**Szakdolgozat 1.**

**Játék alkalmazás fejlesztése webes környezetben**

**Készítette:**

Szabó Bálint

Programtervező informatikus

**Témavezető:**

Dr. Hornyák Olivér

**Szakdolgozat tárgyköre:** játékfejlesztés, webfejlesztés

**Szakdolgozat címe:** Játékalkalmazás fejlesztése webes környezetben

**Tartalomjegyzék**

[1. Bevezetés 2](#_Toc164066787)

[1.1. Logika játékok 2](#_Toc164066788)

[1.2. A játék menete 2](#_Toc164066789)

[2. Hasonló alkalmazások ismertetése 3](#_Toc164066790)

[2.1. Sudoku 3](#_Toc164066791)

[2.2. Grafilogika(Nonogram) 4](#_Toc164066792)

[3. Választott technológiák leírása 6](#_Toc164066793)

[3.1. Egyoldalas webalkalmazások 6](#_Toc164066794)

[3.2. Node JS és az Express 7](#_Toc164066795)

[3.3. HTML 8](#_Toc164066796)

[3.4. CSS 8](#_Toc164066797)

[3.5. MySQL 8](#_Toc164066798)

[3.6. JavaScript 9](#_Toc164066799)

[3.7. Angular 9](#_Toc164066800)

[3.8. JSON 10](#_Toc164066801)

[4. Tervezés és implementáció 10](#_Toc164066802)

[4.1. A projekt felépítése 10](#_Toc164066803)

[4.2. Tervezés 13](#_Toc164066804)

[4.3. Felhasználói felület 13](#_Toc164066805)

[5. Funkcionális leírás 17](#_Toc164066806)

[5.1. Projekt felépítése 18](#_Toc164066807)

[5.2. Az adatbázis felépítése 19](#_Toc164066808)

[5.3. Backend felépítése 19](#_Toc164066809)

[5.4. Kapcsolat a frontend és a backend között 22](#_Toc164066810)

[6. Játék tesztelése 22](#_Toc164066811)

[6.1. Felhasználói visszajelzések 22](#_Toc164066812)

[6.2. Fejlesztési lehetőségek 22](#_Toc164066813)

[7. Összefoglalás 22](#_Toc164066814)

# Bevezetés

## Logika játékok

A logikai játékok mindig is fontosok voltak az emberek számára. Véleményem szerint elengedhetetlen, hogy az esetleges monoton hétköznapokon, megmozgassuk egy kicsit az agyunkat. Azért választottam ezt a témát, mert mindig is szerettem a logikai játékokat és megérteni azok technikáit. Amikor megnyitunk egy logikai játékot, akkor sokszor olyan feladatok tárulnak elénk, amikkel általában nem találkozunk a hétköznapjainkban.

A logikai játékoknak is van többféle előfordulása. Van, amiknél számokkal kell dolgozni, de van olyan, ahol vizuálisan is át kell látni a dolgokat, hogy megtaláld a megfejtést. Személy szerint több éven át verseny szinten sakkoztam, ami szintén egy olyan tábla játék, ami erősen fejleszti a gondolkodást, illetve a térbeli látást is.

**Hogy valóban miért is hasznosak a logikai játékok?**

## A játék menete

A felhasználónak meg kell oldania egy logikai feladatot, webalkalmazás környezetben. A játék koncepciója, hogy az egyes oszlopokban és sorokban annyi mezőt kell kitölteni, hogy az első sorban és oszlopban lévő számmal megegyezzen. Mielőtt elkezdődne a játék, még előtte ki lehet választani a tábla méretét. Ez azért fontos, mert ha valaki még nem játszott ilyen típusú játékkal, akkor ő is meg tudja oldani a könnyebb feladványokat. Emellett pedig van lehetőség nagyobb táblát is választani azoknak, akik nehezebb példát akarnak megoldani, ami már több időt vesz igénybe. Ezek után, ha felkészült játékos a start gombbal elindíthatja a játékot. A ”START” gomb megnyomása után jelenik csak meg a feladvány, illetve az óra is el kezd számolni. A játékos a saját elgondolása szerint kitöltheti a mezőket. Ha úgy érzi végzett a mezők kiválasztásával, akkor ”CHECK” gombot megnyomva tudja meg, hogy sikerült e megoldani vagy sem. Ha sikerült jól megoldani a játékot, akkor megáll az óra és megjelennek a játék eredményei. Viszont, ha nem sikerült helyesen kiválasztani a mezőket, egy visszajelző üzenet jelenik meg, hogy nem jó a megoldás. Az óra megy tovább, addig amíg helyesen nincs kitöltve a táblázat.

# Hasonló alkalmazások ismertetése

Mielőtt belekezdtem volna az alkalmazás elkészítésébe, piackutatást végeztem, milyen az elképzelésemhez hasonló logikai játékok érhetők el. Nem kellett sokáig kutatni, mivel az alábbi két játékkal, amit megemlítek már találkoztam régebben, sőt játszottam is velük. Már akkor elnyerte a tetszésemet az ilyen típusú logikai játékok.

## Sudoku

A Sudoku számjegyeken alapuló logikai fejtörő. A megoldásához koncentrálásra és logikus gondolkodása van szükség. A Sudoku nehézsége az előre kitöltött cellák számától és a megoldásához szükséges módszerektől függ. A rendszeres Sudokuzás javítja a memóriát, a mentális tisztánlátást és csökkenti az agysejtek öregedését.

