UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,

TÎRGU-MUREȘ

SPECIALIZAREA CALCULATOARE

Aplicație web CRS

(Class Response System)

PROIECT DE DIPLOMĂ

Coordonator științific:

Ș.l.dr.ing. Szabó László Zsolt

Absolvent:

Udvari Balázs

2021

UNIVERSITATEA „SAPIENTIA" din CLUJ-NAPOCA **Viza facultății:**
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: **Calculatoare**

LUCRARE DE DIPLOMĂ

Coordonator științific:	Candidat: **Udvari Balázs**
ș.l. dr. ing. Szabó László Zsolt	Anul absolvirii: **2021**

a) Tema lucrării de licență:
Aplicație web CRS (Class Response System)

b) Problemele principale tratate:
- Studiu bibliografic privind aplicații tip CRS
- Realizarea unei aplicații CRS utilizând tehnologia Node.js
- Testare aplicație

c) Desene obligatorii:
- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat, diagrama bază de date proiectat

d) Softuri obligatorii:
-Aplicație web tip CRS

e) Bibliografia recomandată:
-H. Bakonyi V., ifj. Illés Z., Illés Z.: Valós idejű oktatást segítő rendszerek, INFODIDACT 2017, 10. Informatika Szakmódszertani Konferencia, Zamárdi, 23-25.11.2017.
- H. Bakonyi V., Illés Z.: Valós idejű oktatási rendszer, INFODIDACT 2018, 11. Informatika Szakmódszertani Konferencia, Zamárdi, 22-24.11.2018.
-A. Young, B.Mech, M. Cantelon: Node.js in Action, Second edition, Manning, 2017.

f) Termene obligatorii de consultații: săptămânal
g) Locul și durata practicii: Universitatea „Sapientia" din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Primit tema la data de: 31.03.2019.
Termen de predare: 05.07.2021.

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului programului de studiu

Semnătura candidatului

Declarație

SubsemnatulUdvari Balázs, absolvent alspecializării Calculatoare , 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.

Localitatca, Tg-Mureș

Data: 06.07.2021.

Absolvent

Semnătura

Aplicație web CRS
(Class Response System)

Extras

În cadrul proiectului de diplomă am elaborat a aplicație web din categoria *Class Response Systems* (CRS), care vine în ajutorul profesorilor pentru a crește gradul de interactivitate la orelor de curs. Alegerea a fost determinată de preocuparea mea privind transformarea orelor de curs spre interactivitate și experiență educațională trăită cu intensitate. Am studiat avantajele și dezavantajele unor aplicații CRS existente. Scopul meu a fost crearea unei aplicații utilizabilă gratuit, independent de platforme, și aplicabilă atât la cursuri face-to-face cât și în cadrul învățământului online. Aplicația va permite în viitor măsurarea eficienței procesului de predare, precum și a gradului de înțelegere a conceptelor predate în cadrul orelor de curs. Astăzi instituțiile de învățământ găsesc greu soluții pentru a rezolva problema de distragere a atenției elevilor în cadrul orelor cauzată de către dispozitivele conectate la Internet. Dar în același timp, dispozitivele digitale au și un potențial mare de a fi utilizate pentru creșterea eficienței orelor. În cursul lucrării mele am tratat noțiunile care stau la baza aplicațiilor CRS, am comparat câteva sisteme existente, și am prezentat fazele proiectării unei aplicații CRS proprii, precum și modul de testare și utilizare a aplicației. Numită ***answeRS***, aplicația mea este o aplicație web elaborată în mediul Node.js și realizează principalele funcționalități cerute de metodologia CRS. Aplicația asigură prezentarea de către profesori a unor suite de întrebări interactive în cadrul cursurilor. Profesorii au posibilitatea să-și organizeze întrebările conform disciplinelor predate și a grupurilor de elevi sau studenți, să le prezinte interactiv în cadrul cursului și să verifice corectitudinea răspunsurilor date interactiv prin utilizarea de dispozitive mobile de către studenți. Astfel procesul dă profesorilor posibilitatea de-a evalua atenția acordată materiei predate precum și gradul de înțelegere a elevilor în cursul orelor. Aplicația a fost testată și în condiții reale online cu un grup de utilizatori, și a funcționat conform cerințelor.

Cuvinte cheie: Class Response System, echipament clicker, metode timp real în învățământ

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK

VALÓS IDEJŰ OKTATÁST TÁMOGATÓ CRS (CLASS RESPONSE SYSTEM) WEB ALKALMAZÁS

DIPLOMADOLGOZAT

Témavezető:
Dr. Szabó László Zsolt,
egyetemi adjunktus

Végzős hallgató:
Udvari Balázs

2021

Kivonat

Diploma dolgozatként egy valós idejű oktatást segítő web alkalmazást készítettem (Class Response System – CRS típusú alkalmazást). Foglalkoztatott a hagyományos tanórák interaktívabbá, hatékonyabbá, a diákok számára élményalapúbbá tétele, ezért választottam ezt a témát. Tanulmányoztam létező valós idejű oktatást támogató rendszereket és azok előnyeit és hátrányait. Célom egy ingyenesen használható, költséghatékony, platformfüggetlen, hagyományos ("face-to-face") és online oktatásban egyaránt használható, oktatást támogató rendszer létrehozása, annak érdekében, hogy méréseket végezhessek mennyire hatékony egy ilyen rendszer, és mennyivel teszi hatékonyabbá az oktatást. Az oktatási intézmények nehezen tudnak versenyezni az internettel és az okos eszközökkel a diákok figyelmének lekötése terén, akiknek korosztálya már beleszületett az internet és az okos eszközök világába. A digitális eszközökben olyan lehetőség rejlik, amely nem csak felkelti a diákok érdeklődését, hanem fokozzák a tanórák hatékonyságát. Így ezek az eszközök felhasználhatóak CRS alkalmazások válasz adó eszközeiként. Dolgozatomban tárgyalom az oktatást segítő rendszerek alapfogalmait, összehasonlítok néhány elterjedt valós idejű oktatást támogató alkalmazást, továbbá ismertetem az általam készített CRS web alkalmazás tervezését, megvalósítását, működését és tesztelését. A saját, ***answeRS*** nevű alkalmazás egy Node.js környezetben tervezett web alkalmazás, amelyik megvalósítja a CRS rendszerek fontosabb funkcionalitásait. Az alkalmazás lehetőséget ad arra, hogy az okos eszközök oktatásba való bevonásával a tanár kérdéseket intézzen a diákok felé, azok valós időben megválaszolják azokat, majd automatikusan osztályzatot kapjanak rájuk. A tanár számára lehetővé teszi, hogy megtekintse az egyes diákok válaszait, osztályzatait, és nyomon kövesse a diákok egyéni bekapcsolódását az órákba. Az alkalmazás tesztelését elvégeztem online körülmények között, több felhasználóval kapcsolódva a tanóra kérdéssoraihoz.

Kulcsszavak: Class Response System, valós idejű oktatási módszer, clicker eszköz

Abstract

The main subject of my thesis are Class Response Systems, software applications created in order to help real time teaching methods. During the last years one of my main interest was the transformation of high school or university lectures by introducing interactivity and creating a better learning experience for the students. Consequently, my scope was the creation of a freely usable, low cost, platform independent CRS application, usable during traditional and online classes also. Creating the application, my long distance goal is to measure the efficiency of classroom teaching. Nowadays educational institutions frequently face problems caused by the distraction of students attention during classes by Internet connected smart devices. But smart devices, like mobile phones owned by almost all students could be used also for raising the efficiency of understanding during classes. Here are CRS systems coming in. In my thesis I present the basic concepts of CRS applications, compare some of widely used CRS systems, afterwards I present the project phases of my own CRS system, including testing and it's usage in schools. My CRS application, called ***answeRS*** is a Node.js based web application which includes the basic functionalities of a CRS methods. The application assures teachers to display question queues during classes in real time, student responses are collected by using their mobile phones. Teachers can evaluate responses instantly, and consider the result for the next part of the lecture. Testing of the application with multiple users was made under real online condition and requirements were fulfilled.

Keywords: Class Response System, clicker device, classroom real time methods

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. Bevezető

Azért választottam ezt a témát, mert már pedagógia modulos hallgatóként is foglalkoztatott az, hogy hogyan lehetne úgy érdekesebbé, szórakoztatóbbá élményalapúbbá tenni a tanórákat a diákok számára, akár a felső, akár az alsó oktatásban, hogy mellette a tanórák hatékonysága ne csökkenjen, hanem növekedjen és a diákok aktívabbak legyenek.

A Valós idejű oktatást segítő rendszerek c. pedagógia államvizsga dolgozatomat a Class Rasponse System-ek témaköréből írtam, amelyben részleteztem az általános tudnivalókat ezekről a rendszerekről, valamint ismertetem ezek előnyeit, hátrányait és javasolt felhasználási módjukat.

Napjainkban túlnyomó többségben az Y generáció tagjaiból áll az egyetemek hallgatósága, akik az 1980-as és 1999-es évek között születtek és már kisiskolás korukban megtanultak számítógépet és mobiltelefont használni, amelyek nélkül el sem tudják képelni az életüket. [1] Az okos eszközök nem szabályozott használata elvonja a diákok figyelmét a tanórákon, ugyanakkor a jól szabályozott és ellenőrzött használatuk segíti a tananyag elsajátítását és felgyorsítja az információszerzést. Az oktatásba bevont alkalmazások által nyújtott médialehetőségek (kép, videó, hang, látványos diagramok) felkelthetik, illetve fokozhatják a hallgatóság érdeklődését. Az úgynevezett Audience Response System-ek (röviden: ARS) segítségével egy előadó kérdéseket intézhet a hallgatósága felé, és valós időben válaszokat kaphat rá. A Class Response System-ek (röviden: CRS) ezeknek a rendszereknek az oktatásban használt alcsoportja [1].

Célom egy ingyenesen használható, költséghatékony, és platformfüggetlen CRS webalkalmazás létrehozása volt, annak érdekében, hogy méréseket végezhessek mennyire hatékony egy ilyen rendszer, valamint egy olyan alkalmazás létrehozása, amelyet reményeim szerint használni fognak a Sapientia EMTE oktatói és hallgatói.

2. Szakirodalmi tanulmány

2.1. ICT eszközök

Az ICT (Information and Communication Technology) eszközök az információközlést és az oktatást elősegítő infrastruktúrát jelentik [3].

Az ICT eszközök részét képezik az ARS és CRS rendszerek, akárcsak az egyetemek belső könyvtári katalógusa vagy belső helyi elérésű hálózata (LAN - Local Area Network). Az ARS és CRS rendszerekhez tartozik egy ún. clicker eszköz (clicker device) (1. ábra), amely lehet egy külön erre a célra kifejlesztett eszköz, vagy személyi használatra tervezett eszköz, mint például egy okostelefon. Mindezeket a továbbiakban ismertetem.

[1]

1. ábra Clicker eszköz

[1] https://i.ebayimg.com/images/g/bRYAAOSwHmhV7dgY/s-l300.jpg

2.2. Audience Response System (ARS)

Az ARS olyan rendszert jelent, amely elősegíti az interaktivitás létrejöttét az előadó és a hallgatóság között, valamilyen okos eszköz, akár saját okoseszköz (Bring Your Own Device, BYOD) (2. ábra) segítségével. A Bring Your Own Device magyarul annyit tesz, hogy hozd a saját eszközödet [1].

[2]

2. ábra Clicker eszközként használt saját eszköz (BYOD)

A legtöbb ARS szoftver és hardver kombinációját használja kérdések bemutatására, válaszok elmentésére, és visszajelzés nyújtására a hallgatók felé. Az előadó megjeleníti a kérdéseket egy képernyőn, vagy kivetíti egy vászonra, és az előadás résztvevői válaszolhatnak a clicker-t vagy saját eszközt (alkalmazásfüggő) használva [1].

[2] https://www.ombea.com/assets/img/or/ombea-responseapp.png

A clicker egy olyan eszköz, amelyen nyomógombok találhatóak, és a nyomógombok megfelelelhetnek egy-egy, az előadó által a képernyőre vagy vászonra kivetített válasznak, az adott kérdésre. Egy gomb lenyomásával elküldhető a hozzá tartozó válasz, és az eredmény azonnal megjeleníthető egy ábrán, a képernyőn vagy a vásznon [1].

2.3. Class Response System (CRS)

Az oktásban alkalmazott Audience Response System funkciókkal rendelkező alkalmazásokat Class Response System-nek (röviden: CRS) nevezzük, de a Student Response System (röviden: SRS) elnevezés is használatos. A CRS egy gyors módja a hallgatók tudásának és véleményének felmérésére [1].

Segítségével interaktívabbá tehetők az órák, aktivizálhatjuk a hallgatóságot, és jobban leköthetjük a figyelmüket nagy létszámú előadás esetében is. A CRS-ek lehetővé teszik az oktató számára, hogy többválaszos kérdéseket intézzen a hallgatóság felé. A hallgatók válaszai valós időben megtekinthetők, és az oktató a hallgatók szükségleteihez igazíthatja a tanítási folyamatot (3. ábra) [1].

[3]

3. ábra Kérdésre adott válaszok megjelenítése valós időben

[3] https://www.clikapad.com/wp-content/uploads/2015/03/clikapad-hire.png

Az CRS általános tulajdonságai [1]:

- Az oktató könnyedén tehet fel interaktív kérdéseket;
- A hallgatok kockázatvállalásra való ösztönzése anonim válaszadási lehetőséggel;
- Segítségével felmérhető, hogy a hallgatók milyen mértékben értették meg a leadott anyagrészt;
- Lehetőséget ad visszajelzés eredményeinek megbeszélésére;
- A házi feladatokat és teszteket azonnal fogadja, valamint osztályzatot ad rájuk;
- Elmenti az osztályzatokat;
- Számon tartja a jelenlétet;
- Adatokat gyűjt

2.4. E-Lection

Az E-Lection egy CRS funkciókkal rendelkező rendszer, amelyet az Eötvös Loránd Tudományegyetem valósított meg, abból a célból, hogy könnyebben bevonják az egyre csökkenő létszámú és egyre passzívabb hallgatóságot az előadásokba [1].

Anyagi okokból csak az ingyenesen elérhető alkalmazások közül választhattak, amelyekhez nem szükséges clicker eszköz megvásárlása, és azért, mert a legtöbb ilyen rendszer anonim felhasználókkal dolgozik, valamint nem találtak olyan CRS-t, amelynél a válaszok között matematikai kifejezést is meg lehet adni, ezért egy saját webes alkalmazás fejlesztése mellett döntöttek, amelyben clicker-ként a hallgatók saját okos eszközeit használták. Ennek a megvalósításnak nagy előnye, hogy a későbbiekben könnyebben testre szabható újabb igények megjelenése esetén [1].

Az alkalmazás létrehozásához első sorban fel kellett mérjék mennyire elterjedtek a mobileszközök a hallgatók körében, amelyet kérdőívekkel valósítottak meg szlovák partnerek közreműködésével. A felmérés eredménye az 1. táblázatban látható [1]:

1. táblázat Mobil prezentációs kérdőív eredménye

Telefon típusa	**Összesen**	**Magyar**	**Szlovák**
Okos (2015)	95,4%	92,4%	98,7%
Okos (2016)	99,9%	99,9%	99, 9%
Hagyományos (2015)	3,9%	6,3%	1,3%
Hagyományos (2016)	0,0%	0,0% (10 darab)	0,0% (1 darab)
Semmilyen (2015)	0,7%	1,3%	0,0%
Semmilyen (2016)	0,0%	0,0% (1 darab)	0,0%

Mivel a cél az volt, hogy az alkalmazást használó hallgatók tevékenységét egyénien nyomon követhessék, ezért azonosított elérhetőséget választottak a névtelen (anonim) helyett.

