

# **Debreceni SZC Beregszászi Pál Technikum**

## **Fogadási oldal**

Konzulens:

Beke Béla

Készítette:

Sőregi Dávid

Szoftverfejlesztő

Debrecen

2023

# Tartalomjegyzék

1	Bevezetés: .....	4
2	Felhasználói dokumentáció .....	5
2.1	Ismertető .....	5
2.1.1	Célközönség: .....	5
2.2	Program követelmény .....	5
2.2.1	XAMPP telepítése .....	5
2.2.2	A XAMPP és Moduljainak beállítása .....	7
2.2.3	A .NET telepítése .....	10
2.3	A weboldal .....	11
2.3.1	Bejelentkezés & regisztráció .....	11
2.3.2	Jelszó visszaállítás .....	13
2.3.3	A Főoldal .....	13
2.3.4	Pénztárca kezelése .....	14
2.3.5	Fogadás tétel .....	15
2.3.6	Fogadások oldal .....	16
2.3.7	Profil oldal .....	17
2.3.8	Adatok mentése .....	17
3	Fejlesztői dokumentáció .....	18
3.1	Alkalmazott fejlesztői eszközök .....	18
3.1.1	Visual Studio .....	18
3.1.2	Xampp .....	19
3.1.3	Visual Studio Code .....	20
3.2	Adatbázis .....	21
3.2.1	Táblák összekapcsolásai .....	23
3.3	Adatmodell .....	24
3.3.1	Be/ki fizetés tábla .....	24
3.3.2	Felhasználói adatok tábla .....	25
3.3.3	Fogadások tábla .....	26
3.3.4	Fogadási lehetőségek tábla .....	27
3.3.5	Nemzetek tábla .....	28
3.3.6	Játékosok tábla .....	28
3.3.7	Meccs eredmények tábla .....	29
3.3.8	Pénztárca tábla .....	30
3.4	Részletes feladat specifikáció, algoritmusok .....	31

3.4.1	C# program felépítése .....	31
3.4.2	A weblap felépítése .....	39
3.5	Tesztelési dokumentáció .....	50
3.5.1	Tesztelés és eredmények .....	51
3.6	Továbbfejlesztési lehetőségek .....	52
4	Összegzés.....	52
5	Irodalomjegyzék .....	52

# 1 Bevezetés:

A választott témaköröm egy weboldal, amely igazából egy fogadó oldal. Amely oldalon a felhasználó tud fogadni virtuális foci meccsekre nem valódi pénzzel, hanem úgynevezett az oldal által használt játékpénzzel. A fogadás menete egyszerű. Több lehetőség áll rendelkezésre a felhasználónak, amelyre tud fogadni, avagy játékpénzt felhelyezni. Ezek a lehetőségek lehetnek a gólszámra fogadás, a mérkőzés nyertesére való fogadás, a félidőre való fogadás és ezek mellett opcionális lehetőség lehet a lapokra való fogadás vagy a gólt szerző játékos megtippelése. Az utóbbi kettőt csak plusz dolognak szeretném feltüntetni mivel még nem tudom, hogy sikerül-e időben az alap elképzelést megvalósítani. A csapatokat adatbázisban szeretném kezelni és a hozzájuk tartozó adatokat és a meccsek adatait is. A meccseket egy külön álló c# alkalmazásban szeretném lefuttatni. Az itt kapott értékeket az adatbázisban tárolni és innen dolgozna velük az oldal. Az online fogadó oldalak egyik lehetősége a virtuális foci fogadás. Ez egy olyan virtuális sport, amely lehetővé teszi a felhasználók számára, hogy valós időben fogadjanak a virtuális csapatok eredményére. A virtuális foci hasonló a valódi focira, de a játékosok és a csapatok virtuálisak, és az eredményeket számítógép generálja. A virtuális foci fogadás lehetősége a fogadó oldalakon keresztül egyre népszerűbbé válik, és számos fogadási lehetőséget kínál a felhasználóknak. A virtuális foci fogadások lehetnek előzetes fogadások, élő fogadások és kombinált fogadások, és a felhasználók különböző fogadási lehetőségeket választhatnak, mint például a győzelem, a gólok száma, a sárga lapok száma stb. A virtuális foci fogadások nagyszerű lehetőséget kínálnak a futbalszeretőknak, hogy élvezhessék a játékot és egyben pénzt kereshessenek is.

## 2 Felhasználói dokumentáció

### 2.1 Ismertető

A virtuális fogadó oldalakon lehetőség van a virtuális foci fogadásra. A felhasználók valós időben fogadhatnak a virtuális csapatok eredményére, és különböző fogadási lehetőségeket választhatnak, mint például a győzelem, a gólok száma. A virtuális foci fogadás lehetőségei széles körűek, és tartalmazzák az előzetes fogadásokat, az élő fogadásokat. A felhasználók a fogadó oldalakon keresztül figyelemmel kísérhetik a fogadásaikat és a virtuális sporteseményeket, és számos fizetési és kifizetési lehetőséget is használhatnak. Összességében a virtuális fogadó oldalak széles körű sportfogadási lehetőségeket kínálnak a felhasználók számára, és lehetővé teszik, hogy élvezhessék a virtuális foci játékát és egyben pénzt is kereshessenek. A fogadó oldalon ezeket az opciókat érhetjük el. A felhasználó be tud regisztrálni a weboldalra majd képes, ha akar egy bizonyos pénzösszeget feltölteni a profiljához tartozó pénztárcájához. Később a feltöltött pénzből képes fogadni az éppen megjelenő mérkőzésre, amelyek automatikusan váltják egymást. 20 ország csapata közül sorsolódnak ki a meccsek szóval változóak a csapatok így sokfajta mérkőzés születik, amelyeknek az alapja, hogy az algoritmus az előző meccsek alapján sorsolja a meccseket így életszerűbb a fogadás.

#### 2.1.1 Célközönség:

A záródolgozatomat olyan személyeknek ajánlanám, akik szeretik a fogadást és a szerencsejátékokat vagy a labdarúgást esetleg a virtuális labdarúgást. Esetleg olyan személyeknek, akik már régebb óta foglalkoznak fogadással vagy virtuális fogadással.

## 2.2 Program követelmény

Mivel az oldal localhost-on fut így elengedhetetlen az oldal működéséhez a XAMPP alkalmazás és a Microsoft .NET telepítése.

### 2.2.1 XAMPP telepítése

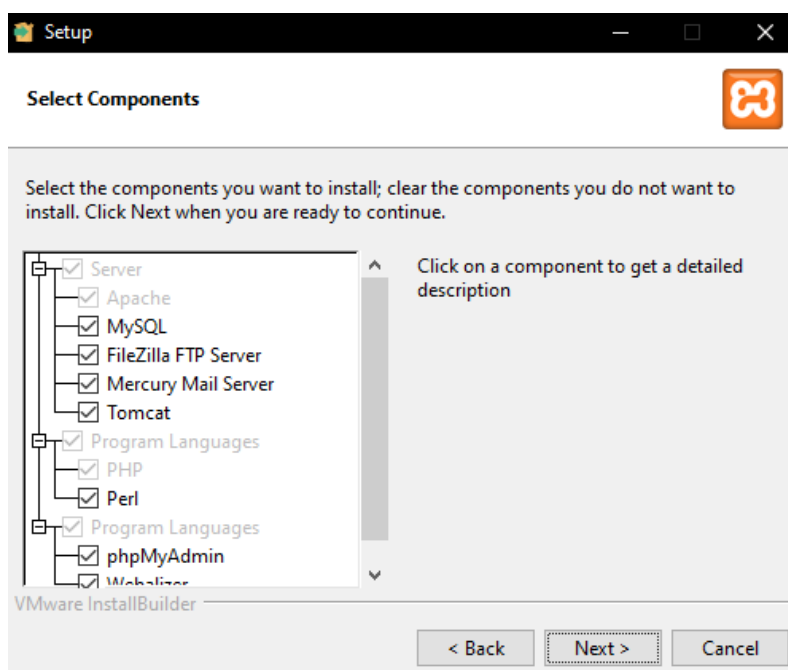
Az XAMPP-al könnyen létrehozhatunk és tesztelhetünk weboldalakat a "htdocs" mappában. Ebben a mappában találjuk az "index.php" fájlt, amely az alapértelmezett fájl az XAMPP-ban. Az XAMPP telepítése és használata egyszerű, és lehetővé teszi a fejlesztők és a webmesterek számára, hogy kényelmesen és hatékonyan teszteljék és fejlesszék webalkalmazásaikat. Az XAMPP egy ingyenes, nyílt forráskódú szoftver, amellyel lehetővé válik egy teljes webalkalmazás-környezet telepítése és futtatása egyetlen gépen. Az XAMPP segítségével

könnyen lehet tesztelni és fejleszteni weboldalakat és webalkalmazásokat, anélkül, hogy szükség lenne egy teljes szerver-konfigurációra. Az XAMPP telepítése könnyű és egyszerű. Az alábbi lépések segítségével telepíthető az XAMPP:

1. Látogassunk el az Apache Friends (<https://www.apachefriends.org>) hivatalos weboldalára, és töltsük le az XAMPP legfrissebb verzióját a letöltési oldalon.

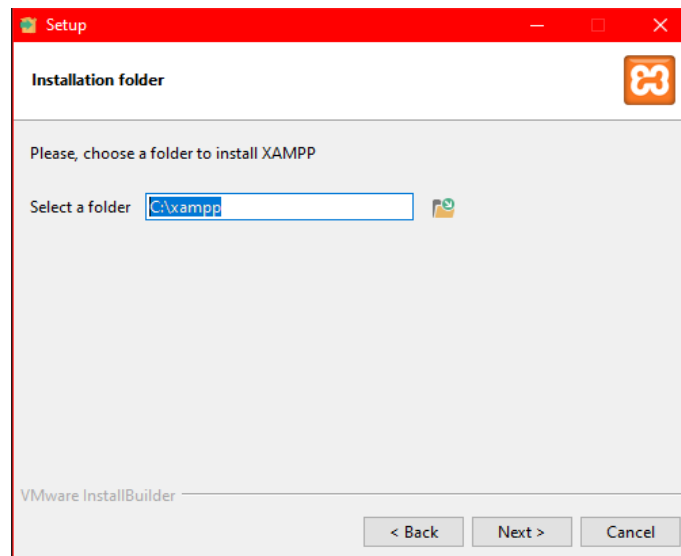


2. A letöltés után indítsuk el a telepítőfájlt, majd válasszuk ki, amit telepíteni szeretnénk.

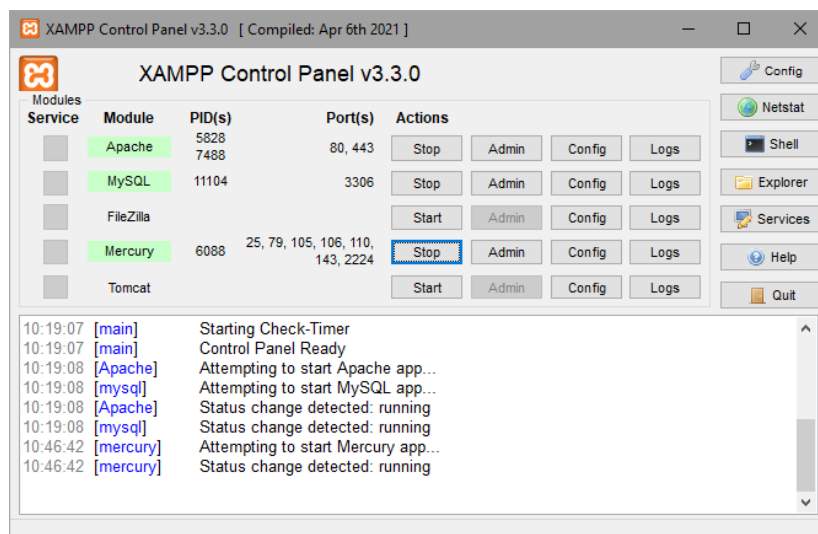


Ha megvan lépünk a Next-gombra.

3. Majd válasszuk ki a telepítési helyet. Az alapértelmezett telepítési hely a "C:\xampp" mappába történik.



4. A telepítő befejezése után indítsuk el az XAMPP Control Panelt, amely lehetővé teszi a szerverkomponensek (Apache, MySQL, PHP stb.) indítását és leállítását. Indítsuk el a APACHE és MySQL Modulokat. A fogadó oldal és a webszerver működéséhez.

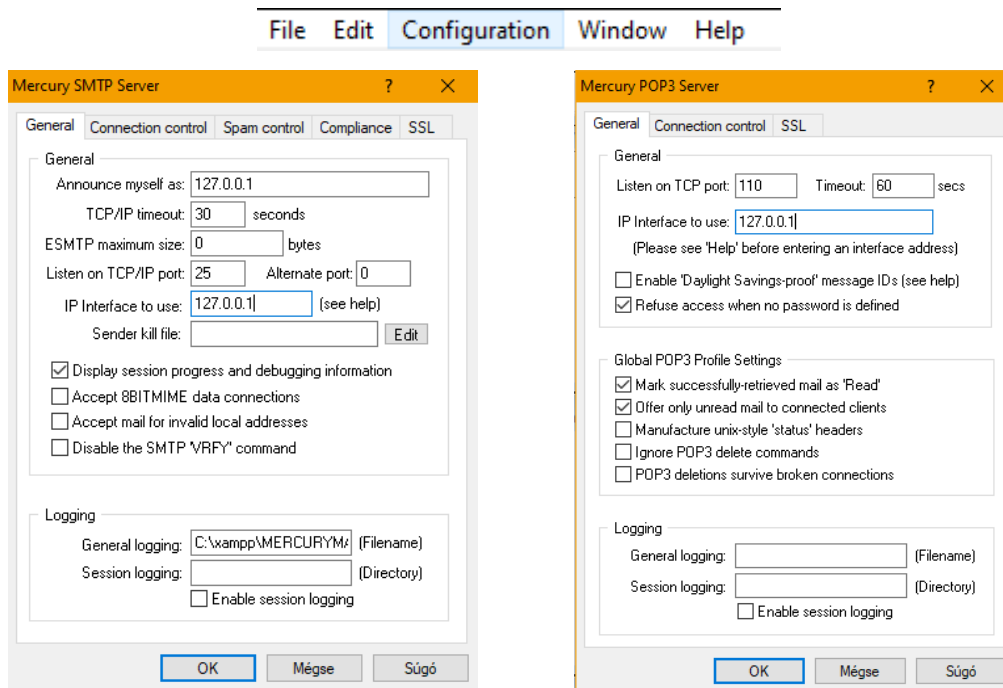


## 2.2.2 A XAMPP és Moduljainak beállítása

A Mercury Modult is elindíthatjuk, ez a jelszó visszaállítás bemutatásához lesz szükséges localhost-on. Csak localhost-on működik.

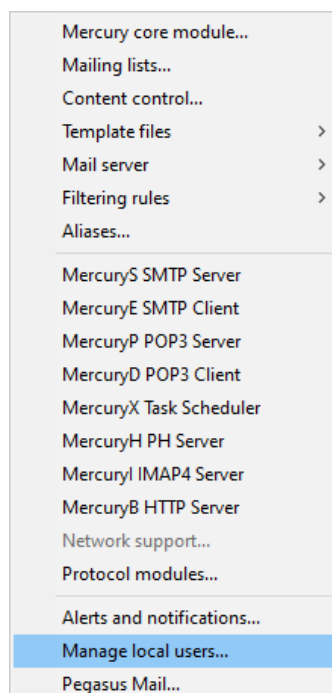
1. Első lépésnek kattintsunk a Mercury modulnak az Admin részére majd a bal felső sarokban látható opciók közül válassza ki a „Configuration” elemet.

„Configuration” elemen belül ki kell választani a „Mercury SMTP Server” -t és a „Mercury POP3 Server” -t.



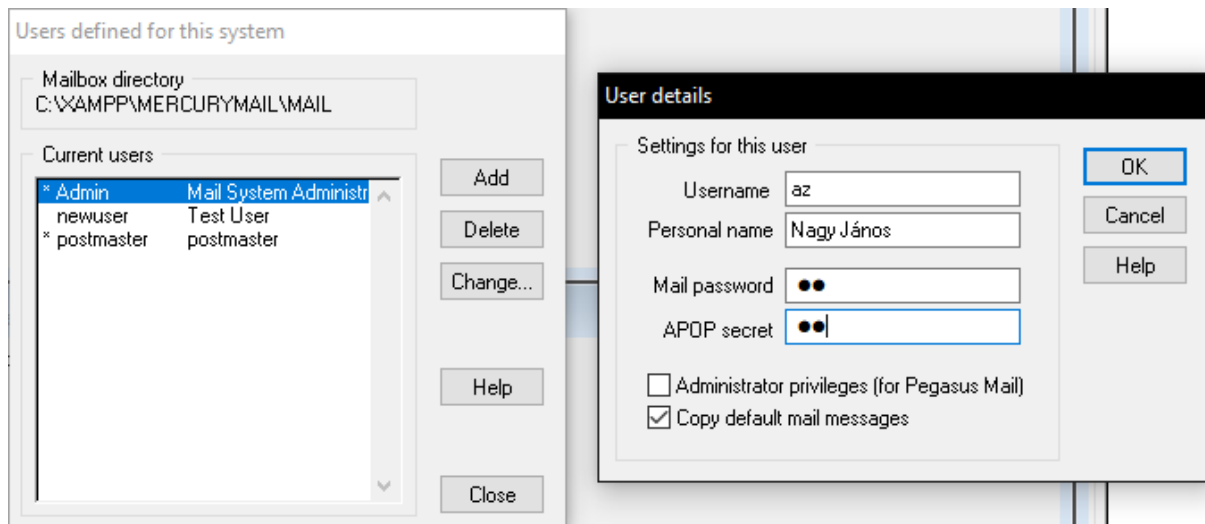
Ezt követően ahogy a képen is látható, ahol szerepel a „127.0.0.1” IP cím oda kell beírni a „127.0.0.1” IP címet.

