Vizsgaremek

Bognár Csilla

Maties Mónika

Tóth Tünde

Vizsgaremek adatlap

A Vizsgaremek készítői:

Neve: Bognár Csilla
E-mail címe: bognarcsini@gmail.com

Neve: Maties Mónika
E-mail címe: monica.maties@gmail.com

Neve: Tóth Tünde
E-mail címe: tothtunde001@gmail.com

A Vizsgaremek témája:

Online felület kialakítása iskolai tanulmányi versenyek és tudásszint felmérés érdekében, melyhez a tanárok számára egy asztali alkalmazás kapcsolódik a feladatlapok közzététele, versenyek kiírása érdekében.

Vizsgaremek címe: Tudástér – ahol hasznos a lógás

Konzulens tanár:

Schneider Dávid

Kelt: Budapest, 2023. május 12.

.....	.....
Bognár Csilla	Tóth Tünde
.....	.....
Maties Mónika	Konzulens

A Vizsgaremek készítőinek aláírása, a konzulens tanár aláírása.

Eredetiség nyilatkozat

Alulírott tanuló kijelentem, hogy a vizsgaremek saját és csapattársa(i)m munkájának eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült vizsgaremekrészét képező anyagokat az intézmény archiválhatja és felhasználhatja.

Budapest, 2023.05.12.

.....

Bognár Csilla
Tanuló aláírása

.....

Maties Mónika
Tanuló aláírása

.....

Tóth Tünde
Tanuló aláírása

Tartalomjegyzék

Bevezető rész 5

1.Fejezet A felhasználói dokumentáció ismertetése 6

1.1. A felület illusztrálása 6

1.1.a) A weboldal kialakítása 6

1.1.b) Az alkalmazás megjelenítése 6

1.1.c) Folyamatábra 6

1.2. Háttérben működő funkciók áttekintése 9

1.2.a) Adatbázis 9

1.2.b) Front-end 10

1.2.c) Back-end 10

1.2.d) Alkalmazás 11

1.3 Használati útmutató 11

2. Fejezet Fejlesztésünk a célkeresztben 12

2.1.Munkamegosztásunk meghatározása, részvételünk a csapatban 12

2.2.Telepítési folyamatok ismertetése 14

2.3.Adatbázisunk részletes elemeinek bemutatása 15

2.3.a) Adatbázisunk grafikusan történő ábrázolása 15

2.3.b) A mezők kialakítása, strukturálása 16

2.4.Fontosabb kapcsolatok közötti összefüggések áttekintése 18

2.5.Alkotásunk komponens szegmensének szemléltetése 24

2.6.Az API főbb összetevőinek kibontása 29

2.7.Tesztelés 34

3. Fejezet Jövőbeli kilátások, fejlesztési ajánlások és lehetőségek 36

Záró rész 38

Felhasznált irodalom 39

Nyomtatott forrás 39

Internetes források 39

Melléklet 40

Tudástér – ahol hasznos a lógás

Bevezető rész

Meghatározó szeretet játszik az életünkben a verseny. Gyerekkorunkat végig kísérik olyan események, amelyek a küzdelemről szólnak. Játszva tanulunk meg részt venni a különböző feladatokban, szabályokat alkotunk és végül megmérettetünk. Az iskolás éveink során számos kihívás ér bennünket, és ezt ha jól sikerül teljesítenünk jutalmat kapunk. Ahhoz, hogy mindez jól működjön elengedhetetlen, hogy olyan személyek koordináljanak és vezessenek minket útjaink során, akik segíthetnek leküzdeni az akadályokat.

Projekt munkaként egy 2023-ban megrendezésre kerülő verseny weboldalának készítését tűztük ki célul. Ezen dokumentációnkban szeretnénk bemutatni azokat a tevékenységi köröket, amelyeket használtunk.

Összefoglaló átfogó komplex képet fogunk adni arról, hogy ki és milyen folyamatot látott el, ehhez kapcsolódóan milyen nehézségekkel kellett szembenéznünk.
Meghatározó szempont volt számunkra, hogy olyan tanulmányi verseny elkészítése legyen, amelyet a tanárok és diákok tudnak használni. A tanároknak létrehozott alkalmazás segítségével lehetőség nyílik arra, hogy újabb kérdéseket tegyenek fel; használhassák a megkapott eredményeket korrepetálásra, valamint a tanulók eredményes munkájának figyelemmel kísérésére.

A tanulóknak támogatást nyújt egy – egy tananyag feldolgozásában, illetve hibájuk kijavításában. A válaszok elérhetősége által láthatóvá válik számukra, mi az amit még nem sajátítottak el kellő alapossággal. Grátiszként kellő motiválást érezhetnek azok a diákok, akik a legjobban teljesítenek.

Elképzelésünkkel reméljük, hogy hatékonyabb oktatás valósítható meg az intézményekben tanuló fiatal talentumoknak, és a jövőben alkalmazható, izgalmas erőpróbák elé állítják majd a prominens növendékeket.

1.Fejezet A felhasználói dokumentáció ismertetése

1.1. A felület illusztrálása

1.1.a) A weboldal kialakítása

A megjelenés szempontjából egyszerűségére törekedtünk kialakítása során. Könnyen kezelhető és átlátható menüpontokkal, rövid tömör megfogalmazása a meghatározott feladatoknak. Célunk, hogy egyértelmű legyen, és a korosztályok könnyen és gyorsan tudják elolvasni az információkat. A tartalmak megjelenése egy a mai kornak megfelelő kialakítás, amely az alap színekre épít elsődlegesen. A képi összhatás egy harmonikus felület, amelyet szívesen használ egy diák, valamint egy tanár. Összehangolt arculat létrehozására[1] törekedtünk, egy – egy játékosabb szín bekapcsolódásával együtt.

