Eötvös Lóránd Tudományegyetem

Informatika Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Tanulási keretrendszer a C++ programozáshoz

Témavezető:

**Pataki Norbert**

Adjunktus, PhD

Szerző:

**Török Richárd Dávid**

Programtervező informatikus, Bsc

Budapest, 2021

Tartalom

[1. Bevezetés 3](#_Toc58261693)

[1.1 Témaválasztás indoka 3](#_Toc58261694)

[1.2 Megoldandó feladat 3](#_Toc58261695)

[2. Felhasználói dokumentáció 4](#_Toc58261696)

[2.1 Rendszerkövetelmények 4](#_Toc58261697)

[2.2 Telepítés 5](#_Toc58261698)

[2.2.1 Első indítás 5](#_Toc58261699)

[2.2.2 Már inicializált adatbázissal 7](#_Toc58261700)

[2.3 Content management system felhasználói felület 8](#_Toc58261701)

[2.3.1 Bejelentkezési képernyő 8](#_Toc58261702)

[2.3.2 Kezdő képernyő 9](#_Toc58261703)

[2.3.4 Leckék szerkesztése 9](#_Toc58261704)

[2.3.5 Feladatok szerkesztése 12](#_Toc58261705)

[2.3.6 Nyitó oldal szerkesztése 12](#_Toc58261706)

[2.4 Alkalmazás felhasználói felület 14](#_Toc58261707)

[2.4.1 Kezdő képernyő 14](#_Toc58261708)

[2.4.2 Lecke nézet 15](#_Toc58261709)

# 1. Bevezetés

## 1.1 Témaválasztás indoka

A koronavírus okozta helyzet rámutatott, hogy mekkora szükségünk van az online elérhető tananyagokra. A diplomamunkám célja, hogy egy olyan környezetet biztosítson a c++ programozási nyelvnek, hogy azt könnyen fogyasztható formában lehessen elsajátítani. A leckékhez feladatok is tartoznak, így a felhasználó egyből tudja gyakorolni a friss tudást. A leckék különböző nehézségi szintek szerint vannak beosztva, így a felhasználó a saját képességeihez mérten választhat a tananyagokból. A leckék egy diasor szerű felületen vannak prezentálva, amik között a felhasználó a navigációs gombok segítségével léphet. Az alkalmazás tartalmaz egy online fordítót is, amivel a lecke tanulása alatt kipróbálhatjuk az új ismereteinket, illetve a programozási feladatokban próbára tehetjük a tudásunk. Ennek működéséről a dokumentáció későbbi szakaszában lesz szó.

## 1.2 Megoldandó feladat

A projekt célja, hogy a végfelhasználónak ne kelljen semmilyen extra programot telepítenie. Egy modern böngészőre legyen szükség mindössze. A kódfordításokhoz nem kell, hogy c++ fordítóprogram legyen az eszközünkön, ez mind szerver oldalon történik. A kód fordítása és futtatása egy külön Docker konténerben történik, szabályozható életidővel. Így ki tudjuk küszöbölni, hogy esetleges káros kód fusson le a szerveren és az kártékony hatásokat hajtson végre a szerver rendszerében. Az oldal irodalmi tartalmát egy headless cms (content management system) szolgáltatja. Ennek tartalma dinamikusan változtatható egy külön felhasználói felületen, ahol akár több szerkesztő felhasználót is létrehozhatunk. Itt tudunk új leckéket és feladatokat létrehozni. Ez a rendszer tartalmaz egy publikációs megoldást is, ami segítséget nyújt, hogy csak az a tartalom jelenjen meg az alkalmazásban, aminek már végeztünk a szerkesztésével.

# 2. Felhasználói dokumentáció

A teljes alkalmazás dockerizálva van, így ha lokálisan szeretnénk futtatni, nincs szükségünk nodeJS-re, vagy egyéb rendszerekre. Két alkalmazást kell telepítenünk, [Docker Desktop](https://www.docker.com/products/docker-desktop), illetve [Postman](https://www.postman.com/downloads/). A docker biztosítja már azt a lehetőséget is hogy a virtualizációhoz Windows-os konténereket használjuk, de válasszuk a Linux alapú konténereket mert ezzel lett fejlesztve az alkalmazás, és nem várt problémák adódhatnak ellenkező esetben. A Postman az adatbázis inicializálásához lesz szükséges.

## 2.1 Rendszerkövetelmények

Docker Desktop for Windows esetén:

* Windows 10 (64-bit) Pro, Enterprise, vagy Education:
  + Build 16299 vagy nagyobb
  + Hyper-V és a Containers Windows funkció engedélyezve kell hogy legyen.
  + Hardveres követelmények:
    - 64 bit-es processzor SLAT támogatással
    - 4GB memória
    - BIOS-szintű hardveres virtualizáció engedélyezése, [bővebben](https://docs.docker.com/docker-for-windows/troubleshoot/#virtualization-must-be-enabled)
* Windows 10 (64-bit) Home:
  + 1903-as vagy nagyobb verzió
  + WSL 2 funkciók engedélyezése, [dokumentáció](https://docs.microsoft.com/en-us/windows/wsl/install-win10)
  + Hardveres követelmények:
    - 64 bit-es processzor SLAT támogatással
    - 4GB memória
    - BIOS-szintű hardveres virtualizáció engedélyezése, [bővebben](https://docs.docker.com/docker-for-windows/troubleshoot/#virtualization-must-be-enabled)

Illetve egy modern böngésző, az alábbi verzió számokkal vagy nagyobbal:

* Chrome (86.0.4240.198)
* Edge (87.0.664.55)
* Firefox (82.0.2)

## 2.2 Telepítés

Miután sikeresen feltelepítettük a Docker Desktop és Postman alkalmazást, a projektet a következő képen tudjuk elindítani.

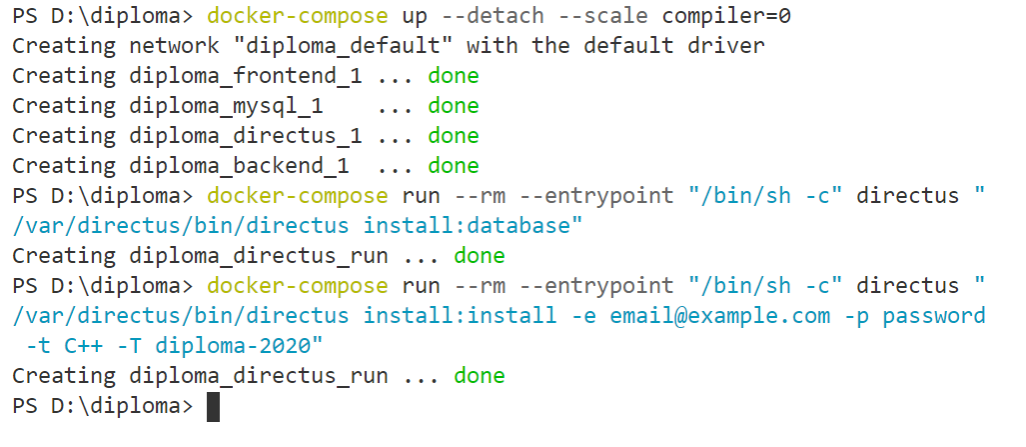
### 2.2.1 Első indítás

Az elsőindítás kicsit bonyolultabb, mivel inicializálnunk kell az adatbázist. Az adatbázis volume-ok segítségével perzisztálva van, így élettartama a konténerétől független. Tehát későbbi indításokkor nem kell a következő lépést végrehajtsuk.