A játékmező egy 9x9-es négyzet, amely 3 cella szélességű kisebb négyzetekre van osztva. A játék célja a számok beírása 1-től 9-ig minden sorba, minden oszlopba és minden 3x3-as négyzetbe úgy, hogy mindegyik számjegy csak egyszer forduljon elő.

A rejtvény kezdőállapota egy részlegesen kitöltött tábla, amelynek a teljes kitöltésére törekszünk. Egy lehetséges kezdeti állapot, amelynek egy megoldása van:

A képen tér, szám, keresztrejtvény, szöveg látható

Automatikusan generált leírás

És ennek a megoldása:

A képen szöveg, képernyőkép, szám, tér látható

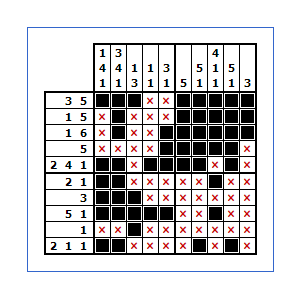
Automatikusan generált leírás

Talán a sudoku volt az első játék, ami az ötletet adta a szakdolgozatom témájához. Egy időben nagyon sok ilyen feladványt oldottam meg és mindig is élveztem

*LINK: https://sudokutable.com/howtoplay/?lang=hu*

## Grafilogika(Nonogram)

A grafilogika egy logikai rejtvényfajta, amely egy japán dizájner, Non Ishida ötletei nyomán alakult ki. 1987-ben azzal az ötletével nyert meg egy grafikai versenyt, hogy egy felhőkarcoló bizonyos ablakait kivilágított, másokat pedig elsötétített, így távolról nézve az épületet, egy kép rajzolódott ki. Ezen ötletén alapul a később grafilogikának nevezett rejtvény is. 1990-ben James Dalgety adta ennek a játéknak a Monogram nevet, amely Magyarországon grafilogikaként vált ismertté.



A rejtvény egy téglalap alakú négyzetrácsos hálóból áll, amelynek az egyik vízszintes és az egyik függőleges oldala mellett számok állnak, amelyek azt jelzik, hogy az adott sorban vagy oszlopban mekkora méretű sötét blokkok követik egymást. Minden beszínezett blokkot tetszőleges számú (legalább egy) üres hely választ el egymástól. A játék célja, hogy minden négyzetrácsról eldöntsük, be van-e színezve vagy nem.

Megoldási technikái:

* Átfedések megkeresése
* Blokkok megtalálása a falaktól kezdve
* Kizárások és üres helyekkel való operálás
* Próbálkozásos módszer

A játék megfejtése során fontos, hogy végig egységesen jelöljük az üres négyzeteket is. Ezeket legegyszerűbben egy X-szel tudjuk jelölni az adott négyzetben. Az üres négyzetekre azért van szükség, hogy jól be tudjuk azonosítani később is azokat a pontokat, ahova nem kerül fekete blokk. Ez a játék előrehaladtával igencsak nagy segítségnek számít! Ahogy a játék vége fele haladunk, konzisztensen jelöljük a már véglegesített, teljes blokkokat a számozásnál is, adott számok kihúzásával. Az egyik leggyakoribb hibaforrás, hogy elfelejtjük kihúzni a már beszínezett blokkot, és ismét berajzoljuk. Ezáltal biztosan nem fog helyesen kijönni a végeredmény, ami számos bosszús percet okozhat nekünk a hibák megkeresésével.

*LINK:* [*https://hu.wikipedia.org/wiki/Grafilogika*](https://hu.wikipedia.org/wiki/Grafilogika)

# Választott technológiák leírása

Ebben a szakaszban bemutatom az összes olyan technológiát, programozási nyelvet, keretrendszert és könyvtárat, amit felhasználtan a szakdolgozatom elkészítése során. Törekszem a tömör magyarázatra. Mindegyiknél próbálom elmagyarázni miért választottam, valamint mik az előnyei, amik számomra kedvezőek voltak.

Mielőtt bele kezdtem a játék fejlesztésébe, több olyan témának is utána néztem, ami jelentősen kapcsolódik a szakdolgozatomhoz. A programozási nyelvek kiválasztása során, információt gyűjtöttem ezekkel kapcsolatban. 2023-ban legtöbbet használt programozási nyelvek világszerte. Ezek közül első két helyen a JavaScript és a HTML/CSS leggyakrabban használt.

## Egyoldalas webalkalmazások

Ha figyelemmel követjük a JavaScript keretrendszerek fejlődését, akkor láthatjuk, hogy egyre nagyobb teret kapnak az egyoldalas alkalmazások (SPA, Single Page Application),a többoldalas alkalmazásokkal (Multi Page Application) szemben, amiket leggyakrabban jQuery vagy Laravel segítségével készítenek. A keretrendszerek népszerűségét a ÁBRÁN szemléltetem. Három elterjedt JavaScript keretrendszert különböztethetünk meg. A Facebook által fejlesztett React-ot, az Angular-t, amit a Google hozott létre és a Vue JS, amit Evan You fejlesztett.

## Node JS és az Express

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

LINK: https://jaydevs.com/most-popular-javascript-frameworks/

A JavaScript nem csupán a legnépszerűbb programozási nyelv, hanem az egyik legelterjedtebb és legszélesebb körben használt szoftverfejlesztési technológia is. Ezt jól illusztrálja a 2.5. ábra diagramja. Korábban főként frontend fejlesztésre alkalmazták, azonban az utóbbi időben egyre elterjedtebbé vált a szerveroldali (backend) alkalmazása is. Az egyik kulcsfontosságú eszköz, ami hozzájárult ennek a változásnak, a Node.js.

