Tartalom

[1. Bevezetés 7](#_Toc196828015)

[1.1 Projekt célja 7](#_Toc196828016)

[1.2 Háttérinformációk 7](#_Toc196828017)

[1.3 Célcsoport és használati környezet 7](#_Toc196828018)

[1.4 Jogszabályi és technikai megfelelés 7](#_Toc196828019)

[2. Funkcionális követelmények 8](#_Toc196828020)

[2.1 Alapvető funkciók 8](#_Toc196828021)

[2.2 Kiegészítő funkciók 8](#_Toc196828022)

[2.3 Nem-funkcionális követelmények 9](#_Toc196828023)

[2.4 Példák a főbb funkcionális követelményekre 9](#_Toc196828024)

[2.5 Funkcionális követelmények összefoglalása 9](#_Toc196828025)

[3. Telepítési útmutató felhasználóknak 10](#_Toc196828026)

[3.1 Rendszerkövetelmények 10](#_Toc196828027)

[3.2 Telepítési lépések 10](#_Toc196828028)

[3.3 Telepítőfájl mellékelése 12](#_Toc196828029)

[4. Használati útmutató 13](#_Toc196828030)

[4.1 Felhasználó 13](#_Toc196828031)

[4.2 Admin felhasználó 21](#_Toc196828032)

[5. Technikai specifikációk 23](#_Toc196828033)

[5.1 Használt technológiák 23](#_Toc196828034)

[5.2 Rendszerkövetelmények 23](#_Toc196828035)

[5.3 Projekt fájl- és mappastruktúra 24](#_Toc196828036)

[5.4 Adatbázis Séma 27](#_Toc196828037)

[5.4.1 Felhasználói Táblák 27](#_Toc196828038)

[5.4.2 Játék Rendszer Táblák 29](#_Toc196828039)

[5.4.3 Közösségi Funkciók Táblái 30](#_Toc196828040)

[5.4.4 Hirdetés és Egyenlegfeltöltés Táblák 32](#_Toc196828041)

[5.4.5 Rendszertáblák 33](#_Toc196828042)

[5.4.6 Adatbázis-kapcsolatok diagram 34](#_Toc196828043)

[5.4.7 Megjegyzések 36](#_Toc196828044)

[6. Fejlesztési környezet – Részletes leírás (2025. április) 37](#_Toc196828045)

[6.1 Verziókezelés 37](#_Toc196828046)

[6.2 Fejlesztési eszközök 37](#_Toc196828047)

[6.3 Tesztelési környezet 39](#_Toc196828048)

[6.4 Frissítési és támogatási irányelvek 39](#_Toc196828049)

[6.5 Verziók összefoglaló táblázat (2025. április) 40](#_Toc196828050)

[7. Telepítési útmutató Fejlesztőknek 41](#_Toc196828051)

[7.1 Rendszerkövetelmények 41](#_Toc196828052)

[7.2 Telepítési lépések 42](#_Toc196828053)

[7.3 Electron Desktop Alkalmazás telepítési útmutató 43](#_Toc196828054)

[7.3.1 Telepítési útvonalak 45](#_Toc196828055)

[7.3.2 Setup fájl generálása 45](#_Toc196828056)

[7.3.3 Build futtatása 46](#_Toc196828057)

[7.3.4 Fontos megjegyzések 46](#_Toc196828058)

[8. Komponensek 47](#_Toc196828059)

[8.1 Általános Komponensek, alapséma 47](#_Toc196828060)

[8.1.1 Layoutok részletes leírása 47](#_Toc196828061)

[8.1.2 Vite hivatkozások magyarázata 50](#_Toc196828062)

[8.1.3 Adatbázis migrációk: 50](#_Toc196828063)

[8.1.4 Útvonalak (web.php) 51](#_Toc196828064)

[8.1.5 API útvonalak (api.php): 52](#_Toc196828065)

[8.1.6 Különbségek a web.php és api.php között 53](#_Toc196828066)

[8.2 Blackjack játék dokumentációja 54](#_Toc196828067)

[8.2.0 BlackJack játék layout (blackjack.blade.php) 54](#_Toc196828068)

[8.2.1 Áttekintés 54](#_Toc196828069)

[8.2.2 Backend működés 54](#_Toc196828070)

[8.2.3 Frontend működés 54](#_Toc196828071)

[8.2.4 Telepítés 55](#_Toc196828072)

[8.3 Coinflip játék dokumentációja 56](#_Toc196828073)

[8.3.0 Coinflip játék layout (coinflip.blade.php) 56](#_Toc196828074)

[8.3.1 Áttekintés 56](#_Toc196828075)

[8.3.2 Backend működés 56](#_Toc196828076)

[8.3.3 Frontend működés 57](#_Toc196828077)

[8.3.4 Telepítés 57](#_Toc196828078)

[8.4 Crash játék dokumentációja 58](#_Toc196828079)

[8.4.0 Crash játék layout (crash.blade.php) 58](#_Toc196828080)

[8.4.1 Áttekintés 58](#_Toc196828081)

[8.4.2 Backend működés 58](#_Toc196828082)

[8.4.3 Frontend működés 59](#_Toc196828083)

[8.4.4 Telepítés 59](#_Toc196828084)

[8.5 Rock Paper Scissors játék dokumentációja 60](#_Toc196828085)

[8.5.0 Rock Paper Scissors játék layout (rock.blade.php) 60](#_Toc196828086)

[8.5.1 Áttekintés 60](#_Toc196828087)

[8.5.2 Backend működés 60](#_Toc196828088)

[8.5.3 Frontend működés 61](#_Toc196828089)

[8.5.4 Telepítés 61](#_Toc196828090)

[8.6 Rulett játék dokumentációja 62](#_Toc196828091)

[8.6.0 Rulett játék layout (roulette.blade.php) 62](#_Toc196828092)

[8.6.1 Áttekintés 62](#_Toc196828093)

[8.6.2 Backend működés 62](#_Toc196828094)

[8.6.3 Frontend működés 63](#_Toc196828095)

[8.6.4 Telepítés 63](#_Toc196828096)

[8.7 Slot Machine játék dokumentációja 64](#_Toc196828097)

[8.7.0 Slot Machine játék layout (slot.blade.php) 64](#_Toc196828098)

[8.7.1 Áttekintés 64](#_Toc196828099)

[8.7.2 Backend működés 64](#_Toc196828100)

[8.7.3 Frontend működés 65](#_Toc196828101)

[8.7.4 Telepítés 66](#_Toc196828102)

[8.8 Mines játék dokumentációja 67](#_Toc196828103)

[8.8.0 Mines játék layout (mines.blade.php) 67](#_Toc196828104)

[8.8.1 Áttekintés 67](#_Toc196828105)

[8.8.2 Backend működés 67](#_Toc196828106)

[8.8.3 Frontend működés 68](#_Toc196828107)

[8.8.4 Telepítés 69](#_Toc196828108)

[8.9 Nem játék komponensek 70](#_Toc196828109)

[8.9.1 Profil statisztika oldal (pages/profile/statistics) – Komponensleírás 70](#_Toc196828110)

[8.9.2 Reklámfelület és hirdetéskezelő modul – Komponensleírás 70](#_Toc196828111)

[8.9.3 Leaderboard (Ranglista) komponens 73](#_Toc196828112)

[8.9.4 Barátok (Friends) és Üzenetek – Komponensleírás 73](#_Toc196828113)

[8.9.4 Fórum komponens – Komponensleírás 74](#_Toc196828114)

[9. Fejlesztői API és Fájlszintű Dokumentáció 76](#_Toc196828115)

[9.0 Layoutok – Fejlesztői, kódszintű leírás 76](#_Toc196828116)

[9.1 Játékok Technikai Dokumentációja 79](#_Toc196828117)

[9.1.1 Blackjack játék – Részletes fejlesztői dokumentáció 80](#_Toc196828118)

[9.1.2 Coinflip (Érmedobás) játék – Részletes fejlesztői dokumentáció 84](#_Toc196828119)

[9.1.3 Crash játék – Részletes fejlesztői dokumentáció 88](#_Toc196828120)

[9.1.4 Rock-Paper-Scissors Játék – Részletes fejlesztői dokumentáció 92](#_Toc196828121)

[9.1.5 Rulett Játék – Részletes fejlesztői dokumentáció 95](#_Toc196828122)

[9.1.6 Slot Machine (Játékgép) – Részletes fejlesztői dokumentáció 99](#_Toc196828123)

[9.1.7 Mines (Aknakereső) Játék – Részletes fejlesztői dokumentáció 103](#_Toc196828124)

[9.2 API és Web útvonalak részletes leírása 107](#_Toc196828125)

[9.2.1 API útvonalak (api.php) 107](#_Toc196828126)

[9.2.2 Web útvonalak (web.php) 110](#_Toc196828127)

[9.2.3 Útvonalhivatkozások 111](#_Toc196828128)

[9.2.4 Fontos megjegyzések 112](#_Toc196828129)

[9.3 Nem játék elemek fejlesztői dokumentációja 113](#_Toc196828130)

[9.3.1 Fejlesztői dokumentáció: Profil statisztika oldal 113](#_Toc196828131)

[9.3.2 Reklámfelület és hirdetéskezelő – Fejlesztői dokumentáció 114](#_Toc196828132)

[9.3.4 Leaderboard – Fejlesztői dokumentáció 117](#_Toc196828133)

[9.3.5 Barátok és üzenetek – Fejlesztői dokumentáció 118](#_Toc196828134)

[9.3.6 Fórum modul – Fejlesztői dokumentáció 120](#_Toc196828135)

[10. Tesztelési módszerek 123](#_Toc196828136)

[10.1 Általános tesztelési stratégiák 123](#_Toc196828137)

[10.1.1 Tesztelési megközelítés 123](#_Toc196828138)

[10.1.2 Tesztdokumentáció 123](#_Toc196828139)

[10.2 Tesztkörnyezet konfiguráció 123](#_Toc196828140)

[10.2.1 Követelmények 123](#_Toc196828141)

[10.2.2 Teszt adatbázis beállítása 123](#_Toc196828142)

[10.2.3 Előfeltételek minden teszthez 124](#_Toc196828143)

[10.3 Coinflip tesztelés 125](#_Toc196828144)

[10.3.1 Coinflip játék alapvető funkcionalitás tesztje 125](#_Toc196828145)

[10.3.2 Coinflip hibakezelés tesztje 126](#_Toc196828146)

[10.4 Admin funkcionalitás tesztelés 127](#_Toc196828147)

[10.4.2 Felhasználó kezelés tesztje 129](#_Toc196828148)

[10.5 Fórum és komment funkció tesztelés 131](#_Toc196828149)

[11. Karbantartás és továbbfejlesztés 132](#_Toc196828150)

[11.1 Rendszeres karbantartási feladatok 132](#_Toc196828151)

[11.2 Új funkciók és játékok bevezetése 133](#_Toc196828152)

[11.3 Technológiai továbbfejlesztés 133](#_Toc196828153)

[11.4 Felhasználói visszajelzések kezelése 134](#_Toc196828154)

[11.5 Jogszabályi és etikai megfelelés 134](#_Toc196828155)

[11.6 Változáskezelés, verziózás 134](#_Toc196828156)

[11.7 Karbantartási ütemterv, felelősségek 135](#_Toc196828157)

[12. Kapcsolat 136](#_Toc196828158)

**Projekt Dokumentáció**

**Spin It Clean Casino Project**

*Bilik Balázs, Krammer Erik, Sitku Péter*

# 1. Bevezetés

## 1.1 Projekt célja

Egy online kaszinó-szimulátor létrehozása, ahol felhasználók virtuális pénzzel játszhatnak különböző játékokat, nyomon követhetik eredményeiket és mások statisztikáit. A rendszer reklámfelületeket integrál, és **nem használ valós pénzt**, így kockázatmentes szórakoztató célokat szolgál.

## 1.2 Háttérinformációk

A projekt célja, hogy egy biztonságos, platformfüggetlen környezetet biztosítson játékosoknak, ahol kipróbálhatják szerencsejáték-stratégiáikat anélkül, hogy pénzügyi kockázatot vállalnának. A csapat a Laravel keretrendszer rugalmasságára és többplatformos kompatibilitására épített.

## **1.3 Célcsoport és használati környezet**

A dokumentáció minden olyan felhasználónak, fejlesztőnek és adminisztrátornak szól, aki a Casino projektet telepíteni, üzemeltetni, fejleszteni vagy tesztelni kívánja. A rendszer oktatási és szórakoztató célokat szolgál, nem kezel valós pénzt, így kockázatmentes környezetet biztosít. A projekt platformfüggetlen, támogatja a modern böngészőket, valamint önálló asztali alkalmazásként is telepíthető Electron segítségével.

## **1.4 Jogszabályi és technikai megfelelés**

A projekt fejlesztése során figyelembe vettük a nemzetközi szerencsejáték-technikai ajánlásokat (pl. SIFS, GLI-12), különös tekintettel a játékosvédelemre, adatnaplózásra, önkorlátozási lehetőségekre és a rendszer átláthatóságára. A rendszer minden jelentős eseményt naplóz, valamint a játékos tranzakciók átlátható követését.

# 2. Funkcionális követelmények

A funkcionális követelmények a rendszer azon képességeit és szolgáltatásait írják le, amelyek közvetlenül támogatják a felhasználók, adminisztrátorok és a rendszer működését. Ezek a követelmények határozzák meg, hogy a Casino alkalmazás mit tud, milyen főbb modulokat és folyamatokat tartalmaz, és hogyan támogatja a felhasználói élményt.

## 2.1 Alapvető funkciók

* **Felhasználói regisztráció és bejelentkezés**  
  A rendszer lehetőséget ad új felhasználók regisztrációjára, valamint meglévő fiókokkal történő biztonságos bejelentkezésre.
* **Virtuális pénzkezelés**  
  Minden felhasználó rendelkezik egyenleggel, amelyet játékokban használhat fel, és amely reklámok megtekintésével, játéknyereményekkel növelhető.
* **Játékmodulok**  
  Többféle kaszinó játék érhető el (pl. blackjack, rulett, slot, crash, coinflip, mines, rock-paper-scissors), mindegyik saját szabályrendszerrel és statisztikával.
* **Nyeremények és statisztikák követése**  
  A rendszer minden játék eredményét naplózza, a felhasználók részletes statisztikákat láthatnak saját profiljukban.
* **Reklámfelületek integrációja**  
  A felhasználók reklámokat nézhetnek, amelyekért jutalmat kapnak, az adminisztrátorok kezelhetik a hirdetéseket.
* **Közösségi funkciók**  
  Barátlista, üzenetküldés, fórum, ranglista, ahol a felhasználók egymással kapcsolatba léphetnek, versenyezhetnek.

## 2.2 Kiegészítő funkciók

* **Játékos viselkedés elemzése**  
  A rendszer elemzi a játékosok aktivitását, szokásait, és személyre szabott statisztikákat, ajánlásokat jelenít meg.
* **Barát- és üzenetkezelő**  
  Lehetőség van barátok hozzáadására, barátkérelmek kezelésére, privát üzenetek váltására.
* **Fórum és hozzászólások**  
  Nyilvános és privát posztok létrehozása, hozzászólás, moderáció (admin vagy poszttulajdonos által).
* **Adminisztrációs modul**  
  Felhasználók felfüggesztése, visszaállítása, végleges törlése; hirdetések hozzáadása, szerkesztése, törlése; statisztikák és jelentések megtekintése.
* **Leaderboard (Ranglista)**  
  A legjobb játékosok rangsorolása egyenleg, győzelmek vagy egyéb statisztikák alapján.
* **Profil és beállítások**  
  Személyes adatok, jelszó, avatar, egyéni statisztikák, önkorlátozás és önkizárás beállítása.

## 2.3 Nem-funkcionális követelmények

* **Teljesítmény**  
  A rendszernek egyszerre legalább 100 aktív felhasználót kell kiszolgálnia akadás nélkül.
* **Biztonság**  
  Minden adatátvitel titkosított (HTTPS), a jelszavak biztonságos hash-eléssel tárolódnak, CSRF-védelem minden űrlapon.
* **Platformfüggetlenség**  
  A rendszer elérhető webes (Chrome, Firefox, Brave), asztali (Electron, Windows/macOS), és mobil böngészőből is.
* **Karbantarthatóság**  
  A kód moduláris, jól dokumentált, minden fő funkcióhoz automata és manuális tesztesetek tartoznak.
* **Adatvédelem**  
  GDPR kompatibilis adatkezelés, felhasználói adatok törlésének és exportálásának lehetősége.

## 2.4 Példák a főbb funkcionális követelményekre

* A felhasználó regisztrálhat, bejelentkezhet.
* A felhasználó bármely játékban tétet tehet, a tét levonódik az egyenlegéből, a játék eredményétől függően nő vagy csökken az egyenlege.
* A felhasználó barátokat kereshet, barátkérelmet küldhet, elfogadhat vagy elutasíthat.
* Az adminisztrátor bármely felhasználót felfüggeszthet, visszaállíthat vagy végleg törölhet.
* A reklámnéző felületen a felhasználó hirdetéseket nézhet, a megtekintés után automatikusan jóváíródik a jutalom.
* A fórumon a felhasználó új posztot hozhat létre (privát vagy nyilvános), hozzászólhat, saját vagy admin által törölhetőek a posztok és kommentek.
* A rendszer minden jelentős eseményt naplóz, a statisztikák és naplók exportálhatók auditálás céljából.

## 2.5 Funkcionális követelmények összefoglalása

A Casino projekt minden fő funkciója úgy lett megtervezve, hogy:

* könnyen bővíthető legyen új játékokkal, közösségi és admin modulokkal,
* a felhasználók számára biztonságos, átlátható és élvezetes játékélményt nyújtson,
* az adminisztrátorok számára teljes körű kontrollt adjon a rendszer felett,
* megfeleljen a modern webes és asztali alkalmazásokkal szemben támasztott elvárásoknak.

# 3. Telepítési útmutató felhasználóknak

## 3.1 Rendszerkövetelmények

* **Szerveroldal**: PHP 8.0+, Composer, Node.js, XAMPP
* **Kliensoldal**:
  + Web: Modern böngészők (Chrome, Firefox).
  + Asztali: Windows 10+ / macOS 10.15+.
  + Mobil: Android 8.0+ / iOS 13+.
  + **Futtatási környezet**: Microsoft Visual Studio Code

## 3.2 Telepítési lépések

1. Klónozd a repót:

git clone https://github.com/SitkuPeter/VizsgaRemek\_BB\_KE\_SP.git

1. Függőségek telepítése:

composer install   
npm install

1. Környezeti változók beállítása (.env fájl).

Amennyiben először indítjuk el a projektet létre kell hozni a gyökérkönyvtárba egy .env fájlt, ezt megtehetjük az .env.example fájl tartalmának kimásolásával és beillesztésével egy általunk létrehozott .env fájlba

Fontos! hogy ekkor már futnia kell a XAMPP control panelen az Apache és MySql szolgáltatásoknak, valamint egy lokális APP\_KEY-t is kell generálnunk, amit az alábbi bash paranccsal tehetünk meg:

php artisan key:generate

1. Adatbázis migrációk futtatása:

php artisan migrate

php artisan migrate:fresh --seed

1. Indítsd el a szervert:

php artisan serve   
npm run dev

## 3.3 Telepítőfájl mellékelése

A dokumentációhoz a felhasználó minden esetben kap egy előre legenerált, Windowsra készült telepítőfájlt is (például: **KasziNo Setup 1.0.0.exe**), amelyet az Electron alkalmazás buildelése során hozunk létre. Ez a telepítő a projekt kiadásakor a dokumentációval együtt kerül átadásra.

A telepítőfájl a következő jellemzőkkel rendelkezik:

* **Könnyen felismerhető név** (pl. „KasziNo Setup 1.0.0.exe”)
* **Egyszerű telepítés**: dupla kattintással elindítható, végigviszi a telepítési folyamatot
* **Alapértelmezett telepítési hely**:  
  C:\Users\<felhasználónév>\AppData\Local\Programs\KasziNo  
  (A telepítő varázslóban a felhasználó módosíthatja is ezt az útvonalat.)
* **Hordozható verzió**: a win-unpacked mappában található EXE fájl közvetlenül is futtatható telepítés nélkül, de a legtöbb felhasználónak a Setup EXE használata javasolt.

