdem k výberu technológií a samotnej implementácií a otestovaniu implementácie. Na záver zanalyzujem celý proces vývoja, čo sa dalo spraviť inak, lepšie a ako sa ďalej môžu tieto moduly rozvíjať v budúcnosti.

Projekt Trainer vznikol v spolupráci s Bc. Ondřejom Wrzieconkom, skupinou študentov z predmetu NI-NUR a BI-SP1 a vedúcim tejto práce Ing. Janom Matouškom.

Cieľ práce

Výstupom tejto práce, je vytvorenie kvízového a testového modulu, ktoré budú integrované do webového portálu projektu Trainer. Pri tvorbe týchto modulov budem dbať na základné princípy softwarového inžinierstva, preto základnými cieľmi ktorých sa v tejto práci budem držať sú:

1. Analýza existujúcich modelov, ktoré riešia rovnakú/podobnú problematiku.
2. Analýza požiadavkov na nami implementované riešenie.
3. Návrh užívateľského rozhrania a základných procesov
4. Implementácia prototypu kvízového a testového modulu na základe analýzy a návrhu z predchádzajúcich bodov.
5. Dôkladné otestovanie prototypu kvízového a testového modulu.

Kapitola 1
Analýza

V tejto kapitole sa budem venovať analýze už existujúcich aplikácií, ktoré sa zaoberajú kvízmi, otázkami v týchto kvízoch a taktiež niektorých aplikácií, ktoré sú veľmi rozšírené hlavne medzi študentmi za účelom samostatného štúdia. Následne sa budem venovať analýze požiadavkov na nami implementované riešenie.

1.1 Analýza existujúcich riešení

Aplikácií, ktoré poskytujú podobnú funkcionalitu ako je tá, ktorú vyvíjam vrámci projektu Trainer je na internete hneď niekoľko. V tejto podkapitole sa preto venujem, niekoľkým aplikáciám, ktoré sa používajú najčastejšie - či už pre spoločné skupinové kvízy, alebo na vytváranie precvičovacích testov pre jednotlivcov, popisujem ich využitie a základné funkcionality, z ktorých sa môžeme inšpirovať pri návrhu a následnej implementácií kvízového resp. testového modulu.

1.1.1 Kahoot

Kahoot [1] je komplexná aplikácia, dostupná nielen ako webstránka, ale aj ako aplikácia na mobilné zariadenia. Ich cieľovou skupinou užívateľov sú nielen školy – teda študenti a učitelia, ale tak isto firmy a individualisti, resp. skupiny ľudí, ktoré nijako nesúvisia so školským alebo pracovným prostredím. Pre každú z týchto "kategórií" poskytuje Kahoot rôzne (v niektorých prípadoch platené) plány. Ďalej Kahoot ponúka možnosť zapájať sa do rôznych vopred pripravených hier a kvízov. Základnou funkcionalitou je plnenie kvízov. Ten je tvorený rôznymi druhmi otázok (výber z n možností, true/false, puzzle, ...).

V prípade potreby mať v zadaní otázky médium – obrázok, gif atď. Kahoot ponúka takéto menu s možnosťou výberu z obrázkov online, alebo uploadu zo zariadenia.

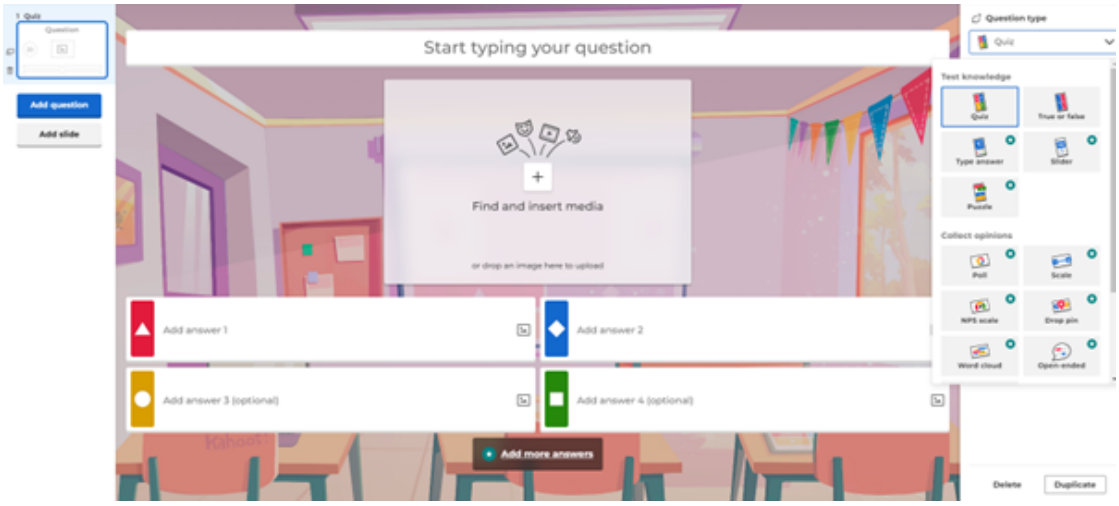
Možnosti výberu je možné zadávať v plaintexte, ako matematické formule, či obrázky. Kedykoľvek počas tvorby kvízu je možné ho uložiť. Aplikácia automaticky upozorňuje na prípadné chyby v kvíze – chýbajúce texty, neoznačená správna odpoveď atď.

Po spustení kvízu najprv vedúci kvízu počká kým sa všetci participanti pripoja do „kvízovej miestnosti". Následne kvíz odštartuje a zobrazí sa prvá otázka.

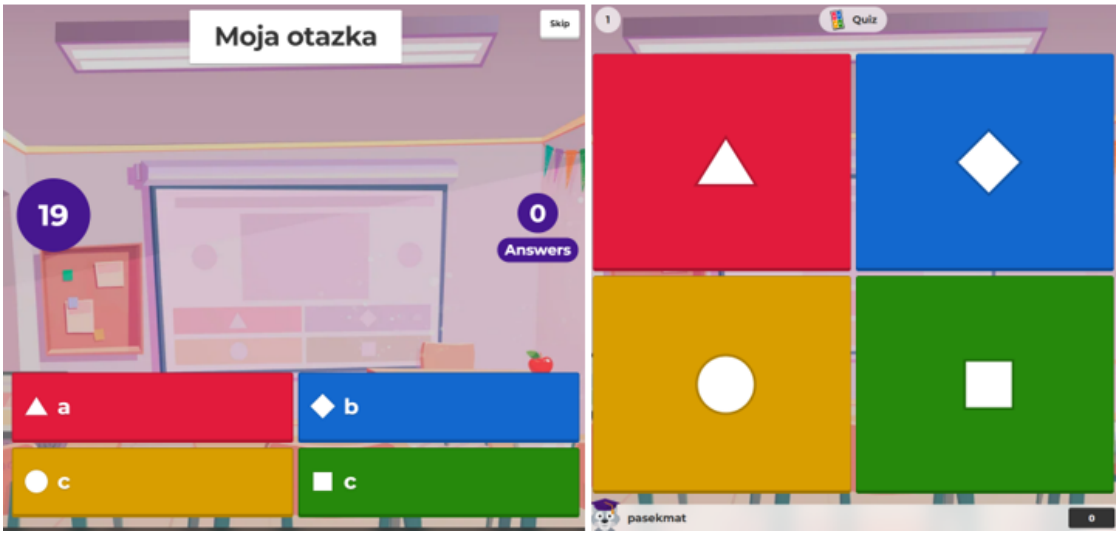
Kvíz resp. každá otázka je spúšťaná učiteľom a synchronizovaná medzi všetkými študentami. Text otázky a odpovedí vidí len učiteľ, študenti naopak vidia len časový odpočet a klikateľné kartičky ktorými vyberajú odpoveď. Z tohto dôvodu je potrebné mať v miestnosti veľkú obrazovku, alebo dataprojektor. Po každej otázke sa zobrazia štatistiky odpovedí participantov.

Po kvíze študenti majú možnosť zobraziť náhľad, prípadne si otázky na ktoré zle odpovedali zopakovať. Okrem klasických kvízov je možné vytvoriť si vlastné otázky a formou kartičiek si

ich opakovať. Prípadne je možné vytvoriť skupinu s ďalšími ľudmi a tieto kartičky v nej zdielať a učiť sa z nich hromadne.



Obr. 1.1 Rozhranie pre tvorbu kvízu v aplikácií Kahoot [1]



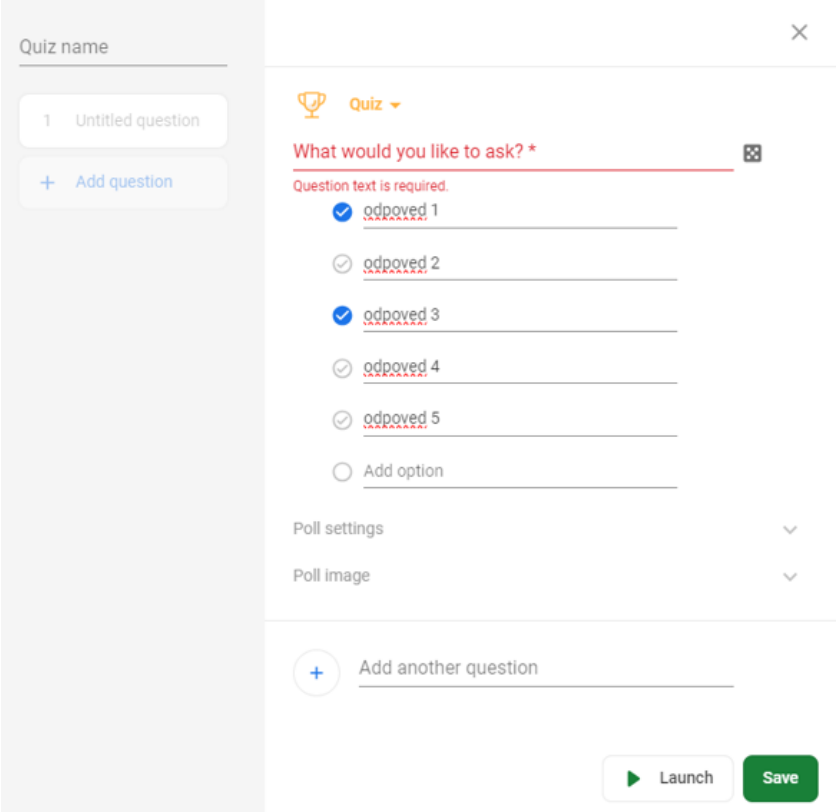
Obr. 1.2 Študentské (vľavo) a učiteľské (vpravo) rozhranie kvízovej otázky v aplikácií Kahoot [1]

1.1.2 Slido

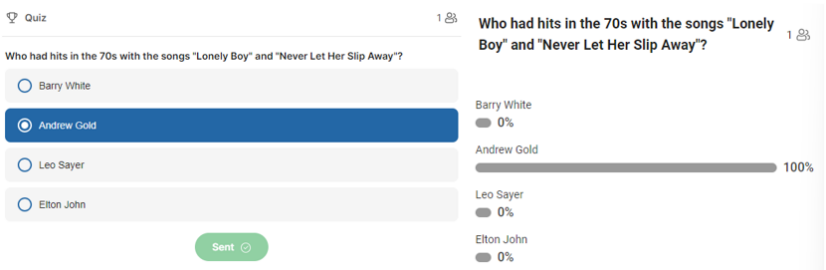
Slido [2] je aplikácia, dostupná nielen na ich webstránke, ale aj vo forme aplikácie na mobilné zariadenia. Je zameraná čisto na tvorbu hlasovaní („Polls“) v rôznych formách. Môže sa jednať napríklad o klasických kvíz alebo rating nejakého filmu. Aplikácia tak isto ponúka možnosť vytvoriť „poll“ z vopred pripravenej šablóny, ktorou môže byť aj už existujúci kvíz.

Po zvolení formy „pollu“ je možné pridať ľubovolný počet otázok. V každej otázke môže byť ľubovolný počet odpovedí. V momente keď je kvíz pripravený je možné ho spustiť. Po spustení vedúci kvízu počká, kým sa pridajú všetci participanti a následne ho spustí.

Nasleduje séria pripravených otázok na ktoré participanti odpovedajú. Po každej otázke sa zobrazia správne výsledky a štatistiky ktorá odpoveď bola vyberaná ako často. Učiteľ navyše odpovede študentov vidí priebežne ako prichádzajú v podobe grafov pri každej z možností.



Obr. 1.3 Rozhranie pre tvorbu kvízu v aplikácií Slido [2]



Obr. 1.4 Študentské (vľavo) a učiteľské (vpravo) rozhranie kvízovej otázky v aplikácií Kahoot [2]

1.1.3 Quizziz

Quizziz [3] je aplikácia slúžiaca na vytváranie interaktívnych kvízov. Je dostupná ako webová stránka a ako aplikácia na mobilné telefóny. Pri prvotnom prihlásení je užívateľ vyzvaný k výberu jeho role – študent, učiteľ atď. Nezávisle na výbere role majú užívatelia možnosť vytvoriť kvíz. Takisto môžu všetci užívatelia vyhľadávať medzi všetkými zverejnenými kvízmi. Učitelia majú

navyše možnosť vybraný kvíz spustiť pre ich skupinu a študenti majú následne možnosť sa do tohto kvízu pripojiť. Študenti môžu taktiež kvíz spustiť, ale iba pre samostatné precvičovanie. Študenti majú možnosť sa pripojiť do prebiehajúceho kvízu prakticky z akejkoľvek časti aplikácie za predpokladu, že aktuálne sa už neúčastnia iného kvízu.

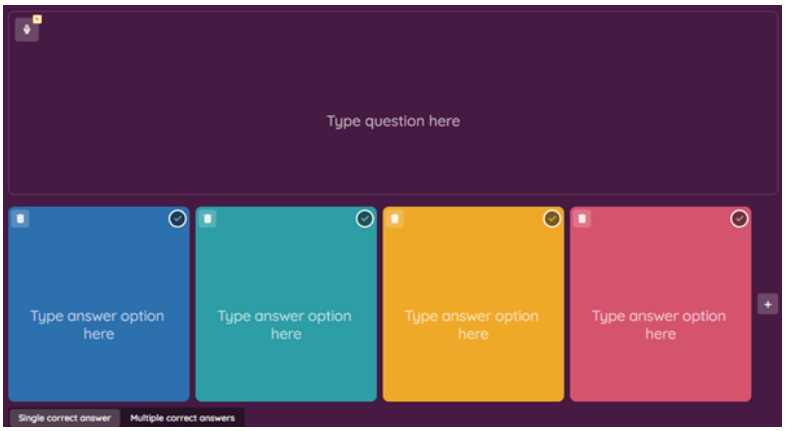
Tvorba kvízu je jednoduchá – spočíva len v pridávaní/importovaní otázok do kvízu, resp. vytváraní otázok a ich pridaním do kvízu. Pri tvorbe otázok je na výber z veľa rôznych typov otázok, pri každom type je ukážka ako daná otázka vo výsledku bude cca vyzerať.

Po vybraní typu otázky je potrebné samotnú otázku vyplniť a po uložení je automaticky pridaná do kvízu. Okrem vyplnenia textov otázky, odpovedí a označení správnych odpovedí je možné upravovať časový limit a bodové ohodnotenie.

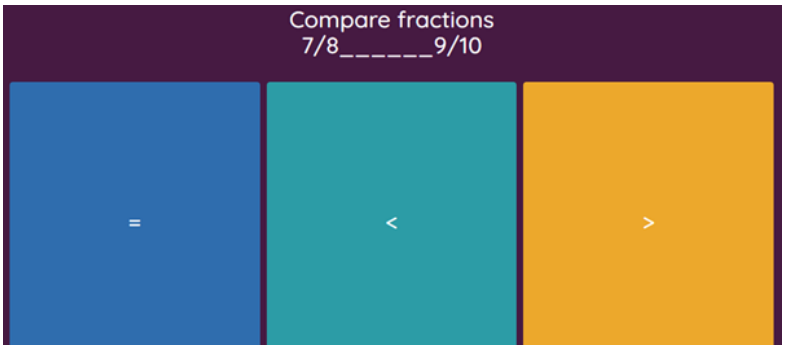
Nakoniec je potrebné kvíz publikovať (komu bude tento kvíz viditeľný sa dá customizovať v platenej verzii) a potom ho je možné spustiť, alebo ho prideliť niekomu ako úlohu. Aplikácia ponúka rôzne módy na spúšťanie kvízu:

- Klasický – každý študent môže ísť vlastným tempom
- Tempom inštruktora – inštruktor určuje, kedy sa zobrazí nová otázka

Po spustení kvízu je zobrazený kód pre študentov v podobe číselnej a v podobe QR-kódu, ktorým sa participanti môžu do kvízu prihlásiť. Po tom ako sa všetci participanti prihlásia inštruktor kvíz odštartuje a v závislosti na móde môžu študenti otázky v tomto kvíze vyplňovať.



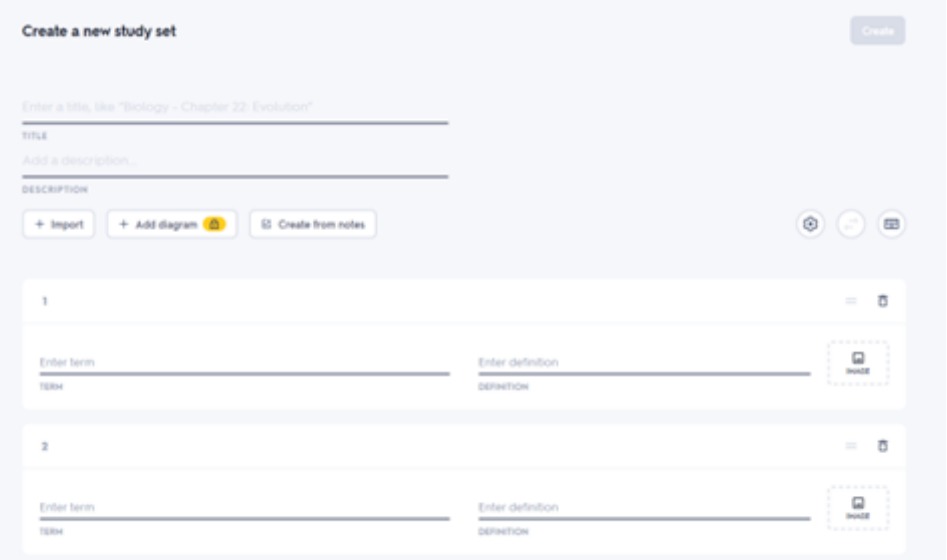
Obr. 1.5 Rozhranie pre tvorbu kvízu v aplikácií Quizziz [3]



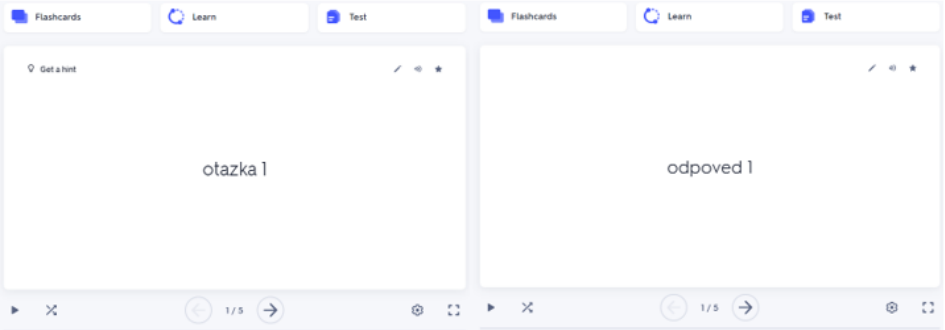
Obr. 1.6 Študentské rozhranie pre kvízovú otázku v aplikácií Quizziz [3]

1.1.4 Quizlet

Quizlet [4] je aplikácia dostupná ako webová stránka aj ako aplikácia pre mobilné telefóny. Jej základnou funkcionalitou je vytváranie setov otázok (a ich následné študovanie), ktoré je následne možné zverejniť vrámci skupiny ľudí – triedy (classy). Tieto sety sú tvorené kartičkami – na jednej strane majú otázku, na druhej strane odpoveď. Študovať sa dá v niekoľkých podobách – tou základnou je, prechádzanie cez všetky kartičky postupne. Ďalšou možnosťou štúdia je vygenerovanie testu s rôznymi typmi otázok, ktoré sú založené na kartičkách v danom sete. Keďže ku každej otázke je na originálnej kartičke len správna odpoveď musí systém niekedy hodnoty pre možnosti výberu odpovedi vygenerovať. Po odpovedaní na všetky otázky a odoslaní odpovedí aplikácia ukáže úspešnosť a študentove odpovede na otázky, prípadne správne odpovede v otázkach, ktoré študent zle zodpovedal. Okrem testu a postupného prechádzania kartičiek Quizlet ponúka mód intenzívneho učenia konkrétneho setu kartičiek. Funguje to tak, že z daného setu vyberie otázky a v rôznych formách ich študentovi postupne pokladá. Keď študent všetky otázky prejde, zobrazia sa mu štatistiky a môže pokračovať v učení, ďalším kolom otázok, ktoré sa môže líšiť od toho predchádzajúceho jedine vo forme v ktorej sú otázky pokladané.



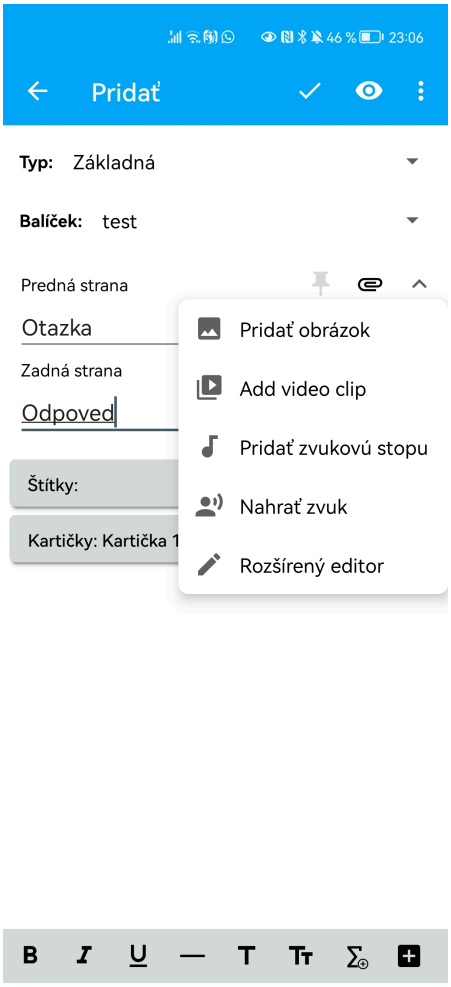
Obr. 1.7 Rozhranie pre tvorbu setu otázok v aplikácií Quizlet [4]



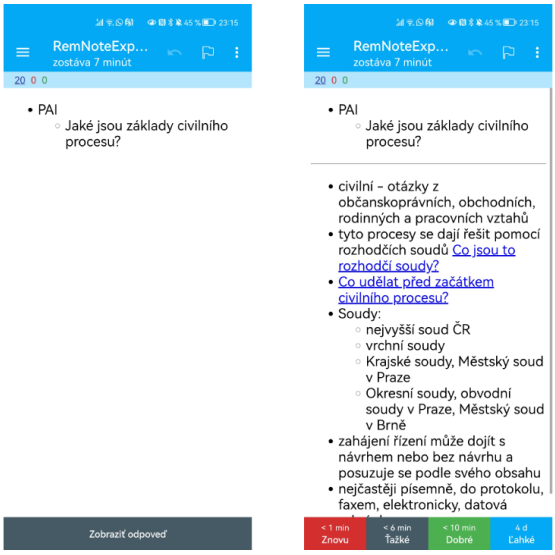
Obr. 1.8 Rozhranie pre otázku resp. kartičku v aplikácií Quizlet [4]

1.1.5 Anki

Anki [5] je aplikácia určená na samostatné štúdium pomocou kartičiek, podobne ako Quizlet – na jednej strane kartičky je otázka, na druhej strane odpoveď. Líši sa však v tom, že po tom ako užívateľ zobrazí odpoveď, či už ju vedel alebo nie, okrem odpovede mu aplikácia dá na výber z možností podľa toho, ako dobre poznal odpoveď na otázku. Na základe tohto výberu mu následne túto konkrétnu otázku po časovom intervale, ktorý je závislý od užívateľovej voľby. Otázky sú organizované do balíčkov, z ktorých si užívateľ môže vybrať, ktorý chce študovať. Do týchto balíčkov je možné otázky pridávať, upravovať, alebo odoberať. Balíčky je tak isto možné exportovať a zdieľať ich s inými užívateľmi, rovnako ich je možné importovať. Tvorba kartičiek (otázok) prebieha jednoducho – užívateľ resp. tvorca otázky, zadá text otázky, odpoveď na túto otázku a tým je kartička pripravená na použitie. Týmto kartičkám môžu byť priradené štítky, typ otázky, prípadne je možné k otázke alebo odpovedi pridať médium – fotku, video, zvukový klip atď.



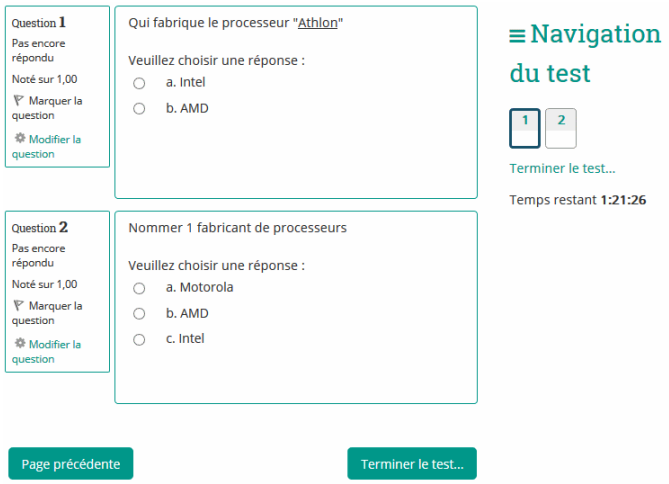
Obr. 1.9 Rozhranie pre tvorbu otázky v aplikácií Anki [5]



Obr. 1.10 Rozhranie pre otázku resp. kartičku v aplikácií Anki [5]

1.1.6 Moodle

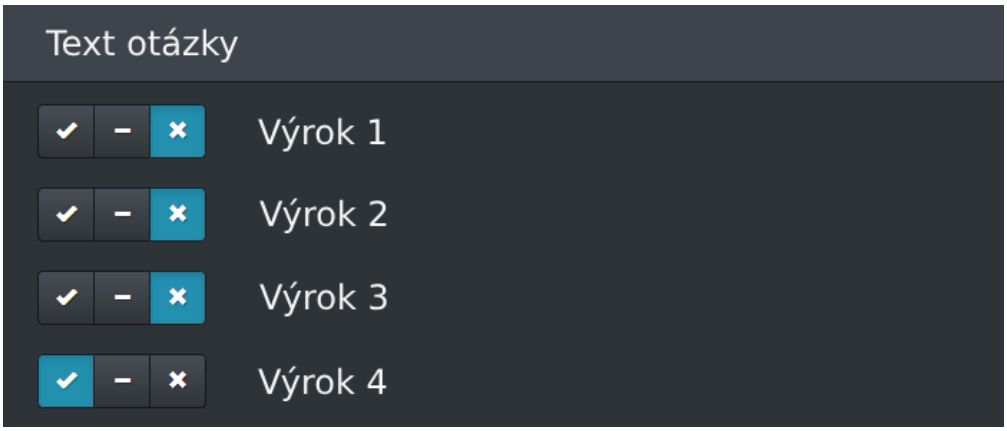
Moodle [6] je využívaný hlavne na školách, ako nástroj výučby. Je možné v ňom vytvárať kurzy a do kurzov pridávať lekcie, súbory, zadávať domáce úlohy atď. Systém tak isto podporuje pridávanie testov do kurzov . Tieto testy je možné konfigurovať – časovo obmedziť, kedy je test prístupný, prísnosť hodnotenia, definovať heslo na spustenie testu (ak je potrebné) atď. Test je zložený z preddefinovaného počtu otázok, študent si môže vybrať, či chce mať všetky otázky zobrazené na jednej stránke, alebo chce mať na jednej stránke zobrazenú len jednu otázku a medzi otázkami sa prepínať tlačítkami. Po zodpovedaní všetkých otázok, je študentovi zobrazený prehľad testu – ktoré otázky zodpovedal, ktoré nie, ktoré si označil, že chce ešte skontrolovať a podobne. Nakoniec tento test študent odošle k vyhodnoteniu a sú mu zobrazené správne odpovede spolu s tými jeho a bodové ohodnotenie.



