
UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Sapi3D tour – UI

PROIECT DE DIPLOMĂ

Coordonator științific:
Ș.l.dr.ing. Szántó Zoltán

Absolvent:
Nagy-Serbán Tünde

2021

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA Facultatea de Științe Tehnice și Umaniste din Târgu Mureș Specializarea: Calculatoare		Viza facultății:
LUCRARE DE DIPLOMĂ		
Coordonator științific: Ș.I. dr. ing. Szántó Zoltán	Candidat: Nagy-Serbán Tünde Anul absolvirii: 2021	
<p>a) Tema lucrării de licență: Sapi3D Tour – UI</p> <p>b) Problemele principale tratate: Această lucrare face parte dintr-un proiect mai mare, care are ca scop realizarea unui model 3D virtual a Universității Sapientia din Tg Mureș. În lucrarea de față accentul se pune pe dezvoltarea unei pagini web în care modelului 3D poate fi accesat. Prin această pagină utilizatorul poate să viziteze și să interacționeze cu principalele puncte de interes ale universității. Pentru a facilita utilizarea sistemului am optat pentru integrarea modelului dezvoltat într-o aplicație web. Pe lângă prezentarea structurală a universității, această pagină are menirea de a ajuta, de a îndruma pe cei cu probleme specifice dar destul de comune, cum ar fi orarul de la secretariat, unde se află laboratorul 243, sau care sunt principalele evenimente organizate în următorul timp etc.</p> <p>c) Desene obligatorii:</p> <ul style="list-style-type: none"> - Schema arhitecturii sistemului întreg - Schema arhitecturii din partea modelului <p>d) Softuri obligatorii:</p> <ul style="list-style-type: none"> - Proiectarea și implementarea interfeței cu utilizator - Proiectarea și executarea animațiilor - Integrarea modelului dezvoltat în aplicația web <p>e) Bibliografia recomandată:</p> <ol style="list-style-type: none"> 1. https://vuejs.org/v2/guide/ 2. https://vuetifyjs.com/en/getting-started/quick-start/ 3. Game Engine Architecture https://www.taylorfrancis.com/books/9781466560062 4. An Introduction to Unreal Engine 4, https://www.taylorfrancis.com/books/9781498765107 5. Using 3D visualization methods in landscape planning: An evaluation of options and practical issues, https://www.sciencedirect.com/science/article/pii/S0169204615000535 		
<p>f) Termene obligatorii de consultații: săptămânal</p> <p>g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș Primit tema la data de: 31.03.2020 Termen de predare: 27.06.2021</p> <div style="display: flex; justify-content: space-between;"> <div> Semnătura Director Departament Semnătura responsabilului programului de studiu </div> <div> Semnătura coordonatorului Semnătura candidatului </div> </div>		


Declarație

Subsemnata Nagy-Serbán Tünde, absolvent al specializării Calculatoarea, promoția 2021 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea: Târgu Mureș,

Data: 2021.07.05

Absolvent: Nagy-Serbán Tünde

Semnătura..........

Extras

Lucrarea de licență prezintă o aplicație web, care reprezintă turul virtual 3D al Universității Sapiientia, Facultatea de Științe Technice și Umaniste, din Târgu Mureș.

În zilele de azi, oamenilor în mare parte o zi depinde de digitalizare. La cele mai multe persoane se găsește un smartphone, calculator și laptop lângă care se alătură și accesul la internet și în contextul acesta posibilitățile sunt infinite.

Digitalizarea introduce niște lucruri foarte bune pentru omenire. Foarte multe probleme se pot rezolva prin intermediul internetului și al dispozitivelor digitale, cum ar fi: plata facturilor, susținerea relației cu nembrii familiei și nu în ultimul rând cu ajutorul tururilor virtuale se pot ajunge în locuri, unde poate că nu se va ajunge niciodată.

În lucrarea de licență este vorba de o aplicație web, care vizualizează un model 3D făcut după modelul clădirii universității, creând turul virtual al acesteia. Oricine poate participa la acest tur virtual și cu acesta participantul primește o imagine despre interiorul clădirii. Lângă acesta, aplicația are și un scop informativ, deoarece sunt afișate multe informații despre facultate, cum ar fi: evenimente (Sapi-Line-Tracer), numele și informațiile de contact al coordonatorilor de specialitate.

Aplicația are și o funcționalitate care se numește "Du-mă la!", care navighează utilizatorul în mod automat la destinația aleasă și arată și traseul la destinație, cu acestea ajutând pe vizitator. Lângă acesta, utilizatorul are posibilitatea să se plimbe în model și cu acesta se poate cunoaște tot modelul.

Sistemul a fost creat pentru suprafețe web și mulțumită acesteia aproape pe toate genurile de dispozitive se poate folosi, atât pe calculator, pe laptop cât și pe telefon. Aplicația distribuie foarte mult ajutor pentru studenții noi, studenți și profesori în vizită, pentru a cunoaște mai ușor clădirea universității.

Cuvinte cheie: aplicatie web, model 3D, tură virtuală.

Abstract

This disertation presents a web application, which includes the 3D virtual tour of Sapientia Hungarian University of Transilvania, Faculty of Tehnical and Human Sciences, from Târgu Mureş.

Nowadays, people in big part of the day are in the influence of digitalization. Most of the people has at least a smatphone, a pc or a laptop and with these also has the connection to the internet, which gives infinite possibilities.

Digitalization introduces a lot of good features for humanity. Problems can be solved with using the internet and a digital device, for example: paying bills, keeping the relation with relatives and not least with the help of the virtual tours, people can reach places, where they never could be.

This dissertation it is about a web application, which uses and displays a 3D model from the building of the university, creating a virtual tour of it. To this tour can attend anyone and with this, the posibility is given to have a look inside the building. Beside this has an informational purpose, because the application shows a lot of information about the university, for example: events (Sapi-Line-Tracer), or the name and the contact of the specialist coordinators.

The application has a feature with the name “Take me!”, which takes the user to the selected place and also shows the route to reach there, giving a lot of help for visitors. Beside this the user has the possibility to walk anywhere inside the model and with this the user can discover the complete model.

The system was made on web interface and thanks to this it can be used almost on all the devices like on pc, laptop and also on smartphone. The application can give a lot of help for the new students and also for the students and teachers who are only in visit to familiarize in a faster way with the main building of the university.

Keywords: web application, 3D model, virtual tour.

**SAPIENTIA ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK**

**Sapi3D tour – UI
DIPLOMADOLGOZAT**

Témavezető:

**Dr. Szántó Zoltán,
egyetemi adjunktus**

Végzős hallgató:

Nagy-Serbán Tünde

2021

Kivonat

A dolgozat egy webalkalmazást mutat be, amely a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhely-i karának a 3D virtuális túráját foglalja magába.

A mai világban az emberek napjait nagyban befolyásolja a digitalizáció. A legtöbb embernél található legalább egy okostelefon, számítógép, laptop amelyek mellé társul az internet is így a lehetőségek száma végtelen.

A digitalizáció az emberek számára nagyon sok jó dolgot vezet be. Rengeteg problémát tudunk megoldani az internet és a digitális eszközök segítségével, mint például: számlák fizetése, távoli rokonokkal könnyebb a kapcsolattartás és nem utolsósorban a virtuális túrák segítségével el tudunk jutni olyan helyekre ahová nem biztos, hogy az életben lesz lehetőségünk.

A dolgozatban egy webes applikációról van szó amely felhasznál egy 3D modellt az egyetemről létrehozva az egyetem úgy nevezett virtuális túráját. A túrán bárki résztvehet ezáltal betekintést nyerhet az egyetem falai mögé. Ezen kívül informatív jelleggel is rendelkezik, mivel az alkalmazás számos információt megjelenít az egyetemmel kapcsolatban, mint például: különböző események (Sapi-Line-Tracer), szakkoordinátorok nevei, elérhetőségei.

Az alkalmazás ugyanakkor rendelkezik a "Vigyél el!" funkcióval, amely a felhasználó által kiválasztott helyiséghez mutatja meg az odavezető utat, sok segítséget nyújtva az egyetemre látogatóknak. Ezen kívül arra is van lehetőség, hogy a felhasználók saját maguk lépeghessenek az egyetem modelljén így még jobban körbe tudják járni azt.

A rendszer webes felületre készült és ennek köszönhetően majdnem minden eszközön meglehetősen tekinteni, úgy a számítógépen mint a laptopon és nem utolsósorban a telefonon is. Az alkalmazás sok segítséget nyújthat az újonnan érkező egyetemistáknak, vendég diákoknak és a vendég tanároknak is az egyetem fő épületében való eligazodásnál.

Kulcsszavak: webalkalmazás, 3D modell, virtuális túra.

Tartalomjegyzék

1. Bevezető	1
2. Célkitűzések	4
3. Szakirodalom áttekintése	6
3.1. Telkom Egyetem 3D kampusza	6
3.2. Mauritiusi egyetem 3D tûrája	7
3.3. Virtuális rendszerek az Old-Segeberg városházán	7
3.4. Kuba-i Nemzeti Mûvészeti iskola virtuális tûrája	8
3.5. Játéktechnika alkalmazása virtuális tûrák esetén	9
4. Technológiai áttekintés	10
4.1. Adatbázisok	10
4.1.1. SQL	10
4.1.2. NoSQL	11
4.1.3. SQL vs NoSQL	11
4.2. Webes keretrendszerek	14
4.2.1. Angular	15
4.2.2. Vue.js	16
4.2.3. Angular vs Vue.js	16
4.2.4. NodeJs	17
4.2.5. Spring Boot	17

4.2.6. Spring Boot vs NodeJs	18
5. Követelmény specifikáció	20
5.1. Felhasználói követelmények	20
5.2. Rendszerkövetelmények	22
5.2.1. Funkcionális követelmények	22
5.2.2. Nem funkcionális követelmények	25
6. Tervezés	26
6.1. A rendszer architektúra	26
6.2. Webalkalmazás architektúra	27
6.3. A fejlesztéshez használt technológiák	29
6.3.1. Front-end	30
6.3.2. Back-end	30
6.3.3. Adatbázis	32
6.4. Modulok leírása	32
6.4.1. Visitor fontosabb komponensei	33
6.4.2. User fontosabb komponensei	34
6.4.3. Admin fontosabb komponensei	39
6.5. Verziókövetés és projektmenedzsment	40
6.5.1. Bitbucket	40
6.5.2. Trello	40
7. A felhasználói felület bemutatása	42
7.1. A 3D modell megjelenése	42
7.2. Fontos egyetemi helyek megjelenése	44
7.3. Események megtekintése	46
7.4. Bejelentkezés és kijelentkezés	47
7.5. Jelszó hitelesítése	47
7.6. User adatainak szerkesztése	48

7.7. User regisztrálása és törlése	48
7.8. Részlegek megadása és szerkesztése	50
7.9. Szakok megadása és szerkesztése	52
7.10. Egyebek megadása	53
7.11. Események megadása	54
8. Összefoglalás	56
8.1. Továbbfejlesztési lehetőségek	57
8.2. Köszönetnyilvánítás	57
Irodalomjegyzék	57

Ábrák jegyzéke

4.1. 6.000.000 rekord, 50% olvasás, 50% írás esetén	12
4.2. 6.000.000 rekord és 5.000 véletlenszerű olvasás esetén	13
4.3. 6.000.000 rekord 5.000 véletlenszerű frissítés esetén	13
4.4. Webes keretrendszerek Github és Stack Overflow pontozások alapján	15
5.1. A rendszer használati eset diagramja	21
6.1. A teljes rendszer architektúrája	26
6.2. A webalkalmazás rendszer architektúrája	28
6.3. A "vigyél el funkció szekvencia diagramja"	29
6.4. Példa a Spring Initializr használatára	32
6.5. Példa Bearer Tokenre	33
6.6. Front-end-ről érkező jelszó JSON objektum	34
6.7. Jelszó hitelesítés ellenőrzése	35
6.8. Jelszó mentése	36
6.9. Bejelentkezés kód szinten	37
6.10. Példa részleg megadásra kód szinten	37
6.11. Front-end-ről érkező esemény adatokat tartalmazó JSON objektum	38
6.12. Példa esemény megadásra kód szinten	38
6.13. E-mail küldés	39
6.14. Validációs token generálása	39
6.15. Példa a Bitbucket verzió követésére	40
6.16. Példa a Trello projekt menedzsment eszközre	41

7.1. 3D modell megjelenése az alkalmazásban	43
7.2. "Vigyél el" funkció részleges bemutatása	43
7.3. Útvonal megmutatása a bejáratától az Auláig	44
7.4. Fontos egyetemi helyek megjelenése az alkalmazásban	45
7.5. Példa a Villamosmérnöki Tanszék megjelenésére	45
7.6. Út a Villamosmérnöki Tanszékhez a bejáratától	46
7.7. Események megtekintése	46
7.8. Bejelentkezési ablak megjelenése az alkalmazásban	47
7.9. Jelszó hitelesítésére szolgáló űrlap	48
7.10. User adatok szerkesztésére használható rész	48
7.11. User regisztrálására használható rész	49
7.12. User törlésére használható rész	50
7.13. Részleg hozzáadására alkalmas űrlap	51
7.14. Részleg szerkesztésére alkalmas űrlap	51
7.15. Szak hozzáadására alkalmas űrlap	52
7.16. Szak szerkesztésére alkalmas űrlap	53
7.17. Egyebek hozzáadására alkalmas űrlap	54
7.18. Az események hozzáadására alkalmas űrlap	55

Táblázatok jegyzéke

4.1. Előnyök Angular és Vue.js között	16
4.2. Hátrányok Angular és Vue.js között	17
4.3. Előnyök Spring Boot és NodeJS között	19

1. fejezet

Bevezető

A mai gyorsan fejlődő világunkban a digitális eszközök a mindennapok elengedhetetlen részei. Nem sok olyan háztartás van ahol nincs egyáltalán legalább egy telefon, számítógép, laptop, okos tv. Az elmúlt években a digitális világ annyira fejlett lett, hogy lassan már ki sem kell mozdulnunk a házból és mindent eltudunk végezni. Például ki tudjuk fizetni a számláinkat, be tudunk vásárolni. Ezekből is következtethetünk, arra hogy egy jó internet kapcsolattal és egy közepes teljesítményű számítógép mellett már egy életet is letudunk élni.

Az elmúlt két évtizedben a „virtuális múzeum” [1] fogalmának meghatározása a gyors technológiai fejlődés következtében megváltozott. A mai rendelkezésre álló 3D technológiák a virtuális múzeumok esetén már nem csupán a gyűjtemények bemutatása az interneten vagy egy kiállítás virtuális bemutatója panorámás fotózás segítségével. Egy virtuális múzeum nem csak az oda látogatóknak hasznos, hanem már tanórákon is feltudják használni a tanárok mivel nem mindig sikerül elvinni a diákokat egy adott városba így egy ilyen virtuális túra segítségével a tanár betudja mutatni az adott múzeum látványosságait. Sőt a gyerekek ezáltal otthon is eltudnak barangolni más országok, kontinensek múzeumaiba virtuálisan.

Világunk fejlődése próbálja biztosítani, hogy egyetlen ember se maradjon le azon helyekről ahová nem tud eljutni, így egyes múzeumokat, iskolákat, kastélyokat, látványosságokat is betudunk járni otthon a négy fal között a 3D virtuális túrák segítségével.

Több okot is fel lehet sorolni annak érdekében, hogy miért is használatosak ezek a 3D modellekkel megalkotott virtuális túrák. Első sorban az új diákok már otthonról be tudnak nézni az egyetem falai

mögé így amikor elérkeznek az új év kezdéséhez akkor otthonosabban érezhetik magukat. Ez azért történhet meg mert már fogják tudni, hogy mit hol találnak nem kell segítséget kérjenek. Egy másik ok, hogyha az adott egyetem rendelkezik egy ilyen típusú modellel akkor jobban felhívhatja a figyelmet a diákok számára. Ez által az egyetem népszerűsítése is megtörténik. Ez mellet a vendégkutató, vendégtanár vagy akár vendéghallgató is könnyebben eltud igazodni az egyetem főépületében.

A 3D jelentése három dimenzió. Ez egy modell, amely tulajdonképpen matematikailag ábrázol egy háromdimenziós objektumot, amely lehet épület, virág, vagy akár egy ember is. A 3D modelleket széles körben használják az orvostudományban, magasabb szintű gyakorlati és elméleti kompetenciák elérésében. A 3D modellekkel nem csak tárgyakat hanem emberi folyamatokat is le lehet szimulálni.

Sok esetben egy szimulált 3D modell segít egy adott problémát jobban átlátni. Ilyen példa lehet amikor az orvos tudományban [2] nagyon szépen letudják előre játszani a műtéteket így biztosabbak a dolgukba és a kockázati lehetőségeket is csökkenthetik. Egy másik példa lehet egy hajóforgalom szimulációs rendszer [2], amely figyelembe veszi a hajókat, vízfelületet és az időjárási viszonyokat. Ezzel a rendszerrel próbálták megmutatni az olajszennyezést [2] a tengerekben, óceánokban. Ezen modellek segítségével a világ látványosságai elérhetővé válhatnak azon emberek számára is, akik nem jutnak el az eredeti országba, városba, hogy megtudják tekinteni az adott látványosságot.

A számítógépes grafikát [3] sok területen alkalmazzák mint például az interaktív médiatervezésben, 3D túrák készítésében és videojáték iparban. Az elmúlt néhány év során számos technológia jelent meg a 3D virtuális túrák megjelenítésére az interneten. A virtuális túrák a felhasználóknak biztosítják az életszerű 3D környezeteket.

A tour szó jelentése utazás, kirándulás. A mai világban túrázni a virtuális világban is lehet. Egy ilyen virtuális túra célja, hogy fejlettebb szimulációs technikák segítségével a nézők élethű 3D-s képet kapjanak a meglévő helyről. Egy híres példa a Second-Life [3] ahol a felhasználók közötti interakció avatárokon keresztül zajlik. Ezen alkalmazáson belül interaktív 3D tour-ok vannak leképezve.

A 3D és a tour (virtuális túra) szavak összetételéből jön ki a 3D tour (3D virtuális túra) amely azt tükrözi, hogy a virtuális világban tudunk megnézni adott épületeket, kilátásokat, látványosságokat. Habár ezen virtuális 3D modelleken alapuló világ még nem tökéletes de folyamatosan fejlődik. A dolgozatomon belül egy ilyen modelltől lesz szó amely a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhely-i karának egy részét tartalmazza. A projekt két részre van bontva, amely két

államvizsga dolgozatot eredményez a 3D modell és a felhasználói felület. Közös munka a modellel történő attrakciók megoldása. Ilyen attrakció lehet a modellben való mozgás, közlekedés. Ezen dolgozatban a felhasználói felületről lesz szó.

