

**BGÉSZC**

**GANZ ÁBRAHÁM** **KÉT TANÍTÁSI NYELVŰ TECHNIKUM**

**Vizsgaremek**

**Lil-Notiz**

Készítette: Pribilla Zalán

Elkán Efraim

Témavezető: Szalainé Török Edit

Méhes József

Budapest, 2025

**Tartalomjegyzék**

[Bevezetés 3](#_Toc190866399)

[Témaválasztás indoklása 3](#_Toc190866400)

[Célkitűzés 3](#_Toc190866401)

[Kiknek szánjuk a weboldalt 3](#_Toc190866402)

[Fejlesztői dokumentáció 5](#_Toc190866403)

[Fejlesztői környezet 6](#_Toc190866404)

[Kialakított adatszerkezet és részletes bemutatása 6](#_Toc190866405)

[Algoritmusok a backend és frontend megvalósításban 6](#_Toc190866406)

[Tesztelés leírása, tesztdokumentáció 6](#_Toc190866407)

[Fejlesztési lehetőségek 7](#_Toc190866408)

[Felhasználói dokumentáció 8](#_Toc190866409)

[Üdvözlet! 8](#_Toc190866410)

[Weboldal használatának részletes ismertetése 9](#_Toc190866411)

[Összefoglalás 10](#_Toc190866412)

[Elkészült munkánk értékelése 10](#_Toc190866413)

[Köszönetnyilvánítás 11](#_Toc190866414)

[Irodalomjegyzék 12](#_Toc190866415)

## Bevezetés

### Feladatok felosztása

Pribilla Zalán:

-Adatbázis tervezés

-Bejelentkezés oldal backend

-Főoldal backend,frontend

-API

Elkán Efraim

-Adatbázis tervezés

-Regisztráció backend,frontend

-Projekt létrehozása oldal backend,frontend

-Főoldal frontend

### Témaválasztás indoklása

A **Lil’Notiz** webalkalmazást azért találtuk ki, hogy a csoportos projektek munkafelosztását, illetve a vele járó részleteket digitálisan is le lehessen bonyolítani, ha a távolság vagy éppenséggel az idő nem kedvez a felhasználóinknak.

Programozóként, szoftverfejlesztőként sok esetben szükséges több emberrel együtt dolgozni egy projekt munka véglegesítése érdekében, azonban megtörténhet, hogy a távolság a csapattagok között nem kedvez, vagy szorít egy határidő, így a személyes megjelenés a tervezés érdekében időt pazarolhat. A **lil’Notiz** webes alkalmazás segíthet időt megtakarítani, mivel digitálisan, online lehet egyszerű kezelhetőséggel megtervezni egy projektmunka menetét.

-Léteznek már hasonló weboldalak, amelyek segítenek egy projektmunka tervezésében, illetve annak haladásának a nyomon követésén, viszont magyarul nem találtunk ilyet, illetve általában vásárlást igényel a weboldal minden szolgáltatásainak kiélvezésére

### 

### Célkitűzés

A **Lil'Notiz** egy könnyen kezelhető, **Blazor** alapú feladatkezelő és projektmenedzsment alkalmazás, amely lehetővé teszi a felhasználók számára, hogy jegyzeteket (**Notizokat**) hozzanak létre, módosítsanak és rendszerezetten tároljanak. Az alkalmazás célja, hogy egyszerű és hatékony módot biztosítson a különböző projektekhez kapcsolódó feladatok nyomon követésére, státuszuk frissítésére és a csapattagok közötti együttműködés elősegítésére.

A rendszer támogatja a **NotizID-k** alapján történő projektkezelést, amely lehetővé teszi a felhasználók számára, hogy könnyedén csatlakozzanak meglévő projektekhez, vagy saját feladataikat egyedi azonosítóval rendszerezzék. Az alkalmazás emellett biztosítja a jegyzetek módosítását, törlését, és egy dinamikusan frissülő felületet kínál az adatok kezelésére.

A projekt célja, hogy a hagyományos jegyzetfüzetek és bonyolult projektmenedzsment szoftverek helyett egy **letisztult, gyors, és hatékony megoldást nyújtson**, amely mind az egyéni, mind a csapatmunkában részt vevő felhasználók számára megfelelő választás lehet.

### Kiknek szánjuk a weboldalt

A Lil' Notiz webalkalmazás célja, hogy minél szélesebb felhasználói kör számára nyújtson hatékony segítséget a feladatok szervezésében és nyomon követésében. Az alábbi felhasználói csoportok számára különösen előnyös lehet:

* **Idősebb korosztály** – Egyértelmű és átlátható rendszere miatt könnyen kezelhető azok számára is, akik nem jártasak a modern digitális eszközökben.
* **Fiatalabb felhasználók** – Kevés száraz, hosszú szöveg, egyszerű dizájn és gyors működés miatt a fiatalabb korosztálynak is szimpatikus és élvezetes a használata.
* **Informatikusok és fejlesztők** – A projektmenedzsment támogatása révén jól alkalmazható fejlesztési munkák, csapatszintű tervezés és feladatkiosztás során is.
* **Hétköznapi emberek** – Ideális a mindennapi feladatok rendszerezésére is, például autószerelési lépések, költözés megtervezése vagy otthoni teendők listázása esetén.
* **Kisvállalkozók és szervezők** – Komplex projektek, például egy nagyobb felhőkarcoló megépítésének összeszervezése is lekövethető és átlátható benne – na jó, kicsit túlzás, de akár ez is!

## Fejlesztői dokumentáció

### Fejlesztői környezet

A **Lil'Notiz** fejlesztése során a modern webes technológiákat és keretrendszereket alkalmaztuk annak érdekében, hogy egy gyors, biztonságos és könnyen kezelhető alkalmazást hozzunk létre. A fejlesztői környezet főbb elemei a következők:

* **Frontend:** Blazor WebAssembly Standalone App – A felhasználói felület kialakításához a Blazor WebAssembly technológiát használtuk, amely lehetővé teszi a gyors és interaktív böngészőalapú működést C# és .NET segítségével.
* **Backend:** ASP.NET Core Web API– A szerveroldali logikát és az adatkezelést egy stabil és skálázható ASP.NET Core Web API backend biztosítja.
* **Adatbázis:** MSSQL – A felhasználói adatok és jegyzetek biztonságos tárolása érdekében.
* **Kommunikáció:** REST API – A frontend és backend közötti adatkapcsolat biztosítására JSON-alapú REST API-kat alkalmazunk.
* **Hitelesítés és jogosultságkezelés:** Entity Framework – A felhasználók biztonságos bejelentkezéséhez és jogosultságkezeléséhez.
* **JavaScript interop:** A Blazor és JavaScript közötti kommunikációhoz, például DOM-manipulációhoz és külső könyvtárak integrálásához.

A fejlesztéshez **Visual Studio** és **Visual Studio Code** környezetet használunk, valamint **Google Drive** verziókezelőt az együttműködés és a kódverziók nyomon követése érdekében. Az alkalmazás könnyen telepíthető és futtatható különböző környezetekben, beleértve a helyi fejlesztői gépeket és felhőalapú megoldásokat.

### 

#### Domain és tárhely

#### Github és Git környezet

### Kialakított adatszerkezet és részletes bemutatása

### Adatbázis táblái:

### **Felhasználók tábla:**

1. **ID**
   * Egyedi azonosító minden felhasználóhoz.
2. **felhasznaloNev**
   * A bejelentkezéshez használt felhasználónév.
3. **jelszo**
   * A felhasználó jelszava (itt egyszerű példa szerepel, de valójában titkosítva kellene tárolni).
4. **email**
   * A felhasználó email címe.
5. **FELHASZNALONOTIZID**
   * Azoknak a jegyzeteknek az ID-ja, amik egy felhasználónak vannak meg



### **Notiz tábla (feladatok)**

1. **ID**
   * Egyedi azonosító minden feladathoz.
2. **NOTIZID**
   * Kapcsolódik a felhasználók táblájához (FELHASZNALOID), ez a közös azonosító.
3. **Tulajdonos**
   * A személy neve, akihez a feladat tartozik.
4. **Feladat**
   * A konkrét teendő neve (pl. kerékcsere, szélvédőcsere, stb.).
5. **Allapot**
   * A feladat aktuális állapota:
     + „Kész” = befejezett,
     + „Nincs elkezdve” = még el sem kezdődött,
     + „Elakadtam” = elkezdődött, de megakadt.
6. **MikorKezdje**
   * A feladat kezdési dátuma (tervezetten).
7. **HatarIdo**
   * A feladat határideje, meddig kell(ene) elkészülnie.
8. **Fontossag**
   * A feladat fontossági szintje:
     + „Sürgős”,
     + „Hanyagolható”,
     + „Fontos”.
9. **Komment**
   * Megjegyzés, egyéb információ a feladathoz.
10. **ProjektNev**
    * A projekt neve, amelyhez a feladat tartozik (itt: „autoszerelés”).



### Algoritmusok a backend és frontend megvalósításban

### Tesztelés leírása, tesztdokumentáció

### **Bejelentkezés tesztelése**

A bejelentkezési funkció tesztelése során az alábbi teszteseteket hajtottuk végre, manuális módon, különböző érvényes és érvénytelen bemenetekkel. A cél annak ellenőrzése volt, hogy a rendszer megfelelően kezeli a hitelesítési folyamatot, biztonságosan bánik az adatokkal, és felhasználóbarát módon reagál a hibás próbálkozásokra.

#### Teszteset: Sikeres bejelentkezés

* **Cél**: Ellenőrizni, hogy az érvényes felhasználónév/jelszó párosítás esetén a felhasználó sikeresen belép az alkalmazásba/weboldalra.
* **Bemenet**: Valós felhasználónév és hozzá tartozó jelszó.
* **Várt eredmény**: Átirányítás a főoldalra bejelentkezett állapotban.

#### Teszteset: Hibás jelszó

* **Cél**: Vizsgálni, hogyan reagál a rendszer hibás jelszó esetén.
* **Bemenet**: Létező felhasználónév, helytelen jelszó.
* **Várt eredmény**: Hibaüzenet („Hibás felhasználónév vagy jelszó”), belépés megtagadása.

#### Teszteset: Nem létező felhasználónév

* **Bemenet**: Nem létező felhasználónév, tetszőleges jelszó.
* **Várt eredmény**: Ugyanaz a hibaüzenet jelenik meg, mint helytelen jelszónál (biztonsági okokból), bejelentkezés meghiúsul.

#### Teszteset: Üres mezők

* **Bemenet**: Üres felhasználónév és/vagy jelszó mező.
* **Várt eredmény**: A bejelentkezés gomb inaktív marad, vagy hibaüzenetet dob a mező(k) kitöltésének hiányáról.

### **A főoldalnál egy újabb notiz léterhozásánál teszteltük érvényes iletve érvénytelen bemenettel. A cél az, hoyg mindenéleképpen legyen egy projektnév az adott projektnek, és ha nem ad a felhasználó projektnevet, akkor ne tudjon egy újat csinálni.**

### **Teszteset 1: Projekt neve megadása nélkül – hibaüzenet megjelenítése**

**Cél**: Annak ellenőrzése, hogy a rendszer nem engedi tovább a felhasználót, ha a „Projekt neve” mező üres marad a NOTIZ létrehozásakor.

#### **Lépések**:

1. Felhasználó a főoldalon rákattint a **„Notiz létrehozása”** gombra.
2. A rendszer átirányítja a projekt létrehozó oldalra.
3. A felhasználó **nem ad meg projektnevet**, csak rákattint a „Tovább” gombra.

#### **Elvárt eredmény**:

* A rendszer **nem engedi tovább** a felhasználót.
* Megjelenik a következő hibaüzenet:  
  **„Kérlek adj meg egy projektnevet.”**

### **Teszteset 2: Projekt neve megadása – sikeres továbbhaladás**

**Cél**: Annak ellenőrzése, hogy a rendszer megfelelően továbbengedi a felhasználót, ha helyesen meg van adva a projekt neve.

#### **Lépések**:

1. Felhasználó a főoldalon rákattint a **„Notiz létrehozása”** gombra.
2. A rendszer átirányítja a projekt létrehozó oldalra.
3. A felhasználó **kitölti a projekt nevet** (pl. „autoszerelés”).
4. A felhasználó rákattint a „Tovább” gombra.

#### **Elvárt eredmény**:

* A rendszer **továbbengedi** a felhasználót a következő oldalra, ahol a projekt részletei megadhatók (pl. feladatok, határidők stb.).

### Fejlesztési lehetőségek

**Archivált jegyzetek visszahozása**

**Backend rész:**

* Az archivált jegyzetek visszahozásának működése az API szinten valósul meg, ahol külön végpontok kezelik a jegyzetek archiválását és visszaállítását.
* Amikor egy jegyzetet archiválnak, az adatbázisban az "aktív" státusz helyett az archivált státuszra vált. Ezáltal a jegyzet nem tűnik el véglegesen, csak elrejtődik a fő listából.
* A jegyzetek visszahozására külön API végpont van, amely a felhasználó kérésére visszaállítja az archivált jegyzetet az aktív státuszra. A backend ezután frissíti az adatbázist, és újra elérhetővé teszi a jegyzetet.

