ETDK-dolgozat

Jákó Dániel

Kiss Krisztián

XXVII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)

Informatika II.: innovatív számítástechnikai termékek, alkalmazások szekció

Kolozsvár, 2024. május 15–18.

AttendEZ

Automatizált jelenlétrögzítő rendszer

Szerzők:

Jákó Dániel

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Kiss Krisztián

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Témavezetők:

dr. Sulyok Csaba, egyetemi adjunktus

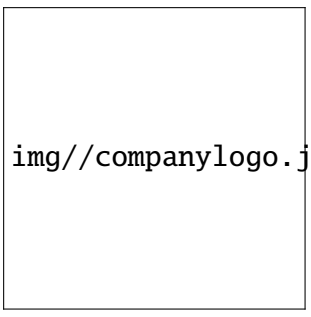
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar

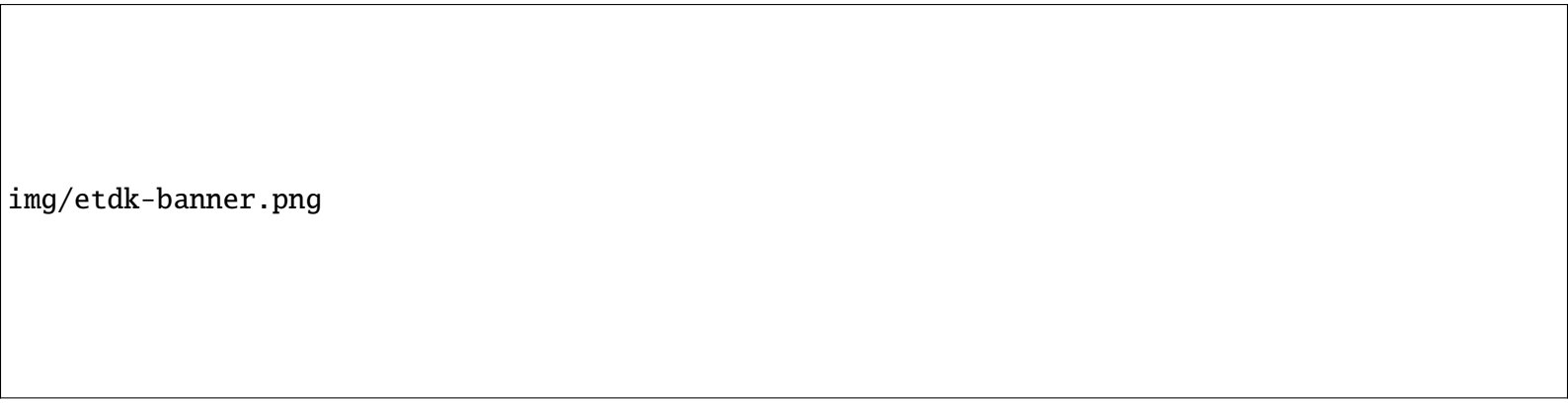
Fekete Árpád-Bence, szoftverfejlesztő,

Codespring

Petkes Kinga, szoftverfejlesztő,

Codespring





AttendEZ: Automatizált jelenlétrögzítő rendszer

Jákó Dániel, Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év
Kiss Krisztián, Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

dr. Sulyok Csaba, egyetemi adjunktus
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar
Fekete Árpád-Bence, szoftverfejlesztő,
Codespring
Petkes Kinga, szoftverfejlesztő,
Codespring

Számos egyetemi tanóra és nagyobb létszámú esemény során fontos a jelenlevők beazonosítása és számba vétele. Az egyetemi órákat tekintve régóta bevett szokás, hogy a jelenlevők maguk írják fel egy kézről-kézre adott jelenléti ívre a nevüket. Bár az eljárás egyszerű, több hátránya is van: időigényes, megszakíthatja a hallgatók figyelmét, illetve az oktatók számára is jelentős többletmunkát jelent a névsor ellenőrzésekor. Ezen túlmenően, a manuális bejegyzés pontatlansága és a könnyen kihasználható csalási lehetőségek miatt a módszer megbízhatósága is csorbul, ami aláássa az akadémiai integritást.

Az AttendEZ projekt célja egy automatizált jelenlétkezelő alkalmazás megvalósítása. A szervezők eseményeket hozhatnak létre és ezek jelenléti listáját kezelhetik. A résztvevők az események jelenléti listáin rögzíthetik a jelenlétüket egy vagy több, az eseményhez tartozó validáció teljesítésével. Egy megvalósított validációs módszer a statikus QR-kód beolvasása mobileszköz segítségével.

Az alkalmazás mikroszolgáltatásokon alapuló szerverarchitektúrára épít. Ez két felhasználói felületet támogat: egy webalkalmazást, ahol a szervezők az események részleteit kezelhetik, illetve egy mobilalkalmazást, amely segítségével a résztvevők rögzíthetik jelenlétüket az események listáin.

Az alkalmazás integrálódik a Canvas LMS rendszerrel, amelyen keresztül a felhasználók bejelentkezhetnek, továbbá a szervezők az általuk kiválasztott tárgyakhoz hozhatnak létre eseményeket, majd ezekhez feltölthetik az események jelenléti listáját.

Tartalomjegyzék

Bevezető

Oktatási környezetben, illetve nagyobb létszámú eseményeknél is gyakran előforduló problémát okoz a résztvevők jelenlétének hatékony nyomonkövetése. A hagyományos papír alapú módszerek nem nyújtanak kellő megbízhatóságot, ami egy akadémiai környezet esetén kulcsfontosságú tényező. Bowen et al. kutatása [8] már 2005-ben rávilágított arra, hogy az elektronikus jelenlétkezelő rendszerek jelentős előnyöket kínálnak a papíralapú rendszerekkel szemben, többek között a valós idejű adatrögzítés és azonnali adathozzáférés lehetőségével. Ezek az elektronikus rendszerek növelik az adatok pontosságát és megbízhatóságát, ezáltal biztosítva az intézmények számára, hogy hatékonyabban reagáljanak a diákok jelenlétével kapcsolatos problémákra.

Több tanulmány [1] [9] [19] is alátámasztja, hogy a diákok rendszeres jelenléte pozitívan befolyásolja az akadémiai sikerességet, továbbá sok esetben szükségszerű a diákok ösztönzése a rendszeres jelenlét fenntartása érdekében. Ezen aspektusok megvalósításához elengedhetetlen egy olyan digitális eszköz, amely lehetőséget teremet a jelenlétrögzítés biztonságos és hatékony gyakorlására. Továbbá, 2024-es adatok [7] szerint az Európában eddig is elterjedt LMS[1] rendszereket használó intézmények száma évente 16.9%-kal növekedik – így az LMS szolgáltatókkal történő integráció kiemlekedően fontos egy jelenlétkezelő rendszer hatékony használatához.

Az *AttendEZ* projekt célja egy olyan digitalizált jelenlétkezelő alkalmazás megvalósítása, amely ezen szempontoknak eleget téve számos moduláris, testreszabható ellenőrzési módszert kínál a csalások hatékony kiküszöbölésére, emellett pedig integrálható külső szolgáltatásokkal is. Az ellenőrzési módszerek testreszabhatósága nagy dinamikusságot biztosít a felhasználók számára, amíg az összetettebb, több lépésből álló ellenőrzési folyamatok fokozott biztonságot nyújtanak.

A projekt a QR-kód[2] alapú validációt támogatja, amely egyszerű és hatékony módja a részvétel igazolásának. Erre olyan felhasználói felületek épülnek, amelyek az események szervezői és résztvevői számára egyaránt hasznos funkciókat látnak el a jelenlétrögzítési folyamatok egyszerű kezeléséhez. A projekt keretén belül a szerzők intézetének infrastruktúráján belül alkalmazott *Canvas LMS*[3] rendszer került integrálásra az alkalmazásba.

A dolgozat 1. fejezete egy kutatás, amely azokat a módszereket vizsgálja, amikkel

[1] Learning Management System
[2] Quick Response-kód; "gyors válasz-kód"; kétdimenziós vonalkód; pontkód
[3] https://www.instructure.com/canvas/

a hatékony és megbízható jelenlétrögzítés megvalósítható, illetve összegzi azon piaci alkalmazásokat, amelyek az *AttendEZ* célkitűzéseihez hasonló funkcionalitásokat tartalmaznak. Gorcső alá kerülnek ezen alkalmazások előnyei és hátrányai, összegzésben pedig kiderül, hogy a vizsgált problémákra miért nem nyújtanak megfelelő megoldást.

A 2. fejezet részletezi a projektben implementált funkcionalitásokat, amelyek meghatározó szerepet játszanak a felhasználók kezelésében, az események rendszerezésében, és a jelenlétrögzítési folyamat teljes működésében.

A 3. fejezet tartalmazza a szoftverrendszer mikroszolgáltatás alapú architektúrájának leírását. A fejezet részletezi az architektúrában szereplő komponensek kialakítását, a köztük történő kommunikációt és kitér az egyes mikroszolgáltatások által ellátott funkcionalitások részletes leírására, miközben átfogó képet ad a szerveroldal és a kliensalkalmazások közötti működésről.

A dolgozat 4. fejezete a webes felület, illetve a mobilalkalmazás működését írja le több technikai szempont figyelembevételével.

Végezetül, az 5. fejezet szemlélteti a rendszer fejlesztésében alkalmazott technológiákat.

A projekt feljesztése a Codespring[4] által szervezett mentorprogram keretén belül kezdődött el. A fejlesztés jelentős része 2023 nyarán, a nyári gyarkorlat részeként történt, majd folytatódott a 2023-2024-es egyetemi tanév első félévében szervezett Csoportos Projekt tantárgy keretein belül.

Köszönet illeti a dolgozat témavezetőjét, dr. Sulyok Csaba egyetmi adjunktust, illetve a Codespring mentorait és szoftverfejlesztő munkatársait, Fekete-Árpád Bencét és Petkes Kingát. Segítségünkre voltak továbbá a Csoportos Projekt tantárgy keretén belül évfolyamtársaink: Boga Zsombor, János-Széll Dávid és Siklódi Zalán.

[4] https://edu.codespring.ro/

1. Hasonló rendszerek

Az jelenléti nyilvántartás automatizálása jelentős előnyöket kínál az oktatási intézmények és más szervezetek számára, különböző technológiák alkalmazásával javítva az adminisztratív folyamatok hatékonyságát és csökkentve az időráfordítást. Az RFID[5] technológia [25] lehetővé teszi a hallgatók gyors és biztonságos azonosítását [26] [4], ahol a diákok NFC[6]-képes kártyákkal jelentkeznek be az előadásokra. A QR-kód alapú rendszerek [15] szintén kihasználják a hallgatók okostelefonjait, lehetővé téve számukra, hogy saját eszközeikkel rögzítsék jelenlétüket, ezzel is csökkentve az előadásokon az adminisztratív feladatokra fordítandó időt. A modern arcfelismerő technológiák [14] továbbfejlesztett biztonsági funkciókat kínálnak, ahol a rendszer a diákok arcjellemzőit elemezve automatikusan rögzíti a jelenlétet, ezzel is növelve az eljárás megbízhatóságát és pontosságát. Ezek a megoldások különösen nagy létszámú előadások esetén bizonyulhatnak hatékonynak, ahol a hagyományos jelenléti ívek kezelése különösen időigényes.

A piacon több olyan jellegű alkalmazás található, amelyek különböző irányokból próbálnak részleges megoldást nyújtani a feltárt problémákra. A következőkben néhány hasonló alkalmazás jellegzetességei kerülnek bemutatásra.

Connecteam

A *Connecteam*[7] egy olyan, főként vállalatok számára fejlesztett alkalmazás, amely lehetőséget teremt az alkalmazottak munkaidejének követésére, emellett számos időgazdálkodási eszközt és ehhez kapcsolódó funkcionalitást biztosít. Az alkalmazottak munkaóráinak és helyének nyomon követésére helymeghatározó technológiát vagy egy külső fizikai eszközt használ, például egy dedikált digitális eszközt, ahova egyedi azonosítót kell felvezetni, vagy RFID chip-olvasót.

aPlus+ Attendance

Az *aPlus+ Attendance*[8] egy olyan sokoldalú alkalmazás, amelyet a jelenlét nyomon követésének és rögzítésének javítására terveztek az oktatási környezetben. Nagy előnyt jelent,

[5]Radio-frequency identification
[6]Near Field Communication
[7]https://connecteam.com/operations/
[8]https://aplusattendance.com/en/he-student-attendance-tracking-software

hogy integrált a *Canvas LMS* rendszerrel. A jelenlétrögzítés folyamatára kínált megoldásai között megtalálható a fizikai RFID-t olvasó eszközök és chipek támogatása, illetve egy kódalapú bejelentkezési folyamat is, amelyben a tanár szolgáltat egy egyedi kódot, melynek beírásával a diák felkerül a jelenléti listára.

Attendance Radar

Az *Attendance Radar*[9] egy olyan akadémiai közegben fejlesztett alkalmazás, amely a legközelebb hozta azon valós problémák megoldását, amely az *AttendEZ* projekt létrejöttét is indítványozta. Az alkalmazás Bluetooth technológia [27] használatával teszi lehetővé a szervezők vagy tanárok számára, hogy a saját eszközeiken keresztül biztonságos Bluetooth jel segítségével jelenlétrögzítést indítsanak el. A résztvevők mobil eszközük segítségével válaszolhatnak a szervező eszköze által kibocsátott Bluetooth jelre, így felkerülve a jelenléti listára. Az alkalmazás fizetős verziója lehetőséget kínál a felsőoktatási intézmények számára a különböző külső rendszerekkel való integrációra is.

Összegzés

A bemutatott alkalmazások a felmerülő probléma eltérő részeire képesek megoldásokat szolgáltatni. A különböző fizikai eszközök bevezetése a rendszerbe (RFID-s chipek és leolvasók), a felhasználók számának növekedésével arányosan teszik bonyolultabbá a rendszer karbantarthatóságát. Ezen eszközök használata korlátozza a rendszer univerzális és helyszíntől független használatát, mivel az eszközök előzetes telepítése és konfigurálása szükséges a hatékony működéshez. Továbbá egyik vizsgált alkalmazás sem nyújtott változtatható vagy összetett ellenőrzési módszereket, hogy a felhasználók által végrehajtott rögzítési folyamatokban csökkenthető legyen a könnyen kihasználható csalási lehetőségek száma. Azok az alkalmazások, amelyek nyújtanak legalább egy hatékony és biztonságos módszert a jelenlétrögzítési próbálkozások ellenőrzésére, illetve szolgáltatnak több felhasználóbarát funkcionalitást, jelentős anyagi költségekkel járnak. Továbbá még ezek esetében is fennáll a külső szolgáltatókkal való integráció problémája.