Obr. 1.11 Rozhranie testu v aplikácií Moodle [7]

1.1.7 Marast

Marast [8] je výučbový systém FIT ČVUT, v ktorom majú študenti prístup k materiálom z vybraných predmetov a tak isto si môžu samostatne precvičovať úlohy. Webová aplikácia umožňuje výber z veľkého množstva rôznych filtrov, na základe ktorých sú študentovi zobrazované následne úlohy. Úlohy sa zobrazujú po jednom – vždy sa zobrazí zadanie, ktoré má študent možnosť vypracovať. Pod zadaním má možnosť zobraziť výsledok, prípadne zobraziť postup riešenia (ak je nejaký pri úlohe nachystaný). Aplikácia tak isto umožňuje organizovať skupinové kvízy. Táto funkcionalita je aktuálne najmä využívaná pri testoch, skúškach atď. Študentom je po otvorení testu (a prípadnom zadaní hesla), pod sebou zobrazených niekoľko otázok, na ktoré má odpovedať. Pod otázkami je možnosť uložiť rozrobenú prácu, alebo svoje riešenie odoslať k vyhodnoteniu. Po vyhodnotení je študentovi zobrazené bodové hodnotenie, spolu s tým ktoré otázky mal správne vyriešené a ktoré nie – nie je mu však ukázané správne riešenie.



Obr. 1.12 Rozhranie otázky v aplikácií Marast [8]

1.1.8 Duolingo

Duolingo [9] je aplikácia určená na samostatné štúdium. Je primárne rozšírená v podobe mobilnej aplikácie, kvôli jednoduchému používaniu a tomu, že vybrať mobil a zapnúť túto aplikáciu je oveľa jednoduchšie a rýchlejšie ako kvôli tomu zapínať počítač. Vďaka tomuto, spolu s faktom, že je dostupná zdarma a rôznym vtipom na sociálnych sieťach je veľmi populárna po celom svete. Táto aplikácia je zameraná na samostatné štúdium cudzích jazykov pomocou interaktívnych otázok a mini hier.

Aplikácia užívateľa pri prvotnom prihlásení vyzve k výberu jazyka, ktorý sa chce tento užívateľ naučiť. Po tom, ako je jazyk vybraný aplikácia zostaví plán, ktorého súčasťou sú lekcie. Tieto lekcie sú zložené z rôznych typov otázok zameraných na precvičovanie slov a gramatiky z vybranej lekcie. Aplikácia podporuje vizuálne typy otázok - kde je úlohou užívateľa pomenovať objekt na obrázku slovom v cudzom jazyku, zvukové typy otázok - kde je cieľom užívateľa zreprodukovať hlasovú nahrávku, ktorú si vypočul v textovej alebo rečovej forme a textové typy otázok, kde je väčšinou úlohou užívateľa preložiť zobrazený text.

Oproti iným aplikáciám, ktoré sú určené na samostatné štúdium táto aplikácia nepodporuje vytváranie vlastných otázok užívateľmi. Toto môže byť na jednej strane výhoda, pokiaľ užívateľa nezaujíma aké otázky dostane. Pre užívateľov, ktorý by si potrebovali zopakovať nejakú konkrétnu sekciu s veľmi konkrétne položenými otázkami by bol fakt, že nemajú možnosť tieto otázky vytvoriť, veľkou nevýhodou.

Obr. 1.13 Rozhranie otázky v aplikácií Duolingo [9]

1.2 Zber požiadavkov

Zber požiadavkov prebiehal v niekoľkých formách počas časového rozpätia asi 6 mesiacov. Niektoré z týchto požiadavkov, boli vyžadované od úplného počiatku, iné vznikli dodatočne z nových ideí po tom, ako bol už základ systému navrhnutý a implementovaný. Hlavnou metódou zberu požiadavkov boli pravidelné konzultácie s vedúcim práce a v prípade potreby pomocou správ cez aplikáciu Discord.

S niekoľkými nápadmi na vylepšenie prišli aj iní vyučujúci na FIT ČVUT, no tieto nápady boli skôr na úrovni prípadov užitia. Na požiadavok, pod ktorý daný prípad užitia spadá, teda tieto nápady mali vplyv, ale nie dostatočne veľký aby som ich klasifikoval ako samostatné požiadavky, či už funkčné, alebo nefunkčné.

1.3 Funkčné požiadavky

Náš systém, alebo konkrétne kvízový a testový modul sa samozrejme bude v niektorých aspektoch líšiť od už existujúcich riešení, ktoré som popísal vyššie. Z opakovaných rozhovorov a konzultácií vyplynuli nasledujúce požiadavky na nami implementované riešenie.

1.3.1 F1 Správa otázok

Základným komponentom akéhokoľvek kvízu sú otázky, na ktoré majú študenti odpovedať. Pri tvorbe kvízu je potrebné umožniť tvorcovi kvízu vytvoriť novú otázku, ktorá bude následne automaticky do kvízu pridaná. Nie vždy sa ale tvorba otázky musí podariť, alebo jednoducho môžu časom stratiť relevanciu, poprípade pravdivosť, preto by bolo vhodné aby systém okrem tvorby otázok umožňoval aj ich úpravu a mazanie.

- **Priorita:** Vysoká, jedná sa o základnú funkcionalitu.
- **Náročnosť:** Vysoká, jedná sa o jednu z hlavných implementovaných funkcionalít. Je potrebné vytvoriť UI, ktoré umožní vyplniť všetky potrebné informácie o danej otázke. Okrem toho je potrebné vytvoriť v databáze tabuľku do ktorej sa všetky otázky budú môcť ukladať a vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami vytvorí, alebo upraví záznam o otázke v databáze, alebo ho z nej odstráni.

1.3.2 F2 Správa kvízov

Predtým ako na cvičení, alebo pri akejkoľvek inej príležitosti mohol učiteľ kvíz spustiť tak ho samozrejme musí pripraviť. Aplikácia umožní vytvoriť kvíz z otázok, ktoré si učiteľ sám vytvorí, alebo z otázok, ktoré už boli vytvorené v minulosti, ale pre potreby daného kvízu – podobné téma, neznalosť študentov pri poslednom kvíze, v ktorom táto otázka bola, atď. sú znovu použiteľné. V prípade, že učiteľ pri tvorbe kvízu urobí chybu je potrebné aby mal možnosť tento kvíz upraviť.

- **Priorita:** Vysoká, jedná sa o základnú funkcionalitu.
- **Náročnosť:** Vysoká, jedná sa o jednu z hlavných implementovaných funkcionalít. Je potrebné vytvoriť UI, ktoré umožní do kvízu pridávať otázky, vytvárať nové otázky, ktoré do tohto kvízu budú pridané automaticky a samozrejme umožní vyplniť akékoľvek dodatočné informácie o kvíze. Okrem toho je potrebné vytvoriť v databáze tabuľku do ktorej sa všetky kvízy budú môcť ukladať a vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami vytvorí, alebo upraví záznam o kvíze v databáze, alebo ho z nej odstráni.

1.3.3 F3 Priebeh kvízu

Pri vhodnej príležitosti – cvičenie, prednáška, atď., je potrebné aby mal učiteľ možnosť daný kvíz pre študentov sprístupniť. Študenti by sa následne mali možnosť do kvízu pripojiť. Po tom, ako sa pripoja všetci študenti môže učiteľ spustiť študentom prvú otázku z ním pripraveného kvízu. Študenti potrebujú mať možnosť jednoducho a pokiaľ možno intuitívne na otázky odpovedať. Tento proces sa opakuje toľko krát, koľko otázok bolo do tohto kvízu pridaných.

- **Priorita:** Vysoká, jedná sa o základnú funkcionalitu.
- **Náročnosť:** Vysoká, jedná sa o jednu z hlavných implementovaných funkcionalít. Je potrebné vytvoriť UI, ktoré intuitívne umožní kvízy spúšťať, prihlasovať sa do nich, prechádzať na nové otázky a odpovedať na ne. Ďalej je potrebné vytvoriť novú tabuľku v databáze, v ktorej budú ukladane študentské odpovede a zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami vytvorí, záznam v databáze o študentovej odpovedi, vyhodnotí túto odpoveď a pripíše študentovi body v prípade, že odpovedal správne.

1.3.4 F4 Náhľad študentských riešení

Aby učiteľ mohol poskytnúť spätnú väzbu študentom, ktorí sa kvízu zúčastnili, je potrebné aby mal prístup k ich odpovediam z tohto kvízu.

- **Priorita:** Stredná. Túto funkcionalitu by bolo vhodné v systéme mať, ale je možné fungovať aj bez nej.
- **Náročnosť:** Stredná. Bude potrebné vytvoriť UI, kde bude učiteľ jednoducho môcť prechádzať medzi všetkými študentmi, ktorí sa kvízu zúčastnili a prezerať ich odpovede na otázky z tohto kvízu. Okrem toho bude potrebné vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok, poskytne potrebné dáta o študentových odpovediach v danom kvíze, resp. na konkrétnu otázku.

1.3.5 F5 Tvorba testového modulu

Študenti často k príprave na testy využívajú aplikácie ktoré som popísal vyššie. Preto by bolo dobré, aby bol do systému implementovaný testový modul, ktorý budú študenti môcť využívať na samo štúdium. V prípade, že si študent chce zopakovať nejakú tematiku nechceme, aby mu

pri generovaní testu vyberalo náhodne otázky zo všetkých otázok, ktoré v systéme existujú, ale ideálne z otázok, ktoré sa týkajú daného tématického okruhu. Toto môže byť docielené tým, že učiteľ pripraví otázky, z ktorých sa študentom tieto testy budú generovať, alebo ako neskôr budem popisovať pomocou kategorizácie - teda by sa pri tvorbe modulu nastavila konkrétna kategória otázok z ktorej by sa testy študentom generovali.

- **Priorita:** Vysoká, jedná sa o základnú funkcionalitu.
- **Náročnosť:** Stredná. Bude potrebné vytvoriť UI, kde bude možné do testového modulu pridať otázky. Podobne ako pri tvorbe kvízového modulu, by malo byť možné pridať otázku, ktorá už existuje, alebo vytvoriť novú otázku, ktorá sa do kvízu pridá automaticky. Ďalej bude potrebné vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami vytvorí, záznam v databáze o novom teste.

1.3.6 F6 Spúšťanie samotestov (testového modulu)

Študenti si v prípade potreby môžu svojvoľne zopakovať látku, z nejakého tématického okruhu, ak im to učiteľ povolí formou testu, ktorí si budú môcť kedykoľvek spustiť. Aplikácia zároveň umožní študentom zadať počet otázok, ktoré následne budú v teste. Po spustení bude vytvorený test s otázkami na ktoré môžu študenti odpovedať podobne, ako v kvíze.

- **Priorita:** Vysoká, jedná sa o základnú funkcionalitu.
- **Náročnosť:** Stredná. Bude potrebné vytvoriť UI, ktoré umožní študentom jednoducho vygenerovať a spustiť inštanciu tohto modulu. Ďalej bude potrebné vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami vytvorí, záznam v databáze o novej inštancií daného testového modulu. V každej inštancií testového modulu, sa môžu nachádzať iné otázky zo skupiny otázok, ktorú učiteľ špecifikoval pri tvorbe tohto modulu.

1.3.7 F7 Spätná úprava hodnotenia otázky

Niekedy sa môže stať, že napriek niekoľko násobnej kontrole sa do kvízu dostane otázka obsahujúca chybu. V takomto prípade by bolo vhodné mať možnosť spätne upraviť otázku a podľa toho nanovo prideliť body študentom.

- **Priorita:** Nízka. Táto funkcionalita bude využívaná len príležitostne v špeciálnych prípadoch.
- **Náročnosť:** Stredná. Bude potrebné vytvoriť UI, ktoré umožní vybrať z kvízu konkrétnu otázku, v ktorej je chyba. Túto otázku je následne potrebné mať možnosť upraviť - konkrétne jej hodnotenie, teda tieto úpravy sa budú pravdepodobne týkať len úpravy správnych odpovedí a v niektorých prípadoch aj možností. Ďalej bude potrebné vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami upraví danú otázku a spolu s ňou upraví hodnotenie študentom, podľa toho ako na túto otázku odpovedali a následne poskytne informáciu o tom, ktorým študentom boli body pripočítané a ktorým boli body odpočítané na základe tejto zmeny v otázke.

1.3.8 F8 Kategorizácia otázok

Postupom času, bude v systéme vznikať viac a viac otázok. Kedže systém umožňuje pridávať pri tvorbe kvízu do neho už existujúce otázky, môže v otázkach vniknúť značný chaos, ktorý bude len narastať s pribúdajúcim počtom otázok. Čiastočne by sa tento problém dal vyriešiť možnosťou vyhľadávať v existujúcich otázkach, ale postupom času by aj toto riešenie bolo veľmi

pravdepodobne nepoužiteľné. Z tohto dôvodu je potrebné implementovať nejakú formu kategorizácie pre otázky, ktorá umožní oveľa jednoduchšie vyhľadávanie a orientáciu v už existujúcich otázkach

- **Priorita:** Stredná. Túto funkcionalitu by bolo vhodné v systéme mať, ale je možné fungovať aj bez nej.
- **Náročnosť:** Vysoká. Je potrebné vytvoriť UI na tvorbu samotných kategórií. Následne je potrebné prispôsobiť návrh a implementáciu UI v niektorých častiach webového portálu. Toto zahŕňa voľbu kategórií pri tvorbe/úprave otázky a filtrovanie otázok na základe kategórií. Ďalej je potrebné vytvoriť zdroj API, ktorý po tom, ako naň príde požiadavok s potrebnými dátami vytvorí novú kategóriu, resp. priradí nejakú konkrétnu kategóriu k otázke v databáze. Aby bolo možné evidovať existujúce a vytvárať nové kategórie by bolo vhodné vytvoriť v databáze novú tabuľku, ktorá by na tento účel slúžila.

1.4 Nefunkčné požiadavky

1.4.1 N1 Synchronizácia kvízu medzi všetkými účastníkmi

Nakoľko kvízy sa budú riešiť primárne skupinovo na cvičeniach a ich priebeh riadi vyučujúci, je potrebné bol kvíz synchronizovaný medzi všetkými účastníkmi - teda, že sa všetkým študentom naraz budú zobrazovať nové otázky budú na ne mať rovnaké množstvo času a že nasledujúca otázka sa študentom zobrazí v momente, kedy to učiteľ uzná za vhodné a nebude možné pre študentov ísť vlastným tempom.

1.4.2 N2 REST API pre komunikáciu s webovou aplikáciou

Ako bolo už v sekcií Funkčné požiadavky spomenuté, je potrebné aby sa pri tvorbe kvízu do neho mohli pridávať už pripravené existujúce otázky. Z tohto dôvodu je potrebné pre našu webovú aplikáciu vytvoriť API, s ktorým bude môcť komunikovať. Toto API bude postavené nad nejakou formou databáze, v ktorej sa budú uchovávať potrebné informácie, nielen o otázkach, ale aj kvízoch samotných, odpovediach študentov atď.

1.4.3 N3 Jednoduchosť používania webovej aplikácie

Aplikácia, resp. konkrétne kvízový a testový modul by mali byť jednoduché a intuitívne na používanie. Napríklad pri pohľade na otázku by mali obsahovať zadanie otázky, možnosti a časový odpočet, čokoľvek navyše by pravdepodobne spôsobilo zmätok pri používaní týchto modulov.

1.4.4 N4 UI vzhľadom na farboslepých užívateľov

Užívateľské rozhranie by malo brať ohľad na farboslepých užívateľov, resp. obecne na užívateľov s akýmkoľvek zdravotným znevýhodnením, avšak čo sa týka UI webovej aplikácie, najväčšou skupinou sú práve farboslepí užívatelia (daltonisti). Existuje niekoľko rôznych foriem farbosleposti, podľa straty vnímania konkrétnej farby - červenej, zelenej alebo modrej.

Aplikácia by teda nemala používať na indikáciu javu čisto niektoré z farieb - konkrétne napríklad na indikáciu správnych odpovedí by sa nemala používať len červená resp. zelená farba. Namiesto farieb, alebo v kombinácií s nimi by sa mal použiť iný indikátor, ktorý nie je závislý na farbe.

1.4.5 N5 Zabezpečenie API

Aby nemohol ktokoľvek s prístupom na internet posielať požiadavky na API a jednoducho z neho dostávať dáta, je potrebné API nejakým spôsobom zabezpečiť. Toto by malo prebiehať na 2 úrovniach.

1. Zamedzenie prístupu k API nepovolaným osobám - osobám, ktoré nie sú prihlásené a autentifikované vo webovej aplikácií
2. a v prípade, že je osoba autentifikovaná tak overiť či je prístup k danému endpointu[1] a vykonaniu danej operácie tejto osobe povolený, na základe autority resp. role tohto užívate.

1.4.6 N6 Obmedzenie prístupu k záznamom v databáze

Ako som už naznačil, je potrebné aj autentifikovaným užívateľom zamedziť prístup k istým častiam aplikácie resp. operáciám v našej aplikácií. To isté sa týka konkrétnych záznamov v databáze, nech už ide o akúkoľvek CRUD(Create, Read, Update, Delete) operáciu.

1.4.7 N7 „Generovanie“ testov

Testový modul, je primárne určený k precvičovaniu danej látky pri akejkoľvek príležitosti a preto ich rozsah v počte otázok môže byť oveľa väčší ako ten v kvízovom module, ktorého zámerom sú skupinové kvízy na cvičeniach. Rovnako nie je potrebné aby mali všetci študenti v teste rovnaké otázky, keďže tento modul je zameraný na individuálne použitie. Z týchto dôvodov by bolo vhodné vytvoriť mechanizmus, ktorý zabezpečí náhodný výber otázok, ktoré budú použité v konkrétnom teste, aby testy neboli zakaždým identické. Otázky budú náhodne vyberané z preddefinovanej skupiny otázok, ktorú vytvorí učiteľ pri vytváraní tohto modulu, ako som už spomínal vo funkčných požiadavkoch.

1.4.8 N8 Podpora markdownu v texte otázok a možností

Markdown je jazyk, ktorý slúži pre úpravu prostého textu. V textoch otázok a odpovedí je často potrebné mať viac ako len čistý text - napríklad môže byť potrebné ho doplniť o matematickú formulu, úryvok zdrojového kódu, alebo len zvýrazniť niektoré časti zadania, na čo je práve vhodné, využiť túto technológiu.

1.5 Prípady užitia

Z analýzy funkčných požiadavkov vyšli najavo nasledujúce prípady užitia kvízového a testového modulu. Prípady užitia sú číslované podľa toho, ku ktorému funkčnému požiadavku sa vzťahujú, teda napr. akýkoľvek prípad užitia začínajúci UC1 (skratka pre Use Case[2] 1) sa vzťahuje k funkčnému požiadavku F1.

1.5.1 UC1.1 Vytváranie otázok

1. Učiteľ sa nachádza na obrazovke kvízového editoru.
2. Učiteľ klikne na tlačidlo „Vytvoriť otázku“.

[1] zdroju
[2] Prípad užitia

3. Zobrazí sa dialógové okno s formulárom na vytvorenie novej otázky.
4. Učiteľ zvolí typ otázky
5. Učiteľ vyplní zadanie otázky, v prípade, že je to potrebné vyplní texty možností, označí, ktoré z možností sú správne a zadá časový limit na danú otázku. Dobrovoľne môže ešte k otázke pridať vysvetlenie.
6. Učiteľ vytvorí otázku kliknutím na tlačidlo „Uložiť“.

1.5.2 UC1.1.1 Výber typu otázky

1. Učiteľ sa nachádza na obrazovke editoru otázky.
2. Učiteľ rozklikne menu „Typ otázky“.
3. Učiteľ vyberie, ktorý typ otázky chce kliknutím na tlačítko reprezentujúce daný typ.
4. Po výbere typu otázky sa prispôsobí editor otázky vybranému typu.

1.5.3 UC1.1.2 Pridávanie ľubovoľného počtu možností

1. Učiteľ sa nachádza na obrazovke editoru otázky.
2. Učiteľ klikne na tlačidlo „Pridať možnosť“.
3. Do editoru je pridaná nová, zatiaľ prázdna možnosť, pripravená na vyplnenie.

1.5.4 UC1.1.3 Odoberanie nepotrebných možností

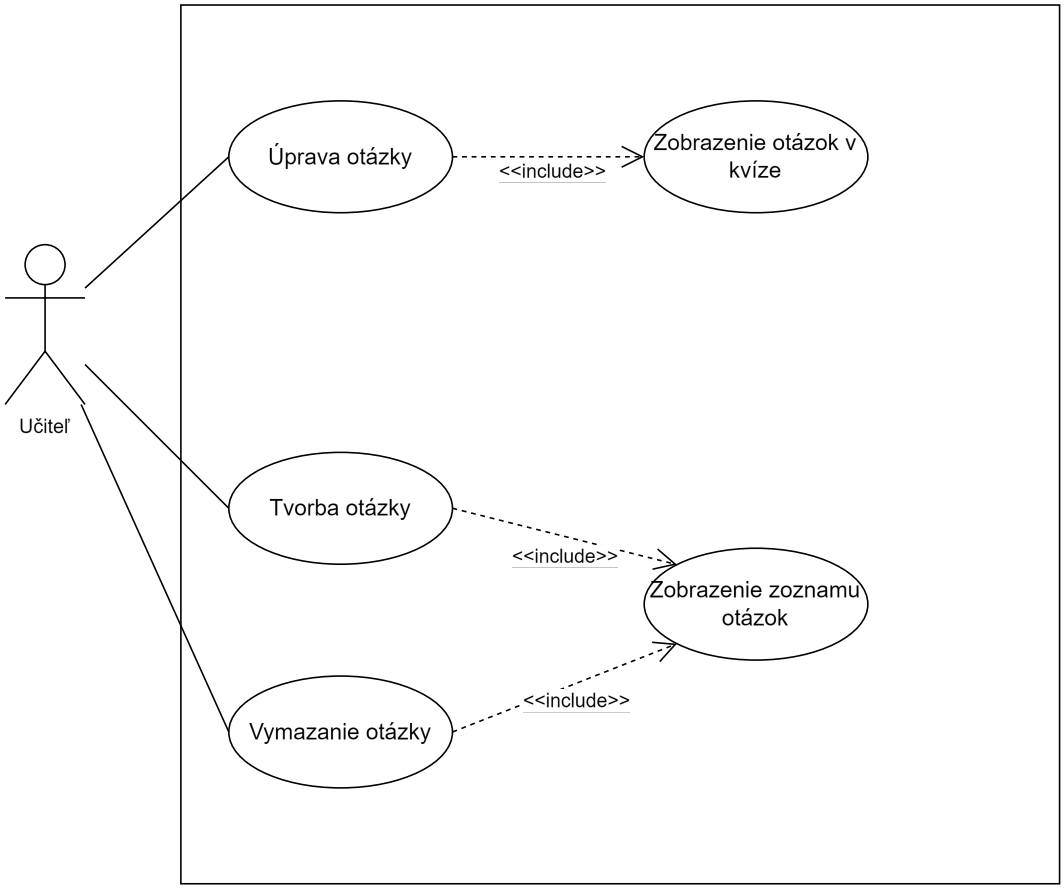
1. Učiteľ sa nachádza na obrazovke editoru otázky.
2. Pri každej vymazateľnej možnosti sa nachádza tlačidlo „Vymazať“.
3. Učiteľ klikne na tlačidlo „Vymazať“ pri niektorej z možností.
4. Daná možnosť je z editoru otázky vymazaná.

1.5.5 UC1.2 Úprava otázok

1. Učiteľ sa nachádza na obrazovke kvízového editoru.
2. Ak sa v kvíze sa už nachádza otázka je možné pokračovať
3. Pri otázke klikne učiteľ na tlačidlo „Upraviť“.
4. Zobrazí sa dialógové okno s formulárom na úpravu otázky s vyplnenými dátami otázky, ktorú chce učiteľ upraviť.
5. Učiteľ upraví otázku podľa svojich potrieb.
6. Učiteľ upraví otázku kliknutím na tlačidlo „Uložiť“ za predpokladu, že sú vyplnené všetky povinne časti otázky – text otázky, možnosti, správne odpovede a časový limit.

1.5.6 UC1.3 Odstraňovanie nepotrebných otázok

1. Učiteľ sa nachádza na obrazovke kvízového editoru.
2. Učiteľ klikne na tlačidlo „Zoznam existujúcich otázok“.
3. Zobrazí sa okno so všetkými otázkami.
4. Učiteľ vyberie otázku a klikne na tlačítko „Odstrániť“.



Obr. 1.14 Diagram prípadov užitia - správa otázok

1.5.7 UC2.1 Vytváranie kvízu

1. Učiteľovi sa zobrazia pripravené lekcie.
2. Učiteľ klikne na tlačidlo „upraviť“ pri niektorej z lekcií.
3. Učiteľovi sa zobrazí obrazovka kde môže upravovať túto lekciu a moduly, ktoré obsahuje.
4. Učiteľ klikne na tlačidlo vytvoriť nový modul.
5. Učiteľ zvolí „Kvízový modul“ ako typ modulu.
6. Učiteľ sa presunie na obrazovku kvízového editoru kliknutím na príslušný tab.

7. a. Ak chce učiteľ pridať novú otázku ďalej postupuje podľa UC1.1
   b. Ak chce učiteľ pridať existujúcu otázku ďalej postupuje podľa UC2.1.1
8. Učiteľ hotový kvíz vytvorí kliknutím na tlačidlo „uložiť“.

1.5.8 UC2.1.1 Pridávanie existujúcich otázok do kvízu

1. Učiteľ sa nachádza na obrazovke editoru kvízu.
2. Učiteľ klikne na tlačidlo „Existujúce otázky“.
3. Zobrazí sa okno so všetkými existujúcimi otázkami a možnosťou filtrovania otázok.
4. Učiteľ zvolí, ktoré otázky chce do kvízu pridať.
5. Učiteľ zatvorí okno a je presmerovaný na obrazovku editoru kvízu, kde sú už zobrazené novopridané otázky.

1.5.9 UC2.1.2 Menenie poradia otázok v kvíze

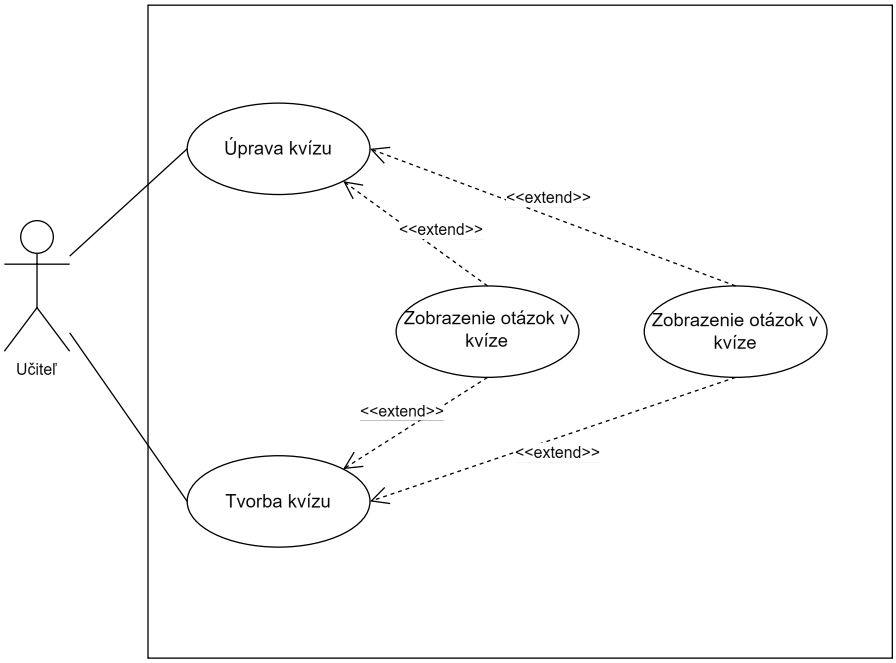
1. Učiteľ sa nachádza na obrazovke editoru kvízu.
2. V prípade, že sú v kvíze aspoň 2 otázky je možné pokračovať.
3. Učiteľ klikne na niektorú z otázok, ktoré sú v kvíze a potiahne ju na miesto medzi otázkami kde ju chce mať.
4. Poradie otázok sa prispôsobí - vybraná otázka je presunutá na žiadané miesto a poradie ostatných otázok je automaticky upravené.

1.5.10 UC2.1.3 Odstránenie otázky z kvízu

1. Učiteľ sa nachádza na obrazovke editoru kvízu.
2. Ak sa v kvíze sa už nachádza otázka je možné pokračovať.
3. Pri otázke klikne učiteľ na tlačidlo „Odstrániť“.
4. Táto otázka je z kvízu odstránená a poradie otázok aktualizované.

1.5.11 UC2.2 Úprava kvízu

1. Učiteľovi sa zobrazia pripravené lekcie.
2. Učiteľ klikne na tlačidlo „upraviť“ pri niektorej z lekcií.
3. Učiteľovi sa zobrazí obrazovka kde môže upravovať túto lekciu a moduly, ktoré obsahuje.
4. Ak je učiteľ autorom modulu, ktorý chce upraviť ukazuje sa mu tlačidlo na úpravu.
5. Učiteľ klikne na tlačidlo pre úpravu modulu.
6. Učiteľovi sa zobrazí detail kvízového modulu so všetkými otázkami ktoré obsahuje.
7. a. Ak chce učiteľ pridať novú otázku ďalej postupuje podľa UC1.1
   b. Ak chce učiteľ pridať existujúcu otázku ďalej postupuje podľa UC2.1.1
   c. Ak chce učiteľ upraviť niektorú z otázok postupuje ďalej podľa UC1.2
8. Učiteľ upravený kvíz upraví kliknutím na tlačidlo „uložiť“.



