Dokumentáció

Epic.CRM

Szoftverfejlesztő és -Tesztelő Technikus Szakma Vizsgaremek

Csapatnév: Hősies meghiúsulás

Csapattagok: Balázs Endre, Tóth Sándor, Fodor Gergő

Vizsgaremek témája: Mini webes vállalatirányítási rendszer kisvállalkozók részére



Tartalom

[Bevezetés, a téma ismertetése, témaválasztás indoklása 3](#_Toc187660146)

[Fejlesztői dokumentáció 4](#_Toc187660147)

[Felhasznált technológiák 4](#_Toc187660148)

[Fejlesztőkörnyezet 4](#_Toc187660149)

[Kialakított adatszerkezet 5](#_Toc187660150)

[Backend Szoftver architektúra 10](#_Toc187660151)

[WebApi architektúra 12](#_Toc187660152)

[Frontend Architektúra 18](#_Toc187660153)

[Továbbfejlesztési lehetőségek 19](#_Toc187660154)

[Felhasználói dokumentáció 21](#_Toc187660155)

[Nyelvváltás 21](#_Toc187660156)

[Bejelentkezés 22](#_Toc187660157)

[Felhasználó kezelés (Adminisztrátor) 23](#_Toc187660158)

[Munkáim 25](#_Toc187660159)

[Ügyfeleim 27](#_Toc187660160)

[Naptár 29](#_Toc187660161)

[Az alkalmazás használata Mobilon 30](#_Toc187660162)

[Tesztelés 32](#_Toc187660163)

[Összefoglalás 35](#_Toc187660164)

# Bevezetés, a téma ismertetése, témaválasztás indoklása

Célunk egy olyan többrétegű webes technológiájú alkalmazás elkészítése, melynek segítségével a regisztrált kisvállalkozók kezelni tudják az ügyélkörüket, valamint a munkáikat.

A mai digitalizált világban azok a vállalkozások, melyek informatikai eszközök használatával támogatják munkáikat tetemes előnyre tehetnek szert a versenytársaikkal szemben. Ám mégis az a tapasztalat, hogy nagyon sok (főként egyéni) vállalkozó továbbra is a „papír-ceruza-kockásfüzet” módszertannal tartja számon munkáit, illetve ügyfeleit. Ez a módszer ma már nem túl hatékony, gondoljunk csak arra, ha egy ügyfél elköltözik, a munkát másik napra kell átütemezni, vagy esetleg keresni kell a sok információ között. Ám akkor mégis miért nem váltottak még ők valami digitális megoldásra? A válasz az, hogy vagy nincs tudomásuk ezekről a lehetőségekről, vagy nem tudnak eligazodni a piacon kínált eszközök között, illetve sok esetben az elérhető eszközök a túl sok és bonyolult funkció miatt nem felelnek meg az említett vállalkozók sokkal egyszerűbb elvárásainak. Ezeknek a vállalkozóknak (ácsoknak, kőműveseknek, asztalosoknak, fodrászoknak, stb) készült az Epic.CRM nevű mini vállalatirányítási rendszer.

Az elkészült alkalmazás az interneten keresztül érhető el. Adminisztrátori szerepkörrel lehet vállalkozókat rögzíteni a rendszerben, akik ezt követően email cím és jelszó megadást követően tudnak belépni a saját felületükre. Ezen a felületen tudják kezelni a munkáikat, valamint az ügyfélkörüket.

Az elkészült alkalmazás kétnyelvű. A felhasználóknak lehetőségük van nyelvet választani. Jelenleg az alkalmazás a magyar és az angol nyelvet támogatja.

# Fejlesztői dokumentáció

## Felhasznált technológiák

* **Backend (.NET CORE):** A backend RESTful API-kat biztosít az adatkezeléshez. A technológiák közé tartozik az ASP.NET Core MVC, Entity Framework Core (adatbázis-hozzáférés). A biztonság JWT (JSON Web Token) alapú hitelesítéssel van biztosítva.
* **Frontend (Angular):** Az Angular keretrendszer biztosítja a felhasználói felületet. A fejlesztés során TypeScript, HTML és CSS kerültek felhasználásra. Keretrendszerek közül az Angular Material Framework-öt használjuk
* **Adatbázis:** A rendszer egy relációs adatbázist használ (Microsoft SQL Server 2019) a következő táblákkal: Felhasználók, Munkák, Ügyfelek, Címek, Munkastátuszok. A táblák között megfelelő relációs kapcsolatok vannak megvalósítva a hatékony adatkezelés érdekében. Ezeken kívül technikai táblák is felhasználásra kerültek az alkalmazás működése érdekében, melyekről részletesebben a további fejezetekben írunk.

## Fejlesztőkörnyezet

Ez a fejezet az Epic.CRM fejlesztéséhez használt hardveres és szoftveres környezetet részletesen leírja. A konzisztens és hatékony fejlesztés érdekében a csapat egységes fejlesztői környezetet alkalmaz.

**Szoftver:**

* **Operációs rendszer:** Microsoft Windows 10 Professional 64-bit, 21H2 build.
* **Integrált fejlesztőkörnyezet (IDE):** Microsoft Visual Studio 2022 Professional, verzió 17.4.5.
* **.NET Runtime:** .NET 8.0 SDK.
* **Adatbázis szerver:** Microsoft SQL Server 2019 Standard Edition, verzió 15.0.2080.27. A SQL Server Management Studio (SSMS) verziója 19.0. Az adatbázis a EPIC\_CRM néven fut, és a master adatbázison belül található.
* **Frontend fejlesztőkörnyezet:** Node.js v16.20.0 LTS, npm 9.8.1. Az Angular CLI verziója 16.1.8.
* **Verziókövetés:** GitLab Community Edition, verzió 15.10.3. A projekt repository címe: https://github.com/Endroo/Epic.CRM. A fő ág neve: main.
* **Terheléses tesztelés:** k6 v0.39.0.
* **Egyéb eszközök:** Postman v10.9.1 (API teszteléshez)

**Konfiguráció:**

A fejlesztői környezet felállítása a következő lépésekből áll:

1. **Operációs rendszer telepítése:** A felsorolt Windows 10 verzió telepítése.
2. **Szoftverek telepítése:** A felsorolt szoftverek telepítése a megadott verziószámokkal.
3. **Adatbázis létrehozása:** Az EPIC\_CRM adatbázis létrehozása a SQL Serverben a megfelelő séma segítségével. A séma leírása a GitLab repository database/schema.sql fájljában található.
4. **Projekt klónozása:** A GitLab repository klónozása a fejlesztői gép egy helyi mappájába.
5. **Függőségek telepítése:** Az Angular projekt függőségeinek telepítése (npm install vagy yarn install).
6. **.NET projekt build-elése:** A .NET projekt build-elése (dotnet build).

**Frissítések:**

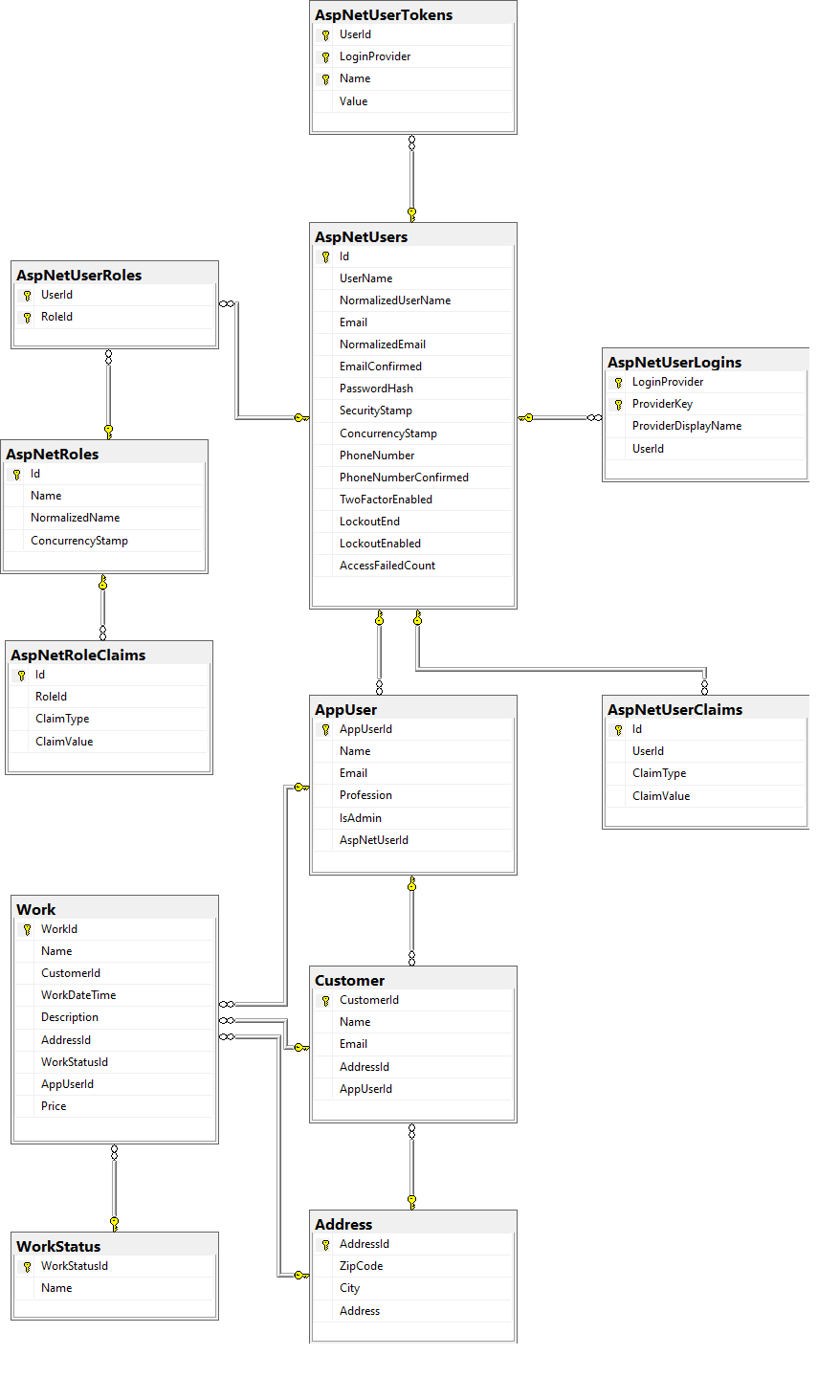
A szoftverek frissítéseit a fejlesztői csapat koordinálja. Minden frissítés előtt tesztelni kell a rendszer működését az új verziókkal. A frissítések a GitLab release notes-ban lesznek dokumentálva.

Ez a dokumentáció segít a fejlesztőknek a fejlesztői környezet felállításában az Epic.CRM projekthez.

## Kialakított adatszerkezet

Az alkalmazás adatait Microsoft SQL Server 2019 relációs adatbázisban tároljuk. Az üzleti és technikai információk egyaránt ebben az adatbázisban kapnak helyet.

Az adatbázis magas szintű adatmodellje az alábbi ábrán látható:



Az alábbiakban pedig a táblák részletes leírása olvasható:

**Address tábla**

Címadatok tárolására használt tábla.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| AddressId | int | Nem | Egyedi azonosítója a címnek |
| ZipCode | int | Igen | Irányítószám |
| City | nvarchar | Igen | Város |
| Address | nvarchar | Igen | Cím részetesen |

**AppUser tábla**

Felhasználók (vállalkozók és adminisztrátorok) tárolására használt tábla.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| AppUserId | int | Nem | Felhasználó egyedi adatbázis azonosítója |
| Name | nvarchar | Nem | A felhasználó neve |
| Email | nvarchar | Igen | Felhasználó email címe |
| Profession | nvarchar | Nem | A felhasználó tevékenységi köre |
| IsAdmin | bit | Nem | A felhasználó Admin szerepkörben tevékenykedik-e |
| AspNetUserId | nvarchar | Nem | A felhasználóhoz kapcsolt authentikációra/authorizációra használt ASP.NET Identity AspNetUserId objektum |

**AspNetRoles**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| Id | nvarchar(450) | Nem | Egyedi azonosítója a szerepnek |
| Name | nvarchar(256) | Igen | A szerep neve |

**AspNetRoleClaims**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| Id | int | Nem | Egyedi azonosító a szerephez kapcsolódó bejegyzéshez |
| RoleId | nvarchar(450) | Nem | Hivatkozás az AspNetRoles táblára |
| ClaimType | nvarchar(max) | Igen | A claim típusa |
| ClaimValue | nvarchar(max) | Igen | A claim értéke |

**AspNetUsers**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| Id | nvarchar(450) | Nem | Egyedi azonosító a felhasználó számára |
| UserName | nvarchar(256) | Igen | A felhasználóneve |
| Email | nvarchar(256) | Igen | A felhasználó e-mail címe |
| PasswordHash | nvarchar(max) | Igen | A jelszó hash-elt verziója |
| SecurityStamp | nvarchar(max) | Igen | Véletlenszerű érték a biztonság érdekében |
| NormalizedUserName | nvarchar(256) | Igen | Normalizált felhasználónév |
| NormalizedEmail | nvarchar(256) | Igen | Normalizált e-mail cím |

**AspNetUserClaims**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| Id | int | Nem | Egyedi azonosító a felhasználóhoz kapcsolódó bejegyzéshez |
| UserId | nvarchar(450) | Nem | Hivatkozás az AspNetUsers táblára |
| ClaimType | nvarchar(max) | Igen | A claim típusa |
| ClaimValue | nvarchar(max) | Igen | A claim értéke |

**AspNetUserLogins**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| LoginProvider | nvarchar(450) | Nem | Az azonosítási szolgáltató neve |
| ProviderKey | nvarchar(450) | Nem | Az azonosítási szolgáltató által biztosított felhasználói kulcs |
| ProviderDisplayName | nvarchar(max) | Igen | Az azonosítási szolgáltató megjelenített neve |
| UserId | nvarchar(450) | Nem | Hivatkozás az AspNetUsers táblára |

**AspNetUserRoles**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| UserId | nvarchar(450) | Nem | Hivatkozás az AspNetUsers táblára |
| RoleId | nvarchar(450) | Nem | Hivatkozás az AspNetRoles táblára |

**AspNetUserTokens**

ASP .NET sztenderd authentikációhoz és authorizációhoz használt technikai tábla

|  |  |  |  |
| --- | --- | --- | --- |
| **Mező neve** | **Adattípus** | **Null érték megengedett** | **Leírás** |
| UserId | nvarchar(450) | Nem | Hivatkozás az AspNetUsers táblára |
| LoginProvider | nvarchar(450) | Nem | Az azonosítási szolgáltató neve |
| Name | nvarchar(450) | Nem | A token neve |
| Value | nvarchar(max) | Igen | A token értéke |

**Customer Tábla**

Vállalkozó ügyfeleinek tárolására használt tábla.

|  |  |  |
| --- | --- | --- |
| Oszlop név | Adattípus | Leírás |
| CustomerId | int | Egyedi azonosító. |
| Name | nvarchar(250) | Az ügyfél neve. |
| Email | nvarchar(250) | Az ügyfél email címe. |
| AddressId | int | Hivatkozás az Address táblára (Foreign Key). |
| AppUserId | int | Hivatkozás az AppUser táblára (Foreign Key). |

**Work Tábla**

Vállalkozó munkáinak tárolására használt tábla.

|  |  |  |
| --- | --- | --- |
| Oszlop név | Adattípus | Leírás |
| WorkId | int | Egyedi azonosító. |
| Name | nvarchar(250) | A munka neve. |
| CustomerId | int | Hivatkozás a Customer táblára (Foreign Key). |
| WorkDateTime | datetime | A munka időpontja. |
| Description | nvarchar(MAX) | A munka leírása. |
| AddressId | int | Hivatkozás az Address táblára (Foreign Key). |
| WorkStatusId | int | Hivatkozás a WorkStatus táblára (Foreign Key). |
| AppUserId | int | Hivatkozás az AppUser táblára (Foreign Key). |
| Price | int | A munka ára. |

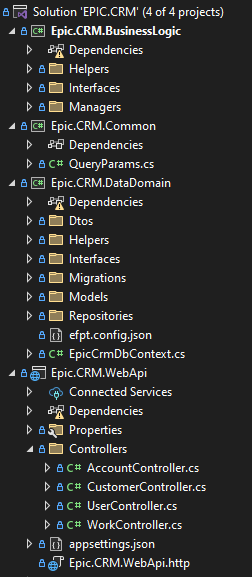
**WorkStatus Tábla**

A munkák lehetséges státuszait tartalmazó felsorolás-típusú tábla

|  |  |  |
| --- | --- | --- |
| Oszlop név | Adattípus | Leírás |
| WorkStatusId | int | Egyedi azonosító. |
| Name | nvarchar(50) | A munka státuszának neve. |

## Backend Szoftver architektúra

Az Epic.CRM egy több rétegű (n-tier) architektúrát követ, amely a fejlesztés, karbantartás és skálázhatóság szempontjából előnyös. A rendszer a következő főbb rétegekből áll:



**Epic.CRM.BusinessLogic**

Ez a réteg tartalmazza az alkalmazás üzleti logikáját. A következő alrétegekre bontható:

* **Dependencies:** Külső függőségek kezelése.
* **Helpers:** Segédfüggvények.
* **Interfaces:** Interfészek a különböző rétegek közötti kommunikációhoz.
* **Managers:** Az üzleti logika implementációja.

