PÉCSI TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

Matematikai és Informatikai Intézet

**MUNKAIDŐ NYILVÁNTARTÁS ASZTALI ALKALMAZÁS**



Témavezető: KISS-VINCZE TAMÁS Készítette:

tanársegéd CSEH MARCELL (FLNYRL)

Programtervező informatikus BSc

Külső konzulens: LIPPAI TAMÁS

PÉCS, 2022

# Tartalomjegyzék

[I. Tartalomjegyzék 2](#_Toc102142113)

[II. Bevezetés 4](#_Toc102142114)

[III. Alkalmazás bemutatása 4](#_Toc102142115)

[1. Admin regisztráció 4](#_Toc102142116)

[2. Bejelentkezés 5](#_Toc102142117)

[3. Főoldal 5](#_Toc102142118)

[4. Profil 6](#_Toc102142119)

[5. Felhasználók 7](#_Toc102142120)

[5.1. Felhasználó profil 8](#_Toc102142121)

[6. Feladatok 10](#_Toc102142122)

[7. Rögzítések 11](#_Toc102142123)

[IV. Fejlesztési módszerek ismertetése 12](#_Toc102142124)

[1. A fejlesztés során használt eszközök 13](#_Toc102142125)

[1.1. Visual Studio 13](#_Toc102142126)

[1.2. MySQL Workbench 13](#_Toc102142127)

[1.3. GitHub 13](#_Toc102142128)

[1.4. Git Bash 14](#_Toc102142129)

[2. A fejlesztés során használt technológiák 14](#_Toc102142130)

[2.1. C# 14](#_Toc102142131)

[2.2. NET Framework 15](#_Toc102142132)

[2.3. NET Core 16](#_Toc102142133)

[2.4. WPF 16](#_Toc102142134)

[2.4.1. MVVM 17](#_Toc102142135)

[2.5. XAML 18](#_Toc102142136)

[2.6. Tesztelés 18](#_Toc102142137)

[2.6.1. Egységtesztelés 19](#_Toc102142138)

[2.7. MySQL 19](#_Toc102142139)

[2.8. Git 20](#_Toc102142140)

[3. A program szerkezeti felépítése 20](#_Toc102142141)

[3.1. Adatelérés 20](#_Toc102142142)

[3.2. Kivételkezelés (Logic) 21](#_Toc102142143)

[3.3. Model 21](#_Toc102142144)

[3.4. Model kiterjesztése 22](#_Toc102142145)

[3.5. Teszt 23](#_Toc102142146)

[3.6. Erőforrás 23](#_Toc102142147)

[3.7. WpfDemo 23](#_Toc102142148)

[V. Összefoglalás 23](#_Toc102142149)

[1. Továbbfejlesztési lehetőségek 23](#_Toc102142150)

[2. Forráskód elérhetősége 24](#_Toc102142151)

[3. Alkalmazás futtatása 24](#_Toc102142152)

[VI. Irodalomjegyzék 25](#_Toc102142153)

[VII. Ábrajegyzék 27](#_Toc102142154)

[VIII. Mellékletek 28](#_Toc102142155)

[1. Relációs diagram 28](#_Toc102142156)

[2. Osztálydiagramok 28](#_Toc102142157)

[NYILATKOZAT 31](#_Toc102142158)

# Bevezetés

A dolgozat bemutatja egy munkaidő nyilvántartás asztali alkalmazás működését és felépítését. Áttekinthetjük, hogy milyen fejlesztői eszközök, programozási nyelvek és keretrendszerek használatával valósítható meg egy ilyen alkalmazás létrehozása.

Az alkalmazás fejlesztésének kezdetekor már az eszközök megfelelő megválasztása is kihívás a lehetőségek nagy számossága miatt. Az általam használt fejlesztői eszközökön kívül sokkal több közül is választhatunk.

Az elképzelés az volt, hogy egy olyan programot hozzak létre, amivel akár napi szinten könnyen be lehet vinni a munkaidőben végzett tevékenységek óraszámát, minél kevesebb időráfordítással.

Az elkészített alkalmazás modern megoldást nyújt a felhasználók munkaidőben végzett tevékenységeik elkönyvelésében, azok követésében és vizsgálatában. A dolgozók rögzítései alapján megtudjuk vizsgálni, hogy mely feladat okoz nehézséget, eképpen tudjuk, hogy azon kell fejlődnie vagy más felhasználónak adni a jövőben a feladatot akinek ez nem okoz nehézséget.

Az elkészült programmal nyomon tudjuk követni a dolgozó tevékenységeit a feladataival.

Lehetőség van a felhasználó munkájának diagram alapján való vizsgálatára, illetve munkájának exportálására. Az exportálás Excel fájlba történik, aminek a célja az időszakos vizsgálat, avagy munka mentése, illetve lehetőség a nyomtatásra.

A fejlesztés során ügyeltem, hogy a programom esztétikus, könnyen kezelhető, felhasználóbarát legyen, illetve lehessen lehetőség továbbfejlesztésre.

A dolgozatban bemutatom a program működését és felépítését, illetve a használt technológiákat és eszközöket.

# Alkalmazás bemutatása

Az alkalmazás működési folyamatait és a felhasználói felületeket igyekeztem minél egyszerűbbnek, logikusabbnak és könnyen kezelhetőnek kialakítani. Az alábbi fejezetben bemutatom a megvalósított felhasználói felületeket és folyamatokat.

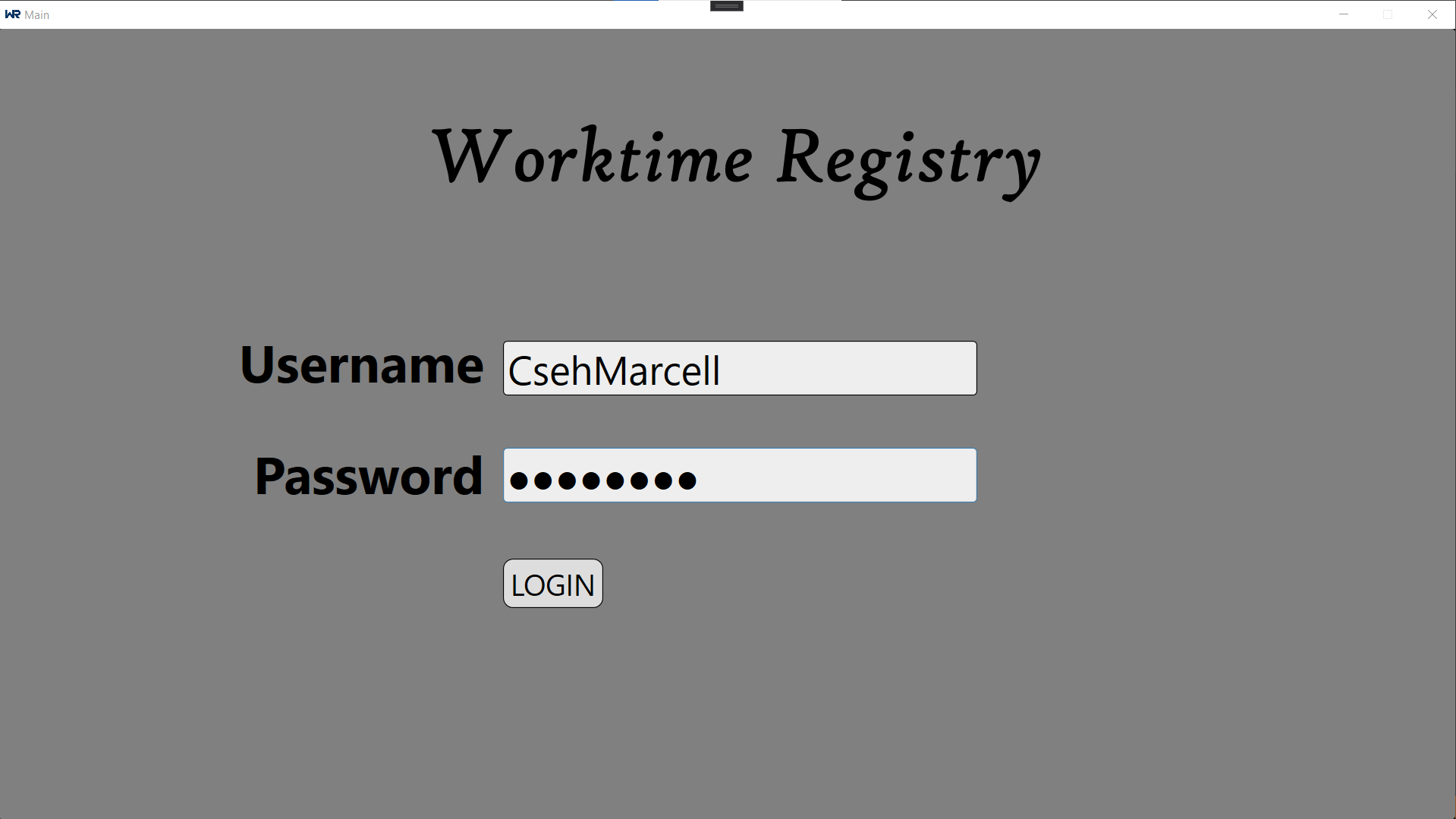
1. Admin regisztráció

Az alkalmazás indításakor kétféle felületre dobhat attól függően, hogy van-e létre hozva admin. Ha nincs létrehozva admin, akkor a (admin) regisztrációs felület jelenik meg, ahol adminként regisztrálsz, aki a főnök szerepet tölti be. Regisztráció során meg kell adnod a Felhasználónevet, jelszót, megerősítő jelszót, keresztnevet, vezetéknevet, email címet, megerősítő email címet, telefonszámot és a cégnek a nevét (illetve annak a megerősítését). Amennyiben a mezők kitöltése hiányos, formailag hibásak vagy a megerősítések nem egyeznek, arról a szoftver hibaüzenet formájában értesíti a felhasználót. Ha minden helyesen van megadva, akkor az alkalmazás létrehozza az admin felhasználót és a megadott emailre kapunk levelet a sikeres regisztrációról, illetve a megadott adatainkról, majd a főoldal felületre irányít át minket az alkalmazás.

Ellenkező esetben, ha már létezik admin, akkor a bejelentkezési felületre kerülünk.

1. Bejelentkezés

A bejelentkezési felületen a felhasználónév és a jelszó megadásával lehet belépni a főoldalra (1. ábra). Nem egyező felhasználónév és jelszó esetén a szoftver hibaüzenetként értesíti a felhasználót.

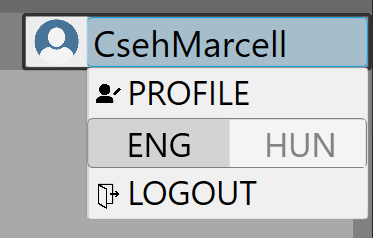


1. ábra – Bejelentkezési felület

1. Főoldal

A főoldal felületén lehet elérni az alkalmazás 3 főbb oldalát a „Rögzítések”, ,,Feladatok” és a ,,Felhasználók” felületet.

