SZAKDOLGOZAT

SZOFTVER FEJLESZTŐ

esti szakképzés

2013-2015

készítette: Sápi Sándor

Budapest, 2015.

Tartalom

Szakdolgozat dokumentáció

HQ40d-tipusú kézi műszer adatainak feldolgozása

A fejlesztés célja

A HQ40d kézi műszer a különböző vízrendszerek vezetőképesség (µs) és a kémhatás (pH) mérésére szolgál. A víz minősége befolyásolja a csőrendszer és a berendezés állapotát, meggátolja a vízkő lerakódását.

A méréseket heti rendszerességgel végzik, adatait folyamatosan nyilvántartják, a paraméterektől való eltérés esetén beavatkoznak és javítják a víz minőségét.

Cégemnél felmerült annak szüksége, hogy ezeket az adatokat a mérések elvégzése után visszamenőlegesen lehessen vizsgálni minden berendezésre vonatkozóan.

Mivel a műszer az adatokat egy .csv típusú fájlban tárolja a fejlesztés fő iránya ennek a fájlnak a feldolgozására fókuszál.

Fejlesztésemben tehát ezen adatokat fogom grafikus diagramon megjeleníteni, fenntartva a lehetőséget arra, hogy minden dátum időpontot szabadon lehessen keresni, az aktuális berendezéseket kiválasztani.

A program általános felépítése

1. A fájl megnyitása, behívása feldolgozásra.
2. Az adatok leválogatása dátum szerint, minden berendezésre vonatkozóan.
3. Az adatok adatbázisban történő letárolása.
4. Megjelenítés vonalas grafikonon x és y koordináták alkalmazásával, ahol x koordináta a dátum, y koordináta a vezetőképesség, vagy kémhatás értéke.
5. Berendezések nevének kijelzése és színekhez rendelése – kiválasztás lehetősége (megjelenítés, vagy sem)
6. Dátum kezdeti értékének megadása, valamint a dátum végső értékének megadása, amellyel kiválasztható az idő intervallum, amit vizsgálni szeretnénk.

7. Opcióként lehetőség arra, hogy a grafikon aktuális értékére állva a mutatóval, ott az aktuális berendezéshez tartozó adatok gyűjtött kijelzése látható legyen.

Fejlesztői környezet, tervezés és programozás

A program fejlesztői környezete C# .NET Framework 3.5 számú környezetben.

Az adatbázis az MS SQL Server 2012 verziójában készült.

Rendszer igény

Minimálisan egy duo core 2-es processzorral szerelt és minimum 4 GB memóriával rendelkező számítógép szükséges a rendszer futtatásához. Operációs rendszer a Windows 7. verziója. Az operációs rendszeren felül telepíteni kell az MS SQL Server 2012. verzióját és a Diagnosztika adatbázist el kell helyezni az adatbázis alapértelmezett mappájában a DATA mappanév alatt. Ezután az adatbázis kezelőben le kell futtatni a CD-n mellékelt Diagnosztika.bak fájlt. Ezután az adatbázis láthatóvá válik a program számára. Ellenkező esetben a program hibaüzenetben kéri a helyes beállításokat. A beállításokhoz általában rendszergazda segítsége szükséges.

Felhasználói leírás

Egy rövid tájékoztatóban bemutatom programom használatát.

Első lépésként a menüben ki kell választani a File menüpontot és benne a File betöltés sort. Ekkor megjelenik a fájl megnyitása ablak, ahol a CD-n levő szakdolgozat mappában ki kell keresni a betölteni kívánt fájlt (pld. „HQ40dUpload_20110211_123234”) néven találhatót. Kiválasztás után a megnyitás gombra kattintva elindul a fájl feldolgozása. A feldolgozás folyamatát a felület jobb felső részén látható adatfeltöltés jelző mutatja. Ha a feldolgozás befejeződött, akkor egy információs panel jelenik meg jelezve a folyamat végét. Ezután a feltöltött adatok megtekinthetők az Adattáblák menüpontnál.