Egy ilyen alkalmazás esetében nem csak maga a modell jelenik meg, hanem rajta kívül számos fontos információ, elérhetőség is. Dolgozatom célja bemutatni a Sapientia Erdélyi Magyar Tudományegyetem 3D Virtual Tour alkalmazás megalkotását, implementálását és nem utolsósorban a felhasznált technológiákat.

2. fejezet

Célkitűzések

A projekt célja, hogy a felhasználók távolról is megtudják nézni az egyetemet. Ez elsősorban az új felvételizőknek és az első éves egyetemistáknak lenne hasznos, mivel ez által már otthon neki foghatnak áttekinteni az egyetem különböző részeit mint például, hogy hol található egy adott tanszék, vagy hol található a dékáni hivatal, vagy hogy hol található a könyvtár. Ilyen alkalmazás hasznos lehet úgy a diákok mint a tanárok, szakkoordinátorok számára, hiszen rengeteg apró de gyakori kérdésben tud segítséget nyújtani. Ugyanakkor egy egyetemen belül sokszor megfordulnak kutatók, vendégtanárok, külföldi diákok (Erasmus) így számukra is hasznos lehet egy ilyen jellegű alkalmazás.

Fontos szempont, hogy a felület legyen elérhető majdnem minden okos eszközön, ezért határoztunk úgy hogy a projekt egy webalkalmazás legyen, mivel ezt ugyan úgy meglehet nézni telefonon, számítógépen, laptopon.

A cél megvalósításának az első lépése az volt, hogy készítsünk egy felületet ahol a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhely-i karának a főépületének a 3D modelljét megtudjuk jeleníteni.

A modell megjelenítése mellett az alkalmazás egyik legfőbb célja megvalósítani egy olyan funkciót, hogy "Vigyél el!". Ez a funkcionalitás azt jelenti, hogy az új egyetemista diák, vendégtanár, vendéghallgató vagy akár vendégkutató bejelöl egy adott helyiséget ahová szeretne eljutni az egyetem területén és az alkalmazás a modell segítségével megmutatja az oda vezető utat.

Az alkalmazásnak szüksége van egy olyan funkcionalitás kivitelezésére is, ahol a felhasználók információkat (tanszékek, szakok, események) tudhatnak meg az egyetemmel kapcsolatban.

Cél az is, hogy az egyetemről egy link gyűjtemény kerüljön be az alkalmazásba amellyel, útbaigazítást végzünk, olyan szempontból, hogyha valakit érdekel akár egy tanszék, hivatal vagy szak, akkor a linkre kattintva több információt tud meg az adott részről.

A hiteles információk, linkek közzététele érdekében szükség van user menedzsmentre. Ez azt jelenti, hogy a bejelentkező felhasználók fogják tudni szerkeszteni, megadni esetleg törölni az egyetemmel kapcsolatos információkat. Ezek mellett ide tartozna a userek közötti menedzselés is, amely által létrejönnének szerepkörök, így biztosítva, hogy a user menedzsment megfelelően fog végrehajtódni.

A bejelentkező személyek számára biztosítani szeretném, a felhasználói adatok biztonságos eltárolását és kezelését is.

Mivel az egyetemre látogatók számára készül és ők még nem ismerik az egyetem keretein belül szervezendő eseményeket sem, így az is a célok közé tartozik, hogy egy eseménynaptárral is bővüljön az alkalmazás. Így az új diákok tisztában lehetnek, hogy milyen események lesznek az egyetem területén.

Szeretnék, egy olyan részt is biztosítani minden felhasználó számára ahol a saját véleményét tudja kifejezni. Ezen véleményeket figyelembe véve szeretném kijavítani az észlelt hibákat.

3. fejezet

Szakirodalom áttekintése

A világhalón való keresés során sok tanulmányt találtam, amely leírja hogy egy virtuális túra egyetemen, múzeumokban, különböző látványosságoknál mekkora befolyásoló képességgel rendelkezik. Megtudhattam, hogy sok egyetemnek van ilyen jellegű túrája más más típusban. Van amelyik egyetem a virtuális túrát képek sorozatában képzelte el, van amelyik panoráma képekben, van amelyik videóban és nem utolsó sorban jönnek azok akik 3D modellekkel valósították meg az egyetemük virtuális túráját. A következő alfejezetekben bemutatok pár hasonló témájú alkalmazást, megközelítést.

3.1. Telkom Egyetem 3D kampusza

Indonézia egyik legnagyobb magánegyeteme a *Telkom University*¹ [4] is a virtuális 3D túrát alkalmazza az új diákok oda vonzására. A túrák tartalmaznak videó és képsorozatokat plusz 3D alapú modelleket is. Tulajdonképpen egy 3D web alapú túrát dolgoztak ki a 3D Vista használatával.

Miközben fejlesztették, ezt a túrát kutatásokat végeztek, hogy mivel lenne jó elkészíteni, más egyetemek milyen technológiákat alkalmaztak hazai területeken. A következő eredmények születtek:

- R F Rahmat előadó a Universitas Sumatera Utara (USU) egyetemen és társai: Az USU egyetem épületeiről információ szolgáltatásokat kivitelezetek.
- Moloo előadó a Mauritius-i egyetemen és társai: virtuális túrát hoztak létre a WebGL és a

¹<https://tour.telkomuniversity.ac.id/>

SketchUp segítségével.

- Fujita a Tokió-i egyetem tagja és társai: mobilis robotokkal készítettek virtuális túrákat.

Szeliski R. aki a fotó technikákat mutatta be [4], szerint is a nagy felbontású fotók és a 3D modellek együttes használatával nagyobb érdeklődési kört lehet elérni, mint ha csak külön használnák őket. Az egyetem kiemelt kutatási stratégia terve volt, hogy ne csak a diákokat vonzzák magukhoz, hanem az IKT (Információs és kommunikációs technológia) technológiák fejlesztésében is érjenek el fejlődéseket. Ennek érdekében használtak 3D modelleket és nagy felbontású fotókat együttesen. A diákok számára nagy érdekességnek számított és ezzel elérték azt, hogy a diákok érdeklődését felkeltették.

3.2. Mauritiusi egyetem 3D túrája

A *mauritiusi*² egyetem [3] is megalkotta saját 3D virtuális túráját. A megalkotást WebGL segítségével történt. A WebGL (Web Graphics Library) könyvtár 3D-s valós idejű megjelenítést kínál. A WebGL az OpenGL-ből származik, és API-t biztosít a 3D grafikához. A WebGL nem teljesen stabil és néhány algoritmus nem hatékonyan látja el a feladatait. Különböző böngészők memóriahasználata és végrehajtási ideje eltérő amelyek mellékhatásokat idézhetnek elő a megírt alkalmazásban.

Az egyetem is szintén valós időbeli megjelenítést alkalmaz és a megfelelő működés érdekében tesztelték az alkalmazást teljesítmény szempontjából, amikor több felhasználó csatlakozik egyidejűleg. A teszt eredményiből az egyetem arra következtetett, hogy minél összetettebb az objektum és ha még textúrával is rendelkezik akkor a teljesítmény nagyon csökken mivel a modell így nagyon nagy erőforrást igényel.

3.3. Virtuális rendszerek az Old-Segeberg városházán

A Németország-i Old-Segeberg [1] város házának is készítettek ilyen virtuális túrát egy Windows alapú interaktív szoftvert és egy virtuális valóság alkalmazást HTC Vive rendszerekkel. Mindkét rendszert kipróbálták a látogatók. A visszajelzések alapján van jövője az ilyen jellegű alkalmazásoknak is.

²<https://www.middlesex.mu/about-mdx-mauritius/campus-virtual-tour>

A virtuális múzeum csúcspontjai a műalkotások pontos ábrázolása.

A Windows alapú interaktív szoftvert egy asztali számítógép segítségével tekinthették meg a felhasználók. A lényege az volt, hogy az oda látogató nem lépett be teljesen a virtuális világba csak kívülről tekintett bele míg a HTC Vive applikáció segítségével már a felhasználó végig ment a város házán virtuálisan. HTC Vive applikáció több interaktivitást nyújt a felhasználóknak viszont mindkét kifejlesztett rendszernek megvannak az előnyei és a hátrányai is. Majdnem minden házban található egy PC számítógép így a Windows alapú rendszert könnyebben népszerűbbé lehet tenni mint a HTC Vive-ot.

3.4. Kuba-i Nemzeti Művészeti iskola virtuális túrája

Az innovatív technológiák új lehetőségeket biztosítottak a kulturális helyszínekről szóló információk gyűjtésére, elemzésére és megosztására. E technológiák közül a gömbszerű képalkotás és a virtuális túrakörnyezet segítségével megalkották a Kuba-i Nemzeti Művészeti iskolát [5] is, pontosabban a Nemzeti Balettiskolát. A virtuális túrán megnézhetőek a tantermek, a fő kupola és az előadó termek. A túrát azért hozták létre mivel az iskola már nagyon régi és rossz állapotban van így ezzel megtudják mutatni az érdeklődőknek, hogy milyen is volt régen az épület.

Ez a virtuális túra kompatibilis számítógépekkel, táblagépekkel és egyéb mobil eszközökkel. A virtuális túrában be vannak építve pdf-ek, linkek amelyek az adott részből bővebb információkat szolgáltatnak a túrázóknak. Összesen 14 gömbpanorámát, 96 hotspotot (műveletek pl. előre lépés, vissza lépés) és 40 linket. Ezen eszközök segítségével biztosítanak a felhasználóknak kényelmes virtuális túrázási lehetőséget. Az képernyőn megjelenik egy menü rendszer is amellyel tudjuk irányítani a túrát.

Ezen tanulmány szerint ennek a túrának a célja az volt, hogy a Balettiskola fennmaradjon a következő nemzedékek számára is legalább virtuális közegbe.

3.5. Játéktechnika alkalmazása virtuális túrák esetén

A virtuális túrák információt nyújtanak multimédiás úton a felhasználóknak, azt a benyomást keltve hogy valós időben navigálnak különböző helyeken. Egy sikeres túra jelentése, hogy a felhasználó el tudja hinni, hogy ő habár virtuálisan is de járt abban az adott helyiségben. Ahhoz hogy ez a benyomás létre jöjjön a modell pontos ábrázolást kell tartalmazzon az adott helyiségről. Ezen tanulmány szerint is az ilyen túrák felhasználhatóak létesítmények népszerűsítésére mivel egy interaktív élményt biztosítanak. Az ilyen jellegű túrákat össze lehet hasonlítani a számítógépes játékokkal is. Hiszen ezek a játékok annyiban különböznek, hogy nem valós időben és nem valós helyeken történnek. Persze vannak kivételek, ahol a játék egy része magába foglal egy valós helyszínt is.

Az Egyesült Királyság [6] számos egyeteme használ ilyen jellegű túrákat annak érdekében, hogy az emberek távolról is betekintést tudjanak nyerni az egyetemek világába. A kutatás szerint tizennégy brit egyetemet tekintve mindegyik weboldalán találtak állóképgalériát és videogalériát viszont meglepő számnak számított hogy 3600 interaktív túrát is találtak. Azért meglepő szám mivel ezek a túrák még nem igazán elterjedtek.

Ezen tanulmány felmérte az Egyesült Királyság egyetemei körében mennyire használatosak a virtuális túrák. Kiderült, hogy a virtuális egyetemi túrák interaktívabbá tették az egyetemek weboldalait és nagyobb fokú szolgáltatást biztosítottak a felhasználók számára hiszen nem csak képeket, videókat láttak az egyetemekről, hanem magát az egyetemet is. Az egyetemek ezen túrák megalkotásában teljes mértékben kihasználták a számítógépes játékokhoz használt grafikai eszközöket. Az modellek segítenek bemutatni az egyetemeket, plusz útvonalakat is tervezhetnek a modellben így már előre fogják tudni, hogy mit hol találnak a felhasználók. A jövőben valószínűleg majdnem minden egyetem fogja alkalmazni az ilyen jellegű túrákat.

4. fejezet

Technológiai áttekintés

A célkitűzésnek megfelelően próbáltam keresni megfelelő adatbázist, keretrendszereket. A következőkben egy pár ilyen jellegű technológiáról lesz szó.

4.1. Adatbázisok

Adatbázisnak nevezzük az azonos jellemzőjű és többnyire strukturált adatok összességét. Az adatbázisokon belül több fajta műveletet tudunk elvégezni [7]: karbantartás, tárolás, lekérdezés, szerkesztés, módosítás és nem utolsósorban az adatok törlése is egy lehetőség. Ezen funkciókat egy adatbázis kezelő rendszer segítségével tudjuk elvégezni. Azonban különbséget kell tennünk az SQL (Structured Query Language) és a NoSQL (Not only Structured Query Language) között.

4.1.1. SQL

Az elmúlt harminc évben a relációs adatbázis volt az alapértelmezett strukturált adatlekérdezési nyelv. Világunk fejlődése miatt egyre több és nagyobb információ adatok robbantak ki így az SQL alapú adatlekérdezés [8] elveszítette hatékonyságát ezzel maga elé állítva azt a kihívást, hogy a nagyobb adatbázisok kezelése jóval megnehezedett. Ebből kifolyólag lehet arra következtetni hogy az SQL alapú szerverek hajlamosak nagy mennyiségű memóriát foglalni, biztonsági kockázatokat és teljesítményproblémákat elkövetni.

4.1.2. NoSQL

A NoSQL adatbázisokat azért alkották meg, mivel az SQL adatbázisok merev struktúrával rendelkeznek aminek következtében, egy adott táblába nem tudunk kihagyni egy oszlopnak a feltöltését sem. Ennek következtében vagy fiktív adatokat, vagy üres mezőket kell megadjunk azokban a mezőkben amelyeket nem akarunk használni. A NoSQL adatbázisok sokkal rugalmasabbak lettek, fő céljuk az adatok könnyű tárolása és visszakeresése függetlenül a szerkezettől és tartalomtól. Automatikusan kezelik az adatkezelést, hibajavítást amelyek költségmegtakarítás szempontjából is fontosak [8].

4.1.3. SQL vs NoSQL

A relációs adatbázisok az egyszerűség miatt leggyakoribb adatbázistípusok. Az adatok több táblára vannak bontva amelyekhez egyszerűen hozzá lehet férni. A CRUD(create, read, update és delete) műveletek nagyon egyszerűen elvégezhetők az SQL által megadott szintaxisok betartásával. Ilyen típusú adatbázis kezelő rendszerek a következők: Oracle, SQL Server, MySQL, PostgreSQL stb. A folyamatos adatmennyiség miatt a relációs adatbázisok hátrányba kerültek a nem relációs adatbázisokkal szemben. A nem relációs adatbázisok sokkal gyorsabban és hatékonyabban tudják elvégezni feladataikat mint a relációsak. Nem relációs adatbázis típusú rendszerek a következők lehetnek [9]: Firebase, MongoDB, GraphQL stb. Sok adat esetén, ha a hatékonyságra törekedünk akkor érdemesebb a nem relációs adatbázisokat használni.

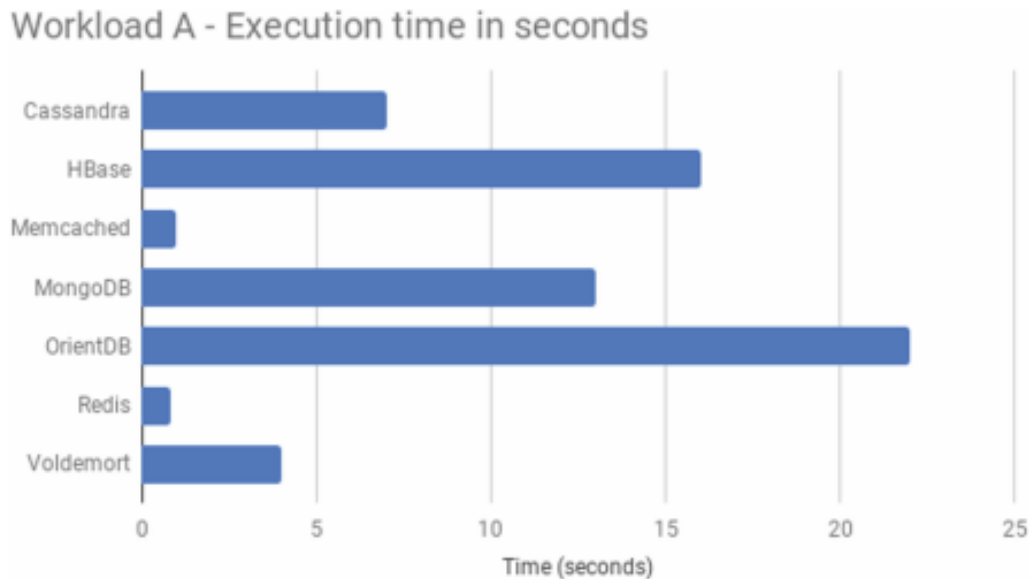
Több tanulmány is szól az adatbázisok teljesítményéről. A következőkben bemutatnék egyet. Ebben a tanulmányban három tesztet végeztek el. A kísérletben szerepelnek a következő NoSQL adatbázis kezelő rendszerek: Cassandra, HBase, Memcached, MongoDB, OrientDB, Redis és a Voldemort [10]. A tanulmányban három tesztet végeztek el. Mindhárom tesztben rendelkezésre állt 6.000.000 rekord. Ez a rekord szám nem is túl sok viszont nem is kevés. A tesztek az adatbázisok teljesítményét tesztelték olvasás, írás és frissítés esetén így elegendőnek bizonyult ennyi adat is. A tesztek időre alapozták, hogy a különböző adatbázisok hány szekundum alatt végzik el a különböző teszteseteket.

Az első tesztben a munkaterhelés fele olvasást és fele frissítést tartalmazott. Az eredmény a 4.1 ábrán látható. Az ábrát tekintve láthatjuk, hogy a legjobb időt a Redis és egy kevéssel lemaradva a Memcached teljesítette. A legrosszabb teljesítményt az OrientDB mutatta.

A második tesztben a munkaterhelés 5.000 véletlenszerű olvasás volt. Az eredmények 4.2 ábrán láthatóak. Az első teszthez hasonlóan itt is a Redis és a Memcached teljesített a legjobban viszont utolsó helyre már a HBase került.

