**Eötvös Loránd Tudományegyetem**

**Informatikai Kar**

**Informatikatudományi Intézet**

**Média- és Oktatásinformatikai Tanszék**

Testreszabható kártyajáték

Szerző: Témavezető:

Schäffer Bálint Gaál Bence

Programtervező informatikus BSc. doktorandusz

**Budapest , 2025**

Tartalomjegyzék

[1. Bevezetés 1](#_Toc87521353)

[2. Felhasználói dokumentáció 2](#_Toc87521354)

[3. Fejlesztői dokumentáció 3](#_Toc87521355)

[4. Összefoglalás és további fejlesztési lehetőségek 4](#_Toc87521356)

[5. Irodalomjegyzék 5](#_Toc87521357)

[6. Melléklet 6](#_Toc87521358)

# Bevezetés

## Szakdolgozati témabejelentő

A szakdolgozatom egy viszonylag gyakori problémára nyújt megoldást. Családi vagy baráti összejöveteleken felmerülhet az az eset, hogy senkinél sincs francia kártya. Az emberek nagy százaléka rendelkezik már okostelefonnal, így könnyen megoldható a probléma. Ez az app egy olyan felületet biztosít a kártyázni vágyóknak, ami valóban úgy működik, mint egy pakli kártya. Alapbeállítások szerint a játékban nincsenek szabályok, az emberek saját becsületességére alapszik, mint ez a hagyományos kártyázás során is megszokhattuk. Az úgynevezett „host” fogja megmondani, hogy kinek mennyi kártyát oszt, a kártya felfedve, rejtve van az asztalon, egymáson, egymás felett (stb…). A játékmesternek is megvan a lehetősége, hogy játsszon, de felvehet csak egy osztó szerepet is. A játékhoz nincs szükség bejelentkezésre, csak egy kódot kell megadni, hogy egy közös lobbyba kerüljenek az emberek. A felület frontendjét Reactben fogom írni, emellett természetesen egy megfelelően működő modern technológiát használó backend is része lesz az alkalmazásnak.

## Témabejelentő módosításai

Az eredeti témabejelentőben az alkalmazást úgy terveztem, hogy nem lesz szükséges hozzá bejelentkezés, viszont a megtervezett funkciók egyértelműbb működése érdekében indokoltnak éreztem, hogy regisztrációhoz, illetve bejelentkezéshez legyen kötve a játék használata. A felhasználók, lobbyk és egyéb erre indokolt adatok az adatbázisban lesznek tárolva (részletek később).

## Motiváció és célközönség

Ez az app a kártyázni vágyó társaságoknak készült. Digitalizálja a kártyázás örömét, összekapcsolja az egymástól távol lévő ismerősöket is. Egyetemi éveim alatt velem is többször előfordult, hogy otthon felejtettük a kártyát egy baráti összejövetelre menet. Ekkor döntöttem el, hogy meg fogom valósítani ezt az appot.



1. ábra: Galaxis útikalauz stopposoknak könyv

Használjátok az automatikus számozást az ábrákhoz. Túl sok ábra esetén érdemes ábrajegyzéket beszúrni a dokumentumban az Irodalomjegyzék után. A folyó szövegben hivatkozzatok minden egyes ábrára és táblázatra is kereszthivatkozással (például így: 1. ábra, vagy táblázat esetén így: 1. táblázat).

1. táblázat: Példatáblázat

|  |  |  |
| --- | --- | --- |
| asdf |  |  |
| 1234 |  |  |

# Felhasználói dokumentáció

## Hardver és szoftver követelmények

### Hardver

Kliens oldalon nem túl nagyok a követelmények, ugyanis a legtöbb számítás szerver oldalon megy végbe. Az applikáció használatához olyan eszköz szükséges, melyek képesek böngészőt futtatni, illetve csatlakozni az internethez.

