Vas Megyei Szakképzési Centrum

Nádasdy Tamás Technikum és Kollégium

Dolgozat Írató Alkalmazás (DIA)

Domnánovich Bálint

Konzulens:
Varga Gábor

2022

Nyilatkozat

Alulírott, Domnánovich Bálint szakos hallgató kijelentem, hogy a Dolgozat Írató Alkalmazás, továbbiakban DIA című szakdolgozat feladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Kőszeg, 2022.04.01 _____

hallgató

Kivonat

Dolgozat Írató Alkalmazás (DIA)

A záródolgozatomban egy olyan szoftvert mutatok be, ami megkönnyíti a tanárok munkáját és próbálja motiválni a diákokat a tanulásra. A DIA tanároknak egy kiváló platform, ahol számon kérhetik a tanulók tudását, a diákoknak pedig egy nagyszerű lehetőség arra, hogy kifizetődjön a tanulásuk.

A tanároknak egy egyszerű felületet biztosít arra, hogy felmérjék a diákok tudását. Segít nekik a javításban és biztosít egy olyan lehetőséget, hogy tiltsák az internet használatát dolgozatírás közben!

Ezenfelül a diákoknak is számos előnyük származik a program használatából. A feladatlapok könnyen értelmezhetőek és kitölthetőek, javításuk után pedig egyből megtekinthetőek. A pontok elköltéséből pedig akár a tanulmányi eredményeiket is javíthatják.

Abstract

Test Writer Application (DIA)

In my final thesis, I present a software that facilitates the work of teachers and tries to motivate students to learn. DIA is an excellent platform for teachers to hold students accountable for their knowledge, and a great opportunity for students to gain advantages for their learning.

It provides the teachers a simple interface to assess students' knowledge. It helps them in fixing and provides an opportunity to ban the use of the Internet while writing a paper!

In addition, students benefit from the use of the program. Task sheets are easy to interpret and fill in, and can be viewed immediately after they are corrected. They can also improve their academic achievements by spending points.

Tartalomjegyzék

1. Bevezetés

A 2022-es záróvizsgám beadandójához témának egy olyan alkalmazást választottam, amiben a tanárok teszteket irattathatnak a diákokkal. Fő célja viszont nem csak az, hogy a diákokat osztályozzák, hanem a diákok tudásának felmérése mellett jutalmakkal ösztönözze őket a tanulásra. A dolgozatok teszt alapúak lehetnek és ezek megírásával pontokat szerezhetnek a diákok. Ezek a pontok a “boltban” beválthatóak különböző jutalmakra.

A választásom azért esett erre a projektre mert nem web alapú dolgozatírásra lehetőséget adó szoftverrel még nem találkoztam. A webesek fő problémája a több oldal megnyitásával való csalás lehetősége. Ezt a problémát a DIA-val megoldottam azzal, hogy a tanárnak van lehetősége engedélyezni vagy tiltani a böngészők használatát dolgozatírás közben.

Tanárként egyszerűen listázhatjuk az osztályokat, ahol tanítanak és az azon belüli összes diákot. Létrehozhatunk teszteket, amelyeket osztályonként és azon belül tanított tantárgyra szűkíteni tudunk. A tesztek teljesen személyre szabhatóak, bármilyen és bármennyi kérdés akármennyi válasszal általunk meghatározott pontszámmal hozhatók létre. Ezek a kérdések és válaszok pedig teljesen összekeverve kerülnek a diákokhoz. Majd később lehetőségünk lesz arra, hogy értékeljük a leadott dolgozatokat.

Diákként ezeket a teszteket kitölthetjük, majd a javítást követően megnézhetjük. Az értékelésük pontban történik, ami hozzáadódik a tanuló pénztárcájához. Ezeket a pontokat a boltban költhetik el különféle jutalmakra, ezzel motiválva őket a tanulásra.

Régebben programozás órán a tanárunk készített egy tesztelő programot, ami nagyon tetszett. Ez is motivált arra, hogy ezt a válasszam a záró dolgozatom témájának.

A szoftver fejlesztéséhez a C#-ban szerzett programozási ismereteimet és a mySQL adatbázis-kezelési készségeimet használtam fel.

- ➔ Az alkalmazáshoz Visual Studio 2022 programozási környezetet használtam C# nyelven, WinForms applikáció-ként.
- ➔ Laragont a lokális környezet kialakításához.
- ➔ Az adatbázis tervezéséhez és elkészítéséhez pedig Laragonon belüli HeidiSQL segített.
- ➔ Az adatok tárolása MySQL szerverben történik.
- ➔ A programban használt képeket, illetve ikonokat az irodalomjegyzékben jelölt oldalról és az Adobe Photoshop 2021 program segítségével készítettem.

2. Hasonló alkalmazások

Jelenleg nagyon kevés ilyen szoftver található az interneten, pedig az online oktatás megmutatta, hogy igen is szükség van egy ilyen alkalmazásra.

A Google űrlapok az az alkalmazás, ami talán hasonló. Teszteket hozhatunk benne létre és segít a javításban. Azonban az űrlapok nem az iskoláknak készült elsősorban. Egyszerűbb tesztekhez és választások lebonyolítására tökéletes, viszont csalás ellen nem próbál meg védekezni és nem lehet benne diákokat és tanárokat létrehozni vagy kezelni.

Egy olyan alkalmazás, amiben a tanárok láthatják a tanulókat és könnyedén irathatnak velük dolgozatokat nagyban segítheti az ő munkájukat, ez a legfőbb indokom a téma választásához.

3. Felhasználói dokumentáció

A program fő célja, hogy segítsen a tanároknak számonkérni a diákokat biztonságos módon és próbálja motiválni a tanulókat a tanulásra.