2. Majd ezt követően megint nyissuk le a „Configuration” elemet. Ezen belül a „Manage local users” -t.

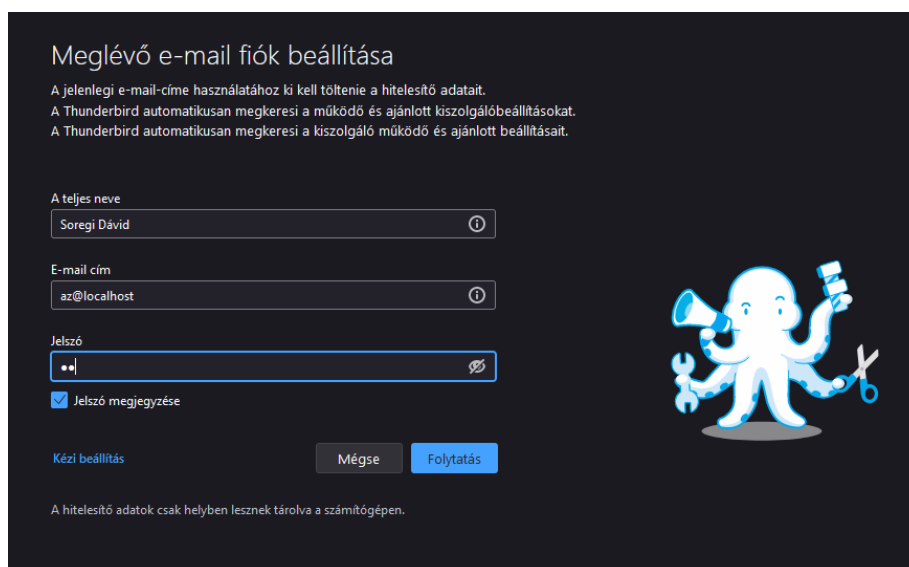




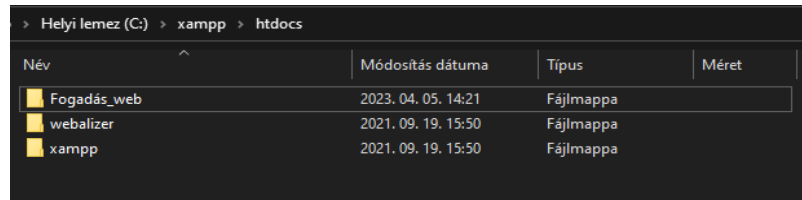
Ezután az „Add gombra” kattintva hozzáadunk egy localhost user-t. A „User details” -on belül „Username” -hez megadjuk annak az email-címnek az elejét, amivel majd regisztrálni akarunk a weboldalra. „Personal name” -nek megadhatunk bármit, amit szeretnénk. Az utolsó kettő mezőbe meg a jelszaunkat adjuk meg a localhost-on használó email-címhez.



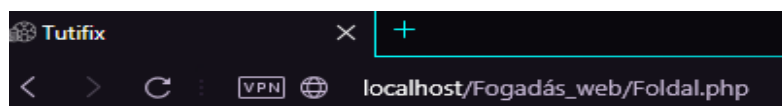
3. A legegyszerűbb program a localhost-on létező email-címünk beállítása a „ThunderBird” nevű alkalmazás. Töltsük le és telepítsük fel. Ezután indítsuk el és lépünk be a már megadott email-címmel. Figyeljünk arra, hogy az email-címben „@localhost” adjunk meg mivel ez localhost-on szimulálja a jelszó vissza állítást.



4. Ha elindítottuk a megfelelő Modulokat akkor nyissuk meg egy Fájlkezelőt majd az elérési útban adjuk meg a XAMPP-nak ezt a mappáját „C:\xampp\htdocs” és a weboldalnak a mappáját másoljuk be ide. Ezután becsukhatjuk az ablakot.



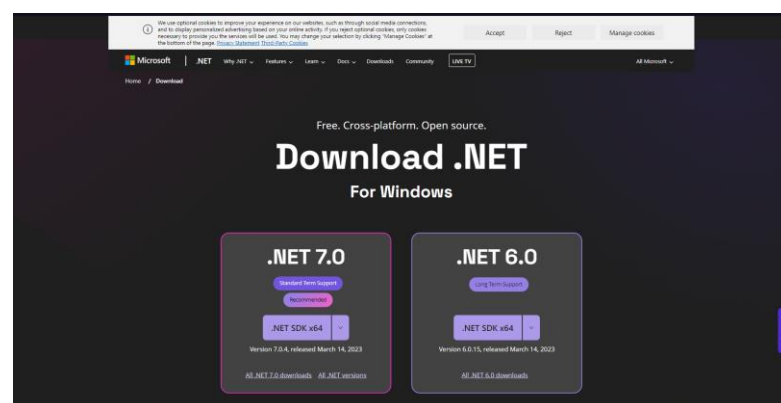
5. Ezután az általunk választott böngészőbe írjuk be „localhost/fogadas\_zarouj/Foldal.php”, hogy elérjük a weboldalt.



### 2.2.3 A .NET telepítése

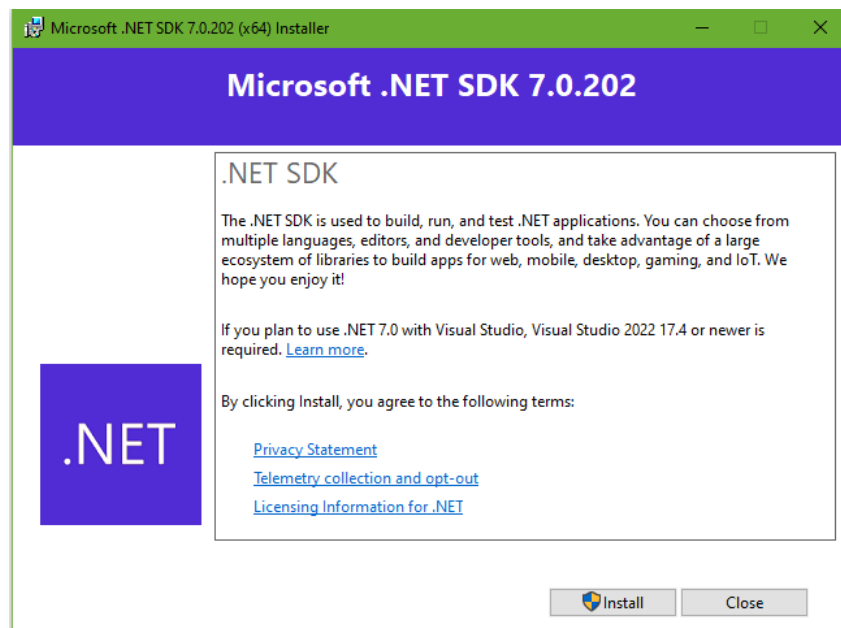
Az oldal működéséhez szükséges a #C program, mivel itt sorsolódnak ki a mérkőzések és itt generálják az algoritmusok le az adott mérkőzések eredményeit is és innen kerülnek be az adatbázisba. Ehhez is szükséges volt a XAMPP alkalmazás. Ahhoz, hogy a felhasználó letudja futtatni a #C programot szüksége lesz a Microsoft .NET-re. Az alábbi lépések segítségével telepíthető az .NET:

1. Látogassunk el a Microsoft (<https://dotnet.microsoft.com/en-us/download>) hivatalos weboldalára, és töltsük le az .NET legfrissebb verzióját a letöltési oldalon.

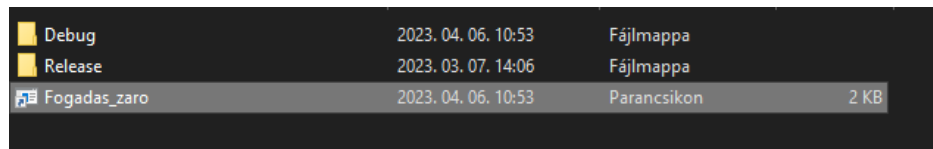


Válasszuk ki a nekünk megfelelőt.

2. A letöltés után indítsuk el a telepítőfájlt, és telepítsük fel a .NET-et.



3. Ezután nyissuk meg a C# programnak mappáját és indítsuk el a „Fogadas\_zaro” -t a program működéséhez és a weboldal megfelelő működéséhez.



## 2.3 A weboldal

Indítsuk el a XAMPP-ot és a megfelelő Modulokat és a #C programot „Fogadas\_zaro” -ot a weboldal működéséhez. Ezután menjünk fel a weboldalra írjuk be a böngészőbe: „localhost/fogadas\_zarouj/Foldal.php”.

### 2.3.1 Bejelentkezés & regisztráció

A weblap betöltése után a főoldalon találjuk magunkat, ahol rendelkezésünkre áll kettő elem az oldal tetején, amivel tudunk a weboldalra regisztrálni vagy bejelentkezni.



Ha a regisztráció gombra kattintunk akkor megjelenik a regisztrációs oldal:

Tutifix fogadó oldal

Fő oldal Regisztráció Bejelentkezés

Felhasználó név:  
Felhasználónév

Email-cím:  
Email-cím

Jelszó:  
Jelszó

Okmány Száma:  
Azonosító szám

Nemem: ☐ Férfi ☐ Nő

Ország:  
Ország

Regisztráció

Van már van fiókod? Jelentkezz be!

Itt tud a felhasználó egy fiókot létrehozni magának, amivel majd később képes lesz bejelentkezni az oldalra. Az általa megadott adatokat az adatbázis eltárolja. A felhasználó megtud adni egy felhasználó nevet, egy email-címet, a fiókja biztonságos eléréséhez egy jelszót, az okmányának a számát, a nemét és esetleg azt, hogy melyik országból való, de ez nem kötelező csak opcionális. Ha minden szükséges adatot megadott a felhasználó akkor a regisztráció gombra kattintva sikeres regisztrációt tud végre hajtani, amit egy felugró ablak jelezni is fog. Ezután, ha sikeres volt a regisztráció akkor egyből átdobja a bejelentkezési oldalra. Ha valamilyen hiba lépne fel például az email-cím vagy a felhasználó név már foglalt lenne akkor azt is jelezni fogja az oldal a felhasználónak. Ha a felhasználó már rendelkezik egy fiókkal akkor kitudja választani azt az opciót, hogy bejelentkezik. Ezt úgy tudja megtenni, hogy a jobb felső sarokban rákattint a „Bejelentkezés” -gombra vagy a „Regisztráció” -gomb alatt talál egy linket „Jelentkezz be!” felirattal. Bármelyiket választja átfogja dobni a bejelentkezési oldalra.

Tutifix fogadó oldal

Fő oldal Regisztráció Bejelentkezés

Bejelentkezés

Email-cím:  
Email-cím

Jelszó:  
Jelszó

Jelszó visszaállítás

Bejelentkezés

© 2023 Sőregi Dávid fogadó oldala

Itt a felhasználó megtudja adni az előzőleg már regisztrált email-címet és a hozzá tartozó jelszót. Ha mindent jól adott meg akkor az oldal befogja tölteni neki a főoldalt, ahonnan most már minden opció elérhető, amelyek bejelentkezés nélkül nem látszódnak. Ha valamit nem jól adott meg a felhasználó akkor ezt egy felugró ablak fogja jelezni. Ebben az esetben a felhasználónak meg kell adnia a megfelelő adatokat, hogy sikeres legyen a bejelentkezés.

### 2.3.2 Jelszó visszaállítás

Ez, ha nem sikerül akkor van egy olyan opció, hogy visszaállítja a jelszavát. Ehhez rákattint „Jelszó visszaállítás” szövegre. Ebben az esetben egy másik oldalon fogja találni magát a felhasználó, ahol meg kell adnia a fiókjához tartozó email-címet.

Erre az email-címre ki fog küldeni egy jelszó visszaállításról szóló email-t az oldal, amiben egy linkre kattintva vissza fogja dobni az weboldalra. Az email-t a „ThunderBird” alkalmazáson belül találja, ha azt használja a felhasználó és ha beállította a „Mercury” -t megfelelően. Ha visszadobta a felhasználót az oldalra akkor megtudja változtatni a jelszavát. Ha sikeres volt a jelszó változtatás akkor az oldal vissza fogja dobni a bejelentkezési oldalra. Ezek után, ha mindent jól adott meg akkor befogja dobni a weboldalra.

### 2.3.3 A Főoldal

A főoldalon így, hogy most már be van jelentkezve a felhasználó így minden opció megjelenik a weboldalon. Bal oldali részen láthatjuk az épen futó vagy már lefutott meccsek eredményeit és adatait például az országok neve és a játékosok neveit. Jobb oldalon láthatjuk az ezekhez a meccsekhez tartozó fogadási lehetőségeket és az ezekhez tartozó szorzókat. A felhasználó láthatja, a menü részben, hogy mennyi pénze van és a felhasználó nevét.

Ha szeretne a felhasználó kijelentkezni akkor azt a „Kijelentkezés” gombra kattintva érheti el.

**Tutifix fogadó oldal**

Fő oldal Fogadásaim Kijelentkezés Pénztárca BOOFT az Profil

4	Eredmények	0
<b>Svédország</b>	<b>Hollandia</b>	
<b>Játékosok:</b>	<b>Gólszerzők:</b>	<b>Játékosok:</b>
Robin Olsen	Mattias Svanberg	Jasper Cillessen
Linus Wahlqvist	Samuel Gustafson	Nathan Aké
Victor Lindelöf	Samuel Gustafson	Virgil van Dijk
Hjalmar Ekdal	Samuel Gustafson	Lutsharel Geertruida
Ludwig Augustinsson		Jurrien Timber
Mattias Svanberg		Kenneth Taylor
Mattias Svanberg		Marten de Roon
Emil Forsberg		Georginio Wijnaldum
Samuel Gustafson		Xavi Simons
Dejan Kulusevski		Memphis Depay
Alexander Isak		Steven Berghuis

**Szorzők**

Végeredmény:

Hazai - 16	Döntetlen - 3.73	Vendég - 4
Hazai Gólszám:	Meccs gólszáma:	Vendég Gólszám:
Hazai +15 Gól - 2.24	+15 Gól - 2.02	Vendég +15 Gól - 5.6
Hazai +25 Gól - 2.88	+25 Gól - 2.87	Vendég +25 Gól - 7.2
Hazai +35 Gól - 4	+35 Gól - 3.71	Vendég +35 Gól - 10
Hazai -35 Gól - 6	+5 Gól - 10.8	Vendég -35 Gól - 2.4

## 2.3.4 Pénztárca kezelése

Mielőtt tovább mennénk a fogadásra elsőnek kell pénzt feltöltenünk az oldalra, hogy a felhasználó képes legyen fogadni a meccsekre. Ha már nyertünk egy bizonyos összeget és szeretnénk ezt a pénzt kiutalni azt is ebben a fülben tudjuk megtenni.

**Tutifix fogadó oldal**

Fő oldal Fogadásaim Kijelentkezés Pénztárca BOOFT az Profil

KÁRTYA SZÁM  
1234 5678 9101 1121

KÁRTYA TULAJDONOS  
Nagy János

ÖSSZEG  
[Input field]

LEJÁRATI DÁTUM  
02/40

CVV  
123

Pénz befizetés

Pénz kifizetés

© 2023 Sőregi Dávid fogadó oldala.

Ezen az oldalon a felhasználónak meg kell adnia a Kártyaszámát, a kártyájának a tulajdonosának nevét a kívánt összeget, amit szeretne be vagy kifizetni, a lejárat dátumot és a CVV-t. Ez az oldal szimulálja a felhasználó részére egy tranzakciót. Ha megadott minden adatot a felhasználó akkor eldöntheti, hogy be szeretné a kívánt összeget fizetni vagy ki. Ezt azzal

választhatja ki, hogy melyik gomra kattint. Az első gomb „Pénz befizetés”, ha erre kattint beviszi a kívánt az összeget, ha a második gombra „Pénz kifizetés” -re kattint akkor kifizeti a kívánt összeget. Ennek is megvannak a határai. Befizetéskor a minimum összeg 1500Ft a maximum 2.000.000Ft. A kifizetésnél 10.000Ft a minimum a maximum 2.000.000Ft. A lejárat dátum ellenőrizve van, a kártya számnak maximum 16 értéket tud megadni a felhasználó. Ha valamit nem jól adna meg akkor azt az oldal jelezni fogja a felhasználónak és akkor nem fog teljesülni a tranzakció. Ha sikerült feltölteni pénzt az oldalra a felhasználónak akkor mehet fogadni.