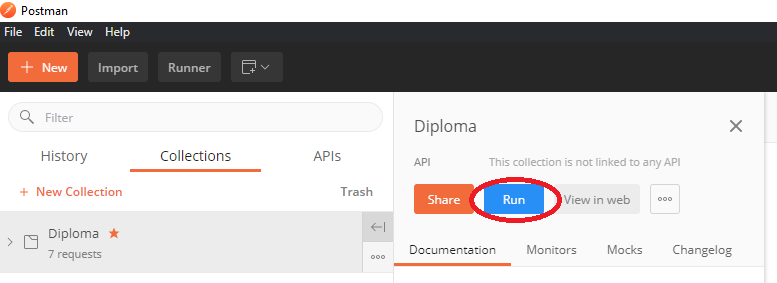
Lépések:

* Service-ek elindítása és adatbázis inicializálása
  + *docker-compose up --detach --scale compiler=0*
  + *docker-compose run --rm --entrypoint "/bin/sh -c" directus "/var/directus/bin/directus install:database"*
* Admin felhasználó létrehozása
  + *docker-compose run --rm --entrypoint "/bin/sh -c" directus "/var/directus/bin/directus install:install -e <email> -p <password> -t C++ -T <access-token>”*
  + Itt az <email> és <password> helyére tetszőleges értékeket írhatunk be.
  + Az <access-token> helyére diploma-2020 értéket írjunk, [lentebb](#access_token) olvasható mi a teendő ha más értéket szeretnénk

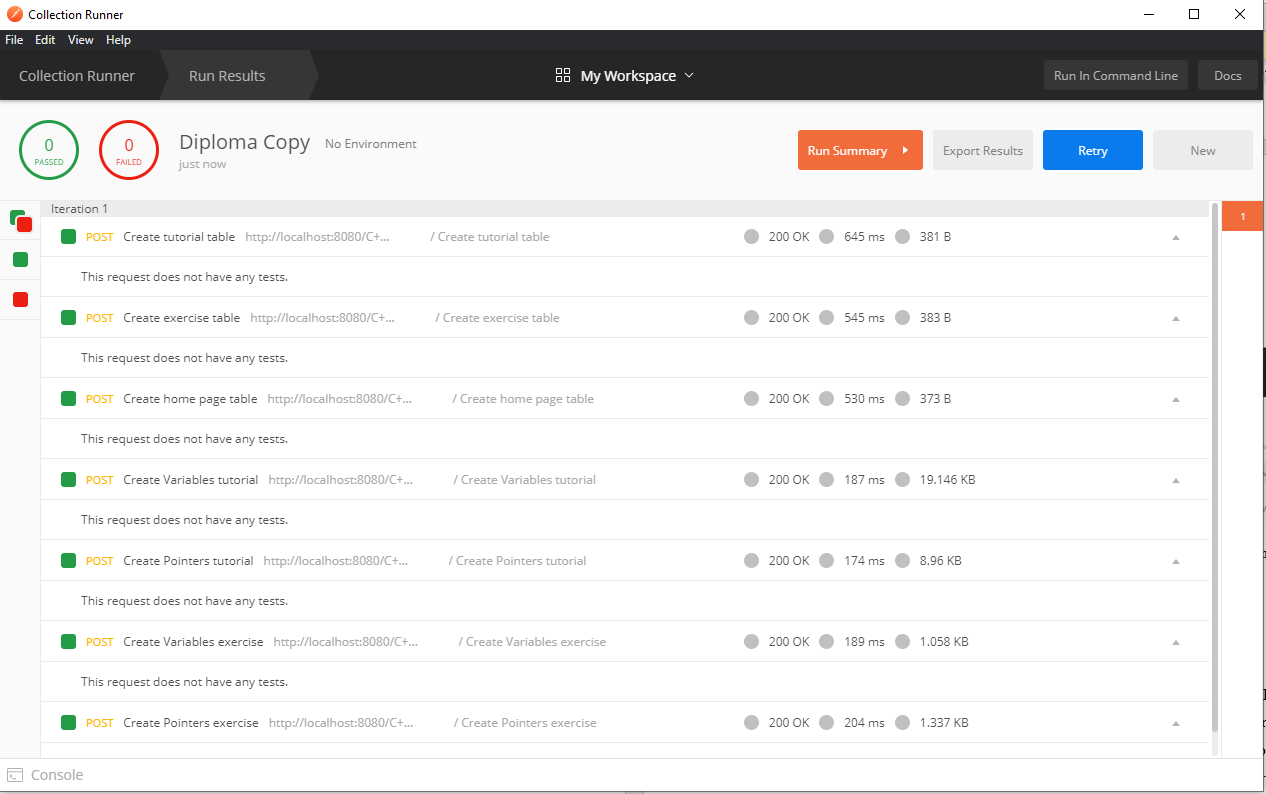
Példa:



1. ábra sikeres indítás konzol kimenetele

* Adatbázis feltöltése
  + Nyissuk meg a Postman alkalmazást
  + Importáljuk be a ./cms/postman\_collection file-t, majd futtasuk le

2. ábra collection fottatása a postman alkalmazásban

* + Állítsunk be 1 másodperces késést a kérések között. Ezt az eredményt kell látnunk: 

3. ábra collection futtatásának eredménye a postman alkalmazásban

* Ha ezeket a lépéseket elvégeztük az alkalmazás futásra kész állapotba került.

Saját access-token érték esetén:

A ’diploma-2020’ értéktől eltérő access token esetén a következő lépéseket kell megtennünk:

* A ./frontend/.env fájlban a **REACT\_APP\_ACCESS\_TOKEN** kulcs értékének a telepítéskor megadott <access-token> értéket kell adni.
* Újrabuildelni a frontend service-t a *docker-compose build frontend* paranccsal
* Leállítani a jelenlegi serviceket (*docker-compose down*)
* Majd újraindítani (*docker-compose up --detach --scale compiler=0*)

### 2.2.2 Már inicializált adatbázissal

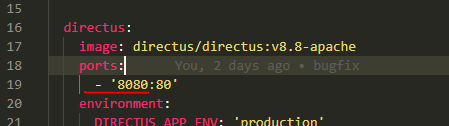
Ha már korábban elvégeztük 2.2.1 lépéseit, a következő képpen tudjuk elindítani az alkalmazást:

* *docker-compose up --detach --scale compiler=0*

## 2.3 Content management system felhasználói felület

Az alkalmazásban megjelenő tartalmat ezenen a felületen keresztük tudjuk szerkeszteni, illetve bizonyos entitások státuszát változtatni.

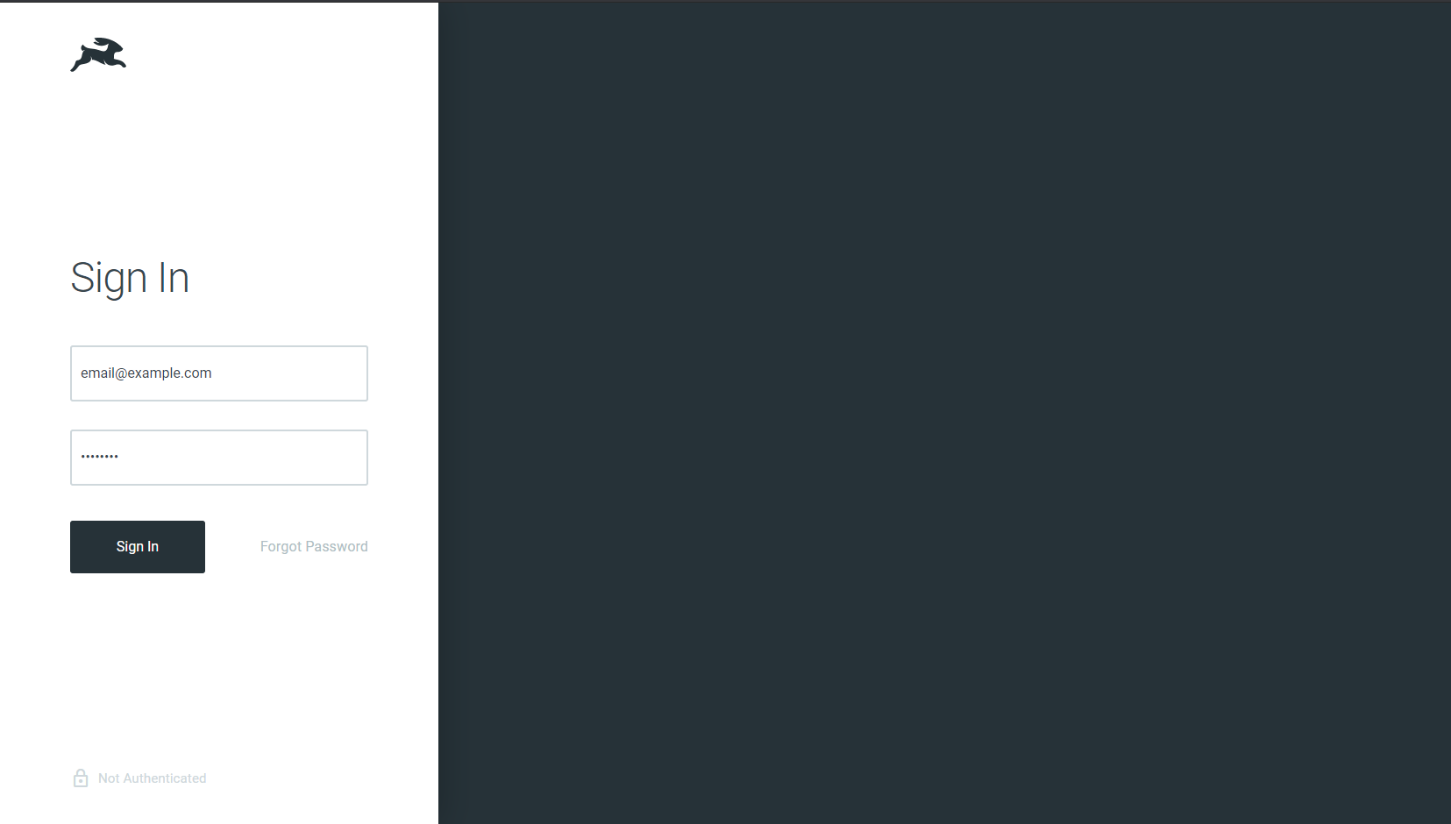
A felületet az alapbeállításokkal a szerver 8080-as portján tudjuk elérni. Esetünkben ez a lokális szerver, tehát az elérési útvonal a következő: <http://localhost:8080>.

