XXIII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2020. május 21–24.

SConES

Diákköri konferenciákat egységesítő és karbantartó online felület

Szerzők:

Rozsnyai Tamás
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, II. év
Tamás Andrea
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Témavezetők:

dr. Sulyok Csaba, tanársegéd
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar
Erőss Ervin, szoftverfejlesztő,
Codespring
Séra Barna, szoftverfejlesztő,
Codespring

Kivonat

A Student Conference Electronic Submission System, röviden SConES nevet viselő projekt célja egy egységes online platform biztosítása nemzetközi, de főként erdélyi diákköri konferenciák meghirdetésére és azokra való jelentkezésre. Valamint kiemelt cél az adminisztrációs folyamatok digitalizálása és egyszerűsítése.

Célközönsége ezen rendezvények szervezői, a jelentkezni kívánó diákok, ezek vezetőtanárai valamint az elbíráló bizottságok tagjai. A platform lehetőséget nyújt a konferenciák böngészésére, részletes adataiknak megtekintésére. Ezek mellett kiemelt funkcionalitás, hogy a diákok megtekinthessék a már beküldött dolgozatukat, figyelemmel követhessék a dolgozat állapotváltozásait, módosíthassák saját adataikat.

A dolgozat részletezi a projekt felépítését, a komponensek közötti kommunikációt, a fejlesztés során felhasznált technológiákat és eszközöket valamint bemutatja az alkalmazás működését.

Tartalomjegyzék

Bevezető **1**

1. A SConES projekt **4**

1.1. Funkcionalitások 4

1.2. Architektúra 6

1.2.1. Komponensek 6

1.2.2. Kommunikáció 7

1.2.3. E-mail küldés 8

1.3. Adatmodell 9

1.4. Nemzetköziesítés és lokalizáció 10

2. Technológiák **11**

2.1. Szerver oldali technológiák 11

2.2. Kliens oldali technológiák 12

3. Alkalmazott módszerek és felhasznált eszközök **16**

3.1. Scrum munkamódszer 16

3.2. Docker 16

3.3. Verziókövetés, CI/CD 17

3.4. Statikus kódelemzés 19

4. Az alkalmazás működése **20**

4.1. Főoldal 20

4.2. Lebegő sáv 21

4.3. Profil icon/menü 21

4.4. Konferencia részletei 21

4.5. Dolgozat feltöltése 22

4.6. Leadott dolgozatok megtekintése 23

4.7. Hiba-, Siker- Jelzések 24

5. Következtetések és továbbfejlesztési lehetőségek **26**

Bevezető

A magyar egyetemi hallgatók életében már régóta fontos szerepet játszik a tudományos diákköri tevékenység. A tudományos diákkör feladata minőségi képzést és tehetséggondozást biztosítani a felsőoktatásban. Az első tudományos diákköri konferenciákat az 1950-1951-es tanévben szervezték meg az Eötvös Loránd Tudományegyetemen és az akkori Veszprémi Vegyipari Egyetemen. Ezt követően több intézményben is elkezdődött a Tudományos Diákköri Konferenciák (TDK) formájában történő tehetséggondozás. Az első országos szintű rendezvény, Tudományos diákkörök első országos konferenciája néven, 1955-ben került megszervezésre. A konferencián összesen 109 előadás hangzott el. A következő években a nagy érdeklődés miatt a konferenciát tudományterületekre tagolták. 2003 óta 16 szekcióban adhatnak elő az egyetemi hallgatók. [2]

Az intézményi Tudományos Diákköri Konferenciák (TDK) lehetőséget nyújtanak a diákoknak, hogy kutatómunkájuk eredményét vagy alkotásaikat, szakértő zsűri előtt bemutathassák. A szakmai visszajelzés mellett a rendezvény megmérettetést is jelent, mivel a legjobbak kijutási lehetőséget kapnak az Országos Tudományos Diákköri Konferenciára (OTDK) [20], amelyet minden második év tavaszán szerveznek meg. Intézményi TDK-k a határon túli erdélyi, felvidéki, kárpátaljai és vajdasági hallgatók számára is megrendezésre kerülnek.

1973-ban alakult meg az Országos Tudományos Diákköri Tanács (OTDT) [21]. Szerepét jogszabályok rögzítik, amelyek mint a tudományos diákköri tevékenység összehangolásáért felelős szervezetet nevezik meg. A szervezet az OTDK gördülékeny megszervezése és az adatok megfelelő tárolása érdekében 2010-ben egy új online rendszert hozott létre. Ez a rendszer minden olyan funkcionalitással rendelkezik, amely lehetővé teszi a diákok regisztrációját a megmérettetésre, a kapcsolattartást, valamint az adattárolást [22].

A különböző intézményes TDK-k nagyrésze is rendelkezik online rendszerrel. Az Erdélyi Tudományos Diákköri Konferenciák (ETDK) közül saját rendszere van a kolozsvári Reál és Humántudományi ETDK-nak [28], a temesvári Műszaki Tudományos Diákköri Konferenciának [8], a marosvásárhelyi TDK-nak, amely egy orvostudományi és gyógyszerészeti konferencia [16]. Saját konferenciát rendez a Sapientia Erdélyi Magyar Tudományegyetem kolozsvári [32], marosvásárhelyi [33] és csíkszeredai [31] kara. Ezeken kívül Erdélyben szervezik meg a Partiumi Tudományos Diákköri Konferenciát, amely szintén rendelkezik saját online rendszerrel [24].

A diákok számára nehéz a különböző konferenciák honlapjainak észben tartása és kezelése is, hiszen mindegyik weboldal más és különböző dolgot vár el a felhasználójától. Például amíg a Vajdasági Magyar Tudományos Diákköri Konferencia (VMTDK) [36] online felületet biztosít

a dolgozatok leadására, addig a Kárpátaljai TDK-ra [37] e-mailen keresztül lehet jelentkezni.

A Student Conference Electronic Submission Sytem (SConES) nevet viselő projekt ötletét az intézményes TDK-k menedzselésének különbözősége adta. Amint a projekt neve is tükrözi, egy egységes felület létrehozása volt a cél, amelyen keresztül megvalósulhat a tudományos diákköri konferenciákra való jelentkezés. Így a SConES online felület létrejöttét főként az erdélyi magyar, de akár a nemzetközi TDK-k meghirdetésének és lebonyolításának egységesítése motiválta. Célközönsége a konferenciák szervezői, a jelentkezni kívánó diákok, ezek vezetőtanárai, valamint az elbíráló bizottságok tagjai.

A projekt célja, hogy felhasználói számára megkönnyítse a konferenciák böngészését, az azokra való jelentkezést. Emellett célja az is, hogy reszponzív megjelenítésével kényelmessé tegye a rendezvények elérését akár mobiltelefon böngészőből is. Tervezéskor fontos szempontot jelentett a konferencia szervezést minél testre szabhatóbbá tenni. Így lehetőség van a rendezvényeken belül a különböző szekciók határidőjének egyenkénti megadására, a kivonat és a dolgozat terjedelmének, formátumának egyéni megválasztására. Ugyanakkor, ha ezek az egyéni beállítások nem történnek meg, a rendszer az alapértelmezetten beépített konfigurációt használja. Ezenkívül a diákok számára biztosított a dolgozat leadása, társszerzők, valamint témavezetők kiválasztása. Lehetőség van a már leadott dolgozatok megtekintésére és a feltöltött dokumentum letöltésére. A projekt megvalósításai közé tartozik még az e-mail küldés bizonyos események bekövetkezésekor, mint amilyen a regisztráció vagy egy dolgozat sikeres leadása.

A SconES projekt bizonyos megvalósításainak ötletét a már 2002 óta működő EasyChair [6] nevű weboldal inspirálta. Ez a felület szintén konferenciák meghirdetésére és azokra való jelentkezésre szolgál. Indulása óta több mint 80000 konferencia létrejöttét segítette elő. Az említett weboldal azonban nem ad lehetőséget diákköri konferenciák megszervezésére. Ugyanis a diákköri konferenciák és standard konferenciák között számottevő különbség van. A TDK-k esetében az elkészült dolgozat létrejöttét és a teljes kutatómunkát vagy alkotói folyamatot vezetőtanári felügyelet és segítség kíséri. Ezek a konferenciák lehetnek az első mérföldkövek a tudományos munkában. Mivel a TDK dolgozatok és a teljes bemutatási folyamat, a zsűri és a közönség kérdéseivel együtt nagyban hasonlít az államvizsga dolgozat megvédéséhez a konferenciákon való részvétel által a hallgatók rutint szerezhetnek. Ezek mellett a TDK-n való bemutatkozás lehetősége nagyban hozzájárul a mesteri és doktori képzésekre való bekerüléshez.

A dolgozat 1. fejezete részletezi a megvalósult funkcionalitásokat, bemutatja az architektúrát, valamint az adatmodellt. Ezt követően a 2. fejezet ismerteti a felhasznált technológiákat szerver és kliens oldalon. A 3. fejezetben sor kerül az alkalmazott módszerek és eszközök bemutatására, amelyet a 4. fejezetben az alkalmazás működésének részletes leírása követ. Az utolsó, 5. fejezet következtetéseket és továbbfejlesztési lehetőségeket tartalmaz.

A projekt fejlesztése a 2018-as Codespring Mentorprogram részeként, a szakmai gyakorlat

alatt indult 2019 nyarán. A fejlesztői csapatot Rozsnyai Tamás és Tamás Andrea alkotta. A fejlesztés elején a csapatnak a mentorokkal együtt lehetősége nyílt egy beszélgetésre olyan szakemberekkel, akik több éves tapasztalattal rendelkeznek TDK szervezés terén. Nekik köszönhetően a SConES projekt megvalósításai közé olyan funkcionalitások kerülhettek, amelyek ténylegesen hiánypótlóak lehetnek. A 2019-2020-as tanév első félévében, a csoportos projekt tantárgy keretén belül a csapat kibővült egy új taggal, név szerint Koncsag Beátával. A projekt létrejöttéért köszönet illeti a Codespring által kijelölt mentorokat: dr. Sulyok Csabát, Erőss Ervint és Séra Barnát.