Az első prototípus kétirányú kommunikációt is lehetővé tett, azaz amellett, hogy a tanár többválaszlehetőséges kérdéseket tehetett fel, a hallgatók az alkalmazás segítségével kérdéseket tehettek fel, vagy jelezhették, ha valamit nem értenek [1].

Egy weboldalon valós időben követni lehetett a tanári kérdésekre adott válaszok eloszlását grafikonokon megjelenítve, a hallgatók kéréseit és jelzéseit pedig lenyitható listákban lehetett elérni. A hallgatók az alkalmazásba azonosítás után léphettek be, a kari központi domain nevüket használva. Minden interakciót elmentettek egy adatbázisba. Később az alkalmazást kibővítették, hogy a jelenlét számon tartására is alkalmas legyen, amelyet széleskörűen használtak a kötelezően látogatandó előadások adminisztrálására.

Biztonsági okokból a hallgatók azonosítását áthelyezték teljesen szerver oldalra. A begyűjtött adatok letölthetőek és feldolgozhatóak. Az oktató listázhatja előadásonként és félévenként, a beérkezett hallgatói jelzéseket és kérdéseket, valamint az általa feltett kérdésekre adott válaszokat.

2017 őszén online kérdőíves felmérést készítettek, az elsős és harmadéves hallgatók körében, hogy pontos képet kapjanak a hallgatók igényeiről. A kérdőív a következő egytől ötig skálázott kérdéseket tartalmazta [1]:

1. A tanórák folyamán használ-e számítógépet vagy mobileszközt?
2. Jónak tartja-e, ha a saját mobil eszközét (telefon, laptop stb.) felhasználhatja az órák alatt?
3. Elősegítené-e a hatékonyabb tanulást, ha az órákon az előadó kérdéseket tenne fel az évfolyamnak az anyaggal kapcsolatban?
4. Hasznosnak tartaná, ha a kérdéseket elektronikus formában kapná meg és válaszolhatná meg?
5. Segítené-e az anyag gyorsabb elsajátítását, ha a kérdésekre adott válaszok helyes és helytelen voltát ott azonnal kiértékelnék?
6. Ha azonnal választ kapna, gyakrabban tenne fel kérdést az órákon?
7. Milyen gyakran fordul elő, hogy annyira elveszti az előadás fonalát, hogy kérdést sem tud megfogalmazni?
8. Szokott-e jelentkezni és kérdést feltenni előadás közben vagy inkább elhallgatja a problémáját, mert szégyelli magát?
9. Kérdezne-e többször az előadótól, ha a hallgató társai előtt anonim maradhatna?

10. Hasznosnak tartaná-e, ha az előadásokon kétoldalú kommunikáció zajlana az előadó és a hallgatóság között?
11. Mit gondol, segítené a felkészülését, ha az előadásokon elhangzó oktatói és hallgatói kérdéseket utólag megkapná, letölthetné?
12. Hasznosnak tartaná-e, hogy on-line elérhető legyen az előadás?
13. Újranézné-e a korábban felvett és már látott órát még egyszer?
14. Mit javasolna, amivel az órák érdekesebbé, vonzóbbá válhatnának?

A kérdőívekből kiderült, hogy a kérdőívet kitöltő hallgatók több, mint 50%-a igényli a kétirányú kommunikáció lehetőségét az előadásokon, továbbá felhasználná és újranézné az előadást videófelvételről. Az anonimitásra szintén nagy igényük volt a hallgatóknak, ezért az alkalmazás jelenlegi verziójában a hallgatók egymás előtt anonimek, viszont az oktató elérheti őket. A kérdőív utolsó kérdésére ("Mit javasolna, amivel az órák érdekesebbé, vonzóbbá válhatnának?") szabad szöveges formában lehetett válaszolni, amelyben két ERASMUS cserediákprogramban részt vett hallgató CRS szoftvert ajánlott. Az egyik szoftver a sli.do volt, a másik az edingburghi egyetem által használt TopHat rendszer [1].

2.5. Hasonló alkalmazások

2. táblázat Class Response System alkalmazások és jellemzőik

Név	**Kahoot** **https://kahoot.com/**	**Socrative** **https://socrative.com/**	**Slido** **https://www.sli.do/**	**VoxVote** **http://www.voxvote.com/**
Technológia	Web, mobil, alkalmazás	Mobil, alkalmazás	Web, mobil	Web, mobil
Eredmény	Mobilalkalmazásban	Mobilalkalmazásban	Slido admin alkalmazásban	Weben
Azonosítás	E-mail cím alapján	Anonim	Anonim	Anonim, becenév
Kérdés	Választható, szabad	Választható, szabad	Választható, szabad	Választható, szabad
Médiaelemek	Kép, Videó, Játék, Zene	Kép	-	-
Kétirányú	Igen, moderálással	Igen, moderálással	Igen, moderálással	Igen, moderálással
Export	Igen	Igen	Igen	
Ingyenes	Ingyenes	50 főig, alapfunkciókkal	Ingyenes alapfunkciók	

Név	Poll Everywhere https://www.polleverywhere.com	Mqlicker http://www.mqlicker.com/	OMBEA http://www.ombea.com/
Technológia	mobil, sms	Web, mobil, böngésző	Web, mobil
Eredmény	PowerPoint, Google Slides	PowerPoint	Weben
Azonosítás	Becenév	E-mail cím	Anonim, becenév
Kérdés	Választható, szabad	Választható	Választható, szabad
Médiaelemek	-	Kép, videó	-
Kétirányú	Igen, moderálással	Igen	Igen, moderálással
Export	Igen	Igen	Igen
Ingyenes	Fizetős	Ingyenes	Oktatásban ingyenes

2.6. Felhasznált technológiák

2.6.1. Node.js és Express.js

A Node.js egy nyílt forráskódú (open-source), platformközi (cross-platform) JavaScript futási környezet (run-time environment), melyet JavaScript kódot hajt végre a böngészőn kívül. A Node.js megengedi a fejlesztőknek, hogy parancssoreszközök (command line tools) írására és szerveroldali szkriptezésre (server-side scripting) használják a JavaScript-et. A szkriptek szerveroldalon való futtatása dinamikus weboldaltartalmat hoz létre, mielőtt az oldal elküldődne a felhasználó webböngészőjéhez. Következésképpen a Node.js egy "JavaScript mindenhol" paradigmát képvisel, amely egyesíti a webalkalmazás-fejlesztést egyetlen programozási nyelv körül, nem pedig szerver- és kliensoldali szkriptek különböző nyelvein. Bár a .js a JavaScript fájl standard fájlnévkiterjesztése, a "Node.js" név nem utal egy adott fájlra ebben az összefüggésben, ez csupán a termék neve. Ezeknek a tervezési döntéseknek a célja, hogy optimalizálják a teljesítményt és a skálázhatóságot a sok bemeneti/kimeneti művelettel rendelkező webalkalmazásokban, valamint a valósidejű webalkalmazásokban (például valósidejű kommunikációs programok és böngészős játékok) [18].

Skálázható internetes alkalmazások, mégpedig webszerverek készítésére hoztak létre. A programok JavaScript-ben írhatók, eseményalapú, aszinkron I/O-val a túlterhelés minimalizálására és a skálázhatóság maximalizálására. A Node.js a Google V8 JavaScript motorjából, a libuvból, és számos beépített könyvtárból áll [17]. A V8 motorra épült, mert nyílt

forráskódú a BSD licenc alatt. Több ezer nyílt forráskódú könyvtár áll rendelkezésére, amelyek többsége az npm webhelyen (https://www.npmjs.com) található [18].

A V8 a JavaScript végrehajtási motorja, amelyet eredetileg a Google Chrome számára fejlesztettek ki. Ezt követően a Google nyílt forráskódúvá tette 2008-ban. C++ nyelven íródva a V8 a JavaScript forráskódot futás közben netív gépi kódra fordítja ahelyett, hogy előzetesen értelmezné. A Node.js a libuvot aszinkron események kezelésére használja. A libuv egy absztrakciós réteg hálózati és fájlrendszer funkcionalításokra Windows és POSIX alapú rendszereken egyaránt úgy, mint a Linux, a macOS, az OSS a NonStop-on és a Unix. A Node.js alapvető funkcionalitásai egy JavaScript könyvtárban találhatók. A Node.js kötések, C++-ban íródván, összekapcsolja ezeket a technológiákat egymással és az operációs rendszerrel [18].

A Node.js implementálja a CommonJS specifikációk egy részét. Tartalmaz továbbá egy REPL rendszert az interaktív teszteléshez [17].

Lehetővé teszi webszerverek és hálózati eszközök létrehozását a JavaScript és különféle alapvető funkciókat kezelő “modulok” gyűjteményének felhasználásával. A Modulok fájlrendszer I/O, hálózati (DNS, HTTP, TCP, TLS/SSL vagy UDP), bináris adatok (pufferek), kriptográfiai függvények, adatfolyamok, és egyéb alapvető funkciók biztosításáért felelősek. A Node.js moduljai olyan API-t használnak, amelyet arra terveztek, hogy csökkentse a szerver-alkalmazások írásának összetettségét [18].

A JavaScript az egyetlen olyan nyelv, amelyet a Node.js natív módon támogat, de nagyszámú JS-re kompilálható (compile-to-JS) nyelv érhető el. Ennek eredményeképpen a Node.js alkalmazások írhatók CoffeScriptben, Dartban, TypeScriptben, ClojureScript-ben és más programozási nyelvekben [18].

Noha a modulrendszer kezdetben CommonJS modulmintán alapult, a modulok nemrégiben történt bevezetése az ECMAScript specifikációba megváltoztatta az ECMAScript modulok használatának irányát a Node.js-ben az alapértelmezett helyett [18].

A Node.js eseményvezérelt programozást hoz be a webszervereknek, lehetővé téve a gyors webszerverek fejlesztését JavaScriptben. A fejlesztők szálkezelés nélkül hozhatnak létre skálázható szervereket az eseményvezérelt programozás egy egyszerűsített modeljével, amely callback függvényeket használ egy adott feladat befejezésének jelzésére. A Node.js összeköti a szkriptnyelv (JavaScript) egyszerűségét a Unix hálózati programozás erejével. [18]

A Node.js bejövő kéréseket egy ciklusban (loop) dolgozza fel, amelyet eseményciklusnak (event loop) hívnak. A Node.js egyetlen szál eseményciklusban működik, nem blokkoló I/O hívásokat

használva, engedélyezve ennek, hogy több tízezer konkurens kapcsolatot támogasson anélkül, hogy a szálkörnyezetváltás költségei felmerülnének. Az egyetlen szál megosztásának tervezése az összes megfigyelő mintát használó kérés között erősen konkurens alkalmazások felépítésére szolgál, ahol minden I/O-t végrehajtó függvénynek callback függvényt kell használnia. A szálkészlet kezeli a párhuzamos feladatok végrehajtását a Node.js-ben. Amikor egy szál a szálkészletben elvégez egy feladatot, értesíti erről a főszálat, amely viszont felébred és végrehajtja a regisztrált callback függvényt [18].

Az npm, amely a világ legnagyobb szoftver tárolója, a Node.js szerverplatform előre telepített csomagkezelője. Telepíti a Node.js programokat az npm nyilvántartásból, megszervezi a harmadik féltől származó (third-party) Node.js programok telepítését és menedzsmentjét [18, 19]. A nyílt forráskód fejlesztői az npm-et használják a csomagok megosztására és kölcsönzésére, és sok szervezet az npm-et használja a magánfejlesztés menedzselésére. Az npm három különálló összetevőből áll: a weboldalból, a parancssori felületből (Command Line Interface – CLI) és a nyilvántartásból. A weboldal segítségével lehet böngészni a csomagok között és be lehet állítani profilokat. A parancssori felület egy terminálon fut, a legtöbb fejlesztő így lép interakcióba az npm-el. Amíg a releváns parancssori felület parancsokat a felhasználói dokumentáció tartalmazza, addig a parancssori felület tartalmaz parancssori súgót. A saját dokumentációs résszel és azonnali segítséggel (man oldalak). A nyilvántartás egy nagy nyilvános adatbázis a JavaScript szoftverekről és a körülvevő meta-információkról [19]. Az npm nyilvántartásban szereplő csomagok az egyszerű segítő könyvtáraktól, mint például a Lodash, egészen a feladat futtatókig (task runners) terjedhetnek, mint például a Grunt [18].

Ha a felhasználó nyilvánosan osztja meg a csomagjait az npm weboldalán, akkor annak nincs költsége. A privát csomagok használatához és megosztásához azonban frissítenie kell a fiókját. Ha a felhasználó meg szeretné osztani másokkal a privát csomagjait, akkor szervezetet kell létrehoznia, ún. npm Orgs-ot, és meghívhat másokat, hogy közösen dolgozzanak, privát módon díj ellenében, vagy nyilvánosan díjmentesen. Vagy feliratkozhat az npm egy privát példányára a vállalata számára, ún. npm Enterprise-ra, így belsőleg fejleszthet csomagokat, amelyek nincsenek megosztva nyilvánosan [19].

A Node.js kombinálható egy böngészővel, egy adatbázissal, amely támogatja a JSON adatokat (például a Postgresszel, a MongoDB-vel vagy a CouchDB-vel) és JSON-nal egy egységes JavaScript fejlesztési veremhez. Lényegében a szerveroldali fejlesztési minták alkalmazásával úgy,

mint az MVC, az MVP, az MVVM stb., a Node.js lehetővé teszi ugyanazon modell és szolgáltatási interfész újra-felhasználását a kliensoldal és a szerveroldal között. [18].

A Node.js együttműködik az operációs rendszerrel, így az operációs rendszer értesíti a csatlakozásokról és kibocsát egy callback függvényt. A Node.js futásidőn belül minden kapcsolat egy kis kupackiosztás (heap allocations). Hagyományosan, a viszonylag nehézsúlyú operációs rendszer folyamatok (process) vagy szálak kezeltek minden kapcsolatot. A Node.js egy eseményciklust használ a skálázhatósághoz a folyamatok vagy szálak helyett. Más eseményvezérelt szerverekkel ellentétben a Node.js eseményciklusát nem szükséges explicit módon meghívni. Ehelyett callback függvények vannak definiálva, és a szerver automatikusan belép az eseményciklusba a callback függvény definíciójának végén. A Node.js kilép az eseményciklusból, amikor nincs további végrehajtandó callback függvény [18].

A Node.js szálak nélküli tervezése nem jelenti azt, hogy ne lehetne kihasználni a környezetben lévő több mag előnyét. A fiú folyamatok létrehozása a child_process.fork() API segítségével történik, és úgy vannak megtervezve, hogy könnyen kommunikálhassanak. Ugyanerre az interfészre épült a klasztermodul, amely lehetővé teszi socketek megosztását a folyamatok között, hogy engedélyezzék a terheléselosztást a magok között [20].

A Node.js-t vállalati felhasználói közé tartozik a GoDaddy, a Groupon, az IBM, a LinkedIn, a Microsoft, a Netflix, a PayPal, a Rakuten, a SAP, a Voxer, a Yahoo! és a Walmart [18].

A Node.js-t hivatalosan támogatott Linuxon, macOS-en és Microsoft Windows 7 (és újabb) rendszereken, a SmartOS és az IBM AIX 2. Szintű támogatásával, valamint a FreeBSD kísérleti támogatásával. OpenBSD-n szintén működik, és elérhetők LTS verziók az IBM i (AS/400) számára. A mellékelt forráskód hasonló operációs rendszerekre is építhető, mint amelyeket hivatalosan támogatnak, vagy amelyeket harmadik fél módosított, hogy támogasson más rendszereket, úgy, mint a NonStop OS és a Unix szerverek [18].