**Felhasználói tapasztalat:**

* A felhasználók nem találják meg az archivált jegyzeteket az alapértelmezett jegyzetek között, mivel azok archivált állapotban vannak.
* A felhasználói felületen elérhető egy lehetőség (pl. egy "Archivált jegyzetek" szűrő vagy külön szekció), ahol a felhasználók megtekinthetik az archivált jegyzeteket.
* Ha egy felhasználó szeretné újra használni vagy módosítani az archivált jegyzetet, egyszerűen kiválaszthatja a visszaállítást, amely automatikusan az aktív jegyzetek közé helyezi.
* A felhasználói felület folyamatos visszajelzést ad, hogy a jegyzet sikeresen vissza lett állítva az aktív listára, és mostantól újra használható.

## Felhasználói dokumentáció

### 

### Weboldal használatának részletes ismertetése

Beépített direktívák és injektált szolgáltatások

**Notiz létrehozása oldal:**

@inherits LayoutComponentBase

@using BlazorApp1.Service

@using CommonLibrary

@inject NavigationManager Navigate

@inject HttpClient httpClient

@inject ItemService akarmi

@inject Microsoft.JSInterop.IJSRuntime JSRuntime

* @inherits LayoutComponentBase: Ez egy layout komponens, így más oldalakat is körbe tud ölelni.
* @inject: Külső függőségek hozzáadása, pl. navigáció (Navigate), HTTP kérés, saját szolgáltatás (ItemService), JS hívások.

**Kijelentkezés**

private async Task Logout()

{

akarmi.Username = null;

Navigate.NavigateTo("/", forceLoad: true);

}

* Törli a felhasználó nevét, majd visszanavigál a főoldalra.
* forceLoad: true újratölti az oldalt – nem csak Blazoron belül vált.

**Notiz ID generálás**

private int notizid = 0;

private void notizIdGeneralas()

{

Random rnd = new();

notizid = rnd.Next(10000, 99999);

}

* A notizid a projekt egyedi azonosítója.
* 5 jegyű random szám generálása.

**Projekt név bekérése + validáció**

private string ProjectName { get; set; } = "";

bool vanProjektnev = false;

void setProjektNev()

{

if (!string.IsNullOrWhiteSpace(ProjectName))

vanProjektnev = true;

else

JSRuntime.InvokeVoidAsync("alert", "Kérlek, adj meg egy projekt nevet!");

}

* A felhasználónak meg kell adnia a projekt nevét.
* Ha nem adja meg, JS alerttel figyelmezteted.

**Személy struktúra (táblázat sora)**

class tablazatSzemelyek

{

public bool IsChecked { get; set; } = false;

public string Szemely { get; set; } = "";

public string Feladat { get; set; } = "";

public string Allapot { get; set; } = "";

public DateTime? Kezdes { get; set; } = DateTime.Today;

public DateTime? Hatarido { get; set; } = DateTime.Today.AddDays(1);

public string Fontossag { get; set; } = "";

public string Komment { get; set; } = "";

}

* Ez egy modellosztály, amit a táblázathoz használunk.
* Az osztályban úgynevezett property-k(tulajdonságok) vannak
* DateTime.Today = mai nap, AddDays(1) = holnap

**Adatok feltöltése a szerverre:**

private async Task Feltoltes()

{

var boolDikkció = new Dictionary<string, bool> { ... };

string igaciFontossag = boolDikkció.FirstOrDefault(x => x.Value).Key;

akarmi.ProjectName = ProjectName;

foreach (var item in ujSzemely)

{

var ujnotiz = new CommonLibrary.Notiz { ... };

await akarmi.AddNotiz(ujnotiz);

}

akarmi.AddFelhasznaloNotizAsync(akarmi.Username, Convert.ToString(notizid));

}

* Beállítja az aktuális fontossági értéket.
* Létrehoz Notiz objektumokat, és elküldi őket egy szerveroldali metódusra (AddNotiz).
* Végül összekapcsolja a projektet a felhasználóval.

**Személyek hozzáadása és törlése**

List<tablazatSzemelyek> ujSzemely = new() { new tablazatSzemelyek() };

async Task UjSor()

{

ujSzemely.Add(new tablazatSzemelyek());

await Task.Delay(100);

await JSRuntime.InvokeVoidAsync("reinitializeDropdowns");

}

private void Torles()

{

ujSzemely = ujSzemely.Where(x => !x.IsChecked).ToList();

}

* UjSor: új sort (személyt) ad hozzá, majd újrainicializálja a dropdownokat.
* Torles: csak azokat hagyja meg, amelyek nincsenek kipipálva.

**Új tábla létrehozása (popup)**

private void UjTabla()

{

ujSzemely = new List<tablazatSzemelyek> { new tablazatSzemelyek() };

HidePopup();

}

* Törli a jelenlegi adatokat, és egy üres új sort hoz létre.

**Másolás vágólapra (JS)**

private async Task Masolas()

{

await JSRuntime.InvokeVoidAsync("copyToClipboard", "masolhatoSzoveg");

}

* A notizid értékét másolja ki a vágólapra a megfelelő HTML elem azonosítója alapján.

**JS inicializáció a komponens renderelés után**

protected override async Task OnAfterRenderAsync(bool firstRender)

{

if (firstRender && !\_jsInitialized)

{

\_jsInitialized = true;

await Task.Delay(500);

await JSRuntime.InvokeVoidAsync("initializeClickEvents", listElement);

await Task.Delay(100);

await JSRuntime.InvokeVoidAsync("initDropdowns");

}

}

* Csak az **első renderelés** után fut.
* Késleltetve inicializál néhány JS funkciót, pl. dropdown animációk.

**Fontosság frissítése soronként**

private void UpdateFontossag(tablazatSzemelyek item, string ujErtek)

{

item.Fontossag = ujErtek;

Console.WriteLine($"Frissített fontosság: {ujErtek}");

}

* Ez frissíti a dropdown értékét adott sorban.

**főoldal:**

**Adatok betöltése:**

private List<Notiz> notizLista = new();

private List<Felhasznalok2> felhasznalokLista = new();

List<string> NOTIZIDLISTA = new();

* notizLista: az összes projekt.
* felhasznalokLista: felhasználói adatok.
* NOTIZIDLISTA: azon Notiz ID-k, amiket a bejelentkezett felhasználóhoz társítunk.

protected override async Task OnInitializedAsync()

{

notizLista = await akarmi.GetNotiz();

felhasznalokLista = await akarmi.GetItems();

foreach (var item in felhasznalokLista)

{

if (item.felhasznaloNev.Equals(akarmi.Username) && item.FELHASZNALONOTIZID != null)

{

for (int i = 0; i < item.FELHASZNALONOTIZID.Count(); i += 5)

{

string aktualis = "";

if (item.FELHASZNALONOTIZID.Length >= i + 5)

{

aktualis = $"{item.FELHASZNALONOTIZID[i]}{item.FELHASZNALONOTIZID[i + 1]}{item.FELHASZNALONOTIZID[i + 2]}{item.FELHASZNALONOTIZID[i + 3]}{item.FELHASZNALONOTIZID[i + 4]}";

if (!NOTIZIDLISTA.Contains(aktualis))

{

NOTIZIDLISTA.Add(aktualis);

}

}

}

}

}

Console.WriteLine($"Betöltött projektek száma: {notizLista.Count}");

}

* Lekéri a projekteket és felhasználókat.
* Kikeresi a bejelentkezett felhasználóhoz tartozó 5 karakteres Notiz kódokat, és hozzáadja a NOTIZIDLISTA-hoz.

**Notiz hozzáadása kód alapján:**

bool? dontes = null;

string kod = "";

public void KodAlapjanBedobas()

{

dontes = true;

bool dobas = true;

foreach (var item in notizLista)

{

if (item.NOTIZID.Equals(kod))

{

dobas = false;

break;

}

}

if (dobas) return;

foreach (var item in NOTIZIDLISTA)

{

if (item.Equals(kod)) return;

}

NOTIZIDLISTA.Add(kod);

}

* A felhasználó kézzel beír egy 5 karakteres kódot. Ha létezik ilyen projekt és még nincs hozzáadva, akkor bekerül a listába.

**Projekt kiválasztás és megjelenítés**

string? kivalasztottKod = null;

private void KivalasztottKodBeallitasa(string kod)

{

kivalasztottKod = kod;

dontes = true;

}

* Beállítja, hogy melyik projekt legyen kijelölve. A dontes flag miatt a táblázat is megjelenik.

**Notiz törlése:**

bool? NotiztorlesFeldobasBool = null;

string TorolniKivantKod = "";

private void NotizTorlesFeldobas() => NotiztorlesFeldobasBool = true;

private void NotizTortlesElrejtes() => NotiztorlesFeldobasBool = false;

private async void NotizVeglegesTorles()

{

NOTIZIDLISTA.RemoveAll(item => item.Equals(TorolniKivantKod));

}

* Megjelenít egy inputot, amibe be lehet írni a törölni kívánt projekt kódját.
* Törli azt a kódot a listából, majd:

private async Task RemoveNotiz()

{

if (string.IsNullOrWhiteSpace(TorolniKivantKod) || TorolniKivantKod.Length != 5)

{

statusMessage = "A NotizID pontosan 5 karakter hosszú kell legyen!";

return;

}

bool success = await akarmi.RemoveNotizIdAsync(akarmi.Username, TorolniKivantKod);

if (success)

{

await JSRuntime.InvokeVoidAsync("alert", "A kívánt Notiz sikeresen törölve!");

}

else

{

await JSRuntime.InvokeVoidAsync("alert", "Nem sikerült a törlés");

}

}

* Meghívja a szerveren az adott kód törlését a felhasználótól.

**Kijelentkezés és mentés**

private async Task FelhasznalohozNotizIdHozzaadas()

{

bool success = await akarmi.AddFelhasznaloNotizAsync(akarmi.Username, kod);

Message = success ? "Sikeresen frissítve!" : "Hiba történt.";

akarmi.Username = null;

Navigate.NavigateTo("/",forceLoad:true);

}

* Hozzáadja a kódot a felhasználóhoz a szerveren, majd kijelentkezteti.

Notiz Módosítása oldal:

**Használt direktívák és beállítások**

@using BlazorApp1.Service

@using CommonLibrary

@inject ItemService akarmi

@inherits LayoutComponentBase

* @using direktívák segítségével más névterek osztályait tesszük elérhetővé.
* @inject ItemService akarmi: dependency injection, ezzel az ItemService nevű szolgáltatás példányát kapjuk meg akarmi néven, amit az adatbetöltéshez és adatmanipulációhoz használunk.
* @inherits LayoutComponentBase: ezáltal a komponens örökli az alap komponens logikát, és képes lesz layout szerepet is betölteni.

**Adatbetöltés**

public List<Notiz> betoltottNotizLista = new();

public List<Notiz> aKIVALASZTOTTAK = new();

public string EfiSegitseg;

* betoltottNotizLista: az összes beérkező Notiz-objektum ide töltődik be az adatbázisból.
* aKIVALASZTOTTAK: ebbe a listába csak az a Notiz kerül, amit éppen módosítani szeretnénk.
* EfiSegitseg: megjeleníti a kiválasztott projekt nevét (UI-n egy címként jelenik meg).

**Komponens inicializálása**

protected override async Task OnInitializedAsync()

{

betoltottNotizLista = await akarmi.GetNotiz();

Console.WriteLine(betoltottNotizLista?.Count());

foreach (var item in betoltottNotizLista)

{

if (item.NOTIZID.Equals(akarmi.ModositaniKivantNotizIdProp))

{

aKIVALASZTOTTAK.Add(item);

EfiSegitseg = item.ProjektNev;

}

}

}

* Aszinkron módon lekérdezi az összes Notiz-t az akarmi (ItemService) segítségével.
* A konzolra kiírja, hogy hány Notiz-t kapott vissza.
* Ezután kiválogatja csak azt az egy Notiz-t, amelynek az ID-je megegyezik a módosítani kívánt Notiz ID-val, amit a ModositaniKivantNotizIdProp tárol.
* A kiválasztott elem bekerül az aKIVALASZTOTTAK listába, így csak az fog megjelenni a UI-on.
* A projekt nevét a EfiSegitseg változóban tárolja, amit majd egy <h1> elem jelenít meg.

**Szerkesztés és mentés kezelése**

Notiz szerkesztettNotiz = null;

string notizString;

void SzerkesztesInditasa(Notiz item)

{

szerkesztettNotiz = item;

}

* A szerkesztettNotiz változó határozza meg, épp melyik sor van szerkesztési módban.
* A SzerkesztesInditasa() metódus dupla kattintásra fut le, és az adott sort "aktiválja" szerkesztéshez.

void Mentes()

{

szerkesztettNotiz = null;

}

A mentés gomb lenyomása után kilép a szerkesztés módból.

Az UpdateNotiz hívás még nincs implementálva, de jelölve van, hogy hová kéne kerülni.

public void UjNotizAdatBazis()

{

// Még nem implementált, új Notiz létrehozására szolgál majd.