1. A SConES projekt

A SConEs rendszer egy egységes platformot nyújt a TDK-k szervezőinek és résztvevőinek, ahova a konferenciára jelentkező diákok könnyen fel tudják tölteni a dolgozatokat. Ugyanitt, a vezető tanárok által jóváhagyott dolgozatok, automatikusan továbbítódnak az elbíráló bizottsághoz. A diákok nyomonkövetik a dolgozat útját és láthatják milyen stádiumban van az.

Ebben a fejezetben a megvalósított funkcionalitásokról, a rendszer architektúrájáról és adatmodelljéről, valamint nemzetköziesítésről, lokalizációról és ezeknek szerepéről lesz szó.

1.1. Funkcionalitások

A SConES weboldalon a főbb funkcionalitások kihasználásához szükséges a felhasználónak bejelentkeznie. Ugyanis ha valaki egy dolgozatot szeretne leadni, egyrészt érzékeny információkat kell megadjon, úgy magáról mint a dolgozatról. Másrészt, ha az illető nyomon szeretné követni, hogy a dolgozata milyen stádiumban van vagy szeretné szerkeszteni a végleges bírálat előtt, bizonyítania kell, hogy az az ő dolgozata. Ezt a SConES egy regisztráció-bejelentkezéses rendszerrel oldja meg. Ezért a nem bejelentkezett felhasználónak, a vagy vendégnek (guest), limitált lehetőségei vannak.

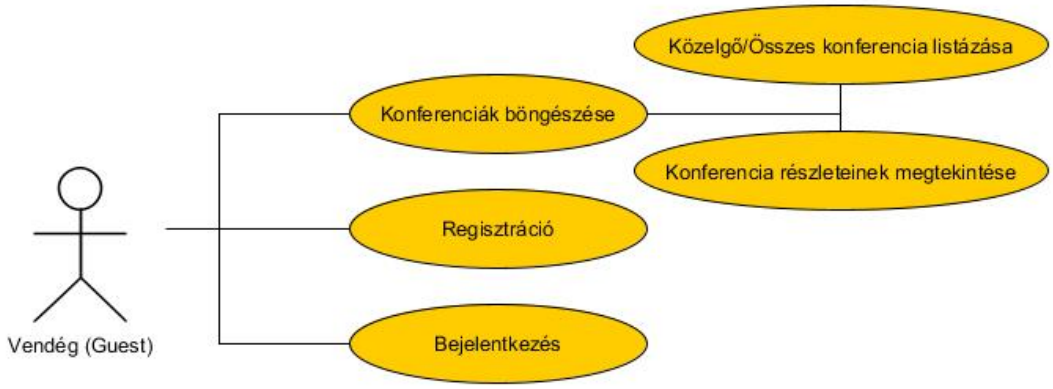
Vendég felhasználó

Amint az 1. ábra is mutatja, egy vendég (nem bejelentkezett) felhasználó böngészhet a konferenciák között, itt külön megnézheti csak a közelgőket, vagyis azokat, amelyekre még lehet dolgozatot leadni, vagy kilistázhatja az összes konferenciát. Ezeket a „Közelgő” illetve az „Összes” kulcsszavak alatt lehet elérni. Hozzáférhet a konferenciák részleteihez, a konferencia saját lapján. Itt megtekintheti a leírását, időpontját, helyszínét, a szekciókat amelyekhez lehet dolgozatot leadni a konferencián belül, leadási határidőket stb. Amennyiben még nem tette meg, regisztrálhat, illetve bejelentkezhet. Sikeres regisztráció esetén, a felhasználó értesítést kap a megadott e-mail címre.

Bejelentkezett felhasználó

Ahogy a 2. ábrán látható, egy bejelentkezett felhasználónak már több funkcionalitás válik elérhetővé. Ő is böngészhet a konferenciák között, de neki már lehetősége van megjelölni azokat a konferenciákat, amik felkeltették az érdeklődését, hogy a jövőben könnyebben visszatérhessen ezekhez.

A közelgő konferenciákhoz leadhat dolgozatot is, konferencia saját lapján. Ezen a

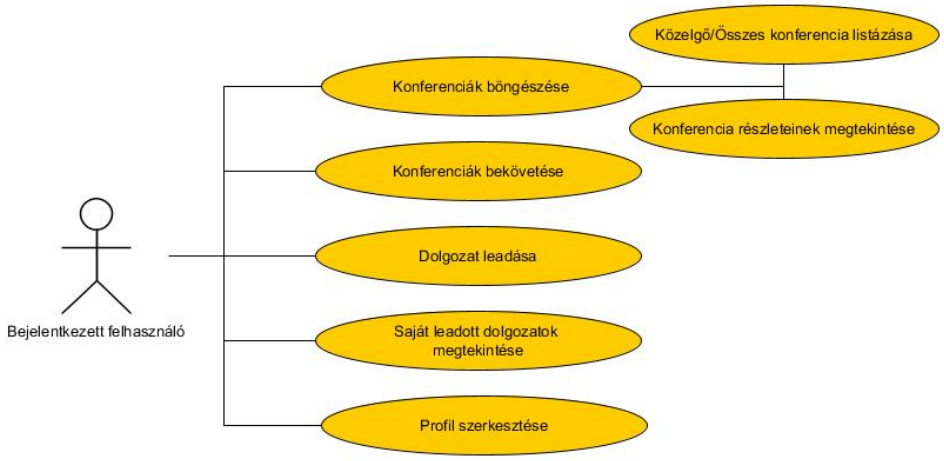


1. ábra. Vendég felhasználó funkcionalitásai

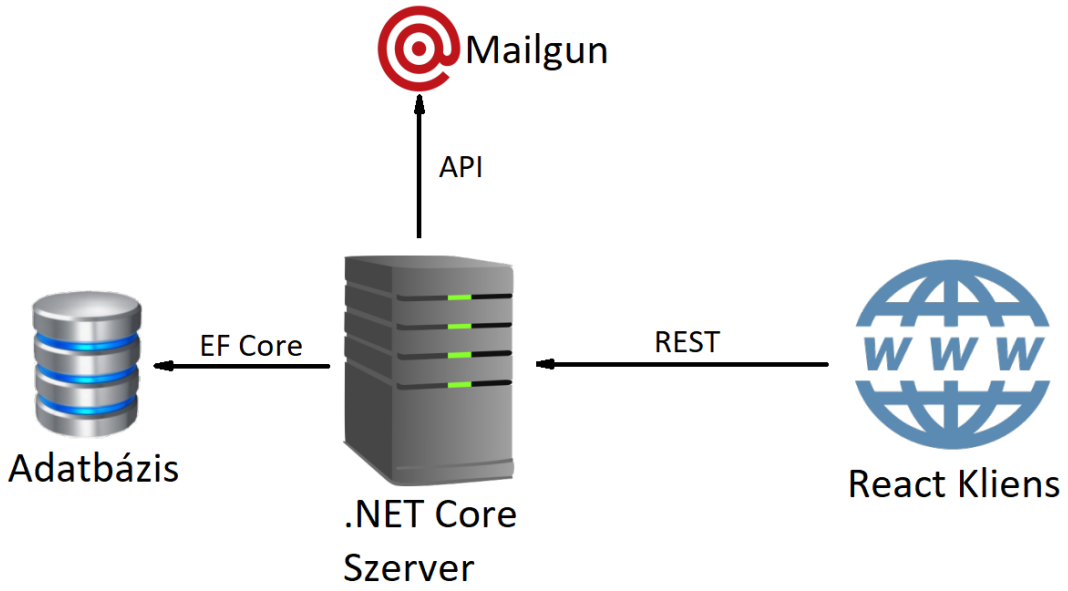
ponton ki kell választani egy szekciót. A szekció meghatározhatja az „absztrakt/kivonat” hosszát, a dolgozat méretét illetve formátumát és akár a leadási határidőt is. Amennyiben ezek a megkötések nincsenek meghatározva az adott szekciónak, örökli a beállításokat a konferenciától. Kap a dolgozat egy címet, egy rövid leírást a kivonathoz és csatolni kell a dolgozatot az elfogadott formátumban. Meg kell jelölni legalább egy felhasználót „Felügyelő tanár”-ként, illetve meg lehet jelölni felhasználókat társszerzőként. Ezek a felhasználók e-mailes értesítést kapnak a dolgozat leadásával kapcsolatban.

A leadott dolgozatokat a felhasználók megtekinthetik a konferenciák saját lapján (csak azok a dolgozatok jelennek meg egy felhasználónak amiknél szerzőként szerepel), illetve külön kilistázva. Itt megjelennek az információk a dolgozatról, letölthető maga a dolgozat is, valamint nyomon követhető a dolgozat állapota („Függőben”, „Jóváhagyott”, „Elfogadott”, „Elutasított”).

A felhasználó megtekintheti és szerkesztheti a személyes adatait (pl. név, egyetem). Ezeknek a változtatása akkor válik relevánssá, ha egy diák felhasználó évfolyamot lép vagy pédául ha szakot, esetleg egyetemet vált és frissíteni szeretné az adatlapját az új információkkal. A leadott



2. ábra. Bejelentkezett felhasználó funkcionalitásai



3. ábra. Architektúra: (balról jobbra) adatbázis - szerver - kliens, Illetve a szerver és az e-mail kuldő külső szolgáltatás (Mailgun) közötti kapcsolatok

dolgozatoknál mindig az aktuális adatok jelennek meg.