A Node.js nem csupán egy keretrendszer vagy könyvtár, hanem egy teljes futtató környezet, mely a Chrome V8-as motorján alapul. Ryan Dahl mutatta be először ezt a technológiát 2009-ben az Európai JavaScript Konferencián. Annak ellenére, hogy már ekkor nagy figyelmet kapott, széles körű elterjedése csak később következett be. 2017-ben érte el csúcspontját, amikor is olyan nagyvállalatok, mint például a LinkedIn, a Netflix, az eBay és az Uber is elkezdték használni. A Node.js széleskörű alkalmazása és hatékonysága révén egyre inkább meghatározó szereplővé vált a modern szoftverfejlesztésben.

## HTML

A HTML-ről (HyperText Markup Language) nem szeretnék hosszasan írni. Talán egy

laikus is tudja, hogy ha webalkalmazásról van szó, vagy akár csak egy honlapról, akkor

megkerülhetetlen a HTML

.

A HTML-t weboldalak készítésére hozták létre, amit később bárki elérhet, aki rendelkezik internet kapcsolattal (feltéve, hogy fel van töltve a weboldal egy domain-re).

Használhatunk benne főcímeket, paragrafusokat, beépített képeket, videókat. Ezeket

úgynevezett tag-ek határozzák meg. Az elején a kezdő tag, a végén pedig a záró tag.

Személy szerint könnyen tudtam kezelni, már korábban is használtam hasonló kisebb méretű oldal elkészítéséhez.

*Link:* [*https://www.altexsoft.com/blog/front-end-development-technologies-concepts/*](https://www.altexsoft.com/blog/front-end-development-technologies-concepts/)

*[Online, 2020.01.30]*

## CSS

A CSS (Cascading Style Sheets) egy stílusleíró nyelv. Alkalmazása a HTML elemek formai megjelenésére irányul, azaz hogyan szeretnénk láttatni a weboldalunkon megjelenő

tartalmakat.

A CSS a HTML elemekre hat, a kommunikálás pedig többek között a selector -ok

segítségével történik. Ezeket a CSS alapból tartalmazza, hivatkozhatunk egy megformálni kívánt paragrafusra, főcímre, valamint megadhatunk saját osztályokat is, amiket

többször fel tudunk használni. A deklaráció tulajdonságokat és értékeket tartalmaz.

Itt szeretnék megemlíteni két dolgot, amit a szakdolgozatom készítése során tapasztaltam. Az egyik, hogy érdekes volt számomra megfigyelni, hogy leginkább ez a rész volt

az, ami felkeltette az érdeklődésem, ebben tudtam maximálisan kiteljesedni. A másik

pedig, hogy megjegyezném az egyre elterjedtebb felhasználási módját a CSS-nek. Ez

pedig nem más, mint a Bootstrap, amit azért hoztak létre, hogy könnyedén készítsünk

reszponzív weboldalakat, azaz minden képernyőméreten szépen megjelenő oldalakat.

Esetemben tudtam, hogy az én alkalmazásomat számítógép képernyőjén, esetleg

más eszközökön, de kizárólag fektetett állapotban lehet használni. Éppen ezért nem

láttam értelmét mélyebben beleásni magam ebbe a világba.

*LINK:* [*https://www.altexsoft.com/blog/front-end-development-technologies-concepts/*](https://www.altexsoft.com/blog/front-end-development-technologies-concepts/)

*[Online, 2020.01.30]*

## MySQL

A MySQL egy Oracle által támogatott nyílt forráskódú relációs adatbázis-kezelő rendszer (RDBMS), amely a strukturált lekérdezési nyelven (SQL) alapul. A MySQL gyakorlatilag minden platformon fut, beleértve a Linuxot, a UNIX-ot és a Windows-t is. Bár számos alkalmazásban használható, a MySQL-t leggyakrabban a webes alkalmazásokhoz és az online publikáláshoz társítják.

A MySQL fontos összetevője a LAMP nevű nyílt forráskódú vállalati stacknek. A LAMP egy olyan webfejlesztési platform, amely operációs rendszerként a Linuxot, webkiszolgálóként az Apache-ot, relációs adatbázis-kezelő rendszerként a MySQL-t, objektumorientált szkriptnyelvként pedig a PHP-t használja. (Néha a PHP helyett Perl vagy Python nyelvet használnak).

A MySQL-t eredetileg a svéd MySQL AB cég tervezte, majd 2008-ban a Sun Microsystems, majd 2010-ben az Oracle vásárolta fel, amikor a Sun-t felvásárolta. A fejlesztők a GNU General Public License(GPL) alapján használhatják a MySQL-t, de a vállalatoknak kereskedelmi licencet kell szerezniük az Oracle-től.

Ma a MySQL a világ számos vezető weboldala és számtalan vállalati és fogyasztói webes alkalmazás, köztük a Facebook, a Twitter és a YouTube mögött álló RDBMS.

Számomra a legfőbb előnyei a technológiának, amiért ezt választottam a következők. Teljesen ingyenesen lehet használni a legtöbb szolgáltatását. Könnyű hozzáférést biztosít az adatokhoz, a Phpmyadmin keresztül. Könnyű az integrálása is, és minimális programozási ismereteket kíván, tehát majdnem bárki be tudja építeni az alkalmazásába.