Obr. 1.15 Diagram prípadov užitia - správa kvízov

1.5.12 UC3.1 Spúšťanie kvízu

1. Učiteľ otvorí detail lekcie.
2. Učiteľ zvolí v zozname modulov pripravený kvízový modul.
3. Učiteľ klikne na tlačidlo „vytvoriť miestnosť“.
4. Zobrazí sa nová obrazovka „čakárne“, kde sa budú postupne zobrazovať mená študentov, ktorý sa do kvízu prihlásili. Na tejto obrazovke je zobrazený kód, ktorým sa študenti do tohto kvízu budú prihlasovať.
5. Po tom, ako sa všetci študenti prihlásia do kvízu učiteľ klikne na tlačidlo spustiť kvíz, čím sa všetkým účastníkom zobrazí naraz prvá otázka.

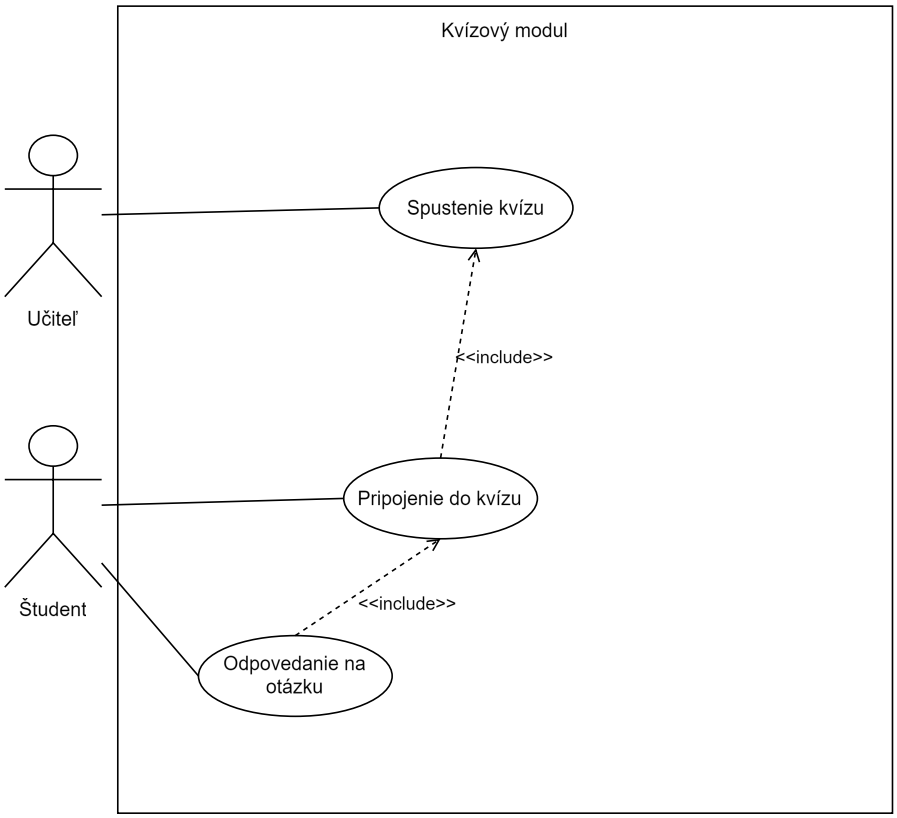
1.5.13 UC3.2 Prihlásenie sa do kvízu

1. Študent otvorí detail lekcie.
2. Študent nájde kvízový modul v zozname modulov a klikne naň.
3. Študent zadá prihlasovací kód, ktorý zverejnil učiteľ a prihlási sa do kvízu.
4. Po prihlásení do kvízu sa zobrazí obrazovka „čakárne“, kde študent vidí svoje meno a čaká až učiteľ kvíz spustí.

1.5.14 UC3.3 Odpovedanie na otázky

1. Prihláseným študentom sa naraz zobrazí otázka s časovým limitom
2. Študent vyberie možnosť/možnosti a klikne na tlačítko odoslať.

3. Ďalej nie je možné odpovedať na otázku
4. Ak študent nestihne odoslať odpoveď, jeho odpoveď bude odoslaná (ako všetky aktuálne vybrané možnosti) automaticky keď časový limit vyprší.



Obr. 1.16 Diagram prípadov užitia - kvízový modul

1.5.15 UC4 Náhľad študentských riešení

1. Učiteľ otvorí detail lekcie.
2. Učiteľ zvolí v zozname modulov pripravený kvízový modul.
3. Učiteľ klikne na tlačidlo „Náhľad výsledkov“.
4. Zobrazí prehľad študentských riešení kvízového modulu.
5. Kliknutím na konkrétneho študenta sa zobrazia všetky pokusy daného kvízového modulu vybraného študenta.
6. V prípade, že študent absolvoval ten istý kvízový modul viac krát, je možné medzi pokusmi prechádzať tlačidlami.
7. V náhľade každého pokusu učiteľ môže vidieť odpovede daného študenta v konkrétnom pokuse na všetky otázky, spolu s prehľadom, ktoré otázky boli zodpovedané správne, a ktoré nesprávne.

1.5.16 UC5.1 Tvorba testového modulu

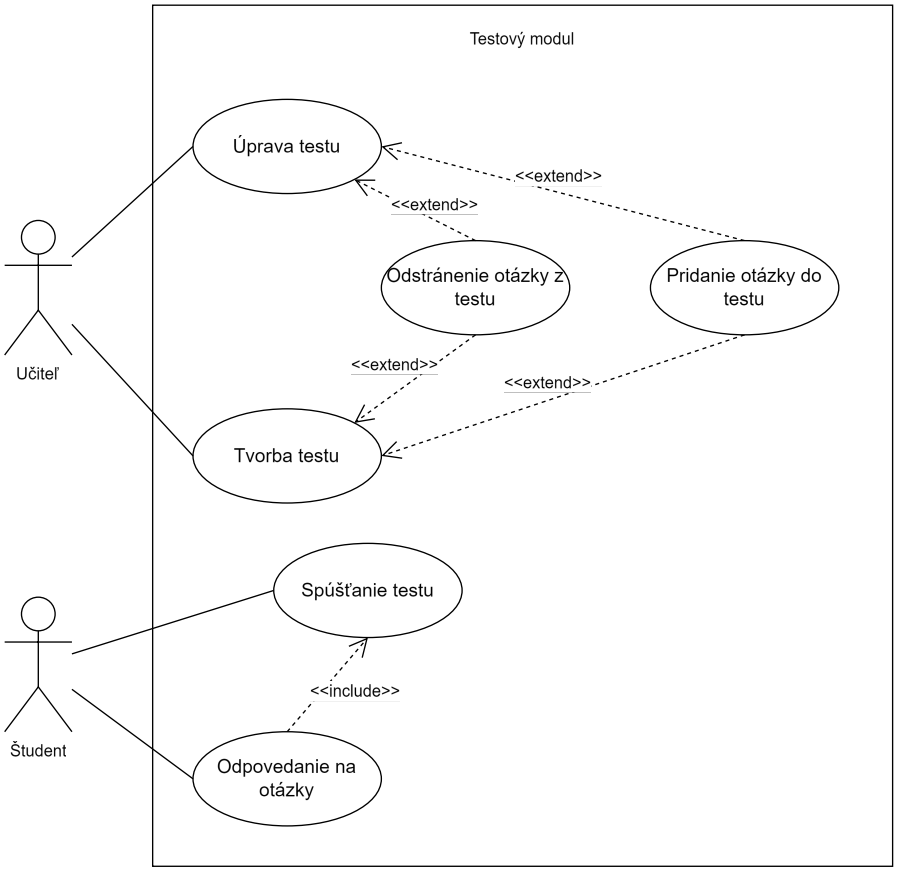
1. Učiteľovi sa zobrazia pripravené lekcie.
2. Učiteľ klikne na tlačidlo „upraviť“ pri niektorej z lekcií.
3. Učiteľovi sa zobrazí obrazovka kde môže upravovať túto lekciu a moduly, ktoré obsahuje.
4. Učiteľ klikne na tlačidlo vytvoriť nový modul.
5. Učiteľ zvolí „Testový modul“ ako typ modulu.
6. Učiteľ sa presunie na obrazovku testového editoru kliknutím na príslušný tab.
7. a. Ak chce učiteľ pridať novú otázku ďalej postupuje podľa UC1.1
   b. Ak chce učiteľ pridať existujúcu otázku ďalej postupuje podľa UC2.1.1
   c. Ak chce učiteľ upraviť niektorú z otázok postupuje ďalej podľa UC1.2
8. Učiteľ hotový testový modul vytvorí kliknutím na tlačidlo „uložiť“.

1.5.17 UC5.2 Úprava testového modulu

1. Učiteľovi sa zobrazia pripravené lekcie.
2. Učiteľ klikne na tlačidlo „upraviť“ pri niektorej z lekcií.
3. Učiteľovi sa zobrazí obrazovka kde môže upravovať túto lekciu a moduly, ktoré obsahuje.
4. Ak je učiteľ autorom modulu, ktorý chce upraviť ukazuje sa mu tlačidlo na úpravu.
5. Učiteľ klikne na tlačidlo pre úpravu modulu.
6. Učiteľovi sa zobrazí detail testového modulu so všetkými otázkami ktoré obsahuje.
7. a. Ak chce učiteľ pridať novú otázku ďalej postupuje podľa UC1.1
   b. Ak chce učiteľ pridať existujúcu otázku ďalej postupuje podľa UC2.1.1
   c. Ak chce učiteľ upraviť niektorú z otázok postupuje ďalej podľa UC1.2
8. Učiteľ upravený testový modul upraví kliknutím na tlačidlo „uložiť“.

1.5.18 UC6 Spúšťanie testov (testového modulu)

1. Študent otvorí detail lekcie.
2. Študent nájde testový modul v zozname modulov a klikne naň.
3. Študent si zvolí, koľko chce aby mal v teste otázok a klikne na tlačidlo „Spustiť test“.
4. Následne je študentovi vygenerovaný test z preddefinovanej skupiny otázok, ktoré vybral učiteľ keď testový modul vytváral.



Obr. 1.17 Diagram prípadov užitia - testový modul

1.5.19 UC7 Spätná úprava hodnotenia otázky

1. Učiteľ otvorí detail lekcie.
2. Učiteľ zvolí v zozname modulov pripravený kvízový modul.
3. Učiteľ klikne na tlačidlo „Náhľad výsledkov“.
4. Zobrazí prehľad študentských riešení kvízového modulu.
5. Kliknutím na konkrétneho študenta sa zobrazia všetky pokusy daného kvízového modulu vybraného študenta.
6. V prípade, že študent absolvoval ten istý kvízový modul viac krát, je možné medzi pokusmi prechádzať tlačidlami.
7. Učiteľ klikne na tlačidlo „Upraviť hodnotenie“.
8. Zobrazí sa okno so zoznamom otázok, ktoré sú súčasťou tohto kvízu.
9. Učiteľ klikne na tlačidlo „Upraviť otázku“. -> pokračovanie editácia otázky
10. V prípade, že učiteľ túto otázku už upravil môže kliknúť na tlačidlo „Prepočítať body“, čím sa upraví hodnotenie všetkým študentom z danej otázky v konkrétnom pokuse kvízu.
11. Zobrazí sa zoznam študentov, ktorým bolo zmenené hodnotenie.

1.5.20 UC8.1 Tvorba kategórie

1. Učiteľ sa nachádza na obrazovke tvorby otázky.
2. Na obrazovke je ponuka výberu kategórie, ktoré je možné priradiť k otázke a tlačidlo „Nová kategória“.
3. Učiteľ klikne na tlačidlo „Nová kategória“.
4. Zobrazí sa formulár na tvorbu novej kategórie.
5. Učiteľ vyplní všetky povinné polia tohto formulára a klikne na tlačidlo „Uložiť“.
6. Nová kategória je vytvorená a učiteľ je presmerovaný na obrazovku tvorby otázky.

1.5.21 UC8.2 Priradenie kategórie k otázke

1. Učiteľ sa nachádza na obrazovke tvorby otázky.
2. Na obrazovke je ponuka výberu kategórie, ktoré je možné priradiť k otázke a tlačidlo „Nová kategória“.
3. Učiteľ otvorí ponuku kategórií.
4. Zobrazia sa existujúce kategórie.
5. Učiteľ vyberie jednu, alebo viac kategórií.
6. Učiteľ klikne na tlačidlo „Uložiť“.
7. Učiteľ je presmerovaný na obrazovku tvorby otázky, kde pri jeho otázke sú priradené vybrané kategórie.

1.5.22 UC8.3 Vyhľadávanie otázok podľa kategórií

1. Učiteľ sa nachádza na obrazovke editoru kvízu.
2. Učiteľ klikne na tlačidlo „Existujúce otázky“.
3. Zobrazí sa okno so všetkými existujúcimi otázkami a možnosťou filtrovania otázok.
4. Učiteľ zvolí „Filtrovanie podľa kategórie“ a zvolí požadovanú kategóriu.
5. V okne sa už zobrazujú len otázky, ktoré patria do danej kategórie.

1.6 Zhrnutie

V tejto kapitole som zanalyzoval niektoré už existujúce riešenia, ktoré sa zaoberajú problematikou kvízov, testov, alebo obecne zodpovedaním otázok za účelom štúdia resp. samoštúdia. Následne som zanalyzoval požiadavky na nami implementované riešenie a posúdil ich náročnosť a prioritu. Postupnou analýzou som prišiel k záveru, že rovnako, ako sa všetky analyzované riešenia v niečom líšili, rovnako naše riešenie bude odlišné od tých už existujúcich.

	F1	F2	F3	F4	F5	F6	F7	F8
Kahoot	✓	✓	✓	✓				
Slido	✓	✓	✓					
Quizziz	✓	✓	✓	✓				✓
Quizlet	✓				✓	✓		✓
Anki	✓							✓
Moodle	✓	✓	✓					
Marast	✓	✓	✓	✓				✓
Duolingo	✓					✓		✓

Tabuľka 1.1 Porovnanie funkčných požiadavkov s existujúcimi modelmi

	F1	F2	F3	F4	F5	F6	F7	F8
UC1.1	✓							
UC1.1.1	✓							
UC1.1.2	✓							
UC1.1.3	✓							
UC1.2	✓							
UC1.3	✓							
UC2.1		✓						
UC2.1.1		✓						
UC2.1.2		✓						
UC2.1.3		✓						
UC2.2		✓						
UC3.1			✓					
UC3.2			✓					
UC3.3			✓					
UC4				✓				
UC5.1					✓			
UC5.2					✓			
UC6						✓		
UC7							✓	
UC8.1								✓
UC8.2								✓
UC8.3								✓

Tabuľka 1.2 Tabuľka pokrytia prípadov užitia

Kapitola 2
Návrh

V tejto kapitole sa budem venovať návrhu aplikácie. V prechádzajúcej kapitole som identifikoval prípady užitia týchto modulov, na základe ktorých následne navrhnem užívateľské rozhranie pre učiteľov aj pre študentov. Ďalej sa budem zaoberať návrhom databázy a serverovej časti aplikácie, kde sa budem venovať architektúre tejto časti aplikácie a návrhu API - konkrétne návrhu zdrojov (endpointov) tohto API a výberu vhodných technológií.

2.1 Návrh užívateľského rozhrania

Na základe analýzy požiadavkov a prípadov užitia, ktoré z tejto analýzy tiež vyšli som navrhol užívateľské rozhranie pre kvízový a testový modul, čo zahŕňa rozhranie pre tvorbu týchto modulov, náhľad výsledkov študentských riešení a rozhranie cez ktoré sa tieto moduly budú ovládať. Niektoré z týchto rozhraní majú veľmi podobnú funkcionalitu a teda ich návrh bude taktiež veľmi podobný. Na druhú stranu niektoré rozhrania sú diametrálne odlišné v ich funkcionalite a využití preto aj ich návrh bude značne odlišný.

2.1.1 Tvorba otázok

Tvorba otázok prebieha v dialógovom okne, ktoré je rozdelené na niekoľko častí. V prvej časti si učiteľ môže vybrať typ otázky. Následne sa podľa výberu typu otázky prispôsobí druhá časť okna, ktorá obsahuje textové editory pre text zadania a texty možností - v prípade, že je nutné aby učiteľ tieto texty zadával, pri niektorých typoch môžu byť možnosti definované vopred. V tretej časti, učiteľ môže nastaviť časový limit na túto otázku a pridať k nej vysvetlenie. Úplne na konci tohto okna sa potom nachádza tlačidlo „Uložiť“, ktorým sa rozpracovaná otázka uloží, ak sú všetky povinné polia vyplnené.

2.1.2 Tvorba modulov

Obrazovka na tvorbu kvízového modulu je takmer identická s obrazovkou na tvorbu testového modulu. Hlavným komponentom je prehľad otázok, ktoré boli už do kvízu pridané. Tento prehľad umožní jednoduché menenie poradia otázok. Pod týmto prehľadom by sa mali nachádzať 3 tlačidlá:

- Tlačidlo „Existujúce otázky“ slúži na zobrazenie prehľadu všetkých otázok. Z tohto prehľadu bude môcť učiteľ pridávať už existujúce otázky do kvízu a tiež, v prípade, že v prehľade vidí nejakú otázku, ktorú nemá v pláne už používať, môže danú otázku zo systému vymazať.

- Tlačidlo „Vytvoriť otázku“ slúži na otvorenie dialógového okna, v ktorom môže učiteľ vytvoriť novú otázku. Po vytvorení otázky, bude táto otázka automaticky pridaná do kvízu.
- Tlačidlo „Uložiť“, ktoré slúži na uloženie kvízu v jeho aktuálnom stave.

2.1.3 Náhľad študentských riešení

Obrazovka s náhľadom študentských riešení sa vždy vzťahuje ku konkrétnemu modulu. Pri rôznych typoch modulov sa môže UI tejto obrazovky značne líšiť, avšak v prípade kvízového a testového modulu je toto UI prakticky identické. Základom obrazovky je prehľad študentov, ktorí sa zúčastnili kvízu, alebo si vygenerovali nejaký test. V prípade potreby je možné vybrať konkrétneho študenta a prezrieť si jeho riešenie. Môže sa stať, že študent absolvoval daný kvíz niekoľko krát, alebo, že v testovom module vygeneroval test a absolvoval ho - druhá z možností je viac pravdepodobná. Pre takéto prípady obrazovka obsahuje číslo pokusu a tlačidlá, ktorými je možné medzi pokusmi jednoducho prechádzať. Z náhľadu študentských riešení je tak isto možné tlačidlom „Upraviť hodnotenie“ upraviť niektorú z otázok a na základe tejto úpravy znova vyhodnotiť danú otázku a prerozdeliť študentom body.

2.1.4 Kvízový modul

Kvízový modul sa skladá z niekoľkých rôznych obrazoviek, ktoré sa líšia v závislosti, či je prihlásený učiteľ, alebo študent. Nižšie popisujem jednotlivé obrazovky s ktorými sa užívatelia počas kvízu stretnú.

Úvodná obrazovka: Prvá obrazovka, ktorú užívateľ po zobrazení modulu uvidí. V učiteľskom rozhraní je spolu s názvom tohto kvízu resp. modulu tlačidlo na otvorenie kvízu. V študentskom rozhraní sa, podobne ako v tom učiteľskom, tiež nachádza názov kvízu. Ďalej sa v ňom nachádza textové pole, do ktorého študenti zadajú heslo, ktoré im sprístupní učiteľ a tlačidlo „Vstúpiť“, ktorým sa študenti prihlásia do kvízu, ak nimi zadané heslo je správne.

Čakáreň: Obrazovka „čakárne“ v učiteľskom rozhraní obsahuje zoznam študentov, ktorí sa prihlásili do kvízu, spolu s ich počtom. Pod týmto zoznamom sa nachádza tlačidlo „Spustiť“, ktorým sa spustí kvíz resp. prvá otázka tohto kvízu.

Obrazovka otázky: Na obrazovke otázky sa nachádzajú dáta k danej otázke. Na hornej časti obrazovky je zadanie, pod ním sú v rovnakom poradí pre všetkých účastníkov kvízu zobrazené karty s možnosťami. Po kliknutí na jednu z možností z týchto kariet, je táto karta označená. Pod kartami s možnosťami je tlačidlo „Odoslať“, ktoré odpoveď - označené možnosti, odošle. Na pravej časti obrazovky je odpočet. Po tom, ako tento odpočet dosiahne 0, sú všetkým študentom automaticky odoslané odpovede, podobne ako keby ich odošlú sami pred vypršaním tohto časového limitu. Jediný rozdiel medzi učiteľským a študentským rozhraním je, že učiteľ nemôže označiť žiadnu z možností a nemá tlačidlo „Odoslať“ v spodnej časti obrazovky.

Vyhodnotenie otázky: Základným komponentom tejto obrazovky je, podobne ako na obrazovke samotnej otázky, zadanie otázky a možnosti. Narozdiel od nej však na nej účastníci kvízu môžu vidieť označené možnosti, ktoré boli správne. V študentskom rozhraní sú navyše označené aj odpovede, ktoré daný študent označil, aj keď neboli správne. Učiteľské rozhranie obsahuje navyše oproti študentskému grafy pri každej s možností s číslom označujúcim počet študentov, ktorí túto možnosť označili. V spodnej časti obrazovky učiteľského rozhrania sa tak isto nachádza rozbaľovacie okno s vysvetlením, ak je pri danej otázke nejaké vysvetlenie pripravené.

Výsledky: Medzi učiteľským a študentským rozhraním je v tomto prípade značný rozdiel. Učiteľské rozhranie obsahuje tabuľku s niekoľkými študentmi, ktorí majú z tohto kvízu najlepšie

skóre - teda odpovedali na najviac otázok správne. Po touto tabuľkou je tlačidlo „Ukončiť kvíz“, ktorým sa kvíz ukončí. V študentskom rozhraní na rozdiel od toho učiteľského, študent vidí len svoj finálny počet bodov. Pod počtom bodov sa nachádza zoznam otázok, z daného kvízu, spolu s informáciou či študent túto otázku zodpovedal správne. Každú z otázok v tomto zozname je možné rozbaliť. Po rozbalení študent môže vidieť detail tejto otázky, spolu s jeho odpoveďami a označením, ktoré z možností boli správne a ktoré možnosti študent označil nesprávne.

2.1.5 Testový modul

Testový modul sa, podobne ako ten kvízový skladá z niekoľkých obrazoviek. Na rozdiel od toho kvízového je však UI pre role učiteľa aj študenta identické. Učitelia pravdepodobne tento modul nepotrebujú vôbec, ale vzhľadom na to, že sa môže stať, že si učiteľ bude po vytvorení testového modulu tento modul otestovať, je vhodné toto UI sprístupniť bez ohľadu na rolu užívateľa.

Úvodná obrazovka: Prvá obrazovka, ktorú užívateľ vidí po otvorení testového modulu. Obsahuje meno testového modulu a rôzne parametre pre generovanie testu - napr. koľko otázok užívateľ chce v teste mať, alebo koľko času chce študent na test mať. Ďalej je v tomto rozhraní tlačidlo „Spustiť“, ktorým sa vygeneruje test podľa zvolených parametrov.

Obrazovka otázky: Na obrazovke otázky sa nachádzajú dáta k danej otázke. Na hornej časti obrazovky je zadanie, pod ním sú v rovnakom poradí pre všetkých účastníkov kvízu zobrazené karty s možnosťami. Po kliknutí na jednu z možností z týchto kariet, je táto karta označená. Pod kartami s možnosťami je tlačidlo „Odoslať“, ktoré odpoveď - označené možnosti, odošle. Na pravej časti obrazovky sa nachádza okno s časovým limitom a niekoľkými tlačidlami. Tlačidlo „K prehľadu“ užívateľa presmeruje na obrazovku „Prehľad testu“ a tlačidlo „Ukončiť test“, bez vyhodnotenia ukončí aktuálne prebiehajúci test a presmeruje užívateľa na úvodnú obrazovku. Na úplnom vrchu a úplnom spodku rozhrania je zobrazené číslo aktuálnej otázky a tlačidlá slúžiace na prechod na predchádzajúcu, alebo nasledujúcu otázku.

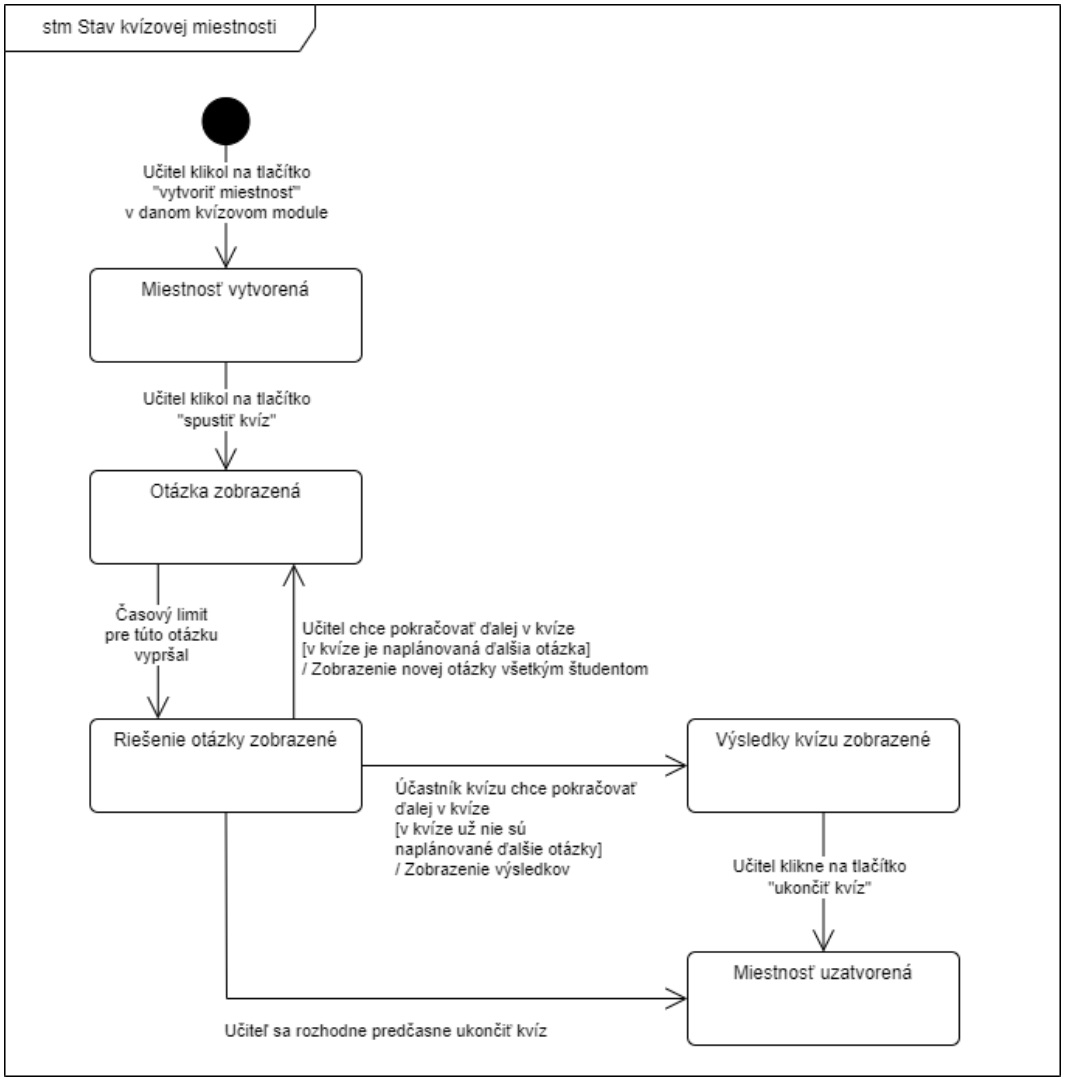
Prehľad testu: Základným komponentom tejto obrazovky je tabuľka obsahujúca prehľad všetkých otázok v aktuálne prebiehajúcom teste spolu s ich statusom - teda či študent odpovedal na danú otázku, alebo nie a tlačidlo „K otázke“, ktoré užívateľa presmeruje na obrazovku danej otázky. Pod touto tabuľkou je tlačidlo „Odoslať“, ktorým sa odošle celý test a zobrazí sa obrazovka s výsledkami. Na pravej časti obrazovky sa nachádza okno s časovým limitom a tlačidlom „Ukončiť test“. Toto tlačidlo, podobne ako na obrazovke otázky, bez vyhodnotenia ukončí aktuálne prebiehajúci test a presmeruje užívateľa na úvodnú obrazovku.