A modern asztali integrált fejlesztői környezetek (IDE - Integrated Development Kit) szerkesztési és hibakeresési funkciókat nyújtanak kifejezetten a Node.js alkalmazásokhoz. Ezek közé az integrált fejlesztői környezetek közé tartozik az Atom, a Brackets, a JetBrains WebStorm, a Microsoft Visual Studio (Node.js eszközökkel a Visual Studió számára vagy TypeScripttel Node definíciókkal), a NetBeans, a Nodeclipse Enide Studio (Eclipse alapú) és a Visual Studio Code. Egyes online webalapú integrált fejlesztői környezetek szintén támogatják a Node.js-t úgy, mint a Codeanywhere, a Codenvy, a Cloud9 IDE, a Koding és a Node-RED vizuális folyamat-szerkesztője [18].

2015-ben, a nagyobb Node.js közösség különféle ágazatai elkezdtek az értékesítés-semleges Node.js Alapítvány (Node.js Foundation) alatt dolgozni. A szervezet kijelentett célja “a Node.js és más kapcsolódó modulok széleskörű elfogadásának lehetővé tétele és a fejlesztés felgyorsításának segítése egy nyílt irányítású modellen keresztül, amely ösztönzi a részvételt, a műszaki hozzájárulást és egy keretrendszert hosszú távú gondnokságra a Node.js sikerébe fektetett ökoszisztéma által”. A Node.js Alapítvány Műszaki Irányítóbizottsága (Technical Steering Committee – TCT) a Node.js Alapítvány műszaki irányító testülete. A Műszaki Irányítóbizottság felelős a Node.js magjának tárházáért (repository), valamint a függő és szomszédos projektekért. Általában a Műszaki Irányítóbizottság ezeknek a projekteknek az ügyintézését munkacsoportokra vagy bizottságokra ruházza rá. Az egyik ilyen csoport az LTS csoport, amely a hosszú távon támogatott (Long Term Support – LTS) kiadványokat kezeli. Egyéb jelenlegi csoportok a Weboldal (Website), az Adatfolyamok (Streams), a Felépítés (Build), a Diagnosztikák (Diagnostics), i18n, az Evangelism (Evangelizáció), a Docker, az Addon API, a Benchmarking, a Post-mortem, az Intl, a Dokumentáció (Documentacion) és a Tesztelés (Testing). A Node.js disztribúciós fejlesztési projektet, amelyet a Node.js Alapítvány irányít, a Linux Alapítvány (Linux Foundation) Együttműködő Projektek (Collaborative Projects) programja segítette elő [18].

A Node.js mag (core) GitHub tárházát az Együttműküdők tartják fenn, akiket a Műszaki Irányítóbizottság folyamatosan vesz fel. Azok a magánszemélyek, akik jelentős és értékes hozzájárulást nyújtanak, Együttműködőkké válnak, és commitolási hozzáférést kapnak a projekthez. Ezeket a magánszemélyeket a Műszaki irányítóbizottság azonosítja, és a kinevezésüket a meglévő Együttműködőkkel vitatják meg [20].

A projektet a Műszaki Irányítóbizottság a Közösségi Bizottsággal (Community Committee – CommComm) közösen vezeti. A Műszaki Irányítóbizottság a projekt magas szintű irányításáért, míg a Közösségi Bizottság a Node.js közösség irányításáért és bővítéséért felelős. A Közösségi Bizottság a Node.js Alapítvány egyik legfelsőbb szintű bizottsága. A Közösségi Bizottság hatáskörébe tartozik a kifelé mutató közösségi tájékoztatási erőfeszítések, beleértve a közösségi evangelizációt (promóciós anyagok készítése, közösség növelése stb.), az oktatási kezdeményezéseket, a Node.js Alapítvány kulturális irányultságát, a közösségi szervezet tájékoztatását, a fordítást és nemzetközivé tételt, a projekt moderálást/közvetítést, valamint a nyilvános tájékoztatást és publikációkat. A Közösségi Bizottsággal való együttműködésnek négy típusa van: Közreműködők (Contributor), Együttműködő (Collaborator), Megfigyelő (Observer) és Tag (Member). Közreműködő minden olyan magánszemély, aki létrehoz egy vitapontot, vagy

hozzászól egy vitaponthoz, valamint létrehoz egy pull kérést, vagy hozzászól egy pull kéréshez. Az Együttműködő olyan közreműködő, aki írási jogot kapott a tárházhoz. Megfigyelő lehet bármely olyan személy, aki kérelmet nyújtott be, vagy felkérést kapott a Közösségi Bizottság ülésén való részvételre. Ez egyben az első lépés a Taggá váláshoz. A Tag olyan szavazati joggal rendelkező Együttműködő, aki teljesítette a részvételi követelményeket és beszavazták a Közösségi Bizottság szavazási folyamata által. A Közösségi Bizottság küldetése a Node.js közösség további kiépítése [20].

A mag munkacsoportokat a Műszaki Irányítóbizottság hozza létre. Az Addon API Munkacsoport felelős a NAN projekt és az annak megfelelő nan csomag fenntartásáért az npm-en. A NAN projekt elérhetővé tesz egy absztrakciós réteget a Node.js natív add-on szerzői számára, segítve a kódírást, amely kompatibilis a Node.js, a V8 és a libuv sok aktívan használt verziójával. Az Addon API felelősségei közé tartoznak a következők: a NAN GitHub tárház fenntartása, beleértve a kódot, a kiadásokat és a dokumentációt; az addon-példák GitHub tárházának fenntartása, szintén beleértve a kódot, a kiadásokat és a dokumentációt; a C++ Addon API és Addon dokumentáció fenntartása a Node.js projektben, a Node.js TSC-nek való alárendeléssel; a nan csomag fenntartása az npm-ben, adott esetben az új verziók kiadása; üzenetek küldése a Node.js és a NAN interfész jövőjéről, hogy a közösséget előre értesítsék a változásokról [20].

A Benchmark Munkacsoport célja, hogy egyetértést érjen el az elfogadott teljesítmény-értékelések halmazával, amelyek felhasználhatók arra, hogy: nyomon kövessék és evangelizálják a Node.js kiadások közötti teljesítménynövekedést; elkerüljék a kiadások közötti teljesítmény-visszaesést. A felelősségek közé tartozik: egy vagy több teljesítmény-értékelés meghatározása, amelyek tükrözik az ügyfelek felhasználását; erőfeszítéseket tenni annak érdekében, hogy a közösség egyetértésre kerüljön a választott listán; a kiválasztott teljesítmény-értékelések rendszeres végrehajtásának hozzáadása a Node.js felépítéseihez (build); a teljesítmény nyomon követése/nyilvánosságra hozása a felépítések (build)/kiadások (release) között [20].

A Felépítési Munkacsoport célja egy elosztott automatizálási infrastruktúra létrehozása és fenntartása. Felelősségei közé tartozik: csomagok előállítása minden célplatformhoz; tesztek futtatása; teljesítménytesztek és összehasonlítások futtatása; építőtárolók (build-container) létrehozása és kezelése [20].

A Diagnosztikai Munkacsoport célja széleskörű, dokumentált és bővíthető diagnosztikai interfészek sorozatának csiszolása a Node.js és a JavaScript virtuális gépek általi használathoz. Felelősségei közé tartozik: együttműködés a V8-al a v8_inspector és a trace-event integrálásához

a Node.js-be; együttműködés a Core-ral az async_wrap és async_hooks finomításához; az operációs rendszer nyomkövető rendszerének integrációjának fenntartása és fejlesztése (például: ETW, LTTNG, dtrace); diagnosztikai képességek és API/k dokumentálása a Node.js-ben és annak komponenseiben; a lehetőségek és hiányosságok feltárása, a tulajdonság-kérelmek (feature request) megvitatása, és a konfliktusok kezelése a Node.js diagnosztikában; diagnosztikai eszközök ökoszisztémájának elősegítése a Node.js számára; interfészek/API-k meghatározása és hozzáadása annak érdekében, hogy szükség esetén lerakóhelyeket lehessen generálni; gyakori struktúrák meghatározása és hozzáadása a létrehozott lerakóhelyekhez annak érdekében, hogy támogassák azokat az eszközöket, amelyek elemezni akarják ezeket a lerakóhelyeket [20].

A Docker Munkacsoport célja hivatalos Docker képek (images) összeállítása, fenntartása és fejlesztése a Node.js projekt számára. Felelősségei közé tartozik: a hivatalos Docker képek frissen tartása összhangban az új Node.js kiadásokkal; a képek tökéletesítésének és/vagy javításának eldöntése és implementálása; a képek dokumentációjának karbantartása és tökéletesítése [20].

Az Evangelizációs Munkacsoport elősegíti a Node.js megvalósítását, és megmondja a közösségnek, hogy hogyan tudnak bekapcsolódni. Felelősségei közé tartoznak: a projekt-üzenetküldések megkönnyítése; a hivatalos projekt közösségi médiájának menedzselése; a találkozók és konferenciák előadóinak, valamint a közösségi rendezvények promóciójának kezelése; rendszeres frissítési összefoglalók és egyéb promóciós tartalmak közzététele [20].

Az i18n Munkcsoportok nem csak a fordításokat kezelik. Végpontok a közösség tagjai számára, hogy egymással együttműködjenek a választot nyelvükön. Minden csapat egy közösen beszélt nyelv köré szerveződött. Ezután minden egyes nyelvi közösség több lokalicáziót készíthet különféle projekterőforrásokhoz. A felelősségek közé tartozik: bármely Node.js anyag lefordítása, amely szerintük releváns a közösségük számára; a folyamatok felülvizsgálata a fordítások naprakészsége és jó minősége érdekében; a közösségi média kezelése és figyelemmel kisérése az anyanyelvükön; a Node.js előadok promoválása a saját nyelvükön tartott találkozókra és konferenciákra. Minden nyelvi közösség fenntartja a saját tagságát [20].

A Kiadási (Release) Munkacsoport a Node.js kiadási folyamatát. A felelősségei közé tartozik: a kiadási folyamat meghatározása; a kiadások tartalmának meghatározása; kiadások generálása, létrehozása és tesztelése; a hosszú távú támogatás (Long Term Support – LTS) és az aktuális ágak (Current branches), beleértve ezekre az ágakra vonatkozó változtatások visszafordítása; meghatározza az irányelvet ahhoz, hogy mi kerül visszafordításra a folyamatok kiadására [20].

A Biztonsági Munkacsoport a Node.js biztonságához kapcsolódó összes szempontot és folyamatot menedzseli. A felelősség körébe tartozik: a biztonsági irányelvek és eljárások meghatározása és fenntartása a mag Node.js Műszaki Irányítóbizottság által fenntartott egyéb projektekre; együttműködés a Node Biztonsági Platformmal (Node Security Platform), a közösség sebezhetőségével kapcsolatos adatok átadása az alapítványnak megosztott tulajdonként; gondoskodás a sebezhetőségi adatok hatékony és időben történő frissítséről; folyamatok felülvizsgálása és javaslása a biztonsági jelentések kezelésére; a Node.js külső, nyílt forráskódú ökoszisztémán belüli biztonsági aggályok meghatározása és fenntartása; a nyilvánosságra hozott biztonsági résekkel kapcsolatos adatok karbantartása és rendelkezésre bocsátása a mag Node.js projekt, a Node.js Alapítvány műszaki csoportja által fenntartott egyéb projektek és a külső Node.js nyílt forráskódú ökoszisztéma számára; a Node,js ökoszisztéma biztonsági gyakorlatainak fejlesztésének elősegítése; biztonsági fejlesztések javaslása a mag Node.js projekthez; az egészséges biztonsági szolgáltatások és a termékszolgáltató ökoszisztéma kibővítésének megkönyítése és támogatása [20].

Az Adatfolyamok Munkacsoport (Streams Working Group) a Node.js és az npm ökoszisztéma által használt Streams API támogatására és fejlesztésére szolgál. Egy kompozitív API létrehozására törekszenek. Az API fejlesztéseit az ökoszisztéma igényei fogják vezérelni. Az együttműködési képesség és a visszafele kompatibilitás más megoldásokkal és korábbi verziókkal alapvető fontosságú. Felelősségei közé tartozik: adatfolyam dokumentációk szerkesztése a Node.js projekten belül; az adatfolyam alosztályok változásainak áttekintése a Node.js projekten belül; a változások átirányítása a Adatfolyamokhoz a Node.js projekttől ehhez a projekthez; az adatfolyam ellátók implementálásának segítése a Node.js-en belül; olvasható adatfolyamok verzióinak ajánlása, hogy be legyenek véve a Node.js-be; üzenetküldés az adatfolyamok jövőjéről, hogy közösséget előre értesítsék a változásokról [20].

A JavaScript szintén közismert nyelv, amely a Node.js-t hozzáférhetővé tette a webfejlesztő közösség számára. A nyílt forráskódot támogató közösség webes keretrendszereket fejlesztett ki az alkalmazások fejlesztésének felgyorsítására, mint például az Express.js, a Socket.io, a Connect, a Feathers.js, a Koa.js, a Hapi.js, a Sails.js, a Meteor, a Derby és még sokan mások [18].

A Node.js fejlesztői közössége leginkább két levelezőlistán található meg, az egyik a Node.js (https://groups.google.com/forum/#!forum/Node.js), a másik pedig a Node.js-dev (https://groups.google.com/forum/#!forum/Node.js-dev), továbbá a #node.js IRC (Internet Relay Chat) csatornán, mely a freenode-on található [17]. Számos fejlesztői konferencia és esemény

támogatja a Node.js közösséget, ide tartozik a NodeConf, a Node Interactive és a Node Summit, valamint számos regionális esemény [18]. Magyarországon a meetup.com keretein belül szerveződik a közösség, mely havi rendszerességgel tartja találkozóit, Nodebp néven [17].

A nagyobb (major) Node.js verziók hat hónapra lépnek a Jelenlegi (Current) kiadási állapotba, amely időt ad a könyvtár szerzőinek, hogy támogatást nyújtsanak hozzá. Hat hónap után a páratlan számú kiadások nemtámogatottá válnak, és a páros számú kiadások pedig aktív hosszú távon támogatott (cktive Long Term Support – Active LTS) állapotba kerülnek és készek az általános használatra. A hosszú távú támogatás általában garantálja, hogy a kritikus hibákat összesen 30 hónapig kijavítják. A termelési alkalmazásoknak csak az aktív hosszú távon támogatott vagy a karbantartási hosszú távon támogatott (Maintenance LTS) kiadásokat kellene használniuk [20].

Az Express.js vagy egyszerűen Express egy webalkalmazási keretrendszer a Node.js számára, amelyet ingyenes és nyílt forráskódú szoftverként adtak ki a MIT Licenc alatt. Webalkalmazások és API-k készítésére tervezték. Úgy nevezték, mint a tényleges (de facto) standard szerverkeretrendszer a Node.js számára [21]. Az Express.js filozófiája az, hogy kicsi, robusztus eszközöket biztosítson a HTTP szerverek számára, ezáltal kiváló megoldást jelent egyoldalas alkalmazásokhoz, weboldalakhoz, hibridekhez vagy nyilvános http API-khoz. Az Express.js nem kényszerít semmilyen specifikus ORM (Object-Relational Mapping - objektum-relációs leképzés) vagy sablonmotor használatára. Több mint 14 sablonmotor támogatásával és a Consolidate.js-el gyorsan létrehozható egy keretrendszer [23].

Az eredeti szerző, TJ Holowaychuk, Sinatra ihlette szerverként írta le, ami azt jelenti, hogy viszonylag minimális, sok plug-in-ként elérhető tulajdonsággal (feature). Az Express.js tulajdonságai: robusztus útvonalkeresés; összpontosítás a nagy teljesítményre; szuper magas tesztlefedettség; HTTP segítők (átirányítás, gyorsítótárazás stb.); látja a rendszertámogató 14+ sablonmotorokat; végrehajtható az alkalmazások gyors generálásához. Az Express.js a MEAN stack back-end komponense, a MongoDB adatbázis szoftverrel és az AngularJS front-end keretrendszerrel együtt. Az Express.js-t az Accenture, a DentiSphere, az Exove, a Fox Sports, az Incicle Technologies, a MuleSoft, a Myntra, a NodeBB, a QuizUp, a Ripjar, a RisingStack, a SparkPost, a PayPal, az Uber, a Yandex, az Agrippa Solutions, a CircleHD, a Teachoo, a Taskade, a Hasura, az iLoveCoding, a Kuali és az IBM is használja [21, 22].