### Szoftver

A program használatához az alábbi böngészőkben biztos a működés:

* Mozilla Firefox
* Google Chrome
* Microsoft Edge

Ezek a böngészők és ezeknek legújabb verzióik ajánlottak.

## Oldalak

### Oldaltérkép

<KÉP>

### Oldalak részletes leírása

Bejelentkezés

A bejelentkező felületen értelemszerűen annak van lehetősége bejelentkezni, aki már hozott létre egy felhasználót a regisztrációs felületen. Sikeres bejelentkezés esetén egy választó felületen találja magát a felhasználó. A felhasználó helyes felhasználónév és jelszó párossal hozzáfér a többi funkcióhoz. Egy új felhasználónak lehetősége van átnavigálni a regisztráció oldalra is, ha még ezt nem tette meg korábban.

<KÉP>

Regisztráció

Ezen az oldalon egy regisztrációs felületet találunk, ahol egy email címet, felhasználónevet és jelszót kell megadni, az alábbi követelményekkel:

* Az email cím formálisan megfelel a követelményeknek (pl.: [minta.janos@gmail.com](mailto:minta.janos@gmail.com))
* Az email cím még nincs használatban
* A felhasználónév legalább 3 karakter hosszú
* A felhasználónév még nincs használatban
* A felhasználónévben nincs space
* A jelszó legalább 8 karakter hosszú
* A jelszónak tartalmaznia kell számot
* A jelszóban nem lehet space
* A jelszónak kis és nagy betűt is tartalmaznia kell

Ha ezeknek a kritériumoknak megfelel, akkor a regisztráció sikeres lesz.

<KÉP>

Választó felület

Ezen a felületen ki lehet választani, hogy a felhasználó egy meglévő lobbyhoz szeretne csatlakozni vagy egy újat hozna létre.

<KÉP>

Új játék

Az új játék oldalon kedvünkre testre szabhatjuk a játék tulajdonságait:

* Játék neve
* Maximum csatlakozó játékosok
* Kártyatípus (magyar, francia, uno)
* Kártyák törlése a pakliból
* Használt paklik száma
* Kezdő kártyák száma a játékos kezében
* Szükséges lesz-e, hogy kártyák kerüljenek az asztalra
* Kezdésnél az asztalon lévő látható kártyák száma
* Kezdésnél az asztalon lévő rejtett kártyák száma

Ha ezek ki vannak töltve lehetősége van a felhasználónak elmenteni ezt a konfigurációt, melyet az oldal jobb oldalán lévő presetek közt érhet el. Itt megtekintheti a saját maga, illetve más felhasználók által mentett konfigurációkat, amelyeket kattintással beilleszt a saját mezőibe.

<KÉPEK>

Csatlakozás játékhoz

Ezen a felületen csak egy beviteli mező található, ahol egy négy számból álló kódot kell begépelnie a felhasználónak. Siker esetén átkerül a lobbyba, ha viszont tévesen adja meg a kódot, akkor értesíti a felület.

<KÉP>

Játékasztal

Ez az oldal a játéknak az újonnan generált vagy már egy meglévő lobbyja. Ameddig a host el nem indítja a játékot, addig a játékosok várakoznak. Indítás után megkezdődik a játék, innentől a felhasználóktól függ a játék folyása.

<KÉPEK>

# Fejlesztői dokumentáció

## Követelményleírás

### Funkcionális követelmények

A program fő funkcionalitása az, hogy lehetővé tegye a játékosok számára, hogy egy kóddal egy lobbyba kerüljenek és egyidejűleg kommunikáljanak egymással. Ahhoz, hogy ez teljesüljön az alfunkcióknak is mind teljesülniük kell:

Felhasználókezelés

* **Regisztráció:** A felhasználók regisztrálhatnak egy felhasználónév, email, jelszó kétszeres megadásával
* **Bejelentkezés:** A helyes felhasználónév és jelszó megadásával a felhasználó be tud jelentkezni
* **Jelszavak:** A jelszavak többszörösen hashelve kerülnek az adatbázisba
* **Frissítés/Internet probléma:** A felületet nem befolyásolja a frissítés, ugyanis visszacsatlakozik.
* **Kijelentkezés:** A felhasználó a program minden állapotában ki tud jelentkezni