**Epic.CRM.Common**

Ez egy közös komponenseket tartalmazó projekt.

* **Dependencies:** Külső függőségek kezelése.
* **Query Params:** Lekérdezési paraméterek kezelése.

**Epic.CRM.DataDomain**

Ez a réteg felelős az adatbázis-hozzáférésért és az adatok kezeléséért.

* **Dependencies:** Külső függőségek kezelése.
* **Dtos:** Adatátviteli objektumok (Data Transfer Objects).
* **Helpers:** Segédfüggvények az adatkezeléshez.
* **Interfaces:** Interfészek a különböző rétegek közötti kommunikációhoz.
* **Migrations:** Az adatbázis migrációit kezelő osztályok.
* **Models:** Az adatbázis táblákhoz tartozó entitások (Entity Framework Core).
* **Repositories:** Az adatbázis interakciót kezelő osztályok (Repository mintázat).
* **efpt.config.json:** Entity Framework Core konfigurációs fájl.
* **EpicCrmDbContext.cs:** Az Entity Framework Core adatbázis kontextus.

**Epic.CRM.WebApi**

Ez a réteg a webes API-t valósítja meg, amely a frontenddel kommunikál.

* **Connected Services:** Kapcsolt szolgáltatások (pl. Identity Server).
* **Dependencies:** Külső függőségek kezelése.
* **Properties:** Konfigurációs beállítások.
* **Controllers:** A webes API végpontokat kezelő kontrollerek (pl. AccountController, CustomerController, UserController, WorkController).
* **appsettings.json:** Az alkalmazás konfigurációs fájlja.
* **Epic.CRM.WebApi.http:** (Swagger dokumentáció)

### WebApi architektúra

#### WorkController

A WorkController osztály a web API végpontokat kezeli a munkák (Work) kezeléséhez. A IWorkManager interfészen keresztül kommunikál az üzleti logikával. Minden végpont ellenőrzi a felhasználó hitelesítését (Authorize attribútum), és csak "Admin" vagy "User" szerepkörrel rendelkező felhasználók férhetnek hozzá.

**Végpontok:**

**1. GET api/Work**

* **Leírása:** Lekéri a bejelentkezett felhasználóhoz tartozó munkák listáját.
* **Kérés paraméterei:** QueryParams objektum (opcionális): page, pageSize, sortBy, sortOrder (oldalszám, oldalméret, rendezési kritériumok).
* **Válasz:** PageResult<IEnumerable<WorkDto>> objektum: A munkák listája oldalszámozva, a teljes elemek számával. 200 OK státuszkód.
* **Hibakezelés:** Ha nincs bejelentkezett felhasználó, 401 Unauthorized státuszkód és ErrorCodes.account\_error\_no\_logged\_user hibaüzenet.
* **Példa:** GET api/Work?page=1&pageSize=10&sortBy=WorkDateTime&sortOrder=desc

**2. POST api/Work/create**

* **Leírása:** Új munka létrehozása.
* **Kérés törzs:** WorkEditRegisterDto objektum: Az új munka adatait tartalmazza.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a kérés törzse érvénytelen, 400 Bad Request státuszkód és a hibák listája. Ha nincs bejelentkezett felhasználó, 401 Unauthorized státuszkód és ErrorCodes.account\_error\_no\_logged\_user hibaüzenet.
* **Példa:** POST api/Work/create (JSON formátumban a WorkEditRegisterDto objektum adataival).

**3. PUT api/Work/{id}**

* **Leírása:** Meglévő munka módosítása.
* **Útvonal paraméter:** id: A módosítandó munka azonosítója.
* **Kérés törzs:** WorkEditRegisterDto objektum: A módosítandó munka adatait tartalmazza.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a kérés törzse vagy az id érvénytelen, 400 Bad Request státuszkód és a hibák listája.
* **Példa:** PUT api/Work/123 (JSON formátumban a WorkEditRegisterDto objektum adataival).

**4. DELETE api/Work/{id}**

* **Leírása:** Meglévő munka törlése.
* **Útvonal paraméter:** id: A törlendő munka azonosítója.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha az id érvénytelen, 400 Bad Request státuszkód és a hibák listája.
* **Példa:** DELETE api/Work/123

**5. GET api/Work/calendar**

* **Leírása:** Lekéri a bejelentkezett felhasználóhoz tartozó munkák listáját naptár nézetben.
* **Válasz:** DataResult<IEnumerable<CalendarDto>> objektum: A munkák listája CalendarDto objektumokként, amelyek tartalmazzák a dátumot, a címet és a leírást. 200 OK státuszkód.
* **Hibakezelés:** Ha nincs bejelentkezett felhasználó, 401 Unauthorized státuszkód és ErrorCodes.account\_error\_no\_logged\_user hibaüzenet.
* **Példa:** GET api/Work/calendar

#### AccountController

Az AccountController osztály a felhasználói fiókok kezeléséhez tartozó web API végpontokat definiálja. A SignInManager<IdentityUser>, UserManager<IdentityUser>, és IAppUserManager osztályokon keresztül kezeli a felhasználói bejelentkezést, kijelentkezést és az aktuális felhasználó adatainak lekérdezését.

**Végpontok:**

**1. GET api/Account/getCurrentUser**

* **Leírása:** Lekéri a bejelentkezett felhasználó adatait.
* **Válasz:** LoggedInUserDto objektum: A bejelentkezett felhasználó neve, email címe és adminisztrátori jogosultsága. 200 OK státuszkód. Üres objektumot ad vissza, ha nincs bejelentkezett felhasználó.
* **Hibakezelés:** Ha a felhasználó nem található a rendszerben, 400 Bad Request státuszkód és ErrorCodes.account\_error\_no\_user\_found hibaüzenet.
* **Példa:** GET api/Account/getCurrentUser

**2. POST api/Account/login**

* **Leírása:** Felhasználói bejelentkezés.
* **Kérés törzs:** LoginDto objektum: A felhasználónév és a jelszó.
* **Válasz:** DataResult<LoggedInUserDto> objektum: A bejelentkezett felhasználó adatai (LoggedInUserDto), vagy hibaüzenet. 200 OK státuszkód sikeres bejelentkezés esetén.
* **Hibakezelés:**
  + Ha a felhasználónév vagy jelszó hiányzik, 400 Bad Request státuszkód és ErrorCodes.account\_error\_username\_or\_password\_null hibaüzenet.
  + Ha a felhasználónév vagy jelszó helytelen, 400 Bad Request státuszkód és ErrorCodes.account\_error\_wrong\_username\_or\_password hibaüzenet.
  + Ha a felhasználó nem található, 400 Bad Request státuszkód és ErrorCodes.account\_error\_no\_user\_found hibaüzenet.
* **Példa:** POST api/Account/login (JSON formátumban a LoginDto objektum adataival).

**3. POST api/Account/logout**

* **Leírása:** Felhasználói kijelentkezés.
* **Válasz:** true értéket ad vissza sikeres kijelentkezés esetén. 200 OK státuszkód.
* **Hibakezelés:** Belső szerverhiba esetén 400 Bad Request státuszkód és ErrorCodes.common\_error\_internal\_server\_error hibaüzenet.
* **Példa:** POST api/Account/logout

#### CustomerController

A CustomerController osztály a web API végpontokat kezeli az ügyfelek (Customer) kezeléséhez. Az ICustomerManager interfészen keresztül kommunikál az üzleti logikával. Minden végpont ellenőrzi a felhasználó hitelesítését (Authorize attribútum), és csak "Admin" vagy "User" szerepkörrel rendelkező felhasználók férhetnek hozzá.

**Végpontok:**

**1. GET api/Customer**

* **Leírása:** Lekéri a bejelentkezett felhasználóhoz tartozó ügyfelek listáját.
* **Kérés paraméterei:** QueryParams objektum (opcionális): page, pageSize, sortBy, sortOrder (oldalszám, oldalméret, rendezési kritériumok).
* **Válasz:** PageResult<IEnumerable<CustomerDto>> objektum: Az ügyfelek listája oldalszámozva, a teljes elemek számával. 200 OK státuszkód.
* **Hibakezelés:** Ha nincs bejelentkezett felhasználó, 401 Unauthorized státuszkód és ErrorCodes.account\_error\_no\_logged\_user hibaüzenet.
* **Példa:** GET api/Customer?page=1&pageSize=10&sortBy=Name&sortOrder=asc

**2. GET api/Customer/{id}**

* **Leírása:** Lekéri egy adott ügyfél adatait az azonosítója alapján.
* **Útvonal paraméter:** id: Az ügyfél egyedi azonosítója.
* **Válasz:** DataResult<CustomerDto> objektum: Az ügyfél adatai, vagy hibaüzenet. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha az id érvénytelen vagy az ügyfél nem található, akkor a DataResult objektum a ResultStatus mezőben Fail értéket és a hibaüzenetet tartalmazza.
* **Példa:** GET api/Customer/123

**3. POST api/Customer/create**

* **Leírása:** Új ügyfél létrehozása.
* **Kérés törzs:** CustomerEditRegisterDto objektum: Az új ügyfél adatait tartalmazza.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a kérés törzse érvénytelen, 400 Bad Request státuszkód és a hibák listája. Ha nincs bejelentkezett felhasználó, 401 Unauthorized státuszkód és ErrorCodes.account\_error\_no\_logged\_user hibaüzenet.
* **Példa:** POST api/Customer/create (JSON formátumban a CustomerEditRegisterDto objektum adataival).

**4. PUT api/Customer/{id}**

* **Leírása:** Meglévő ügyfél módosítása.
* **Útvonal paraméter:** id: A módosítandó ügyfél azonosítója.
* **Kérés törzs:** CustomerEditRegisterDto objektum: A módosítandó ügyfél adatait tartalmazza.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a kérés törzse vagy az id érvénytelen, 400 Bad Request státuszkód és a hibák listája.
* **Példa:** PUT api/Customer/123 (JSON formátumban a CustomerEditRegisterDto objektum adataival).

**5. DELETE api/Customer/{id}**

* **Leírása:** Meglévő ügyfél törlése.
* **Útvonal paraméter:** id: A törlendő ügyfél azonosítója.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha az id érvénytelen, 400 Bad Request státuszkód és a hibák listája.
* **Példa:** DELETE api/Customer/123

#### UserController

A UserController osztály a web API végpontokat kezeli a felhasználók (AppUser) kezeléséhez. Ez a controller csak adminisztrátori jogosultsággal rendelkező felhasználók számára elérhető (Authorize(Roles = "Admin")). Az IAppUserManager interfészen keresztül kommunikál az üzleti logikával.

**Végpontok:**

**1. GET api/User**

* **Leírása:** Lekéri a felhasználók listáját.
* **Kérés paraméterei:** QueryParams objektum (opcionális): page, pageSize, sortBy, sortOrder (oldalszám, oldalméret, rendezési kritériumok).
* **Válasz:** PageResult<IEnumerable<AppUserDto>> objektum: A felhasználók listája oldalszámozva, a teljes elemek számával. 200 OK státuszkód.
* **Példa:** GET api/User?page=1&pageSize=10&sortBy=Name&sortOrder=asc

**2. GET api/User/id:{id}**

* **Leírása:** Lekéri egy adott felhasználó adatait az azonosítója alapján.
* **Útvonal paraméter:** id: A felhasználó egyedi azonosítója.
* **Válasz:** DataResult<AppUserDto> objektum: A felhasználó adatai, vagy hibaüzenet. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a felhasználó nem található, a DataResult objektum a ResultStatus mezőben Fail értéket és a hibaüzenetet tartalmazza.
* **Példa:** GET api/User/id:123

**3. GET api/User/username:{userName}**

* **Leírása:** Lekéri egy adott felhasználó adatait a felhasználóneve alapján.
* **Útvonal paraméter:** userName: A felhasználó neve.
* **Válasz:** DataResult<AppUserDto> objektum: A felhasználó adatai, vagy hibaüzenet. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a felhasználó nem található, a DataResult objektum a ResultStatus mezőben Fail értéket és a hibaüzenetet tartalmazza.
* **Példa:** GET api/User/username:john.doe

**4. POST api/User/register**

* **Leírása:** Új felhasználó regisztrációja.
* **Kérés törzs:** AppUserRegisterDto objektum: Az új felhasználó adatait tartalmazza.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a kérés törzse érvénytelen, 400 Bad Request státuszkód és a hibák listája.
* **Példa:** POST api/User/register (JSON formátumban az AppUserRegisterDto objektum adataival).

**5. PUT api/User/{id}**

* **Leírása:** Meglévő felhasználó módosítása.
* **Útvonal paraméter:** id: A módosítandó felhasználó azonosítója.
* **Kérés törzs:** AppUserEditDto objektum: A módosítandó felhasználó adatait tartalmazza.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:** Ha a kérés törzse vagy az id érvénytelen, 400 Bad Request státuszkód és a hibák listája.
* **Példa:** PUT api/User/123 (JSON formátumban az AppUserEditDto objektum adataival).