Výsledky: Na tejto obrazovke užívateľ nájde prehľad všetkých otázok, ktoré práve vyhodnotený test obsahoval, spolu s označenými správnymi možnosťami a možnosťami, ktoré zvolil užívateľ aj napriek tomu, že správne nie sú. Na pravej časti obrazovky sa nachádza okno, obsahujúce celkové bodové ohodnotenie študenta a tlačidlo „Ukončiť test“, po kliknutí na ktoré, je užívateľ presmerovaný na úvodnú obrazovku.

2.2 Návrh priebehu kvízu

Priebeh kvízu, od jeho spustenia až po jeho ukončenie bude mať niekoľko rôznych fáz resp. stavov. Väčšinou sa medzi týmito stavmi kvízu prechádza na základe akcie učiteľa, v niektorých prípadoch, ale môžu stav svojho kvízu študenti zmeniť sami, ako je zobrazené na diagrame nižšie.

Jedným z nefunkčných požiadavkov, konkrétne N1, je synchronizácia kvízu medzi všetkými užívateľmi ktorí sú doň prihlásení. To znamená, že potrebujeme nielen, aby bol synchronizovaný stav kvízu medzi všetkými užívateľmi - čo je zabezpečené pomocou atribútu „room_state“ v tabuľke Quizroom, ale aby sa všetkým užívateľom zobrazila tá istá otázka v ten istý moment.



Obr. 2.1 Stavový diagram kvízovej miestnosti

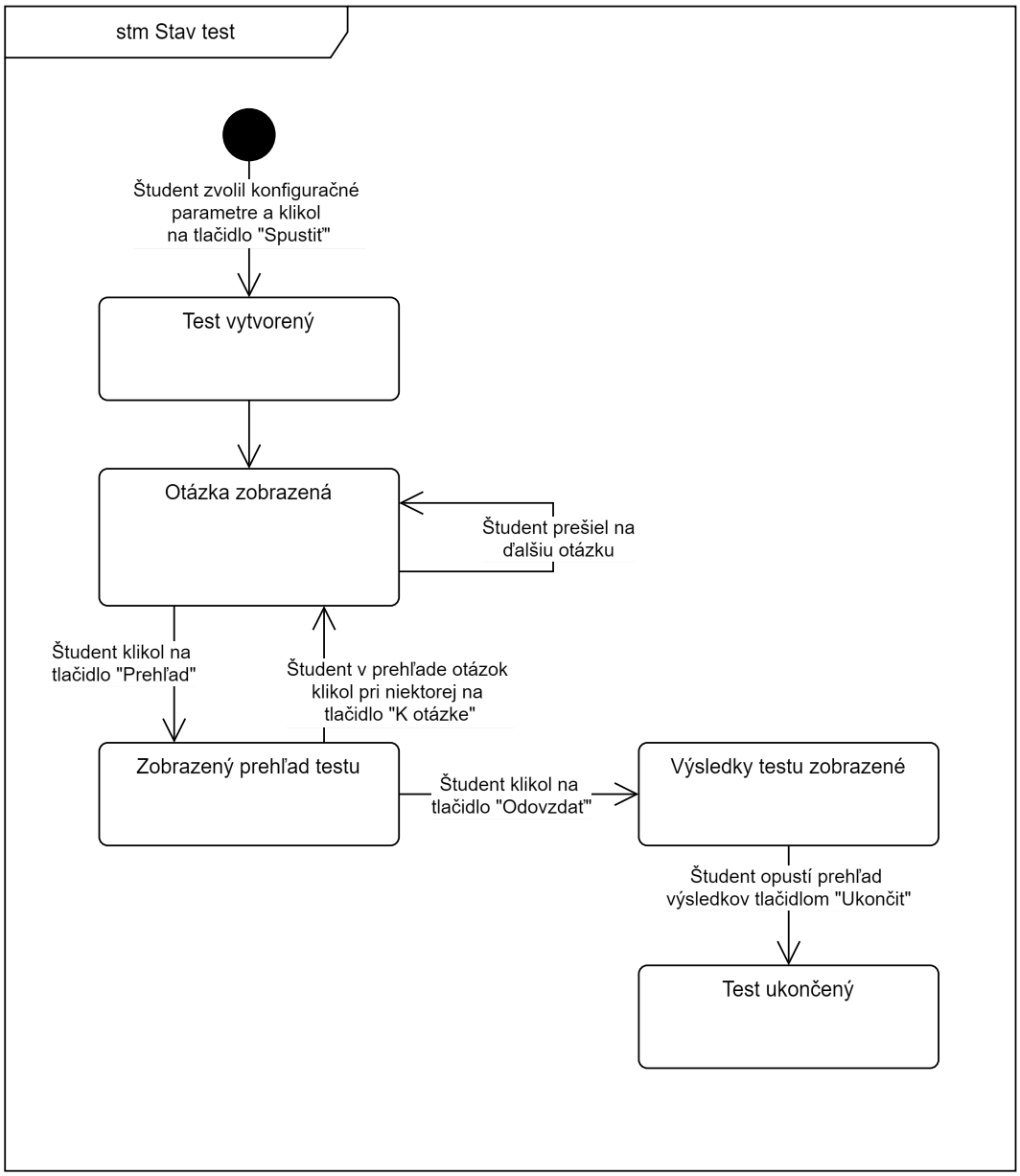
To je možné zabezpečiť pomocou technológie „Websocket“, ktorá využíva protokol TCP. Pri prihlásení do kvízu začne klient automaticky počúvať na špecifickej URI a keď sú na túto URI odoslané dáta, tieto dáta sú prijaté naraz všetkými klientmi počúvajúcimi na danej URI. Hlbšie sa tejto technológií budem venovať v sekcií „Technológie - Internetové protokoly“.

2.3 Návrh priebehu „samotestu“

Priebeh „samotestu“ je značne odlišný od priebehu kvízov, nakoľko kvízy sú určené k tomu, aby boli vypracovávané v skupine (napr. na cvičení). „Samotesty“ sú určené na individuálne používanie študentom, najmä za účelom prípravy na výuku alebo blížiaci sa test. Celý proces spúšťania a prechádzania „samotestu“ je teda plne v réžií individuálneho študenta.

Na úvodnej obrazovke si študent zvolí parametre pre test, ktorý mu bude vygenerovaný a klikne na tlačidlo „Spustiť“. Následne je vygenerovaný test a študentovi je zobrazená prvá otázka. Študent môže medzi otázkami prechádzať v ľubovoľnom poradí, po prejdení na inú otázku je automaticky uložená jeho odpoveď na otázku, z ktorej odišiel. Toto platí aj v prípade, že sa

študent rozhodne prejsť na prehľad tohto testu. V prehľade je zobrazená tabuľka otázok, spolu so stavom týchto otázok - či na túto otázku študent odpovedal, alebo nie. V prípade že je študent spokojný so svojimi odpoveďami, môže tento test odoslať k vyhodnoteniu. Po odoslaní mu je zobrazený prehľad otázok z tohto testu s jeho odpoveďami na ne, spolu s celkovými výsledkami tohto testu - počet získaných bodov atď. Pre opustenie tohto prehľadu je študentovi k dispozícii tlačidlo „Ukončiť test“.



Obr. 2.2 Stavový diagram priebehu samostatného testu

2.4 Architektúra serverovej časti aplikácie

V tejto sekcií sa budem venovať rôznym typom softwarových architektúr, ktoré sa používajú pri vývoji aplikácií. Vytvorím prehľad o rôznych architektúrach a následne vyberiem jednu z nich, ktorou sa následne bude riadiť implementácia projektu.

2.4.1 Monolitická architektúra

V monolitickej architektúre všetko funguje na jednej vrstve. Túto architektúru je možné využiť na malé projekty bez príliš komplexnej funkcionality. Na väčšie a komplexnejšie aplikácie je vhodné použiť viac-vrstvovú architektúru. [10]

2.4.2 Dvojvrstvová architektúra

Dvojvrstvová architektúra, oddeľuje logiku a správu dát do separátnych vrstiev. Bežne sa tieto vrstvy označujú ako prezentačné a dátové. Prezentačná vrstva má na starosť vystavovanie dát ktoré sú uložené na dátovej vrstve, prípadne prijímanie dát na uloženie na dátovej vrstve. Dátová vrstva má na starosti správu dát a poskytovanie týchto dát prezentačnej vrstve. Akákoľvek aplikačná logika aplikácie používajúcej túto architektúru sa môže nachádzať na jednej, alebo druhej vrstve, niekedy do konca aj na oboch. Toto má za následok to, že koncový užívateľ má priamo prístup k dátam na dátovej vrstve, preto sa táto architektúra zriedka používa k poskytovaniu dát užívateľovi. Naopak táto architektúra sa využíva najčastejšie v aplikáciách ktoré, prijímajú požiadavky od iných aplikácií a poskytujú im dáta. [10]

2.4.3 Trojvrstvová architektúra

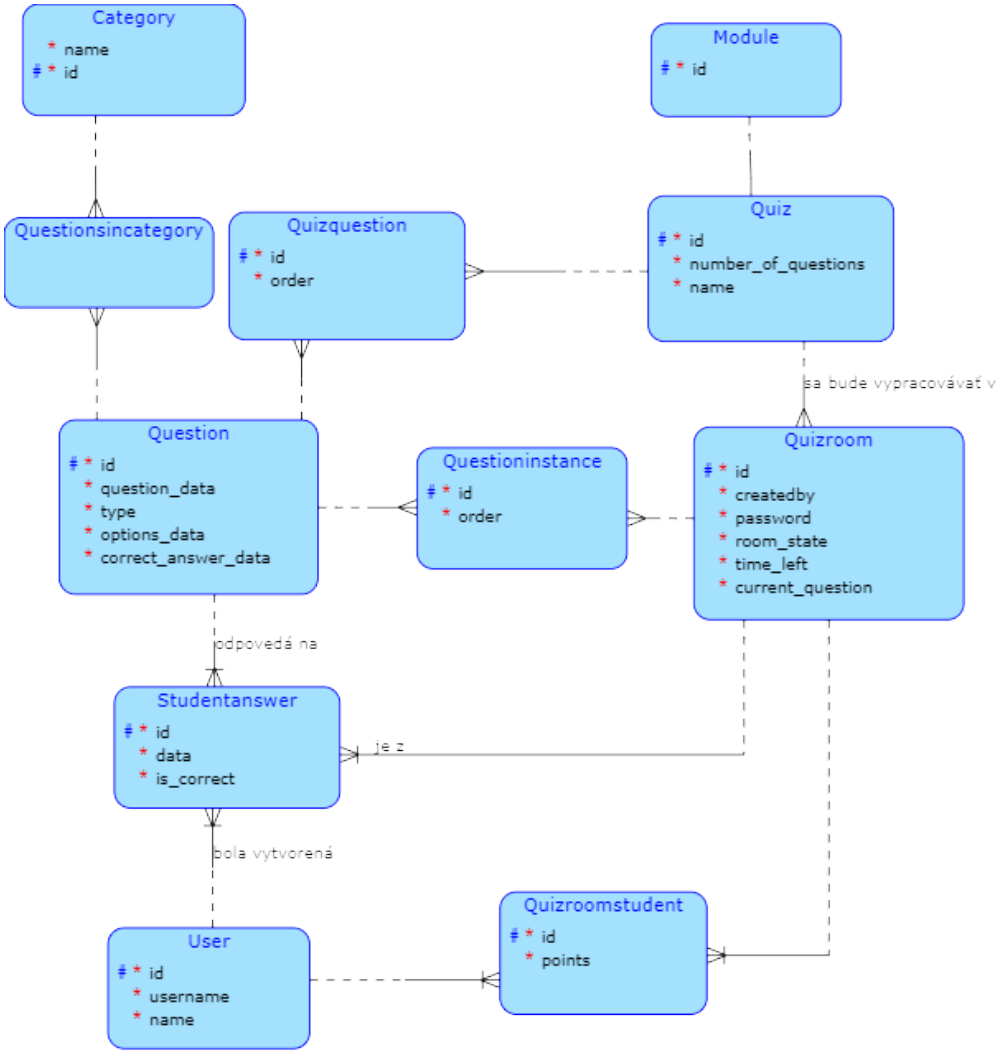
Trojvrstvová architektúra je v niektorých ohľadoch podobná tej dvojvrstvovej. Na rozdiel od nej však táto architektúra má navyše vrstvu aplikačnej logiky, teda má 3 vrstvy - prezentačnú, aplikačnú a dátovú. Dátová vrstva má na starosti správu dát a prezentačná vrstva má na starosti prijímanie požiadavkov od klienta, či už je klientom fyzický užívateľ, alebo iná aplikácia. Aplikačná vrstva má na starosti samotné spracovanie požiadavkov. Pod týmto si môžme predstaviť jednoduché procesy ako kontrolu správnosti a kompletnosti dát, zavolanie CRUD operácie, alebo zložitejšie procesy ako napríklad rôzne výpočty, atď. [11]

2.4.4 Vybraná architektúra

Pre tento projekt som sa rozhodol využiť trojvrstvovú architektúru. Toto rozhodnutie som učinil na základe analýzy zložitosti tejto domény a operácií, ktoré aplikácia, ktorá túto doménu implementuje musí vykonávať. Tieto operácie môžu byť relatívne jednoduché ako napr. vytvorenie otázky, zložitejšie vytvorenie kvízu, ktorý bude nejaké otázky samozrejme obsahovať, až po zložité operácie ako zobrazovanie študentských riešení, alebo spätná oprava hodnotenia otázky. Všetky tieto operácie by bolo pravdepodobné možné realizovať na dátovej alebo prezentačnej vrstve, ale zdrojový kód by bol vo výsledku pravdepodobne veľmi neprehľadný a ťažko rozšíriteľný nakoľko by dochádzalo k miešaniu zodpovedností jednotlivých vrstiev.

2.5 Návrh databázy

Z analýzy vyšlo najavo, že je potrebné niektoré dáta ukladať. Nižšie je možné vidieť návrh jednotlivých entít resp. tabuliek, ktoré sú potrebné pre ukladanie dát podľa analýzy funkčných požiadavkov.



Obr. 2.3 Diagram databázovej štruktúry

2.5.1 Question

Tabuľka Question obsahuje všetky potrebné dáta o otázkach, ktoré budú užívatelia - prevažne učitelia, vytvárať. Jediné dáta, ktoré priamo táto tabuľka neobsahuje, sú informácie o tom, do ktorých kategórií konkrétna otázka patrí. Toto je zabezpečené M:N väzbou medzi tabuľkami Category a Question, čo spôsobí, že po dekomponovaní tejto väzby vznikne nová entita QuestionsInCategory, obsahujúca dáta o tom, do akých kategórií patria otázky, uložené v databáze.

2.5.2 Quiz

Tabuľka Quiz obsahuje všetky potrebné dáta o kvízových a testových moduloch, ktoré budú učitelia vytvárať, okrem samotných otázok. Tabuľka Quiz je naviazaná na tabuľku Question M:N väzbou, čo spôsobí, že po de-komponovaní tejto väzby vznikne nová tabuľka QuizQuestion. Tabuľka Quiz je naviazaná ešte na tabuľku Module väzbou 1:1, nakoľko tabuľka kvíz reprezentuje kvízový resp. testový modul. Táto tabuľka, spoločne s tabuľkou User je v časti databáze, ktorú

navrhuje Bc. Ondřej Wrzieconko v jeho diplomovej práci.

2.5.3 Quizroom

Tabuľka Quizroom obsahuje dáta o konkrétnom behu kvízu resp. testu (kvízového resp. testového modulu) - teda po tom, ako učiteľ sprístupní kvíz študentom a je vytvorená tzv. nová kvízová resp. testová miestnosť, je vytvorený v tejto tabuľke nový riadok so záznamom o tejto miestnosti. Táto tabuľka je naviazaná na tabuľku Quiz väzbou N:1, čo umožňuje opakované spustenie toho istého kvízového resp. testového modulu viacnásobne. Dá sa teda o tejto entite uvažovať akoby o inštancií konkrétneho modulu, nezávisle na tom či sa jedná o kvízový, alebo testový modul.

2.5.4 Quizquestion a QuestionInstance

Tabuľky Quizquestion a QuizInstance plnia veľmi podobný účel - udržujú informácie o otázkach, ktoré sú vyplňované v danom kvíze resp. kvízovej miestnosti (v závislosti na entite na ktorú sú naviazané). Entita QuizQuestion udržuje otázky, ktoré sú prítomné v kvízovom module a otázky z ktorých budú v testovom module generované študentom testy. Entita QuesionInstance je využívaná len v testovom module a jej úlohou, je ukladať otázky, z ktorých je tvorený vygenerovaný „samotest“.

2.5.5 StudentAnswer

Tabuľka StudentAnswer obsahuje dáta o odpovediach študentov na otázky v kvízoch a testoch. Táto entita má väzby na tabuľky Question, Quizroom a User, všetky typu N:1. Tieto väzby sú podstatné obzvlášť pri hľadaní odpovedí študenta v databáze, nakoľko vďaka N:1 väzbám na ostatné entity, je dosiahnuté, že každá odpoveď je jednoznačne identifikovateľná podľa otázky (Question), kvízovej miestnosti (Quizoom) a užívateľa (User).

2.5.6 Category

Tabuľka Category obsahuje dáta o rôznych kategóriách otázok. Na tabuľku Question je naviazaná M:N väzbou, čo spôsobí, že po dekomponovaní tejto väzby vznikne nová entita QuestionsInCategory, ako som popísal už vyššie. Tabuľku Category by malo byť možné použiť aj na kategorizáciu iných inštancií, nielen otázok. Túto tabuľku navrhol ako obecné riešenie na kategorizáciu entít tím študentov predmetu BI-SP1.

2.5.7 User

Tabuľka User obsahuje dáta o všetkých užívateľoch systému Trainer. Spoločne s tabuľkou Module patrí do časti systému, ktorú navrhuje Bc. Ondřej Wrzieconko v jeho diplomovej práci.

2.5.8 Module

Tabuľka Module obsahuje dáta o existujúcich moduloch systému Trainer, nech už sú akéhokoľvek typu. Ako som už vyššie spomenul, táto tabuľka patrí do časti systému, ktorú navrhuje Bc. Ondřej Wrzieconko v jeho diplomovej práci.

2.6 Návrh API

Ako som už v analýze funkčných požiadavkov pri niektorých požiadavkoch naznačil, bude potrebné implementovať zdroje API, na ktoré užívatelia budú môcť posielať požiadavky. Pri každom zdroji uvádzam HTTP metódu a príslušnú príponu zdroju.

2.6.1 Zdroj /questions

Zdroj /questions má na starosti spracovávanie požiadavkov, týkajúcich sa otázok. Systém pod URI /questions vystavuje nasledujúce zdroje.

GET (/) : Prijíma dobrovoľný parameter „author“. V prípade, že je tento parameter poskytnutý vystaví zoznam otázok, ktoré vytvoril užívateľ s užívateľským menom zhodným s tým z parametru „author“. V opačnom prípade vystaví zoznam všetkých existujúcich otázok.

GET (/{id}) : Vystaví detail otázky so špecifikovaným id. Neobsahuje informáciu o správnej odpovedi a vysvetlení danej otázky.

GET (/{id}/correct) : Vystaví správnu odpoveď a vysvetlenie otázky so špecifikovaným id.

POST (/) : Z dát, ktoré prijme vytvorí novú otázku v databáze.

PUT (/{id}) : Upraví otázku so špecifikovaným id (ak existuje), podľa prijatých dát.

DELETE (/{id}) : Vymaže otázku so špecifikovaným id z databázy.

2.6.2 Zdroj /quizzes

GET (/) : Vystaví zoznam všetkých existujúcich kvízov.

GET (/{id}) : Vystaví detail kvízu so špecifikovaným id.

GET (/module/{moduleId}) : Vystaví detail kvízu, ktorý má väzbu na modul so špecifikovaným id. Takýto kvíz môže byť maximálne 1, nakoľko väzba medzi tabuľkami Quiz a Module je 1:1.

POST (/) : Z dát, ktoré prijme vytvorí nový kvíz v databáze.

PUT (/{id}) : Upraví kvíz so špecifikovaným id (ak existuje), podľa prijatých dát.

DELETE (/{id}) : Vymaže kvíz so špecifikovaným id z databázy.

2.6.3 Zdroj /rooms

GET (/) : Vystaví zoznam všetkých existujúcich kvízových miestností, v ktorých niekedy prebiehal kvíz.

GET (/{id}) : Vystaví detail kvízovej miestnosti so špecifikovaným id.

GET (/{id}/students) : Vystaví zoznam študentov, ktorí sa prihlásili do kvízovej miestnosti so špecifikovaným id.

GET (/{id}/students/{studentId}) : Vystaví detail študenta v kvízovej miestnosti. Tento detail obsahuje napríklad informáciu o bodoch, ktoré tento študent v tejto kvízovej miestnosti doteraz získal. Kvízová miestnosť a študent sú špecifikované v parametroch id resp. studentId.

POST (/) : Z dát, ktoré prijme vytvorí novú kvízovú miestnosť v databáze.

POST (/{id}/students) : Pridá študenta do kvízovej miestnosti, teda vytvorí záznam v tabuľke QuizroomStudent, s kvízovou miestnosťou špecifikovanou v parametre id a študentovi špecifikovanom v dátach, ktoré tento zdroj prijme.

PUT (/{id}) : Upraví kvízovú miestnosť so špecifikovaným id (ak existuje), podľa prijatých dát.

PUT (/{id}/students/{studentId}) : Upraví záznam študenta, špecifikovaného v parametre studentId, v kvízovej miestnosti, špecifikovanej v parametre id, podľa prijatých dát. Tento zdroj môže byť využitý napríklad k úprave počtu získaných bodov u konkrétneho študenta v danej kvízovej miestnosti.

DELETE (/{id}) : Vymaže kvízovú miestnosť so špecifikovaným id z databázy.

2.6.4 Zdroj /studentAnswers

GET (/) : Prijíma nepovinné parametre „quizroom“, „question“ a „student“. V prípade že niektorý z parametrov bol poskytnutý, alebo ich kombinácia bola poskytnutá vystaví zoznam všetkých študentských odpovedí, ktoré boli vytvorené v špecifikovanej kvízovej miestnosti (quizroom), na špecifikovanú otázku (question), špecifikovaným študentom (student). Ak nie je žiadny z parametrov poskytnutý vystaví zoznam všetkých existujúcich študentských odpovedí.

GET (/{id}) : Vystaví odpoveď študenta so špecifikovaným id.

POST (/) : Z dát, ktoré prijme vytvorí novú študentskú odpoveď v databáze.

PUT (/{id}) : Upraví odpoveď so špecifikovaným id (ak existuje), podľa prijatých dát. Úprava študentskej odpovede prebehne len v prípade, že sa jedná o odpoveď v testovom module. V kvízovom module študenti nemajú možnosť svoju odpoveď po odoslaní upraviť, v testovom module áno.

DELETE (/{id}) : Vymaže odpoveď so špecifikovaným id z databázy.

2.7 Technológie - Databáza

2.7.1 Ukladanie do súborov

Jedným z možných spôsobov ukladania dát je ukladanie do súborov. Toto môže prebiehať buď štruktúrovane - dáta sú ukladané do súborov podľa nejakého formátu napríklad JSON, alebo CSV, alebo neštruktúrovane – dáta sú ukladané tak ako prídu na vstup bez nejakej ďalšej úpravy alebo formátovania. Keďže dáta, ktoré sa budú v tomto projekte ukladať sú v rámci jednej tabuľky rovnaké resp. atribúty sú rovnaké, dáta sú samozrejme rozdielne, bolo by určite vhodnejšie použiť štruktúrovanú variantu. Navyše jazyky ako napr. Python používajú tzv. Dictionaries [1], ktoré sú veľmi podobné dátam vo formáte JSON a tým pádom by bol načítavanie resp. ukladanie dát do súborov relatívne jednoduché.

[1] Slovníky

2.7.2 Relačné databázy

Relačné databázy sa oproti iným spôsobom ukladania dát odlišujú tým, že dáta ukladajú do oddelených tabuliek, ktoré majú vlastné atribúty. Medzi tabuľkami sú potom vytvárané väzby, ktoré vyjadrujú nejaké previazanie medzi nimi. Okrem týchto základných prvkov je možné vytvoriť ďalšie pravidlá – tzv. integritné obmedzenia, ktoré budú ďalej obmedzovať logiku databázovej štruktúry.

2.7.2.1 MySQL

MySQL [12] je najpopulárnejší open-source typ relačnej databáze, vďaka čomu má veľkú komunitu. To môže, podobne ako u väčšiny technológií, byť príjemný benefit v prípade, že vývojár narazí na problém a nevie ako ho vyriešiť, je veľká pravdepodobnosť, že niekto z tejto komunity tento problém už v minulosti riešil. Navyše je dostupná k použitiu úplne zdarma, čo je pravdepodobne jeden z dôvodov vysokej popularity. MySQL databáze sú obecne relatívne jednoduché na vývoj a používanie – vytvorenie MySQL nezaberie vývojárovi, ktorý vie čo robí viac ako 10 minút. Ďalej sa vyznačuje jej spoľahlivosťou, bezpečnosťou a vysokým výkonom.

2.7.2.2 PostgreSQL

PostgreSQL [13] je špeciálny typ relačnej databáze, tzv. objektovo-relačná databáza. Vo veľa ohľadoch sú PostgreSQL a MySQL databáze veľmi podobné a ich použiteľnosť je takmer identická. PostgreSQL databázy je vhodné použiť napríklad keď očakávame, že táto databáza bude spracovávať veľké množstvo požiadavkov na zmenu dát, nakoľko v tomto ohľade sú tieto databázy rýchlejšie ako MySQL databázy. Naopak MySQL je vhodné použiť ak očakávame veľké množstvo požiadavkov na čítanie dát, z rovnakého dôvodu. Tento databázový typ je navyše dostupný a použiteľný zdarma a je, podobne ako MySQL, open-source.

2.7.3 Vybraná technológia

Z analýzy a návrhu je jasné, že dáta, ktoré budú odosielané na server, budú štukturované. Ukladanie dát do súborov, by preto z hľadiska manažovania dát by bolo veľmi náročné na údržbu. Bc. Ondřej Wrzieconko sa rozhodol použiť pre jeho časť systému MySQL databázu a keďže kvízový a testový modul tento systém dopĺňajú som sa aj ja rozhodol použiť ako dátové úložisko MySQL databázu, ktorá bude následne integrovaná do databázy implementovanej Bc. Ondřej Wrzieconkom, čím vznikne jedna spoločná databáza pre celý systém.

2.8 Technológie - Internetové protokoly

Na komunikáciu medzi serverom a klientom je vždy treba použiť nejaký internetový protokol. Na bežnú komunikáciu, ktorá nevyžaduje nič viac ako odoslanie dát zo serveru klientovi, ktorý tieto dáta vo vhodnej forme zobrazí, je možné použiť protokol HTTP. Nakoľko je však potrebné, aby sa otázky v kvízovom module zobrazili všetkým účastníkom naraz, HTTP protokol nie je sám o sebe dostatočný a je potrebné použiť protokol TCP.

2.8.1 HTTP

HTTP [14], alebo Hypertext Transfer Protocol, je protokol na prenos dát cez internet. Na to, aby bolo možné komunikovať pomocou tohto protokolu je potrebné poznať URI, na ktorú je potrebné poslať požiadavok, čím sa vytvorí medzi odosielateľom a URI spojenie. Tento požiadavok je úplne nezávislý od akýchkoľvek iných požiadavkov, ktoré na túto URI prichádzajú. Následne,

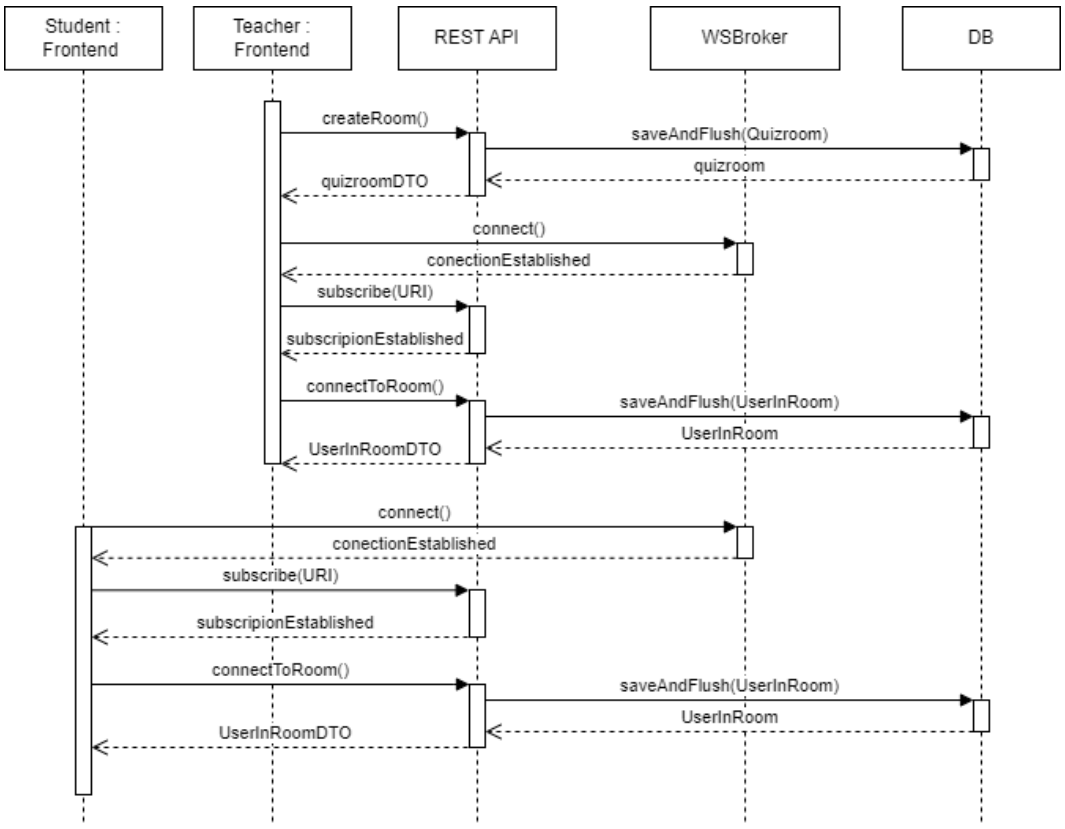
po spracovaní požiadavku je odoslaná z tejto URI odpoveď na požiadavok a tým je spojenie ukončené. Existuje niekoľko typov požiadavkov - GET, POST, PUT, DELETE (a PATCH).