### 2.3.5 Fogadás tétel

Amikor megjelenik az új meccs akkor tud a felhasználó fogadni rá. Választani tud a lehetőségek közül, amelyek adottak. Fogadni tud a végeredményre, hogy ki nyert a mérkőzésen, a mérkőzés gólszámára vagy arra, hogy külön-külön a hazai és vendégcsapat hány gólt fog szerezni a mérkőzésen. Itt a felhasználó láthatja az adott fogadáshoz lévő szorzót is. Ennek fényében döntheti el mire akar fogadni.

Szorzók		
Végeredmény:		
Hazai - 2	Döntetlen - 2,67	Vendég - 2
Hazai Gólszám:	Meccs gólszáma:	Vendég Gólszám:
Hazai +15 Gól - 2,8	+15 Gól - 2,13	Vendég +15 Gól - 2,8
Hazai +25 Gól - 3,6	+25 Gól - 2,95	Vendég +25 Gól - 3,6
Hazai +35 Gól - 5	+35 Gól - 3,77	Vendég +35 Gól - 5
Hazai +35 Gól - 3	+5 Gól - 10,64	Vendég +35 Gól - 3

Ha megvan melyikre szeretne fogadni ezek közül a felhasználó kattintással kiválasztja azt. Ennek hatására felugrik egy ablak, ahol képes lesz végbe vinni a fogadást. Itt látja a fogadás adatait. A fogadás nevét, mekkora szorzóra fog fogadni és itt tudja bevinni a kívánt összeget mennyit szeretne felrakni a fogadásra. Ha sikeres volt a fogadás megtétele akkor az oldal átfogja dobni a „Fogadásaim” nevű oldalra, ahol láthatja, az előző fogadásait és azt is, amit éppen megfogadott. Az oldal automatikusan ellenőrzi, hogy a felhasználó fogadása nyertes vagy vesztes lett. Ha nyert bővülni fog a pénztárcájának az értéke. Amint megrakja a fogadást levonja a pénztárcájából az összeget. Ha mégsem szeretné megtenni a fogadást szimplán az „X” -el jelzet gombra kattintva becsukja az ablakot vagy bárhová kattint az oldalon. Ekkor befog csukódní az ablak.

Fontos megjegyezni, hogy lehet egy meccsre több fogadást is megtenni, de mindegyiket egyesével kell megtennie a felhasználónak.

Hazai (1.6x)

Válassz összeget:

Küldés
X

### 2.3.6 Fogadások oldal

Ezen az oldalon tudja a felhasználó megtekinteni, hogy az előző fogadásai nyertesek vagy vesztesek lettek-e. Éppen, ha van egy fogadása, ami még nem futott le akkor azt bal oldalt az „Futó fogadásaid” táblázatban láthatja, ha nincs futó fogadása akkor üres a táblázat. Jobb oldalt láthatja a felhasználó az utolsó 10 db már le futott előző fogadásait a „Lefutott fogadásaid” nevű oszlopban.

Tutifix fogadó oldal

[Fő oldal](#)
[Fogadásaim](#)
[Kijelentkezés](#)
[Pénztárca](#)
2940Ft
[az](#)
[Profil](#)

Fogadásaid

Futó fogadásaid

Csapatok	Tét	Profit/Bukó	Szoró	Fogadás neve
Dánia — Csehország	1000	0	2	Vendég
Dánia — Csehország	1000	0	3	Döntetlen
Dánia — Csehország	1000	0	25	Hazai

Lefutott fogadásaid

Csapatok	Tét	profit/bukó	Szoró	Fogadás neve	Eredmény
Ukraina — Portugália	1000	-1000	2.25	Vendég	Vesztes
Ukraina — Portugália	1000	2940	2.94	Döntetlen	Nyertes
Ukraina — Portugália	1000	-1000	2.16	Hazai	Vesztes
Ukraina — Szerbia	1000	-1000	2.33	Vendég	Vesztes
Ukraina — Szerbia	1000	-1000	2.95	Döntetlen	Vesztes
Ukraina — Szerbia	1000	2000	2.1	Hazai	Nyertes
Hollandország — Dánia	1000	-1000	1.6	Vendég	Vesztes
Hollandország — Dánia	1000	-1000	3.73	Döntetlen	Vesztes
Hollandország — Dánia	1000	4000	4	Hazai	Nyertes
Portugália — Dánia	1000	3000	3	Vendég	Nyertes

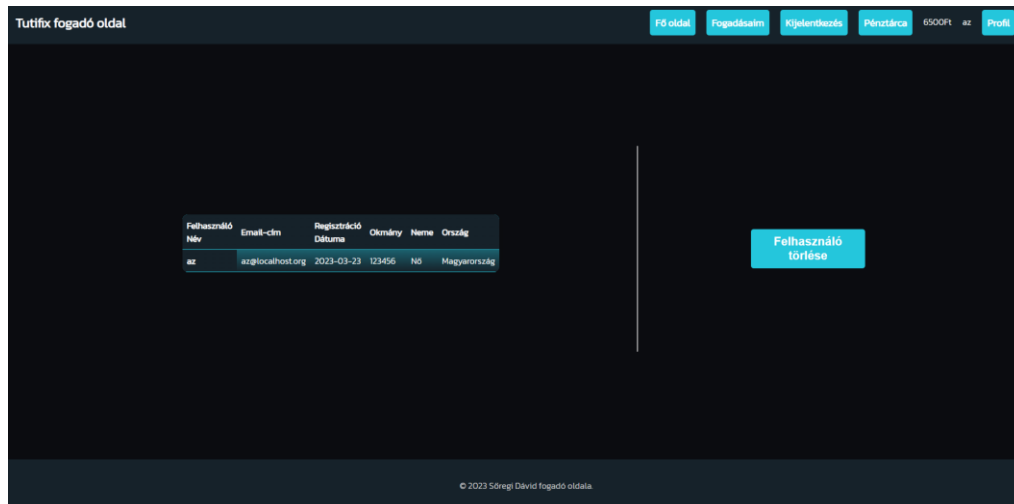
© 2023 Sőregi Dávid fogadó oldala.

A táblázatokban látható a csapatok nevei olyan sorrendben, hogy az első mindig a hazai csapat a második mindig a vendégcsapat. Ezután a felhasználó láthatja a tétet mennyivel fogadott az adott mérkőzésre. Tovább haladva láthatja a felhasználó mennyit nyert vagy bukott az adott fogadáson, a szorzót is és a fogadásnak a nevét. A jobb oldali táblázatban láthatja azt is, hogy a lefutott fogadásai nyertesek voltak-e vagy vesztesek.



### 2.3.7 Profil oldal

A fő oldalról tudja elérni a felhasználó ezt az oldalt. Ezen az oldalon láthatja a felhasználó milyen adatokat adott meg regisztrációkor. Láthatja a felhasználói nevét, az email-címét, regisztráció dátumját, okmány számát, nemét és az országát. Ezen az oldalon képes törölni a felhasználó a fiókját, ha szeretné. A „Felhasználó törlése” gombra kattintva érheti ezt el.



### 2.3.8 Adatok mentése

Minden adat, amivel dolgozik a weboldal adatbázisban van eltárolva és kezelve nincs olyan adat, ami nem lenne eltárolva és nem látna a felhasználó. A felhasználó adatok visszatérnek az oldalra ezt is adatbázis kezeli.

## 3 Fejlesztői dokumentáció

### 3.1 Alkalmazott fejlesztői eszközök

Az integrált fejlesztői környezet, vagyis rövidítve IDE (Integrated Development Environment) egy olyan egy olyan szoftver, amelyben a megírni kívánt forráskódot lehet szerkeszteni. Az IDE nem csak a forráskód szerkesztési lehetőségét adja, hanem számos beépített funkciót tartalmaz. Főbb eszközei, funkciói:

- Szintaxis kiemelése: Az IDE-szerkesztő általában szintaktikai kiemelést biztosít, ilyenkor mind a vizuálisan különböző színekkel és betű-hatásokkal rendelkező szintaktikai hibákat képes megjeleníteni.
- Kód kiegészítés: a kódkiegészítés egy fontos IDE funkció, amelynek célja a programozás felgyorsítása. A modern IDE-knek már van intelligens kód befejezése is. Az intelligens kódkiegészítése egy környezet tudatos funkció, amely felgyorsítja az alkalmazások kódolási folyamatát az elírások és más gyakori hibák csökkentésével. Erre általában gépelés közbeni automatikus kiegészítés felugró ablakon keresztül történik.
- Hibakeresés: Az IDEs-t hibakeresésre is használják, az integrált hibakeresőt használva, támogatva a töréspontok beállítását, amely a kód adott részénél megállítja a programot így megtudhatjuk hol jelenetezik a hiba vagy mi okozza azt, illetve nyomon követhetjük a változók értékeit is.
- Vizuális programozás: A Visual Basic lehetővé teszi a felhasználók számára, hogy új alkalmazásokat hozzanak létre programozás, építőelemek vagy kód csomópontok áthelyezésével, hogy folyamatábrákat vagy szerkezet diagramokat hozzon létre, amelyeket aztán lefordítanak vagy értelmeznek.

#### 3.1.1 Visual Studio

A Visual Studio a Microsoft integrált fejlesztői környezete (IDE). Számítógépes programok, valamint webhelyek, webes alkalmazások, mobil alkalmazások fejlesztésére szolgál. A Visual Studio 36 különböző programozási nyelvet támogat, és lehetővé teszi, hogy a kódszerkesztő és a hibakereső támogassa szinte bármilyen programozási nyelvet, feltéve, hogy létezik nyelvspecifikus szolgáltatás.

A Visual Studio-nak a legnagyobb változást a .NET-keretrendszert használó felügyelt kódfejlesztő környezet bevezetése jelentette. A .NET használatával fejlesztett programokat nem gépi nyelvre fordítják (mint például a C++), hanem a Microsoft Intermediate Language

(MSIL) vagy a Common Intermediate Language (CIL) formátumra. Amikor egy CIL-alkalmazás fut, akkor az lefordítása közben a végrehajtás alatt álló platformnak megfelelő gépi nyelvre kerül, ezáltal a kód több platformon is hordozhatóvá válik. A CIL-be fordított programok csak olyan platformokon futtathatók, amelyek rendelkeznek Common Language Infrastructure implementációval.

A .NET keretrendszer azon részét, amely elvégzi a konverziót köztes kód és a gépkód között, közös futtató környezetnek Common Language Runtime (CLR) nevezzük. Ennek a megoldásnak az igazi ereje abban rejlik, hogy a .NET rendszerben támogatott összes nyelv először MSIL-re fordul. Így a CLR számára teljesen mindegy, hogy a kód, amelyet lefordít, eredetileg Visual Basic .NET, J#, vagy esetleg C# nyelven volt megírva. Az IL tehát elfedi a .NET által támogatott nyelvek közötti különbségeket is.<sup>1</sup>

A C# a Visual Basic mellett a .NET fő programozási nyelve. 1999 –ben Anders Hejlsberg vezetésével kezdték meg a fejlesztését. A C# tisztán objektumorientált, típusbiztos, általános felhasználású nyelv. A tervezésénél a lehető legnagyobb produktivitás elérését tartották szem előtt. A nyelv elméletileg platform független (létezik Linux és Mac fordító is), de napjainkban a legnagyobb hatékonyságot a Microsoft implementációja biztosítja.

Így a választásom a Visual Studio-ra esett mivel, minden megtalálható benne a számomra könnyű munkavégzésre. A meccsek sorsolása és meccs eredmények sorsolása is itt történik a fogadások ellenőrzése is és végül ezeknek az adatoknak a kiírtatása is itt történik.

### 3.1.2 Xampp

A XAMPP egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB (korábban a MySQL) adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői (végrehajtó rendszerei). Ez a szoftvercsomag egy integrált rendszert alkot, amely webes alkalmazások készítését, tesztelését és futtatását célozza, és ehhez egy csomagban minden szükséges összetevőt tartalmaz. A rendszer egyik nagy előnye az összehangolt elemek könnyű telepíthetősége.

A XAMPP egyetlen tömörített (zip, tar, 7z vagy exe formátumú) állományba van csomagolva, telepítéséhez mindössze ezt a fájlt kell letölteni és futtatni. A telepítés elvégzi az alapbeállításokat, azokon csak nagyon keveset vagy éppen semmit nem kell változtatni, ezután

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio)

a rendszer készen áll a webservert és a mintaalkalmazások futtatására. A XAMPP-csomagot rendszeresen frissítik, így az mindig az Apache, MariaDB, PHP és Perl legújabb változatát tartalmazza, valamint más kiegészítőket is, mint pl. az OpenSSL és a phpMyAdmin. Verziószáma a csomagban található PHP-értelmező verziójával egyezik meg (jelenleg az 5.5.x, 5.6.x és 7.0.x PHP-verziókhoz egyaránt elérhető különálló csomag). A XAMPP-csomag több példányban is telepíthető a gazdagépre, a különálló telepítések képesek önmagukban, a többi csomag megzavarása nélkül működni, ráadásul az installált példányok egyszerűen átmásolhatók egy másik gépre. A szoftver három változatban is elérhető: teljes, általános és kisméretű csomagok léteznek. Hivatalosan a XAMPP tervezői az eszközt egy fejlesztőrendszernek szánták, amellyel a web-tervezők és programozók internetes kapcsolat nélkül fejleszthetik és tesztelhetik alkalmazásaikat. Ennek érdekében több fontos biztonsági funkció alapértelmezésben ki van kapcsolva a csomagban, ennek ellenére a XAMPP szoftvert valódi webes szolgáltatóként is használják. A csomag egy külön eszközt tartalmaz a legfontosabb részek jelszavas védelmének beállítására. A XAMPP többféle adatbázis-kezelő használatát is támogatja, ilyenek pl. a MySQL és az SQLite és mások. A XAMPP telepítése után a helyi gép (a localhost) hálózati gépként is hozzáférhetővé válik, pl. FTP kliensprogrammal elérhető. Alkalmazható pl. FileZilla fájlkezelő, telepíthető tartalomkezelő rendszerek, mint a WordPress vagy a Joomla! A localhost közvetlenül is kezelhető egyes HTML vagy általános szövegszerkesztőkből, az FTP protokoll használatával. Az alapbeállítás szerinti FTP felhasználónév a „newuser”, jelszava „wampp”. Az alapbeállítás szerinti MySQL felhasználónév a „root”, jelszó nélkül.<sup>2</sup> A adatbázishoz és a weboldal megfelelő működéséhez a Xampp-ot választottam.

### 3.1.3 Visual Studio Code

A Visual Studio Code, más néven VS Code a Microsoft által az Electron keretrendszerrel készített forráskód-szerkesztő program, Windows, Linux és macOS operációs rendszerekhez, amelynek jellemzői közé tartozik a hibakeresés támogatása, a szintaxis kiemelése, az intelligens kódkiegészítés, a snippetek, a kód refaktorálása és a beágyazott Git. A felhasználók megváltoztathatják a témát, a billentyűparancsokat, a preferenciákat, és telepíthetik a funkciókat bővítő bővítményeket. A Stack Overflow 2022 fejlesztői felmérésében a Visual Studio Code a 71 010 válaszadó körében a legnépszerűbb fejlesztőkörnyezeti eszköznek bizonyult, 74,48%-uk jelezte, hogy használja. A Visual Studio Code egy forráskód-szerkesztő,

---

<sup>2</sup> <https://hu.wikipedia.org/wiki/XAMPP>

amely számos programozási nyelvvel használható, többek között C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. A program az Electron keretrendszeren alapul, amelyet a Blink elrendezési motoron futó Node.js webes alkalmazások fejlesztésére használnak. A Visual Studio Code ugyanazt a szerkesztő komponenst (kódnevén "Monaco") használja, amelyet az Azure DevOps (korábbi nevén Visual Studio Online és Visual Studio Team Services).