A telepítőfájl a projekt buildelése után például az alábbi helyen található meg:

C:\CustomBuilds\MyElectronApp\KasziNo Setup 1.0.0.exe

A felhasználó számára tehát a dokumentációval együtt átadandó:

* maga a dokumentáció (PDF, HTML vagy egyéb formátumban)
* a legenerált telepítőfájl (EXE)
* (opcionálisan) a hordozható, telepítés nélküli futtatható verzió (win-unpacked mappa tartalma)

**Fontos!**

Mivel a projekt jelenleg teszt builddel rendelkezik nincs interneten keresztül elérhető url, így az asztali alkalmazás működése érdekében elengedhetetlen, hogy fusson a szerver az eszközünkön (3.2 5. lépés), és a Blackjack Játék futtatása is elengedhetetlen (8.2.4)

# 4. Használati útmutató

## 4.1 Felhasználó

1. **A képen félkarú rabló, szöveg, képernyőkép látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Regisztrálj egy felhasználót a /register oldalon,**
2. **A képen szöveg, félkarú rabló, képernyőkép, fedett pályás látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Jelentkezz be a /login oldalon.**
3. **A képen képernyőkép, szöveg, Multimédiás szoftver, szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Válassz játékot a játék menüből.**
4. **A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Virtuális pénzedet felhasználva játsz.**
5. **A képen szöveg, képernyőkép, diagram, szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, Betűtípus látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Kövesd a statisztikákat a profilodban.**
6. **Versenyezz a ranglistán**

A képen képernyőkép, szöveg, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

1. **A képen szöveg, képernyőkép, Multimédiás szoftver, szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Jelölj be barátokat és kövesd a statisztikáikat**

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, Betűtípus, embléma látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

1. **A képen kör, sárga látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Üzenj a barátaidnak**

**A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**

1. **A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Posztolj a fórumra, másoknak, vagy szimplán a barátaidnak**

**A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**

1. **A képen szöveg, képernyőkép, Multimédiás szoftver, szoftver látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.Kommentelj a posztok alá**
2. A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

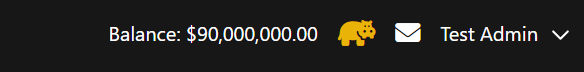
   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, Betűtípus, képernyőkép, Grafika látható

   Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**Szerezz eljátszható pénzt reklámok megtekintésével**

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## 4.2 Admin felhasználó

1. **Az admin dashboard megnyitásához kattints a Víziló ikonra**
2. **Menedzseld a felhaszálókat**

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

1. **Menedzseld a reklámokat**

**A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**

# 5. Technikai specifikációk

## 5.1 Használt technológiák

* **Backend**: Laravel (PHP) – rugalmas és biztonságos API-k készítéséhez.
* **Frontend**:
  + Web: Laravel Frontend, Vite
  + Asztali: Electron.js
  + Mobil: Laravel Frontend, Vite
* **Adatbázis**: MySQL (strukturált adatokhoz).

## 5.2 Rendszerkövetelmények

* **Szerveroldal**: PHP 8.0+, Composer, Node.js, XAMPP
* **Kliensoldal**:
  + Web: Modern böngészők (Chrome, Firefox).
  + Asztali: Windows 10+ / macOS 10.15+.
  + Mobil: Android 8.0+ / iOS 13+.

## 5.3 Projekt fájl- és mappastruktúra

Az alábbiakban bemutatjuk a Casino projekt főbb mappáit és azok szerepét. Ez a struktúra segíti a fejlesztőket az eligazodásban, a kód gyors áttekintésében és a fejlesztési folyamatokban.

KASZINO/

│

├── app/ # Laravel backend fő mappa

│ ├── Helpers/ # Segédfüggvények, util osztályok

│ ├── Http/ # Kontroller, Middleware, Request osztályok

│ ├── Models/ # Adatbázis modellek (pl. User, GameLog)

│ ├── Observers/ # Modellek eseményfigyelői

│ ├── Policies/ # Jogosultsági szabályok

│ ├── Providers/ # Laravel service providerek

│ ├── Services/ # Üzleti logika, szolgáltatások

│ └── View/ # (Ha van) nézetlogika

│

├── bootstrap/ # Laravel bootstrap fájlok

├── config/ # Konfigurációs fájlok (adatbázis, mail, stb.)

├── database/ # Migrációk, seederek, factory-k

├── deckofcards-master/ # Külső Python alapú kártya API (Blackjack-hez)

├── electron-app/ # Asztali alkalmazás forrása (Electron)

├── node\_modules/ # NPM csomagok (automatikusan generált)

├── public/ # Publikus elérhető assetek (index.php, képek, stb.)

│

├── resources/ # Frontend források

│ ├── css/ # CSS fájlok (globális és komponens szintű)

│ └── js/ # JavaScript fájlok (játékokhoz, funkciókhoz)

│ ├── app.js

│ ├── BJScript.js

│ ├── bootstrap.js

│ ├── coinflip.js

│ ├── crash.js

│ ├── forum.js

│ ├── mines.js

│ ├── post.js

│ ├── rockpaperscissors.js

│ ├── roulette.js

│ └── slot.js

│

├── resources/views/ # Blade nézetek (felhasználói felület)

│ ├── layouts/ # Alap layoutok (app, game, navigation, stb.)

│ ├── pages/ # Oldalak (admin, fórum, játékok, profil, stb.)

│ ├── admin/

│ │ └── ads/ # Admin hirdetéskezelés nézetei

│ ├── forum/ # Fórum oldalak

│ ├── games/ # Játék oldalak (blackjack, coinflip, stb.)

│ └── profile/ # Profil oldalak

│

├── routes/ # Laravel útvonalak

│ ├── api.php # API végpontok (JSON, játéklogika)

│ ├── auth.php # Authentikációs útvonalak

│ ├── console.php # Parancssoros (artisan) útvonalak

│ └── web.php # Webes (Blade) oldalak útvonalai

│

├── storage/ # Cache, logok, feltöltött fájlok

├── tests/ # Tesztek (unit, feature)

├── vendor/ # Composer csomagok (automatikusan generált)

│

├── .env, .gitignore, README.md, artisan, composer.json, package.json, stb. # Gyökérfájlok

**Főbb mappák és fájlok rövid magyarázata**

* **app/**: A backend logika, kontrollerek, modellek, segédfüggvények, jogosultsági szabályok és szolgáltatások helye.
* **resources/js/**: Minden játékhoz és funkcióhoz tartozó JavaScript logika (pl. mines.js, crash.js, slot.js).
* **resources/views/**: Blade sablonok, komponensek, teljes oldalnézetek.
  + **layouts/**: Alapvető elrendezések (pl. navigáció, játék layout).
  + **pages/games/**: Minden játék önálló nézete (blackjack.blade.php, coinflip.blade.php, stb.).
  + **pages/forum/**: Fórumhoz kapcsolódó nézetek.
  + **pages/admin/ads/**: Admin felület hirdetéskezelő nézetei.
* **routes/**:
  + **api.php**: Backend API végpontok (játéklogika, JSON kommunikáció).
  + **web.php**: Webes oldalakat kiszolgáló útvonalak (Blade nézetek).
  + **auth.php, console.php**: Authentikáció, artisan parancsok.
* **deckofcards-master/**: Külső Python alapú kártya API a blackjack játékhoz.
* **electron-app/**: Asztali alkalmazás (Electron) forráskódja.
* **public/**: Közvetlenül elérhető statikus fájlok, képek, index.php.
* **tests/**: Automata backend és frontend tesztek.
* **vendor/**, **node\_modules/**: Csomagkezelő által generált fájlok (ne szerkeszd kézzel).
* **.env, composer.json, package.json, README.md, artisan**: Konfigurációs és futtatási segédfájlok.

## 5.4 Adatbázis Séma

A Casino alkalmazás komplex adatbázis-szerkezettel rendelkezik, amely több modulból áll: felhasználókezelés, játékok, közösségi funkciók, hirdetéskezelés és statisztikák. Az adatbázisban a következő fő táblák és kapcsolatok találhatók:

### 5.4.1 Felhasználói Táblák

**users**

A rendszer központi felhasználói táblája, amely tartalmazza a játékosok profiladatait és egyenlegét.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| is\_admin | boolean | Admin jogosultság jelölése |
| name | string | Felhasználó neve |
| email | string | Egyedi email cím |
| email\_verified\_at | timestamp | Email megerősítés időpontja |
| password | string | Titkosított jelszó |
| balance | decimal(10,2) | Felhasználó egyenlege |
| remember\_token | string | "Emlékezz rám" token |
| created\_at, updated\_at | timestamp | Létrehozás és módosítás időpontjai |
| deleted\_at | timestamp | Soft delete időpontja |
| suspended\_by | foreign key | Felfüggesztést végrehajtó felhasználó ID-ja |
| restored\_by | foreign key | Visszaállítást végrehajtó felhasználó ID-ja |
| destroyed\_by | foreign key | Törlést végrehajtó felhasználó ID-ja |

**Kapcsolatok:**

* Egy felhasználóhoz több game\_stats, game\_logs, posts, comments, messages, balance\_uploads tartozhat

**sessions**

Felhasználói munkamenetek tárolása.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | string | Munkamenet egyedi azonosítója |
| user\_id | foreign key | Felhasználó azonosítója |
| ip\_address | string | IP cím (max 45 karakter) |
| user\_agent | text | Böngésző/eszköz információk |
| payload | longtext | Munkamenet adatok |
| last\_activity | integer | Utolsó aktivitás időbélyege |

### 5.4.2 Játék Rendszer Táblák

**game\_logs**

Minden egyes játékmenet részletes naplózása.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| user\_id | foreign key | Játékos azonosítója |
| game\_type | string | Játék típusa (blackjack, coinflip, stb.) |
| bet\_amount | decimal(10,2) | Megtett tét összege |
| win\_amount | decimal(10,2) | Nyeremény összege (alapértelmezett: 0) |
| is\_win | boolean | Nyertes játék jelölése |
| game\_data | json | Játékspecifikus adatok (kártyák, szorzók, stb.) |
| played\_at | timestamp | Játék időpontja |

**game\_stats**

Felhasználók játékstatisztikáinak összesítése.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| user\_id | foreign key | Felhasználó azonosítója |
| total\_games\_played | integer | Összes játszott játék száma |
| total\_wins | integer | Nyertes játékok száma |
| total\_losses | integer | Vesztes játékok száma |
| total\_winnings | decimal(15,2) | Összes nyeremény |
| total\_losses\_amount | decimal(15,2) | Összes veszteség |
| last\_played\_at | timestamp | Utolsó játék időpontja |
| game\_statistics | json | Játékokra bontott részletes statisztikák |

### 5.4.3 Közösségi Funkciók Táblái

**friendships**

Felhasználók közötti baráti kapcsolatok.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| sender\_id | foreign key | Küldő felhasználó azonosítója |
| receiver\_id | foreign key | Fogadó felhasználó azonosítója |
| status | enum | Kapcsolat állapota (pending, accepted, declined) |

**Egyedi kulcs:** (sender\_id, receiver\_id) kombinációra

**friend\_requests**

Függőben lévő barátfelkérések.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| sender\_id | foreign key | Kérés küldőjének azonosítója |
| receiver\_id | foreign key | Kérés fogadójának azonosítója |
| status | enum | Kérés állapota (pending, accepted, rejected) |

**messages**

Felhasználók közötti közvetlen üzenetek.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| sender\_id | foreign key | Küldő felhasználó azonosítója |
| receiver\_id | foreign key | Fogadó felhasználó azonosítója |
| message | text | Üzenet tartalma |
| is\_read | boolean | Olvasottsági állapot |

**posts**

Fórum bejegyzések.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| user\_id | foreign key | Szerző azonosítója |
| title | string | Bejegyzés címe |
| body | text | Bejegyzés tartalma |
| is\_private | boolean | Privát állapot jelölése |
| deleted\_by | foreign key | Törlést végrehajtó admin azonosítója |

**Kapcsolatok:**

* Egy bejegyzéshez több comments kapcsolódhat

**comments**

Fórum bejegyzésekhez tartozó hozzászólások.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| post\_id | foreign key | Bejegyzés azonosítója |
| user\_id | foreign key | Hozzászóló azonosítója |
| body | text | Hozzászólás tartalma |
| deleted\_by | foreign key | Törlést végrehajtó admin azonosítója |

### 5.4.4 Hirdetés és Egyenlegfeltöltés Táblák

**advertisements**

Rendszerben elérhető hirdetések, amiket megtekintve a felhasználók egyenleget szerezhetnek.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| user\_id | foreign key | Feltöltő azonosítója |
| updated\_by | foreign key | Frissítést végző felhasználó azonosítója |
| title | string | Hirdetés címe |
| description | text | Hirdetés leírása |
| duration\_seconds | integer | Időtartam másodpercben |
| reward\_amount | decimal(10,2) | Jutalom összege |
| image | string | Eredeti kép neve |
| media\_type | string | Média típusa (image/video) |
| image\_path | string | Kép elérési útvonala |
| video\_path | string | Videó elérési útvonala |
| is\_active | boolean | Aktív állapot |
| deleted\_by | foreign key | Törlést végrehajtó admin azonosítója |

**balance\_uploads**

Felhasználók egyenlegfeltöltései hirdetések megtekintésével.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| user\_id | foreign key | Felhasználó azonosítója |
| advertisement\_id | foreign key | Megtekintett hirdetés azonosítója |
| duration\_watched | integer | Megtekintett időtartam (mp) |
| reward | decimal(10,2) | Jóváírt jutalom |
| status | string | Feltöltés állapota (pending, completed, stb.) |

### 5.4.5 Rendszertáblák

**admins**

Admin felhasználók táblája.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| created\_at, updated\_at | timestamp | Létrehozás és módosítás időpontjai |

**personal\_access\_tokens**

API tokenek Sanctum-hoz.

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| id | bigint | Elsődleges kulcs |
| tokenable\_type, tokenable\_id | morphs | Polimorfikus kapcsolat tokenhez |
| name | string | Token neve |
| token | string | Titkosított token (max 64 karakter) |
| abilities | text | Token jogosultságok |
| last\_used\_at | timestamp | Utolsó használat időpontja |
| expires\_at | timestamp | Lejárat időpontja |

**cache és cache\_locks**

Rendszer gyorsítótár táblák.

### 5.4.6 Adatbázis-kapcsolatok diagram

users

├── game\_stats (1:1)

├── game\_logs (1:N)

├── posts (1:N)

├── comments (1:N)

├── messages\_sent (1:N)

├── messages\_received (1:N)

├── balance\_uploads (1:N)

├── advertisements (1:N)

├── friendships\_sender (1:N)

├── friendships\_receiver (1:N)

└── friend\_requests (1:N)

posts

└── comments (1:N)

advertisements

└── balance\_uploads (1:N)

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

### 5.4.7 Megjegyzések

* A rendszer a legtöbb táblánál soft delete-et használ (deleted\_at mező)
* A pénzösszegek 10,2 formátumban vannak tárolva (10 számjegy, 2 tizedesjegy)
* A game\_data és game\_statistics JSON mezők rugalmas adattárolást biztosítanak a különböző játéktípusokhoz
* A barátságok és üzenetek elkülönülnek, de összekapcsolhatók a sender\_id és receiver\_id mezőkkel

# 6. Fejlesztési környezet – Részletes leírás (2025. április)

A fejlesztési környezet részletes bemutatása elengedhetetlen a projekt hosszú távú karbantartásához és bővíthetőségéhez. Az alábbiakban minden fontosabb fejlesztői és tesztelési platform, eszköz, valamint a jelenlegi (2025. áprilisi) verziók kerülnek felsorolásra.

## 6.1 Verziókezelés

* **Verziókezelő rendszer**: Git
  + Ajánlott GUI: GitHub Desktop, Sourcetree vagy VS Code beépített Git
  + Tároló: publikus GitHub repository

## 6.2 Fejlesztési eszközök

**Fő fejlesztői környezet**

* **Operációs rendszer**:
  + *Windows 11* (ajánlott, stabil):
    - **23H2** (OS build: 22631.3527) – jelenleg a legelterjedtebb stabil verzió
    - **24H2** (OS build: 26100.3775) – 2025. áprilisi frissítés, fokozatos bevezetés alatt, új funkciókkal és teljesítményjavításokkal
  + *Alternatívák*:
    - Windows 10 (korlátozott támogatás), macOS 12+ (Monterey), Ubuntu 22.04+ (Linux)
* **Fejlesztői szoftverek**:
  + **Visual Studio Code** (ajánlott, stabil): 1.89.x (2025 áprilisi verzió)
  + **DBeaver** – adatbázis-kezeléshez
  + **Postman** – API teszteléshez
  + **Node.js** – 20.x LTS vagy újabb
  + **Composer** – 2.7.x vagy újabb (PHP csomagkezelő)
  + **XAMPP** – 8.1.x vagy újabb (PHP, MySQL, Apache)
  + **Python** – 3.12.x (DeckOfCards API-hoz)
  + **npm** – 10.x vagy újabb

**Böngészők – Jelenlegi stabil verziók (2025. április)**

* **Google Chrome**
  + **Stabil verzió**: 135.0.7049.96 (Windows, Mac, Linux)[4](https://chromereleases.googleblog.com/2025/04/)
  + **Beta**: 136.x (elérhető, de nem ajánlott éles fejlesztéshez)
* **Mozilla Firefox**
  + **Stabil verzió**: 137.0.2 (2025. április 23.)
* **Brave**
  + **Stabil verzió**: 1.77.101 (Chromium 135.0.7049.115, 2025. április 23.)

*Megjegyzés*:  
A projekt minden funkcióját ezekben a böngészőkben teszteljük, és csak ezekhez garantált a teljes kompatibilitás.

**Asztali alkalmazás (Electron)**

* **Electron**: 29.x vagy újabb (ajánlott)
* **Build eszköz**: electron-builder (ajánlott)
* **Telepítő generálása**:
  + A buildelt setup EXE/DMG/DEB fájlok a electron-app/dist/ mappába kerülnek
  + Windows telepítés: alapértelmezett hely C:\Users\<felhasználónév>\AppData\Local\Programs\<AppNév>, de a build könyvtárban (pl. C:\custombuilds\myelectronapp-) is megtalálható a setup és a win-unpacked mappa (hordozható futtatás)[7.3.1]

**Mobil fejlesztéshez (opcionális)**

* **Android**: 10.0 (Q) vagy újabb, Chrome/Firefox mobil verziókkal
* **iOS**: 14.0 vagy újabb, Safari/Chrome/Brave mobil verziókkal

## 6.3 Tesztelési környezet

* **Teszt adatbázis**: különálló, frissített tesztadatbázis (pl. casino\_test)
* **Automata tesztelés**: Laravel PHPUnit, Cypress (frontend)
* **Manuális tesztelés**: minden főbb böngészőben és platformon, a 10. fejezet szerinti tesztdokumentáció alapján

## 6.4 Frissítési és támogatási irányelvek

* **Windows 11**: A projektet minden újabb 23H2, illetve 24H2 builden tesztelni kell, a támogatás a hivatalos Microsoft támogatási ciklushoz igazodik2.
* **Böngészők**: Az aktuális stabil Chrome, Firefox, Brave verziókhoz igazítjuk a támogatást. A projekt minden főbb funkcióját ezekkel teszteljük, és csak ezekhez garantáljuk a hibamentes működést.

## 6.5 Verziók összefoglaló táblázat (2025. április)

|  |  |
| --- | --- |
| Szoftver/Platform | Javasolt/tesztelt verzió (2025. április) |
| Windows 11 | 23H2 (22631.3527), 24H2 (26100.3775, preview) |
| Chrome | 135.0.7049.96 |
| Firefox | 137.0.2 |
| Brave | 1.77.101 (Chromium 135.0.7049.115) |
| Node.js | 20.x LTS |
| Composer | 2.7.x |
| PHP | 8.1.x vagy újabb |
| Python | 3.12.x |
| Electron | 29.x |
| Visual Studio Code | 1.89.x |

# 7. Telepítési útmutató Fejlesztőknek

## 7.1 Rendszerkövetelmények

* **Tesztelt Futtatási környezet**: Microsoft Visual Studio Code
* **Szerveroldal**: PHP 8.0+, Composer, Node.js, XAMPP
* **Kliensoldal**:
  + Web: Modern böngészők (Chrome, Firefox).
  + Asztali: Windows 10+ / macOS 10.15+.
  + Mobil: Android 8.0+ / iOS 13+.