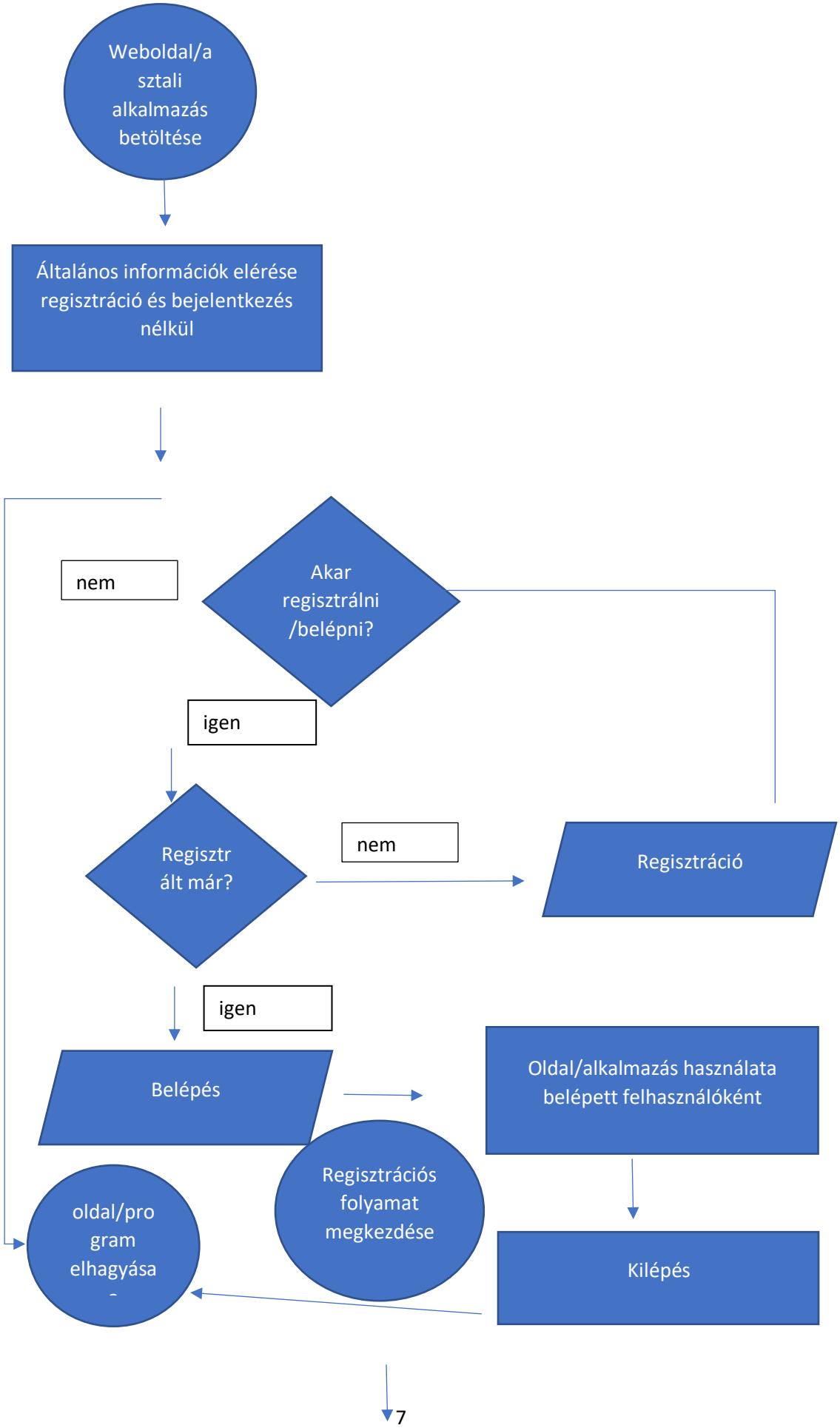
1.1.b) Az alkalmazás megjelenítése

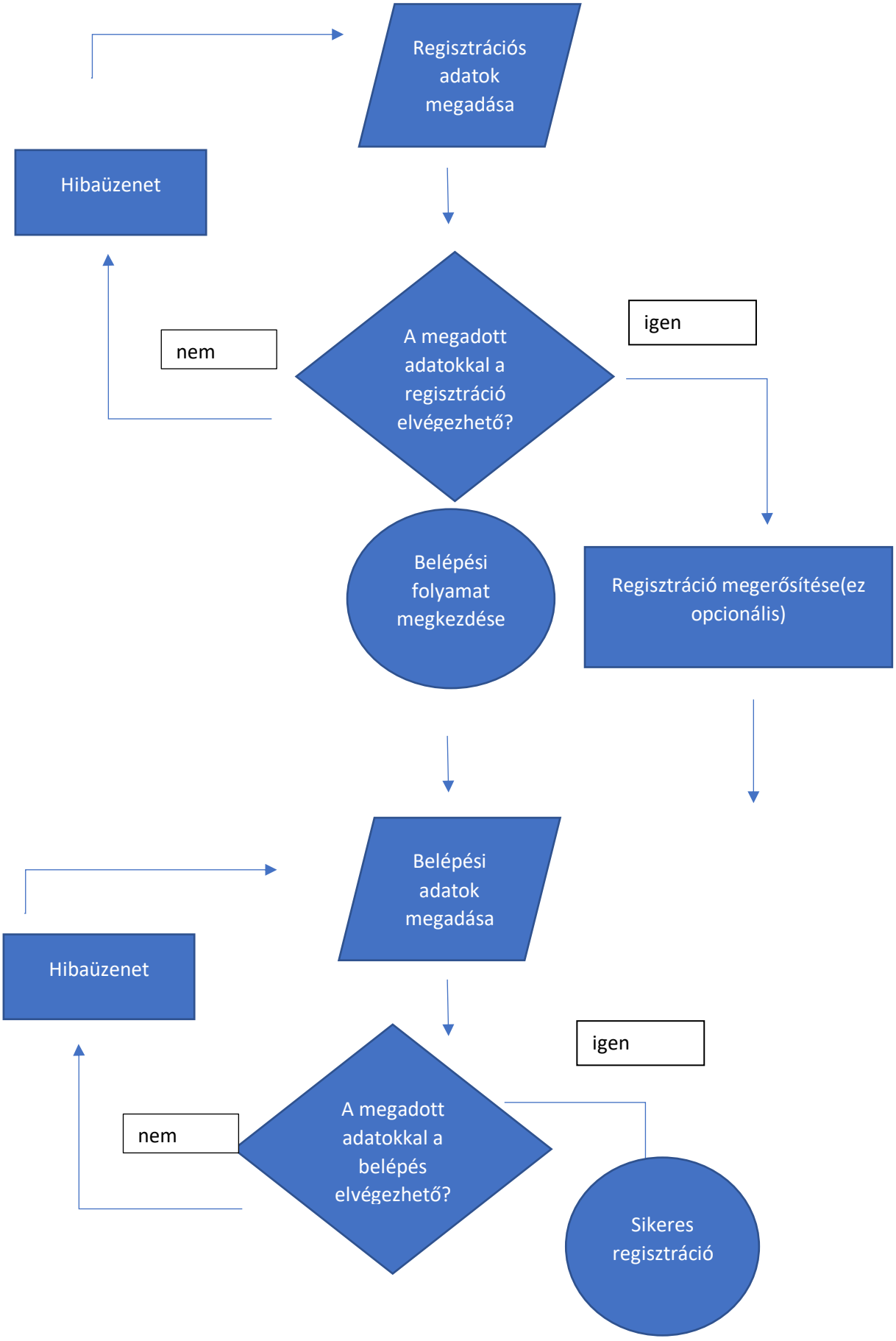
Az grafikus felület egy könnyen kezelhető, lágy színekkel megkonsturált felület. Törekedtünk arra, hogy a tanárok minden korosztályának egyértelmű legyen, igyekeztünk minden bonyolult funkciót és megfogalmazást kerülni, hogy minél bátrabban használják, és alkalmazzák munkájuk során. A felesleges információk átadása, egy hosszú nap után is nehezebb felismerést eredményez. Nap mint nap rengeteg inger ér minket, és ezen behatások által csökken a teljesítő képességünk, tudatunk inkább a rövidebb kifejezéseket fogja át ezek után, a letisztult elemek ilyen estben pedig igazi kincset érnek.

1.1.c) Folyamatábra

Ez azoknak az elemeknek a megértéséhez szükséges, amelyeken végighalad a koncepciónk, és amely által az elképzelésünk megvalósult. Kezdeti lépésünk az elgondolás volt, amelynek útján ha elindulunk sok fontos apró gondolatmenetet tekinthetünk át, és így tud megvalósulni az amit megterveztünk.
Az alábbi ábrával szeretnénk szemléltetni a web valamint az asztali alkalmazást:

[1] Weboldal tippek, 10 weboldal tipp
https://weblapdesign.hu/weboldal-keszites/weboldal-tippek/
(Letöltés ideje: 2023.05.10.)


Weboldal/asztali alkalmazás betöltése
Általános információk elérése regisztráció és bejelentkezés nélkül
Akar regisztrálni /belépni?
nem
igen
Regisztrált már?
nem
Regisztráció
igen
Belépés
Oldal/alkalmazás használata belépett felhasználóként
Regisztrációs folyamat megkezdése
oldal/program elhagyása
Kilépés


Regisztrációs adatok megadása
Hibaüzenet
A megadott adatokkal a regisztráció elvégezhető?
nem
igen
Belépési folyamat megkezdése
Regisztráció megerősítése(ez opcionális)
Belépési adatok megadása
Hibaüzenet
A megadott adatokkal a belépés elvégezhető?
nem
igen
Sikeres regisztráció

A folyamatábra abban segít egy projekt megvalósításánál, hogy vizuálisan megpróbáljuk elképzelni, milyen utat kell bejárnunk ahhoz, hogy a legjobb megoldást megtaláljuk. Ez által lehet kiindul, és megtervezni az elkövetkező lépéseket. Amennyiben jó gondolatmenetet követünk, és sikerül egy jó ábrát elkészíteni, akkor később nem kell módosítani, vagy elölről kezdeni az egész feladatot.
Így sokkal jobban megértjük az egyes részeket, és elemeket, amelyeken keresztül kell menni, és amely által megvalósulhat a projekt.
Ez a leghatékonyabb szemléltetési módja szerintünk, és kezdetben ez segített a legtöbbet abban, hogy az adatbázisunkat megalkothassuk.

1.2. Háttérben működő funkciók áttekintése

1.2.a) Adatbázis

Az adatbázis[2] fogalma a következő lenne:,, Az adatbázis olyan számítógépen tárolt és rendezett adathalmaz, melynek elemei összetartoznak. Az adatbázisok célja, hogy az adatokat hosszútávon, tartósan lehessen tárolni, illetve, hogy a tartalma gyorsan kereshető legyen.”[3] Egy olyan relációs adatbázist hoztunk létre, amely egyeddel, attribútummal, táblákkal, mezőkkel, rekordokkal rendelkezik. Az egyed az adatok tárolását szolgálja, az attribútum az egyedi tulajdonságokat foglalja össze, a táblák amelyek átjáróként kapcsolódnak egymáshoz. A mező az egy – egy oszlopot jelent, a rekord pedig az adatok tárolásához használt sorokat foglalja magába. Fontos hogy az adatbázis kapcsolatai a következőképpen alakulhatnak:
1:1 kapcsolatnál minden egyes egyedhez pontosan csak egy másik egyed tartozhat. Ennek jelölése: a kapcsolatot úgy kötjük az egyedekhez, hogy minden egyed felé mutat nyíl.
1:N kapcsolatnál (vagy "egy a sokhoz" kapcsolat) alapján az egyik egyedhez több másik egyedet tudunk hozzá rendelni, de a másik csoport minden egyes példányához azonban csak pontosan egyet társíthatunk.
N:M kapcsolatnál (vagy "sok a sokhoz" kapcsolat) alapján egy egyed példány több másikkal áll relációban, és ez fordítva is igaz.

[2] Mik azok az adatbázisok?
https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-are-databases/?cdn=disable
(Letöltés ideje:2023.05.10.)
[3] Az alábbi oldalon olvasható az idézet szöveg:
https://webiskola.hu/sql-ismeretek/relacios-adatbazis-sql-fogalmak-peldak/
(Letöltés ideje: 2023.05.07.)

1.2.b) Front-end

A front-end megfogalmazása a következő lenne: ,,A *front-end* (néha *frontend* vagy *front end* formában is írják) a programoknak**,** weboldalaknak az a része, amelyik a felhasználóval közvetlenül kapcsolatban van**.** Feladata az adatok megjelenése, befogadása a felhasználó (vagy ritkábban egy másik rendszer) felől."[4]
Keretrendszerként a React-ot alkalmaztuk, ahol JavaScript programozási nyelvvel valósítottuk meg az interaktív weboldalunk létrehozását. A böngésző a forrás kódban szereplő nyelvet letölti, majd értelmezi és futtatja. A formázásokat a CSS[5] azaz a „Cascading Style Sheet" használata által hajtottuk végre. Ez az ami a weboldal kinézetéért felel, ami az úgynevezett külalak megjelenítésért felel.

A HTML kódok ismerete által valósítható meg a fent nevezett Js, valamint CSS, mivel ezen elemekkel kapcsolódnak össze.

A teljesen reszponzív webes frontendről elmondható, hogy standard technológiákkal készült, valamint nagyon hasznos, mert támogatja az összes elterjedt/ismert böngésző használatát, mind az asztali gépeken és mobil eszközökön egyaránt. A front-end létrehozása során igyekeztünk megragadni azokat a látvány elemeket, amelyek szükségesek egy ilyen verseny még vonzóbbá tételéhez.