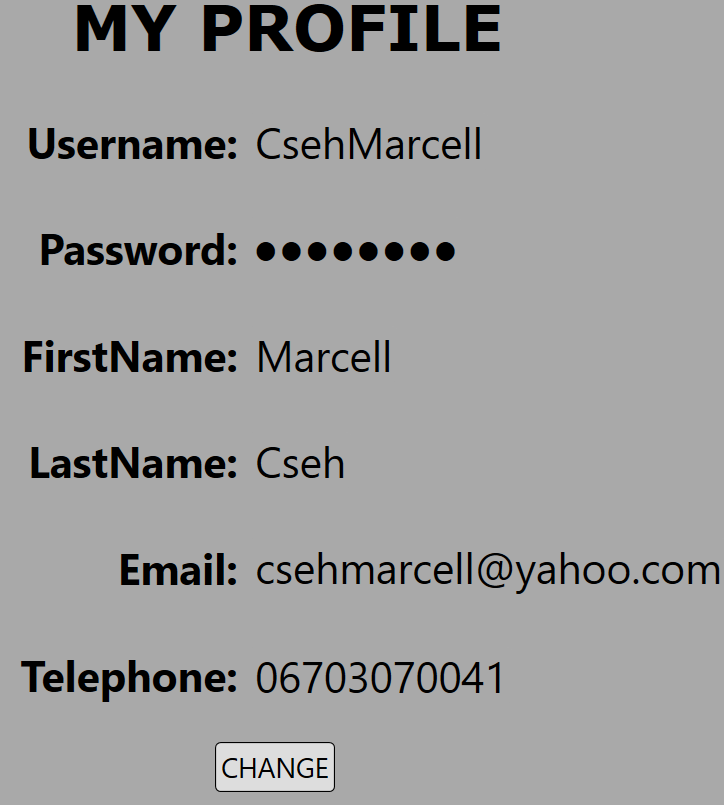
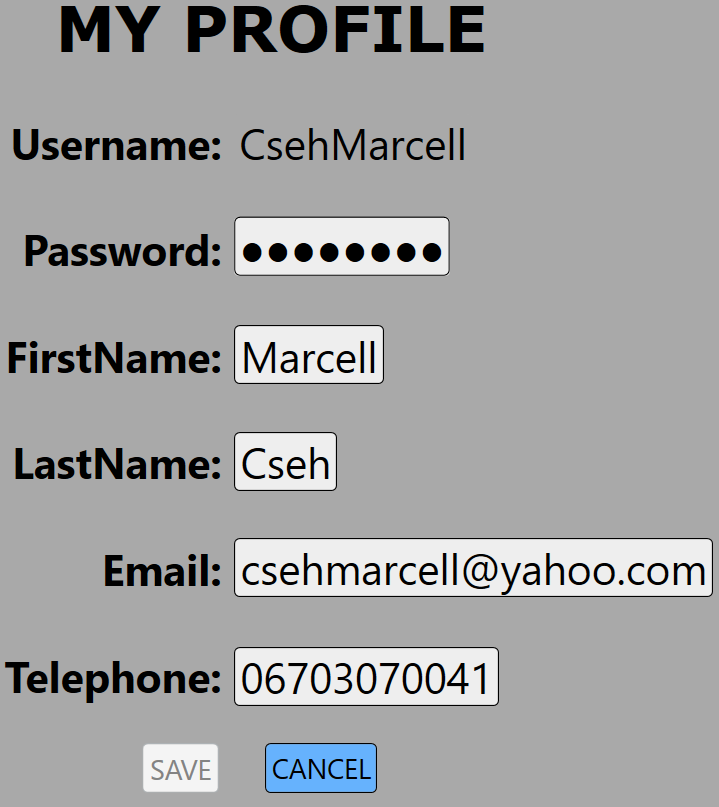
A felület jobb felső részénél kiírja a bejelentkezett felhasználónak a felhasználónevét, amire ha rákattintunk, akkor lenyílik egy menü (2. ábra). A menü első választása a saját profilunk megtekintése, amivel a profil ablak nyílik meg. A menü második választásával be lehet állítani az alkalmazás nyelvét angolra vagy magyarra. A menü harmadik és egyben utolsó választásával ki lehet jelentkezni, amivel a bejelentkezési felületre kerülünk.



2. ábra – Főoldal lenyíló menü

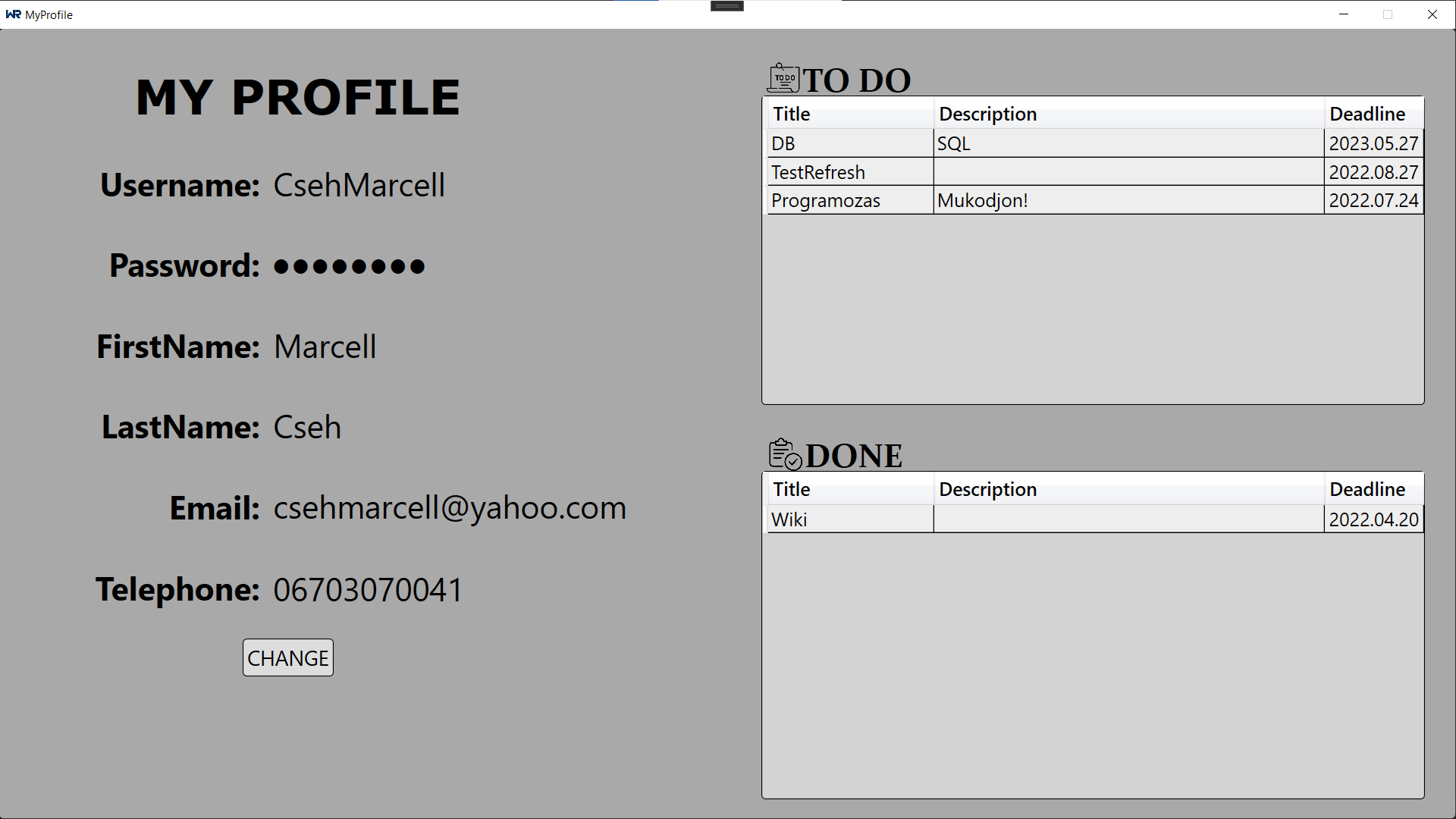
1. Profil

A profil bal oldalán láthatjuk a saját adatainkat (3. ábra). Az adataink közül a felhasználónév kivételével bármelyiket tudjuk módosítani (4. ábra), sikeres módosítás esetén emailt kapunk.

3. ábra – Profil: felhasználói adataim 4. ábra – Profil: adataim, ha megakarnám változtatni

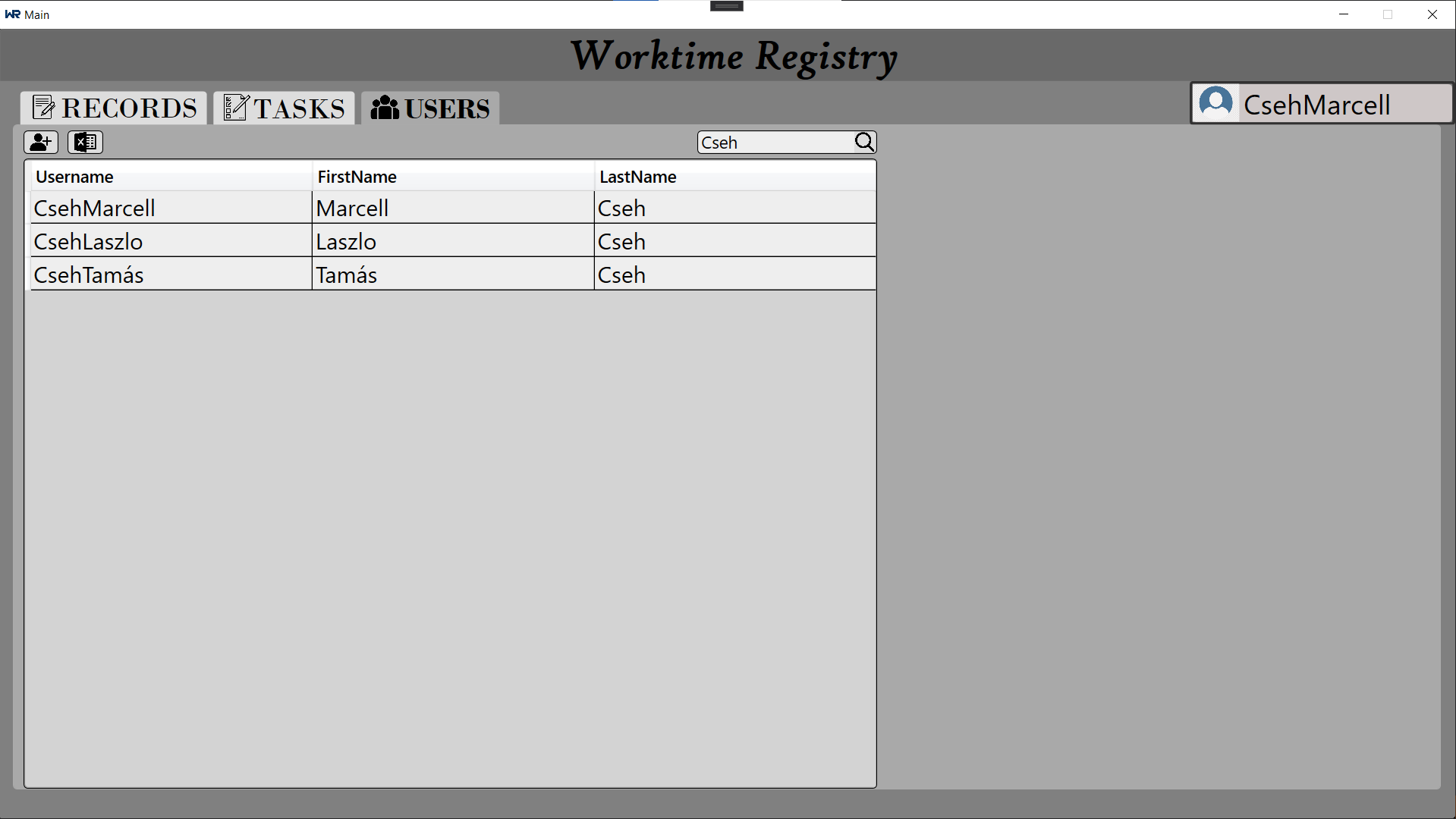
A felület jobb oldalán 2 lista látható (5. ábra). A felső listában szerepelnek az elvégzendő feladataink és az alatta lévő listában pedig az elvégzett feladataink. Ha a felhasználó nem rendelkezik elvégzendő vagy elvégzett feladatokkal, akkor azt szöveggel szemléltetjük.



5. ábra – Profil felület

1. Felhasználók

A ,,Felhasználók” az egyik lapja a főoldalnak. A felület bal oldalán egy lista található, ami szemlélteti az összes felhasználót. A listában kiírja a felhasználók felhasználónevét, keresztnevét és vezetéknevét. A lista jobb felső sarkanál van egy keresési mező, ahol a bevitt érték alapján keres a listában lévő felhasználónevek, keresztnevek és vezetéknevek közül. Ha a keresési érték egyezik a listában lévő értékek valamelyikével, akkor szűri arra a listát a keresés gomb megnyomása után (6. ábra).