## 7.2 Telepítési lépések

1. Klónozd a repót:

git clone https://github.com/SitkuPeter/VizsgaRemek\_BB\_KE\_SP.git

1. Függőségek telepítése:

composer install   
npm install

1. Környezeti változók beállítása (.env fájl).

Amennyiben először indítjuk el a projektet létre kell hozni a gyökérkönyvtárba egy .env fájlt, ezt megtehetjük az .env.example fájl tartalmának kimásolásával és beillesztésével egy általunk létrehozott .env fájlba

Fontos! hogy ekkor már futnia kell a XAMPP control panelen az Apache és MySql szolgáltatásoknak, valamint egy lokális APP\_KEY-t is kell generálnunk, amit az alábbi bash paranccsal tehetünk meg:

php artisan key:generate

1. Adatbázis migrációk futtatása:

php artisan migrate --seed

1. Indítsd el a szervert:

php artisan serve   
npm run dev

### 7.3 Electron Desktop Alkalmazás telepítési útmutató

Az alábbi útmutató részletezi, hogyan kell telepíteni és futtatni a Casino projekt Electron alapú asztali alkalmazását. Mivel ez egy fejlesztői/tesztelői build, a megfelelő beállítások szükségesek a helyi környezetben való futtatáshoz.

Telepítési előfeltételek

Az Electron alkalmazás futtatásához a következőkre van szükség:

1. **Node.js és npm**: Töltsd le és telepítsd a legfrissebb verziót a [Node.js hivatalos oldaláról](https://nodejs.org/)
2. **Működő Laravel környezet**: Az 5.1-5.3 pontokban leírt telepítés szerint
3. **DeckOfCards API**: A 8.2.4 pontban leírt telepítés szerint

Electron Telepítése

A Casino projekt tartalmazza az Electron alkalmazást az electron-app mappában. A következő lépésekkel telepítheted:

# Lépj be az electron-app mappába

cd electron-app

# Telepítsd a függőségeket

npm install

Futtatási sorrend

Az asztali alkalmazás futtatásához a következő szolgáltatásokat kell elindítani a megadott sorrendben:

**1. Laravel szerver indítása**

# A projekt gyökérkönyvtárában

php artisan serve

Ez elindítja a Laravel szervert a http://127.0.0.1:8000 címen.

**2. DeckOfCards API indítása**

A 8.2.4 szekcióban leírtak szerint:

# Lépj be a deckofcards-master mappába

cd deckofcards-master

# Indítsd el az API szervert

python manage.py runserver 127.0.0.1:8001

**3. Electron alkalmazás indítása**

# Az electron-app mappában

cd electron-app

# Alkalmazás indítása

npm start

Az Electron alkalmazás működése

Az alkalmazás elindítja a főablakot és betölti a Laravel szervert ([http://127.0.0.1:8000](http://127.0.0.1:8000/)). A főablakban a következő funkciók érhetők el:

* **Menü nélküli felület**: A natív menüsáv el van rejtve, tiszta felhasználói élményt biztosítva
* **Játékok külön ablakban**: A /games/ útvonalon található játékok külön ablakban nyílnak meg
* **Optimalizált méretezés**: A főablak (1200x800) és a játékablakok (1000x600) mérete optimalizált a játékélményhez

**Fontos megjegyzések**

* Az Electron alkalmazás csak akkor működik megfelelően, ha mind a Laravel szerver, mind a DeckOfCards API fut a megadott portokon
* A Laravel szervernek a 8000-es porton kell futnia (php artisan serve alapértelmezetten ezt használja)
* A DeckOfCards API-nak a 8001-es porton kell futnia (a kódban ez van beállítva)
* Ha a portok foglaltak, módosítsd a megfelelő konfigurációs fájlokat és az Electron kódban a URL-eket

**Hibaelhárítás**

* **Fehér képernyő**: Ellenőrizd, hogy a Laravel szerver fut-e a 8000-es porton
* **Blackjack nem működik**: Ellenőrizd, hogy a DeckOfCards API fut-e a 8001-es porton
* **"Cannot find module" hiba**: Futtasd újra az npm install parancsot az electron-app mappában

### 7.3.1 Telepítési útvonalak

Az Electron alkalmazás telepítője a következő helyekre telepíti az alkalmazást alapértelmezés szerint:

* **Windows**:  
  C:\Users\<Felhasználónév>\AppData\Local\Programs<AppNév>  
  *Megjegyzés*: Ha admin jogokkal telepíti a felhasználó, a Program Files mappába is kerülhet.
* **macOS**:  
  /Applications/<AppNév>.app
* **Linux**:  
  /usr/lib/<appnév> vagy a felhasználó által választott könyvtár

### 7.3.2 Setup fájl generálása

A telepítőfájl létrehozásához használd az electron-builder-t:

cd electron-app

npm run build

A generált telepítők a dist/ mappában lesznek:

* Windows: .exe és .msi fájlok
* macOS: .dmg és .zip
* Linux: .deb, .rpm, stb.

### 7.3.3 Build futtatása

Parancsok a különböző platformokhoz:

# Windows

npm run build -- --win

# macOS

npm run build -- --mac

# Linux

npm run build -- --linux

A generált fájlok a dist mappában lesznek elérhetők, készen a terjesztésre.

### 7.3.4 Fontos megjegyzések

1. **Code Signing**:  
   Éles környezetben digitális aláírás szükséges a telepítőhöz (pl.: Windows esetén EV kódaláíró tanúsítvány).
2. **Környezetfüggőségek**:  
   A buildeléshez platform-specifikus eszközök szükségesek (pl.: Windows-on NSIS, macOS-en Xcode).
3. **Laravel szerver**:  
   Éles környezetben a Laravel backend-et külön webszerveren kell hosztolni, nem a php artisan serve-en.
4. **Asztali alkalmazás fájljai**:  
   Windows rendszeren: alapértelmezett hely C:\Users\<felhasználónév>\AppData\Local\Programs\<AppNév>, de a build könyvtárban (pl. C:\custombuilds\myelectronapp-) is megtalálható a setup tool és a win-unpacked mappa (hordozható futtatás, ebben van a futtatáshoz szükséges .exe fájl)[7.3.1]

# 8. Komponensek

## 8.1 Általános Komponensek, alapséma

### 8.1.1 Layoutok részletes leírása

A Casino projekt minden oldalához egységes, újrahasznosítható Blade layoutokat használ. Ezek a layoutok biztosítják a konzisztens megjelenést, a közös navigációt, valamint a reszponzív és sötét-világos mód támogatását. Az alábbiakban bemutatjuk a fő layoutokat és azok felépítését, funkcióit, valamint a navigation.blade.php komponens szerepét és működését, a dokumentáció általános sémáját követve23.

**Alap layoutok**

**app.blade.php**

* **Fő cél:** Az oldal globális, minden oldalra kiterjedő elrendezése, amely tartalmazza a navigációs sávot, a fő tartalmi blokkot és opcionálisan további komponenseket (pl. chat bubble).
* **Felépítés:**
  + HTML5 alapstruktúra, dinamikus lang attribútummal.
  + Meta tag-ek: karakterkódolás, viewport, CSRF token.
  + Dinamikus oldal cím: @yield('pageTitle', config('app.name', 'Spin It Clean')).
  + Külső assetek: FontAwesome ikonok, jQuery, Vite által generált CSS és JS fájlok.
  + **Navigáció:** @include('layouts.navigation') – a navigációs sáv minden oldalon megjelenik.
  + **Tartalom:** <main class="container mx-auto py-8"> @yield('content') </main> – minden oldal egyedi tartalma itt jelenik meg.
  + **Chat Bubble:** Bejelentkezett felhasználóknak egy fix pozíciójú chat ikon jelenik meg, amely a barátok üzeneteihez vezet.
  + **Sötét mód támogatás:** A dark: prefix és a megfelelő Tailwind CSS osztályok biztosítják a sötét-világos mód közötti váltást.
* **Főbb jellemzők:** Reszponzív, modern design, gyors asset betöltés Vite segítségével, biztonságos CSRF védelem, egységes navigáció2.

**game.blade.php**

* **Fő cél:** Kifejezetten a játékoldalakhoz készült layout, amely minimalizált, letisztult felületet biztosít a játékélményhez.
* **Felépítés:**
  + Hasonló szerkezetű, mint az app.blade.php, de nem tartalmaz navigációs sávot.
  + Kizárólag a játék fő tartalmát jeleníti meg a <main> blokkban.
  + Assetek: Vite által generált CSS és JS, FontAwesome, jQuery.
  + Sötét mód támogatás.
* **Főbb jellemzők:** Maximális fókusz a játékra, zavaró elemek nélkül, gyors betöltés, reszponzív kialakítás2.

**guest.blade.php**

* **Fő cél:** Vendég (nem bejelentkezett) felhasználók számára biztosít egyedi, látványos kezdőoldal layoutot.
* **Felépítés:**
  + Saját, vizuálisan kiemelkedő háttér (kaszinó témájú kép), átlátszó kártyák, glow effektusok.
  + Vendég navigáció: csak bejelentkezés és regisztráció gombok, mobilbarát menü.
  + Egyedi stílus: beágyazott CSS, egyedi betűtípusok, animációk.
  + Tartalom: fő blokkok (hero section, feature section, games section, CTA, footer).
* **Főbb jellemzők:** Modern, figyelemfelkeltő design, mobilbarát navigáció, vendégek számára optimalizált funkciók2.

**bettingarea.blade.php**

* **Fő cél:** Fogadási összeg kiválasztására szolgáló, minden játékoldalon újrahasznosítható komponens.
* **Felépítés:**
  + Számbeviteli mező a tét megadásához.
  + Aktuális tét kijelző.
  + Reszponzív, központosan elhelyezett elrendezés.
* **Főbb jellemzők:** Egységes fogadási logika minden játékban, könnyű integráció más layoutokba2.

**Navigációs komponens: navigation.blade.php**

A navigation.blade.php a fő navigációs sávot valósítja meg, amely minden bejelentkezett és vendég felhasználó számára egységes, de dinamikusan igazodik a jogosultságokhoz és az aktuális oldalhoz:

* **Reszponzív menü:** Mobilon hamburger menü, asztali nézetben teljes menüsor.
* **Menüpontok:**
  + Bejelentkezett felhasználóknak: főoldal, játékok, barátok, ranglista, fórum, shop, admin (ha jogosult).
  + Vendégeknek: bejelentkezés, regisztráció.
* **Felhasználói funkciók:**
  + Egyenleg kijelző, amely linkként szolgál a wallet oldalra.
  + Barátkérelmek értesítő (ikon, badge, lenyíló lista).
  + Profilmenü: profil szerkesztés, statisztikák, kijelentkezés.
* **Sötét mód támogatás:** Minden elemhez tartozik világos és sötét színvilág.
* **Alpine.js használata:** A mobil menü és a dropdownok vezérléséhez3.
* **Példa menüpont szerkezet:**

<a href="{{ route('pages.mainpage') }}" class="hover:text-yellow-500 ...">Main Page</a>

<a href="{{ route('pages.games') }}" class="hover:text-yellow-500 ...">Games</a>

...

@if (auth()->user()->is\_admin)

<a href="{{ route('admin.dashboard') }}">Admin Control</a>

@endif

**Layoutok összehasonlítása**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layout | Fő cél | Navigáció | Sötét mód | Tartalom helye | Speciális funkciók |
| app.blade.php | Általános oldal | Igen | Igen | @yield('content') | Chat bubble, friend chat |
| game.blade.php | Játékoldalak | Nem | Igen | @yield('content') | Minimál, játékfókusz |
| guest.blade.php | Vendég kezdőoldal | Igen | Igen | {{ $slot }} | Casino háttér, animációk |
| bettingarea.blade.php | Fogadási komponens | N/A | Igen | Tét input, kijelző | Újrahasznosítható |

**Login rendszer**

A bejelentkezési és regisztrációs rendszer a Laravel Breeze csomag alapértelmezett beállításaira épül, amely biztonságos, session-alapú authentikációt, regisztrációt, jelszó-visszaállítást és email validációt biztosít. A Breeze által generált sablonok és route-ok lettek integrálva a projektbe, így minden auth-funkció a Laravel ajánlott, biztonságos módján működik2.

### 8.1.2 Vite hivatkozások magyarázata

A Vite egy modern eszköz, amely gyorsítja az assetek (CSS, JS) betöltését. Az alábbi fájlokat tölti be:

**CSS fájlok:**

* resources/css/app.css: Globális stílusok
* resources/css/(komponens).css: Specifikus stílusok az adott komponensekhez.

**JavaScript fájlok:**

* resources/js/app.js: Globális interaktív funkciók.
* resources/js/(komponens).js: Specifikus logika az adott komponensek működéséhez.

### 8.1.3 Adatbázis migrációk:

* Felhasználók táblája (users) tartalmazza az egyenleget és statisztikákat.
* Játéklogok táblája (game\_logs) rögzíti az összes játszott játékot.

### 8.1.4 Útvonalak (web.php)

A web.php fájl a Laravel alkalmazás webes útvonalait definiálja. Felelős azért, hogy a felhasználói kéréseket a megfelelő vezérlőkhöz (Controller) irányítsa. Főbb elemei:

**Kulcsfontosságú részek:**

1. **Oldalhoz tartozó útvonalak:**

* Főoldal (/), irányítva a welcome nézetre.
* Játékok útvonalai (pl.: /games/blackjack), amelyek a GameController metódusait hívják meg.

Route::get('/games/blackjack', [GameController::class, 'blackjack'])->name('blackjack');

1. **Hitelesítéshez kötött útvonalak:**

* Profilkezelés (/profile), barátok listája (/friends), főmenü (/mainpage).

Route::middleware('auth')->group(function () {

Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');

});

1. **Admin útvonalak:**

* Admin dashboard (/admin), felhasználói adatok megjelenítése (/admin/{id}).

Route::get('/admin', [AdminController::class, 'dashboard'])

->middleware(['auth', AdminMiddleware::class])

->name('admin.dashboard');

1. **Névvel ellátott útvonalak:**

* Például ->name('blackjack') lehetővé teszi útvonalak dinamikus hivatkozását a név alapján.

### 8.1.5 API útvonalak (api.php):

Az api.php fájl a backend API végpontokat definiálja, amelyeken keresztül a frontend kommunikál a szerverrel. A végpontok JSON válaszokat adnak vissza. Főbb elemei:

**Kulcsfontosságú részek:**

1. **Sanctum hitelesítés:**

* Csak hitelesített felhasználók érhetik el a csoportba tartozó végpontokat.

Route::middleware('auth:sanctum')->group(function () {

Route::get('/user', function (Request $request) { return $request->user(); });

});

1. **Játékhoz kapcsolódó végpontok:**

* Minden játékhoz tartozik start és end művelet (pl.: /blackjack/start).

Route::post('/blackjack/start', [BlackjackController::class, 'start']);

Route::post('/blackjack/end', [BlackjackController::class, 'end']);

1. **Egyéb végpontok:**

* Egyenleg lekérdezése: /user/balance.
* Kártyahúzás proxy végpontja: /proxy-draw-cards (külső API hívás kezelése).
* Token generálás: /generate-token (fejlesztési célokra).

### 8.1.6 Különbségek a web.php és api.php között

| **Tulajdonság** | **web.php** | **api.php** |
| --- | --- | --- |
| **Cél** | Oldalak megjelenítése | Adatcserére optimalizált API végpontok |
| **Válasz formátuma** | HTML nézetek | JSON |
| **Hitelesítés** | Session-based | Sanctum token |
| **Middleware** | auth, verified | auth:sanctum |
| **Példa útvonal** | /games/blackjack | /blackjack/start |

**Fontos megjegyzés**

* A **web.php** útvonalak a felhasználói felületet szolgálják ki, míg az **api.php** a játéklogika és adatkezelés mögötti műveleteket vezérli.
* Az API végpontokhoz a frontend **Axios** vagy **Fetch API** segítségével kapcsolódik (lásd: BJScript.js).

## 8.2 Blackjack játék dokumentációja

### 8.2.0 BlackJack játék layout (blackjack.blade.php)

Ez az oldal specifikus a Blackjack játékhoz. Tartalmazza:

* **Balance kijelző:** Megjeleníti a felhasználó aktuális egyenlegét.
* **Játék konténer:** Dealer és Player szekciók, valamint akciógombok (Hit, Stand, New Game).
* **Fogadási terület:** Az @include('layouts.betingarea') segítségével integrálva.
* **Stílus és script betöltése:** A Vite segítségével (BJStyle.css, BJScript.js).

### 8.2.1 Áttekintés

A Blackjack játék a Laravel keretrendszerben készült, és a Deck of Cards API-t használja kártyák kezelésére. Az API kódját egy helyi szerveren futtatjuk, amely az open-source GitHub projektből tölthető le. A játék működéséhez a backend és frontend kommunikációja biztosított, azonban a kód nem tartalmaz explicit backend kommunikációs metódusokat, csak az API-t használja.

### 8.2.2 Backend működés

**Controller (BlackjackController):**

* A játék indítása (start()): Ellenőrzi, hogy van-e elegendő egyenleg, majd levonja a tétet.
* A játék befejezése (end()): Kiértékeli az eredményt (győzelem, döntetlen vagy veszteség), frissíti az egyenleget és statisztikákat ment.
* Privát metódus (checkBlackjack()): Ellenőrzi, hogy van-e blackjack kéz.

**API útvonalak (api.php):**

* /blackjack/start: A játék indításához.
* /blackjack/end: A játék befejezéséhez.

### 8.2.3 Frontend működés

**JavaScript logika BJscript.js:**

* Kártyák kezelése (húzás, pontszám számítás).
* Játékállapot frissítése (győzelem/veszteség/döntetlen).
* API hívások (Deck of Cards API és backend endpointok).

**UI elemek:**

* Dealer és Player kártyák megjelenítése dinamikusan.
* Akciógombok állapotának kezelése (aktív/inaktív).
* Fogadási összeg megadása és kijelzése.

### 8.2.4 Telepítés

**Telepítési útmutató:**

**Laravel projekt telepítése:**

Lásd:

A Projekt már tartalmazza a programot, amik szükségesek az opensource DeckOfCardsApi lokális futtatásához, viszont a függőségeket magunknak kell beimportálnunk és telepítenünk

Először is telepítenünk kell A Python-t az eszközünkre (a tesztelés 3.12.x-es verziószámú Python-nal történt, a biztos működés érdekében 3.13 előttire lesz szükség): <https://www.python.org/downloads/>

Amennyiben a (Casino projekt) telepítve van és sikeresen elindítottuk, lennie kell a projekten belül egy *deckofcards-master* mappának, ez tartalmazza a kártyákért felelős alkalmazást

**Az alkalmazás telepítése:**

Először be kell lépni terminal-on belül az alkalmazást tartalmazó mappába:

cd deckofcards-master

Ezt követően futtassuk a következő parancsokat, ezek feltelepítik a szükséges függőségeket, konfigurálják a működést:

pip install -r requirements.txt

python manage.py migrate

Ez után futtassuk a használathoz a következő parancsot:

python manage.py runserver 127.0.0.1:8001

A runserver utáni cím átírásával kedvünkre változtathatjuk a szerver eléréséhez használt címet, ha a telepítés sikeres volt akkor minden indításkor csak ezt a kettő parancsot kell futtatnunk:

cd ’deckofcards-master’

python manage.py runserver 127.0.0.1:8001

Győződjünk meg arról, hogy az API szerver fut (pl.: http://127.0.0.1:8001/api/deck/new/shuffle/).  
Amennyiben a szerver fut a BlackJack játék indításra készen áll

## 8.3 Coinflip játék dokumentációja

### 8.3.0 Coinflip játék layout (coinflip.blade.php)

Ez az oldal specifikus a "Fej vagy Írás" játékhoz. Tartalmazza:

* **Balance kijelző**: Megjeleníti a felhasználó aktuális egyenlegét.
* **Játékinterfész**:
  + Választó gombok (Fej/Írás).
  + Animált érme megjelenítése.
  + Eredménykijelző és újrajátszás gomb.
* **Fogadási terület**: Az @include('layouts.betingarea') segítségével integrálva.
* **Stílus és script betöltése**: A Vite segítségével (coinflip.css, coinflip.js).

### 8.3.1 Áttekintés

A Coinflip játék egy egyszerű esélyjáték, ahol a felhasználó kiválasztja, hogy az érme melyik oldalára fogad. A backend a Laravel keretrendszert, a frontend pedig JavaScript animációkat használ. A kimenetel 50-50% eséllyel generálódik.