1.2.c) Back-end

A backend fogalma az alábbi lenne: ,,A *back-end* (néha *backend* vagy *back end* formában is írják) a programoknak**,** weboldalaknak a hátsó, a felhasználó elől rejtett, a tényleges számításokat végző része**.** Feladata a front-end (a felhasználóval kapcsolatban lévő rész) felől érkező adatok feldolgozása, és az eredményeknek a front-end felé történő visszajuttatása."[6]

---

[4] Az alábbi oldalon olvasható az idézet szöveg:
https://lexiq.hu/front-end
(Letöltés ideje: 2023.05.08.)
[5] Mi az a CSS? A CSS bemutatása
https://webiskola.hu/css-ismeretek/mi-az-a-css-a-css-bemutatasa/
(Letöltés ideje: 2023.05.09.)
[6] Az alábbi oldalon olvasható az idézett szöveg:
https://lexiq.hu/back-end
(Letöltés ideje:2023.05.09.)

Megvalósításához a laravel[7] keretrendszert alkalmaztuk, amely egy nyílt forráskódú PHP rendszer. Gyorsította a folyamatunkat, biztonságosan lehet vele dolgozni. Az adatok gond nélkül migrálhatóak a fejlesztés során. Jelszavak létrehozása során biztonságosnak tekinthető, mivel PHP kódot használnak az SQL kód helyett. Így az SQL injekciós támadásokkal[8] szemben egy védettebb felületet kapunk.
A Back-end létrehozása során igyekeztünk olyan logikai struktúrát kialakítani, amely stabil működést eredményez a háttérben.

1.2.d) Alkalmazás

Megalkotásához a Visual Studio[9] fejlesztőkörnyezetet használtuk, ahol a grafikus tervező felület került kiválasztásra. A sablonok közül a Windows Forms Application-t választottuk, és a Toolbox segítésével, különböző komponenseket helyeztünk el az űrlapon. Beállítottuk a szükséges tulajdonságokat, funkciókat és az eseményvezérlés elvét követtük a különböző műveletek során. Alapvető input-output műveleteket végeztünk a C# nyelvben használt változó típusokkal. Definiáltunk konstansokat, függvényeket alkottunk és osztályokat hoztunk étre. Ezen kifejezések a számítógép kiértékeli, az utasításokat végrehajtja majd az eredményeket felhasználja a precedencia elv mentén.

1.3 Használati útmutató

A különböző felületek használatához egy alap szintű ismerettel kell rendelkezni a felhasználónak. Az alábbi programok telepítését javaslom az áttekintésük során: Visual Studio,

[7] Laravel (angol)
https://laravel.com/
(Letöltés ideje: 2023.05.09.)
[8] Az egyik leggyakoribb támadási forma a weboldalak ellen. Többféle adat megszerzésére irányul, az SQL adatbázisunkból szeretne adatot, információhoz hozzájutni a támadó fél.
[9] Visual Studio (angol)
https://visualstudio.microsoft.com/
(Letöltés ideje: 2023.05.09.)

PhpStorm[10], Xampp[11], React[12], Laravel. A Visual Studio, a PhpStorm és a Xampp telepítése a klasszikus weboldalakon a letöltés gombbal alkalmazhatóak, figyelni kell arra, hogy milyen operációs rendszerrel rendelkezik a számítógépünk, és milyen feltételeket, követelményeket határoz meg az adott program telepítése során. A továbbiakban a két keretrendszerre térnék ki, mivel ezek azért specifikáltabb esetek.

A React telepítéséhez feltételezem, hogy már a Node telepítve van. A következő lépeseket kell megtenni:

- Hozz létre egy könyvtárat a React projektednek.
- Lépj be a könyvtárba
- Indítsd el a parancssort (vagy terminált, a PHP Storm-ban), itt most a Windows parancssort alkalmazzuk!
- A következő parancsot kell kiadni: npx create-react app
- Amennyiben a Node fel van telepítve a következő üzenet fog megjelenni: „Happy Hacking!”
- Az npm start paranccsal indítsd el a programot.
- Az ENTER billentyű lenyomása után elindul az alapértelmezett böngészőben az alkalmazás a localhost:3000 címen.

A Laravel telepítése következőképp néz ki:

- Laravel letöltése
- Készíts egy új mappát a gépeden, vagy ha helyi webszervert használsz akkor nyisd meg a gyökérkönyvtárát.
- A terminálba írd be a következő parancsot: composer create-project --prefer-dist laravel/laravel mappacska

[10] PhpStorm (angol)
https://www.jetbrains.com/phpstorm/promo/?source=google&medium=cpc&campaign=14335686429&term=phpstorm&content=604081944634&gad=1&gclid=CjwKCAjwx_eiBhBGEiwA15gLN_juVBCEQbOJnHXGIASuBoX3cvavuWZ0q0C5HkaB7_B9Iu3RzqyWaBoCprkQAvD_BwE
(Letöltés ideje: 2023.05.09.)
[11] Mi a Xampp?
https://www.apachefriends.org/hu/index.html
(Letöltés ideje: 2023.05.09.)
[12] React
https://react.dev/
(Letöltés ideje:2023.05.09.)

- Megjegyzem, hogy a fenti parancs létrehozza a mappacska mappában az első laravel projektet.
- Ha elkészült nyisd meg IDE-dben.
- A projektedet a .env kiterjesztésű tejtett fájlban tudod konfigurálni. Itt tudod megadni az elrési útvonalat, adatbázishoz történő kapcsolódást stb. Legelső alkalommal nincs még meg ez a fájl, ezt létre kell hozni .env.example másolásával.
- A projektet a PHP-ban elindítod, a terminálban kiadott php artisan serve paranccsal.
- A böngészőben a következő címet nyisd meg:http://127.0.0.1:8000

Javaslom, hogy mindig ellenőrizzük le, hogy minden karaktert a megfelelően írtunk be, illetve a lépéseket ne cseréljük fel, mert itt minden egymás után következik. Nagyon fontos a sorrendiség, és az hogy a kezdetben meghatározott programok elérhetőek legyenek, mert a két keretrendszer működéséhez ez elengedhetetlen. Alaposan és körültekintően járjunk el, figyeljünk oda, hogy kipihent állapotban hajtsuk végre a fent meghatározottakat.

2. Fejezet Fejlesztésünk a célkeresztben

2.1.Munkamegosztásunk meghatározása, részvételünk a csapatban

A folyamatok kialakítása során igyekeztünk egymás között egy – egy területet külön létrehozni. Az adatbázist együtt készítettük el, ahol kezdetben ábrákkal próbáltuk meg szemléltetni a kapcsolatok, meg szerettük volna érteni az elsődleges és az idegen kulcsokat, és ezek általi összefüggéseket, a versenyhez szükséges elemeket alakítottuk ki. Mindig lépésről lépésre haladtunk, és ha éppen hibába ütköztünk, igyekeztük együtt megbeszélni, átlátni és együtt megtalálni a megoldást.

A feladatok megoldása a következőként nézett ki:

- Mónika a C#-os alkalmazás létrehozását látta el.
- Csilla a Back-end megalkotását tudhatta magáénak.
- Tünde pedig a Front-end kialakításában és megálmodásában játszott jelentős szerepet.

Összehangolt tevékenység folyt a projektmunka kialakítása során, ha nem volt világos, mi az amit szeretnénk, akkor egymáshoz fordultunk és igyekeztünk tisztázni mit akarunk megcsinálni. A különböző területeket együtt leellenőriztük, a kódokat mindannyian megbeszéltük, kiegészítettük egymás területét, javaslatokat fogalmaztunk meg, vagy éppen hozzá építettünk egy – egy fontosabbnak vélt elemet. A csapatunk minden tagja maximálisan kivette a részét egymás munkája során is, javítottuk egymás hibáját ha ez éppen szükséges volt. A kommunikáció során teljes mértékben odafigyeltünk egymás kérésére, igényére, és igyekeztünk szabadidőnkben tartósan megbeszéléseket tartani. Minden héten tájékoztatót tartottunk az aktuális helyzetünkről, és különböző felületeken mutattuk meg ki hol tart éppen. Munkánk során használtuk a teams-t, a google meet csatornáját, a discord-on is folytattunk tanulásunkat elősegítő konzultációt. A github felületét is állandóan használtuk, és a csapatunk tagjai, valamint a konzulensünk is nyomon követhette fejlődésünket, ellenőrizte hogyan tevékenykedünk hétről hétre hogyan haladunk előre.
Összességében egy hihetetlenül jó szövetséget alkottunk, eredményes zárással.

2.2. Telepítési folyamatok ismertetése

Az alkalmazások elindításához szükséges szoftverek az alábbiak lennének: Xamp, PHP storm, Visual studio. Elindításához szükséges Xamp, az adatbázishoz egy Verseny táblát kell

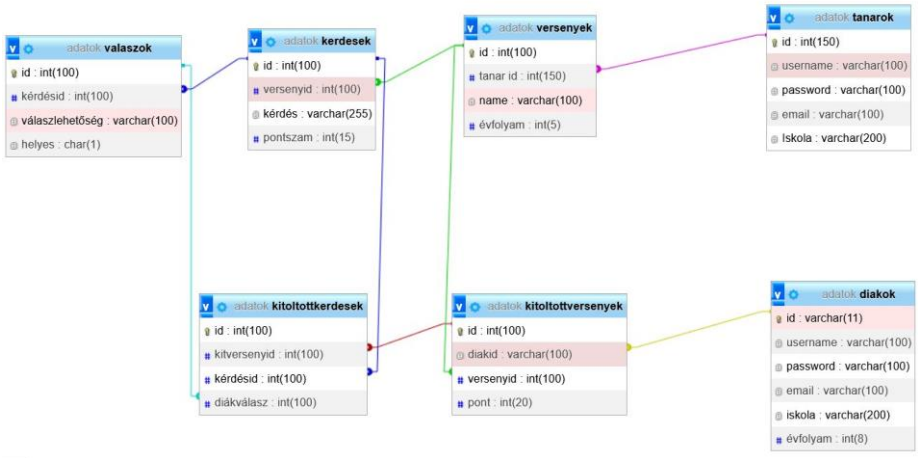
létrehozni, és beimportálni az általunk elkészítettet sql adatbázist. Back-end- service-ek Apinál a következő szükséges: php artisan migrate: fresh parancs beírása, tesztadat betöltése TestAdat.sql file-ból, majd az Api indítása: php aritsan serve
Front-end esetén, verseny mappába navigálás, szükséges package telepítése.