**6. DELETE api/User/{id}**

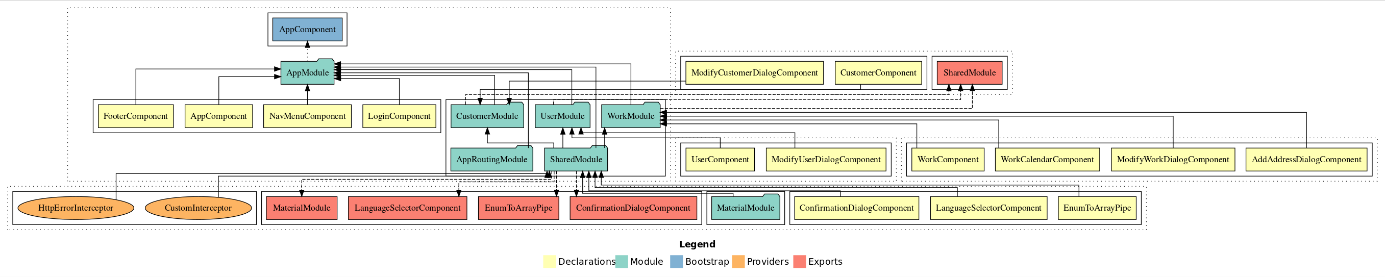
* **Leírása:** Meglévő felhasználó törlése.
* **Útvonal paraméter:** id: A törlendő felhasználó azonosítója.
* **Válasz:** Result objektum: A művelet sikerességét jelzi (ResultStatusEnum.Success vagy ResultStatusEnum.Fail), hibák esetén a hibák listáját tartalmazza. 200 OK státuszkód siker esetén.
* **Hibakezelés:**
  + Ha az id érvénytelen, 400 Bad Request státuszkód és a hibák listája.
  + Ha a bejelentkezett felhasználó törlését kérik, 400 Bad Request státuszkód és ErrorCodes.user\_error\_cant\_delete\_loggedin\_user hibaüzenet.
* **Példa:** DELETE api/User/123

**Összefoglalás**

Az Epic.CRM architektúrája jól strukturált és karbantartható. A rétegek elkülönítése biztosítja a kód modularitását és a könnyű tesztelhetőséget. Az adatbázis-hozzáférés az Entity Framework Core segítségével történik, a Repository mintázatot alkalmazva. Az alkalmazás konfigurációja az appsettings.json fájlban van meghatározva.

## Frontend Architektúra

Az Epic.CRM frontend architektúrája Angular-ban készült, moduláris felépítéssel. A következő ábra szemlélteti a komponensek és modulok közötti kapcsolatokat. A lenti leírás ezt az ábrát követi.



**7.1 AppModule**

Az AppModule a fő Angular modul, amely a többi modult összekapcsolja és inicializálja az alkalmazást. Az AppModule a következő komponenseket és modulokat tartalmazza:

* **AppComponent:** A fő alkalmazás komponens, amely a többi komponenst tartalmazza.
* **AppRoutingModul:** Az alkalmazás útvonalkezelését végzi.
* **FooterComponent:** A weboldal lábléce.
* **NavMenuComponent:** A navigációs menü.
* **LoginComponenet:** A bejelentkezési felület.

**7.2 CustomerModule, UserModule, WorkModule**

Ezek a modulok a rendszer fő funkcionális területeit reprezentálják:

* **CustomerModule:** Ügyfélkezelés. Tartalmazza a CustomerComponent-et (ügyfelek listázása) és a ModifyCustomerDialogComponent-et (ügyfél módosítása).
* **UserModule:** Felhasználókezelés (adminisztrátoroknak). Tartalmazza a UserComponent-et (felhasználók listázása) és a ModifyUserDialogComponent-et (felhasználó módosítása).
* **WorkModule:** Munkák kezelése. Tartalmazza a WorkComponent-et (munkákat listázása), a WorkCalendarComponent-et (munkákat naptár nézetben), a ModifyWorkDialogComponent-et (munka módosítása) és az AddAddressDialogComponent-et (új cím hozzáadása).

**7.3 SharedModule**

Ez a modul a különböző modulok által megosztott komponenseket, szolgáltatásokat és pipe-okat tartalmazza. A megosztott elemek újra felhasználhatók, javítva a kód karbantarthatóságát. Tartalmazza többek között:

* **MaterialModule:** Az Angular Material komponenseket tartalmazza.
* **ConfirmationDialogComponent:** Megerősítő dialógus ablak.
* **LanguageSelectorComponent:** Nyelvválasztó komponens.
* **EnumToArrayPipe:** Enum típusokat tömbbé alakító pipe.

**7.4 Interceptores**

* **HttpErrorInterceptor:** A HTTP hibák kezelését végzi.
* **CustomInterceptor:** Egyedi interceptor a HTTP kérések módosításához.

**7.5 Összefoglalás**

Az Angular moduláris felépítése jól strukturált és karbantartható rendszert eredményez. A SharedModule hatékonyan kezeli a megosztott komponenseket, így csökkentve a kód redundanciáját. A komponensek és modulok közötti kapcsolatok világosan láthatóak a diagramon. A rendszer kiterjeszthető új modulok és komponensek hozzáadásával.

## Továbbfejlesztési lehetőségek

Az Epic.CRM jelenlegi funkcionalitása egy stabil alapot biztosít a kis- és mikrovállalkozások számára. Azonban számos további fejlesztési lehetőség kínálkozik a rendszer funkcionalitásának bővítésére és a felhasználói élmény javítására. Az alábbiakban négy konkrét továbbfejlesztési ötletet vázolunk fel:

**6.1 Jelentéskészítés és Analitika**

A jelenlegi rendszer lehetővé teszi az adatok tárolását és kezelését, de hiányzik a komplexebb jelentéskészítési és adatelemzési funkcionalitás. A továbbfejlesztés során beépíthető lenne egy modul, amely lehetővé teszi a felhasználók számára, hogy testreszabható jelentéseket generáljanak a munkákról, az ügyfelekről és a bevételekről. Ez magában foglalhatja a különböző diagramokat, grafikonokat és exportálási lehetőségeket (pl. PDF, Excel). Az adatelemzés segítségével a felhasználók hatékonyabb döntéseket hozhatnak.

**6.3 Integráció más rendszerekkel**

A rendszer integrálható lenne más vállalati rendszerekkel (pl. számlázó programok, CRM rendszerek). Ez automatizálná bizonyos feladatokat, és javítaná az adatátvitelt más rendszerekkel. Például, az integráció egy számlázó rendszerrel automatikusan generálhatna számlákat a befejezett munkák alapján.

**6.4 Időkövetés és Projektmenedzsment**

A rendszer kiegészíthető lenne egy időkövető modullal, amely lehetővé teszi a felhasználók számára, hogy nyomon kövessék az egyes munkákhoz szánt időt. Ez segíthet a hatékonyság javításában és a költségek optimalizálásában. A projektmenedzsment funkciók (pl. határidők, feladatok) bevezetése is hasznos lehet komplexebb projektek kezeléséhez.

**6.5 Felhasználói Jogosultságkezelés**

A jelenlegi rendszerben a jogosultságkezelés csak adminisztrátor és felhasználó szerepköröket különböztet meg. Továbbfejlesztésként több szerepkör bevezetése indokolt lehet (pl. munkatárs, vezető), amely lehetővé teszi a finomabb jogosultságkezelést. Ez biztosíthatja, hogy minden felhasználó csak a számára engedélyezett adatokhoz és funkciókhoz férhessen hozzá. Ez fokozott biztonságot és adatvédelmet biztosít.

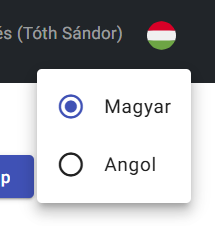
Ezek a továbbfejlesztési lehetőségek jelentősen javíthatják az Epic.CRM rendszer funkcionalitását és felhasználói élményét, szélesebb körű felhasználói bázist biztosítva. A fejlesztési prioritások meghatározása a felhasználói visszajelzések és a piaci igények alapján történik.

# Felhasználói dokumentáció

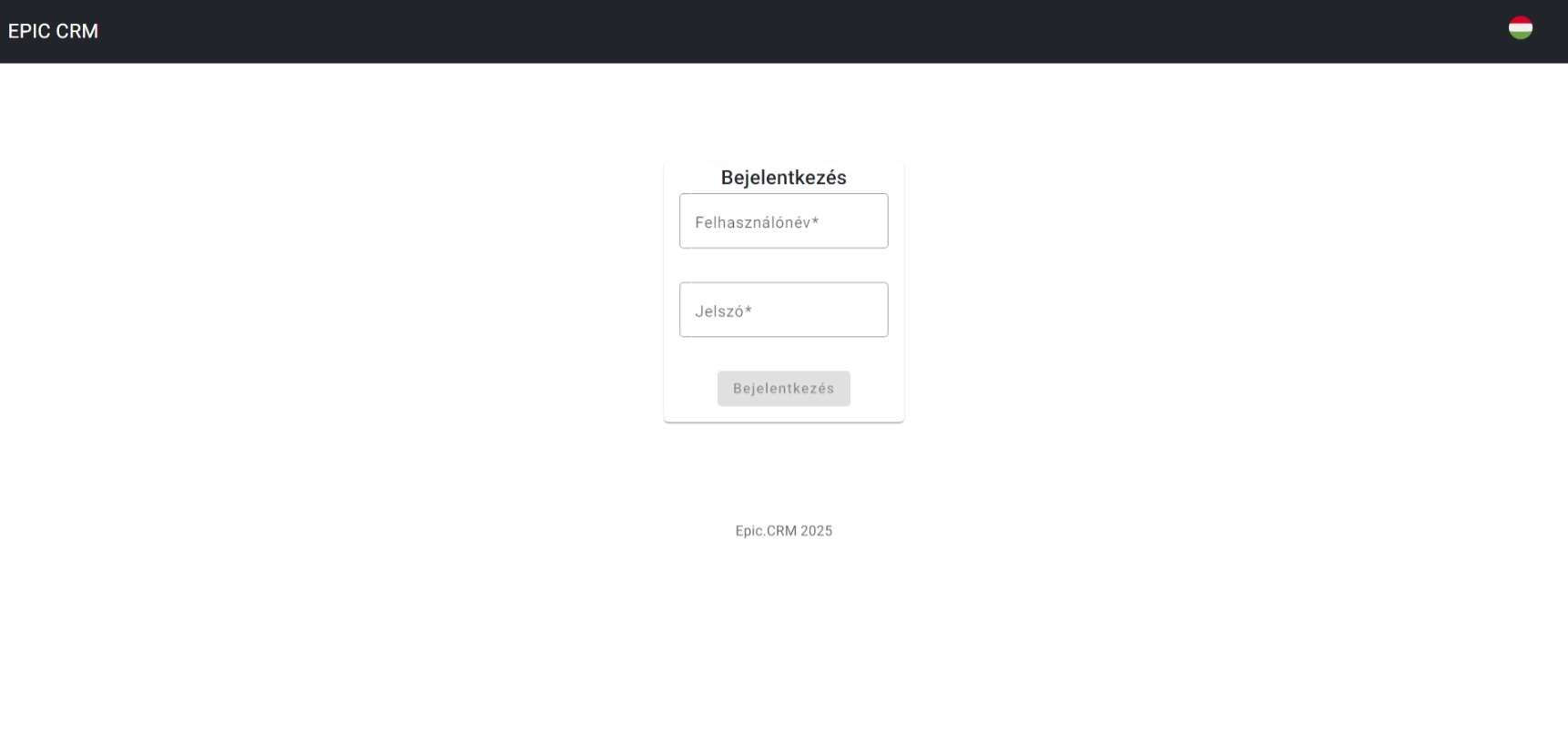
Az alkalmazás egy webes alkalmazás, amely az internetről érhető el, böngészőn keresztül.

## Nyelvváltás

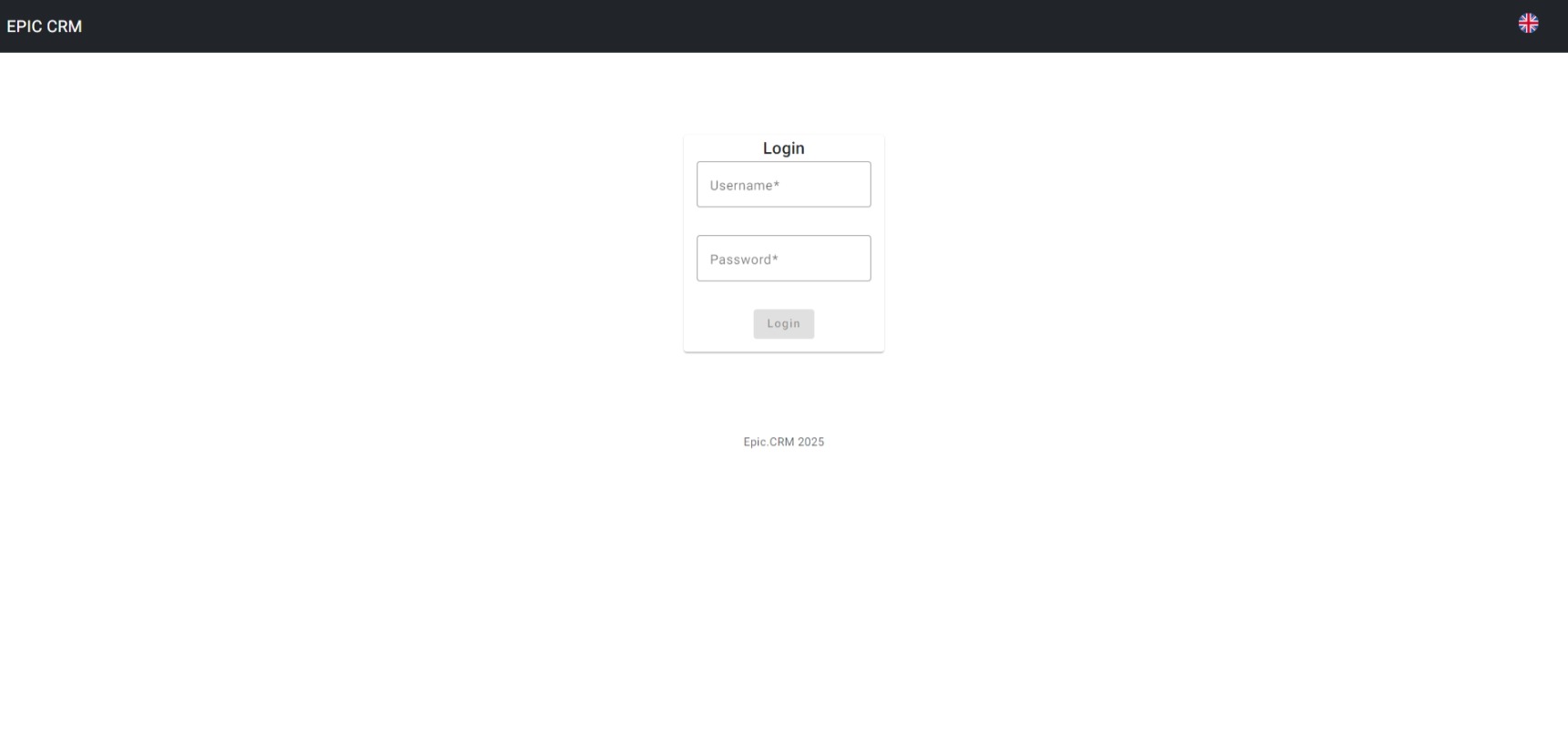
Az alkalmazás többnyelvű. Jelenleg angol és magyar nyelv támogatott. A jobb felső sarokban lévő zászló ikonra kattintva lehet nyelvet váltani:



1. ábra Nyelvváltás

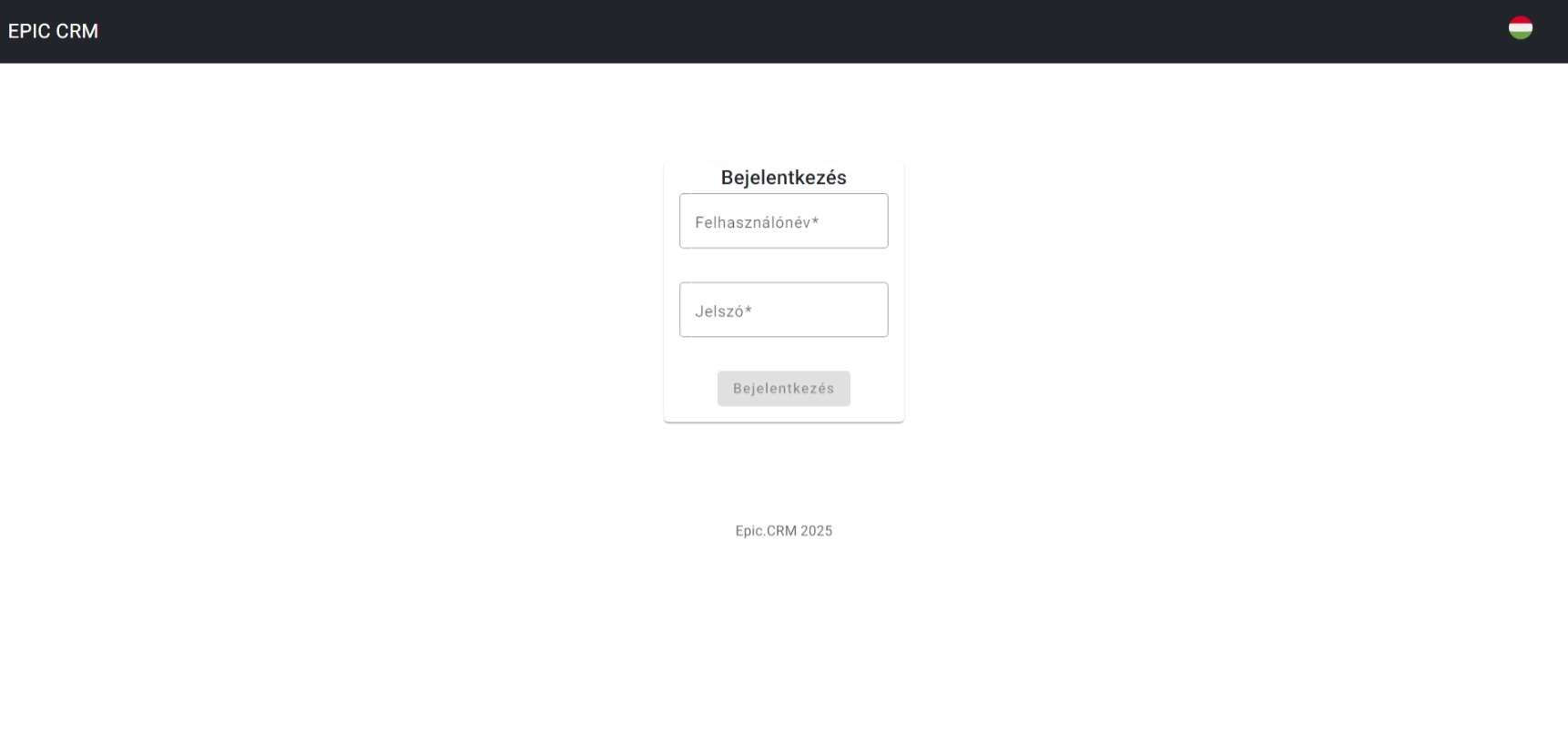


2. ábra bejelentkezési felület magyar nyelven



3. ábra Bejelentkezési felület angol nyelven

## Bejelentkezés



A rendszer eléréséhez a felhasználóknak be kell jelentkezniük a főoldalon található űrlap segítségével. Ez az űrlap két beviteli mezőt tartalmaz:

* **Email:** A regisztrált e-mail cím megadása.
* **Jelszó:** A megfelelő jelszó megadása.

Sikeres bejelentkezés után a felhasználó a rendszer főképernyőjére kerül, amely a szerepkörének megfelelő funkciókat jeleníti meg.

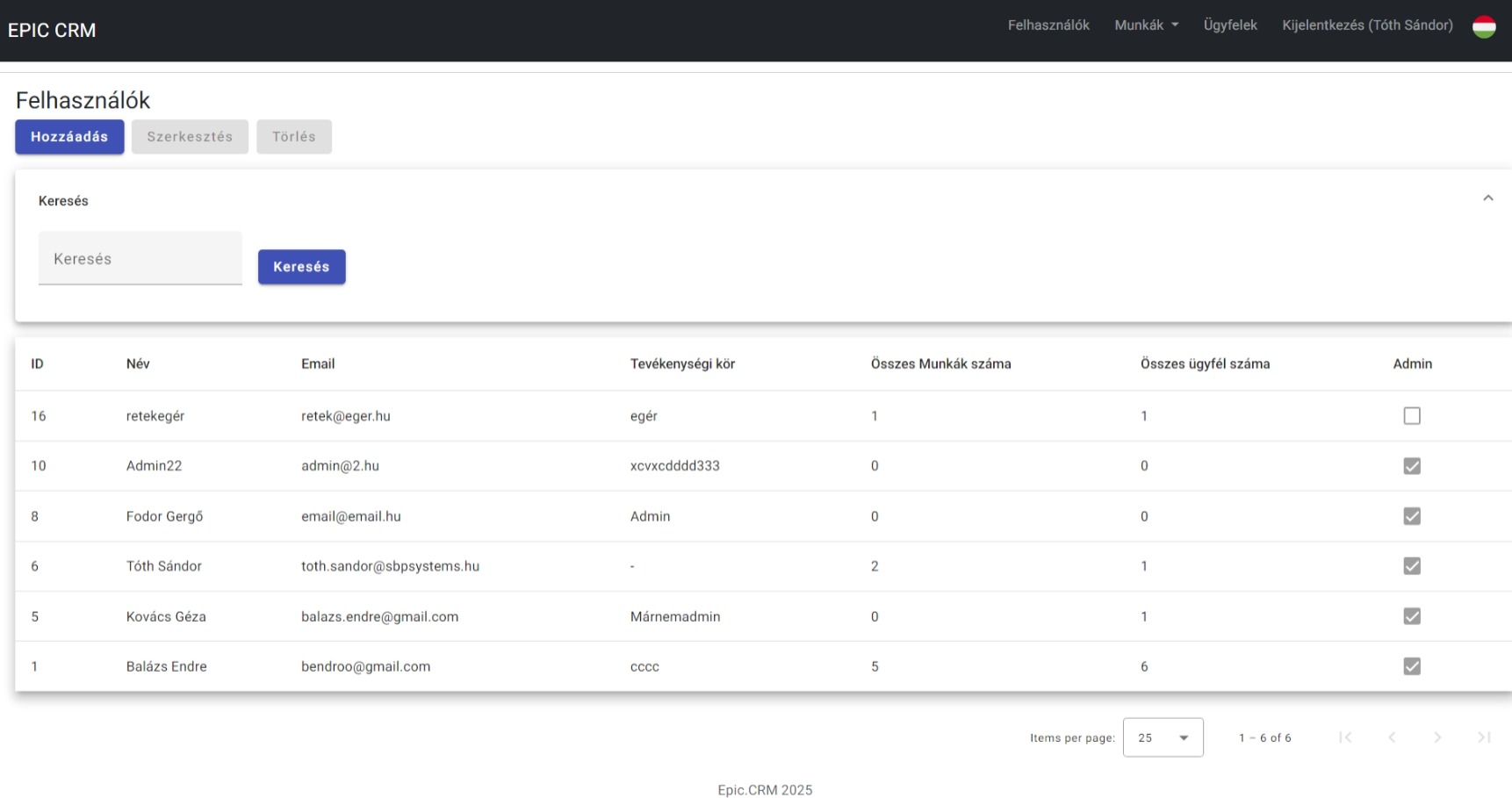


4. ábra Bejelenkezett fejlác Admin felhasználónál

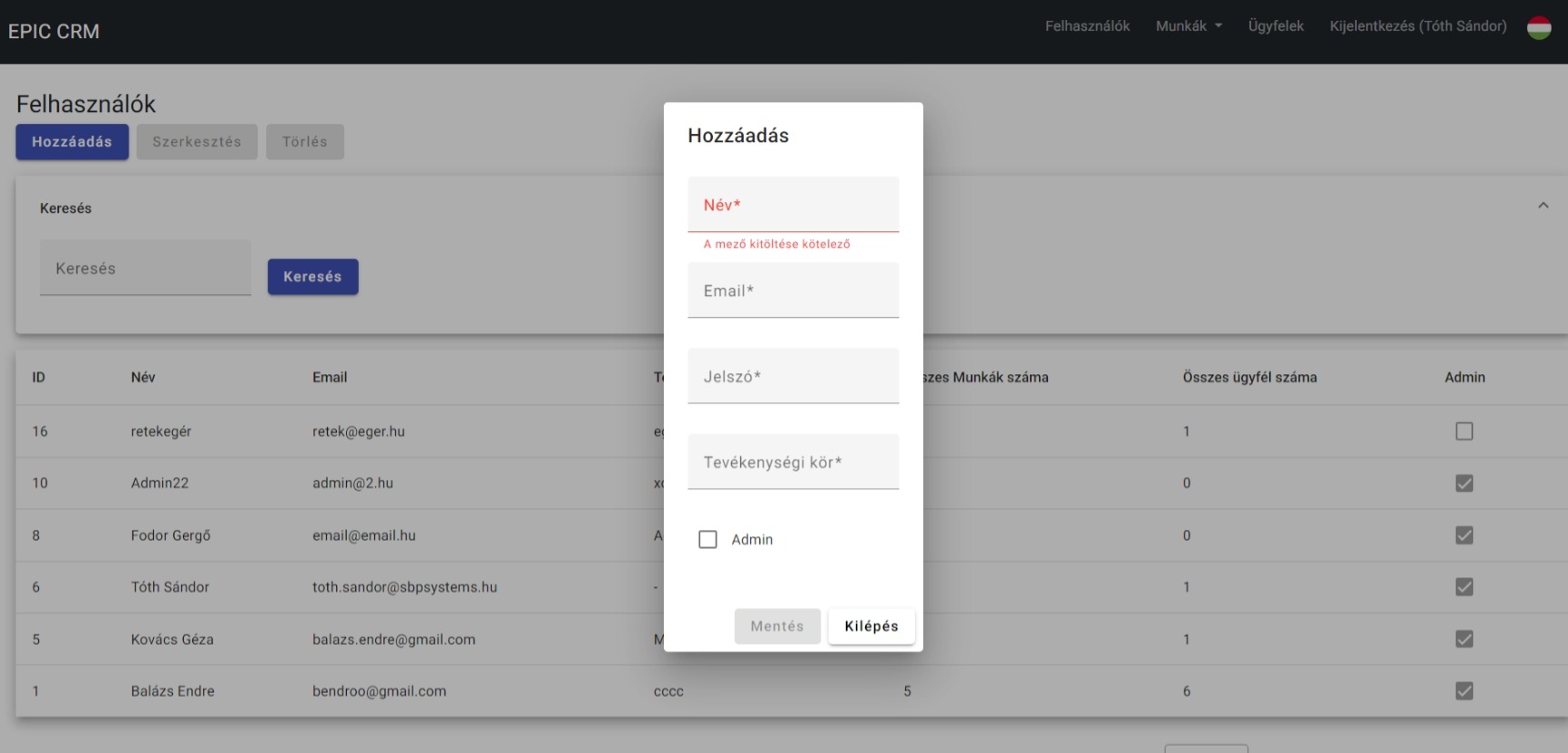


5. ábra Bejelentkezett fejlác Vállalkozóval

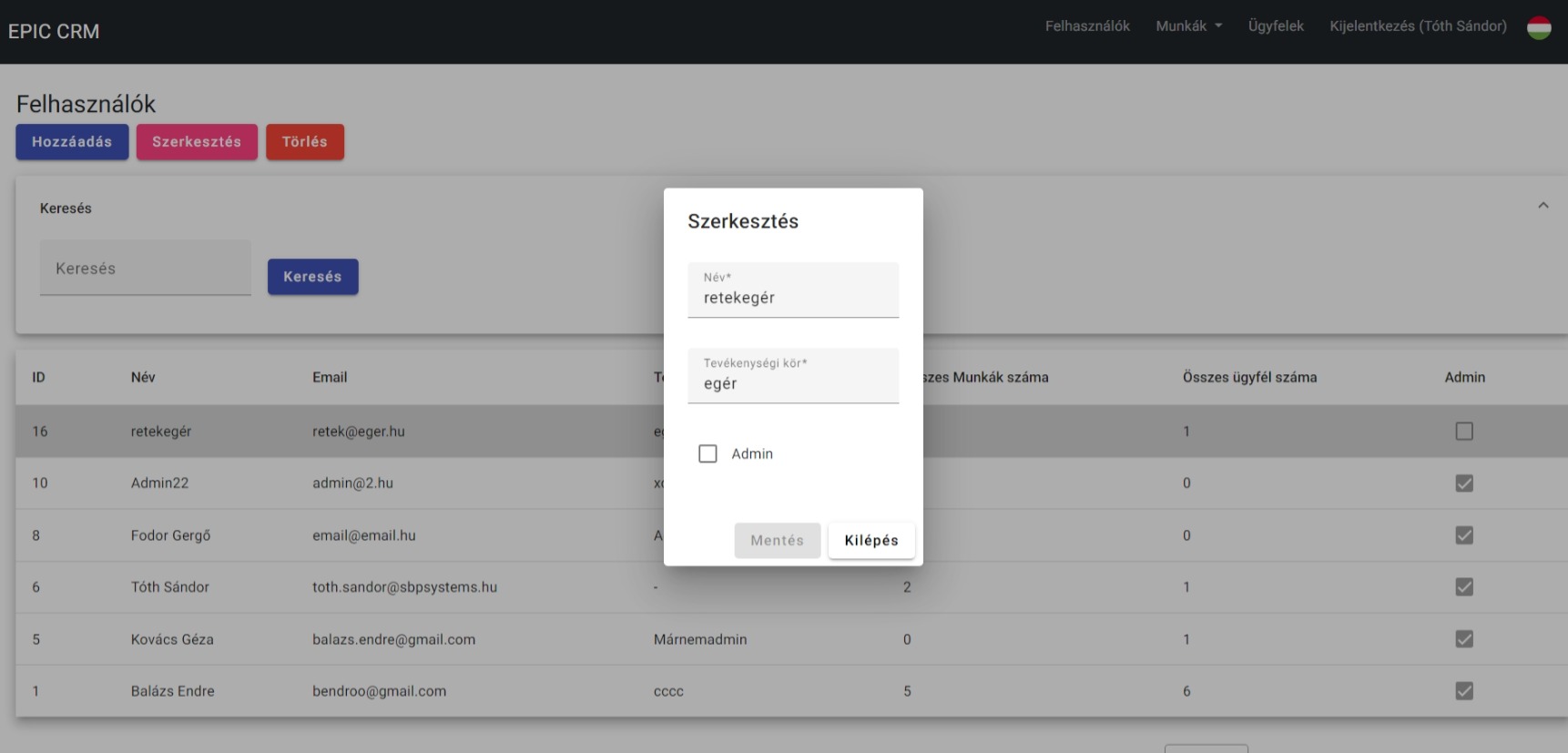
## Felhasználó kezelés (Adminisztrátor)