A port számot módosíthatjuk, ha a projekt gyökér szintén levő *docker-compose.yml* file-ban átírjuk az alábbi értéket:

4. ábra részlet a docker-compose.yml fájlból

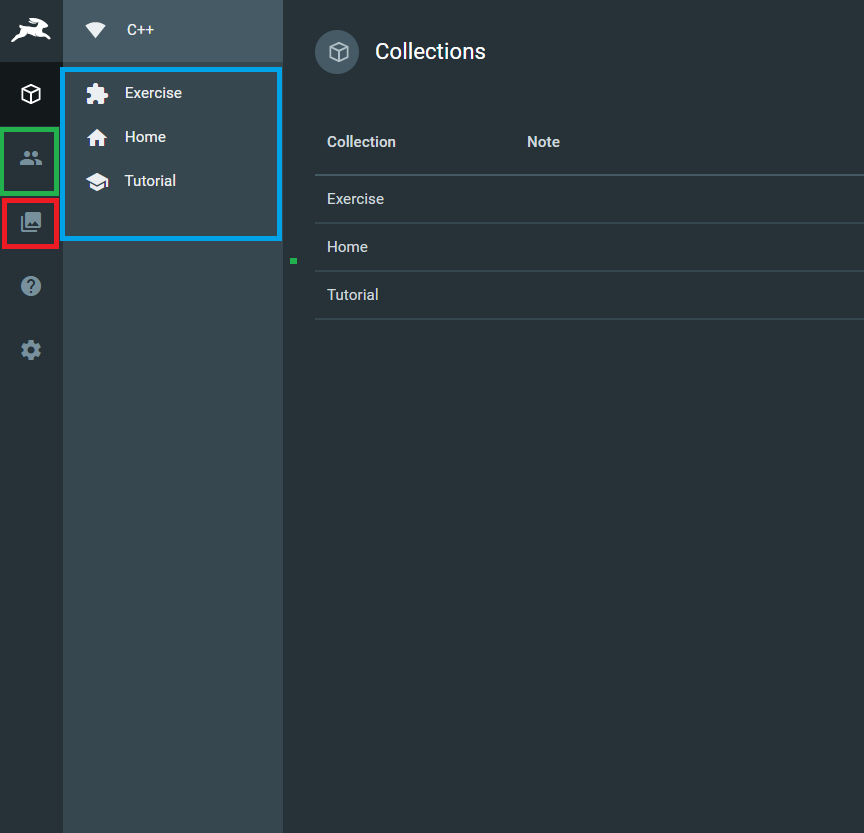
### 2.3.1 Bejelentkezési képernyő

A következő felülettel találkozhatunk, ha meglátogatjuk a fentebb található linket. Itt a telepítésnél megadott email és jelszó párossal léphetünk be. A rendszer sütik segítségével megjegyzi a belépési adatainkat, így azokat nem kell minden alkalommal újra megadnunk.



5. ábra CMS belépő képernyő

### 2.3.2 Kezdő képernyő



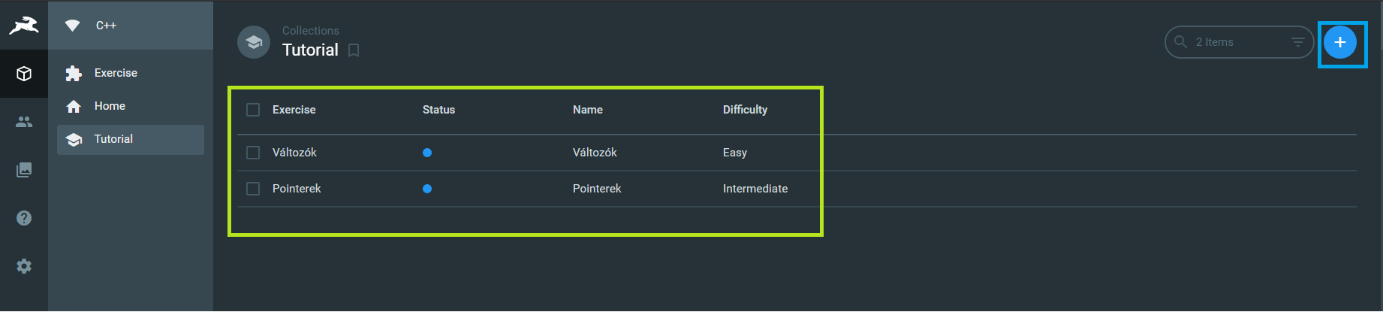
6. ábra CMS főmenű

A kékkel jelölt dobozban láthatjuk a collection-öket, ezek az egyes elemek csoportosítására szolgálnak. Itt tudjuk az elemeket kilistázni, bizonyos elemekre szűrni, illetve új elemet hozzáadni. Fontos megjegyezni, hogy a Home collection egyke, ami annyit jelent, hogy a másik kettővel ellenkezőleg csak egy darab példány létezik. Rákattintva egyből annak az egy elemnek a szerkesztő képernyője jelenik meg.

Zölddel látható a felhasználók kezelésére szolgáló felület, itt lehet új felhasználókat hozzáadni, vagy meglévők adatait, státuszukat szerkeszteni.

Piros színnel pedig a cms-be feltöltött fileokat-, média tartalmakat tekinthetjük meg, illetve adhatunk hozzá újakat.

### 2.3.4 Leckék szerkesztése



7. ábra CMS-ben kilistázott leckék

Két féle képen tudjuk megjeleníteni a részletes szerkesztői nézetet. A kék színnel jelölt hozzáadás gomb megnyomásával, ekkor egy új leckét hozunk létre. Illetve, ha a kilistázott elemek egyikére rákattintunk. A státusz oszlopban a kék pötty jelzi azt, hogy az adott lecke publikálva van, és listázva van az alkalmazásban. Piszkozat esetén szürkével jelölt.

Egy leckének nyolc szerkeszthető mezője van:

* status
* name
* url\_alias
* difficulty
* icon
* description
* slides
* exercise

Status:

Három választási lehetőségünk van, *Published, Draft, Deleted*. Draft esetén nincs listázva az adott lecke az alkalmazásban. Fontos hogy csak olyan leckét állítsunk Published-re, amihez a feladatok részét is elkészítettük már, és össze is kötöttük a kettőt az exercise mező segítségével. Ez egy lenyíló mező, ahol kiválaszthatjuk név alapján a megfelelő feladatot.

Url\_alias:

Ez egy kötelező szövegmező. Az angol ábécé kis betűi, illetve kötőjel használható benne. Ez a szöveg fog megjelenni a böngésző url részében, ha megnyitjuk a leckét.  
Példa:



8. ábra url alias megjelenése

Difficulty:

Ez egy kötelező lenyíló mező. Három lehetőségünk van *Easy, Intermediate, Professional*. Azt befolyásolja ez a mező, hogy a menüben melyik kategória alá kerüljön a lecke, illetve jelzést is az a felhasználónak a lecke nehézségéről.