Az Express.js-t TJ Holowaychuk alapította, a jelenlegi vezető karbantartó Douglas Chritopher Wilson. Az első kiadás az Express.js GitHub tárházának megfelelően, 2010. május 22-én volt. 2014 júniusában a StrongLoop vállalat megszerezte a projekt menedzselésének jogát. 2015

szeptemberében az IBM vállalat megszerezte StrongLoop-ot. 2016 januárjában az IBM bejelentette, hogy az Express.js-t a Node.js Alapítvány inkubátorának intézősége alá fogja helyezni. [21].

Az Express.js műszaki bizottsága kéthetente online ülésezik (szükség szerint), hogy megvitassák az Express.js fejlesztését és fenntartását, és az Express.js projekttel kapcsolatos egyéb kérdéseket. Az egyes üléseket általában egy expressjs/vita kiadványban hirdetik meg, egy hivatkozással a csatlakozáshoz vagy az ülés megtekintéséhez, amely minden megfigyelő számára nyitva áll. Az üléseket rögzítik, amelyek megtalálhatók az Express.js YouTube csatornáján (https://www.youtube.com/channel/UCYjxjAeH6TRik9Iwy5nXw7g) [22].

Az Express.js közössége bővítmények széles választékát hozta létre, köztes szoftver (middleware) modulokat és magasabb szintű keretrendszereket. Továbbá, az Express.js közössége modulokat tart fenn a következő két GitHub szervezetben (orgs): [22].

- jshtttp (https://jshttp.github.io) – modulok, amelyek hasznos segédfüggvényeket biztosítanak;
- pillarjs (https://pillarjs.github.io) – alacsony szintű modulok, amelyeket az Express.js belsőleg használ.
- Számos népszerű keretrendszer épült az Express.js-re: [22].
- Feathers: Segítségével prototípusok és készíthetők percek alatt, és termelésre kész valós idejű alkalmazások napok alatt;
- ItemsAPI: back-end-ek kereshetők vele web- és mobilalkalmazásokhoz az Express.js-re és az Elasicsearch-re építve;
- KeystoneJS: Weboldal és API alkalmazás-keretrendszer/CMS automatikusan generált React.js adminisztrátori felhasználui felülettel (Admin UI);
- Poet: Könnyű Markdown Blog-motor azonnali lapszámozással, címke- és kategórianézetekkel;
- Kraken: Biztonságos és méretezhető réteg, amely bővíti az Express.js-t struktúra és konvenció biztosításával;
- LoopBack: Nagyon bővíthető, nyílt forráskódú Node.js keretrendszer végponttól végpontig (end-to-end) REST API-k gyors létrehozásához;
- MEAN: Fullstack JavaScript kertrendszer, amely egyszerűsíti és felgyorsítja a webalkalmazások fejlesztését;

- Sails: MVC keretrendszer a Node.js számára praktikus, gyártásra kész alkalmazások készítéséhez;
- Hydra-Express: egy könnyűsúlyú (light-weight) könyvtár, amely megkönnyíti a Node.js Microservices létrehozását az Express.js használatával;
- Blueprint: egy SOLID keretrendszer API-k és backend szolgáltatások létrehozásához;
- Locomotive: Erőteljes MVC webkeretrendszer a Node.js számára a Passport.js készítőjétől;
- graphql-yoga: Teljes funkcionalitású, mégis egyszerű és könnyűsúlyú GraphQL szerver;
- Express Gateway: Teljes funkcionalitású és bővíthető API Gateway, amely az Express.js-t használja alapként;
- Dinoloop: Rest API alkalmazáskeretrendszer, a TypeScript által működtetve, függőségi befecskendezéssel (dependency injection);
- Kites: Sablonalapú webalkalmazás keretrendszer;
- FoalTS: Új generációs keretrendszer vállalati szintű Node.js alkalmazások készítéséhez (TypeScript);
- NestJs: Egy progresszív Node.js keretrendszer hatékony, skálázható és vállalati szintű szerveroldali alkalmazások készítéséhez a TypeScript és a JavaScript tetején (ES6, ES7, ES8);
- Expressive Tea: Egy kis keretrendszer modulálható, tiszta, gyors és leíró szerveroldali alkalmazásokhoz a TypeScript és az Express.js használatával.
- Néhány nyílt forráskódú projekt, amelyek az Express.js-t használják:
- Builder Book: Nyílt forráskódú webalkalmazás dokumentumok, vagy könyvek közzétételéhez. A React, a Material-UI, a Next, az Express.js, a Mongoose és MongoDB használatával készült;
- SaaS Boilerplate: Nyílt forráskódú webalkalmazás saját Saas termék készítéséhez. A React, a Material-UI, a Next, a Mobx, az Express.js, a Mongoose, a MongoDB és a TypeScript használatával készült;
- BitMidi: Nyílt forráskódú webalkalmazás, amelyet az Express.js működtet. A BitMidi a web korai korszakából származó MIDI (Musical Instrument Digital Interface) fájlok történelmi archívuma. A legújabb modern webtechnológiát használja, beleértve a WebAssembly-t és Web Audio-t, hogy életre keltse a MIDI-t.

Az Express.js és az expressjs szervezet többi projektje a GitHub-on a Node.js Alapítvány projektjei. Ezeket a projekteket a Node.js Alapítvány általános politikái és irányelvei szerint vezetik, valamint a következő kiegészítő iránymutatások vonatkoznak rájuk: Technikai bizottság (Technical committee), Közösségi közreműködő útmutató (Community contributing guide), Együttműködési útmutató (Collaborator's guide) és Biztonsági eljárások és procedúrák (Security policies and procedures) [22].

2.6.2. jQuery

A jQuery egy ingyenes, nyílt forráskódú, népszerű, gyors, kicsi és tulajdonságokban gazdag JavaScript könyvtár, amely a HTML kód és a kliensoldali JavaScript közötti kapcsolatot hangsúlyozza, és célja a HTML kliens oldali szkriptjének egyszerűsítése. John Resig hozta létre 2006-ban. A függvénykönyvtár MIT és GNU kettős licenc alatt jelent meg. [8, 9, 11].

Célja, hogy segítsen minél inkább leválasztani a JavaScript kódot a HTML-ről, és kényelmes kommunikációt biztosítson a weblap elemeivel – eseményvezérlők és azonosítók (ún. CSS szelektorok) használatával [9].

Sokkal egyszerűbbé teszi a HTML dokumentumok bejárását és manipulálását, eseménykezelést, animációt, és az Ajax sokkal egyszerűbb egy könnyen használható API-val, amely sok böngészővel működik. A sokoldalúság és a kibővíthetőség kombinációjával a jQuery megváltoztatta azt a módot, ahogyan emberek milliói írják a JavaScriptet [8]. A jQuery mottója: "Írj kevesebbet, tégy többet". A jQuery alkotója kifejezetten arra hozta létre ezt a könyvtárat, hogy a gyakori feladatok triviálissá váljanak és könnyen megtanulhatók legyenek, és megoldják a böngésző összeférhetetlenségéből fakadó problémákat [11].

A jQuery lehetővé teszi a fejlesztők számára plug-in-ok létrehozását a JavaScript könyvtár tetején. Ez lehetővé teszi a fejlesztők számára, hogy absztrakciókat hozzanak létre alacsony szintű interakciókhoz és animációkhoz. A jQuery könyvtár moduláris megközelítése lehetővé teszi nagy teljesítményű dinamikus weboldalak és webes alkalmazások létrehozását [10].

A jQuery alapvető tulajdonságainak halmaza – DOM (Document Object Model) elemek kiválasztása, bejárása és manipulációja – a szelektormotorja által engedélyezve ("Sizzle" néven az 1.3-mas verzióból) egy új "programozási stílust" hozott létre, egybeolvasztva algoritmusokat és DOM adatstruktúrákat. Ez a stílus befolyásolták más JavaScript keretrendszerek (framework), mint például a YUI v3 és a Dojo architektúráját, később ösztönözve a standard Selectros API létrehozását. Később ezt a stílust növelték egy mélyebb algoritmus-adat egybeolvadással a jQuery örökösében, a D3.js keretrendszerben [10].

Egy internetes elemzés szerint ez a legszélesebb körben alkalmazott JavaScript könyvtár. 3-4-szer nagyobb felhasználással rendelkezik, mint bármely más JavaScript könyvtár [10]. Számos ismert IT cég és weboldal is alkalmazza a jQuery-t saját projektjeiben, mint például az Amazon, Dell, Etsy, Netflix, Best Buy, Instagram, Fox News, GoDaddy, és még sokan mások [11]. A Microsoft erre építette a Visual Studio-ban is elérhető ASP.NET AJAX platformját [9].

A jQuery-vel való fejlesztés alapelvei a következők:

- A JavaScript és HTML elválasztása: A jQuery könyvtár (library) egyszerű szintaxist biztosít eseménykezelők hozzáadására a DOM-hoz JavaScript használatával, ahelyett, hogy HTML eseményattribútumokat adna hozzá a JavaScript függvények meghívásához. Így arra ösztönzi a fejlesztőket, hogy teljesen különítsék el a JavaScript kódot a HTML jelölésektől [10].
- Rövidség és egyértelműség: a jQuery elősegíti a rövidséget és az egyértelműséget olyan tulajdonságokkal (feature), mint a "láncolható" ("chainable") függvények és a és a rövidített függvénynevek [10].
- A böngészők közötti inkompatibilitások kiküszöbölése: A különböző böngészők JavaScript motorjai (engine) kissé eltérnek, így az egyik böngészőben működő kód nem biztos, hogy működik egy másikban is. Más JavaScript eszközkészlethez (toolkit) hasonlóan a jQuery kezeli az összes ilyen böngészőközti inkonzisztenciát, és biztosít egy konzisztens interfészt, amely a különböző böngészők közt működik [10].
- Bővíthetőség: Az új események, elemek, és metódusok könnyen hozzáadhatók, és plug-inként újra-felhasználhatók [10].

A jQuery magába foglalja a következő tulajdonságokat (feature) [10]:

- DOM elemek kiválasztása a Sizzle több böngészős, nyílt forráskódú szelektormotorjával, a jQuery melléktermékével
- A DOM manipuláció CSS szelektorokon alapul, amely elemek neveit és attribútumait használja – mint például az azonosító (id) és az osztály (class) – kritériumkén csomópontok kiválasztására a DOM-ban
- Események (Event)
- Hatások (effect) és animációk
- Ajax
- Halasztott (Deferred) és ígéret (Promise) objektumok az aszinkron feldolgozás vezérlésére
- JSON elemzés

- Bővíthetőség plug-in-okon keresztül
- Segédprogramok, mint a tulajdonság észlelése
- Kompatibilitási metódusok, amelyek alapból elérhetőek a modern böngészőkben, de visszaesésre van szükség régebbi böngészők számára
- Böngészők közti támogatás

A jQuery 3.0 és újabb verziók támogatják a Firefox (és ESR), Chrome, Safari és az Edge, valamint az Internet Explorer 9, jelenlegi stabil verzióját, és az azt megelőző verziót. Mobiltelefonon támogatja az iOS 7 és újabb, valamint az Android 4.0 és újabb verziókat [10]

A jQuery könyvtárat általában egyetlen JavaScript fájlként terjesztik, amely meghatározza az összes interfészét, beleértve a DOM-ot, az eseményeket, és az Ajax függvényeket. Beilleszthető egy weboldalba egy helyi másolatra mutató hivatkozással, vagy a nyilvános szerverekről elérhető sok másolat egyikére mutató hivatkozással. A jQuery rendelkezik a MaxCDN által üzemeltetett tartalomszolgáltató hálózattal (content delivery network - CDN) [10]

A jQuery két féle függvényt kínál: a statikus segédfüggvényeket és a jQuery objektum metódusokat. Mindegyiknek megvan a saját használati stílusa. Mindkettő a jQuery fő azonosítóján keresztül érhető el: a jQuery-n keresztül. Ennek az azonosítónak van egy másik neve, a $. Az összes függvény elérhető a két név bármelyikén keresztül [10].

A jQuery függvény egy vagy több DOM csomópontot (node) képviselő jQuery objektum létrehozására szolgáló gyárfüggvény (factory). A jQuery objektumoknak vannak metódusaik ezen csomópontok manipulálására. Ezek a metódusok (néha parancsoknak nevezik) láncolhatók, mivel minden metódus jQuery objektumot térít vissza. Több DOM csomópont elérése és manipulálása a jQuery-ben általában a $ függvény CSS szelektor-karakterlánc meghívásával kezdődik. Ez egy jQuery objektumot térít vissza, amely HTML oldal összes megfelelő elemére vonatkozik. A $(“div.test”) például egy jQuery objektumot térít vissza a test osztály összes div elemmel. Ez a csomóponthalmaz manipulálható a visszatérített jQuery objektumon való metódushívásokkal [10].

A statikus segédprogramok közvetlenül nem hatnak a jQuery objektumra. Statikus metódusokként érhetők el a jQuery vagy a $ azonosítón keresztül. Például a $.ajax() egy statikus metódus [10].

A jQuery biztosít egy $.noConflict() függvény, amely abbahagyja a $ név irányítását. Ez akkor hasznos, ha jQuery-t egy weboldalon használják, amelyhez szintén hozzá van kapcsolva egy másik könyvtárral, amely szintén égényeli a $ szimbólumot, mint azonosítóját. Konfliktis nélküli módban a fejlesztők használhatják a jQuery-t, mint a $ helyettesítőjét a funkcionalitás elvesztése nélkül [10].

A jQuery-t általában az inicializálási kód és az eseménykezelő függvények $(handler)-be való helyezésével használják. Ezt a jQuery váltja ki, amikor a böngésző befejezte a DOM felépítését az aktuális weboldalhoz. Történelmileg, a $(document).ready(callback) tényleges kifejezésmód a kód futtatásához, miután a DOM készen áll. Azonban a jQuery 3.0-óta a fejlesztőket arra ösztönzik, hogy a sokkal rövidebb $(handler) írásmódot használják [10].

A jQuery objektum metódusok általában szintén jQuery objektumot térítenek vissza, amely lehetővé teszi a metódusláncok (method chain) használatát [10].

A jQuery-n keresztül a meglévő DOM csomópontokhoz való hozzáférés mellett új DOM csomópontok is létrehozhatók, ha a $() gyárfüggvénynek argumentumként átadott karakterlánc HTML-ként néz ki [10].

A $.ajax()-al Ajax kéréseket hozhatunk létre (böngészőközi támogatással) távoli adatok betöltésére és manipulálására [10].

A jQuery architektúrája lehetővé teszi a fejlesztők számára, hogy plug-in kódot hozzanak létre, hgoy kibővítsék annak funkcióját. Több ezer jQuery plug-in érhető el az interneten, amelyek számos funkciót lefednek, mint például Ajax segítőket, webszolgáltatásokat (Web service), adatrácsokat (datagrid), dinamikus listákat, XML és XSLT eszközöket, “drag and drop”-ot, eseményeket, süti (cookie) kezelést és modális (modal) ablakokat. A jQuery plug-in-ok egy fontos forrása a jQuery Projekt plug-in aldomainje. Azonban ennk az aldomainnek a plug-in-jait kitörölték 2011 decemberében, hogy megszabadítsák a webhelyt a spam-től. Az új webhely a GitHub által üzemeltetett tárház (repository), amely a megkérte a fejlesztőket, hogy küldjék be újra a plug-in-jaikat és alkalmazkodjanak az új követelésekhez. A jQuery biztosít egy “Oktatóközpontot” (“Learning Center”), amely segíthet a JavaScript megértésében és a jQuery plug-in-ok fejlesztésének elkezdésében [10].