Lobby management

* **Lobby létrehozás:** Minden bejelentkezett játékos létrehozhat lobbyt
* **Csatlakozás lobbyba:** Minden bejelentkezett játékos csatlakozhat lobbyba
* **Kilépés lobbyból:** Egy lobbyban tartózkodó játékos ki tud lépni a lobbyból
* **Játék indítása:** A host képes elindítani a játékot
* **Játék vége:** A host képes befejezni a játékot
* **Játék újraindítása:** A host képes újraindítani a játékot
* **Host jog átruházása:** A host át tudja ruházni a host jogát más lobbyban lévő játékosra
* **Játékos kirúgása:** A host ki tud rúgni játékost a lobbyból
* **Frissítés:** A játékosok minden akciójuknál a lobbyban tartózkodók azonnal megkapják a lobby új állapotát

Játékmechanika

* **Kártyahúzás:** A játékos tud kártyát húzni a húzópakliból
* **Kártya megjátszás:** A játékos képes kártyát megjátszani valamely paklijából
* **Kártya áthelyezése:** A játékos képes kártyákat áthelyezni a saját paklijai közt
* **Paklik cseréje:** Két játékos meg tudja cserélni a kezükben lévő paklit
* **Kártya kézbe adása:** A host képes bármely játékos kezébe kártyát adni
* **Dobópakli bekeverése:** A host be tudja keverni a dobópaklit a húzópakliba.

Navigáció

* Minden oldal biztosít egyértelmű és elégséges navigációs lehetőségeket
* **Login:** A játékos minden felületről képes idejutni a kijelentkezésre kattintva, illetve a regisztráció elületről is
* **Register:** Ide a játékos a Login felületről tud eljutni
* **Létrehozás vagy csatlakozás:** A játékos regisztráció vagy bejelentkezés után és lobbyból való kilépés után ide jut
* **Játék létrehozása:** Ide a választó felületről jut a „New Game” gombra kattintva
* **Csatlakozás játékhoz:** Ide a választó felületről jut a „Join game” gombra kattintva
* **Játékasztal:** Ide egy helyesen megadott lobby kóddal jut a játékos

### Nem funkcionális követelmények

Teljesítmény/Hatákonyság

* A használat hálózati csatlakozást igényel
* A szervernek képesnek kell lennie egyszerre több játékos kezelésére (X lobby Y játékossal)
* Lobby frissítések alacsony késleltetéssel történnek (játékosok azonos állapotot lássanak)
* A játék adatai (kártyák, állapotok) a szerver memóriájában tárolódnak. Ez adatbázis szinten sokkal erőforrásigényesebb, lassabb lenne.

Felhasználói élmény

* Az alkalmazás reszponzív, a program igénybevételéhez megfelelő eszközökön minden tisztán látható
* Internetprobléma vagy oldalfrissítés esetén a játékost újracsatlakoztatja socket id és user id segítségével

*Biztonság*

* **Jelszó:** A jelszó bcryptjs segítségével már titkosítva kerül az adatbázisba, így szinte lehetetlen visszafejteni
* **Validáció:** A validáció szerver oldalon történik, kliens oldalra nem kerülnek titkosított adatok

Megbízhatóság

* **Hibaüzenetek:** A következő oldalakon hibaüzenetek jelenhetnek meg:
  + **Bejelentkezés:** Rossz belépési adatok
  + **Regisztráció:** A megszorításoknak nem elégséges adat
  + **Új játék:** Nincs elegendő kártya a játék elindításához a maximum játékos szám és a kezdő lapok függvényében
  + **Csatlakozás lobbyhoz:** A kód helytelen
  + **Csatlakozás lobbyhoz:** A lobby tele van, vagy a játék már elindult

Az itt felsorolt hibaüzeneteken kívül minden oldalon adatbázissal kapcsolatos hibaüzenetek is megjelenhetnek. Ezekről későbbi szekcióban lesz még szó.