### 8.3.2 Backend működés

**Controller (CoinflipController)**

* **Játék indítása (start())**:
* Levonja a tétet a felhasználó egyenlegéből.

$user->balance -= $betAmount;

$user->save();

* **Játék befejezése (end())**:
* Kiszámolja a nyereményt (duplázás győzelem esetén).
* Frissíti a felhasználó statisztikáit és menti az adatbázisba.

if ($result === 'win') {

$winAmount = $betAmount \* 2;

$user->balance += $winAmount;

}

**API útvonalak (api.php)**

* /coinflip/start: Fogadás elfogadásához.
* /coinflip/end: Eredmény rögzítéséhez.

**Adatbázis integráció**

A game\_logs tábla rögzíti:

* Felhasználó választását (user\_choice).
* Érme eredményét (coin\_result).

### 8.3.3 Frontend működés

**JavaScript logika (coinflip.js)**

1. **Eseménykezelők**:

* Gombnyomásra indítja a játékot (#heads-btn, #tails-btn).

1. **Animációk**:

* 3D-s érme forgatás CSS transform és animation segítségével.

coin.style.animation = "flip 2s ease-in-out";

1. **API kommunikáció**:

* Küldi a fogadási összeget a /coinflip/start végpontra.
* Leküldi az eredményt a /coinflip/end végpontra.

**UI elemek**

* **Érme animáció**: HTML/CSS 3D transzformációk.
* **Eredménykijelző**: Szöveges visszajelzés a nyereményről/veszteségről.

### 8.3.4 Telepítés

A Coinflip játék működéséhez nincs szükség külső API-ra. Elég a Laravel szervert és az adatbázist futtatni:

php artisan serve

npm run dev

## 8.4 Crash játék dokumentációja

### 8.4.0 Crash játék layout (crash.blade.php)

Ez az oldal specifikus a Crash játékhoz. Tartalmazza:

* **Balance kijelző**: Megjeleníti a felhasználó aktuális egyenlegét.
* **Játékinterfész:**
  + Indítás és kifizetés gombok (Start Game, Cash Out).
  + Multiplikátor kijelző és progress bar.
  + Játékelőzmények listája.
* **Fogadási terület:** Az @include('layouts.betingarea') segítségével integrálva.
* **Stílus és script betöltése:** A Vite segítségével (crash.css, crash.js).

### 8.4.1 Áttekintés

A Crash játékban a felhasználó egy növekvő multiplikátorral fogad, amely bármikor "összeomolhat". A játék célja, hogy a felhasználó a megfelelő pillanatban kifizettesse a tétet a crash előtt. A backend a Laravel keretrendszert, a frontend dinamikus animációkat és valós idejű frissítéseket használ.

### 8.4.2 Backend működés

**Controller (CrashController)**

**Játék indítása (start()):**

* Ellenőrzi, hogy a felhasználónak nincs-e aktív játéka (Cache segítségével).
* Levonja a tétet a felhasználó egyenlegéből.

Cache::put('crash-active-'.$user->id, true, now()->addMinutes(2));

$user->balance -= $betAmount;

$user->save();

**Játék befejezése (end()):**

* Kiszámolja a nyereményt a kifizetési multiplikátor alapján.
* Cache segítségével megakadályozza a dupla tranzakciókat.

if ($validated['result'] === 'win') {

$user->balance += $validated['winnings'];

}

**API útvonalak (api.php)**

* /crash/start: Játék indításához.
* /crash/end: Eredmény rögzítéséhez.

**Adatbázis integráció**

A game\_logs tábla rögzíti:

* Crash pontot (crash\_point).
* Kifizetési multiplikátort (cashout\_multiplier).

### 8.4.3 Frontend működés

**JavaScript logika (crash.js)**

* Multiplikátor növelése:
* A multiplikátor másodpercenként 0.01x-el nő.

currentMultiplier += 0.01;

multiplierDisplay.textContent = `${currentMultiplier.toFixed(2)}x`;

**Crash generálása:**

A crash pontot valószínűségi algoritmus generálja (1/x eloszlás).

function getRandomCrashPoint() {

return Math.max(1.00, (1 / (1 - Math.random())));

}

**API kommunikáció:**

* Küldi a fogadási összeget a /crash/start végpontra.
* Leküldi az eredményt a /crash/end végpontra.

**UI elemek**

* Progress bar: A multiplikátor növekedését mutatja.
* Játékelőzmények: Legutóbbi crash pontok listája.

### 8.4.4 Telepítés

A Crash játék működéséhez nincs szükség külső API-ra. Elég a Laravel szervert és az adatbázist futtatni:

php artisan serve

npm run dev

## 8.5 Rock Paper Scissors játék dokumentációja

### 8.5.0 Rock Paper Scissors játék layout (rock.blade.php)

Ez az oldal specifikus a "Kő-Papír-Olló" játékhoz. Tartalmazza:

* **Balance kijelző**: Megjeleníti a felhasználó aktuális egyenlegét.
* **Játékinterfész**:
  + Választó gombok (Kő, Papír, Olló).
  + Animált számítógépes választás.
  + Eredmény- és pontszámkijelző.
* **Fogadási terület**: Az @include('layouts.betingarea') segítségével integrálva.
* **Stílus és script betöltése**: A Vite segítségével (rockpaperscissors.css, rockpaperscissors.js).

### 8.5.1 Áttekintés

A "Kő-Papír-Olló" játékban a felhasználó egy tétet helyez el, majd választ egy eszközt (kő, papír, olló). A számítógép **nem választhatja a felhasználó által választott eszközt**, így növelve a stratégiai elemet. A győztes a tét duplázását kapja, döntetlen esetén a tét visszakerül.

### 8.5.2 Backend működés

**Controller (RockPaperScissorsController)**

**Játék indítása (start())**:

* + Levonja a tétet a felhasználó egyenlegéből.

$user->balance -= $betAmount;

$user->save();

**Játék befejezése (end())**:

* + Kiszámolja a nyereményt (győzelem esetén duplázás, döntetlen esetén tét visszatérítés).

if ($result === 'win') {

$user->balance += $betAmount \* 2;

} elseif ($result === 'tie') {

$user->balance += $betAmount;

}

**API útvonalak (api.php)**

* /rock-paper-scissors/start: Fogadás elfogadásához.
* /rock-paper-scissors/end: Eredmény rögzítéséhez.

**Adatbázis integráció**

A game\_logs tábla rögzíti:

* Felhasználó és számítógép választását (player\_choice, computer\_choice).
* Eredményt (result).

### 8.5.3 Frontend működés

**JavaScript logika (rockpaperscissors.js)**

**Játéklogika**:

* A számítógép választása **kizárja a felhasználó által választott eszközt**, így nehezítve a játékot.

function getComputerChoice(playerChoice) {

const options = ["rock", "paper", "scissors"];

const filteredOptions = options.filter(choice => choice !== playerChoice);

return filteredOptions[Math.floor(Math.random() \* filteredOptions.length)];

}

**Animációk**:

* Számítógépes választás animációja (gyors váltás a lehetőségek között).

animateComputerChoice(finalChoice, () => { /\* ... \*/ });

**API kommunikáció**:

* Küldi a fogadási összeget a /rock-paper-scissors/start végpontra.
* Leküldi az eredményt a /rock-paper-scissors/end végpontra.

**UI elemek**

* **Pontszámkijelző**: Nyert körök száma.
* **Eredményüzenet**: Szöveges és emojis visszajelzés.

### 8.5.4 Telepítés

A játék működéséhez **nincs szükség külső API-ra**. Elég a Laravel szervert és az adatbázist futtatni:

php artisan serve

npm run dev

## 8.6 Rulett játék dokumentációja

### 8.6.0 Rulett játék layout (roulette.blade.php)

Ez az oldal specifikus a Rulett játékhoz. Tartalmazza:

* **Balance kijelző**: Megjeleníti a felhasználó aktuális egyenlegét.
* **Játékinterfész**:
  + Rulett kerék animációval.
  + Fogadási típus választó (even, odd, red, black, green).
  + Spin gomb és előző nyeremények listája.
* **Fogadási terület**: Az @include('layouts.betingarea') segítségével integrálva.
* **Stílus és script betöltése**: A Vite segítségével (roulette.css, roulette.js).

### 8.6.1 Áttekintés

A Rulett játékban a felhasználó a következőkre fogadhat:

* **Szín** (piros, fekete, zöld).
* **Páros/páratlan**.

A nyeremény a fogadási típus alapján változik (zöld esetén 35x, egyébként 2x). A backend a Laravel keretrendszert, a frontend pedig dinamikus animációkat használ.

### 8.6.2 Backend működés

**Controller (RouletteController)**

**Játék indítása (start())**:

* Ellenőrzi a tétet és levonja a felhasználó egyenlegéből.

$user->balance -= $betAmount;

$user->save();

**Játék befejezése (end())**:

* Kiszámolja a nyereményt a fogadási típus alapján.

if ($result === 'win') {

$user->balance += $winnings;

}

* A getPayoutMultiplier() metódus határozza meg a szorzót (pl.: zöld esetén 35x).

**API útvonalak (api.php)**

* /roulette/start: Fogadás elfogadásához.
* /roulette/end: Eredmény rögzítéséhez.

**Adatbázis integráció**

A game\_logs tábla rögzíti:

* Fogadási típust (bet\_type).
* Nyeremény szorzót (payout\_multiplier).

### 8.6.3 Frontend működés

**JavaScript logika (roulette.js)**

**Rulett kerék animáció**:

* A kerék forgatása véletlenszerű szám kiválasztásához.

let delay = 50;

setTimeout(spinStep, delay); // Gyorsuló animáció

**Eredmény meghatározása**:

* A nyerő szám színét és paritását ellenőrzi.

const isEven = winner.number % 2 === 0 ? "Even" : "Odd";

**API kommunikáció**:

* Küldi a fogadási összeget a /roulette/start végpontra.
* Leküldi az eredményt a /roulette/end végpontra.

**UI elemek**

* **Előző nyeremények**: Legutóbbi 10 nyerő szám megjelenítése.
* **Színkódolt eredmény**: A nyerő szám színe szerinti kiemelés.

### 8.6.4 Telepítés

A játék működéséhez **nincs szükség külső API-ra**. Elég a Laravel szervert és az adatbázist futtatni:

php artisan serve

npm run dev

## 8.7 Slot Machine játék dokumentációja

### 8.7.0 Slot Machine játék layout (slot.blade.php)

Ez az oldal specifikus a Slot Machine (játékgép) játékhoz. Tartalmazza:

**Balance kijelző:** Megjeleníti a felhasználó aktuális egyenlegét.

**Játékinterfész:**

* + 5 db pörgethető henger (reel) szimbólumokkal.
  + "Spin Reels" (pörgetés) és "Auto Spin" gombok.
  + Nyereménytörténet lista és aktuális játék eredmény kijelző.
  + Multiplikátor kijelző és progress bar.
  + Info gomb, amely modális ablakban mutatja a szabályokat és a kifizetési táblázatot.

**Fogadási terület:** Az @include('layouts.betingarea') segítségével integrálva.

**Stílus és script betöltése:** A Vite segítségével (slot.css, slot.js).

### 8.7.1 Áttekintés

A Slot Machine játék egy klasszikus, többhengeres játékgépet szimulál, ahol a játékosok különböző szimbólumokat pörgethetnek ki. A cél, hogy legalább három azonos szimbólum jelenjen meg a hengereken, a kifizetés a szimbólumok típusától és számától függ. A frontend látványos animációkkal és élő egyenlegfrissítéssel támogatja a játékélményt.

### 8.7.2 Backend működés

**Controller (SlotController)**

**Játék indítása (start()):**

* + Ellenőrzi, hogy a felhasználó egyenlege elegendő-e a tét megtételéhez.
  + Levonja a tét összegét az egyenlegből, majd menti az új egyenleget.
  + Naplózza a tranzakciót.
  + Visszaadja az új egyenleget JSON válaszban.

**Játék befejezése (end()):**

* + Fogadja a játék eredményét, a pörgetett szimbólumokat és a nyeremény összegét.
  + Ha a játékos nyert, jóváírja a nyereményt az egyenlegén.
  + Naplózza az eredményt és frissíti a felhasználó statisztikáit.
  + Létrehoz egy új bejegyzést a game\_logs táblában, amely tartalmazza a játék adatait (hengerek, eredmény, tét, nyeremény).
  + Visszaadja az új egyenleget JSON válaszban.

**API útvonalak (api.php):**

* /slot-machine/start: A játék indításához (tét levonása).
* /slot-machine/end: A játék befejezéséhez (eredmény és nyeremény rögzítése).
* /user/balance: Az aktuális egyenleg lekérdezése.

### 8.7.3 Frontend működés

**JavaScript logika (slot.js):**

* **Szimbólumok és kifizetési tábla:**
  + 5-féle szimbólum: 🍒, 🍋, 🍉, ⭐, 💎
  + Kifizetések: legalább 3 azonos szimbólum esetén, a kifizetés a szimbólum típusától és a találatok számától függ (részletesen a payoutTable objektumban).
* **DOM elemek kezelése:**
  + Hengerek, gombok, egyenlegkijelző, eredménykijelző, nyereménytörténet, modális ablak.
* **Játék indítása:**
  + A "Spin Reels" gomb eseménykezelője érvényesíti a tétet, majd POST kérést küld a /api/slot-machine/start végpontra.
  + Animált pörgetés: a hengerek szimbólumai gyorsan változnak, majd megállnak a végleges szimbólumokon.
* **Eredmény kiértékelése:**
  + A kód kiszámolja, hány azonos szimbólum jelent meg, és ezek alapján meghatározza a nyereményt.
  + POST kérést küld a /api/slot-machine/end végpontra az eredménnyel, tét összeggel, hengerek állásával és nyereménnyel.
  + A válaszban kapott egyenleget frissíti a felületen.
  + A nyereménytörténet listában megjeleníti a legutóbbi nyerő kombinációkat.
* **Auto Spin:**
  + Az "Auto Spin" gombbal a játék automatikusan többször is pörget, amíg van elegendő egyenleg.
* **Modal kezelés:**
  + Az info gomb és a modális ablak eseménykezelői biztosítják a szabályok és kifizetések gyors elérhetőségét.
* **Stílus:**
  + A modal animációk, gombok és egyéb vizuális elemek CSS-sel lettek kialakítva a játékélmény fokozásához.

### 8.7.4 Telepítés

A Slot Machine játék működéséhez elegendő a Laravel szervert és az adatbázist futtatni:

php artisan serve

npm run dev

A szükséges CSS és JavaScript fájlokat a Vite tölti be (slot.css, slot.js).

## 8.8 Mines játék dokumentációja

### 8.8.0 Mines játék layout (mines.blade.php)

Ez az oldal a Mines (aknakereső) játékhoz készült, főbb elemei:

* **Balance kijelző:** Megjeleníti a felhasználó aktuális egyenlegét.
* **Játékinterfész:**
  + 5x5-ös rács (25 mező), amelyen 7 akna van elrejtve.
  + "Start Game" gomb a játék indításához, "Cash Out" gomb a kiszálláshoz.
  + Multiplikátor kijelző és progress bar, amely mutatja a várható nyereményszorzót.
  + Eredménykijelző, játéktörténet lista.
  + Info gomb, amely modális ablakban mutatja a szabályokat.
* **Fogadási terület:** Az @include('layouts.betingarea') komponenssel integrálva.
* **Stílus és script betöltése:** Vite segítségével (mines.css, mines.js).

### 8.8.1 Áttekintés

A Mines játék egy kaszinó-aknakereső, ahol a játékos tétet helyez el, majd a rácson mezőket felfedve próbál minél több drágakövet találni anélkül, hogy aknára lépne. Minden sikeres felfedés növeli a szorzót. A játékos bármikor kiszállhat ("Cash Out"), ekkor a tétje és a szorzó szorzata lesz a nyereménye. Ha aknára lép, elveszíti a tétjét. A játék végén az összes akna helye felfedésre kerül.

### 8.8.2 Backend működés

**Controller (MinesController):**

* **Játék indítása (start()):**
  + Ellenőrzi a tét érvényességét és hogy a felhasználó egyenlege elegendő-e.
  + Levonja a tétet a felhasználó egyenlegéből, majd menti az új egyenleget.
  + Visszaadja az új egyenleget JSON válaszban.
* **Játék befejezése (end()):**
  + Fogadja az eredményt (win vagy lose), a tét összegét, az aknák számát, a felfedett mezők számát és a nyereményt.
  + Nyeremény esetén jóváírja azt a felhasználó egyenlegén, veszteség esetén nem történik jóváírás.
  + Frissíti a felhasználó statisztikáit.
  + Létrehoz egy új bejegyzést a game\_logs táblában, amely tartalmazza a játék adatait (aknák száma, felfedett mezők, szorzó).
  + Visszaadja az új egyenleget JSON válaszban.

**API útvonalak (api.php):**

* /mines/start: A játék indításához (tét levonása).
* /mines/end: A játék befejezéséhez (eredmény és nyeremény rögzítése).

### 8.8.3 Frontend működés

**JavaScript logika (mines.js):**

* **Inicializálás:**
  + DOM elemek kiválasztása, eseményfigyelők beállítása (start, cash out, info modal, rács kattintás).
  + A rácsot a createGrid() függvény hozza létre.
* **Játék indítása:**
  + A "Start Game" gomb POST kérést küld a /api/mines/start végpontra a tét összegével.
  + Sikeres indítás után a mezők állapota alaphelyzetbe áll, a tét és a szorzó nullázódik, a játék aktív lesz.
  + A rács 7 aknával és 18 drágakővel töltődik fel véletlenszerűen.
* **Játék menete:**
  + A játékos mezőkre kattintva felfedheti azok tartalmát.
  + Drágakő esetén nő a szorzó, akna esetén a játék véget ér.
  + A szorzó növekedése a felfedett mezők számától és az aknák számától függ; a progress bar vizuálisan is mutatja a szorzó alakulását.
* **Cash Out:**
  + A "Cash Out" gombbal a játékos bármikor kiszállhat, ekkor a tét és a szorzó szorzata lesz a nyeremény.
  + Az összes akna helye felfedésre kerül.
  + POST kérés megy a /api/mines/end végpontra a nyereménnyel, aknák számával, felfedett mezőkkel.
* **Játék vége:**
  + Akna felfedésekor vagy cash out után a játék véget ér, az eredmény és a nyeremény megjelenik.
  + A játéktörténet frissül a legutóbbi játék eredményével.
* **Stílus:**
  + A mezők, aknák, drágakövek, cash out és veszteség animációk, valamint a progress bar CSS-sel lettek kialakítva.

### 8.8.4 Telepítés

A Mines játék működéséhez elég a Laravel szervert és az adatbázist futtatni:

php artisan serve

npm run dev

A szükséges CSS és JavaScript fájlokat a Vite tölti be (mines.css, mines.js).

## 8.9 Nem játék komponensek

### 8.9.1 Profil statisztika oldal (pages/profile/statistics) – Komponensleírás

A profil statisztika oldal a felhasználó kaszinós eredményeit és játékstatisztikáit jeleníti meg, modern, reszponzív kártyás elrendezésben, sötét/világos mód támogatással. Az oldal a layouts.app layoutot használja, és a fő tartalmi blokkban jeleníti meg a statisztikai adatokat.

**Főbb felépítés:**

* **Statisztika kártyák**:  
  Grid-elrendezésben jelennek meg a legfontosabb mutatók (egyenleg, összes játék, győzelmek, vereségek, összes nyeremény, veszteség, nettó profit, utolsó játék).
* **Játékstatisztika diagramok**:  
  Minden játékhoz külön kártya tartalmaz egy kördiagramot (Chart.js), amely a győzelmek és vereségek arányát mutatja.
* **Hiányzó statisztika figyelmeztetés**:  
  Ha nincs játékstatisztika, figyelmeztető kártya jelenik meg.
* **Reszponzív, modern UI**:  
  Tailwind CSS grid, hover effektek, sötét mód támogatás.