2.8.2 TCP

TCP [15], alebo Transmission Control Protocol, je podobne ako HTTP, protokol slúžiaci na prenos dát cez internet. Oproti HTTP však funguje veľmi odlišne. Spojenie medzi klientom a serverom je vytvorené po tom, ako sa klient pokúsi pripojiť na adresu, na ktorej server beží a ako reakciu na túto akciu server pošle klientovi tzv. acknowledgement [2], čím dá klientovi vedieť, že spojenie bolo nadviazané. Na tomto spojení môže následne prebiehať komunikácia až dokým sa jedna zo strán nerozhodne toto spojenie ukončiť.

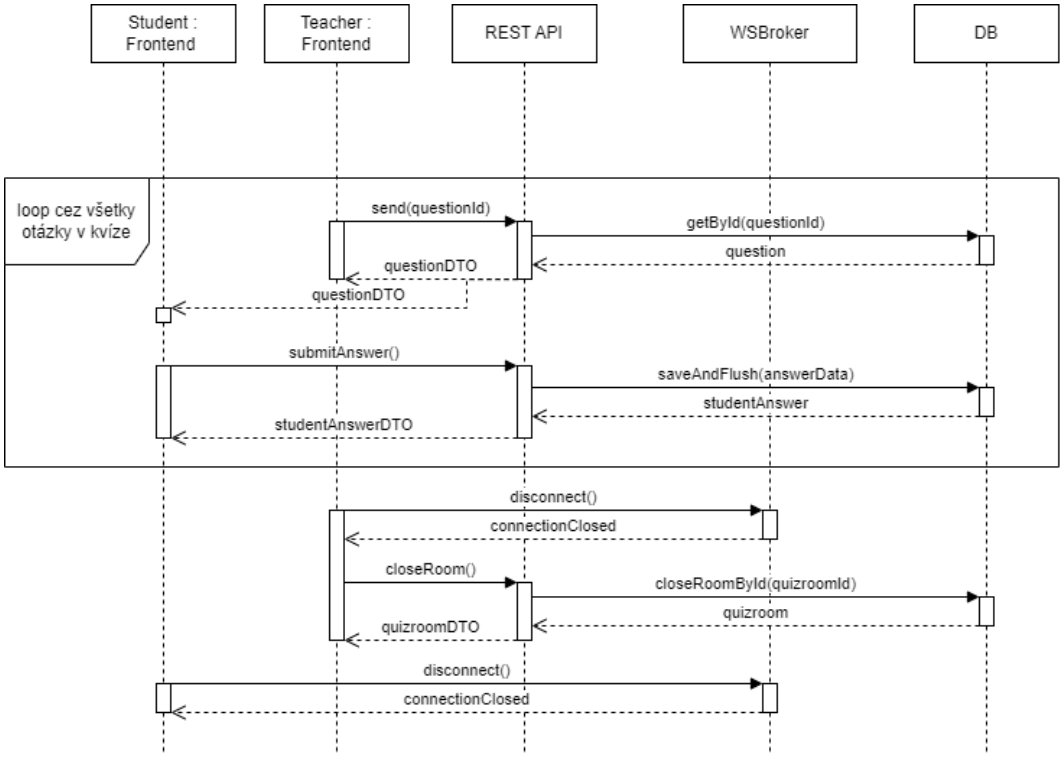
2.8.3 Popis komunikácie

V diagrame nižšie je popísaná komunikácia medzi backendom a frontendom. Vlákna REST a WSBroker, reprezentujú rozhrania s ktorými komunikuje klient pomocou HTTP (REST) a TCP (WSBroker) protokolov. Na začiatku komunikácie je vytvorené spojenie medzi klientom a WSBrokerom a následne začne klient počúvať na vybranej URI, kam budú chodiť správy s dátami obsahujúcimi nové otázky. Tieto dáta budú na tejto URI sprístupnené v rovnaký čas pre všetkých pripojených klientov - keď učiteľ odošle „send“ požiadavok na túto URI.



Obr. 2.4 Sekvenčný diagram komunikácie počas kvízu - časť 1.

[2] Špeciálny packet obsahujúci špecifickú sekvenciu čísel



Obr. 2.5 Sekvenčný diagram komunikácie počas kvízu - časť 2.

2.9 Technológie - Server

2.9.1 Framework

V tejto sekcií sa venujem výberu vhodného frameworku pre našu aplikáciu. Analyzujem niekoľko potenciálne použiteľných technológií a na konci sekcie vyberiem jednu, ktorá bude následne použitá pri implementácií serverovej časti aplikácie.

2.9.1.1 Django

Django [16] je framework pre tvorbu serverovej časti webových aplikácií. Tento framework je naprogramovaný v programovacom jazyku Python, preto je možné tento framework použiť v kombinácií práve s týmto programovacím jazykom. Tento framework je navyše open-source a je zadarmo.

2.9.1.2 Rails

Ruby on Rails [17], alebo skrátene Rails, je framework naprogramovaný v programovacom jazyku Rubi. Tento framework narozdiel od iných poskytuje podporu pre tvorbu serverovej aj klientskej časti webových aplikácií. To môže byť značnou výhodou pre vývojárov, ktorí majú skúsenosti s programovacím jazykom Rubi, keďže sa nemusia zapodievať výberom a učením ďalšieho frameworku, či už pre serverovú, alebo klientsku časť webovej aplikácie. Navyše Rails je možné využiť k vývoju zadarmo.

2.9.1.3 ASP.NET

.NET je framework vyvinutý spoločnosťou Microsoft. ASP.NET [18] je súčasťou tohto frameworku, ktorá ho rozširuje o nástroje na tvorbu webových aplikácií. Tento framework dokáže pracovať s niekoľkými možnými programovacími jazykmi, konkrétne C#, F# a Visual Basic. Toto dáva vývojárom viac flexibility a vďaka tomuto môže oveľa viac vývojárov tento framework využívať. Podobne ako Rubi, ASP.NET je možné využiť na tvorbu serverovej aj klientskej časti webových aplikácií, dokonca je možné klientsku časť webovej aplikácie napísať priamo pomocou jazyku C#, čo značne uľahčuje prácu vývojárom, ktorí nemajú skúsenosti s vývojom webových rozhraní pomocou jazykov ako napríklad Javascript. Tento framework je taktiež zadarmo a môže ho použiť k vývoju ktokoľvek.

2.9.1.4 Spring

Spring [19] je framework, ktorý poskytuje platformu pre vývoj veľkých enterprise aplikácií pomocou programovacích jazykov ako Java alebo Kotlin. Je určený najmä pre tvorbu serverovej časti týchto aplikácií. Spring uľahčuje vývojárom vývoj aplikácie tým, že aplikuje princíp IOC[3], čím Spring prevezme zodpovednosť za niektoré low-level procesy a umožňuje tak vývojárovi sústrediť sa na vývoj samotnej aplikačnej logiky. Spring je navyše použiteľný k vývoju zadarmo a je je open-source.

2.9.1.5 Flask

Flask [20] je framework pre tvorbu serverovej časti webových aplikácií. Je založený na WSGI [21] - Web Server Gateway Interface, čo je špecifikácia, ktorá popisuje komunikáciu medzi webovými servermi a webovými aplikáciami, alebo frameworkami vyvinutými v jazyku Python. Umožňuje veľmi jednoduchý vývoj v programovacom jazyku Python. Rovnako ako vyššie spomenuté frameworky, Flask je možné k vývoju webových aplikácií využiť zdarma.

2.9.1.6 Vybraná technológia

Podobne ako pri výbere databázy Bc. Ondřej Wrzieconko, na jeho časť systému, na ktorej kvízový a testový modul stavia použil framework Spring. Z tohto dôvodu, som teda bol aj ja „nútený“ pracovať s týmto frameworkom, čo ale nepredstavovalo žiadny problém, nakoľko s týmto frameworkom mám bohaté skúsenosti.

2.9.2 Programovací jazyk

2.9.2.1 Java

Java [22] je objektovo orientovaný programovací jazyk, ktorý je prevažne interpretovaný. To znamená, že všetky zdrojové kódy sú preložené do vnútornej formy – v prípade Javy sa táto forma volá bajtkód (byte code). Pre vývoj aplikácií v Jave je potrebné mať na zariadení inštalovaný JRE (Java Runtime Enviroment), ktorý obsahuje JVM [23] (Java Virtual Machine) a základné triedy a knižnice pre Javu. Vďaka JVM, ktorá poskytuje prostredie v ktorom je možné vyššie spomínaný bajtkód exekvovať, je nezávislá na platforme a architektúre operačného systému stroja na ktorom beží. Implementuje svoj vlastný GC (Garbage Collector), čo vývojárovi značne uľahčuje prácu s pamäťou. Java je navyše pre vývojárov dostupná zdarma a rovnako sa môže ktokoľvek podieľať na jej vývoji.

[3] Inversion of Control

2.9.2.2 Kotlin

Kotlin [24] je staticky typovaný objektovo orientovaný programovací jazyk, ktorý sa vo veľa ohľadoch podobá na Javu. Prvotná verzia Kotlinu bola dokonca implementovaná pomocou Javy, no dnes sú už nové verzie Kotlinu resp. kompilátoru Kotlinu, vyvíjané priamo v Kotline. Tento programovací jazyk vo veľa ohľadoch stavia na nedostatkoch Javy a napráva ich. Navyše, keďže bol pôvodne napísaný v Jave je akýkoľvek program napísaný v Kotline kompatibilný s Java kódom. To znamená, že je možné aby bola časť projektu vyvinutá v Jave a časť v Kotline. Podobne ako Java je možné tento programovací jazyk použiť k vývoju zdarma.

2.9.2.3 C#

C# [25] [26]je objektovo orientovaný programovací jazyk vyvinutý spoločnosťou Microsoft a je to jeden z jazykov používaných na platforme .NET. Aj vďaka platforme dotnet má veľmi veľa využití - od vývoju hier až po vývoj webových aplikácií. Je to jeden z najviac obecne použiteľných programovacích jazykov. Navyše je veľmi populárny a teda má širokú komunitu vývojárov, ktorá ho používa. Toto môže značne uľahčiť prácu pri narazení na nejaký problém, nakoľko je veľmi pravdepodobné, že na daný problém už niekto narazil v minulosti. Navyše C# je cross-platform, čo znamená, že podobne ako Javu a Kotlin, je možné tento programovací jazyk využiť pre vývoj na rôznych operačných systémoch (platformách).

2.9.2.4 Python

Python [27] je programovací jazyk vyvíjaný organizáciou Python Software Foundation. Tento programovací jazyk má veľmi jednoduchú syntax, čo ho robí relatívne jednoduchým na naučenie pre nováčikov v IT sfére. Vývojári v tomto jazyku majú k dispozícií veľmi široký výber knižníc, ktoré je možné použiť v ich programoch. Často sa jedná o knižnice Pandas, Numpy a Scipy, ktoré vývojárom uľahčujú prácu pri analýze dát a náročných výpočtoch. Práve najmä vďaka týmto knižniciam je Python jedným z obľúbených jazykov pre dátovú analýzu. Okrem dátovej analýzy je však tento jazyk možné použiť na rôzne typy aplikácií. Napríklad sa môže jednať o webovú aplikácia pomocou frameworku Flask, v ktorom aj neskúsený vývojár dokáže vytvoriť jednoduché RESTové API v priebehu niekoľkých minút, alebo sa môže jednať o hru vytvorenú pomocou knižnice Pygame.

2.9.2.5 Ruby

Ruby [28] je objektovo orientovaný skriptovací jazyk. Má jednoduchú syntax, vďaka čomu je relatívne populárny medzi novými vývojármi. Je použiteľný na vývoj širokej škály aplikácií, najbežnejšie sú v ňom však vyvíjané desktopové a webové aplikácie. Na vývoj webových aplikácií je navyše s týmto programovacím jazykom často použitý framework Rails. [4] Ruby ja navyše možné k vývoju použiť zdarma a je open-source, čo znamená, že na vývoji tohto jazyku sa môže podieľať prakticky ktokoľvek.

2.9.2.6 Vybraná technológia

Ako som v predchádzajúcej sekcií už spomínal, ako na implementáciu serverovej časti aplikácie som zvolil framework Spring, kvôli tomu, že s týmto frameworkom, rovnako ako s jazykmi ako Java a Kotlin mám skúsenosti. Preto bol výber programovacieho jazyka rovno obmedzený na výber medzi tieto dva jazyky. Bc. Ondřejom Wrzieconko sa rozhodol na implementáciu jeho časti systému použiť jazyk Kotlin. Implementácia serverovej časti kvízového a testového modulu, mohla prebehnúť aj v jazyku Java, nakoľko Java je kompatibilná s jazykom Kotlin. Napriek tomuto faktu som sa rozhodol na implementáciu použiť aj ja jazyk Kotlin a to z dôvodu, že (ako

[4] resp. Ruby on Rails

som už niekoľko krát spomenul) syntax týchto jazykov je veľmi podobná, miestami až identická a taktiež som sa tento programovací jazyk chcel naučiť.

2.9.3 Komunikácia s databázou

Každý z frameworkov rieši pripojenie k databáze odlišným spôsobom. Ako som spomenul v predchádzajúcej sekcií o frameworkoch, pre implementáciu som zvolil framework Spring. Spring poskytuje hneď niekoľko rozhraní pre komunikáciu s databázou.

2.9.3.1 JdbcTemplate

Trieda JdbcTemplate poskytuje rozhranie cez ktoré je možné komunikovať s databázou. Toto je možné docieliť niekoľkými spôsobmi, základom je však definícia atribútu typu JdbcTemplate v triede, ktorú chceme používať na vykonávanie operácií nad nejakou tabuľkou v DB. Následne je možné využiť niektorú z metód, ktoré toto rozhranie poskytuje - medzi často využívané metódy patria metódy queryForObject a queryForList. Tieto metódy bez akejkoľvek ďalšej konfigurácie vrátia tzv. ResultSet, čo pre ďalšie použitie výsledkov tohto príkazu nie je vždy dostatočné. Je preto ďalej možné implementovať tzv. Mapper, ktorý výsledky daného príkazu dokáže mapovať na atribúty nejakej triedy.

2.9.3.2 JPA

Rozhranie JPA, ktoré poskytuje Spring, poskytuje už implementované základné CRUD operácie. To znamená že vývojárovi pri implementácií nejakej triedy jeho dátovej vrstvy stačí aby toto rozhranie použil - urobil z z tejto triedy potomka JpaRepository a metódy ktoré toto rozhranie poskytuje je možné následne volať z iných tried v aplikácií. Okrem týchto metod je možné v prípade potreby vytvoriť ďalšie a to niekoľkými spôsobmi. Najjednoduchší spôsob, aplikovateľný obzvlášť v prípadoch keď je logika metody jednoduchá, je definovanie operácie ktorá sa bude vykonávať nad tabuľkou v DB pomocou mena danej metody. Tu je potrebné dodržať formu mena danej metódy, inak tento spôsob nebude fungovať. Alternatívou je použiť JPQL[5], čo spočíva v pridaní anotácie JpqlQuery nad definíciu metódy a následnej definície SQL príkazu v tele tejto anotácie. Tento príkaz bude následne vykonávaný nad konkrétnou tabuľkou v DB vždy, keď bude daná metóda zavolaná. Tento postup je využiteľný najmä pri príkazoch, ktoré majú komplikovanejšiu logiku - spájanie tabuliek a pod.

2.9.3.3 Vybraná technológia

Pre komunikáciu s databázou sa Bc. Ondřejom Wrzieconko rozhodol využiť rozhranie JPA. Nakoľko som s týmto rozhraním tiež mal skúsenosti a rozhranie JdbcTemplate mi prišlo oveľa zložitejšie na použitie pri implementácií, som učinil rozhodnutie toto rozhranie použiť taktiež.

2.9.4 API

API, alebo Application Programming Interface, je súbor definícií rôznych metód pre komunikáciu a integráciu aplikácií. Existuje niekoľko rôznych druhov API, preto je potrebné vybrať jeden, ktorý sa pre tento projekt bude hodiť najviac.

[5] Java Persistence Query Language

2.9.4.1 SOAP

SOAP API[29], alebo Simple Object Access Protocol API je druh API, ktorý komunikuje pomocou protokolu HTTP a dátového formátu XML [6]. Vďaka tomu, že používa HTTP ho teda je možné použiť na čítanie, tvorbu, úpravu, či mazanie dát (pomocou na to určených HTTP metód). Pri komunikácií s týmto druhom API špecifikujeme operáciu, ktorú je potrebné vykonať priamo v tele správy, ktorú posielame. [30]

2.9.4.2 REST

Podobne ako SOAP API, aj REST API ku komunikácií využíva protokol HTTP a jeho typy metód. Vďaka tomu, je ho, podobne ako SOAP API, možné použiť na čítanie, tvorbu, úpravu, či mazanie dát, pomocou na to určených HTTP metód. Narozdiel od SOAP API však dokáže pracovať s rôznymi dátovými formátmi, nielen s XML. Asi najrozšírenejšie používaný dátový formát s REST API je JSON, ale podporovaný je aj čistý text, HTML, či vyššie spomínané XML. Ďalším rozdielom je posielanie požiadavkov na tento druh API. REST API má pre každú operáciu, ktorú poskytuje, vystavený zdroj (endpoint), na ktorý je možné posielať požiadavky a po prijatí týchto požiadavkov je daná operácia vykonaná. [30]

2.9.4.3 GraphQL

GraphQL [31] je dotazovací jazyk pre API a operácie s týmto API spojené. Podobne ako SOAP a REST, umožňuje čítanie, upravovanie a mazanie dát. Narozdiel od týchto typov API však GraphQL používa na komunikáciu úplne inú syntax - syntax, ktorá je založená na tom, že aké dáta si užívateľ vypýta v tele požiadavku, také dostane. Nie je preto potrebné poznať mená vystavovaných funkcií, alebo URI zdrojov pre jednotlivé operácie. Naopak stačí odoslať požiadavok na adresu, kde beží toto API, v ktorého tele špecifikujeme o aké dáta, prípade objekty a ich atribúty, máme záujem. Tento požiadavok následne toto API spracuje a odošle odpoveď obsahujúcu presne dáta, ktoré užívateľ požadoval. [32]

2.9.4.4 Vybraná technológia

Z vyššie spomenutých technológií mám skúsenosti s dvomi - REST a SOAP. SOAP API umožňuje výmenu dát len vo formáte XML a nie napríklad vo formáte JSON, čo je značne limitujúce. Dáta odoslané vo formáte JSON dokáže webová aplikácia veľmi jednoducho pomocou jedného príkazu spracovať, dáta odoslané vo formáte XML nie. Z tohto dôvodu som sa rozhodol pre API použiť technológiu REST.

2.10 Technológie - Frontend

2.10.1 Programovacie jazyky

Pri vývoji webových aplikácií je prakticky nemožné vyhnúť sa jazykom HTML a CSS. Nižšie preto veľmi stručne popisujem načo tieto technológie využívam. Pri voľbe programovacieho jazyku však už je možnosť výberu, preto použiteľné technológie zanalyzujem a vyberiem z nich jednu, ktorú následne použijem pri implementácií webového rozhrania.

2.10.1.1 HTML

HTML alebo HyperText Markup Language je značkovací jazyk. Používa sa na vývoj webových stránok väčšinou v kombinácií s nižšie spomenutými technológiami. HTML konkrétne definuje

[6] Extensible Markup Language

štruktúru v akej sú dáta užívateľom zobrazované. Každá webová stránka je zložená z nejakého počtu HTML elementov. Tieto elementy sa odlišujú pomocou tzv. tagov. [33]

2.10.1.2 CSS

CSS alebo Cascading Style Sheets, je rozšírenie základného HTML. Poskytuje možnosť upravovať vzhľad webovej stránky a komponent, ktoré sú na nej vyobrazené. Narozdiel od HTML, ktorý definuje štruktúru webovej stránky, pomocou CSS túto stránku je možné štylizovať a dodať jej tak prívetivé rozhranie. [34]

2.10.1.3 Javascript

Javascript je skriptovací programovací jazyk, známy najmä vďaka jeho veľmi rozšírenému používaniu pre vývoj webových stránok. Tejto popularite tiež prispieva fakt, že existuje niekoľko frameworkov pre vývoj webových stránok, ktoré používajú tento programovací jazyk - niektoré z nich sú spomenuté nižšie v sekcií Framework. Javascript navyše podporuje rôzne druhy programovacích štýlov - objektovo orientované, imperatívne, alebo funkcionálne programovanie, čo z neho robí skvelú voľbu pre vývojárov, ktorý so všetkými týmito paradigmami resp. štýlmi nemajú skúsenosti. Javascript je tak isto možné použiť, väčšinou v kombinácií s nejakým frameworkom, na vývoj backendu webových aplikácií. [35]

2.10.1.4 PHP

PHP [36] je, podobne ako Javascript, rozšírený skriptovací programovací jazyk, ktorý je špeciálne zameraný na vývoj webových aplikácií. Jedným zo základných rozdielov oproti „frontendovému“ Javascriptu je, že všetok kód je spúšťaný na servery, kde je z neho vygenerované HTML, ktoré je následne poslané klientovi. PHP je relatívne jednoduché na porozumenie pre nováčikov, čo je jeden z dôvodov jeho stálej popularity, no zároveň poskytuje veľa funkcionalít pre pokročilých vývojárov, ktoré ale nie sú potrebné pre základy vývoja v tomto jazyku, a ktorým by nováčikovia mali problém porozumieť.

2.10.1.5 Zhrnutie

Pre implementáciu webového klienta som sa rozhodol použiť, spolu s HTML a CSS, programovací jazyk Javascript. K tomuto rozhodnutiu som prišiel na základe mojich osobných skúseností a preferencií a taktiež som bol značne ovplyvnený kolegom Wrzieconkom, ktorý sa pre implementáciu jeho časti systému rozhodol použiť túto kombináciu programovacích jazykov.

2.10.2 Framework

Keďže som sa rozhodol na implementáciu logiky webového rozhrania použiť Javascript, som limitovaný vo výbere webových frameworkov na tie, ktoré podporujú tento jazyk. Nižšie sa preto venujem niektorým z najznámejších frameworkov pre tento jazyk.

2.10.2.1 VueJS

VueJS [37] je framework pre tvorbu UI webových aplikácií. Integruje v sebe HTML, CSS a Javascript a tým poskytuje vývojárom nástroj, ktorý je jednoduchý na používanie. Je založený na component-based[7] programovaní, teda vo väčšine prípadov (obzvlášť pri veľkých aplikáciách), je aplikácia rozdelená do komponentov a každý komponent je spravidla vyvíjaný v zvláštnom súbore

[7] Programovanie po komponentoch

tzv. SFC [8]. Navyše Vue poskytuje 2 rozdielne prístupy k vývoju komponentov - Composition a Options API. Oba tieto „postupy“ sa líšia len v syntaxy no obomi je možné vyvinúť identické UI. Všetok kód v týchto komponentoch je rozdelený do 3 blokov - Script, Template a Style. V Script bloku je spravidla písaný Javascriptový kód, ktorý spravuje logiku tohto komponentu. V bloku Template je priestor na vývoj samotného vzhľadu webovej stránky, teda sa v tomto bloku nachádza prevažne HTML a CSS kód s ojedinelou logikou programovanou v Javascripte, pri renderovaní vybraných elementov.

2.10.2.2 ReactJS

ReactJS [38], alebo skrátene React je framework pre tvorbu dynamických a interaktívnych webových aplikácií. Rovnako ako VueJS aj ReactJS využíva component-based programovanie, čo znamená, že webová aplikácia sa rozdelí na komponenty, ktoré sú vyvíjané oddelene. Jednou z výhod tohto prístupu, okrem prehľadnejších zdrojových kódov, fakt, že tieto komponenty (po tom ako sú vyvinuté) sú použiteľné v aplikácií kdekoľvek je potreba. Z osobnej skúsenosti môžem povedať, že napriek podobným princípom ako vo VueJS, sú niektoré koncepty tohto frameworku, aspoň podľa môjho názoru zložitejšie na pochopenie. Pre nováčika teda môže byť zložitejšie začať vyvíjať v tomto frameworku, ako napríklad vo VueJS. [39]

2.10.2.3 Angular

Angular [40] je, podobne ako ReactJS a VueJS, framework pre tvorbu webových rozhraní. Avšak narozdiel od týchto frameworkov, Angular používa programovací jazyk TypeScript. Tento programovací jazyk je veľmi podobný Javascriptu, no oproti Javascriptu pridáva typy premenných. To znamená, že narozdiel od kódu v Javascripte, neni možné vytvárať premenné bez explicitne deklarovaných dátových typov. Toto môže, v závislosti na vývojárovi, byť výhoda aj nevýhoda tohto programovacieho jazyku. Rovnako ako vyššie spomenuté frameworky využíva component-based programovanie s menším rozdielom. Každá komponenta je zložená z 2 hlavných súborov (častí). Jeden súbor je formátu HTML, v tomto súbore je deklarované rozhranie a štylizácia. Druhý súbor obsahuje logiku danej komponenty v typescriptovom kóde. Angular je volne dostupný a je ho možné pre vývoj použiť zdarma.

2.10.2.4 Vybraná technológia

Podobne ako pri výbere ostatných technológií, aj tu som bol ovplyvnený kolegom Bc. Ondřejom Wrzieconkom, ktorý v čase keď ja som začal na projekte pracovať už mal technológie, ktoré použije premyslené. Avšak narozdiel od serverovej časti, kde som mal priestor na použitie odlišných technológií, pri vývoji webovej stránky by použitie odlišných technológií, ako napr. odlišných frameworkov alebo jazykov, bolo značne problematické, až nemožné. Navyše som bol motivovaný sa v tomto frameworku naučiť pracovať a preto sme sa po spoločnej dohode s kolegom Wrzieconkom rozhodli použiť framework VueJs.

[8] Single File Component

Kapitola 3
Implementácia

V tejto kapitole sa budem venovať samotnej implementácií kvízového a testového modulu. Popíšem aké postupy som pri vývoji zvolil, niektoré problémy s ktorými som sa pri vývoji stretol a popíšem jeho postup, vrátane niektorých ciest ktorými som sa vo vývoji vybral a nakoniec nikam neviedli.

3.1 Server

Na implementáciu backendu bola použitá trojvrstvová architektúra. To znamená, že je rozdelený na dátovú, aplikačnú/business a prezentačnú vrstvu. Dátová vrstva je zložená z 2 balíčkov tried - Entity a Repository. Triedy v týchto balíčkoch majú na starosti správu dát v databáze a poskytujú operácie nad tabuľkami v nej. Aplikačnú vrstvu implementuje balíček Service. Tento balík resp. triedy v tomto balíku obsahujú drvivú väčšinu aplikačnej logiky - od obmedzenia prístupov k niektorým operáciám, až po rôzne zložité operácie, výpočty atď. Vrstva prezentačná je obsiahnutá v balíčku api. Triedy v tomto balíčku obsahujú implementáciu REST API, ktoré je poskytované webovému frontendu na komunikáciu a výmenu dát. Posledným balíčkom, ktorý takpovediac spadá pod 2 vrstvy - aplikačnú a prezentačnú, je balík DTO [1]. Tento balík obsahuje triedy, ktoré definujú, ako budú vyzerať JSON objekty, ktoré budú posielané cez API na frontend a naopak. Táto sekcia sa venuje implementácií týchto vrstiev a tried obsiahnutých v týchto balíkoch. Prakticky v každej vrstve je aspoň 1 centrálne rozhranie ktoré všetky ostatné triedy v danej vrstve implementujú. Tieto centrálne rozhrania implementoval kolega Bc. Ondřej Wrzieconko, rovnako ako autentifikáciu požiadavkov a zabezpečenie API, ktoré sú súčasťou týchto rozhraní, alebo sú implementované v niektorom z jeho potomkov.