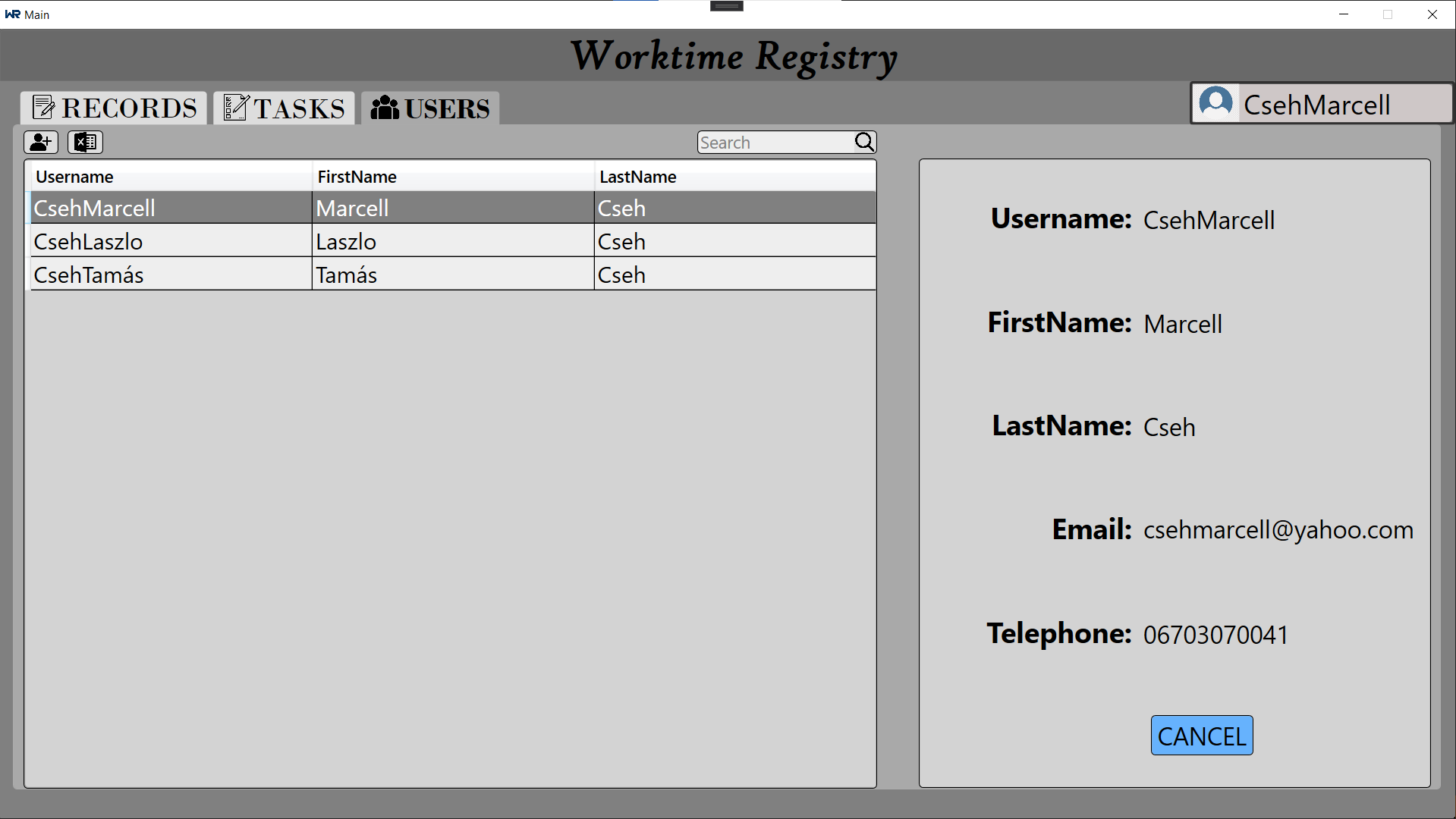
6. ábra – Felhasználók felület: lista szűrés után

Admin esetén a lista bal felső sarkánál van 2 gomb. A bal oldali gomb megnyomásával az admin létre tud hozni új felhasználókat, ami a felület jobb oldalán jelenik meg és ott kell megadni a felhasználónevét és az email címet. Amennyiben a kitöltendő mezők hibásak vagy nem egyedi a felhasználónév a rendszer hibaüzenet formájában jelzi. Sikeres létrehozás esetén a megadott email címre érkezik levél, amiben szerepelni fognak a felhasználó ideiglenes adatai azaz a felhasználóneve, véletlenül generált 10 karakter hosszú jelszó és az email címe. A felhasználó, így be tud lépni és megtudja adni a hiányzó adatait, illetve képes megváltoztatni a jelszavát vagy email címét.

A másik gomb pedig lementi a jelenlegi listát egy Excel fájlba.

A listában lévő felhasználóra kattintva megjelenik a felület jobb oldalán a felhasználó adatai a jelszaván kívül (7. ábra). Dupla bal klikk esetén megnyitja a felhasználó profiljának ablakát.

Az admin tudja törölni a listában lévő felhasználót, ha jobb klikkel kattint rá és kiválasztja a ,,Törlés” lehetőségre.



7. ábra – Felhasználók felület: felhasználóra kattintva (megjelenik jobb oldalt a felhasználó adatai)

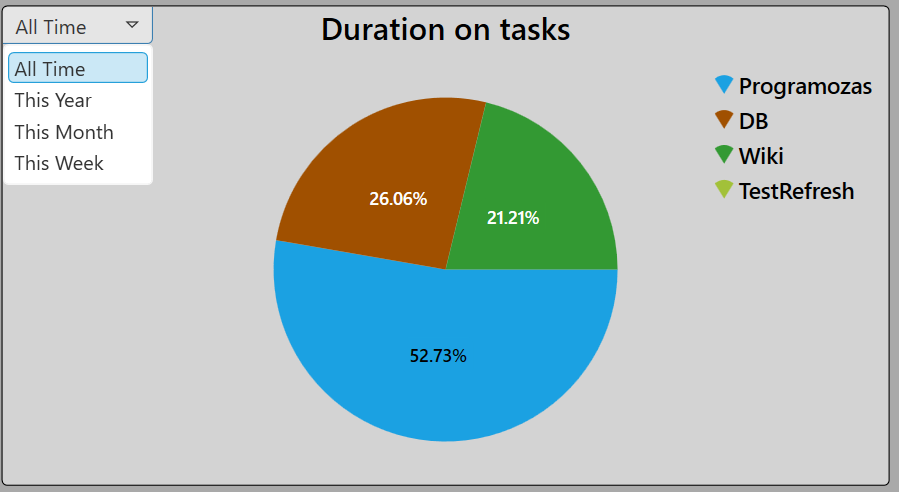
* 1. Felhasználó profil

A felület tetején kiírja a kiválasztott felhasználó felhasználónevét, hogy nehogy elfelejtsük kinek a profilját tekintjük át.

A felület felső részén 2 lista található (9. ábra). A bal oldali lista a felhasználó feladatait, a jobb oldali peidg a rögzítéseit szemlélteti. A listák alatt diagramok szemléltetik a felhasználó tevékenységeit. A feladat lista alatt, bal oldalt egy kördiagram mutatja meg, hogy a feladatokkal mennyi időt foglalkozott a felhasználó. A felhasználói felület jobb alsó részén pedig egy oszlopdiagram szemlélteti a kiválasztott feladat rögzítéseinek időtartamait.

A feladat listában kiírja a feladatok címét, leírását, határidejét, státuszát. Ha a felhasználó nem rendelkezik feladattal, akkor azt szöveggel szemléltetjük. A listában lehet keresni cím, leírás, státusz alapján és lehet szűrni határidő szerint. Ha a keresési érték egyezik a listában lévő értékek valamelyikével, akkor szűri arra a listát. A keresés gomb mellett van egy visszaállító gomb, aminek a célja, hogy a szűrő határidőket visszaállítja alapértékeire és így a lista is frissül. A feladatra kattintva megjelenik a mellette lévő rögzítés listában a feladathoz tartozó rögzítések, illetve az oszlopdiagram feltöltődik a megfelelő rögzítések időtartamaival. A feladatra jobb klikkelve feljön sima felhasználó esetén csak az ,,Eltöltött idő” lehetőség, ami kiírja nekünk, hogy mennyit foglalkozott vele az adott felhasználó. Admin esetén pedig jobb klikkre még feljön a ,,Törlés” lehetősége. Admin esetén duplán klikkelve lehet a feladatot módosítani, illetve a lista mellett megjelenik egy hozzáadás gomb, amivel létre lehet hozni új feladatot a felhasználónak.

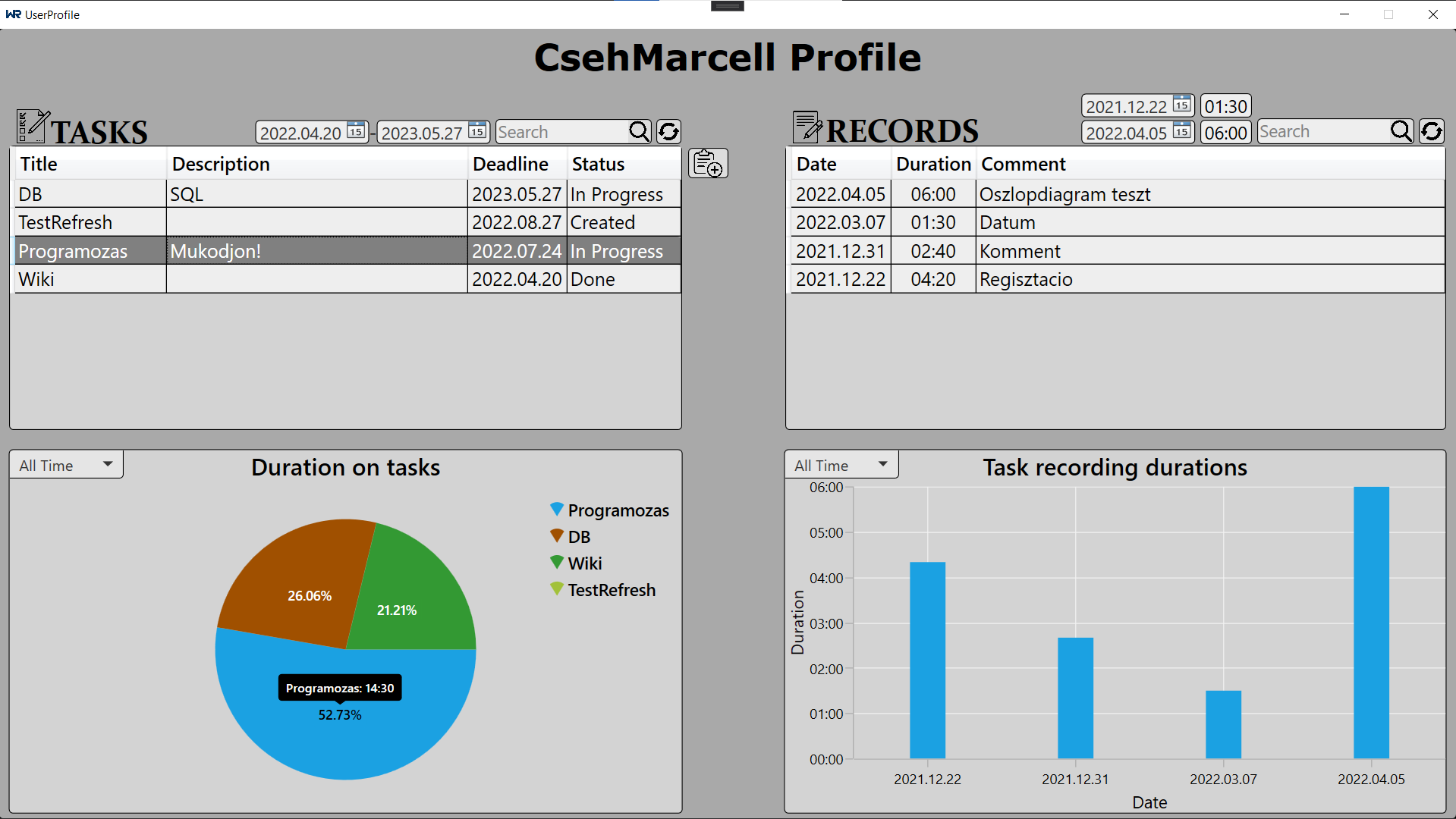
A kördiagramba szereplő feladathoz tartozó időtartamot lehet szűrni: hétre, hónapra, évre és mindenkorra (8. ábra).



8. ábra – Felhasználói profil: kördiagram

A felhasználói profil felület job oldali listájában kiírja a kiválasztott feladathoz tartozó munkavégzés rögzítéseket, azaz a dátumot, időtartamot, megjegyzést. Ha nincs kiválasztva feladat vagy a kiválasztott feladathoz nem tartozik munkavégzés rögzítés, akkor azt szöveggel szemléltetjük. A listában szintén lehet keresni még pedig a megjegyzés alapján, illetve lehet szűrni a dátum és időtartam szerint. Ha a keresési érték egyezik a listában lévő értékek valamelyikével, akkor szűri arra a listát. A keresés gomb mellett ugyanúgy van egy visszaállító gomb, ami a szűrési dátumokat és szűrési időtartamokat állítja vissza alapértékére és a lista is frissül.

Az oszlopdiagramba lévő rögzítések időtartamát lehet szűrni: hétre, hónapra, évre és mindenkorra.