[9] https://attendanceradar.com/

2. Funkcionalitások

E fejezet célja az *AttendEZ* rendszer működési mechanizmusának és a benne rejlő funkcionális elemeknek bemutatása. A szoftver funkcionális aspektusai középpontjában a felhasználók kezelése, az események létrehozása és kezelése, valamint a résztvevők jelenlétének digitális rögzítése állnak. Bemutatásra kerül továbbá az eseménycsoportok, illetve a felhasználói fiókok és az ezekhez tartozó jogok automatikus kezelése is.

2.1. Felhasználók

Egy közösségorientált, szervezeti kontextusban használt szoftver alapvető funkcionalitása a rendszert igénybe vevő felhasználók pontos beazonosítása és karbantartása. Emellett fontos szempont a felhasználók identitásának biztonságban tartása, azaz annak biztosítása, hogy minden egyes felhasználó kizárólag saját fiókjához férhessen hozzá. [6] [10] Említésre méltó továbbá a felhasználók engedélyeinek kezelése, azaz annak a meghatározása, hogy melyik felhasználó, melyik erőforrással és milyen műveletet végezhet. Ezt az engedély alapú hozzáférésvezérlés valósítja meg. [11]

Ezen elvek mentén az alkalmazást kizárólag felhasználói fiókkal rendelkező személyek érhetik el. A rendszer felhasználóinak fiókjai megkülönböztetésre kerülnek a szolgáltatójuk által: léteznek belsőleg létrehozott és karbantartott felhasználói fiókok, amelyek kezelését teljes mértékben az *AttendEZ* rendszer végzi; illetve felhasználói fiókok jönnek létre a rendszerbe integrált külső szolgáltatásokon keresztül való belépéskor. A projekt esetén a *Canvas LMS* szolgáltatás integrált a rendszerbe.

Mindkét felhasználói fiók-típus esetén a szolgáltató kontextusában egyedien megkülönbözteti a fiókokat az e-mail-címük: tehát létezhet belső felhasználó és külső szolgáltatótól származó felhasználó is ugyanazzal az e-mail-címmel, viszont két ugyanolyan típusú felhasználó nem osztozhat ugyanazon az e-mail-címen. Továbbá, minden felhasználói fiókhoz egyedi identitás is társul, ami lehetőséget biztosít a jövőben ugyanazon személy különböző fiókjainak összekapcsolására. Az identitáshoz kapcsolódik még a felhasználó adminisztratív joga is, azaz hogy tud-e más felhasználók fiókján műveleteket végezni.

A következő alfejezetekben a két felhasználói fiók-típus sajátosságairól lesz szó.

2.1.1. Belső felhasználó

Az alkalmazás lehetőséget teremt a felhasználóknak a regisztrációra. Ennek keretein belül a rendszer a következő adatokat várja el a felhasználótól:

1. *Teljes név*: ezeket külön-külön vezetéknév, illetve keresztnév formájában szükséges megadni, a rendszer viszont az ezekből képezett teljes formát használja fel.

2. *E-mail-cím*: a rendszer nem ellenőrzi, hogy a felhasználó valóban létező e-mail-címet ad meg, csupán azt, hogy a megadott érték megfelel-e a helyes e-mail-cím szintaxisnak.

3. *Jelszó*: a felhasználói fiókok védelme érdekében előírások határozzák meg a jelszavak minimális hosszát és karakterkészletét.

Amennyiben a megadott e-mail-cím még nem foglalt egy létező felhasználó által és a többi adat is megfelel a követelményeknek, az új felhasználói fiók sikeresen rögzítésre kerül, az alapértelmezetten non-adminisztratív jogot tartalmazó identitással, illetve alapértelmezett eseménycsoport-szintű engedélyekkel együtt – ez utóbbi szerepét a 2.3. alfejezet szemlélteti. Ellenkező esetben megfelelő hibaüzenet tájékoztatja a regisztrálni próbáló felhasználót.

Bejelentkezéskor a regisztráció során megadott e-mail-cím és jelszó megadása szükséges. Ha az e-mail-cím létezik a rendszerben és a hozzá tartozó jelszó helyesen lett megadva, a felhasználó megkapja a hozzáférést a rendszerhez és a számára hozzáférhető erőforrásokhoz. Ellenkező esetben tájékoztatást kap a hibájáról.

2.1.2. Külső szolgáltatásból származó felhasználó

A belső rendszerrel ellentétben, amely esetén szabadon regisztrálható felhasználói fiók, a külső szolgáltatót használók számára nincs dedikált regisztrációs lépés a fiókjuk létrehozása során. Amikor egy felhasználó legelőször jelentkezik be a külső szolgáltatón keresztül, a belső rendszerben automatikusan létrejön számára egy felhasználói fiók, az alapértelmezett identitásával együtt. A rendszer felhasználja a külső szolgáltató által rendelkezésre bocsátott adatokat, hogy belsőleg eltárolja az új felhasználó nevét és e-mail-címét, illetve hogy az eseménycsoportokat és a hozzájuk tartozó engedélyeket szinkronizálja – erről bővebben a 2.3. alfejezetben lehet olvasni.

2.2. Eseménycsoportok és események

Az alkalmazás központi eleme az *esemény* (*session*) entitás, ami a felhasználók valós életbeli tevékenységeinek digitális reprezentációját teszi lehetővé. Ez az entitás olyan konkrét eseményeket tükröz, mint az előadások, konferenciák vagy egyéb találkozók, ahol szükség van a résztvevők jelenlétének dokumentálására.

Az események *eseménycsoportokba* (*session groups*) rendeződnek. Ezen csoportok kizárólag automatikusan jönnek létre. A *Nyilvános* (*Public*) eseménycsoport alapértelmezetten létezik az alkalmazásban, továbbá egy külső szolgáltatóval való integrációs folyamat során kinyert információk segítségével újabb eseménycsoportok jöhetnek létre. Egy adott eseménycsoportba tartozó események száma nincs korlátozva, viszont az események címének egyediségét a csoporton belül szükséges biztosítani.

Egy esemény több meghatározó attribútummal rendelkezik, mint például cím, leírás, validációs metódusok, elérhetőségi állapotjelző és az eseménycsoport, amelybe tartozik. Egy esemény tulajdonosa az a felhasználó, aki létrehozta azt. Kizárólag ennek a felhasználónak van joga kezelni az adott eseményt – például módosítani az attribútumait, aktívvá tenni, jelenlevőket hozzáadni vagy törölni. Egy esemény nem mozgatható át abból az eseménycsoportból, amelyikben létre lett hozva.

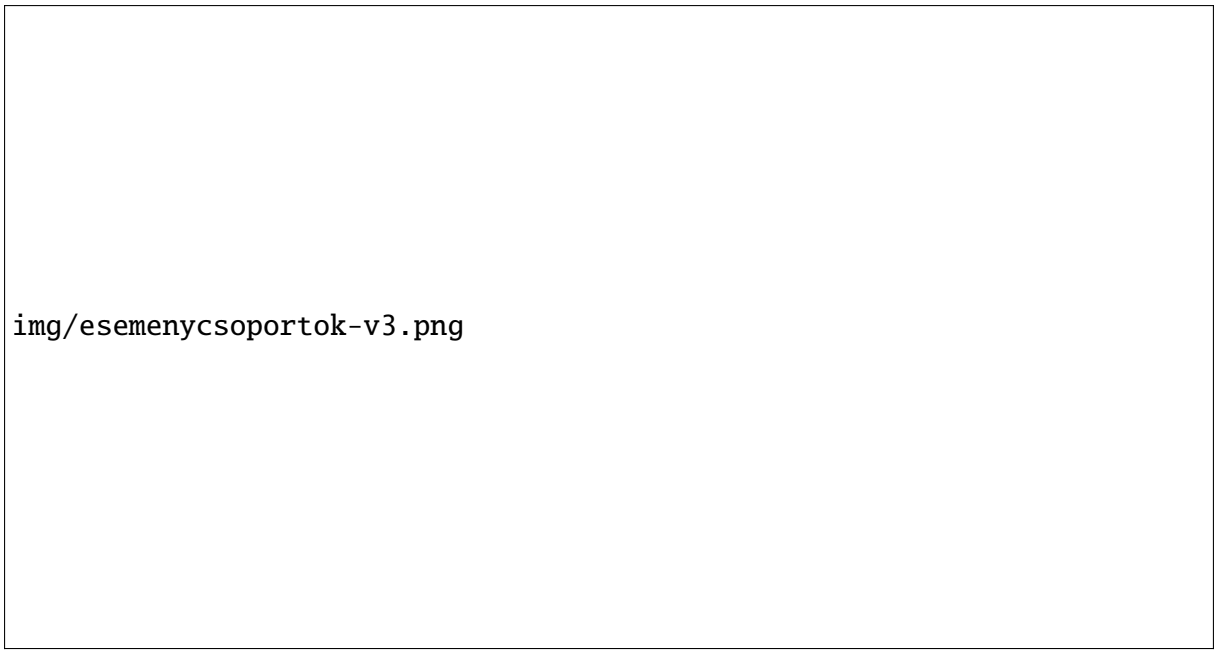
A *Canvas LMS*-ből származtatott eseménycsoportok nevét az onnan átvett tantárgyak neve és a tantárgyak kódja alkotja, ez utóbbi zárójelben szerepel. Például az *Algoritmika* tantárgy, melynek kódja *ALGHU23-24* az *Algoritmika (ALGHU23-24)* nevű eseménycsoportra képeződik le. Ez biztosítja, hogy az eseménycsoport neve egyaránt olvasható és egyedi.

A rendszerben leképezett eseménycsoportokat és eseményeket példaként az 1. ábra szemlélteti.

2.3. Felhasználók és eseménycsoportok kapcsolata

Az eseménycsoportok bevezetése lehetővé teszi a rendszerszintű felhasználójogok kiosztásának elkerülését. Továbbá a felhasználói jogosultságok eseménycsoportokkénti meghatározása által, egy felhasználó egyetlen fiókkal is rendelkezhet eltérő jogosultságokkal a különböző eseménycsoportokban. Két féle engedélyt különíthetünk el egy adott eseménycsoportra vonatkozóan: események létrehozására való jogosultságot és résztvevőként jelenlét rögzítésére való jogosultságot.

Az *AttendEZ* rendszer által karbantartott felhasználók alapértelmezett résztvevői



1. ábra. Példa az alapértelmezett *Nyilvános* és a *Canvas LMS* tantárgyaknak megfelelő eseménycsoportokra. Az *AttendEZ* rendszer által alapértelmezetten létrehozott és karbantartott eseménycsoportot a kék keret, illetve a *Canvas LMS* tárgyaknak megfelelő csoportokat a piros keret jelzi.

jogosultságot kapnak a *Nyilvános* eseménycsoportban. Ezen jogosultság megléte szükséges ahhoz, hogy egy felhasználó validáción keresztül saját jelenlétét rögzítse a *Nyilvános* csoportban levő, aktív események keretein belül. Azon felhasználók, akik regisztrációkor olyan e-mail-címet adnak meg, amely tartalmazza a `create` kulcsszót, a résztvevői jog mellett létrehozói jogot is szereznek – ez kizárólag a fejlesztés tesztelésének céljából lett megvalósítva.

A *Canvas LMS* rendszer egy tantárgyának keretén belül a felhasználók különböző szerepköröket tölthetnek be, mint például: tanár, tanársegéd és diák. Egy felhasználó *Canvas LMS* rendszeren keresztül végrehajtott bejelentkezését követően, a felhasználóhoz tartozó *Canvas LMS* specifikus adatok automatikusan szinkronizálódnak az *AttendEZ* rendszerével. A *Canvas LMS* rendszer tantárgyainak megfelelő, *AttendEZ* eseménycsoportokban a felhasználók meghatározott jogosultságokat kapnak az *AttendEZ* rendszer által. Tanári vagy tanársegédi jogosultságokkal rendelkező felhasználók eseménylétrehozó jogokat kapnak, míg a diákok résztvevői jogosultságot szereznek a tárgyaknak megfelelő eseménycsoportokban.

2.4. A jelenlétrögzítési próbálkozások validációja

Annak érdekében, hogy egy felhasználó felkerüljön egy esemény digitális jelenléti listájára, szükséges, hogy megfeleljen az eseményhez társított validációnak. A validáció magját a validációs metódusok képezik. Ezek különböző formákban létezhetnek, mint például QR-kód ellenőrzés, arcfelismerés vagy helymeghatározás. A validációs metódusok lehetőséget biztosítanak a szervezők számára, hogy az esemény típusához és biztonsági követelményeihez igazítsák a résztvevők hitelesítésének módját.

Minden eseményhez annak létrehozója egy vagy több validációs metódust társít, amelyek *mindegyikét* sikeresen kell teljesítenie a résztvevőnek, hogy az esemény jelenléti listájára felkerüljön. Ez a megközelítés lehetővé teszi, hogy az események különböző biztonsági és szervezési igényeit hatékonyan kezeljék.

Az alkalmazás rugalmasságát az adja, hogy általános célú rendszert alkalmaz a validációs metódusok kezelésére. Új validációs metódusok hozzáadása csupán az adott metódus-specifikus implementációt igényli.

2.4.1. A QR-kód alapú validáció

A projektben implementált validációs metódus egy QR-kódot használ a validációhoz. Azon esemény, amelyhez ez a metódus társított, rendelkezik egy titkos értékkel, ami véletlenszerűen generálódik minden alkalommal, amikor az esemény aktívvá válik. Ebből jelenlétrögzítés során a szervező által használt kliensalkalmazás QR-kódot rajzol ki. Ezt a kódot a résztvevő a saját kliensalkalmazását felhasználva beolvassa. Amennyiben a helyes QR-kód került beolvasásra, illetve a résztvevőnek joga van a részvételhez az eseményhez rendelt eseménycsoportban, a metódus általi validáció kimenetele sikeres.

2.5. Jelenlétrögzítés

Egy eseményhez tartozó jelenlétrögzítés folyamatát két különböző felhasználó szemszögéből lehet megközelíteni: az esemény létrehozására jogosult és a jelenlétének rögzítésére jogosult felhasználó részéről.

Az esemény létrehozására jogosult felhasználó aktívvá teheti az eseményt, majd ezt követően hozzáférhet az eseményhez tartozó titkos információkhoz, amelyet a validációs metódusnak megfelelően szolgáltathat a felhasználó számára. Továbbá lehetősége van a kívánt

felhasználóknak a jelenléti listához történő manuális hozzáadására, amely e-mail-cím alapján történik.