}

#### A program céljának és lényegesebb funkcióinak összefoglalása

A webes alkalmazás célja, hogy egyszerű és átlátható módot nyújtson a felhasználóknak jegyzeteik (Notizok) rendszerezésére és megosztására. Mivel a rendszer regisztrációhoz kötött, mindenki csak a saját jegyzeteit látja, kivéve, ha valaki kifejezetten megosztja vele az adott Notizt.

Főbb funkciók:

* **Notiz létrehozása** – új jegyzet felvitele cím, tartalom és projekt megadásával.
* **Notiz módosítása** – meglévő jegyzetek szerkesztése.
* **Notiz törlése** – már nem szükséges jegyzetek eltávolítása.
* **Notiz megosztása** – más felhasználók meghívása egy adott jegyzethez.

A rendszer lehetőséget ad arra, hogy mindenki a saját munkáját kezelje, de ha szükséges, másokkal is könnyedén együtt tudjon dolgozni.

#### bejelentkezés/regisztráció oldal:

**Használt szolgáltatások / direktívák**

@inherits LayoutComponentBase

@using BlazorApp1.Service

@using CommonLibrary

@inject ItemService akarmi

@inject NavigationManager Navigation

@inject Microsoft.JSInterop.IJSRuntime js

* **LayoutComponentBase:** A komponens örököl az alapértelmezett Blazor elrendezésből.
* **ItemService akarmi:** Backend szolgáltatás a felhasználók lekérésére, új felhasználók mentésére, valamint a bejelentkezés hitelesítésére.
* **NavigationManager:** Oldalra navigálás vezérlése.
* **IJSRuntime js:** JavaScript kóddal való kommunikációhoz használjuk, pl. vizuális visszajelzésekhez.

**OnAfterRenderAsync**

protected override async Task OnAfterRenderAsync(bool firstRender)

* Ez a metódus minden renderelés után fut. A firstRender azt jelzi, hogy ez az első render-e.

if (firstRender && !\_jsInitialized)

* Ha ez az első render és még nem futott le az inicializálás…

\_jsInitialized = true;

* ... akkor beállítjuk, hogy többé ne fussunk újra.

await Task.Delay(100);

* Várakozunk 100ms-ot, hogy biztosan minden HTML elem betöltődjön.

await js.InvokeVoidAsync("switchToLogin");

* Meghívunk egy JS függvényt, ami visszavált a bejelentkezési nézetre.

**Bejelentkezés változói**

string nevB = "";

string jelszoB = "";

string emailB = "";

**Regisztrációs változók**

private string regFelhasznaloNev="";

private string regEmail = "";

private string regJelszo = "";

* A regisztrációs formban megadott adatok.

**Ellenőrző boolok**

private bool? megvan = null;

private bool? megvan2 = null;

* Jelzi, hogy sikeres volt-e a bejelentkezés (megvan) vagy a regisztráció (megvan2).

### Felhasználók listája

public List<Felhasznalok2> felhasznalokLista = new();

* Tartalmazza az összes regisztrált felhasználót.

OnInitializedAsync

protected override async Task OnInitializedAsync()

* Ez a metódus akkor fut, mikor a komponens először betöltődik.

felhasznalokLista = await akarmi.GetItems();

* Lekéri az összes felhasználót az adatbázisból a GetItems() segítségével.

**Bejelentkezési logika**

private void Megnezes()

Ez a metódus akkor hívódik meg, amikor a felhasználó be akar jelentkezni.

foreach (var item in felhasznalokLista)

Végigmegyünk az összes felhasználón.

if (item.felhasznaloNev.Equals(nevB) && item.jelszo.Equals(jelszoB))

Megnézzük, hogy egyezik-e a felhasználónév és jelszó.

akarmi.Username = nevB;

Beállítjuk a bejelentkezett felhasználó nevét a szolgáltatásban.

megvan = true;

break;

* Beállítjuk, hogy sikeres volt a bejelentkezés és kilépünk a ciklusból.

else { megvan = false; }

* Ha nem találtunk egyezést, akkor hibás a bejelentkezés.

**Regisztráció logika**

private async Task Regisztracio()

* A metódus regisztrációkor fut le.

if (felhasznalokLista.Any(van=> van.felhasznaloNev.Equals(regFelhasznaloNev)))

* Ha már létezik ilyen felhasználónév...

megvan2 = false;

await js.InvokeVoidAsync("RegHaLetezikFelh");

return;

* ... akkor hibát jelzünk és kilépünk.

**Új felhasználó objektum létrehozása**

var ujFelhasznalo = new Felhasznalok2

{

felhasznaloNev = regFelhasznaloNev,

email = regEmail,

jelszo = regJelszo

};

**Validációk**

if (string.IsNullOrWhiteSpace(regJelszo))

* Ha nincs megadva jelszó → hibaüzenet JS-ből

if (string.IsNullOrWhiteSpace(regFelhasznaloNev))

* Ha nincs felhasználónév → szintén JS hiba

**Sikeres regisztráció**

if (!string.IsNullOrWhiteSpace(...))

* Ha minden mező ki van töltve...

await akarmi.AddItem(ujFelhasznalo);

* ... elmentjük az új felhasználót.

felhasznalokLista = await akarmi.GetItems();

* Frissítjük a felhasználók listáját.

await js.InvokeVoidAsync("RegSikeres");

await Task.Delay(2000);

await js.InvokeVoidAsync("switchToLogin");

* Sikerüzenet + automatikus váltás a bejelentkezésre.

**JavaScript: részletes soronkénti magyarázat**

function switchToLogin() {

* Ez egy függvénydefiníció, ami akkor hívódik meg, amikor vissza akarjuk váltani a nézetet a bejelentkezés oldalra.

document.getElementById("login-form").style.display = "block";

* A bejelentkezési űrlapot (login-form) megjelenítjük (display: block).

document.getElementById("register-form").style.display = "none";

* A regisztrációs űrlapot (register-form) elrejtjük (display: none).

function switchToRegister() {

* Ez a függvény akkor fut le, ha át akarunk váltani a regisztrációs nézetre.

document.getElementById("login-form").style.display = "none";

* A bejelentkezési űrlapot elrejtjük.

document.getElementById("register-form").style.display = "block";

* A regisztrációs űrlapot megjelenítjük.

function RegHaLetezikFelh() {

* Függvény, amit akkor hívunk, ha a regisztráció során a megadott felhasználónév már létezik.

alert("Ez a felhasználónév már foglalt!");

* Figyelmeztető üzenetet jelenítünk meg a felhasználónak (klasszikus alert()).

function RegSikeres() {

* Függvény, amelyet akkor hívunk, ha a regisztráció sikeres volt.

alert("Sikeres regisztráció! Átirányítás a bejelentkezéshez...");

* Visszajelzés a felhasználónak, hogy minden rendben ment, és jön az átirányítás.

**CSS**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Poppins', sans-serif;

}

* Ez a globális reset: minden elemről leszedjük az alapértelmezett margókat, paddinget, és beállítjuk a box-sizing-et, hogy a padding és border is beleszámítson a szélességbe/magasságba. A betűtípus mindenhol 'Poppins'.

body {

background-color: black !important;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background: url('background.jpg') no-repeat;

background-size: cover;

background-position: center;

}

* A body a teljes képernyőt lefedi (100vh), középre igazítja a tartalmat (flex), és egy fix háttérkép van beállítva, amit nem ismétel, kitölti a helyet, és középre pozicionálja.

header {

position: fixed;

top: 0;

left: 0;

width: 100%;

backdrop-filter: blur(35px);

padding: 20px 100px;

display: flex;

justify-content: space-between;

align-items: center;

z-index: 99;

}

* A fejléc (header) fixen van rögzítve a képernyő tetejére, teljes szélességben, elmosott háttérrel (backdrop-filter). A belső elemeket egy flexbox rendezi el balra és jobbra, így lesz hely a logónak és a menünek.

header nav {

display: flex;

align-items: center;

justify-content: flex-end;

}

* A navigációs sáv (nav) is flexboxos, és jobbra igazítja az elemeket (például menüpontokat).

.logo {

font-size: 2em;

color: #fff;

user-select: none;

}

* A logó szöveges, nagy méretű (2em), fehér színű, és nem lehet kijelölni (user-select: none), így nem zavaró UX szempontból.

.navigation a {

position: relative;

font-size: 1.1em;

color: white;

user-select: none;

cursor: pointer;

text-decoration: none;

font-weight: 500;

margin-left: 40px;

}

* Itt a navigációs linkek (a tag-ek) stílusát látjuk: közepes méret, fehér, nem kijelölhető, kis térköz balról. A position: relative kell a későbbi animált aláhúzás miatt.

.navigation a:hover {

color: white !important;

}

.navigation a::after {

content: '';

position: absolute;

left: 0;

bottom: -6px;

width: 100%;

height: 3px;

background: #fff;

border-radius: 5px;

transform-origin: right;

transform: scaleX(0);

transition: transform .35s;

}

* Hover állapotban marad fehér (biztos, ami biztos).
* Az ::after egy kis aláhúzás effekt, amit animálva jelenítünk meg. Kezdetben nincs skálázva (scaleX(0)), de animálódik, ha hover jön.

.navigation a:hover::after {

transform-origin: left;

transform: scaleX(1);

}

* Amikor föléviszed az egeret, az aláhúzás balról jobbra nő ki. Kis dizájnos animáció, szép sima.

.navigation .btnLogin-popup {

width: 150px;

height: 50px;

background: transparent;

border: 2px solid;

outline: none;

border-radius: 6px;

cursor: pointer;

font-size: 1.1em;

color: #fff;

font-weight: 500;

margin-left: 40px;

transition: .5s;

}

* Ez a "Bejelentkezés" vagy hasonló gomb. Üveg hatású (transparent háttér), fehér betű, lekerekített sarkok, kis animált áttűnés.

.navigation .btnLogin-popup:hover {

background: #fff;

color: #162938;

}

* Hovernél megfordul a szín: fehér háttér, sötétkékes szöveg. Klasszikus kontrasztos gomb-effekt.

.wrapper {

position: fixed;

width: 400px;

height: 440px;

user-select: none;

background: transparent;

border: 2px solid rgba(255, 255, 255, .5);

border-radius: 20px;

backdrop-filter: blur(50px);

box-shadow: 0 0 30px rgba(0,0,0, .5);

display: flex;

justify-content: center;

z-index: 2;

align-items: center;

overflow: hidden;

transform: scale(0);

transition: transform .5s ease, height .2s ease;

}

* Ez maga az ablak, amiben a bejelentkezés/regisztráció van.
* Kezdetben kicsi (scale(0)), áttetsző üveges stílusú (backdrop-filter), árnyékos, fix pozíciójú. Ha aktiválod, kinagyul animálva.

.wrapper.active-popup {

transform: scale(1);

}

.wrapper.active {

height: 520px;

}

* Az .active-popup nyitja meg az ablakot. Az .active emellett növeli a magasságot is, ha pl. regisztrációs form nyílik meg.

.wrapper .icon-close {

position: absolute;

top: 0;

right: 0;

width: 45px;

height: 45px;

background: #162938;

font-size: 2em;

color: #fff;

display: flex;

justify-content: center;

align-items: center;

border-bottom-left-radius: 20px;

cursor: pointer;

z-index: 1;

}

* Ez az X ikon jobb felül, amivel bezárható az ablak. Kerekített sarka van, sötét háttér, középre rendezve az ikon.

.wrapper .icon-close:hover {

color: #162938;

background-color: #fff;

}

.wrapper .form-box {

padding: 40px;

}

.wrapper .form-box.login {

transition: transform .18s ease;

transform: translateX(0);

}

* Ez maga a formázott doboz a belsejében. A .login verzió alapból a helyén van (translateX(0)), és kap egy kis animációt, ha elmozdul.

.wrapper.active .form-box.login {

transition: none;

transform: translateX(-400px);

}

.wrapper .form-box.register {

position: absolute;

transition: none;

transform: translateX(400px);

}

.wrapper.active .form-box.register {

transition: transform .18s ease;

transform: translateX(0);

}

* Itt történik a varázs: amikor .wrapper.active aktiválódik (pl. átváltasz regisztrációra), a login doboz elcsúszik balra, a register meg beúszik jobbról.

.form-box h2 {

font-size: 2em;

color: #162938;

text-align: center;

}

* A form fejléce (pl. "Bejelentkezés") nagy betűs, sötétkék és középre van igazítva.

.input-box {

position: relative;

width: 100%;

height: 50px;

border-bottom: 2px solid #162938;

margin: 30px 0;

}

* Ez a keret az input mezők körül. Kis alsó vonalas dizájn, viszonylag tág margóval.

.input-box label {

position: absolute;

top: 50%;

left: 5px;

transform: translateY(-50%);

font-size: 1em;

color: #162938;

font-weight: 500;

pointer-events: none;

transition: .5s;

}

* A label-ek az inputokon belül úsznak. Amikor belekattintasz, felugranak (mint a Material UI-nál). Ezt segíti a transition és a pointer-events: none.

.input-box input:focus ~ label,

.input-box input:valid ~ label {

top: -5px;

}

* Ha fókuszálsz vagy helyesen ki van töltve a mező, a label felmegy a mező fölé. Ezzel lesz "lebegő" hatása a feliratnak.