Lehetőségeink tanárként:

- ➔ Osztályok és diákok listázása, diákok vásárlásainak kezelése (beváltása)
- ➔ Saját profil szerkesztése
- ➔ Dolgozatok létrehozása, kijavítása, visszakövetése

Lehetőségeink diákként:

- ➔ Dolgozatok írása és megtekintése
- ➔ Saját profil szerkesztése
- ➔ Pontok elköltése a boltban

Lehetőségeink adminként:

- ➔ Tanárok hozzáadása, szerkesztése, pozíciójának kezelése
- ➔ Diákok hozzáadása, szerkesztése, vásárlásaiknak kezelése
- ➔ Osztályok, tantárgyak létrehozása, szerkesztése

3.1. Hardware és Software igény

Az applikáció futtatásához (ha az adatbázis nem lokálisan fut) hálózati kapcsolat szükséges. Továbbá a következő specifikáció ajánlott:

- ➔ Legalább 1 GHz-es processzor
- ➔ 1 GB szabad memória
- ➔ Windows 7 vagy újabb operációs rendszer
- ➔ Microsoft .Net 5 vagy újabb
- ➔ 100 MB szabad tárhely

3.2. A program telepítése, elérése

A szoftver előre telepítve megtalálható a meghajtón és a mappán belüli DIA_Project.exe file-al futtatható, amennyiben az SQL szerver már fut!

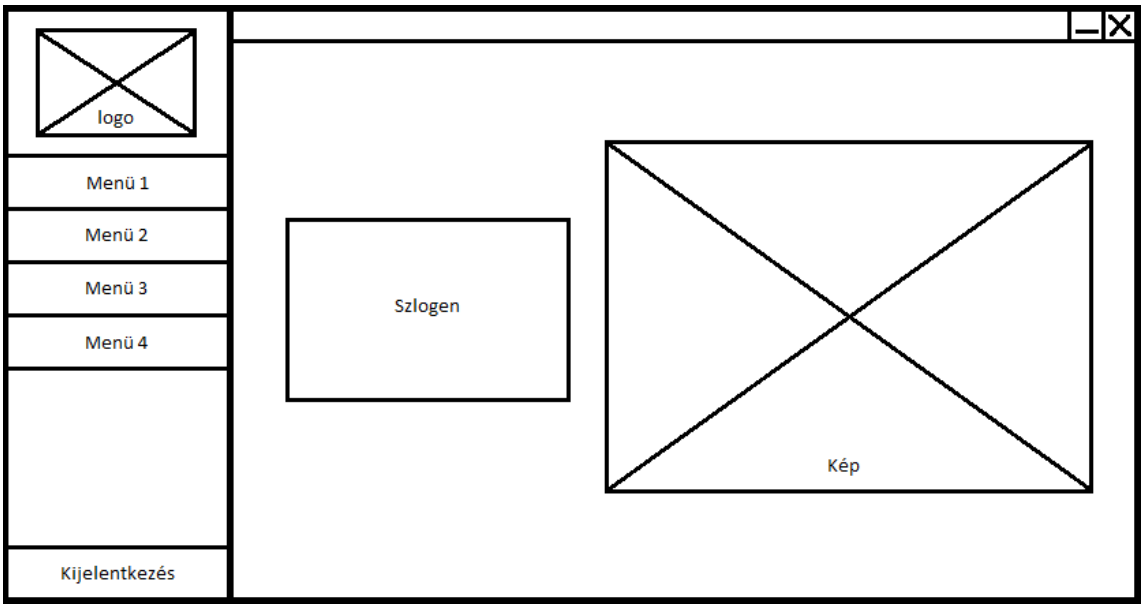
3.3. Belépés és regisztráció

Az alkalmazásba előre létrehozott profil vagy regisztráció nélkül nem léphető be! Fiók létrehozására viszont csak az admin jogosult.
A regisztrációhoz szükségünk van egy 6 karakterből álló kódra (“-” jellel elválasztva és anélkül is működik) amelyet az admin hozhat létre. A karakterek birtokában egy teljes név, felhasználónév, jelszó és e-mail cím megadása után regisztrálhatunk.

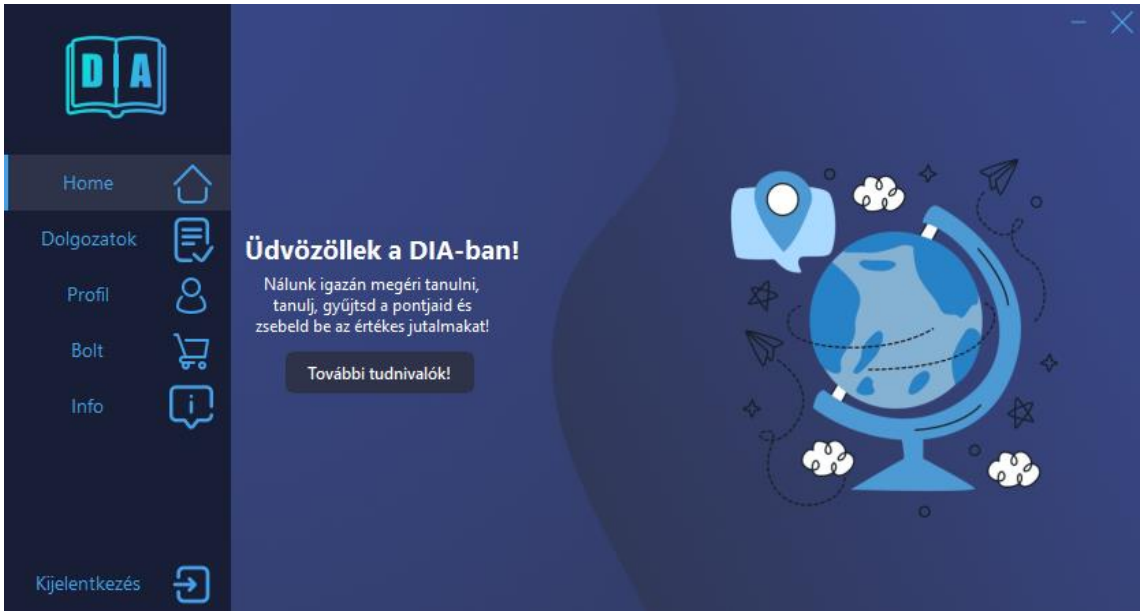
Belépéshez a megfelelő felhasználónév és jelszó párost kell beírni, majd a “Belépés” -re kattintva vagy az “Enter” -lenyomásával megjelenik a főoldal.