*Link:* [*https://www.techtarget.com/searchoracle/definition/MySQL*](https://www.techtarget.com/searchoracle/definition/MySQL)

*[2018 july]*

## JavaScript

A JavaScript (JS) egy könnyű, értelmezett vagy just-in-time fordított programozási nyelv első osztályú funkciókkal. Bár leginkább a weboldalak szkriptnyelveként ismert, számos nem böngésző környezetben is használják, például a Node.js, az Apache CouchDB és az Adobe Acrobat. A JavaScript egy prototípus-alapú, többparadigmás, egyszálú, dinamikus nyelv, amely támogatja az objektumorientált, imperatív és deklaratív (pl. funkcionális programozási) stílusokat.

A JavaScript dinamikus képességei közé tartozik a futásidejű objektumkonstrukció, a változóparaméter-listák, a függvényváltozók, a dinamikus szkriptkészítés, az objektum-introspekció és a forráskód-helyreállítás.

*LINK:* [*https://developer.mozilla.org/en-US/docs/Web/JavaScript*](https://developer.mozilla.org/en-US/docs/Web/JavaScript)

## Angular

Az Angular egy platform és keretrendszer egyoldalas kliensalkalmazások építésére HTML és TypeScript használatával. Az Angular TypeScript nyelven íródott. Az alapvető és opcionális funkciókat TypeScript könyvtárak készleteként valósítja meg, amelyeket importálhat az alkalmazásaiba.

Egy Angular alkalmazás architektúrája bizonyos alapvető koncepciókra épül. Az Angular keretrendszer alapvető építőkövei az Angular komponensek.

A komponensek nézeteket definiálnak, amelyek képernyőelemek halmazai, amelyek közül az Angular választhat és módosíthat a program logikájának és adatainak megfelelően.

A komponensek szolgáltatásokat használnak, amelyek a nézetekhez közvetlenül nem kapcsolódó háttérfunkciókat, például az adatok lehívását biztosítják. Az ilyen szolgáltatásokat függőségként be lehet injektálni a komponensekbe, így a kód moduláris, újrafelhasználható és hatékony lesz.

Számos Angular Material design komponenst tudtam felhasználni az alkalmazásom felépítéséhez. Ezek nem csak megkönnyítik a weblap felépítését, de stílusos megjelenést biztosítanak az egyes elemeknek.

*LINK:* [*https://angular.io/guide/architecture*](https://angular.io/guide/architecture)

## JSON

A JSON (JavaScript Object Notation) egy szövegalapú, nyílt szabványú adatátviteli formátum, melyet kifejezetten emberi olvasási és feldolgozási célokra terveztek. Gyakran alkalmazzák JavaScript alapú alkalmazásokban és hálózati kommunikáció során.

Az egyik fő előnye, hogy könnyen kezelhető a legtöbb dinamikus programozási nyelvben, mivel közvetlenül támogatja az alapvető adattípusokat, mint például a string, number és boolean értékeket. Emellett lehetőséget biztosít objektumok létrehozására, melyeket kulcsokkal hivatkozva tudunk kezelni. Személyes tapasztalatom alapján kifejezetten előnyös nagy adathalmazokkal dolgozó szakemberek számára, mivel segítségével hatékonyan lehet kereséseket végezni, ami különösen fontos a webfejlesztés terén.

# Tervezés és implementáció

Ebben a fejezetben bemutatom az elkészült programot a számomra legérdekesebb programrészletekkel. Nem térek ki minden funkcióra és programkódra, csak a lényegesebb

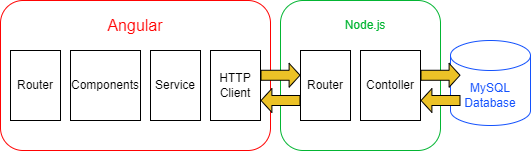
egységekre. Részletezem a frontend kivitelezését, amelybe tartozik a Angular JS, HTML és

CSS kódok. Ezután a backend számításokat mutatom be, amelyet Node JS környezetben

valósítottam meg.

## A projekt felépítése

Mielőtt belekezdek a megoldásaim, függvények, programkódok részletezésébe, először bemutatom a projekt felépítését, hogy egyszerűbben érthető legyen, kezdve a webalkalmazás felépítésével.



A webalkalmazás frontend részét a már említett Angular JS-ben valósítottam meg. Ebben

vannak a komponensek és a -router, amely az oldalak közti navigációt végzi

el. A frontend és a backend közötti kommunikációt egy API adja meg. Az adatbázis

lokálisan helyezkedik el és csak a backend kommunikál vele, illetve használja a

Kód átláthatóságát, rendszerezettségét nagyon fontosnak tartottam a projekt felépítésekor. Mások számára értelmezhető legyen. Meghatározott céllal rendelkezzen és jól strukturált legyen. Egyes függvények értelmezhetősége, ez által az eleminek könnyen megoldható változtatása. Fontos, hogy jól kezelhető és logikus legyen a mappastruktúra felépítése. Külön komponensekként kezelve a webalkalmazás oldalait, ezzel is törekedve az átláthatóságra.

A képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás![A képen fekete látható

Automatikusan generált leírás](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAE8AAABlCAYAAAAMA77WAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAADsSURBVHhe7dChAcAwDMCwblfk/0c7sgtiKhFzPzNzDyvvXxbMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzAvMC8wLzFs75wOdMAIRSL2LrQAAAABJRU5ErkJggg==)

## Tervezés

A játék tervezésének az elején nagyon fontos volt azt meghatározni, hogy miként épüljön fel a webalkalmazás. Ötletelés után összeírtam azokat a funkciókat, amik elengedhetetlennek tartottam. Ezt követően bővítettem további elemekkel a tervezést. Elsősorban fontosnak találtam azt, hogy legátláthatóbb és legegyszerűbb legyen a felhasználó számára a kezelése. Elengedhetetlennek tartom az autentikációt, felhasználók megkülönböztetésére. Emellett véleményem szerint a felhasználó is biztonságosabbnak talál egy olyan webalkalmazást, ahol saját különálló fiókkal rendelkezik. Nem mellesleg az előre elképzelt funkcióknál is fontos szerepet játszik, a felhasználókhoz tartozó egyedi felhasználónév.

## Felhasználói felület

NAVIGÁCIÓS FEJLÉC

A navigációs fejléc lehetőséget biztosít az oldalak közötti egyszerű lépkedésre. Ennek a megvalósításához egy külön komponenst hoztam létre. Ezt meghívtam az app komponensben, így mindegyik oldalon elérhetővé válik a navigációs fejléc használata.

Az Angular JS egyik előnye, hogy komponensekből épül fel, melyeket többször fel tudunk

használni. Nekem az egyik ilyen komponensem a navigációs fejléc. Ezt mindegyik

oldalon alkalmazom, hiszen a felhasználó ezen keresztül tud váltani az oldalak között.

Másik sajátossága a view-router, ami megvalósítja az oldalak közötti ugrást, ezek az

fájlban vannak definiálva.

<mat-toolbar class="toolbar">

  <span>

    <img src="/assets/images/lightbulb2.jpeg">

  </span>

   <a routerLinkActive="active" [routerLinkActiveOptions]="{exact: true}" routerLink="/home" mat-button>Home</a>

   <a routerLinkActive="active" routerLink="/results" mat-button>Results</a>

   <a routerLinkActive="active" routerLink="/game" mat-button>Game</a>

   <div class="icon">

    <mat-icon (click)="profileClick()">account\_circle</mat-icon>

   </div>

   <div class="icon">

    <mat-icon (click)="logoutClick()">logout</mat-icon>

   </div>

 </mat-toolbar>

A webalkalmazásban több menüpont is elérhető lesz a felhasználó részére. Próbáltam logikusan felépítnei mik is azok a legfontosabb információk a játékkal kapcsolatban, amikhez szüksége külön menüpont.

AUTENTIKÁCIÓ

submit() {

    const username = this.registerForm.value.username;

    const password = this.registerForm.value.password;

    const passwordRepeat = this.registerForm.value.passwordRepeat;

    if(password == passwordRepeat){

      this.authService.register(username,password).subscribe({

        next:results=>{

        this.router.navigate(['/login']);

        },

        error:(err=>{

        this.error = err.error.message;

      })

      });

    }

    else {

      this.error = "Passwords don't match";

    }

  }

PROGRAMKÓD

Az oldal megnyitásakor mindenekelőtt egy bejelentkezéshez szükséges űrlap jelenik meg felhasználó előtt. Az oldal többi része nem elérhető, amíg meg nem történt a sikeres bejelentkezés. Ha először került megnyitásra az oldal és nem regisztrált még felhasználó, akkor ezzel kell kezdeni. Ha úgy próbál bejelentkezni, hogy nincs regisztrálva még fiókja, akkor egy *Incorrect username or password* hibaüzenetet ad vissza az űrlap. A regisztrációt követően, a megadott adatai eltárolódnak az adatbázisban, így amikor már a bejelentkezéshez tölti ki a mezőket átirányítja az oldal többi részéhez.

Lehetősége lesz az eredetileg megadott adatait módosítani, amit a profil ikonra kattintva érhet el. Az egyéni felhasználónév azért fontos, ahogy az általa megoldott feladványok eredményei nyomon követhetőek legyenek.

Ennek a két oldalnak a HTML része nagyon egyszerű. Egy űrlapot, belül két szöveges beviteli mezőt, egy regisztrációs linket, ezen kívül egy gombot találunk. A regisztrációs link átirányít egy másik űrlaphoz, amin három szöveges beviteli mező van. Az első a felhasználónév, a második a jelszó, a harmadik pedig a jelszó megismétlésének megadására szolgál. A sikeres regisztrációt követően visszairányít az először megjelent bejelentkező oldalra.

  submit() {

    const username = this.loginForm.value.username;

    const password = this.loginForm.value.password;

    this.authService.login(username,password).subscribe({next:results=>{

      localStorage.setItem('currentUser',JSON.stringify({

        id: results.id,

        username: results.username

      }));

      this.router.navigate(['/game']);

    },error:(err=>{

      this.error = err.error.message;

    })

  });

  }

Az autentikációt viszonylag egyszerű elvégezni a MySQL segítségével. Regisztrációnál az fent megjelenített függvényt használva hozunk létre egy felhasználót. Ehhez egy felhasználónév és egy jelszó tartozik. Ezt el is tárolja nekünk a MySQL database-ben. Ezután bejelentkezésnél az alábbi függvényt használjuk, amely megkapja az felhasználónév és jelszó párost, és ha ez egyezik, már bent is vagyunk a főoldalon.

HOME

Miután megtörtént a sikeres regisztráció és bejelentkezés, erre az oldalra irányítja a felhasználót az alkalmazás. Ebben a menüpontban megtalálható minden szüksége információ a játékkal kapcsolatban. Ismerteti a játékkal kapcsolatos szabályokat. Leírja hogyan kell használni a felületet, és mi a játék menete. Tudomást szerezhet az oldal funkcióiról, illetve az egyes menüpontok tartalmáról és használatáról.