3.1.1 Entity

Každá databázová tabuľka je reprezentovaná jednou triedou z balíčku Entity. Tento balík je možné prirovnať napríklad modelovej časti architektúry MVC [2], obsahuje triedy, ktoré definujú ako sú dáta ukladané. Navyše na základe definícií týchto tried sú vytvorené samotné databázové tabuľky, spolu s ich atribútmi a dátovými typmi týchto atribútov a integritnými obmedzeniami. Toto je zabezpečené pomocou JPA [3] a frameworku hibernate, ktoré umožňujú ORM. [4] Všetky triedy v tomto baličku navyše dedia od spoločného rozhrania IEntity, ktoré definuje 1 atribút - id

[1] Data Transfer Object
[2] Model-View-Controller
[3] Java Persistence API
[4] Object–relational mapping teda objektovo-relačné mapovanie

a 2 metódy - canView a canEdit, ktoré sa vyžívajú na kontrolu prístupu k vybraným operáciám. Nižšie je možné vidieť príklad implementácie triedy Quiz z tohto balíčku.

```
@Entity
class Quiz(

    @Column(nullable = false) val numOfQuestions : Int,
    @Column(nullable = false) val name : String,
    @Column(nullable = true) val numOfAttempts : Int,

    @OneToMany(mappedBy = "quiz")
    @OnDelete(action = OnDeleteAction.CASCADE)
    val quizQuestions : List<QuizQuestion>,

    @OneToMany(mappedBy = "quiz")
    val quizRooms : List<Quizroom>,

    @OneToOne
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JoinColumn(name = "module_id", nullable = true)
    val module: Module,

    override val id: Int = 0
) : IEntity(id) {
    override fun canView(user: User?): Boolean {
        return module.canView(user)
    }

    override fun canEdit(user: User): Boolean {
        return module.canEdit(user)
    }
}
```

Výpis kódu 3.1 Implementácia triedy Quiz

3.1.2 Repository

V balíčku Repository sú zdrojové kódy k rozhraniam, ktoré slúžia na komunikáciu s databázou. Každé z týchto rozhraní dedí (rozširuje) centrálne rozhranie IRepository. Toto centrálne rozhranie dedí od rozhrania JpaRepository, ktoré poskytuje framework Spring, resp. konkrétne JPA, ktoré framework Spring v sebe zaobaľuje. Toto rozhranie navyše vyžaduje aby mu vývojár poskytol triedu, ktorá je mapovaná na niektorú z databázových tabuliek a dátový typ identifikátora tejto tabuľky. JpaRepository navyše už obsahuje implementované základné CRUD operácie nad touto tabuľkou a je možné,v prípade potreby pridať ďalšie jedným z dvoch spôsobov, ktoré som popisoval v sekcií JPA v kapitole návrh. Nižšie je zobrazená ukážka rozhrania QuizRepository.

```
@Repository
interface QuizRepo : IRepository<Quiz> {

    fun existsByModule(module: Module) : Boolean
    fun getByModule(module: Module) : Quiz
}
```

Výpis kódu 3.2 Implementácia rozhrania QuizRepository

3.1.3 DTO

Balíček DTO obsahuje dátové triedy, ktoré reprezentujú štruktúru dát, ktoré API posiela na frontend, alebo naopak prijíma v tele požiadavkov, ktoré na API prídu. V aplikácií rozoznávam 4 rôzne, takmer na každom zdroji API používané, druhy DTO - GetDTO, FindDTO, PostDTO, UpdateDTO. Každý z týchto druhov má definované vlastné rozhranie, z ktorého sú následne pomocou dedičnosti definované DTO pre všetky potrebné triedy z balíku Entity. Nižšie sa nachádza ukážka kódu týchto tried pre triedu Quiz.

```
//-> out
data class QuizFindDTO(override val id: Int, val name : String,
                       val numOfQuestions: Int, val numOfAttempts: Int)
                       : IFindDTO
//-> out
data class QuizGetDTO(override val id: Int, val name : String,
                      val numOfQuestions: Int, val numOfAttempts: Int,
                      val questions: List<Int>) : IGetDTO

//<- in
data class QuizUpdateDTO (val name : String?, val numOfQuestions: Int?,
                          val numOfAttempts: Int?, val moduleId : Int?,
                          val questions: List<Int>?) : IUpdateDTO
//<- in
data class QuizCreateDTO (val name : String, val numOfQuestions: Int,
                          val numOfAttempts: Int, val questions: List<Int>,
                          val moduleId : Int) : ICreateDTO
```

Výpis kódu 3.3 Implementácia rozhrania QuizRepository

3.1.4 Service

Balík Service obsahuje triedy, ktoré majú na starosti drvivú väčšinu logiky tejto aplikácie. Podobne ako v ostatných vrstvách/balíkoch aj tu všetky triedy dedia od spoločného predka, no tentokrát týmto predkom nie je len rozhranie s niekoľkými obecnými metódami, ale aj trieda, ktorá niektoré z týchto metód implementuje a navyše implementuje metódy určené na identifikáciu užívateľa, od ktorého prišiel požiadavok na vykonanie vybranej operácie a prípadné zamedzenie prístupu k tejto operácií. Nižšie je možné vidieť časť implementácie jednej z týchto tried. Implementácia týchto tried sa spravidla skladá z 2 častí.

1. V prvej časti sú implementované metódy, ktoré majú na starosti časť logiky aplikácie. Spravidla sa jedná o metódy, ktoré volá API, po tom ako naň príde požiadavok od užívateľa.
2. V druhej časti je implementovaný postup konverzie dát, ktoré majú byť odoslané na frontend na špecifikovaný typ DTO a opačne, v závislosti od požiadavku a typu DTO.

```
@Service
class QuizService(override val repository: QuizRepo,
                  private val moduleRepo: ModuleRepository,
                  private val quizQuestionRepo : QuizQuestionRepo,
                  private val questionRepo: QuestionRepo,
                  userRepo: UserRepository
)
    : IServiceImplOld<Quiz, QuizFindDTO, QuizGetDTO, QuizCreateDTO,
                      QuizUpdateDTO>(repository, userRepo){

    // part 1
```

```
fun getQuizByModuleId(moduleId : Int, user: UserFindDTO?) = tryCatch {
    if(!moduleRepo.existsById(moduleId))
        throw ResponseStatusException(HttpStatus.NOT_FOUND)

    if(!repository.existsByModule(moduleRepo.findById(moduleId).get()))
        throw ResponseStatusException(HttpStatus.NOT_FOUND)

    repository.getByModule(moduleRepo.findById(moduleId)
        .get()).toGetDTO()

// part 2

override fun Quiz.toGetDTO() = QuizGetDTO(id, name, numOfQuestions,
    numOfAttempts, quizQuestions.map { it.question.id })

override fun QuizCreateDTO.toEntity() : Quiz {
    questions.forEach{
        if(!questionRepo.existsById(it))
            throw ResponseStatusException(HttpStatus.NOT_FOUND)
    }

    if(!moduleRepo.existsById(moduleId)){
        throw ResponseStatusException(HttpStatus.NOT_FOUND)
    }


    return Quiz(numOfQuestions, name, numOfAttempts, emptyList(),
        emptyList(), moduleRepo.findById(moduleId).get())
}
```

}

Výpis kódu 3.4 Implementácia triedy QuizService

3.1.5 API

Balík API obsahuje triedy, implementujúce samotné REST API, ktoré vystavuje zdroje na komunikáciu webovému frontendu. Zdroje tohto API sa veľmi podobajú tomu, čo bolo pôvodne navrhnuté, no v niektorých prípadoch bolo potrebné implementovať dodatočné zdroje pre niektoré operácie. Navyše keďže je potrebné aby aplikácia podporovala komunikáciu pomocou TCP protokolu bolo implementované rozhranie na komunikáciu pomocou tohto protokolu.

3.1.5.1 REST

Všetky triedy z tohto balíku, ktoré definujú zdroje API, dedia od spoločného rozhrania, podobne ako tomu je v ostatných vrstvách aplikácie. Avšak podobne ako v prípade tried z balíku Service, tu tiež okrem tohto rozhrania, boli v oddelenej triede, ktorá je potomkom tohto rozhrania implementované obecné metódy, ktoré je možné pri implementácií konkrétnych zdrojov použiť. Rovnako je tu implementovaná aj logika pre autentifikáciu požiadavkov, pomocou dát v ich hlavičke. Avšak v prípade, že vývojár nepotrebuje túto triedu využiť, môže namiesto nej jednoducho implementovať základné rozhranie. Nižšie je zobrazená implementácia zdroju /quizzes. Implementovaná je tu len 1 metóda, nakoľko všetky ostatné implementuje trieda IControllerImpl, ktorú som popisoval vyššie.

```
@RestController
@Visibility
@RequestMapping("/quizzes")
class QuizController(service : QuizService, userService: UserService,
                     private val quizService: QuizService)
    : IControllerImpl<Quiz, QuizFindDTO, QuizGetDTO, QuizCreateDTO,
                      QuizUpdateDTO>(service, userService){

    @GetMapping("/module/{moduleId}")
    fun getByModuleId(@PathVariable moduleId : Int,
                      request: HttpServletRequest)
        = authenticate(request, VisibilitySettings.LOGGED) {
            user -> quizService.getQuizByModuleId(moduleId, user)
        }
}
```

Výpis kódu 3.5 Implementácia REST API zdroju /quizzes

3.1.5.2 Websocket controller

Pre komunikáciu pomocou protokolu TCP je možne použiť technológiu Websocket, ktorá umožňuje vytvorenie komunikačného kanálu na vybranej URI. Framework Spring na toto poskytuje rozhranie WebSocketMessageBrokerConfigurer, pomocou ktorého je možné konfigurovať tento komunikačný kanál. Na URI, ktorá je definovaná pri konfigurácií sa užívatelia následne môžu pripojiť a počúvať správy, ktoré server na túto URI pošle. Všetkým užívateľom bude táto správa tým pádom zverejnená naraz. Nižšie je možné vidieť ukážku konfigurácie tohto komunikačného kanálu.

```
@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig : WebSocketMessageBrokerConfigurer {

    override fun registerStompEndpoints(registry: StompEndpointRegistry) {
        registry.addEndpoint("/broker")
            .setAllowedOriginPatterns("*")
            .withSockJS()
    }

}
```

Výpis kódu 3.6 Implementácia WebSocketMessageBrokerConfigurer

3.2 Frontend

Frontend je v aktuálnom stave projektu dostupný ako webové rozhranie na stránke Trainer. Kvízový a testový modul užívatelia portálu môžu vidieť (ak nejaký z týchto modulov vyučujúci pripravil) po tom, ako otvoria detail niektorej z lekcií a daný modul vyberú kliknutím na jeho názov v postrannom menu. Používanie tohto portálu v svojej diplomovej práci dôkladnejšie popisuje Bc. Ondřej Wrzieconko. Celá aplikácia je rozdelená do komponent. Kvízový a testový modul sa skladajú z niekoľkých komponent, v závislosti na tom, koľko rôznych rozhraní pri používaní týchto modulov je potrebných.

3.2.1 Service

Balíček Service obsahuje niekoľko súborov so zdrojovými kódmi, ktoré implementujú rôzne požiadavky, ktoré sú pri behu aplikácie posielané na server pomocou HTTP protokolu. Javascript poskytuje niekoľko spôsobov, ako vytvárať HTTP požiadavky, kolega Wrzieconko sa rozhodol použiť knižnicu axios pre jeho časť systému a keďže som s tou knižnicou aj ja mal skúsenosti použil som ju taktiež. Rôzne požiadavky sú rozdelené do súborov podľa toho, ku ktorej časti aplikácie patria. Všetky požiadavky, ktoré sa týkajú kvízov sa nachádzajú v súbore quizApi.js, no aj kvízový aj testový modul na niektorých miestach využívajú aj požiadavky, ktoré nie sú obsiahnuté v tomto súbore ale v inom. Nižšie je možno vidieť ukážku definícií týchto HTTP požiadavkov. Ako je zrejmé jedná sa o set asynchrónnych funkcií v Javascripte, v ktorých je odoslaný 1 HTTP požiadavok na URI špecifikovanú v tele funkcie, ktorú poskytuje knižnica axios. Funkcie ktoré táto knižnica poskytuje reprezentujú rôzne typy HTTP požiadavkov, ich používanie je teda veľmi jednoduché na porozumenie a implementácia je rovnako veľmi rýchla a jednoduchá, ako je vidno aj z ukážky kódu nižšie.

```
import axios from 'axios'

export default {

    async createQuiz(quiz) {
        const response = await axios.post("/quizzes", quiz)
        return response.data
    },

    async editQuiz(id, quiz) {
        const response = await axios.patch(`/quizzes/${id}`, quiz)
        return response.data
    },

    async listQuizzes() {
        const response = await axios.get("/quizzes")
        return response.data
    },

    async quizDetail(id) {
        const response = await axios.get(`/quizzes/${id}`)
        return response.data
    },

    async quizDetailByModule(moduleId) {
        const response = await axios.get(`/quizzes/module/${moduleId}`)
        return response.data
    }
}
```

Výpis kódu 3.7 Ukážka HTTP požiadavkov

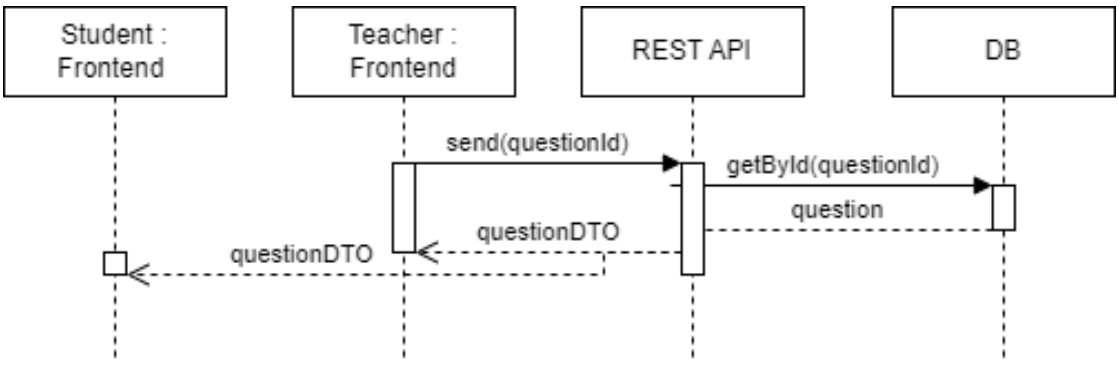
3.2.2 Komunikácia pomocou websocketov

Komunikácia pomocou technológie websocket prebieha exkluzívne v kvízovom module, nakoľkona len v tomto module je potrebné, aby bol kvíz synchronizovaný medzi všetkými pripojenými užívateľmi. Navyše je pomocou tejto technológie zabezpečené, že akcia učiteľa vyvolá reakciu u všetkých účastníkov.

```
import SockJS from "sockjs-client";
import Stomp from "webstomp-client";

const stompClient = ref(null)
const connected = ref(false)
let socket = new SockJS(axios.defaults.baseURL + "/broker");

function connect() {
  stompClient.value = Stomp.over(socket);
  stompClient.value.debug = () => {};   //disables debug messages in console
  stompClient.value.connect(
    {},
    () => {
        connected.value = true

        stompClient.value.subscribe("/messages/questions", q => {
          questionData.value = JSON.parse(q.body)
        });
      },
      error => {
        console.log(error);
        connected.value = false;
      }
  );
}
```

Výpis kódu 3.8 Komunikácia pomocou websocketov

Vo funkcií connect sa deje niekoľko vecí. Najprv je inicializovaná premenná stompClient, ktorá reprezentuje rozhranie/klienta, pomocou ktorého je možné komunikovať pomocou TCP, po vytvorení spojenia. Toto spojenie je vytvorené pomocou tohto rozhrania, konkrétne metódy connect. Následne je možné využiť metódy subscribe. Po zavolaní tejto metódy, za predpokladu, že nestane nejaká chyba začne tento klient počuvať na špecifikovanej URI. Na túto URI bude server posielať správy ako reťazce vo formáte JSON, kedykoľvek na to príde požiadavok - v našom prípade to nastane keď sa učiteľ rozhodne prejsť na ďalšiu otázku. Po prijatí týchto dát klientom sú pomocou metódy JSON.parse() textové reťazce dekódované a uložené ako JSON objekt. Proces komunikácie vyzerá teda približne takto:



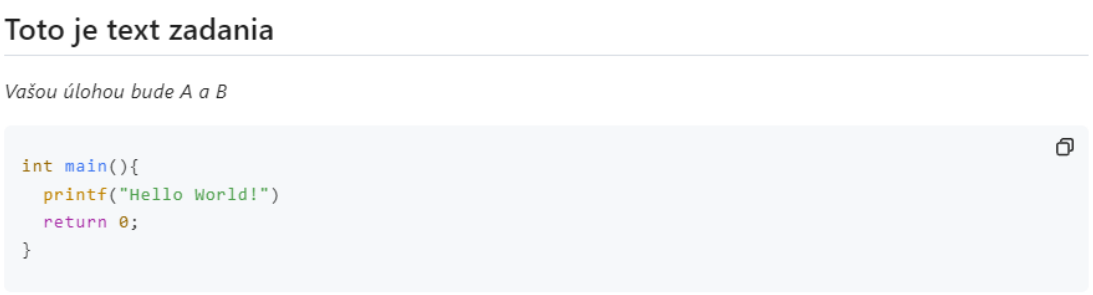
Obr. 3.1 Ukážka komunikácie prijímania otázky

3.2.3 Renderovanie markdownu

Pre renderovanie markdownu som sa rozhodol použiť javascriptovú knižnicu md-editor-v3. Táto knižnica umožňuje jednoduché renderovanie markdownu pomocou komponentu MdPreview. Navyše taktiež obsahuje komponentu MdEditor, ktorá slúži ako textové pole, do ktorého je možné písať text v markdowne. Kolega Bc. Ondřej Wrzieconko, sa navyše tento editor rozšíril v jeho vlastnom komponente, ktorý nazval TextEditor o MdPreview. Je teda možné pri tvorbe textu v markdowne sledovať ako bude tento text vyzerať pri používaní. Nižšie je možné vidieť ukážky kódov a UI implementovaného pomocou tejto komponentov z tejto knižnice.

```
<MdPreview language="en-US" preview-theme="github"
            :theme="userStore.darkMode ? 'dark' : 'light'"
            :no-iconfont="true" :read-only="true"
            class="px-4 pb-2 pt-3 text-left" style="width: 85%"
            v-model="options[index]"/>
```

Výpis kódu 3.9 Príklad použitia komponentu MdPreview



Obr. 3.2 Ukážka renderovania textu s MdPreview

Komponent MdPreview som pri implementácií využil prakticky kdekoľvek je potrebné zobraziť text v markdowne, napríklad pri zobrazení otázky je pomocou tohto komponentu zobrazené zadanie spolu so všetkými možnosťami.

```
<TextEditor :placeholder="t('$vuetify.quiz_module.option')"
            style="height: 200px; width: 80%"
            editable="true" v-model="option">
</TextEditor>
```

Výpis kódu 3.10 Príklad použitia komponentu TextEditor

Komponent TextEditor je, ako som už v úvode tejto sekcie spomínal, komponent, ktorý v sebe spája MdPreview a MdEditor do jedného komponentu. Vďaka tomu je možné jednoducho písať text v markdowne do textového pola naľavo a napravo je automaticky zobrazené ako bude tento text vyzerať kdekoľvek bude zobrazený pomocou komponentu MdPreview.

3.2.4 Príklad implementovaného komponentu

Ako som v úvode sekcie spomínal, kvízový a testový modul, rovnako ako celá webová aplikácia, sú zložené z niekoľkých komponentov resp. rozhraní. Každý VueJs komponent je zložený z 3 sekcií - script, template a style. Sekcia script obsahuje kód v Javascripte s implementovanými všetkými potrebnými funkciami pre správne fungovanie webového rozhrania. Samotný vzhľad tohto

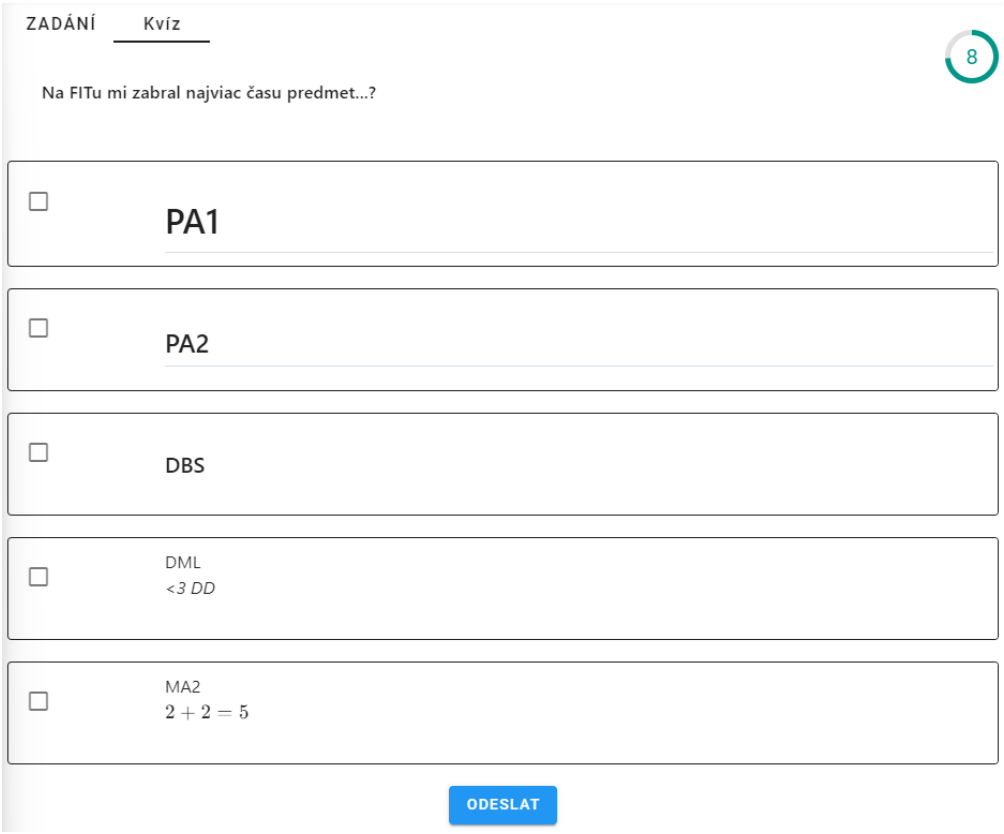
rozhrania je implementovaný v sekcií template. Sekcia style je určená na definovanie špeciálnych CSS štýlov, ktorými je možné upravovať vzhľad UI. Nižšie je ukážka jednoduchého VueJS komponentu napísaného pomocou Composition API.

```
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">You clicked me {{ count }} times.</button>
</template>

<style scoped>
</style>
```

Výpis kódu 3.11 Jednoduchý VueJS komponent [37]

Rozhranie tohto komponentu obsahuje jedno tlačidlo obsahujúce text „You clicked me *X* times“. Namiesto *X* je zobrazené číslo, ktoré sa vždy zdvihne o 1, po tom ako na toto tlačidlo klikne užívateľ. (Pôvodne je teda na tomto mieste zobrazená 0).

Práve z takýchto komponentov, aj keď oveľa komplexnejších, je zložený kvízový a testový modul. Nižšie je možné vidieť komponent, ktorý má na starosti pravdepodobne najpodstatnejšie rozhranie v oboch moduloch - rozhranie pre zobrazenie otázky na ktoré je možno odpovedať.



Obr. 3.3 Rozhranie pre odpovedanie na otázku

Toto rozhranie má značne zložitejšiu implementáciu, z ktorej najpodstatnejšie časti uvádzam

v nasledujúcej ukážke kódu.

```
<script setup>

const submitAnswer = () => {
    if (answers.value.length === 0)
        return
    answers.value = [...new Set(answers.value)]
    const answer = {
        lesson: lessonId,
        quizroom: quizroomData.value.id,
        student: userStore.user.id,
        question: questionData.value.id,
        data: JSON.stringify(answers.value)
    }

    api.createAnswer(answer)
        .catch((err) => {
            console.log(err)
            error.value = `Chyba řpi čnaítání: ${err.code}`
        })
    answersSubmitted.value = true
}
</script>

<template>
    <v-card-title style="top: 200px">
        <MdPreview language="en-US" preview-theme="github"
            :theme="userStore.darkMode ? 'dark' : 'light'"
            :no-iconfont="true" :read-only="true"
            class="px-4 pb-2 text-left"
            :toolbars="[]" v-model="questionText"/>
    </v-card-title>

    <v-card
        v-for="(option, index) in options"
        :key="option"
        :style="[userStore.darkMode ? {'background-color': 'black'}
            : {'background-color': 'white'}]"
        class="mt-5"
        @click="selectOrUnselectOption(options[index])"
        :variant="answers.indexOf(options[index]) > -1
            ? 'tonal' : 'outlined'">
      <v-row>
        <v-checkbox class="ma-5" style="width: 10px" :value="options[index]"
            v-model="answers" :disabled="answersSubmitted"></v-checkbox>
        <MdPreview language="en-US" preview-theme="github"
            :theme="userStore.darkMode ? 'dark' : 'light'"
            :no-iconfont="true" :read-only="true"
            class="px-4 pb-2 pt-3 text-left"
            style="width: 85%" v-model="options[index]"/>
      </v-row>
    </v-card>

</template>
```

Výpis kódu 3.12 Rozhranie pre odpovedanie na otázku

Na implementáciu rozhrania systému Trainer sme spoločne s kolegom Wrzieconkom použili framework Vuetify, ktorý poskytuje veľmi veľké množstvo UI komponent. V tejto ukážke je konkrétne možno vidieť použité komponenty v-checkbox, v-card a v-row.

V sekcií script sa v reálnej implementácií nachádza viac funkcií, no zďaleka najpodstatnejšia je funkcia na odosielanie odpovedí - submitAnswer(). Táto funkcia je volaná v 2 prípadoch - keď študent klikne na tlačidlo odoslať a keď študentovi vyprší čas na odpovedanie. Najprv vo funkcií dôjde ku kontrole vstupných dát - študentových odpovedí. V prípade, že študent nezvolil žiadnu z možností sa táto funkcia ukončí. V opačnom prípade sa odstránia duplicity vybraných možností, ktoré by síce nemali vznikať, ale je dobré tam túto ochranu mať. Následne sa vytvorí objekt, s potrebnými dátami, ktorý je pomocou asynchrónnej funkcie definovanej v balíčku Service odoslaný na server, kde je jeho odpoveď spracovaná a vyhodnotená.

V bloku template môžeme vidieť dve hlavné časti. Navrchu stránky je pomocou MdPreview renderované zadanie otázky. Pod týmto zadaním je potom niekoľko možností. Tieto sú zobrazené v komponentoch v-card, ktoré obsahujú checkbox na vyberanie odpovedí a opäť komponentu MdPreview, ktorá renderuje text danej možnosti. Toto je zabezpečené pomocou direktívy v-for, ktorá vie iterovať cez všetky prvky poľa a podla týchto iterácií vytvorí požadovaný počet komponent, pri ktorých je táto direktíva špecifikovaná - v našom prípade je teda vytvorených n komponent v-card, kde n je rovné počtu možností. Okrem toho tieto karty menia svoj vzhľad podla toho, či ich študent zvolil, alebo nie.

3.2.5 Nginx

Nginx [41] je reverse proxy server, využívaný najmä za účelom zvýšenia bezpečnosti, load balancingu a kešovania dát, čo môže mať za následok lepší výkon. Pre projekt Trainer bol nginx konfigurovaný pre komunikáciu pomocou HTTP protokolu, čo implementoval Bc. Ondřej Wrzieconko a tak isto pre komunikáciu pomocou Webosocketov cez protokol TCP - táto časť bola konfigurovaná mnou. Nižšie je možné vidieť ukážku tejto konfigurácie.

```
location /api/broker/ {
        proxy_pass  http://backend:8080/broker/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $host;
}
```

Výpis kódu 3.13 Ukážka konfigurácie proxy

Táto konfigurácia nginx serveru docieli to, že kedykoľvek bude na URI /api/broker odoslaný požiadavok, tento server to odchytí a presmeruje tento požiadavok na adresu http://backend:8080 /broker/. Toto presmerovanie je potrebné, nakoľko frontend aj backend tejto aplikácie beží na rovnakom porte, no backendový server má na konci URI navyše reťazec /api. V skutočnosti na servery žiaden REST API zdroj ani Websocketový „broker“ nepočúva. Tento „broker“ počúva práve na URI /broker. Navyše pri tomto presmerovaní nginx nastaví do hlavičky HTTP požiadavku, informáciu o tom, že chce upgradovať spojenie na komunikáciu pomocou websocketov - konkrétne sa jedná o:

- proxy_set_header Upgrade $http_upgrade;
- proxy_set_header Connection $connection_upgrade;

Broker na servery tento požiadavok spracuje a v prípade úspechu odošle odpoveď s HTTP statusom 101 Switching Protocols, ktorá indikuje úspešné upgradovanie pripojenia.