3.4. Főoldal és tartalma

A főoldal minden felhasználó típusnál megegyezik. Bal oldalt a logó és a navigációs menüsor, felül egy az ablak mozgatására képes keskeny panel, kilépés és tálca gombok, középen pedig a tartalom, ami változik menüpont-tól függően. A főoldal esetében ez egy szlogennel ellátott kép.



1. ábra a főoldal design tervéről



2. ábra a végleges főoldalról

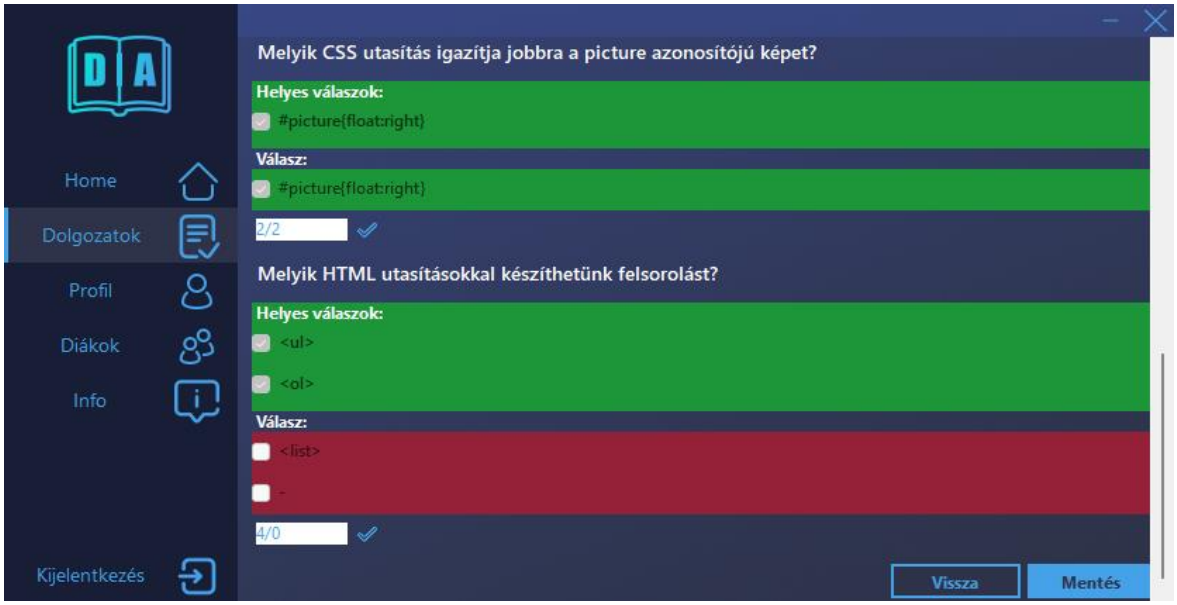
3.5. Funkciók ismertetése

3.5.1. Dolgozatok létrehozása

Csak a tanár fiókok tudnak létrehozni teszteket. A “Dolgozatok” menüpont alatt elérhetik az összes általuk kiadott dolgozatot visszamenőleg is. Itt hozhat létre dolgozatot.
Az “új” gombra kattintva elkészítheti a dolgozatát, ahol meg kell adnia melyik osztálynak milyen tantárgyból és milyen határidőkkel szeretne dolgozatot íratni. A “hozzáadás” gombbal adhat kérdéseket és válaszokat is hozzá, az “X” gombbal pedig visszavonhatja őket. Itt engedélyezheti vagy tilthatja az internet használatát is. A “mentés” gombbal adhatja ki a feladatsort a diákoknak.

3.5.2. Dolgozatok javítása

Szintén a Dolgozatok menüpont alatt a “Megnyitás” gombbal tudja megnézni, hogy kik adták le a dolgozatokat a tanár. A késve leadott teszteket is jelzi a program. A már leadott feladatsorokat tudja kijavítani a tanár. Minden kérdést automatikusan kijavítja az alkalmazás és felajánl egy pontozást a tanárnak, aki a pipára kattintva fogadhatja azt el, de így nyújt lehetőséget a felülírásra is.