9. ábra – Felhasználói profil felület

1. Feladatok

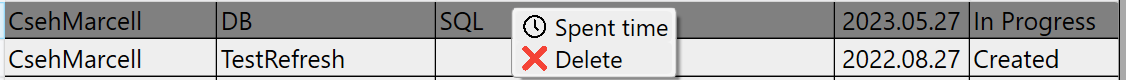
A ,,Feladatok” az egyik lapja a főoldalnak. A felület bal oldalán egy lista szemlélteti a felhasználó feladtait határidők szerint rendezve. A listában kiírja a feladatok címét, admin esetén a felhasználót (aki kapta a feladatot), leírást, határidőt, státuszt. Ha a felhasználó nem rendelkezik feladattal, akkor azt szöveggel szemléltetjük. A lista fölötti jobb oldalon van egy keresési mező, ahol a bevitt érték alapján keres a listában lévő címek, admin esetén felhasználók, leírás és státusz alapján, illetve lehet szűrni határidő szerint. Ha a keresési érték egyezik a listában lévő értékek valamelyikével, akkor szűri arra a listát a keresés gomb megnyomása után. A keresés gomb mellett van a visszaállító gomb, mely a szűrő határidőket visszaállítja alapértékeire és így a lista is frissül.

A lista fölött bal oldalon admin esetén van feladat létrehozása gomb, amivel a felület jobb oldalánál jelenik meg a feladat és ott kell megadni a felhasználót akinek kiadja a feladatot, a feladat címét, leírást (opcionális), határidőt és a státusz ilyenkor alapból be lesz állítva létrehozottra. Amennyiben a kitöltendő mezők hibásak a rendszer hibaüzenet formájában jelzi. Sikeres létrehozáskor a kitűzött felhasználó emailt kap a feladatról, illetve értesítést a feladathoz. A mellette lévő gomb az Excel gomb, ami exportálja a jelenlegi listát egy Excel fájlba.

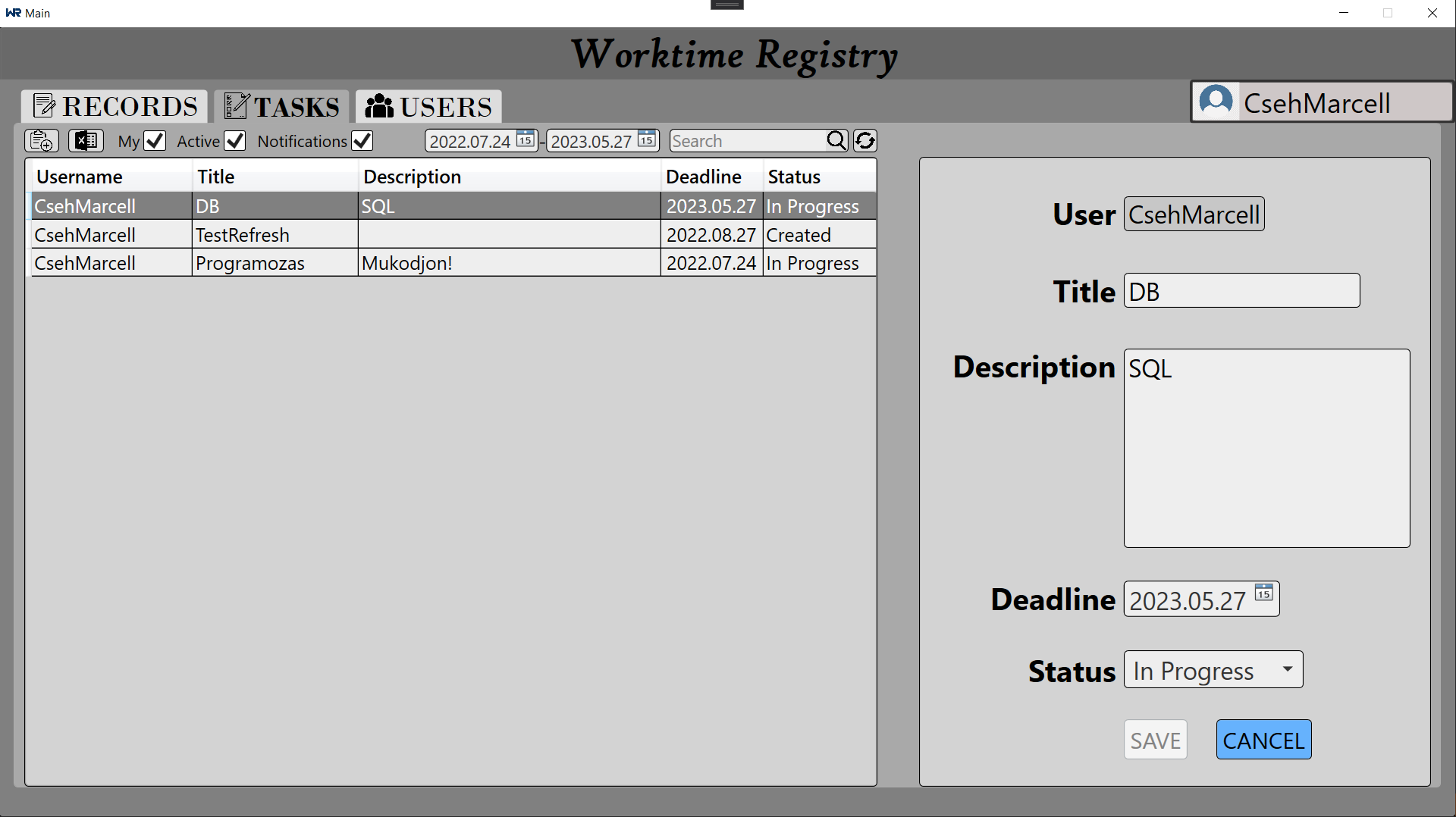
Admin esetén van ,,Saját” checkbox, ami alapból ki van pipálva, de ha nincs, akkor látja mindenkinek a feladatát. Ezen kívül még van 2 checkbox. Az egyik az ,,Aktív”, amivel az elvégzendő feladatokat listázza ki és az elvégzetteket pedig nem. A másik checkbox pedig az ,,Értesítések”, amivel a feladathoz tartozó megfelelő értesítést mutatja (ha van).

A listában lévő feladatra kattintva tudjuk megtekinteni a feladatot és a módosítani azt (11. ábra). A sima felhasználó csak a feladat státuszát tudja átállítani, amiről az admin értesítést kap a feladathoz. Az admin pedig a feladat címét, leírását, határidejét és a sajátjának a státuszát tudja módosítani, ha nem a sajátját módosítja, akkor az a felhasználó akinek a feladatát módosította, kap értesítést a feladathoz arról, ami változott.

A listában lévő feladatra jobb klikkelve feljön az ,,Eltöltött idő” lehetősége, ami kiírja nekünk, hogy mennyit foglalkozott vele az adott felhasználó. Admin esetén még feljön a ,,Törlés” lehetősége is (10. ábra).



10. ábra – Feladatok felület: feladatra kattintva lenyíló menü (admin esetén)



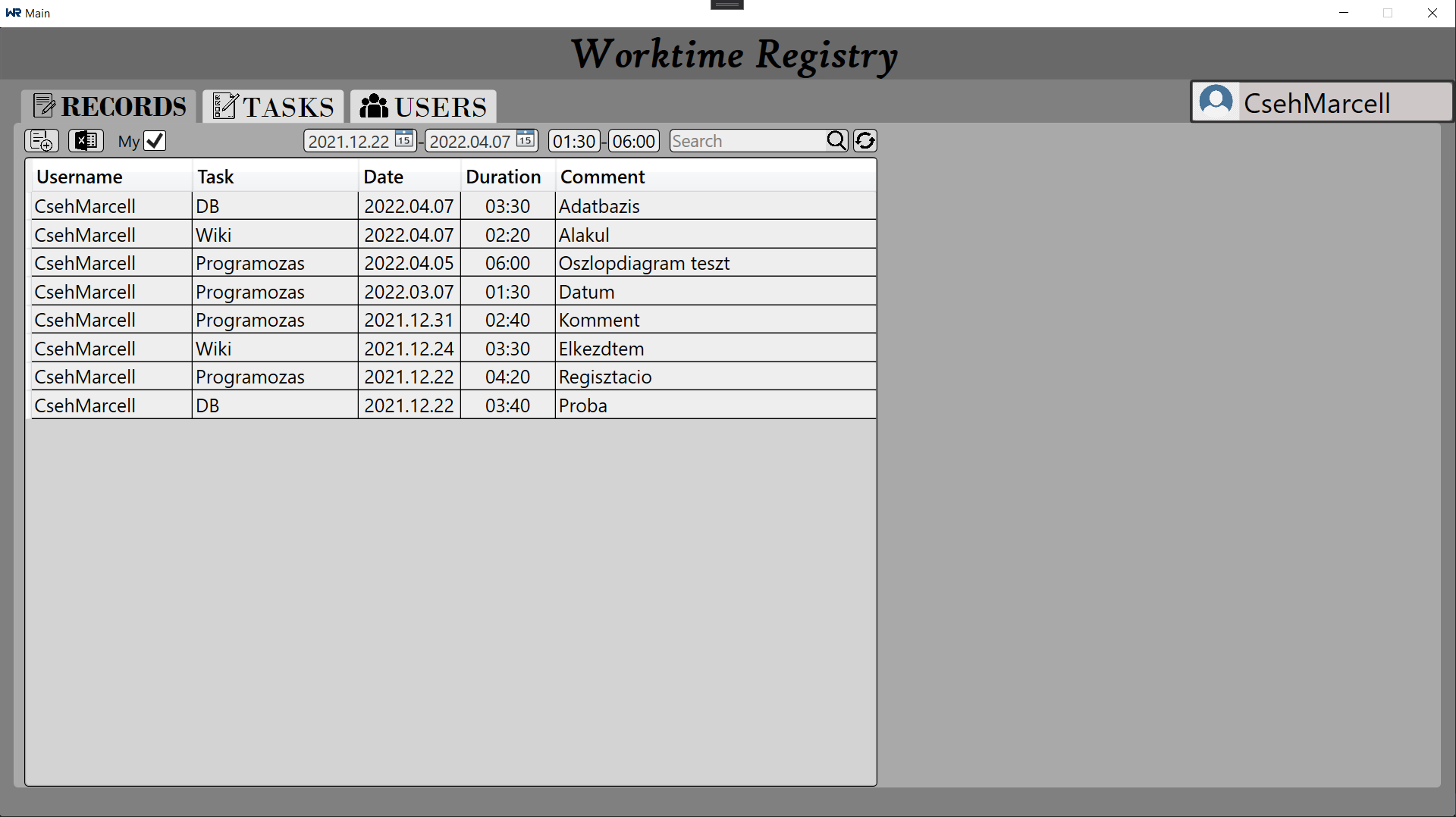
11. ábra – Feladatok felület: feladatra kattintva (megjelenik jobb oldalt a feladat)

1. Rögzítések

A ,,Rögzítések” az egyik lapja a főoldalnak (12. ábra). A felület bal oldalán egy lista szemlélteti a felhasználó munkavégzés rögzítéseit dátum szerint rendezve. A rögzítés listában kiírja a feladatot (amihez tartozik a munkavégzés rögzítés), admin esetén a felhasználót, dátumot, időtartamot, megjegyzést. Ha a felhasználónak nincs rögzítése, akkor azt szöveggel szemléltetjük. A lista fölött baloldalt van 2 gomb. A baloldali gombbal létre lehet hozni munkavégzés rögzítéseket, míg a mellette lévővel a jelenlegi listát exportálja egy Excel fájlba. Admin esetén az Excel gomb mellett van egy ,,Saját” checkbox, amit ha nem pipál ki akkor láthatja más rögzítéseit, de az alapból ki lesz pipálva. A lista fölött jobbra van egy keresési mező, ahol a bevitt érték alapján keres a listában lévő feladat, megjegyzés, admin esetén felhasználónév alapján és lehet szűrni dátum, illetve időtartam szerint a keresés gombbal. Ha a keresési érték egyezik a listában lévő értékek valamelyikével, akkor szűri arra a listát. A keresés gomb mellett van a visszaállító gomb, ami a szűrési dátumokat és szűrési időtartamokat állítja vissza alapértékére és ekképpen a lista is frissül.

Új munkavégzés létrehozásánál a felület jobb oldalánál jelennek meg a munkavégzés adatai, ahol meg kell adnod a feladatot, a feladattal eltöltött időtartamot, ennek dátumát és megjegyzést (opcionális). Amennyiben a kitöltendő mezők hibásak a rendszer hibaüzenet formájában jelzi.

A listában lévő rögzítésre kattintva tudod azt módosítani, viszont az admin másét már nem módosíthatja. A kiválasztott munkavégzésre jobb klikkelve jön fel a ,,Törlés” lehetőség.