Azon felhasználók, akik résztvevőként jogosultak a jelenlétük rögzítésére végrehajthatják az eseményhez társított összes szükséges validációt, amennyiben az esemény létrehozója előzőleg aktívvá tette az eseményt.

2.6. Jelenléti lista megtekintése és exportálása külső szolgáltatóhoz

Az alkalmazás lehetőséget nyújt egy esemény jelenléti listájának megtekintésére, amely tartalmazza azokat a felhasználókat, akik sikeresen hozzá lettek adva a validáció teljesítésével vagy manuálisan az esemény létrehozója által.

Emellett, azon események esetén, amelyek külső szolgáltatótól származó eseménycsoportokhoz tartoznak, elérhető a jelenléti lista feltöltése is az eseményt létrehozó felhasználó számára. Az integrált *Canvas LMS* rendszer esetén a feltöltés ahhoz a *Canvas LMS* tantárgyhoz történik, amelyikből az adott eseménycsoport származik. Ezen belül létezhet több publikált, *igennel* vagy *nemmel* minősíthető feladat, ezek közül kerül kiválasztásra egyik az esemény létrehozója által. A jelenléti listán levő felhasználók számára a *Canvas LMS* rendszerben az adott feladat *igennel* minősítődik.

3. Mikroszolgáltatások

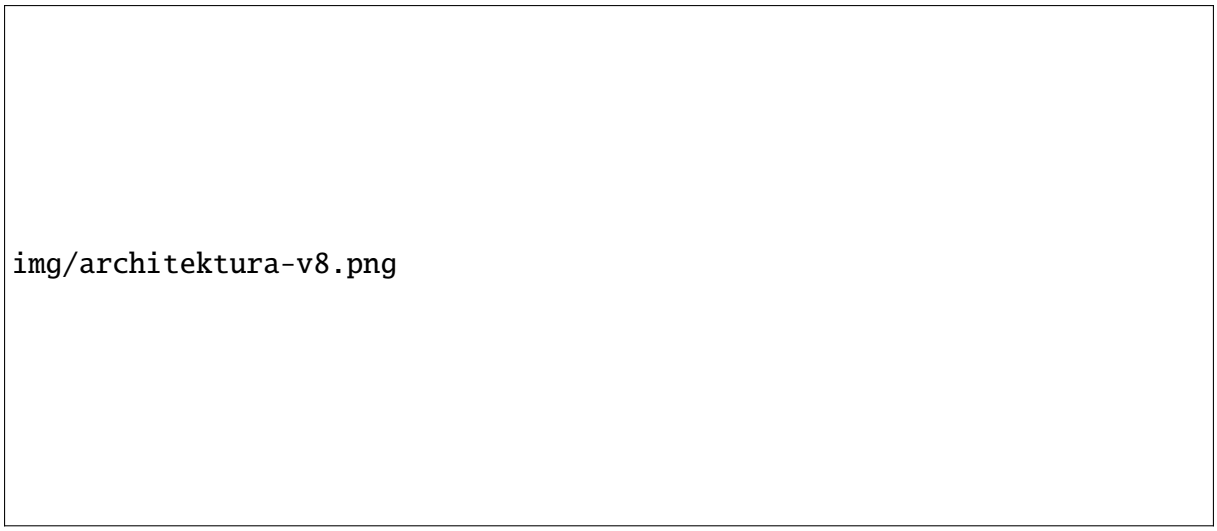
A projekt a mikroszolgáltatások architektúra paradigmájára [13] épül, ami átalakította a szoftverfejlesztési módszertanokat azzal, hogy lehetővé tette az alkalmazások szegmentálását kisebb, önállóan működő komponensekre. Az elosztott, független komponensekből összetevődő szerkezet lényegesen hozzájárul a fejlesztési agilitás, a technológiai diverzitás és a rendszerek skálázhatóságának növeléséhez, miközben elősegíti a folyamatos integrációt és kitelepítést (CI/CD[10]). A fejlesztési függőségeket tekintve, egy komponens módosítása nem indukál automatikusan változásokat a többi komponensben. Technológiai szemszögből a mikroszolgáltatások lehetővé teszik, hogy a különféle komponensek eltérő megoldásokat alkalmazzanak, szemben a monolitikus architektúrával, ahol az egész rendszer azonos technológiai alapokra épül. Üzemeltetési kontextusban a komponensek frissítései vagy módosításai nem követelik meg a teljes alkalmazás újbóli telepítését, ezzel csökkentve az operatív kockázatokat és növelve a rendszer stabilitását. Skálázhatósági aspektusból a függőségek minimalizálása azt eredményezi, hogy a komponensek teljesítményét izoláltan lehet finomítani és igény szerint skálázni.

Az említett architektúra nem mentes a kihívásoktól, többek között a szolgáltatások közötti kommunikáció koordinálása, az adatkezelési stratégiák kidolgozása, valamint a rendszer összetettségének és megbízhatóságának kezelése területén. A projekt kiaknázza a mikroszolgáltatás-alapú architektúra alkalmazásából származó előnyöket, optimalizálva ezzel a fejlesztést és elősegítve a rendszer hatékony skálázhatóságát és megbízhatóságát.

A projekt keretében kialakított szerveroldali mikroszolgáltatás-orientált architektúra alapul szolgál a rendszer számára, ahol minden egyes komponens egy adott üzleti logikai funkciót teljesít. Ennek eredményeképpen hat mikroszolgáltatás került azonosításra és implementálásra: *Felhasználókezelő*, *Hitelesítő szolgáltatás*, *Eseménykezelő*, *Jelenlétkezelő*, *Validációs szolgáltatás*, és *Canvas adapter*. Ezek a komponensek meghatározott szerepet töltenek be a rendszerben, ahol a "*kezelő*" megnevezéssel ellátott szolgáltatások és a *Canvas adapter* közvetlen kapcsolatot tartanak saját adatbázisukkal. A komponensek közötti kommunikáció a RabbitMQ üzenetbróker[11] felhasználásával valósul meg. Az architektúra átfogó vázát a 2. ábra szemlélteti.

A rendszer architektúrája magába foglalja az adatok replikálását a különböző szolgáltatások

[10]Continuous Integration and Continuous Deployment
[11]https://www.rabbitmq.com/



2. ábra. Az *AttendEZ* alkalmazás mikroszolgáltatás-architektúrája.

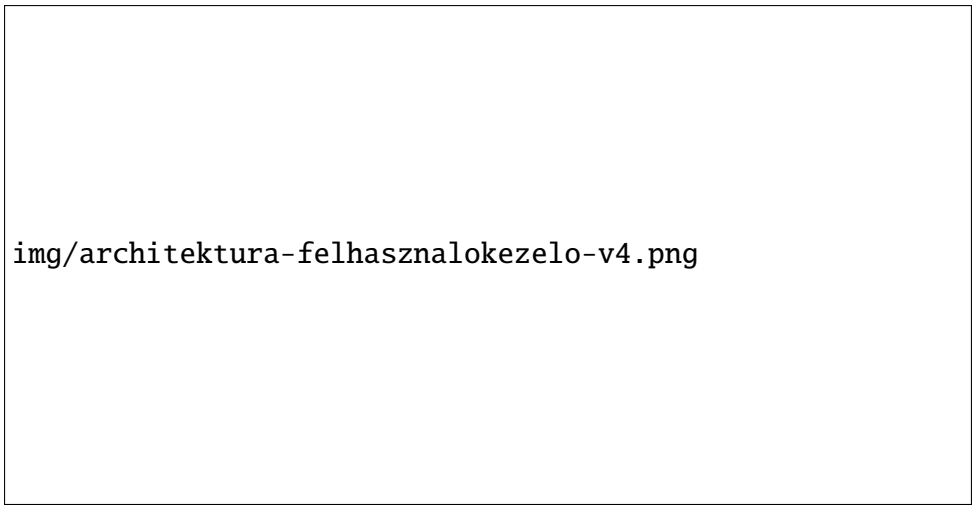
között. Ez lehetővé teszi a gyors és hatékony adathozzáférést a szolgáltatások számára, azonban szükségessé teszi az adatok közötti konzisztencia fenntartását szinkronizációs mechanizmusok segítségével. Ezt a feladatot a RabbitMQ bróker látja el, amely aszinkron üzenetek továbbításával koordinálja az adatok létrehozását, törlését és módosítását a mikroszolgáltatások között.

3.1. Felhasználókezelő

Ezen mikroszolgáltatás a felhasználói fiókok és a hozzájuk tartozó identitások kezelését látja el. Az architektúra többi szolgáltatásával a RabbitMQ brókeren keresztül kommunikál. A kliensalkalmazásokat HTTP[12] REST[13] API[14] végpontokon keresztül szolgálja ki. Az architektúrában elfoglalt helyét a 3. ábra szemléltet

Az alkalmazásban megkülönböztetésre kerülnek a felhasználói fiókok a szolgáltatójuk szerint. Ennek megfelelően léteznek teljes mértékben belsőleg kezelt felhasználók, illetve olyan fiókok, amelyek a *Canvas LMS* rendszerrel való bejelentkezés részeként kerülnek be a rendszerbe. Mindkét felhasználói fiók-típus entitásai egységesen kerülnek eltárolásra a *Felhasználókezelőhöz* társított adatbázisban. Egy attribútum jelzi mindegyik fiók esetén, hogy az melyik szolgáltatótól származik.

[12] Hypertext Transfer Protocol
[13] REpresentational State Transfer
[14] Application Programming Interface



3. ábra. A *Felhasználókezelő* architektúrában elfoglalt helye. A kliensalkalmazások felhasználói kéréseit HTTP REST API-n keresztül fogadja. A kérések kiszolgálása érdekében adatbázis műveleteket hajt végre, illetve üzenetsorokon keresztül kommunikál a rendszer többi szolgáltatásával.

Felhasználók létrehozása

A szolgáltatás a felhasználói fiókok létrehozására vonatkozó kéréseket a hozzácsatolt üzenetsoron figyeli. Ehhez a *Hitelesítő szolgáltatás* és a *Canvas adapter* a megfelelő gyűjtőpontba helyezik kéréseiket. Látható, hogy a fiókok létrehozása nem közvetlenül a kliensalkalmazásokon keresztül történik. Ennek oka, hogy mindkét említett szolgáltatás saját üzleti logikát hajt végre a létrehozás folyamatában.

A *Canvas LMS* rendszertől érkező fiókok esetén nincs szó lokális hitelesítésről az OAuth 2.0[15] folyamat természete miatt. Ezzel szemben, a belsőleg kezelt felhasználóknál eltárolásra kerül a jelszavuk is. Ez eleve hasított formában érkezik a *Hitelesítő szolgáltatástól*. A jelszavak hasított formában történő átadása és tárolása csökkenti az illetéktelen hozzáférés kockázatát a rendszerben.

A *Felhasználókezelő* a felhasználói fiókok létrehozásakor a RabbitMQ bróker gyűjtőpontjába (exchange) üzenetet küld, amely tartalmazza a felhasználó adatait. Erre a *Jelenlétkezelő* figyel, az ő üzleti logikája megköveteli, hogy értesüljön az adat létrehozásáról és replikálja azt a saját adatbázisában.

A felhasználók adatainak elküldése mellett ezen szolgáltatás felelős azért, hogy a belsőleg kezelt felhasználók a *Nyilvános* eseménycsoportban résztvevői, illetve létrehozói jogot kapjanak regisztrációkor – ezt az *Eseménykezelő* tárolja. Ezt szintén a RabbitMQ brókeren küldött üzenet

[15] https://oauth.net/2/

segítségével valósítja meg a rendszer.

Identitások létrehozása

Felhasználói fiókok létrehozásakor automatikusan létrejön az ezeknek megfelelő identitás is. Ez az absztrakció lehetőséget nyújt rendszerszintű jogok meghatározására, illetve több felhasználói fiók összekötésének jövőbeli megvalósítására. A rendszer jelenlegi állapotában rendszerszintű az adminisztratív jog, amely egy felhasználónak engedélyt ad más felhasználók adminisztratív jogának kezelésére.

Felhasználók és identitások lekérdezése és kezelése

A *Felhasználókezelő* a RabbitMQ bróker megfelelő üzenetsorain üzeneteket vár a felhasználók és az identitások lekérdezésére egyaránt. Ennek segítségével más szolgáltatások ellátják a saját üzleti logikájukat, például a *Hitelesítő szolgáltatás* ezen kommunikáció segítségével kéri le egy felhasználó e-mail-címét és jelszavát bejelentkezés során.

A felhasználókra és azok identitására vonatkozó műveletek HTTP REST API végpontok formájában is elérhetőek, azon felhasználók számára, akik adminisztratív engedélyt birtokló identitással rendelkeznek. A `GET /api/users` végpont listázza a rendszer felhasználóit, míg a `PUT /api/users/:userId/admin` végponton keresztül beállítható egy felhasználó identitásának adminisztratív engedélye.

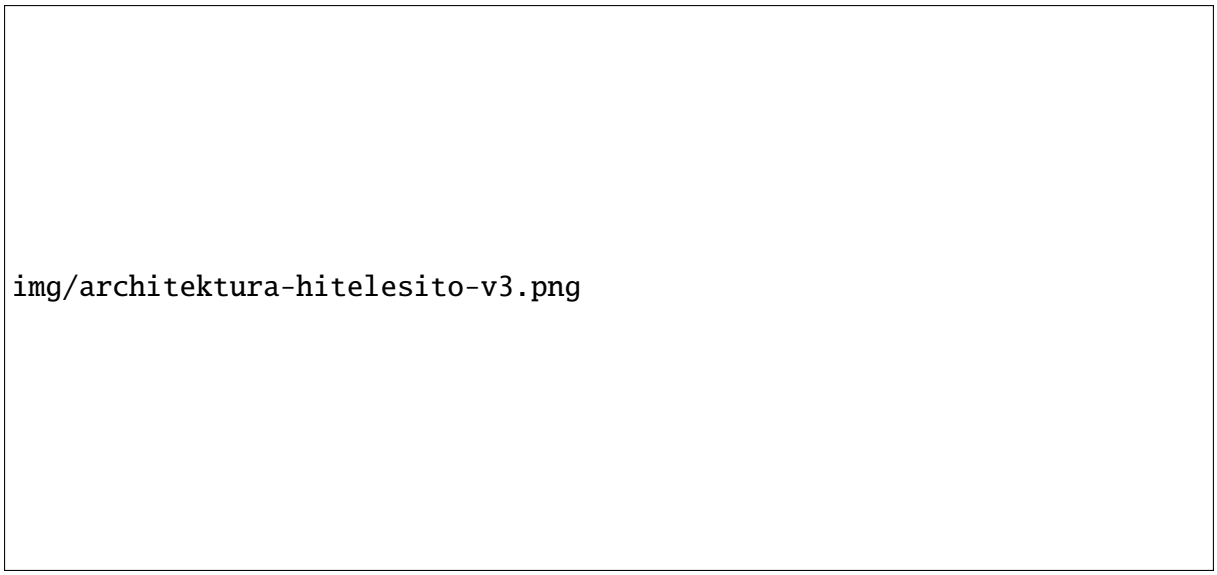
3.2. Hitelesítő szolgáltatás

A *Hitelesítő szolgáltatás* fő felelőssége a felhasználói hitelesítés és azonosítás kezelése. Ezen szolgáltatás látja el a felhasználók regisztrációs és bejelentkezési kéréseit, illetve állít ki hitelesítő tokeneket annak érdekében, hogy a felhasználók kérései azonosítva legyenek az architektúra többi szolgáltatásával szemben. A szolgáltatás HTTP REST API végpontokat, illetve a RabbitMQ bróker által biztosított üzenetsorokat használ a kommunikációra. Az architektúrában elfoglalt helyét a 4. ábra szemlélteti.