2.3.Adatbázisunk részletes elemeinek bemutatása

A következő ábrán szemléltetném a főbb kapcsolatokat. Minden tábla rendelkezik id-vel, mivel ez számokat jelöl ezért int-ként definiáltuk, kivételt képez ez alól az adatok tábla diákok felirata ahol varchar található, mivel ők nem csak számokat adhatnak meg. Az adatok válaszok és az adatok kérdések táblát a kérdés id és az id köti össze a következő táblával és így tovább kapcsolódnak egymáshoz az adatok kérdések verseny id-ja és az adatok versenyek id-ja, valamint az adatok versenyek id-ja a tanár id-val. Az adatok válaszok táblája az adatok kitöltött kérdéseknél az id-val és a diák válaszával áll kapcsolatban. Az adatok kérdések id-ja a az adatok kitöltött kérdések kérdés id-val kapcsolódik egymáshoz. Az adatok kitöltött kérdések kitöltött verseny id-ja az adatok kitöltött versenyek id-val függ össze. Az adatok verseny id-ja a verseny id-val áll szorosabb viszonyban, és végül az adatok kitöltött versenyek diák id-ja az adatok diákok id-val hozható összefüggésbe. Az alábbi pontban található, hogyan terveztük ezt meg, és hogyan néz ki az imént ismertetett leírásom.

2.3.a) Adatbázisunk grafikusan történő ábrázolása

Kapcsolatok ábrázolás főbb pontjai:



Adatbázis kapcsolatok kép

2.3.b) A mezők kialakítása, strukturálása

A következő táblákat tartalmazza végül: diakok, kerdesek, kitoltottkerdesek, kitoltottversenyek, migrations, personal_access_tokens, tanarok, versenyek.
A diakok tábla szerkezete a következően néz ki:

- Van egy id-ja elsődleges kulccsal ellátva, bigint típussal.
- Az id-hoz tartozik egy username idegen kulccsal, vacrhar típussal.
- A password, az email, a fullname, a school, a class, mező típusa szintén varchar.
- A created_at és az update_at timestap adattípusként jelenik meg, mivel ezzel a további adatmódosítás elkerülhető.

A kerdesek tábla szerkezete a következően épül fel:

- Van egy id-ja, elsődleges kulccsal ellátva, bigint típussal.
- A competitionId, question, Answer1, Answer2, Answer3, Answer4 mezők varchar típussal rendelkezik.
- A correctAnswer mező típusa pedig int.
  Szintén a created_at és az update_at timestap adattípusként jelenik meg, mivel ezzel a további adatmódosítás elkerülhető.
  A kitoltottkerdesek tábla szerkezetétnek ismertetése az alábbi lenne:
- Az id-ja bigint típusú, és elsődleges kulccsal van ellátva.
- A submittedCompetitionId, a questionId, studentAnswer mezők int típussal van ellátva.
- Ugyancsak a created_at és az update_at timestap adattípusként jelenik meg, mivel ezzel a további adatmódosítás elkerülhető.
  A kitoltottversenyek szerkezete a soron következő elemekből áll:
- Az id-ja itt is elsődleges kulcsként szerepel bigint típussal.
- A studentId, competitionId mező varchar típusú.
- Egyaránt a created_at és az update_at timestap adattípusként jelenik meg, mivel ezzel a további adatmódosítás elkerülhető.

A migrations tábla az alábbiakat tartalmazza:

- Int típusú id-val rendelkezik, amely elsődleges kulcs.
- A migration mező varchar típusú.
- A batch mező pedig int típusú.

A personal_access_tokens tábla felépítése a következő lenne:

- Elsődleges kulccsal ellátott id-ja, bigint típussal.
- Másodlagos kulccsal ellátott tokanable_type, varchar típussal.
- Továbbá másodlagos kulccsal van hozzárendelve a tokanable_id-hoz, bigint típussal.
- A soros name mezőhöz varchar típus tartozik.
- A token szintén másodlagos kulccsal bír, varchar típussal.
- Az abilities mező text típussal rendelkezik.
- Itti is megjelenik a a created_at és az update_at valamint a last_used_at és az experies_at timestap adattípusként, mivel ezzel a további adatmódosítás elkerülhető.

A tanárok tábla az alábbiakat foglalja magába:

- Az id-ja elsődleges kulccsal van ellátva, bigint típusként.
- A username másodlagos kulcsként szerepel, varchar típusként.
- A password, az email, a fullname, a subject, a class, mező típusa szintén varchar.
- A created_at és az update_at timestap adattípusként jelenik meg, mivel ezzel a további adatmódosítás elkerülhető.

A versenyek tábla szerkezete a következőképpen néz ki:

- Az id-ja elsődleges kulccsal van ellátva, bigint típusként.
- A competition_name és a description mező varchar típusként szerepel.
- Ugyancsak a created_at és az update_at timestap adattípusként jelenik meg, mivel ezzel a további adatmódosítás elkerülhető.

Összességében az alábbi állításokkal szeretnénk implementálni az fenti szerkezetet:
A táblák adatokat tartalmaznak, amelyekkel a később lekérdezéseket tudunk végrehajtani. Az oszlopok nevei létrehozásánál ügyeltünk arra, hogy ne tartalmazzanak duplikációt, adattípus eltérés esetén sem. A szintakti kialakításánál az oszlopok deklarációja során megadtuk az adattípusokat, mivel ez egy kötelező elem.

A szöveg adattípust nálunk a varchar amely változó hosszúságú karakterláncot jelöl, valamint a text amely szöveg tárolására képes. A szövegnél tulajdonképpen string-ként tárolódnak el az adatok.

Szám adattípusként használtuk az int és a bigint típusokat. Az int közepes méretű egész szám tárolását foglalja magába, a bigint egy nagyobb tartomány jelöl, minden másban megegyezik egyénként az int-tel.

A dátum és idő adattípusként a timestamp-et alkalmaztuk, ez az úgynevezett időbélyeg.
Az alábbi korlátozások azok, amelyeket alkalmaztunk, és ki lehet olvasni az adatbázis tábláinkból:
NOT NULL kifejezés esetén a mező értéke nem lehet nulla, vagyis üres.
PRIMARY KEY azaz elsődleges kulcs esetén egyértelműen beazonosíthatóvá válik a rekordunk.
FOREIGN KEY vagyis másodlagos/idegen kulcs esetén egy másik tábla elsődleges kulcsaként hivatkozunk.

A projektünk alapja tehát a jól felépített adatbázisunk, hiszen az adatok, amelyek a weboldalon szereplő verseny kitöltéséhez, regisztrálásához, belépéséhez stb. valamint az alkalmazásban a kérdések megadásához, a tanár felület elérhetővé tételéhez mind – mind ezekben a táblákban tárolódnak, a megadott mezőkkel és kapcsolatokkal együtt jöhetnek létre.

2.4. Fontosabb kapcsolatok közötti összefüggések áttekintése

A c# programnyelven[13] készített felülethez a Visual Studio grafikus fejlesztői környezetet használtuk. Tartalmaz egy komponens könyvtárat, amelyet a Toolboxon érhetünk el, vagy létrehozhatunk több komponenseket is. A következő leírásban feltárom a főbb komponenseket, fontosabb tulajdonságait az eseményeivel kapcsolatosan. A leggyakrabban használt elemeket fogom bemutatni és leírni. A Toolbox által az űrlapra helyeztük a következőket, Button, Textbox, Label.

A button vagyis a gomb tulajdonságai az alábbiak lennének:

- Text vagyis a gomb felirata.
- Image azaz a gombon lévő kép.
- ImageAlign a kép elhelyezkedését jelöli a gombon belül.
- A Click a gomb megnyomásakor következik be.

[13] Illés Zoltán: Programozás C# nyelven. Jedlik Oktatási Stúdió, Budapest, 2005.

A szövegdoboz tulajdonságai az alábbiak lennének:

- Text vagy a szövegdobozban szereplő szöveg itt állítható be, olvasható el.
- Multiline amelynek az értéke ha igaz több sor szöveget is írhatunk bele a szövegdobozunkba.
- UseSystemPasswordChar amelynek az értéka ha igaz, akkor a szöveg amit gépelünk, adatként beviszünk, annak helyére ezzel párhuzamosan egy karakter jelenik meg, így nem lehet kiolvasni a jelszót olyan egyszerűen.
- TextChanged akkor következik be változás, ha a szövegdoboz szövege megváltozik.

Kiegészítve a Labellel, amely olyan vezérlő amit a leggyakrabban használunk. A fő célja a szöveg megjelenítése. A leggyakrabban használt esemény egyébként az egérkattintás vagyis a „Click”.
A felhasználó a számítások során kapott eredmények a képernyőn/kijelzőn látja meg(output), a program futása esetén az input értékeket, adatokat kérjük be.

Azt a tartalmat amely a változhat a program futása közben, változónak nevezzük. A változókra az alábbi szabályok érvényeresek:

- névvel/azonosítóval
- típussal
- tartalommal
- aktuális értékkel
- hatókörrel
- élettartammal ellátott.