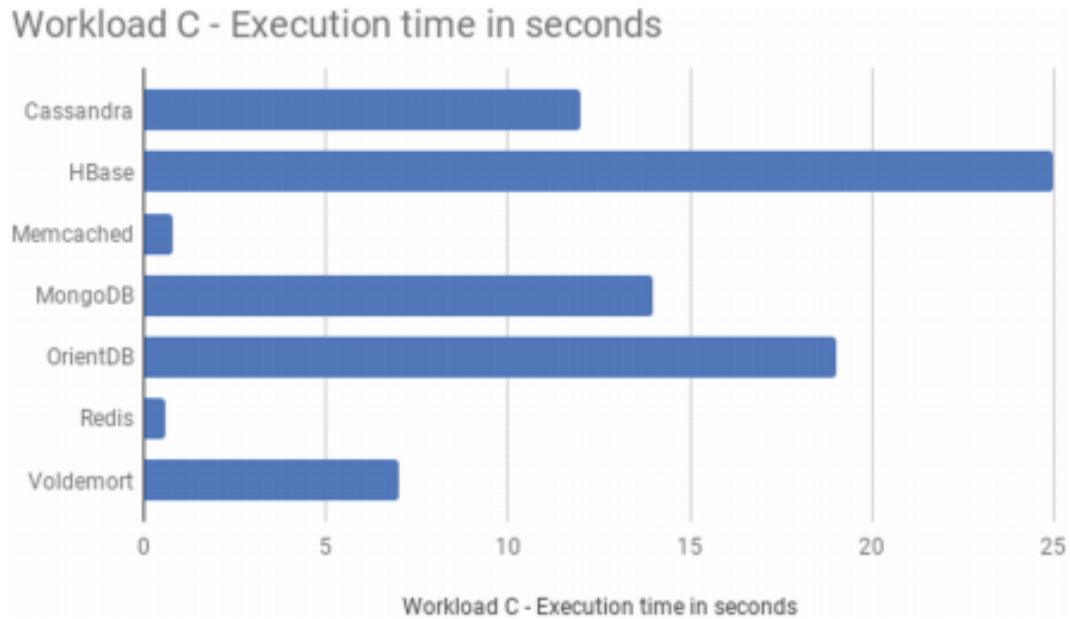
A harmadik tesztben a munkaterhelés 5.000 véletlenszerű frissítés volt. Az eredmények 4.3 ábrán láthatóak. Az első helyeket ismét a Redis és a Memcached szerezte meg míg az utolsó helyre visszakerült az OrientDB.

A tanulmányban leírt teszteket és NoSQL adatbázisokat tekintve a legjobb választás a Redis vagy a Memcached lenne. A legrosszabb választás az OrientDB lenne. A tanulmányban résztvevő többi adatbázis teljesítménye körülbelül megegyező volt, közepes teljesítménnyel, így a tanulmányt tekintve ezeknek a használata is megfontolandó.

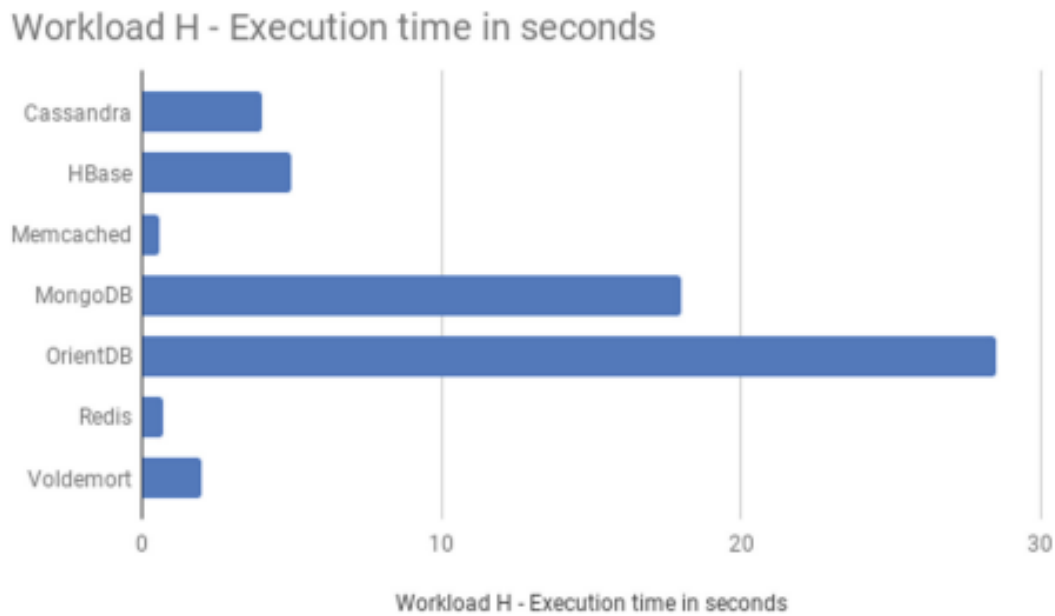


4.1. ábra. 6.000.000 rekord, 50% olvasás, 50% írás esetén¹

¹https://www.researchgate.net/figure/Workload-A_fig1_332028074



4.2. ábra. 6.000.000 rekord és 5.000 véletlenszerű olvasás esetén²



4.3. ábra. 6.000.000 rekord 5.000 véletlenszerű frissítés esetén³

²https://www.researchgate.net/figure/Workload-C_fig2_332028074

³https://www.researchgate.net/figure/Workload-H_fig3_332028074

Az adatbázis kezelő rendszer kiválasztásánál fontos szerepet játszott, hogy a használata egyszerű legyen és a Spring Boot is tudjon kapcsolatot teremteni vele. Mindezek mellett az is fontos szerepet játszott a döntéshozatalban, hogy az adatbázis kezelő rendszernek a használatáért ne kelljen fizetni. Igaz, hogy a fent leírt tanulmányban a Redis adta a legjobb eredményeket, viszont azért a teljesítményért fizetni kell. Ezeknek a szempontoknak a figyelembe vétele után döntöttünk úgy, hogy a MongoDB-t választjuk, mivel a használata könnyű, a Spring Boot is tud vele kapcsolatot teremteni és nem utolsó sorban ingyenes is. Ezek mellett a fent leírt tanulmányban is közepes szintű teljesítményt mutatott és az általunk készített alkalmazás nem fog ekkora mennyiségű adatot feldolgozni.

4.2. Webes keretrendszerek

A keretrendszerek [11] napjainkban felkapott rendszerek lettek a felhasználók között. Ezek a rendszerek sokoldalúak, robusztusak és hatékonyak. A különböző alkalmazások fejlesztéséhez az ilyen jellegű rendszerek segítségével csak a magasabb szintű funkcionalitások elvégzésére kell koncentrálni. Erre az a magyarázat, hogy a keretrendszer gondoskodik az alacsonyabb funkcionalitásokról, amelyek már rengeteg tesztelt kódot tartalmaznak, így mi ezeket a funkcionalitásokat nem kell külön megírjuk és leteszteljük, hogy helyes-e a megírt kód. Számos előnyt lehet felsorolni, hogy miért jó ha használjuk ezeket a rendszereket:

- Elősegíti a tervezési minták megfelelő kialakítását.
- Biztonságosabb kódolás.
- A redundáns kód elkerülése.
- Következetes kódfejlesztés kevesebb hibával.
- Megkönnyíti a kód tesztelését és a hibakeresést is.
- Az alkalmazás fejlesztéséhez szükséges idő lecsökken.

Az 4.4 ábrán látható egy rangsorolás a Github és StackOverflow által készített pontozások alapján a Front-end-et megvalósító keretrendszereknek. A Front-end foglalkozik a kliens felületek megjelenítésével. Az ábrán látható hogy az első helyen a React van összesített 98 ponttal. A második helyen

áll az ASP.NET MVC 95 ponttal viszont itt észlelhető az is, hogy a Githubról nem jött pontozás erről a keretrendszerről. Az ábrán látható további keretrendszerek ugyan annyi összpontozást kaptak. Az ábra alapján a legjobb döntés egy webes keretrendszer választásra a React lenne viszont a többi keretrendszer sincs sokkal lemaradva az első helyezettől.

Framework	Github Score	Stack Overflow Score	Overall Score
React	99	97	98
ASP.NET MVC		95	95
Angular	91	96	93
Ruby on Rails	87	99	93
AngularJS	90	97	93
Vue.js	100	87	93

4.4. ábra. Webes keretrendszerek Github és Stack Overflow pontozások alapján⁴

2021-ben a 10 leghasználatosabb Back-end keretrendszer a Spring - Spring Boot, NodeJS, Laravel, Django, Flask, Ruby, Play, Asp.Net, CakePHP és Symphony [12][13]. A Back-end segít a Front-end-nek egy dinamikus alkalmazás elkészítésében. Ez azt jelenti, hogy a Back-end biztosítja az adatokat a Front-end-nek.

A következőkben olvashatunk részletesebben néhány keretrendszerről mint például: az Angular, Vue.js, Spring Boot és a NodeJs.

4.2.1. Angular

A Google munkatársai 2008-ban fejlesztettek a JavaScript alapú Angular keretrendszert [14]. Abban az időben a webhelyek zöme többoldalas alkalmazás megközelítésén alapult. A több oldalas alkalmazások lassúnak bizonyultak így bevezették az egy oldalas alkalmazásokat. Az egy oldalas alkalmazások abban jobbak a több oldalas alkalmazásoknál, hogy weboldal betöltése folyamán nem kell mindig mindent újra betölteni, hanem csak azok a részek töltődnek be ahol változások fognak megjelenni. Az Angular volt az egy oldalas alkalmazások első kerete. Az egyik fő előnye, hogy a felhasználók egyszerű struktúrával kell dolgozzanak. Megtanulják az Angular sajátos felépítését ezáltal gyorsan és

⁴<http://hotframeworks.com/>

optimálisabban tudnak benne fejleszteni. Az sem elhanyagolható, hogy a fejlesztők egy részletes és egyértelmű dokumentációval szolgáltak a felhasználóknak.

4.2.2. Vue.js

A Vue.js (röviden: Vue) [14] tekinthető az egyik legújabb keretrendszernek. Hasonlít az Angularhoz, mindkét keretrendszer TypeScript típusú. Használható kisebb, egyszerűbb projekteknek. Mindig egy oldalas alkalmazásokat lehet benne elkészíteni ami a felhasználói élményt segíti elő. Fő érdeme a skálázhatóság és különlegessége, hogy egy nyílt forráskódú közösség fejlesztette ki, nem pedig egy nagyobb vállalat, így a problémák megoldására rengeteg segítséget lehet kapni. Komponens alapú keretrendszer amely azt jelenti, hogy komponenseket különböztetünk és jelenítünk meg. Egy komponensen belül tudunk írni HTML, CSS és Script elemeket is. A megjelenítés egy oldalon történik ezért szükséges használni a útválasztást (rout).

4.2.3. Angular vs Vue.js

Mindkét keretrendszernek megvannak az előnyei 4.1 táblázat és a hátrányai is 4.2 táblázat.

4.1. táblázat. *Előnyök Angular és Vue.js között [15]*

	Angular	Vue.js
1	TypeScript használata	TypeScript használata, részletes dokumentáció
2	Részletes dokumentációval rendelkezik	Egy oldalas alkalmazások készítése
3	Gyorsítja a fejlesztést	Könnyű integráció a meglévő struktúrákba
4		Kihasználja virtuális DOM előnyeit
5		Sebessége és rugalmassága optimális

4.2. táblázat. *Hátrányok Angular és Vue.js között [15]*

	Angular	Vue.js
1	Számos különféle struktúrát kínál, nehezíti a tanulást	Kevesebb erőforrást kínál
2	Lassabb teljesítmény mert működik a reális DOM	

4.2.4. NodeJs

A NodeJs [16] egy olyan szoftver platform, amely a Chrome V8 JavaScript futási idején épül fel. Fontos tulajdonsága, hogy skálázható ezért is sokan használják. Eseményvezérelt, nem blokkoló I/O modellt használ, amely könnyűvé és hatékonyá teszi a valós idejű alkalmazások megvalósítását, amelyek elosztott rendszeren futnak át. Tulajdonképpen közvetlenül a natív gépi kódra fordítja le a JavaScript-et, amelynek hatására az adatformátum egy lépése megszűnik, ezzel növelve az alkalmazás sebességét. Legtöbb esetben a NodeJs használatakor adatbázisnak NoSQL adatbázisokat választanak.

NodeJs a következő különlegességeket tartalmazza [17]:

- A NodeJS alkalmazások fejlesztésének elindítása könnyű.
- Agilis fejlesztési módszertant követi.
- Alkalmas a skálázható alkalmazásfejlesztési szolgáltatásokra.
- Nagy projekteknél gyorsabban működik mint a Java.
- Hatalmas könyvtárakkal rendelkezik, amelyek segítik a fejlesztőket.

4.2.5. Spring Boot

A Spring keretrendszer [18] egy programozási és konfigurációs modellt kínál a Java alapú alkalmazásokhoz. A Spring Boot [19] célja a Spring alkalmazás fejlesztés egyszerűsítése. Megtalálhatóak benne a következő tulajdonságok és különlegességek [17]:

- Automatikus konfigurációk - Az alkalmazások Springként való működése érdekében.
- Indítófüggőségek - Biztosítja a felhasználóknak a szükséges függőségek (dependency) beillesztését. Ilyen lehet a Maven, Hibernate validátor, adatbázis elérések stb.
- Parancssori tolmács.
- Működtetés - A console-ba megjelennek az alkalmazás működésével kapcsolatos információk. Ilyen információ lehet a hiba, az elvégzett művelet stb.
- Radikálisan gyorsabb és széles körben hozzáférhető, érthető Spring fejlesztést nyújt.
- Számos funkciót kínál: beágyazott szervereket, metrikákat, ellenőrzéseket, külső konfigurációkat.
- Egyszerű, minden eszköz és operációs rendszer támogatja.
- Beépített nyelvbiztonsági funkciókkal rendelkezik, amelyeket a Java Compiler beágyaz.
- Robusztus kódot alkalmaz.
- Integrációs képessége jó.
- Beágyazott HTTP-kiszolgálókat, például Jetty, Tomcat használ és egyszerűen teszteli a webes alkalmazásokat.

4.2.6. Spring Boot vs NodeJs

A Spring Boot és a NodeJs közötti néhány előnyt [20] az 4.3 táblázatban tudjuk megtekinteni.

4.3. táblázat. *Előnyök Spring Boot és NodeJS között*

	Spring Boot	NodeJs
1	Java nyelv már nagyon ismeretes így ha elakadás van könnyű segítséget kérni/keresni	A JavaScript futásidejének köszönhetően gyors az adatfeldolgozásban
2	Többszálú programozás	Memória takarékos
3	Nagyszámú erőforrás áll rendelkezésre (például: az autentikáció már előre megvan írva)	NPM(Node Package Manager)-nek köszönhetően egyre több szolgáltatás érhető el.

Minden keretrendszer között találhatunk előnyöket is meg hátrányokat is. A Spring Boot és a NodeJs között előnyöket az 4.3 táblázatban már tárgyaltuk. A következőkbe néhány hátrányról [20] lesz szó:

- A NodeJS nem tud hatékonyan teljesíteni nagy számítások esetén.
- A NodeJS mivel még új technika, ezért nincs teljesen kifejlesztve, tesztelve így sok olyan hibába ütközhetünk amire nincs ideális megoldás.
- A Spring Boot legnagyobb hátránya, hogy a memóriaigénye nagyon nagy.
- A Spring Boot egy másik hátránya, hogy a hiba keresés meglehetősen nehéz mivel sok beépített kód található benne.

Összefoglalva mindkét technológia a maga módján megfelelő különböző alkalmazások tervezésénél. Ha az alkalmazás sok bemeneti/kimeneti feladatot (például: sok regisztráció) tartalmaz akkor mindenképp a NodeJs-t érdemes választani. Viszont ha biztonságos és önálló alkalmazást tervezünk amely intenzív CPU használatot igényel akkor a Spring Boottal érdemes foglalkozni.

5. fejezet

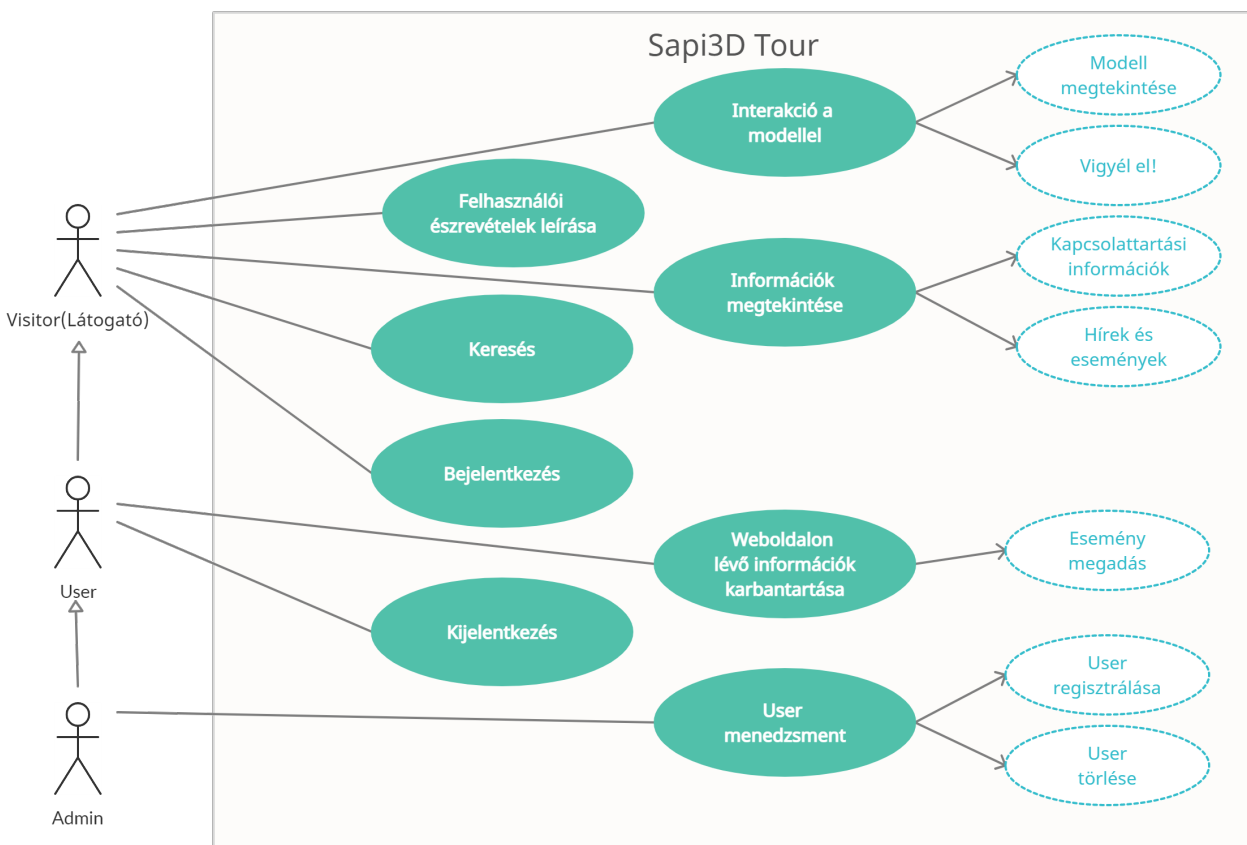
Követelmény specifikáció

5.1. Felhasználói követelmények

A Sapi3D alkalmazás web alapú, ezért mindenki számára elérhető. A fő célja, hogy egy 3D modellként jelenítse meg a Sapientia EMTE Marosvásárhely-i karának a főépületét, illetve ennek fontosabb helyeit, mint pl. tanszékek, titkárság stb. A rendszer fontosabb funkcionálisait és az ezeket igénybe vevő szerepköröket a 5.1 ábra szemlélteti.