RESULTS

Ebben a menüpont, tekinthető vissza az addigi megoldott feladványok eredményei. Elsősorban a fiókhoz tartozó felhasználónév, a játék idejének hossza és a megoldott tábla mérete. Ezt megtekintve tudja a felhasználó, hogy mennyivel teljesített jobban vagy rosszabbul az elmúlt játékokban. Megnézheti mennyivel gyorsabban old meg egy kisebb táblázatot, mint egy nagyot. Egy szűrő segítségével, rá tud szűrni az eredményeire akár idő, akár nehézségi szint szerint. Ezek mellett a felhasználókat is ABC sorrendbe lehet rakni.

Az eredmények között nem csak a saját, hanem a mások eredményeit is láthatja. Össze tudja hasonlítani a saját idejét a többi játékos idejével, ezzel tudomást szerezve, hogy mennyire ügyes a többiekhez képest.

Az eredményeket tartalmazó táblázatokat egy <mat-tab-group> tag-ben adtam, aminek segítségével 3 különböző táblázatok között tudunk válogatni. Az első táblázat tartalmazza az eddigi összes játékos eredményeit. a második táblázatban találjuk a legjobb eredményeket. A harmadik táblázatban pedig a felhasználó a saját eredményeit látja. Tehát az aktuális bejelentkezve lévő profil eredményei jelennek meg.

Mindezeket az eredményeket a játék nehézségi szintje szerint látjuk. Így be lehet állítani, hogy csak a 3-as nehézségi szintű eredményeket, vagy például csak a 8-asokat szeretné látni. Ennek a megvalósításához egy SLIDER-t használtam, amin a 3-as, a 10-es és az ezek közötti értékeket lehet kiválasztani.

GAME

Ezen az oldalon történik magának a játéknak a lebonyolítása. Többféle fontos elem jelenik meg a játék szempontjából.

Az oldal balfelső részén helyezkedik el egy számláló. A számláló alatt három gomb helyezkedik el. Egy START, egy RESET és egy CHECK gomb. A START gombra rákattintva kezdődik a játék és ekkor kezdi el számolni az eltelt időt. Mindaddig számolni fog az óra, amig nincs sikeresen megoldva a táblázat. A számláló, a percet, a másodpercet, a tized és század másodperceket mutatja. A START megnyomása után a RESTART gombra változik. Ennek az a funkciója, hogy erre kattintva egyből újrakezdheti a játékos, újraindul a számlálás és új tábla generálódik. A RESET gomb visszaállítja az órát és táblázatot is alapállapotba

A felhasználó nehézségi szintet is be tud állítania, ami azt jelenti, hogy eldöntheti mekkora táblázatot generáljon a játék. A nehézségi szintet az előző oldalnál említett SLIDER segítségével tudja beállítani. Például, ha a játékos könnyű nehézségi szintű feladványt szeretne választani, akkor lehetősége van akár egy 3x3-as méretű táblát megoldani. Ha nehezebbet szeretne megoldani, akár 10x10-es táblát és létre tud hozni. SLIDER értékét változtatva változik a táblázat mérete.

A játékos a mezőbe kattintva jelöli meg, hogy szerinte melyik mező helyes. A játék közben figyelembe kell venni, hogy ha az oszlopban lévő szám szerint akarunk mezőt választani, akkor nem mehetünk el a mellett, hogy milyen szám van a vele megegyező sorban.

Ha megtörtént a táblázatnak a kitöltése, akkor a számláló alatti CHECK gombra kattintva derül ki, hogy jó-e a megoldás. Ha helyes, akkor egy felugró ablak jelenik, ami kiírja, hogy a játékos nyert, illetve egyből láthatjuk a játék eredményeit. Ha helytelen, akkor egy visszajelző üzenet jelenik meg.

Kibővítettem egy olyan feltétellel a játék felületét, hogy a játék kezdése előtt a CHECK gomb, illetve a táblázat mezők egy DISABLED tulajdonságot kapnak, amihez egy logikai változót rendeltem. Ekkor nem lehet használni se a gombot, se a mezőket kijelölni. Ezt a tulajdonságot a játék elindítása után is alkalmaztam. Mivel a játék úgy érhet véget, hogy ha sikeresen megoldják, újat generálják, vagy visszaállítják alapállapotba, így a játék menete közben a SLIDER használatát is és megvontam, hogy ne lehessen játék közben kisebb vagy nagyobb táblázatot generálni.

A játék által generált táblázat megoldása:

A képen képernyőkép, tér, Téglalap látható

Automatikusan generált leírás

PROFILE SETTINGS

Erre az oldalra, a navigációs fejléc jobb oldalán található profil ikon gomb megnyomásával lehet eljutni. Ezen az oldalon a regisztrációhoz és bejelentkezés hasonló űrlap jelenik meg. Itt lehet megváltoztatni a regisztrációkor megadott adatokat megváltoztatni. A felhasználónév megváltoztatásához csak egy új felhasználónevet kell megadni. A változtatás csak akkor lehetséges, ha olyan nevet adunk meg, ami még nincs használatban. A jelszó megváltoztatásához meg kell adni a régi jelszót, illetve az új jelszót, és annak a megismétlését. A felhasználónév és jelszó megváltoztatása gombot megnyomva, egy üzenet jelenik meg a sikeres változtatásokról. Ezentúl, ha újra be akar lépni a felhasználó, akkor már csak az újonnan megadott adatokkal tud belépni.