A Visual Studio Code alapszintű támogatást tartalmaz a legtöbb elterjedt programozási nyelvhez. Ez az alaptámogatás magában foglalja a szintaxis kiemelést, a zárójelek illesztését, a kód összehajtását és a konfigurálható snippeteket. A Visual Studio Code IntelliSense-t is tartalmaz a JavaScript, TypeScript, JSON, CSS és HTML számára, valamint hibakeresési támogatást a Node.js számára. További nyelvek támogatását a VS Code Marketplace-en szabadon elérhető bővítményekkel lehet biztosítani. A projektrendszer helyett lehetővé teszi a felhasználók számára, hogy egy vagy több könyvtárat nyissanak meg, amelyeket aztán munkaterületekre menthetnek a későbbi újrafelhasználás érdekében. Ez lehetővé teszi, hogy nyelv-agnosztikus kódszerkesztőként működjön bármilyen nyelvhez. Számos programozási nyelvet és nyelvenként eltérő funkciókészletet támogat. A nem kívánt fájlok és mappák a beállításokon keresztül kizárhatók a projektfából. A Visual Studio Code számos funkciója nem a menükön vagy a felhasználói felületen keresztül érhető el, hanem a parancspalette-n keresztül. A Visual Studio Code bővítményekkel bővíthető, amelyek egy központi tárolón keresztül érhetők el. Ez magában foglalja a szerkesztő és a nyelvtámogatás bővítéseit. Figyelemre méltó funkció a bővítmények létrehozásának lehetősége, amelyek új nyelvek, témák, hibakeresők, időutazó hibakeresők támogatását adják hozzá, statikus kódelemzést végeznek, és a Language Server Protocol segítségével kódlintereket adnak hozzá.<sup>3</sup>

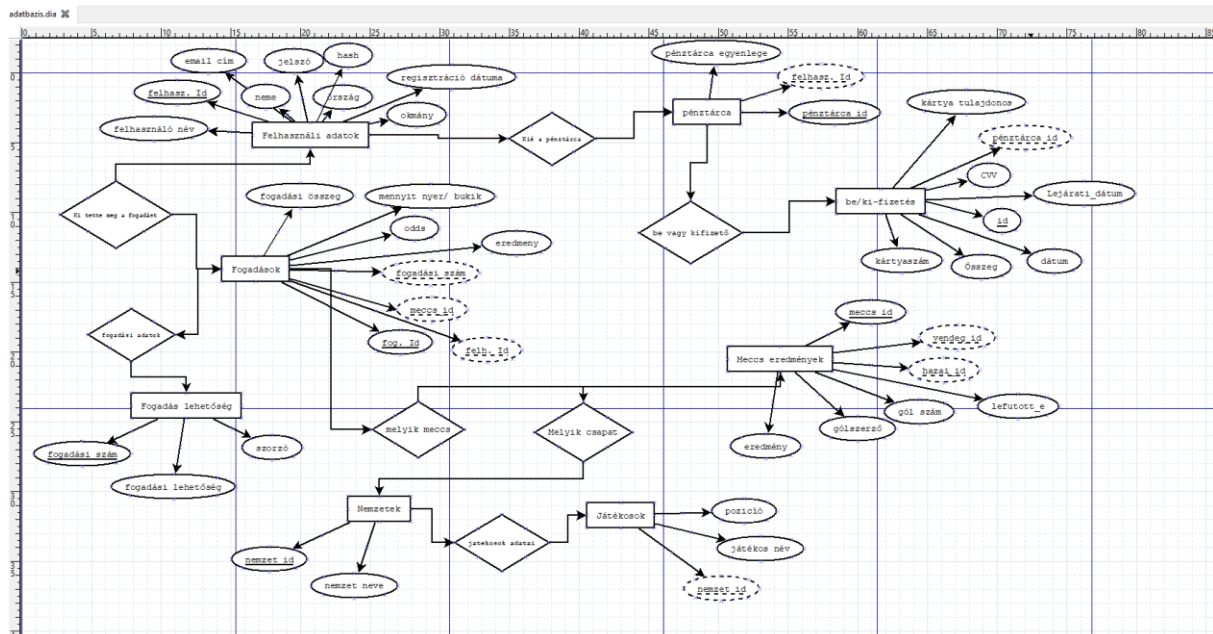
## 3.2 Adatbázis

Az adatbázist úgy próbáltam megtervezni és kivitelezni, hogy ha később szeretnék bővíteni az adatok benne vagy az oszlopokat több attribútummal akkor is zökkenőmentesen működjenek. Az adatbázisomat megpróbáltam normalizálni a lehető legjobb módon, hogy futás közben minél gyorsabban fusson a weboldal vagy sql utasítás közben gyorsabban tudjanak lefutni. Fő feladata az adatbázisomnak, hogy egy olyan adatszerkezetet hozzak létre vele, amely a weboldal és a hozzá tartozó programot segíti a tökéletes működésben. Úgy, hoztam létre az adatbázist, hogy ha a későbbiekben hozzá szeretnék adni plusz attribútumokat

---

<sup>3</sup> [https://en.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)

akkor az adatbázis továbbra is megfelelően fusson. Az adatbázishoz tervezés során készítettem egy ER modellt, amelyben próbáltam megtervezni az adatbázisomat majd ebből létrehozni a végleges formáját.



Ez az ER diagram már a végleges adatbázisomat ábrázolja. Ahhoz, hogy így nézzen ki felvezettem többször. Elsőnek 3 nagyobb táblám volt és ezeket normalizáltam úgy, hogy ne legyen benne ismétlődés csak ha tábla összekapcsolás miatt szükséges volt származtatni az adatot a másik táblába. Próbáltam nem belevinni felesleges adatokat csak annyit amennyit kell. Az adatbázisban több tábla is található, amelyek különböző adatokat tárolnak:

- **Pénztárca adatai (penztarca):** A felhasználók pénztárcájának adatait tárolja.
- **Játékosok tábla (jatekosok):** A nemzetekhez tartozó játékosokat tárol, és megadja, hogy milyen poszton játszanak.
- **Csapatok tábla (nemzetek):** A csapatok helyezési számát tartalmazza, amely alapján a C# alkalmazás dönteni tudja majd a meccsek eredményéről. Összesen 20 csapat van az adatbázisban.
- **Meccsek tábla (meccs\_eredmeny):** Tárolja a már lefutott meccsek eredményeit, beleértve a győztest, a gólszámot, a szögletek számát és a gól szerző játékos nevét is.
- **Felhasználói tábla (felhasznaloi\_adatok):** A regisztrációnál bevitt adatokat tárolja, mint például az e-mail címet, a felhasználónevet és a jelszót.
- **Tranzakciók tábla (be\_ki\_fizetes):** Tárolja a tranzakciók adatait.

- Ha szükséges, bármelyik tábla bővíthető újabb adatokkal a jövőben.



- Az `be_ki_fizetes` tábla `penztarca_id` oszlopa a `penztarca` tábla `penztarca_id` oszlopához kapcsolódik.
- Az `fogadas` tábla `fogadasi_szam` oszlopa a `fogadasi_lehetoseg` tábla `fogadasi_szam` oszlopához, az `felhasz_id` oszlopa az `felhasznaloi_adatok` tábla `felhasz_id` oszlopához, míg a `meccs_id` oszlopa az `meccs_eredmeny` tábla `meccs_id` oszlopához kapcsolódik.
- Az `jatekosok` tábla `nemzet_id` oszlopa a `nemzetek` tábla `nemzet_id` oszlopához kapcsolódik.
- Az `meccs_eredmeny` tábla `hazai_id` és `vendeg_id` oszlopai a `nemzetek` tábla `nemzet_id` oszlopához kapcsolódnak.
- Az `penztarca` tábla `felhasz_id` oszlopa az `felhasznaloi_adatok` tábla `felhasz_id` oszlopához kapcsolódik.

## 3.3 Adatmodell

### 3.3.1 Be/ki fizetés tábla

#	Név	Típus
1	id 🔑	int(10)
2	penztarca_id 🔑	int(11)
3	datum	date
4	kartyaszam	varchar(255)
5	lejarati_datum	varchar(255)
6	cvv	varchar(255)
7	osszeg	double
8	kartya_tulajdonos	varchar(255)

A `be_ki_fizetes` nevű tábla a felhasználók által végrehajtott befizetések és kifizetések rögzítésére szolgál. A tábla 8 oszlopa van, amelyek a következő információkat tartalmazzák:

1. `id`: Az egyedi azonosító értéket tartalmazza, amely minden rekordban megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
2. `penztarca_id`: Ez az oszlop azonosítja a pénztárcát, amelyhez a befizetés vagy kifizetés kapcsolódik. A pénztárca azonosítója segítségével lehet követni, hogy melyik felhasználó melyik pénztárcájához kapcsolódó tranzakciókról van szó.
3. `datum`: Ez az oszlop a tranzakció dátumát tartalmazza. A dátum az adott tranzakció létrejöttének időpontját jelöli.
4. `kartyaszam`: Ez az oszlop a használt bankkártya számát tartalmazza. A bankkártya száma szükséges a tranzakció végrehajtásához.
5. `lejarati_datum`: Ez az oszlop a bankkártya lejárat dátumát tartalmazza. A lejárat dátum az adott bankkártya érvényességi idejét jelöli.
6. `cvv`: Ez az oszlop a bankkártya CVV kódját tartalmazza. A CVV kód szükséges a bankkártya használatához.
7. `osszeg`: Ez az oszlop a tranzakció összegét tartalmazza. Az összeg a befizetés vagy kifizetés összegét jelöli.
8. `kartya_tulajdonos`: Ez az oszlop a bankkártya tulajdonosának nevét tartalmazza.

Összességében a `be_ki_fizetes` tábla a felhasználók által végrehajtott tranzakciók adatait tartalmazza, amelyekre a weboldalon való ki- és befizetések során van szükség. A tábla



azonosítja az érintett felhasználókat, pénztárcákat és bankkártyákat, valamint rögzíti a tranzakció dátumát és összegét.

### 3.3.2 Felhasználói adatok tábla

#	Név	Típus
1	<b>felhasz_id</b> 🔑	int(11)
2	<b>felhasz_nev</b>	varchar(255)
3	<b>email</b>	varchar(255)
4	<b>jelszo</b>	varchar(255)
5	<b>hash</b>	varchar(255)
6	<b>reg_datum</b>	date
7	<b>okmany</b>	varchar(255)
8	<b>neme</b>	varchar(255)
9	<b>orszag</b>	varchar(255)

A felhasználói\_adatok nevű tábla a felhasználók adatait tartalmazza, akik regisztráltak a weboldalon. A tábla 9 oszlopa van, amelyek a következő információkat tartalmazzák:

1. **felhasz\_id**: Az egyedi azonosító értéket tartalmazza, amely minden rekordban megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
2. **felhasz\_nev**: Ez az oszlop a felhasználó nevét tartalmazza, amelyet a felhasználó regisztrációkor adott meg.
3. **email**: Ez az oszlop a felhasználó e-mail címét tartalmazza, amelyet a felhasználó regisztrációkor adott meg.
4. **jelszo**: Ez az oszlop a felhasználó jelszavát tartalmazza, amelyet a felhasználó regisztrációkor adott meg. A jelszót általában hash-elik vagy titkosítják, hogy biztonságosabb legyen.
5. **hash**: Ez az oszlop a felhasználó jelszavának hash-ját tartalmazza. A hash értéket a jelszó alapján számítják ki, és általában a jelszó titkosítására használják.
6. **reg\_datum**: Ez az oszlop a felhasználó regisztrációjának dátumát tartalmazza, amelyet a felhasználó regisztrációkor adott meg.

7. okmany: Ez az oszlop az okmányok típusát tartalmazza, amelyekkel a felhasználó igazolni tudja a személyazonosságát. A megadott okmány típusát általában a felhasználó választja ki a regisztrációkor.
8. neme: Ez az oszlop a felhasználó nemét tartalmazza, amelyet a felhasználó regisztrációkor adott meg.
9. orszag: Ez az oszlop az országot tartalmazza, amelyben a felhasználó él, és amelyet a felhasználó regisztrációkor adott meg.

Összességében a felhasználói\_adatok tábla az adatokat tartalmazza, amelyekre a weboldalon regisztrált felhasználóknak szükségük van ahhoz, hogy ki- és befizetéseket hajtsanak végre. A tábla azonosítja a felhasználókat, és rögzíti a felhasználó nevét, e-mail címét, jelszavát, regisztrációs dátumát, okmány típusát, nemét és országát.

### 3.3.3 Fogadások tábla

#	Név	Típus
1	felhasz_id 🔑	int(11)
2	fog_id 🔑	int(11)
3	fogadasi_osszeg	int(11)
4	profit_buko	double
5	meccs_id 🔑	int(11)
6	fogadasi_szam 🔑	int(11)
7	eredmeny	varchar(255)
8	odds	double


A fogadas nevű tábla a fogadásokat tartalmazza, amelyeket a felhasználók hajtanak végre a weboldalon. A tábla 8 oszlopa van, amelyek a következő információkat tartalmazzák:

1. felhasz\_id: Ez az oszlop azonosítja a felhasználót, aki a fogadást elhelyezte. Ez az oszlop idegen kulcsként szolgál, amely kapcsolódik a felhasználói\_adatok táblában található azonosítóhoz.
2. fog\_id: Ez az oszlop az egyedi azonosítót tartalmazza, amely minden fogadás esetében megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
3. fogadasi\_osszeg: Ez az oszlop a fogadásra helyezett összeg értékét tartalmazza.
4. profit\_buko: Ez az oszlop a fogadás nyereségének vagy veszteségének értékét tartalmazza. A nyereség vagy veszteség a fogadás kimenetelétől függően változik.

5. `meccs_id`: Ez az oszlop azonosítja a meccset, amelyre a fogadás vonatkozik. Az azonosító idegen kulcsként szolgál, amely kapcsolódik a `meccs_eredmeny` táblában található azonosítóhoz.
6. `fogadasi_szam`: Ez az oszlop a fogadási számot tartalmazza. A fogadási szám idegen kulcsként szolgál, amely kapcsolódik a `fogadasi_lehetoseg` táblában található azonosítóhoz.
7. `eredmeny`: Ez az oszlop a fogadás eredményét tartalmazza. Az eredmény lehet nyertes vagy vesztes a fogadás kimenetelétől függően.
8. `odds`: Ez az oszlop az odds értéket tartalmazza, amely a fogadás kötésekor volt érvényes. Az odds a fogadás esélyét jelöli, és határozza meg a fogadás potenciális nyereségét.

Összességében a fogadas tábla azokat az adatokat tartalmazza, amelyekre a weboldalon fogadásokat elhelyező felhasználóknak szükségük van. A tábla azonosítja a felhasználókat, és rögzíti a fogadás összegét, a fogadás eredményét, az odds értékét, valamint a fogadási számot.

### 3.3.4 Fogadási lehetőségek tábla


#	Név	Típus
1	<code>fogadasi_szam</code> 	<code>int(11)</code>
2	<code>fogadas_neve</code>	<code>varchar(255)</code>
3	<code>szorzo</code>	<code>double</code>

A `fogadasi_lehetoseg` nevű tábla a weboldalon elérhető fogadási lehetőségeket tartalmazza. A tábla 3 oszlopa van, amelyek a következő információkat tartalmazzák:

1. `fogadasi_szam`: Ez az oszlop az egyedi azonosítót tartalmazza, amely minden fogadási lehetőség esetében megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
2. `fogadas_neve`: Ez az oszlop a fogadási lehetőség nevét tartalmazza, amelyet a felhasználók a weboldalon választhatnak ki. A nevek általában leírják a fogadási lehetőséget vagy a meccs eredményére vonatkozó kimenetelt.
3. `szorzo`: Ez az oszlop az odds értéket tartalmazza, amely a fogadási lehetőségre vonatkozik. Az odds értékének magasabbnak kell lennie, ha a fogadás valószínűsége alacsonyabb, és fordítva.

Összességében a fogadasi\_lehetoseg tábla azokat az adatokat tartalmazza, amelyekre a weboldalon fogadásokat elhelyező felhasználóknak szükségük van ahhoz, hogy kiválasszák a megfelelő fogadási lehetőséget. A tábla azonosítja a fogadási lehetőségeket, és rögzíti a fogadási lehetőség nevét és az odds értékét.

### 3.3.5 Nemzetek tábla


#	Név	Típus
1	nemzet_id 	int(11)
2	nemzet_nev	varchar(255)

A nemzetek nevű tábla az országokat tartalmazza, amelyek részt vesznek az adott sporteseményekben. A tábla 2 oszlopa van, amelyek a következő információkat tartalmazzák:

1. nemzet\_id: Ez az oszlop az egyedi azonosítót tartalmazza, amely minden ország esetében megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
2. nemzet\_nev: Ez az oszlop az ország nevét tartalmazza, amely részt vesz az adott sporteseményben. Az országok neve általában a hivatalos nevük vagy a sportban használt nevük lehet.

Összességében a nemzetek tábla azokat az országokat tartalmazza, amelyek részt vesznek az adott sporteseményekben. A tábla azonosítja az országokat, és rögzíti a nevüket. A tábla segítségével a felhasználók könnyen megtalálhatják az országokat, és az adminisztrátorok könnyen kezelhetik az országokkal kapcsolatos adatokat.

### 3.3.6 Játékosok tábla

#	Név	Típus
1	nemzet_id 	int(11)
2	jatekos_nev	varchar(255)
3	pozicio	varchar(255)

A játékosok nevű tábla az adott sportesemény játékosait tartalmazza. A tábla 3 oszlopa van, amelyek a következő információkat tartalmazzák:

1. **nemzet\_id**: Ez az oszlop azonosítja az országot, amelynek a játékosa részt vesz az adott sporteseményben. Az azonosító idegen kulcsként szolgál, amely kapcsolódik a nemzetek táblában található azonosítóhoz.
2. **jatekos\_nev**: Ez az oszlop a játékos nevét tartalmazza, aki részt vesz az adott sporteseményben.
3. **pozicio**: Ez az oszlop a játékos pozícióját tartalmazza a sporteseményen belül. A pozíció lehet például hátvéd, középpályás, támadó vagy kapus.