Az alkalmazás eléréséhez, használatához egyetlen feltételnek kell eleget tenni, amely az internet kapcsolat megvalósítása lenne. Tekintsük meg a 5.1 ábrát amely bemutatja a rendszert, amit három különböző felhasználó vehet igénybe: Visitor, User és Admin, az utóbbi kettőhöz bejelentkezés szükséges.

Visitor (látogató), amelynek lehetősége van megtekinteni az elkészített weboldalt, igénybe veheti a "vigyél el" opciót és nem utolsó sorban saját kezűleg is végig tud menni az egyetem 3D modelljén. Vannak olyan Visitor-ok akik betudnak jelentkezni, így átalakulnak User-ré. A User az aki felelős az alkalmazás karbantartásáért. A karbantartás alatt kell érteni azt, hogy az oldalon megjelenő információk napra készek legyenek. Az Admin felhasználó az, akinek a rendszerben van a legnagyobb felelőssége. Ő felel azért, hogy mely Visitorok kaphatnak engedélyt a bejelentkezéshez. Ez mellett az ő hatáskörébe tartozik, hogy ki lesz kitörölve a rendszerből.



5.1. ábra. A rendszer használati eset diagramja

Bármely felhasználó, aki egy böngészőből megnyitja az oldalt, a Visitor kategóriába kerül. Ez a fajta felhasználó megtekintheti a 3D modellt, körbe sétálhatja és el tud jutni pl. titkárságra. Amint megnyílt az oldal rögtön látható az egyetemről készített modell. Ezen a modellen tud nézelődni, esetleg körbe is tudja járni, vagy adott helységekre el is tud jutni (például: titkárság, adott tanszék). Ezen kívül lehetősége van információk, elérhetőségek, események részleteinek elolvasására is. Minden Visitor ugyan akkor leírhatja saját véleményét, meglátásait az oldallal kapcsolatban is. A fent említett műveletek elvégzéséhez nem kell sem bejelentkezés, sem regisztráció.

A bejelentkezéshez szükséges megadni egy már regisztrált e-mail címet és egy már hitelesített jelszót is. A User engedélyezése nem regisztráció alapján történik, hanem az admin joggal rendelkezők osztják ki, mivel a rendszer úgy van megtervezve, hogy nincs regisztráció, hanem csak egy Admin tudja beregisztálni az új User-eket. A regisztráció azért történik így, mert az egyetemen csak bizonyos

személyeket lehet erre kinevezni, különben bárki invalid adatokkal tudná ellátni az oldalt. Egy User képes különböző műveletek elvégzésére, mint például: eseményeket megadni, az egyetemmel kapcsolatos információkat megadni, módosítani. Ezeken kívül a Usereknek lehetősége van a kijelentkezésre is.

Az Admin nem csak a Userek regisztrálásával kell foglalkozzon, hanem azoknak a törlésével is. Ezek mellett foglalkozik a teljes adatbázis menedzselésével és a rendszer újraindításával is.

5.2. Rendszerkövetelmények

5.2.1. Funkcionális követelmények

Visitor

- A használatához a digitális eszköz lehet laptop, asztaligép, tablet vagy akár telefon is.
- Interakció a 3D modellel a megjelenített gombok használatával. A gombok biztosítják az előre hátra jobbra és balra menést.
- A "Vigyél el" funkció használata. A modell fölött található legördülő listából való kiválasztás esetén egy új gomb megjelenésével és annak használatával látható lesz az út a bejáratától a kiválasztott helyre.
- Fontos egyetemi helyek menüpont megtekintése, ahol az adatbázisban található adatok jelenítődnek meg. A címekre kattintva megjelennek a bővebb információt tartalmazó linkek. A címek mellett megjelenik egy térkép ikon is, amelyre ha rákattintanak akkor vissza kerülnek a 3D modellhez és a "Vigyél el" funkciót használva ismét megmutatódik az út a bejáratától a kiválasztott helyig.
- Az események menüpont alatt találhatóak azon esemény kártyák, amelyek nem tartoznak egy adott tanszékhez, részleghez az egyetemen belül. A kártyákra kattintva az oldal tovább visz az adott esemény bővebb leírásához.

User

- Jelszó hitelesítése. Az e-mail-ben kapott validációs tokenet felhasználva a jelszó hitelesítés oldalon megadni a jelszót, a megfelelő formátummal (Legyen benne legalább kisbetű, nagybetű és szám. Legyen legalább 5 karakter hosszú). Az e-mailben kapott validációs token lejáratí idővel rendelkezik.
- Bejelentkezés a regisztrált e-mail és hitelesített jelszó megadásával történik, amint a User megnyomja a "Bejelentkezés" gombot.
- Hozzáférés az adatbázisban tárolt adatokhoz.
- User képes szerkeszteni a saját adatait (név, e-mail cím, telefonszám), ha rákattint a jobb sarokban lévő ember ikonra. Ezek után egy ablakba előjönnek az adatai amelyek szerkeszthetőek. A szerkesztés után a "Adatok szerkesztése" gombra kattintva az adatok bekerülnek az adatbázisba. A megadott adatok elegett kell tegyenek a megfelelő formátumoknak.
- Az egyetemi részleg megadása, szerkesztése funkcionál egy rádiógomb segítségével ki tudja választani a User, hogy új részleget szeretne megadni, vagy már egy létező részleget szeretne módosítani. Ha a részleg megadását választja akkor meg kell adja a részleg nevét (példaul: Dékáni Hivatal) és egy *linker*¹ amely elvisz a részleget leíró oldalra. Ezek után a "Hozzáadás" gomb megnyomásával az adatok bekerülnek az adatbázisba. A szerkesztés esetén lehet módosítani a nevet is és a linket is. A "Szerkesztés" gomb segítségével az adatbázisban szereplő adatok átíródnak.
- A szak megadása és szerkesztése funkcionál szintén egy rádiógomb segítségével tudja eldönteni, hogy megadni szeretne egy új szakot, vagy a meglévőket szeretné módosítani. Az új szak megadásánál a következő információkat kell megadni: szak neve; szakkoordinátor neve; szakkoordinátor e-mail címe; terem száma, ahol a szakkoordinátor tudja fogadni az érdeklődőket; egy linket a szak részletes leírásához és egy legördülő lista segítségével, hogy melyik tanszékhez tartozik az adott szak. A "Hozzáadás" gomb segítségével az adatok bekerülnek az adatbázisba.

¹<https://ms.sapientia.ro/hu/munkatarsak/dekani-hivatal>

A szerkesztés esetén a fent említett adatokat lehet szerkeszteni. A szerkesztés akkor lesz végleges ha a User megnyomja a "Szerkesztés" gombot.

- Egy részleghez, nem csak szakot lehet megadni hanem eseményeket, tevékenységeket is ha az Egyebek hozzáadását használja a User. Itt megkell adni egy legördülő lista segítségével, hogy a mely részleghez tartozik (például Villamosmérnöki Tanszék), egy nevet (például: Sapi-Line-Tracer), és egy linket ahol több információ van leírva az eseményről, tevékenységről. Itt is szintén a "Hozzáadás" gomb megnyomásával kerülnek be az adatok az adatbázisba.
- A User feladatai közé tartozik az események hozzáadása is. Ezt az Esemény hozzáadása címmel ellátott űrlap segítségével teheti meg. A sikeres esemény megadáshoz kötelezően megkell adni egy esemény nevet, esemény szervezőt, egy képet az eseményről és egy linket ahol további információkat tudnak kapni a Visitorok az adott eseményről.
- A User ha elvégezte a hozzáadásos, szerkesztéses feladatait, akkor a jobb felső sarokban található "Kijelentkező" gombra kattintva ki is tud jelentkezni.

Admin

- Hozzáférés az adatbázisban tárolt adatokhoz.
- Userek regisztrálása és az adatok validálása a teljes név, e-mail cím, telefonszám megadásával. Az e-mail cím és a telefonszám eleget kell tegyen a megfelelő formátumoknak. A telefonszám csak számokból állhat és 10 karakterből kell álljon. Ezek után a "Hozzáadás" gomb lenyomásával az adatok eltárolása a NoSql adatbázisba és egy e-mail küldés a Usernek a jelszó hitelesítéséhez, amely tartalmaz egy validációs tokent, ami a regisztráció pillanatában generálódik.
- Userek törlése: a User e-mail címének kiválasztása egy legördülő listából, majd a "Felhasználó keresése" gomb lenyomásával a User adatainak megkeresése. Ezek után a "Törlés" gomb használatával a User adatainak kitörlése az adatbázisból.

5.2.2. Nem funkcionális követelmények

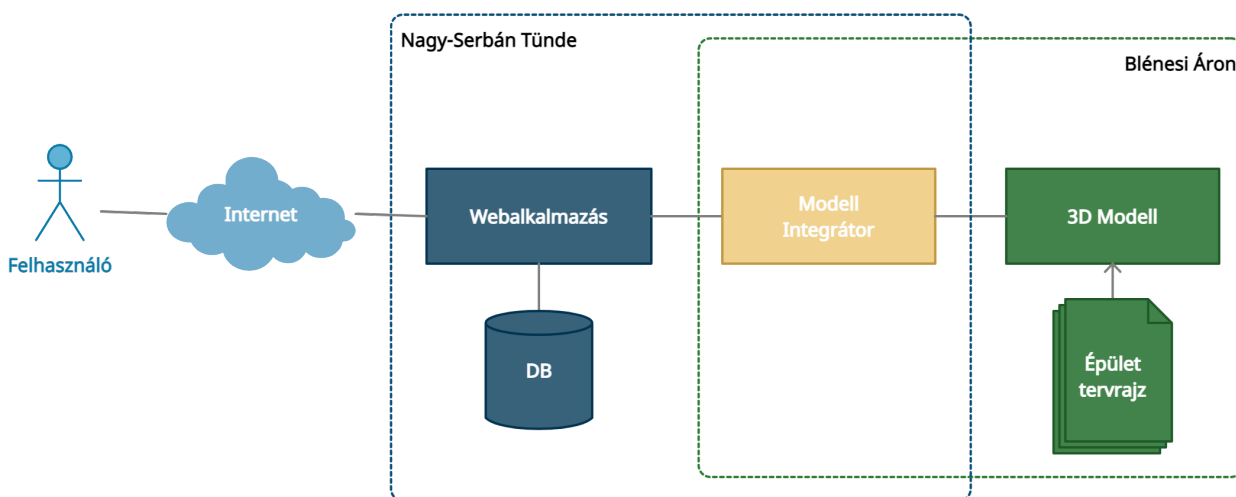
- A rendszer használatához a felhasználóknak szüksége van Internetre és egy digitális eszközre amelyen megtudják nyitni a webalkalmazást.
- A felhasználók nincsennek egy adott operációs rendszerhez kötve.
- Az operációs rendszer lehet Windows, Linux, Android és iOS is.
- Külön telefonos applikáció még nincs, viszont a telefonokon található böngészők bármelyikében meglehet tekinteni az alkalmazást.
- A regisztrált felhasználók adatai egy NoSQL adatbázisban (MongoDB) vannak eltárolva, amely nem lokálisan van jelen a felhasználók eszközén.
- Az azonosítás a Spring Boot beépített autentikációs moduljával történik meg.

6. fejezet

Tervezés

6.1. A rendszer architektúra

A rendszer több részből áll, melyen két diák is dolgozik, így két dolgozat is készül. Egyrészt a 3D modell, és másrészt az ehhez tartozó UI. Ezeket foglalja össze az 6.1 ábra.



6.1. ábra. A teljes rendszer architektúrája

A 3D modell tervezését, implementálását a kollégám, Blénesi Áron készíti el míg a webalkalmazást, adatbázis tervezést én. A két projektet egy komponensen keresztül kötjük össze. Ez a komponens a web alkalmazáson belül van létrehozva, és általa jelenítődik meg a 3D modell.

A kolléga dolgozata arról szól, hogy a modell hogyan lett megtervezve, elkészítve és nem utolsósorban hogyan lett beépítve egy weboldalba. A modell az egyetem valós tervrajzai alapján készült el törekedve a pontosságra, hiszen biztosítani szeretnénk volna a felhasználónak azt az élményt, hogy életnagyságban járhasse be a digitalizált épületet. A feldolgozáshoz használt segédeszköz a Blender, ingyenes modellező software volt. A komplex modell elkészítése után ez exportálásra kerül egy glTF (Grafikus nyelvátviteli formátum) kiterjesztésű fájlba. Ez a fájl majd beépítésre kerül a webalkalmazásba a modell komponensen keresztül. Ez a komponens felel a 3D-s modell megjelenítéséért és az épület bejárési logikájának megvalósításáért. Ezen az oldalon a felhasználó interakcióba léphet a modellel, szabadon bejárhatja azt, illetve úticélt választhat a megadott listából és a kamera egy jólmeghatározott útvonalat követve lépésenként elér a kiválasztott úticélig.

6.2. Webalkalmazás architektúra

A tervezés során kideült, hogy négy nagy komponensre lesz szükség: Frontend, Webszerver, Adatbázis és a 3D modell.

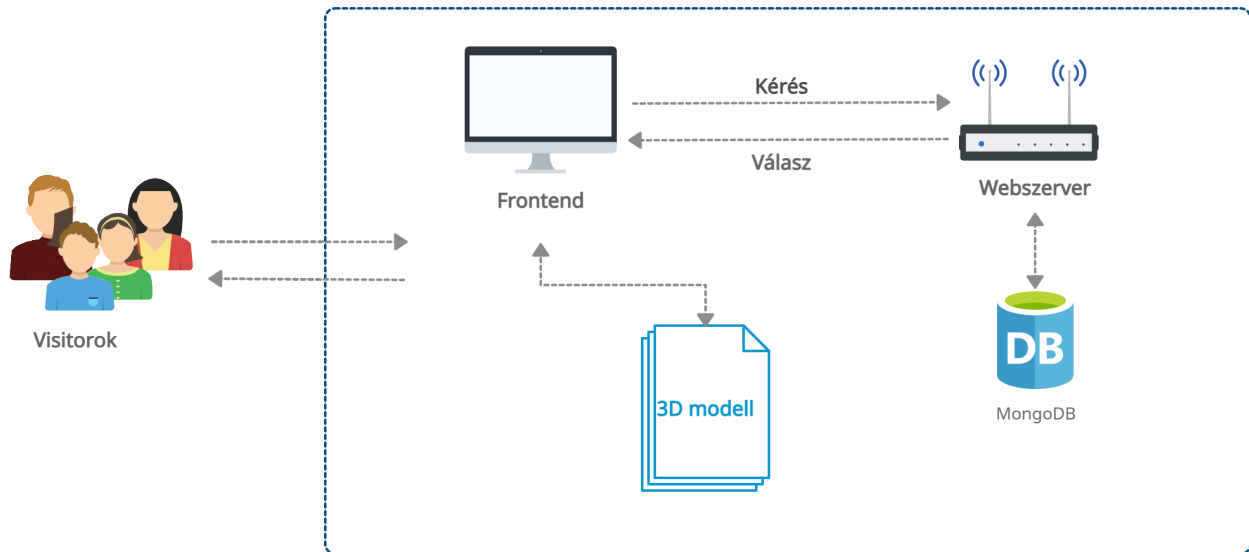
Amint a 6.2 ábrán is látszik a Visitorok a Frontenden keresztül érik el az alkalmazást. A Frontend fogja biztosítani a felhasználói felületet, vagyis itt fognak megjelenni az adatok az adatbázisból és a 3D modell is ide fog betöltődni.

A Webszerver segítségével lesznek elérhetőek a Frontend számára az adatbázisban tárolt adatok. Tulajdonképpen a Webszerver biztosít egy kommunikációs csatornát a Frontend és az adatbázis között. A Frontend kéréseket (GET, PUT, POST) küld a webszervernek és a webszerver a kérésnek megfelelő eredményt szolgáltatja vissza.

Az adatbázis fogja tárolni a Frontendről érkező adatokat, valamint vissza is szolgáltatja azokat a megfelelő kérések esetén a Webszerveren keresztül.