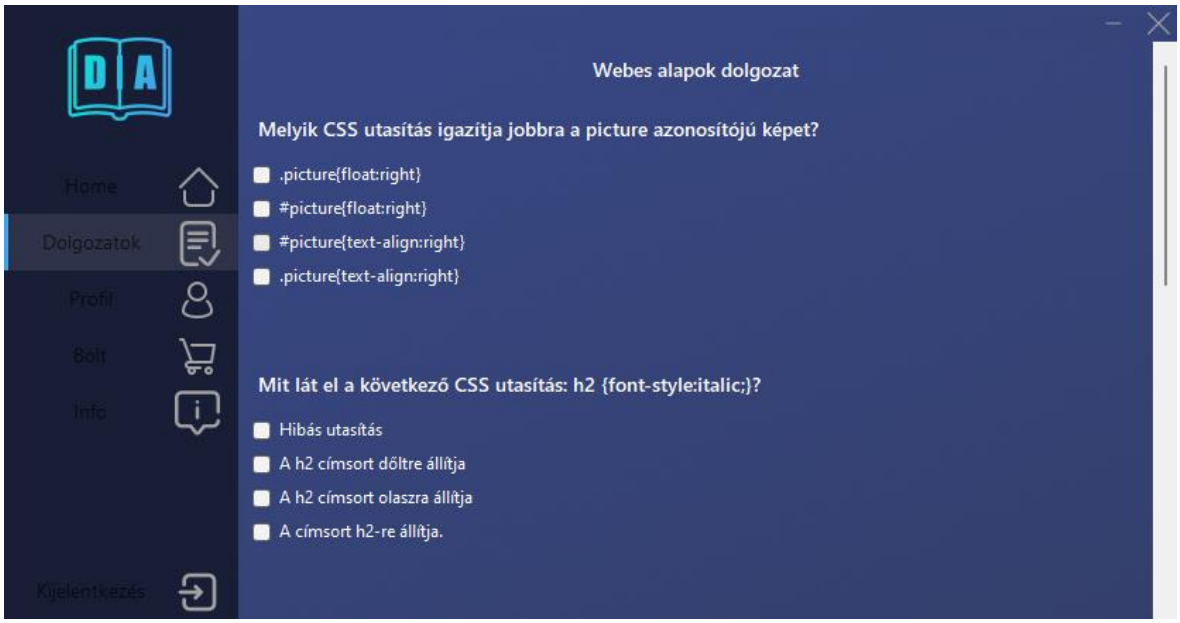


3. ábra a dolgozat javításáról

3.5.3. A feladatsorok kitöltése

A diákok a dolgozatok menüpont alatt találhatják meg a már megírt és újonnan nekik kiadott teszteket. Lehetőségük van szűkíteni tantárgya és megjeleníteni, illetve elrejteni azokat a teszteket, amelyek már kijavítottak vagy javítás alatt vannak.

Egy dolgozatra kétszer kattintva vagy kijelölve és a megnyitás gombra menve az új dolgozatokat kitölthetik a már megírtakat megtekinthetik. Dolgozatírás előtt egy figyelmeztető ablak jelenik meg ahol az igenre kattintva már tölthetjük is ki a feladatsort. Amennyiben a dolgozat megírásához nincs engedélyezve az internethasználat, akkor erre is figyelmeztet. Ha mégis elindítunk egy böngészőt (Opera, Chrome, Edge, Firefox) akkor az a dolgozat akkori állapotában történő leadását eredményezi.



4. ábra a teszt kitöltéséről

3.5.4. Pontok elköltése

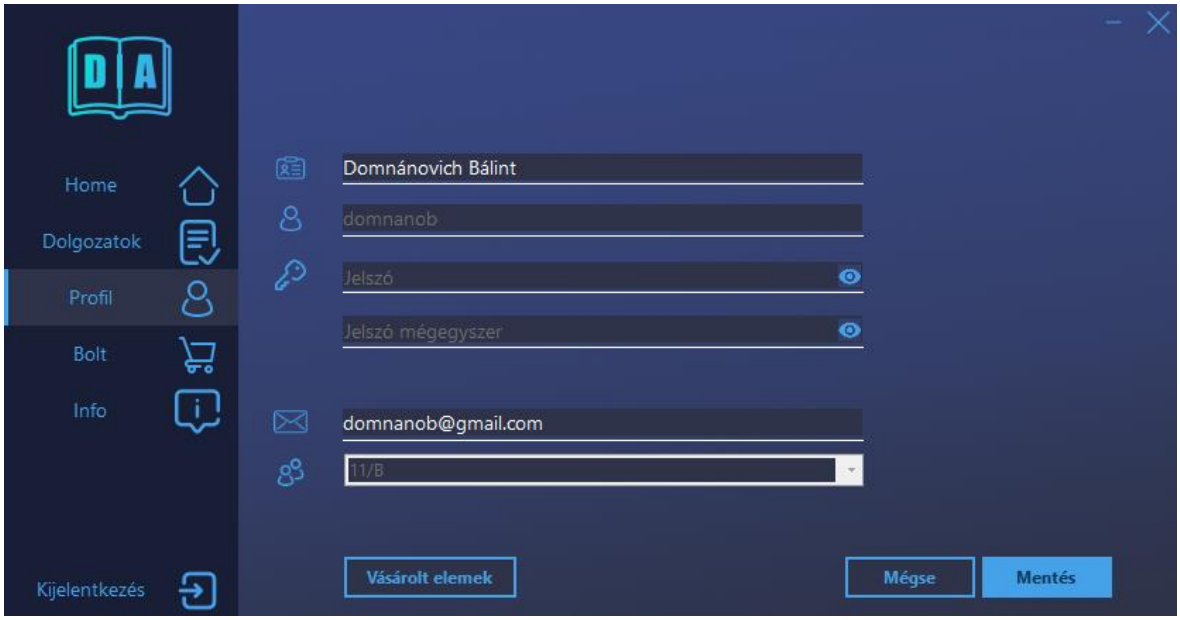
Minden teszt az eredményétől függően pontokat ad a diákoknak. Ezeket a “Bolt” menüpont alatt költheti el. Mindegyik lehetőségből egyszerre csak 1 lehet egy diáknak, ha elhasználta akkor vehet újra abból a típusú elemből.

3.5.5. Pontok beváltása

Mivel a vásárolt elemek olyan segítségeket nyújtanak a diákoknak, amelyek a programon belül nem érvényesíthetőek ezek a tanár feladata. Ilyen például egy jeles érdemjegy, amit a naplóban kell rögzítenie. A vásárolt elemet ezután a “Diákok menüpontban” törölheti a tanár. Itt az összes általa tanított diák megjelenik. Lehetősége van szűkíteni osztályra vagy keresni névre is.

3.5.6. Profil szerkesztése

Mindegyik fióknak lehetősége van arra, hogy szerkessze az adatait. Ilyen adatok a név az email cím és a jelszó. A felhasználónév nem módosítható mivel ez az elsődleges azonosítója minden felhasználónak.



5. ábra a profil menüről

3.5.7. Gyakran ismételt kérdések (FAQ)

Minden felhasználónak van egy “Info” menüje, ahol megtalálhat néhány gyakori kérdést a programmal kapcsolatban. A kérdések listája később természetesen bővíthető.

