Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

**MuseStream**

**Aplikácia pre online hudobné hodiny**

**Projekt z predmetu MTAA**

19.4.2022

Sebastián Petrík (ID 110872), Štefan Otto Novák

Predmet: **Mobilné technológie a aplikácie**

Cvičiaci: Ing. Peter Trúchly, PhD

Akademický rok 2021/2022, 2. ročník

# Popis funkcionality

Aplikácia slúži na správu hudobných lekcií kde cieľová skupina sú žiaci a učitelia. Učiteľ môže vytvoriť triedu, ktorá má v sebe študentov.

Každý študent triedy má pridelené lekcie. V rámci lekcie je možné posielať správy medzi učiteľom a študentom. Študent môže požiadať o pridanie do triedy alebo je pridaný učiteľom.

Lekcia poskytuje učiteľovi a študentovi pripojiť sa do videohovoru.

# Návrh obrazoviek

Obrazovky sme navrhli v nástroji Figma Jam. Obrazovky sú rozdielne pre rozdielne typy používateľov – máme 3 typy používateľov – student, teacher a admin.

Každá obrazovka má menu (dostupné cez hamburger tlačidlo).

Pre lepšie prezeranie obrazoviek je možne navštíviť návrh cez link:

<https://www.figma.com/file/dKFQ21pcRz7ubFZ0WihLdG/MTAA-JAM?node-id=0%3A1.ň>

## Úvodné obrazovky

Graphical user interface, application, Teams

Description automatically generated

Obr.: Úvodné obrazovky

## Obrazovky študenta

A picture containing diagram

Description automatically generated

Obr.: Menu študenta

Graphical user interface, text, application, chat or text message

Description automatically generated

Obr.: Požiadanie učiteľa o pridanie do triedy

Graphical user interface, application

Description automatically generated

Obr.: Priebeh od zobrazenia tried študenta až po samotný videohovor

A picture containing timeline

Description automatically generated

Obr.: Nastavenia študenta (rovnaké pre učiteľa)

## Obrazovky učiteľa

A picture containing diagram

Description automatically generated

Obr.: Menu učiteľa

Graphical user interface, text, application, chat or text message

Description automatically generated

Obr.: Priebeh toku od zobrazenia tried až po videohovor pre učiteľa

Graphical user interface, application, chat or text message

Description automatically generated

Obr.: Vytvorenie triedy, zobrazenie študentov, lekcie študenta a

pridanie študenta do triedy

Graphical user interface, text, application, chat or text message

Description automatically generated

Obr.: Správa požiadavok na pridanie do triedy pre učiteľa

## Obrazovka súborov

Graphical user interface, text, application, chat or text message

Description automatically generated

Obr.: Zoznam súborov s tlačidlami pre stiahnutie a odoslanie

## Obrazovky administrátora

Graphical user interface

Description automatically generated

Obr.: Obrazovky administrátora – hlavná obrazovka, používatelia,

úprava používateľa

# Dokumentácia API endpointov

Dokumentácia API bola písaná vo formáte OpenAPI a je dostupná na linku:

<https://app.swaggerhub.com/apis-docs/Plasmoxy/MTAA-api/1>.

Kód dokumentácie je dostupný na linku:

<https://app.swaggerhub.com/apis/Plasmoxy/MTAA-api/1>.

# Návrh dátového modelu

Dátový model je možne prezrieť si na diagrame na linku:

<https://dbdiagram.io/d/62279eb761d06e6eadc66795>.

Graphical user interface, application, Teams

Description automatically generated

Obr.: Dátový model

# Implementácia backendu

Backend je implementovaný vo frameworku AdonisJS v jazyku TypeScript. Framework umožňuje komunikáciu s databázou cez ORM a pohodlnú implementáciu endpointov. Počas implementácie backendu bolo potrebné urobiť voči návrhu zopár úprav.

Kód backendu je možné si pozrieť na linku <https://github.com/Plasmoxy/musestream-backend-adonis> .

Každý endpoint podľa openapi dokumentácie (viď link nižšie) je implementovaný v určitom controlleri, ktorý ho obsluhuje. Jednotlivé controllery sú vo folderi „app/Controllers/Http“. Tieto controllery sú napájané na endpointy cez súbor „start/routes.ts“.

Využili sme zabudovanú funkcionalitu AdonisJS na napr. automatickú serializáciu dát, autentifikáciu cez tokeny, Lucid ORM pre definovanie modelov databázy a prístup k nej, migrácie tabuliek databázy, riešenie výnimiek cez stavové kódy, upload a servovanie súborov.

## Úprava modelu

Celkový návrh modelu zostal podobný, bolo potrebné odstrániť niektoré zbytočné polia a opraviť názvy kľúčov pri vzťahoch. Celkovo sme vybrali možno príliš komplikovaný spôsob riešenia databázy, možno by dokumentová databáza bola jednoduchšia. Relačný model nám ale umožňuje efektívne získavať dáta.