1.2. Architektúra

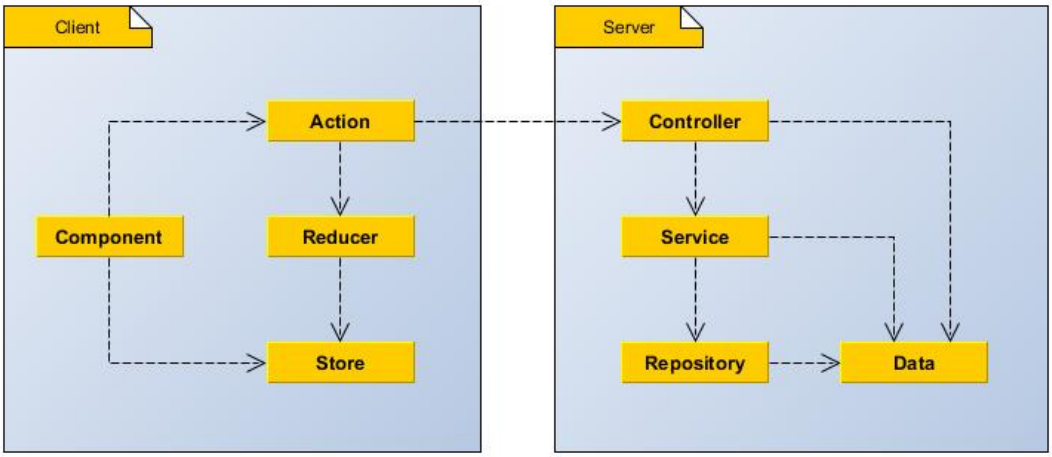
A SConES alkalmazás rendszer két fő részből áll (lásd 3. ábra): a kliens oldalon van egy webalkalmazás, melyet a felhasználó egy böngészőn keresztül tud használni, illetve a háttérben egy szerver, amely a kliens által küldött kérésekre válaszol, adatokat dolgoz fel és tárol. A két komponens között HTTP kommunikáció van. Továbbá a szerver egy külső, Mailgun [15] nevű szolgáltatáson keresztül e-maileket küld ki a felhasználóknak.

1.2.1. Komponensek

A 4. ábra részletesen bemutatja a kliens illetve a szerver főbb komponenseit és ezeknek az egymással való kapcsolatát.

A kliensoldal a tipikusan követett MVC mintákhoz képest egy alternatív megközelítést vonultat fel: a „Component” kapcsolatban áll úgy a „Store”-ral, mint az „Action”-okkal. Ezek a Componentek a weboldal interaktív építő elemei. Az Actionok kommunikálnak a szerverrel és „Reducer”-eken keresztül változtatják a Store állapotát. A Store-ban tárolt adatok alapján a Component változtatja az állapotát.

A szerver oldalán a „Repository” réteg felel az adatbázissal végzett műveletekért. Ezt használja a „Service” réteg, ami az üzleti logikáért felel. Bármiféle adatmanipuláció ezen a szinten történik. Legfelső szinten helyezkedik el a „Controller” réteg. Itt találhatóak a belépési pontok, amiket elér a kliens webalkalmazás. Ez a réteg hívja meg a Service réteg nyújtotta



4. ábra. Komponensek: Kliens oldalon a Component a Store állapotára épül, Action-okkal kommunikál a szerverrel. Az Action-ok változtatják a Storeban az adatokat. Szerver oldalon a Repository és a Data biztosítja az adatelérést, a Service az üzleti logikát, a Controller peddig belépési pontot a kliensnek valamint hozzáférést a Service-ekhez.

szolgáltatásokat. A „Data” tartalmazza a különböző DTO-kat és az adatokat reprezentáló osztályokat. A szerver oldalon minden rétegnek hozzáférése van ehhez a komponenshez.

1.2.2. Kommunikáció

A webkliens és szerver közötti kommunikáció RESTful szolgáltatásokon keresztül valósul meg. A REST (Representational State Transfer), amelyet Roy T. Fielding ismertetett doktori disszertációjában 2000-ben [10], egy architektúrális minta kliens-szerver kommunikáció megvalósítására. RESTful-nak nevezik azokat a szolgáltatásokat, amelyek a minta elveinek eleget tesznek. Ezáltal egyértelműen elérhetővé teszik az adatokat egy egységes felületen keresztül, kliens-szerver alapú kommunikációt valósítanak meg, cache-elési lehetőséget biztosítanak. Valamint jellemzőjük az állapotmentesség, tehát a szerver nem tárol információt a kliensről. Az állapotmentesség hozzájárul a skálázhatósághoz is, ugyanis mivel a szerver nem tárol információt bármely szerver bármely kérést kiszolgálhatja. A RESTful szolgáltatások rétegelést biztosítanak a háttérben, ami azt jelenti, hogy a szerver oldali folyamatok akár mindegyike különböző szerveren mehet végbe. Mindez úgy valósul meg, hogy a felhasználó nem tudja, hogy a folyamat, amelyet elindított éppen hol megy végbe. [30]

A kliens oldalon minden komponens saját könyvtárral rendelkezik. A könyvtárakon belül az action mappákban helyezkednek el azok a fájlok, amelyekben a komponens adatainak feltöltéséhez szükséges `fetch` hívások találhatóak. A szervertől megkapott eredményekkel egy megfelelő típusú action jön létre, amelyre a megfelelő reducer reagál. A reducerek feladata az alkalmazás állapotának frissítése az új információk alapján.

A szerver oldalon használt ASP.NET Core támogatja a web API létrehozását, amely

```
[Produces("application/json")]
[EnableCors("MyPolicy")]
[Route("/api/[controller]")]
[ApiController]
public class ConferencesController : Controller
{
    //...
}
```

1. kódrészlet. Részlet a `ConferenceController` osztályból

a klienstől érkező kérések fogadását és kiszolgálását úgynevezett kontrollerek segítségével valósítja meg. Ennek megfelelően a projekt szerver oldali komponensében egy Controller réteg van kialakítva, amely egy-egy Controllert tartalmaz mindegyik modell számára. Ezen osztályok implementációja, ahogyan az 1. kódrészlet szemlélteti az `EDU.CS.TDK.Controllers` névtéren belül helyezkedik el. Az osztály az `[ApiController]` attribútummal van megjelölve, valamint rendelkezik még egy `[Route()]` attribútummal is, amely a beérkezett kérések megfelelő kontrollerhez való eligazítását teszi lehetővé.

Az adatok JavaScript Object Notation (JSON) [13] objektumok formájában vannak továbbítva, ezt a kontroller osztályok `[Produces(„application/json’’)]` attribútuma biztosítja. A JSON egy adatcserére és tárolásra alkalmas formátum. Emberek számára egyszerűen írható és olvasható kulcs-érték párok formájában tárolja az információt. Minden modern programozási nyelv támogatja, ezért nagyban megkönnyíti két különböző nyelvben írt komponens közötti kommunikációt.

A kért információ a megfelelő HTTP állapotkódokkal együtt küldődik el a kliens számára.

1.2.3. E-mail küldés

A projekt célkitűzései között szerepel a felhasználóval való kapcsolattartás, a felhasználó értesítése bizonyos események bekövetkezésekor. Így például megfelelő e-mail küldése regisztráció esetén vagy a dolgozat sikeres leadásakor.

Az említett funkcionalitások megvalósítása érdekében szükség volt egy e-mail küldő rendszer létrehozására. Ennek kivitelezése a MailGun [15] szerver szolgáltatásainak felhasználásával valósul meg. A SConES projekt esetében a MailGun szerverhez való csatlakozást a `System.Net.Mail` névtér `SmtpClient` osztályának az `appsettings.json` állományban található adatok segítségével történő konfigurálása teszi lehetővé. Az üzenetek tartalmának generálása egyszerű szöveges sablonok segítségével történik. A levél nyelve pedig attól függ, hogy a felhasználó milyen nyelvi beállítással (1.4) használta a webes felületet regisztrációkor. A megfelelő e-mail generálása után a már említett `SmtpClient` osztály `send()`

```html
<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <p>Kedves, {0}</p>
    <p>
      Köszöntünk a Scones-nál.
    </p>
    <p>
      Amennyiben nem te regisztráltál a weboldalunkra,
      hagyd figyelmen kívül ezt az e-mailt.
    </p>
    <p>
      Minden jót, Scones Team
    </p>
  </body>
</html>
```

(a) Magyar nyelvű e-mail sablon regisztrációkor.

```html
<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <p>Dear, {0}</p>
    <p>
      Welcome to Scones.
    </p>
    <p>
      If you did not sign up for a Scones account
      please disregard this email.
    </p>
    <p>
      Happy confering!
    </p>
    <p>
      Best regards, Scones Team
    </p>
  </body>
</html>
```

(b) Angol nyelvű e-mail sablon regisztrációkor.

metódusa küldi el az üzenetet a MailGun szervernek, amely aztán továbbítja a felhasználónak.

1.3. Adatmodell

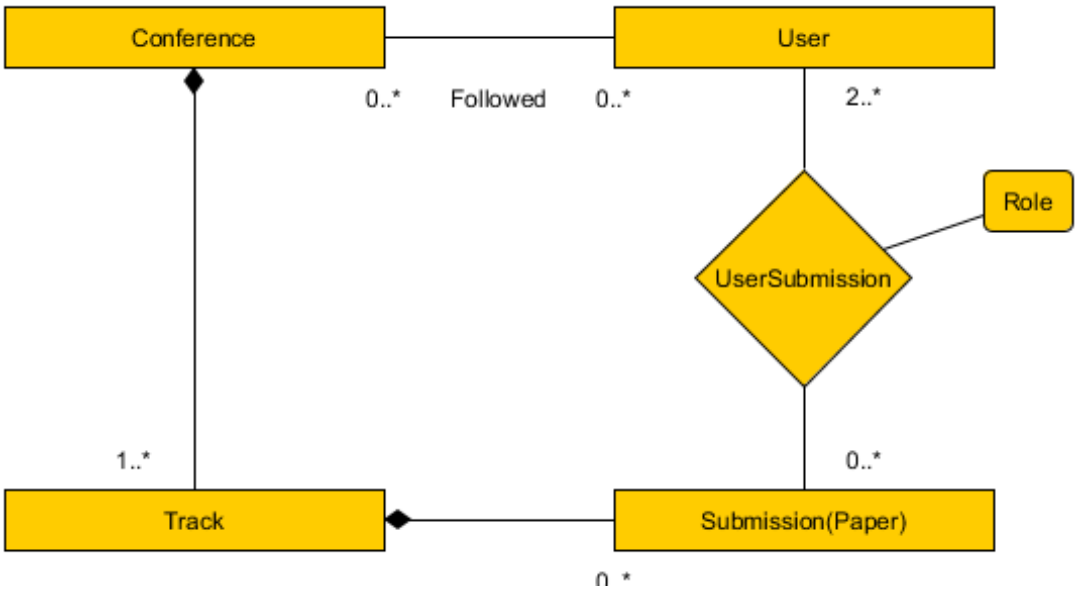
Az adatmodell (a 6 ábra) négy fő táblából és az ezek közötti kapcsolat táblákból áll: Conference, Track, User, Submission.