12. ábra – Rögzítések felület

# Fejlesztési módszerek ismertetése

Az általam választott technológiák alapja a C# programnyelvre épülő .NET keretrendszer. A program az MVVM (Model-View-ViewModel) programtervezési mintára épül. Az alkalmazásban a Backend és a Frontend részeket is egy projekten belül kezeljük. A felhasználó-, és jogosultságkezelésre a keretrendszer által nyújtott technológiát használjuk. Az adatokat a szoftver MySQL adatbázisban tárolja, melyet az Entity Framework ORM segítségével használ.

1. A fejlesztés során használt eszközök

A szoftverek fejlesztése során a fejlesztőknek rengeteg eszköz áll rendelkezésre, amelyek nagyban megkönnyítik a munkájukat. Ezen eszközök közül szabadon válogathatnak, így egyszerűsítve a bonyolultabb folyamatokat. A program fejlesztése során én is több olyan eszközt használtam, melyek részben, vagy egészben kiválthatók lettek volna parancssoros felhasználói felület használatával, azonban használatuk megkönnyítette és gyorsította a fejlesztési folyamatokat.

* 1. Visual Studio

A fejlesztés során a Visual Studio 2019 alkalmazást használtam. Ez egy több programozási nyelvet tartalmazó fejlesztőkörnyezet, amit a Microsoft fejleszt. A szoftverbe integrált úgynevezett IntelliSense nagyban segíti a fejlesztést az éppen használt programnyelvhez igazodva automatikusan felajánlott súgóüzenetekkel, illetve kódkiegészítés funkcióival. Az elérhető bővítmények segítségével könnyedén testre szabható az alkalmazási területhez [1].

* 1. MySQL Workbench

Az alkalmazás fejlesztése során az adatbázis kezeléshez a MySQL Workbench szoftvert használtam. A MySQL Workbench egy vizuális adatbázis-tervező eszköz, amely integrálja az SQL fejlesztést, adminisztrációt, adatbázis-tervezést, -létrehozást és –karbantartást egyetlen integrált fejlesztői környezetbe a MySQL adatbázisrendszer számára [2].

* 1. GitHub

A fejlesztés során, hogy ne vesszen el a munkám bármi oknál fogva, a GitHub-ot használtam tárolásként, illetve fájljaim és alkalmazás verzióim nyomon követésére. A GitHub a [Git](https://hu.wikipedia.org/wiki/Git) segítségével szoftverfejlesztési verziókövetés-szolgáltatást, hozzáférés-kezelést és számos együttműködési funkciót nyújt, mint például [bug követés](https://hu.wikipedia.org/wiki/Bug_(informatika)), szolgáltatáslekérés, feladatkezelés, valamint [wikiket](https://hu.wikipedia.org/wiki/Wiki) minden projekthez.

A GitHubon található projekteket a Git parancssorral lehet elérni és használni és minden alapvető Git parancs működik rajta. A GitHub továbbá lehetővé teszi regisztrált és nem regisztrált felhasználóknak is nyilvános tárolók böngészését az oldalon. Több asztali alkalmazást és Git plugint is létrehozott a GitHub és egyéb harmadik felek akik egyesülnek a platformmal [3].

* 1. Git Bash

A program fejlesztése során a Git parancssori kezeléséhez a Git Bash alkalmazást használtam. A Git Bash egy Microsoft Windows környezetekhez készült alkalmazás, amely emulációs réteget biztosít a Git parancssori élményhez. A Bash a Bourne Again Shell rövidítése. A shell egy terminálalkalmazás, amelyet az operációs rendszerrel írásos parancsokon keresztül kommunikálnak [4].

1. A fejlesztés során használt technológiák

Az alkalmazott technológiák kiválasztása során a rendelkezésre álló számos lehetséges alternatíva közül igyekeztem olyan naprakész, modern technológiákat kiválasztani, amelyeket folyamatosan fejlesztenek és az iparban is használják őket. Ezekkel kapcsolatban már egy részével gyakorlati tapasztalatokkal is rendelkeztem, néhány pedig új volt számomra.

* 1. C#

A C# egy modern objektumorientált és erősen típusos programozási nyelv. A nyelv a C programozási nyelv alapjaira épül, és sok hasonlóságot mutat a C, C++, Java és JavaScript nyelvekkel [5]. A C#-ot úgy tervezték, hogy meglegyen az egyensúly a fejlesztő nyelvi szabadsága és a gyors alkalmazásfejlesztés lehetősége között. A C# az a programozási nyelv, ami a legközvetlenebb módon tükrözi az alatta működő, minden .NET programot futtató .NET keretrendszert, valamint erősen függ is attól: nincsen nem menedzselt, natív módban futó C# program. A primitív adattípusai objektumok, a .NET típusok megfelelői. Szemétgyűjtést használ, valamint az absztrakcióinak többsége (osztályok, interfészek, delegáltak, kivételek...) a .NET futtatórendszert használja közvetlen módon. A C vagy C++ nyelvhez hasonlítva a C# több korlátozást és továbbfejlesztést is tartalmaz. A legtöbb objektumhozzáférés csak biztonságos hivatkozásokon keresztül tehető meg, és a legtöbb művelet ellenőrzött túlcsordulás szempontjából. Az objektumok nem szabadíthatók fel közvetlen módon, ehelyett a szemétgyűjtő szabadítja fel őket, mikor már nincs rájuk hivatkozás. Ez a módszer kizárja a nem létező objektumokra való hivatkozás lehetőségét. A nyelv csak egyszeres öröklődést támogat, de egy osztály több interfészt is megvalósíthat. A C# sokkal típusbiztosabb, mint a C++. Az egyetlen implicit konverzió a biztonságos konverzió, úgy mint az egészek tágabb intervallumba konvertálása vagy a leszármazott osztályok alaposztályba konvertálása. Nincs implicit konverzió az egészek és a logikai típus (boolean) között, a felsorolás tagok és az egészek között. Nincsenek void mutatók (bár az Object osztályra mutató mutatók hasonlóak), valamint bármely, a felhasználó által definiált implicit konverziót explicit módon meg kell jelölni. A felsorolás adattagjai a saját névterükben helyezkednek el. Tulajdonságok (Properties) használhatók, amelyek úgy tesznek lehetővé kódfuttatást mezők beállításakor és olvasásakor, mintha mezőhozzáférés történne. Teljes reflexió elérhető. A legtöbb programozási nyelvtől eltérően a C# megvalósítások nem rendelkeznek önálló, eltérő osztály- vagy függvénykönyvtárakkal. Ehelyett a C# szorosan kötődik a .NET keretrendszerhez, amitől a C# kapja a futtató osztályait és függvényeit. A .NET keretrendszer osztálykönyvtárat tartalmaz, ami a .NET nyelvekből felhasználható egyszerű feladatok (adat reprezentáció és szöveg manipuláció) végrehajtásától kezdve a bonyolult (dinamikus ASP.NET weblapok generálása, XML feldolgozás és reflekció) feladatokig. A kód névterekbe van rendezve, mely a hasonló funkciót ellátó osztályokat fogja össze. Például System.Drawing a grafikai, System.Collections az adatstruktúra és System.Windows.Forms a Windows Forms funkciókat fogja össze [6].

* 1. NET Framework

A [Microsoft](https://hu.wikipedia.org/wiki/Microsoft) által készített .NET keretrendszer (a .NET Framework) [gyors alkalmazásfejlesztést](https://hu.wikipedia.org/wiki/Gyors_alkalmaz%C3%A1sfejleszt%C3%A9s), [platformfüggetlenséget](https://hu.wikipedia.org/wiki/Platformf%C3%BCggetlens%C3%A9g) és hálózati átlátszóságot támogató [szoftverfejlesztői](https://hu.wikipedia.org/wiki/Szoftver) platform.

A .NET Framework eszköztára a szoftverfejlesztés szinte minden aspektusát ([kliens](https://hu.wikipedia.org/wiki/Kliens)-, illetve [szerveroldali](https://hu.wikipedia.org/wiki/Szerver) megoldások, [adatbázisok kezelése](https://hu.wikipedia.org/wiki/Adatb%C3%A1zis-kezel%C5%91_rendszer), játékfejlesztés stb.) lefedi. A .NET Framework alapját a CLI, vagyis a [Common Language Infrastructure](https://hu.wikipedia.org/wiki/Common_Language_Infrastructure) képezi. Ez nem más, mint azon szabályok halmaza, amelyek leírnak egy nyelvfüggetlen fejlesztői környezetet, a futtatókörnyezetet, típusrendszert stb. A CLI-t úgy tervezték, hogy bármilyen objektumorientált programozási nyelvet támogasson, megosztva egy közös objektum modellt és egy nagy, közös osztálykönyvtárat.

A beépített nyelvek a [C#](https://hu.wikipedia.org/wiki/C_Sharp), [C++](https://hu.wikipedia.org/wiki/C%2B%2B)-hoz és a [Sun](https://hu.wikipedia.org/wiki/Sun_Microsystems) [Java](https://hu.wikipedia.org/wiki/Java_(programoz%C3%A1si_nyelv)) nyelvéhez hasonló objektumorientált nyelv [7].

* 1. NET Core

A .NET (.NET Core) egy ingyenes és nyílt forráskódú, felügyelt számítógépes szoftver keretrendszer Windows, Linux és macOS operációs rendszerekhez. Ez egy többplatformos utódja a .NET-keretrendszernek [8].

A .NET négy platformközi felületet támogat: ASP.NET Core webalkalmazások, parancssori/konzolalkalmazások, könyvtárak, és univerzális Windows Platform alkalmazások. A .NET Core 3.0 előtt nem valósította meg a Windows Forms-t vagy a Windows Presentation Foundation-t (WPF), amelyek az asztali szoftverek szabványos grafikus felhasználói felületét jelenítik meg Windowson. Most azonban a .NET Core 3.0 már támogatja a Windows Forms, WPF technológiákat. Lehetőség van többplatformos grafikus alkalmazások írására is a .NET használatával.

A .NET parancssori felület végrehajtási belépési pontot kínál az operációs rendszerek számára, és olyan fejlesztői szolgáltatásokat nyújt, mint a fordítás és a csomagkezelés [9].

A .NET támogatja a NuGet-csomagok használatát [10].

* 1. WPF

A Windows Presentation Foundation (WPF) [grafikus felhasználói felületek](https://hu.wikipedia.org/wiki/Grafikus_felhaszn%C3%A1l%C3%B3i_fel%C3%BClet) készítéséhez használatos osztálykönyvtár, fejlesztője a [Microsoft](https://hu.wikipedia.org/wiki/Microsoft). A WPF a [.NET keretrendszer](https://hu.wikipedia.org/wiki/.NET_keretrendszer) 3.0 verziójában jelent meg [11].

A WPF egyik fő újítása a korábbi ablaktervező megoldáshoz képest a felület és az üzleti logika szétválasztása. Az ablakok tervezése egy XML alapú jelölőnyelvvel, az [XAML](https://hu.wikipedia.org/wiki/XAML)-lel történik. A jelölőnyelv, illetve a feladatok szétválasztása a programozók és látványért felelős munkatársak könnyebb együttműködését teszi lehetővé.

A WPF alkalmazások grafikai elemei [vektorgrafikusak](https://hu.wikipedia.org/wiki/Vektorgrafika), ezáltal lehetővé teszik az esztétikus átméretezést, és lényegesen kevesebb tárterületet foglalnak.

Az adatkötés a [felhasználói felület](https://hu.wikipedia.org/wiki/Felhaszn%C3%A1l%C3%B3i_fel%C3%BClet) és az üzleti logika között teremti meg a kapcsolatot. Ez több úton-módon történhet, az egyirányú, egyszeri kapcsolattól a kétirányú, szinkronban tartott kapcsolatig.

Az alkalmazások fejlesztéséhez, felületük kialakításához számos [integrált fejlesztői környezet](https://hu.wikipedia.org/wiki/Integr%C3%A1lt_fejleszt%C5%91i_k%C3%B6rnyezet) létezik, a legelterjedtebb a [Microsoft Visual Studio](https://hu.wikipedia.org/wiki/Microsoft_Visual_Studio). A WPF alkalmazásfejlesztés számos [programozási nyelven](https://hu.wikipedia.org/wiki/Programoz%C3%A1si_nyelv) történhet, közülük a legtöbbet használt a [C#](https://hu.wikipedia.org/wiki/C).

A WPF az MVVM (Model-View-ViewModel) programtervezési mintára épül [12].

2.4.1. MVVM

A modell-nézet-nézetmodell minta (MVVM) egy [architekturális minta](https://hu.wikipedia.org/wiki/Architektur%C3%A1lis_minta).