A QUnit egy teszt-automatizálási keretrendszer, amelyet a jQuery projekt tesztelésére használnak. A jQuery csapata egy házon belüli (in-house) egység tesztelési könyvtárként fejlesztette ki. A jQuery csapata a kód és a plug-in-ok tesztelésére használja, de tesztelni tud bármely generikus JavaScript kódot, beleértve a szerveroldali (server-side) JavaScript kódot is. 2011-től a jQuery Tesztelőcsapat a QUnit-ot TestSwarm-mal arra használja, hogy teszteljen minden egyes jQuery kódbázisú kiadást [10].

A jQuery Mobile egy érintésoptimalizált (touch-optimalized) webkeretrendszer (úgy is ismert, mint mobilkeretrendszer), pontosabban egy JavaScript könyvtár, amelyet a jQuery projekt csapata fejlesztett ki. A fejlesztés az okostelefonok és táblagépek széles választékával kompatibilis

keretrendszer létrehozására összpontosít, amelyet az okostelefonok és táblagépek növekvő és heterogén piaca tesz szükségessé. A jQuery Mobile keretrendszer kompatibilis más mobilalkalmazás-keretrendszerekkel és olyan platformokkal, mint például a PhoneGap és a Worklight [12].

A jQueri UI, amely először 2007-ben jelent meg, egy vizuális effektekkel animált GUI widget-ek és jQuery-vel (JavaScript könyvtár), sorba kapcsolt stíluslapokkal (Cascading Style Sheet) és HTML-lel implementált témák gyűjteménye. A JavaScript elemző szolgáltatás, a Libscore, szerint az egymillió legnépszerűbb weboldal közül 197.000-en használják a jQuery UI-t, így a második legnépszerűbb JavaScript könyvtárrá téve azt. Jelentős felhasználói a Pinterest, a PayPal, az IMDb, a Huffington Post és a Netflix. A jQuery és a jQuery UI egyaránt ingyenes és nyílt forráskódú szoftver, amelyet a jQuery Foundation terjeszt a MIT Licenc alatt [13].

2.6.3. MongoDB

A MongoDB egy NoSQL (Not Ony SQL). valós idejű (Real Time) adatbázis. Az adatok tárolására JSON formátumot használ. Előnye, hogy egy adott eltárolt dokumentum nem szükségszerű, hogy egy megadott sémát kövessen.

2.6.4. Handlebars

A Handlebars Chris Wanstrarh által létrehozott Mustache sablon nyelv kiterjesztése. A Handlebars és a Mustache egyaránt logika nélküli sablonnyelvek, amelyek elválasztják egymástól a nézetet (view) és a kódot [4]. A Handlebars.js-t a MIT Licenc alatt adták ki [7].

A Handlebars sablon úgy néz ki, mint a hagyományos HTML beágyazott Handlebars kifejezésekkel. A Handlebars kifejezés egy {{ és valamely tartalom, amelyet }} követ [5]. Egy sablont és egy bemeneti objektumot használ HTML vagy más szöveges formátum generálásához [6].

A Handlebars nagyban kompatibilis a Mustache sablonokkal. A legtöbb esetben ki lehet cserélni a Mustache-t a Handlebars-al, és folytatni lehet a jelenlegi sablon használatát [5].

A Handlebars az ún. segítőkben (Helpers) és beágyazott útvonalakban (Nested Paths) különbözik a Mustache-tól. A segítők (Heplers) a blokk-kifejezésekben (amelyeket szakaszoknak hívnak a Mustache-ban) engedélyezhetnek egyedi funkcionalitásokat az adott blokkhoz tartozó, felhasználó által írt, kód segítségével. Amíg a Mustache-ban a specifikáció meghatározza a kevésbé testre szabható szakaszok funkcionalitását. A Mustache csak egyszerű útvonalakat engedélyez, míg a

Handlebars támogatja az objektumok nagyobb beágyazódását [4]. Van néhány Mustache viselkedés, amelyet a Handlebars nem implementál:

- A Handlebars némileg eltér a Mustache-tól, mivel alapértelmezetten nem végeznek rekurzív keresést. A fordítási idő **compant** jelző (flag) be kell legyen állítva ennek a funkciónak a engedélyezéséhez. A felhasználóknak figyelembe kell venniük azt, hogy ennek a jelzőnek a bekapcsolása teljesítény költséggel jár. A pontos költség sablonként változik, de ajánlott, hogy a teljesítményfüggő műveletek kerüljék ez a módot, és inkább explicit útvonal-referenciákat használjanak [7].
- Az opcionális Mustache stílusú lambdák nem támogatottak. Ehelyett a Handlebars biztosítja a saját lambda-felbontását, amely követi a segítők viselkedését [7].
- Az alternatív elválasztók (delimiter) nem támogatottak [7].

A Handlebars néhány további funkcióval egészíti ki a sablonok írásának megkönnyítését, és megváltoztatja a partíciók működésének apró részleteit [6] Különféle beépített segítőket kínál, pédául az if feltételt és az each iterátort, valamint számos API-t az alkalmazásokhoz és segítőkhöz (Helpers) [5].

A Handlebars-t úgy tervezték, hogy bármilyen ECMAScript 3 környezetben működhessen. Ez magéba foglalja a következőket: Node.js, Chrome, Firefox, Safari 5+, Opera 11+, IE 6+. A régebbi verziók valószínűleg működni fognak, de ezeket formálisan még nem tesztelték [7].

Egy durva teljesítményteszt során az előre kompilált Handlebars.js sablonok (a Handlebars.js eredeti változatában) körülbelül fele annyi idő alatt készültek, mint a Mustache sablonok. Az újraírt Handlebars (jelenlegi verzió) gyorsabb, mint a régi verzió, és 5-7-szer gyorsabb, mint a Mustache megfelelője [7].

2.6.5. Bootstrap

A Bootstrap egy ingyenes és nyílt forráskódú (open-source) frontend web keretrendszer (framework), amelyet reszponzív, főként mobil frontend webfejlesztésre szántak [14]. A Bootstrap segítségével tervezett webhelyek és alkalmazások kompatibilisek mind az iOS, mind az Android rendszerekkel [16] CSS- és (opcionálisan) JavaScript-alapú tervezősablonokat tartalmaz tipográfia, űrlapok (form), gombok, navigációs és egyéb interfész-összetevők számára. A Bootstrap a harmadik legcsillagozottabb projekt a GitHub-on, több mint 131.000 csillaggal [14]. A 2010-es évek közepén jött létre, és a készítők arra helyzeték a hangsúlyt, hogy nyílt forráskódú legyen [15].

A Boostrap-et, eredeti nevén Twitter Blueprint-et, Mark Otto és Jacob Thornton fejlesztette ki a Twitteren, mint olyan keretrendszert, amely konzisztenciára ösztönöz a belső eszközökön át. Jelenleg a Bootstrap-et egy kis fejlesztői csapat tartja karban a GitHub-ot használva [15]. A Bootstrap előtt különféle könyvtárakat használtak felületfejlesztésre, ami inkonzisztenciát és magas karbantartási terhet okozott [14].

A Bootstrap olyan webes keretrendszer, amely az informatív weboldalak fejlesztésének egyszerűsítésére (ellentétben a webalkalmazásokkal) összpontosít. A webes projekthez történő hozzáadás elsődleges célja az, hogy a Bootstrap szín-, méret-, betűtípus- és elrendezési lehetőségeit alkalmazza a projektre. Adott projekthez hozzáadva a Bootsrap alapvető stílusdefiníciókat biztosít az összes HTML elem számára. Végeredménye egy egységes megjelenés a szövegeknek, táblázatoknak és űralapelemeknek a böngészőkön keresztül. Ezenkívül a fejlesztők kihasználhatják a Bootstrap-ben definiált CSS osztályokat, hogy tovább testre szabhassák a tartalmaik megjelenését. Például a Bootstrap rendelkezett világos és sötét színű táblázatokkal, oldalfejlécekkel, szembetűnőbb idézet- és szövegkiemelésekkel [14].

A Bootstrap számos JavaScript komponenssel együtt jQuery plug-in-ok formájában érkezik. További felhasználói felületi elemeket biztosít, úgy, mint párbeszédablakok (dialog box), eszköztippeket és carousel-ek [14]. A Bootstrap különféle tervezősablonokat tartalmaz, amelyek a következőkön alapszanak: HTML, CSS, Less (3-mas verzió), Sass (4-es verzió) és opciós JavaScript bővítmények (extension) [16]. Bővítik néhány meglévő interfész elem funkcionalitását, beleértve például a bemeneti mezők automatikus kitöltését [14].

A Bootstrap legjelentősebb alkotóelemei az elrendezéséi komponensek, mivel egy teljes weboldalra vonatkoznak. Az alapvető elrendezési összetevő “Konténer”-nek (Container) hívják, mivel az oldal minden más eleme benne van elhelyezve. A fejlesztők választhatnak egy rögzített szélességű és egy folyadék szélességű konténer között. Míg az utóbbi mindig, kitölti a weboldal szélességét, addig az előbbi a négy előre meghatározott rögzített szélesség egyikét használja, a képernyő szélességétől függően: kisebb mint 576 pixel; 576-768 pixel között; 768-992 pixel között; 992-1200 pixel között; nagyobb, mint 1200 pixel. Miután a konténer a helyén van, a többi Bootstrap elrendezési komponens CSS rácsos elrendezést (grid layout) valósít meg a sorok és oszlopok meghatározásával [14]. Annak ellenére, hogy a Bootstrap reszponzív tizenkét oszlopos rácsokban, elrendezésekben és komponensekben van tervezve, a testreszabás is nagyon egyszerű. Függetlenül attól, hogy rögzített vagy reszponzív rácsra van szükség, néhány változtatással

megvalósítható. Az oszlopok eltolását és egymásba ágyazását szintén könnyű megvalósítani a CPU-alapú és a mobilalapú böngészőrácsokban (browser grid) is [16].

A Bootstrap egyik kiemelkedő tulajdonsága a reszponzív segédosztályai (utility class). A reszponzív segédosztályok használatával készíthető úgy egy adott tartalom, hogy megjelenése vagy elrejtése csak a használt képernyő méretétől függjön. Ez a tulajdonság rendkívül hasznos azoknak a tervezőknek, akik a weboldalok mobil és táblagépbarát verzióját szeretnék elkészíteni [16].

Néhány komponens, amely előre stilizált a Bootstrap-ben: legördülő listák (drop-down), gombok (buttons), navigációs menü (navigation), kitűzők figyelmeztetései (badges alert), haladásmutató (progress bar). A legördülő lista komponens egy weboldal reszponzív additív tulajdonsága. Egy weboldalba való beillesztéshez sok különféle plug-in-t teszteltek, melyek főként Jáva-alapúak. De Bootstrap-el és annak könnyek testre szabható lehetőségeivel ez percek alatt könnyen megtehető. A könnyen elérhető sablonok megkönnyítik a tapasztalatlan felhasználók számára egy weboldal létrehozását a Bootsrap weboldalán elérhető egyszerű oktatóanyag vagy demó alapján [16].

A Bootstrap előrekompilált verziója elérhető egy CSS fájl és három JavaScript fájl formájában, amelyek bármely projekthez könnyedén hozzáadhatók. A Bootstrap nyers formája azonban lehetővé teszi a fejlesztőknek, hogy további testreszabásokat és méretoptimalizálásokat implementáljanak. Ez a nyers forma moduláris, azaz a fejlesztő eltávolíthatja a felesleges összetevőket, alkalmazhat egy témát, és módosíthatja a nem kompilált Sass fájlokat [14].

3. A rendszer specifikációi és architektúrája

A dolgozat célja egy CRS webalkalmazás készítése, amelyet tanár és diákok akár online, akár hagyományos „face-to-face“ oktatás esetén, használhatnának a tanórák érdekesebbé tételére, az interaktivitás növelésére, és a jelenlét egyszerűbb követésére. Az alkalmazás tervezésénél figyelembe vettem, hogy a rendszert három féle felhasználó fogja használni: rendszergazda, tanár és diák.

3.1. Követelmény specifikáció

Ebben a fejezetben bemutatom a saját CRS alkalmazás specifikációját. A felhasználóknak szükségük van internetkapcsolatra és egy böngészőre. A webalkalmazás platformfüggetlen, bármely internetre csatlakoztatható és böngészőt futtatni képes eszközről használható. Az alkalmazást böngészőn keresztül érhetik el a felhasználók.

Rendszergazda típusú felhasználónak egy eszközre van szüksége, tanár típusú felhasználónak egy eszközre, de két képernyőre (az egyik amit csak ő lát, a másik amit a diákokkal is megoszt). Diák típusú felhasználónak online oktatás esetén szüksége van két eszközre, vagy egy számítógépre két képernyővel, hagyományos „face-to-face“ oktatás esetén egy eszköz is elegendő számára. A diák online oktatás esetén egy két képernyős számítógép használatakor az egyik képernyőn látná azt, amit megoszt vele a tanár, a másik képernyőt használná clicker eszközként. Szintén online oktatás esetén a diák használhat egy egyképernyős számítógépet, amelyen az jelenne meg, amit megosztott vele a tanár, clicker eszközként pedig használhatja a saját okostelefonját, tabletjét vagy egy másik, a közvetlen közelében levő számítógépet.
Rendszergazda esetében, a funkcionalitásai miatt, teljesen mindegy, hogy telefonról, tabletről vagy számítógépről használja. A tanár funkcionalitásai miatt tanóra közben ajánlott számítógépről dolgoznia, és két képernyő (vagy egy képernyő és egy vetítő) használata elengedhetetlen. Tanórán kívül a tanár is bármely eszközt használhat. Diákok esetében mindegy, hogy milyen eszközt használnak. Számukra hagyományos „face-to-face“ oktatás esetén elegendő egy eszköz (legkényelmesebb ebben az esetben egy okostelefon használata), online oktatás esetén pedig szükség van egy számítógépre két képrenyővel, vagy két bármilyen, az fenneb leírt követelményeknek megfelelő, külön állá eszközre (legkényelmesebb ebben az esetben egy laptop és egy okostelefon használata).
Az alkalmazás költséghatékony, mert nem szükséges speciális clicker eszközök megvásárlása, hanem a diákok a saját okos eszközüket vagy számítógépüket használhatják. Ennek a megoldásnak egy másik előnye, hogy a diákoknak nem kell elsajátítaniuk egy clicker eszköz használatát.
A rendszer módosítható, illetve testre szabható újabb igények megjelenése esetén.

3.1.1. Funkcionális követelmények

Ebben a részben a rendszer működését és a rendszer által biztosított funkcionalitásokat írom le. A rendszer funkcionalitás követelményei három fő csoportra oszthatók a rendszer felhasználói szerint:

- rendszergazda típusú,
- tanár típusú
- és diák típusú felhasználó funkcionalitásai.