3.5.8. Admin lehetőségek

Az admin-nak lehetősége van listázni, módosítani, hozzáadni, illetve törölni adatokat. A “Diákok” menüben láthatja az összes diákot. Szűkítheti őket osztályokra és kereshet is közöttük. A szerkesztés gombbal módosíthatja adataikat, kivéve a felhasználóneveket az adatintegritás megőrzése végett. Lehetőség van a deaktiválásra is, ami után az a felhasználó nem tud bejelentkezni. Továbbá a vásárlásaik is módosíthatóak itt. Az új gombbal létrehozhatóak diákok. Az adatok kitöltésével és a mentés gombbal készíthetünk manuálisan, de akár a regisztráció kód segítségével generálhatunk egy 6 karakterből álló kódot, amivel a tanulók is regisztrálhatnak. Ezek a kódok megjegyzik a hozzájuk tartozó osztályt is és automatikusan beosztja a megfelelőbe a diákokat. Valamint ezek a regisztrációs “zsetonok” később törölhetőek.

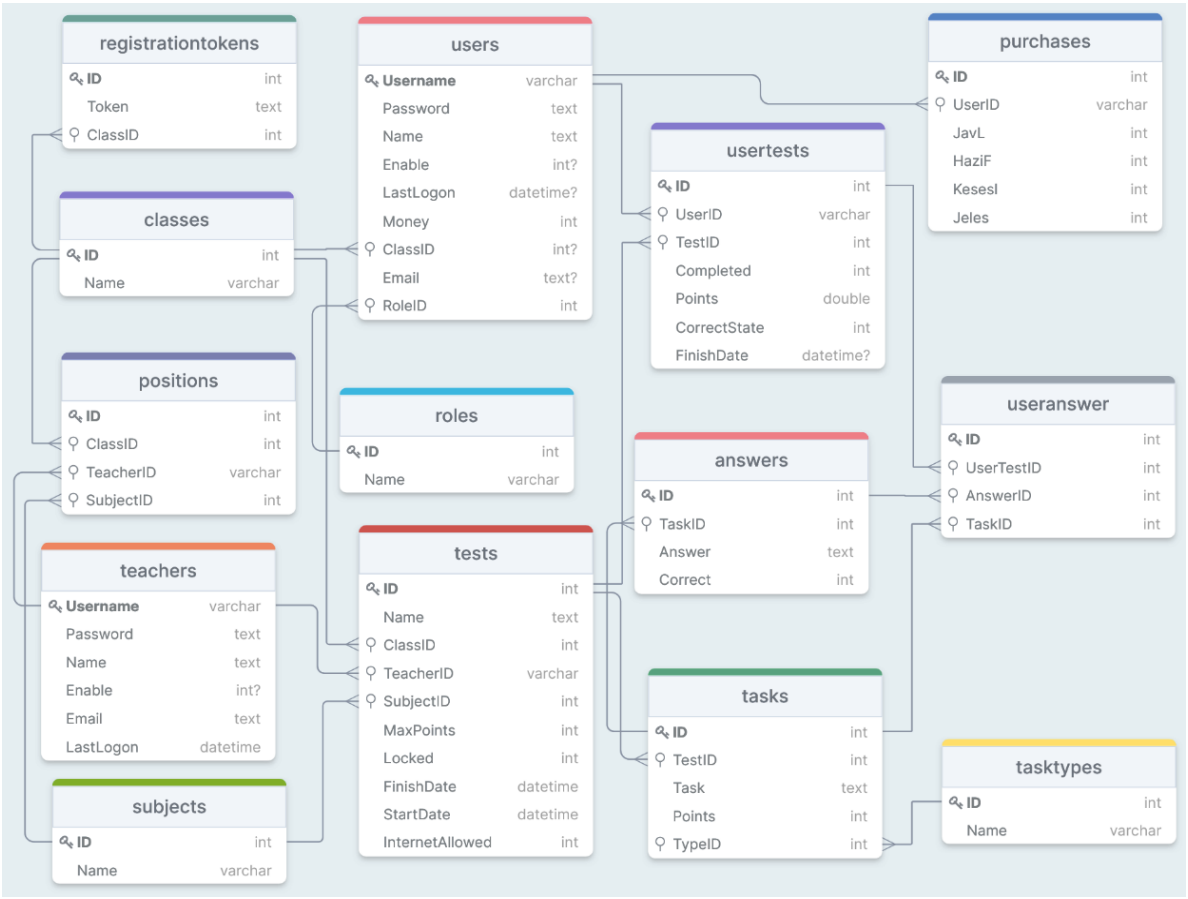
A “Tanárok” menüben eléri az összes tanári felhasználót. Az adataik ugyanúgy módosíthatóak mint a diákoknak. Nekik viszont a pozíciójuk is megadható. Egy tanár akár több osztályban és egy osztályon belül több tantárgyból is taníthat. Ezek a pozíciók akár törölhetőek is.

A tantárgyakat is tudja kezelni. A következő menüpontban megjelenik az összes tantárgy. Létrehozható új tárgy és neveik is módosíthatóak azonban az adat összetartozás nem engedi a törlésüket.

Az osztályok is hasonlóképp működnek, azonban itt a listában megjelenik az adott osztályba járó diákok száma is.

4. Fejlesztői dokumentáció

4.1. Az adatbázis táblái és kapcsolatai



6. ábra az adatbázis struktúrájáról

Az alkalmazáshoz tartozó adatok tárolására dia_db nevű adatbázis nyújt lehetőséget. Az adatbázisnak a lokális szerveren kell futnia jelszó nélkül. Jelenleg az adatbázis 14 táblából és közöttük 18 kapcsolatból áll. Továbbá alapból tartalmaz adatokat, amelyek szükségesek a működéséhez teszteléséhez.

4.1.1. users tábla

➔ A felhasználók tárolására szolgáló tábla.

Mezőnév	Típus	Megjegyzés
Username	VARCHAR, PK, NN	Felhasználónév
Password	TEXT, NN	Jelszó
Name	TEXT, NN	Teljes név
Enable	INT, NN, def.:1	Státusz (logikai)
LastLogon	DATETIME	Utolsó belépés
Money	INT, NN, def.:0	Pontjainak száma
ClassID	INT, FK, NN	Osztályának azonosítója
Email	TEXT	Email cím
RoleID	INT, FK, NN, def.:0	Jogosultságának azonosítója

4.1.2. teachers tábla

→ A tanárok adatai találhatóak itt.