A Conference táblában vannak tárolva a konferenciák és ezek tulajdonságai, információi. A Track táblában vannak tárolva a „Szekciók” és megkötéseik. A User táblában a „Felhasználók” adatai találhatóak. A Submission táblában vannak eltárolva a leadott dolgozatok.

A Conference és a Track táblák között van egy egy-a-többhöz (one-to-many) kapcsolat. Minden konferenciához tartozik legalább egy szekció, és minden szekció csak egy konferenciához tartozik az egyedi megkötések miatt. Például az idei (fiktív) F-TDK2020-ra a meghirdetett szekciók, ahova dolgozatot lehet leadni: fizika, matematika, számítástechnika. Ezek a tavalyi F-TDK2019 szekcióitól különböznek: fizika és matematikánál változott az elfogadott kiterjesztés, számítástechnikánál a feltöltendő állományok mérete. Ezek a tulajdonságok csak ehhez a konferenciához kötődnek. Ezért nem lehet több-a-többhöz a kapcsolat szekció és konferencia között.

A Track és a Submission táblák között is egy-a-többhöz kapcsolat van, minden szekcióhoz le lehet adni több dolgozatot, viszont egy dolgozatot egy szekcióhoz egy egyedi submission-nal lehet leadni.

A Submission és a User táblák között több-a-többhöz (many-to-many) kapcsolat van. Egy dolgozathoz több felhasználónak is köze van, ugyanis minden dolgozatnak kell legyen legalább egy szerzője és legalább egy vezető tanára. Illetve egy felhasználó több dolgozatot



6. ábra. Adatmodell

adhat le (vagy felügyelhet). Mindez egy köztes kapcsolat táblával („UserSubmission”) van megvalósítva. Itt van eltárolva egy adott felhasználó egy adott dolgozatnál betöltött szerepköre is („Szerző” vagy „Felügyelő tanár”). Például ha egy dolgozatnak két szerzője van, az egyik létrehozza a submission-t, megjelöli a társát mint társszerző és bejelöli a vezető tanárt (a társ és a vezető tanár is regisztrálva kell legyen a SConES rendszerben). Ekkor a UserSubmission köztes tábla bővül három adatsorral, amelyből kettő szerző és egy felügyelő tanár.

A Conference és a User tábla között is egy több-a-többhöz kapcsolat van, mely a konferenciák „bekövetését” („Follow”) hivatott ábrázolni.

1.4. Nemzetköziesítés és lokalizáció

A SConES projekt egyaránt kívánja kiszolgálni az erdélyi magyar és a nemzetközi diákkonferenciákat. Ennek érdekében szükség van az alkalmazás globalizációjára [17]. A globalizáció egy gyűjtőfogalom a nemzetköziesítésre és lokalizációra. A nemzetköziesítés teszi lehetővé, hogy az online felület több nyelven is elérhető legyen. A lokalizáció pedig azért felelős, hogy a nyelvi konvenciók, például a dátum formátum, teljesüljenek. Az SConES online felületének megnyitása után a felhasználónak lehetősége van váltani a rendelkezésére álló nyelvek közül, amelyek jelenleg az angol és a magyar.

A nemzetköziesítés elve abban áll, hogy a nyelv szerint változó üzeneteket különböző erőforrás-állományokban tárolja. Ezeknek köszönhetően a nyelvek közötti váltás nem igényel újrakompilálást. Ennek a mechanizmusnak köszönhetően a SConES projekt egyszerűen bővíthető további nyelvekkel. A megvalósításhoz felhasznált technológia részletezése a 2.2. alfejezetben található.

2. Technológiák

A SConES felület megvalósítását különböző technológiák teszik lehetővé. A jelen fejezet célja ezen technológiák részletes ismertetése a szerver és a kliens oldalon.

2.1. Szerver oldali technológiák

A szerver oldali komponens `C#`-ban íródott ASP.NET Core keretrendszer felhasználásával. Az adatok tárolására a PostgreSQL relációs adatbázis-kezelő rendszer szolgált. A következő alfejezetek bemutatják ezen technológiákat, valamint a szerver és az adatbázis közötti kommunikációt megvalósító Entity Framework Core keretrendszert.

PostgreSQL

A PostgreSQL [25] egy nyílt forráskódú relációs adatbázis-kezelő rendszer. Használja és kiterjeszti az SQL nyelvet, kibővítve azt egyéb olyan funkcionalitásokkal, melyek lehetővé teszik bonyolult adathalmazok biztonságos tárolását és kezelhetőségét.

ASP.NET Core

Az ASP.NET Core [1] egy cross-platform .Net keretrendszer webalkalmazások fejlesztésére. Ingyenes és nyílt forráskódú keretrendszer, amelynek kulcsfontosságú tulajdonsága a teljesítmény. Ezt a független TechEmpower felmérése is igazolja. Az ASP.NET Core 2.2-es verziója képes volt válaszolni 7 milliónál több, egy másodpercen belül küldött HTTP kérésre. [35]

Az ASP.NET Core megengedi moduláris alkalmazások létrehozását, ami azt jelenti, hogy elkerülhető a teljes .NET Framework telepítése. Az alkalmazások megírásához elég csak azoknak a NuGet csomagoknak a hozzáadása, amelyekre szükség van. Ez könnyű fejleszthetőséget és magas teljesítményt eredményez, valamint kevesebb memóriát használ.

Entity Framework Core

Az Entity Framework Core [7] egy pehelykönnyű, bővíthető, nyílt forráskódú, cross-platform, az objektum-relációs leképezés (ORM, Object-relational mapping) technikáját megvalósító keretrendszer. Az ORM megvalósítja az objektum orientált nyelvekben használt objektum modellek megfeleltetését az adatbázis relációs modelljeinek és fordítva. Így a fejlesztők az általában használt adat-hozzáférési kód megírása nélkül C# objektumok segítségével dolgozhatnak az adatbázissal. Az adathozzáférés modellek segítségével valósul meg. Egy modell entitás osztályokból és egy kontextus objektumból áll. Ez utóbbi egy adatbázis

kapcsolatot hivatott reprezentálni. A modell és az adatbázis szinkronizálása három módon valósulhat meg:

- a már létező adatbázis alapján generálódhat a modell
- a fejlesztő kézzel létrehozhat egy modellt, amelynek felépítése egyezik az adatbázis szerkezetével
- az EF Migrations segítségével a megírt modell alapján jöhet létre az adatbázis, folyamatosan igazodva a modell változásaihoz

Az adatokhoz való hozzáférés Language Integrated Query (LINQ) [14] segítségével történik. A LINQ egy átmeneti nyelv, szintaxisa nem függ az adatbázis-kezelő rendszertől. Az általa létrehozott lekérdezések futási időben a megfelelő adatbázisban értelmezhető lekérdezésekké fordulnak. A LINQ a lekért adatokat objektumokká alakítja, így megkönnyítve a fejlesztő számára a további műveletvégzést.

2.2. Kliens oldali technológiák

A SConES projekt kliens oldali komponense egy webalkalmazás, amely a React és Redux keretrendszerek felhasználásával íródott. Az esztétikus és reszponzív megjelenítés kialakítása érdekében a Semantic UI keretrendszer szolgált segítségül. A következő alfejezetek ezen technológiák részletezését tartalmazzák.

React

A React [26] egy nyílt forráskódú JavaScript könyvtár felhasználói felületek létrehozására, amelyet a Facebook fejlesztett ki. A React lehetővé teszi interaktív felhasználói felületek megvalósítását, az alkalmazás minden állapota számára egyszerű kinézet (view) megszerkesztését és hatékonyan frissíti az alkalmazás megfelelő komponenseit az oldal újra töltése nélkül.

A React komponens alapú, ami azt jelenti, hogy egymástól jól elkülöníthető, önállóan működő egységek meghatározására van lehetőség. A komponensek általában az XML szintaxisához hasonló JSX-ben íródnak, ami lehetővé teszi HTML kód használatát a JavaScript kódon belül. Az előfeldolgozó egységeknek köszönhetően a HTML kód a JavaScript számára is értelmezhető objektumokká alakul, aminek következtében a fejlesztőnek lehetősége nyílik a komponensek egymásba ágyazására.

A React komponensek saját állapottal rendelkeznek és azt menedzselik. Ahhoz, hogy egymás állapotát befolyásolni tudják az állapotot egy olyan szülő komponensbe kell helyezni, amelynek mindkét komponens leszármazottja. Kis méretű alkalmazások esetében ez a fajta

```
class RegisterPage extends Component {
  constructor(props) {
    super(props);
  }
  render() {
    return (
      //...
    );
  }
}
```

2. kódrészlet. Részlet a regisztrációs oldal komponséből.

állapot menedzsment jól tud működni, azonban ahogy az alkalmazás mérete nő úgy lesz egyre bonyolultabb.

A SConES projekt kliens oldala egy sok komponensből álló React alkalmazás. Megtörténhet, hogy egy állapotot két olyan komponens között kell megosztani, amelyek nagyon messze vannak egymástól a komponens fában. Az állapotok elhelyezésének folyamatos változtatása bonyolulttá és nehezen értelmezhetővé teheti a kódot. Ennek elkerülése érdekében szükség van egy állapotmenedzsment eszköz használatára, amely a SConES projekt esetében a Redux.

Redux

A Redux [29] egy nyílt forráskódú JavaScript könyvtár, amelynek fő feladata az alkalmazás állapotának menedzselése. Dan Abramov fejlesztette ki 2015-ben. Általában a React vagy Angular könyvtárakkal együtt használják felhasználói felületek megalkotására.

A Redux bevezetése a SConES projekt React kliensébe megoldotta a 2.2. alfejezetben leírt állapotmegosztási problémát. A Redux ugyanis egy központi store-ban tárolja az alkalmazás teljes állapotát. Ehhez a store-hoz minden komponens hozzáfér és használhatja a számára szükséges részt.