### 8.9.2 Reklámfelület és hirdetéskezelő modul – Komponensleírás

#### 8.9.2.1 Wallet (Hirdetésnéző és jutalom) felület

A wallet.blade.php a felhasználók számára elérhető pénztárca és reklámnéző oldal. Itt a játékosok aktív hirdetéseket tekinthetnek meg, amelyekért virtuális jutalmat kapnak. A komponens főbb elemei:

* **Egyenlegkártya**:  
  Látványos, gradient hátterű kártya mutatja az aktuális egyenleget.
* **Elérhető hirdetések**:  
  Grid-elrendezésben minden hirdetés kártyán jelenik meg (cím, leírás, időtartam, jutalom, "Watch Ad" gomb).
* **Hirdetéslejátszó modal**:  
  Külön modal ablakban jelenik meg a reklám (videó vagy kép), progress bar-ral, szünet/feladás lehetőséggel, forfeit megerősítő ablak.
* **Interaktív vezérlés**:  
  A néző progress bar-jának végén automatikus jutalom, részleges nézés esetén arányos jutalom, idő előtti bezárásnál feladás opció.
* **Stílus**:  
  Modern, sötét/világos mód támogatás, wallet.css-ben egyedi modal, kártya, progress bar stílusok.

**Blade példa:**

<div class="balance-card ...">

<h2>Current Balance</h2>

<p id="balance-amount">${{ number\_format($balance, 2) }}</p>

</div>

<div class="ad-card ...">

<h3>{{ $ad->title }}</h3>

<p>{{ $ad->description }}</p>

<span>Duration: {{ $ad->duration\_seconds }}s</span>

<span>Reward: ${{ number\_format($ad->reward\_amount, 2) }}</span>

<button class="ad-btn" data-ad-id="{{ $ad->id }}" ...>Watch Ad</button>

</div>

**Főbb JS logika (wallet.js):**

* Hirdetés gombra kattintva modal nyílik, progress bar indul, videó/kép betöltődik.
* Szüneteltetés, folytatás, idő előtti bezárás (forfeit) támogatott.
* Progress végén AJAX POST /wallet/process-ad-ra, reward és új egyenleg frissítése.
* Hiba vagy megszakítás esetén figyelmeztetés.

**CSS (wallet.css):**

* Modal, progress bar, kártyák, forfeit ablak egyedi stílusai.
* Hover effektek, árnyék, átmenetek.

#### 8.9.2.2 Admin hirdetéskezelő felület

Az admin hirdetéskezelő a pages/admin/ads nézetekből épül fel, és lehetővé teszi a reklámkampányok teljes körű kezelését.

* **Listázás (index)**:  
  Táblázatos nézet minden hirdetésről (cím, típus, időtartam, jutalom, készítő/frissítő, akciók: szerkesztés, törlés).
* **Létrehozás/szerkesztés (form)**:  
  Űrlap hirdetés adatainak (cím, leírás, időtartam, jutalom, média típus, thumbnail, fő média) rögzítéséhez/módosításához.
* **Média kezelés**:  
  Kép és videó feltöltés, meglévő média előnézete.
* **Akciók**:  
  Szerkesztés, soft delete, végleges törlés (force delete), visszaállítás.
* **Stílus**:  
  Sötét/világos mód, reszponzív, modern admin UI.

**Blade példa (index):**

<table>

<thead>

<tr>

<th>Title</th>

<th>Type</th>

<th>Duration</th>

<th>Reward</th>

<th>Created By</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

@foreach ($ads as $ad)

<tr>

<td>{{ $ad->title }}</td>

<td>{{ $ad->media\_type }}</td>

<td>{{ $ad->duration\_seconds }}s</td>

<td>${{ number\_format($ad->reward\_amount, 2) }}</td>

<td>{{ $ad->creator?->name ?? 'N/A' }}</td>

<td>

<a href="{{ route('admin.ads.edit', $ad) }}">Edit</a>

<form action="{{ route('admin.ads.destroy', $ad) }}" method="POST">...</form>

</td>

</tr>

@endforeach

</tbody>

</table>

### 8.9.3 Leaderboard (Ranglista) komponens

A Leaderboard oldal a legjobb játékosokat rangsorolja az egyenlegük alapján. Célja, hogy motiválja a felhasználókat, és közösségi élményt nyújtson. A ranglista mindenki számára elérhető, a felület reszponzív, világos és sötét módban is jól használható.

**Főbb elemek:**

* **Fejléc:** Oldalcím és rövid leírás.
* **Lista:**
  + Minden sorban: helyezés, játékos neve (avatar), egyenleg, utolsó aktivitás.
  + Az első három helyezett vizuálisan kiemelve (pl. arany, ezüst, bronz háttér).
  + Adminok külön megjelölve.
* **Lapozás:** Ha sok a játékos, a lista több oldalra van bontva.

**Főbb működés:**

* Az adatokat a backendről kapja, ahol a felhasználók egyenleg szerint csökkenő sorrendben vannak lekérve.
* A lapozás automatikus, Laravel paginátorral.
* A vizuális kiemelés segíti a legjobb játékosok azonosítását.

### 8.9.4 Barátok (Friends) és Üzenetek – Komponensleírás

A barátok funkciók lehetővé teszik, hogy a felhasználók egymást megtalálják, barátnak jelöljék, barátkérelmeket kezeljenek, valamint privát üzeneteket váltsanak. A felület modern, reszponzív, sötét/világos módot is támogat.

**Add Friends oldal**

* **Kereső és rendező sáv:**
  + Név szerinti keresés élő szűréssel.
  + Rendezés név vagy egyenleg szerint (növekvő/csökkenő).
* **Felhasználó kártyák:**
  + Profilkép vagy kezdőbetű, név, egyenleg.
  + "Add Friend" gomb, amely POST kérést küld a barátkéréshez.
* **Interaktív lista:**
  + Keresés és rendezés JavaScript-tel történik, minden változás azonnal frissíti a listát.

**Barátlista és kérelmek (My Friends)**

* **Két tab:**
  + *Friends*: Elfogadott barátok listája kártyákkal (profil, név, email, egyenleg, "Show Info" gomb).
  + *Pending Requests*: Függőben lévő kérelmek, elfogadás/elutasítás gombokkal.
* **Tabok közötti váltás:**
  + JavaScript váltja a tartalmat, alapértelmezett a barátok lista.
* **Barát badge:**
  + Kiemeli, hogy az adott felhasználó már barát.

**Üzenetküldő felület (Messages)**

* **Kétpaneles elrendezés:**
  + Bal oldalt barátlista, kattintva kiválasztható a chat partner.
  + Jobb oldalon a kiválasztott baráttal folytatott beszélgetés (üzenetlista, küldő űrlap).
* **Üzenetküldés:**
  + Űrlap POST-tal, azonnali frissítés a beszélgetésben.
* **Stílus:**
  + Sötét/világos mód, kiemelt saját üzenetek, időbélyeg.

### 8.9.4 Fórum komponens – Komponensleírás

A fórum modul lehetővé teszi a felhasználók számára közösségi interakciót, posztok létrehozását és kommentelést. A főbb elemek:

**Fórum főoldal (forum.blade.php)**

* **Posztlista:**
  + Minden poszt megjeleníti a szerzőt, címét, rövid tartalmat, kommentek számát és létrehozás dátumát
  + Privát posztok jelölve vannak sárga badge-el
  + Reszponzív grid elrendezésben
* **Szűrési lehetőségek:**
  + "All Posts" és "Friends' Posts" tabok
  + Rendezés: legújibb, legrégebbi, legtöbb komment
* **Új poszt gomb:**
  + Modális ablak nyílik új tartalom létrehozásához
  + Privát poszt opcióval (csak barátok láthatják)

**Poszt részletező oldal (post.blade.php)**

* **Poszt tartalma:**
  + Teljes cím és tartalom megjelenítése
  + Szerző avatárja (név kezdőbetűjével)
  + Kommentszekció hierarchikus megjelenítéssel
* **Kommentelés:**
  + Bejelentkezett felhasználók számára elérhető
  + Kompakt űrlap a poszt alatt
  + Saját kommentek törölhetők

**Moderációs funkciók**

* Poszt és komment törlése (adminok és tulajdonosok számára)
* Privát posztokhoz való hozzáférés korlátozása

# 9. Fejlesztői API és Fájlszintű Dokumentáció

Ebben a fejezetben részletesen bemutatjuk a projekt fejlesztői szemszögből lényeges technikai aspektusait. A leírás célja, hogy a rendszer minden fontosabb játékához és moduljához átfogó, kódszintű referenciát nyújtson.

A fejezet tartalmazza a főbb komponensekhez, játékokhoz tartozó JavaScript és backend logika, API-végpontok, osztályok, függvények, változók és algoritmusok részletes ismertetését.

## 9.0 Layoutok – Fejlesztői, kódszintű leírás

Ebben a részben a projekt főbb layoutjainak fejlesztői dokumentációját találod. A cél, hogy gyorsan átlásd, hogyan épülnek fel a layoutok, mik a főbb újrahasznosítható blokkok, és hogyan lehet őket bővíteni vagy testreszabni.

**app.blade.php**

**Globális layout, minden főoldalhoz.**

* **Navigáció:**

@include('layouts.navigation')

Minden oldal tetején megjelenik, automatikusan betölti a dinamikus menüpontokat, egyenlegkijelzőt, barátkérelmeket, profilmenüt3.

* **Tartalomhely:**

<main class="container mx-auto py-8">

@yield('content')

</main>

Az aktuális oldal tartalma ide kerül.

* **Chat bubble (bejelentkezett felhasználónak):**

@auth

<a href="{{ route('friends.messages') }}" class="fixed bottom-6 right-6 ...">

<i class="fas fa-comments text-xl"></i>

</a>

@endauth

Fix pozícióban, csak autentikált felhasználónak jelenik meg.

* **Sötét mód támogatás:**  
  A <html> és <body> tagekben Tailwind dark: osztályok, automatikus színváltás.
* **Assetek betöltése:**

@vite(['resources/css/app.css', 'resources/js/app.js'])

Vite build rendszerrel, gyors fejlesztői reload és cache-elés.

**game.blade.php**

**Minimal layout, játékoldalakhoz.**

* **Navigáció nélkül:**  
  Csak a fő tartalom jelenik meg, a navigációs sáv teljesen hiányzik, hogy a játékra koncentráljon a felhasználó.
* **Tartalomhely:**

<main class="container mx-auto py-8">

@yield('content')

</main>

* **Assetek:**  
  Ugyanúgy Vite-tel tölt be minden szükséges CSS/JS-t, mint az app layout.

**guest.blade.php**

**Vendég layout, látványos kezdőoldalhoz és auth oldalakhoz.**

* **Egyedi háttér és vizuális effektek:**  
  A <body>-ban .casino-bg háttér, átlátszó kártyák, glow effektek, egyedi betűtípusok.
* **Vendég navigáció:**  
  Csak bejelentkezés és regisztráció gombok jelennek meg, mobilon hamburger menü, amely JS-sel nyílik le.
* **Tartalomhely:**t

<div class="... content-container">

<div class="...">

{{ $slot }}

</div>

<!-- Footer -->

</div>

* **Példa gomb:**

<a href="{{ route('login') }}" class="text-yellow-400 hover:text-yellow-300 ...">Sign In</a>

**bettingarea.blade.php**

**Újrahasznosítható fogadási komponens minden játékhoz.**

* **Fogadási input:**

<input type="number" id="bet-input" min="1" placeholder="Enter bet amount" ...>

* **Aktuális tét kijelző:**

<p id="current-bet" ...>Current Bet: $0</p>

* **Integráció:**  
  Játékoldalakon:

@include('layouts.bettingarea')

**Login rendszer**

A bejelentkezési és regisztrációs logika a Laravel Breeze csomag alapértelmezett implementációját használja, minden auth-funkció (login, register, jelszó reset, email validáció) a Breeze által generált route-okon és view-kon keresztül történik, külön egyedi fejlesztés nélkül2.

**Példák layout használatra egy oldalon**

@extends('layouts.app')

@section('content')

<!-- Oldalspecifikus tartalom -->

@endsection

Játékoldalon:

text

@extends('layouts.game')

@section('content')

@include('layouts.bettingarea')

<!-- Játék UI -->

@endsection

**További megjegyzések fejlesztőknek**

* Minden layoutban a <meta name="csrf-token"> automatikusan generálódik, így AJAX/Fetch hívásoknál a CSRF védelem biztosított.
* A navigációs sáv Blade komponensben van, így könnyen bővíthető új menüpontokkal vagy funkciókkal (pl. admin panel link, értesítések).
* A layoutokban a Vite assetek miatt a fejlesztői környezetben automatikus HMR (Hot Module Reload) működik, éles környezetben cache-elés aktív.
* A bettingarea komponens minden játékhoz egységes fogadási logikát biztosít, JS-ből könnyen elérhető az input mező (#bet-input).

## 9.1 Játékok Technikai Dokumentációja

Az alábbi alfejezetekben minden játékhoz külön-külön, egységes szerkezetben található meg a részletes fejlesztői dokumentáció.

Ezek a szakaszok tartalmazzák:

* Az adott játék frontend (JavaScript) és backend (PHP/Laravel) logikájának főbb funkcióit,
* A fontosabb függvények, metódusok, eseménykezelők és változók leírását,
* Az API-kommunikáció menetét, végpontokat, paramétereket,
* A játék működésének algoritmikus magyarázatát, valamint a kód működését illusztráló példákat.

### 9.1.1 Blackjack játék – Részletes fejlesztői dokumentáció

**Layout: blackjack.blade.php**

A Blackjack játék felhasználói felülete a resources/views/pages/games/blackjack.blade.php fájlban található. Főbb elemek:

* **Balance kijelző:**  
  Megjeleníti a felhasználó aktuális egyenlegét (#user-balance).
* **Dealer szekció:**
  + Kártyák megjelenítése (#dealer-cards)
  + Pontszám kijelző (#dealer-score)
* **Akciógombok:**
  + Új játék indítása (#new-game-button)
  + "Hit" (lap húzása) (#hit-button)
  + "Stand" (megállás) (#stand-button)
* **Player szekció:**
  + Kártyák megjelenítése (#player-cards)
  + Pontszám kijelző (#player-score)
* **Fogadási terület:**  
  Az @include('layouts.betingarea') komponenssel integrálva.
* **Állapotüzenetek:**
  + Játékállapot (#game-status)
  + Eredmény (#game-result)
* **Token és assetek:**
  + API token rejtett mezőben (#api-token)
  + Stílus és script betöltése: @vite('resources/css/BJStyle.css'), @vite('resources/js/BJScript.js')

**Frontend logika: BJScript.js**

A Blackjack játék frontend logikája a resources/js/BJScript.js fájlban található. Főbb funkciók:

**Inicializálás és DOM elemek:**

* Betöltéskor kiválasztja a fő DOM elemeket (dealer/player kártyák, pontszámok, gombok, egyenleg, státuszok, fogadási input).
* Ellenőrzi az API tokent.

**Játék indítása (startNewGame):**

* Ellenőrzi a tétet, majd POST kérést küld a /api/blackjack/start végpontra.
* Siker esetén elrejti a tétmezőt, letiltja az új játék gombot, engedélyezi a "Hit" és "Stand" gombokat.
* Új paklit kér a Deck of Cards API-tól, majd kiosztja a kezdő lapokat a játékosnak és az osztónak.
* Ha a játékos blackjack-et kap, azonnal győzelem.

**Kártyakezelés:**

* fetchNewDeck(): új pakli lekérése a külső API-tól.
* drawCards(count): lap(ok) húzása a pakliból proxy-n keresztül.
* updateUI(): dealer és player kártyák, valamint pontszámok frissítése a felületen.

**Pontszám-számítás:**

* calculateScore(hand): figyelembe veszi az ászok értékét (1 vagy 11), figurák értéke 10.

**Játékos akciók:**

* **Hit:** új lap húzása, ha 21 fölé megy, automatikus veszteség.
* **Stand:** az osztó automatikusan húz lapokat 17-ig, majd összehasonlítás, eredmény meghatározása.

**Játék vége (endGame):**

* Eredmény szöveg megjelenítése, gombok tiltása, tétmező visszaállítása.
* POST kérés a /api/blackjack/end végpontra az eredmény, tét, játékos és osztó lapjaival.
* Egyenleg frissítése a backend válasza alapján.

**Egyéb funkciók:**

* Hibakezelés, státusz- és hibaüzenetek megjelenítése.
* Gombok eseménykezelői: új játék, hit, stand.

**Backend: BlackjackController.php**

A backend logika a app/Http/Controllers/BlackjackController.php fájlban található.

**Fő metódusok:**

* **start(Request $request):**
  + Ellenőrzi az egyenleget, levonja a tétet, menti az új egyenleget.
  + Naplózza a tranzakciót.
  + Válasz: { "success": true, "balance": <új egyenleg> } vagy hibaüzenet.
* **end(Request $request):**
  + Lekéri az eredményt, tétet, játékos és osztó lapjait.
  + Nyeremény esetén a tét kétszeresét jóváírja, döntetlen esetén visszaadja a tétet.
  + Frissíti a statisztikákat, létrehoz egy naplóbejegyzést a game\_logs táblában.
  + Válasz: { "success": true, "balance": <új egyenleg> }
* **checkBlackjack(array $hand):**
  + Ellenőrzi, hogy a kéz blackjack-e (ász + 10-es értékű lap).

**Naplózás és adatbázis:**

* Minden jelentős eseményt logol (Log::info).
* Minden játék eredménye bekerül a game\_logs táblába (felhasználó, tét, nyeremény, lapok, eredmény, időpont).

**API végpontok**

* **POST /api/blackjack/start** – Játék indítása, tét levonása.
* **POST /api/blackjack/end** – Játék befejezése, eredmény és nyeremény rögzítése.

**Folyamatábra**

1. **Játék indítása:**
   * Tét megadása → /api/blackjack/start → egyenleg frissítése
   * Pakli inicializálása (Deck of Cards API)
   * Kezdő lapok kiosztása
2. **Játékos akciók:**
   * "Hit": új lap húzása → UI frissítése, pontszám ellenőrzés
   * "Stand": osztó húz, eredmény meghatározás
3. **Játék vége:**
   * /api/blackjack/end hívása → eredmény, lapok, tét elküldése
   * Egyenleg és statisztikák frissítése

**Példakódok**

**Játék indítása frontendről:**

fetch('/api/blackjack/start', {

method: 'POST',

headers: { 'Content-Type': 'application/json', 'Authorization': `Bearer ${apiToken}` },

body: JSON.stringify({ bet\_amount: 100 })

})

.then(res => res.json())

.then(data => { /\* Egyenleg frissítése \*/ });

**Játék befejezése frontendről:**

fetch('/api/blackjack/end', {

method: 'POST',

headers: { 'Content-Type': 'application/json', 'Authorization': `Bearer ${apiToken}` },

body: JSON.stringify({

result: 'win',

bet\_amount: 100,

player\_hand: [...],

dealer\_hand: [...]

})

})

.then(res => res.json())

.then(data => { /\* Egyenleg frissítése \*/ });

### 9.1.2 Coinflip (Érmedobás) játék – Részletes fejlesztői dokumentáció

**Layout: coinflip.blade.php**

A Coinflip játék felhasználói felülete a resources/views/pages/games/coinflip.blade.php fájlban található. Főbb elemek:

* **Balance kijelző:**  
  Megjeleníti a felhasználó aktuális egyenlegét (#user-balance).
* **Játékinterfész:**
  + Választó gombok: "Heads" (#heads-btn) és "Tails" (#tails-btn)
  + Animált érme (#coin-container, #coin)
  + Eredménykijelző (#result)
  + "Play Again" gomb (#reset-btn)
  + Fogadási terület: az @include('layouts.betingarea') komponenssel integrálva
* **Token és assetek:**
  + API token rejtett mezőben (#api-token)
  + CSRF token meta tag
  + Stílus és script betöltése: @vite('resources/css/coinflip.css'), @vite('resources/js/coinflip.js')

**Frontend logika: coinflip.js**

A Coinflip játék frontend logikája a resources/js/coinflip.js fájlban található. Főbb funkciók:

* **Inicializálás:**
  + DOM elemek kiválasztása betöltéskor
  + Eseményfigyelők beállítása a fej/írás gombokra és a reset gombra
* **Játék indítása (startGame):**
  + Ellenőrzi a tétet, majd POST kérést küld a /api/coinflip/start végpontra
  + Siker esetén frissíti az egyenleget, kijelzi a tétet
* **Érmefeldobás logika:**
  + Véletlenszerűen generálja az érme eredményét (flipCoin())
  + Animált érmeforgatás CSS animációval (coin.style.animation = "flip 2s ease-in-out")
* **Játék befejezése (endGame):**
  + Összehasonlítja a játékos választását az érme eredményével
  + POST kérést küld a /api/coinflip/end végpontra az eredménnyel, tét összeggel, választással és érme eredménnyel
  + Frissíti az egyenleget és a kijelzőket a backend válasza alapján
* **Eredménymegjelenítés:**
  + Szöveges visszajelzés, hogy a játékos nyert vagy vesztett
  + Reset gombbal új játék indítható
* **Hibakezelés:**
  + Hibás tét vagy szerverhiba esetén figyelmeztető üzenet jelenik meg

**Backend: CoinflipController.php**

A backend logika a app/Http/Controllers/CoinflipController.php fájlban található.