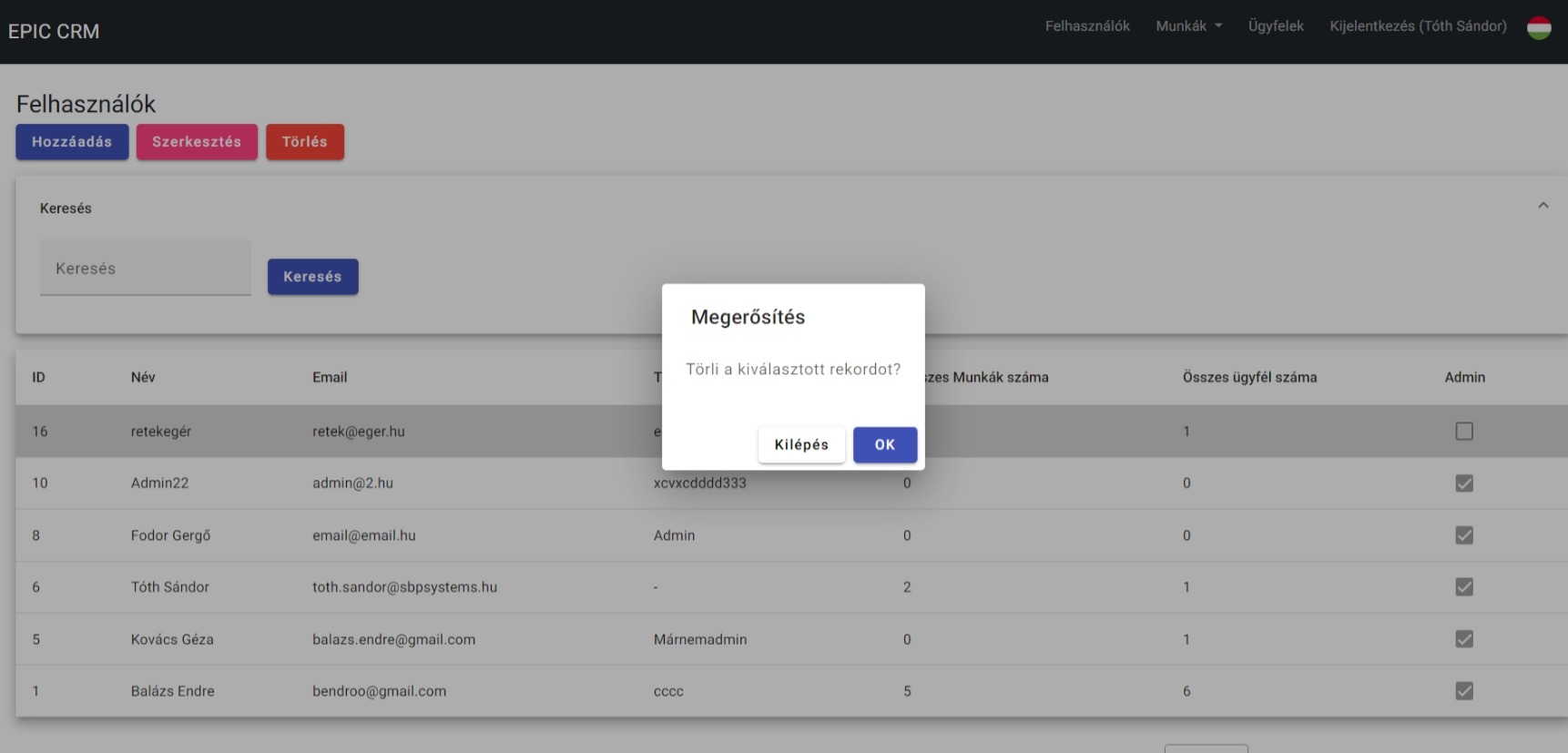
6. ábra Felhasználók listája



7. ábra Felhasználó hozzáadása



8. ábra Felhasználó szerkesztése



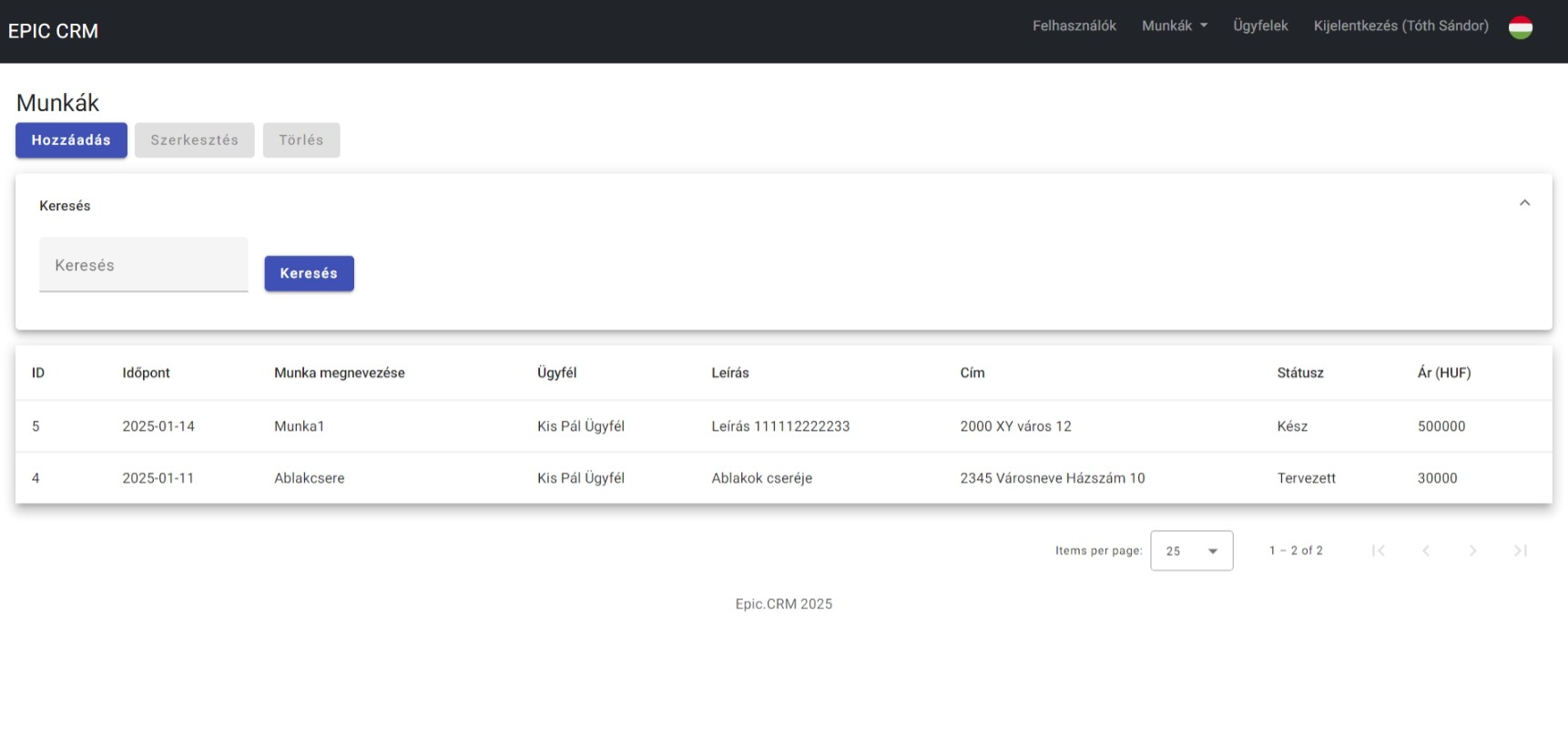
9. ábra Felhasználó törlése

Csak az adminisztrátori jogosultsággal rendelkező felhasználók férhetnek hozzá ehhez a funkcióhoz. Ez a felület táblázatosan jeleníti meg az összes regisztrált felhasználót, az alábbi oszlopokkal:

* **ID:** A felhasználó egyedi azonosítója.
* **Név:** A felhasználó neve.
* **Email:** A felhasználó e-mail címe.
* **Tevékenységi kör:** A felhasználó tevékenységi köre (pl. ács, kőműves).
* **Összes Munkák száma:** Az adott felhasználó által kezelt munkák száma.
* **Összes Ügyfél száma:** Az adott felhasználó által kezelt ügyfelek száma.

Az adminisztrátorok új felhasználókat adhatnak hozzá, valamint szerkeszthetik és törölhetik a meglévőket. Az új felhasználó hozzáadásához egy űrlap jelenik meg, amely a felhasználói adatok bevitelére szolgál.

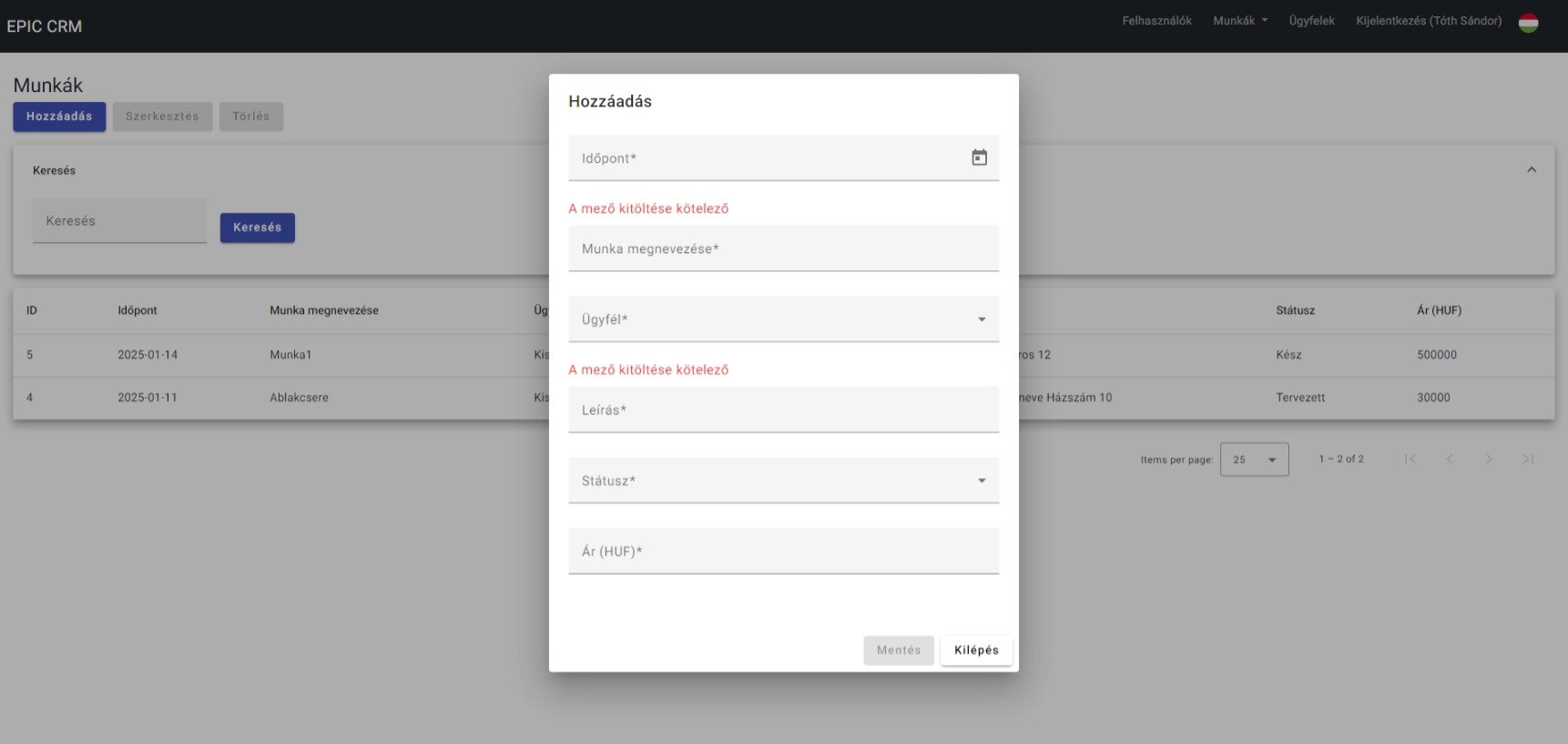
## Munkáim



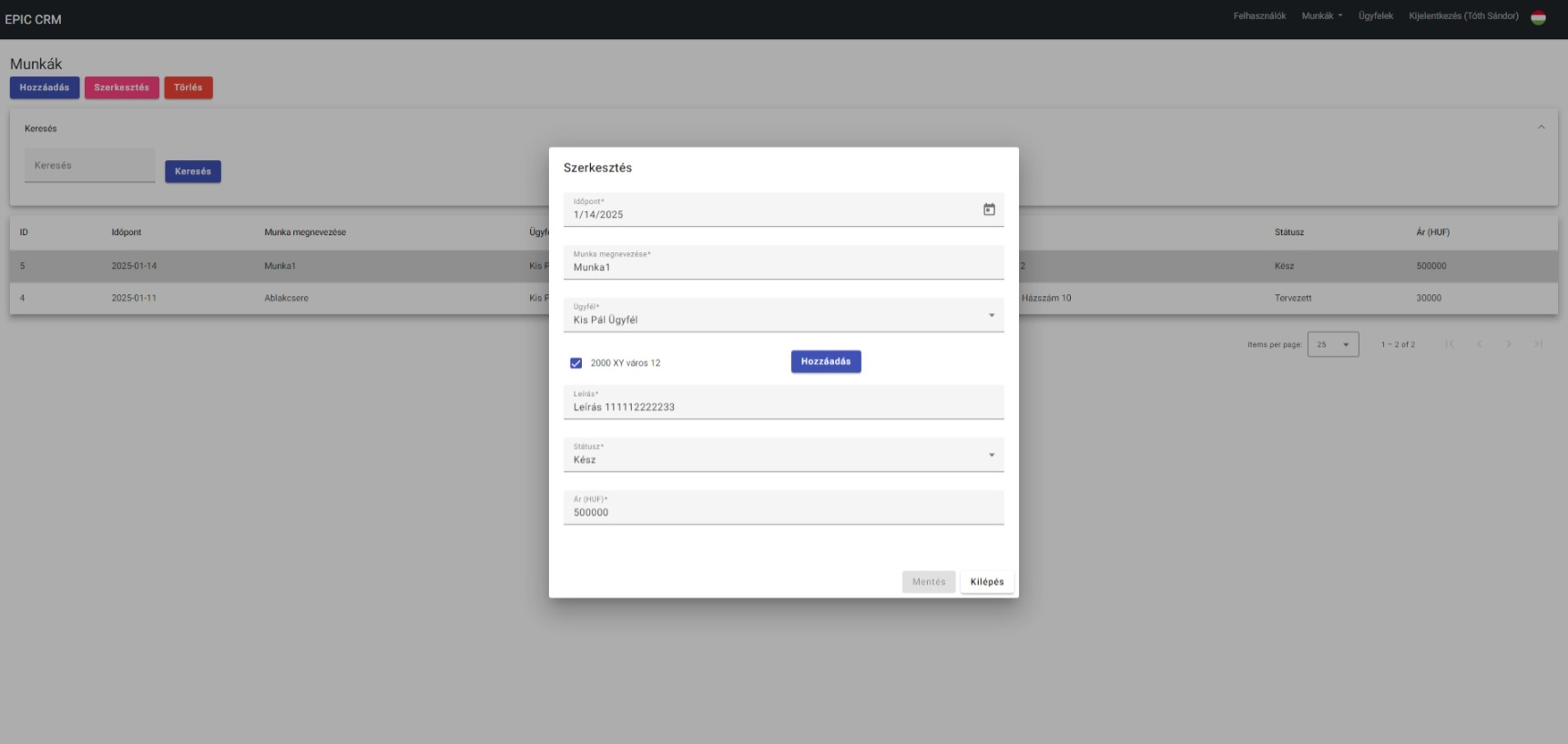
10. ábra Munkák listázása

Ez a felület az aktuális felhasználó által kezelt munkákat jeleníti meg táblázatos formában:

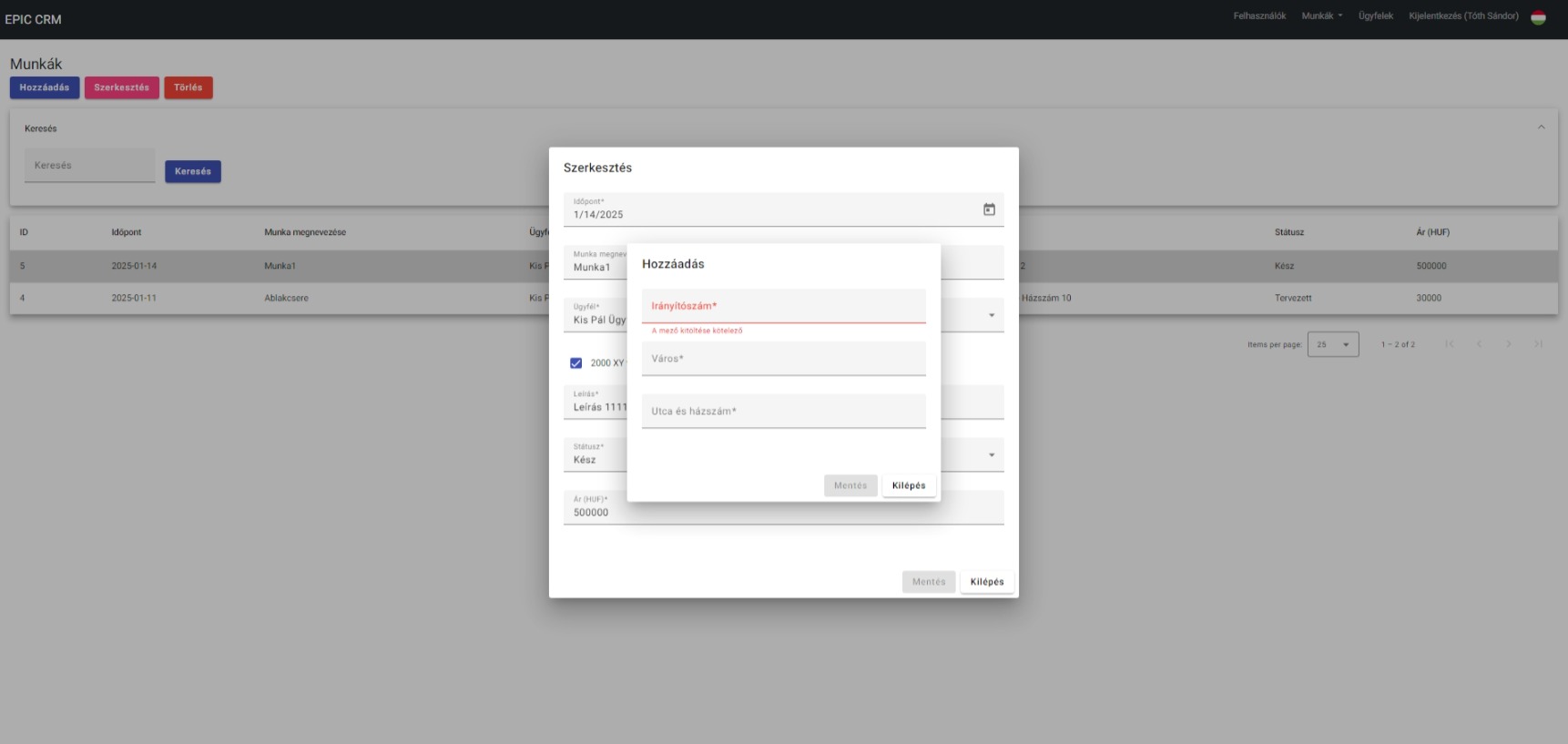
* **ID:** A munka egyedi azonosítója.
* **Időpont:** A munka tervezett vagy elvégzett időpontja.
* **Munka megnevezése:** A munka rövid leírása.
* **Ügyfél:** A munkát megrendelő ügyfél neve.
* **Leírás:** A munka részletesebb leírása.
* **Cím:** Az irányítószám, település, utca és házszám összefűzött megjelenítése.
* **Státusz:** A munka aktuális státusza (pl. tervezett, folyamatban, befejezve).
* **Ár:** A munka ára.



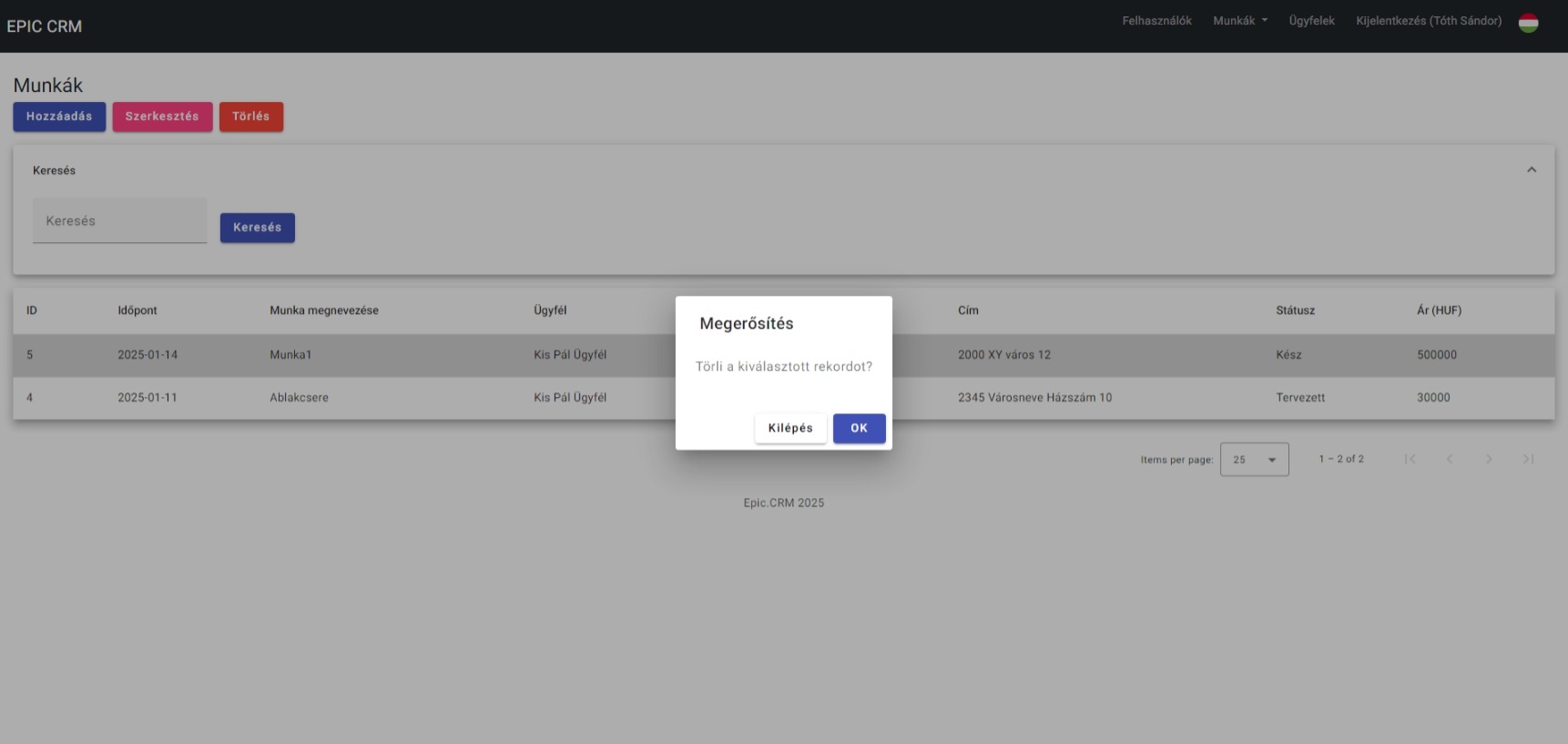
11. ábra Új munka hozzáadása



12. ábra Munka szerkesztése



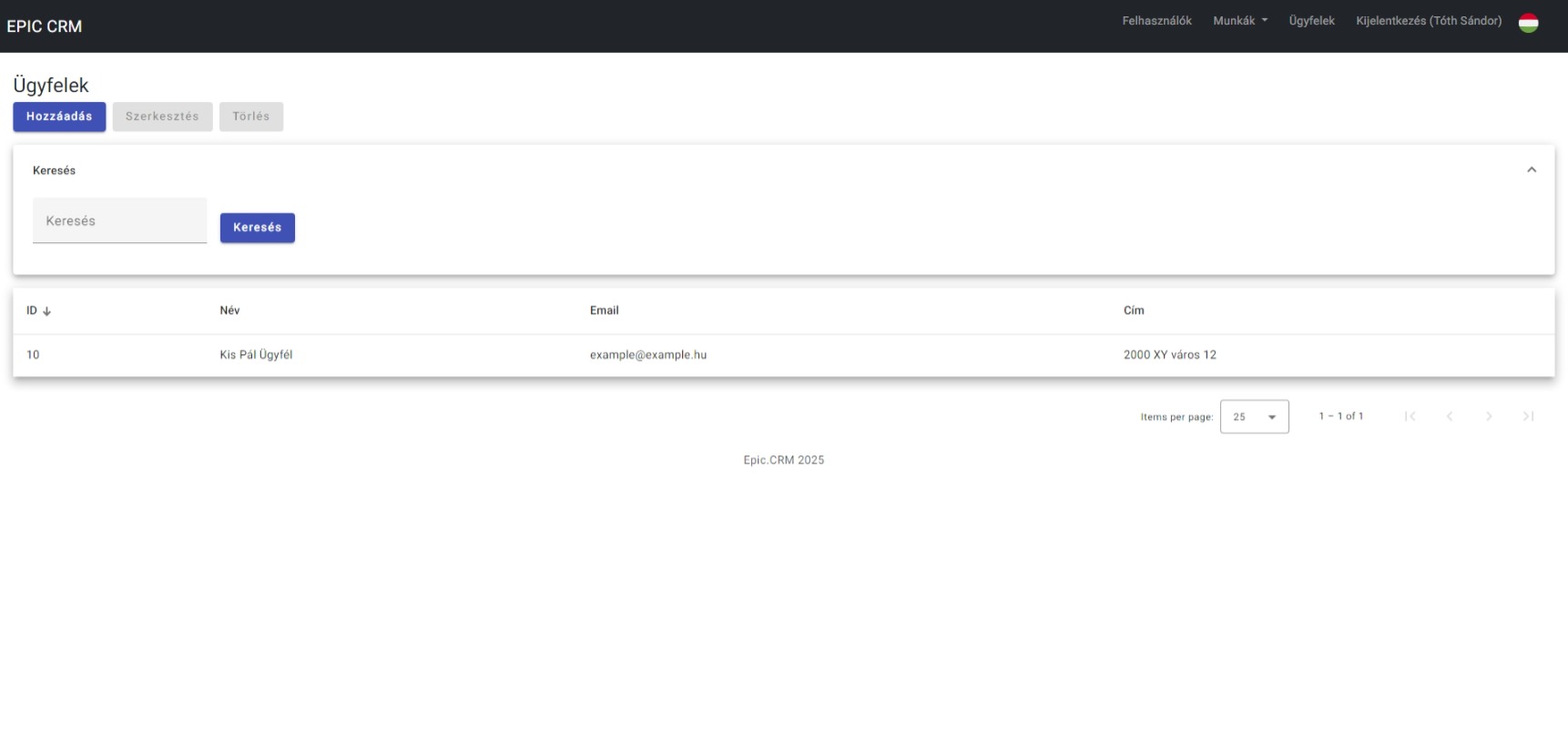
13. ábra Munka szerkesztése új cím megadásával



14. ábra Meglévő munka törlése

Új munka hozzáadásához egy űrlap jelenik meg, amely lehetővé teszi a munka adatainak bevitelét. Az űrlap tartalmaz egy ügyfél kombóboxot, egy dátumválasztót, egy új cím hozzáadására szolgáló funkciót, valamint egy státusz kombóboxot. A meglévő munkák szerkeszthetők és törölhetők.