Összességében a **jatekosok** tábla azokat a játékosokat tartalmazza, akik részt vesznek az adott sporteseményben. A tábla azonosítja a játékosokat, és rögzíti a nevüket és a pozíciójukat.

### 3.3.7 Meccs eredmények tábla

#	Név	Típus
1	<b>meccs_id</b> 🔑	int(11)
2	<b>eredmeny</b>	varchar(255)
3	<b>gol_szerzo</b>	varchar(255)
4	<b>golszam</b>	int(11)
5	<b>hazai_id</b> 🔑	int(11)
6	<b>vendeg_id</b> 🔑	int(11)
7	<b>lefutott_e</b>	int(1)

A **meccs\_eredmeny** nevű tábla az adott sportesemény meccseinek eredményét tartalmazza. A tábla 7 oszlopa van, amelyek a következő információkat tartalmazzák:

1. **meccs\_id**: Ez az oszlop az egyedi azonosítót tartalmazza, amely minden meccs esetében megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
2. **eredmeny**: Ez az oszlop az adott meccs eredményét tartalmazza. Az eredmény lehet hazai győzelem, vendég győzelem vagy döntetlen.
3. **gol\_szerzo**: Ez az oszlop a meccsen szerzett gólok szerzőit tartalmazza.
4. **golszam**: Ez az oszlop az adott meccsen szerzett összes gól számát tartalmazza.
5. **hazai\_id**: Ez az oszlop azonosítja a hazai csapatot, amely részt vesz az adott meccsen. Az azonosító idegen kulcsként szolgál, amely kapcsolódik a nemzetek táblában található azonosítóhoz.

6. `vendeg_id`: Ez az oszlop azonosítja a vendégcsapatot, amely részt vesz az adott meccsen. Az azonosító idegen kulcsként szolgál, amely kapcsolódik a nemzetek táblában található azonosítóhoz.
7. `lefutott_e`: Ez az oszlop azt jelzi, hogy az adott meccs már lefutott-e vagy sem. Az értéke 1, ha a meccs már lefutott, és 0, ha még nem.

Összességében a `meccs_eredmeny` tábla azokat az adatokat tartalmazza, amelyekre szükség van ahhoz, hogy az adott sportesemény résztvevői és a felhasználók nyomon követhessék a meccsek eredményeit. A tábla azonosítja a meccseket, és rögzíti az eredményüket, a gólszerzőket és a gólok számát.

### 3.3.8 Pénztárca tábla

#	Név
1	<code>felhasz_id</code> 🔑
2	<code>penztarca_id</code> 🔑
3	<code>egyenleg</code>

A `penztarca` nevű tábla a felhasználók pénztárcájának információit tartalmazza. A tábla 3 oszlopa van, amelyek a következő információkat tartalmazzák:

1. `felhasz_id`: Ez az oszlop azonosítja a felhasználót, akinek a pénztárcájáról van szó. Az azonosító idegen kulcsként szolgál, amely kapcsolódik a felhasználók táblában található azonosítóhoz.
2. `penztarca_id`: Ez az oszlop az egyedi azonosítót tartalmazza, amely minden pénztárca esetében megtalálható. Ez az oszlop a tábla elsődleges kulcsa, amely segítségével az adatokat könnyen lehet azonosítani és kezelni.
3. `egyenleg`: Ez az oszlop a pénztárca egyenlegét tartalmazza. Az egyenleg lehet pozitív vagy negatív, és azt mutatja meg, hogy mennyi pénz áll rendelkezésre a felhasználónak a weboldalon való fogadásokhoz.

Összességében a `penztarca` tábla azokat az adatokat tartalmazza, amelyekre szükség van ahhoz, hogy nyomon lehessen követni a felhasználók pénztárcájában található pénzügyi információkat. A tábla azonosítja a felhasználók pénztárcáit, és rögzíti az egyenlegeket.

## 3.4 Részletes feladat specifikáció, algoritmusok

Itt a program lényeges függvényeinek, az osztályok metódusainak a specifikációja (mit valósít meg az adott függvény, illetve metódus, milyen paraméterei vannak, mi a visszatérési érték).

### 3.4.1 C# program felépítése

A meccsek eredményeinek és a végkimenetelésnek generálás súlyozott, azaz figyelembe veszi a csapatok utóbbi 5 mérkőzésén mennyi meccset tudott nyerni a két csapat és azt is, hogy a kettő csapat az előző 5 mérkőzésén átlagban hány gólt tudott szerezni összesen. Ennek megfelelően dönti el, hogy melyik csapatnak van nagyobb esélye gólt löni, mennyi gól lesz a mérkőzésen és melyik csapat nyeri meg a meccset. Az eredményeket és a statisztikákat az adatbázisba küldi el a C# program, ahol a megfelelő táblákban tárolja őket. Ezek az adatok lesznek majd az oldalon megjelenítve a php segítségével. Az eredmények valóságghűek és nem generálnak irracionális eredményeket.

```
class Database
{
    static public MySqlCommand command;
    static public MySqlConnection connection;

    1 reference
    public Database(string server = "localhost", string user = "root", string password = "", string db = "fogadas_zaro")
    {
        MySqlConnectionStringBuilder builder = new MySqlConnectionStringBuilder();
        builder.Server = server;
        builder.UserID = user;
        builder.Password = password;
        builder.Database = db;
        connection = new MySqlConnection(builder.ConnectionString);
        if (Kapcsolatok())
        {
            command = connection.CreateCommand();
        }
    }

    1 reference
    private bool Kapcsolatok()
    {
        try
        {
            connection.Open();
            connection.Close();
            return true;
        }
        catch (MySqlException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }
}
```

Ez egy C# osztály, amely az adatbázis kapcsolatot kezeli. Az osztály neve "Database". Az osztály két statikus publikus változót tartalmaz: "command" és "connection". Ezek a változók az adatbáziskapcsolatot és az adatbázis lekérdezéseket kezelik. Az osztály konstruktora beállítja az adatbázis-kapcsolat paramétereit, majd létrehozza az adatbázis-kapcsolatot. Ha sikeres a kapcsolat, akkor létrehozza az adatbázis-lekérdezést is. Az osztály privát bool típusú függvénye a "Kapcsolatok()", amely az adatbáziskapcsolatot teszteli. Megpróbálja megnyitni az adatbáziskapcsolatot, majd, ha sikeresen megnyílt, bezárja azt, és igazzal tér vissza. Ha hiba történik a kapcsolat nyitása közben, akkor a hibát kiírja a konzolra, és hamissal tér vissza.

Összességében ez az osztály az adatbázis-kapcsolatot és az adatbázis-lekérdezéseket kezeli a C# programban.

```
Random rng = new Random(Guid.NewGuid().GetHashCode());
Database database = new Database();
int[] csapatok = database.csapatsorsolas(rng);
string[] csapatnev = database.csapatnev(csapatok);
int hazainyer = database.getcsapatnyer(csapatok[0]);
int vendegnyer = database.getcsapatnyer(csapatok[1]);
```

```
int[] osszes = database.csapatgolkok();
int hazaigolszam = osszes[0];
int vendeggolszam = osszes[1];
double golsum = Convert.ToDouble(hazaigolszam + vendeggolszam);
double normal = 2 - (golsum / 80);
double inverznorm = 1 + (golsum / 80);
double golnormal = golsum / 80;
```

Valószínűségi számítási program része, amely sportesemények eredményeire vonatkozó szorzókat és adatokat számít ki. A kód első sorában egy új Random objektumot hoz létre, amelynek inicializáló paramétere a Guid.NewGuid().GetHashCode() függvény eredménye. Ez a függvény egy új GUID-t generál, majd ennek hash kódját adja vissza, ami egyedi random számot eredményez. A második sorban egy új Database objektumot hoz létre, amely egy adatbázis-kezelő osztály, amely lehetővé teszi a program számára, hogy adatokat írjon és olvasson az adatbázisból.

```
public int[] csapatsorsolas(Random rng)
{
    int[] adatvissza = new int[2];
    command.Parameters.Clear();
    command.CommandText = "SELECT COUNT(`nemzet_id`) AS 'idcount' FROM `nemzetek`";
    connection.Open();

    using (MySqlDataReader dr = command.ExecuteReader())
    {
        dr.Read();
        int temp = dr.GetInt32("idcount") + 1;
        adatvissza[0] = rng.Next(1, temp);
        bool a = true;
        while (a)
        {
            adatvissza[1] = rng.Next(1, temp);
            if (adatvissza[0] != adatvissza[1])
            {
                a = false;
            }
        }
    }
    connection.Close();
    return adatvissza;
}
```

A következő sorokban a program lekérdezi adatbázisból a kisorsolt csapatok nevét és a csapatok előző 5 mérkőzésnek győzelmeinek számát és ezek segítségével fogja legenerálni az eredményt és ehhez a szorzókat is. Lekérdezi az előző 5 meccsüknek az összes gólszámát és átlagot számol és ezeket fogja felhasználni a gólok generálására a csapatok góljainak generálására is és ezeknek a megfogadásához szükséges szorzókat is.



```

public int getcsapatnyer(int csapat_id)
{
    int temp = 0;
    command.Parameters.Clear();
    command.CommandText = "SELECT * FROM 'meccs_eredmeny' WHERE 'hazai_id' = @id OR 'vendeg_id' = @id ORDER BY meccs_id DESC LIMIT 5;";
    command.Parameters.AddWithValue("@id", csapat_id);
    connection.Open();
    using (MySqlDataReader dr = command.ExecuteReader())
    {
        while (dr.Read())
        {
            int hazai = dr.GetInt32("hazai_id");
            int vendeg = dr.GetInt32("vendeg_id");
            int[] eredmeny = Array.ConvertAll(dr.GetString("eredmeny").Split("-"), int.Parse);
            if (hazai == csapat_id)
            {
                if (eredmeny[0] > eredmeny[1])
                {
                    temp++;
                }
            }
            else if (vendeg == csapat_id)
            {
                if (eredmeny[0] < eredmeny[1])
                {
                    temp++;
                }
            }
        }
    }
    connection.Close();
    return temp;
}

```

Miután ezeket az adatokat a program bekérte ezeknek a segítségével kisorsolja a szorzókat. Elsőnek megadtam a módszernek a szükséges adatokat. Amelyek a két csapat 5 meccsből megnyert meccsek száma.

```

double[] multiplier = GenerateMultiplier(hazainyer, vendegnyer);

```

A kód két tömbben generál szorzókat, egyet a hazai csapat számára (hazaiSzorzok) és egyet a vendégcsapat számára (vendegSzorzok). Mindkét tömbben 7 szorzó található, amelyek különböző súlyozással rendelkeznek a csapatok eredményei alapján. A szorzók generálása a hazai csapat számára a következő lépéseken keresztül történik:

1. Először meghatározzuk az összes nyerési esetet a hazai és vendég csapatok között, és azok összegét (osszWinHazai).
2. Ezután kiszámítjuk az alap szorzókat mindkét csapatra, a hazai csapat esetében ez a hazaiSzorzoAlap.
3. Ha a hazai csapat alap szorzója nagyobb, mint a vendég csapaté (vendegSzorzoAlap), akkor a vendégcsapat alap szorzóját megszorozzuk 1,2-vel (vendegSzorzoAlap = vendegSzorzoAlap \* 1.2). Ezután kiszámítjuk a további szorzókat a hazai csapat számára, amelyek eltérő súlyozással rendelkeznek.
4. Ezután a többi szorzót a hazai csapat számára arra, hogy hány gólt fog lőni a hazai. Mivel a hazai csapat az esélyesebb így rá kisebb szorzókat fog generálni a program. Úgy lett beállítva, hogy az esélyesebb csapatnak nagyobb az esélye, hogy több gólt fog lőni így erre kevesebb a szorzó és így fordítva a gyengébb csapatra. Arra, hogy kevés gólt fog lőni az erősebb csapat meg logikusan nagyobb a szorzó mivel az várható, hogy sok gólt fog szerezni a meccsen.

A szorzók generálása a vendégcsapat számára hasonlóan történik, csak a szorzók súlyozása és az alap szorzók meghatározása a vendégcsapat eredményei alapján történik. Az indok az, hogy a szorzók súlyozása a csapatok eredményei alapján történik, és a nagyobb szorzó azt jelenti, hogy a hazai csapat jobb eredményeket ért el a múltban, és nagyobb eséllyel nyer a következő mérkőzésen. Ezért a vendégcsapat szorzóit kell növelni, hogy jobban tükrözzék a hazai csapat dominanciáját. Ha pont fordítva lenne a helyzet, és a vendégcsapat alap szorzója lenne nagyobb, akkor a hazai csapat szorzóit kellene növelni, hogy jobban tükrözzék a vendégcsapat gyengeségét.

```
// Vendég szorzók kiszámítása
double osszWinVendeg = vendegnyer + 1 + hazainyer + 1;
double vendegSzorzoAlap2 = osszWinVendeg / (vendegnyer + 1);
double hazaiSzorzoAlap2 = osszWinVendeg / (hazainyer + 1);

if (vendegSzorzoAlap2 > hazaiSzorzoAlap2)
{
    hazaiSzorzoAlap2 = hazaiSzorzoAlap2 * 1.2;
}
if (vendegSzorzoAlap2 < hazaiSzorzoAlap2)
{
    vendegSzorzoAlap2 = vendegSzorzoAlap2 * 1.2;
}

vendegSzorzo[0] = Math.Round(vendegSzorzoAlap2, 2);
vendegSzorzo[1] = Math.Round(hazaiSzorzoAlap2 * 1.2, 2);
vendegSzorzo[2] = Math.Round(hazaiSzorzoAlap2 * 1.4, 2);
vendegSzorzo[3] = Math.Round(hazaiSzorzoAlap2 * 2.0, 2);
vendegSzorzo[4] = Math.Round(hazaiSzorzoAlap2 * 1.3, 2);
vendegSzorzo[5] = Math.Round(hazaiSzorzoAlap2 * 1.7, 2);
vendegSzorzo[6] = Math.Round(hazaiSzorzoAlap2 * 2.3, 2);
```

```
double[] szorzok = new double[14];
szorzok[0] = hazaiSzorzo[0];
szorzok[1] = vendegSzorzo[0];
szorzok[5] = hazaiSzorzo[1];
szorzok[6] = hazaiSzorzo[2];
szorzok[7] = hazaiSzorzo[3];
szorzok[8] = hazaiSzorzo[4];
szorzok[9] = hazaiSzorzo[5];
szorzok[10] = hazaiSzorzo[6];
szorzok[2] = vendegSzorzo[1];
szorzok[3] = vendegSzorzo[2];
szorzok[4] = vendegSzorzo[3];
szorzok[11] = vendegSzorzo[4];
szorzok[12] = vendegSzorzo[5];
szorzok[13] = vendegSzorzo[6];
return szorzok;
```

Sikeres generálás után egy double tömbben tárolom a szorzókat. Ezek mellé legenerálom még a döntetlenre a szorzókat. Elsőnek ennek is bekérem az adatokat. Amelyek a már legenerált hazai és vendég csapatokra a szorzók.

```
double drawMultiplier = Math.Round(GenerateDrawMultiplier(multiplier[0], multiplier[1]), 2);
```

Ezután egy függvénnyel le generálom a szorzót a döntetlenre.

1. Az első lépésben összegezzük a hazai és vendég csapatok szorzóit (homeTeamMultiplier és awayTeamMultiplier), majd ezt az összeget 1,5-vel osztjuk.
2. Az így kapott értéket adjuk vissza a függvény visszatérési értékeként, amely az alkalmazandó szorzót jelenti, ha a mérkőzés döntetlen eredménnyel zárul.

A függvény tehát egy olyan szorzót generál, amely figyelembe veszi mindkét csapat szorzóját, és a döntetlen esetén alkalmazandó szorzó értéket adja vissza. A szorzó értéke az

adott csapatok formájától és eredményeitől függ, és általában kisebb, mint a győzelmi szorzók.