Icon:

9. ábra lecke kártyák

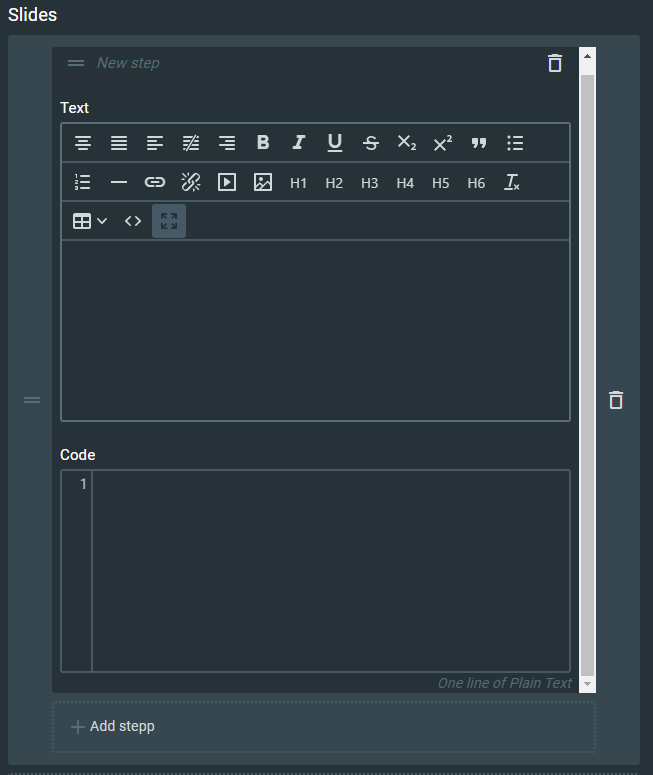
Opcionális mező, a leckekártyán megjelenő illusztráció. Ha nem adjuk meg, helyét a nehézséghez rendelt tölti ki.

Description:

Kötelező szövegmező, ami összefoglalja a lecke lényegét, és érdeklődést kelt fel. A lecke kártyán jelenik meg, illetve a megnyitott lecke neve alatt. 

10. ábra lecke leírás és cím

Slides:

Ez egy összetett mező, itt lehet megadni a tartalom lényegi részét. A lecke anyaga egy slideshow szerű felületet van prezentálva. Két entitásunk van, slide és step. Egy slide-on belül több step-ünk is lehet, és egy step-ben lehet kódrészlet vagy szöveges tartalom. A slideshow-n van egy léptető amivel a következő step-et jeleníthetjük meg, vagy léphetünk vissza az előzőre. Ha el szeretnénk tüntetni az előző stepeket a slideshow-ról, kezdjünk egy új slide-ot és arra rakjunk új step-eket. A szöveg valójában html tartalmat hoz létre, tehát lehet beszúrni képeket és multimédiás tartalmat is. Van egy teljes képernyős gomb is, amivel megkönnyebbíthetjük a szerkesztési élményt. Ez a fenti ábrán a világosabb szürke hátterű gomb. Fontos, hogy igaz két mezőt látunk egy step-en belül, de csak az egyikbe írjuk tartalmat, így elkerülve a sorrendiségi problémákat. A step-eket, illetve a slide-okat átrendezhetjük, ha csupánt áthúzzuk a sorrendjüket a jobboldali sáv segítségével.

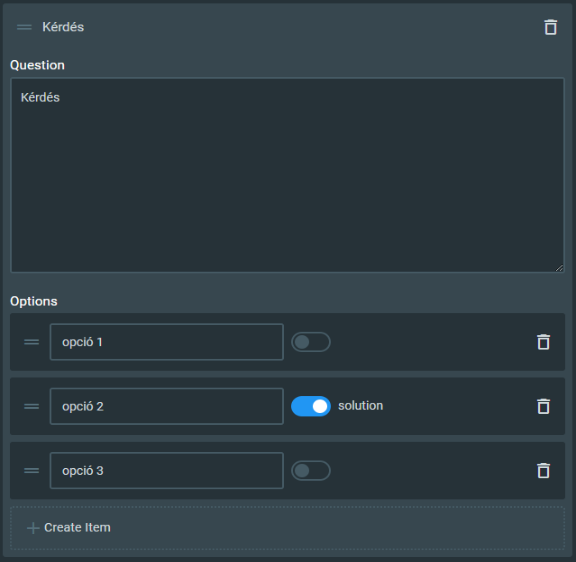
11. ábra slide szerkesztése

Exercise:

Lenyíló mező, a rendszerben tárolt leckéket listázza ki név szerint.

### 2.3.5 Feladatok szerkesztése

Meglévő feladatot ugyanúgy szerkeszthetünk vagy hozhatunk létre mint a leckék esetén. Egy leckéhez tartozó feladat kétfajta feladatból állhat.

**Kvíz feladatok**, ezeknek száma 0-tól a kívánt mennyiségig terjedhet. Nem csak egy helyes válasz lehetséges, egy kis kapcsolóval lehet választani, hogy a választási lehetőség helyes-e. Sajnos a cms rendszerben van egy ismert hiba, nem menti el a kezdetben hamisnak jelölt és úgy is hagyott opciókat. Ennek egy egyszerű, de kissé idegesítő megjavítása ha egyszerűen ki be kapcsoljuk az igaz/hamis értéket.

**12**. ábra quiz szerkesztése

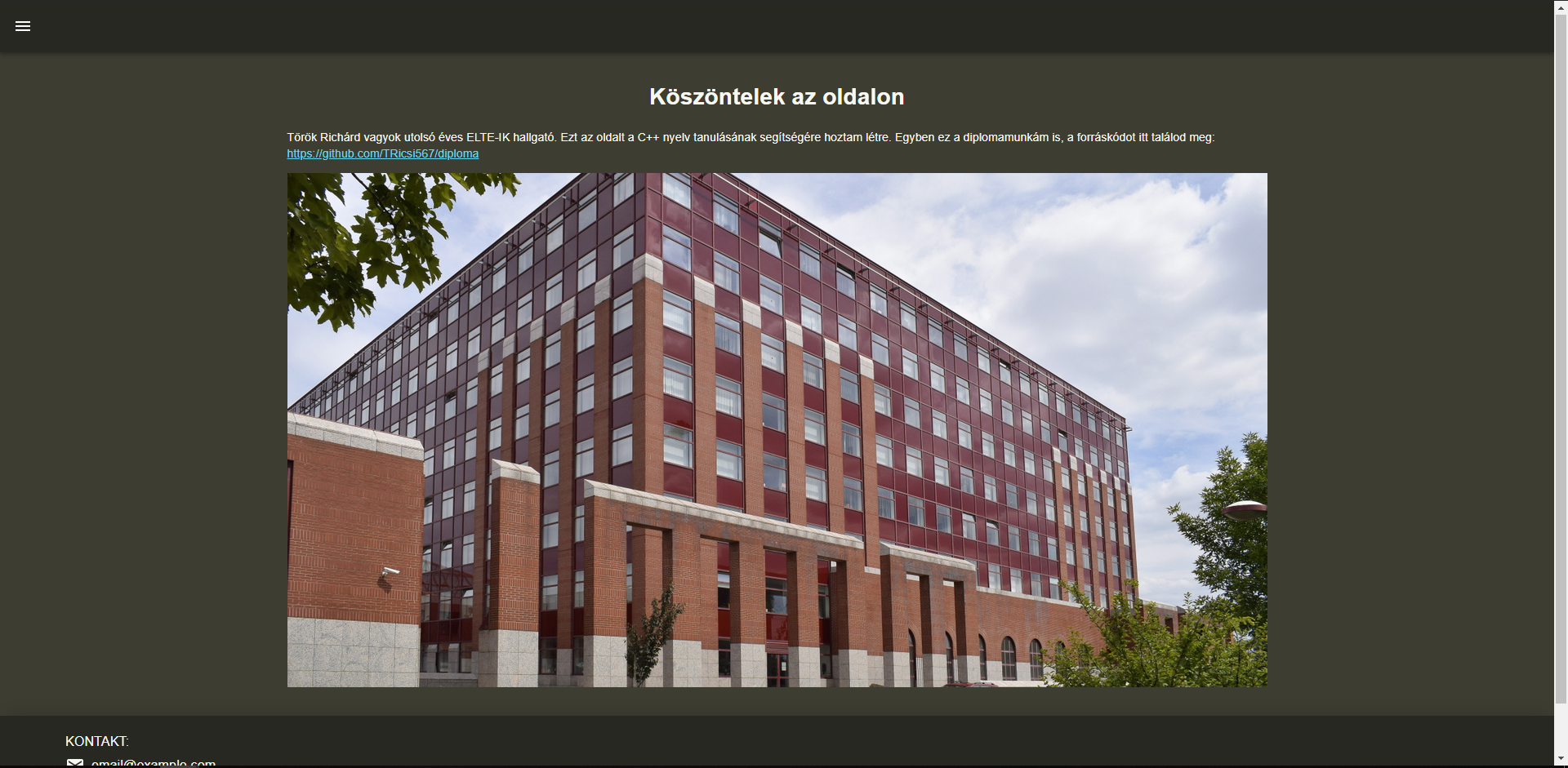
**Programozási feladat,** ebből feladatsoronként egy létezhet. Ennek a entitásnak három mezője van, *question, description, solution*. A description kivételével kötelező mezők. Az opcionális description mezőbe írhatunk tippeket a feladat megoldásához, illetve kis segítséget. Fontos kihangsúlyozni, hogy a rendszer úgy ellenőrzi a feladatot hogy a program output-ját hasonlítja össze a solution mezőbe írt értékkel. Itt a betűméret nem számít, a program átalakítja a program kimenetét és a megoldás szövegét egyaránt kisbetűssé.