Kapitola 4
Testovanie

V tejto kapitole popíšem testovaciu fázu vývoja aplikácie. Túto kapitolu som sa rozhodol rozdeliť na 2 sekcie - automatizované testovanie a užívateľské testovanie. Pri automatizovanom testovaní popisujem rôzne typy testov a ich samotnú implementáciu. Pri užívateľskom testovaní vysvetľujem rôzne spôsoby, akým toto testovanie môže prebehnúť a taktiež aplikácie, ktoré tento druh testovania môžu uľahčiť.

4.1 Automatizované testovanie

Automatizované testy sú spôsobom, ako zabezpečiť správne fungovanie aplikácie prakticky kedykoľvek, ale špeciálne po tom, ako prebehnú v implementácií zmeny, prípadne celkový refaktor implementácie. Ich hlavnou výhodou oproti užívateľským testom je to, že eliminujú ľudský faktor a vždy skončia tým istým výsledkom ak testujú deterministickú funkcionalitu. Na automatizované testovanie, konkrétne na jednotkové testy, som použil frameworky Kotest [42] a Mockito[43]. Na integračné testy som použil aplikáciu Postman, v ktorej som vytvoril testovaciu sadu požiadavkov, ktoré je možné jednoducho odoslať kliknutím na jedno tlačidlo.

4.1.1 Jednotkové testy

Jednotkové testy sú testy, ktoré testujú funkcionalitu jednej metódy vybranej triedy. Takto je možné (a vhodné) otestovať všetky metódy v každej triede, alebo aspoň tie, pri ktorých to dáva zmysel - nie je napríklad potrebné testovať funkcionalitu getterov [1]. V prípade, že táto metóda je nejakým spôsobom závislá na inej triede - vstupný parameter, lokálna premenná a pod., je možné použiť mock objekt, ktorý poskytuje framework mockito, ktorý túto závislosť nahradí správaním, aké očakávame.

```
class QuizServiceTests (
    @MockBean val repository: QuizRepo,
    @MockBean val userRepo: UserRepository,
    @MockBean val moduleRepo: ModuleRepository,
    service: QuizService
): StringSpec({
    // sem patria samotné testy
})
```

Výpis kódu 4.1 Ukážka mockovania závislostí testovacej triedy

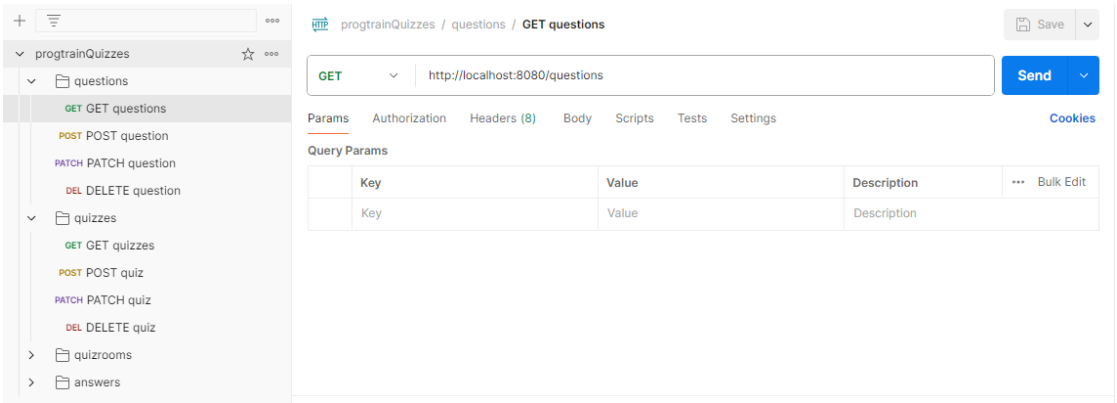
[1] Metódy, ktoré vrátia hodnotu vybraného atribútu. Väčšinou obsahujú len 1 príkaz.

```
val module1 = Module("quiz", ModuleType.QUIZ, "", ModuleDifficulty.EASY,
                     true, false, false, false, 0, "",
                     Timestamp.from(Instant.now()),emptyList(), emptyList(),
                     emptyList(), userDummy, 1)

val quiz1 = Quiz(1,"quiz", 1, emptyList(), emptyList(), module1, 1)
val quiz1DtoG = QuizGetDTO(quiz1.id, quiz1.name, quiz1.numOfQuestions,
                           quiz1.numOfAttempts, emptyList())


"getByModule" {
        given(moduleRepo.existsById(1)).willReturn(true)
        given(moduleRepo.findById(1)).willReturn(Optional.of(module1))
        given(repository.existsByModule(module1)).willReturn(true)
        given(repository.getByModule(module1)).willReturn(quiz1)

        service.getQuizByModuleId(1, userDto) shouldBe quiz1DtoG

        verify(repository).getByModule(module1)
}
```

Výpis kódu 4.2 Ukážka implementácie automatického testu

4.1.2 Integračné testy

Integračné testy testujú celkovú funkcionalitu vybranej časti systému ako celku, teda či používané triedy a metódy medzi sebou interagujú podľa očakávaní a dodávajú konzistentné výsledky. Na tento typ testov som použil aplikáciu Postman, v ktorej som vytvoril testovaciu sadu pre väčšinu požiadavkov, ktoré je možné poslať na REST API tohto systému a týkajú sa nejakým spôsobom kvízového a testového modulu.



Obr. 4.1 Ukážka integračných testov v aplikácií Postman

4.2 Užívateľské testovanie

Užívateľské testy obecne testujú rozhranie systému - v našom prípade je to webová stránka a teda pri testovaní môže hrať rolu veľmi veľa faktorov, ako napr. zariadenie užívateľa, používaný

prehliadač atď. Okrem toho je bežné, že ľudia pri testovaní spravia chybu a test tým pádom skončí nesprávnym výsledkom. Toto automatizované testy úplne eliminujú. Čo však automatizované testy nedokážu, je posúdiť samotné rozhranie a jeho celkovú použiteľnosť. Na toto je dobré, až priamo potrebné, využiť testovanie užívateľmi a to najmä kvôli možnosti spätnej väzby na UI a UX, ktorú automatizované testy nemajú ako poskytnúť. Vo väčšine prípadov je však dobré poskytnúť užívateľom testovacie scenáre, podľa ktorých sa môžu riadiť pri testovaní aplikácie. Toto je špeciálne platné ak títo užívatelia túto aplikáciu nepoznajú. Testovanie prebiehalo priebežne a podieľal som sa na ňom ja, tím študentov predmetu BI-SP1, vedúci práce Ing. Jan Matoušek, kolega Bc. Ondřej Wrzecionko a niektorí ďalší vyučujúci, menovite Otto Šleger a Jan Šuráň, ktorý poskytli naozaj veľké množstvo spätnej väzby. V rámci tohto testovania sme testovali všetky mnou implementované rozhrania s maximálnym dôrazom na ich funkčnosť, no taktiež s dôrazom na ich použiteľnosť z pohľadu UI a UX.

4.2.1 Jira

Jira je aplikácia určená na správu úloh a testovacích scenárov pre vývoj softwaru. Poskytuje široké spektrum prehľadov úloh zadaných vo vybranom projekte vďaka čomu je jednoduchá na používanie ako pre vývojárov a testerov, tak pre manažérov. Plná verzia tejto aplikácie je platená, no existuje verzia zdarma, ktorá ale má značne okliešténú funkcionalitu.

Tvorba testovacích scenárov v Jire je veľmi jednoduchá a spočíva vo vyplnení názvu a testovacích krokov, prípadne ďalších polí, ktoré už sú závislé na tom, v akom tíme užívateľ pracuje. Okrem toho Jira tak isto poskytuje možnosť nahrať testovací scenár zo súborov vo formáte CSV [2]. To znamená, že v prípade preferencie, je možné testovací scenár pripraviť v aplikácií, ktorá podporuje CSV, napríklad Microsoft Excel, a následne tento súbor importovať priamo Do Jiry, ktorá po troche konfigurácie z tohto súboru pripraví testovací scenár.

4.2.2 Gitlab

Gitlab ponúka rozhranie pre správu testovacích scenárov v každom projekte. Toto rozhranie sa nachádza v každom projekte v záložke Build -> Test cases. Na tejto obrazovke je možné vidieť všetky testovacie scenáre, ktoré boli v danom projekte vytvorené. V scenároch je možné vyhľadávať podľa niekoľkých kritérií - napríklad podľa mena, dátumu vytvorenia, autora atď. Tieto scenáre je ďalej možno zaraďovať do skupín pomocou labelov[3], podľa ktorých je tak isto možné tieto scenáre vyhľadávať. Tvorba scenáru na gitlabe je veľmi jednoduchá, podľa môjho názoru jednoduchšia ako v Jira. Užívateľ jednoducho založí nový testovací scenár a v následne zobrazených poliach vyplní názov a text daného scenáru, kde môže popísať kroky, alebo daný scenár popísať inak.

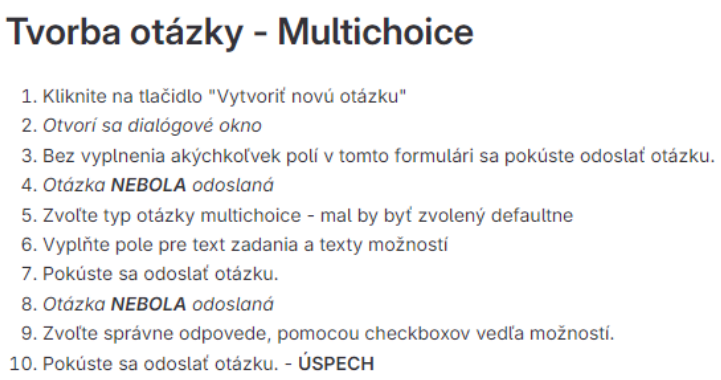
4.2.3 Použitá aplikácia

Pre správu testovacích scenárov som zvolil Gitlab issues. Hlavným dôvodom je to, že je Gitlab je zadarmo a používam ho už niekoľko rokov. Navyše tento systém je určený pre používanie na FIT ČVUT, takže potreba mať tieto testovacie scenáre prístupné z externého zdroja sa nejaví ako potrebné. Ďalším dôvodom je, že počet testovacích scenárov je aktuálne relatívne malý, takže ich správa je jednoduchá. Avšak v prípade, že sa časom počet testovacích scenárov radikálne zväčší, nemal by byť problém prejsť na iný systém pre správu týchto scenárov ak tento systém už nebude dostatočný.

[2] Comma-Separated Values
[3] štítkov

4.2.4 Testovací scenár

Testovací scenár je jednoduchý popis, podľa ktorého dokáže vybraná osoba otestovať funkcionalitu, ktorej sa tento scenár týka. Nižšie je ukážka testovacieho scenáru na otestovanie vytvárania otázky.



Obr. 4.2 Ukážka testovacieho scenáru

4.2.5 Spätná väzba

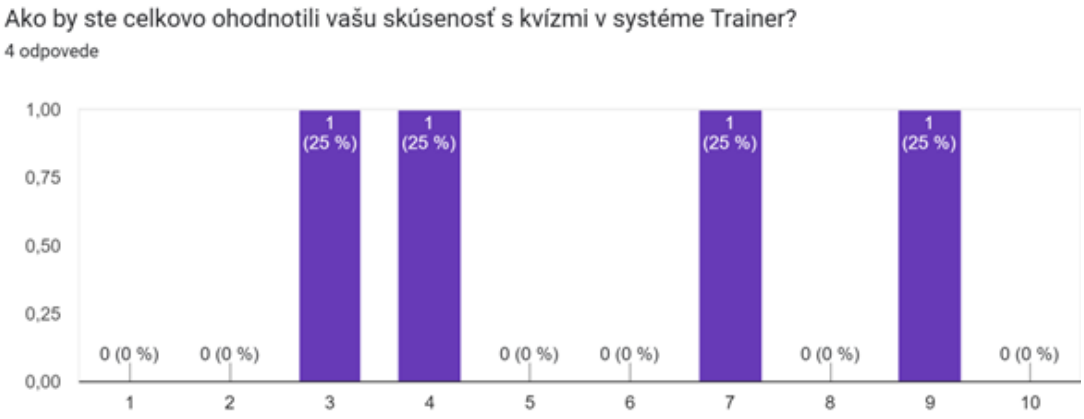
Keďže kvízový modul je používaný na cvičeniach vybraných predmetov na FIT ČVUT už 2. semester, mal som možnosť priebežne zbierať spätnú väzbu. Tento zber prebieha pomocou Gitlab Issues na stránke https://gitlab.fit.cvut.cz/trainer-fit/feedback, kde užívatelia môžu založiť novú issue v prípade, že nájdu bug [4], alebo majú akúkoľvek pripomienku, týkajúcu sa používania systému. Okrem toho som v priebehu letného semestru vytvoril dotazník spokojnosti s kvízovým modulom pre študentov a učiteľov. Dotazník spokojnosti pre testový modul som zatiaľ považoval za bezpredmetný nakoľko tento modul aktuálne je vyvinutý, ale ešte nie je nasadený na server, kde beží systém Trainer, no keď sa začne používať určite bude dobré takýto dotazník vytvoriť. Z tohto dôvodu som o testovanie a spätnú väzbu požiadal tím študentov z predmetu BI-SP1 a ich spätnú väzbu som spracoval a zohľadnil v nasledujúcej kapitole (Diskusia).

4.2.5.1 Učiteľský dotazník

Učiteľský dotazník obsahoval oproti tomu študentskému otázky, ktoré sa týkali aj rozhraní na tvorbu otázok a kvízov, spolu s otázkami, ktoré sa týkali spokojnosti s rozhraniami samotného kvízu. Vzorka je rádovo menšia oproti študentskému dotazníku, nakoľko učiteľov je rádovo menej ako študentov. Odpovede na otázky boli buď ako otvorený text, alebo ako výber na škále 1-10, kde 1 je najhorší možný výsledok a 10 najlepší. Dotazník dopadol prevažne pozitívne, no jasne z neho vzišlo, že UI a UX tohto modulu - špeciálne UI rozhrania pre tvorbu otázok a kvízov bude potrebovať veľmi veľa práce, aby bolo jasné a príjemné na používanie. Tieto rozhrania dostali v dotazníku priemerné hodnotenie 5/10, čo by som ja osobne pripísal, mojím, prakticky neexistujúcim, skúsenostiam s návrhom užívateľských rozhraní a takisto nemalému počtu bugov, na ktoré sa našťastie prišlo veľmi rýchlo. Čo sa týka rozhraní, ktoré sú používané pri hraní kvízov, tu sú výsledky dotazníku značne lepšie. V závislosti od konkrétneho rozhrania (celkový počet = 5), boli výsledky dotazníku v priemere medzi 8,5 - 9,5. Na otázku ohľadom celkovej spokojnosti v priemere učitelia odpovedali hodnotením 5.7, čo pripisujem najmä rozhraniam pre tvorbu otázok a kvízov.

[4] chyba, alebo anomália v systéme

Okrem hodnotení spokojnosti respondenti tak isto poskytli návrhy na vylepšenia a rozšírenia tohto modulu, z ktorých niektoré popisujem v závere tejto práce.



Obr. 4.3 Graf celkovej spokojnosti učiteľov s kvízovým modulom

4.2.5.2 Študentský dotazník

Študentský dotazník obsahoval otázky, týkajúce sa spokojnosti s rozhraniami, s ktorými študenti pri používaní kvízového modulu prichádzajú do kontaktu a tak isto priestor pre návrhy na časti tohto modulu, ktoré je priestor vylepšiť. Odpovede na otázky boli buď ako otvorený text, alebo ako výber na škále 1-10, kde 1 je najhorší možný výsledok a 10 najlepší. Celková spokojnosť študentov s týmto modulom je v priemere medzi 6.9 - 7. Najčastejším dôvodom horších hodnotení bolo rozhranie pre správne riešenie otázky, nakoľko niektorím študentom neprišlo na prvý pohľad intuitívne. Dojem z tohto modulu tak isto študentom pokazil bug, na ktorý sa v prišlo približne v polovičke letného semestru, pri ktorom nefungovalo správne vyhodnocovanie odpovedí. Podstata tohto bugu bola v priebehu niekoľkých dní identifikovaná a bug opravený.

Obr. 4.4 Graf celkovej spokojnosti študentov s kvízovým modulom

Kapitola 5
Diskusia

V tejto kapitole sa budem stručne venovať spokojnosti užívateľov s výsledkom praktickej časti tejto práce, rovnako ako niektorým možným budúcim rozšíreniam týchto dvoch modulov. V predchádzajúcej kapitole som stručne popísal výsledky dotazníkov spokojnosti, ktorých súčasťou boli otázky, v ktorých som sa študentov aj učiteľov pýtal na ich nápady na nové funkcionality týchto modulov, prípadne, ktoré z existujúcich funkcionalít by radi videli vylepšené.

5.1 Potrebné vylepšenia

Nižšie popisujem časti modulov (najmä kvízového, keďže narozdiel od testového, tento modul je používaný už niekoľko mesiacov), ktoré potrebujú značné vylepšenia, aby bolo zaistené jednoduchšie a intuitívnejšie používanie týchto modulov. Taktiež sa v krátkosti budem venovať niektorým technologickým aspektom, ktoré priamo s UI ani s UX nemajú moc spoločné. Tieto poznatky pre vylepšenia vznikli z dotazníkov spokojnosti a taktiež zo sebareflexie, nakoľko v dobe kedy píšem túto prácu mám značne väčšiu skúsenosť s vývojom UI, ako keď som tieto moduly pôvodne implementoval.

5.1.1 UI pre tvorbu otázok

Rozhranie pre tvorbu otázok, už prešlo niekoľkými iteráciami. V aktuálnej forme by toto rozhranie potrebovalo vylepšenie v ohľade funkčnom - občasný výskyt bugov. Taktiež z dotazníku spokojnosti učiteľov vyšlo najavo, že by bolo dobré implementovať jednoduché menenie poradia možností.

5.1.2 UI pre tvorbu kvízov

Podobne ako rozhranie pre tvorbu otázok, toto rozhranie taktiež prešlo niekoľkými iteráciami. Zo spätnej väzby od učiteľov však vyšlo najavo, že jeho používanie nie je natoľko intuitívne ako som si myslel pri jeho pôvodnej implementácií. Okrem toho by bolo dobré identifikovať a odstrániť prípadné bugy, ktorých bolo v predchádzajúcich verziách niekoľko, no všetky známe bugy by v tomto čase mali byť opravené.

5.1.3 Zobrazenie otázok v zozname pri tvorbe modulu

V aktuálnej verzií projektu je každá otázka reprezentovaná vlastnou kartou, na ktorej je text danej otázky a možnosti odpovede. Tieto texty sú však zobrazené ako čistý text (nie markdown).

To spôsobuje občasnú nečitateľnosť obsahu otázky obzvlášť v prípadoch kedy táto otázka obsahuje obrázok. Bolo by preto dobré docieliť aby tieto texty boli zobrazované tak ako majú byť a nie v čistom texte.

5.1.4 Zobrazenie riešenia otázky

Aktuálne je každá možnosť, ktorá je správna zobrazená zelenou farbou a vedľa nej ikona ✓, nehľadiac na to, či túto možnosť študent zvolil, alebo nie. Taktiež sú v tomto prehľade indikované možnosti, ktoré študent označil a v prípade, že vybraná možnosť bola označená nesprávne je táto možnosť indikovaná červenou farbou. Od niekoľkých študentov som v dotazníku spokojnosti dostal spätnú väzbu, že tento prehľad nie je jednoznačne pochopiteľný. Bolo by preto vhodné analyzovať podobné aplikácie/programy/modely a zistiť, ako sa v nich zobrazujú podobné prehľady a toto následne implementovať do systému Trainer.

5.1.5 Kvalita frontendového kódu

Tento požiadavok je jediný, ktorý vznikol len ako súčasť sebareflexie, nakoľko okrem niekoľkých kolegov, nemá k zdrojovým kódom prístup takmer nikto. Keď som začínal s vývojom rozhrania pre kvízový modul bol som prakticky úplným nováčikom čo sa týka používania frameworku VueJS. Tomu odpovedá kvalita zdrojového kódu kvízového modulu. Nakoľko testový modul bol vyvíjaný niekoľko mesiacov po kvízovom kvalita kódu je tu značne lepšia, aj keď vylepšenia by sa určite dali spraviť.

5.2 Možné rozšírenia

5.2.1 Priradenie kategórie otázok testovému modulu

Pri tvorbe testového modulu nebude nutné priraďovať postupne všetky otázky, ale bude stačiť k tomuto modulu priradiť kategóriu otázok a z otázok v nej budú následne testy generované. Prípadne bude možné použiť jednu z možností:

1. Priradenie kategórie/kategórií,
2. priradenie rôznych otázok (aktuálny stav projektu),
3. kombinácia týchto možností.

5.2.2 Náhľad otázky pri tvorbe modulu

Pri tvorbe kvízového aj testového modulu sa aktuálne zobrazuje zoznam otázok v danom module. Tieto otázky je možné upravovať, alebo z kvízu odstrániť, no nie je možné ich nijako zobraziť bez toho, aby musel učiteľ danú otázku začať upravovať. Bolo by preto vhodné dorobiť tlačidlo, ktoré zobrazí vybranú otázku v read-only móde. [1]

5.2.3 Vyhľadanie konkrétneho pokusu kvízu/testu

Tento nápad by si vyžadoval značnú analýzu a návrh, jednalo by sa však o jednoduché vyhľadávanie/filter v náhľade študentských riešení. Ponúka sa možnosť vyhľadávať podľa otázky, počtu nesprávnych/správnych odpovedí a pod. Toto by bolo využiteľné najmä pri náhľade odpovedí z testového modulu, nakoľko na kvízový modul väčšinou študenti majú len jeden pokus, no testový modul si môžu spustiť koľko krát chcú.

[1] mód bez možnosti úprav

5.2.4 Možnosť pridať komentár k študentovej odpovedi na otázku

Pri náhľade študentských odpovedí aktuálne chýba možnosť tejto spätnej väzby – učiteľ teda musí študentovi napísať správu, alebo si ho vytypovať na cvičení. Tento nápad by toto eliminoval tým, že by pri každej študentovej odpovedi na otázku mal možnosť napísať krátku správu.

5.2.5 Personalizácia vzhľadu modulov

Tento nápad je veľmi obecný a ktokoľvek by ho chcel realizovať môže prísť s niečím iným. Jednou z mojich ideí bol výber emotikonu v kvízovej miestnosti, ktorí sa zobrazí učiteľovi v čakárni, ak má zapnutý anonymný mód. To by trochu pokazilo aktuálnu logiku, kde sa náhodne vyberá z niekoľkých fixne definovaných emotikonov, preto by bolo treba napr. Konkrétne túto zmenu analyzovať a konzultovať.

5.2.6 Podpora ďalších typov otázok

Aktuálne je na výber len medzi 2 typmi otázok, malo by byť relatívne jednoduché tento výber rozšíriť, čo by dalo učiteľom väčšiu mieru flexibility pri tvorbe otázok a modulov. Najviac sa asi ponúka otvorený typ otázky, pri tomto type otázok však býva problematický proces vyhodnocovania, keďže v niektorých prípadoch, napr. výsledkoch matematických rovníc, môže byť tvarov správnych výsledkov viac. Rovnako môže spôsobiť problémy diakritika, alebo formátovanie odpovede. Všetky tieto faktory bude potrebné vzať do úvahy pri návrhu a implementácií.

5.2.7 Odporúčaný čas na otázku

Pri tvorbe otázky je aktuálne predvolený čas, ktorý budú študenti mať na zodpovedanie danej otázky. Jednou z ideí, ktoré som dostal vrámci spätnej väzby z učiteľského dotazníku bolo vytvoriť mechanizmus, ktorý by automaticky vyplnil toto pole na základe nejakého faktoru, napríklad dĺžky textu danej otázky a odpovedí.

Kapitola 6
Záver

Hlavným cieľom tejto práce bolo vytvorenie kvízového a testového modulu a ich integrácia do systému Trainer.

Výsledkom tejto práce sú dva plne funkčné moduly, ktoré sú v tomto čase už používané v systéme Trainer, učiteľmi aj študentmi. Tieto moduly boli vyvinuté v 2 častiach – server ktorý vystavuje REST API, ktorý bol vyvinutý v programovacom jazyku Kotlin a frameworku Spring a webová stránka, ktorá bola vyvinutá pomocou HTML, CSS a Javascriptu pomocou frameworku VueJS. Nakoľko sa jednalo o moju prvú prácu takéhoto rozmeru a nebol som skúsený s návrhom UI a UX, niektoré časti týchto modulov by sa určite dali vylepšiť a možno aj rozšíriť o ďalšiu funkcionalitu. Naopak, čo sa týka napr. logiky kvízov a technológií resp. protokolov, ktoré som sa rozhodol použiť, s týmto výberom som veľmi spokojný, najmä kvôli tomu, že pomocou TCP protokolu, sú rozdiely medzi tým kedy sa užívateľom zobrazí nová otázka naozaj minimálne, väčšinou až nebadateľné.

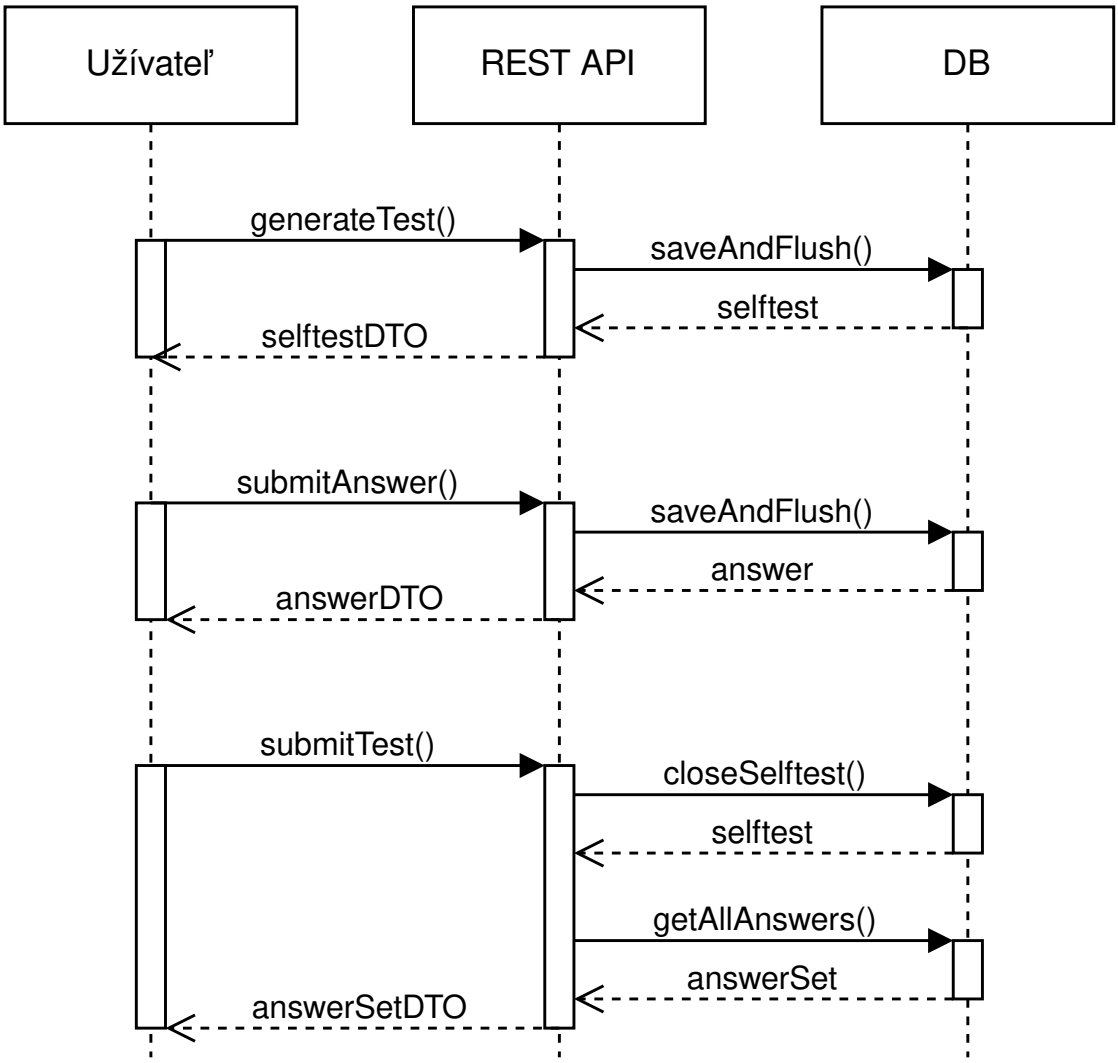
Ako som spomínal vyššie, nie všetko sa počas vývoja podarilo tak, ako som si to pôvodne predstavoval. Navyše postupom času prichádzali postupne myšlienky a nápady na novú funkcionalitu z rôznych zdrojov - ja, učitelia, ale aj aktívny študenti. Niektoré z týchto nápadov boli už implementované, aj popísané v tejto práci, no zďaleka nie všetky a som si istý, že ľudia, ktorí budú na tomto projekte pracovať po mne, určite prídu s ďalšími vlastnými nápadmi.