**Fő metódusok:**

* **start(Request $request):**
  + Ellenőrzi a felhasználó egyenlegét, levonja a tétet, elmenti az új egyenleget
  + Naplózza a tranzakciót
  + Válasz: { "success": true, "balance": <új egyenleg> } vagy hibaüzenet
* **end(Request $request):**
  + Lekéri az eredményt (result), tétet (bet\_amount), játékos választását (user\_choice), és az érme eredményét (coin\_result)
  + Győzelem esetén a tét kétszeresét jóváírja, veszteség esetén nem történik jóváírás
  + Frissíti a felhasználó statisztikáit, létrehoz egy naplóbejegyzést a game\_logs táblában
  + Válasz: { "success": true, "balance": <új egyenleg> }

**Naplózás és adatbázis:**

* Minden jelentős eseményt logol (Log::info)
* Minden játék eredménye bekerül a game\_logs táblába (felhasználó, tét, nyeremény, választás, érme eredmény, időpont)

**API végpontok**

* **POST /api/coinflip/start** – Játék indítása, tét levonása
* **POST /api/coinflip/end** – Játék befejezése, eredmény és nyeremény rögzítése

**Folyamatábra**

1. **Játék indítása:**
   * Tét megadása → /api/coinflip/start → egyenleg frissítése
2. **Játékos választása:**
   * "Heads" vagy "Tails" gomb → érme animáció, véletlen eredmény generálása
3. **Játék vége:**
   * /api/coinflip/end hívása → eredmény, tét, választás, érme eredmény elküldése
   * Egyenleg és statisztikák frissítése

**Példakódok**

**Játék indítása frontendről:**

fetch('/api/coinflip/start', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${apiToken}`

},

body: JSON.stringify({ bet\_amount: 10 })

})

.then(res => res.json())

.then(data => { /\* Egyenleg frissítése \*/ });

**Játék befejezése frontendről:**

fetch('/api/coinflip/end', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${apiToken}`

},

body: JSON.stringify({

result: 'win',

bet\_amount: 10,

user\_choice: 'heads',

coin\_result: 'heads'

})

})

.then(res => res.json())

.then(data => { /\* Egyenleg frissítése \*/ });

### 9.1.3 Crash játék – Részletes fejlesztői dokumentáció

**Layout: crash.blade.php**

A Crash játék felhasználói felülete a resources/views/pages/games/crash.blade.php fájlban található. Főbb elemek:

* **Balance kijelző:**  
  Megjeleníti a felhasználó aktuális egyenlegét (#user-balance).
* **Játékinterfész:**
  + "Start Game" gomb (#start-game)
  + "Cash Out" gomb (#cash-out)
  + Multiplikátor kijelző (#current-multiplier) és progress bar
  + Játéktörténet lista (#crash-history)
  + Fogadási terület: az @include('layouts.betingarea') komponenssel integrálva
* **Token és assetek:**
  + API token rejtett mezőben (#api-token)
  + Stílus és script betöltése: @vite('resources/css/crash.css'), @vite('resources/js/crash.js')

**Frontend logika: crash.js**

A Crash játék frontend logikája a resources/js/crash.js fájlban található. Főbb funkciók:

* **Inicializálás:**  
  DOM elemek kiválasztása és eseményfigyelők beállítása a gombokhoz.
* **Játék indítása (startGameLogic):**
  + Generál egy egyedi játékazonosítót (currentGameId)
  + Véletlenszerű crash pont generálása (getRandomCrashPoint())
  + Intervallum indítása a multiplikátor növelésére (másodpercenként +0.01x)
  + POST kérés a /api/crash/start végpontra a tét levonásához
* **Multiplikátor logika:**

intervalId = setInterval(() => {

currentMultiplier += 0.01;

multiplierDisplay.textContent = `${currentMultiplier.toFixed(2)}x`;

progressFill.style.width = `${(currentMultiplier / 10) \* 100}%`;

if (currentMultiplier >= crashPoint) {

endGame('lose', currentBetAmount);

}

}, 200);

* **Cash Out kezelése:**
  + Megállítja a multiplikátor növekedését
  + POST kérés a /api/crash/end végpontra a nyereménnyel
  + UI frissítése a nyeremény megjelenítéséhez
* **Crash pont generálás:**

function getRandomCrashPoint() {

return Math.max(1.00, (1 / (1 - Math.random())));

}

Valószínűségi eloszlás: P(crash ≥ x) = 1/x

**Backend: CrashController.php**

A backend logika a app/Http/Controllers/CrashController.php fájlban található.

**Fő metódusok:**

* **start(Request $request):**
  + Ellenőrzi, hogy nincs-e aktív játék (cache lock)
  + Levonja a tétet a felhasználó egyenlegéből
  + Cache-be menti az aktív játék állapotát (2 percre)
  + Válasz: { "success": true, "balance": <új egyenleg> }
* **end(Request $request):**
  + Ellenőrzi a kérés érvényességét (game\_id, result, bet\_amount stb.)
  + Cache lock használatával megakadályozza a dupla feldolgozást
  + Nyeremény esetén jóváírja az egyenleget
  + Frissíti a felhasználó statisztikáit
  + Létrehoz egy naplóbejegyzést a game\_logs táblában

**Adatbázis struktúra:**

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| cashout\_multiplier | float | Kiszálláskori szorzó |
| crash\_point | float | Crash pont értéke |
| game\_id | string | Egyedi játékazonosító |

**API végpontok**

* **POST /api/crash/start**
  + Paraméterek: bet\_amount (number)
  + Válasz:

{

"success": true,

"balance": 950.00

}

* **POST /api/crash/end**
  + Paraméterek:

{

"game\_id": "abc123",

"result": "win",

"bet\_amount": 50,

"cashout\_multiplier": 3.45,

"crash\_point": 4.20,

"winnings": 172.50

}

* + Válasz:

{

"success": true,

"balance": 1122.50

}

**Folyamatábra**

1. **Játék indítása:**
   * Tét megadása → /api/crash/start → egyenleg frissítése
   * Multiplikátor indítása 1.00x-ről
2. **Játék menete:**
   * Multiplikátor másodpercenként +0.01x
   * Felhasználó bármikor kiszállhat ("Cash Out")
3. **Játék vége:**
   * Automatikus vége crash pont elérésekor
   * /api/crash/end hívása → eredmény rögzítése
   * Cache lock törlése

**Példakódok**

**Frontend indítás:**

fetch('/api/crash/start', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer xyz123'

},

body: JSON.stringify({ bet\_amount: 50 })

})

**Backend válasz hibás tét esetén:**

{

"success": false,

"message": "Insufficient balance"

}

### 9.1.4 Rock-Paper-Scissors Játék – Részletes fejlesztői dokumentáció

**Layout: rock.blade.php**

A Rock-Paper-Scissors játék felhasználói felülete a resources/views/pages/games/rock.blade.php fájlban található. Főbb elemek:

* **Balance kijelző:**  
  Megjeleníti a felhasználó aktuális egyenlegét (#user-balance).
* **Játékinterfész:**
  + Választó gombok (✊ Kő, ✋ Papír, ✌️ Olló)
  + Eredménykijelző (#result)
  + Játékos és számítógép választásának megjelenítése (#player-choice, #computer-choice)
  + Pontszámkijelzők (#player-score, #computer-score)
  + Fogadási terület: az @include('layouts.betingarea') komponenssel integrálva
* **Token és assetek:**
  + API token rejtett mezőben (#api-token)
  + Stílus és script betöltése: @vite('resources/css/rockpaperscissors.css'), @vite('resources/js/rockpaperscissors.js')

**Frontend logika: rockpaperscissors.js**

A játék frontend logikája a resources/js/rockpaperscissors.js fájlban található. Főbb funkciók:

* **Inicializálás:**
  + DOM elemek kiválasztása (gombok, kijelzők)
  + Eseményfigyelők beállítása a választó gombokhoz
* **Játéklogika:**
  + getComputerChoice(playerChoice): A számítógép választása kizárja a játékosét (nehézségi szint növelése)
  + determineWinner(): Győztes meghatározása klasszikus szabályok alapján
  + animateComputerChoice(): Számítógép választásának animált megjelenítése
* **API kommunikáció:**
  + /api/rock-paper-scissors/start: Tét levonása a játék indításakor
  + /api/rock-paper-scissors/end: Eredmény rögzítése a játék végén
* **Egyedi funkciók:**
  + Emoji-alapú vizuális visszajelzés
  + 1200ms animáció a számítógép választásához
  + Pontszám követés (playerScore, computerScore)

**Backend: RockPaperScissorsController.php**

A backend logika a app/Http/Controllers/RockPaperScissorsController.php fájlban található.

**Fő metódusok:**

* **start(Request $request):**
  + Ellenőrzi a felhasználó egyenlegét
  + Levonja a tétet és menti az új egyenleget
  + Válasz: { "success": true, "balance": <új egyenleg> }
* **end(Request $request):**
  + Feldolgozza az eredményt (win, lose, tie)
  + Nyeremény esetén jóváírja a tét 2x-esét, döntetlennél visszatéríti
  + Frissíti a statisztikákat és naplózza a játékot

**Adatbázis struktúra:**

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| player\_choice | string | Játékos választása (rock/paper/scissors) |
| computer\_choice | string | Számítógép választása |
| result | string | Játék eredménye |

**API végpontok**

* **POST /api/rock-paper-scissors/start**

{

"bet\_amount": 50

}

* **POST /api/rock-paper-scissors/end**

{

"result": "win",

"bet\_amount": 50,

"player\_choice": "rock",

"computer\_choice": "scissors"

}

**Játékmenet folyamata**

1. **Játék indítása:**
   * Tét megadása → /api/rock-paper-scissors/start hívás
   * Balance frissítése backend válasz alapján
2. **Választás és animáció:**
   * Játékos választ → számítógép választása generálódik
   * Animált választásmegjelenítés (1200ms)
3. **Eredmény kiértékelés:**
   * Győzelem: +2x tét, vesztés: -tét, döntetlen: tét visszatér
   * Pontszámok frissítése
   * /api/rock-paper-scissors/end hívás eredménnyel

**Egyedi algoritmusok**

**Számítógép AI (nehézségi szint):**

function getComputerChoice(playerChoice) {

const options = ["rock", "paper", "scissors"];

const filteredOptions = options.filter(choice => choice !== playerChoice);

return filteredOptions[Math.floor(Math.random() \* filteredOptions.length)];

}

Megjegyzés: A számítógép sosem választhatja a játékos által választott elemet, így 50% helyett 66% esélyt kap a győzelemre.

### 9.1.5 Rulett Játék – Részletes fejlesztői dokumentáció

**Layout: roulette.blade.php**

A Rulett játék felhasználói felülete a resources/views/pages/games/roulette.blade.php fájlban található. Főbb elemek:

* **Rulett kerék:**  
  Dinamikusan generált számokkal és színekkel (#roulette-container).
* **Fogadási terület:**
  + Tét összegének megadása (@include('layouts.betingarea'))
  + Fogadási típus választó (#bet-type): Even, Odd, Red, Black, Green
* **Spin gomb:**  
  Játék indításához (#spin-button).
* **Eredménykijelző:**  
  Nyert szám, szín és típus (#result).
* **Előző nyeremények:**  
  Legutóbbi 10 nyerő szám megjelenítése (#previous-winners-container).
* **Balance kijelző:**  
  Felhasználó aktuális egyenlege (#user-balance).

**Frontend logika: roulette.js**

A játék frontend logikája a resources/js/roulette.js fájlban található. Főbb funkciók:

* **Inicializálás:**
  + Rulett számok és színek definiálása (0-36 + zöld 0)
  + Fogadási opciók betöltése a select elembe
* **Spin animáció:**
  + Gyorsuló kerékforgás szimulációja CSS és JavaScript kombinációval
  + Véletlenszerű nyerőszám kiválasztása

let delay = 50;

let currentIndex = Math.floor(Math.random() \* rouletteNumbers.length);

const spinStep = () => {

// Animáció frissítése minden lépésben

if (iterations >= maxIterations) {

showResult(currentElement);

return;

}

currentIndex = (currentIndex + 1) % rouletteNumbers.length;

setTimeout(spinStep, delay);

};

* **Eredményfeldolgozás:**
  + Nyertes szám színének és paritásának meghatározása
  + Nyeremény kiszámítása a fogadási típus alapján

const showResult = (winner) => {

const isEven = winner.number !== 0 && winner.number % 2 === 0 ? "Even" : "Odd";

// API hívás a nyeremény rögzítéséhez

fetch('/api/roulette/end', { ... });

};

**Backend: RouletteController.php**

A backend logika a app/Http/Controllers/RouletteController.php fájlban található.

**Fő metódusok:**

* **start(Request $request):**
  + Ellenőrzi a tétet (1-1000 közötti szám)
  + Levonja a tétet a felhasználó egyenlegéből
  + Naplózza a tranzakciót

public function start(Request $request)

{

$validated = $request->validate([

'bet\_amount' => 'required|numeric|min:1|max:1000',

'bet\_type' => 'required|in:even,odd,red,black,green'

]);

$user->balance -= $betAmount;

$user->save();

}

* **end(Request $request):**
  + Kiszámolja a nyereményt a fogadási típus alapján
  + Frissíti a felhasználó statisztikáit
  + Létrehoz egy naplóbejegyzést a game\_logs táblában

private function getPayoutMultiplier(string $betType): int

{

return match($betType) {

'green' => 35, // Zöld (0) esetén 35x

default => 2 // Egyéb fogadások 2x

};

}

**API végpontok**

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP Métodus | URL | Leírás | Paraméterek |
| POST | /api/roulette/start | Játék indítása, tét levonása | bet\_amount, bet\_type |
| POST | /api/roulette/end | Eredmény rögzítése | result, bet\_amount, bet\_type, winnings |

**Példa válaszok:**

// /api/roulette/start válasza

{

"success": true,

"balance": 950.00

}

// /api/roulette/end válasza

{

"success": true,

"balance": 1000.00

}

**Adatbázis struktúra**

A game\_logs tábla tartalmazza:

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| bet\_type | string | Fogadási típus (even/red/green stb.) |
| payout\_multiplier | integer | Nyereményszorzó (2x vagy 35x) |
| game\_data | JSON | Eredmény részletei (szín, paritás) |

**Egyedi implementációk**

1. **Gyorsított animáció:**  
   A kerékforgás animációja kezdetben gyorsan indul, majd fokozatosan lassul:

delay = Math.min(delay + 30, 500); // Aggresszív lassítás

1. **Színkódolt eredmények:**  
   Az előző nyeremények szín szerint jelennek meg:

`<div style="background-color: ${w.color};">${w.number}</div>`

1. **Komplex nyereményszabályok:**
   * Zöld (0): 35x tét
   * Piros/Fekete: 2x tét
   * Páros/Páratlan: 2x tét

### 9.1.6 Slot Machine (Játékgép) – Részletes fejlesztői dokumentáció

**Layout: slot.blade.php**

A Slot Machine játék felhasználói felülete a resources/views/pages/games/slot.blade.php fájlban található. Főbb elemek:

* **Balance kijelző:**  
  Megjeleníti a felhasználó aktuális egyenlegét (#user-balance).
* **Játékinterfész:**
  + 5 db pörgethető henger (reel) szimbólumokkal (#reel1–#reel5)
  + "Spin Reels" (pörgetés) és "Auto Spin" gombok
  + Nyereménytörténet lista (#win-history)
  + Aktuális játék eredmény kijelző (#game-result)
  + Multiplikátor kijelző és progress bar (#current-multiplier, #progress-fill)
  + Info gomb, amely modális ablakban mutatja a szabályokat és a kifizetési táblázatot
* **Fogadási terület:**  
  Az @include('layouts.betingarea') komponenssel integrálva.
* **Token és assetek:**
  + API token rejtett mezőben (#api-token)
  + Stílus és script betöltése: @vite('resources/css/slot.css'), @vite('resources/js/slot.js')

**Frontend logika: slot.js**

A Slot Machine játék frontend logikája a resources/js/slot.js fájlban található. Főbb funkciók:

* **Szimbólumok és kifizetési tábla:**
  + 5-féle szimbólum: 🍒, 🍋, 🍉, ⭐, 💎
  + Kifizetések: legalább 3 azonos szimbólum esetén, a payoutTable objektumban részletezve
* **DOM elemek kezelése:**
  + Hengerek, gombok, egyenlegkijelző, eredménykijelző, nyereménytörténet, modális ablak
* **Játék indítása:**
  + A "Spin Reels" gomb eseménykezelője POST kérést küld a /api/slot-machine/start végpontra, amely levonja a tétet
  + Animált pörgetés: a hengerek szimbólumai gyorsan változnak, majd megállnak a végleges szimbólumokon
* **Eredmény kiértékelése:**
  + A kód kiszámolja, hány azonos szimbólum jelent meg, és ezek alapján meghatározza a nyereményt
  + POST kérést küld a /api/slot-machine/end végpontra az eredménnyel, tét összeggel, hengerek állásával és nyereménnyel
  + A válaszban kapott egyenleget frissíti a felületen
  + A nyereménytörténet listában megjeleníti a legutóbbi nyerő kombinációkat
* **Auto Spin:**
  + Az "Auto Spin" gombbal a játék automatikusan többször is pörget, amíg van elegendő egyenleg
* **Modal kezelés:**
  + Az info gomb és a modális ablak eseménykezelői biztosítják a szabályok és kifizetések gyors elérhetőségét
* **Stílus:**
  + A modal animációk, gombok és egyéb vizuális elemek CSS-sel lettek kialakítva a játékélmény fokozásához

**Backend: SlotController.php**

A backend logika a app/Http/Controllers/SlotController.php fájlban található.