A minta leválasztja a [grafikus felhasználói felületet](https://hu.wikipedia.org/wiki/Grafikus_felhaszn%C3%A1l%C3%B3i_fel%C3%BClet) és az üzleti logikát (adatmodell). A nézetmodell értékkonverter, ez a felelős az adatok átalakításáért a könnyű kezelhetőséghez és reprezentálásához. A nézetmodell inkább modell, mint nézet, de az hozza létre a megjelenítés logikáját is. Megvalósíthatja a [közvetítő programtervezési mintát](https://hu.wikipedia.org/wiki/K%C3%B6zvet%C3%ADt%C5%91_programtervez%C3%A9si_minta) is, megszervezve a hozzáférést a nézet használati esetei számára.

A modell-nézet-nézetmodell esetén a prezentációs modell nézete nem függ a felhasználó platformjától. A modell-nézet-nézetmodell és a prezentációs modell származtatható a [modell-nézet-vezérlő mintából](https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C5%91) [13].

Mindkét mintát a Microsoft fejlesztette ki, hogy egyszerűsítsék a felhasználó interfészek eseményvezérelt programozását. A modell-nézet-nézetmodellre úgy is hivatkoznak, mint modell-nézet-összekötő, különösen a .NET platformján kívül.

Részei a modell, nézet és nézetmodell.

A modell a tartománymodellre utal, ami a tartalom állapotát reprezentálja (objektumorientált módban), vagy adathozzáférési réteg (adatközpontú megközelítés).

A nézet egy szerkezet, elrendezés vagy megjelenítés, amit a felhasználó a képernyőn láthat. Ez a GUI (Grafikus felhasználói felület).

A nézetmodell a nézet absztrakciója, ami publikus tulajdonságokat és metódusokat tartalmaz. Az MVC modell vezérlője vagy az MVP megjelentője helyett az MVVM összekötőt tartalmaz, ami a nézetmodellben közvetít a nézet és az adatok között. A nézetmodellt az adatok állapotaként írják le a modellben.

Az összekötőt a minta implicit tartalmazza. A Microsoft solution stackjében az összekötő egy leírónyelv, a [XAML](https://hu.wikipedia.org/wiki/XAML). Ez segíti a programozót abban, hogy ne kelljen szószátyár kódot írnia a nézetmodell és a nézet között. Általában az ennek megvalósítására szolgáló technológia kulcsfontosságú a minta megvalósításában.

Az MVVM-et úgy tervezték, hogy a WPF adatkapcsolati függvényeket használja, hogy jobban elkülönítse a nézet fejlesztését a minta többi részétől. Virtuálisan eltávolítja a GUI kódját a nézet rétegből. A direkt kódolás helyett leírónyelvet használhatnak, és adatkapcsolatokat hozzanak létre a nézetmodellel, amit alkalmazásfejlesztők készítenek és tartanak karban. A szerepek elkülönítése miatt a nézet készítőinek nem kell foglalkozniuk az üzleti logikával. A rétegeket külön személyek vagy csapatok fejleszthetik, így jobban be lehet tartani a határidőket.

A minta egyesíteni próbálja az MVC és az adatösszekötés előnyeit, miközben a keretrendszert az adatok összekapcsolásával olyan tisztán tartja, emennyire csak lehet. Használja az összekötőt, a nézetmodellt és az üzleti réteg adatellenőrzését a bejövő adatok ellenőrzéséhez. Ennek eredményeként a modell és a keretrendszer végzi el a legtöbb műveletet, kiküszöbölve vagy minimalizálva az alkalmazáslogikát, amivel közvetlenül kezeli a nézetet.

A mintának a hatránya, hogy egyszerű műveletek a minta használata esetén lelassulnak, különösen nagyobb alkalmazások esetén. Az adatösszekötés nagyon nagy méretű alkalmazásokban a memóriafogyasztást is érezhetően megnöveli [14].



13. ábra - MVVM

* 1. XAML

Az XAML egy deklaratív [XML](https://hu.wikipedia.org/wiki/XML) alapú leíró nyelv, melyet a [Microsoft](https://hu.wikipedia.org/wiki/Microsoft) fejlesztett ki. Strukturált értékek és objektumok inicializálására használható. A betűszó eredetileg az „Extensible Avalon Markup Language” rövidítése volt.

Az XAML-t jelentős mértékben használják a [.NET keretrendszerben](https://hu.wikipedia.org/wiki/.NET_keretrendszer), ezen belül különösképpen a Windows Presentation Foundationben (WPF), ahol a felhasználói felület jelölőnyelveként szolgál, annak adatelemeit, adatkötéseit, eseményeit és más szolgáltatásokat XAML-lel kell definiálni [15].

* 1. Tesztelés

Egy projekt életében mindig meghatározó elem a tesztelés. Ez nem csak a megfelelő működés érdekében fontos, hanem azért is, mivel rákényszeríti a fejlesztőt, hogy könnyen tesztelhető függvényeket hozzon létre, melyek csak egy feladatot látnak el, ezzel újrafelhasználható és redundancia mentes lesz a kód. A hagyományos megközelítés szerint a tesztelés célja az, hogy a fejlesztés során létrejövő hibákat minél korábban felfedezze, és ezzel csökkentse azok kijavításának költségeit. A tesztelés szintjei a következők lehetnek: egységteszt, integrációs teszt, rendszerteszt és átvételi teszt.

2.6.1. Egységtesztelés

A legalacsonyabb szintű tesztelés. A programot felépítő egységek tesztelése. Unit: egy rendszer legkisebb önálló egységként tesztlehető része. Unit tesztekkel ellenőrizhető, hogy egy unit (egység) az elvárásoknak megfelelően működik. Egy unit függvényeiről ellenőrizzük, hogy különböző bemetek esetén megfelelő eredményt, vagy hibát produkálnak. Az egységeket egymástól izoláltan kell tesztelni.

Előnyei:

* A hibák sokkal korábban észlelhetőek.
* Minden komponens legalább egyszer tesztelt.
* Az egységek elkülönítése miatt a hibák helyének meghatározása könnyű.
* A funkciók könnyen módosíthatóak átalakíthatók.
* Dokumentációs szerep: példákat biztosít egyes funkciók használatára.

Elvek:

* Gyors: A teszteknek gyorsan kell futnia, lassú teszteket senki nem futtatja gyakran, így a hibák nem derülnek ki idejében.
* Független: A teszteknek egymástól függetlennek és bármilyen sorrendben végrehajthatónak kell lennie.
* Megismételhető: A teszteknek bármilyen környzetben, hálozat nélkül is végrehajthatónak kell lennie. A különöbző futtatások eredményének meg kell egyeznie.
* Ön ellenőrző: A tesztek eredménye egy boolean, futásuk vagy sikeres, vagy sikertelen.
* Automatikus: A tesztek automatikusan, interakció nélkül futnak [16].
  1. MySQL

A MySQL a jelenleg legnépszerűbb nyílt forráskódú SQL (Structured Query Language) alapú relációs adatbázis-kezelő rendszer, amelyet az Oracle Corporation fejleszt. A MySQL nagyon gyors, többszálú robosztus adatkiszolgálást biztosít [17].

Az adatbázisok olyan tárolási struktúrák, amelyekből az információ kinyerhető. A MySQL egy relációs típusú adatbázis, amely azt jelenti, hogy a relációs adatmodell egy oly módú leképezése, amely az információkat sorokból és oszlopokból álló táblákban jeleníti meg. Ezeket a táblákat relációknak nevezzük abban az értelemben, hogy azonos típusú (oszlopok) objektumok (sorok) gyűjteménye, a táblákban lévő adatok pedig összekapcsolhatók egymással [18].

* 1. Git

A Git egy nyílt forráskódú elosztott verziókezelő rendszer.

A verziókezelő rendszer, vagy más néven VSC (Version Control System), olyan rendszer, amelynek segítségével nyomon követhetjük fájlok életútját, rögzíti azok változásait, hogy később bármikor előállíthassuk a korábbi verziókat.

Az elosztott verziókezelő rendszer, vagy DVSC (Distributed Version Control System) olyan verziókezelő rendszer, amelyben a kliensek nem csak a fájlok legfrissebb verzióját kapják meg, hanem azok teljes történetét is. Ez a rendszer lehetővé teszi, hogy egy esetleges kiszolgálói hiba esetén a kiszolgáló teljes tartalma visszaállítható legyen, mivel minden egyes klón az összes adat teljes biztonsági másolata [19].

A projekt fejlesztése során számos Git által nyújtott lehetőséggel nem éltem, mivel a projekt fejlesztésében egyedül vettem részt. Ha egy projektet több ember fejleszt, abban az esetben az ágak használata nagyban megkönnyíti a párhuzamos munkát és a kooperációt. Ezenkívül a már élesben futó projektek esetében is célszerű leágazásokat használni [20].

1. A program szerkezeti felépítése

A programkód szerkezetileg hét fő részre, projektre tagolható, amelyek között nem minden esetben lehet éles határvonalat húzni. Alapvetően ezeken a részeken belül is elkülönülnek az egyes funkciócsoportok.

* 1. Adatelérés

Az adatbázis elérésére és az adatbázisban szereplő tárolt eljárások lefuttatására szolgál ez a projekt.

A projektben a *DBHelper* osztály segít a MySQL adatbázis kapcsolat létrehozásában. A projektben interfészeket használtam, melyek leírják az őket implementáló osztályok viselkedését. A *Logic* osztályok fogják az interfészek metódusait felhasználni, melyek az adatbázis tárolt eljárásait futtatják le. Ezen fontosabb oszályok a *UserLogic*, *TaskLogic*, *RecordLogic* és a *NotificationLogic.*

A *UserLogic* osztályban hívódik meg azon metódusok, mint admin regisztrációja, felhasználó létrehozása/olvasása/frissítése/törlése és a belépéshez a validáció.

A *TaskLogic* osztályban azon metódusok szerepelnek, melyek a feladat létrehozásához, különféle olvasásához, frissítéséhez és törléséhez szükségesek.

A *RecordLogic* osztályban olyan tárolt ejárásokat hívó metódusok szerepelnek, melyek a munkavégzés rögzítés létrehozásához, olvasásához, frissítéséhez és törléséhez szükségesek.

A *NotificationLogic* osztály az értesítés létrehozásához, olvasásához és törléséhez szükséges metódusokat tartalmaz.

* 1. Kivételkezelés (Logic)

A projektben a kivétlekezelés folyamata zajlik. Minden modell osztályhoz tartozik egy *Repository* osztály. Ezekben a *Repository* osztályokban történik az előző témában megemlített interfészek metódusainak a kivételkezelése, még pedig a később említésre kerülő validációs osztályok segítségével.

* 1. Model

A projektben az egyes egyedeket modell osztályba sorolom őket, ahol az egyedek tulajdoságait adom meg. A getter és setter metódus implementálásával a tulajdonság olvasható és írható.

A főbb modellek a felhasználó, feladat, rögzítés, értesítés és cég.

A felhasználó modell tartalmazza a következőket:

* IdUser: felhasználónak az azonosítója.
* Username: felhasználónak a felhasználóneve.
* Password: felhasználónak a jelszava.
* FirstName: felhasználónak a keresztneve.
* LastName: felhasználónak a vezetékneve.
* Email: felhasználónak az email-címe.
* Telephone: felhasználónak a telefonszáma.
* Status: felhasználónak a munkahelyi státusza.

A feladat modell tartalmazza a következőket:

* IdTask: feladatnak az azonosítója.
* Title: feladatnak a címe.
* Description: feladatnak a leírása.
* Status: feladatnak a státusza.
* Deadline: feladatnak a határideje.
* User\_IdUser: annak a felhasználónak az azonosítója, akihez tartozni fog a feladat.
* CreationDate: feladat létrejöttének dátuma.

A rögzítés modell tartalmazza a következőket:

* IdRecord: rögzítésnek az azonosítója.
* Date: rögzítésnek a dátuma.
* Comment: rögzítésnek a megjegyzése.
* Duration: rögzítésnek a időtartama, azaz hogy meddig dolgozott a feladattal.
* User\_idUser: annak a felhasználónak az azonosítója, aki a rögzítést készítette.
* Task\_idTask: annak a feladatnak az azonosítója, amihez történt a rögzítés.

Az értesítés modell tartalmazza a következőket:

* IdNotification: értesítésnek az azonosítója.
* Message: értesítésnek az üzenete.
* NotificationFor: az értésítés adminnak vagy dolgozónak szól.
* Task\_IdTask: annak a feladatnak az azonosítója, melyhez az értesítés tartozik.

A cég modell tartalmazza a következőket:

* IdCompany: cégnek az azonosítója.
* CompanyName: cégnek a neve.
  1. Model kiterjesztése

A projektben a felhasználóhoz, feladathoz és rögzítéshez vannak publikus statikus validációs osztályok, mellyek felelnek a mezők kitöltésekor a hibaüzenetek generálásának kivételek (Exceptionok) esetén.

A felhasználói validációs osztály a felhasználónév, jelszó, keresztnév, vezetéknév, email cím és telefonszám ellenőrzésére felel.