### 2.3.6 Nyitó oldal szerkesztése

A home collection alatt tudjuk szerkeszteni a kezdőképernyőn megjelenő tartalmat, a láblécben megjelenő kontakt emailt, és telefonszámot, valamint az oldalsó menüben megjelenő ’hasznos linkek’ szekciót. Ha nem adunk meg egyetlen hasznos linket sem akkor, maga a szekció címkéje se fog megjeleni az oldalsó menüben. Maga a kezdőoldalon megjelenő tartalom a *content*  mezővel szerkeszthető, ez is egy html tartalmat generál, el lehet helyezni benne különböző méretű fejsorokat, multimédiás tartalmakat, képeket, és még sok mást.

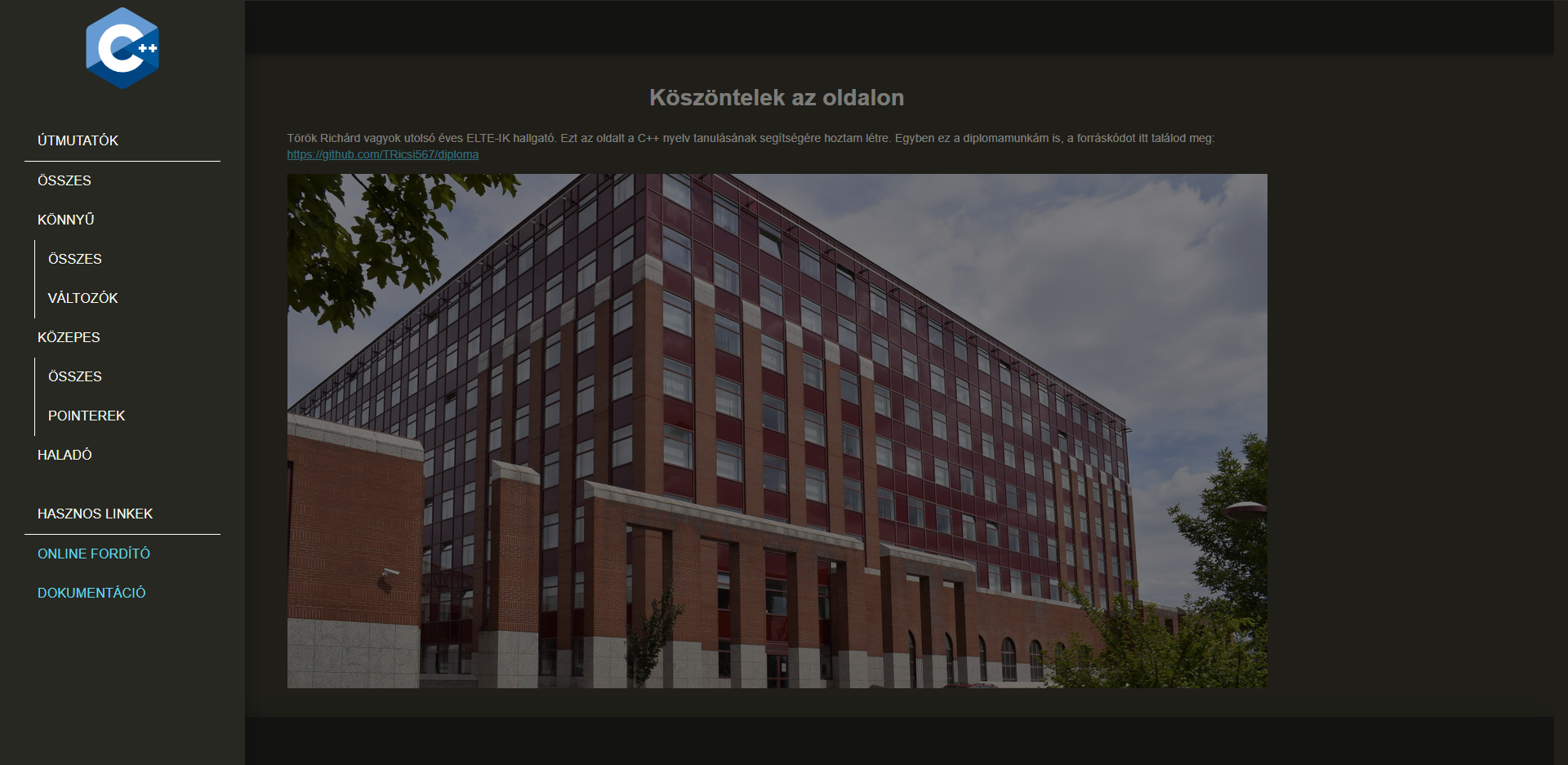
## 2.4 Alkalmazás felhasználói felület

### 2.4.1 Kezdő képernyő



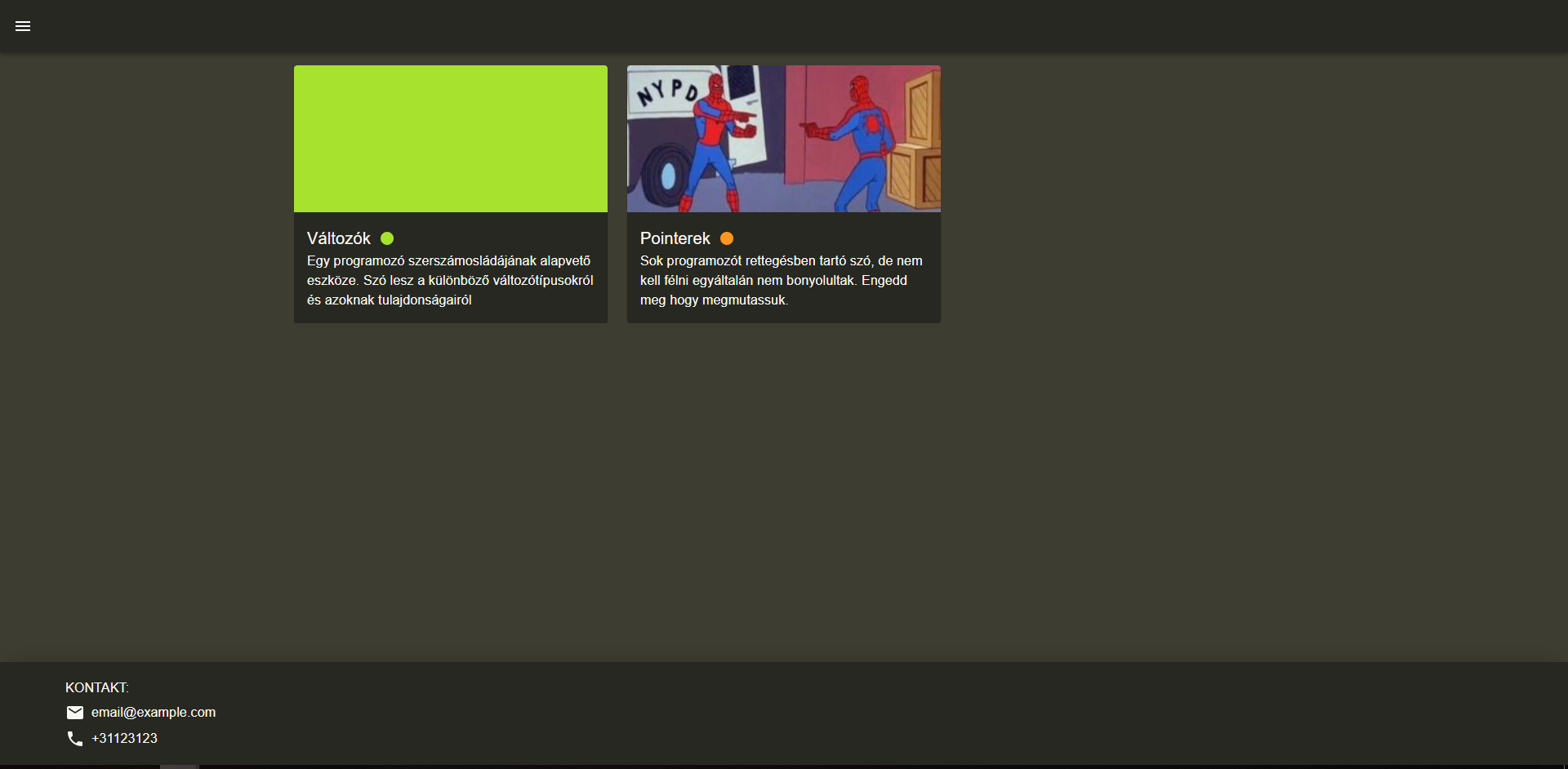
13. ábra kezdő képernyő

Az alkalmazást betöltve az 13. ábrához hasonló nézettel találkozunk. Itt a tartalom a szerkesztőtől függ. Ezen az oldalon különböző cikkeket olvashatunk, amit a szerkesztő létrehoz, vagy esetleg más hivatkozásokat is hozzáadhat.A láblécben