A változóknak egyik különleges típusa a literál, amelyet akkor használunk, amikor programunkban egy konkrét értéket szeretnénk igénybe venni. Továbbá állandókat is definiáltunk azaz konstansokat.
Az iteráció által ugyanazt a tevékenységi sort hajtottuk végre, ilyenkor használtunk ciklusokat. Itt többféle ciklus alkalmazása volt adott elől és hátul tesztelős ciklus. A foreach-re szeretnénk most külön kitérni, mivel ez elég különleges és többször, több helyen is előfordult. Ezt akkor használjuk, ha valamilyen összetett adattípus elemeit egytől egyig fel akarjuk dolgozni.

Feltételes ciklust akkor használunk, ha nem ismerjük az ismétlések számát, ebben az esetben pedig egyetlen feltételtől tehetjük függővé az ismétlések számát. Ezt meg is tettük és alkalmaztuk is.

Az .xaml kiterjesztéssel rendelkező elemekhez tartozó .xaml.cs amely namespaceként globálisan összefogja a kódokat, így átláthatóbb, letisztultabb formát kapunk a megértésükben.

Majd ezen belül kerül létrehozásra az osztály, amely átadja a benne szereplő utasításokat, illetve meghívja a különböző eljárásokat és függvényeket.

Az alábbiakban bemutatok kód részleteket, amelyeket megjelöltünk és magyarázattal is ellátunk, ez nem a teljes program bemutatása csak egyes elemeit szeretném szemléltetni a megértésükben, működésükben:

Itt történik meg a navigálás az új oldalra, ahol a versenyt lehet létrehozni:

```
private void NewCompetition_Click(object sender, RoutedEventArgs e)
    {
        mainFrame.Navigate(new CreateCompetitionPage(mainFrame, username));
        }
```

Itt történik a verseny api meghívása:

```
HttpResponseMessage response = client.GetAsync("verseny/tanar/list").Result;
```

Meghívjuk a bejelentkezés API-t:

```
await client.DeleteAsync($"verseny/tanar/delete/{versenyId}");
```

Hibaüzenet kiírása:

```
if (!body.Status.Equals("OK"))
    {

        Console.WriteLine(body.Message);
```

```
            return;
                }
```

Egy tetszőleges konstruktor eleje és vége:

```
        public LoginPage(Frame navigationFrame)
    {
        InitializeComponent();
        client.BaseAddress = new Uri("http://127.0.0.1:8000/api/");
        mainFrame = navigationFrame;

    }
```

A kérés body-ját az alábbi kódrészlet tartalmazza:

```
private async void Login_Click(object sender, RoutedEventArgs e)
    {
            var loginData = new
        {
            username = UserTextBox.Text,
            password = PasswordTextBox.Text,
        };
        string jsonData = JsonConvert.SerializeObject(loginData);
        var content = new StringContent(jsonData.ToString(), Encoding.UTF8,
"application/json");

        // Meghívjuk a bejelentkezés API-t
        HttpResponseMessage response = client.PostAsync("tanarok/login", content).Result;
        if (!response.IsSuccessStatusCode)
        {
            Console.WriteLine(response);
            MessageLabel.Content = "Bejelentkezés sikertelen, ismeretlen hiba.";
            return;
```

```
        }

        var responseBody = await response.Content.ReadAsStringAsync();
        var body = JsonConvert.DeserializeObject<LoginResponse>(responseBody);

        if (!body.Status.Equals("OK")) {
            // Kiírjuk a hibaüzenetet a response-ból
            MessageLabel.Content = body.Message;
            return;
        }

        // Sikeres bejelentkezés
        MessageLabel.Content = body.Message;



        //Itt kell átirányítani a verseny oldalra
        mainFrame.Navigate(new CompetitionList(mainFrame, UserTextBox.Text));

    }
```

A felhasználó létrehozása és regisztrációs folyamat egy része a gomb kattintással:

```
    private async void CreateUser_Click(object sender, RoutedEventArgs e)
    {
        // Létrehozzuk a request body-t
        var registrationData = new
        {
            username = UserNameTextBox.Text,
            password = PasswordTextBox.Text,
            email = EmailTextBox.Text,
            fullname = FullnameTextBox.Text,
            subject = "none",
            @class = "none"
        };
```

```
        string jsonData = JsonConvert.SerializeObject(registrationData);
        var content = new StringContent(jsonData.ToString(), Encoding.UTF8,
"application/json");

        // Meghívjuk a regisztráció API-t
        HttpResponseMessage response = client.PostAsync("tanarok/signup", content).Result;
        if (!response.IsSuccessStatusCode)
        {
            Console.WriteLine(response);
            MessageLabel.Content = "Regisztráció sikertelen. Ismeretlen hiba.";
        }

        var responseBody = await response.Content.ReadAsStringAsync();
        var body = JsonConvert.DeserializeObject<RegistrationResponse>(responseBody);

        if (!body.Status.Equals("OK"))
        {
            // Kiírjuk a hibaüzenetet a response-ból
            MessageLabel.Content = body.Message;
            return;
        }

        // Sikeres regisztráció
        MessageLabel.Content = body.Message;
        //mainFrame.Navigate(new LoginPage(mainFrame));

    }
```

Az említett kód részletekkel szerettünk volna egy – egy folyamatot átadni, amelyet a vizsgamunkánk során kiemelten fontosnak tartottunk az alkalmazás megalkotása során. Ezek olyan folyamatok és elemek, amelyek egymásra épülnek, és a szerkezetük megértéséhez egy – egy sajátos algoritmikus gondolkodás szükséges. Talán így jobban megérti az olvasó, mik azok a legfontosabb elemek amelyre épül.

2.5.Alkotásunk komponens szegmensének szemléltetése

A Front-end felülethez használt fejlesztői környezet a React volt, amelynél az alábbi fontosabb elemeket szeretnénk bemutatni:
A Components menu mappában került létrehozásra a MenutItems.js, Navbar.js, és a NavbarStyles.css kiterjesztésű fájl. Az export kifejezéssel egy másik Js fájlban tudom majd használni a const-ként megjelölt MenuItsems elemeit, amelyek a következők lennének: kezdőlap, verseny, diákok, tanárok, kapcsolat. Itt kapják meg a title-t(azaz a címet) valamint az url-t az elérési útvonalban szereplő nevüket. Kód részlete így néz ki:

```
{
        title:”verseny”,
        url:”/verseny”
}
```

A Navbar.js-ben először importálni szükséges azokat az elemeket, amelyeket fontosnak tartok, ilyen például a css, így tudja ezt gyakorlatilag használni. A **from** kulcsszó után sztringként kell megadni az elérési utat. Tehát amit használni szeretnék az az importálással lehetséges, itt oda kell figyelni, hogy melyik mappában szerepel, hogy írom le stb., hogy a későbbiekben ne okozzon validációs problémát.
Mint például:

```
import ”./NavbarStyles.css”;
import {MenuItems} from ”./MenuItems”;
```

A Javascript megírása során kötelezően be kell írni a függvények elé a function kulcsszót, amelyet közvetlenül a függvény neve fog követni.
Korábban[14] csak a var kulcsszóval lehetett változót deklarálni. Azonban most már a const és a let is használatos. A const-ot később sem szokás felülírni, inkább állandó típusként tekintünk rá, a let pedig egy blokk hatókörében érvényesül.

A változókkal kapcsolatos szabály a következő lenne:

- A nevének kötelezően egyedinek lennie!

[14] 2015 előtti időszakban.

- A neve a hosszára vonatkozó kötelező jelleg, hogy minimum 1 karakterből kell állnia vagy akár több karakterből is állhatnak.
- A nevei az alábbi karaktereket tartalmazhatják: számok, aláhúzások és dollárjelek.
- Nevei létrehozása estén vagy csak karakterrel, vagy aláhúzással vagy dollárjellel kezdődhetnek, **számmal pedig egyáltalán nem.**
- Tudniillik, hogy a nevek kis- és nagybetűre érzékenyek (tehát a példa kedvéért: a “JS” és a “js” két különböző változó)
- Vannak előre lefoglalt, kötött kulcsszavai a JavaScriptnek, amelyeket nem lehet használni (például let, var, const, if stb.)
-

A HTML leíró nyelv során tanultakat is alkalmazzuk az alábbi részletben:

```
<div>
        <nav className="NavbarItems">
          <h1 className="navbar-logo">Verseny</h1>

          <ul className="nav-menu">
            {MenuItems.map((item, index) => {
              return (
                <li key={index}>
                  <Link className="nav-links" to={item.url}>
                    <i className={item.icon}></i>{item.title}
                  </Link>
                </li>
              )
            })}

            {button}
          </ul>
          <div className="logintext mr-2">
            {loginMessage}
          </div>
        </nav>

      </div>
```

Itt csak szerettük volna szemléltetni, hogyan is valósul meg a gyakorlatban a JavaScript és a html együttes szerepe. A kettő terület kéz a kézben jár, mindkettő teljes ismerete szükséges volt a feladat megalkotásához.

A NavbarStyles.css során állítjuk be kinézete, különböző interakciókat a felhasználói felületen. Néhányat említenék: háttér szín, magasság, árnyékolás, margó, igazítások, betűk és stílusok, dekoráció, betűszín és a többi. Számos beállítási mód lehetséges, ezek részletezésre most nem térnénk ki, mert amit csak tudunk itt meg lehet valósítani a legapróbb részletekig, ha szeretnénk. Megjegyzem, hogy az Assets mappában található a jegvirag1.jpg amelyet háttérképként alkalmaztunk, a design kedvéért. Ez jpg ingyenesen használható és elérhető.