A feladat validációs osztály a feladathoz tartozó felhasználónak, feladat címének, határidejének és státuszának ellenőrzésére felel.

A rögzítési validációs osztály a megfelelő feladatnak a kiválasztására, a munkavégzés dátumának és feladattal töltött időtartammának ellenőrzésére felel.

* 1. Teszt

A forráskódot Unit (egység) tesztekkel ellenőrzöm, hogy a felhasználói, feladat és rögzítési validációk elérik-e a kitűzött céljukat.

* 1. Erőforrás

A projektben két Resources fájl van, ahol az egyik tartalmazza a magyar kifejezéseket a másik pedig az angol kifejezéseket a kliensoldalon megjelenő kiírásokhoz, üznetekhez [21]. A projektben még szerepel egy publikus statikus kezelő osztály, ami a két erőforrás fájl kiválasztásában, elérésében segít.

* 1. WpfDemo

Ez a fő projekt, amivel az alkalmazást el lehet indítani. Ebben a rétegbe használtam ki leginkább a NuGet csomagok lehetőségét [10]. A projektet három részre osztottam.

Van egyszer a nézetek, mely az alkalmazás azon rétege, amely a megjelenítésért felel, minden olyan rész, amivel a felhasználó interakcióba léphet. Ez a réteg kommunikál a nézetmodell réteggel.

Nézetmodellek, ahol az osztályok a kliensoldalon használt adatok leírását szolgáljak. Ez a réteg kommunikál a model osztályaival.

Komponensek, melyek a kiegészítő osztályok és a felhasznált képek, ikonok.

# Összefoglalás

Jelen dolgozat keretei között készített programom egy munkaidő nyilvántartás asztali alkalmazás, mellyel a felhasználó a feladataival való foglalkozását rögzíteni tudja. Nyomon tudjuk követni a dolgozó munkaidőben végzett tevékenységeit a feladataival. Lehetőség van a felhasználó munkájának diagramikus vizsgálatára, illetve munkájának exportálására.

A fejlesztés során ügyeltem, hogy a programom esztétikus, könnyen kezelhető, felhasználóbarát legyen. Továbbá az elkészítés során olyan modern technológiákat használtam, melyek nyílt forráskódúak.

1. Továbbfejlesztési lehetőségek

A szoftvert többféleképpen is lehetne továbbfejleszteni.

Elsősorban az adatbázis elérését lehetne módosítani, még pedig a szerver-kliens architektúra alkalmazásával, azaz egy REST APIT-t létrehozni.

Lehetne a felhasználóknak több pozíciót adni, minden pozíciónak meglenne a saját jogosultsága. A feladatokat nem csak személynek adni, hanem egy csoportnak, ekképpen dolgozó csoportok jönnének létre.

Esetleg még felhasználók közötti kommunikációs lehetőséggel is lehetne bővíteni a programot, viszont van épp elég alkalmazás, ahol a kapcsolatot lehet tartani egymással, így ez egyrészt hátrány is lehet, hogy mégegyet figyelni kellene.

A programon belül az alsó részhez létre lehetne hozni státuszmezőt, ami mutatná a visszajelzést, ha létrehozás, módosítás vagy törlés történt.

A programban be lehetne építeni egy stopperórát, hogy amikor elkezdjük a feladatunkat, akkor pontosan mérje nekünk az azzal töltött időnket.

Az angol és magyar nyelven kívül többel kiegészíteni.

1. Forráskód elérhetősége

A forráskód elérhető a GitHub-on: <https://github.com/Marszu99/Szakdoga>

1. Alkalmazás futtatása

A program futtatásához telepítenünk kell a Visual Studio-t melyet a következő weboldalról letölthetünk: <https://visualstudio.microsoft.com/downloads/>, illetve szükségünk van még az adatbázis eléréséhez a MySQL Workbench alkalmazás letöltéséhez, melyet a <https://dev.mysql.com/downloads/workbench/> weboldalról tölthetünk le.

A Visual Studio-ban tudjuk megnyitni a GitHub-ról letöltött forráskódot.

Ahhoz, hogy fusson a program a Workbench-ben létre kell hozni egy szervert, majd egy *schema*-t létrehozni, de akár ki is választhatjuk az alapból létrehozott *mydb* sémát. A kiválasztott sémában pedig le kell futtatni a <https://github.com/Marszu99/Szakdoga>

weboldalon található SzakdogaDB SQL fájlt, mely létrehozza a szükséges táblákat, nézeteket és tárolt ejárásokat.

Végül a programkódban a *WpfDemo* projektben lévő *App.Config* fájlban át kell írni a *connectionString*-nél a *server*-t a szeverünk címére, *uid*-t a szerverünk nevére, *pwd*-t a szerverünk jelszavára és a *database*-t a kiválasztott séma nevére.

# Irodalomjegyzék

1. ,,Microsoft Visual Studio” [Online]. Elérhető: <https://hu.wikipedia.org/wiki/Microsoft_Visual_Studio> [Hozzáférés dátuma: 2022. 04. 10.].
2. ,,MySQL Workbench” [Online]. Elérhető: <https://dev.mysql.com/doc/workbench/en/wb-intro.html> [Hozzáférés dátuma: 2022. 04. 10.].
3. ,,Git and Github essentials for Docs” [Online]. Elérhető: <https://docs.microsoft.com/en-us/contribute/git-github-fundamentals> [Hozzáférés dátuma: 2022. 04. 10.].
4. ,,Working on Git Bash” [Online]. Elérhető: <https://www.geeksforgeeks.org/working-on-git-bash/> [Hozzáférés dátuma: 2022. 04. 10.].
5. „A tour of the C# language” [Online]. Elérhető: [https://docs.microsoft.com/en- us/dotnet/csharp/tour-of-csharp/](https://docs.microsoft.com/en-%20us/dotnet/csharp/tour-of-csharp/) [Hozzáférés dátuma: 2022. 04. 10.].
6. ,,Reiter István: C# programozás lépésről lépésre” [Online]. Elérhető: <https://www.libri.hu/konyv/reiter_istvan.c-programozas-lepesrol-lepesre.html> [Hozzáférés dátuma: 2022. 04. 10.].
7. ,,Életciklus GYIK - .NET Framework” [Online]. Elérhető: <https://docs.microsoft.com/hu-hu/lifecycle/faq/dotnet-framework> [Hozzáférés dátuma: 2022. 04. 10.].
8. ,,Bakonyi Viktória: C#: egy nyelv ezernyi lehetőség - ASP.NET Core” [Online]. Elérhető: <http://tehetseg.inf.elte.hu/jegyzetek/ASPCore_vegleges.pdf> [Hozzáférés dátuma: 2022. 04. 10.].
9. ,,Életciklus GYIK - .NET Core” [Online]. Elérhető: <https://docs.microsoft.com/hu-hu/lifecycle/faq/dotnet-core> [Hozzáférés dátuma: 2022. 04. 10.].
10. ,,An introduction to NuGet” [Online]. Elérhető: <https://docs.microsoft.com/en-us/nuget/what-is-nuget> [Hozzáférés dátuma: 2022. 04. 10.].
11. ,,Introduction to WPF [Online]. Elérhető: <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/introduction-to-wpf> [Hozzáférés dátuma: 2022. 04. 10.].
12. ,,Windows Presentation Foundation” [Online]. Elérhető: <https://hu.wikipedia.org/wiki/Windows_Presentation_Foundation> [Hozzáférés dátum: 2022. 04. 10.].
13. ,,Patterns - WPF Apps With The Model-View-ViewModel Design Pattern” [Online]. Elérhető: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern> [Hozzáférés dátum: 2022. 04. 10.].
14. ,,Modell-nézet-nézetmodell” [Online]. Elérhető: <https://hu.wikipedia.org/wiki/Modell-nézet-nézetmodell> [Hozzáférés dátum: 2022. 04. 10.].
15. ,,XAML overview (WPF .NET) [Online]. Elérhető: [https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/](https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0) [Hozzáférés dátum: 2022. 04. 10.].
16. ,,Unit test” [Online]. Elérhető: <https://swap.web.elte.hu/2018191_pt2e/ea08.pdf> [Hozzáférés dátuma: 2022. 04. 10.].
17. „MySQL 8.0 Reference Manual - General Information” [Online]. Elérhető: <https://dev.mysql.com/doc/refman/8.0/en/introduction.html> [Hozzáférés dátuma: 2022. 04. 10.].
18. „A Relational Database Overview” [Online]. Elérhető: <https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html> [Hozzáférés dátuma: 2022. 04. 10.].
19. ,,About Git” [Online]. Elérhető: <https://docs.github.com/en/github-ae@latest/get-started/using-git/about-git> [Hozzáférés dátuma: 2022. 04. 10.].
20. B. S. Scott Chacon, Pro Git, Berkeley, CA: Apress, 2014. [Online]. Elérhető: <https://doi.org/20.500.12657/28155> [Hozzáférés dátuma: 2022. 04. 10.].
21. ,,Resources (WPF)” [Online]. Elérhető: <https://docs.microsoft.com/hu-hu/dotnet/desktop/wpf/advanced/resources-wpf?view=netframeworkdesktop-4.8> [Hozzáférés dátuma: 2022. 04. 10.].

# Ábrajegyzék

[1. ábra – Bejelentkezési felület 5](#_Toc102142520)

[2. ábra – Főoldal lenyíló menü 6](#_Toc102142521)

[3. ábra – Profil: felhasználói adataim 6](#_Toc102142522)

[4. ábra – Profil: adataim, ha megakarnám változtatni 6](#_Toc102142523)

[5. ábra – Profil felület 7](#_Toc102142524)

[6. ábra – Felhasználók felület: lista szűrés után 7](#_Toc102142525)

[7. ábra – Felhasználók felület: felhasználóra kattintva (megjelenik jobb oldalt a felhasználó adatai) 8](#_Toc102142526)

[8. ábra – Felhasználói profil: kördiagram 9](#_Toc102142527)

[9. ábra – Felhasználói profil felület 10](#_Toc102142528)

[10. ábra – Feladatok felület: feladatra kattintva lenyíló menü (admin esetén) 11](#_Toc102142529)

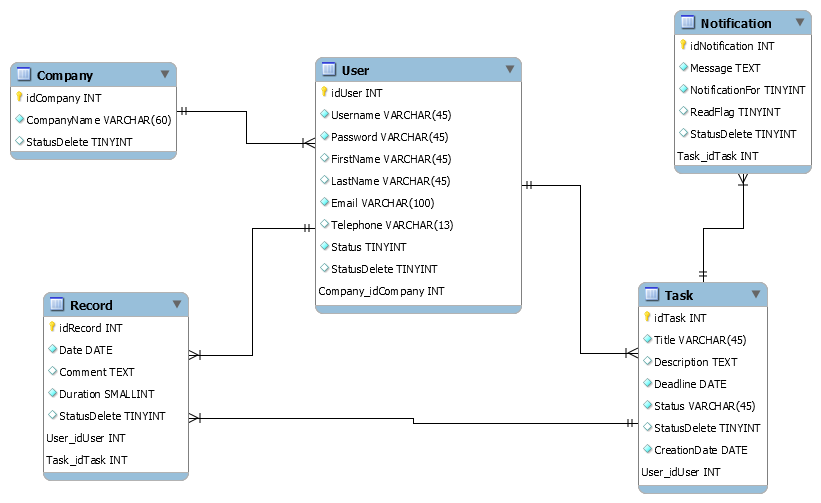
[11. ábra – Feladatok felület: feladatra kattintva (megjelenik jobb oldalt a feladat) 11](#_Toc102142530)

[12. ábra – Rögzítések felület 12](#_Toc102142531)

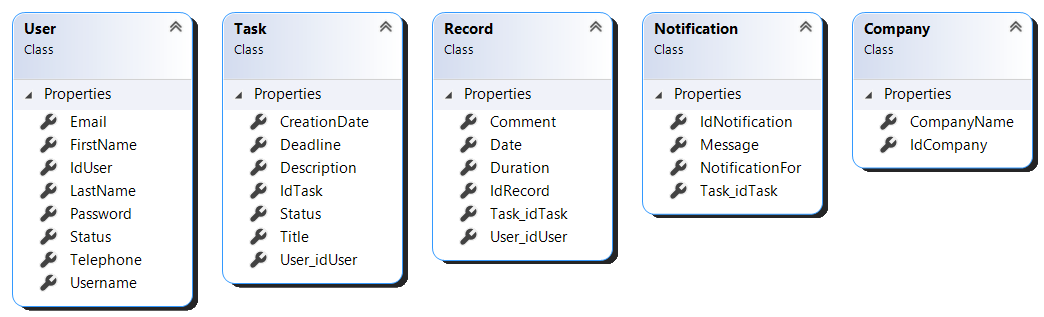
[13. ábra - MVVM 18](#_Toc102142532)