Mezőnév	Típus	Megjegyzés
Username	VARCHAR, PK, NN	Felhasználónév
Password	TEXT, NN	Jelszó
Name	TEXT, NN	Teljes név
Enable	INT, NN, def.:1	Státusz (logikai)
Email	TEXT, NN	Email cím
LastLogon	DATETIME	Utolsó belépés

4.1.3. roles tábla

➔ Ez a tábla segít elkülöníteni a jogosultságokat a felhasználók között. Jelenleg ez az admin és a diák, de megléte továbbfejlesztési lehetőségre szolgál.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	VARCHAR, NN	Jogosultság Megnevezése

4.1.4. classes tábla

➔ Segédtábla, itt tárolja a program az osztályokat

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	VARCHAR, NN	Osztály megnevezése

4.1.5. subjects tábla

➔ Segédtábla, itt találhatóak meg a tantárgyak

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	VARCHAR, NN	Tantárgy megnevezése

4.1.6. purchases tábla

➔ A felhasználók vásárlásai itt tárolódnak.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
UserID	VARCHAR, FK, NN	A felhasználó azonosítója
JavL	INT, NN, def:0	Vásárolható elemek (logikai)
HaziF	INT, NN, def:0	
KesesI	INT, NN, def:0	
Jeles	INT, NN, def:0	

4.1.7. positions tábla

➔ A tanárok munkakörei vannak itt rögzítve.

➔ Több a többhöz kapcsolat érvényesül, mivel egy tanár taníthat több tantárgyból több osztályt is.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
TeacherID	VARCHAR, FK, NN	Tanár azonosítója
ClassID	INT, FK, NN	Osztály azonosítója
SubjectID	INT, FK, NN	Tantárgy azonosítója

4.1.8. registrationTokens tábla

➔ Tárolja a regisztrációhoz szükséges 6+1 (“-” jel) karakterből álló kódot.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Token	TEXT, NN	Regisztrációs kód
ClassID	INT, FK, NN	Osztály azonosítója

4.1.9. tests tábla

➔ A kiadott dolgozatok mentésére szolgáló tábla.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	TEXT, NN	A dolgozat címe
TeacherID	VARCHAR, FK, NN	Tanár azonosítója
SubjectID	INT, FK, NN	Tantárgy azonosítója
ClassID	INT, FK, NN	Osztály azonosítója
MaxPoints	INT, NN	Elérhető pontok száma
Locked	INT, NN, def.:0	Dolgozat állapota
FinishDate	DATETIME, NN	Dolgozat kezdetének időpontja
StartDate	DATETIME, NN	Dolgozat kezdetének időpontja
InternetAllowed	INT, NN, def.:0	Internet engedélyezése (logikai)

4.1.10. tasks tábla

➔ Minden dolgozatnak vannak kérdései, ezek tárolását teszi lehetővé ez a tábla.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
TestID	INT, FK, NN	Teszt azonosítója
Task	TEXT, NN	Maga a kérdés
Points	INT, NN	A kérdés pontértéke
TypeID	INT, FK, NN	Típusának azonosítója

4.1.11. taskTypes tábla

➔ Segédtábla a kérdéstípusok tárolására.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	VARCHAR, NN	Kérdés típusának megnevezése

4.1.12. answers tábla

➔ A kérdésekhez tartozó válaszlehetőségeket tárolja.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
TaskID	INT, FK, NN	Kérdés azonosítója
Answer	TEXT, NN	Maga a válasz
Correct	INT, NN, def.:0	Helyes-e (logikai)

4.1.13. userTest tábla

➔ Ez a tábla kapcsolja össze a dolgozatokat a diákokkal, hogy kinek melyikeket kell megírnia, illetve tárolja megírás után az eredményt.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
UserID	VARCHAR, FK, NN	Felhasználó azonosítója
TestID	INT, FK, NN	Teszt azonosítója
Completed	INT, NN, def.: 0	Leadási állapota (logikai)
Points	INT, NN	Elért pontszám
CorrectState	INT, NN, def.: 0	Javítási állapota (logikai)
FinishDate	DATETIME	Leadási idő

4.1.14. userAnswers tábla

➔ Ebben a táblában a diákok válaszai kerülnek mentésre.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
UserTestID	INT, FK, NN	Felhasználó dolgozatának azonosítója
AnswerID	INT, FK, NN	A válasz azonosítója
TaskID	INT, FK, NN	Kérdés azonosítója

4.2. Formok működése

Az alkalmazás rugalmas és újra töltés mentes működéséért szülő, illetve gyermek Formokat hoztam létre. A szülő (Parent) Formok adják a keretét a programnak, ezek nem záródnak be csak kilépéskor és kijelentkezéskor. Ezekbe a Formok belsejébe töltődnek be és cserélődnek a gyermek (Child) Formok.

4.3. Metódusok, függvények és események

A már említett Formok működését és betöltését a *OpenChildForm(Form childForm)* metódus teszi lehetővé. Bezárja a jelenlegi gyermekosztályt és létrehoz egy megadott újat, amit betölt a szülőosztály belső paneljébe.

A *LoadingDataSources()* és a *Reload()* metódusok felelősek az adatok betöltésére. Ezek általában az adatbázisból kérik le az adatokat majd töltik bele listákba, Labelek-be, DataGridek-be vagy ComboBoxok-ba.

A *BorderRadius.cs* tartalmazza a szükséges forrásokat és függvényeket ahhoz, hogy kerekíteni lehessen a panelek, gombok sarkait.

A *FrameMover.cs*-ben megtalálhatóak a nélkülözhetetlen importokat és függvényt a keret nélküli ablak mozdításához.

A *SecurePasswordHasher.cs* segítségével lehet a jelszavakat hash-elni. Jelenleg a 20 karakteres hash-t egy 16 karakteres “salt” -al támogatja a feltörések ellen. Továbbá lehetőséget ad arra, hogy összehasonlítsunk jelszavakat is.