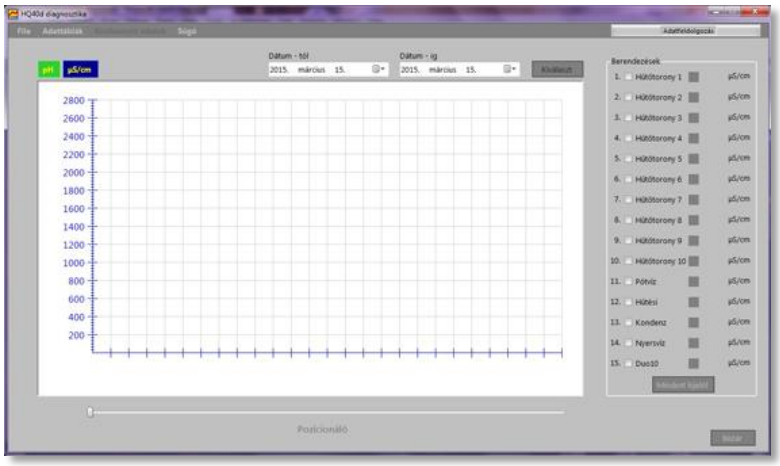
Második lépésként a „Dátum-tól” elnevezésű lenyíló felületen állítsuk be a kiinduló dátumot, a következő „Dátum-ig” elnevezésű felületen pedig a végső dátumot. Evvel megadtuk a dátum intervallum értékét. A Kiválaszt gombra kattintva az adatok betöltődnek a rendszerbe. A betöltődés végét egy információs panel jelzi és tájékoztat a bevitt dátumokról. A felület jobb oldalán látható „Berendezések” nevű panelen bármelyik berendezést kiválaszthatjuk és láthatóvá válnak a grafikonok. A felület bal felső részén található kiválasztó gombokkal a két mérés típust lehet kiválasztani (ph, vagy µs/cm).

Amennyiben az adatok túlcsordulnak a felület megjelenési lehetőségein, akkor két görgető gomb jelenik meg, mellyel jobbra és balra görgethető a grafikon.

Ha konkrét adatokat szeretnénk a grafikonokról, használjuk a pozícionáló csúszkáját. Ennek elmozdításával egy függőleges vonalat mozgathatunk és ha ez metszi a grafikon adatpontjait akkor megjelenik egy apró négyzet, benne a berendezés sorszámával. Az adatponthoz tartozó pontos adat a felület jobb oldalán látható berendezés nevek mellett olvasható le.

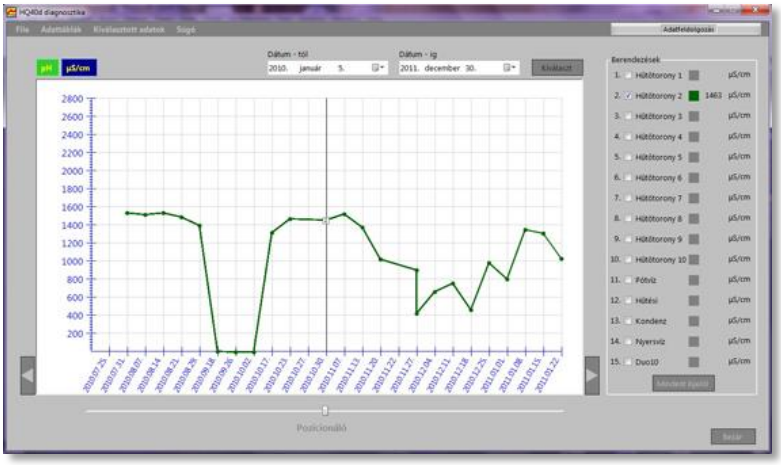
A tervezés fázisai

A felhasználói felület megtervezése – ahol a felhasználhatóság és ergonómia figyelembe vételével kellett a grafikus ablakokat kialakítani. Fő szempont az egyszerű kezelhetőség és egyértelmű, minden felesleges külsőségtől mentes adat megjelenítés. *A főoldal képe az 1. árán látható.*