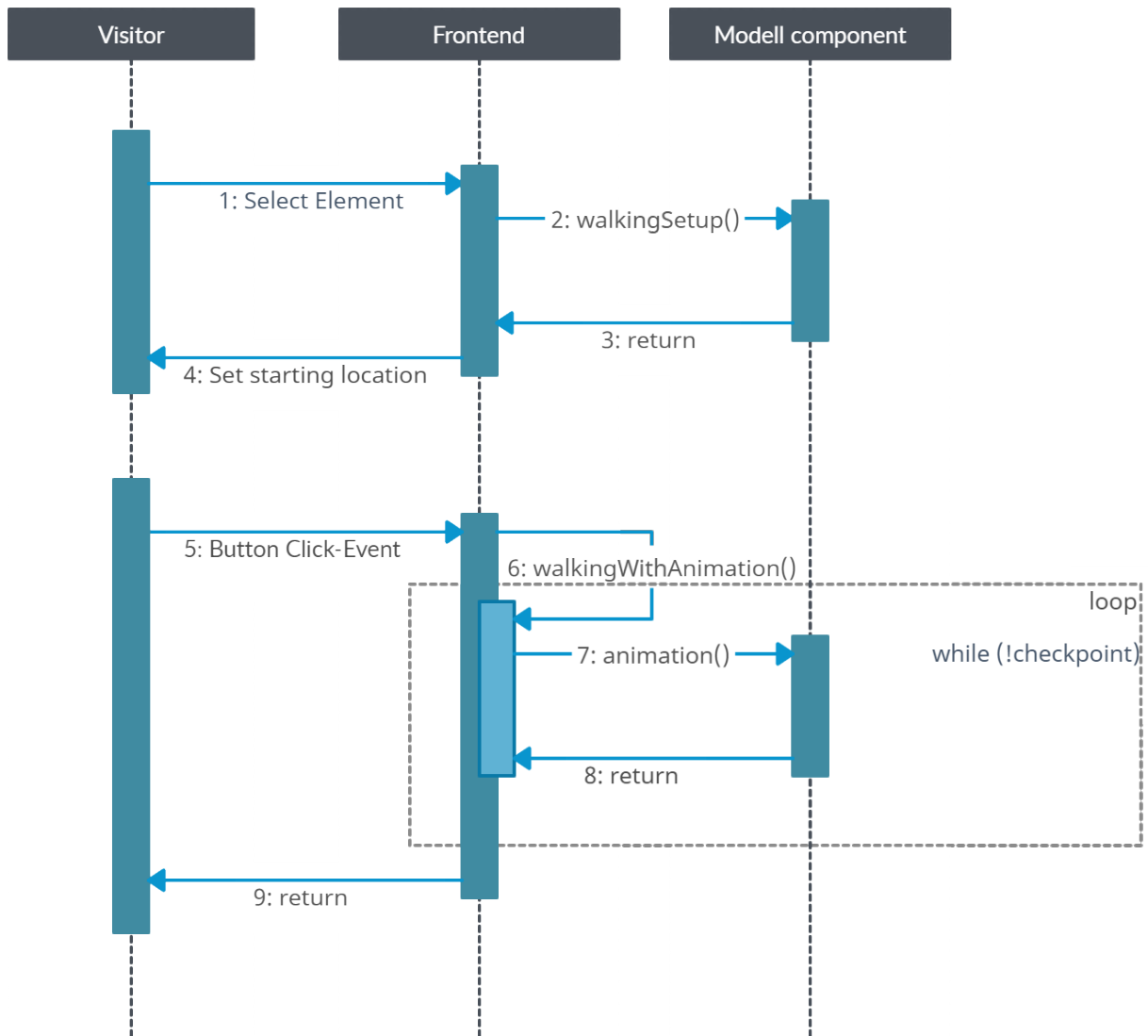
A 6.1 fejezetben említett 3D modell egy statikus fájlban lesz elhelyezve, amelyet a Frontend ér el és onnan fogja betölteni a modellt egy Frontenden belüli komponensbe. A statikus fájl tartalmazza azokat az elemeket, amelyek megjelennek az alkalmazáson belül, viszont az adatbázisban nem lehetséges vagy nagyon nehéz eltárolni. Ilyen elem a 3D modell is mivel a mérete meghaladja a 30 MB-ot. A Frontenden belüli komponensen keresztül fog összekapcsolódni a 6.1 fejezetben említett

két alrészleg, maga a webapplikáció és a 3D modell.



6.2. ábra. A webalkalmazás rendszer architektúrája

Az alkalmazás egyik lefgontosabb funkciója a "vigyél el" funkció, amelynek szekvencia diagramját a 6.3 ábrán tekinthetjük meg, amely ezen funkció használatát írja le. Az első lépésben a Visitor a legördülő listából kiválaszt egy helyet ahová elszeretne jutni. Ezek után a Frontend beállítja a túra paramétereit mint például: a checkpointokat a túrához. Ha ezek a beállítások megtörténtek megjelenik egy gomb amely, a túra elindításáért felel. A következő lépésben ezt a gombot megnyomva elindul a túra. Ezt követően elindul a háttérben egy animációs függvény amely két checkpoint között animál. Az animáció a kamerák állításával közelítésével történik. Ha befejeződik az animáció azt jelenti elértük a következő checkpointot és újra meg kell nyomni a gombot amely ismét megjelent.



6.3. ábra. A "vigyel el funkció szekvencia diagramja"

6.3. A fejlesztéshez használt technológiák

A rendszer egy webalkalmazásként lett fejlesztve. Az alkalmazás fejlesztéséhez első sorban szükséges volt egy operációs rendszer amely támogatja a Vue.js-t, Spring Boot-t, MongoDB-t.

6.3.1. Front-end

A Frontend rész HTML/JavaScript-ben implementálódott, keretrendszernek pedig a Vue.js volt használva, mivel egy új, progresszív keretrendszer ezért, folyamatosan új dolgok használatát biztosítja a fejlesztők számára, így a keretrendszerek használatának felmérési listáján is elől szerepel. A Vue.js használatához a következő telepítéseket, utasításokat, parancsokat kellett végrehajtani ebben a sorrendben:

- nodejs és npm letöltése és telepítése: segítségével könnyen tudjuk telepíteni a Vue-t.
- npm install vue: a Vue telepítésének parancsa.
- vue init webpack sapi3dtour: a projekt inicializása.
- npm install vue-router: segített a routolás megoldásában.
- vue add vuetify - prototyp: a projektben használt HTML tageknek változatosabb stílust add hozzá.
- npm install @fortawesome/fontawesome-free: a *Font Awesome*¹ ikonok használatát tette lehetővé.
- npm install --save axios: segítségével tudtam kérni és küldeni adatokat a Backendre.
- npm install vue-3d-model --save: a 3D modell betöltéséért felelős csomag.

A Frontend indításához először bele kell lépni a sapi3dtour könyvtárba, ott nyitni egy console ablakot és lefutatni a következő parancssort: *npm run dev*.

6.3.2. Back-end

A 6.1 ábrán látható Webszerver része, amely fogadja az adatokat, kéréseket a Front-end-től és vissza is küldi neki a megfelelő válaszokat.

¹<https://fontawesome.com/icons?d=gallery&p=2&m=free>

A Backend rész Java-ba lett implementálva, keretrendszernek a Spring Bootot használtam, mivel a segítségével nem kell külön foglalkozni a Maven vagy Gradle fájlok, Tomcat szinkronizálásával. Ez annak köszönhető, hogy a Spring Boot fejlesztői kifejlesztettek egy weboldalt, ahol Spring Boot típusú projekteket lehet létrehozni. Ez az oldal nem csak a projekt struktúrájának létrehozásában segít, hanem a különböző dependenciákat is betudjuk állítani egyszerűen. A Spring Boot projekt létrehozásához a következő lépéseket végeztem el:

- A legújabb Java (15.0.1) instalálása
- Spring Boot keretrendszer telepítése
- *Spring Initializr*² segítségével létre hozni a projektet különböző dependenciák megadásával.
- A következő dependenciákat adtam hozzá:
 - MongoDB: az adatbázis elérésében segít.
 - Spring Web: lehetővé teszi a Cors filterek beállítását.
 - Spring Security: a beépített autentikációt teszi lehetővé.
 - Validation: a beérkező adatok validálásával foglalkozik.
 - Java Mail Sender: elősegíti az e-mail küldést.

Az 6.4 ábrán látható egy példa a Spring Initializr használatára. A példa pontosan a fent leírtakat mutatja be.

²Spring Initializr (a név helyesen van leírva): <https://start.spring.io/>

Project

☒ Maven Project
 ☐ Gradle Project

Language

☒ Java
 ☐ Kotlin
 ☐ Groovy

Spring Boot

☐ 2.5.0 (SNAPSHOT)
 ☐ 2.5.0 (M3)
 ☐ 2.4.5 (SNAPSHOT)
 ☒ 2.4.4
 ☐ 2.3.10 (SNAPSHOT)
 ☐ 2.3.9

Project Metadata

Group

Sapi3DTourMongo

Artifact

Sapi3DTourMongo

Name

Sapi3DTourMongo

Description

Sapientia 3D Tour backend implementation

Package name

Sapi3DTourMongo.Sapi3DTourMongo

Packaging

☒ Jar
 ☐ War

Java

☒ 16
 ☐ 11
 ☐ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Data MongoDB

NOSQL

Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Spring Security

SECURITY

Highly customizable authentication and access-control framework for Spring applications.

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Validation

I/O

Bean Validation with Hibernate validator.

Java Mail Sender

I/O

Send email using Java Mail and Spring Framework's JavaMailSender.

6.4. ábra. Példa a Spring Initializr használatára

6.3.3. Adatbázis

Az adatbázis MongoDB NoSql lett, amelyet az operációs rendszernek megfelelő Workbench-ben kezeltem. Mivel az adatokat nem lehet egy séma alapján felállítani ezért van szükség a NoSQL adatbázisra.

Az adatbázis létrehozásához a következő lépéseket végeztem el:

- Telepítettem a MongoDB adatbázis kezelő rendszert.
- A MongoDB-hez tartozó Workbench telepítése
- Létrehoztam az adatbázist a Workbench segítségével
- Kollekciókat nem itt hoztam létre. Az a Spring Boot feladata.

6.4. Modulok leírása

Ebben a fejezetben a modulok leírásáról lesz szó. A Back-end-en megírt modulokról tudni kell, hogy minden modul rendelkezik egy controller-el, service-el és egy service implementációval. A controller biztosítja a kérések (GET, PUT, DELETE) megfelelő kezelését, és a megfelelő válasz visszaküldését

is. A service egy interfész amelyet használ a controller. A service tartalmazza a metódusok deklarációját. A service implementáció a service-ből származik és ő felel a service-ben található deklarációknak az implementációjáért.

Ezen kívül megjelennek a modellek, amelyek az adatbázis táblákat írják le. Ezeket a modelleket egy-egy repository kezeli. A repository által tudunk az adatbázisban lévő adatokon módosítani, új adatokat hozzáadni, törölni és lekérdezni is.

A Back-end-en van biztosítva az is, hogy egyes kéréseknél kötelező a bejelentkezés. Ezt úgy oldottam meg, hogy a kérés mellé kell egy Headert is csatolni, ami kell tartalmazzon egy Bearer Token-t. A Bearer Token egy generált kód, amely a User adataiból és az érvényességi időből áll össze, lásd 6.5 ábrát. A Header segítségével lesz igazolva a User, hogy bevan jelentkezve vagy sem. Ha a Header tartalmazza a Bearer token-t akkor a User be van jelentkezve különben nem. Ennek a segítségével tudja eldönteni a webszerver, hogy adott kérésekre milyen választ küldjön vissza, mivel a kérések megvannak különböztetve a Userok és a Visitorok között.

6.5. ábra. Példa Bearer Tokenre

```
1 {  
2   "BearerToken" : "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlc2Vc2FwaTNkQGdtYWlsLm  
3     NvbSIsImIhdCI6MTYxMzEzMDYzMiwiaXNjaXZlbnRvbmUiOiJ0dWwudGVzLnR1bmQyYnV5IiwiaWF0IjoxNTY0ODAwMDAuanB4InQ.  
4     2yBuysIO77x3C80CDav9iJ4dWB7ksO5gfEeYkyvmtyB-YwRuxpD1d0VKm2lVlKfZUGw"  
5 }
```

6.4.1. Visitor fontosabb komponensei

- **Interakció a 3D modellel** Az interakciót a modellel gombok segítségével oldottuk meg. A gombok biztosítják az előre, hátra, jobbra, balra irányba való menést. A modell betöltéséért felelős Vue package által használt kamera pozíciójának állításával tudtuk beállítani ezeket a lépés konfigurációkat.
- **"Vigyél el" funkció** A "Vigyél el" funkció megvalósításánál is szintén létrehoztunk egy gombot amely két checkpoint között navigál el. De ezek mellett egy legördülő listát is megadtunk annak érdekében, hogy a Visitor tudja kiválasztani azt a helyet ahová szeretne menni. Ha kiválasztunk a legördülő listából egy helyet akkor az a bizonyos navigációs gomb azonnal megjelenik. A navigáció egy előre megírt tárolóból veszi ki a checkpointokat. Erről a részről viszont a kollégám

(Blénesi Áron: Sapi3D tour – 3D model) dolgozatában [21] lehet bővebben olvasni.

- **Fontosabb egyetemi helyek megtekintése** A Visitorok megtekinthetik a Userok által megadott információkat. Viszont ezek az információk az adatbázisban vannak eltárolva így a Visitorok közvetlen nem tudják elérni. A webszerver segítségével a Front-end lekéri az adatokat az adatbázisból. A webszerver válaszként visszaküldi az adatokat a Front-endnek, amely a megkapott információkat megjeleníti a Visitorok számára. A "Vigyél el" funkció itt is megjelenik. A részlegek címei mellett megjelenik egy gomb amely segítségével a 3D modellen megmutatódik a bejárattól az odavezető útig.
- **Események megtekintése** A Visitorok a fontosabb egyetemi helyek mellett egy külön részben megtekinthetik az eseményeket amelyeket az egyetemen belül szerveznek. Ezeknek az eseményeknek a megjelenítése hasonlóan működik, mint a fontosabb egyetemi helyek esetén. Az adatok az adatbázisban vannak eltárolva. A megjelenítés a webszerver és Front-end segítségével történik.

6.4.2. User fontosabb komponensei

- **Jelszó hitelesítése** A regisztráció véglegesítéséhez szükséges a jelszó hitelesítése, amely azt takarja hogy a User megkell adja a saját jelszavát és a validációs codeToken, amelyet a Front-end elküld egy JSON objektumként a Back-end-nek, lásd a 6.6 ábrán. Amint látható a jelszót nem kódolva küldtem mivel a bejelentkezéshez szükséges e-mail cím és jelszó páros nem jelenik meg együttesen, így a feltörés nem történhet meg. Ezek mellett a továbbiakban belesz vezetve a HTTPS protokoll, amely egy TLS 1.2 réteget fog használni 1024 bites kulcsokkal. Így a rendszer feltörése nagyon nehéz lesz.

6.6. ábra. Front-end-ről érkező jelszó JSON objektum

```
1  {  
2    "codeToken" : "-WfqPn0u68StNLg_m1Jnn4g8KjVb2Qu0",  
3    "password"  : "Almakorte12",  
4    "passwordAgain" : "Almakorte12"  
5  }
```

Ezek után a Back-end fogadja az adatokat és elkezd feldolgozni. A feldolgozás első fázisa

a különböző ellenőrzések, amelyeknek többsége a codeToken-t ellenőrzik. A codeToken az adatbázisban van eltárolva és automatikusan generálódik amikor a regisztráció történik és ezt kell a User megadja a Front-end részen. A 6.7 ábrán az első sortól a hatodik sorig történő ellenőrzés azt vizsgálja, hogy a codeToken amit a felhasználó megadott létezik-e. Ha létezik akkor megy a következő lépésre, ha nem akkor visszaküld egy hibát. A második ellenőrzés a 12. sortól a 15. sorig történik és azt ellenőrzi, hogy ezzel a codeTokennel már volt-e megadva jelszó. Ha igen akkor hibát küld vissza a Back-end, ha nem akkor lépik tovább az ellenőrzés fázison. A harmadik ellenőrzés a codeToken lejáratási idejének ellenőrzése, amelyet a 17. sortól a 19. sorig láthatunk. Itt is ha az idő lejárt akkor hibát kapunk válasznak, ha nem lépünk tovább. A 20-as sortól a 25. sorig történő ellenőrzés arról szól, hogy a két megadott jelszó egyezik-e. Ha egyezik akkor átlépünk a második fázisba.

6.7. ábra. Jelszó hitelesítés ellenőrzése

```
1      if(!registartionRepository.existsByToken(  
2          passwordRequest.getCodeToken()  
3      ))  
4      {  
5          throw new Exception("Wrong token!");  
6      }  
7      Registration registartion = registartionRepository  
8          .findByToken(passwordRequest.getCodeToken())  
9          .orElseThrow(() -> new RuntimeException(  
10             "Error: Registartion is not found."  
11         ));  
12      if (registartion.isEnable())  
13      {  
14          throw new Exception("Already exist!");  
15      }  
16      Calendar cal = Calendar.getInstance();  
17      if (registartion.getValidDate().before(cal.getTime())) {  
18          throw new Exception("Time done!");  
19      }  
20      if (!passwordRequest.getPassword().equals(  
21          passwordRequest.getPasswordAgain()  
22      ))  
23      {  
24          throw new Exception("The passwords not equals!");  
25      }
```

A sikeres ellenőrzések után következik a jelszó elmentése az adatbázisba, amelyre láthatunk kód részletet a 6.8 ábrán. Látható a kódnak a 4. sorában, hogy egy encoder segítségével kódoljuk a jelszót és csak úgy mentjük el az adatbázisba.