**Fő metódusok:**

* **start(Request $request):**
  + Ellenőrzi, hogy a felhasználó egyenlege elegendő-e a tét megtételéhez
  + Levonja a tét összegét az egyenlegből, majd menti az új egyenleget
  + Naplózza a tranzakciót
  + Visszaadja az új egyenleget JSON válaszban
* **end(Request $request):**
  + Fogadja a játék eredményét, a pörgetett szimbólumokat és a nyeremény összegét
  + Ha a játékos nyert, jóváírja a nyereményt az egyenlegén
  + Naplózza az eredményt és frissíti a felhasználó statisztikáit
  + Létrehoz egy új bejegyzést a game\_logs táblában, amely tartalmazza a játék adatait (hengerek, eredmény, tét, nyeremény)
  + Visszaadja az új egyenleget JSON válaszban

**Adatbázis integráció**

A game\_logs tábla a következőket rögzíti:

* Felhasználó azonosítója
* Játék típusa (slot-machine)
* Tét összege
* Nyeremény összege
* Eredmény (is\_win)
* Hengerek végleges állása (game\_data)
* Játék időpontja

**API végpontok**

* **POST /api/slot-machine/start** – A játék indításához (tét levonása)
* **POST /api/slot-machine/end** – A játék befejezéséhez (eredmény és nyeremény rögzítése)
* **GET /api/user/balance** – Az aktuális egyenleg lekérdezése

**Példakódok**

**Játék indítása frontendről:**

fetch('/api/slot-machine/start', {

method: "POST",

headers: {

Authorization: `Bearer ${document.getElementById("api-token").value}`,

"Content-Type": "application/json",

},

body: JSON.stringify({ bet\_amount: betAmount }),

})

.then(res => res.json())

.then(data => { /\* Egyenleg frissítése \*/ });

**Játék befejezése frontendről:**

fetch('/api/slot-machine/end', {

method: "POST",

headers: {

Authorization: `Bearer ${document.getElementById("api-token").value}`,

"Content-Type": "application/json",

},

body: JSON.stringify({

result: winnings > 0 ? "win" : "lose",

bet\_amount: betAmount,

reels: result.join(","),

winnings,

}),

})

.then(res => res.json())

.then(data => { /\* Egyenleg frissítése \*/ });

**Folyamatábra**

1. **Játék indítása:**
   * Tét megadása → /api/slot-machine/start → egyenleg frissítése
   * Animált hengerek pörgetése
2. **Eredmény kiértékelése:**
   * Kombinációk vizsgálata, nyeremény számítása
   * /api/slot-machine/end hívása → eredmény, tét, hengerek, nyeremény elküldése
   * Egyenleg és statisztikák frissítése
3. **Auto Spin:**
   * Automatikus pörgetés, amíg van elegendő egyenleg

### 9.1.7 Mines (Aknakereső) Játék – Részletes fejlesztői dokumentáció

**Layout: mines.blade.php**

A Mines játék felhasználói felülete a resources/views/pages/games/mines.blade.php fájlban található. Főbb elemek:

* **Balance kijelző:**  
  Megjeleníti a felhasználó aktuális egyenlegét (#user-balance).
* **Játékinterfész:**
  + 5x5 rács (#mines-grid) 25 mezővel
  + "Start Game" és "Cash Out" gombok
  + Multiplikátor kijelző (#current-multiplier) és progress bar
  + Játéktörténet lista (#game-history)
* **Info modális:**  
  Szabályok és nyereményszorzók megjelenítése
* **Fogadási terület:**  
  Az @include('layouts.betingarea') komponenssel integrálva

**Frontend logika: mines.js**

A játék frontend logikája a resources/js/mines.js fájlban található. Főbb funkciók:

**1. Grid inicializálás**

function createGrid() {

for (let i = 0; i < 25; i++) {

const tile = document.createElement("div");

tile.className = "mine-tile";

tile.dataset.index = i;

minesGrid.appendChild(tile);

}

}

**2. Játék indítása**

* Tét validálása
* API hívás a /api/mines/start végpontra
* 7 akna véletlenszerű elhelyezése

async function startGame() {

const response = await fetch("/api/mines/start", {

method: "POST",

body: JSON.stringify({ bet\_amount: currentBet })

});

// ...

}

**3. Mezőkattintás kezelése**

function handleTileClick(e) {

const index = parseInt(e.target.dataset.index);

if (grid[index] === "mine") endGame("lose");

else updateMultiplier();

}

**4. Multiplikátor számítás**

function updateMultiplier() {

const safeCells = GRID\_SIZE - minesCount - revealedCount;

multiplier = 1 + (maxMultiplier - 1) \* (revealedCount / safeCells);

multiplierDisplay.textContent = multiplier.toFixed(2) + "x";

}

**5. Cash Out logika**

async function cashOut() {

const winnings = betAmount \* multiplier;

await fetch("/api/mines/end", {

method: "POST",

body: JSON.stringify({

result: "win",

revealed\_cells: revealedCount,

winnings: winnings

})

});

}

**Backend: MinesController.php**

**Fő metódusok:**

public function start(Request $request) {

// Tét levonás

$user->balance -= $request->bet\_amount;

$user->save();

}

public function end(Request $request) {

// Nyeremény számítás

if ($request->result === 'win') {

$user->balance += $request->winnings;

}

// Naplózás

GameLog::create([

'game\_data' => json\_encode([

'mines\_count' => 7,

'multiplier' => $request->winnings / $request->bet\_amount

])

]);

}

**Adatbázis struktúra:**

|  |  |  |
| --- | --- | --- |
| Mező | Típus | Leírás |
| mines\_count | integer | Aknák száma (fix 7) |
| revealed\_cells | integer | Felfedett biztonságos mezők |
| multiplier | float | Kiszálláskori szorzó |

**API végpontok**

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Paraméterek | Válasz |
| POST | /api/mines/start | bet\_amount | {balance: új\_egyenleg} |
| POST | /api/mines/end | result, bet\_amount, revealed\_cells | {balance: új\_egyenleg} |

**Folyamatábra**

1. Start Game → API hívás → Tét levonás

2. Rács generálás 7 aknával

3. Mező felfedés:

- Akna → Veszteség → Minden akna felfedése

- Biztonságos → Szorzó növelés

4. Cash Out → Nyeremény jóváírás

5. Játék végén API hívás eredmény rögzítésére

**Példa API kommunikáció**

**Start kérés:**

fetch('/api/mines/start', {

method: 'POST',

body: JSON.stringify({ bet\_amount: 50 })

})

**End kérés:**

fetch('/api/mines/end', {

method: 'POST',

body: JSON.stringify({

result: "win",

bet\_amount: 50,

revealed\_cells: 12,

winnings: 600

})

})

## 9.2 API és Web útvonalak részletes leírása

### 9.2.1 API útvonalak (api.php)

**Főbb jellemzők:**

* **Kötelező hitelesítés**: Minden útvonal a auth:sanctum middleware alatt fut
* **JSON válaszok**: Minden végpont JSON formátumban ad vissza adatokat
* **Játékok logikája**: Minden játékhoz külön start és end végpont

**Útvonal lista:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| HTTP Métodus | URL | Controller Művelet | Leírás | Paraméterek | Példa hívás |
| GET | /sanctum/csrf-cookie | CsrfCookieController::show | CSRF token generálás | - | Axios.get('/sanctum/csrf-cookie') |
| GET | /user | Closure | Felhasználói adatok | - | axios.get('/api/user') |
| POST | /blackjack/start | BlackjackController::start | Blackjack játék indítása | bet\_amount | axios.post('/api/blackjack/start', {bet\_amount: 50}) |
| POST | /blackjack/end | BlackjackController::end | Blackjack eredmény | result, player\_hand, dealer\_hand | axios.post('/api/blackjack/end', {...}) |
| POST | /coinflip/start | CoinflipController::start | Érmedobás indítás | bet\_amount | axios.post('/api/coinflip/start', {bet\_amount: 20}) |
| POST | /coinflip/end | CoinflipController::end | Érmedobás eredmény | result, user\_choice | axios.post('/api/coinflip/end', {result: 'win'}) |
| POST | /rock-paper-scissors/start | RockPaperScissorsController::start | Kő-Papír-Olló indítás | bet\_amount | axios.post('/api/rock-paper-scissors/start', {bet\_amount: 30}) |
| POST | /rock-paper-scissors/end | RockPaperScissorsController::end | KPO eredmény | result, player\_choice | axios.post(...) |
| POST | /slot-machine/start | SlotController::start | Játékgép indítás | bet\_amount | axios.post('/api/slot-machine/start', {bet\_amount: 10}) |
| POST | /slot-machine/end | SlotController::end | Játékgép eredmény | result, reels | axios.post(...) |
| POST | /roulette/start | RouletteController::start | Rulett indítás | bet\_amount, bet\_type | axios.post(...) |
| POST | /roulette/end | RouletteController::end | Rulett eredmény | result, number | axios.post(...) |
| POST | /crash/start | CrashController::start | Crash játék indítás | bet\_amount | axios.post(...) |
| POST | /crash/end | CrashController::end | Crash eredmény | result, multiplier | axios.post(...) |
| POST | /mines/start | MinesController::start | Aknakereső indítás | bet\_amount | axios.post(...) |
| POST | /mines/end | MinesController::end | Aknakereső eredmény | result, revealed\_cells | axios.post(...) |
| GET | /user/balance | Closure | Egyenleg lekérdezés | - | axios.get('/api/user/balance') |
| POST | /generate-token | AuthController::generateToken | Fejlesztői token | - | axios.post(...) |

### 9.2.2 Web útvonalak (web.php)

**Főbb jellemzők:**

* **Névvel ellátott útvonalak:** Minden route-nak van neve (->name())
* **Middlewerek:** Alapértelmezett web middleware csoport (session, CSRF)
* **Blade template-ek:** Minden útvonal Blade nézetet renderel

**Főbb útvonal csoportok:**

**1. Hitelesítés nélküli útvonalak**

Route::get('/', fn() => view('welcome'));

Route::get('/home', fn() => redirect()->route('dashboard'));

**2. Authentikált útvonalak (auth middleware)**

// Profilkezelés

Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');

// Főoldalak

Route::get('/mainpage', [MainPageController::class, 'index'])->name('pages.mainpage');

Route::get('/games', [GameController::class, 'index'])->name('pages.games');

// Játékok

Route::get('/games/blackjack', [GameController::class, 'blackjack'])->name('blackjack');

Route::get('/games/mines', [GameController::class, 'mines'])->name('mines');

// ...hasonlóan a többi játékhoz

**3. Admin útvonalak**

Route::middleware(['auth', AdminMiddleware::class])->prefix('admin')->group(function() {

Route::get('/', [AdminController::class, 'dashboard'])->name('admin.dashboard');

Route::patch('/{user}/suspend', [UserController::class, 'suspend']);

});

**4. Resource útvonalak**

// Fórum

Route::post('/forum/post', [ForumController::class, 'store'])->name('forum.post.store');

Route::delete('/forum/post/{id}', [ForumController::class, 'destroy']);

// Barátok

Route::post('/friends/request/{id}', [FriendController::class, 'sendRequest']);

### 9.2.3 Útvonalhivatkozások

**API útvonalak hivatkozása:**

// Példa Axios hívás

axios.post(route('blackjack.start'), {

bet\_amount: 50

}).then(response => {

console.log(response.data);

});

**Web útvonalak hivatkozása Blade-ben:**

<a href="{{ route('blackjack') }}">Blackjack</a>

<form action="{{ route('forum.post.store') }}" method="POST">

@csrf

...

</form>

**Paraméteres útvonalak:**

// Útvonal definíció

Route::get('/forum/post/{id}', [PostController::class, 'show'])->name('forum.post.show');

// Hivatkozás

<a href="{{ route('forum.post.show', ['id' => $post->id]) }}">

{{ $post->title }}

</a>

### 9.2.4 Fontos megjegyzések

1. **Middlewerek különbségei:**
   * **API útvonalak:** auth:sanctum middleware (token alapú hitelesítés)
   * **Web útvonalak:**web middleware (session alapú hitelesítés)
2. **CORS konfiguráció:**

// config/cors.php

'paths' => ['api/\*', 'sanctum/csrf-cookie'],

'allowed\_methods' => ['\*'],

'allowed\_origins' => ['http://localhost:3000'], // React/Vue frontend

1. **Rate Limiting:**

// App\Http\Kernel

'api' => [

'throttle:api',

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

1. **Validáció példa:**

public function start(Request $request)

{

$validated = $request->validate([

'bet\_amount' => 'required|numeric|min:0.1'

]);

// ...

}

## 9.3 Nem játék elemek fejlesztői dokumentációja

### 9.3.1 Fejlesztői dokumentáció: Profil statisztika oldal

**Layout és főbb komponensek**

* **Fájl:** resources/views/pages/profile/statistics.blade.php
* **Layout:** @extends('layouts.app')
* **Grid szerkezet:**
  + 2 oszlop mobilon, 4 oszlop desktopon a fő statisztikáknak.
  + Játékstatisztika diagramok 1 vagy 2 oszlopban.

**Adatmodell és backend**

* $user->balance: aktuális egyenleg
* $user->gameStat: kapcsolódó GameStat modell
* $user->gameStat->game\_statistics: asszociatív tömb, minden játékhoz külön statisztika (pl. wins, losses)

**Controller példa:**

public function statistics()

{

$user = Auth::user()->load('gameStat');

return view('pages.profile.statistics', compact('user'));

}

**Frontend logika**

* **JavaScript**:
  + A játékstatisztikák JSON formátumban érkeznek a backendről.
  + Minden játékhoz egyedi canvas ID generálódik.
  + A script minden játékhoz inicializál egy doughnut chartot, amely vizuálisan mutatja a győzelmek és vereségek arányát.
* **Stílus:**
  + .chart-container gondoskodik a reszponzív méretezésről.
  + Tailwind CSS biztosítja a sötét mód támogatást.

**Bővíthetőség és fejlesztői megjegyzések**

* Új játék hozzáadásakor csak a backendnek kell bővítenie a game\_statistics mezőt.
* A komponens automatikusan kezeli, ha nincs statisztika (figyelmeztető kártya).
* Minden főbb mutató (nyeremény, veszteség, profit) külön kártyán jelenik meg, így könnyen hozzáadható más kártya is a meglévők változtatása nélkül is, a meglévőket könnyen lehet egymástól függetlenül változtatni

### 9.3.2 Reklámfelület és hirdetéskezelő – Fejlesztői dokumentáció

#### 9.3.2.1 Wallet modul – kódszintű áttekintés

**Blade és fő DOM elemek**

* **Fájl:** resources/views/pages/wallet.blade.php
* **Balance kijelző:** #balance-amount
* **Hirdetés kártyák:** .ad-card, gomb: .ad-btn
* **Modal:** #progress-modal, progress: #progress-bar, szünet: #pause-button, videó: #ad-video, kép: #ad-image, forfeit ablak: #forfeit-option

**JS logika (wallet.js)**

* **Ad modal megnyitása:**  
  .ad-btn gombra kattintva openAdModal(adId, duration, mediaType, title) inicializálja a modalt, betölti a megfelelő médiát (videó vagy kép), elindítja a progress animációt.
* **Progress animáció:**  
  animateProgress() requestAnimationFrame-mel növeli a progress bart, a teljes idő eléréséig.
* **Szünet/folytatás:**  
  togglePause() leállítja/újraindítja az animációt, gomb színe és felirata vált.
* **Forfeit logika:**  
  Bezárás előtt, ha nincs kész a progress, szünetel, megjelenik a forfeit modal, döntés szerint folytat vagy lemond a jutalomról.
* **Jutalom feldolgozás:**  
  closeModal(noReward) POST-ot küld /wallet/process-ad végpontra, paraméterek: ad\_id, watched\_seconds. Siker esetén egyenleg frissül, alertben jutalom összege.
* **Hibakezelés:**  
  Hálózati vagy logikai hibák esetén alert, progress reset.

**CSS (wallet.css) főbb elemek**

* Modal, progress bar, kártyák, forfeit ablak egyedi, reszponzív stílus.
* Hover, árnyék, átmenetek, sötét/világos mód támogatás.

**Backend logika (WalletController)[wallet]**

* **index()**:  
  Lekéri a felhasználó egyenlegét és az aktív hirdetéseket.
* **processAd(Request $request)**:  
  Validálja a hirdetés ID-t és a nézett időt, kiszámolja az arányos jutalmat, tranzakcióban frissíti az egyenleget és logolja a feltöltést (balance\_uploads), visszaadja az új egyenleget és jutalmat JSON-ban.
* **Jutalom számítás:**  
  calculateAdReward($adId, $watchedSeconds) – a nézett idő arányában számolja a jutalmat (max a teljes reward).

#### 9.3.2.2 Admin hirdetéskezelő – kódszintű áttekintés

**Blade nézetek**

* **index:**  
  Hirdetések listázása táblázatban, minden sorban szerkesztés, törlés gomb.
* **form:**  
  Új hirdetés vagy szerkesztés űrlap (cím, leírás, időtartam, jutalom, média típus, thumbnail, fő média), meglévő média előnézete.

**Controller (AdvertisementController)**

* **index()**:  
  Paginált hirdetéslista, átadja az összes adatot a nézetnek.
* **create()/edit()**:  
  Létrehozó/szerkesztő nézet.
* **store()/update()**:  
  Validálás, hirdetés mentése/frissítése, médiafájlok kezelése.
* **destroy()/forceDestroy()**:  
  Soft delete és végleges törlés, médiafájlok törlése.
* **handleMediaUpload()**:  
  Kép és videó feltöltés, elérési utak mentése az adatbázisba.

**Adatmodell**

* **advertisements**:  
  Minden hirdetéshez: cím, leírás, időtartam, jutalom, típus (kép/videó), thumbnail, média elérési út, státusz (is\_active), készítő/frissítő.

#### 9.3.2.3 Admin Dashboard – felhasználókezelés

**Blade nézet (admin/dashboard)**

* **Felépítés:**
  + Főcím, üdvözlés, "Manage advertisements" gomb.
  + Flash üzenetek (siker/hiba).
  + Felhasználók táblázata: ID, név, email, egyenleg, szerepkör (admin/user), akciók (info, felfüggesztés/visszaállítás, végleges törlés).
  + Soft delete vizuális megjelenítése (suspended → halvány sor).
* **Akciók:**
  + Info: felhasználói statisztikák megtekintése.
  + Suspend/Restore: PATCH kérés, soft delete vagy visszaállítás.
  + Delete: csak felfüggesztett felhasználón, végleges törlés.

**Controller (AdminController, UserController)**

* **dashboard()**:  
  Minden felhasználó (aktív és felfüggesztett) listázása.
* **showById($id)**:  
  Felhasználó adatainak és statisztikáinak megjelenítése (JSON dekódolás, stat nézet).
* **suspend() / destroy()**:  
  Soft delete (felfüggesztés) és végleges törlés, státusz visszajelzés.

**Adatbázis séma (összefoglaló)**

* **advertisements**:
  + id, title, description, duration\_seconds, reward\_amount, media\_type, image, image\_path, video\_path, is\_active, (soft delete)
* **balance\_uploads**:
  + id, user\_id, advertisement\_id, duration\_watched, reward, status
* **users**:
  + id, is\_admin, name, email, balance, (soft delete)

### 9.3.4 Leaderboard – Fejlesztői dokumentáció

**Layout és főbb elemek**

* **Fájl:** resources/views/pages/leaderboard.blade.php
* **Layout:** @extends('layouts.app')
* **Fő DOM elemek:**
  + Lista: minden sor egy játékos adatait tartalmazza (helyezés, név, egyenleg, utolsó aktivitás)
  + Adminok külön stílusos címkével
  + Lapozás: {{ $users->links() }}

**Backend logika**

* **Controller:** LeaderboardController
* **Fő metódus:**

public function index()

{

// Felhasználók lekérése egyenleg szerint csökkenő sorrendben, lapozva

$users = User::orderBy('balance', 'desc')->paginate(20);

return view('pages.leaderboard', compact('users'));

}

* **Adatforrás:**
  + Minden felhasználó, akinek van egyenlege, bekerül a listába.
  + Az adminok is megjelennek, külön címkével.

**Frontend viselkedés**

* **Kiemelés:**
  + Az első három helyezett külön színnel, nagyobb számmal jelenik meg.
* **Reszponzív:**
  + Mobilon kevesebb oszlop látszik, desktopon minden adat elérhető.
* **Sötét mód:**
  + Minden elem Tailwind dark: osztályokkal támogatja a sötét témát.

**Lapozás**

* **Laravel paginator:**
  + A paginate(20) automatikusan kezeli a lapozást és az oldalszámokat.
  + A Blade-ben a {{ $users->links() }} elem jeleníti meg a lapozót.

**Bővíthetőség**

* Később lehetőség van további oszlopok (pl. győzelmek, játékok száma) hozzáadására.
* A rangsorolás módja módosítható (pl. statisztika, aktivitás alapján).

### 9.3.5 Barátok és üzenetek – Fejlesztői dokumentáció

**Blade nézetek és fő DOM elemek**

* **Add Friends (pages/friends.blade.php):**
  + Keresőmező: #search
  + Rendező select: #sort
  + Felhasználó kártyák: .user-card (adat attribútumok: data-name, data-balance)
  + Barátkérés gomb: form POST route('friends.request', $user->id)
* **Barátlista (pages/friendslist.blade.php):**
  + Tabok: #tab-friends, #tab-pending
  + Lista tartalom: #tab-content-friends, #tab-content-pending
  + Elfogadás/elutasítás gomb: form POST az adott request id-vel
* **Üzenetek (pages/messages.blade.php):**
  + Barátlista: bal oldali panel, linkek a chat partnerekhez
  + Üzenetlista: üzenetek időbélyeggel, saját üzenetek kiemelve
  + Üzenetküldő form: input + submit, POST route('friends.messages.send')

**JavaScript logika**

* **Keresés és rendezés (Add Friends):**
  + Keresés: minden billentyűleütésnél szűri a .user-card elemeket.
  + Rendezés: select változásra sorba rendezi a DOM elemeket (név vagy egyenleg szerint).
* **Tabváltás (Friendslist):**
  + showTab(tab) függvény mutatja a kiválasztott tartalmat, tabok stílusát váltja.
  + Oldalbetöltéskor hash alapján automatikusan a megfelelő tab aktív.
* **Üzenetküldés:**
  + Form submit POST-tal, siker után redirect a beszélgetéshez.