# Funkcionális leírás

Ebben a fejezetben bemutatom az elkészült programot a számomra legérdekesebb programrészletekkel. Nem térek ki minden funkcióra és programkódra, csak a lényegesebb egységekre. Részletezem a frontend kivitelezését, amelybe tartozik az Angular, HTML és CSS kódok. Ezután a backend számításokat mutatom be, amelyet Node JS környezetben valósítottam meg.

## Játék elemzése

A táblázatban generált számokat, az első sorban, illetve oszlopban külön tömbben kezeljük TÖMB NEVEK. A táblázat méretével együtt így a tömbök mérete is változhat. Így tudjuk elvégezni a logikai műveletek, illetve meghatározni a helyes mezőket. Feltételeket határozunk meg a generálás során, hogy olyan táblázatot kapjunk, amit valóban lehetséges megoldani. Az 1. legfontosabb vizsgálat, hogy a két tömb elemeinek az összege megegyezzen, mivel enélkül helytelen táblázatot kapnánk. A mezők vizsgálatakor egyesével haladunk végig soronként. Az előbb említett vizsgálatot az alábbi programkód részlet vizsgálja:

Array.from(buttons).forEach(button => {

      //console.log(button.children[0].ariaPressed);

      let correct : Results = {

        row: button.children[0].id.substring(0,1), //row and coloumn id

        col: button.children[0].id.substring(2,3),

        pressed: button.children[0].ariaPressed

      }

      console.log(correct);

      corrects.push(correct);

    });

PROGRAMKÓD RÉSZLET

A kitöltött táblázat ellenőrzéséhez két újabb tömböt hozunk létre TÖMB NEVEK. A mezők vizsgálatánál nagyon fontos az egyes mezők azonosítása.

PROGRAMKÓD RÉSZLET

A táblázatban használt gombokhoz tartozik egy olyan logikai vizsgálat, aminek az értékéből eldönthető, hogy ki van jelölve a mező vagy sem:

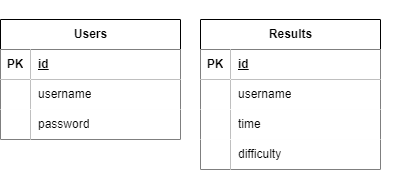
PROGRAMKÓD RÉSZLET

A táblázat generálásakor létrehozott tömbök és az kitöltött mezők vizsgálásakor használt tömböket összehasonlítva tudjuk ellenőrizni a táblázat megoldásának eredményét. Fontos, hogy a sort és az oszlopot jelölő tömböknek egyaránt meg kell egyezni, mivel, ha valamelyik eltér, akkor már hibás a megoldás.

A játék készítése során rájöttem, hogy az általam kitalált leegyszerűsített verzióban olyan táblák generálódnak, amiknek többféle megoldása is lehet. Ez a kevés kikötés miatt történhet. Ezt úgy lehetne kibővíteni, mint a Nonogram játékban. Ott azt is meghatározzuk, hogy hány mezőként legyen egy üres mező.

## Az adatbázis felépítése

RESULTS TÁBLÁZATOK

Az adatbázis két táblából áll. Az egyik a users tábla, ami a felhasználók bejelentkezéskor megadott adatokat tartalmazza. A másik pedig a results tábla, ami a felhasználó játékainak az eredményeit tárolja.

## Backend felépítése

* nodejs
* express, express.Router(), létrehozunk api útvonalakat amelyeket meghívva a frontendből adatokat tudunk kinyerni vagy bejutattani az adatbázisba.  
  Routeok: users, result -> kérések bemutása , /login /getResults stb.
* connection: Adabázis kapcsolat, myslq node package használatával és a .env environment válzotóban eltárolt adatok segítségével létrehozzuk a kapcsolatot
* error handling, megfelelően lekezelt hibák. megfelelő status code responsok (ha valamilyen server oldali hiba van pl: status code: 500, status code: 404 not authorized, stb);
* cors: Cross Origin Policy, az backend és a frontend között szabályozza, hogy honnan érkezhetne kérések (olvass utána).

LOGIN

Az router.post az Express Router objektumának egy metódusa. A Router objektumok olyan útvonalakat definiálnak, amelyeket a kliensek kéréseivel kezelni szeretnénk. A router.post konkrétan azt jelenti, hogy az alkalmazásunk fogadni fog POST kéréseket az adott útvonalon. A post metódus a POST HTTP kéréseket fogja kezelni, amelyeket a megadott útvonalra küldenek a kliensek.

Létrehozunk api útvonalakat, amelyeket meghívva a frontendből adatokat tudunk kinyerni vagy bejutattani az adatbázisba.

A kód először lekéri a kért felhasználónevet és jelszót a kérés testéből. Majd egy SQL lekérdezést futtat a users táblában, hogy ellenőrizze, létezik-e már ilyen felhasználónév. Ha a felhasználó még nem létezik, akkor egy másik SQL lekérdezést futtat, hogy beszúrja az új felhasználót a táblába.