A store-ban tárolt állapot megváltoztatható bizonyos események hatására. Az események (action) különböző típusúak lehetnek és adattagokkal rendelkezhetnek. Adott típusú esemény kiváltása adott felhasználói művelet hatására történik meg. A reducer-ek felismerik a különböző típusú eseményeket, értelmezik azokat és a megfelelő módon módosítják a kezdeti állapotot az action által továbbított adattag segítségével. Az állapot módosítása maga után vonja a felhasználói felület módosulását is.

```
{
   "register": {
       "lastName": "Vezetéknév",
       "firstName": "Keresztnév",
       "email": "E-mail",
       "password": "Jelszó",
       "submit": "Regisztráció"
    }
}
```

3. kódrészlet. Részlet a magyar nyelvű szövegek tárolására szolgáló állományból.

```
<Form.Field required>
   <label>
      {this.props.t("register.firstName")}
   </label>
   <input
      name="firstName"
   />
</Form.Field>
```

4. kódrészlet. Részlet a regisztrációs űrlapot megvalósító kódból, ahol jól látszik az i18n használata.

Globalizáció

A globalizáció megvalósítását a `react-i18next` [27] csomag teszi lehetővé. A lefordítandó kifejezések kulcs-érték párosok formájában vannak megadva a `translation.json` fájlokban, amelyek a /public/locales mappán belül minden választható nyelv könyvtárában megtalálhatóak. A felhasználó által aktuálisan kiválasztott nyelv a localstorage-ba kerül, ahonnan az i18n `changeLanguage()` metódusa segítségével betöltődik a megfelelő nyelvet leíró JSON fájl. Ebből az állományból futási időben az i18n fordító függvénye visszatéríti a megadott kulcsnak megfelelő értéket.

A különböző nyelvek fordítását meghatározó állományokhoz hasonlóan, minden nyelv könyvtárában jelen van egy `localeFormats.json` fájl, amely az adott nyelvre vonatkozó lokalizációs formátumokat tárolja, többek között a dátum formátumának kulcs-érték páros általi megadását. Ezeknek az adatoknak a betöltése szintén az i18n fordító függvénye segítségével valósul meg.

A 3. kódrészletben a regisztációs űrlap megvalósításánál használt kifejezések magyar megfelelői láthatóak kulcs-érték párosokban. A 4. kódrészlet pedig szemlélteti az i18n felhasználásával a megfelelő nyelvű szöveg beállítását.

Semantic UI

A Semantic UI [34] egy olyan fejlesztői keretrendszer, amely lehetővé teszi esztétikus, reszponzív elrendezések létrehozását a felhasználói felületen, barátságos és könnyen használható HTML segítségével. A keretrendszer előre felépített szemantikus komponenseket tartalmaz, tömör HTML-t és intuitív JavaScript-et használ, azért, hogy a front-end fejlesztést megkönnyítse és egy élvezhető tevékenységgé formálja.

A Semantic UI keretrendszer több keretrendszerbe is integrálható, mint például a React, Angular, Ember vagy Meteor, hogy elősegítse a felhasználói felület kinézetének elrendezését

```
<Card>
  <Card.Content>
     <Card.Header>
     //...
        <FavoriteButton/>
     //...
```

5. kódrészlet. Részlet a konferenciák megjelítéséről a Semantic UI Card használatával.

7. ábra. A kód által eredményezett kártya.

az alkalmazás logikájával összhangban.

A SConES projekt megvalósítása során kézenfekvő volt a Semnatic UI használata. A konferenciák és a leadott dolgozatok listázásánál is hasznosak voltak a keretrendszer által biztosított kártyák. Segítségükkel a listák elemei egységesek lettek és további elemek hozzáadása is egyszerűvé vált. A kártyák használatát az 5. kódrészlet és a 7. ábra szemlélteti.

3. Alkalmazott módszerek és felhasznált eszközök

Hogy a fejlesztés során megfelelő ütemű előrehaladás valósulhasson meg, elengedhetetlen a munkafolyamat megtervezése, a gyakori egyeztetés, valamint a megvalósításokra való visszatekintés. Ugyanakkor folyamatos hangsúlyt kell fektetni a kód minőségére, hogy minél egységesebb, átláthatóbb végeredményt kapjunk. A következő részben azok a módszerek és eszközök kerülnek megemlítésre, amelyek nagyban hozzájárultak a fent említettekhez.

3.1. Scrum munkamódszer

A SConES projekt megvalósítása a Scrum [5] szoftverfejlesztési stratégia alkalmazásával történt, amely iteratív és inkrementális tulajdonságokkal rendelkező módszer. A módszeren belüli iterációkat sprinteknek nevezik, melyeket minden esetben megelőz egy tervezési fázis, amelynek során a fejlesztők megbecsülik a funkcionalitásokat takaró user storykra szánt időt, valamint eldöntik, hogy az aktuális sprintben mely story-kat kívánják megvalósítani. A sprinteket egy az újonnan létrejött funkcionalitásokat szemléltető bemutató (demo) követi, valamint egy visszatekintő kiértékelésre is sor kerül.

Ezek alapján a nyári gyakorlat során kéthetes sprintekkel dolgozott a projekt fejlesztői csapata, mindennapos megbeszélésekkel. Majd a csoportos projekt tantárgy keretein belül a sprintek három hét hosszúságúra nőttek, valamint a megbeszélések száma heti egyre csökkent.

3.2. Docker

A modern alkalmazások a kézi kitelepítés hátrányainak kiküszöbölése végett operációs rendszer szintű virtualizációt használnak. Ennek köszönhetően elkerülhető az operációs rendszer inkompatibilitás, foglalt portok jelenléte, tűzfal nem megfelelő beállításai, eszközök verzióinak eltérése és még sorolható. Virtualizáció során az alkalmazások konténerekben futnak. A konténerek hasonlóak a virtuális gépekhez, de a hardver mellett az operációsrendszert is virtualizálják, így nagyon pehelysúlyúak és egy absztrakciós szintet valósítanak meg az alkalmazás és annak minden függősége fölött.

A Docker [18] egy a fent leírt virtualizáció megvalósítására alkalmas eszköz, amely 2013-ban jelent meg. A Docker esetében a konténerek az úgynevezett docker image-ekből hozhatóak létre, amelyek egy virtuális gép lefagyasztott állapotát reprezentálják. Az image-ek készítése Dockerfile alapján történik, amelyben kötelezően szerepelnie kell a kiinduló forrás image-nek (FROM). Opcionálisan jelen lehet még a dolgozó könyvtár beállítása (WORKDIR), a megfelelő állományok átmásolása a lokális gépről az image-re (COPY), valamint a konténer indító parancsa (CMD).

```
#a kiinduló forrás image megadása, amely az új image alapjául szolgál
FROM mcr.microsoft.com/dotnet/core/sdk:3.1
#a dolgozó könyvtár beállítása
WORKDIR /app

#a függőségek beállításához szükséges állományok másolása
#a lokális gépről az image-re
COPY *.csproj ./
#a függőségek beállítása az átmásolt állományok alapján
RUN dotnet restore

#további szükséges állományok másolása a lokális gépről az image-re
COPY . ./
#az applikáció felépítése (build) a Release konfigurációk alapján
RUN dotnet publish -c Release -o out
#új migrációs állomány létrehozása a kódbeli modellek alapján
RUN dotnet ef migrations add latest
#az adatbázis frissítése a létrehozott migrációs
#állomány alapján és a program futtatása
CMD dotnet ef database update && dotnet out/backend.dll
```

6. kódrészlet. Szerver oldali Dockerfile

A SConES projekt esetében a szerver oldali komponens és a kliens oldali komponens is külön Dockerfile-al rendelkezik, a szerver oldalon megtalálható Dockerfile tartalma az 6. kódrészletben látható.

3.3. Verziókövetés, CI/CD

A hatékony csapatmunka érdekében a projekt fejlesztése során szükség volt egy verziókövető rendszerre, amely a Git volt. A Git [11] egy osztott verziókövető rendszer, lehetővé teszi az egyidejű fejlesztést, a kódbázisban történő változások számontartását, valamint korábbi verzióra való visszatérést, ha erre szükség van, amivel nagyban elősegíti és megkönnyíti a fejlesztési folyamatot.

A Git Flow [4] megközelítés értelmében minden funkcionalitás fejlesztése számára egy-egy különálló ág áll rendelkezésre, amely csak egy alapos felülvizsgálat után kerül be a stabil változatot tartalmazó ágra. Az ágak, feladatok nyilvántartásának és követésének megkönnyítése érdekében szükség volt a GitLab [12] projektmenedzsment eszközre. A GitLab a szoftver teljes életciklusát lefedő fejlesztői eszközöket biztosít. A SConES projekt fejlesztői csapata ennek segítségével tárolta a forráskódot, használta a beépített KanBan board-ot, melynek segítségével látványosan követhetővé vált a user storyk állapota és naplózhatóvá vált az egyes funkcionalitások fejlesztésével eltöltött idő.

Ezen kívül a GitLab által biztosított pipeline rendszer segítségével valósul meg a

```yaml
services:
  dk-reverse-proxy:
    image: r.edu.codespring.ro/dk/dk-compose
    links:
      - dk-backend:dk-backend
      - dk-web:dk-web
    # ...

  dk-postgres:
    image: postgres:11.5-alpine
    # ...


  dk-backend:
    image: r.edu.codespring.ro/dk/dk-backend
    links:
      - dk-postgres:dk-postgres
    # ...

  dk-web:
    image: r.edu.codespring.ro/dk/dk-frontend
    # ...
```

7. kódrészlet. Részlet a docker-compose.yml állományból

Continuous Integration (CI), vagyis a folyamatos integráció is, amely által elkerülhető, hogy az újonnan beépített funkcionalitások elrontsák a már meglévő, helyesen működő alkalmazást. A pipeline-ok fázisokba (stage) rendezett munkákból (job) állnak, amelyek a .gitlab-ci.yml állományban vannak konfigurálva. Minden push esetén elindul egy pipeline és sorrendben végrehajtódnak a megfelelő fázisok. A SConES projekt esetében a CI pipeline két vagy három fázisból áll, a statikus kódellenőrzésből, a kód lefordításából, valamint a master ágra történő push esetében még egy a kitelepítéshez szükséges Docker image-et létrehozó fázis is jelen van. Az elkészült image bekerül egy privát tárolóba, a GitLab Container Registry-be.