Všetky zdrojové kódy sa nachádzajú na fakultnom gitlabe a samotná webová aplikácia beží na webovej stránke pod fakultnou doménou. Prístup na tieto stránky je však obmedzený na ľudí s platnou ČVUT identitou, projekt na gitlabe je naviac obmedzený len na ľudí ktorým je špeciálne udelený prístup.

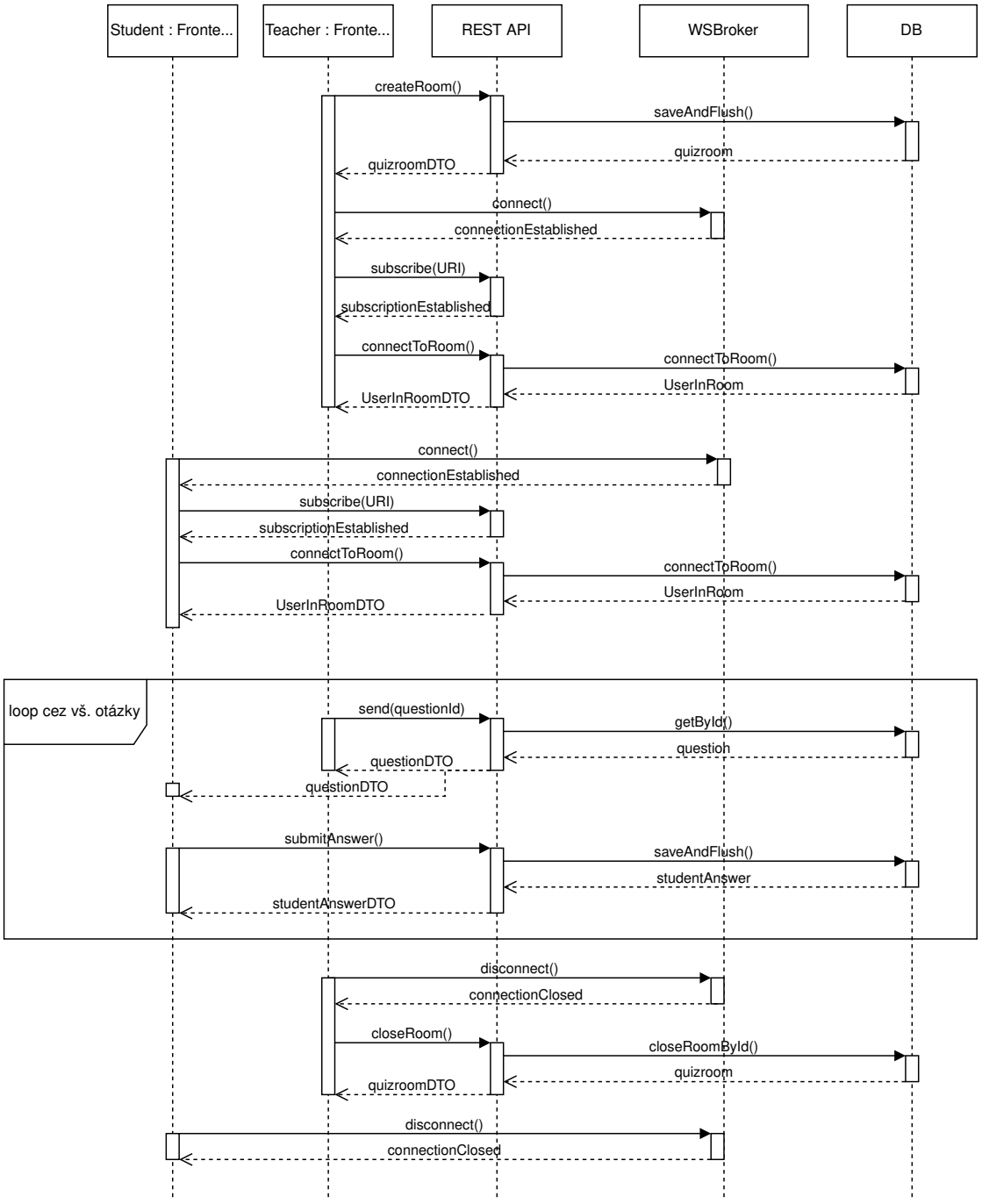
Vzhľadom k veľkosti tohto projektu sa to, že som sa na tomto projekte dohodol s vedúcim ešte v lete 2023, ukázalo ako veľmi dobré rozhodnutie. Dalo mi to dosť času postupne implementované riešenie vylepšovať, pretože prvotné webové rozhranie odpovedalo mojim dovtedy neexistujúcim skúsenostiam s návrhom UI a UX. Som preto veľmi rád, že rozhranie, v jeho aktuálnej podobe, je relatívne prívetivé a jednoduché na používanie, aj keď v ňom je stále určite veľa priestoru na vylepšenia. Ďalej som veľmi rád za to, že som sa naučil vyskúšal vývoj s dovtedy mne neznámymi technológiami ako Kotlin, VueJS, ale hlavne s protokolom TCP, s ktorým som mal dovtedy len teoretické skúsenosti.

Dodatok A

Návrh komunikácie



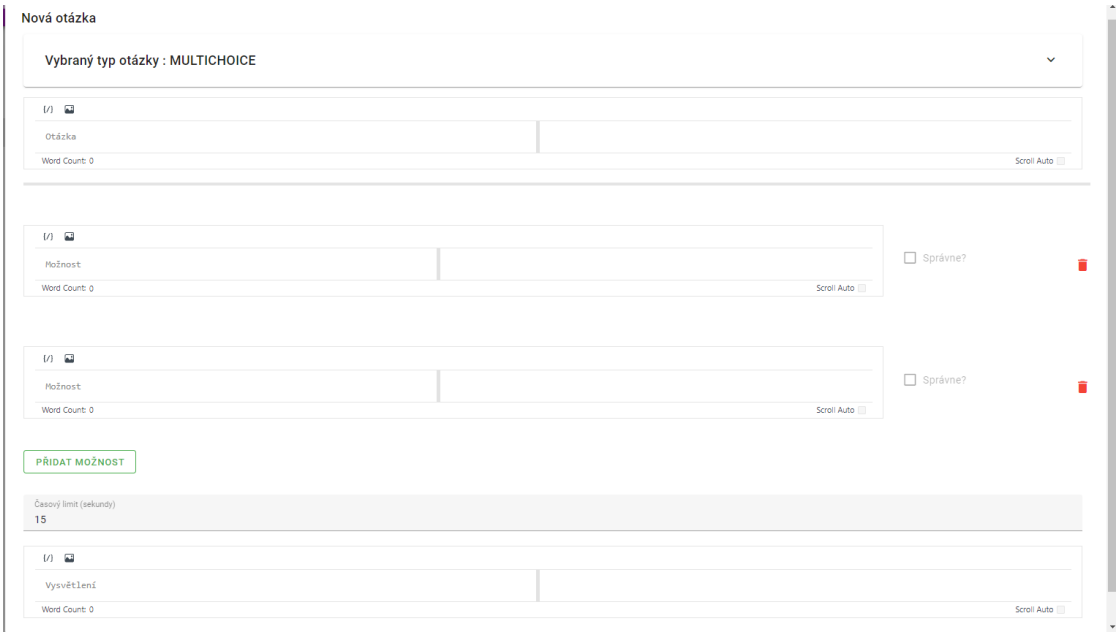
Obr. A.1 Sekvenčný diagram komunikácie počas plnenia testu



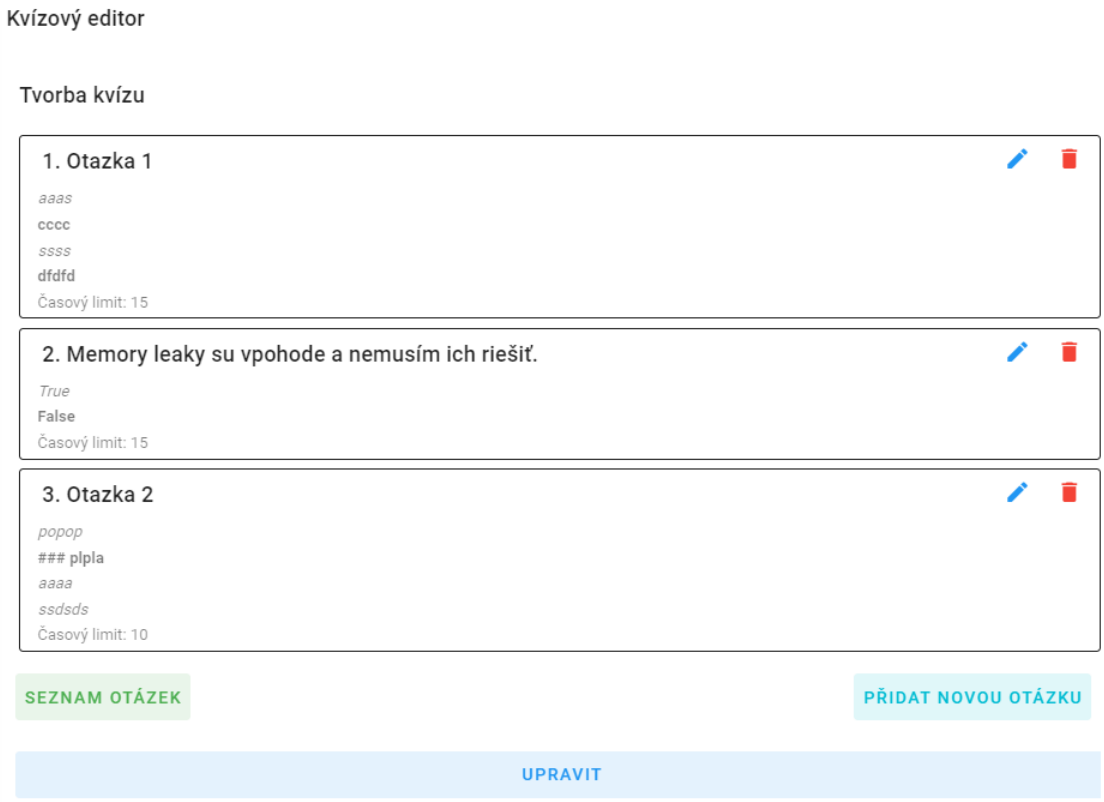
Obr. A.2 Sekvenčný diagram komunikácie počas plnenia kvízu

Dodatok B

Užívateľské rozhrania



Obr. B.1 Ukážka webového rozhrania pre tvorbu otázky



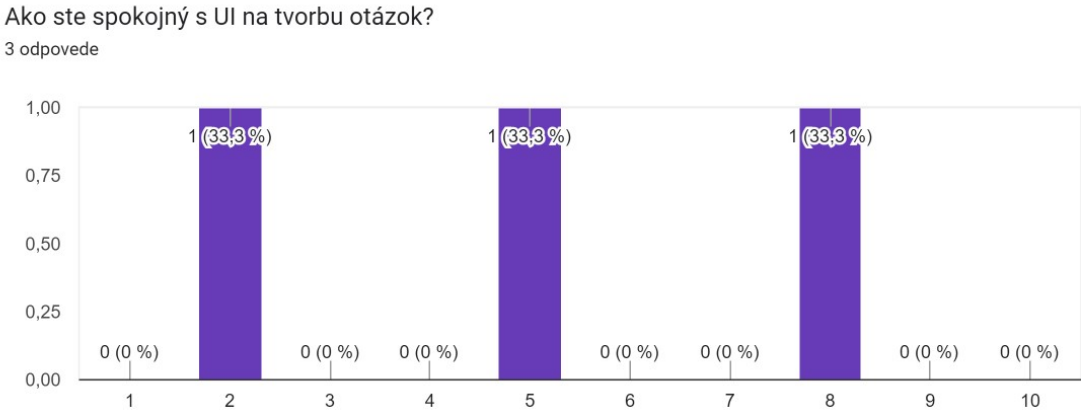
Obr. B.2 Ukážka webového rozhrania pre tvorbu kvízového a testového modulu

Obr. B.3 Ukážka používania TextEditoru

Dodatok C

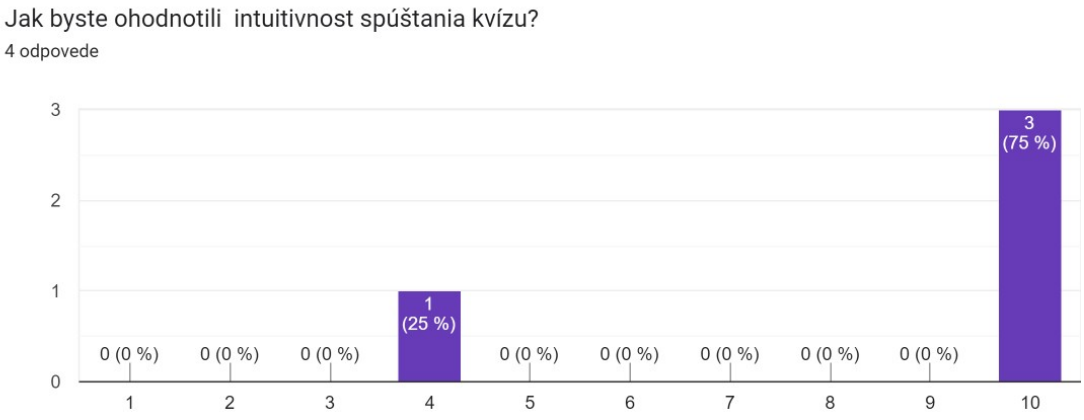
Dotazníky spokojnosti

Obr. C.1 Graf spokojnosti učiteľov s tech. spoľahlivosťou kvízov

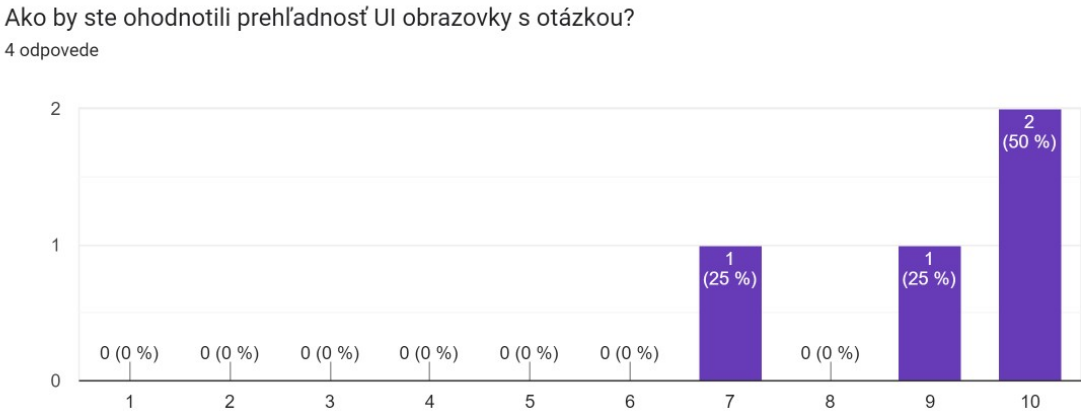


Obr. C.2 Graf spokojnosti učiteľov s UI pre tvorbu otázky

Obr. C.3 Graf spokojnosti učiteľov s UI pre tvorbu kvízu

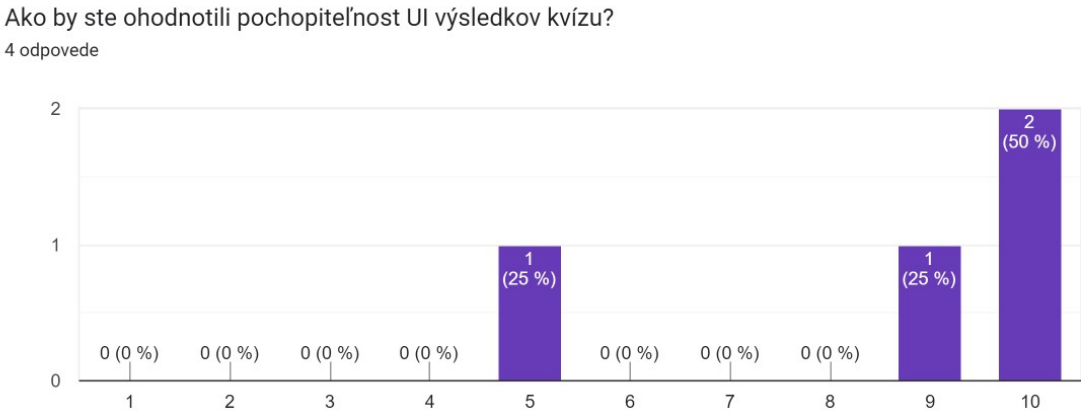


Obr. C.4 Graf spokojnosti učiteľov s UI pre spúšťanie kvízu

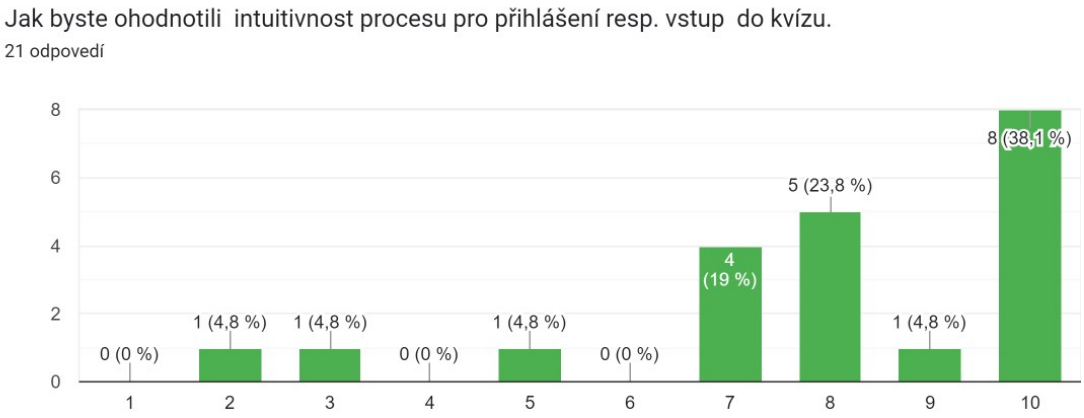


Obr. C.5 Graf spokojnosti učiteľov s UI otázky v kvíze

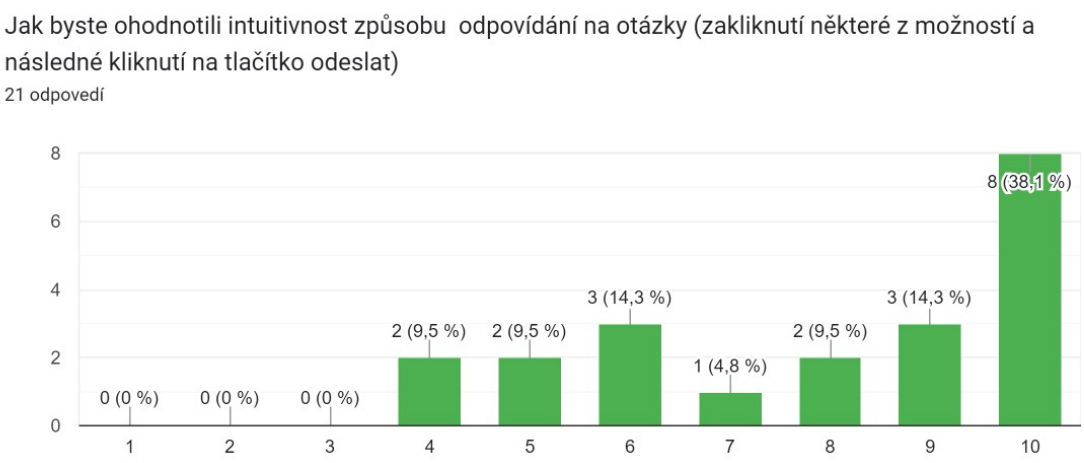
Obr. C.6 Graf spokojnosti učiteľov s UI výsledkov otázky



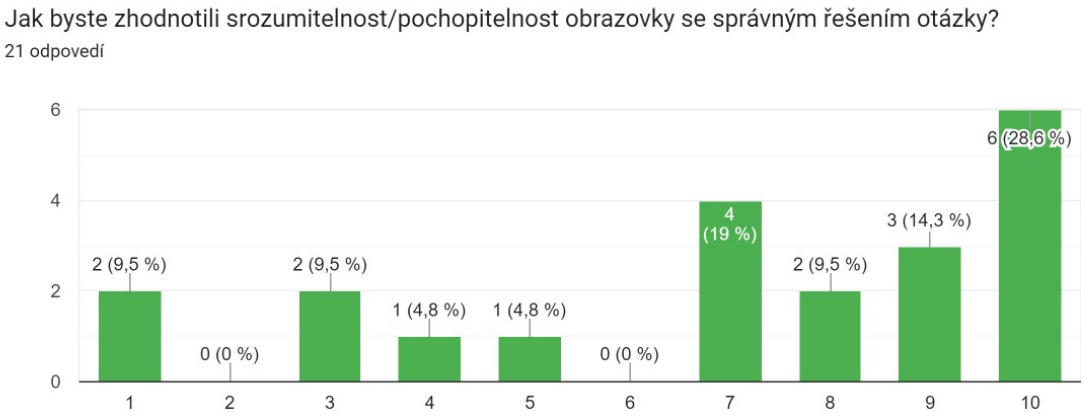
Obr. C.7 Graf spokojnosti učiteľov s UI výsledkov kvízu



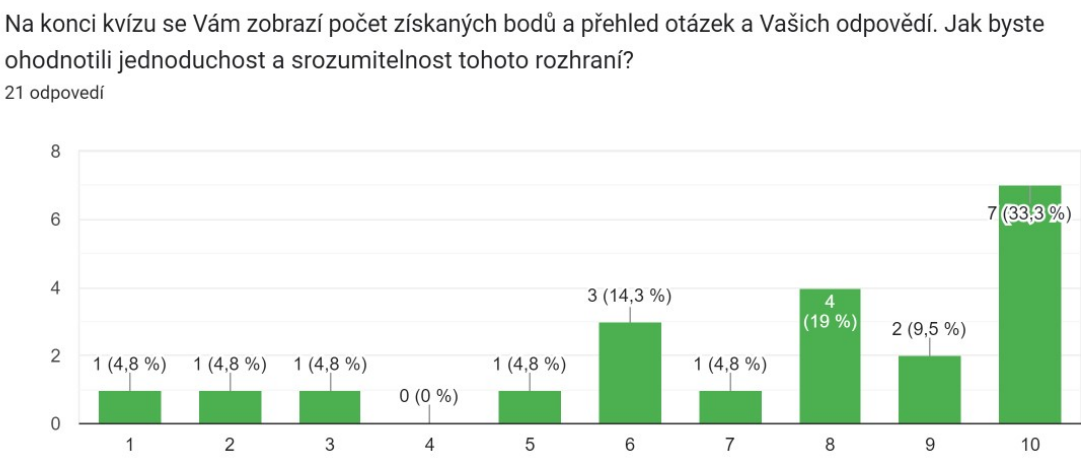
Obr. C.8 Graf spokojnosti študentov s UI vstupu do kvízu



Obr. C.9 Graf spokojnosti študentov s UI otázky v kvíze



Obr. C.10 Graf spokojnosti študentov s UI riešenia otázky



Obr. C.11 Graf spokojnosti študentov s UI výsledkov kvízu

Bibliografia

1. KAHOOT! *Kahoot! | Learning games | Make learning awesome!* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://kahoot.com/`.
2. CISCO SYSTEMS. *Slido - Audience Interaction Made Easy* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://www.slido.com/`.
3. QUIZIZZ. *Quizizz | Free Online Quizzes, Lessons, Activities and Homework* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://quizizz.com/`.
4. QUIZLET. *Your Sets | Quizlet* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://quizlet.com/latest`.
5. ANKITECTS. *Anki - powerful, intelligent flashcards* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://apps.ankiweb.net/`. publisher:
6. MOODLE. *Home | Moodle.org* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://moodle.org/`.
7. MOODLE. *Test — MoodleDocs* [online]. 2024. [cit. 2024-05-07]. Dostupné z : `https://docs.moodle.org/4x/fr/Test`.
8. KAM FIT ČVUT. *MARAST* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://marast.fit.cvut.cz/`.
9. DUOLINGO. *Learn a language for free* [online]. 2024. [cit. 2024-04-25]. Dostupné z : `https://www.duolingo.com/`.
10. HARTINGER, David. *Lekce 2 - Monolitická a dvouvrstvá architektura* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://www.itnetwork.cz/monoliticka-a-douvrstva-architektura`.
11. IBM. *What Is Three-Tier Architecture? | IBM* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://www.ibm.com/topics/three-tier-architecture`.
12. ORACLE. *What is MySQL?* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://www.oracle.com/in/mysql/what-is-mysql/`.
13. AMAZON WEB SERVICES. *What is PostgreSQL? – Amazon Web Services* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://aws.amazon.com/rds/postgresql/what-is-postgresql/`.
14. MOZILLA DEVELOPER NETWORK. *An overview of HTTP - HTTP | MDN* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview`.
15. JAVATPOINT. *What is Transmission Control Protocol (TCP)? Header, Definition - javatpoint* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://www.javatpoint.com/tcp`.

16. DJANGO. *Django* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://www.djangoproject.com/.
17. RAILS TEAM. *Ruby on Rails* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://rubyonrails.org/.
18. MICROSOFT. *What is ASP.NET? | .NET* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet.
19. BROADCOM. *Spring home* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://spring.io/.
20. RONACHER, Armin. *Welcome to Flask — Flask Documentation (3.0.x)* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://flask.palletsprojects.com/en/3.0.x/.
21. BUILT IN. *What Is a WSGI (Web Server Gateway Interface)? | Built In* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://builtin.com/data-science/wsgi.
22. ORACLE. *Java | Oracle* [online]. 2024. [cit. 2024-05-07]. Dostupné z : https://www.java.com/en/.
23. TPOINT TECH. *JVM | Java Virtual Machine - Javatpoint* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://www.javatpoint.com/jvm-java-virtual-machine.
24. JETBRAINS. *Kotlin Programming Language* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://kotlinlang.org/.
25. BILLWAGNER. *A tour of C# - Overview - C#* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.
26. COURSERA. *C# Programming: What It Is, How It's Used + How to Learn It* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://www.coursera.org/articles/c-sharp.
27. PYTHON SOFTWARE FOUNDATION. *Welcome to Python.org* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://www.python.org/.
28. RUBY COMMUNITY. *Ruby Programming Language* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://www.ruby-lang.org/en/.
29. INDEED. *What Is SOAP API? (Plus Comparison to REST API and Benefits)*. 2024. Dostupné tiež z: https://www.indeed.com/career-advice/career-development/what-is-soap-api.
30. AMAZON WEB SERVICES, INC. *SOAP vs REST - Difference Between API Technologies - AWS* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://aws.amazon.com/compare/the-difference-between-soap-rest/.
31. THE GRAPHQL FOUNDATION. *GraphQL | A query language for your API* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://graphql.org/.
32. HYGRAPH. *What is GraphQL?* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://hygraph.com/learn/graphql.
33. MOZILLA FOUNDATION. *HTML: HyperText Markup Language | MDN* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://developer.mozilla.org/en-US/docs/Web/HTML.
34. MOZILLA FOUNDATION. *CSS: Cascading Style Sheets | MDN* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://developer.mozilla.org/en-US/docs/Web/CSS.
35. MOZILLA FOUNDATION. *JavaScript | MDN* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://developer.mozilla.org/en-US/docs/Web/JavaScript.
36. THE PHP GROUP. *PHP: Hypertext Preporocessor*. 2024. Dostupné tiež z: https://www.php.net/.
37. EVAN, You. *Vue.js* [online]. 2024. [cit. 2024-05-06]. Dostupné z : https://vuejs.org/.

38. META OPEN SOURCE. *React* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://react.dev/`.
39. GEEKSFORGEEKS. *React Tutorial* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://www.geeksforgeeks.org/react-tutorial/`.
40. GOOGLE. *Angular* [online]. 2024. [cit. 2024-05-06]. Dostupné z : `https://angular.io/`.
41. SOLARWINDS WORLDWIDE. *What is Nginx: Everything You Need to Know* [online]. 2024. [cit. 2024-05-07]. Dostupné z : `https://www.papertrail.com/solution/guides/nginx/`.
42. KOTEST TEAM. *Kotest / Kotest* [online]. 2024. [cit. 2024-05-07]. Dostupné z : `https://kotest.io/`.
43. MOCKITO. *Mockito framework site* [online]. 2024. [cit. 2024-05-07]. Dostupné z : `https://site.mockito.org/`.

Obsah příloh