Hitelesítő tokenek: JWT

A rendszer a JSON[16] Web Tokeneket [12], röviden JWT-ket használ a kérések hitelesítése érdekében. A JWT egy kompakt módon kódolt szabványosított token formátum, amelyet

[16] JavaScript Object Notation



4. ábra. A *Hitelesítő szolgáltatás* architektúrában elfoglalt helye. A kliensalkalmazások felhasználói kéréseit HTTP REST API-n keresztül fogadja. A kérések kiszolgálása érdekében üzenetsorokon keresztül kommunikál a rendszer többi szolgáltatásával. A *Canvas adapter* szolgáltatás számára JWT-t állít ki.

felhasználói munkamenetek azonosítására és a kliens-szerver közötti hitelesítésre használnak.

A *Hitelesítő szolgáltatás* a JWT-ket RSA[17] algoritmussal írja alá, SHA-256[18] aszimetrikus titkosítást használva. Ehhez szükség van egy privát-nyilvános kulcspárosra, amelyek a mikroszolgáltatás indításakor generálódnak. A *Hitelesítő szolgáltatás* az egyedüli, aki ismeri a privát kulcsot, amelyet az aláírások készítéséhez használ, míg a publikus kulcsot a többi mikroszolgáltatás használja az aláírás hitelesítésére. A JWT-k előállításának központosítása a *Hitelesítő szolgáltatásban* növeli a rendszer biztonságát, mivel ezáltal egységes és következetes módon ellenőrizhető, hogy minden token érvényes és hiteles forrásból származik.

A publikus kulcs megosztása egy JSON Web Key Set (JWKS) [2] formájában történik, a `GET /api/jwks` REST API végponton keresztül. A JWKS egy dinamikus eszköz, amely lehetővé teszi a kulcsok újragenerálását és a kulcspárok cseréjét újratelepítés nélkül. Támogatja az időszakos automatikus kulcsrotációt is, ezáltal növelve a biztonságot. A rendelkezésre bocsátott JWKS-t a rendszer valamennyi mikroszolgáltatása felhasználja a JWT-k, és ezzel együtt a kérések hitelesítése érdekében.

A JWT struktúrája három részből áll, amelyek együttesen biztosítják az adatok integritását és hitelességét:

- A **fejléc** a szabványnak megfelelően tartalmazza az aláíráshoz használt algoritmust – a

[17] Rivest-Shamir-Adleman
[18] Secure Hash Algorithm

rendszer esetén ennek értéke `RS256`.

- A **tartalom** a tokenben tárolt állításokat (claims) foglalja magában. Ez tartalmazza a felhasználó egyedi azonosítóját (ID-ját) (`sub`), a kibocsátót (`iss`), a token kiállításának és lejártának időpontját (`iat` és `exp`), valamint a token egyedi azonosítóját (`jti`). A további attribútumok, mint a felhasználó teljes neve (`fullName`), e-mail-címe (`email`), fiókjának szolgáltatója (`provider`), adminisztrátori joga (`isAdmin`), illetve aggregált esemény-létrehozói joga (`canCreateSession`) személyre szabott információkat biztosítanak a többi szolgáltatásnál való hatékony hitelesítés érdekében. A token tartalmára példa az 5. ábrán szereplő kódrészlet.

- Az **aláírás** a fejléc és a tartalom hasított kombinációja a privát kulccsal aláírva. Ez biztosítja, hogy a token tartalma módosítatlan marad.

A felhasználók kérései HTTP végpontokon keresztül érkeznek a szolgáltatásokhoz, a hitelesítő token a kérések `Authorization` fejlécében kerül átadásra, a `Bearer` séma alapján. [18]

Regisztráció és bejelentkezés folyamata

Azon felhasználók kérései, akik az alkalmazás által natívan felajánlott regisztrációval illetve bejelentkezéssel élnek, a *Hitelesítő szolgáltatáshoz* kerülnek elküldésre. Ezen kommunikáció HTTP REST API végpontokon keresztül történik.

A `POST /api/auth/register` végpontra küldött kérés törzsében szerepelnie kell a regisztrálandó felhasználó e-mail-címének, nevének, illetve jelszavának. A rendszer elkészíti a jelszó hasított változatát, a `BCryptPasswordEncoder` osztály példányát

```
{
  "sub": "6",
  "iss": "authentication-service",
  "iat": 1714144082,
  "exp": 1714230482,
  "jti": "ea11eb19-8fa8-4ab6-b25f-0fdea4d79caf",
  "fullName": "John Doe",
  "email": "example@mail.com",
  "provider": "ATTENDEZ",
  "isAdmin": true,
  "canCreateSession": false
}
```

5. ábra. Példa a JWT tartalmára.

használva, alapértelmezett beállítások mellett. [24] A regisztrációs kérés továbbításra kerül a *Felhasználókezelőhöz* a RabbitMQ bróker megfelelő gyűjtőpontján keresztül.

A bejelentkezés hasonlóképpen működik. A `POST /api/auth/login` végpontra küldött kérés tartalmazza a bejelentkezni kívánó felhasználó e-mail-címét és jelszavát. A *Hitelesítő szolgáltatás* a felhasználó adatainak lekérését üzenetsoron keresztül intézi, amelyet a *Felhasználókezelő* fogad. Amennyiben található ilyen felhasználó, és a bejelentkezési kérelemben megadott jelszó hasított változata megegyezik az eltárolt hasított értékkel, a bejelentkezés sikeres.

Ezen funkcionalitások sikeres kimenetele esetén a *Hitelesítő szolgáltatás* összegyűjti azokat az adatokat, amelyek a token tartalmában személyre szabottak: az e-mail-címet, a teljes nevet, az adminisztratív jogot, illetve az összesített létrehozói engedélyt, azaz, hogy a felhasználónak van-e létrehozói joga legalább egy eseménycsoportban – ez megkönnyíti az autorizáció kiértékelését több kérés esetén is. Ezen információk összegyűjtése érdekében a *Hitelesítő szolgáltatás* üzenetsoron keresztül kommunikációt folytat a *Felhasználókezelővel* és az *Eseménykezelővel*. Végül kiállítja a felhasználó számára a JWT-t, az előző alfejezetben leírtaknak megfelelően.

A *Hitelesítő szolgáltatás* fontos szerepet játszik a *Canvas LMS* rendszert használók esetén is. A *Canvas LMS* felhasználók hitelesítése a *Canvas adapter* szolgáltatás feladatköre, amely ezen folyamat sikeres kimenetelében a *Hitelesítő szolgáltatáshoz* intéz kérést az üzenetsoron keresztül a JWT előállítása érdekében. A folyamatról részletesen a 3.6. alfejezet tárgyal.

3.3. Eseménykezelő

Az *Eseménykezelő* mikroszolgáltatás több funkcionalitást kínál az események és az ezekhez tartozó strukturális elemek hatékony kezelésére, mint például az eseménycsoportok, a felhasználók eseménycsoportokhoz társított jogaik és a validációs metódusok. Ezen szolgáltatás egy dedikált adatbázissal rendelkezik, amely nem csak az események tárolását teszi lehetővé, hanem az ezzel kapcsolatos, előzőleg felsorolt elemek tárolását is.

A szolgáltatás két típusú kommunikációt valósít meg: egyrészt a kliensalkalmazásokkal, REST API végpontokra küldött HTTP kéréseken keresztül; másrészt a többi mikroszolgáltatással, mint például a *Canvas adapter*, *Felhasználókezelő*, *Validációs szolgáltatás* és *Jelenlétkezelő*, üzenetsorok használatával. Az architektúrában elfoglalt helyét a 6. ábra szemlélteti.

6. ábra. Az *Eseménykezelő* architektúrában elfoglalt helye. A kliensalkalmazások felhasználói kéréseit HTTP REST API-n keresztül fogadja. Ez felelős az eseménycsoportok, események és felhasználók eseménycsoportokkénti jogaiknak tárolásáért. A kérések kiszolgáláshoz szükséges információt a többi szolgáltatással történő szinkron és aszinkron kommunikáció révén nyeri ki, üzenetsorokat használva.

Eseménycsoportok

Az alkalmazás architektúrájának alapvető elemei az *eseménycsoportok*. Ezekhez a csoportokhoz vannak hozzárendelve az események, továbbá a felhasználói jogosultságok kezelése is eseménycsoport szinten történik.

A rendszer tervezése révén az eseménycsoportok létrehozása automatikusan történik, így a felhasználóknak nincs lehetőségük manuálisan új csoportokat létrehozni, már meglévőket törölni vagy módosítani. A szolgáltatás feladata, hogy indításkor, egy `ApplicationRunner` osztály segítségével létrehozza a *Nyilvános* eseménycsoportot, amennyiben ez még nem létezik. A további eseménycsoportok a *Canvas LMS* rendszerrel való szinkronizáció eredményeképpen, a mikroszolgáltatások közötti belső kommunikáció során jöhetnek létre.

A különböző eseménycsoportok létrehozására vonatkozó kéréseket, amelyeket a *Canvas adaptertől* kap, az *Eseménykezelő* egy üzenetsorba gyűjti össze. Az üzenetek feldolgozását, a Spring Boot AMQP[19] [23] által szolgáltatott `RabbitListener` metódus valósítja meg, amely dedikáltan ezen üzenetsorba érkező információkra vár.

[19] Advanced Message Queuing Protocol

Felhasználók eseménycsoportok szerinti jogainak tárolása

A felhasználók jogai eseménycsoportok szerint változhatnak. Az eseménycsoportokra vonatkozó jogok alapján a felhasználók két részre oszthatóak: azokra, akik eseményeket hozhatnak létre, és azokra, akik résztvevőként jelenlétüket rögzíthetik. Ez a mikroszolgáltatás a felhasználók jogaira vonatkozó adatokat két forrásból kaphatja: a *Felhasználókezelő* és a *Canvas adapter* mikroszolgáltatásoktól.

Az üzenetsorokkal zajló kommunikáció keretében a *Felhasználókezelő* és a *Canvas adapter* az információt a megfelelő gyűjtőpontokba küldi. Innen a RabbitMQ bróker továbbítja az adatokat az *Eseménykezelő* által létrehozott üzenetsorokba, ahonnan a szolgáltatás Listener-ek segítségével kiolvassa őket.

Az *Eseménykezelő* a felhasználók eseménycsoportokhoz kötődő jogainak kezeléséért felelős, illetve ezeket biztosítja a többi mikroszolgáltatás számára is. Ilyen esetek közé tartozik egy adott felhasználó jogainak bejelentkezéskor történő lekérdezése, a *Hitelesítő szolgáltatás* és *Canvas adapter* részéről, vagy egy adott felhasználó jogainak jelenlét-rögzítési kérelménél történő lekérdezése, a *Validációs szolgáltatás* részéről.

Validációs metódusok kezelése

Az *Eseménykezelő* felelős a validációs metódusok tárolásáért is. Mivel ezek a metódusok egymástól erősen eltérőek és mindegyik saját implementációt igényel, a rendszer nem teszi lehetővé a validációs metódusok módosítását, törlését, illetve valamely új metódus manuális bevezetését a felhasználók számára. A meglévő metódusok a `GET /api/methods` REST API végponton keresztül kérhetőek le. A rendszer kialakításának köszönhetően az új validációs metódusok integrálása egymástól függetlenül megvalósítható, a már meglévő implementáció módosítása nélkül.

Fejlesztés során egy validációs metódus került bevezetésre, a QR-kód alapú validáció. Azon események aktivvá tétele esetén, amelyek mellé ez a validációs metódus társul, a rendszer minden alkalommal egy új kódot generál. Ez a kód egy 4-es verziójú UUID[20], ami egy véletlenszerű és 128 bit nagyságú érték. [21] A végső QR-kód felépítéséért a kliens alkalmazások felelősek – erről bővebben a 4. fejezet ír.

[20]Universally Unique Identifier

Események kezelése

Az eseményekhez tartozó műveletek REST API végpontok segítségével kezelhetőek. A `/api/sessions` végponton keresztül érhető el minden olyan funkcionalitás amely az események kezelésére vonatkozik. Az események létrehozására irányuló kérések során mindig ellenőrizve van, hogy az adott felhasználó rendelkezik-e létrehozói jogosultsággal azon esemény csoporton belül, ahol az adott eseményt kívánja létrehozni. Az esemény létrehozása során a rendszer vizsgálja bizonyos előfeltételek meglétét, például az esemény csoporton belüli címének egyediségét és a legalább egy kiválasztott validáció metódust.

A létrehozás után az esemény alapértelmezetten inaktív állapotba kerül. A felhasználók számára fenntartott `PUT /api/sessions/:sessionId/active` végpont lehetővé teszi az események elérhetőségét jelző attribútumának módosítását. Továbbá a felhasználók lekérhetik az adott eseményhez tartozó validációs metódusok információit, viszont ezen esetben a szerver különböző adatokat szolgáltat, a felhasználó jogainak megfelelően. Azok számára, akik nem rendelkeznek létrehozói jogosultsággal, csak az eseményhez kapcsolódó validációs metódusok listája érhető el. Ezzel szemben az esemény létrehozója hozzáférést kap a validációs metódusok titkos információihoz is, mint például a QR-kódhoz.