A funkcionalitások listája:

1. Rendszergazda, tanár és diák közös funkcionalitásai:

1.1.Bejelentkezés

1.2.Kijelentkezés

2. Rendszergazda és tanár közös funkcionalitásai:

2.1.Kulcs létrehozása (regisztrációs vagy csatlakozási kulcsok)

2.2.Kulcsok listázása

2.3.Kulcs szerkesztése

2.4.Kulcs törlése

3. Tanár és diák közös funkcionalitása:

3.1.Regisztráció

4. Tanár funkcionalitásai:

4.1.Tantárgy létrehozása

4.2.Tantárgyak listázása

4.3.Tantárgy szerkesztése

4.4.Tantárgy törlése

4.5.Tanóra létrehozása

4.6.Tanórák listázása

4.7.Tanóra szerkesztése

4.8.Tanóra törlése

4.9.Kérdéssor létrehozása

4.10. Kérdéssorok listázása

4.11. Kérdéssor szerkesztése

4.12. Kérdéssor törlése

4.13. Kérdés és hozzá tartozó válaszok létrehozása

4.14. Kérdések és a hozzájuk tartozó válaszok listázása

4.15. Kérdés és hozzá tartozó válaszok szerkesztése

4.16. Kérdés és hozzá tartozó válaszok törlése

4.17. Tanóra előkészítése csatlakozásra

4.18. Tanóra elindítása

4.19. Kérdéssor elindítása

4.20. Kérdés elindítása (automatikus)

4.21. Kérdéssor megszakítása

4.22. Kérdéssor befejezése (automatikus)

4.23. Osztályzatok létrehozása befejezet kérdéssorhoz (automatikus)

4.24. Jelenlét készítése

4.25. Tanóra befejezése

4.26. Leadott válaszok listázása

4.27. Jelenlétek listázása

5. Diák funkcionalitásai:

5.1.Csatlakozás tanórához

5.2.Kérdés megválaszolása

Az alábbiakban néhány funkcionalitást részletesen bemutatok a dolgozatban. Terjedelem miatt a többi nem került ide (de megtalálható egy saját munka dokumentumban).

1. Tanóra létrehozása (4.5.) funkcionalitás: Tanár típusú felhasználó bejelentkezése után megjelenik a menü, amelyben a „Létrehozás” menüponton belül a lenyíló menüelemek között a „Tanóra létrehozása” menüpontra kattintva betöltődik az az oldal, amely a tanórát létrehozó űrlapot tartalmazza. A tanárnak meg kell adnia a tanóra címét, egy lenyíló menüben ki kell választania, hogy melyik tantárgyhoz szeretné hozzárendelni, és ha szeretne, megadhat egy rövid leírást is (de ez opciós), majd a „Létrehozás” gombra kell kattintania. A tanóra létrehozása akkor sikeres, ha az adatbázisban nem szerepel a megadott névvel megegyező, a kiválasztott tantárgyhoz tartozó tanóracím, a tanóracím és a leírás nem haladja meg a mezők maximális hosszát, és csak a megengedett karaktereket tartalmazzák. Ha ezek közül valamelyik nem teljesül, akkor az űrlap felett a hibáknak megfelelő hibaüzenet jelenik meg. Két tanóra neve megegyezhet, ha más-más tantárgyhoz tartozik. Az űrlapban megadott adatokon kívül az adatbázisban eltároljuk a tanóra azonosítóját (mint elsődleges kulcsot) és egy dátumot, hogy mikor hozták létre a tanórát.
2. Csatlakozás tanórához: Miután a tanár előkészítette a tanórát, megosztotta a diákokkal a böngészőablakot, amelyben látható a „Tanóra előkészítése csatlakozásra” oldal, létrehozta a csatlakozási kulcsot ezen az oldalon, a diák bejelentkezés után a saját felhasználói felületén megjelenik egy szövegdoboz, amelyben a diáknak meg kell adnia a tanár által közzétett kulcsot, majd a „Csatlakozás” gombra kell kattintania. A csatlakozás akkor sikeres, ha a megadott kulcs létezik, és típusa csatlakozási kulcs. Ha ezek közül valamelyik nem teljesül, akkor az űrlap felett a hibáknak megfelelő hibaüzenet jelenik meg. Sikeres csatlakozás esetén a tanár, diákokkal megosztott „Tanóra előkészítése csatlakozásra”,

oldalán megjelenített csatlakozott diákok száma egyel növekszik. A diák felhasználói felületén megjelenik a következő szöveg: „Csatlakozva tanárához: XYZ tanóracím”.

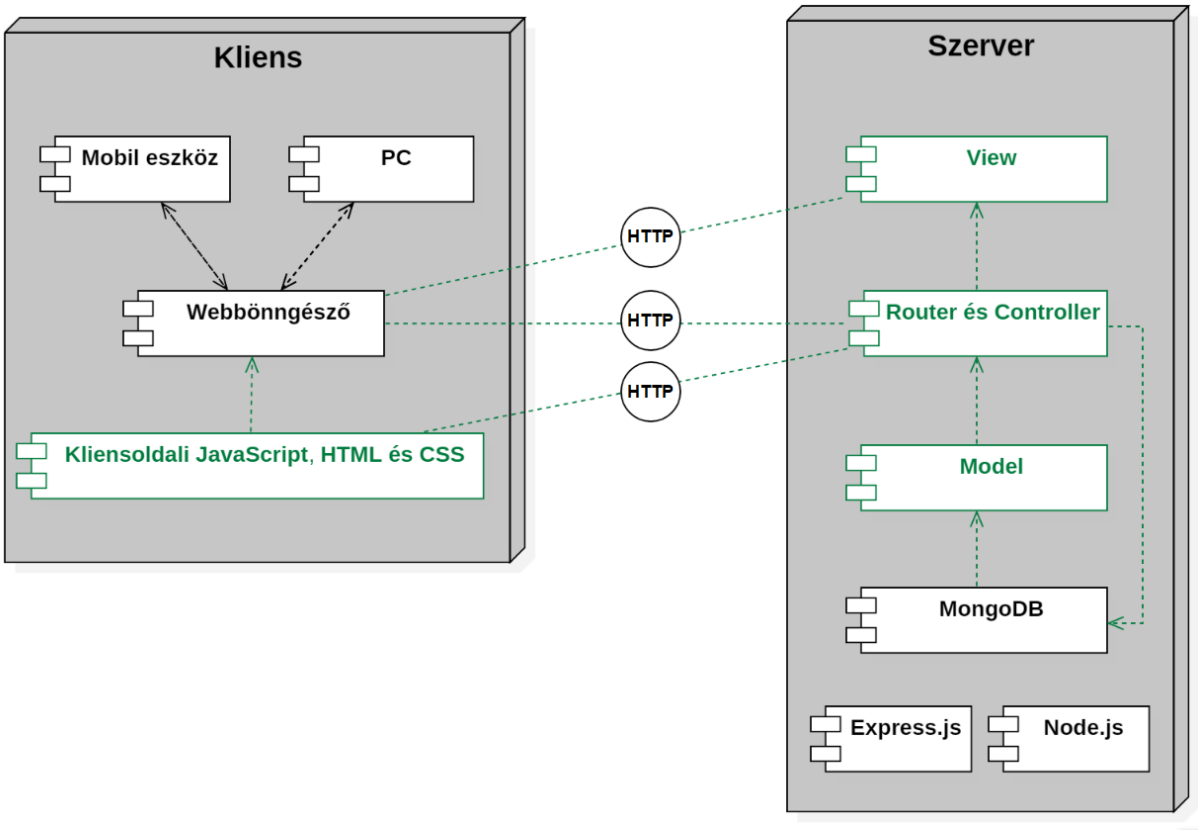
3. Tanóra elindítása: Amint a tanár úgy látja, hogy elegendő diák csatlakozott, akkor a „Tanóra elindítás” gombra kattintva elindíthatja a tanórát. Amint a tanár elindította a tanórát az indítás pillanata mentésre kerül az adatbázisban. A tanóra elindítása után a tanár elindíthatja a tanórához rendelt kérdéssorok bármelyikét. A diákok csatlakozása a tanóra elindítása után is megengedett, amíg tart a tanóra.

3.1.2. Nem funkcionális követelmények

Az alkalmazás elérhető kell legyen interneten és jó válaszidővel kell rendelkezzen. A webes felületet biztonságosan kell elérni, ezért ajánlott a HTTPS protokoll használata a webszerver és felhasználók között. A diákoknak el kell fogadniuk az alkalmazás felhasználói feltételeit.
A felhasználói interfésznek nagyon barátságosnak és könnyen használhatónak kell lennie.

3.2. Az alkalmazás architektúrája

A rendszer két fő komponense osztható: kliens és szerver. A kliens és a szerver alkomponensekből áll, amelyet a 4. ábra szemléltet. Az általam írt alkomponensek az ábrán zöld színnel vannak jelölve. Látható, hogy a szerver oldali alkalmazás Node.js környezet alatt fut, így az Express web szervert és a MongoDB adatbázist használja. A technológiának megfelelően az alkalmazás router, kontroller és view (megjelenítő) Node.js modulokból épül fel szerver oldalon. Kliens oldalon egy HTML, JavaScript és CSS technológiával megvalósított kliens oldali alkalmazás fut, ahol a megjelenítésben nagy szerepe van a Bootsrap könyvtárnak.



4. ábra A rendszert alkotó komponensek

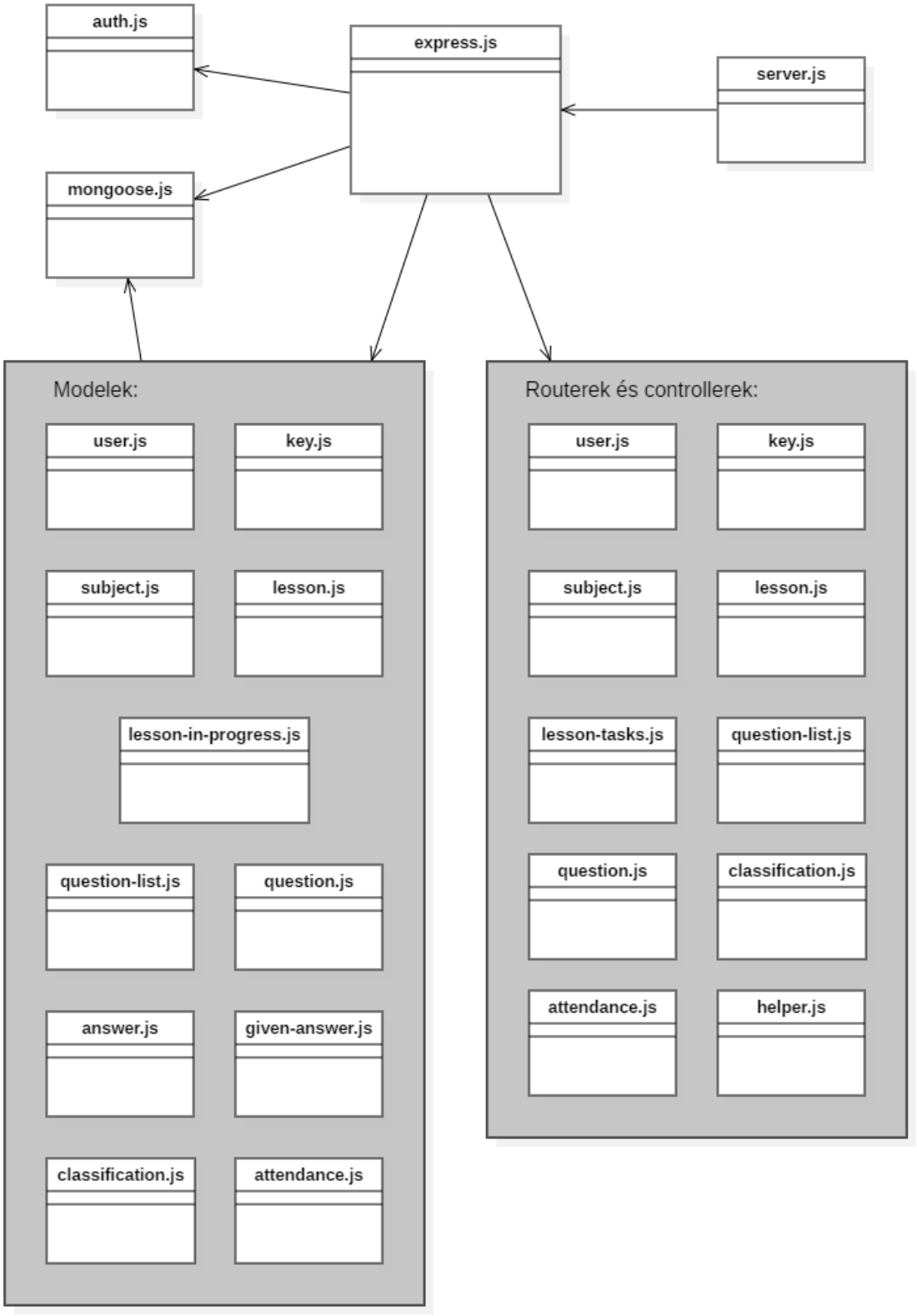
4. Részletes tervezés

4.1. Az alkalmazás Node.js moduljai közti kapcsolatok

A Node.js környezetben tervezett alkalmazások konkrétan JavaScript modulok nevű egységekben vannak megírva. A moduloknak jól meghatározott szerepe van a http kérések kiszolgálása folyamán lefutó folyamatokban. A http kérések kiszolgálása során elinduló JavaScript függvényeket sokféleképpen lehet összefűzni. Az alábbi moduláris felépítés egy viszonylag egyszerű kiszolgálási láncot alakít ki. Az alkalmazás általam megírt, valamit a felhasznált express.js, modulok közti kapcsolatokat az 5. ábrán látható.

Az alkalmazás modul (server.js) egy express.js modult használ a http kérések kiszolgálására. A kérés konkrét kiszolgálása előtt az auth.js nevű authentikációs modul (middleware) fut le, amelyik a felhasználók folyamatos azonosítását végzi. Az Express Router osztályának segítségével hozzuk létre azokat az útvonalakat (route) csoportosító Router modulokat, amelyekben megvalósítjuk az egyes funkcionalitásokat kezelő HTTP kérések kiszolgálását, így jönnek létre a routeren belül a kontroller függvények. Az alkalmazás adatait a Mongoose modul segítségével kódoljuk

modellekbe, és ezeket használjuk adat elérésre a routerekből. Az auth.js middleware modul a beléptetéshez szükséges funkcionalitásokat kezeli.