Hordozhatóság

Minden internethez csatlakozó, böngészővel rendelkező eszközön biztosított a használat.

## Megvalósításhoz használt technológiák

Ebben a szekcióban a projektemhez felhasznált technológiákat fogom bemutatni, működésüket megmagyarázni.

Visual Studio Code

A fejlesztéshez a Visual Studio Code jelenlegi legfrissebb verzióját használtam (1.98.2). Ez egy nyílt forráskódú Microsoft által fejlesztett fejlesztő környezet, amely Windows, Linux és macOS operációs rendszereken is használható. Használata kényelmes a számtalan elérhető kiegészítő miatt. Git integrációval is rendelkezik, ami számomra is jelentősen megkönnyítette a verziókezelést.

HTML 5

A HTML (HyperText Markup Language) a weboldalak létrehozására és strukturálására használt szabványos nyelv. Címkék segítségével rendszerezi a weboldal tartalmát és elrendezését.

A HTML5, a legújabb verzió olyan kulcsfontosságú fejlesztéseket hoz, mint például a multimédia jobb támogatása, új szemantikai címkék és hatékony API-k, megkönnyítve és interaktívabbá téve a webfejlesztést.

DaisyUI / Tailwind

A DaisyUI egy Tailwind plugin, ami lehetővé teszi, hogy a fejlesztő komponensek, osztálynevek segítségével meggyorsítsa, hatékonyabbá tegye a fejlesztési folyamatot azáltal, hogy csökkentik a manuálisan megírt HTML és CSS kódok mennyiségét.

A Tailwind egy rendkívül modern és népszerű CSS keretrendszer. Sok apró, előre definiált stílusosztályt kínál, ezek közvetlenül hozzáadhatók az elemekhez. Az oldalak reszponzivitásáért nagyrészt ezek az osztályok felelnek. A projektemhez a daisyui 4-es és a tailwind 3-as verzióit használtam.

DaisyUI/Tailwind használata:

1. Csomagok letöltése: Ahhoz, hogy használni tudjuk először le kell töltenünk a csomagokat. Ha külön kezelve van szerver és kliens oldal, akkor a kliens oldali gyökér mappába. A letöltés elhelyezni a csomagokat a „node modules” mappába, illetve a „package.json” fájlban elhelyezni a nevét, verziószámát.

npm install tailwindcss daisyui

1. Konfig fájl struktúra: át kell írni a „tailwind.config.js” fájlt erre a struktúrára:

module.exports = {

content: ['./src/\*\*/\*.{js,ts,jsx,tsx}'],

plugins: [require('daisyui')],

};

1. Stíluslap kiegészítése: a használt stíluslapunk elejére be kell illeszteni a következő sorokat:

@tailwind base;

@tailwind components;

@tailwind utilities;

Ezek után a lépések után ReactJS környezetben már használhatjuk is mindkét csomagot.

<KÉP>

ReactJS

A React egy JavaScript könyvtár, amit a META közössége fejleszt. Célja a felhasználói felületek komponens alapú fejlesztése.

A fájlok kiterjesztése JSX, ami a JavaScript XML fájlkiterjesztése. Ezt a kódot Babel fordítja át JavaScript kóddá. A Babel egy olyan fordító, ami különböző jelölő vagy programozási nyelvet JavaScriptre fordít.