# VIII. Mellékletek

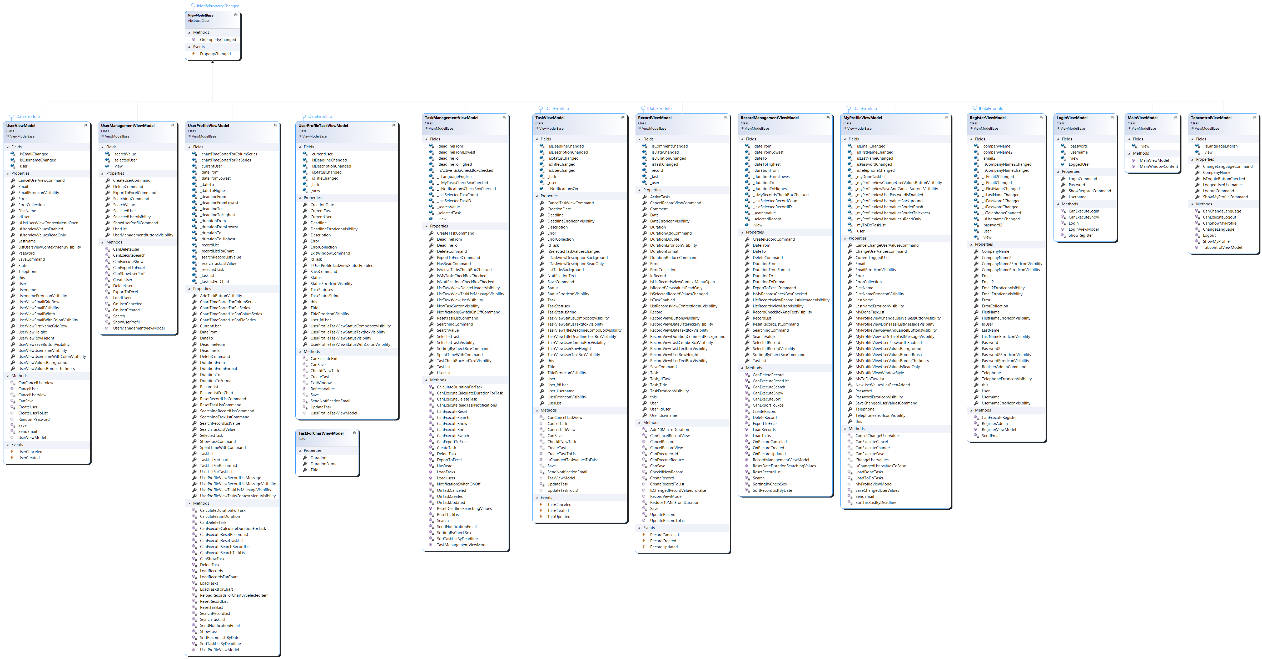
1. Relációs diagram



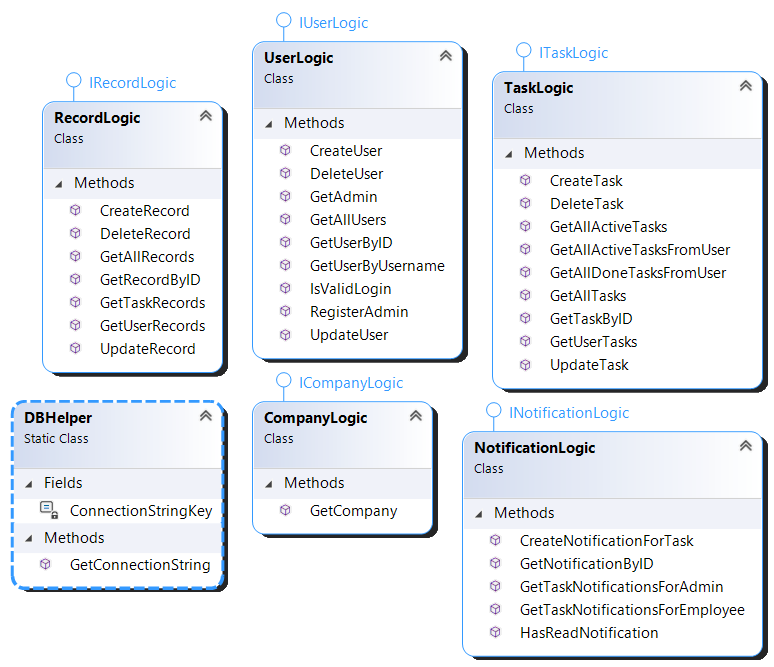
2. Osztálydiagramok

**

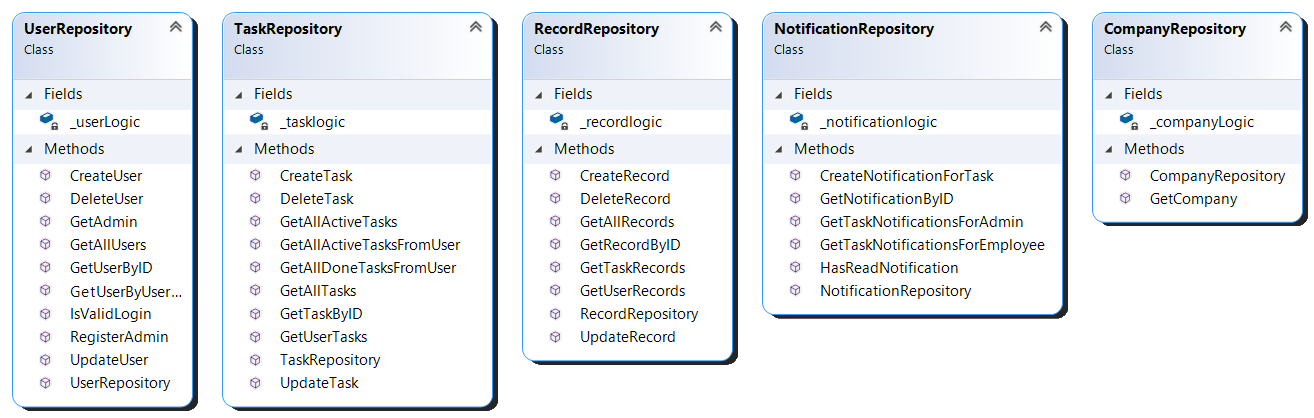
*Modellek osztálydiagramja*

**

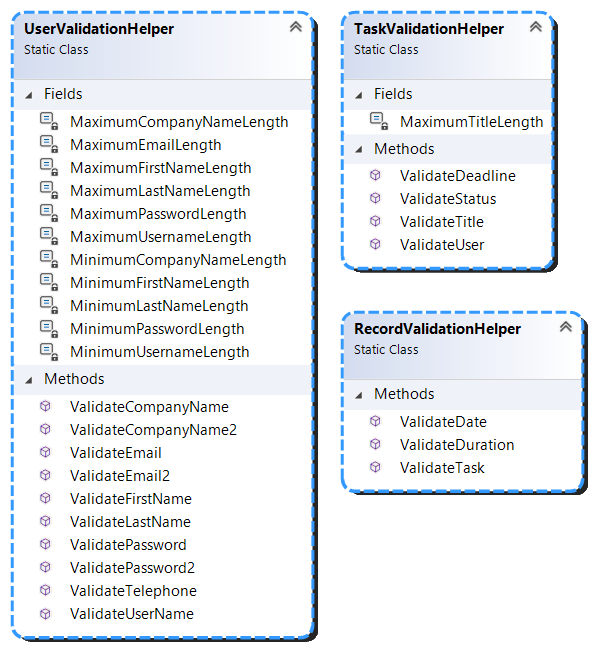
*ViewModellek osztálydiagramja*



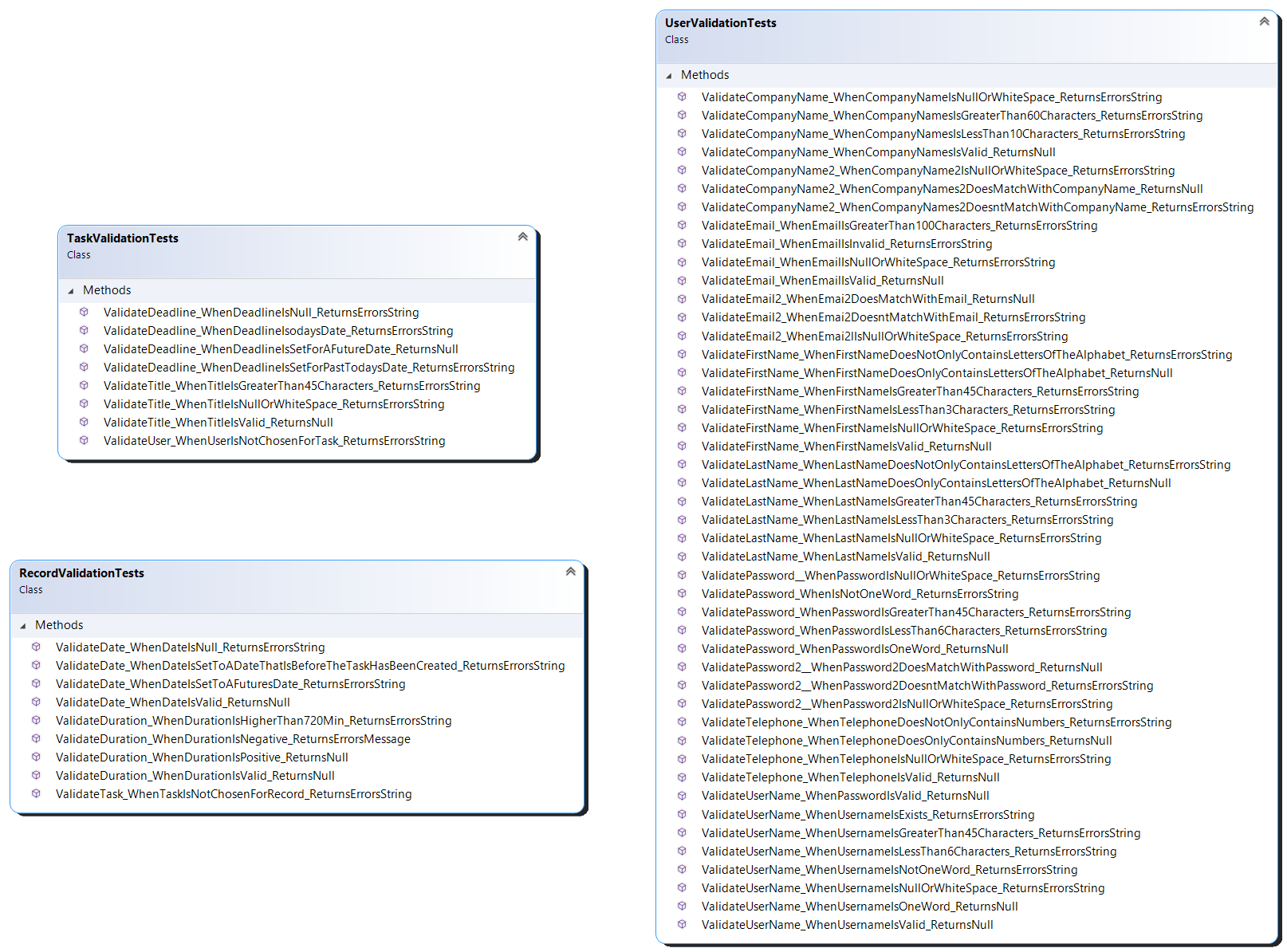
*Logicok osztálydiagramja*



*Repositoryk osztálydiagramja*



*Validationok osztálydiagramja*

**

*Tesztek osztálydiagramja*

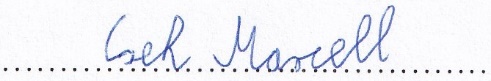
# NYILATKOZAT

az írásmű eredetiségéről

*(PTE SZMSZ 5. sz. mellékletének 14/1. számú melléklete alapján)*

Alulírott Cseh Marcell (FLNYRL), büntetőjogi felelősségem tudatában kijelentem, hogy Alkalmazásfejlesztés releváns problémákhoz című írásomban foglaltak saját, önálló munkám eredményei, ennek elkészítéséhez kizárólag a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel, írásomat a Pécsi Tudományegyetem vonatkozó szabályzatainak betartásával készítettem. Tudomásul veszem, hogy a szerzői jogi szabályok betartását a Pécsi Tudományegyetem plágiumkereső rendszeren keresztül ellenőrizheti.

Pécs, 2022. 04. 29.



hallgató aláírása