5. ábra Az alkalmazás Node.js moduljai és a köztük levő kapcsolatok

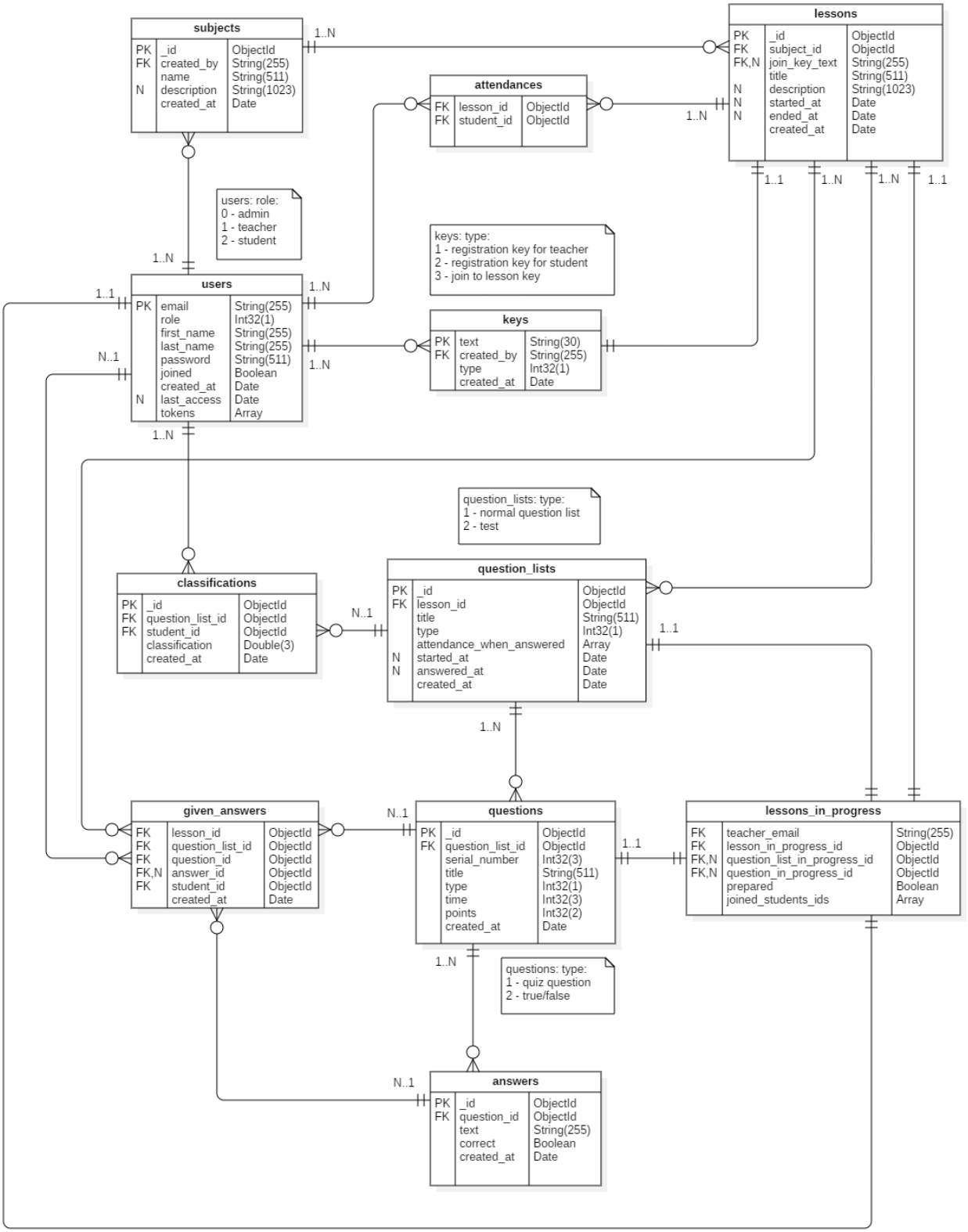
4.2. Az adatbázis tervezése

Elemeztem a funkcionális követelményeket, és ezek alapján úgy döntöttem, hogy a következő listában található adatokat kell ábrázolni a rendszerben:

1. Felhasználók (users tábla)
2. Regisztrációs és csatlakozási kulcsok (keys tábla)
3. Tantárgyak (subjects tábla)
4. Tanórák (lessons tábla)
5. Folyamatban levő tanórák (lessons_in_progress tábla)
6. Kérdéssorok (question_lists tábla)
7. Kérdések (questions tábla)
8. Válaszok (answers tábla)
9. Leadott válaszok (given_answers tábla)
10. Osztályzatok (classifications tábla)
11. Jelenlétek (attendances tábla)

Úgy döntöttem, hogy az adatok ábrázolását MongoDB NoSQL típusú adatbázisban valósítom meg, mert egyes táblában szereplő dokumentumok estében indokolt az eltérő struktúrájú dokumentumok eltárolása. Az adatok közti relációkat az . ábra szemlélteti.

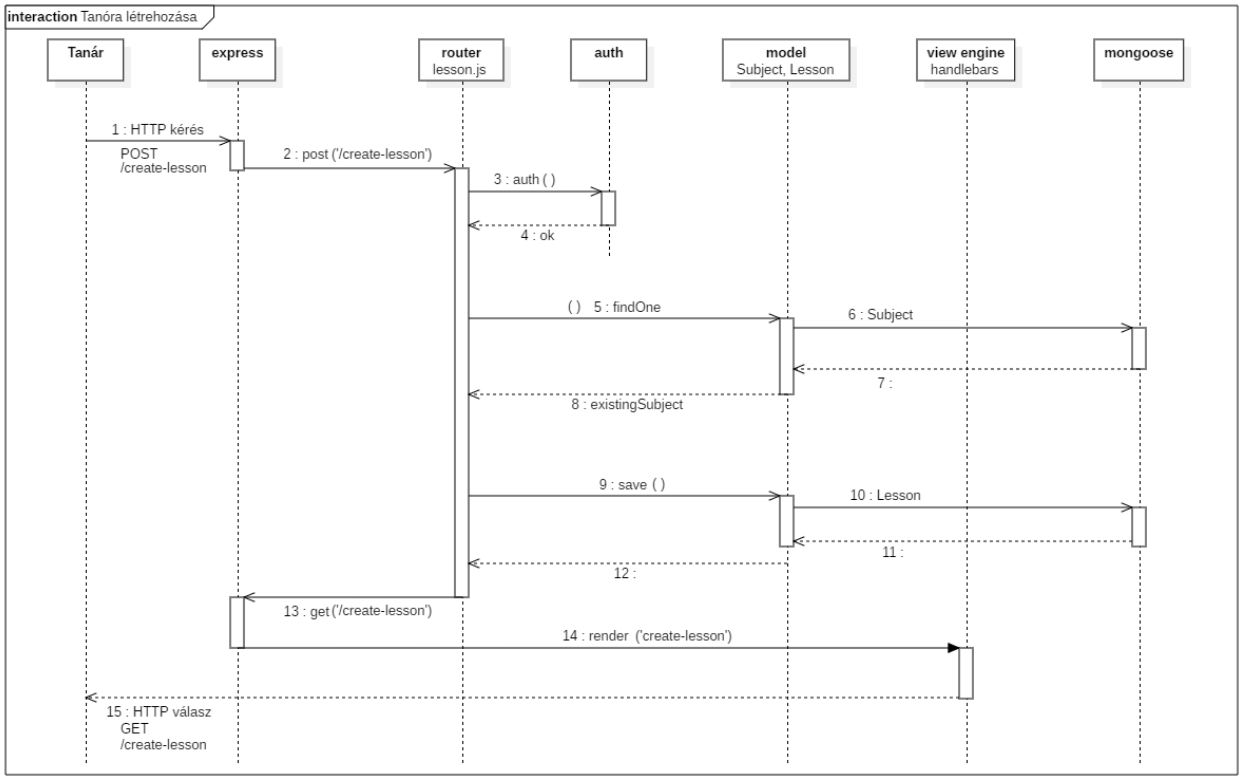
Az adatok MongoDB adatbázisban lettek tárolva, és a végrehajtó részekből a Mongoose könyvtáron keresztül értem el őket. Ennek megfelelően az adatoknak a Mongoose követelményeinek megfelelő modelleket írtam, ezek megtalálhatóak az alkalmazás models nevű könyvtárában. Az adatbázis diagramja megtekinthető a 6. ábrán.



6. ábra Adatbázis diagram

4.3. Egy funkcionalitás végrehajtásának tervezése

Egy általános funkcionalitás, például a ***Tanóra létrehozás*** (4.5.) funkcionalitás szekvencia diagramja az 7. ábrán látható, ebben a sorrendben futnak az előzőleg bemutatott Node.js modulok függvényei a http kérések kiszolgálásakor.



7. ábra Tanóra létrehozás szekvencia diagram

Az alkalmazás egyes funkcionalitásai nem egyszerű http kérések és válaszok lefutásából állnak, hanem kliens oldali Ajax kérések által vannak vezérelve. Ilyen például a csatlakozott diákok számának megjelenítése a Tanóra előkészítése csatlakozásra oldalon funkcionalitás, amelyet az alábbiakban ismertetek.

A kliensoldalról 2 másodpercenként érkezik egy Ajax GET kérés (/poll-lesson) a szerver felé, hogy küldje el az előkészített tanórához csatlakozott diákok darabszámát. Az Ajax kéréseket a JQuery könyvtárral valósítottam meg, egy ilyen kód látható a 8. ábrán.

```
121         {{>bootstrap}}
122
123         <input type="hidden" id="lesson_id" value="{{lesson._id}}">
124
125         <script>
126             $(function () {
127                 function poll() {
128                     setTimeout(poll, 2000);
129
130                     $.get("/poll-lesson?lessonId=" + document.getElementById("lesson_id").value)
131                         .done(function (numOfJoineds) {
132                             console.log("GET:" + numOfJoineds);
133                             $("#numOfJoinedStudents").html(numOfJoineds);
134                         })
135                         .fail(function (error) {
136                             console.log(error);
137                         });
138                 };
139                 poll();
140             });
141         </script>
142     </body>
143 </html>
```

8. ábra Kérés a szerver felé

A kérést a **routers/lesson.js** router és egyben controller hajtja végre. A controler a 9. ábrán látható.

```
171 // Csatlakozott diákok számának lekérdezése az adatbázisból,
172 // és megjelenítése a Tanóra előkészítése oldalon:
173 router.get('/poll-lesson', auth, async (req, res) => {
174     const lessonId = req.query.lessonId
175
176     const preparedLesson = await LessonInProgress.findOne({ lesson_in_progress_id: lessonId })
177     if (preparedLesson) {
178         const numOfStudents = preparedLesson.joined_students_ids.length
179
180         res.write(`${numOfStudents}`)
181         res.end()
182     }
183 })
184
```

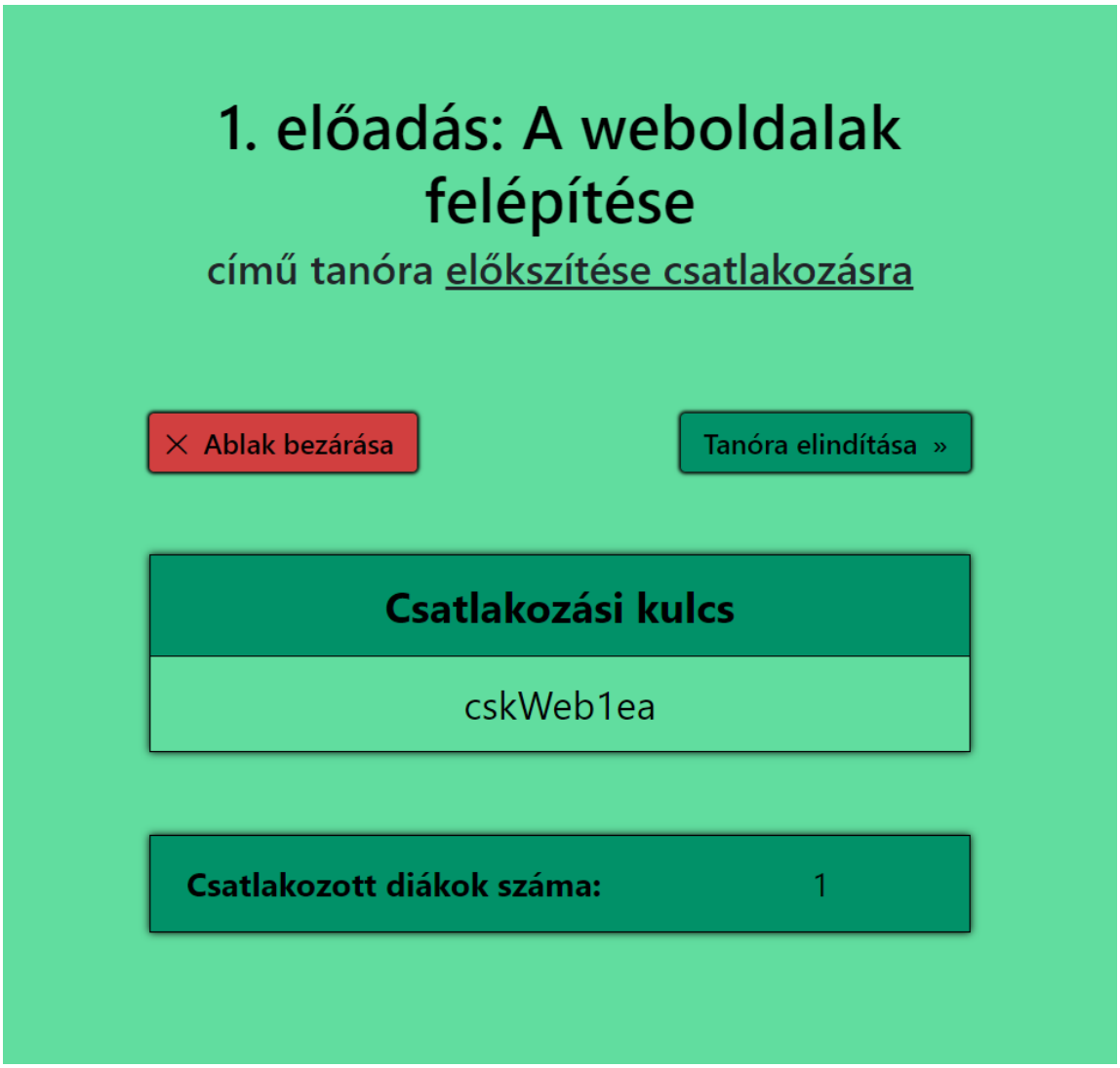
9. ábra Az Ajax GET kérést végrehajtó controller

A **res.write()** beépített függvény a csatlakozott diákok darabszámát tartalmazó **numOfStudent**s változót a 8. ábrán a **131.** sorban **.done()-on** belül definiált függvény **numOfJoineds** változójába téríti vissza, és a **133.** sorban levő kódrészlet beírja a változó értékét a **numOfJoinedStudents** azonosítójú HTML elemébe. A **prepare-lesson.hbs** view kódrészlete a 10. ábrán látható:

```
80      <div class="row d-flex justify-content-center mt-5 pt-3 m-0 w-100 h-min-content">
81          <div class="col-xl-6 font-size-21pt border-black box-shadow-black m-0 p-0">
82              <div class="row color-black bg-color-4 text-center m-0 p-3 w-100">
83                  <div class="col-md-8 d-flex justify-content-start align-items-center t
84                      <b>Csatlakozott diákok száma:</b>
85                  </div>
86                  <div id="numOfJoinedStudents" class="col-md-4 d-flex justify-content-c
87                      0
88                  </div>
89              </div>
90          </div>
91      </div>
```

10. ábra Tanóra előkészítése csatlakozásra oldal view (HTML részlete)

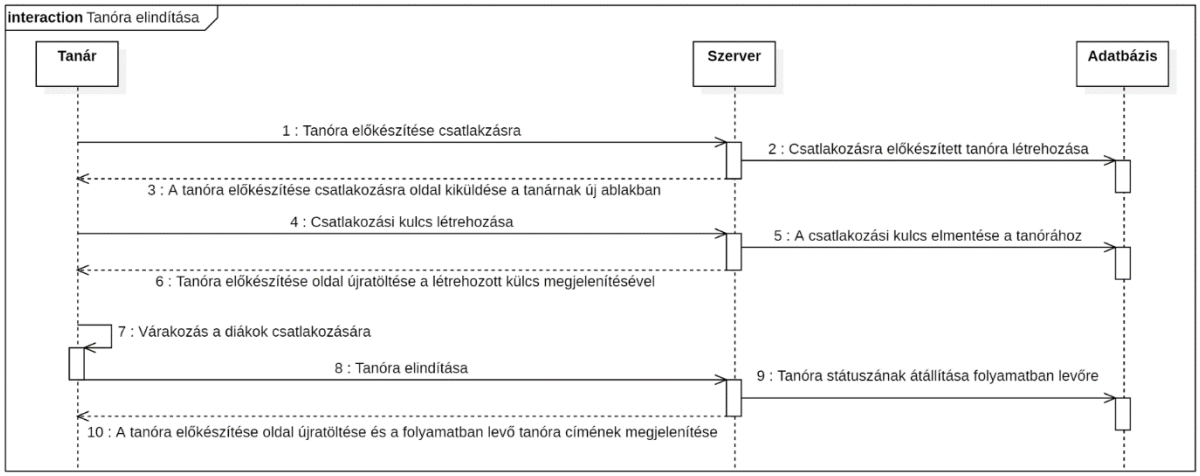
Egy csatlakozott diák esetén a Tanóra előkészítése oldal Csatlakozott diákok száma után megjelenik egy 1-es, amelyet a 11. ábra szemléltet:



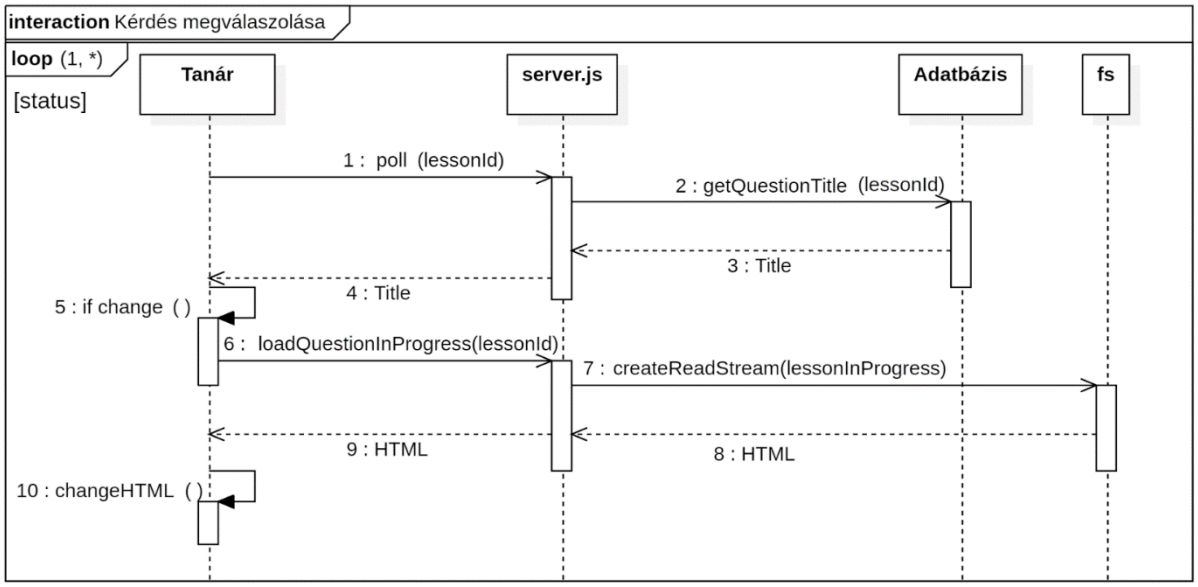
11. ábra Tanóra előkészítése csatlakozásra oldal tartalma sikeres Ajax kérés után

4.4. A Tanóra indítása és a diák Kérdés megválaszolása funkcionalitások

A 12. ábrán a tanár Tanóra indítása, a 13. ábrán pedig a diák Kérdés megválaszolása funkcionalitást szemléltetem szekvenciadiagramok segítségével.



12. ábra Tanóra indítása szekvencia diagram



13. ábra Kérdés megválaszolása szekvencia diagram

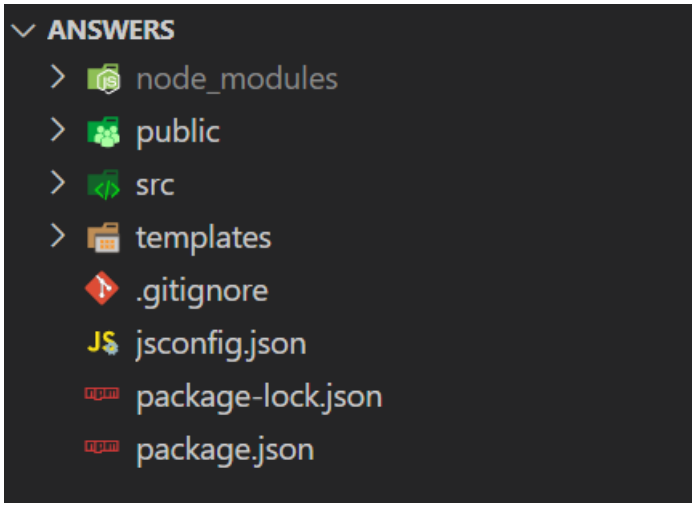
A kérdéseket a lesson-tasks.js fájlban levő start-task route által elindított questionTask() nevű időzítő függvény cserélgeti a kérdés leírásából (questions collection az adatbázisban) kiolvasott időzítések szerint.

4.5. Az alkalmazás könyvtár szerkezete

A gyökérkönyvtár alatt 14. ábra) találhatóak az alábbi kód szervezést megoldó könyvtárak:

- a node_modules: az alkalmazás által használt Node.js modulok
- public: a webszerver által webről elérhető könyvtárak
- src: itt találhatóak a tulajdonképpeni forrás fájlok:
  - models: Mongoose adat modellek
  - routers: routerek
  - db: a Mongoose modul
  - middleware: Express middleware modulok
  - server.js az alkalmazást indító modul
  - helper.sj: segéd függvényeket tartalmazó modul.
- templates: a Handlears sablonokat tartalmazó megjelenítést megvalósító fájlok.

A .gitignore, jsconfig.json, package-lock.json és package.json fájlok az npm és git segéd fájljai, illetve a .git könyvtár tartalmazza a git fájljait.



14. ábra Könyvtárszerkezet - gyökérkönyvtár

5. Üzembe helyezés és kísérleti eredmények

5.1. Üzembe helyezési lépések

A projekt GitHub tarolóban van és mellékeljük a dolgozathoz egy zip fájlban is. A zip kipakolása után vagy a git clone művelet után (ha rendelkezünk a tároló elérési adataival), az alkalmazás előkészíthető indításra.

Az elindításhoz szükséges a Node.js legalább 14. verziója. A MongoDB legalább 4.4-es verziója kell fusson, és annak elérhetőségét be kell konfigurálni a src/db/mongoose.js fálban.

Az alkalmazás gyökér könyvtárában ki kell adni az
npm install
parancsot, amennyiben a node_modules-ban nincsenek meg a telepített modulok.
Mindezek után be kell olvasni a MongoDB-be az adatbázist. A db_backups könyvtárban ki kell csomagolni az adatbázist az ott levő zip fájlból, majd importálni:
mongo answeRS --eval "db.dropDatabase();"
mongorestore -d answeRS answeRS

Utána az alkalmazás a pm2 vagy nodemon felügyelő programokkal indítható, pl.
nodemon src/server.js -e js,hbs
Folyamatos háttérben való futtatáshoz ajánlottabb a pm2 felügyelőt használni:
pm2 start src/server.js

Az alkalmazás a Node.js alapértelmezett 3000-es portján fut, ez megváltoztatható az indító parancsban a port megadásával.
Teszteléshez az alkalmazást egy Ubuntu 18 Linux szerveren futtattuk a 3000-es port-on. Internetről a https://answers.ms.sapientia.ro címen volt elérhető. Ez a cím az adott névvel rendelkező egyetemi webszervert hívta meg a 80-as porton, ahol egy Apache webszerverrel volt megoldva a HTTPS titkosítás és átirányítás az Ubuntu gép 3000-es portjára.

5.2. Kísérleti eredmények és mérések

Az alkalmazást a megadott https://answers.ms.sapientia.ro címen teszteltük. Az első teszt folyamán négy diák felhasználó válaszolt meg egy három kérdésből álló kérdéssort. Ennek eredményét az oktató az alábbi 15. ábra szerint tudta megtekinteni. Az alkalmazás helyesen működött 10 felhasználóval is.
Tekintettel a jelenlegi járványügyi helyzetre való tekintettel a felhasználók különböző helyszínekről érték el az alkalmazást és nem egy osztályteremből.

15. ábra Diákok által leadott válaszok és a kapott osztályzatok megjelenítése a tanár számára

6. A rendszer felhasználása

A rendszer felhasználásakor fontos szem előtt tartani néhány pedagógiai szempontot:

- Időben értesítsük a hallgatókat, hogy a tanóra ideje alatt legyen náluk olyan eszköz, amelyet clicker eszközként használhatnak.
- 10-20 perces időközöknél gyakrabban ne indítsunk el egy kérdéssort [1].
- A kérdéssor összeállításánál tartsuk szem előtt, hogy mekkora létszámú tanóra keretein belül szeretnénk felhasználni. Ha nagy létszámú tanórán szeretnénk elindítani a kérdéssort, akkor az egyes kérdések létrehozásánál állítsunk be legalább egy percet a rendelkezésre álló megválaszolási időnek kérdésenként [1].
- A rendszert ne használjuk pusztán jelenlét készítésére [1].

6.1. A rendszer felhasználása hagyományos oktatás esetén

Tanóra közben, amikor a tanár használja a rendszert, szüksége van egy számítógépre, amelynek képernyőjét csak ő látja (ezen vezéreli a folyamatban levő a tanórát), és egy vetítőre vagy nagy képernyőre, amelyet a számítógépére csatlakoztat. Ezt a vetítőt/képernyőt megosztja a diákokkal is, és ezen jelennek meg számukra a megválaszolandó kérdések, a hozzá tartozó válaszok, és a válaszokhoz tartozó betűjelek (A/B/C/D vagy I/H).

Tanórán kívül, amikor a regisztrációs kulcsokat. tantárgyakat, tanórákat, kérdéssorokat, kérdéseket és válaszokat hozza létre, szerkeszti, törli, akkor elégséges egy képernyős számítógép használata is.

A diákok regisztráción kívül csak tanórákon használják a rendszert. Egy adott diáknak csak egy clicker eszközre van szüksége a tanórába való teljes értékű bekapcsolódáshoz. Ez az eszköz lehet okostelefon, tablet vagy laptop, amely rendelkezik internetkapcsolattal. Ezen a clicker eszközön jelennek meg a válaszlehetőségek betűjelei, amelyek közül egyet kell kiválasztania a diáknak és a válasz automatikusan elmentődik.

6.2. A rendszer felhasználása online oktatás esetén

Online oktatás esetében valamivel nehézkesebb a rendszer használata, mert szükség van egy videókommunikációt lehetővé tevő platformra. A tanárnak szüksége van egy számítógépre, amelynek főképernyőjét csak ő látja (ezen vezéreli folyamatban levő a tanórát) és egy másik képernyőre, amelyet a számítógépére csatlakoztat. Ezt a képernyőt megosztja a diákokkal a videókommunikációt lehetővé tevő platform (pl.:Google Meet, Skype) segítégével, és ezen jelennek meg számukra a megválaszolandó kérdések, a hozzá tartozó válaszok, és a válaszokhoz tartozó betűjelek.

Az adott diáknak szüksége van egy clicker eszközre (okostelefon, tablet vagy laptop amely rendelkezik internetkapcsolattal). Ezen a clicker eszközön jelennek meg a válaszlehetőségek betűjelei, amelyek közül egyet kell kiválasztania a diáknak. Az online oktatás olyan szempontból hátrányosabb a rendszerem esetében, hogy a diáknak legalább két eszközre van szüksége, vagy egy eszközre két képernyővel, azért hogy egyiken tudja nézni azt a felületet, amit a tanár osztott meg vele, a másikon, pedig tudja leadni a válaszait az egyes kérdésekre.

7. Következtetések

7.1. Megvalósítások

A dolgozatban sikerült megvalósítani a kitűzött CRS rendszerekhez kötődő fontosabb funkcionalitásokat. A rendszer jelenlegi állapotában a rendszergazda képes regisztrációs kulcsot létrehozni a tanárok számára, a tanárok képesek regisztrációs kulcsot létrehozni diákok számára. A rendszergazda és a tanár kilistázhatja, szerkesztheti, törölheti az általa létrehozott kulcsokat.

A tanár létre tud hozni tantárgyak, tanórákat, csatlakozási kulcsot a tanórához, két különböző típusú kérdéssort (normál kérdéssort és felmérőt), két különböző típusú kérdést (kvíz kérdés 2-4 válaszlehetőséggel, igaz/hamis kérdés). A tanár elindíthat tanórát, kérdéssort (a kérdéssorhoz tartozó kérdések automatikusan indulnak), befejezhet tanórát és szükség szerint megszakíthat kérdéssort. A diákok válaszolhatnak a kérdéssorokra és a tanár megtekintheti az eredményeket. Mindezt sikerült több diák felhasználóval kipróbálni.

7.2. Összegzés

Az ARS és CRS rendszerek olyan médiaelemeket tartalmazhatnak, amelyek segítségével interaktívabbá tehetők az előadások, ezáltal pozitív irányban befolyásolható a diákok tanuláshoz való hozzáállása, nem is beszélve arról, hogy a diákok aktivitása és fejlődése egyénenként nyomon követhető, és az adatok adatbázisba menthetők. Az adatbázisból kinyert értékes információk befolyásolhatják az oktatók által használt módszereket, illetve a tananyag megközelítésének módját.

Az általam készített CRS rendszer használható kurzusokon és szemináriumokon egyaránt, hagyományos („face-to-face”) és online oktatásban is.

8. Irodalomjegyzék

[1] H. Bakonyi V., ifj. Illés Z., Illés Z.: Valós idejű oktatást segítő rendszerek, INFODIDACT 2017, 10. Informatika Szakmódszertani Konferencia, Zamárdi, 23-25.11.2017.

[2] Bakonyi, V.H. and Illés, Z., 2018, October. Real time classroom systems in teachers training. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (pp. 206-215). Springer, Cham.

[3] Margaret R.: ICT (information and communication technology, or technologies), https://whatis.techtarget.com/definition/ICT-information-and-communications-technology-or-technologies, utolsó elérés: 2021.07.05.

[4] Wikipédia (angol): Mustache (template system), https://en.wikipedia.org/wiki/Mustache_(template_system) , utolsó elérés: 2021.07.05.

[5] handlebars, http://handlebarsjs.com/, utolsó elérés: 2021.07.05.

[6] GitHub: handlebars.js, https://github.com/wycats/handlebars.js#differences-between-handlebarsjs-and-mustache, utolsó elérés: 2021.07.05.

[7] npm: handlebars, https://www.npmjs.com/package/handlebars, utolsó elérés: 2021.07.05.

[8] jQuery, https://jquery.com/, utolsó elérés: 2021.07.05.

[9] Wikipédia (magyar): jQuery, https://hu.wikipedia.org/wiki/JQuery, utolsó elérés: 2021.07.05.

[10] Wikipédia (angol): jQuery, https://en.wikipedia.org/wiki/JQuery, utolsó elérés: 2021.07.05.

[11] B. Bibeault, Y. Katz, and A. De Rosa: jQuery in Action, Third Edition, Manning Publications, 2015 augusztus, ISBN 9781617292071

[12] Wikipédia: (angol): jQuery Mobile, https://en.wikipedia.org/wiki/JQuery_Mobile, utolsó elérés: 2021.07.05.

[13] Wikipédia (angol): jQuery UI, https://en.wikipedia.org/wiki/JQuery_UI, utolsó elérés: 2021.07.05.

[14] Wikipédia (angol): Bootstrap (front-end framework), https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) , utolsó elérés: 2021.07.05.

[15] Bootstrap, https://getbootstrap.com/docs/4.3/about/overview/, utolsó elérés: 2021.07.05.

[16] DZone: G. Singh: Bootstrap and its Features, https://dzone.com/articles/bootstrap-and-its-features, utolsó elérés: 2021.07.05.

[17] Wikipédia (magyar): Node.js, https://hu.wikipedia.org/wiki/Node.js, utolsó elérés: 2021.07.05.

[18] Wikipédia (angol): Node.js: https://en.wikipedia.org/wiki/Node.js, utolsó elérés: 2021.07.05.

[19] npm: npm Documentation, https://docs.npmjs.com/, utolsó elérés: 2021.07.05.

[20] Node.js, https://nodejs.org/, utolsó elérés: 2021.07.05.

[21] Wikipédia (angol): Express.js, https://en.wikipedia.org/wiki/Express.js, utolsó elérés: 2021.07.05.

[22] Express: https://en.wikipedia.org/wiki/Express.js, utolsó elérés: 2021.07.05.

[23] npm: express: https://www.npmjs.com/package/express, utolsó elérés: 2021.07.05.