A projektemhez a React 18-as verzióját használtam a felhasználói felület megvalósítására. A React alapból nem tartalmaz page routing-ot, ezért ezt a problémát **React Routerrel** oldottam meg. Ennek használata olyan dolgokat tesz lehetővé mint:

* Hatékony navigáció több oldalas alkalmazások esetén
* URL-ek olvashatósága, ez keresőoptimalizáció szempontjából is előnyös
* Nincs teljes oldalfrissítés, hiszen csak az adott útvonalhoz tartozó komponensek frissülnek

Én a React alkalmazás létrehozásához a „Vite” nevű build toolt használtam, a következő paranccsal hoztam létre a projekt alapját:  
  
npm create vite@latest projeect-name -- --template react

Socket.io

A Socket.io egy olyan JavaSctript alapú könyvtár, amely lehetővé teszi a valós idejű kétirányú kommunikációt a szerver és a kliensek közt. Alapméretezetten a WebSocket protokollt használja, de szükség esetén más protokollt is képes használni (pl. HTTP), így a legtöbb böngészővel kompatibilis.

A projektemben azért választottam ezt a technológiát, mert gyors és késleltetésmentes kommunikációra volt szükségem, ugyanis a játékosok folyamatosan interaktálnak egymással. Fontos szempont volt a szobák (rooms) támogatása is, ugyanis így bizonyos adatokat, eseményeket csak egy szoba résztvevői kapnak meg. A kliens és a szerver tud eseményeket küldeni, hallgatni, ezeken keresztül adatot átvinni WebSocketek segítségével.

A technológiát az alábbi módokon használom fel:

* Játékos csatlakozások/újracsatlakozások: Minden felhasználó egyedi socket id-t kap már abban a pillanatban, hogy a Login felületre kerül. Ezzel az id-val van azonosítva később, ennek használatával lehet kezelni, ha esetleg frissítene az oldal és újracsatlakozna a felhasználó.
* Játék állapotainak frissítése: Minden játékasztalon történő állapotváltozás ezzel van lekezelve. A játék indításától, a kártyák húzásán, megjátszásán, pakolásán keresztül egészen a játék végéig egy update eseményen keresztül kapják a változtatásokat.
* Lobby létrehozás: Mint ahogy korábban említettem, a szobákat is támogatja, így a helyes lobby kód megadása után egy szobába kerül a felhasználó, el alapján kapja a frissítéseket és egyéb eseményeket.

A használathoz a következő lépések szükségesek:

1. Csomagok telepítése:
   1. Szerver oldalon: npm install socket.io express
   2. Kliens oldalon: npm install socket.io-client
2. Importok:
   1. Szerver:  
      const express = require("express");

const http = require("http");

const { Server } = require("socket.io");

* 1. Kliens:  
     import { io } from "socket.io-client";

Bcrypt

Jelszavakat sima szövegként tárolni nagy biztonsági kockázatot jelent egy esetleges adatbázis feltörése esetén. Ezt a problémát az én projektemben a bcyrptjs oldja meg. Ez egy teljes mértékben JavaScript alapú könyvtár (van natív C++ verzió is), mely egy rendkívül erőforrásigényes titkosítást biztosít, így ellenáll a brute-force támadásoknak.

Használatához a következő lépések szükségesek:

1. Csomag letöltése: npm install bcryptjs
2. Import: const bcrypt = require("bcryptjs");

Sequelize

A Sequelize egy modern TypeScript és Node.js ORM (Object-Relational Mapping) eszköz, amely támogatja többek között az Oracle, Postgres, MySQL, MariaDB, SQLite és SQL Server adatbázisokat. Olyan funkciókkal rendelkezik, mint a megbízható tranzakciókezelés, kapcsolatok, eager és lazy loading (előre vagy késleltetett betöltés), olvasási replikáció és további hasznos lehetőségek.

A Sequelize lényegesen egyszerűbb az adatbáziskezelés. Az adatok lekérése a létrehozott modellek statikus metódusai keresztül történik. Példa:

const user = await User.create({

        username: req.body.name,

        email: req.body.email,

        password: hashedPw,

      });

# Összefoglalás és további fejlesztési lehetőségek

# Irodalomjegyzék

# Melléklet