.input-box input {

width: 100%;

height: 100%;

background: transparent;

border: none;

outline: none;

font-size: 1em;

color: #162938;

font-weight: 600;

padding: 0 35px 0 5px;

}

* Az input maga teljes szélességet kap, átlátszó háttérrel, csak szöveg van benne. A padding jobb oldalon nagyobb, mert oda jön az ikon.

.input-box .icon {

position: absolute;

right: 8px;

font-size: 1.2em;

color: #162938;

line-height: 57px;

}

* Az ikonok (pl. kis lakat, user) jobbra vannak igazítva, fix magasságban — mindig egy szintben az input szöveggel.

remember-forgot {

font-size: .9em;

color: #162938;

font-weight: 500;

margin: -15px 0 15px;

display: flex;

justify-content: space-between;

}

* Ez a kis szakasz a checkbox („Emlékezz rám”) és az „Elfelejtetted a jelszavad?” link között. Kis betűméret, sötétkék szín, két oldalra igazítva.

.remember-forgot label input {

accent-color: #162938;

margin-right: 3px;

}

.remember-forgot a {

color: #162938;

text-decoration: none;

}

.remember-forgot a:hover {

text-decoration: underline;

}

* A checkbox kap egy kis sötétkék árnyalatot (accent-color), a link alapból sima, hoverre aláhúzott — klasszik webes UX.

.btn {

width: 100%;

height: 45px;

background: #162938;

border: none;

outline: none;

border-radius: 6px;

cursor: pointer;

font-size: 1em;

color: #fff;

font-weight: 500;

}

* A gomb a form alján: teljes szélesség, sötét háttér, fehér szöveg, lekerekítve, semmi fölös dizájn. Pont amilyen kell.

ZalanSzoveg {

text-align: center;

margin-left:15px;

background-color: transparent;

border-radius: 10px;

font-weight:600;

text-shadow:white 2px 2px 4px;

color: black;

}

* Ez valami speckó, név alapján talán egy személyes belső poén vagy cím? Szép kis árnyékkal fehér szövegárnyék és fekete szöveg – fancy, de átlátszó háttérrel.

.login-register {

font-size: .9em;

color: #162938;

text-align: center;

font-weight: 500;

margin: 25px 0 10px;

}

* A szöveg: „Nincs még fiókod?” vagy „Már van fiókod?” középre igazítva, kis betű, de hangsúlyos, hogy észrevegyék.

.login-register p a {

color: #162938;

text-decoration: none;

font-weight: 600;

}

.login-register p a:hover {

text-decoration: underline;

}

* Ez a link benne: erősebb betű, hoverre aláhúzódik – tiszta, átlátható UX, semmi cicoma.

responsivemenu {

display: none;

visibility: hidden;

}

.responsivewrapper {

display: none;

}

* Ez a két osztály kezdetben rejtve van. Ezek a mobilnézethez tartozó menüelemek, amiket csak kisebb képernyőkön jelenítünk meg. display: none és visibility: hidden biztosítja, hogy sem vizuálisan, sem interaktívan ne legyenek jelen.

@media only screen and (max-width: 630px) {

* A következő blokk egy media query, amely kizárólag 630 pixel szélesség alatti képernyőkre vonatkozik. Ez a tipikus határvonal, amely mobiltelefonokra szabja az oldalt.

.responsivemenu {

display: block;

visibility: visible;

color: #fff;

font-size: 50px;

font-weight: 500;

}

* Mobilnézetben az .responsivemenu osztály aktiválódik. Ez egy hamburgermenü, mobilra optimalizált navigációs elem. Nagyméretű, jól látható betűkkel jelenik meg, fehér színnel.

header {

padding: 30px;

display: flex;

justify-content: space-between;

align-items: center;

z-index: 99;

}

header .logo {

font-size: 1.5em;

}

header nav {

display: none;

}

* Mobilos megjelenésnél a fejléc továbbra is jól strukturált, de a hagyományos navigáció (nav) el van rejtve (display: none), és helyette a responsivemenu veszi át a szerepet. A logó mérete kisebb, hogy jobban illeszkedjen a kompakt felületre.

.navigation .btnLogin-popup {

width: 150px;

height: 25px;

border: 2px solid;

border-radius: 6px;

font-size: 1em;

font-weight: 500;

margin-left: 40px;

}

* A bejelentkező gomb is kisebb méretű, alacsonyabb magasságú. A formai jellemzők – lekerekítés, szegély, betűméret – megmaradnak, de arányosan kisebbek lesznek, hogy illeszkedjenek a kisebb kijelzőmérethez.

.bemutat {

color: white;

background-color: black;

position: relative;

top: 130vh;

}

.content h2 {

width: 50%;

}

* A bemutat szekciót mobilon még lejjebb tolja (top: 130vh), feltehetően azért, hogy ne lógjon bele az előző tartalomba. A cím (h2) szélessége is növekszik (30% → 50%), így a mobilos elrendezésben jobban kitölti a rendelkezésre álló helyet.

.responsivewrapper {

position: fixed;

top: 100px;

right: 20px;

background-color: white;

display: inline-block;

border-radius: 15px;

}

* Ez a konténer a mobilos menü opciókat tartalmazza. Fix pozícióban jelenik meg a jobb felső sarokban. A háttér világos, lekerekített sarkokkal – barátságos, letisztult megjelenés.

.responsivewrapper a {

cursor: pointer;

user-select: none;

display: flex;

padding: 15px;

margin: 10px;

}

* A mobilos menüpontok linkként viselkednek, nem lehet őket kijelölni (user-select: none), és megfelelő méretű paddinget kapnak az érintőkijelzős használhatóság érdekében.

.responsivewrapper button {

border-radius: 10px;

outline: none;

text-decoration: none;

border: 1px black solid;

background-color: transparent;

padding: 15px;

margin: 0 15px 15px;

transition: all 0.2s ease 0.1s;

}

.responsivewrapper button:hover {

background-color: black;

color: #fff;

}

* A mobilos gombokhoz tartozó stílus. Alapból átlátszó, fekete szegéllyel, de hover-re fekete háttér és fehér betűszín jelenik meg. Az all tranzíció biztosítja a gördülékeny animációt.

Notiz Módosítása oldal:

<HeadContent>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css' rel="stylesheet">

</HeadContent>

 <HeadContent>: Blazor-specifikus komponens, amely lehetővé teszi a <head> szekció dinamikus módosítását komponensekből.

 <meta charset="UTF-8">: A karakterkódolást határozza meg, UTF-8-ra állítva (ez támogatja a magyar ékezetes karaktereket is).

 <meta name="viewport" ...>: Mobilbarát nézet beállítása; a reszponzív megjelenítés kulcsa.

 <link href=...>: A **Boxicons** ikoncsomag betöltése a felhasználói felülethez szükséges ikonokhoz.

<div class="dropdown">

<div class="select">

<span class="selected">

<span class="icon">

<ion-icon name="person"></ion-icon>

</span>

<span class="fiokom">

Fiókom

</span>

<span class="caret">

<i class="bx bx-chevron-down"></i>

</span>

</span>

</div>

 <div class="dropdown">: A lenyíló menü fő konténere.

 <div class="select">: A dropdown látható része, ami a kiválasztott opciót mutatja.

 <span class="selected">: A kiválasztott állapotot reprezentálja, egy csomagolt tartalommal.

 <ion-icon name="person">: Felhasználó ikont jelenít meg.

 <span class="fiokom">Fiókom</span>: Megjelenített címke, amely utal arra, hogy ez a felhasználói fiók kezelő része.

 <span class="caret">: Lenyíló nyíl ikon, vizuálisan jelzi, hogy a menü lenyitható.

 <i class="bx bx-chevron-down">: A Boxicons könyvtárból származó lefelé mutató nyíl.

<ul class="menu" id="dropmenu">

<li>

<span class="navikon">

<ion-icon name="settings"></ion-icon>

</span>

Profilom beállításai

</li>

<li>

<span class="navikon">

<ion-icon name="log-out"></ion-icon>

</span>

<span @onclick="FelhasznalohozNotizIdHozzaadas"> Kijelentkezés </span>

</li>

</ul>

* <ul class="menu" id="dropmenu">: Egy nemrendezett lista (ul), amely tartalmazza a lenyíló menü elemeit. Az id="dropmenu" lehetővé teszi a JavaScript vagy CSS általi célzást.
* <li>: Listaelem, az egyes lehetőségeket (pl. profilbeállítás, kijelentkezés) képviseli.
* <span class="navikon">: Egy külön span, amely ikonokat tartalmaz, vizuálisan is elkülönítve a funkció nevétől.
* <ion-icon name="settings">: Beállítások ikonja, amely utal a profil konfigurációs lehetőségeire.
* „Profilom beállításai”: Szöveg, amely megnevezi az adott menüpont funkcióját.
* <ion-icon name="log-out">: Kijelentkezés ikon.
* <span @onclick="FelhasznalohozNotizIdHozzaadas">: Blazor @onclick direktíva, amely egy metódus hívását köti a kattintási eseményhez — itt például kijelentkezési funkció vagy naplózási tevékenység kapcsolódik hozzá.

<h1 style="text-shadow:1px 1px 5px black;">@EfiSegitseg</h1>

 <h1>: A legnagyobb szintű címsor, amely a felhasználó számára fontos információt emel ki.

 style="text-shadow:1px 1px 5px black;": Egy inline CSS-stílus, amely fekete árnyékot ad a szövegnek, vizuális kiemelés céljából.

 @EfiSegitseg: Blazor szintaxis, amely egy szerveroldali változót jelenít meg. Ez a változó tartalmazhat egy szöveget, ami útmutatás, státusz vagy egyéb kontextus lehet.

<thead>

<tr>

<td>Tulajdonos</td>

<td>Feladat</td>

<td>Állapot</td>

<td>Mikor kezdje</td>

<td>Határidő</td>

<td>Fontosság</td>

<td>Komment</td>

</tr>

</thead>

* <thead>: A táblázat fejléce. Itt vannak megadva az oszlopnevek.
* <tr>: Egy sor a táblázatban (itt: a fejlécsor).
* <td>: Táblázatcella (data cell), ezekben jelennek meg az oszlopnevek:
  + **Tulajdonos**: Kié az adott projekt vagy részfeladat.
  + **Feladat**: Milyen konkrét teendő tartozik hozzá.
  + **Állapot**: A feladat státusza (pl. kész, elakadt stb.).
  + **Mikor kezdje**: Tervezett kezdési időpont.
  + **Határidő**: Befejezési határidő.
  + **Fontosság**: Prioritás szint (sürgős, fontos stb.).
  + **Komment**: Megjegyzés vagy kiegészítő információ.

<td style="user-select:none;">

@if (szerkesztettNotiz == item)

{

<input @bind="item.Tulajdonos" readonly />

}

else

{

@item.Tulajdonos

}

</td>

 <td>: A "Tulajdonos" cella.

 user-select:none;: Nem engedi, hogy a felhasználó kijelölje a szöveget.

 @if (szerkesztettNotiz == item): Ha az adott sor épp szerkesztés alatt áll:

* <input @bind="item.Tulajdonos" readonly />: Szövegmező jelenik meg, de nem lehet szerkeszteni.

 else: Egyébként csak szimplán megjelenik a tulajdonos neve szövegként.

<td>

@if (szerkesztettNotiz == item)

{

<input @bind="item.Feladat" />

}

else

{

@item.Feladat

}

</td>

A "Feladat" mező dinamikusan jelenik meg:

* Ha szerkesztési módban van: egy input mező (@bind – adatkapcsolat az objektumhoz).
* Máskülönben: egyszerű szövegként látjuk a feladat nevét.

<td>

@if (szerkesztettNotiz == item)

{

<select class="form-select" @bind="@item.Allapot">

<option value="" style="display:none;">Állapot beállítása</option>

<option>Kész</option>

<option>Elkezdve</option>

<option>Elakadtam</option>

<option>Nincs elkezdve</option>

</select>

}

else

{

@item.Allapot

}

</td>

* Ez a cella az "Állapot" értékét jeleníti meg:
  + Ha szerkeszthető a sor: egy legördülő lista (<select>) jelenik meg előre definiált opciókkal.
  + Egyébként: megjelenik az aktuális állapot szövegként.

<td>

@if (szerkesztettNotiz == item)

{

<input @bind="item.MikorKezdje" />

}

else

{

@item.MikorKezdje

}

</td>

* Mikor kezdje” oszlop.
* Ha a sor szerkesztés alatt áll, megjelenik egy beviteli mező az adat kötésével (@bind).
* Ha nem szerkesztjük: csak a dátum/idő jelenik meg sima szövegként.

<td>

@if (szerkesztettNotiz == item)

{

<input @bind="item.HatarIdo" />

}

else

{

@item.HatarIdo

}

</td>

 „Határidő” oszlop.

 Hasonlóan az előzőhöz:

* Szerkesztés esetén: input mező.
* Egyébként: kiírásra kerül a határidő szöveges formában.