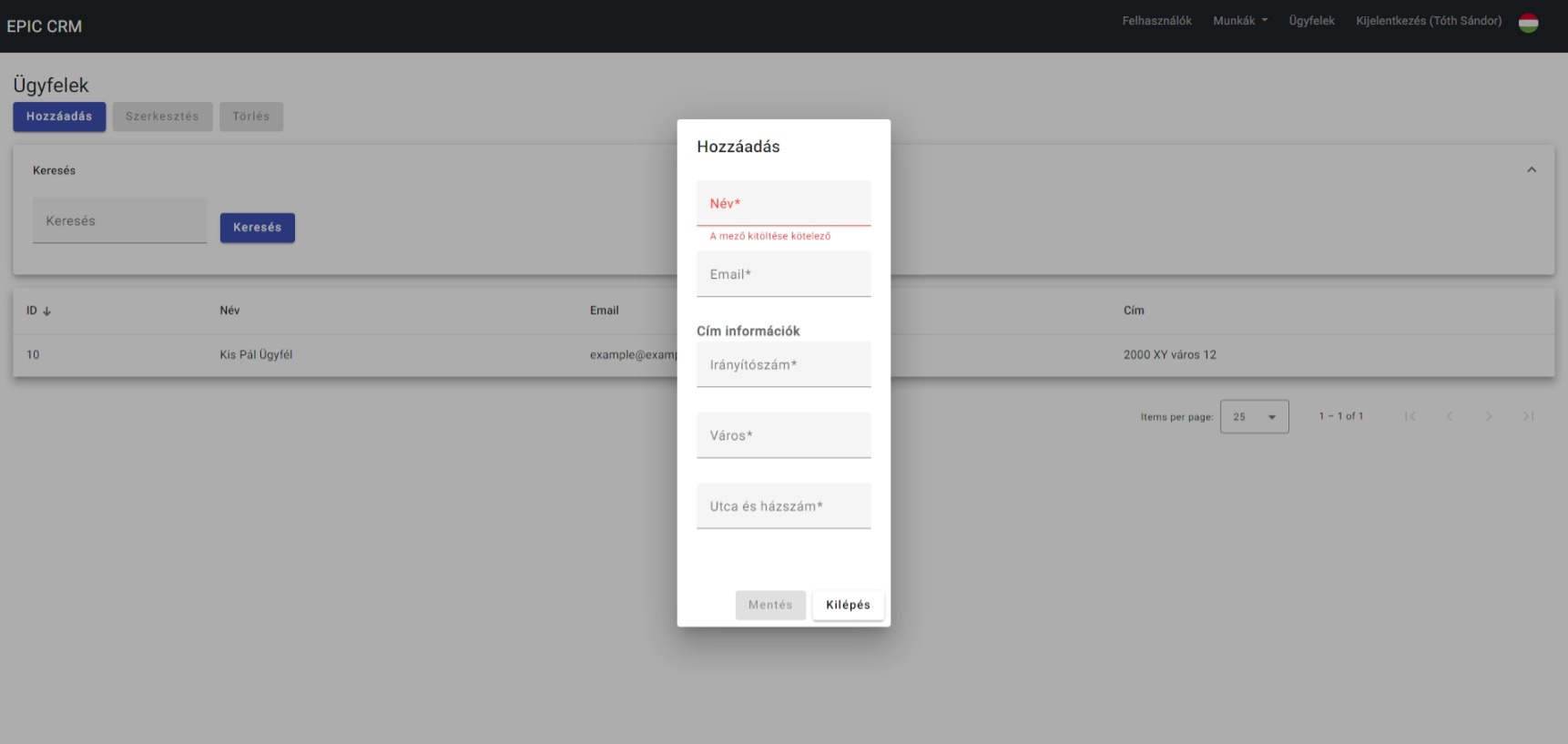
## Ügyfeleim



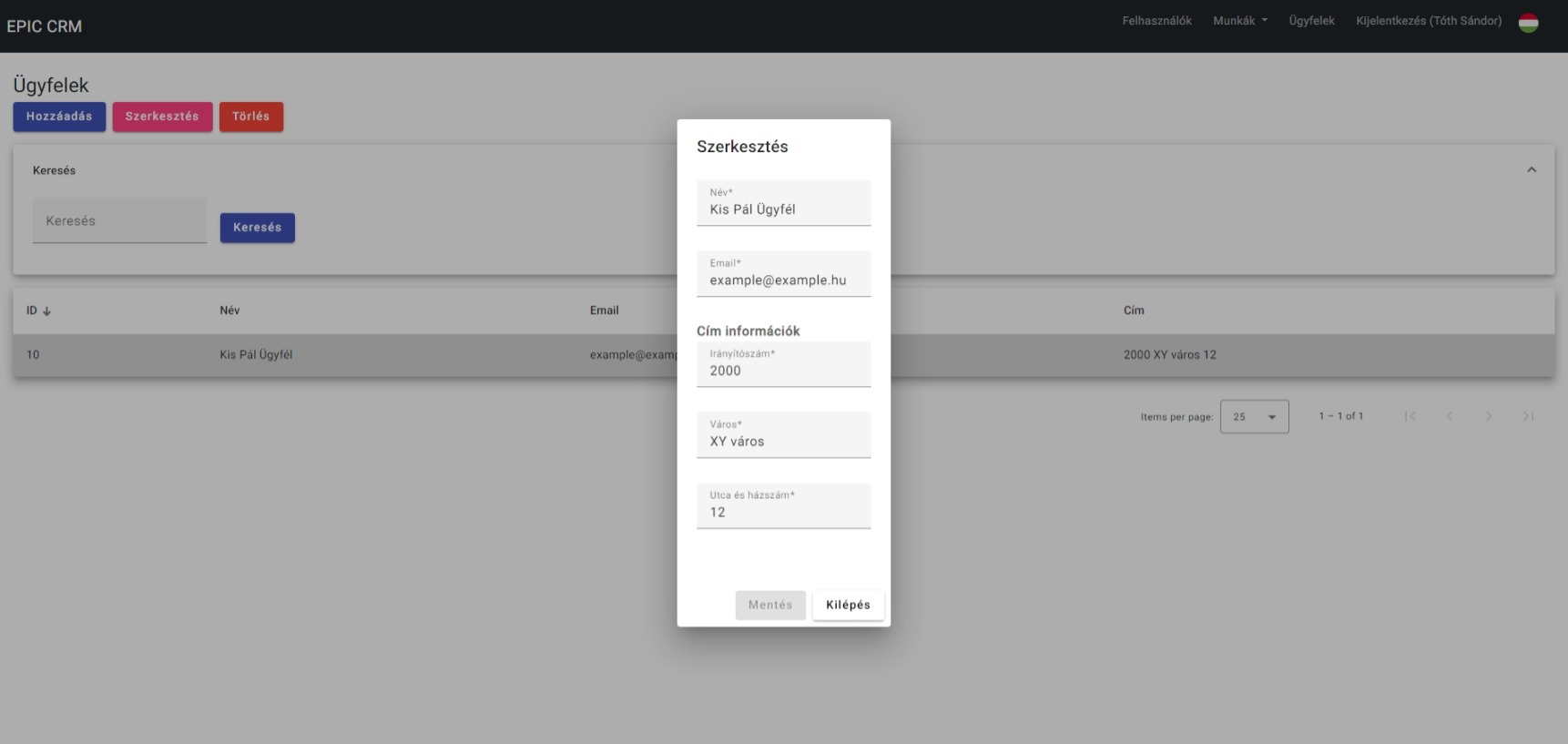
15. ábra Ügyfelek listája

Ez a felület az aktuális felhasználó által kezelt ügyfeleket jeleníti meg táblázatos formában:

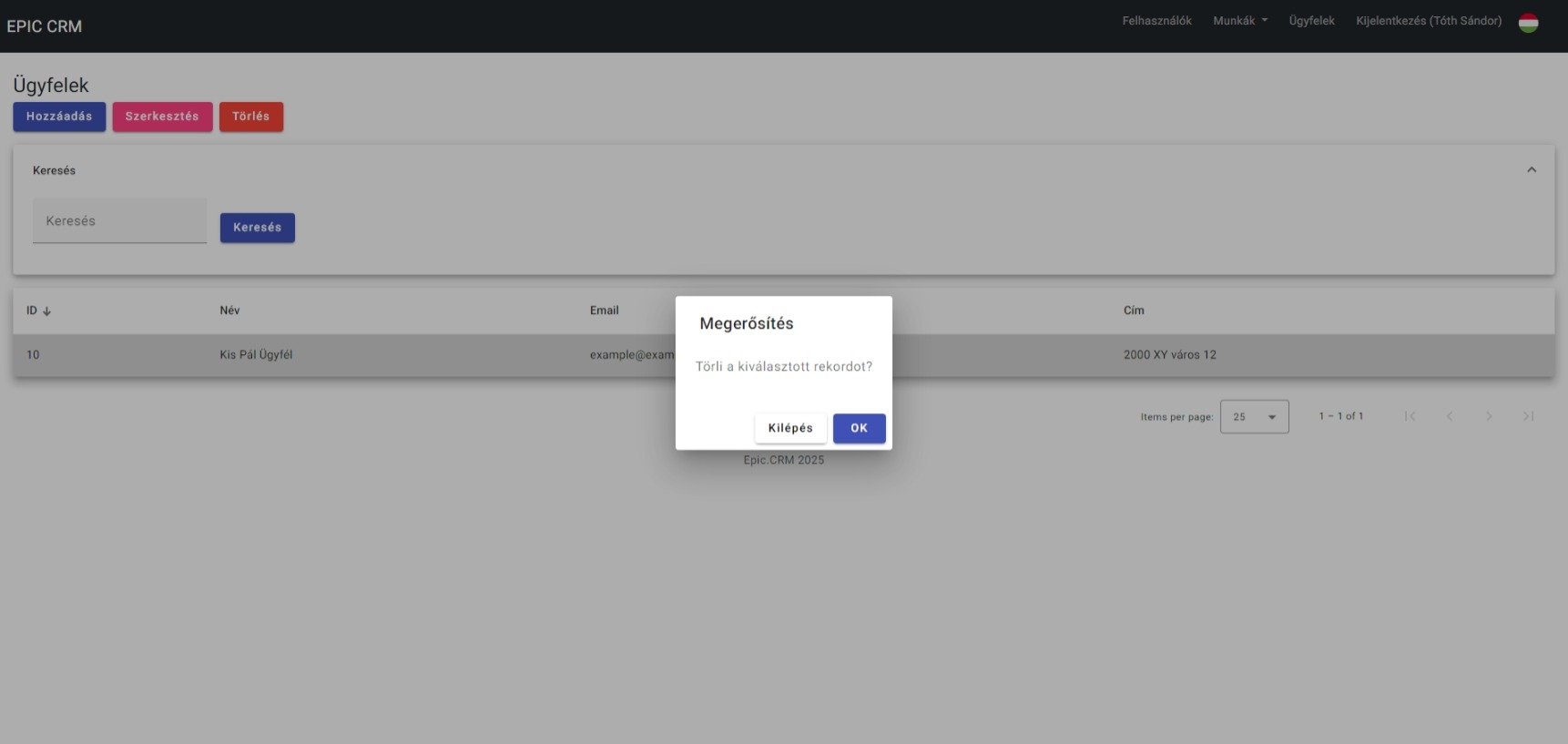
* **Név:** Az ügyfél neve.
* **Email:** Az ügyfél e-mail címe.
* **Cím:** Az irányítószám, település, utca és házszám összefűzött megjelenítése.



16. ábra Új ügyfél hozzáadása



17. ábra Ügyfél szerkesztése

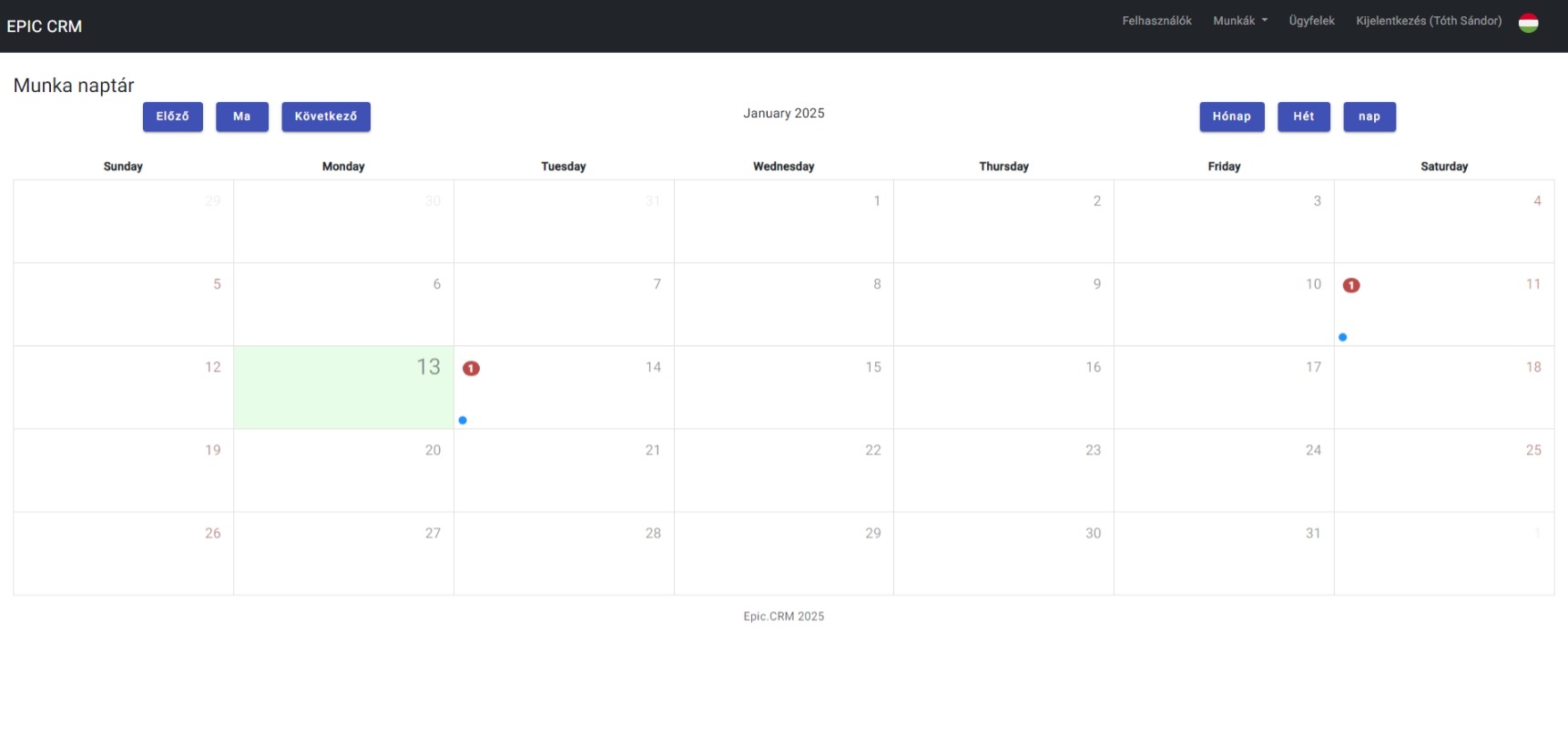


18. ábra Ügyfél törlése

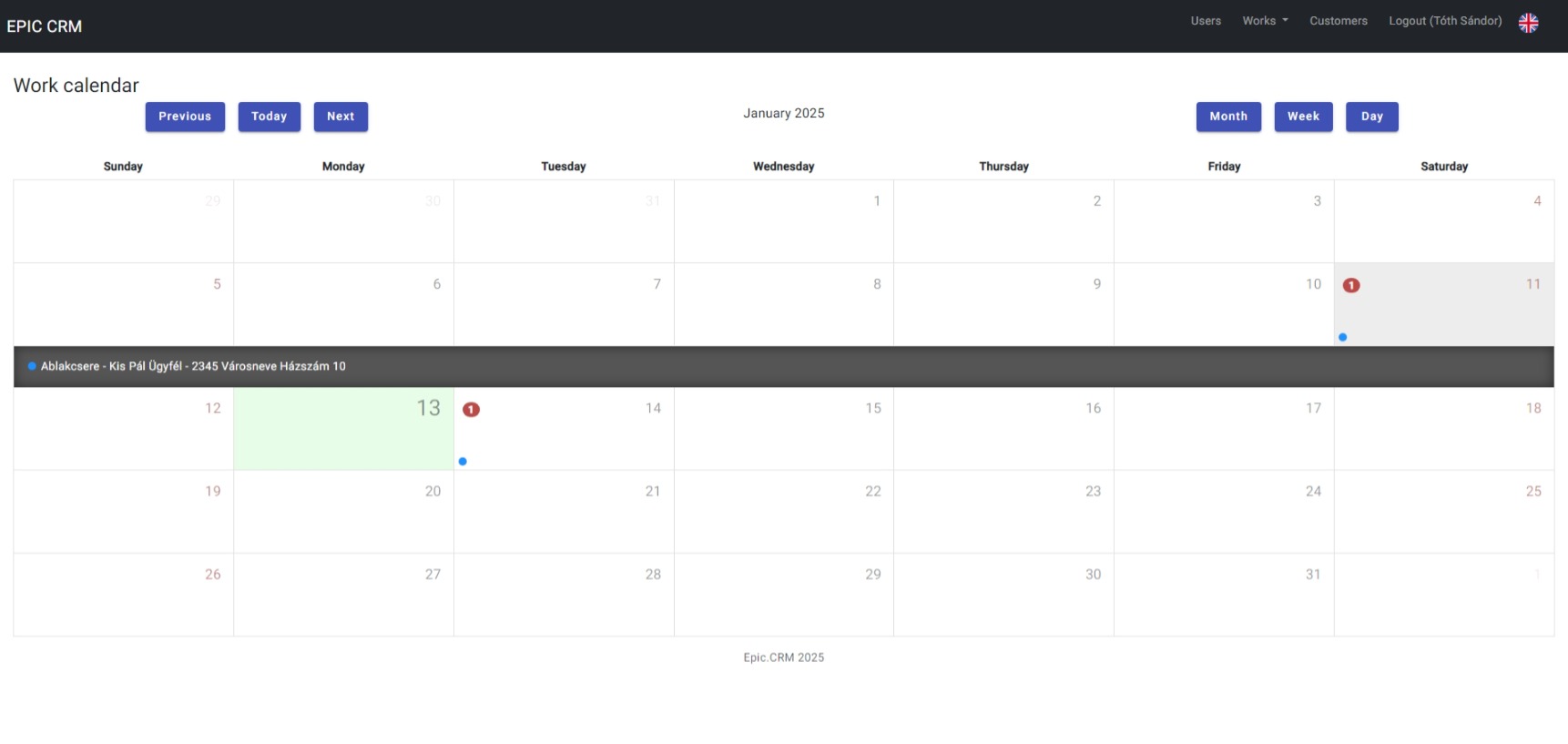
Új ügyfél hozzáadásához egy űrlap jelenik meg, amely lehetővé teszi az ügyfél adatainak bevitelét, beleértve egy új cím hozzáadásának lehetőségét is. A meglévő ügyfelek szerkeszthetők és törölhetők.