**Backend logika**

* **Felhasználók keresése és rendezése:**
  + A controller kizárja a már barátokat, függőben lévő kérelmeket és saját magát.
  + Keresés és rendezés query paraméterek alapján.
* **Barátkérés küldése:**
  + POST /friends/request/{id} – csak ha nincs már kapcsolat vagy függőben lévő kérelem.
* **Barátkérelem elfogadása/elutasítása:**
  + Elfogadás: barátság rekord létrehozása, kérelem törlése.
  + Elutasítás: kérelem törlése.
* **Barátlista és kérelmek lekérdezése:**
  + Elfogadott barátok és függőben lévő kérelmek külön gyűjtve.
* **Üzenetek:**
  + Kétirányú üzenetlista lekérdezése a kiválasztott baráttal.
  + Új üzenet validálása és mentése.

**Adatbázis séma**

* **users**: felhasználók
* **friendships**: baráti kapcsolatok (kétirányú, mindkét fél ID-ja)
* **friend\_requests**: függőben lévő kérelmek (sender\_id, receiver\_id, status)
* **messages**: privát üzenetek (sender\_id, receiver\_id, message, created\_at)

**Bővíthetőség, megjegyzések**

* A kereső és rendező logika bővíthető további szempontokkal (pl. aktivitás).
* Az üzenetküldés AJAX-szal is megvalósítható a gyorsabb UX érdekében.
* A barátlista és üzenetpanel mobilon is jól használható, minden komponens reszponzív.
* A chat bubble a layoutban minden oldalon elérhető, gyors hozzáférést ad az üzenetküldéshez.

### 9.3.6 Fórum modul – Fejlesztői dokumentáció

**Blade nézetek és fő DOM elemek**

* **Fórum főoldal (forum.blade.php):**
  + Posztlista: minden poszt kártyán jelenik meg, szerző, cím, rövid tartalom, kommentek száma, privát badge.
  + Szűrők: barátok posztjai / minden poszt tab, rendező select (legújabb, legrégebbi, legtöbb komment).
  + Új poszt modal: gombbal nyitható, cím, tartalom, privát opció.
* **Poszt részletező (post.blade.php):**
  + Poszt tartalom: szerző, dátum, privát státusz, teljes szöveg.
  + Kommentlista: hozzászólások szerzővel, időbélyeggel, törlés gombbal (saját vagy admin).
  + Komment űrlap: textarea és küldés gomb, csak bejelentkezett felhasználóknak.
* **Stílus:**
  + Reszponzív grid, sötét/világos mód, Tailwind CSS.
  + Privát poszt badge, törlés gombok, hover effektek.

**Backend logika**

* **Fórum főoldal (ForumController):**
  + Posztok lekérdezése, ahol nincs törölve.
  + Szűrés: csak barátok posztjai, privát/nyilvános posztok jogosultság szerint.
  + Rendezés: legújabb, legrégebbi, legtöbb komment.
  + Minden poszt szerzővel és kommentek számával együtt kerül átadásra.
* **Poszt létrehozás/törlés:**
  + Új poszt: cím, tartalom, privát opció, szerző automatikusan az aktuális felhasználó.
  + Törlés: csak a poszt szerzője vagy admin végezheti.
* **Poszt részletező (PostController):**
  + Jogosultság-ellenőrzés privát poszt esetén (csak szerző vagy barát láthatja).
  + Kommentek listázása, hozzászólás hozzáadása, komment törlése (csak szerző vagy admin).

**Adatbázis séma**

* **posts**:
  + user\_id, title, body, is\_private, deleted\_by, timestamps
* **comments**:
  + post\_id, user\_id, body, timestamps

**Frontend viselkedés**

* **Új poszt modal:**
  + JavaScript kezeli a modal nyitását/zárását, ESC-re és háttérre kattintva is zárható.
* **Tabváltás, szűrés:**
  + Tabok és rendező select segítségével a lista azonnal frissül.
* **Kommentelés:**
  + Sikeres küldés után a kommentlista frissül, hibák esetén validációs üzenet jelenik meg.

**Bővíthetőség, fejlesztői megjegyzések**

* A jogosultsági logika könnyen bővíthető további szűrőkkel (pl. kategóriák, címkék).
* A posztokhoz és kommentekhez like vagy reakció funkció is egyszerűen hozzáadható.
* A fórum minden komponense reszponzív, mobilon és asztali nézetben is jól működik.
* A moderációs funkciók (törlés) csak szerzőnek vagy adminnak elérhetők, minden törlés soft delete.

# 10. Tesztelési módszerek

## 10.1 Általános tesztelési stratégiák

A Casino projekt tesztelése során különböző stratégiákat alkalmazunk, hogy biztosítsuk a rendszer minden komponensének megfelelő működését. Az alábbiakban részletezzük a tesztelési folyamatokat, módszereket és eszközöket.

### 10.1.1 Tesztelési megközelítés

A tesztelés a következő megközelítésekkel történik:

* **Egységtesztek**: Az egyes funkciók (játékok, API végpontok) izolált tesztelése.
* **Integrációs tesztek**: Komponensek együttműködésének tesztelése (pl. játék és egyenlegkezelés).
* **Funkcionális tesztek**: Felhasználói esetek teljes folyamatainak tesztelése.
* **Manuális tesztek**: Strukturált tábblázatokban dokumentált, lépésenkénti vizuális ellenőrzés.

### 10.1.2 Tesztdokumentáció

A manuális tesztelés dokumentálására egységes formátumú tesztesetet használunk, amely tartalmazza:

* A teszt forgatókönyv és eset azonosítóját, leírását
* A teszt végrehajtási lépéseit részletesen
* Az elvárt és tényleges eredményeket
* A tesztek státuszát és a tesztelő megjegyzéseit

## 10.2 Tesztkörnyezet konfiguráció

### 10.2.1 Követelmények

A teszteléshez az alábbi környezetet használjuk:

* **Szerverkonfiguráció**: XAMPP 8.1 vagy újabb, PHP 8.1+, MySQL 8.0+
* **Böngészők**: Chrome (legfrissebb), Firefox (legfrissebb), Brave
* **Eszközök**: Desktop (Windows 10/11), mobil (Android 10+)

### 10.2.2 Teszt adatbázis beállítása

php artisan migrate:fresh --seed

### 10.2.3 Előfeltételek minden teszthez

* Működő teszt környezet
* Megfelelő teszt felhasználói fiókok (admin, normál felhasználó)
* Ellenőrzött kezdeti egyenlegek

## 10.3 Coinflip tesztelés

### 10.3.1 Coinflip játék alapvető funkcionalitás tesztje

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt forgatókönyv azonosító | Coinflip-1 | Teszt eset azonosító | Coinflip-1A |
| Teszt eset leírása | Coinflip játék indítása és játékmenet | **Teszt prioritás** | Magas |
| Előfeltétel | Bejelentkezett felhasználó, minimum 100$ egyenleg | **Utófeltétel** | N/A |

**Teszt végrehajtási lépések:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sorszám | Művelet | Bemenetek | Elvárt kimenet | Tényleges kimenet | Teszt böngésző | Teszt eredmény | Teszt megjegyzések |
| 1 | Coinflip oldal megnyitása | <http://localhost:8000/games/coinflip> | Coinflip játék felület betölt | Coinflip játék felület betölt | Chrome | Sikeres | [2025-04-27 20:00] Oldal sikeresen betöltve |
| 2 | Tét megadása | Tét: 50 | Tét érték beállítva 50$-ra | Tét érték beállítva 50$-ra | Chrome | Sikeres | [2025-04-27 20:01] Tét sikeresen beállítva |
| 3 | "Fej" választása | Kattintás a "Heads" gombra | Tét levonódik, érme animáció elindul | Tét levonódik, érme animáció elindul | Chrome | Sikeres | [2025-04-27 20:02] Játék elindult, tét levonva |
| 4 | Eredmény megjelenítés | - | Eredmény megjelenik (nyerés/vesztés), egyenleg frissül | Eredmény megjelenik, egyenleg frissül | Chrome | Sikeres | [2025-04-27 20:03] Eredmény helyesen jelent meg |

### 10.3.2 Coinflip hibakezelés tesztje

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt forgatókönyv azonosító | Coinflip-2 | Teszt eset azonosító | Coinflip-2A |
| Teszt eset leírása | Coinflip érvénytelen tét kezelése | **Teszt prioritás** | Közepes |
| Előfeltétel | Bejelentkezett felhasználó, 50$ egyenleg | **Utófeltétel** | N/A |

**Teszt végrehajtási lépések:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sorszám | Művelet | Bemenetek | Elvárt kimenet | Tényleges kimenet | Teszt böngésző | Teszt eredmény | Teszt megjegyzések |
| 1 | Coinflip oldal megnyitása | <http://localhost:8000/games/coinflip> | Coinflip játék felület betölt | Coinflip játék felület betölt | Firefox | Sikeres | [2025-04-27 20:10] Oldal sikeresen betöltve |
| 2 | Érvénytelen (magas) tét megadása | Tét: 100 | Hibaüzenet megjelenése: "Elégtelen egyenleg" | Hibaüzenet megjelenése: "Elégtelen egyenleg" | Firefox | Sikeres | [2025-04-27 20:11] Hibakezelés működik |
| 3 | Érvénytelen (negatív) tét megadása | Tét: -10 | Gomb inaktív vagy hibaüzenet | Gomb inaktív | Firefox | Sikeres | [2025-04-27 20:12] Negatív érték nem elfogadott |

## 10.4 Admin funkcionalitás tesztelés

**10.9.1 Hirdetés kezelés tesztje**

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt forgatókönyv azonosító | Admin-Ads-1 | Teszt eset azonosító | Admin-Ads-1A |
| Teszt eset leírása | Új hirdetés létrehozása | **Teszt prioritás** | Magas |
| Előfeltétel | Admin jogosultságú felhasználó | **Utófeltétel** | N/A |

**Teszt végrehajtási lépések:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sorszám | Művelet | Bemenetek | Elvárt kimenet | Tényleges kimenet | Teszt böngésző | Teszt eredmény | Teszt megjegyzések |
| 1 | Admin bejelentkezés | Email: tes.@example.com Jelszó: 123 | Sikeres bejelentkezés, admin vezérlőpult betölt | Sikeres bejelentkezés, admin vezérlőpult betölt | Chrome | Sikeres | [2025-04-27 20:30] Bejelentkezés sikeres |
| 2 | Hirdetéskezelő megnyitása | Kattintás a "Hirdetések kezelése" gombra | Hirdetéslista oldal betölt | Hirdetéslista oldal betölt | Chrome | Sikeres | [2025-04-27 20:31] Oldal betöltött meglévő hirdetésekkel |
| 3 | "Új hirdetés" gombra kattintás | - | Hirdetés létrehozási űrlap betölt | Hirdetés létrehozási űrlap betölt | Chrome | Sikeres | [2025-04-27 20:32] Űrlap helyesen jelenik meg |
| 4 | Hirdetés adatok kitöltése és elküldése | Cím: "Teszt hirdetés" Leírás: "Teszt leírás" Időtartam: 30 másodperc Jutalom: 50 Média típus: Kép Teszt kép feltöltése | Űrlap sikeresen elküldetett, átirányítás a listára, sikeres üzenet | Űrlap sikeresen elküldetett, átirányítás a listára, sikeres üzenet | Chrome | Sikeres | [2025-04-27 20:33] Hirdetés sikeresen létrehozva |

### 10.4.2 Felhasználó kezelés tesztje

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt forgatókönyv azonosító | Admin-User-1 | Teszt eset azonosító | Admin-User-1A |
| Teszt eset leírása | Felhasználó felfüggesztése | **Teszt prioritás** | Magas |
| Előfeltétel | Admin jogosultságú felhasználó, aktív felhasználói fiók | **Utófeltétel** | N/A |

**Teszt végrehajtási lépések:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sorszám | Művelet | Bemenetek | Elvárt kimenet | Tényleges kimenet | Teszt böngésző | Teszt eredmény | Teszt megjegyzések |
| 1 | Admin vezérlőpult megnyitása | <http://localhost:8000/admin> | Admin vezérlőpult betölt felhasználói táblázattal | Admin vezérlőpult betölt felhasználói táblázattal | Firefox | Sikeres | [2025-04-27 20:40] Vezérlőpult sikeresen betöltött |
| 2 | Célfelhasználó keresése | Görgetés a "Test" megtalálásához | Felhasználó megjelenik a táblázatban | Felhasználó megjelenik a táblázatban | Firefox | Sikeres | [2025-04-27 20:41] Felhasználó megtalálva a listában |
| 3 | "Felfüggesztés" gombra kattintás | Kattintás a "Felfüggesztés" gombra a "Test" sorában | Megerősítő ablak megjelenik | Megerősítő ablak megjelenik | Firefox | Sikeres | [2025-04-27 20:42] Megerősítés megjelent |
| 4 | Felfüggesztés megerősítése | Kattintás az "OK" gombra | Felhasználó státusza "Felfüggesztett"-re változik, sikeres üzenet megjelenik | Felhasználó státusza "Felfüggesztett"-re változik, sikeres üzenet megjelenik | Firefox | Sikeres | [2025-04-27 20:43] Felhasználó sikeresen felfüggesztve |

## 10.5 Fórum és komment funkció tesztelés

|  |  |  |  |
| --- | --- | --- | --- |
| Teszt forgatókönyv azonosító | Forum-Admin-1 | Teszt eset azonosító | Forum-Admin-1A |
| Teszt eset leírása | Fórum poszt törlése (admin) | **Teszt prioritás** | Közepes |
| Előfeltétel | Admin jogosultságú felhasználó, létező fórum poszt | **Utófeltétel** | N/A |

**Teszt végrehajtási lépések:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sorszám | Művelet | Bemenetek | Elvárt kimenet | Tényleges kimenet | Teszt böngésző | Teszt eredmény | Teszt megjegyzések |
| 1 | Fórum megnyitása | <http://localhost:8000/forum> | Fórum főoldal betölt posztokkal | Fórum főoldal betölt posztokkal | Brave | Sikeres | [2025-04-27 20:50] Fórum oldal betöltött |
| 2 | Célposzt megnyitása | Kattintás a "Teszt poszt" címére | Poszt részletező oldal betölt a tartalommal és hozzászólásokkal | Poszt részletező oldal betölt a tartalommal és hozzászólásokkal | Brave | Sikeres | [2025-04-27 20:51] Poszt részletek oldal megnyílt |
| 3 | "Poszt törlése" gombra kattintás | Kattintás a "Poszt törlése" gombra | Megerősítő ablak megjelenik | Megerősítő ablak megjelenik | Brave | Sikeres | [2025-04-27 20:52] Megerősítés megjelent |
| 4 | Törlés megerősítése | Kattintás az "OK" gombra | Átirányítás a fórum főoldalra, sikeres üzenet, poszt többé nem látható | Átirányítás a fórum főoldalra, sikeres üzenet, poszt többé nem látható | Brave | Sikeres |  |

# 11. Karbantartás és továbbfejlesztés

A Casino projekt hosszú távú fenntarthatósága és folyamatos fejlődése érdekében elengedhetetlen a rendszeres karbantartás, a hibák gyors javítása, valamint az új funkciók és játékok bevezetése. Ebben a fejezetben részletesen ismertetjük a karbantartási és továbbfejlesztési folyamatokat, javaslatokat, valamint a kapcsolódó szervezési és technikai feladatokat.

## 11.1 Rendszeres karbantartási feladatok

* **Biztonsági frissítések**
  + A szerveroldali (Laravel, PHP, Composer) és kliensoldali (Node.js, Electron, npm csomagok) függőségek rendszeres frissítése.
  + A kritikus biztonsági hibák mielőbbi javítása, különös tekintettel a külső csomagokra és API-kra.
* **Adatbázis karbantartás**
  + Rendszeres adatmentés (backup) és visszaállítási tesztek.
  + Adatbázis migrációk futtatása verzióváltáskor.
  + Elavult, felesleges adatok törlése (pl. soft delete rekordok véglegesítése).
* **Logok, naplók és monitoring**
  + Laravel logok, szerverlogok, hibajelentések rendszeres ellenőrzése.
  + Monitoring eszközök (pl. Sentry, NewRelic) bevezetése a hibák gyors felismeréséhez.
* **Automata és manuális tesztek futtatása**
  + Minden frissítés vagy új funkció bevezetése előtt a 10. fejezet szerinti tesztesetek végrehajtása.
  + Új hibák azonnali dokumentálása és javítása.
* **Dokumentáció frissítése**
  + A fejlesztési, telepítési és felhasználói dokumentáció naprakészen tartása minden jelentős változtatás után.
  + Verziószámok, rendszerkövetelmények, új funkciók és ismert hibák pontos vezetése.

## 11.2 Új funkciók és játékok bevezetése

* **Új játékmodulok hozzáadása**
  + Pl.: Póker, Baccarat, Lottó vagy egyéb kaszinó játékok.
  + Új játékok fejlesztésekor a 8–9. fejezet szerinti dokumentációs és fejlesztési sémát kell követni (layout, backend, frontend, API, tesztelés).
* **Közösségi és közösségépítő funkciók**
  + Barátlista bővítése, csoportos chat, versenyek, ranglisták fejlesztése.
  + Fórum, privát üzenetek, értesítések továbbfejlesztése.
* **Reklám- és jutalomrendszer fejlesztése**
  + Új hirdetési formátumok (pl. interaktív reklámok, mini-játékos hirdetések).
  + Dinamikus jutalmazási logika, achievement rendszer.
* **AI-alapú elemzések**
  + Játékos viselkedés elemzése, statisztikai riportok, csalásdetektálás.
  + Személyre szabott ajánlások, tippek, játékstílus-elemzés.

## 11.3 Technológiai továbbfejlesztés

* **Frissítések és migrációk**
  + Laravel, Electron, Node.js, PHP, MySQL verzióváltások támogatása.
  + Újabb frontend technológiák (pl. Vue 4, React 19, Tailwind új verziók) bevezetése.
* **Teljesítményoptimalizálás**
  + Kód refaktorálás, felesleges lekérdezések és duplikációk megszüntetése.
  + Asset optimalizálás (CSS/JS minifikáció, képek tömörítése).
  + Szerveroldali cache, CDN integráció.
* **Skálázhatóság és üzemeltetés**
  + Dockerizáció, CI/CD pipeline bevezetése.
  + Felhőalapú hoszting, horizontális skálázás előkészítése.

## 11.4 Felhasználói visszajelzések kezelése

* **Visszajelzési csatornák**
  + E-mail, fórum, in-app feedback, GitHub issue tracker.
* **Visszajelzések feldolgozása**
  + Rendszeres review meetingek, priorizált hibajegyek és feature-kérések.
  + Felhasználói igények alapján funkciók finomhangolása.

## 11.5 Jogszabályi és etikai megfelelés

* **Adatkezelési megfelelés**
  + GDPR és egyéb adatvédelmi szabályozások betartása.
  + Felhasználói adatok törlésének, exportálásának biztosítása.
* **Szerencsejáték szabályozás**
  + Bár a rendszer nem használ valós pénzt, a kaszinó-jelleg miatt a szerencsejáték szabályozások figyelemmel követése ajánlott.

## 11.6 Változáskezelés, verziózás

* **Verziókezelés**
  + Minden jelentős fejlesztés, javítás vagy új funkció külön branch-en történjen, a Git workflow szerint.
  + Verziószámok, changelog vezetése.
* **Változások dokumentálása**
  + Minden változást (bugfix, új funkció, refaktor) röviden dokumentálni kell a fejlesztői naplóban és a fő README-ben.

## 11.7 Karbantartási ütemterv, felelősségek

* **Rendszeres karbantartási ciklus**
  + Havonta legalább egyszer függőségfrissítés, negyedévente teljes körű audit.
* **Felelősök kijelölése**
  + Minden fő modulhoz (játék, közösségi funkciók, reklám, admin) legyen kijelölt felelős fejlesztő.
* **Események, hibák, fejlesztések naplózása**
  + Minden jelentős eseményt, hibát, frissítést naplózni kell (pl. CHANGELOG.md).

# 12. Kapcsolat

* **Fejlesztők**: Sitku Péter, Bilik Balázs, Krammer Erik.
* **Email**: peter.sitku@gmail.com
* **GitHub**: https://github.com/SitkuPeter/VizsgaRemek\_BB\_KE\_SP.git