<td>

@if (szerkesztettNotiz == item)

{

<select class="form-select" @bind="@item.FOntossag">

<option style="display:none;" value="" disabled selected>Válassz prioritást</option>

<option>Sürgős</option>

<option>Hanyagolható</option>

<option>Fontos</option>

<option>Nem fontos</option>

</select>

}

else

{

@item.FOntossag

}

</td>

 „Fontosság” oszlop.

 Szerkesztés esetén a felhasználó választhat prioritást egy legördülő listából.

 Ellenkező esetben az aktuális érték jelenik meg szövegként.

<td>

@if (szerkesztettNotiz == item)

{

<input @bind="item.Komment" />

}

else

{

@item.Komment

}

</td>

 Komment” oszlop.

 Szerkeszthető input mező jelenik meg, ha a sor aktív szerkesztési módban van.

 Egyébként csak megjeleníti a szöveget.

if (firstRender && !\_jsInitialized)

{

\_jsInitialized = true;

await Task.Delay(100);

await js.InvokeVoidAsync("switchToLogin");

}

Ha ez az első renderelés, és a JavaScript még nem lett inicializálva, akkor:

* \_jsInitialized értékét true-ra állítja,
* késleltet 100 ms-ot (biztonsági várakozás),
* meghívja a frontenden definiált switchToLogin nevű JavaScript függvényt.

string nevB = "";

string jelszoB = "";

string emailB = "";

Három mező, amelyek a bejelentkezéshez és e-mailhez kapcsolódó adatokat tárolják.

private bool? megvan = null;

private bool? megvan2 = null;

Két nullable logikai érték, amelyek visszajelzést adnak arról, hogy a felhasználó megtalálása (megvan) vagy a regisztráció sikeressége (megvan2) megtörtént-e.

private string regFelhasznaloNev="";

private string regEmail = "";

private string regJelszo = "";

A regisztrációs űrlaphoz tartozó adatokat tároló változók.

protected override async Task OnInitializedAsync()

{

felhasznalokLista = await akarmi.GetItems();

}

A komponens inicializálásakor lekéri az összes felhasználót a GetItems() aszinkron metódussal, és betölti a felhasznalokLista listába.

public List<Felhasznalok2> felhasznalokLista = new();

Az összes felhasználót tartalmazó lista. A Felhasznalok2 egy osztály, amely egy felhasználó adatait tartalmazza.

foreach (var item in felhasznalokLista)

{

if (item.felhasznaloNev.Equals(nevB) && item.jelszo.Equals(jelszoB))

{

akarmi.Username = nevB;

megvan = true;

break;

}

else

{

megvan = false;

}

}

Végigmegy az összes felhasználón, és ha a megadott név és jelszó egyezik, akkor:

* beállítja a felhasználónevet a központi akarmi objektumban,
* a megvan értéke true lesz,
* a ciklus megszakad.  
  Ha nincs egyezés, akkor a megvan értéke false.

private async Task BillentyuKezelo(KeyboardEventArgs e)

{

if (e.Key == "Enter")

{

Megnezes();

}

}

Egy eseménykezelő metódus, amely figyeli, ha az Enter billentyű le lett nyomva — ekkor meghívja a Megnezes() függvényt.

#### Szükséges hardvereszközök és szoftverek felsorolása

A **Lil’notiz** webalkalmazás fejlesztéséhez és működtetéséhez az alábbi hardver- és szoftverkörnyezetre van szükség:

**Hardvereszközök:**

* **Fejlesztői oldalon**:
  + Legalább középkategóriás számítógép vagy Laptop
  + Billentyűzet, egér, esetlegesen két monitor, az átláthatóság kedvéért
  + Stabil internetkapcsolat a verziókezeléshez és teszteléshez.
* **Felhasználói oldalon**:
  + Asztali vagy hordozható számítógép, tablet vagy okostelefon, modern böngészővel.
  + Internetkapcsolat a webalkalmazás eléréséhez.

**Szoftverek:**

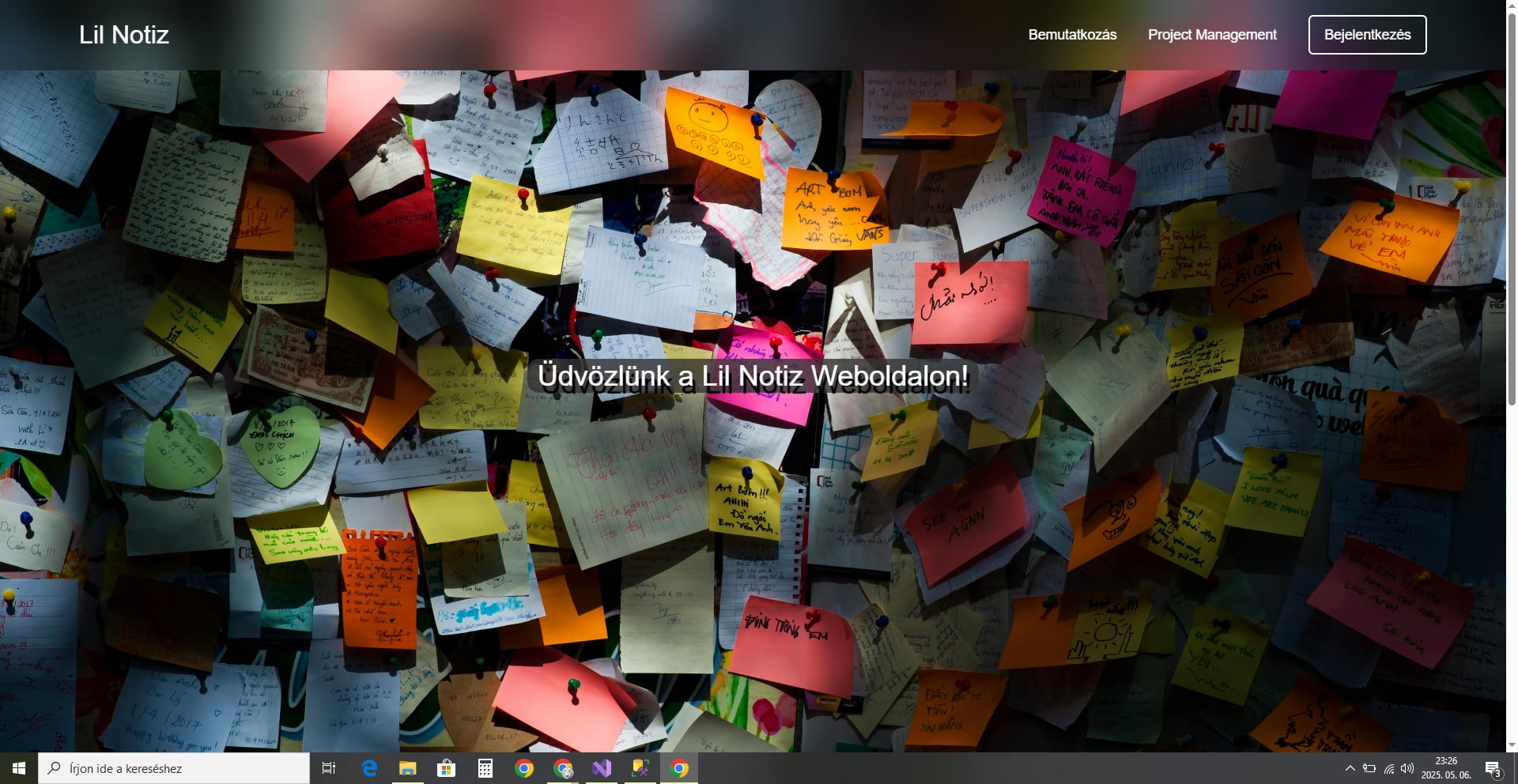
* **Fejlesztéshez**:
  + Operációs rendszer: Windows.
  + Fejleszttői környezet: *Visual Studio 2022*.
  + Verziókezelés: *Git*, *GitHub*, *kisebb mellékmódosításokra google drive.*
  + Backend környezet: *ASP.NET Core Web API (C#), Entity Framework Core.*
  + Frontend keretrendszer: *HTML, CSS, JavaScript*.
  + Adatbázis-kezelő: *MSSQL*.
  + Böngésző alapú teszteléshez: *Chrome*,  *Microsoft Edge*.
* **Felhasználói oldalon**:
  + Bármely korszerű böngésző (Chrome, Firefox, Edge).
  + Nem szükséges telepíteni semmilyen plusz szoftvert, mivel az alkalmazás teljes mértékben webes felületen működik.

#### Telepítés és indítás lépéseinek ismertetése

A Lil'notiz webalkalmazás telepítése és elindítása néhány egyszerű lépésben elvégezhető. A következő útmutató segít a fejlesztői környezet előkészítésében, a projekt beállításában és a futtatásban.

1. **Követelmények telepítése:**
   1. Töltsük le és telepítsük a [Visual Studio 2022](https://visualstudio.microsoft.com/) vagy újabb verzióját.
   2. Győződjünk meg róla, hogy a szükséges adatbázis-kezelő is elérhető.
2. **Projekt letöltése:**
   1. Klónozzuk a projektet Git-ről, vagy más módon másoljuk le a projekt mappát a gépre.
3. **Adatbázis inicializálása:**
   1. A backend alkalmazás első indításakor az Entity Framework Core automatikusan létrehozhatja az adatbázist (Migrations használatával).
4. **Projekt indítása:**
   1. Visual Studio-ban nyissuk meg a .sln fájlt.
   2. Állítsuk be a backend projektet induló projektként, majd a blazort követketőnek.
   3. Nyomjunk a **Start** gombra vagy használjuk a dotnet run parancsot a terminálban.
5. **Frontend elérés:**
   1. Először az API indul el, biztosítva az adatszolgáltatást, majd betöltődik a Blazor frontend a böngészőben.
   2. A felhasználó innen elérheti a webes felületet, bejelentkezhet, létrehozhat jegyzeteket, kategóriákat stb.

#### A program részletes bemutatása



### 1. **Fejléc (header)**

A weboldal tetején egy **átlátszó, elmosott hátterű** fejléc található, amely teljes szélességében végigfut a képernyőn. A fejléc a következő elemeket tartalmazza:

* **Bal oldalon:**
  + A weboldal neve: **"Lil Notiz"**, egyszerű, letisztult betűtípussal, világos (fehér) színnel jelenik meg, jól láthatóan.
* **Jobb oldalon:**
  + **Navigációs menüpontok**, egy sorban, jobbra igazítva:
    - „**Bemutatkozás**”
    - „**Project Management**”
    - Egy kiemelt gomb/jellegű link: **„Bejelentkezés”**, sötétebb háttérrel és világos betűszínnel, vizuálisan elkülönülve a többitől

A fejléc **rögzített pozícióban** van (position: fixed), azaz görgetéskor is látható marad a képernyő tetején.

### 2. **Háttér**

A teljes kezdőoldal háttere egy **színes, élénk fotó**, amely különféle **post-it jegyzeteket** ábrázol, kaotikus, de mégis kreatív hangulatot árasztva. A kép főként a következő jellemzőkkel bír:

* Sötét és világos színek vegyesen szerepelnek
* A fókusz középre esik: a kép középső része van jobban megvilágítva
* A színes jegyzetlapok vizuálisan összhangban vannak a „jegyzetelős” témával

A háttér stílusát a következő CSS-sel állították be:

### 3. **Főcím (Hero Section)**

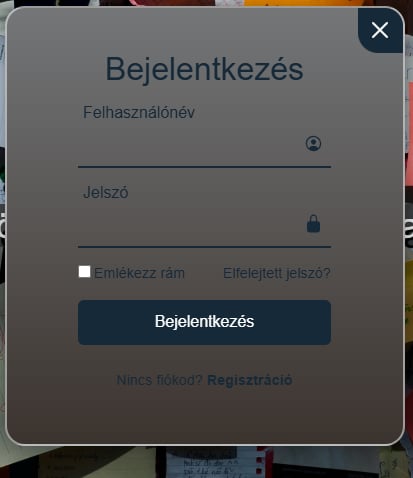
Középen, **a háttérkép előterében** egy **fehér színű szöveg** jelenik meg:

* **„Üdvözlünk a Lil Notiz Weboldalon!”**
* A szöveg:
  + Középre van igazítva
  + Nagyobb betűméretű (font-size), modern sans-serif betűtípussal
  + Színe: **fehér** (color: white)
  + Nincs háttérdoboz vagy kiemelés mögötte, de kontrasztos a mögöttes sötétebb képrészlettel

Ez a szakasz **vizuálisan hívogató**, és egyértelműen közli a látogatóval, hová érkezett.

### 4. **Felhasználói élmény és általános benyomás**

* Az oldal **letisztult**, modern megjelenést biztosít
* A dizájn vizuálisan erős: a háttérkép érzelmi és tematikus kapcsolódást nyújt a weboldal funkciójához (jegyzetelés, emlékek, gondolatok rendszerezése)
* A navigáció **egyszerű és könnyen értelmezhető**
* A tipográfia és az elrendezés alapján a célközönség fiatalabb, digitálisan jártas felhasználó



### **Bejelentkezési felület (Login Panel)**

#### **Fejléc:**

* **Szöveg:** „Bejelentkezés”
* **Funkció:** Egyértelműen jelzi a felhasználónak, hogy ez az oldal a bejelentkezésre szolgál.