A *CheatDetector.cs* egy olyan osztály, ami segít megállítani a csalókat dolgozatírás közben. A *DetectBrowser()* függvénye megnézi, hogy a fut-e böngésző a háttérben. Ezt az eljárást használja fel az *_Anticheat()* esemény, amely egy külön szálon hívja meg az ellenőrzést. Ez az esemény eredményét pedig egy időzítő figyeli 1000ms-os frissítéssel. Az időzítő létjogosultságát a szálak összekeverésének elkerülése adja.

A *NativeWinAPI.cs* néhány paramétert és importot tartalmaz, amelyekkel próbálja az alkalmazás teljesítményét javítani.

4.4. Osztályok és Modellek

A modellek felelősek az adatbázis adatainak tárolására a programon belül. Minden táblának megvan a megfelelője modellként és pontosan ugyanazokat az adattípusokat tartalmazzák, mint az adatbázisban. Ezeket az adatokat azonban kilistázni a felhasználónak nem célszerű mivel tartalmazhat olyan adatokat, amiket tilos látniuk, vagy kulcsokat, ezért készültek fordító osztályok. Ezek az osztályok csak azokat az adatokat jelenítik meg egy bizonyos táblából, amelyeket szeretném és elvégzi a kapcsolatokat is a táblák között. Mindezt a konstruktor használatával értem el. Bemeneti paraméterben megkapja a nyers modellt és átalakítja kilistázható formára.

4.5. User Control-ok

A User Control-ok olyan elemek, amelyek több különálló elemeket képesek tárolni egyben. Ezek tették lehetővé a dolgozatok felépítését. Minden dolgozat User Control-okból épül fel. Dolgozat létrehozásakor a *MultipleChoiceUC*-ból építi fel az oldalt az alkalmazás. Ezek a kérdés bemeneti mezőjéből, hozzáadás gombból, törlés ikonból és egy másik User Control-ból a *RadioButtonUC*-ból áll, ami pedig egy bemeneti mezőből, egy gombból és egy törlés ikonból.
A dolgozatok kijavítása, megírása és megtekintése is külön, de hasonló User Control-okat használ.

7. ábra a User Control fejlesztői nézetéről

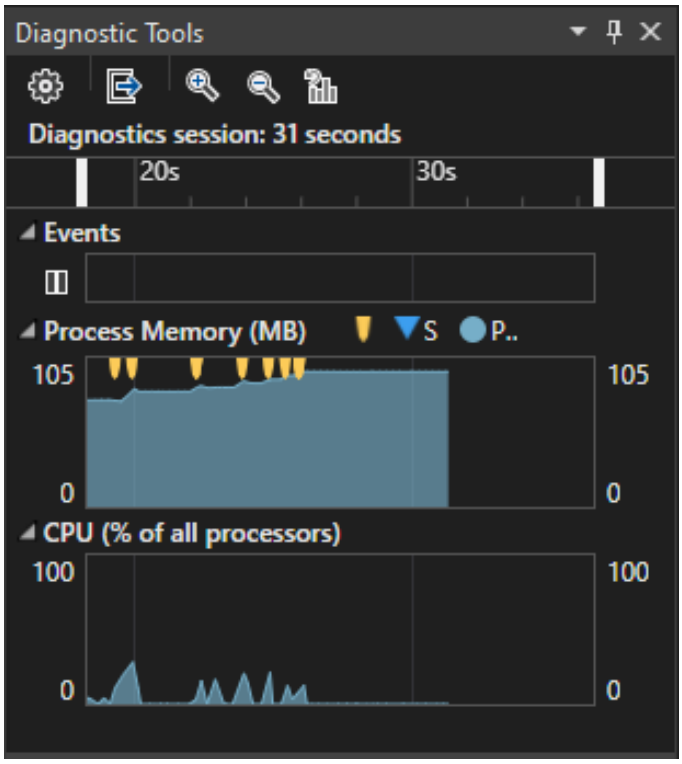
4.6. Tesztdokumentáció

A tesztelés a fejlesztéssel párhuzamosan ment, majd elkészültjekor többször kipróbáltam a működését többféle helyzetben. A teszteket a saját gépemen végeztem melynek lényeges specifikációi:

- ➔ Intel Core I5 7300HQ processzor
- ➔ 8 GB RAM

A program futása a WinForms korlátjainak tudatában gördülékenyen futott a memória használata sosem érte el az 500 MB-ot.

Kikértem a programozásban nem jártas szüleim véleményét az alkalmazással kapcsolatban és egyszerű könnyen kezelhetőnek jellemezték.



8. ábra a diagnosztikai adatokról

A képen jól látszódik a Garbage Collector működése (sárga jelölések). A nem használt adatokat takarítja ezzel csökkentve a memória használatát.

5. Az alkalmazás design-ja

Már a program ötletének kitalálásakor tudtam milyen kinézetet akarok az applikációnak. Legyen sötét témájú, egyszerű, ikonokkal és matricákkal díszítve. A kék alapszín is ekkor lett egyértelmű. A kinézetet folyamatosan készítettem a kódolással párhuzamosan, mivel gyakran ez is befolyásolta azt, hogy mi és hogyan fog működni az egyes oldalakon. A felhasznált ikonokat internetről letöltve majd photoshopban átszerkesztve, átszínezve használtam fel. A logót saját kezűleg készítettem ugyanabban a szoftverben. A program a nevét a lényegéről kapta (dolgozat írató alkalmazás). Ezen felül egy lány név becézése, ami könnyeddé és hétköznapivá teszi kiejtését. Ezen felül az angol „diamond” az az gyémánt szó rövidítése is, ami a védelmére és a kinézetének színválasztására utal.