6.8. ábra. Jelszó mentése

```
1  Iterator<User> itr = registartion.getUser().iterator();
2  BCryptPasswordEncoder passwordEncoder =
3      new BCryptPasswordEncoder();
4  String encodedPassword = passwordEncoder.encode(
5      passwordRequest.getPassword()
6  );
7  User user = userRepository.findById(itr.next()
8      .getId()
9      .toHexString())
10     .orElseThrow(() -> new RuntimeException(
11         "Error: User is not found."
12     ));
13  user.setPassword(encodedPassword);
14  userRepository.save(user);
15  registartion.setEnable(true);
16  registartionRepository.save(registartion);
```

- **Bejelentkezés** A bejelentkezés egy regisztrált e-mail címmel és egy hitelesített jelszó párossal történik. A Front-end elküldi az adatokat JSON objektumként a Back-end-nek, amely feldolgozza, lásd a 6.9 ábrán. A beépített authentication manager segítségével létrehozunk egy autentikációs párost a megadott e-mail címből és jelszóból. Security Context Holder segítségével ellenőrizzük, hogy az autentikációs páros benne van-e az adatbázisba. Ha benne van akkor, a beépített jwt (JSON Web Token) token generálással generálunk egy Bearer Tokenet. Ezek után lekérdezzük az User adatait a getPrincipal() függvénnyel. A következő lépésben megkeressük, hogy a User milyen szerepkörrel rendelkezik. Az utolsó lépésben válaszként vissza küldjük a Bearer Tokenet, a User id-t és a szerepkört. A Front-end megkapja ezeket az adatokat és eltárolja a localStorage nevezetű tárolóhelyben.
- **Saját adatok szerkesztése** A User tudja szerkeszteni saját adatait, amelyeket újra küld a Front-end a Back-endnek. A Back-end a User id-ja alapján megkeresi az adatbázisba a Usert és módosítja az adatokat.
- **Egyetemi részleg, szak és egyebek megadása** Mindhárom elem megadása hasonló logika alapján épül fel. A Front-end küld egy JSON objektumot, amelyet a Back-end dolgoz fel. A JSON objektumok változnak. Az egyetemi részleg (tanszék, hivatal) esetén a JSON objektum tartalmazza a részleg nevét és egy linket. A szak megadásánál megjelenik a szak név, szakkoordinátor név, szakkoordinátor e-mail cím, egy terem szám, egy link és egy tanszék. Az egyebek megadásánál, a User dönti el, hogy milyen információkat ad meg. Lehet több link, több szöveg, cím

6.9. ábra. Bejelentkezés kód szinten

```
1 Authentication authentication =
2   authenticationManager.authenticate(
3     new UsernamePasswordAuthenticationToken(
4       loginRequest.getEmailAddress(),
5       loginRequest.getPassword()
6     )
7   );
8 SecurityContextHolder.getContext()
9   .setAuthentication(authentication);
10 String jwt = jwtUtils.generateJwtToken(authentication);
11 UserDetailsImpl userDetails =
12   (UserDetailsImpl) authentication.getPrincipal();
13 List<String> roles = userDetails.getAuthorities()
14   .stream()
15   .map(item -> item.getAuthority())
16   .collect(Collectors.toList());
17 return ResponseEntity.ok(new LoginResponse(jwt,
18   userDetails.getId(),
19   roles)
20 );
```

is. A Back-end amiután megkapta ezeket a JSON objektumokat feldolgozza őket. A 6.10 ábrán láthatjuk, hogy a Back-end hogyan ad hozzá egy új részleget. A másik két esetben is hasonlóan történik, csak az adatok változnak. Az első részben leván ellenőrizve, hogy a részleg létezik-e, ha nem létezik akkor departmentRepository segítségével elmentjük az adatbázisba, különben nem sikerül az elmentés és egy hibát fog visszatéríteni a Back-end a Front-endnek.

6.10. ábra. Példa részleg megadásra kód szinten

```
1 if (departmentRepository.existsByDepartmentName(
2   depReq.getDepartmentName()
3 ))
4 {
5   throw new Exception("Department already exist!");
6 }
7 try {
8   Department dep = new Department(
9     depReq.getDepartmentName(),
10    depReq.getLink()
11  );
12  departmentRepository.save(dep);
13 } catch (Exception e) {
14   System.out.println(e);
15   throw new Exception("Wrong add department!");
16 }
17 }
```

- **Egyetemi részleg és szak szerkesztése** Az egyetemi részleg és szak szerkesztése hasonló módon működik mint a hozzáadás, a Front-end küld egy-egy JSON objektumot a Back-end-nek majd a Back-end fogadja és feldolgozza. Első sorban megkeresi az adott részleget, tanszéket

majd a megkapott objektum elemeit átírja majd elmenti.

- **Esemény hozzáadása** Az esemény hozzáadása is hasonló képpen működik, mint a szakok, részlegek vagy akár az egyebek hozzáadása is. A Back-end kap egy POST kérést, amellyel együtt kap egy JSON objektumot is. A JSON objektum tartalmazza az esemény adatait, amint látható 6.11 ábrán. Az objektum kell tartalmazzon egy címet (title), szervezőt (artist), egy kép Base64-ként kódolt változata (src) és egy linket (link).

A feldolgozás kód szintű leírását a 6.12 ábrán tekinthetjük meg, amelynek 2 sorától az ötödik soráig létrehozunk egy Event modelt, míg a hatodik sorban elmentjük a létrehozott model adatait az adatbázisba. A Back-end az adatok feldolgozása után visszaküld egy üzenetet a Front-end-nek, hogy a hozzáadás sikeres volt vagy sem. Az üzenet típusától függően a Front-end ad egy visszajelzést a Usernek a hozzáadás sikerességéről, sikertelenségéről.

6.11. ábra. Front-end-ről érkező esemény adatokat tartalmazó JSON objektum

```
1  {  
2    "title" : "Regisztrációs het",  
3    "artist" : "Hallgatói Önkormányzat (HOK)",  
4    "src" : "data:image/jpeg;base64,/9j/4AAQSkZJRgAB...",  
5    link : "https://ms.sapientia.ro/hu/hirek/tanevkezes-2020"  
6  }
```

6.12. ábra. Példa esemény megadásra kód szinten

```
1  try {  
2    Event event = new Event(eventAddRequest.getTitle(),  
3    eventAddRequest.getSrc(),  
4    eventAddRequest.getArtist(),  
5    eventAddRequest.getLink(), "");  
6    eventRepository.save(event);  
7  } catch (Exception e) {  
8    System.out.println(e);  
9    throw new Exception("Wrong add event!");  
10 }
```

- **Kijelentkezés** Ha a User a Front-enden a kijelentkező gombra kattint, akkor a Front-enden tárolt id-ja és a Bearer Tokenje kitörlődik a localStorage-ból, amely egy tároló hely és a Bejelentkezésnél tárolt adatokat tartalmazza (user id, szerepkör - admin vagy user, a Bearer token).

6.4.3. Admin fontosabb komponensei

- **User regisztrálása** Az alkalmazás karbantartását a Userok végzik, ezért szükség volt a regisztrálásra. A regisztrálás elvégzéséhez az Admin megkell adja a User teljes nevét, egy valid e-mail címet, egy telefonszámot és a User szerepkörét is. A valid e-mail címre azért van szükség mivel a regisztráció véglegesítéséhez kell hitelesíteni a User jelszavát. Ezt a User tudja megtenni, az e-mailel amit a webszerveren belüli Back-end rész küld el, lásd 6.13 ábrán.

6.13. ábra. E-mail küldés

```
1      MimeMessage mimeTypeMessage = emailSender.createMimeMessage();
2      MimeMessageHelper helper =
3          new MimeMessageHelper(mimeTypeMessage, "utf-8");
4      String htmlMsg = "Üdvözlünk!
5          Ont sikeresen jelentkezett az
6          alkalmazás használatára.
7          A jelentkezés véglegesítésére
8          a következő linken adja meg jelszavát!
9          http://localhost:8080/#/password
10         Azonosító kódja a következő:"
11         + token +
12         " Figyelem a link elérhetősége korlátozott!";
```

Az e-mailban megjelenik egy oldal, ahol a hitelesítés kell megtörténni és egy validációs token amely igazolja a Usert. A validációs token a Back-end generálja ki. A kigenerálás folyamatát meglehet tekinteni a 6.14 ábrán. A token maga bájtokból épül fel amelyeket Spring Boot beépített `secureRandom.nextBytes` függvényével generáltattunk ki. Ezt majd a szintén beépített encoder segítségével kódoltunk. A kódolt token szintén lementettük az adatbázisba a névvel, e-mail címmel, a telefonszámmal és a User szerepkörrel.

6.14. ábra. Validációs token generálása

```
1      private String generateNewToken() {
2          byte[] randomBytes = new byte[24];
3          secureRandom.nextBytes(randomBytes);
4          return base64Encoder.encodeToString(randomBytes);
5      }
```

- **User törlése** A User törlését is szintén az Admin végzi. A törlés e-mail cím alapján történik. A Front-end-en az Admin elküldi a kérést, hogy az adott Usert kisseretné törölni. A kérésben van egy JSON paraméter amely a kitörölni kívánt User e-mail címét tartalmazza. Kérést a Back-end feldolgozza és vissza küld egy választ a Front-end-nek, hogy a törlés sikeres volt vagy sem.

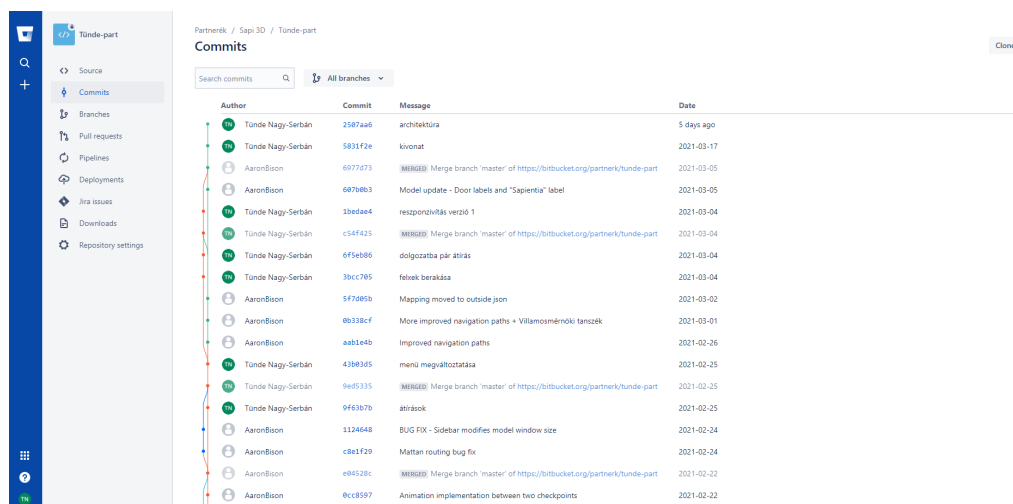
6.5. Verziókövetés és projektmenedzsment

A rendszer fejlesztéséhez szükséges volt egy verziókövető rendszert választani. Erre azért volt szükség, mivel ketten készítettük a projektet, a verziókövető rendszer segítséget nyújtott, a sikeres közös fejlesztéshez. A verziókövető rendszer mellett egy projekt menedzsment eszközt is választottunk, ahol a feladatainkat (task) tudtuk nyilvántartani.

6.5.1. Bitbucket

Kódverzió követő rendszer [22], amelyet a fejlesztők számára hoztak létre. Lehetőséget biztosít a git adattárak kezeléséhez és a fejlesztési folyamat végigvezetéséhez.

A 6.15 ábrán látható, a projekt kommitolásnak visszakövetése, oldalból meg, hogy még milyen opciókkal rendelkezünk, például: ha rálépünk a Source opcióra eljön a forráskódunknak az utolsó verziója. Tulajdonképpen nyomon követhetjük a közös fejlesztés lépéseit, visszanezhetjük a kódon történt változtatásokat és egy biztonságos helyen tárolhatjuk a projekt forrás kódját.



Author	Commit	Message	Date
Tunde Nagy-Serbán	2587aa6	architektúra	5 days ago
Tunde Nagy-Serbán	5831f2e	kiadat	2021-03-17
AaronBison	6972773	Merge branch 'master' of https://bitbucket.org/partnerk/tunde-part	2021-03-05
AaronBison	687b863	Model update - Door labels and "Sapientia" label	2021-03-05
Tunde Nagy-Serbán	1bedae4	responzivitás verzió 1	2021-03-04
Tunde Nagy-Serbán	c54f425	Merge branch 'master' of https://bitbucket.org/partnerk/tunde-part	2021-03-04
Tunde Nagy-Serbán	6f5e886	dolgozatba pár átírás	2021-03-04
Tunde Nagy-Serbán	3bdc785	feltek berakása	2021-03-04
AaronBison	5f7085b	Mapping moved to outside json	2021-03-02
AaronBison	0b33bcf	More improved navigation paths + Villamosmérnöki tanácsok	2021-03-01
AaronBison	aab1e4b	Improved navigation paths	2021-02-26
Tunde Nagy-Serbán	43883d5	menü megváltoztatása	2021-02-25
Tunde Nagy-Serbán	9e03335	Merge branch 'master' of https://bitbucket.org/partnerk/tunde-part	2021-02-25
Tunde Nagy-Serbán	9f63b7b	átírások	2021-02-25
AaronBison	1124648	BUG FIX - Sidebar modifies model window size	2021-02-24
AaronBison	c8e1f29	Mattan routing bug fix	2021-02-24
AaronBison	e04528c	Merge branch 'master' of https://bitbucket.org/partnerk/tunde-part	2021-02-22
AaronBison	0cc8997	Animation implementation between two checkpoints	2021-02-22

6.15. ábra. Példa a Bitbucket verzió követésére

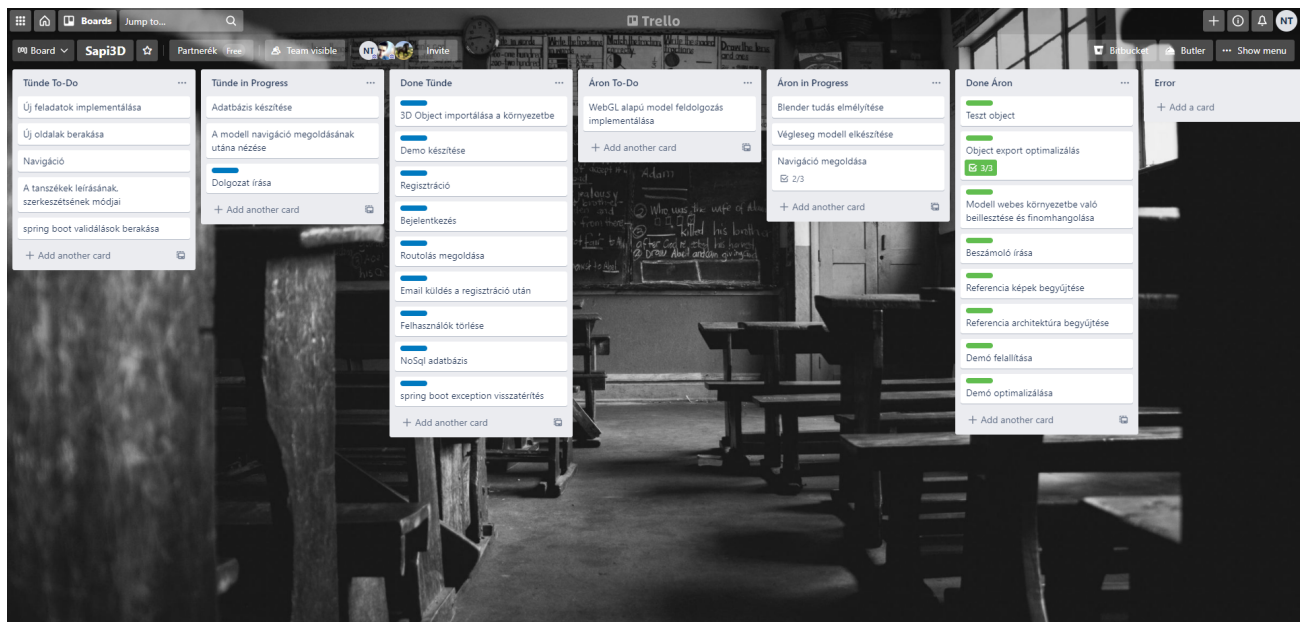
6.5.2. Trello

A Trello projektmenedzsment eszköz [23], amely a projekteket különböző táblákba rendezi. A 6.16 ábrán látható a saját projektmanadzselés. Megfigyelhetők a különböző oszlopok, amelyekben látha-

tunk kisebb feladatokat.

Az eszköz a *Kanban módszer*³ alapján a feladat folyamatát jelzi, mint például a To-Do oszlop jelenti azokat a feladatokat amelyeket még el sem kezdtek a fejlesztők. A Progress oszlop jelzi, hogy mely feladatokon dolgoznak a fejlesztők. A Done oszlop jelzi, hogy mely feladatok lettek elvégezve. A fejlesztők eltudják látni a feladatokat egy-egy színnel is, amely jelzi a többi fejlesztőnek, hogy azt a feladatot ki végzi. A feladatokat lehet mozgatni állapotuktól függően.

Egy jó tulajdonsága a Trello-nak, hogy össze lehet kötni a Bitbucket-tel így nem kell külön megnyitani a két felületet, hanem egy felület alatt meglehetősen nézni mindkettőt.



6.16. ábra. Példa a Trello projekt menedzsment eszközre

³<https://promanconsulting.hu/kanban-rendszer/>

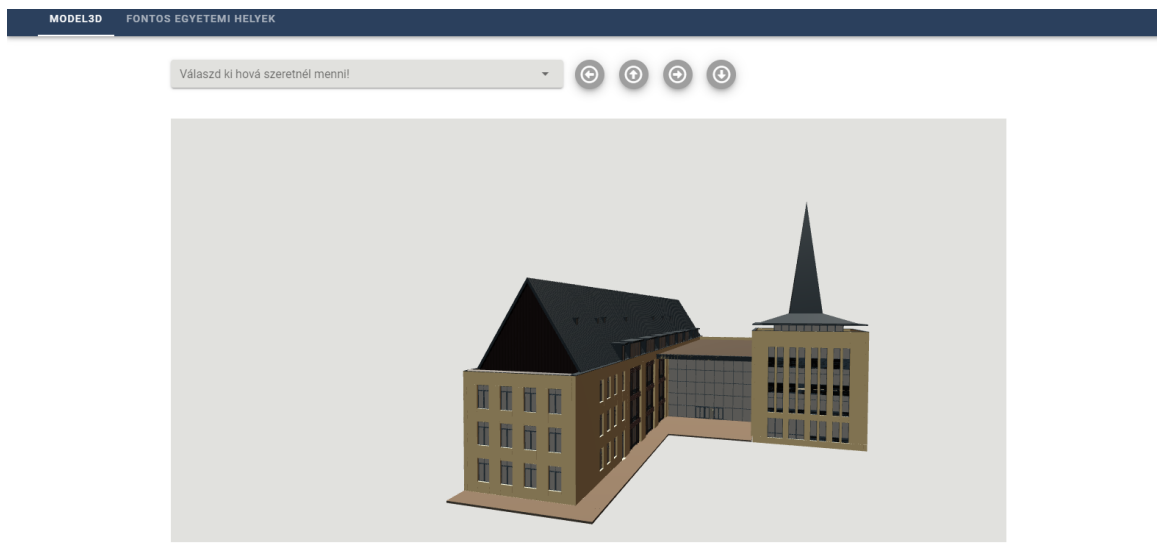
7. fejezet

A felhasználói felület bemutatása

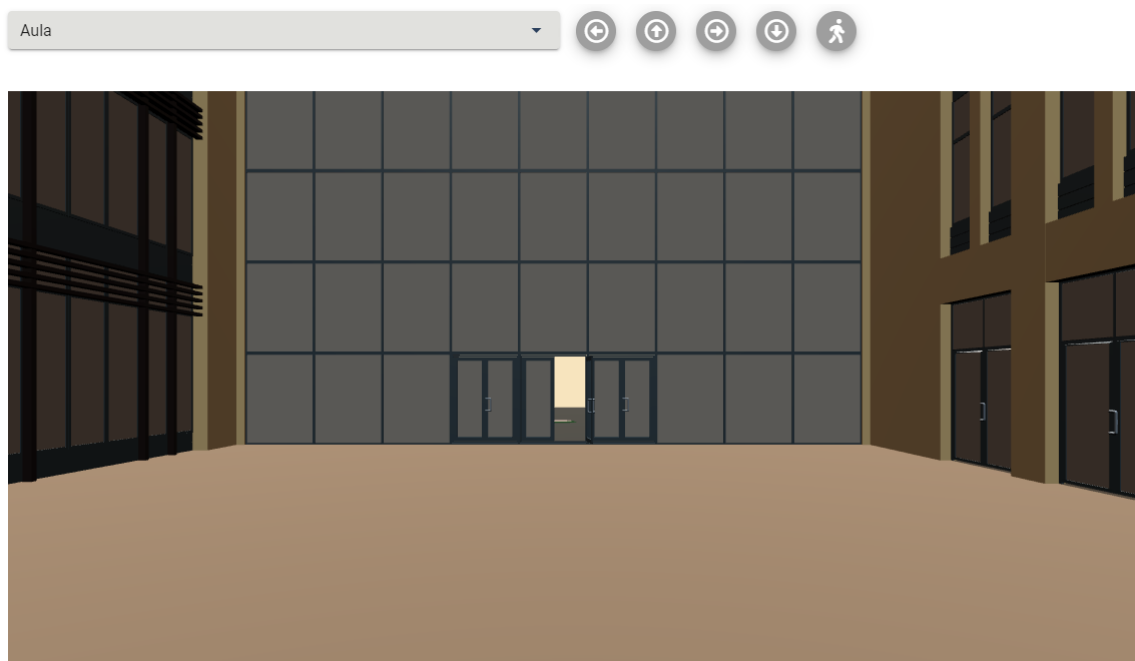
A felhasználó felületnek a képekben való bemutatása nehéz, főleg a "Vigyél el" funkció. Ezért van egy demó verzió amelyet jelenleg meglehet tekinteni a következő címen: <http://193.226.0.198:20024>.