A screenshot of a computer

Description automatically generated with low confidence

## Úprava endpointov

V rámci riešenia backendu bolo potrebné zmeniť niektoré endpointy, no funkcionalita zostala zachovaná ako pri návrhu.

Endpointy **sú definované a opísané** cez OpenAPI špecifikáciu a ich dokumentáciu je možné si pozrieť na linku <https://app.swaggerhub.com/apis-docs/Plasmoxy/MTAA-api/1> .

Medzi úpravy endpointov patria:

* zmena prihlasovacieho systému – prihlasovanie funguje na princípe tokenov cez endpointy /login a /register, autentifikácia je riešená cez Bearer token
* pridanie niektorých chýbajúcich endpointov, napr. endpoint pre pridanie Lesson pre Class Studenta.
* úprava endpointov pre súbory – stačia 2 endpointy = jeden na zoznam súborov v triede, druhý na upload súboru do triedy
* pridané statické servovanie súborov cez endpoint /static
* drobné úpravy v schéme endpointov a vo vrátených stavových kódoch.

## Úprava akceptačných testov

Vzhľadom na to, že sa zmenili backend endpointy, bolo potrebné opraviť niektoré akceptačné testy. Celkové idey testov sú zachované.

# Implementácia aplikácie

Aplikácia je implementovaná vo frameworku Flutter a využíva rôzne knižnice (viď pubspec.yaml súbor). Aplikácia pozostáva z množstva obrazoviek, väčšina podľa návrhu (bolo potrebné niekde upraviť design, niekde zasa zmeniť umiestnenie prvkov).

Využíva HTTP protokol na komunikáciu s backendom, a teda využíva väčšinu volaní, ktoré sme implementovali. Niektoré volania bolo potrebné upraviť/pridať, napríklad pri získavaní tried študenta bolo potrebné pripojiť prepojené modely učiteľa/študenta z dôvodu zobrazovania triedy.

Git repozitar: <https://github.com/Plasmoxy/musestream-flutter> .

Web verzia: <https://seb-mtaa.surge.sh/#/> (pre ukazkove ucely, nemusi fungovat/byt aktualna).

## Úprava funkcionality oproti návrhu

* Zmena akceptovania požiadaviek na pridanie do triedy – je to teraz riešené priamo v triede, pretože každá trieda má vlastné požiadavky – **opravený akceptačný test 2**
* Pri triedach a lekciách bolo pridané tlačidlo pridať a upraviť, pridaná osobitná obrazovka na tieto funkcionality
* Odstránené správy počas videohovoru, správy by mali byť pri triede/lekcii, nie videohovore, potrebné implementovať cez websockety
* Odstránené manuálne pridanie študenta do triedy – je potrebné používať class requests

## Konečný popis funkcionality

* Aplikácie funguje pre 3 typy používateľa – student, teacher, admin. Rozhranie aplikácie sa mení v závislosti od typu, napr. študentovi nedovolí editovať triedu atď.
* Učiteľ môže vytvárať svoje triedy, editovať ich, vymazať, pridať/editovať/vymazať lekciu pre študenta v danej triede. Môže potvrdiť/zamietnuť žiadosti na pridanie do triedy.
* Aj učiteľovi aj študentovi sa zobrazujú priradené triedy a lekcie. Môžu si pozrieť ich detaily.
* Používatelia si môžu nastaviť meno a email cez nastavenia.
* Aplikácia podporuje prihlásenie a registráciu.
* Administrátor môže meniť typ používateľov cez jeho obrazovku.
* Učiteľ môže uploadovať súbory do triedy, študenti ich následne môžu prezerať/sťahovať.
* Aplikácia podporuje pripojenie do videhovoru v rámci 1 lekcie, učiteľ najprv spustí videohovor, následne sa študent pripojí.

## Implementácia WebRTC

Na implementáciu hovoru sme použili knižnicu flutter\_webrtc, ktorú využívame v súbore signaling.dart. Postupovali sme podľa tutoriálu Maxa Webera <https://github.com/md-weber/webrtc_tutorial>, kde bolo potrebné niektoré veci opraviť.

Na prenos a uchovanie SDP dát používame Firestore, ktorý umožňuje počúvanie na zmeny. Využívame Google STUN servery, hovor je možné uzatvoriť aj cez NAT. Hovor podporuje aj audio, aj video. Hovor funguje na princípe miestností = „rooms“, jeden účasník vytvorí miesnosť a uloží tam SDP dáta, druhý účastník sa do miestnosti potom pripojí.