A következő nagyobb részünk a message mappa ahol 10 darab Js fájl található, valamint egy bejelentkezes.css fájl szerepel.
A bejelentkezes.js során megtörténik a szükséges elemek importálása, majd ezt követően a function kulcsszóval a függvény hívása. Ez a felület tartalmazza azokat a metódusokat, amelyek a bejelentkezéshez szükségesek. Itt a következő történik a **fetch-el** JavaScript függvényt hívjuk meg, aminek paraméterként átadásra került az API linkje, aminek a végén az a nagy JSON adat található. A fetch egy promise-t ad majd nekünk vissza.
Kódokban a példánk:

```
function Bejelentkezes() {
    const [username, setUsername] = useState(null);
    const [password, setPassword] = useState(null);

    const navigate = useNavigate();

    const login = (e) => {
        e.preventDefault();
        fetch('http://127.0.0.1:8000/api/diakok/login', {
            method: "POST",
            headers: { "content-type": "application/json" },
            body: JSON.stringify({
                "username": username,
                "password": password
            })
```

```
    })
    .then(response => {
        return response.json();
    })
        .then(response => {
        // Sikeres bejelentkezés esetén eltároljuk a userId-t
        if (response.status === 'OK') {
            localStorage.setItem("userId", response.id);
            localStorage.setItem("userName", username);
            navigate("/verseny");
        }
        else {
            //Hiba esetén kiírjuk a hibaüzenetet
            alert(response.message);
        }
    })
    .catch(error => {
        console.log(error.message);
    })

}
```

A diakok, a tanarok, a kezdolap a már fent említett egyszerűbb kódokkal rendelkezik, részleteire nem térnék ki újra.

A kapcsolat.js function-ben találhatóak a következő const-ok: név, email, tárgy, üzenet. Nyilván ez az a felület ahol a felhasználó tud küldeni egy szöveges üzenetet a nevével és az email címével ellátva. Fontosnak tartottuk, hogy két oldalú kommunikációt valósítsunk meg, mivel ha valamilyen nem várt esemény, vagy hiba következne be, akkor is elérhetővé tudjunk válni azok számára, akik számítanak ránk. Ez talán így egy kicsit komolyabb és minőségibb oldalra utal, mivel egy felületen keresztül válunk elérhetővé. Az email ugyanígy egy opció lehetne, de az elég egyszerű lenne. Így modernebb és azonnal elérhető, könnyebb ide beírni bármit, mintsem belépni az email fiókunkba, ahol sok információ lefoglalhat bennünket a napi

tevékenységünk során. Tehát ez egy gyorsan elérhető, azonnali megoldás lehetőségét nyújtja annak, aki a weboldalunkra látogat.

A kerdes.js-ban történik meg a kérdések kitöltése, és a válaszok tovább küldése a verseny-form.js-nek. A function hívja meg azokat a függvényeket, amelyek a const-ban szerepelnek. A let által behatároltuk a válaszok számát. Külön kitértünk a rossz válasz megadására is. A további html leíró nyelv elemeként div-ekként helyezzük el, mint például gombok, valamint jó és rossz válaszok.
A regisztracio.js az ahol a regisztrációs folyamatok mennek végbe.
A const-ok azok a változók amelyek a következőket tartalmazzák: felhasználó név, jelszó, teljes név, email, iskola és osztály. Ezek azok, amelyeket az adatbázissal kapcsolunk majd össze, amelyeket korábban már említettem. Tehát ez az amit a felhasználó lát, amikor éppen a mi oldalunkra beregisztrál. A regisztrációt a fetch-en keresztül hajtja végre, és sikeres létrehozás esetén tovább navigál a bejelentkezés felületre.

A versenyform.js van lehetőségünk beküldeni a válaszokat, itt lehet hozzá rendelni a kérdésekhez a pontszámokat, és vissza is léphetünk. Ha nem vagyunk bejelentkezve, akkor átnavigál minket a bejelentkezési oldalra.

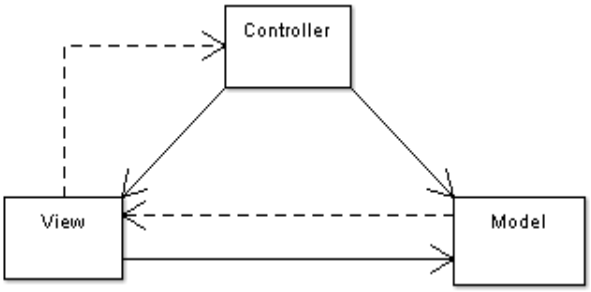
A verseny-items.js ahol az eredmény és a kitöltés valósul meg, visszavezetve a versenyhez.

Az App.js talán az egyik legfontosabb elem, hiszen itt történik meg az importálása a függvényeknek, gyakorlatilag itt hívjuk meg, és a Route során tud megjelenni a böngésző url-jében az általunk megnevezett/meghatározott oldalunk.

Az index.js az ahol a böngésző gyökér könyvtárát rendereljük az App-unkhoz.

A front-end elemi igazán meghatározzák egymást, megírása során kellő figyelemmel és alapossággal kell eljárnunk, mert a megírása során minden részletre oda kell figyelnünk, megfelelően kell alkalmaznunk az importálásokat, meghatározni az elérési útvonalakat. Véleményem szerint itt minden mindennel összefügg.

2.6.Az API főbb összetevőinek kibontása



A fenti ábrával szeretnénk bemutatni, hogy működik a folyamata a controllernek, modellnek, felhasználói felületnek. A controller a vezérlői szerkezet másnéven, és azért felelős, hogy válaszoljon a felhasználói imputokra, és módosításokat hajt végre a modellekben. Gyakorlatilag feldolgozzák az adatokat, és kimenő adatot szolgáltatnak a felhasználó számára. A controllerben megírt metódusok döntik el, hogy melyik felületet adja vissza, melyik html fog éppen rendelődni.

A back-end során a legfontosabb elemekre térnénk ki kezdetben a controllersekre, valamint a modelsekre. A http mappában található a Contorllers mappa, és azon belül került kialakításra a Controller.php, a DiakController.php, a KerdesController.php, a TanarController.php, a VersenyDiakController.php valamint a VersenyTanarController.php fájl.
Ezekben a rétegekben történik meg az adatvalidációk, illetve innen indul ki az adatbázisműveletek is. A két legfontosabb elem a kommunikáció során a GET és a POST request lenne.

Először egy POST kéréssel indítunk, amely az új diákot regisztrálja, ellenőrzi a hiányzó adatokat, valamint ellenőrzi, hogy létezik e már a megadott felhaszálónév. Vagyis a DiakController osztályban történik meg a function függvény meghívása, amely a bejelentkezés kérést validálja. A validációs kérések az alábbiakra irányulnak: felhasználónév, jelszó, teljes név, email, iskola, osztély. Amennyiben ez hibás (fails) akkor kiírja üzenet formájában ezt a szöveget: Hiányzó adatok a regisztrációban.

A következő folyamatban érkezik egy kérés létező felhasználóvére és amennyiben foglalt az az üzenet jelenik meg a felhasználó számára: Ez a felhasználónév már foglalt.
Az új diák beszúrása a következőképpen néz ki:
Diak::create($request->all());
Ekkor visszatért (return) az alábbi üzenettel: Sikeres regisztráció.

A következő POST kérésnél történik meg a diák bejelentkeztetése; ellenőrzi az adatbázisban a felhasználónév-jelszó párost, találat esetén visszaadja a diák azonosítóját.
Ez a function általi login függvény hívás esetén a következőképpen néz ki:
A request-ből kivesszük a szükséges két elemet, majd megpróbáljuk megtalálni a diákot a username-password alapján.

Ez így néz ki:

```
$diak = Diak::query() -> where([
        ['username', '=',  $username],
        ['password', '=',  $password]])->first();
```

Amennyiben nem találja meg hibát fog eredményezni, és a következő üzenet jelenik meg: Hibás felhasználónév vagy jelszó.
Amennyiben sikeres a bejelentkezés, akkor returnként visszaadja az értékét, és a következő üzenet jelenik meg: Sikeres bejelentkezés.

A KerdesController.php egy GET kéréssel indít, ahol a kérdések kilistázása történik meg egy adott versenyhez, valamint a verseny azonosító paraméterben történő megadás is ebben a folyamatban szerepel. A kérdés listázása esetén, egy queri lekérés irányul a competitonId-hoz, a versenyId-nál. Majd visszatér a státusz OK értékével.

Új kérdés indítása esetén meg kell adni a verseny azonosítót, a válaszlehetőségeket, valamint a helyes választ is. Először validáljuk a kérdéseket és a hozzá megkreált válaszokat, majd failsként ha hiányos, a következő üzenet jelenik meg: Hiányzó adatok a kérdés létrehozásához. Amennyiben minden rendben volt, visszatér a következő üzenettel: A kérdés sikeresen létrehozva.

Az úgynevezett PUT kéréssel indítjuk a kérdés módosítását.
Itt az előző folyamathoz hasonlóan történik validáljuk a kérdéseket és a hozzá létrehozott válaszokat, majd failsként ha hiányos, a következő üzenet jelenik meg: Hiányzó adatok a kérdés módosításához. Amennyiben minden rendben volt, visszatér a következő üzenettel: Sikeres kérdés módosítás.

A DELETE kéréssel történik a kérdés törlése. A message pedig a következő lesz: A kérdés sikeresen törölve.

A következő POST kérésnél történik meg az új tanár regisztrációja, a hiányzó adatokat ellenőrzése, továbbá, hogy létezik e már a megadott felhasználónév. Vagyis a TanarController osztályban történik meg a function függvény meghívása, amely a bejelentkezés kérést validálja. A validációs kérések az alábbiakra irányulnak: felhasználónév, jelszó, teljes név, email, iskola, osztály. Amennyiben ez hibás (fails) akkor kiírja üzenet formájában ezt a szöveget: Hiányzó adatok a regisztrációban.
A következő folyamatban érkezik egy kérés létező felhasználóvére és amennyiben foglalt az az üzenet jelenik meg a felhasználó számára: Ez a felhasználónév már foglalt.