Az *Eseménykezelő* az eseményekhez tartozó validációs metódusok titkos adatainak megszerzésére irányuló kéréseket egy üzenetsoron keresztül fogadja. Ilyen adat például a UUID, amely alapján az eseményhez tartozó QR-kód generálódik. Ez a kommunikáció a RabbitMQ bróker által biztosított `sendAndReceive` szinkron kommunikációs módszer által van megvalósítva, amely keretén belül a RabbitMQ bróker automatikusan létrehoz egy ideiglenes válasz-üzenetsort, ahova az *Eseménykezelő* visszaküldheti a válaszát a kérés feldolgozása következtében. Ezen kommunikációs csatorna felhasználója a *Validációs szolgáltatás*, amely egy adott jelenlétrögzítési kérelem ellenőrzéséhez kéri le a megfelelő esemény adatait.

3.4. Jelenlétkezelő

Minden *esemény* entitáshoz egy jelenléti lista kapcsolódik, melyeknek kezelését a *Jelenlétkezelő* mikroszolgáltatás látja el. Az architektúrában elfoglalt helyét a 7. ábra szemlélteti.

Egy eseményhez tartozó jelenléti listák műveletei a `GET/POST/DELETE /api/sessions/:sessionId/attendances` REST API végponton keresztül érhetőek

7. ábra. A *Jelenlétkezelő* architektúrában elfoglalt helye. A kliensalkalmazások felhasználói kéréseit HTTP REST API-n keresztül fogadja. Az eseményekhez tartozó jelenléti listák tárolásáért felelős. Üzenetsorokon keresztül kommunikál a rendszer többi szolgáltatásával, ebből nyeri ki a felhasználókkal és eseményekkel kapcsolatos információkat és szolgáltatja egy eseményekhez tartozó jelenléti listát.

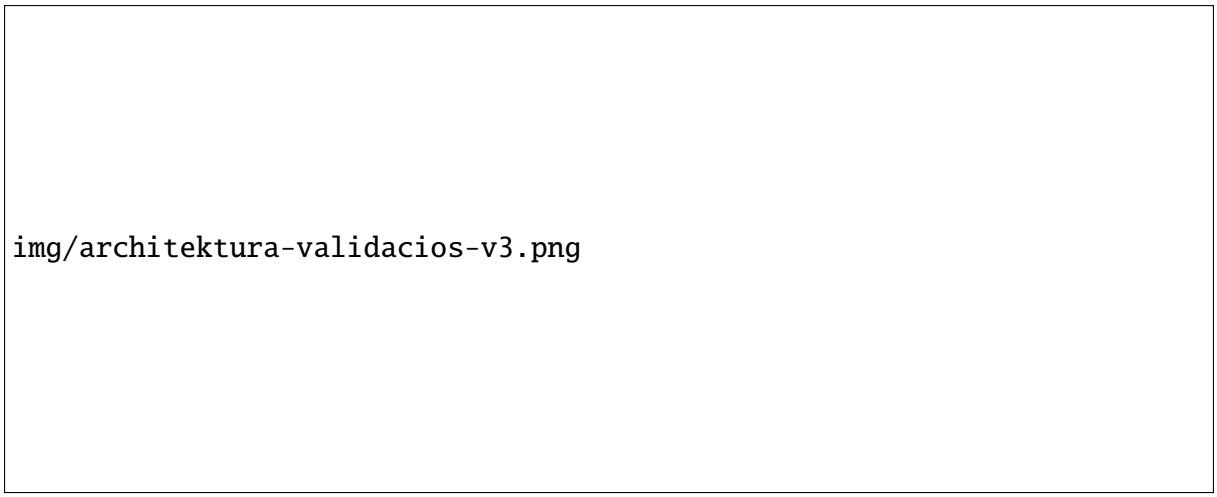
el a felhasználók számára. Minden API híváskor, amely a megadott végpontra érkezik, a rendszer ellenőrzi, hogy a felhasználó az adott esemény tulajdonosa-e, biztosítva ezzel, hogy a más felhasználók által létrehozott események jelenlét listái ne legyenek manipulálhatóak.

A résztvevőként jelenlétüket rögzíteni kívánó felhasználók nem közvetlenül a *Jelenlétkezelő* mikroszolgáltatással kommunikálnak; rögzítési próbálkozásaikat a *Validációs szolgáltatás* ellenőrzi, majd sikeres validáció esetén a jelenlétük rögzítéséhez szükséges adatokat egy RabbitMQ gyűjtőpontba továbbítja. Ezt követően a *Jelenlétkezelő* egy dedikált üzenetsoron keresztül fogadja és dolgozza fel a *Validációs szolgáltatástól* érkező üzeneteket.

Az események létrehozói jogosultak a jelenléti listák lekérdezésére, amely során hozzáférhetnek a listában szereplő felhasználók adatmodelljének megfelelő információkhoz, többek között az e-mail-címekhez és felhasználók teljes nevéhez. Ezen felül a *Jelenlétkezelő* egy dedikált üzenetsort biztosít a *Canvas adapter* számára, amely lehetővé teszi egy esemény jelenléti listájának lekérdezését az adatok a *Canvas LMS* rendszerbe való exportálása során.

A jelenléti lista módosítása

Az eseményeket létrehozó felhasználók két fő módosítási műveletet hajthatnak végre a jelenléti listákon: egy felhasználót hozzáadhatnak a listához, valamint egy már hozzáadott



8. ábra. A *Validációs szolgáltatás* architektúrában elfoglalt helye. A felhasználók jelenlétrögzítési kéréseit HTTP REST API-n keresztül fogadja. Nem rendelkezik saját adatbázissal, ezért a kérések kiszolgálása érdekében üzenetsorokon keresztül kommunikál a rendszer többi szolgáltatásával. Ezáltal nyeri ki a jelenlétrögzítési kérések validáláshoz szükséges információt.

felhasználó eltávolíthatnak. A jelenléti listához történő felhasználók hozzáadását, illetve törlését a már említett REST API végponton keresztül küldött HTTP `POST`, illetve `DELETE` kérés révén hajtják végre.

A *Jelenlétkezelő* felelős az adott eseménycsoportba tartozó felhasználók listázáért is, akiket az esemény létrehozója hozzáadhat egy esemény jelenléti listájához. Ezen funkcionalitás a `GET /api/session-groups/:sessionGroupId/users` REST API végponton keresztül érhető el.

3.5. Validációs szolgáltatás

A *Validációs szolgáltatás* architektúrában elfoglalt helyét a 8. ábra szemlélteti.

A *Validációs szolgáltatás* egyetlen funkcionalitása az eseményekhez tartozó validációs metódusokra tett kísérletek ellenőrzése. A jelenlétüket rögzíteni kívánó felhasználók ezen szolgáltatás által nyújtott `POST /api/sessions/:sessionId/validation` REST API végpontra küldik el a jelenlétrögzítési kísérleteiket. Mivel egy eseményhez többféle validációs metódus is tartozhat, az ide érkező kérések tömbként tartalmaznak információkat minden metódusról a kérés törzsében. Az ellenőrzési folyamat során a *Validációs szolgáltatás* szinkron kommunikációt folytat az *Eseménykezelővel* üzenetsorok segítségével, amelyből lekéri az eseményhez tartozó validációs metódusok listáját és az adott felhasználó jogosultságát az esemény rögzítésére.

A validációs folyamat központi része a `ValidateAggregator` osztály, amely összefogja egy eseményhez tartozó összes validációs metódus ellenőrzési folyamatát és egy összesített választ ad a validációs folyamat eredményéről. A fejlesztése során a QR-kód alapú metódus lett implementálva, de a rendszer kialakítása hozzájárul az újabb metódusok zökkenőmentes integrálhatóságához.

A validációs metódusok jellemzői

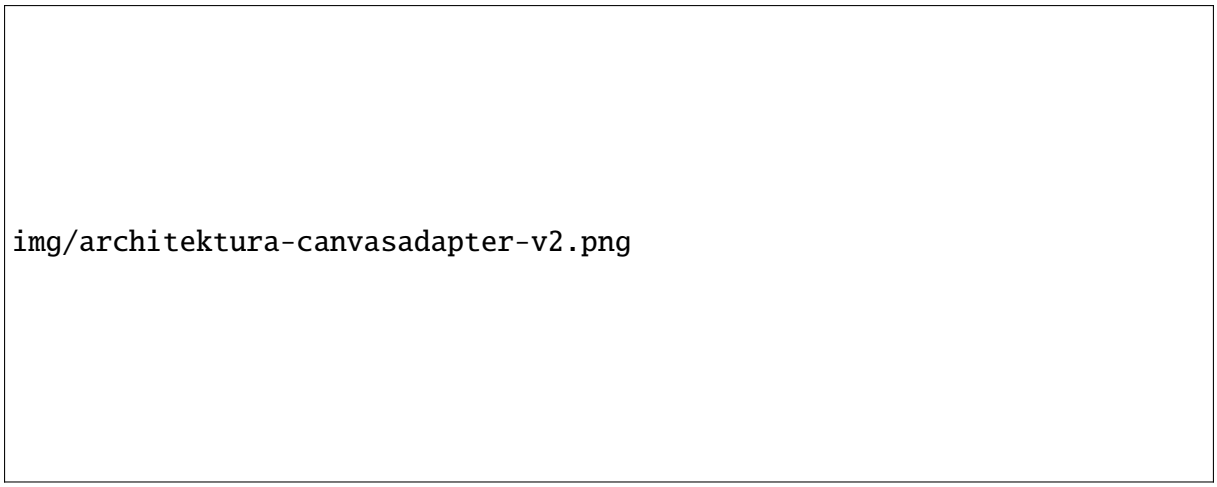
Mindegyik bevezetett validációs metódus rendelkezik egy rendszer szinten egyedi, `shortcode` attribútummal. A felhasználók által küldött kérésekben és az *Eseménykezelőtől* kapott információkban egyaránt, minden validációs metódushoz tartozó titkos információ egy kulcs-érték párokat tartalmazó szótár (map) adatszerkezetben kerül továbbításra. Ezáltal az eltérő validációs metódusok különböző specifikus paraméterei egységes módon kerülnek feldolgozásra. A QR-kód alapú ellenőrzés esetében ezen titkos információ egy UUID-ból áll, amelyet az *Eseménykezelő* generál újra az esemény minden aktiválásakor.

A metódusok ellenőrzési folyamata

A `ValidateAggregator` osztály kulcsszerepet tölt be az ellenőrzés folyamatában azáltal, hogy feldolgozza a két irányból érkező információkat: a felhasználó kérését és az *Eseménykezelőtől* kapott adatokat. Minden validációs metódus egy saját `Validator` osztállyal rendelkezik, amely az adott metódushoz tartozó ellenőrzési logikát implementálja. A QR-kód ellenőrző metódus esetén a `CodesEqualMethodValidator` validációs osztály a két titkos kód pontos egyezését kell vizsgálja. Ezen osztályok kötelezően megvalósítják a `MethodValidator` interfészt, amely előírja az `isMatching` metódus implementálását. Ez a metódus két `Map` objektumot kap paraméterként: a felhasználó által küldött titkos információkat és az *Eseménykezelő* által szolgáltatott adatokat.

Az ellenőrzési folyamatot tekintve a `MethodValidatorProvider` osztály szolgáltatja minden egyes metódushoz, annak shortcode-ja alapján, a hozzá tartozó `Validator` osztályt. Ezáltal egy új validációs metódus bevezetése a rendszerbe, pusztán egy megfelelő `Validátor` osztály implementációjával és annak a `MethodValidatorProvider`-be való elhelyezésével jár.

A `ValidateAggregator` osztály az ellenőrzésre váró validációs metódusok tömbjén végighaladva, lekéri az egyes validációs metódusokhoz tartozó validációs osztályt, majd az



9. ábra. A *Canvas adapter* architektúrában elfoglalt helye. A kliensalkalmazások felhasználói kéréseit HTTP REST API-n keresztül fogadja. A kérések kiszolgálása érdekében adatbázis műveleteket hajt végre, üzenetsorokon keresztül kommunikál a rendszer többi szolgáltatásával, illetve HTTP REST API-n keresztül intéz további kéréseket a *Canvas LMS* rendszer fele. Tárolja a Canvas tárgyaknak megfelelő eseménycsoportokat a karbantarthatóság és hatékony kommunikáció fenntartása érdekében.

`isMatching` metódus segítségével ellenőrzi a kapott adatok helyességét. Amennyiben a validációs folyamat sikertelen, a felhasználó tájékoztatást kap minden egyes validációs metódus eredményéről; sikeres validáció esetén egy megerősítő választ kap és az adatai elküldésre kerülnek a *Jelenlétkezelő* szolgáltatáshoz.

3.6. Canvas adapter

A *Canvas adapter* mikroszolgáltatás az *AttendEZ* és a *Canvas LMS* rendszerek közötti integrációt kezeli. Ezen szolgáltatás feladatai közé tartozik a felhasználók hitelesítése, a *Canvas LMS* rendszerből származó adatok átalakítása az *AttendEZ* saját modelljére, valamint a jelenléti adatok exportálása a *Canvas LMS* rendszerbe. Ebben a *Canvas LMS* által nyújtott HTTP REST API és OAuth 2.0 protokoll segít. A szolgáltatás saját adatbázissal tart kapcsolatot, ahova a *Canvas LMS*-en keresztül bejelentkezett felhasználókat és azok tokenjeit, illetve a létrehozott eseménycsoportokat tárolja. A kliensalkalmazásokkal HTTP REST API-n keresztül kommunikál, a többi szolgáltatással pedig a RabbitMQ brókeren keresztül. Az architektúrában elfoglalt helyét a 9. ábra szemlélteti.