7.1. A 3D modell megjelenése

Az alkalmazás indításakor elsőre a 3D modell jelenik meg. Ez látható a 7.1 ábrán. A modellen kívül megjelenik egy legördülő lista, ahol a Visitorok kitudnak választani egy fontos egyetemi helyet és plusz gomb megjelenésével eltudnak navigálni oda. Ez megfigyelhető a 7.2 ábrán. Látható hogy ki lett választva az Aula és 7.1 ábrától eltérően megjelent egy újabb gomb egy emberkével. Ez a gomb jelzi az alkalmazásnak, hogy a Visitor elszeretne jutni valahová. Ha ezt a gombot nyomjuk akkor az alkalmazás bemutatja a bejáratról az utat a kiválasztott helyre, esetünkben a bejáratról az Auláig.



7.1. ábra. 3D modell megjelenése az alkalmazásban



7.2. ábra. "Vigyél el" funkció részleges bemutatása

A 7.3 ábrán látható, hogy az ábrának az első részében még ott vagyunk a bejárat előtt. Ha megnyomjuk a navigálásra alkalmas gombot akkor átkerülünk az ábrának a második részében látható aulába. Ezen kívül az egér segítségével tudunk forogni és a navigációs gomb mellett láthatunk olyan

gombokat is amelyeken nyilak vannak. Ezeknek a nyilaknak a segítségével tudunk előre, jobbra, balra és hátra mozogni a modellben. A jobb megértés érdekében érdemes ellátogatni a fent említett címre és kipróbálni a funkciókat.



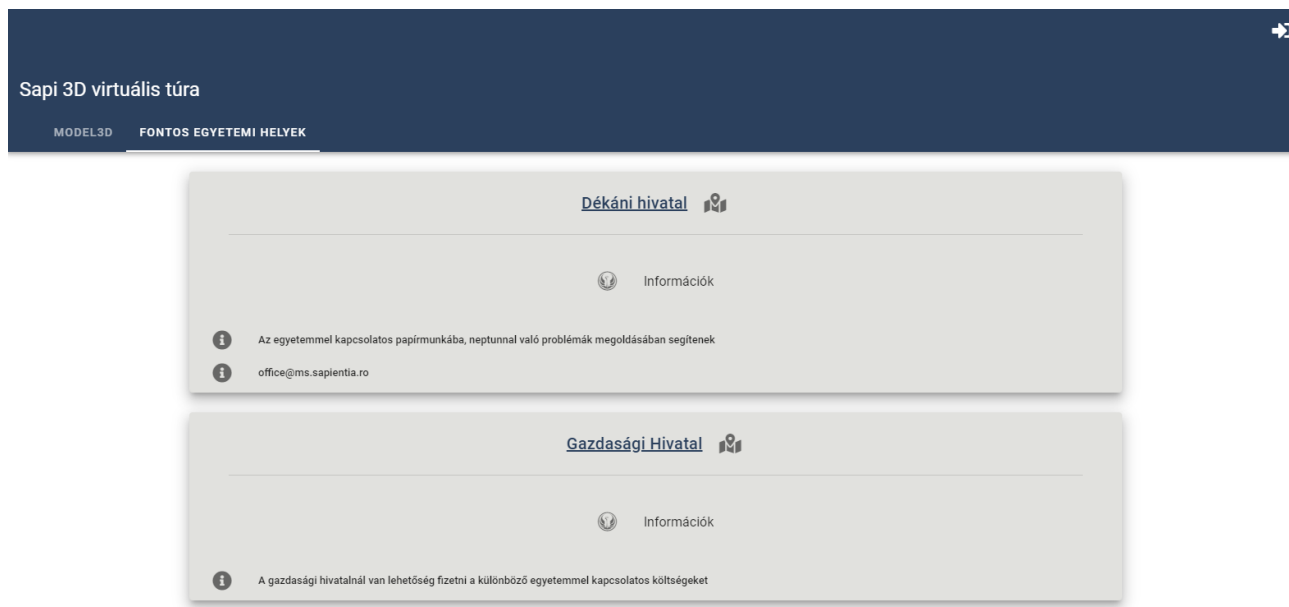
7.3. ábra. Útvonal megmutatása a bejárattól az Auláig

7.2. Fontos egyetemi helyek megjelenése

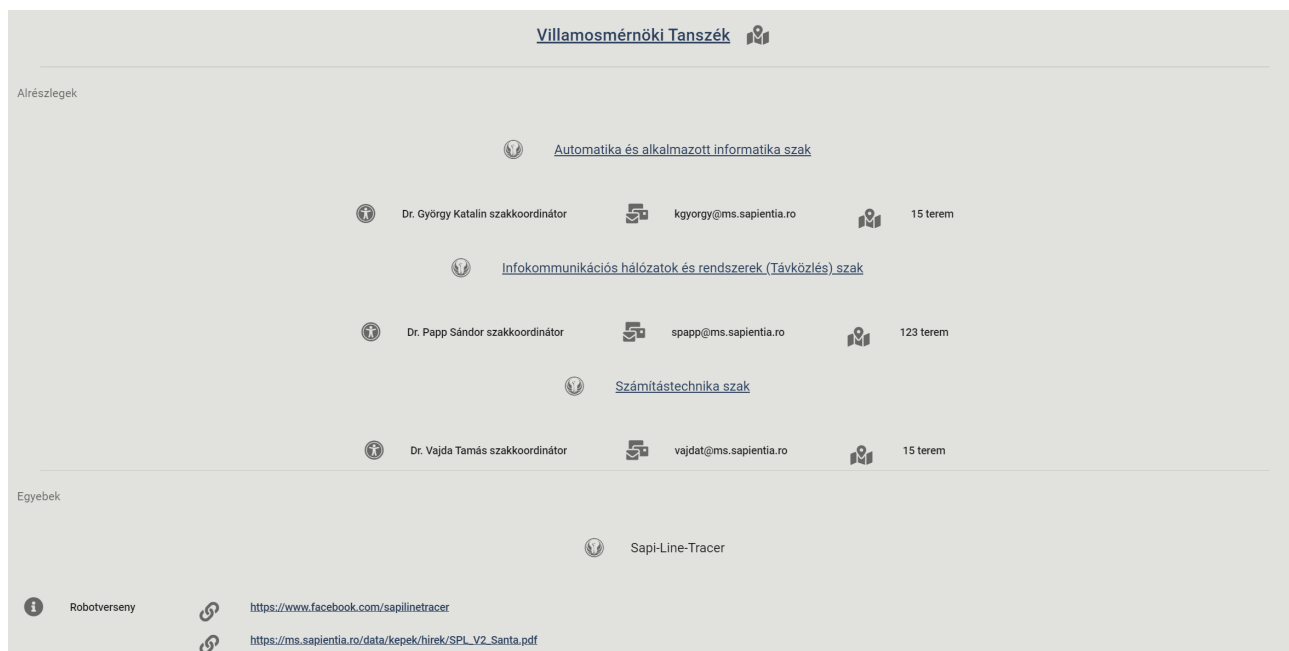
A fontos egyetemi helyeknek a megtekintéséhez átkell lépniünk a következő menüpontra. Ha átléptünk a második menüpontra, akkor a 7.4 ábrán látható képpel fogunk találkozni. Itt megjelennek a tanszékek és különböző hivatalok. Ezeken belül megjelennek a különböző alrészlegek vagy egyéb fontos információk az adott részleggel kapcsolatban. A címekre kattintva átkerülünk egy másik oldalra ahol, bővebb információkat találunk az adott részlegről. A cím mellett található ikonra kattintva vissza kerülünk a modellhez és az útvonal lesz bemutatva, úgy ahogy az előző részben bemutatódott a bejárat és az Aula között út.

A 7.5 ábrán bemutatnám miként jelenik meg egy tanszék az alkalmazáson belül. Először is megjelenik a tanszék neve, esetünkben a Villamosmérnöki Tanszék. Ezek után két csoportosítást vehetünk észre. Az alrészlegek tartalmazzák a tanszékhez tartozó szakokat. A jelenlegi ábrán látható az Automatika és alkalmazott informatika, Infokommunikációs hálózatok és rendszerek (Távközlés) és Számítástechnika szakok. A szakok alatt láthatóak információk az adott szakkal kapcsolatban, mint például a szakkoordinátorok neve. Ezek után megjelennek az egyebek kategória, ami az adott tanszékhez tartozó versenyeket (Sapi-Line-Tracer robotverseny), eseményeket írnak le. A 7.6 ábrán látható egy kép amely bemutatja, hogyan lehet eljutni a Villamosmérnöki tanszékre. A bejárattól kezdődően indulva beérkezünk az aulába, ahonnan tovább haladva elérünk a Gépészmérnöki Tanszék bejára-

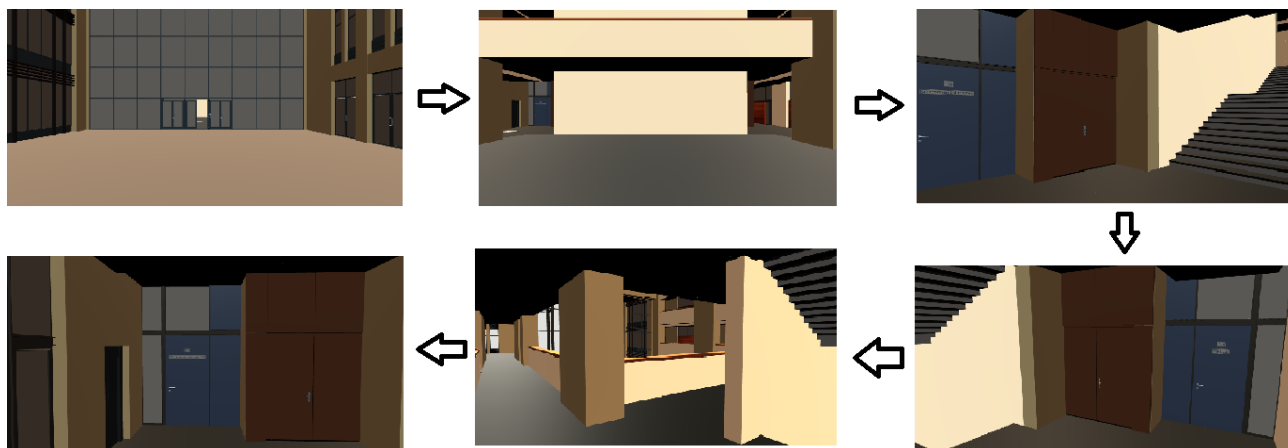
tához. A következő lépésben felmegyünk a lépcsőn és megérkezünk a Titkárság ajtajához. Innen megfordulva és tovább haladva eljutunk a célállomáshoz a Villamosmérnöki Tanszék ajtója elé.



7.4. ábra. Fontos egyetemi helyek megjelenése az alkalmazásban



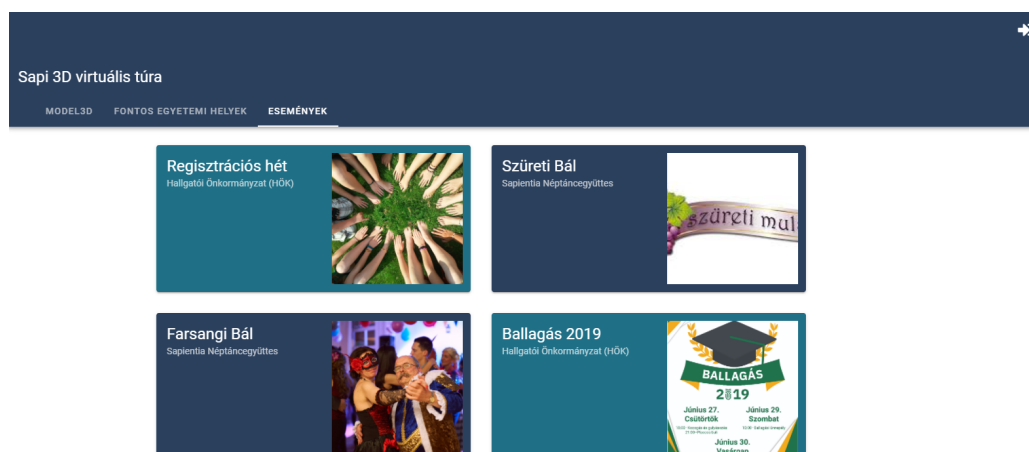
7.5. ábra. Példa a Villamosmérnöki Tanszék megjelenésére



7.6. ábra. Út a Villamosmérnöki Tanszékhez a bejáratától

7.3. Események megtekintése

Az események menüpont alatt tekinthetők meg az egyetemen rendezett események listája. Itt nem csak azok az események jelenhetnek meg, amelyeket egy tanszéken belül rendeznek, hanem a diákok által szervezett események is feltüntethetők. A felhasználói felület látható 7.7 ábrán, ahol látható, hogy az események kártyákon jelennek meg. A kártyák tartalmazzák a az esemény nevét, a szervező nevét és egy képet az eseményről. A kártyákra kattintva az oldal átirányít egy másik oldalra, ahol több információt lehet megtudni az adott eseményről.

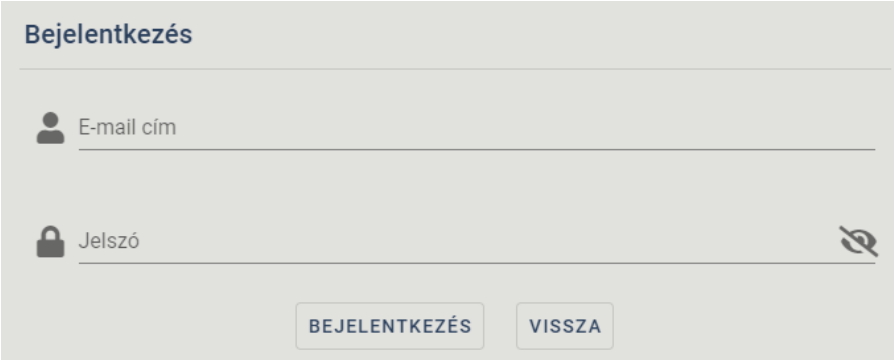


7.7. ábra. Események megtekintése

7.4. Bejelentkezés és kijelentkezés

A bejelentkezés ablakáról láthatunk képet a 7.8 ábrán, ahová a felhasználó betudja írni az e-mail címet és a jelszót. Mindkét mezőt kötelező kitölteni és mindkét mezőbe a megfelelő formátumot kell írni. Jelszó esetében kis- és nagybetűt valamit számot. A Bejelentkezés gombra kattintva tudunk bejelentkezni, ha megfelelő jelszóval és e-mail címmel rendelkezünk. A jelszót megis tudjuk tekinteni ha rálépünk a label végében található áthúzott szemre.

A kijelentkezést a weboldal sarkában megjelenő kijelentkező gomb használatával tehetjük meg.



The image shows a login form with a light gray background. At the top, the title 'Bejelentkezés' is displayed in a dark blue font. Below the title, there are two input fields. The first field is labeled 'E-mail cím' and has a person icon to its left. The second field is labeled 'Jelszó' and has a lock icon to its left. To the right of the password field, there is a small icon of an eye with a diagonal line through it, indicating a toggle for password visibility. At the bottom of the form, there are two buttons: 'BEJELENTKEZÉS' and 'VISSZA', both in a light gray box with dark blue text.

7.8. ábra. Bejelentkezési ablak megjelenése az alkalmazásban

7.5. Jelszó hitelesítése

A jelszó hitelesítésére szolgáló űrlapot a 7.9 ábrán láthatjuk, ahol megjelenik három label, amelyeket ki kell tölteni a tokennel, a jelszóval és a jelszó újrával. A sikeres jelszó megadás akkor lesz, ha a token létezik és a felhasználási idő nincs lejárva. Ezen kívül a két megadott jelszó meg kell egyezzen és kell tartalmazzanak kis- és nagybetűt valamint számot is.

Jelszó megadása

Kedves Felhasználó! Regisztrációja véglegesítéséhez szükséges megadnia a jelszavát és azonosító kódját. A jelszó és az azonosító kód megadásakor kötelezően kell szerepeljen kis, nagy és szám karakter.