Az új tanár beszúrása a következőképpen néz ki:
Tanar::create($request->all());
Ekkor visszatért (return) az alábbi üzenettel: Sikeres regisztráció.

A következő POST kérésnél történik meg a tanár bejelentkeztetése; ellenőrzi az adatbázisban a felhasználónév-jelszó párost, találat esetén visszaadja a tanár azonosítóját.
Ez a function általi login függvény hívás esetén a következőképpen néz ki:
A request-ből kivesszük a szükséges két elemet, majd megpróbáljuk megtalálni a tanárt a username-password alapján.

Ez így néz ki:

```
$tanar = Tanar::query() -> where([
        ['username', '=',  $username],
        ['password', '=',  $password]])->first();
```

Amennyiben nem találja meg hibát fog eredményezni, és a következő üzenet jelenik meg: Hibás felhasználónév vagy jelszó.
Amennyiben sikeres a bejelentkezés, akkor returnként visszaadja az értékét, és a következő üzenet jelenik meg: Sikeres bejelentkezés.

A VersenyDiakController egy GET kéréssel indít, amely a versenyeket listázza, és minden versenynél meg van jelölve, hogy megtörtént e a diákok részéről a feladat beküldése.
A kitöltött versenyId-ket az alábbiként gyűjtjük össze:

$kitoltottVersenyekId = $this->collectSubmittedCompetitionIds($diakId);

Majd az összes verseny listázása történik meg így:

$versenyek = Verseny::all();

Ezek után beállítjuk, hogy a versenyeket beküldték e, ehhez egy sima foreach-et használunk, majd visszatérünk az OK értékkel.

A következő GET kérésnél a kérdések kilistázása történik meg a versenyekhez. Ha a diák még nem küldte be a feladat megoldását, akkor a helyes válaszok nem látszanak. Azonban ha már küldött megoldást, akkor mind a helyes, és mind a beküldött válaszokat is tartalmazza. A további komponenseken haladva szemléltetnénk:

- Kitöltött versenyId-k összegyűjtése.
- Kérdések listázása az adott versenyhez.
- Majd egy elágazás következik, ha már ki van töltve, akkor a jelölt válaszok, és a helyes válaszok is hozzáadódnak; ha még nincs kitöltve akkor csak az alap adatok adódnak hozzá (a helyes válasz elrejtése történik meg).

A következő POST kérésnél meg kell adni a diák és verseny azonosítót, valamint az összes kérdésre a választ, amely a megoldás beküldéséhez szükséges a verseny esetén.

Input validálása így néz ki:

```
$validator = Validator::make($request->all(), [
        'studentId' => 'required',
        'competitionId' => 'required'
    ]);
```

Amennyiben hiányos, a következő üzenettel találkozunk: Hiányzó adatok a verseny beküldéséhez.

Ezek után leellenőrizzük, hogy beküldésre került e már korábban a kitöltött versenyünk. Amennyiben beküldtük, ez az üzenet jelenik meg: Erre a versenyre már beadtad a megoldást.

Végül az adatbázis részére történő adatok átvitele történik meg és ha ez sikerült akkor a következő üzenetet láthatjuk: A megoldás a versenyre sikeresen beküldve.

A VersenyTanarController GET kéréssel indít, ahol az összes verseny kilistázása történik meg először.

Ezt egy újabb POST kérés követi, és az új verseny létrehozásával kapcsolatba kell megadni a leírást és a nevet. Ismételten a validátor folyamatán megyünk keresztül, és amennyiben fails-t eredményez a következő üzenetet kapjuk: Hiányzó adatok a verseny létrehozásához.
Amennyiben a statusonk OK, akkor ez válik olvashatóvá: A verseny sikeresen létrehozva.

Továbbiakban a PUT kéréssel módosítjuk, ahol validációs folyamaton haladunk újra, amennyiben fails, ezt a mondatot kapjuk: Hiányzó adatok a verseny módosításához.
Viszont ha a statusonk OK, akkor a return által a következőket olvashatjuk: Sikeres verseny módosítás.
A DELETE kéréssel történik meg a verseny törlése, és ezzel az üzenettel tér vissza: A verseny sikeresen törölve.

A Models-ek által valósul meg a kommunikáció az adatbázissal. Alapjáraton ha egy models neve megegyezik az adatbázisunkban szereplő táblával, az adatbázissal kapcsolatos műveletek végrehajtódnak.

A következő Modelseket hoztuk létre: Diak.php, Kerdes.php, KitoltottKerdes.php, KitoltottVerseny.php, Tanar.php, User.php, Verseny.php.

A Model megfogalmazása az alábbi lenne: „A View-ben leképezett/felhasznált adatokat az úgynevezett Model osztályok példányaiban tároljuk. A modell osztály megalkotásakor a fejlesztők rengeteg opció közül választhatnak, a legnépszerűbbek az ORM megoldások, mint pl. az Entity Framework. A Model bármilyen adatforrásból dolgozhat, nem csak relációs adatbázisokból.”[15]

Validálás kifejezés alatt azt értjük, hogy a felhasználó által az adatok elküldésére több opció is adott. A szerver oldalon lehetőség van egyedi logika validálására, illetve attributumok felhasználására. Továbbá az attributumok nem csak validálásra használhatóak fel, hanem segítségükkel egyedi logika megvalósítása is implementálható. A mi általunk használt egyik ilyen attributum egyébként a required.

[15] A következő oldalon olvasható az idézett fogalom:
http://nyelvek.inf.elte.hu/leirasok/ASP.NET/index.php?chapter=5
(Letöltés ideje: 2023.05.09.)

2.7.Tesztelés

A következő adatokkal dolgoztunk a tesztelés során:

Oszlop	Teszt adatok		
id	1	2	3
username/felhasználónév	Nagy Béla	Kiss_Pista	Apró_Ildikó
password/jelszó	1a2b3c012345	4d5e6f012345	7q8w9e012345
email	nagy.bela@gmail.com	kiss.pista@gmail.com	apro.ildiko@gmail.com
fullname/teljes név	Nagy Béla	Kiss Pista	Apró Ildikó
school/iskola	Nemes Nagy Ágnes	Bláthy Otto Titusz Informatikai Szakközép Iskola	Vajda János Gimnázium
class/osztály	5	8	3

oszlop	Teszt adatok				
id	2	3	4	5	7
competitionId	2	1	1	2	1
question/kérdés	A víznek sok formája van. A természetben folyékony (tavak, folyók), szilárd (jég) és légnemű (pára) alakban fordul elő. Az alábbiak közül melyik még a légnemű alakja?	Hány oldala van egy körnek?	Hány oldala van egy trapéznak?	Hány füle van egy nyúlnak?	Hány oldala van egy körnek?
answer1/válasz	a gőz	valamennyi	1	nincs füle	4
anwer2/válasz	a jégvirág	0	2	1	3
answer3/válasz	a zuzmara	pont annyi	3	2	2
answer4/válasz	a dér	nem tudom	4	2 pár	0

correctAnswer /helyesVálasz	1	2	4	3	4

Oszlop	Teszt adatok		
id	1	2	3
username/felhasználónév	Nagy_István	Kiss_József	Apró_Nárcisz
password/jelszó	1p2l3d025795	8d3e9f010354	3q7w1e016345
email	nagy.istvan@gmail.com	kiss.jozsef@gmail.com	apro.narcisz@gmail.com
fullname/teljes név	Nagy István	Kiss József	Apró Nárcisz
subject/tantárgy	matek	környezet	informatika
class/osztály	10.b	10.b	11.b

A weboldalt különböző nézetben teszteltük, Windows és Mac általi IOS rendszerek által nyitottuk meg, ahol minden rendben volt. A menü pontjai és weboldal kialakítása a responzív követelményeknek megfelelt. Fontos tudni azt, hogy bizonyos webhelyek nem engedélyezik a keretben való megjelenítést, ekkor csak külön ablakban – a felkínált link-re kattintva - lehet a felkeresett webhelyen használni.

Az adatokat a fent megadott táblázatban az adatbázisunkban rögzítettük, majd jelentkeztünk be a megadottakkal. Sikeres bejelentkezés volt, azonban ha nem jó kombinációt adtunk meg mint felhasználónév és jelszó, akkor ez nem működött, megkaptuk a hibaüzeneteket, amelyeket korábban már a Back-end-ben rögzítettünk. Ugyan ez az eset áll fenn akkor amikor egy új regisztráción megyünk keresztül, ha mindent jól töltünk ki, akkor abban az esetben sikerül a regisztráció és megtörténik a navigálás a bejelentkezés felületére. Azonban ha nem töltünk ki mindent a regisztráció során, akkor nem lehet befejezni ezt a tevékenységet.

A versenyeknél sikeresen zárult a verseny kitöltése, ahol megnéztük, hogy valóban a jó válaszokra adja a pontot, és valóban az választható ki amit a felhasználó megjelöl. Emlékszik e a jó válaszokra a programunk, és mutatja e azokat, ha végig megyünk a folyamatokon. Ugyanígy végrehajtásra került az az opció, amikor nem töltünk ki minden kérdésre adott választ, és el szeretnénk küldeni, beadni. Ekkor is megjelenik a fent már jól ismertetett üzenet.

A tanárok felületén ugyan ezeket a folyamatokat hajtottuk végre mind a regisztráció, mind a bejelentkezés tekintetében. Itt kicsit összetettebb volt abból a szempontból, hogy ők adhatnak meg kérdést, és törölhetnek, vagy módosíthatnak is. A hangsúly itt azon volt, hogy a kérdést csak akkor engedje tovább, ha a feltételeknek megfelelő válaszokat adott meg és megjelölte azt, amelyik helyes.
Summázva a tesztelést sikeresen zártuk, és eredményesen megoldottuk a projektünket.