OAuth 2.0

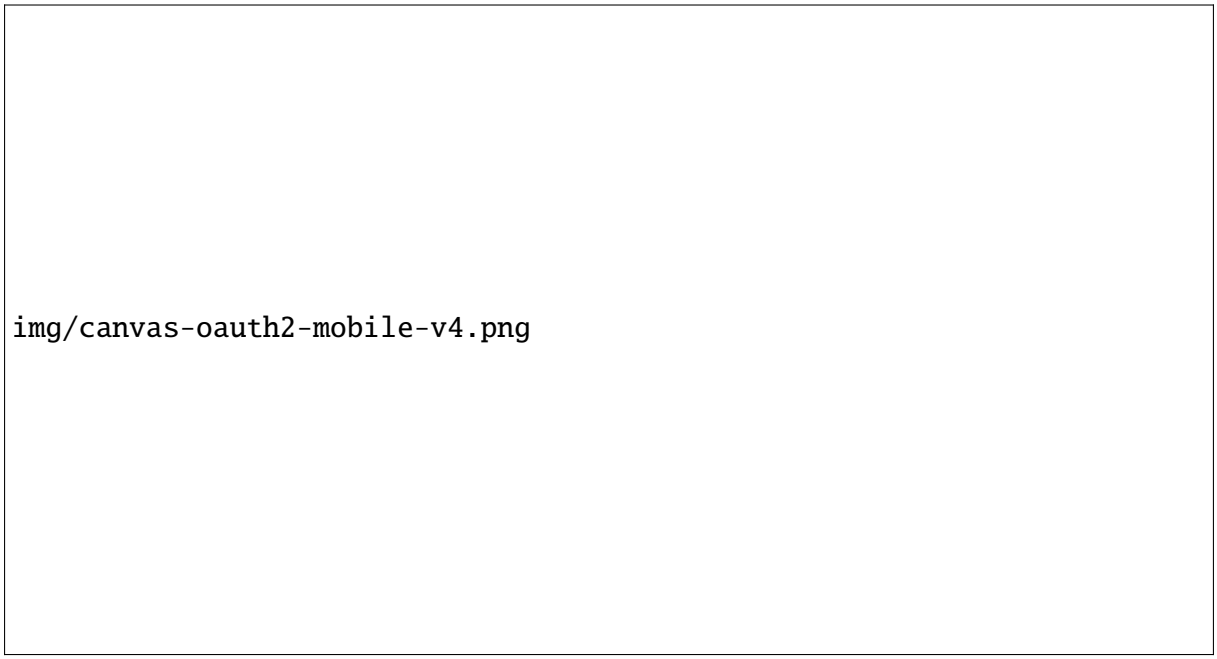
Az OAuth 2.0 egy széles körben elterjedt ipari-standard hitelesítő protokoll. Fejlesztésében a kliensalkalmazásokban való használat egyszerűsége, illetve a különböző hitelesítő folyamatok támogatása kapott hangsúlyos szerepet. A protokoll használatának lényege, hogy egy rendszer felhasználója a jelszava megosztása nélkül egy harmadik féltől származó alkalmazás számára engedélyt ad, hogy a nevében műveleteket hajtson végre olyan erőforrásokon, amelyeket az eredeti szolgáltatás kezel. [3]

Az OAuth 2.0 protokollt számos LMS rendszer és egyéb külső szolgáltató támogatja [17] [5] [16], ezek között a rendszerbe integrált *Canvas LMS* is. E protokoll implementálása megkönnyíti további rendszerek integrálását a meglévő infrastruktúrába.

Az *AttendEZ* rendszer felhasználja a *Canvas LMS* által támogatott OAuth 2.0 folyamatot a felhasználók bejelentkeztetése, illetve a *Canvas LMS*-beli erőforrásaikhoz való hozzáférés megszerzése érdekében. Ehhez a standard alapján szükség van egy rendszerszintű `client_id`-ra és `client_secret`-re, amelyet a projekthez a szerzők intézményének környezetében kitelepített *Canvas LMS* üzemeltetői biztosítanak. A hitelesítési folyamat lényege, hogy a felhasználó megerősítésének eredményeképp a *Canvas LMS* rendszer egy `code` értéket generál, amelyet az *AttendEZ* rendszer bevált egy *hozzáférési* (*access*) és egy *frissítési* (*refresh*) tokenre. A hozzáférési token birtokában a rendszer a felhasználó nevében műveleteket hajthat végre, a frissítési token segítségével pedig újabb hozzáférési tokeneket igényelhet a *Canvas LMS* rendszertől ezek lejárta után, a felhasználó beavatkozása nélkül. Az OAuth 2.0 folyamatról bővebben a [3] cikk tárgyal, a *Canvas LMS* OAuth 2.0 dokumentációja pedig a [20] forrásban érhető el.

A bejelentkezés folyamata a webes és mobilos kliensalkalmazások között helyenként eltér. A webes alkalmazás átirányításokat használ, míg mobil esetén a *Canvas LMS* által felkínált `urn:ietf:wg:oauth:2.0:oob` visszatérítési URI jelzi, hogy nincs szükség átirányításra, valamint a `code` értékét az alkalmazás nyeri ki és dolgozza fel. Ezen folyamatokat a 10. és 11. ábrák szemléltetik.

10. ábra. *Canvas LMS* OAuth 2.0 folyamat web kliens esetén.



11. ábra. *Canvas LMS* OAuth 2.0 folyamat mobil kliens esetén.

Szinkronizációs folyamat

A hitelesítés részeként elindul egy szinkronizációs folyamat is. Ennek keretén belül meghatározásra kerül, hogy a felhasználó adatai léteznek-e már a rendszerben. Amennyiben a felhasználó már létezik, a tokenjei frissítésre kerülnek. Amennyiben új felhasználóról van szó, a rendszer elküldi az új fiók rögzítésére vonatkozó kérelmet a *Felhasználókezelőnek* a RabbitMQ

bróker megfelelő gyűjtőpontján keresztül, sikeres válasz esetén pedig beszúrja az adatokat a saját adatbázisába is, a hozzáférési és frissítési tokenekkel együtt.

Ezután következik az eseménycsoportok szinkronizálása. Meghatározásra kerül, hogy melyek azok a *Canvas LMS* tantárgyak, amelyekben a felhasználó tanári vagy tanársegédi jogosultsággal bír. Ezen tantárgyak képeződnek le eseménycsoportokra a rendszerben. A *Canvas adapter* megállapítja, melyek azok a tantárgyak, amelyeknek megfelelő eseménycsoport még nem létezik. Ezekhez lekérdezi a felhasználókat is, ezek közül pedig kiszűri azokat, akik szerepelnek az *AttendEZ* rendszerben. Nekik meghatározza a megfelelő résztvevői vagy létrehozói jogosultságokat is az eseménycsoporton belül, a tantárgyban betöltött szerepüknek megfelelően. Az eseménycsoportok és jogok létrehozására a RabbitMQ megfelelő gyűjtőpontján keresztül üzenetet küld az *Eseménykezelőnek*, a válaszok alapján pedig hozzáadja a létrehozott eseménycsoportokat a saját adatbázisához is.

Jelenléti lista exportálása

Egy lista exportálásának előfeltétele, hogy a *Canvas adapter* lekérdezi a *Canvas LMS* rendszerből az eseménycsoportnak megfelelő azon feladatok listáját, amelyek publikáltak és *igennel* vagy *nemmel* minősíthetőek. Ez a funkció a *Canvas adapter* `GET /api/session-groups/:sessionGroupId/assignments` REST API végpontján érhető el.

Egy jelenléti lista exportálása a `POST /api/session-groups/:sessionGroupId/assignments/:assignmentId?sessionId=:sessionId` végponton keresztül történik. A jelenléti lista a RabbitMQ gyűjtőpontján tett kérésre érkezik válaszként, a *Jelenlétkezelőtől*. A megfelelő modellátalakítás után a *Canvas adapter* mikroszolgáltatás a *Canvas LMS* rendszer fele egyetlen kérésben feltölti a jelenléti listát, a *Canvas LMS* `POST /api/v1/courses/:courseId/assignments/:assignmentId/submissions/update_grades` REST API végpontját használva.

4. Kliens oldali alkalmazások

Az *AttendEZ* projekthez két kliensalkalmazás tartozik: egy webalkalmazás és egy mobilalkalmazás. A webalkalmazás azon felhasználók számára tartalmaz releváns funkcionalitásokat, akik rendelkeznek eseménylétrehozói jogosultsággal legalább egy eseménycsoport keretén belül. A mobilalkalmazás fejlesztése a könnyű hordozhatóság és a különböző validációs metódusok kihasználása miatt jelent előnyt a projekt számára.

4.1. Webalkalmazás

Az AttendEZ webalkalmazás a React[22] nyílt forráskódú keretrendszert használja, amely elősegíti gazdag kliensfelületek (rich client) és dinamikus felhasználói felülettel rendelkező alkalmazások fejlesztését. Az alkalmazás forráskódja TypeScript[21] nyelvben íródik, amely a JavaScript nyelv egy típusosságot biztosító kiterjesztése. Az alkalmazásban használt stilizált komponenseket az Ant Design[22] könyvtár biztosítja.

A webalkalmazás szerkezete különböző oldalaknak megfelelő komponensekből épül fel. Minden felhasználó számára elérhetőek a bejelentkezési és regisztrációs oldalak, ahol az ezekhez szükséges form komponensek találhatóak, illetve a *Canvas LMS* rendszerrel történő bejelentkezési folyamatot indító gomb. Az alkalmazás fő funkcionalitásai közé tartozik a felhasználó által létrehozott események listázása, új események létrehozása és a már meglévő események módosítása, törlése és aktiválása. A *Canvas LMS* rendszerrel bejelentkezett felhasználók számára a webalkalmazás lehetőséget nyújt a jelenléti lista feltöltésésre.

Oldalak

Az alkalmazáson belül több funkcionalitásnak megfelelő oldal különíthető el. Az alapvető oldalak elérésére szolgál a bal oldalon található navigációs menü. Ebben a menüben megtalálható események (*Sessions*) menüpont nyújt lehetőséget a felhasználó által létrehozott események listázására – lásd 12. ábra. Az oldalon minden eseménynek egy `Card` komponens felel meg, amelyen található gomb segítségével a felhasználó elérheti az esemény specifikus oldalát. Az új események létrehozására szolgáló oldal a 13. ábrán látható, amely az eseményeket listázó oldalról érhető el. Egy eseménynek megfelelő specifikus oldalon lehetőség van az adott esemény törlésére, frissítésére, aktiválására és a hozzá tartozó jelenléti lista feltöltésére,

[21] https://www.typescriptlang.org/
[22] https://ant.design/docs/react/introduce/

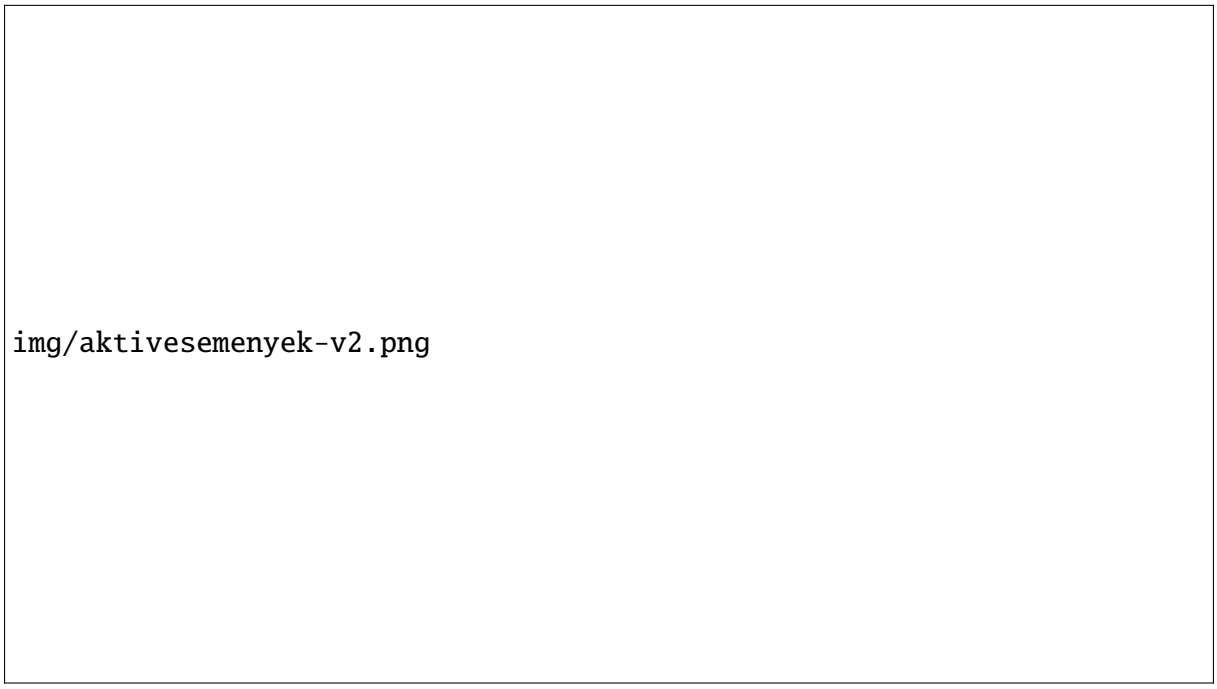
12. ábra. Eseményeket listázó oldal. Az eseményeknek megfelelő `Card` componens jobb felső sarkában levő állapotjezlő aktív események esetén kék, inaktív események esetén szürke színű.

amennyiben az esemény egy *Canvas LMS* rendszerből származó eseménycsoporthoz tartozik. Az aktív események számára az alkalmazás egy különálló nézetet biztosít, ahol a felhasználó elérheti az eseményhez tartozó jelenléti listát, hozzáadhat új résztvevőket, illetve megtekintheti az eseményhez kapcsolódó QR-kódot, az implementált validációnak megfelelően.

QR-kód felépítése

A QR-kód alapú validációs metódus esetében a kliensalkalmazások felelősek a QR-kód kirajzolásáért. A webalkalmazásban QR-kód kirajzolását az Ant Design könyvtár által szolgáltatott `QRCode` komponens biztosítja. A QR-kód mögött egy karakterlánc áll, amely tartalmazza az esemény egyedi azonosítóját (ID), a validációs metódusnak megfelelő `shortcode` attribútumot és az eseményhez tartozó titkos kódot. Ezen részek egy `#` karakterrel történő összefűzése szolgáltatja a QR-kód mögött található egyedi karakterláncot.

13. ábra. Események létrehozására szolgáló oldal.



14. ábra. Aktív esemény oldala, ahol látható az eseményhez tartozó QR-kód és a jelenléti lista.

Oldalak közötti navigáció

Az oldalak közötti navigáció a `react-router-dom` csomaggal van megvalósítva, amely biztosítja a `Routes` és a `Route` komponenseket és a `useNavigation` hookot. A `useNavigation` segítségével valósítható meg az oldalak közötti dinamikus navigáció és az elérési útvonalak megváltoztatása. Minden oldal egyedi elérési útvonallal rendelkezik, amely a `Route` komponens path attribútumaként adható meg. Az oldalak adatait – elérési útvonal, cím, hozzá tartozó komponens, stb. – a `PagesData` konstansokat tartalmazó fájl szolgáltatja. Bejelentkezés nélkül csak a `/login` és `/register` útvonalakon levő oldalak érhetőek el az alkalmazásban. Az alkalmazáson belüli navigáció dinamikusan változik annak függvényében, hogy a bejelentkezett felhasználó milyen jogosultságokkal rendelkezik. A megfelelő útvonalak és hozzájuk tartozó oldalak kiválasztása a felhasználó adatainak `isAdmin` és `canCreateSession` attribútumai alapján történik.