Hatékonyan kezeli a felhasználók regisztrációját az adatbázisban, ellenőrzi a felhasználónevek egyediségét és biztosítja a megfelelő választ minden lehetséges forgatókönyv esetén.

router.post('/register', async (req,res)=>{

  const username = req.body.username;

  const hashedPassword =  req.body.password;

  var checkQuery  = "select \* from users where username=?";

  connection.query(checkQuery,[username],(err,results)=>{

      if(!err){

          if(results.length <= 0){

              var insertQuery = "insert into users(username,password) values (?,?)";

              connection.query(insertQuery,[username,hashedPassword],(err,results)=>{

                  if(!err){

                      console.log(results);

                      return res.status(200).json({message: "Registered successfully"});

                  }

                  else {

                      console.log(err);

                      return res.status(500).json(err);

                  }

              });

          }

          else {

              console.log("Username already exists");

              return res.status(400).json({message: "Username already exists"});

          }

      }

      else {

          console.log(err);

          return res.status(500).json(err);

      }

  })

});

Ezen az elven alapul a bejelentkezés, a jelszó változtatás, és a felhasználónév változtatás is, kérések is. A regisztrációnál azt kell ellenőrizni, hogy a megadott felhasználó létezik e már vagy sem. A bejelentkezésnél egyértelműen ellenőrizni kell a felhasználó nevet, illetve a jelszót, hogy tartalmazza-e az adatbázis. A felhasználónév és jelszó változtatást is külön kezelem. Az elsőnél csak azt vizsgáljuk, hogy a létezik az a felhasználónév, amire változtatni szeretnénk. A jelszónál viszont, szükség van a régi jelszóra, majd az új jelszót is meg kell adni kétszer.

Mind a négy post kérésnél, ügyeltem a hibakezelésre. Számolva azzal a lehetőségekkel, hogy kezelje azt is, ha valami szerver oldali hiba lenne.

* Ha a beszúrás sikeres, a kód visszaküld egy 200-as státuszkódot, ami jelzi a sikeres regisztrációt.
* Ha az ellenőrzés során kiderül, hogy már létezik ilyen felhasználó, a kód visszaküld egy 400-as státuszkódot, ami jelzi, hogy a felhasználónév már foglalt.
* Ha bármilyen hiba történik a lekérdezések végrehajtása során, a kód visszaküld egy 500-as státuszkódot, ami tartalmazza a hibaüzenetet.

router.post('/get',(req,res,next)=>{

    const difficulty = req.body.difficulty;

    var query = "select \* from results where difficulty=?";

    connection.query(query,[difficulty],(err,results)=>{

        if(!err){

            return res.status(200).json(results);

        }

        else {

            return res.status(500).json(err);

        }

    });

});

Összes mező értéke a lekérdezi a results táblából, illetve a hozzátartozó nehézségi szint szerint.

var query = "select username,difficulty,min(time) as time from results where difficulty=? group by username";

Azokat mezőket kérdezi le, ahol a legrövidebb a megoldás ideje, illetve a hozzátartozó nehézségi szint szerint.

var query = "select username,difficulty,min(time) as time from results where difficulty=? and username=? group by username";

Azokat mezőket kérdezi le, ahol a username az aktuális felhasználó és ahol a legrövidebb a megoldás ideje, illetve a hozzátartozó nehézségi szint szerint.

CORS

ACORS (Cross-Origin Resource Sharing ) egy HTTP-fejlécen alapuló mechanizmus, amely lehetővé teszi a kiszolgáló számára, hogy a sajátján kívül bármely más eredetű (tartomány, séma vagy port) jelezze, ahonnan a böngészőnek engedélyeznie kell az erőforrások betöltését. A CORS egy olyan mechanizmusra is támaszkodik, amelynek segítségével a böngészők "preflight" kérést intéznek a cross-origin erőforrást tároló szerverhez annak ellenőrzése érdekében, hogy a szerver engedélyezi-e a tényleges kérést. Ebben az előzetes ellenőrzésben a böngésző olyan fejléceket küld, amelyek jelzik a tényleges kérés során használt HTTP-módszert és fejléceket.

LINK: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

## Kapcsolat a frontend és a backend között

.env fájl

PORT = 8080

DB\_PORT = 3306

DB\_HOST = localhost

DB\_USERNAME = api

DB\_PASSWORD = 123456

DB\_NAME = game

Itt határozzuk meg az alkalmazás környezeti változóinak értékeit.

PORT: Ez a környezeti változó az alkalmazás számára kijelölt portszámot tárolja. Az backend mindig a 8080 porton kommunikál frontend-del.

DB\_PORT: A MySQL adatbázissal történő kapcsolódáshoz használt portszám.

DB\_HOST: Az adatbázis szerver hosztneve vagy IP címe, amelyen a MySQL szolgáltatás fut.

DB\_USERNAME: Az adatbázis kapcsolódáshoz használt felhasználónév.

DB\_PASSWORD: Az adatbázis kapcsolódáshoz használt jelszó.

DB\_NAME: Az alkalmazás által használt adatbázis neve.

CONNNECTION

MySQL adatbázishoz történő kapcsolódást itt hozom létre. A kapcsolat beállításai az alkalmazás környezeti változóiból (.env) származnak, ami lehetővé teszi a konfiguráció dinamikus kezelését.

var connection = mysql.createConnection({

    port: process.env.DB\_PORT,

    host: process.env.DB\_HOST,

    user: process.env.DB\_USERNAME,

    password: process.env.DB\_PASSWORD,

    database: process.env.DB\_NAME

});

# Játék tesztelése

A képen szöveg, képernyőkép, szám, diagram látható

Automatikusan generált leírás

## Felhasználói visszajelzések

## Fejlesztési lehetőségek

Előfizetés integrálása az alkalmazásba

# Összefoglalás