3. Fejezet Jövőbeli kilátások, fejlesztési ajánlások és lehetőségek

A jövőben több alternatíva közül is választhatunk, amellyel még népszerűbbé tehetjük az általunk létrehozott Versenyeket.

Színesíthetjük más – más témakörökkel, tantárgyakkal, olyanokkal amelyekkel mindenki szívesen megismerkedne, vagy tanulhatna. Ezeket specializálni lehetne akár az érettségre vonatkozóan, vagy a 8.osztályos tanulók továbbtanulásához szükséges tananyag elsajátításához.

Használhatóvá válik képzések megtartásához, ahol az a cél, hogy a résztvevő személyek tudását fel tudjuk mérni, kellő alapossággal ki tudjuk deríteni mely terület mélyebb megismeréséhez van még szükség, milyen újabb kérdéseket kell összeállítani ahhoz, hogy minden tudást ténylegesen a magunkénak tudhassunk.

Nagyon hasznosnak tartanám ha az eljövendő időszakban a tanórákon akár mint játék rendelkezésre állna a diákok számára, így ha egymással kell versenyezniük, akár egyénileg vagy csapatban történő megvalósítás esetén sokkal inkább élmény dúsabb és színvonalasabb oktatásban lenne részük. Még azon szereplőket is bevonhatnánk, akiket nem érdekel az adott tananyag elsajátítása, hiszen ha játszva tanulunk akkor az azoknak is megmarad, akik érdektelenek a téma iránt. Úgy gondolom, hogy a feladatokat egyénileg és csoportosan is ki lehet próbálni, milyen úgy együtt dolgozni, amikor odafigyelünk egymásra, a kérdésekre, és a válaszokra. Házi versenyeken lehetne jutalmazni a legjobbakat, ezekről az adott oktatásban résztvevő személyek dönthetnének.

Tovább lehetne fejleszteni és egy diákoknak számára létrehozott alkalmazást, amelyet a saját telefonjukon használnának. A mobil telefon mindig kéznél van, ezeket az egyéni versenyek

kitöltéséhez ajánlanám igazán. Gyorsabb, könnyebb, kezelhetőbb és egyszerűbb. A weboldalon a tanároknak létre lehetne hozni hasonlóan egy külön felület, ugyanúgy mint a diákoknak, mert gondolnunk kell azokra a generációkra is, akik még nehezebben kezelik a mobiltelefonokat, és egy webes felületen jobban szemléltethetőbb, láthatóbb az ami éppen történik.

Javasolnék egy online chat felület elkészítését, és így online formában vagy otthonról is elérhetővé válnának egymásnak a kommunikációs csatornák. Itt inkább az együtt töltött társas kommunikáció lenne a cél, amely egy jó közösség formálásához elengedhetetlen. A diákok szívesebben vesznek részt olyan tanulási formákban, ahol megismerik egymást, így jobban meg tudnak nyílni, egy folyamatot sikeresebben lehet végre hajtani, a kívánt célokat könnyebb elérni. A tanároknak szintén hasznosnak tartom, mert megoszthatnák egymással a tapasztalataikat, interaktív módon kicsit kikapcsolódhatnának, és egy kölcsönös bizalom épülne ki a munkahelyükön. Nagyon fontosnak tartjuk hogy mind az oktatók, mind a diákok egy minőségi élményben részesüljenek, amelyekre szívesen emlékeznek majd vissza. Így az életük különböző területeibe is be tudják építeni azokat a megoldásokat, vagy siker élményeket amelyeket itt kaptak.

Javasolnék még intézmények közötti versenyt is, így az iskolák színeiben is megmérettetnének a tanárok és diákok együtt. A sportversenyek igen népszerűek, és nagyon szeretnek küzdeni, és kihívásokat teljesíteni a hallgatók és az oktatók egyaránt. Ilyenkor tényleg életbe lép a csoportszellem és az az érzés amelyet közvetítenek összekovácsolja még a leggyengébb láncszemeket is.

Létre lehetne hozni a kapcsolatoknál egy – egy intézmény bemutatását, koordinátáját a térképen megtalálható útvonal megjelölésével együtt. Itt az intézmények bemutatása pár mondatban történne meg, és megismerhetnénk, hogy melyik éppen miben erős, miben jobb szolgálgatást nyújt. Ez által ismételten növelnénk az iskolák vezetőit azzal kapcsolatban, hogy minél jobbak és jobbak akarjanak lenni az oktatási tevékenységükben.

Továbbá javasolnám, hogy egy – egy ünnepet, vagy történelmi eseményt így dolgozzanak fel a tanulók önállóan, mert tényleg hasznos lehet otthoni időtöltésnek, ez által nem esnek ki a tanulási folyamatokból, és ha éppen várni kell valakire, vagy valamire, akkor ezzel is el lehet ütni az időt.

Szerintem számtalan területen lehetne még ezt alkalmazni, és tovább fejleszteni, bővíteni. Reméljük, hogy ezzel másnak is adunk ihletet és motiválunk majd fejlesztőket, hogy ez irányba induljanak el.

Záró rész

Összegezve, a projektünk egy hihetetlenül összetett, bonyolult parancsokat, utasításokat, logikai folyamatokat ölel magában. Egy igazi csapatmunka, és agilis fejlesztői munkamorál birtokában, képességeinket használva sikeresen vettük az elénk állított akadályokat. A nehézségeken azért tudtunk átlendülni, mert szerettünk volna egy játékos versenyt átadni, szerettünk volna egy olyan élményt nyújtani, amelynek mi is nagyon szívesen örülnénk. Motiváló hatást akartunk kiváltani a jövőben azokból, akik ezzel a területtel szívesen foglalkoznának. Online mindent könnyebb és gyorsabb megvalósítani, ezért egy verseny lebonyolításához, több szereplő által történő részvétellel ezek kinyilatkoztatása elérhetővé vált. Reméljük, hogy fejlesztésünk ötletét mások is tovább viszik majd, újabb irányokat adunk azoknak, akik kissé talán demotiváltabbá váltak a környezeti behatások által. Reméljük, hogy megújító szereplővé váltunk a mi projektünknek hála.

Felhasznált irodalom

Nyomtatott forrás

1. Illés Zoltán: Programozás C# nyelven. Jedlik Oktatási Stúdió, Budapest, 2005.

Internetes források

1. Weboldal tippek, 10 weboldal tipp
https://weblapdesign.hu/weboldal-keszites/weboldal-tippek/
(Letöltés ideje: 2023.05.10.)

2. Mik azok az adatbázisok?
https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-are-databases/?cdn=disable
(Letöltés ideje:2023.05.10.)

3. Az alábbi oldalon olvasható az idézet szöveg:
https://webiskola.hu/sql-ismeretek/relacios-adatbazis-sql-fogalmak-peldak/
(Letöltés ideje: 2023.05.07.)

4. Az alábbi oldalon olvasható az idézet szöveg:
https://lexiq.hu/front-end
(Letöltés ideje: 2023.05.08.)

5. Mi az a CSS? A CSS bemutatása
https://webiskola.hu/css-ismeretek/mi-az-a-css-a-css-bemutatasa/
(Letöltés ideje: 2023.05.09.)

6. Az alábbi oldalon olvasható az idézett szöveg:
https://lexiq.hu/back-end
(Letöltés ideje:2023.05.09.)

7. Laravel (angol)
https://laravel.com/
(Letöltés ideje: 2023.05.09.)

8. Visual Studio (angol)
https://visualstudio.microsoft.com/
(Letöltés ideje: 2023.05.09.)

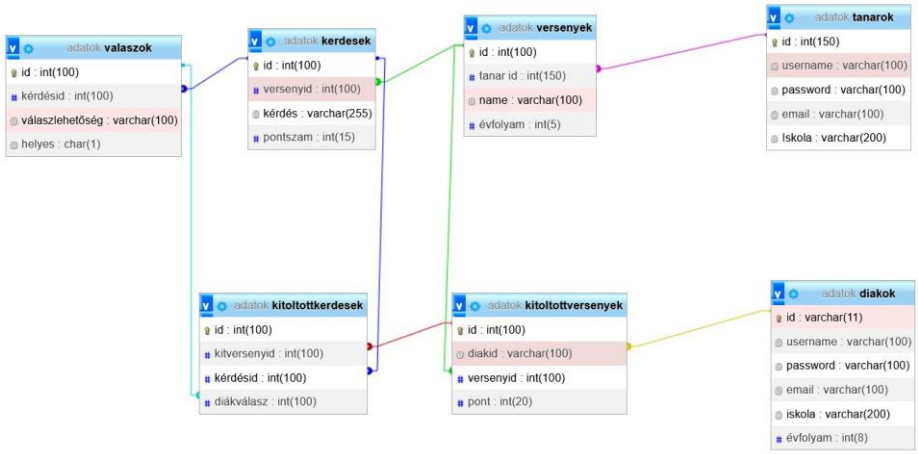
9. PhpStorm (angol)
https://www.jetbrains.com/phpstorm/promo/?source=google&medium=cpc&campaign=14335686429&term=phpstorm&content=604081944634&gad=1&gclid=CjwKCAjwx_eiBhBGEiwA15gLN_juVBCEQbOJnHXGIASuBoX3cvavuWZ0q0C5HkaB7_B9Iu3RzqyWaBoCprkQAvD_BwE
(Letöltés ideje: 2023.05.09.)

10. Mi a Xampp?
https://www.apachefriends.org/hu/index.html
(Letöltés ideje: 2023.05.09.)

11. React
https://react.dev/
(Letöltés ideje:2023.05.09.)

12. A következő oldalon olvasható az idézett fogalom:
http://nyelvek.inf.elte.hu/leirasok/ASP.NET/index.php?chapter=5
(Letöltés ideje: 2023.05.09.)

Melléklet



Adatbázis kapcsolatok kép