## Naptár

Az alkalmazásban kialakításra került egy naptár komponens, amely a „Naptár” menüpont alatt érhető el és az adott felhasználó munkáit tartalmazza naptár nézetben, ahol könnyedén követhető, hogy mikor milyen feladat vár a vállalkozóra.



19. ábra Naptár nézet magyar nyelven

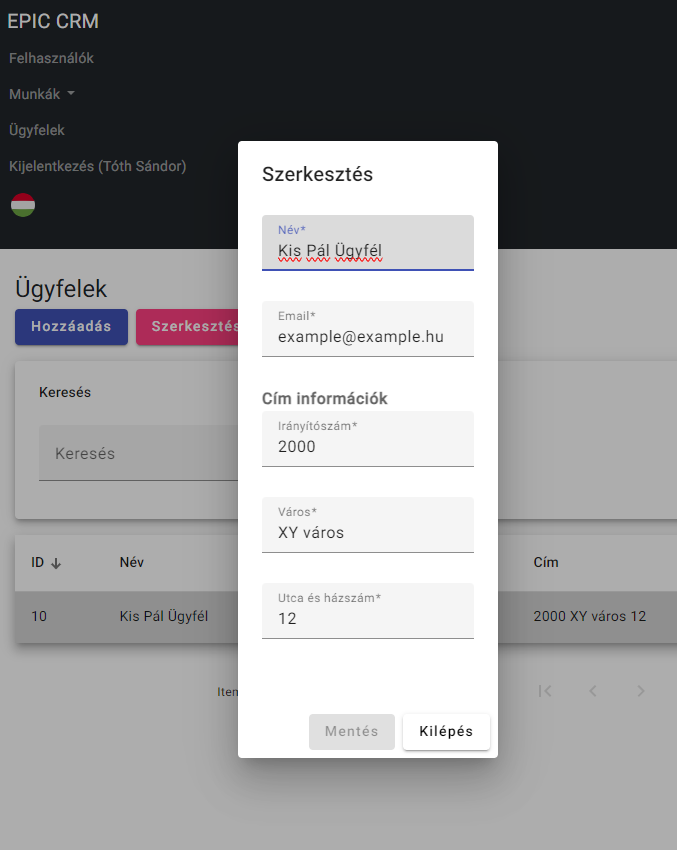


20. ábra Naptár nézet angol nyelven, munka megjelenítésével

A naptárban kék pöttyel vannak jelölve azok a dátumok, amire rögzítve van munka a rendszerben. A napra kattintva lenyílik egy sáv, ahol az aznapra felvett munkák listája látható az ügyfél nevével, címével és a munka megnevezésével.

## Az alkalmazás használata Mobilon

Az alkalmazás oly módon készült, hogy reszponzív legyen és használata mobilon is elérhető az interneten keresztül. Mobilon az alábbi módon jelenik meg az alkalmazás:



21. ábra Alkalmazás megjelenése Mobilos felbontáson

# Tesztelés

A tesztelés célja az Epic.CRM rendszer funkcionalitásának, megbízhatóságának, teljesítményének és biztonságának ellenőrzése. A tesztelés több fázisból áll, különböző teszt típusokat alkalmazva.

**Teszt Típusok**

* **Funkcionális tesztek:** Ellenőrzik, hogy a rendszer megfelel-e a specifikációban meghatározott funkcionális követelményeknek. Ez magában foglalja az egyes funkciók (pl. bejelentkezés, felhasználó kezelés, munka kezelés, ügyfél kezelés) tesztelését különböző bemeneti adatokkal és felhasználói interakciókkal.
* **Integrációs tesztek:** Ellenőrzik, hogy a rendszer különböző komponensei (pl. frontend, backend, adatbázis) megfelelően működnek-e együtt.
* **Teljesítménytesztek:** Ellenőrzik a rendszer teljesítményét különböző terhelések mellett (pl. egyidejű felhasználók száma, adatmennyiség). A terheléses tesztekhez a k6 eszközt használjuk.
* **Biztonsági tesztek:** Ellenőrzik a rendszer biztonságát, beleértve a hitelesítést, a jogosultságkezelést és a védelmet a különböző támadásokkal szemben.

**Teszt Esetek**

A tesztelés során számos teszteset kerül végrehajtásra.

**Regisztráció és Bejelentkezés:**

* **TE1 (Sikeres regisztráció):** Új felhasználó regisztrációja érvényes adatokkal. Ellenőrzés: sikeres regisztráció, aktiváló email érkezése.
* **TE2 (Sikertelen regisztráció - hibás adatok):** Regisztráció hiányos adatokkal (kötelező mezők hiánya), vagy érvénytelen email címmel. Ellenőrzés: megfelelő hibaüzenetek jelennek meg.
* **TE3 (Sikertelen regisztráció - létező email):** Regisztráció már létező email címmel. Ellenőrzés: megfelelő hibaüzenet jelenik meg.
* **TE4 (Sikeres bejelentkezés):** Bejelentkezés helyes email címmel és jelszóval. Ellenőrzés: sikeres belépés a rendszerbe.
* **TE5 (Sikertelen bejelentkezés - hibás adatok):** Bejelentkezés hibás email címmel és/vagy jelszóval. Ellenőrzés: megfelelő hibaüzenet jelenik meg.
* **TE6 (Sikertelen bejelentkezés - zárolt fiók):** Több sikertelen bejelentkezési kísérlet után a fiók zárolása. Ellenőrzés: a fiók zárolva van, megfelelő üzenet jelenik meg.

**Felhasználó Kezelés (Admin):**

* **TE7 (Felhasználó létrehozása):** Új felhasználó létrehozása adminisztrátorként. Ellenőrzés: a felhasználó megjelenik a felhasználói listában.
* **TE8 (Felhasználó szerkesztése):** Meglévő felhasználó adatainak módosítása. Ellenőrzés: a módosítások mentésre kerülnek.
* **TE9 (Felhasználó törlése):** Meglévő felhasználó törlése. Ellenőrzés: a felhasználó törlődik a listából.

**Munka Kezelés:**

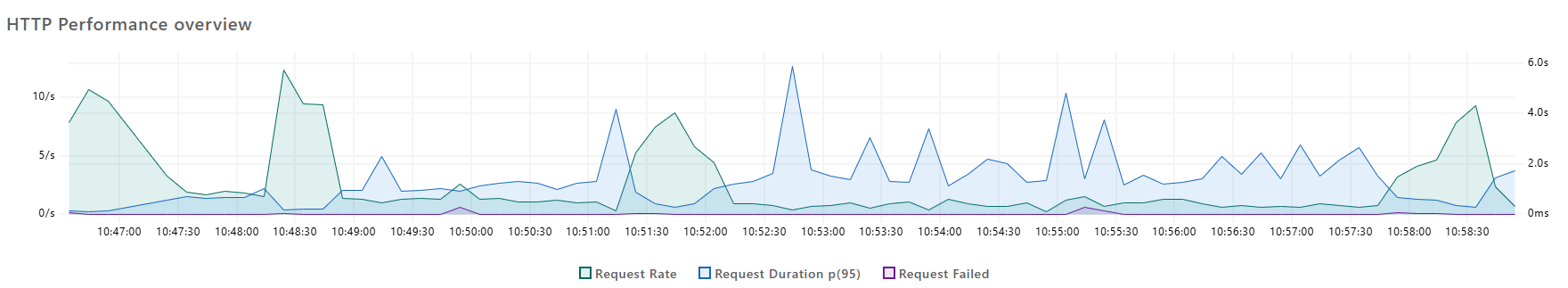
* **TE10 (Munka létrehozása):** Új munka létrehozása minden szükséges adat megadásával (ügyfél kiválasztása, cím megadása, státusz kiválasztása). Ellenőrzés: a munka megjelenik a munkák listájában.
* **TE11 (Munka szerkesztése):** Meglévő munka adatainak módosítása. Ellenőrzés: a módosítások mentésre kerültek.
* **TE12 (Munka törlése):** Meglévő munka törlése. Ellenőrzés: a munka törlődik a listából.
* **TE13 (Státusz módosítás):** Munka státuszának módosítása. Ellenőrzés: a státusz változás rögzítve van.

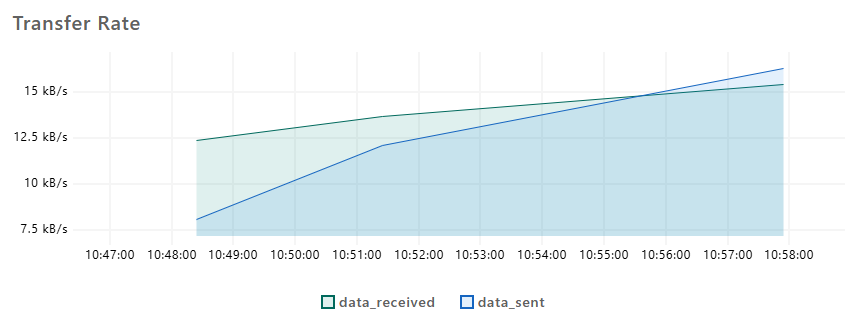
**Ügyfél Kezelés:**

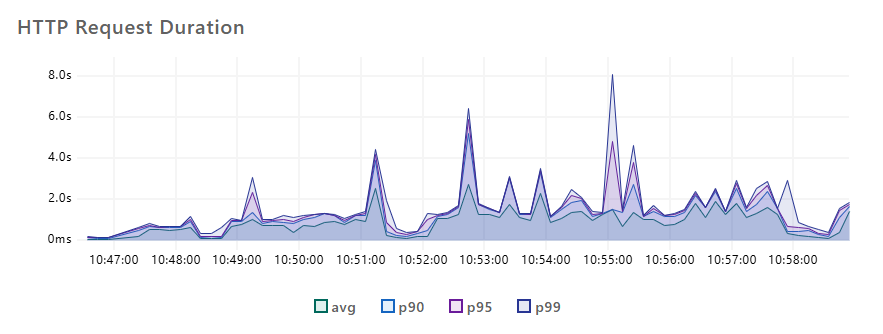
* **TE14 (Ügyfél hozzáadása):** Új ügyfél hozzáadása. Ellenőrzés: az ügyfél megjelenik az ügyfél listában.
* **TE15 (Ügyfél szerkesztése):** Meglévő ügyfél adatainak módosítása. Ellenőrzés: a módosítások mentésre kerültek.
* **TE16 (Ügyfél törlése):** Meglévő ügyfél törlése. Ellenőrzés: az ügyfél törlődik a listából, és a hozzá tartozó munkák is megfelelően kezelődnek.

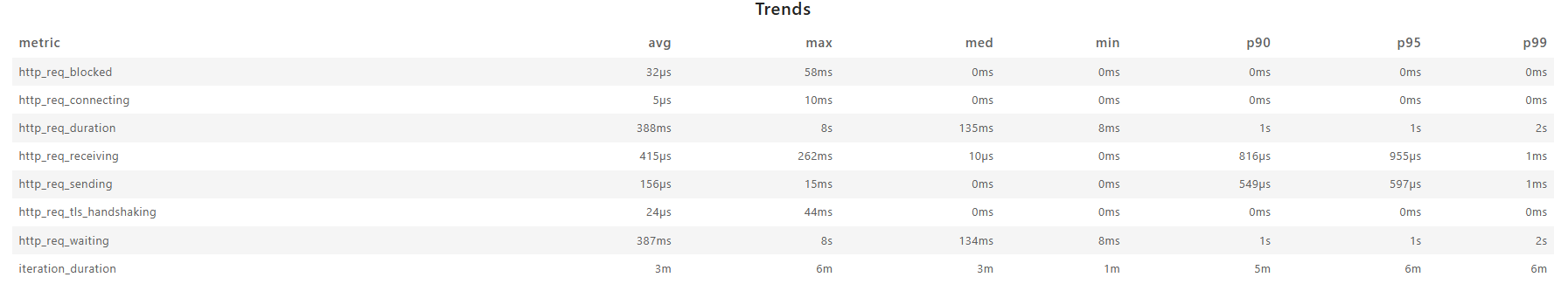
**Teljesítmény Tesztelés**

A k6 eszközzel a rendszer teljesítményét teszteljük különböző terhelések mellett. A teszt forgatókönyvek a rendszer különböző funkcióit szimulálják egyidejűleg több felhasználóval. A tesztelés során mért paraméterek: válaszidő, kérések/másodperc, hibaszázalék.









A terheléses teszt eredményekből az látszik, hogy folyamatosan növekvő terhelés mellett (összesen 100 felhasználóval) nagymértékű teljesítmény beszakadást nem tapasztaltunk. 15 kB/sec transfer rate mellett értünk el 8 sec válaszidőt tüskeszerűen, tehát egy-egy hosszabb válaszidő után ismét visszaállt a teljesítmény a terheléses teszt alatti átlagos 388 ms körüli értékre.

**Biztonsági Tesztelés**

A biztonsági tesztelés magában foglalja a rendszer sebezhetőségének vizsgálatát különböző támadásokkal szemben (pl. SQL injection, XSS).

# Összefoglalás

A fejlesztési munkák folyamán az Epic.CRM mini vállalkozásirányitási rendszer nagyon közel került a szívünkhöz. A projekt elején nem gondoltuk volna, de valóságosan potenciált és lehetőséget látunk benne. A valós életben való versenyképes használathoz a fejlesztés közben számos ötlet felmerült bennünk, ezek közül néhányat felsoroltunk a Továbbfejlesztési lehetőségek fejezetben. A munka során a legnagyobb kihívást a munka összehangolása és a feladatok priorizálása okozta. Ám ezt is sikerült abszolválnunk és sikerrel vettük az akadályokat, egymás közt kiosztottuk a feladatokat és rendszeresen egyeztettünk, összehangoltuk a munkafolyamatokat. Mindent összevetve nagyon élveztünk a projekten dolgozni és büszkék vagyunk az elkészült alkalmazásra, illetve ezen dokumentációra.