A kitelepítés megvalósítása egy a szerver és kliens oldal tárolójától különálló dk-compose nevű tároló által valósul meg. A kitelepítést elősegíti a docker-compose [3], amely lehetővé teszi több konténerből álló alkalmazások futtatását. A manuálisan végrehajtható kitelepítés során a docker-compose létrehoz egy belső hálózatot az alkalmazásszerver, a webszerver és az adatbázis számára és megnyit egy portot a külvilág felé, amelyen keresztül elérhető az alkalmazás. A 7. kódrészletben is látható reverse-proxy, egy módszer arra, hogy egyetlen domain név alatt legyen elérhető az alkalmazás. A hívások megfelelő továbbítását az alkalmazásszerver vagy a webszerver felé egy NGINX [19] szerver valósítja meg.

3.4. Statikus kódelemzés

A fejlesztés során a kód átláthatóságának és minőségének megőrzése érdekében fontos az előre meghatározott konvenciókhoz való igazodás. Ennek megkönnyítésére szolgálnak a statikus kódellenőrző eszközök, melyek az adott nyelvre vonatkozó specifikus szabályok szerint elemzik a kód minőségét.

A kliens oldali JavaScript forráskód ellenőrzésére az ESlint [9] eszközt alkalmazza a projekt, míg a szerver oldali C# kód megfelelő minőségét a StyleCop [23] biztosítja. Mindkét kódellenőrző konfigurálható saját igényeink szerint, valamint a figyelmeztetés módját, erősségét is személyre szabhatjuk. A kód ellenőrzése mindkét komponens esetén a CI pipeline-ban megtörténik, így minden push után visszajelzés érkezik a kód minőségéről.

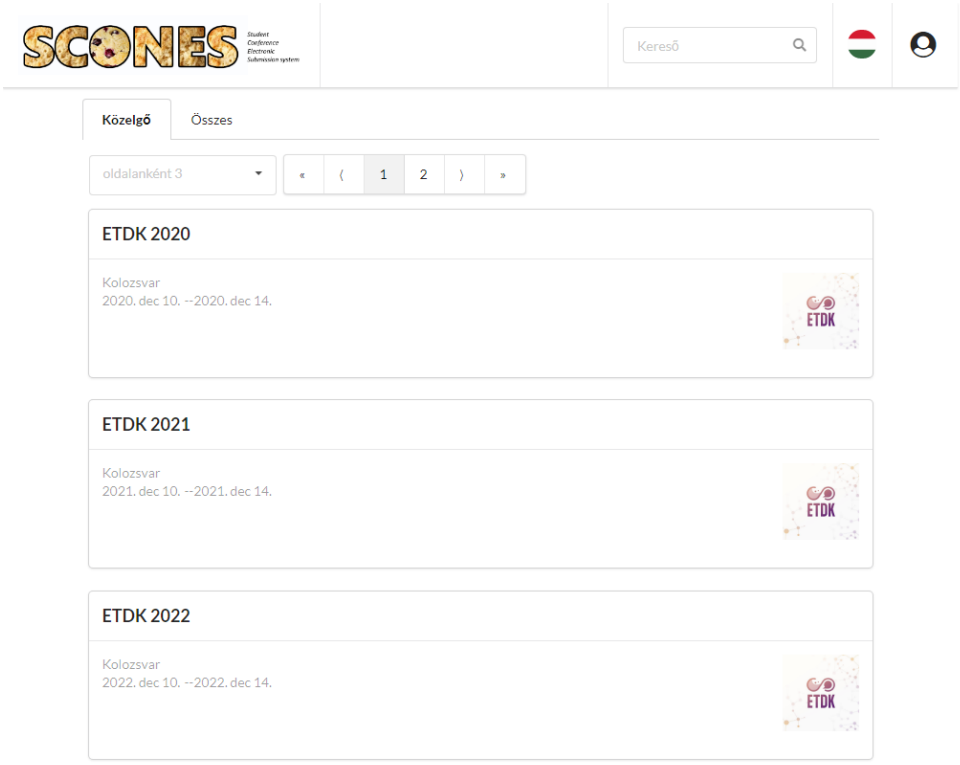
4. Az alkalmazás működése

A jelen fejezetben részletesen be van mutatva, hogy hogyan lehet használni a weboldalt.

4.1. Főoldal

A SConES-nak főoldalaként a „Közelgő” („Upcoming”) konferenciák kilistázása szolgál (a 8 ábra). Ezt az oldalt a felhasználó többféleképpen is elérheti, ez fogadja a felhasználót amikor a weboldalra navigál, vagy amikor a weboldalon a SConES logo-ra kattint. A középpontban a konferenciák állnak, egymás alatt. Átláthatóság érdekében limitálva van az, hogy egyszerre hány konferencia jelenjen meg. Az alapértelmezett beállítás a három konferencia oldalanként, de a felhasználó beállíthat oldalanként ötöt is egy lenyíló menüből, amely a lista fölött helyezkedik el. Ugyanitt helyezkedik el egy panel amellyel lehet navigálni a konferenciák között. E fölött található két fül, melyekkel lehet változtatni a listázást a „Közelgő” illetve az „Összes” („All”) konferenciák között.

A konferenciákhoz fel van tüntetve pár alapinformáció:név, helyszín, időintervallum (mikor kerül megszervezésre) illetve egy plakát.



8. ábra. A SConES főoldala: közelgő konferenciák kilistázva, oldalanként három konferencia van megjelenítve, két oldalra van osztva

4.2. Lebegő sáv

A weboldal legtetején található egy rögzített, lebegő sáv, amelyen bal oldalt fel van tüntetve a SConES logo, illetve, jobb oldalt, pár hasznos eszköz. Itt található egy kereső, mely a konferenciák szűrését hivatott szolgálni, egy legördülő menü a nyelv váltásra: a weboldal magyarul és angolul használható (minden ami nem konferencia/dolgozat specifikus le van fordítva magyarra meg angolra, e mellett a dátum is a kiválasztott nyelv konvenciói szerint van formázva), valamint egy profil ikon, melynek funkcionalitása attól függ, hogy be van-e jelentkezve valaki a weboldalra vagy sem.

4.3. Profil icon/menü

Az korábban említett lebegő sávban található profil icon egy lenyíló lebegő menühöz nyújt hozzáférést. Amennyiben nincs valaki már bejelentkezve, e-mail jelszó párossal ezt megteheti. Ugyanitt található egy regisztrációs gomb (link), felhasználó fiók létrehozására. Az ehhez szükséges adatok: keresztnév, vezetéknév, intézmény ahol tanul/dolgozik az illető, részletes információ az utóbbihoz (kar, szak, év), e-mail cím és jelszó. A felhasználó jelszaván kívül, az információk megjelenhetnek az oldalon különböző helyeken, vagyis elérhetőek más felhasználók számára.

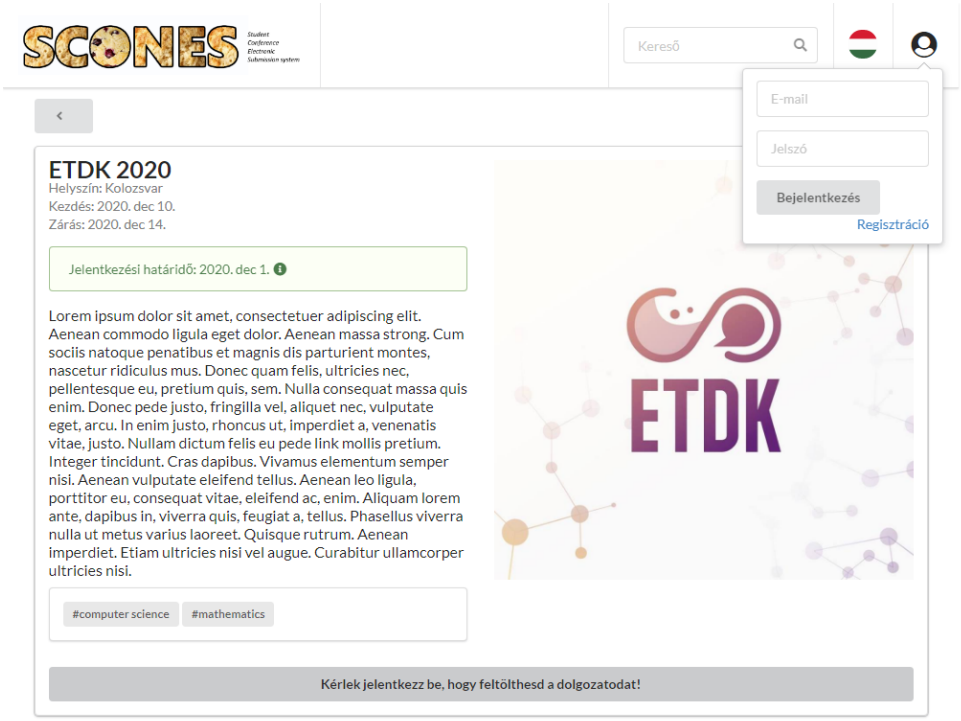
Egy bejelentkezett felhasználó a lenyíló menüben hozzáfér az általa leadott dolgozatokhoz, navigálhat a „Közelgő konferenciák”-hoz (főoldal), a „Profilom” menüponton keresztül megtekintheti valamint szerkesztheti személyes adatait, és ki tud jelentkezni.

4.4. Konferencia részletei

Amint a felhasználó rákattint egy konferencia kártyájára, a weboldal átnavigál az adott konferencia profil oldalára (a 9 ábra), ahol minden tudnivaló megjelenik. Az alapinformációk itt bővülnek a szekciók (témakörök, mellyel foglalkozik a konferencia) megnevezésével, a leadási határidőkkel, mely szekciónként változhat, egy részletes leírással a konferenciáról, valamint egy gombbal, melynek szerepe változó.