Ezt követően már csak a mérkőzésre generáltam le a szorzókat. Bekértem az adatokat a függvénynek. Az adatbázisból bekértem a kisorsolt vendég és hazai csapat góljainak számát és ezt tovább adtam a függvénynek már a le kezelt módon, amit már mutattam képen. Így végül tovább adtam a normal és inverznormál vétőzóban az adatokat.

```
command.Parameters.Clear();
command.CommandText = "SELECT SUM(golszam) FROM (select `meccs_eredmeny`.`golszam` from meccs_eredmeny WHERE `hazai_id` = @hazaid ORDER BY meccs_eredmeny.golszam) AS temp";
command.Parameters.AddWithValue("@hazaid", temp[0]);
using (MySqlDataReader dr = command.ExecuteReader())
{
    if (dr.Read())
    {
        int hazaiGol = dr.GetInt32("SUM(golszam)");
        adatvissza2[0] = hazaiGol;
    }
}

command.Parameters.Clear();
command.CommandText = "SELECT SUM(golszam) FROM (select `meccs_eredmeny`.`golszam` from meccs_eredmeny WHERE `vendeg_id` = @vendegid ORDER BY meccs_eredmeny.golszam) AS temp";
command.Parameters.AddWithValue("@vendegid", temp[1]);
using (MySqlDataReader dr = command.ExecuteReader())
{
    if (dr.Read())
    {
        int vendegGol = dr.GetInt32("SUM(golszam)");
        adatvissza2[1] = vendegGol;
    }
}

connection.Close();
return adatvissza2;
```

```
double[] odds = GenerateTotalGoalsOdds(normal, inverznormál);
```

Amelyek a gól számok voltak. Az előző 5 meccs átlag gól számai.

```
public static double[] GenerateTotalGoalsOdds(double normal, double inverznormál)
{
    // Az over/under szorzók kiszámítása
    double[] odds = new double[7];

    odds[0] = Math.Round(normal - 0.3, 2);
    odds[1] = Math.Round(normal * 1.3, 2);
    odds[2] = Math.Round(normal * 2.1, 2);
    odds[3] = Math.Round(normal * 6.4, 2);
    odds[4] = Math.Round(inverznormál * 1.1, 2);
    odds[5] = Math.Round(inverznormál * 1.6, 2);
    odds[6] = Math.Round(inverznormál * 2.1, 2);

    return odds;
}
```

Létrehoztam egy 7 elemű double tömböt az adatok tárolására. Ezután le generálom a gólszámokra a szorzókat olyan módon, hogy az első négy az (1.5,2.5,3.5,5) -nél több gólra szól. Maradék 3 szorzó pedig (3.5,2.5,1.5) -nél kevesebb gólra szól. A normal-t és az inverznormál-

t úgy szoroztam, hogy reális szorzók alakuljanak ki a meccsekre. Mivel a gólszámok is ezeknek a segítségével van legenerálva így nem randomok a szorzók és a gólok sem hanem az előző meccsek alapján történik minden.

```
database.adatkiiratasszorzo(multiplier, drawMultiplier);  
database.adatki("0-0", csapatok[0], csapatok[1]);
```

Miután szorzók le lettek generálva kiíratam őket az adatbázisba, hogy az oldal tudja kezelni azokat. Ezt egy függvény meghívásának a segítségével csináltam meg. Plusz kiíratom adatbázisba a kisorsolt csapatokat is.

```
database.szorzooverunder(odds);
```

A meghívott függvények az adatbázisba írják ki az adatokat. SQL parancsok segítségével.

```
public void adatki(string eredmény, int hazai_id, int vendeg_id)  
{  
    command.Parameters.Clear();  
    command.CommandText = "INSERT INTO `meccs_eredmeny` (eredmeny, `hazai_id`, `vendeg_id`, lefutott_e) VALUES (@eredmeny, @hazai, @vendeg, '0')";  
    command.Parameters.AddWithValue("@eredmeny", eredmény);  
    command.Parameters.AddWithValue("@hazai", hazai_id);  
    command.Parameters.AddWithValue("@vendeg", vendeg_id);  
    connection.Open();  
    command.ExecuteNonQuery();  
    connection.Close();  
}
```

Hasonló módon történik az összes kiírtatás.

```
public void adatkiiratasszorzo(double[] szorzo, double dontetlen)  
{  
    connection.Open();  
    command.Parameters.Clear();  
  
    command.CommandText = "UPDATE fogadasi_lehetoseg SET szorzo = @szorzo WHERE fogadasi_szam = 1;";  
    command.Parameters.AddWithValue("@szorzo", szorzo[0]);  
    command.ExecuteNonQuery();  
    command.Parameters.Clear();  
}
```

Minden egyes szorzót egyesével íratok ki az adatbázisba a megfelelő helyére a nevéhez kötve. A WHERE feltételben megadom nekik melyik fogadási lehetőséghez tartozik.

```
public void szorzooverunder(double[] odds)  
{  
    connection.Open();  
    command.Parameters.Clear();  
    command.CommandText = "UPDATE fogadasi_lehetoseg SET szorzo = @szorzo WHERE fogadasi_szam = 4;";  
    command.Parameters.AddWithValue("@szorzo", odds[0]);  
    command.ExecuteNonQuery();  
    command.Parameters.Clear();  
}
```

Miután lefutott az összes kiírtatás 30 másodpercre alvásra tettem a programot, ez idő alatt tud a felhasználó fogadni a meccsre a weboldalon.

```
Thread.Sleep(30000);
```

Ezután tovább fog haladni a program és lefogja sorsolni a meccs eredményeit. Azt, hogy ki nyert mennyi gól született melyik csapat mennyi gólt lőtt és az adott csapatból melyik játékos lőtt gólt.

Elsőnek is a meccs eredményének a sorsolását szeretném bemutatni. A sorsolás úgy történik, hogy súlyozott generálást használok. Amely úgy működik, hogy megnézem, hogy az előző 5 meccsen hány meccset nyert meg a hazai és a vendégcsapat. Ezt eltárolom és átalakítom úgy, hogy feltudjam osztani a kettő csapatnak az esélyeit 1 és -1 közé. Tehát mindkettő csapatot az esélyek alapján elosztom ebbe a range-be. Figyelembe véve az előző meccseken mennyit nyertek így az egyik, ha többet nyert az előző meccsen akkor nagyobb részt kap a gyengébbik csapat kisebbet és ebbe meg megfelelő elosztással és eséllyel beleépítettem a döntetlenre is az esélyt.

```
double suly = Convert.ToDouble(vendegnyer - hazainyer) / 10;  
double hazaisuly = 0;  
if (Math.Abs(suly) == suly)  
{  
    hazaisuly = suly * 1.5;  
}  
else  
{  
    hazaisuly = suly;  
}  
double vendegsuly = 0;  
if (Math.Abs(suly) != suly)  
{  
    vendegsuly = suly * 1.5;  
}  
else  
{  
    vendegsuly = suly;  
}  
  
double hazairange = 0.2 + hazaisuly;  
double vendegrange = -0.2 + vendegsuly;
```

Ezeket az elosztott részeket eltárolom. Majd egy függvényben felhasználom a meccs eredményének az eldöntésére. Elsőnek is meghívom a kellő adatokat a metódusba. Átadom az elkészített elosztást a gólszámoknál elkészített gól átlagot és egy random generált számot, ami 1-től -1-ig generál egy számot. Ez fogja majd eldönteni a meccs eredményét.

```
int[] eredmeny = sorsolas(hazairange, vendegrange, rng, golnormal);
```

A bekerült adatokkal felhasználva eldöntjük a meccs eredményét. Mégpedig legenerálódik az elosztott részhez a random szám így meglesz az eredményünk, amit eltárolunk. Ezután gólszámokat generáljuk le a meccshez az áthozott normállal, ami befolyásolja így gólszámot

úgy, hogy a két csapat előző meccsinek gól átlagjával fogja generálni a mostani mérkőzésre a gólokat. Ezután egy ellenőrzést hajtok végre, hogy ki nyert. Ezután következik a gól sorsolás csapatokra nézve. Le sorsolom a gólokat úgy, hogy a győztesnek több gólja legyen logikusan. Arra is figyelve, hogy ne irracionális eredmény szülessen. Bár mivel van esély rá, hogy a gyengébb nyerjen így arra is van esély, hogy kicsit furább eredmények szülessenek.

```
static int[] sorsolas(double hazairange, double vendegrage, Random rng, double golnormal)
{
    int[] az = new int[3];
    double random = Convert.ToDouble(rng.Next(-1000, 1001)) / 1000;
    int fix = rng.Next(1, 3+Convert.ToInt32(Math.Ceiling(1*golnormal)));
    if (random > hazairange)
    {
        double szorzo = ((random - hazairange) / (1 - hazairange));
        double hazaipont = Math.Round(fix + szorzo * fix);
        double vendegpont = Math.Floor(fix - szorzo * fix);
        int golszam = Convert.ToInt32(hazaipont + vendegpont);
        Console.WriteLine("Gólszámok/ eredmény:");
        Console.WriteLine();
        Console.WriteLine($"{golszam} gól volt a meccsen");
        Console.WriteLine($"{hazaipont}-{vendegpont} az eredmény");
        Console.WriteLine("Hazai nyert");
        Console.WriteLine("-----");
        az[1] = Convert.ToInt32(hazaipont);
        az[2] = Convert.ToInt32(vendegpont);
        az[0] = 1;
        return az;
    }
}
```

A fenti példa kód a hazai csapatot mutatja be, ha a hazai nyer. A benne lévő kiírtatások csak a C# program ellenőrzését szolgálták. A vendégcsapatnál és a döntetlennél nagyjából ugyan ez a szerkezetű kód, algoritmus fut le. Ha megvolt ezeknek a legenerálása akkor eltárolom a hazai csapat gólszámát és a vendégcsapat gólszámát, utolsónak pedig egy „1” -est, ha hazai nyert „2” -est, ha vendég és „3” -ast, ha döntetlen lett a mérkőzés. Ezt követően kisorsolom melyik játékosok lőtték a gólokat. Ezt egy függvény segítségével oldottam meg ahol a csatárnak van a legnagyobb esélye a gólszerzésre és így szépen haladva lefelé az eséllyel ahogy haladunk a pozíciókkal hátrafelé a pályán. Tehát csatár, középpályás, védő, kapus.

```

List<Jatekos> golovok = new List<Jatekos>();
for (int i = 0; i < golok_szama; i++)
{
    Jatekos goljelolt = null;
    while (goljelolt == null)
    {
        int valasztottJatekosIndex = rng.Next(jatekosok.Count);
        Jatekos valasztottJatekos = jatekosok[valasztottJatekosIndex];

        switch (valasztottJatekos.Pozicio)
        {
            case "Kapus":
                break;
            case "Védő":
                if (rng.Next(0, 10) == 0) //10% esély védőre
                {
                    goljelolt = valasztottJatekos;
                    golovok.Add(goljelolt);
                }
                break;
            case "Középpályás":
                if (rng.Next(0, 5) == 0) //20% esély középpályásra
                {
                    goljelolt = valasztottJatekos;
                    golovok.Add(goljelolt);
                }
                break;
            case "Csatár":
                if (rng.Next(0, 2) == 0) //50% esély csatárra
                {
                    goljelolt = valasztottJatekos;
                    golovok.Add(goljelolt);
                }
                break;
        }
    }
}

```

A legenerált játékosokat visszaadom a fő program résznek. Ahol eltárolom őket és kiíratom a többi adattal őket az adatbázisban, hogy ezt is megtudjam jeleníteni a weblapon a felhasználónak.

```

List<Jatekos> hazaigollovok = database.gol_lovo(rng, csapatok[0], eredmény[1]);
List<Jatekos> vendeggollovok = database.gol_lovo(rng, csapatok[1], eredmény[2]);

```

Ezután egy függvény segítségével kiíratom vagyis felülírom a már meccs lefutása előtt felvitt adatokat a mérkőzésről és így a mérkőzés lefutott.

```

database.adatupdate($"[{eredmény[1]}]-[{eredmény[2]]", hazaigollovok.Concat(vendeggollovok).ToList(), eredmény[1] + eredmény[2], database.getlastmeccs(),
Thread.Sleep(30000);

```

Ezután ugyan úgy alszik a program egy 30 másodpercet. Ez az idő lesz az új meccs közötti idő. Így néz ki a C# program felépítése és a meccsek és a meccsek eredményeinek generálása.

### 3.4.2 A weblap felépítése

- Regisztrációs oldal

A regisztrációs oldal egy nagyon fontos része lehet egy weboldalnak, hiszen ez teszi lehetővé az új felhasználók számára, hogy hozzáférjenek a weboldal szolgáltatásaihoz, és használják azokat. A regisztrációs oldal létrehozása során fontos, hogy a felhasználói élményt szem előtt tartsuk, és biztosítsuk a felhasználók számára a lehető legjobb felhasználói élményt. Az oldal

nagyon egyszerű és átlátható felépítéssel rendelkezik. Az űrlapban a felhasználók megadhatják a személyes adataikat, beleértve a felhasználónevüket, az e-mail címüket, a jelszavukat, az okmány számukat, a nemüket és az országukat. Az űrlap minden mezője kötelező, így biztosítva van, hogy a felhasználók minden szükséges információt megadnak. Az oldal a felhasználói adatokat biztonságosan kezeli, és a jelszavakat hash-elve tárolja az adatbázisban, hogy megakadályozza az illetéktelen hozzáférést. Az oldal az adatokat ellenőrzi a rendszerben található adatokkal, hogy megakadályozza a duplikátumokat. Ha a felhasználónév vagy az e-mail cím már foglalt, akkor egy üzenet jelenik meg, amely tájékoztatja a felhasználót erről. Ha minden adat helyesen van megadva, akkor a felhasználói adatokat az adatbázisba mentjük, majd a felhasználó átirányítva lesz a bejelentkezési oldalra. Az oldal a felhasználói regisztráció után automatikusan hozzárendel egy pénztárcát a felhasználóhoz. Ez a funkció nagyon fontos lehet egy weboldalon, hiszen lehetővé teszi a felhasználók számára, hogy vásároljanak a weboldalon található termékekből, és fizetéseket hajtsanak végre.

Az oldal felépítése és funkcionalitása nagyban befolyásolja a felhasználói élményt. Az átlátható felépítés és a biztonságos adatkezelés fontos szempontok, amelyeket mindig szem előtt kell tartani a regisztrációs oldal létrehozása során.

```
if (isset($_POST['regisztracio'])) {
    $felhasznalo_nev = $db->security($_POST['felhasznalo_nev']);
    $email = $db->security($_POST['email']);
    $password = password_hash($db->security($_POST['jelszo']), PASSWORD_DEFAULT);
    $okmany = $db->security($_POST['okmany']);
    $neme = $db->security($_POST['nem']);
    $orszag = $db->security($_POST['orszag']);
    //ellenőrzés
    $sql = "SELECT * FROM felhasznaloi_adatok WHERE felhaszn_nev = '$felhasznalo_nev'";
    $result = $db->RunSQL($sql);
    $F_ellenorzes = $result->fetch_assoc();

    $sql = "SELECT * FROM felhasznaloi_adatok WHERE email = '$email'";
    $result = $db->RunSQL($sql);
    $E_ellenorzes = $result->fetch_assoc();
    if ($F_ellenorzes != false) {
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translate(-50%, -50%);>';
        echo '<p>Az email-cím megfelelő, de a felhasználónév már foglalt!</p>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = none;});';
    }elseif ($E_ellenorzes != false) {
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translate(-50%, -50%);>';
        echo '<p>Az email-cím már foglalt!</p>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = none;});';
    }else {
        $hash = $db->Hash($felhasznalo_nev);
        $sql = "INSERT INTO felhasznaloi_adatok VALUES ('null', '$felhasznalo_nev', '$email', '$password', '$okmany', '$neme', '$orszag')";
        $result = $db->RunSQL($sql);
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translate(-50%, -50%);>';
        echo '<p>Sikeres regisztráció!</p>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = none;});';
        header('Location: login.php');
    }
}

$sql2 = "SELECT felhaszn_id FROM felhasznaloi_adatok ORDER BY felhaszn_id DESC LIMIT 1";
$result2 = $db->RunSQL($sql2);
$fid = $result2->fetch_assoc()['felhaszn_id'];
$sql1 = "INSERT INTO penztarca VALUES ('$fid', 'null', '0')";
$result = $db->RunSQL($sql1);
```



- Bejelentkezési oldal

Bejelentkezési űrlap tartalmazza a kellő php és html részeket, amely lehetővé teszi a felhasználók számára, hogy hozzáférjenek a weboldal szolgáltatásaihoz, miután sikeresen bejelentkeztek. Az űrlap az e-mail cím és a jelszó megadását kéri, majd ellenőrzi, hogy a felhasználói adatok helyesek-e. Az űrlap elküldése után a kód először ellenőrzi, hogy a „bejelentkezés” gombra kattintottak-e, majd a megadott e-mail cím alapján lekérdezi a felhasználó adatait az adatbázisból. Ezután összehasonlítja a megadott jelszót a felhasználói adatokkal, amelyeket hash-elve tároltak az adatbázisban. Ha a bejelentkezés sikeres volt, akkor a felhasználói adatokat elmenti a PHP \$\_SESSION változóiban, majd átirányítja a felhasználót a weboldal főoldalára. Ha a bejelentkezés sikertelen volt, akkor a kód egy modális ablakot jelenít meg, amely tájékoztatja a felhasználót arról, hogy az e-mail cím vagy a jelszó nem volt megfelelő. Ezután a modális ablakot automatikusan bezárja, hogy a felhasználói élményt ne zavarja meg. Az űrlap felépítése egyszerű és átlátható, ami lehetővé teszi a felhasználók számára, hogy gyorsan és könnyen bejelentkezzenek. Az űrlap funkciói közé tartozik a jelszó visszaállításának lehetősége, amely lehetővé teszi a felhasználók számára, hogy új jelszót állítsanak be, ha elfelejtették a régiét. Az átlátható felépítés és a biztonságos adatkezelés fontos szempontok, amelyeket mindig szem előtt kell tartani a bejelentkezési űrlap létrehozása során.

```
if (isset($_POST['bejelentkezés'])) {
    $email = $_POST['email'];
    $password = $_POST['jelszo'];

    $sql = "SELECT * FROM felhasznaloi_adatok INNER JOIN penztarca ON penztarca.felhasz_id = felhasznaloi_adatok.felhasz_id WHERE email = '$email'";
    $result = $db->runSQL($sql);
    $felhasznalo = $result->fetch_assoc();

    if ($felhasznalo != false && password_verify($password, $felhasznalo['jelszo'])) {
        $_SESSION['email'] = $felhasznalo['email'];
        $_SESSION['felhasz_nev'] = $felhasznalo['felhasz_nev'];
        $_SESSION['felhasz_id'] = $felhasznalo['felhasz_id'];
        $_SESSION['penztarca_id'] = $felhasznalo['penztarca_id'];
        header('Location: Foldal.php');
    } else {
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 20px; border: 1px solid #ccc; width: 400px; text-align: center;>';
        echo '<p>Sikertelen bejelentkezés!</p> <p>Az email-cím vagy a jelszó nem volt megfelelő!</p>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
    }
}
```