9. ábra az alkalmazás logójáról

6. Összefoglalás

6.1. A szakdolgozat célja

A szakdolgozatom célja egy olyan applikáció volt, amellyel a tanárok egyszerűen irathatnak dolgozatokat a diákjaikkal. Egy ilyen alkalmazás nagyon jól tud jönni akár az online oktatás alatt is. Segít osztályozni, illetve felmérni az aktuális tudását a diákoknak. Emellett a diákoknak is hasznuk származik belőle, mivel a jó eredmények kifizetődnek és jutalmakkal segíthetik tanulmányaikat.

6.2. Megvalósítása

Először az adatbázis tervének elkészítése volt a célom. Ez alatt kirajzolódott bennem néhány működési mechanizmus és az adatok kezelési módja is.

Kódolásban először a bejelentkezési felületnek álltam neki. Ehhez kapcsolódott a jelszavak titkosítása és a felhasználók jogosultságának kezelése is. Ekkor már tudtam milyen design-t szeretnék a programnak adni.

Ezt követte a főoldal és a keretrendszer kialakítása. Létrehoztam a Parent Formokat és egy sablon Child Formot. Ezt a sablon Formot másoltam és alakítottam minden oldal elkészítésekor. Először mindig felvázoltam mit szeretnék az adott Formon megjeleníteni majd ezután nekiláttam csak neki a kódolásnak.

Először az egyszerűbb Formokat, mint a profil, bolt, dolgozatok és osztályokat készítettem el. Ezután álltam neki annak, hogy lehessen készíteni dolgozatot. Ennek a működésének a kitalálása vett igénybe a legtöbb időt. A WinForms-os alkalmazások nem igazán rugalmasak és megjelenítésük után nehezen adható hozzájuk új elem, ami nagy problémát jelentett. Erre a már említett User Control-ok jelentettek megoldást. Fejlesztés közben gyakran megesett, hogy módosítanom kellett az adatbázison.
A diákok és a tanárok elkészülte után álltam neki az admin felületnek. Ez már könnyebb volt fejlesztés szempontjából, azonban mivel sok mindent tartalmaz sok időt is jelentett.

Legvégül többször is teszteltem az alkalmazást. Néhány “bug” kijavítása után pedig el is készült a program. Egyetlen “hibája” az a WinForms-os alkalmazás határai. Sajnos ez a fajta applikáció nem tud grafikus gyorsítót felhasználni ezért a megjelenés helyenként lassan esetleg villódzva történik. Próbáltam *dupla pufferelést* is alkalmazni a probléma elhárítására mi a sebességen és a memória használaton segített azonban a megjelenítésen nem változtatott.

6.3. Fejlesztési lehetőségek

- ➔ A program átfejlesztése WPF alkalmazásra, ami már képes grafikus gyorsító használatára.
- ➔ API szerver kiépítésével az adatok egyszerre is tudnának módosulni és rengeteg lehetőséget tárna fel (akár web-re és telefonra is portolható lenne).
- ➔ Többféle feladattípus hozzáadása a dolgozatok készítéséhez.
- ➔ A bolt is bővíthető lehetne további elemekkel, ehhez azonban a tanárok egyetértése is kell.

7. Irodalomjegyzék

7.1. Internetes Irodalomjegyzék

A weboldalakat 2022.03.18-án használtam fel utoljára.

- ➔ Programozási problémák megoldásai: https://stackoverflow.com
- ➔ C#-os dokumentáció: https://docs.microsoft.com/en-us/?view=net-6.0
- ➔ Ikonok és matricák: https://www.flaticon.com
- ➔ Adatbázistervező: https://drawsql.app

8. Mellékletek

8.1. Programkódok

8.1.1. OpenChildForm(Form childForm) metódus

```
public void OpenChildForm(Form childForm)
        {
            if (OldChildForm != childForm)
            {
                if (OldChildForm != null)
                {
                    OldChildForm.Dispose();
                    OldChildForm.Close();
                }

                childForm.FormBorderStyle =FormBorderStyle.None;
                childForm.TopLevel = false;
                childForm.Dock = DockStyle.Fill;

                if (childForm.GetType() == new
MainForm().GetType())
                {
                    DesktopP.BackgroundImage =
Properties.Resources.WinFormBg2D;
                }
                else
                {
                    if (OldChildForm.GetType() == new
MainForm().GetType())
                    {
                        DesktopP.BackgroundImage =
childForm.BackgroundImage;
                    }
                }
                DesktopP.Controls.Add(childForm);
                DesktopP.Tag = childForm;

                childForm.BringToFront();
                childForm.Show();
```

```
                OldChildForm = childForm;
            }
        }
```

8.1.2. Példa a LoadingDataSources() metódusra a dolgozatok betöltésnél

```
public void LoadingDataSources() {
            positions.Clear();
            using (SQL sql = SQL.MySql())
            {
                foreach (var item in sql.positions)
                {
                    positions.Add(item);
                }
                foreach (var item in sql.classes.OrderBy(x =>
x.Name))
                {
                    foreach (var p in positions)
                    {
                        if (p.TeacherID ==
CurrentTeacher.Username && p.ClassID == item.ID)
                        {
                            if
(!ClassesCB.Items.Contains(item.Name))
                            {
                                ClassesCB.Items.Add(item.Name);
                            }
                        }
                    }
                }
                tests = sql.tests.Where(x => x.TeacherID ==
CurrentTeacher.Username).ToList();
                DGVLoad(tests);
            }
        }
```

8.1.3. A DetectBrowser() metódus

```
 public static bool DetectBrowser() {
            Process[] processes = Process.GetProcesses();
            bool futE = false;
            foreach (Process p in processes)
            {
                if (!String.IsNullOrEmpty(p.MainWindowTitle))
                {
                    if (new string[] { "opera", "chrome",
"firefox", "msedge" }.Contains(p.ProcessName))
                    {
                        futE = true;
                    }
                }
            }
            if (futE)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
```

8.2. Egy darab CD_ROM a következő tartalommal

- ➔ szakdolgozat digitális változata,
- ➔ eredetiségnyilatkozat,
- ➔ a szakdolgozathoz tartozó programrendszer forráskódja,
- ➔ az adatbázist leíró SQL utasítás,
- ➔ egy tesztelési útmutató.