Bejelentkezés

A webalkalmazásban bejelentkezett felhasználó adatainak tárolásáért és kiosztásáért az `AuthContext` globális állapotkezelő eszköz felelős, amely a React által biztosított `createContext` függvény segítségével hozható létre. A bejelentkezés során az alkalmazás elmenti a szerver által kiállított JWT-t a böngésző helyi tárolójába (*local storage*), ahonnan a weboldal újratöltése esetén ezen információ kiolvasható és a szerveroldal által fenntartott végponton ellenőrizhető annak érvényessége. Ezáltal megvalósítható a felhasználó bejelentkezett állapotának megtartása a böngészési munkamenetek között.

Az alkalmazás egy különálló bejelentkezési folyamatot biztosít a *Canvas LMS* rendszer felhasználói számára. Amennyiben ezt a bejelentkezési módot választja a felhasználó, az alkalmazás a *Canvas* által szolgáltatott bejelentkezési oldalra irányítja át. A *Canvas LMS* és az *AttendEZ* szerveroldala közötti sikeres adatcsere után, a szerveroldal átirányítja a folyamatot a `/login-success` útvonalra, ahol a feldolgozásra és tárolásra kerülnek a felhasználó adatai, az *AttendEZ* által biztosított bejelentkezéshez hasonlóan. Ezt követően a felhasznaló automatikusan átirányítódik az alkalmazás főoldalára.

Kommunikáció a szerveroldallal

Az alkalmazás és a szerveroldal közötti kommunikáció az Axios[23] HTTP kliens segítségével történik. Az Axios könyvtár egyik előnye, hogy lehetővé teszi alapértelmezett értékek beállítását több kérés esetén – ezt kihasználva, például a JWT automatikusan társul minden API kérés `Authorization` fejlécéhez a `Bearer` sémának megfelelően, egy hívást megelőző (interceptor) függvény segítségével.

Az API réteghez tartozó metódusok további optimalizációjához a TanStack Query[24] könyvtár által szolgáltatott komponensek kerültek felhasználásra. Ez a könyvtár optimalizálja a gyakran lekért adatok újrahasznosítását és segíti az adatfrissítési folyamat kezelését. Minden kérésnek megfelelően egy egyedi hook került létrehozásra, amely lehetővé teszi a szerveroldaltól érkező hibaüzenetek egységes módon történő kezelését és leképezését az webalkalmazáson belül meghatározott üzenetekre.

4.2. Mobilalkalmazás

Az *AttendEZ* mobilalkalmazás fejlesztése a React Native [22] nyílt forráskódú keretrendszer használatával történik. A React Native lehetővé teszi egyetlen kódbázis felhasználásával, az alkalmazások fejlesztését Android és iOS platformokra. A fejlesztést az Expo[25] keretrendszer is támogatja, amely elősegíti a React Native alkalmazások gyors fejlesztését és publikálását a saját szerverén keresztül. Az alkalmazás funkcionalitásai TypeScript programozási nyelv segítségével kerülnek megvalósításra, illetve a felhasználói felület megjelenítésért a React Native Elements[26] könyvtár felelős, amely biztosítja az előre stilizált komponenseket. A fejlesztés folyamatában előnyt jelent az Expo Go[27] kliensalkalmazás, amelyen keresztül lehetőség van a rendszer mobilkészülékekre történő gyors kitelepítésére.

A mobilalkalmazás felhasználói közé tartoznak az események szervezői és azok, akik jelenlétüket kívánják rögzíteni. Az alkalmazás több összetett szerkezeti egységre épül, mint például a *nézetek* (*screens*), a *komponensek* (*components*) és a *kommunikációs modulok* (*api*), amelyek az alkalmazás belső logikájának működését biztosítják. Az alkalmazás komponensekre való felbontása elősegíti a karbantarthatóságot és ezek újrahasznosítása

[23] https://axios-http.com/docs/intro/
[24] https://tanstack.com/query/latest/docs/framework/react/overview/
[25] https://docs.expo.dev/
[26] https://reactnativeelements.com/
[27] https://expo.dev/go/

csökkenti a redundanciát. A szerveroldallal folytatott kommunikáció a webalkalmazásban implementált megvalósításhoz hasonlóan történik.

Az alkalmazáson belüli nézetek különböző funkcionalitások elérését teszik elérhetővé a felhasználók számára, azok jogosultságainak megfelelően. Alapértelmezetten, az alkalmazás elindításakor a felhasználókat egy bejelentkezési lehetőséget kínáló nézet fogadja. Ezen ablakban található egy bejelentkezési form komponens és egy gomb, amely a *Canvas LMS* bejelentkezési folyamatot indítja el.

A felhasználó adatainak kezelése

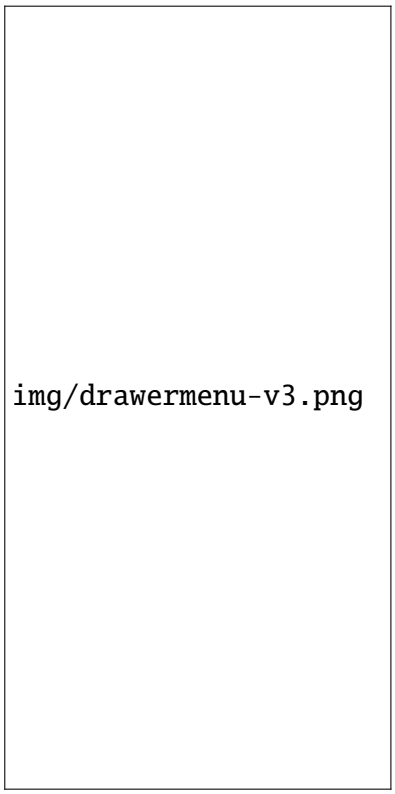
Az alkalmazásban történő bejelentkezés a megadott form kitöltésével történik, ahol a felhasználónak az e-mail-címét és jelszavát kell megadnia. A regisztrációs folyamat eléréséhez a felhasználónak a navigációs menüben levő regisztrációs nézetre kell váltania. Hasonlóan a bejelentkezési nézethez, a felhasználó itt is egy form mezőit kell kitöltse a következő adatokkal: e-mail-cím, kereszt- és családnév, jelszó és annak megerősítése. Az adatok bevitelére szolgáló form komponenseket a Formik[28] könyvtár szolgáltatja, amely validációs sémát is biztosít a formok mezőinek ellenőrzéséhez.

Egy adott felhasználó sikeres regisztrációját vagy bejelentkezését követően szükségszerű a felhasználói adatok alkalmazásszintű tárolása és ezeknek különböző komponensekben történő lekérdezése. A felhasználó adatainak – lásd a JWT személyre szabott állításait az 5. kódrészletben – globális tárolásáért és azok továbbadásáért az `AuthContext` és `AuthProvider` komponensek felelősek. A lekérdezésre nyújt kézenfekvő megoldást a `useContext` React Hook[29], egy olyan állapotmenedzsment eszköz, amely lehetőséget nyújt az adatok globális kezelésére, ezáltal ezen adatok lekérhetőek lesznek az alkalmazás bármely komponensében. A hitelesítési folyamatok helyes végkifejleteként, az alkalmazás hozzájut a felhasználó adataihoz, amelyet a fent említett `AuthContext` helyes beállítása követ.

Nézetek közötti navigáció

Az alkalmazás különböző nézetei között történő navigáció a React Navigation[30] csomag által szolgáltatott `Drawer.Navigation` komponensek felhasználásával valósul meg. A felhasználók számára a nézetek közötti váltást szolgáltató menüpont a képernyő bal felső

[28] https://formik.org/
[29] https://react.dev/reference/react/hooks/
[30] https://reactnavigation.org/



(a) Navigációs menüpontok eseménylétrehozók számára.

(b) A felhasználó által létrehozott események listázása.

(c) A rendszerben található validációs metódusok listázása.

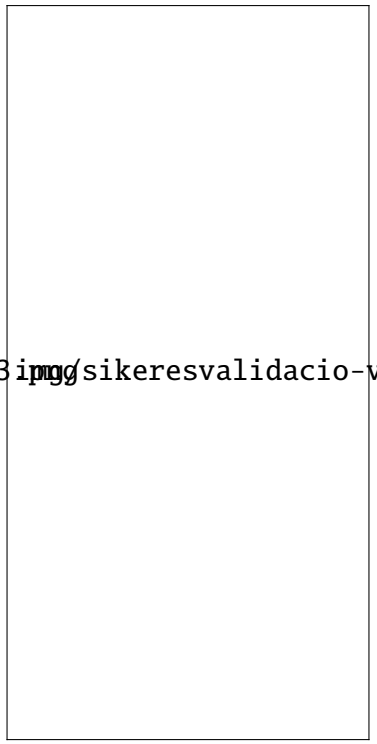
15. ábra. Nézetek közötti navigáció.

sarkában érhető el. Az alkalmazásban elérhető nézetek dinamikusan változnak a felhasználó jogainak megfelelően. Felhasználó fiók nélkül csak a bejelentkezési és a regisztrációs nézetek elérhetőek minden felhasználó számára. Az események létrehozására jogosult személyek számára elérhető egy nézet, ahol az általuk létrehozott események listája (*My Sessions*) jelenik meg és elérhetik bármely eseménynek megfelelő saját nézetet, ahol aktívvá tehetik az adott eseményt. Ez utóbbi nézeten az aktiváláshoz szükséges gomb és az aktív eseményekhez tartozó QR-kód rajzolódik ki – ennek felépítése a 4.1. alfejezetben leírtaknak megfelelően történik. Minden felhasználó elérheti a számára aktív események listáját (*Available Sessions*) és a jelenlétrögzítési nézetet (*Mark Attendance*) – a nézeteket és a nézetek közötti navigációt a 15. ábra szemlélteti.

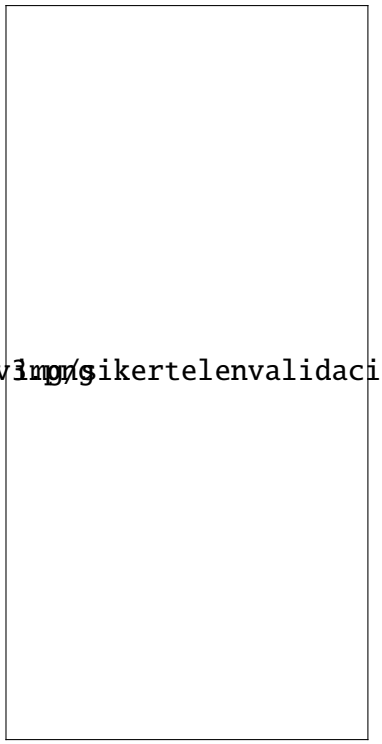
Jelenlétrögzítési folyamat

A résztvevőként jelenlétüket rögzíteni kívánó felhasználók a jelenlétrögzítési menüpont alatt érhetik el a rendszerbe bevezetett validációs metódusok listáját. Minden validációs metódusnak megfelel egy gomb, amely segítségével a felhasználó elindíthatja a jelenlétrögzítési

(a) Kameranézet a QR-kód beolvasására.



(b) Sikeres validáció esetén kapott üzenet.



(c) Hibás kód olvasása esetén kapott visszajelzés.

16. ábra. Jelenlétrögzítési próbálkozás lépései.

folyamatot. Az alkalmazásba bevezetett QR-kód alapú validációs folyamathoz szükséges a felhasználó mobilkészülékén található kamerához való hozzáférés az alkalmazás számára. A kamera használatához az alkalmazás az `expo-barcode-scanner` könyvtár komponenseit használja. A szükséges engedélyek megadása után a felhasználó automatikusan átirányítódik egy olyan nézetre, ahol a mobilkészüléke hátlapi kamerájának képe látható. Ennek segítségével beolvashatja egy adott esemény létrehozója által felmutatott QR-kódot, majd egy másik, automatikusan megjelenő nézet keretén belül láthatja a próbálkozása sikerességét vagy a felmerülő hiba okát – a folyamat során megjelenő nézetek szemléletetésére szolgál a 16. ábra.

5. Felhasznált technológiák és eszközök

Ez a fejezet az *AttendEZ* projekt fejlesztéséhez használt technológiák bemutatásával foglalkozik, amelyek lehetővé teszik a mikroszolgáltatás alapú architektúra megfelelő fejlesztését és hatékony működését.

Java és Gradle

A szerveroldali mikroszolgáltatások az Open Java Development Kit (röviden OpenJDK) 17-es verziója[31] alatt kerültek implementálásra. Emellé a projekt a Gradle projektépítő eszközt[32] alkalmazza, amelynek alprojekteket való támogatása kiaknázza a mikroszolgáltatások monorepóban való elhelyezését. Mindegyik szolgáltatásnak megfelel egy alprojekt, illetve létezik egy `common` alprojekt is, amely az közös, elengedhetetlen függőségeket tartalmazza. A szolgáltatásokat együttesen a gyökérprojektből, illetve individuálisan is kezelni, építeni és futtatni lehet.

Spring Boot Starter

A Spring Boot[33] egy népszerű keretrendszer a Java alkalmazások gyors és hatékony fejlesztéséhez, amely egyszerűsíti az önálló, produkciókész alkalmazások létrehozását beágyazott szerverekkel. A Spring Boot automatizálja a konfigurációs folyamatot, kezeli az alkalmazásfüggőségeket, és a Starter függőségeken keresztül kész megoldásokat kínál, amelyekkel gyorsan integrálhatóak a különböző modulok és szolgáltatások – ezáltal a fejlesztők több időt fordíthatnak az üzleti logika megvalósítására és kevesebbet a konfigurációra.