#### **Űrlapmezők:**

1. **Felhasználónév**
   * **Szövegmező (placeholder):** „Felhasználónév”
   * **Ikon:** Egy felhasználó ikonja (👤)
   * **Funkció:** Ide kell beírni a felhasználónak a regisztráció során megadott felhasználónevét.
2. **Jelszó**
   * **Szövegmező (placeholder):** „Jelszó”
   * **Ikon:** Egy lakat ikon (🔒)
   * **Funkció:** A felhasználó jelszavát kell ide beírni. Rejtett karakterekkel jelenik meg.

#### **Egyéb opciók:**

* **„Emlékezz rám” jelölőnégyzet:**
  + **Funkció:** Ha be van pipálva, a rendszer megjegyzi a felhasználót, így a következő bejelentkezésnél nem kell újra megadnia az adatokat.
* **„Elfelejtett jelszó?” hivatkozás:**
  + **Funkció:** A felhasználó új jelszót kérhet, ha elfelejtette a meglévőt.

#### **Bejelentkezés gomb:**

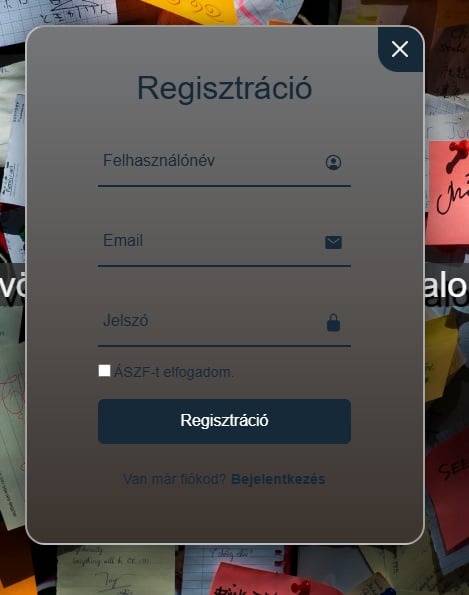
* **Felirat:** „Bejelentkezés”
* **Funkció:** A beírt felhasználónév és jelszó alapján belépteti a felhasználót a rendszerbe.

#### **Alsó rész:**

* **Szöveg:** „Nincs fiókod? Regisztráció”
  + **„Regisztráció” hivatkozásként szerepel**
  + **Funkció:** Új fiók létrehozására visz, ha a felhasználó még nem rendelkezik fiókkal.

#### **Bezáró ikon (X):**

* **Funkció:** Bezárja a bejelentkezési panelt (feltehetően egy felugró ablak része).



### **Regisztrációs felület (Sign Up Panel)**

#### **Fejléc:**

* **Szöveg:** „Regisztráció”
* **Funkció:** Egyértelműen jelzi, hogy a felhasználó új fiók létrehozására szolgáló űrlapot tölt ki.

#### **Űrlapmezők:**

1. **Felhasználónév**
   * **Szövegmező (placeholder):** „Felhasználónév”
   * **Ikon:** Felhasználó ikon (👤)
   * **Funkció:** A kívánt felhasználónév megadása. Ez lesz az azonosító a rendszerben.
2. **Email**
   * **Szövegmező (placeholder):** „Email”
   * **Ikon:** Boríték ikon (📧)
   * **Funkció:** A felhasználó e-mail címének megadása, melyre visszaigazoló e-mailek érkezhetnek.
3. **Jelszó**
   * **Szövegmező (placeholder):** „Jelszó”
   * **Ikon:** Lakat ikon (🔒)
   * **Funkció:** A fiókhoz tartozó jelszó beállítása. Rejtett karakterként jelenik meg.

#### **Egyéb opciók:**

* **ÁSZF elfogadása jelölőnégyzet:**
  + **Szöveg:** „ÁSZF-t elfogadom.”
  + **Funkció:** A felhasználónak el kell fogadnia az Általános Szerződési Feltételeket a regisztrációhoz. Kattintható négyzet.

#### **Regisztráció gomb:**

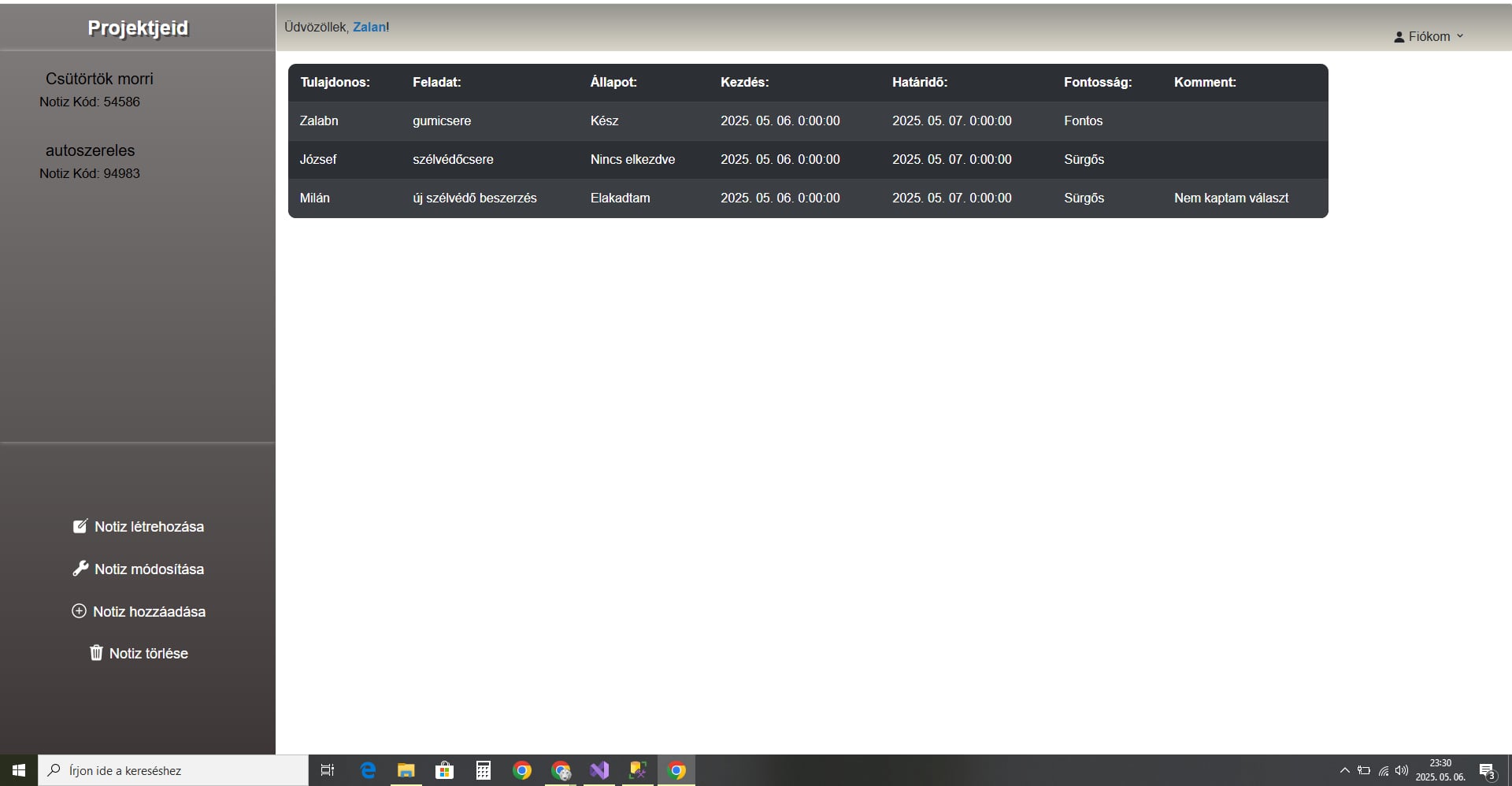
* **Felirat:** „Regisztráció”
* **Funkció:** Beküldi az űrlapot, ha minden mező helyesen ki van töltve és az ÁSZF el van fogadva.

#### **Alsó rész:**

* **Szöveg:** „Van már fiókod? Bejelentkezés”
  + **„Bejelentkezés” hivatkozásként jelenik meg**
  + **Funkció:** Átvált a bejelentkezési felületre, ha a felhasználónak már van fiókja.

#### **Bezáró ikon (X):**

* **Funkció:** Bezárja a regisztrációs panelt (feltehetően felugró ablak).



### **Felhasználó köszöntése:**

* **Szöveg:** „Üdvözöllek, Zalán!”
* **Funkció:** Köszönti a bejelentkezett felhasználót név szerint.

### **Jobb felső sarok:**

* **Szöveg:** „Fiókom ▾”
* **Funkció:** Egy legördülő menüt nyit, ahol a felhasználó módosíthatja a profilját, kijelentkezhet stb.

### **Projektek (menüszekció):**

1. **csütörtök morri**
   * **Notiz kód:** 94648
2. **autószerelés**
   * **Notiz kód:** 94649

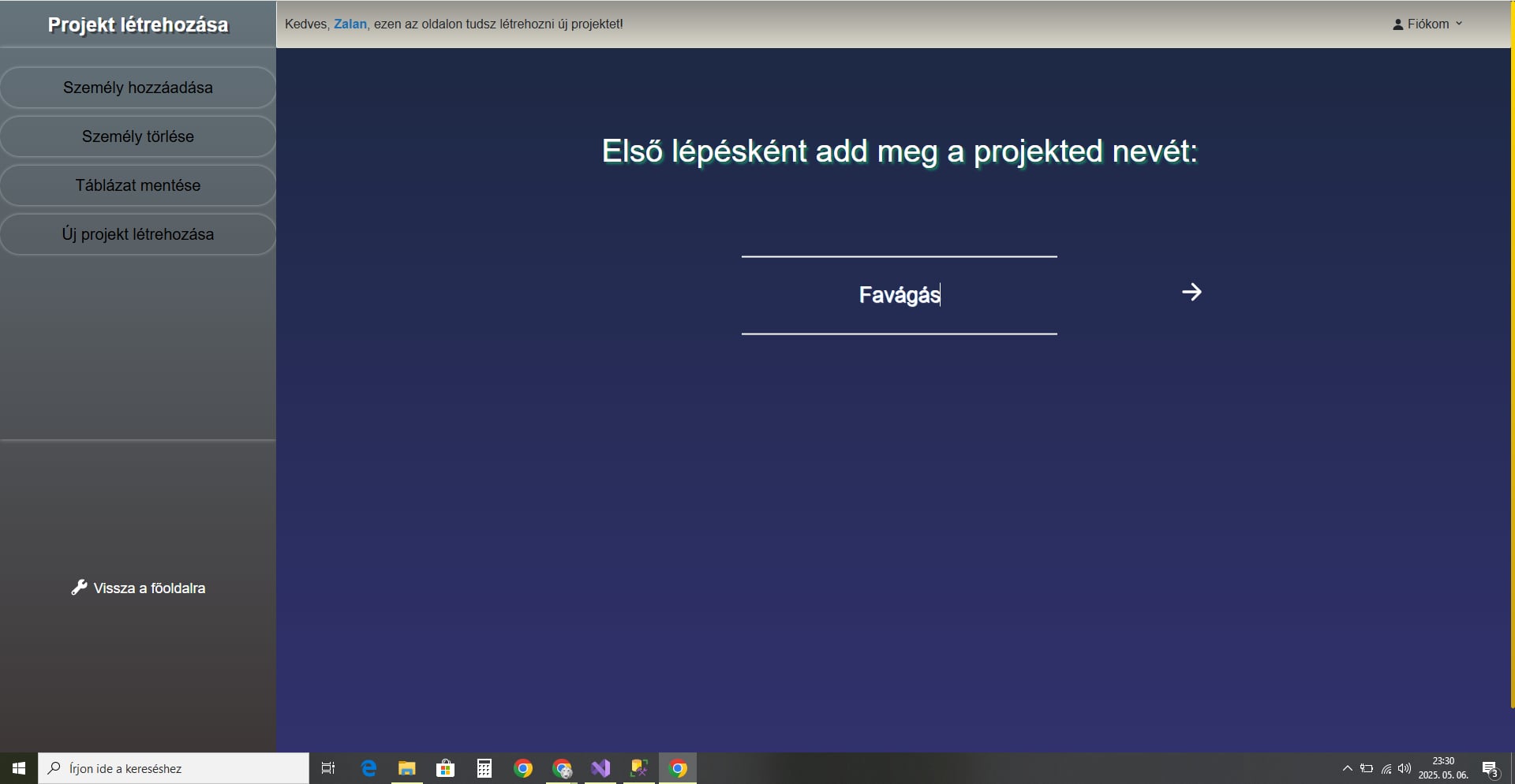
Ezek a feltehetően a felhasználó projektjei, amelyeket ki lehet választani. Mellettük a „Notiz kód” azonosítók jelennek meg.

### **Menüpontok:**

* **„Notiz létrehozása”**
  + Új projekt létrehozására szolgál.
* **„Notiz módosítása”**
  + A kiválasztott projekt szerkesztése.
* **„Notiz hozzáadása”**
  + Új elem vagy alprojekt hozzáadása.
* **„Notiz törlése”**
  + A kiválasztott projekt törlése.