találtható a kontakt email és telefonszám. Ezek linként is szolgálnak, rájuk kattintva egyből elindul a számítógépünkön alapértelmezett levelező, ittelve telefonáló rendszer. A bal felső sarokban található a menü gomb. A menüt megnyit va a következő nézetet kapjuk:



14. ábra kinyitott menü

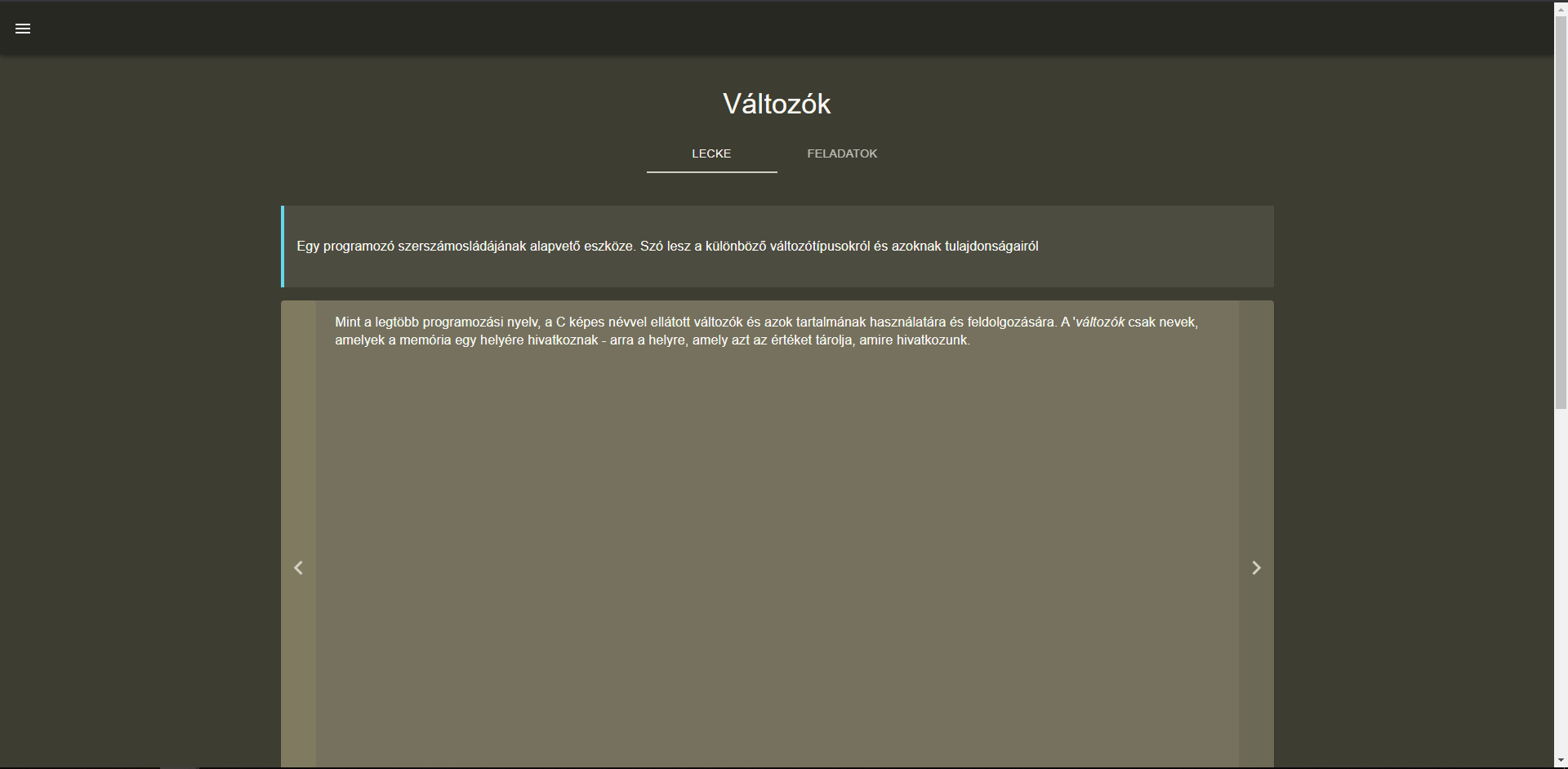
A C++ ikonra kattintva visszajutunk a főoldalra. Az útmutatók cimke alatti könnyű, közepes és haladó menüpontok újabb menüpontokat nyitnak meg. Bármely kategóriának az ’összes’ menüelemére kattintva a 15. ábrán látható egyik összegző oldalra jutunk. Konkért leckére kattintva a választott tananyaghoz navigál az applikáció. A hasznos linkek alatti elemek a cms-ben adhatóak meg. Ezek főlegkülső hivatkozások.



15. ábra összegző képernyő

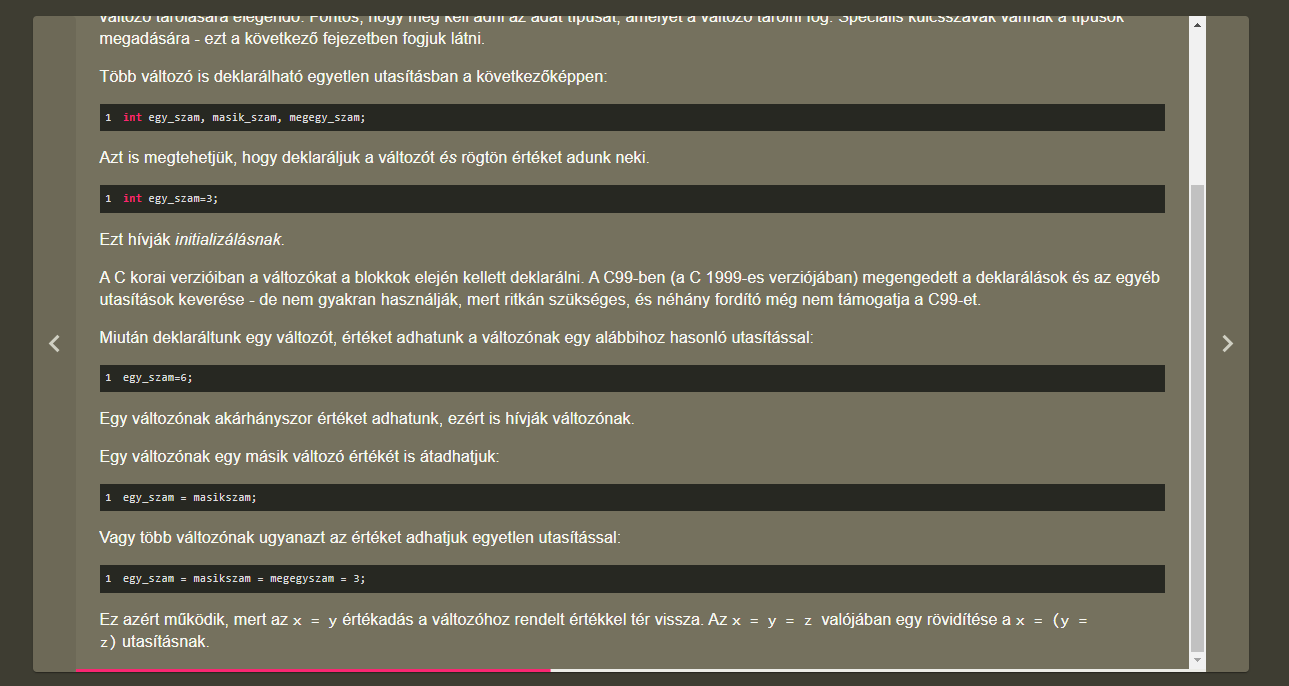
### 2.4.2 Lecke nézet

A főoldalhoz hasonloan itt is megmarad az oldal szerkezete, fejléc benne a menüvel és a lábléc. Az oldal tartalma változik csak.