Ha a felhasználó nincs bejelentkezve, az említett gomb lenyitja a bejelentkező űrlapot. Ha be van jelentkezve a felhasználó és a konferenciának már lejárt a jelentkezési határideje, a gomb nem elérhető és magyarázatként fel van tüntetve, hogy „Sajnos a jelentkezési határidő lejárt”. Abban az esetben, ha a határidő még nem járt le, a gomb átnavigál a dolgozat feltöltését megvalósító űrlaphoz.

A bejelentkezett felhasználónak, amennyiben már van feltöltve, egy vagy több, dolgozata az adott konferenciához, „Dolgozataim” („My Submissions”) címszó alatt, ezek listázva vannak: címmel, szekcióval, leadás dátumával és idejével, valamint a dolgozat állapotával.



9. ábra. Nem bejelentkezett felhasználó így látja az ETDK2020 nevű konferencia profilját. Jobb felső sarokban, le van nyitva a bejelentkező űrlap.

4.5. Dolgozat feltöltése

A dolgozat feltöltése (a 10 ábra), egy űrlap kitöltésével kezdődik. Az űrlap automatikusan kitöltődik a felhasználó vezeték- és keresztnevével.

Ezután megjelölhet egy vagy több társszerzőt a regisztrált felhasználok közül. A „Társszerző(k)” („Co-Author(s)”) alatti cellában, három karakter után, a weboldal ajánlatokat ad a beírt részlet alapján, melyek közül lehet kiválasztani a társszerzőt. A rendszer név és e-mail címben is keresi a megadott részletet. Társszerző megjelölése opcionális, míg az ezután következő adatokat kötelező megadni.

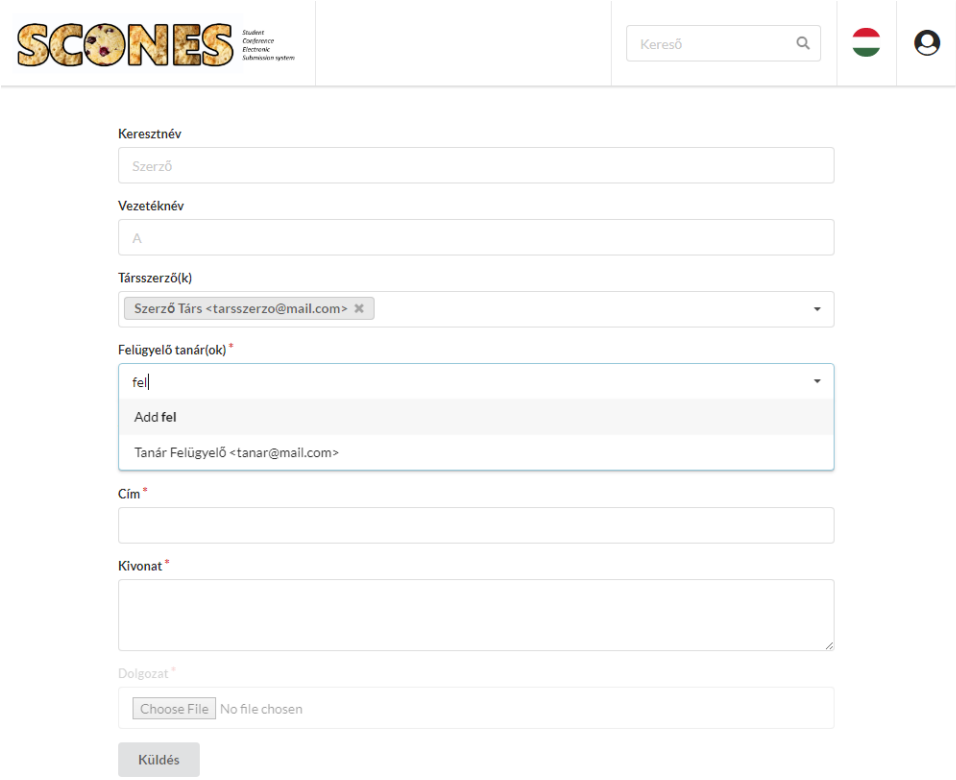
„Felügyelő tanár(ok)” („Supervisor(s)”) alatt, az előbb említett módszerrel, meg kell jelölni legalább egy felhasználót, aki a vezető tanár szerepet tölti be a dolgozatnál.

Ezután kell kiválasztani a „Szekció”-t („Track”), amelyhez a dolgozatot szeretné leadni a felhasználó. Egy legördülő menüben jelennek meg azok a szekciók, melyeknek még nem járt le a leadási határideje. A kiválasztott szekció meghatározza későbbiekben a „Kivonat” („Abstrakt”) hosszát és a feltöltendő állomány kiterjesztését valamint maximális méretét.

Következőben a szerző megadja a dolgozat „Cím”-ét („Title”) majd a kivonatot.

„Dolgozat” („Paper”) címszó alatt, megjelöli a szerző a saját eszközéről a feltöltendő dolgozatot.

Végül a „Küldés” („Submit”) gombra kattintva, ha minden rendben, feltöltheti a dolgozatot.

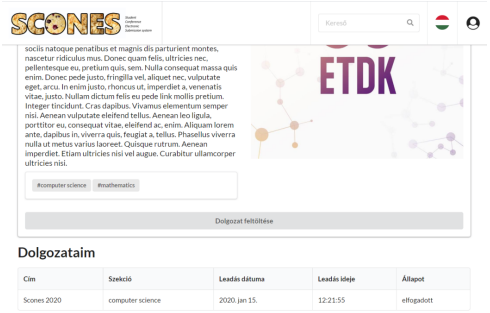


10. ábra. Dolgozat leadási űrlapja

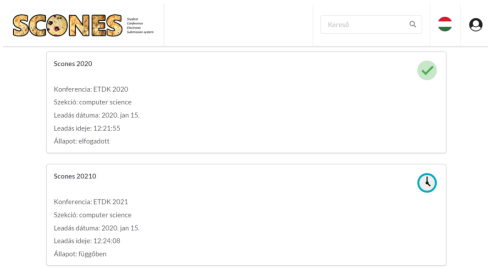
4.6. Leadott dolgozatok megtekintése

A megfelelő konferencia profilja alatt, a felhasználó megtekintheti azoknak a dolgozatoknak az állapotát, amelyeknél szerzőként (vagy társszerzőként) szerepel (a 12(a) ábra). A profil menüből (jobb felső sarok) azonban, a „Dolgozataim”-on („My submissions”) keresztül, megtekintheti az összes dolgozatot kilistázva, aminél szerzőként szerepel (a 12(b) ábra). A felhasználó a listában láthatja, többek között, a dolgozatok címét, melyik konferenciához, azon belül melyik szekcióhoz volt leadva és az állapotot amiben a dolgozat épp van. A dolgozat állapota a Következő négy egyike lehet: „Függőben” - ez az első állapot amibe kerül feltöltés után, várja a vezető tanár jóváhagyását; „Jóváhagyott” - a vezető tanár által jóváhagyott dolgozat az elbíráló bizottság kezébe került; „Elfogadott” - a bíráló bizottság elfogadta a dolgozatot; „Elutasított” - a bíráló bizottság elutasította a dolgozatot.

Amennyiben a felhasználó rákattint a listán egy elemre, a weboldal átnavigál a dolgozat részletes oldalára (a 12 ábra). Itt az előbb megjelenített részletek mellett információ látható a többi szerzőről illetve a felügyelő tanárról. Található egy link annak a konferenciának a profil oldalára, ahova a dolgozat le volt adva, valamint a „Töltsd le a dolgozatodat innen:” („Download your paper from here:”) felirat melletti ikonra kattintva, le lehet tölteni a dolgozatot.



(a) Konferencia profil oldala alatt megjelenő Dolgozataim kilistázva. Első jóváhagyott állapotban, második függőben.



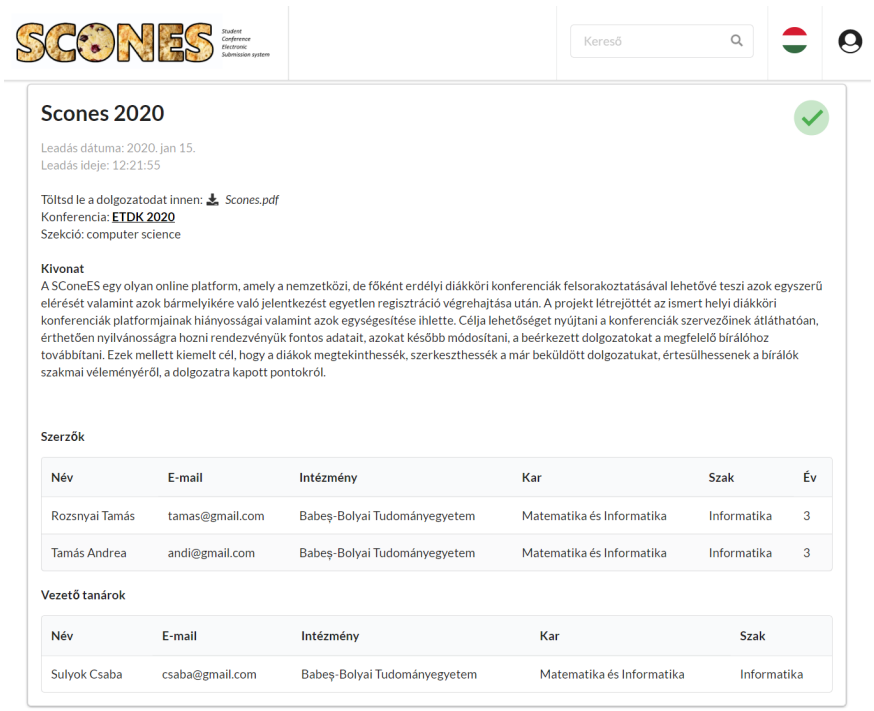
(b) Dolgozataim kilistázva. Első jóváhagyott

4.7. Hiba-, Siker- Jelzések

Sikeres regisztráció vagy dolgozat leadás után a weboldalon középen fent, megjelenik zölddel egy sikeres (pozitív) üzenet. Pár másodperc után ez eltűnik.