- Kijelentkezési oldal

A PHP kód az aktuális felhasználói munkamenetet zárja le és átirányítja a felhasználót a weboldal főoldalára. A session\_start() függvény az aktuális munkamenetet indítja el, amely lehetővé teszi a felhasználók számára, hogy hozzáférjenek a weboldal szolgáltatásaihoz, miután sikeresen bejelentkeztek. A session\_destroy() függvény az aktuális munkamenetet zárja le, és eltávolítja az összes munkameneti változót és adatot. Ezután a header() függvény átirányítja a felhasználót a weboldal főoldalára. Ez a kód hasznos lehet például akkor, ha a felhasználó ki

szeretne jelentkezni a weboldalról, vagy ha a felhasználónak ki kell jelentkeznie a biztonsági okokból. A munkamenet lezárása biztonságosabbá teszi a felhasználói adatokat, és megakadályozza az illetéktelen hozzáférést a felhasználói fiókhoz.

```
<?php
session_start();
session_destroy();
header('Location: Foldal.php');
?>
```

- Jelszó visszaállító oldal

Ez az oldal egy jelszóvisszaállító űrlapot tartalmaz, amely lehetővé teszi a felhasználók számára, hogy visszaállítsák a jelszavukat, ha elfelejtették azt. Az űrlap a felhasználó által megadott e-mail cím alapján ellenőrzi, hogy az e-mail cím létezik-e az adatbázisban. Ha az e-mail cím létezik, akkor az űrlap elküldi az e-mailt a felhasználónak, amely tartalmazza az új jelszó beállításához szükséges linket. Az első sor az `include('header.php')` függvényt tartalmazza, amely betölti a fejléc tartalmát. Az `isset()` függvénnyel ellenőrzi, hogy a felhasználó elküldte-e az űrlapot, majd a felhasználó által megadott e-mail cím alapján ellenőrzi, hogy az e-mail cím létezik-e az adatbázisban. Ha az e-mail cím létezik, akkor az űrlap elküldi az e-mailt a felhasználónak, amely tartalmazza az új jelszó beállításához szükséges linket. Az e-mail tartalma tartalmazza a felhasználó nevét, a linket az új jelszó beállításához, és a feladó e-mail címét. Ezután megjelenít egy modal üzenetet, amely megerősíti, hogy az e-mail sikeresen elküldésre került, majd néhány másodperc múlva eltűnik. Ha az e-mail cím nem létezik az adatbázisban, akkor az űrlap hibaüzenetet generál, és figyelmezteti a felhasználót arra, hogy az e-mail cím nincs regisztrálva. Ebben az esetben is megjelenik egy modal üzenet, amely megerősíti a hibaüzenetet, majd néhány másodperc múlva eltűnik. Az űrlap egyszerű és átlátható kialakítást kapott, amely lehetővé teszi a felhasználók számára, hogy könnyen használják, és megfelelően tájékozódjanak a jelszóvisszaállítás folyamatáról. A modal üzenetek használata tovább növeli az űrlap felhasználóbarát jellegét.

```

<?php
include('header.php');
if (isset($_POST['jelszovissza'])) {
    $visszaemail = $_db->security($_POST['reset_email']);
    $sql = "SELECT * FROM felhasznaloi_adatok WHERE email='\$visszaemail'";
    $result = $_db->RunSQL($sql);
    $felhasznalo = $result->fetch_assoc();

    if ($felhasznalo != false) {
        $to = $visszaemail;
        $subject = 'Password Reset';
        $message = '
        Elfelejtett jelszó

        -----
        Felhasználói név: ' . $felhasznalo['felhaszn_nev'] . '

        Kérjük, kattintson erre a linkre új jelszó beállításához:
        http://localhost/Fogadas_web/newpassword.php?email=' . $visszaemail . '&hash=' . $felhasznalo['hash'];
        $headers = 'from:admin@localhost.org';
        mail($to, $subject, $message, $headers);
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 10px; border: 1px solid #ccc;">
        echo '<p>Az E-mailt elküldtük a megadott e-mail címre</p></div>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
    } else {
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 10px; border: 1px solid #ccc;">
        echo '<p>Nem talált emailt!</p></div>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
    }
}
}
?>

```

- Új jelszó létrehozó oldal

Ez az oldal az új jelszó beállítását teszi lehetővé a felhasználók számára, akik elfelejtették a jelszavukat, és kértek egy új jelszó beállítását. Az űrlap megkapja az e-mail címet és a hash-kódot, amelyet az e-mail tartalmazott, amelyet korábban elküldtek a felhasználónak. Az űrlap ellenőrzi, hogy az e-mail cím és a hash-kód helyesek-e, majd lehetővé teszi a felhasználó számára, hogy új jelszót állítson be. Ha a felhasználó megadja az új jelszót, és az új jelszó egyezik a megerősítéssel, akkor az űrlap frissíti a felhasználó jelszavát az adatbázisban.

A frissítés után a felhasználó átirányításra kerül a bejelentkezési oldalra. Ha az új jelszó beállítása sikertelen, akkor az űrlap hibaüzenetet jelenít meg a felhasználó számára. A kód a MySQL adatbázis kapcsolatát és az adatbázis lekérdezéseket tartalmazza. Az űrlap biztonságos adatfeldolgozást használ, hogy megakadályozza a rosszindulatú szereplők általi bejelentkezést és az adatok manipulálását.

```

<?php
include('header.php');

$jelszo1 = $jelszo2 = "";
$email = $_db->security($_GET['email']);
isset($_GET['hash']) ? $hash = $_db->security($_GET['hash']) : $hash = "";
if (isset($_POST['megerosites'])) {
    $jelszo1 = $_db->security($_POST['jelszo1']);
    $jelszo2 = $_db->security($_POST['jelszo2']);
    if ($jelszo1 == $jelszo2) {
        $jelszo = password_hash($_POST['jelszo1'], PASSWORD_DEFAULT);
        $sql = "UPDATE felhasznaloi_adatok SET jelszo = '$jelszo' WHERE email = '$email' AND hash = '$hash'";
        $result = $_db->RunSQL($sql);

        if ($result > 0) {
            echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 10px; border: 1px solid #ccc;">
            echo '<p>Jelszó beállítva!</p></div>';
            echo '</div>';
            echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
            header('Location: login.php');
        } else {
            echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 10px; border: 1px solid #ccc;">
            echo '<p>A jelszó módosítása nem lehetséges!</p></div>';
            echo '</div>';
            echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
        }
    } else {
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 10px; border: 1px solid #ccc;">
        echo '<p>A jelszavak nem egyeznek!</p></div>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
    }
}
}
?>

```

- Profilt tartalmazó oldal

Ez a kód egy felhasználói profil megjelenítését teszi lehetővé, amely tartalmazza a felhasználó adatait, mint például a felhasználónév, az e-mail cím, a regisztráció dátuma, az okmány típusa, a nem és az ország. Az adatokat az adatbázisból lekéri a felhasználói azonosító alapján, és az űrlap megjeleníti a táblázat formájában. Az űrlap továbbá lehetővé teszi a felhasználó számára, hogy törölje a profilját. Ha a felhasználó erre a gombra kattint, akkor az űrlap törlődik az adatbázisból, és a felhasználó átirányításra kerül a kijelentkezési oldalra.

```
<div class="kozepre">
  <div class="table-container">
    <table class="table">
      <thead>
        <th>Felhasználó Név</th>
        <th>Email-cím</th>
        <th>Regisztráció Dátuma</th>
        <th>Okmány</th>
        <th>Neme</th>
        <th>Ország</th>
      </thead>
      <tbody>
        <tr>
          <th>
            <?= $adatok[0][0]; ?>
          </th>
          <td>
            <?= $adatok[0][1]; ?>
          </td>
          <td>
            <?= $adatok[0][2]; ?>
          </td>
          <td>
            <?= $adatok[0][3]; ?>
          </td>
          <td>
            <?= $adatok[0][4]; ?>
          </td>
          <td>
            <?= $adatok[0][5]; ?>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

- Pénztárca oldal

Ez az oldal egy webes alkalmazás részét képezi, amely lehetővé teszi a felhasználók számára, hogy pénzt helyezzenek el vagy vonjanak ki a saját pénztárcájukból. Amikor a felhasználó pénzt helyez el, meg kell adnia a kártyaszámát, a kártya érvényességi idejét, a CVV kódot és az összeget, amit befizetne. Ha a megadott érvényességi idő még nem járt le, és az összeg megfelelő (1500-2000000 forint között), akkor az alkalmazás frissíti a felhasználó egyenlegét a megadott összeggel. Ha a felhasználó pénzt szeretne kivenni, akkor szintén meg kell adnia a kártyaszámát, a kártya érvényességi idejét, a CVV kódot és az összeget, amit ki szeretne venni. Ha a megadott érvényességi idő még nem járt le, és az összeg megfelelő (10000-2000000 forint között), és a felhasználó rendelkezik elegendő egyenleggel, akkor az alkalmazás frissíti a felhasználó egyenlegét a kivett összeggel. Ha a felhasználó rossz adatokat ad meg, vagy az érvényességi idő már lejárt, az alkalmazás hibaüzenetet jelenít meg, majd automatikusan

eltűnteti azt néhány másodperc múlva. Van még rajta egy HTML kód, ami egy fizetési űrlapot jelenít meg. A felhasználó itt adhatja meg a bankkártyájának adatait és az összeget, amit befizetni vagy kifizetni szeretne. A kártyaszám, a kártyatulajdonos neve, az összeg, a lejárat dátum és a CVV mezők kitöltése kötelező. A kártyaszám mezőbe csak számokat lehet beírni, a kártyatulajdonos nevébe pedig csak betűket és szóközt. Az összeg mező minimum 50 és maximum 2000000 lehet. A lejárat dátum mezőbe csak számokat és a "/" karaktert lehet beírni, és a formátumnak "hh/ee" -nek kell lennie. A CVV mezőbe csak számokat lehet beírni, maximum 3 karakter hosszúságban. A "Pénz befizetés" és "Pénz kifizetés" gombokkal a felhasználó tudja elindítani a tranzakciót. A jobb oldalon megjelenik egy képzeletbeli bankkártya, amelyen szintén láthatók a megadott adatok. Alul látható egy JavaScript fájl hivatkozása és a footer.php fájl beillesztése is.

```
<?php include('header.php');
$sql6 = "SELECT egyenleg FROM penztarca WHERE felhasz_id = " . $_SESSION['felhasz_id'] . " ";
$result6 = $db->runSQL($sql6);
$result6 = $result6->fetch_assoc()['egyenleg'];

if (isset($_POST['penzbe'])) {
    $sql2 = "SELECT penztarca_id FROM penztarca WHERE penztarca_id = " . $_SESSION['penztarca_id'] . " ";
    $result2 = $db->runSQL($sql2);
    $fid = $result2->fetch_assoc()['penztarca_id'];
    $kartyaszam = $db->security($_POST['kartyaszam']);
    $kartyat = $db->security($_POST['kartyat']);
    $osszeg = $db->security($_POST['osszeg']);
    $valid = $db->security($_POST['valid-thru-text']);
    $cvv = $db->security($_POST['cvv-text']);
    $szetvalid = explode('/', $valid);
    $expires = \DateTime::createFromFormat('my', $szetvalid[0] . $szetvalid[1]);
    $now = new \DateTime();

    if ($expires > $now) {
        if ($osszeg >= 1500 && $osszeg <= 2000000) {
            $sql1 = "INSERT INTO be_ki_fizetes VALUES (NULL, '$fid', " . date("Y-m-d H:i:s") . ", '$kartyaszam', '$valid', '$cvv', '+$osszeg', '$kartyat')";
            $result = $db->runSQL($sql1);
            $sql = "UPDATE penztarca SET egyenleg = egyenleg + $osszeg WHERE penztarca_id = '$fid'";
            $result = $db->runSQL($sql);
            header('Location:Foldal.php');
        }
    }
}
```

```
if (isset($_POST['penzki'])) {
    $sql2 = "SELECT penztarca_id FROM penztarca WHERE penztarca_id = " . $_SESSION['penztarca_id'] . " ";
    $result2 = $db->runSQL($sql2);
    $fid = $result2->fetch_assoc()['penztarca_id'];
    $kartyaszam = $db->security($_POST['kartyaszam']);
    $kartyat = $db->security($_POST['kartyat']);
    $osszeg = $db->security($_POST['osszeg']);
    $valid = $db->security($_POST['valid-thru-text']);
    $cvv = $db->security($_POST['cvv-text']);
    $szetvalid = explode('/', $valid);
    $expires = \DateTime::createFromFormat('my', $szetvalid[0] . $szetvalid[1]);
    $now = new \DateTime();

    if ($expires > $now) {
        if ($osszeg <= $result6) {
            if ($osszeg >= 10000 && $osszeg <= 2000000) {
                $sql = "INSERT INTO be_ki_fizetes VALUES (NULL, '$fid', " . date("Y-m-d H:i:s") . ", '$kartyaszam', '$valid', '$cvv', '-$osszeg', '$kartyat')";
                $result = $db->runSQL($sql);
                $sql3 = "UPDATE penztarca SET egyenleg = egyenleg - $osszeg WHERE penztarca_id = '$fid'";
                $result = $db->runSQL($sql3);
                header('Location:Foldal.php');
            }
        }
    }
}
```

```
<script src="js/penztarca.js"></script>
<?php include('footer.php'); ?>
```

- Pénztárca JavaScript felépítése

Ez a JavaScript kód egy online fizetési űrlap számára íródott. A kód az űrlapon megadott kártyaszám, kártyatulajdonos neve, lejárat dátum és CVV kód formázásáért felelős. A kártyaszám mező automatikusan formázódik úgy, hogy az első 12 számmal együtt az elválasztó szóközők is megjelennek. Az utolsó 4 szám beírásakor szintén automatikusan hozzáadódnak az elválasztó szóközők. A kártyatulajdonos neve nagybetűsre változik, amint beírják. A lejárat dátum formátuma "hónap/év" lesz, és a beírt adatok formázódnak, hogy ezt a formátumot kövessék. A CVV kód beírása után azonnal megjelenik a kód az űrlapon. Ez a kód egy nagyon hasznos funkcióval látja el a fizetési űrlapot, amely segít a felhasználóknak könnyebben és pontosabban kitölteni az űrlapot.