A screenshot of a person

Description automatically generated with low confidenceGraphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generated

# Frontend akceptačné testy

|  |  |
| --- | --- |
| **Test 1:** Žiadosť oregistráciu na triedu (positive) | |
| **Vstupné podmienky:** | Používateľ, ktorý má rolu študenta je prihlásený v aplikácii, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Študent vytvorí žiadosť pre registráciu na triedu |
| **Postup**: | 1. Používateľ stlačí tlačidlo HAMBURGER IKONA a následne sa otvorí menu na ľavej strane 2. Používateľ stlačí tlačidlo REQUEST CLASS 3. Zobrazí sa zoznam dostupných tried 4. Používateľ stlačí na vybranú triedu 5. Zobrazia sa informácie o vybranej triede 6. Používateľ stlačí tlačidlo ASK TEACHER TO JOIN THIS CLASS |

|  |  |
| --- | --- |
| **Test 2:** Potvrdenie žiadosti oregistráciu na triedu (positive) | |
| **Vstupné podmienky:** | Používateľ, ktorý má rolu učiteľa je prihlásený v aplikácii, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Študentovi ale aj učiteľovi sa zobrazí potvrdená trieda vo vlastnej sekcie My Classes |
| **Postup**: | 1. Používateľ otvorí detail triedy 2. Zobrazí sa okrem iného aj zoznam čakajúcich žiadosti s informáciami 3. Používateľ potvrdí žiadosť cez tlačidlo s ikonou fajky (ACCEPT) |

|  |  |
| --- | --- |
| **Test 3:** Prihlásenie do aplikácie (positive) | |
| **Vstupné podmienky:** | Používateľ otvorí aplikáciu a nie je prihlásený, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Požívateľ sa prihlási do aplikácie |
| **Postup**: | 1. Používateľ otvorí aplikáciu 2. Zobrazí sa Login obrazovka 3. Používateľ vloží správne prihlasovacie údaje a stlačí tlačidlo LOGIN 4. Databáza preverí správnosť údajov a sa zobrazí hlavná stránka aplikácie |

|  |  |
| --- | --- |
| **Test 4:** Prihlásenie do aplikácie (negative test) | |
| **Vstupné podmienky:** | Používateľ otvorí aplikáciu a nie je prihlásený, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Požívateľ sa neprihlási do aplikácie |
| **Postup**: | 1. Používateľ otvorí aplikáciu 2. Zobrazí sa Login obrazovka 3. Používateľ vloží nesprávne prihlasovacie údaje a stlačí tlačidlo LOGIN 4. Databáza preverí správnosť údajov a vypíše správu “The username or password is incorrect” |

|  |  |
| --- | --- |
| **Test 5:** Pripojenie ku video hovoru (negative test) | |
| **Vstupné podmienky:** | Používateľ, ktorý má rolu študenta je prihlásený v aplikácii, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Študent sa nepripojí ku video hovoru |
| **Postup**: | 1. Používateľ stlačí tlačidlo HAMBURGER IKONA a následne sa otvorí menu na ľavej strane 2. Používateľ stlačí tlačidlo MY CLASSES 3. Zobrazí sa zoznam tried ku ktorým je prihlásený 4. Používateľ stlačí na vybranú triedu 5. Zobrazia sa informácie o vybranej triede 6. Používateľ stlačí na lekciu, ktorá ešte nezačala 7. Zobrazia sa informácie o vybranej triede 8. Tlačidlo pre pripojenie do videhovoru je vypnuté, pretože videohovor ešte nezačal |

|  |  |
| --- | --- |
| **Test 6:** Zmena údajov (negative test) | |
| **Vstupné podmienky:** | Používateľ, ktorý má rolu učiteľa je prihlásený v aplikácii, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Údaje sa neaktualizujú |
| **Postup**: | 1. Používateľ stlačí tlačidlo HAMBURGER IKONA a následne sa otvory menu na ľavej strany 2. Používateľ stlačí tlačidlo SETTINGS 3. Zobrazí sa obrazovka s textovými poliami 4. Používateľ vyplní všetky textové polia, okrem pole Email 5. Používateľ stlačí tlačidlo CONFIRM CAHNGES 6. Údaje sa neaktualizujú, pri textovom poli emailu sa zobrazí error |

| **Test 7:** Registrácia účtu (negative test) | |
| --- | --- |
| **Vstupné podmienky:** | Používateľ otvorí aplikáciu a nie je prihlásený, aplikácia má pripojenie na internet |
| **Výstupné podmienky**: | Požívateľ nevytvorí nový účet |
| **Postup**: | 1. Používateľ otvorí aplikáciu 2. Zobrazí sa Login obrazovka 3. Používateľ stlačí tlacidlo SIGN UP 4. Zobrazí sa obrazovka s textovým polom pre Username a Password, aj informácie používateľa 5. Požívateľ vyplní pole Username s používateľským meno, ktoré už používa iný používateľ, vyplní ostatné polia 6. Databáza preverí použite používateľské mena a vypíše správu “This username has already been used” |