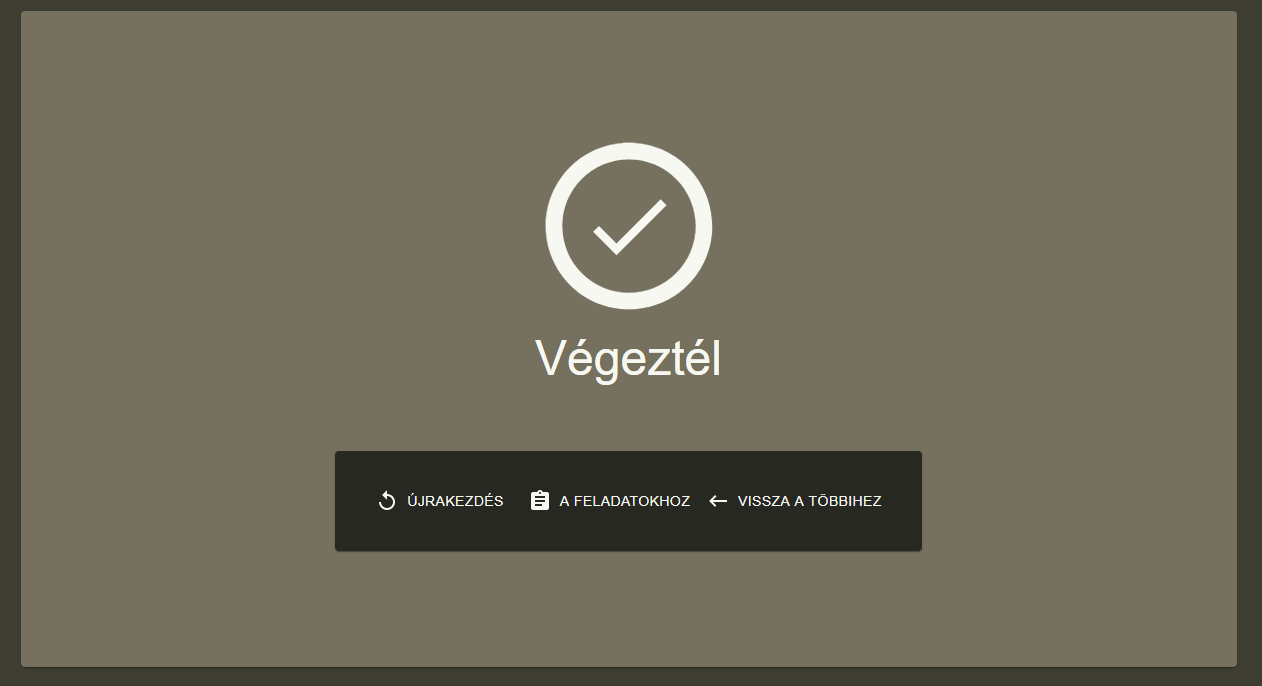
16. ábra lecke nézet

Az első sorban a lecke címe található, alatta egy menü amivel válthatunk a lecke és feladatok között. A menü alatt pedig a lecke leírása található. Itt az egész szöveg meg van jelenítve, mivel megeshet hogy a leckekártyán a szöveget levágjuk ha az túl hosszú. Alatta pedig a slideshow található, amiben a lecke anyaga van. Itt a nyilakkal jelölt gombokkal lehet előre vagy hátra haladni az anyagban.



17. ábra slideshow

Alul egy rózsaszín csík jelzi hol tartunk az anyagban. Ha az anyag végére értünk a következő képernyőt látjuk:

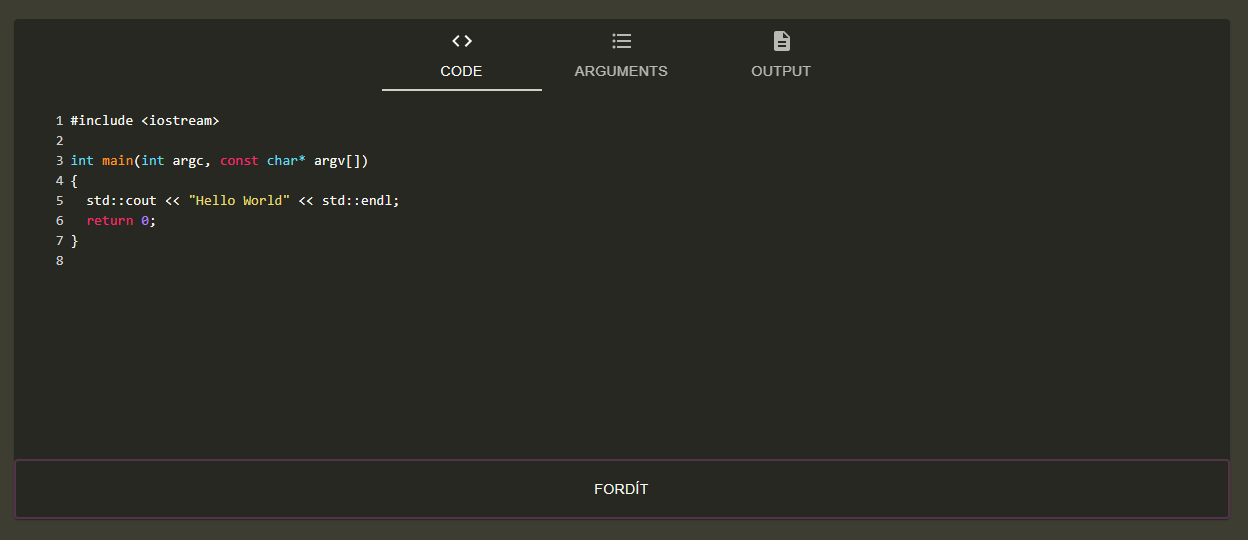


18. ábra befejezett lecke

Három opciót kínál fel nekünk a felület. Újrakezdhetjük a tananyagot az elejétől, a feladatok szekcióra ugorhatunk, vagy visszamehetünk a jelenlegi lecke szinjével megegyező összegző képernyőre.