```
const form = document.querySelector("#credit-card");

const cardNumber = document.querySelector("#card-number");
const cardHolder = document.querySelector("#name-text");
const cardExpiration = document.querySelector("#valid-thru-text");
const cardCVV = document.querySelector("#cvv-text");

const cardNumberText = document.querySelector("#number-v1");
const cardHolderText = document.querySelector("#name-v1");
const cardExpirationText = document.querySelector("#expiration-v1");
const cardCVVText = document.querySelector("#cvv-v1");

cardNumber.addEventListener("keyup", (e) => {
  if (!e.target.value) {
    cardNumberText.innerHTML = "1234 5678 9101 1121";
  } else {
    const valuesOfInput = e.target.value.replaceAll(" ", "");

    if (e.target.value.length > 14) {
      e.target.value = valuesOfInput.replace(/(\d{4})(\d{4})(\d{4})(\d{0,4})/, "$1 $2 $3 $4");
      cardNumberText.innerHTML = valuesOfInput.replace(/(\d{4})(\d{4})(\d{4})(\d{0,4})/, "$1 $2 $3 $4");
    } else if (e.target.value.length > 9) {
      e.target.value = valuesOfInput.replace(/(\d{4})(\d{4})(\d{0,4})/, "$1 $2 $3");
      cardNumberText.innerHTML = valuesOfInput.replace(/(\d{4})(\d{4})(\d{0,4})/, "$1 $2 $3");
    } else if (e.target.value.length > 4) {
      e.target.value = valuesOfInput.replace(/(\d{4})(\d{0,4})/, "$1 $2");
      cardNumberText.innerHTML = valuesOfInput.replace(/(\d{4})(\d{0,4})/, "$1 $2");
    } else {
      cardNumberText.innerHTML = valuesOfInput;
    }
  }
});

cardHolder.addEventListener("keyup", (e) => {
  if (!e.target.value) {
    cardHolderText.innerHTML = "Nagy János";
  } else {
    cardHolderText.innerHTML = e.target.value.toUpperCase();
  }
});
```

- Header oldal

A kód az include('database.php') sorral betölti az adatbáziskapcsolatot biztosító database.php fájlt. Ezután a session\_start() függvény segítségével inicializálja a munkamenetet, és ellenőrzi, hogy van-e bejelentkezett felhasználó. Ha van, akkor lekérdezi az aktuális egyenlegüket a penztarca táblából, majd megjeleníti a navigációs gombokat és a felhasználó nevét és egyenlegét. Ha nincs bejelentkezett felhasználó, akkor csak a "Regisztráció" és a "Bejelentkezés" gombok jelennek meg. A felhasználói adatok és a fogadások lekérdezése a fogadas, fogadasi\_lehetoseg, meccs\_eredmeny és nemzetek táblákból történik. A fogadas tábla tartalmazza a felhasználó által leadott fogadásokat, a fogadasi\_lehetoseg tábla tartalmazza az

egyes fogadási lehetőségeket és azokhoz tartozó odds-okat, a `meccs_eredmeny` tábla tartalmazza a meccsek eredményeit, míg a `nemzetek` tábla tartalmazza a meccsekben résztvevő nemzetek adatait. Végül, ha van bejelentkezett felhasználó, akkor a "Fogadásaim" gombra kattintva a felhasználó által leadott fogadásokat lehet megtekinteni a `fogadasok.php` oldalon. A "Pénztárca" gombra kattintva pedig az aktuális egyenleg és a pénzmozgások adatai tekinthetők meg a `penztarca.php` oldalon. A "Profil" gomb a felhasználó profiljának szerkesztésére vezet a `profil.php` oldalon.

- Footer oldal

Ez a kód a weboldal láblécét jelenti. A footer elemen belül a szerzői jogi információ jelenik meg, amely tartalmazza a szerző nevét és a weboldal nevét. Egy többszintű elrendezés része, mint például egy container vagy wrapper elem, amely tartalmazza az összes oldaltartalmat. A lábléc fontos eleme a weboldalnak, mivel a felhasználók gyakran az oldal alján keresik a kapcsolatfelvételi és jogi információkat. A szerzői jogi információ megjelenítése fontos az intellektuális tulajdon védelme és a jogi követelmények teljesítése szempontjából. A &copy; szimbólum az "©" karaktert jelenti, amely a szerzői jogi védelem jele. A kód a weboldal teljesítése érdekében fontos, és hozzájárul az oldal összképéhez.

- Fogadások oldal

Ez a PHP kód a `fogadasok.php` oldal tartalmát határozza meg. Az oldal felső részén egy cím, majd két táblázat jelenik meg, amelyek a futó és lefutott fogadásokat tartalmazzák. A táblázat oszlopai a csapatok, a tét, a profit/bukó, a szorzó és a fogadás neve. A táblázatok a `$result1` és `$result` változók értékei alapján jönnek létre, amelyek a fogadás táblában tárolt futó és lefutott fogadásokat tartalmazzák. Az adatbázisból történő adatlekérdezés az `$db` objektum `RunSQL()` metódusának meghívásával történik, amely visszaadja az eredményt egy asszociatív tömbként. A `foreach` ciklus segítségével az összes fogadás adatai megjelennek a táblázatban. A ciklus minden egyes iterációja során a kód lekéri a meccsben résztvevő hazai és vendég csapatok nevét a `nemzetek` táblából. Ezután az adatokat hozzáadja a táblázat soraihoz. Az alsó részen található lábléc a szerzői jogi információt tartalmazza. Ez fontos az intellektuális tulajdon védelme és a jogi követelmények teljesítése szempontjából. Az oldal alján található JavaScript kód az oldal automatikus frissítését biztosítja. Az `XMLHttpRequest` objektum segítségével az oldal frissítése nélkül tölthető be az új adat, és a `setInterval` függvény segítségével az oldal 1 másodpercenként automatikusan frissül. Ez hasznos lehet, ha a felhasználónak gyors és friss információra van szüksége a fogadásokkal kapcsolatban.

```

<script>
function refresh() {

    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
        document.getElementById("refresh").innerHTML = this.responseText;
    }

    xhttp.open("GET", "fogadasok.php");
    xhttp.send();
}
refresh();
setInterval(function() {
    refresh();
}, 1000);
</script>

```

```

<table class="table">
<thead>
<th>Csapatok</th>
<th>Tét</th>
<th>profit/buko</th>
<th>Szorzó</th>
<th>Fogadás neve</th>
<th>Eredmény</th>
</thead>
</thead>

<tr>
<?php foreach ($result as $value) {
    $sql3 = "SELECT * FROM nemzetek WHERE nemzet_id = '" . $value['hazai_id'] . "'";
    $result3 = $db->RunSQL($sql3);
    $sql4 = "SELECT * FROM nemzetek WHERE nemzet_id = '" . $value['vendeg_id'] . "'";
    $result4 = $db->RunSQL($sql4);
    $hazai_nev = $result3->fetch_assoc()['nemzet_nev'];
    $vendeg_nev = $result4->fetch_assoc()['nemzet_nev'];
}
<th>
<? $hazai_nev; ?> --
<? $vendeg_nev; ?>
</th>
<td>
<? $value['fogadasi_osszeg']; ?>
</td>
<td>
<? $value['profit_buko']; ?>
</td>
<td>
<? $value['odds']; ?>
</td>
<td>
<? $value['fogadas_neve']; ?>
</td>
<td>
<? $value['eredmeny']; ?>
</td>
</tr>
<?php }
?>
</table>

```

- fogadások JavaScript

Ez egy JavaScript kódrészlet, amely egy modal ablakot vezérel. A kód figyeli az egérgattintásokat, és ha azok a modal ablakon kívül esnek, akkor bezárja az ablakot. A kódban található openBetModal függvény segítségével megnyitható a modal ablak, amelyben egy adott csapat, egy odd és egy value2 értékek jelennek meg. Az ablakon belül lehetőség van a csapatra történő fogadásra. Az ablak bezárásához szükséges gombra kattintva az ablak eltűnik.

- Content mappa és content fájlok

Ezek a php oldalakon vannak az adatok betöltése a Főoldalra és a modal modulba mikor fogadni akar a felhasználó. Ezek mellett néhány lekérdezés szerepel itt, amely szükséges az oldal működéséhez.

- Adatbázis oldal

Ez egy PHP osztály, amely adatbázis kapcsolatot hoz létre és biztonsági funkciókat nyújt. Az osztályban található security metódus a kapott adatokat HTML entitásokká alakítja, így védi a weboldalt a cross-site scripting (XSS) támadásoktól. A Hash metódus SHA512 titkosítási algoritmust használ a kapott adatok titkosítására. Az osztály általános adatbázis műveleteket is tartalmaz, mint például a RunSQL metódus, amely végrehajt egy SQL utasítást, és visszaadja a lekérdezés eredményét vagy az érintett sorok számát. Az RunSQLPrms metódus ugyanazt teszi, mint az RunSQL, de lehetővé teszi a paraméterek használatát a lekérdezésben.



- Főoldal

```
if (isset($_POST['kuld'])) {
    if (isset($_SESSION['felhasz_id']) ? $_SESSION['felhasz_id'] = $db->security($_SESSION['felhasz_id']) : $_SESSION['felhasz_id'] = "") {
        $sql1 = "SELECT meccs_id, lefutott_e FROM meccs_eredmeny ORDER BY meccs_id DESC LIMIT 1";
        $result1 = $db->RunSQL($sql1);
        $result1 = $result1->fetch_assoc();
        if ($result1['lefutott_e'] == 0) {
            $osszeg = $_POST['betAmount'];
            $fid = $_POST['fid'];
            $odd = $_POST['odd'];
            $sql6 = "SELECT egyenleg FROM penztarca WHERE felhasz_id = " . $_SESSION['felhasz_id'] . " ";
            $result6 = $db->RunSQL($sql6);
            $egyenleg = $result6->fetch_assoc()['egyenleg'];
            if ($osszeg <= $egyenleg && $osszeg >= 50) {
                $sql = "INSERT INTO 'fogadas' ('felhasz_id', 'fogadasi_osszeg', 'profit_buko', 'meccs_id', 'fogadasi_szam', odds) VALUES ('" . $_SESSION['felhasz_id'] . "', '" . $osszeg . "', '0', '" . $meccs_id . "', '" . $fogadasi_szam . "', '" . $odd . "')";
                $result = $db->RunSQL($sql);
                $sql3 = "UPDATE penztarca SET egyenleg = egyenleg - $osszeg WHERE felhasz_id = '" . $_SESSION['felhasz_id'] . "'";
                $result3 = $db->RunSQL($sql3);
                header('Location: fogadasok.php');
            } else {
                echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 20px; border: 1px solid #ccc; text-align: center;">';
                echo '<p>Nincs elegendő pénze megrakni a fogadást!</p>';
                echo '</div>';
                echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
            }
        } else {
            echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 20px; border: 1px solid #ccc; text-align: center;">';
            echo '<p>Már lefutott a meccs!</p>';
            echo '</div>';
            echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
        }
    } else {
        echo '<div id="modal" style="position: fixed; top: 20%; left: 50%; transform: translateX(-50%); background-color: #fff; padding: 20px; border: 1px solid #ccc; text-align: center;">';
        echo '<p>Nem vagy bejelentkezve!</p>';
        echo '</div>';
        echo '<script>setTimeout(function(){document.getElementById(\'modal\').style.display = \'none\';}, 4000);</script>';
    }
}
```

Az oldal első részében egy űrlap található, amely lehetővé teszi a felhasználók számára, hogy fogadásokat helyezzenek el a meccsek eredményére. Az űrlap tartalmazza az összeget, amelyet a felhasználó fogadni szeretne, valamint az egyes meccsek eredményeire vonatkozó fogadási számokat és szorzókat. Az űrlap elküldése után a PHP kód ellenőrzi az adatokat, és ha valami nincs rendben, hibaüzenetet jelenít meg. Ha az adatok megfelelőek, akkor a fogadás az adatbázisba kerül mentésre, és az egyenleg csökkentése is megtörténik. Az oldal második részében egy AJAX kérés található, amely frissíti az oldalon megjelenő adatokat. Az AJAX kérés lekérdezi az adatokat az adatbázisból, majd dinamikusan frissíti az oldalon megjelenő adatokat. Az oldal tartalmaz egy gombot is, amely lehetővé teszi a felhasználók számára, hogy manuálisan frissítsék az oldalon megjelenő adatokat. Az oldal harmadik részében az egyes meccsek eredményeinek frissítése látható. Az oldal különböző szorzókat is tartalmaz, amelyeket a felhasználók használhatnak a fogadásaikhoz. Az oldal JavaScript kódot is tartalmaz, amely az AJAX kérésekkel frissíti az oldalon megjelenő adatokat. Az oldal egy egyszerű és hatékony felületet biztosít a felhasználók számára, hogy fogadásokat helyezzenek el és kövessék nyomon a meccsek eredményét.

```

<?php include('footer.php'); ?>
<script>
function refresh() {
    for (let index = 1; index < 7; index++) {
        const xhttp = new XMLHttpRequest();
        xhttp.onload = function () {
            document.getElementById("refresh" + index).innerHTML = this.responseText;
        }
        xhttp.open("GET", "content/content" + index + ".php");
        xhttp.send();
    }
}
refresh();
setInterval(function () {
    refresh();
}, 500);
</script>

```

```

<main>
<div class="kozos">
    <div class="mecssek" id="mecssek">
        <div class="grid-container">
            <div class="col1" id="refresh1">
                </div>
            <div class="col2" id="refresh2">
                <h1>Meccs adatai mindjárt frissülnek.</h1>
            </div>
            <div class="col3" id="refresh3">
                </div>
        </div>
    </div>
    <div class="mecssek2">
        <h1>Szorzók</h1>
        <div class="grid-container-szorok">
            <div class="col1-szorok">
                <span id="refresh4">
                    </span>
            </div>
            <div class="col2-szorok">
                <span id="refresh5">
                    </span>
            </div>
            <div class="col3-szorok">
                <span id="refresh6">
                    </span>
            </div>
        </div>
    </div>
</div>

```

- Css rész

Az oldal CSS fájlja az oldal kinézetéért és elrendezéséért felelős. Az oldal stílusát egyszerű és modern módon definiálja, amely megegyezik a weboldal témájával és céljával. Az oldal betűtípusa és háttérszíne harmonizál, és a fejléc, a fő tartalom és a lábléc is jól elkülönül egymástól. Az űrlapok és a gombok is stílusosan vannak megjelenítve, hogy a felhasználók könnyen megtalálják a szükséges információkat és funkciókat. A CSS fájl továbbá tartalmazza a responsive design szabályait is, amely lehetővé teszi az oldal megfelelő megjelenítését a különböző eszközökön, így a felhasználók könnyen böngészhetik az oldalt bármilyen készüléken.

## 3.5 Tesztelési dokumentáció

Tesztelendő funkciók:

1. Bejelentkezés & Regisztráció:

Bejelentkezés gomb, Regisztrációs gomb, Beviteli mezők, Jelszó visszaállítás, Új jelszó beállítása.

2. Főoldal:

Az összes menü gomb a megfelelő helyre vigyen, az adatok megfelelő megjelenítése.

### 3. Fogadás tétel:

Kiválasztott fogadásnál megfelelő adatokkal felugró ablak. A megtett fogadás adatainak megfelelő eltárolása. A fogadási összeg levonása. A megfelelő ellenőrzések működése.

Ha nyertes a tipp akkor a pénztárca növelése a megfelelő értékkel.

### 4. Fogadás ellenőrzés:

Felhasználó látja az előző 10 fogadását és az éppen futó fogadást is. Az adatok megfelelőek.

Frissülnek az adatok a meccs lefutásakor.

### 5. Pénztárca működése:

Pénz feltöltés és kifizetés megfelelő működése. Az ellenőrizendő adatok ellenőrzése.

Adatok frissülése.

### 6. Profil oldal:

A profil adatok megfelelő megjelenítése. A fiók törlés gomb megfelelő működése.

### 7. Kijelentkezés:

A kijelentkezés gomb megfelelő működése.

### 8. Hibás adat megadása:

A megfelelő hibaüzenet megjelenítése felugró ablakban.

## 3.5.1 Tesztelés és eredmények

### 1. Bejelentkezés & Regisztráció:

- Elvárt kimenet: Regisztráció és Bejelentkezés megfelelően működik. A beviteli mezők is átadják az adatokat az adatbázisnak. Jelszó visszaállító email is megfelelően működik és az új jelszó beállítása is. A kelendő ellenőrzések is működnek.

### 2. Főoldal:

- Elvárt kimenet: A gombok megfelelően működnek arra az oldalra irányítanak át ahova kell és az a megfelelő adatokat jeleníti meg az oldal.

### 3. Fogadás tétel:

- A felugró ablak a jó adatokat viszi át. A fogadás eltárolódik és az összeget levonja és ha nyertes a tipp bővíti az értéket. Az ellenőrzések is megfelelően működnek.

### 4. Fogadás ellenőrzés:

- A fogadás előzményei jól mutatják az adatokat. Az éppen futó fogadások is automatikusan frissülnek. Az adatok megfelelően frissülnek.

### 5. Pénztárca működése:

- A pénz feltöltés és kifizetés teljesen az elvártként működik. Hozzáadja vagy levonja a kívánt összeget a pénztárcából. Az adatok ellenőrzése is hibátlan.
6. Profil oldal:
- A profil oldalon az adatok megfelelően vannak kiírva. A felhasználó képes törölni a fiókját.
7. Kijelentkezés:
- A felhasználó képes kijelentkezni. Ezután az oldalon tud maradni és újra bejelentkezni.
8. Hibás adat megadása:
- A hibásan megadott adatok esetén az oldal mindig közli a felhasználóval, hogy valamit nem jól adott meg.

### 3.6 További fejlesztési lehetőségek

Az oldal bővíthet, mint pl.: a gólt szerző játékosra való fogadás vagy a sárga és piros lapok generálása egy meccsen és ezekre való fogadás. Későbbiekben bővíthető lenne az oldal, hogy egyszerre több meccs fusson és több meccsre lehessen fogadni. Külön statisztikai oldal rész, ahol kereséssel ellátva a legtöbb meccset vissza lehessen kérdezni az adatbázisból. Több fogadási lehetőség bevitele. A felhasználó tudjon egy kieséses szakaszt létrehozni saját csapatokkal és ezekre fogadni.

## 4 Összegzés

A webalkalmazás fejlesztése alatt HTML, Css, JavaScript és PHP ismereteimet bővítettem. Az oldalhoz szükséges programmal a C# ismereteimet bővítettem. Élveztem az ezzel eltöltött időt. Nagyon sok dolgot tanultam és sok érdekességet tanultam, hogy is működnek bizonyos dolgok.

## 5 Irodalomjegyzék

A dokumentáció elkészítéséhez szerzett ismeret:

<https://infojegyzet.hu/webszerkesztes/dokumentacio/>

A weboldalhoz szükséges elemek ismerete:

<https://www.w3schools.com>

A PHP-hoz használt dokumentáció:

<https://www.php.net/manual/en/>

A C# használt dokumentáció:

<https://learn.microsoft.com/en-us/dotnet/csharp/>