Azonosító kód 0 / 32

Jelszó ✖

Jelszó újra ✖

JELSZÓ MEGERŐSÍTÉSE

7.9. ábra. Jelszó hitelesítésére szolgáló űrlap

7.6. User adatainak szerkesztése

A Usernek a bejelentkezés után lehetősége van a saját adatainak a szerkesztésére. A bejelentkezés után a weboldal jobb sarkában megjelenik egy ember ikon, amelyre ha rákattint előjönnek az adatai és a szerkesztési lehetőség, lásd a 7.10 ábrán. A szerkesztés esetén egy sort sem lehet üresen hagyni, különben a szerkesztés gomb nem fog működni.

Személyes adatok változtatása

Név
Nagy Tunde

E-mail
usersapi3d@gmail.com

Telefonszám
0323636518

SZERKESZTÉS **VISSZA**

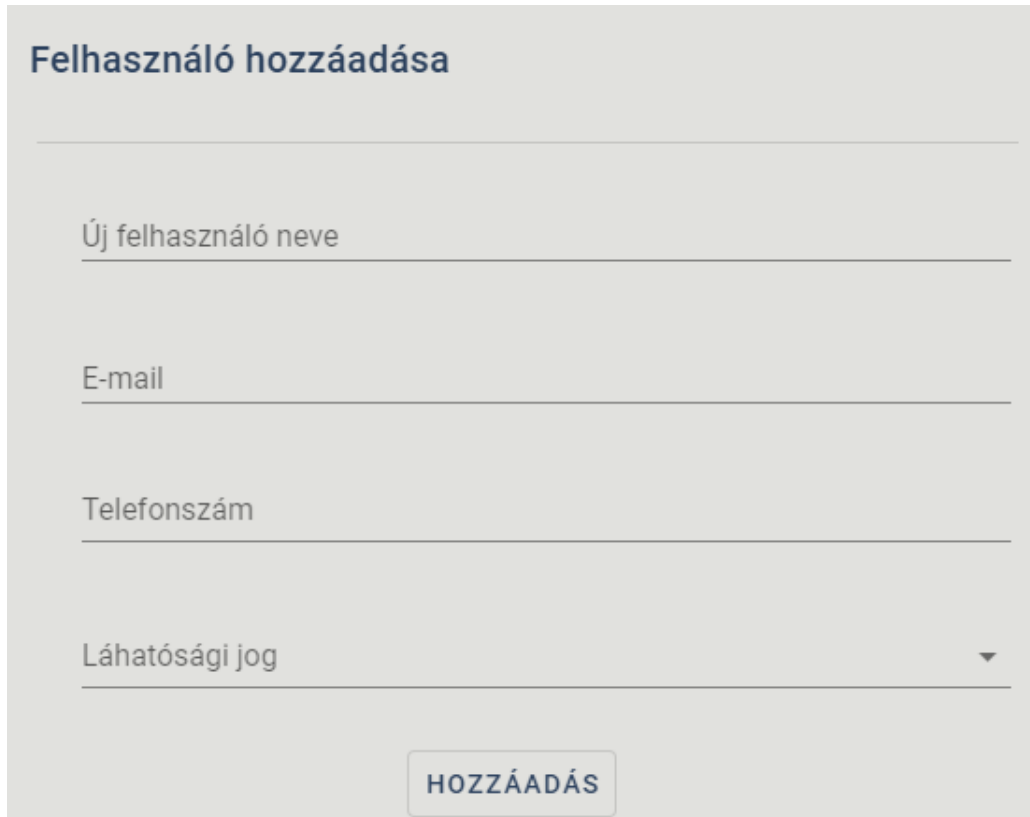
7.10. ábra. User adatok szerkesztésére használható rész

7.7. User regisztrálása és törlése

A User regisztrálását és törlését az Admin végzi. 7.11 ábrán látható egy űrlap, amellyel az új User adatait tudjuk regisztrálni. Be kell írni a User nevét, a telefonszámát és az e-mail címét. A Hozzáadás

gomb megnyomásával adjuk, hozzá a User-t. Attól függően, hogy a hozzáadás sikeres volt vagy nem a gomb alatt megfog jelenni egy üzenet.

A User törlésére is szintén egy űrlapot kell használni, amely a 7.12 ábrán látható. Első sorban megjelenik egy legördülő lista a Userek e-mail címével. Innen kiválasztva a megfelelő e-mail címet lekérjük a User adatait. Majd ha az adatok alapján a megfelelő személyt választottuk ki, akkor TÖRLÉS gomb lenyomásával kitöröljük a User-t.



Felhasználó hozzáadása

Új felhasználó neve

E-mail

Telefonszám

Láthatósági jog ▼

HOZZÁADÁS

7.11. ábra. User regisztrálására használható rész

Felhasználó törlése

Válaszd ki a törlendő Felhasználót!

almakorte@gmail.com

FELHASZNÁLÓ KERESÉSE

Név	Tunde
E-mail cím	almakorte@gmail.com
Telefonszám	0253564515

TÖRLÉS

7.12. ábra. User törlésére használható rész

7.8. Részlegek megadása és szerkesztése

A részlegek megadása és szerkesztése is egy-egy űrlap segítségével történik. A részlegek megadását a 7.13 ábrán láthatjuk, ahol látszik, hogy megkell adni egy nevet és egy linket, ahol több információt tudunk elérni az adott részlegről, lásd a 7.14. Itt az első lépés a részleg kiválasztása, majd amiután megjelentek a részleg részletei lehet őket szerkeszteni és a szerkesztés gomb megnyomásával véglegesítődik a szerkesztés.

Egyetemi részleg megadása, szerkesztése

☒ Hozzáadás ☐ Szerkesztés

Részleg neve _____

A részleg részletes leírásához a link _____

HOZZÁADÁS

7.13. ábra. Részleg hozzáadására alkalmas űrlap

Egyetemi részleg megadása, szerkesztése

☐ Hozzáadás ☒ Szerkesztés

Válaszd ki a részleget!
Villamosmérnöki Tanszék ▼

RÉSZLEG KERESÉSE

Részleg neve
Villamosmérnöki Tanszék

Részleg link
<https://ms.sapientia.ro/hu/tanszekek/villamosmernoki-tanszek>

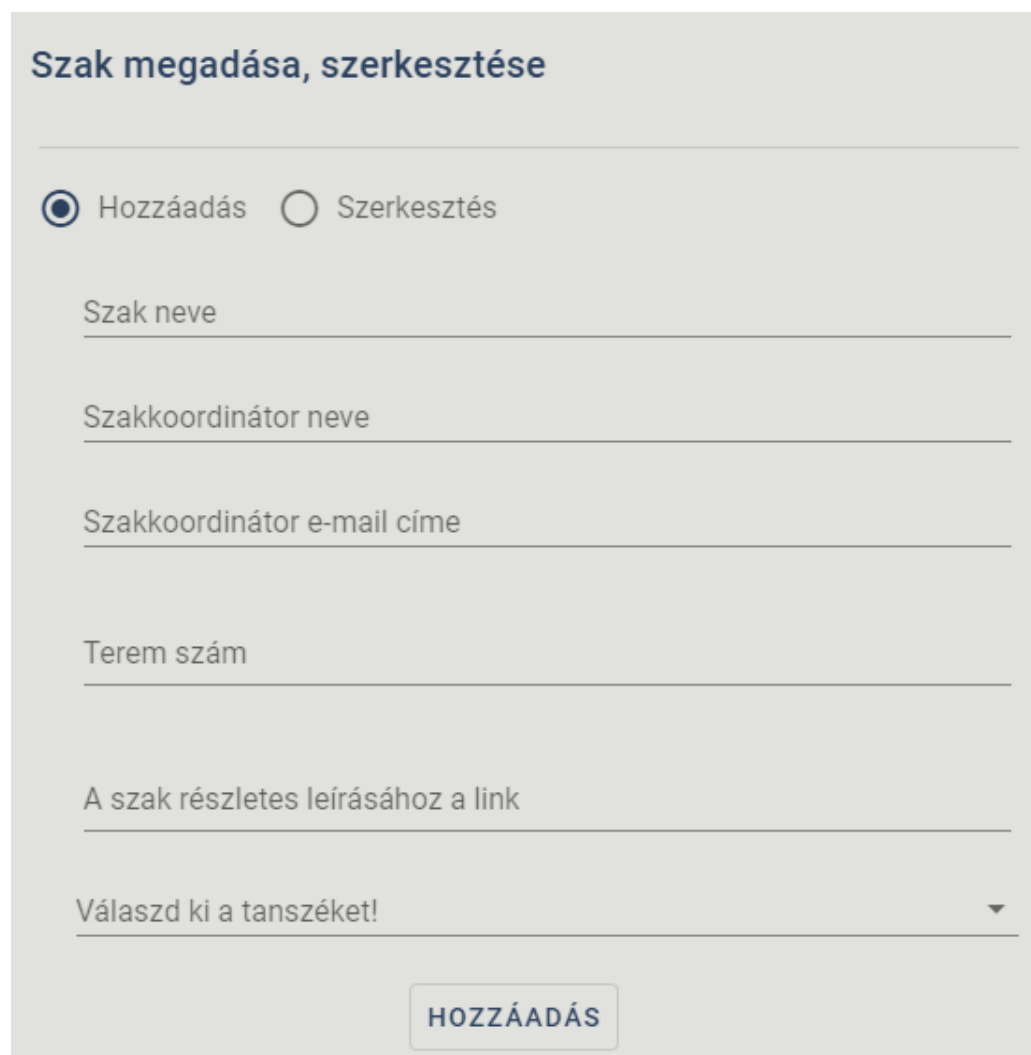
SZERKESZTÉS

7.14. ábra. Részleg szerkesztésére alkalmas űrlap

7.9. Szakok megadása és szerkesztése

A szakok megadása és szerkesztése hasonlóan működik, mint a részlegek megadása és szerkesztése.

Az erre alkalmas űrlapokat a 7.15 és a 7.16 ábrákon láthatóak.



The screenshot shows a web form titled "Szak megadása, szerkesztése" (Department addition/editing). At the top, there are two radio buttons: "Hozzáadás" (Add) which is selected, and "Szerkesztés" (Edit). Below this, there are six text input fields: "Szak neve" (Department name), "Szakkoordinátor neve" (Department coordinator name), "Szakkoordinátor e-mail címe" (Department coordinator email address), "Terem szám" (Room number), "A szak részletes leírásához a link" (Link to the detailed description of the department), and a dropdown menu labeled "Válaszd ki a tanszéket!" (Select the faculty). At the bottom center, there is a button labeled "HOZZÁADÁS" (Add).

7.15. ábra. *Részleg hozzáadására alkalmas űrlap*

Szak megadása, szerkesztése

☐ Hozzáadás ☒ Szerkesztés

Válaszd ki a szakot!

Automatika és alkalmazott informatika szak

SZAK KERESÉSE

Szak neve

Automatika és alkalmazott informatika szak

Szakkoordinátor neve

Dr. György Katalin

Szakkoordinátor e-mail címe

kgyorgy@ms.sapientia.ro

Terem szám

15

A szak részletes leírásához a link

<https://ms.sapientia.ro/hu/alapkepzesrol-bovebben/alapkepzesi-ta>

SZERKESZTÉS

7.16. ábra. Részleg szerkesztésére alkalmas űrlap

7.10. Egyebek megadása

Az egyebek megadására az űrlap a 7.17 ábrán látható, ahol ki kell választani egy részleget, majd lehet hozzáadni szövegrészeket, linkeket. A szövegrészekhez szükséges label-t az űrlap alján található billentyűzet ikonra való kattintással végezhetjük el. A mellette található ikonnal tudjuk a linkek hoz-

záadását elvégezni. Egy ilyen egyéb kategória megadásánál legalább ki kell tölteni egy szöveges labelt és egy linket tartalmazó labelt is.



7.17. ábra. Egyebek hozzáadására alkalmas űrlap


7.11. Események megadása

Az események megadására az űrlap a 7.18 ábrán látható, ahol kötelezően meg kell adni az esemény nevét, az esemény szervezőjét és egy eseménylinket. Ezek mellett kötelezően fel kell tölteni egy képet az adott eseményről. Az adatok megadása és a kép feltöltése után a hozzáadás gomb segítségével tudjuk elvégezni a hozzáadást.

Esemény hozzáadása

Esemény neve

Esemény szerveője

 Esemény kép

Esemény linkje

HOZZÁADÁS

7.18. ábra. Az események hozzáadására alkalmas űrlap

8. fejezet

Összefoglalás

A projekten belül egy olyan szoftver rendszer jött létre, amely segítséget nyújt a Sapientia EMTE Marosvásárhely-i karának főépületében való tájékozódásra egy 3D modell segítségével. A megvalósított tervek a következőkben lehet olvasni:

- Az alkalmazás fejlesztése közben, figyelembe lett véve, hogy minél hatékonyabb és kezelhetőbb legyen az esetleges látogatók számára.
- Segítséget nyújt a Visitoroknak, amelyek lehetnek vendégtanárok, diákok és nem utolsó sorban kutatók.
- Az alkalmazás, nem csak a tájékozódásban segíthet, hanem információkat is szolgáltat a Visitorok számára a tanszékekről, hivatalokról és a különböző eseményekről is.
- A "Vigyél el" funkció részleges megvalósítása.
- Egy link gyűjtemény összegyűjtése, amelyek segítséget nyújtanak a Visitoroknak.
- Az Admin és a User szerepkörök megvalósítása.

Összességében sikerült megvalósítani egy működő alkalmazást, amely nem csak segít a tájékozódásban, hanem az egyetemet is népszerűsíti.

8.1. Továbbfejlesztési lehetőségek

A céloknak megfelelően néhányat sikerült megvalósítani viszont még rengeteg ötlet van. A következőkben pár ilyen ötletéről lehet olvasni:

- A modellben való tájékozódás kiegészítése, pontosítása.
- Több útvonal megadása.
- Még több információ közlés a Visitorok számára (például: Melyik terembe milyen óra van és az órát ki tartja).
- A HTTPS protokoll bekonfigurálása a rendszerbe.

8.2. Köszönetnyilvánítás

Köszönetet szeretnék mondani konzulens tanáromnak dr. Szántó Zoltánnak, a Sapientia EMTE egyetemi adjunktusának, akinek segítségével jöhetett létre ez a dolgozat. Köszönetet szeretnék mondani a Sapientia EMTE Marosvásárhely-i Karának aki biztosított egy számítógépet, amely segítségével az alkalmazás nyilvánosságra kerülhetett. Szintén köszönetet szeretnék mondani kollégámnak, Blénesi Áronnak aki megalkotta a 3D modellt.

Irodalomjegyzék

- [1] T. P. Kersten, F. Tschirschwitz, and S. Deggim, „Development of a virtual museum including a 4d presentation of building history in virtual reality,” *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 361, 2017.
- [2] D. Dedov, M. Krasnyanskiy, A. Obukhov, and A. Arkhipov, „Design and development of adaptive simulators using 3d modeling,” *International Journal of Applied Engineering Research*, vol. 12, no. 20, pp. 10415–10422, 2017.
- [3] R. K. Moloo, S. Pudaruth, M. Ramodhin, and R. B. Rozbully, „A 3d virtual tour of the university of mauritius using webgl,” in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 2891–2894, IEEE, 2016.
- [4] D. Perdana, A. I. Irawan, and R. Munadi, „Implementation of a web based campus virtual tour for introducing telkom university building,” *International Journal of Simulation—Systems, Science & Technology*, vol. 20, no. 1, pp. 1–6, 2019.
- [5] R. Napolitano, I. Douglas, M. Garlock, and B. Glisic, „Virtual tour environment of cuba’s national school of art,” *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 42, no. 2, p. W5, 2017.
- [6] C. Maines and S. Tang, „An application of game technology to virtual university campus tour and interior navigation,” in *2015 international conference on developments of E-systems engineering (DeSE)*, pp. 341–346, IEEE, 2015.
- [7] „Database Management System Tutorial.” <https://www.tutorialspoint.com/dbms/index.htm>.

- [8] S. Venkatraman, K. Fahd, S. Kaspi, and R. Venkatraman, „Sql versus nosql movement with big data analytics,” *Int. J. Inform. Technol. Comput. Sci*, vol. 8, pp. 59–66, 2016.
- [9] A. Gupta, S. Tyagi, N. Panwar, S. Sachdeva, and U. Saxena, „Nosql databases: Critical analysis and comparison,” in *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, pp. 293–299, IEEE, 2017.
- [10] P. Martins, M. Abbasi, and F. Sá, „A study over nosql performance,” in *World Conference on Information Systems and Technologies*, pp. 603–611, Springer, 2019.
- [11] „Frameworks.” <https://hackr.io/blog/what-is-frameworks>.
- [12] „Top 10 emerging backend frameworks in 2021.” <https://www.decipherzone.com/blog-detail/top-10-backend-development-frameworks>.
- [13] „The Most Popular Backend Frameworks for Web Development in 2020.” <https://www.intagleo.com/blog/most-popular-backend-frameworks-for-web-development-in-2019/>.
- [14] E. Wohlgethan, *Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js*. PhD thesis, Hochschule für Angewandte Wissenschaften Hamburg, 2018.
- [15] „Vue and Angular comparison.” <https://www.educative.io/blog/react-angular-vue-comparison>.
- [16] N. Js and N. JS, „Node.js,” *Tradução de: SILVA, AG Disponível em*, 2016.
- [17] „NodeJs vs Spring Boot.” <https://www.chapter247.com/blog/node-js-vs-springboot-java-which-one-to-choose-and-when/>.
- [18] „Spring Framework.” <https://spring.io/projects/spring-framework>.
- [19] Ž. Jovanović, D. Jagodić, and Vujičić, „Java spring boot rest web service integration with java artificial intelligence weka framework,” in *International Scientific Conference “UNITECH 2017*, pp. 323–327, 2017.

- [20] „NodeJS vs Spring Boot : Picking up the right Technology.” <https://www.inexture.com/nodejs-vs-spring-boot-choosing-the-best-technology/>.
- [21] B. Áron, „Sapi3d tour – 3d model, tudományos diákköri dolgozat,” *Sapientia EMTE*, 2021.
- [22] „Bitbucket: What is Bitbucket?.” <https://confluence.atlassian.com/confeval/development-tools-evaluator-resources/bitbucket/bitbucket-what-is-bitbucket>.
- [23] „What is Trello?.” <https://help.trello.com/article/708-what-is-trello>.

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

Ș.l. dr. ing Szabó László Zsolt