**Táblázat funkciója:** A projekthez tartozó feladatokat listázza a következő adatokkal:

* **Tulajdonos:** Ki felel a feladatért
* **Feladat:** A feladat rövid leírása
* **Állapot:** A feladat aktuális állapota
* **Kezdés / Határidő:** Mikor kezdődik és mikorra kell befejezni
* **Fontosság:** Jelzi, mennyire sürgős vagy fontos
* **Komment:** Megjegyzések a feladathoz



## ****Oldal neve: Projekt létrehozása****

Ez az oldal egy új projekt létrehozását szolgálja a LilNotiz alkalmazásban. A célja, hogy a felhasználó néhány egyszerű lépésben megadja a projekt adatait és résztvevőit.

### **1. Oldal felépítése**

#### **Bal oldali navigációs panel:**

Ez a rész egy több lépéses folyamatot vezet végig, ahol a felhasználó megadhatja a projekt részleteit.

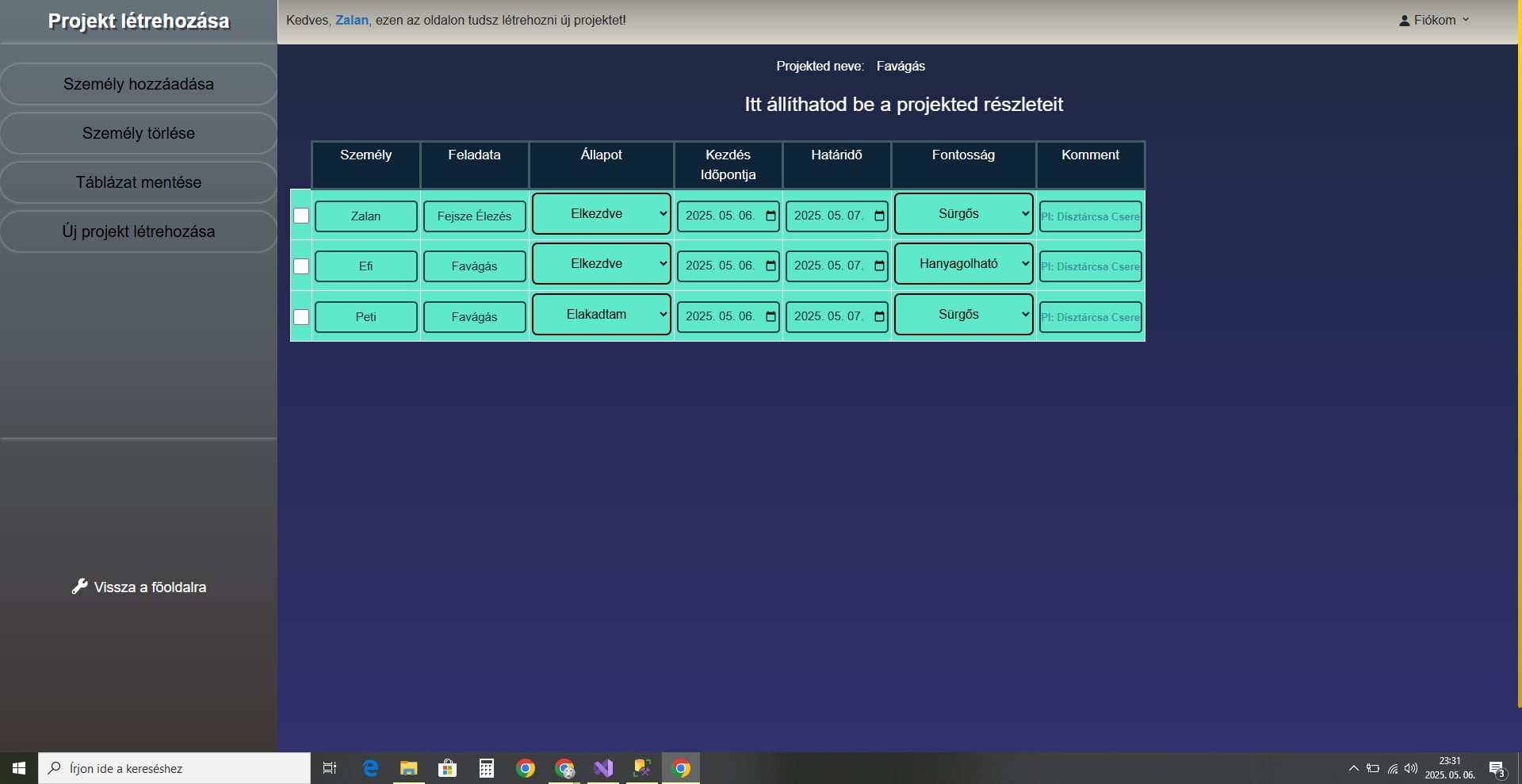
* **Személy hozzáadása**: Gomb, amely egy oldalra visz, ahol új csapattagot lehet felvenni a projekthez.
* **Személy törlése**: Gomb, amellyel a hozzáadott személyek törölhetők.
* **Táblázat mentése**: Elmenti az aktuális projekt struktúráját (csapattagok és feladataik).
* **Új projekt létrehozása**: Visszavisz a projekt létrehozási folyamat elejére, új projekt indításához.
* **Vissza a főoldalra** (alul): Gomb, amely visszairányítja a felhasználót az alkalmazás kezdőlapjára vagy dashboardjára.

#### **Felső sáv:**

* **Szöveg:** „Kedves Zalan, ezen az oldalon tudsz létrehozni új projektet.”  
  → A felhasználót személyesen megszólítja, ami segít a felhasználói élmény javításában. A név dinamikusan kerülhet megjelenítésre (pl. bejelentkezett felhasználó alapján).
* **Profil ikon (jobb felső sarokban)**: Egy menüt nyit meg a felhasználói fiókhoz kapcsolódó lehetőségekkel (profil, kijelentkezés, beállítások stb.).

#### **Fő tartalom (középen):**

* **Cím:** „Első lépésként add meg a projekted nevét:”  
  → Ez irányítja a felhasználót, hogy az első lépés a projekt elnevezése.
* **Szövegmező:** Már tartalmazza a „Favágás” nevű projektcímet.  
  → Ide kell beírni az új projekt nevét. A felhasználó által szabadon választott szöveg.
* **Jobbra mutató nyíl gomb (→):**  
  → A következő lépésre visz a projekt létrehozás folyamatában a személyek hozzáadásához.



### **1. Oldal elrendezése és funkciói**

#### C:\Users\Én\Downloads\fcde81c1-56d1-4a73-bdca-ec09c7ab16fa.jpg**Bal oldali navigációs panel (megegyezik az előző képernyővel):**

* **Személy hozzáadása**: Új csapattag hozzáadása a táblázathoz.
* **Személy törlése**: Kijelölt személy törlése a projektből.
* **Táblázat mentése**: Az aktuálisan módosított projektadatok mentése.
* **Új projekt létrehozása**: Új projekt indítása.
* **Vissza a főoldalra**: Visszavisz a főoldalra.

#### **Felső sáv:**

* **Felhasználó köszöntése:** „Kedves Zalan, ezen az oldalon tudsz létrehozni új projektet”
* **Profil ikon (jobb felső sarok)**: Fiókhoz kapcsolódó opciók elérése.
* **Projekt neve megjelenítve:** „Projekt neve: Favágás”  
  → Ez az aktuálisan szerkesztett projekt neve.
* **Oldalcím:** „Itt állíthatod be a projekted részleteit”  
  → Jelzi, hogy itt van lehetőség a projekt részletes beállításaira.

### **2. Központi tartalom: Projekttábla**

Ez a szakasz egy **interaktív táblázat**, amelyben minden sor egy személyhez tartozik, és különböző projektadatokat jelenít meg.

| **Oszlop neve** | **Leírás** |
| --- | --- |

|  |  |
| --- | --- |
| **Személy** | A csapattag neve (pl. Zalan, Eli, Peti). |

|  |  |
| --- | --- |
| **Feladat** | Az adott személyhez rendelt feladat (pl. Faipari Eszköz, Favágás). |

|  |  |
| --- | --- |
| **Állapot** | A feladat állapota: pl. „Elkészült”, „Elakadtam”. |

|  |  |
| --- | --- |
| **Kezdés időpontja** | A feladat kezdetének dátuma (pl. 2025. 05. 06). |

|  |  |
| --- | --- |
| **Határidő** | A feladat befejezési határideje (pl. 2025. 05. 07). |

|  |  |
| --- | --- |
| **Fontosság** | A feladat prioritása (pl. „Sürgős”, „Nagyon sürgős”). |

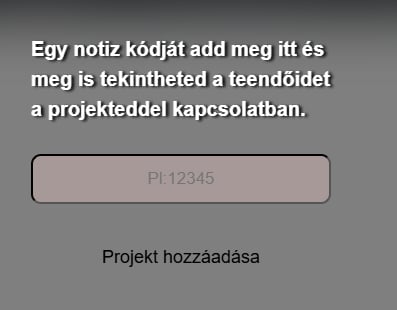
|  |  |
| --- | --- |
| **Komment** | További megjegyzések a feladathoz (pl. „Elsőként ezt kell csinálni”). |

#### **Táblázat színezése és vizuális elemek:**

* A sorok világoskék háttérrel jelennek meg –a jobb olvashatóság érdekében.
* Az állapot és fontosság cellák dropdown (lenyíló lista) elemeknek tűnnek.
* Kommentmezők szövegbeviteli mezők vagy szintén lenyílóak.

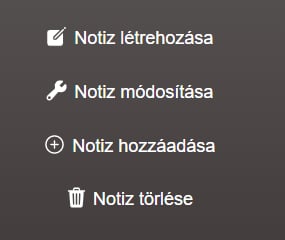
### **3. Funkcionalitás összegzése**

| **Elem** | **Funkció** | **Leírás** |
| --- | --- | --- |
| Személy neve | Információ | Ki végzi a feladatot. |
| Feladat | Szövegmező | A személyhez rendelt konkrét feladat. |
| Állapot | Dropdown | Állapotváltozás követése (pl. kész, elakadt). |
| Dátum mezők | Naptár vagy input | Kezdés és határidő beállítása. |
| Fontosság | Dropdown | A feladat prioritásának jelzése. |
| Komment | Szövegdoboz | További leírás vagy megjegyzés a feladathoz. |



A ”Notiz hozzáadása” gombra rákattintva felugrik az ablak, amely a notizkódot kéri

Ha egy Notizkódot módunkban áll elkérni, annak ide beillesztésével számunkra is látható lesz a jegyzet



### Notiz műveletek:

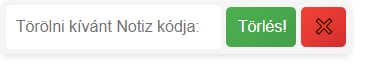
A menü négy elemből áll, mindegyik ikon és szöveg kombinációja, amelyek különböző műveletek végrehajtására szolgálnak egy „Notiz” nevű entitáson.

### Menüelemek:

1. **📝 Notiz létrehozása**
   * **Ikon**: Ceruza papíron
   * **Funkció**: Új Notiz (jegyzet vagy bejegyzés) létrehozása.
   * **Hatás**: megnyit egy új űrlapot vagy modált, ahol a felhasználó új Notiz-t rögzíthet.
2. **🔧 Notiz módosítása**
   * **Ikon**: Csavarkulcs
   * **Funkció**: Meglévő Notiz szerkesztése.
   * **Hatás**: Lehetővé teszi egy kiválasztott Notiz adatainak módosítását.
3. **➕ Notiz hozzáadása**
   * **Ikon**: Plusz jel
   * **Funkció**: Egy új Notiz hozzáadása egy meglévő lista vagy projekt keretéhez.
   * **Megjegyzés**: Funkciója hasonló lehet a létrehozáshoz, de kontextus-függő (pl. hozzárendelés).
4. **🗑️ Notiz törlése**
   * **Ikon**: Szemetes kuka
   * **Funkció**: Meglévő Notiz végleges törlése.
   * **Hatás**: A felhasználó kiválaszt egy Notiz-t, amely ezután eltávolításra kerül az adatbázisból.

### Általános jellemzők:

* **Stílus**: Letisztult, szürke háttéren fehér ikonok és szöveg.
* **Elrendezés**: Függőleges menü, jól strukturált, egységes megjelenésű.
* **Felhasználás célja**: Egyszerű és gyors hozzáférés a Notiz-ek kezeléséhez.



### Notiz törlése:

#### 1. **Szövegmező (input mező)**

* **Felirat**: „Törölni kívánt Notiz kódja:”
* **Funkció**: A felhasználó ebbe a mezőbe írja be annak a jegyzetnek (Notiz) a kódját, amelyet törölni szeretne.
* **Típus**: Szöveges bemeneti mező (input type="text").