Kitöltendő űrlapok esetén (bejelentkezés, regisztrálás, dolgozat feltöltése), ha valamelyik információ nem felel meg a kérelmeknek, az adott cella (általában instant) piros színűvé változik, ezzel jelezve a hibát. Pl. e-mail cím esetében - nem e-mail cím formájú a beírt adat, jelszó esetében - nem elég hosszú a jelszó, esetleg hiányzik vagy nem megfelelő (dolgozat leadásnál pl. Az állomány formátuma) valamely kötelezően megadandó adat. Utóbbit leadáskor (submit) jelzi a weboldal.

Ha valamilyen hiba lép fel a háttérben (back-end), a weboldalon a hibaüzenet (akár a siker üzenet) fent középen jelenik meg, annyi különbséggel, hogy ez pirossal lesz bekeretezve. Pár



12. ábra. Leadott dolgozat részletei

másodperc után a hibaüzenet eltűnik. Hibaüzenet jelenhet meg pl. Regisztrációnál - ha az e-mail cím már regisztrálva van; bejelentkezésnél - ha a megadott e-mail cím vagy jelszó páros nem helyes; dolgozat leadásánál - ha a dolgozat már le volt egyszer adva; vagy ha bármilyen más hiba történik a háttérben.

5. Következtetések és továbbfejlesztési lehetőségek

A magyar tudományos diákköri tevékenységet folytató egyetemi hallgatók számára nem egyszerű az anyaországi és határon túli intézményi TDK-k követése. Az online rendszerekbeli különbségek vagy azok teljes hiánya megnehezíti a jelentkezés folyamatát. A jelen dolgozatban bemutatott SConES projekt ezen konferenciák egységesítését tűzte ki célul. A fejlesztés során sikerült létrehozni egy olyan online felületet, amely hiánypótló lehet a magyar vagy akár nemzetközi TDK-k menedzselésében. A SConES felület használatával a diákok számára könnyebbé válhat a konferenciák megtalálása, böngészése és ennek következtében a szervezők is könnyebben elérhetik célközönségüket. Az online rendszer használatával leegyszerűsödhet a dolgozatok leadása és a kommunikáció is.

Az eddig megvalósított funkcionalitások a regisztráció, jelentkezés, leadott dolgozat megtekintése, személyes adatok szerkesztése mellett több olyan funkcionalitás megvalósítása van tervben, amelyek még eredetibbé, testre szabhatóbbá tehetik a konferenciaszervezést.

A fejlesztés során felmerült továbbfejlesztési lehetőségek között szerepel a konferenciák keresése több különböző kritérium alapján, szekciók szerint, dátum szerint, név szerint, megrendezési hely szerint. A szervezőknek engedélyezni a konferencia létrehozását, annak testre szabását, diákoknak lehetőséget nyújtani a már beküldött dolgozat kivonatának szerkesztésére, új dokumentum feltöltésére az adott konferencia szabályzatának megfelelően vagy az esetleges visszavonásra. A vezető tanár jóváhagyását elektronizálni, a feltöltött dolgozatokat automatikusan tovább küldeni a kijelölt elbírálóknak, a dolgozatokat kommentálni, pontozni.

Hivatkozások

[1] *ASP.NET Core hivatalos dokumentáció*. URL: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core (elérés dátuma: 2020. ápr. 16.)

[2] Szendrő Péter Cziráki Szabina. *A TDK szerepe, szervezeti keretei és jövője a felsőoktatási tehetséggondozásban*. URL: https://www.felvi.hu/pub_bin/dload/felsooktatasimuhely/FeMu/2012_4/Felsooktatasi_Muhely_2012_4_73-54.pdf (elérés dátuma: 2020. ápr. 28.)

[3] *Docker compose hivatalos dokumentáció*. URL: https://docs.docker.com/compose/ (elérés dátuma: 2020. ápr. 22.)

[4] Vincent Driessen. *A successful Git branching model*. URL: https://nvie.com/posts/a-successful-git-branching-model/ (elérés dátuma: 2020. ápr. 28.)

[5] Claire Drumond. *Scrum - what it is, how it works, and why it's awesome*. URL: https://www.atlassian.com/agile/scrum (elérés dátuma: 2020. ápr. 22.)

[6] *EasyChair weboldal*. URL: https://easychair.org/ (elérés dátuma: 2020. ápr. 21.)

[7] *Entity Framework Core hivatalos dokumentáció*. URL: https://docs.microsoft.com/en-us/ef/core/ (elérés dátuma: 2020. ápr. 16.)

[8] *Erdélyi Műszaki Tudományos Diákköri Konferencia hivatalos weboldal*. URL: https://mtdk.tmd.ro/index.php/site/page/ (elérés dátuma: 2020. ápr. 28.)

[9] *ESLint hivatalos dokumentáció*. URL: https://eslint.org/ (elérés dátuma: 2020. ápr. 22.)

[10] Roy T Fielding és Richard N Taylor. *Architectural styles and the design of network-based software architectures*. 7. kötet. University of California, Irvine Irvine, 2000.

[11] *Git hivatalos dokumentáció*. URL: https://git-scm.com/ (elérés dátuma: 2020. ápr. 21.)

[12] *GitLab hivatalos dokumentáció*. URL: https://about.gitlab.com/what-is-gitlab/ (elérés dátuma: 2020. ápr. 21.)

[13] *JavaScript Object Notation hivatalos dokumentáció*. URL: https://www.json.org/json-en.html (elérés dátuma: 2020. ápr. 29.)

[14] *Language Integrated Query hivatalos dokumentáció*. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ (elérés dátuma: 2020. ápr. 29.)

[15] *Mailgun weboldal*. URL: https://www.mailgun.com (elérés dátuma: 2020. máj. 3.)

[16] *Marosvásárhelyen megrendezett Tudományos Diákköri Konferencia hivatalos weboldal*. URL: https://tdk.mmdsz.ro/ (elérés dátuma: 2020. ápr. 28.)

[17] Kenneth A Mckethan és Graciela White. „Demystifying software globalization". *Translation Journal* 9.2 (2005), 1–8. oldal.

[18] Dirk Merkel. „Docker: lightweight linux containers for consistent development and deployment”. *Linux journal* 2014.239 (2014), 2. oldal.

[19] *NGINX hivatalos dokumentáció*. URL: https://www.nginx.com/resources/glossary/nginx/ (elérés dátuma: 2020. ápr. 29.)

[20] *Országos Tudományos Diákköri Konferencia hivatalos weboldala*. URL: http://otdk.hu/ (elérés dátuma: 2020. ápr. 28.)

[21] *Országos Tudományos Diákköri Tanács hivatalos weboldala*. URL: http://otdt.hu/ (elérés dátuma: 2020. ápr. 28.)

[22] *Országos Tudományos Diákköri Tanács online adatkezelő rendszere*. URL: https://online.otdk.hu/ (elérés dátuma: 2020. ápr. 28.)

[23] Michael Parker. *Linting C# in 2019 – StyleCop, Sonar, Resharper, Visual Studio and Roslyn*. URL: https://medium.com/@michaelparkerdev/linting-c-in-2019-stylecop-sonar-resharper-and-roslyn-73e88af57ebd (elérés dátuma: 2020. ápr. 22.)

[24] *Partiumi Tudományos Diákköri Konferencia hivatalos weboldala*. URL: https://ptdk.partium.ro/hu (elérés dátuma: 2020. ápr. 28.)

[25] *PostgreSQL hivatalos dokumentáció*. URL: https://www.postgresql.org/about/ (elérés dátuma: 2020. ápr. 15.)

[26] *React hivatalos dokumentáció*. URL: https://reactjs.org/ (elérés dátuma: 2020. ápr. 19.)

[27] *React i18next hivatalos dokumentáció*. URL: https://react.i18next.com/ (elérés dátuma: 2020. ápr. 22.)

[28] *Reál és Humántudományi Erdélyi Tudományos Diákköri Konferencia hivatalos weboldal*. URL: https://etdk.kmdsz.ro/ (elérés dátuma: 2020. ápr. 28.)

[29] *Redux hivatalos dokumentáció*. URL: https://redux.js.org/ (elérés dátuma: 2020. ápr. 21.)

[30] *REST API hivatalos dokumentáció*. URL: https://restfulapi.net/ (elérés dátuma: 2020. ápr. 29.)

[31] *Sapientia Erdélyi Magyar Tudományegyetem csíkszeredai karának Kari Diákköri Konferencia hivatalos weboldala*. URL: https://tdk.cs.sapientia.ro/index.php/site/page/ (elérés dátuma: 2020. ápr. 28.)

[32] *Sapientia Erdélyi Magyar Tudományegyetem kolozsvári karának Kari Diákköri Konferencia hivatalos weboldala*. URL: https://tdk.kv.sapientia.ro/index.php/site/page/ (elérés dátuma: 2020. ápr. 28.)

[33] *Sapientia Erdélyi Magyar Tudományegyetem marosvásárhelyi karának Kari Diákköri Konferencia hivatalos weboldala*. URL: https://tdk.ms.sapientia.ro/index.php/site/page/ (elérés dátuma: 2020. ápr. 28.)

[34] *Semantic UI hivatalos dokumentáció*. URL: https://semantic-ui.com/ (elérés dátuma: 2020. ápr. 21.)

[35] *TechEmpower hivatalos weboldala*. URL: https://www.techempower.com/benchmarks/#section=data-r16&hw=ph&test=plaintext (elérés dátuma: 2020. ápr. 28.)

[36] *Vajdasági Magyar Tudományos Diákköri Konferencia hivatalos oldala*. URL: http://vmtdk.edu.rs/ (elérés dátuma: 2020. ápr. 28.)

[37] *XIV. Kárpátaljai Tudományos Diákköri Konferencia hivatalos meghirdetése*. URL: http://kmf.uz.ua/hu/xiv-karpataljai-tudomanyos-diakkori-konferencia-2020-majus-15-beregszasz/ (elérés dátuma: 2020. ápr. 28.)