# Backend akceptačné testy

|  |  |
| --- | --- |
| **Test 1: Registrácia používateľa a prihlásenie (positive)** | |
| **Vstupné podmienky:** | Používateľ nie je prihlásený ani zaregistrovaný. |
| **Výstupné podmienky:** | Vytvorí sa nový používateľ na backende a prihlási sa = dostane token |
| **Postup:** | 1. Aplikácia pošle POST na /register s JSON dátami pre vytvorenie používateľa (name, password) 2. Backend odpovie 200 OK – používateľ zaregistrovaný 3. Aplikácia pošle GET na /login s dátami používateľa rovnakými, pod akými sa registroval 4. Backend odpovie 200 OK – dané prihlasovacie údaje sú správne a vráti JSON objekt s poľom token, v ktorom je token |
| **Výsledok:** PASS/FAIL | |

|  |  |
| --- | --- |
| **Test 2: Vytvorenie novej triedy ako učiteľ (positive)** | |
| **Vstupné podmienky:** | Používateľ je prihlásený ako učiteľ (požiadavky Bearer token učiteľa). |
| **Výstupné podmienky:** | Vytvorí sa nová trieda, objaví sa v zozname tried daného učiteľa. |
| **Postup:** | 1. Aplikácia pošle POST na /classes s JSON dátami pre vytvorenie novej triedy (title, description, instrument) 2. Backend odpovie 200 OK – trieda bola vytvorená, vráti JSON dáta triedy, pričom učiteľ triedy je daný prihlásený učiteľ. 3. Aplikácia pošle GET na /classes. 4. Backend odpovie 200 OK so zoznamom tried daného učiteľa vrátane novovytvorenej triedy. |
| **Výsledok:** PASS/FAIL | |

|  |  |
| --- | --- |
| **Test 3: Akceptovanie požiadavky študenta o prijatie do triedy (positive)** | |
| **Vstupné podmienky:** | Používateľ je prihlásený ako učiteľ. |
| **Výstupné podmienky:** | Požiadavka o prijatie do triedy bude vymazaná, študent bude pridaný do danej triedy. |
| **Postup:** | 1. Aplikácia pošle POST na /requests/{requestId}, kde parameter requestId je ID danej požiadavky 2. Backend odpovie 200 OK – požiadavka bola akceptovaná a študent pridaný do triedy, požiadavka o prijatie bude vymazaná 3. Aplikácia pošle GET na /classes/{id}/students, kde parameter id je ID triedy, na ktorú bola požiadavka o prijatie adresovaná. 4. Backend odpovie 200 OK so zoznamom študentov danej triedy, obsahujúcim študenta, od ktorého bola odoslaná požiadavka na prijatie do triedy. |
| **Výsledok:** PASS/FAIL | |

|  |  |
| --- | --- |
| **Test 4: Registrácia používateľa s už existujúcim menom (negative)** | |
| **Vstupné podmienky:** | Používateľ nie je prihlásený – požiadavky nemajú dáta o autorizácii, v databáze už existuje používateľ s menom „admin“. |
| **Výstupné podmienky:** | Backend nedovolí registráciu používateľa s rovnaký menom, aké má už existujúci používateľ |
| **Postup:** | 1. Aplikácia pošle POST na /register s JSON dátami pre vytvorenie používateľa, kde meno používateľa je „admin“. 2. Backend odpovie 400 BAD REQUEST – nemožné zaregistrovať používateľa 3. Aplikácia pošle GET na /login s menom a heslom rovnakým, s akými sa pokúšal registrovať 4. Backend odpovie 400 BAD REQUEST – dané prihlasovacie údaje nie sú správne |
| **Výsledok:** PASS/FAIL | |

|  |  |
| --- | --- |
| **Test 5: Aktualizácia dát lekcie (negative)** | |
| **Vstupné podmienky:** | Používateľ je prihlásený ako učiteľ – požiadavky majú dáta o autorizácii. |
| **Výstupné podmienky:** | Backend nedovolí upraviť lekciu s dátumom ukončenia skorším, než je dátum začatia lekcie |
| **Postup:** | 1. Aplikácia pošle PUT na /lessons/{id}, kde parameter id je ID existujúcej lekcie, pričom požiadavka obsahuje JSON s atribútmi „start“ a „end“, čo sú dátumy (ISO) začatia a ukončenia lekcie. Dátum ukončenia je skorší než dátum začatia. 2. Backend odpovie 400 BAD REQUEST – neplatné aktualizačné dáta 3. Aplikácia pošle GET na /lessons/{id}, kde parameter id je ID danej lekcie. 4. Backend odpovie 200 OK, s nezmenenými dátami lekcie (obsahujúc správne dátumy ešte pred pokusom o aktualizáciu). |
| **Výsledok:** PASS/FAIL | |