#### 2. **Zöld gomb**

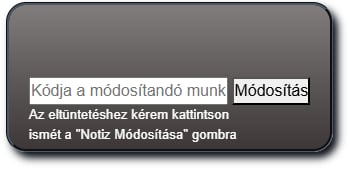
* **Felirat**: „Törlés!”
* **Szín**: Zöld (#28a745 körüli árnyalat)
* **Funkció**: A gomb megnyomásával végrehajtódik a törlési művelet. Az alkalmazás törli az adatbázisból azt a jegyzetet, amelynek kódja a szövegmezőbe lett beírva.
* **UI viselkedés**: Sikeres törlés után megjelenik egy visszajelzés

#### 3. **Piros gomb**

* **Ikon**: Piros színű négyzet, benne egy X ikon.
* **Funkció**: A gomb megnyomására a művelet megszakad, azaz a törlési folyamat megszüntethető vagy visszavonható.
* **Alternatív funkció**: Lehetséges, hogy a piros gomb a törlési ablak bezárására szolgál.

### Elrendezés és dizájn

* A mezők egy sorban helyezkednek el, ami letisztult és felhasználóbarát kezelőfelületet eredményez.
* A színek szándékosan utalnak a funkcióra: zöld a megerősítéshez, piros a megszakításhoz vagy figyelmeztetéshez.
* A vizuális hierarchia egyértelmű: a törlés művelet hangsúlyos (zöld gomb), a megszakítás másodlagos (piros ikon).



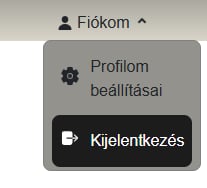
### Notiz módosítása funkció:

#### 1. **Szövegmező (input mező)**

* **Felirat (részben látható)**: „Kódja a módosítandó munk...” – feltehetően: „Kódja a módosítandó munkának” vagy „jegyzetnek”.
* **Funkció**: Ide kell beírni annak a jegyzetnek a kódját, amelyet a felhasználó módosítani szeretne.
* **Típus**: Szöveges bemeneti mező.

#### 2. **Gomb**

* **Felirat**: „Módosítás”
* **Szín**: Alap stílusú (nincs színes háttér, fekete-fehér dizájn)
* **Funkció**: A gomb megnyomásával elindul a megadott kódú jegyzet módosítási folyamata.
* **Várható működés**: A gomb aktivál egy szerkesztő nézetet vagy betölti a jegyzetet szerkeszthető formában.



### Fiók legördülő sáv:

* **Főmenü felirat**: „Fiókom” – lenyíló menü ikon kíséretében
* **Lenyíló elemek**:
  1. **Profilom beállításai** – Fogaskerék ikon, profil testreszabására vezet.
  2. **Kijelentkezés** – Kilépés ikon, a felhasználó kijelentkezik a rendszerből.

### Funkció:

* **Felhasználói fiókkezelés**: A menü lehetőséget ad a profilbeállítások megnyitására vagy a rendszerből való kilépésre.



**Módosítandó notiz:**

**autoszerelés** – A tábla címe, amely a modul témáját jelöli (autószerelési feladatok nyilvántartása).

### Adattábla oszlopai:

1. **Tulajdonos** – Az érintett személy neve.
2. **Feladat** – A szerelési feladat megnevezése.
3. **Állapot** – A feladat aktuális státusza (pl. „Kész”, „Nincs elkezdve”, „Elakadtam”).
4. **Mikor kezdje** – A feladat megkezdésének időpontja (formátum: év. hó. nap. óra:perc:mp).
5. **Határidő** – A feladat teljesítésének határideje, ugyanolyan formátumban.
6. **Fontosság** – A feladat fontossági szintje (pl. „Fontos”, „Sürgős”).
7. **Komment** – Szabad szövegű megjegyzés (pl. „Nem kaptam választ”).

### Funkcionalitás:

* Ez a nézet a feladatok módosítására szolgál.
* dupla kattintás esetén a következő módon tudjuk módosítani:



### Változások és funkciók:

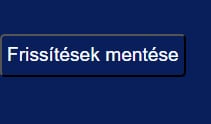
* Ezen képen látható, hogy a táblázat szerkeszthető módban van.
* Az első sor (Zalán) mezői szerkeszthetővé váltak, az alábbi módon:

#### Interaktív mezők:

1. **Tulajdonos**: szövegmező – pl. „Zalán”
2. **Feladat**: szövegmező – a felhasználó épp írja: „akami m…”
3. **Állapot**: legördülő lista – kiválasztott érték: „Kész”
4. **Mikor kezdje**: dátum-idő mező – érték: 2025. 05. 06. 00:00:00
5. **Határidő**: dátum-idő mező – érték: 2025. 05. 07. 00:00:00
6. **Fontosság**: legördülő lista – kiválasztott: „Fontos”
7. **Komment**: szövegmező – jelenleg üres

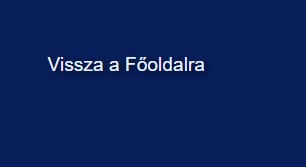
### Funkcionalitás:

* A táblázat sorai szerkeszthetők, gombnyomás hatására.
* A változtatásokat egy mentés funkció rögzíti az átírt adatokat



### Frissítések mentése:

* **Felirat**: „Frissítések mentése”
* **Típus**: Gomb (button)
* **Stílus**: Sötét háttér világos szöveggel, enyhe keret vagy árnyék
* **Funkció**: A szerkesztett vagy módosított adatok mentésére szolgál. A felhasználó ezzel a gombbal véglegesíti az aktuális változtatásokat (pl. táblázat sorainak módosítása).



### Vissza a főoldalra link:

* **Felirat**: „Vissza a Főoldalra”
* **Típus**: Navigációs link vagy gomb
* **Stílus**: Egyszerű szöveg, linkként működik (kattintható)
* **Funkció**: A felhasználót visszairányítja a kezdőlapra vagy főmenüre.

#### Helytelen használatból adódó hibajelzések magyarázata

A rendszer különböző helytelen felhasználói interakciók esetén hibajelzéseket küld a felhasználóknak, hogy jelezze, ha valamilyen adat nem található, vagy ha a szükséges információk hiányoznak. Az alábbiakban ismertetjük a leggyakoribb hibajelzéseket és azok jelentését:

1. **Bejelentkezéskor helytelen adat megadása:**
   1. Ha a felhasználó olyan adatokat ad meg a bejelentkezés során, amelyek nem találhatóak meg az adatbázisban (például helytelen felhasználónév vagy jelszó), a rendszer hibaüzenetet jelenít meg. A hibaüzenet figyelmeztet arra, hogy a megadott adatok nem egyeznek a rendszerben tároltakkal, és újra kell próbálni helyes adatokat megadva.
   2. **Hibaüzenet példa:** *"Helytelen felhasználónév vagy jelszó. Kérjük, próbálja újra."*
2. **Regisztrációkor hiányos adatok:**
   1. Ha a felhasználó regisztrációkor nem ad meg minden szükséges adatot (például név, e-mail cím, vagy egyéb kötelező mezők), akkor a rendszer hibaüzenetet küld, amely tájékoztatja, hogy a regisztráció nem sikerült, mert a megadott adatok nem teljesek.
   2. **Hibaüzenet példa:** *"Kérjük, töltse ki az összes kötelező mezőt!"*
3. **Nem létező NotizID keresése:**
   1. Ha a felhasználó olyan NotizID-ra próbál rákeresni, amely nem létezik az adatbázisban, a rendszer egy hibaüzenetet ad, amely figyelmeztet, hogy az adott NotizID nem található. Ez segít a felhasználónak abban, hogy helyes NotizID-t adjon meg a kereséshez.
   2. **Hibaüzenet példa:** *"A megadott NotizID nem található. Kérjük, ellenőrizze a megadott kódot!"*
4. **Helytelen NotizID törléskor:**
   1. Ha a felhasználó olyan NotizID-t próbál törölni, amely nem található meg az adatbázisban, a rendszer szokásos hibaüzenetet ad, amely figyelmeztet arra, hogy a megadott NotizID helytelen vagy nem létezik.
   2. **Hibaüzenet példa:** *"A törölni kívánt NotizID nem található. A törlés nem sikerült"*
5. **Hibás adat bevitel új Notiz létrehozásakor:**
   1. Ha a felhasználó új Notiz létrehozása során üresen hagyja a "Projekt neve" mezőt, a rendszer nem engedi tovább a műveletet, és egy hibaüzenet jelenik meg, amely figyelmezteti a felhasználót a kötelező mező hiányára.
   2. **Hibaüzenet példa:** *"A projekt neve mező kitöltése kötelező. Kérjük, adjon meg egy projekt nevet!"*

#### Információkérés lehetőségeinek megadása

A felhasználók számára az alábbi módokon biztosítunk alapvető tájékoztatást a webalkalmazásról:

* **Bemutatkozó szöveg a bejelentkezési felületen:**  
  Ismerteti az oldal célját, a jegyzetkezelés szerepét, valamint rövid áttekintést ad a projektmenedzsment alapjairól.
* **Projekt struktúrájának leírása:**  
  Segít megérteni, hogyan épül fel egy projekt, mik a fő elemei (pl. cél, időbeosztás, erőforrások), és hogyan illeszkedik mindez a Lil'Notiz rendszerébe.
* **Egyszerű, felhasználóbarát felület:**  
  A felület kialakítása önmagában is segíti a tájékozódást – a funkciók logikusan követik egymást, így külön útmutató nélkül is könnyen használható az oldal.

Bár külön súgó vagy GYIK nem áll rendelkezésre, de a bejelentkezés alatti rövid bemutatkozás egy átláthatóbb kezelőfelület nyújt a kezdő felhasználók számára a rendszer használatának elsajátítására.

VISZONT!

A jövőben tervezzük e-mail alapú ügyfélszolgálat és helpdesk rendszer bevezetését is — ha majd egyszer befut a weboldal.

## Összefoglalás

### Elkészült munkánk értékelése

Az elkészült projekt visszatekintve nem csupán egy technikai feladat, hanem egy komplex tanulási folyamat leképezése is volt. A kitűzött szakmai célokat sikerült strukturáltan és következetesen végigvinnünk: egy reszponzív, esztétikailag igényes, modern felhasználói felületet hoztunk létre, amely mind funkcionálisan, mind vizuálisan megállja a helyét a webes alkalmazások világában.

A legnagyobb kihívást egyértelműen a különböző technológiák összefűzése jelentette – különösen a JavaScript eseménykezelés és az adaptív dizájn összehangolása során. Ezek a részek azonban egyben a legtöbbet is tanították nekünk: megtapasztalhattuk, hogyan válik egy ötlet működőképes, interaktív webes élménnyé, és közben mélyebb technikai tudásra is szert tettünk.

A munka során nemcsak a front-end fejlesztés alapjait sikerült elmélyítenünk, hanem ráébredtünk arra is, hogy a felhasználói élmény mennyire kulcsfontosságú egy alkalmazás sikerében. A jövőben szeretnénk ezt a projektet továbbfejleszteni, például adatbázis-kapcsolattal, autentikációval, vagy akár admin felülettel kiegészítve – így teljes körű, életszerű alkalmazássá válhat.

Ezúton szeretnénk köszönetet mondani mindazoknak, akik támogattak minket ebben a folyamatban: oktatóinknak a szakmai útmutatásért, családtagjainknak pedig a türelemért és háttérország biztosításáért. Minden támogatás, akár technikai, akár lelki, hozzájárult ahhoz, hogy ezt a munkát nemcsak elkészíteni tudtuk, hanem valóban a sajátunknak is érezzük.

### Köszönetnyilvánítás

Ezúton szeretnénk kifejezni őszinte hálánkat és nagyrabecsülésünket Méhes József és Szalainé Török Edit tanáraink iránt, aki türelemmel, szakértelemmel és támogató hozzáállással segített minket a fejlesztés során. Minden hasznos tanács, visszajelzés és útmutatás, amit tőle kaptunk, jelentősen hozzájárult a projekt sikeréhez és a saját szakmai fejlődésünkhöz is.

Továbbá köszönetet mondunk azoknak a barátainknak, családtagjainknak, akik türelmesen viselték, hogy a projekt idején néha hosszú órákra eltűntünk a kódsorok és tesztelések fejlesztése végett. Az ő megértésük és támogatásuk nélkül ez a munka jóval nehezebben valósulhatott volna meg.

Szeretnénk megköszönni a közösségi fórumokon, fejlesztői dokumentációkban és open-source platformokon aktívan tevékenykedő fejlesztők munkáját, hiszen az ő tudásmegosztásuk lehetővé tette, hogy tanuljunk, ihletet merítsünk, és elkerüljünk olyan hibákat, amik tapasztalat híján talán sokáig akadályoztak volna minket.

**Végül, de nem utolsósorban: köszönjük szépen a megtisztelő figyelmet!**  
Reméljük, hogy munkánk nemcsak technikailag állta meg a helyét, hanem inspiráló vagy tanulságos is lehet mások számára.

## Irodalomjegyzék

Űrlap-elemek megvalósítása HTML-ben

[http://**infojegyzet.hu/**webszerkesztes/html/urlapok/](http://infojegyzet.hu/webszerkesztes/html/urlapok/)

PHP alapok - szerveroldali programozás

<http://infojegyzet.hu/webszerkesztes/php/alapok/>

Dinamikus képgaléria

<http://infojegyzet.hu/webszerkesztes/php/kepgaleria/>

Záródolgozat témák, szempontok, ötletek

<http://infojegyzet.hu/webszerkesztes/zarodolgozat/>

Az irodalomjegyzék általában rá szokott férni 1 oldalra.