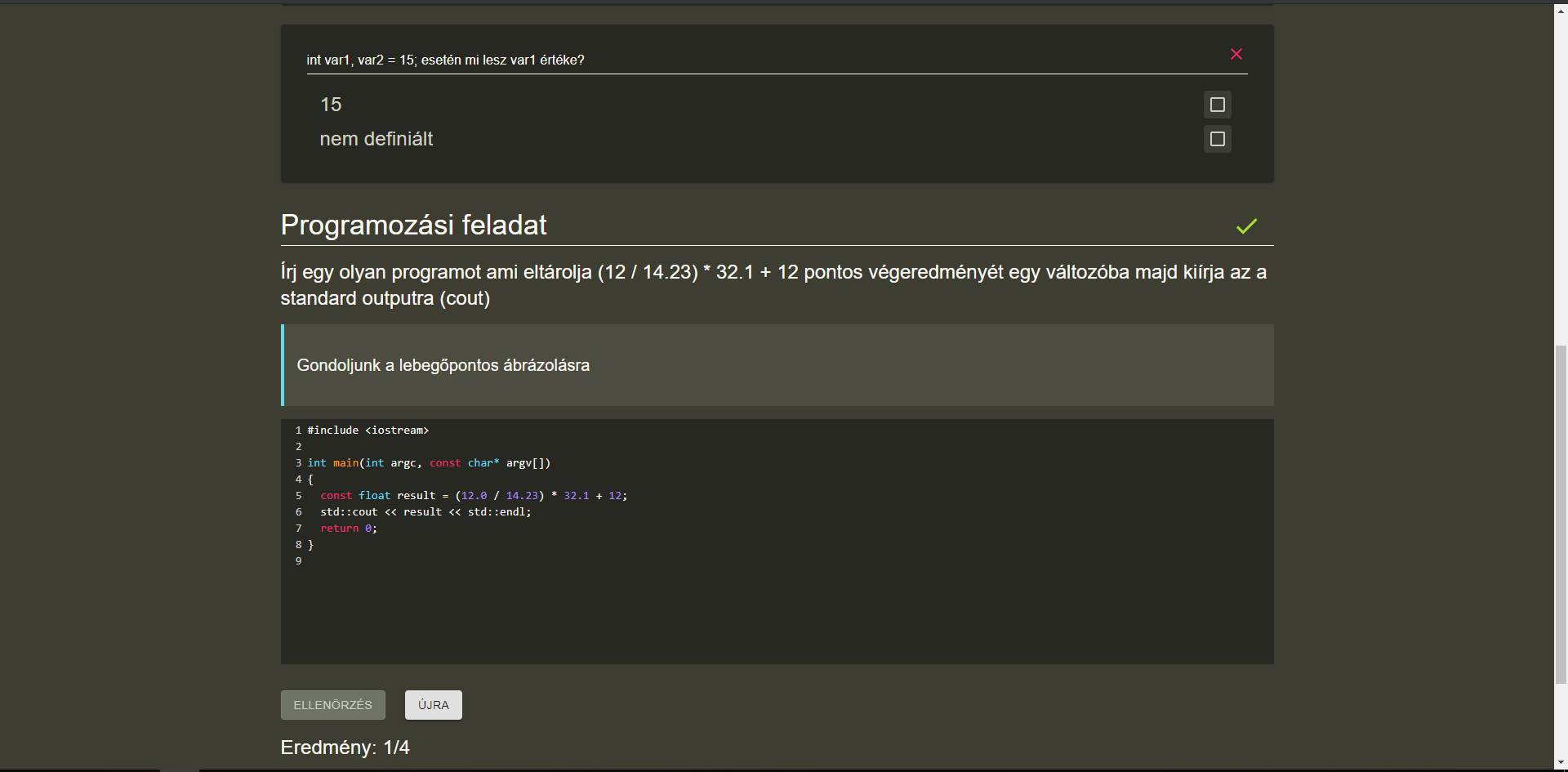
A slideshow alatt található az online kódfordító rész, a 19 ábrán láthatjuk. Ez három fülből áll, code, arguments és output. A code fülön tudjuk szerkeszteni a forrás kódot amit majd le szeretnénk futtatni, az arguments fül alatt adhatjuk meg az argumentumokat amiket a program kap meg futtatáskor. Az output fülön pedig a program futásának kimenetelét láthatjuk. Ha hibás a forráskódunk és fordítási hibánk van azt is ezen a fülön láthatjuk. A fordít gomb lenyomásával indíthatjuk el a fordítási folyamatot, ameddig ez fut a gomb inaktiv lesz. Ha véget a folyamat átvált automatikusan a output fülre. Lényegében három féle képpen végződhet a folyamat.

* Sikeresen lefut a program, ekkor a program kimenetele kerül kiírásra
* Fordítási hibánt van, ekkor a fordító által írt hibaüzenetet írjuk ki
* A folyamat túllépi az időkorlátot, ekkor a *timed out* üzenetet írjuk ki



19. ábra online fordító

A feladatokat két féle képpen tudjuk megjeleníteni, ha a fenti almenüben a ’Feladatok’ menüpontot választjuk (16. ábra), vagy miután végeztünk a leckével a középső ’A feladatokhoz’ opcióra kattintunk. Először a kvízek vannak kilistázva, ha megadtunk belőlük legalább egyet a cms-ben, utána pedig a programozási feladat.



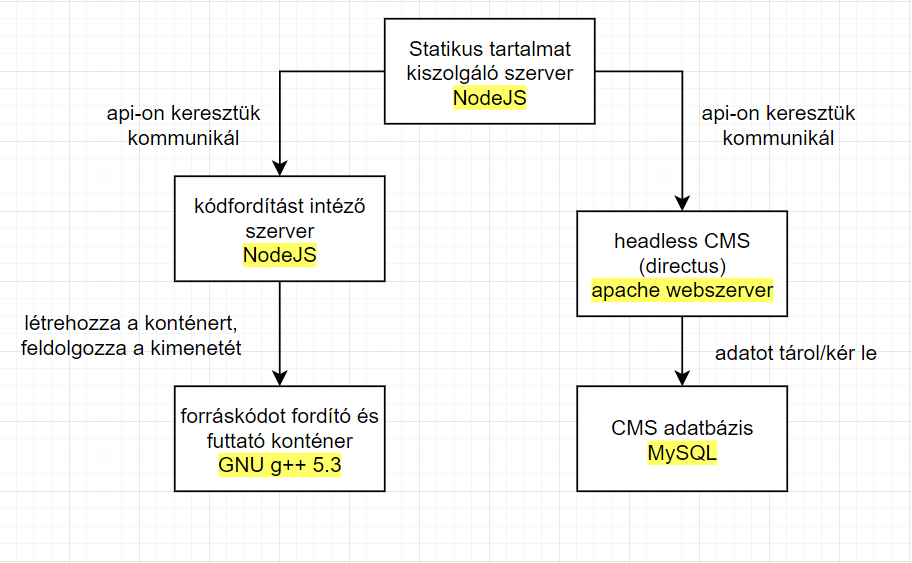
. ábra leellenörzött feladatmegoldás

Az ellenörzés gombra kattintva, elküldjük a válaszainkat kiértékelésre. Ameddig ez tart az ’újra’ és ’ellenörzés’ gomb is inaktív állapotba kerül. Ha végzetta program a kiértékeléssel minden kérés kérdést megjelöl egy zöld pipával vagy pirosz ikszel a válasz helyességétől függően. Erre a 20 ábrán láthatunk példát. A gombok alatt pedig kiírja hány kérdésre adtunk jó választ az összesből. Az újra gomb megnyomásával a feladatlap visszaáll az eredeti állapotába és újból próbálkozhatunk. Érdemes megjegyezni, ha a feladatok és lecke fül között váltakozunk, a feladatlap mindig visszaáll a kezdetleges állapotába, így nem lehet olyan könnyen lenézni a választ a lecke anyagár

# 3. Fejlesztői dokumentáció

## 3.1 Alkalmazás felépítése

Az alkalmazás minden modulja egy Docker konténerben van elhelyezve, így könnyítve a jövőbeli a skálázhatóságot, és éles környezetbe való helyezését. A forráskódban több docker-compose fájl is szerepel, a gyökér szinten található az amiben az alkamazásban használt összes modul/service szerepel. A compose fájlok fontosságát érdemes kiemelni, ugyanis ezek sok kényelmi funkcióval szolgálnak, mint például volume-ok helybeli létrehozása, amivel a futások közti adatokat tudjuk perzisztálni, vagy például egy alap hálózatot, amin keresztül a konténerek tudnak kommunikálni, és egyfajta címfordítást is biztosít, aminek segítségével könnyebben tudjuk megadni az elérési címeket. Az alkalmazás architektúráját a következőképp ábrázolható:



ábra az alkalmazás szerkezete

A node JS szervereken futó alkalmazások a 14.15.0 verzióval lettek buildelve, illetve azon is futnak. A build is egy ideiglenes Docker konténerben készül, nem szükséges hogy a saját eszközünkön is ez a verzió szerepeljen. A buildelő és futtató konténereknek egyaránt 3.12-es verziójú alpine linux az operációs rendszere, a lehető legkisebb image méret érdekében.

### 3.1.1 Statikus megjelenési tartalom

A frontend elkészítéséhez a ReactJS keretrendszert használtam. A create-react-app script által generált alap projektre fekszik. Ez magában foglalja a Jest unit tesztelési keretrendszert, Webpack eszközt ami a csomagolást és buildelést végzi, illetve egy fejlesztői szervert is tartalmaz. A csomag magába foglalja még a BabelJS-t is ami a kód transplieolásához szükséges.