A projekt a Spring Boot Starter keretrendszer több komponensét is felhasználja, hogy a mikroszolgáltatások hatékonyan és környezetfüggetlenül működhessenek. Ilyen alapkomponensek például a `spring-boot-starter-web`, amely a webes, HTTP protokollon keresztül kommunikáló alkalmazások alapvető szolgáltatásait biztosítja, a `spring-boot-starter-security` a biztonsági keretrendszer integrálásához, a `spring-boot-starter-oauth2-resource-server` az OAuth 2.0 erőforrás-szerver funkcionalitásokhoz. Ezenkívül a projekt specifikus szükségleteihez illeszkedően a `spring-boot-starter-data-jpa` az adatkezelést egyszerűsíti a JPA használatával,

[31] https://openjdk.org/projects/jdk/17/
[32] https://docs.gradle.org/current/userguide/userguide.html
[33] https://spring.io/projects/spring-boot

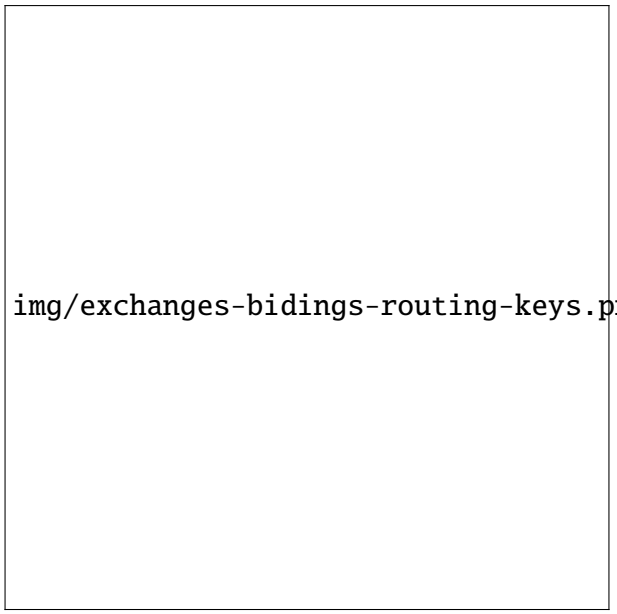
a `spring-boot-starter-validation` az adatok validálását segíti elő, valamint a `spring-boot-starter-actuator`-on keresztül a mikroszolgáltatások egészségi állapota és teljesítménye monitorizálható. Az üzenetközvetítési képességek a `spring-boot-starter-amqp` integrálásával vannak biztosítva, amely a RabbitMQ-val való együttműködést támogatja.

Docker és Docker Compose

Az alkalmazás architektúrájának egyik alapvető eleme a Docker[34] és a Docker Compose[35] technológiák használata, amelyek lehetővé teszik a mikroszolgáltatások izolált és egységesített fejlesztési és üzemeltetési környezetének megvalósítását. A `Dockerfile`-ok, amelyek a különböző mikroszolgáltatásokhoz készülnek, többfázisú építési folyamatot követnek. Az első fázisban a Gradle konténerképet használják a szükséges függőségek letöltésére és a forráskód fordítására. Ez a folyamat biztosítja, hogy minden szükséges komponens, beleértve a közös modulokat is, bekerüljön a végső futtatható állományba. A fordítás eredményeként létrejött `JAR`[36] állomány a következő fázisban egy Java Runtime Environment-et tartalmazó konténerbe másolják, ahol a kész alkalmazás futtatása történik.

A Docker Compose definiálja és kezeli a mikroszolgáltatások és azok függőségei közötti kapcsolatokat egyetlen, összetett alkalmazás részeként. A mikroszolgáltatások, a RabbitMQ üzenetbróker, illetve a PostgreSQL adatbázis definíciói a saját `docker-compose.yml` elnevezésű deszkriptor állományukban találhatóak. Ebben a konfigurációban minden egyes szolgáltatás egy külön konténerben fut, amelyek közötti kommunikáció a Docker hálózati beállításai által van kezelve. A projekt gyökere összefogja a különálló szolgáltatásokat egyetlen deszkriptor állományban, ezzel is elősegítve a szerveroldal egységes kezelését. Ennek a megközelítésnek köszönhetően a fejlesztők lokálisan, egyszerű parancsokkal képesek indítani és leállítani az összes szükséges szolgáltatást, amelyek egységesen meghatározott beállítások mellett futnak, ezzel konzisztenciát biztosítva különböző környezetekben való futtatások között.

[34] https://docs.docker.com/
[35] https://docs.docker.com/compose/
[36] Java archive



17. ábra. Közvetlen gyűjtőpont működése a RabbitMQ bróker esetén. Az 1. és 5. pontok szemléletetik az üzenet küldését és fogadását. A 2. pontban szemléltetve a gyűjtőpont, ahova a küldő helyezi az üzenetet. A 3. pont szemlélteti az üzenetsorok gyűjtőponthoz történő kötését. A 4. pontban láthatóak a létrehozott üzenetsorok.

PostgreSQL

A PostgreSQL[37] egy nyílt forráskódú objektum-relációs adatbázis, amely megfelelő rugalmasságot nyújtott az *AttendEZ* projekt architektúrájának felépítéséhez. A PostgreSQL adatbázis kiterjeszthetősége, megbízhatósága és teljesítménye miatt egy ideális választás a mikroszolgáltatás-alapú architektúrához, melyben az entitások modelljei jól meghatározott struktúrákat követnek és nem igényelnek gyakori változtatásokat.

Liquibase

A Liquibase[38] egy nyílt forráskódú adatbáziskezelő eszköz, amely lehetővé teszi adatbázisséma-változtatások nyomon követését, kezelését és automatizálását. A Liquibase használatának megfelelően a mikroszolgáltatások sémáinak változásait a *changelog* fájlok dokumentálják. Ebben a struktúrában a *changelog-master* egy központi jegyzék, amely más *changelog* fájlokat von be az `include` kulcsszó segítségével. A Liquibase segítségével a sémaváltoztatásokat különálló egységekre, *changelog* fájlokra bontjuk, amelyek elősegítik a változtatások könnyű követhetőségét és a hibák esetén történő visszaállíthatóságot. A *changelog*

[37] https://www.postgresql.org/
[38] https://www.liquibase.com/

fájlok platformfüggelten formátumának köszönhetően nem szükséges az adatbázis-specifikus parancsok ismerete, így lehetővé válik a különböző adatbázis típusok közötti migráció is.

RabbitMQ

A RabbitMQ[39] egy nyílt forráskódú üzenetközvetítő rendszer, amely rugalmas és kiterjeszthető üzenetkezelési és kommunikációs megoldást kínál a mikroszolgáltatás alapú rendszer számára. A mikroszolgáltatások gyűjtőpontokat (exchange) hoznak létre, ahova üzeneketet küldenek a megfelelő útvonalcímzéssel (routing key). Az üzenetek fogadói kötések (binding) segítségével megadják a megfelelő gyűjtőpontot, útvonalcímzést és az adott üzenetsort, ahonnan az üzeneteket fogják kiolvasni. Ez a típusú kommunikáció közvetlen gyűjtőpontokat (direct exchange) használ – működését a 17. ábra[40] szemlélteti.

A RabbitMQ rendszer lehetőséget nyújt szinkron kommunikáció megvalósítására is, amely során létrehoz egy automatikus válaszüzenetsort az üzenetet küldő fél számára, ahova a fogadó fél által visszatérített üzenet kerül elküldésre.

[39] https://www.rabbitmq.com/

[40] Forrás: https://www.cloudamqp.com/img/blog/exchanges-bidings-routing-keys.png, utolsó elérés dátuma: 2024-04-29

Következtetések és továbbfejlesztési lehetőségek

Az *AttendEZ* projekt sikeresen válaszol a jelenlétrögzítés kihívásaira az oktatási intézményekben és nagyobb létszámú rendezvényeken. A fejlett technológiai megoldások, mint a QR-kód alapú validációs módszer és az integráció a *Canvas LMS* rendszerrel, lehetővé teszik az adatok pontos és megbízható kezelését, miközben jelentősen csökkentik az adminisztratív terheket. Az *AttendEZ* rendszer rugalmas és testreszabható funkcionalitása biztosítja, hogy minden intézmény saját igényeihez igazíthassa a jelenlétkezelési folyamatokat, ezzel növelve az oktatási eredményeket és javítva a rendezvények szervezésének hatékonyságát. Ezáltal az *AttendEZ* nem csupán technológiai innovációt jelent, hanem hozzájárul a diákok akadémiai elkötelezettségének és a részvételi arányok növelésének támogatásához is. Ezt egy webes, illetve egy mobilalkalmazás támogatja, amelyek hozzáférhetővé teszik a különböző funkcionalitásokat a szervezők és a résztvevők számára egyaránt.

A rendszer továbbfejleszthető több irányban:

- további validációs metódusok implementálása a nagyobb biztonság elérése érdekében – ilyenek például az NFC alapú validáció, az arcfelismerés, illetve a helymeghatározás;
- felhasználókhoz tartozó eszközök számának korlátozása a nagyobb biztonság elérése érdekében;
- a meglévő QR-kód alapú validáció biztonságosabbá tétele a kód periodikus változtatásával, amely nem igényel külső beavatkozást a szervező részéről;
- egy felhasználó több fiókjának összekapcsolása a fiókok identitása által – például, ha egy felhasználó rendelkezik belsőleg regisztrált és *Canvas LMS* fiókkal egyaránt, ezeknek közös lehet az identitása, ami által a felhasználó bármelyik fiókon keresztül lép be, mindkét fiók erőforrásaihoz hozzáférhet;
- egy eseményhez tartozó jelenléti lista különböző formátumokban történő exportálása – például `json`, `csv`, `xlsx`;
- olyan menüpont a résztvevők számára, ahol a múltban látogatott eseményeiket tekinthetik meg;
- további külső rendszerekkel való integráció – például *Microsoft Teams*, *Moodle LMS*.

Hivatkozások

[1] Vincenzo Andrietti. „Does lecture attendance affect academic performance? Panel data evidence for introductory macroeconomics”. *International Review of Economics Education* 15 (2014), 1–16. oldal. ISSN: 1477-3880. DOI: https://doi.org/10.1016/j.iree.2013.10.010. URL: https://www.sciencedirect.com/science/article/pii/S1477388013000613.

[2] Auth0. *JSON Web Key Sets*. URL: https://auth0.com/docs/secure/tokens/json-web-tokens/json-web-key-sets (elérés dátuma: 2024. ápr. 25.)

[3] Gbadebo Bello. *What is OAuth 2.0?* URL: https://blog.postman.com/what-is-oauth-2-0/ (elérés dátuma: 2024. ápr. 29.)

[4] Balazs Benyo és mások. „Student attendance monitoring at the university using NFC”. *Wireless Telecommunications Symposium 2012*. IEEE. 2012, 1–5. oldal.

[5] Blackboard. *OAuth2 User and Admin document*. 2024-05-03. URL: https://help.blackboard.com/sites/default/files/documents/2018-10/OAuth2%20User%20and%20Admin%20document.pdf.

[6] Askar Boranbayev, Mikhail Mazhitov és Rinat Yamalutdinov. „Managing User Accounts Across Heterogeneous Information Systems In The University”. *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering és Applied Computing (WorldComp). 2013, 1. oldal.

[7] Bouchrika, Imed. *51 LMS Statistics: 2024 Data, Trends & Predictions*. 2024. URL: https://research.com/education/lms-statistics/ (elérés dátuma: 2024. máj. 4.)

[8] Eleri Bowen és mások. „Improving the quantity and quality of attendance data to enhance student retention”. *Journal of Further and Higher Education* 29.4 (2005), 375–385. oldal.

[9] Donald C. Richter és Doris M. Munson. „A Formal Research Study on Correlating Student Attendance Policies to Student Success”. 2013. URL: https://api.semanticscholar.org/CorpusID:56101188.

[10] Egnyte. *What is User Management?* 2022. URL: https://www.egnyte.com/guides/governance/user-management (elérés dátuma: 2024. ápr. 14.)

[11] Heimdal Security. *Policy Based Access Control: What Is It And How Does It Work*. https://heimdalsecurity.com/blog/policy-based-access-control/. 2023. (Elérés dátuma: 2024. ápr. 22.)

[12] JWT.io. *JSON Web Tokens Introduction*. URL: https://jwt.io/introduction/ (elérés dátuma: 2024. ápr. 25.)

[13] James Lewis és Martin Fowler. „Microservices: a definition of this new architectural term”. *MartinFowler.com* 25.14-26 (2014), 12. oldal.

[14] Samuel Lukas és mások. „Student attendance system in classroom using face recognition technique”. *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2016, 1032–1035. oldal.

[15] Fadi Masalha és Nael Hirzallah. „A students attendance system using QR code”. *International Journal of Advanced Computer Science and Applications* 5.3 (2014).

[16] Microsoft. *Microsoft identity platform and OAuth 2.0 authorization code flow*. URL: https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-auth-code-flow (elérés dátuma: 2024. máj. 3.)

[17] Moodle. *OAuth 2 API*. URL: https://docs.moodle.org/dev/OAuth_2_API (elérés dátuma: 2024. máj. 3.)

[18] Mozilla Developer Network. *HTTP Authentication: Bearer*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication#bearer (elérés dátuma: 2024. ápr. 28.)

[19] Loretta Newman-Ford és mások. „A large-scale investigation into the relationship between attendance and attainment: a study using an innovative, electronic attendance monitoring system”. *Studies in Higher Education* 33.6 (2008), 699–717. oldal. DOI: 10.1080/03075070802457066. URL: https://doi.org/10.1080/03075070802457066.

[20] *OAuth2 - Canvas LMS REST API Documentation*. URL: https://canvas.instructure.com/doc/api/file.oauth.html (elérés dátuma: 2024. ápr. 27.)

[21] Oracle. *UUID - Java Platform SE 17*. URL: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/UUID.html#randomUUID() (elérés dátuma: 2024. ápr. 22.)

[22] M. Sakhniuk és A. Boduch. *React and React Native: Build cross-platform JavaScript and TypeScript apps for the web, desktop, and mobile*. Packt Publishing. ISBN: 9781805126874. URL: https://books.google.ro/books?id=QxAEEQAAQBAJ (elérés dátuma: 2024. máj. 3.)

[23] Spring Boot. *Messaging with AMQP - Spring Boot Reference Documentation*. URL: https://docs.spring.io/spring-boot/reference/messaging/amqp.html (elérés dátuma: 2024. ápr. 22.)

[24] Spring Security. *BCryptPasswordEncoder - Spring Security*. URL: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html (elérés dátuma: 2024. ápr. 25.)

[25] TechTarget. *RFID (radio-frequency identification)*. URL: https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification (elérés dátuma: 2024. ápr. 17.)

[26] Mutammimul Ula és mások. „A new model of the student attendance monitoring system using RFID technology". *Journal of Physics: Conference Series*. 1807. kötet. 1. IOP Publishing. 2021, 12026. oldal.

[27] Bruno Zorić és mások. „Design and development of a smart attendance management system with Bluetooth low energy beacons". *2019 Zooming Innovation in Consumer Technologies Conference (ZINC)*. IEEE. 2019, 86–91. oldal.