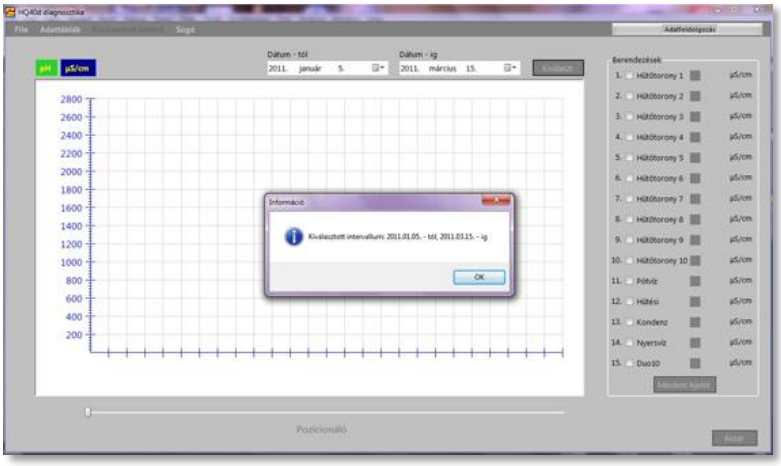
1. ábra - A fő oldal képe

A felület fő részét a grafikonokat megjelenítő fehér terület alkotja, ahol rácsos formában látható a grafikon x és y koordinátája. Az x koordináta a kiválasztott adatok dátumát mutatja, míg az y koordináta az adatok vezetőképességét, vagy kémhatását mutatja választástól függően. A választás lehetőségét a felület bal felső részén található, két különböző színnel megjelölt gombja biztosítja. (zöld „pH” = kémhatás – kék „µs/cm” = vezetőképesség) Az adatok pozícionálását a pozícionáló csúszka elmozdításával lehet elvégezni. Ha az adatok mennyisége nagyobb a grafikon felosztott felületénél akkor két mozgató gomb jelenik meg, amellyel az adatok jobbra – balra görgethetők a felületen. *A megjelenő felület a 2. ábrán látható.*



2. ábra - Görgethető felület

Az adatokat a dátum-tól dátum-ig jelzésű kontrolokon keresztül lehet megadni és a „Kiválaszt” gomb megnyomásával lehet betölteni. *A felület a 3. ábrán látható.*

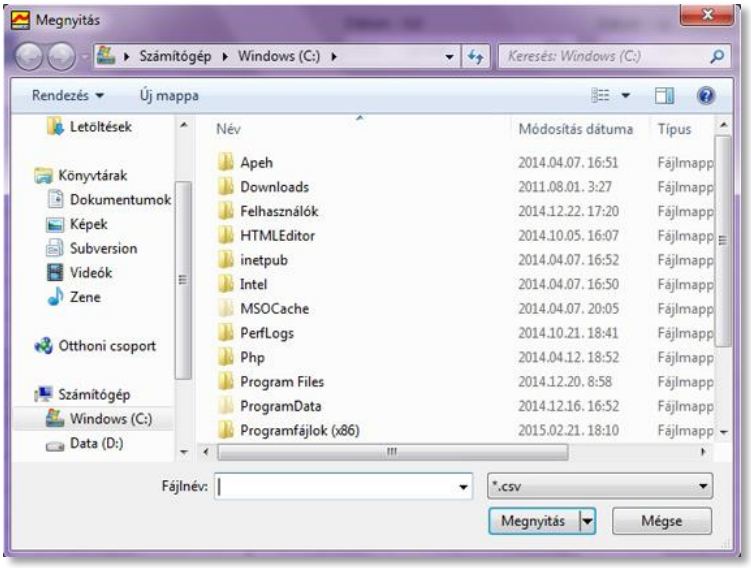


3. ábra - Dátum intervallum kiválasztása

A felület jobb oldalán a „Berendezések” szekció alatt megjelenő kiválasztó gombokkal a különböző berendezések jelölhetők meg megjelenésre.

Menürendszer:

A felület felső sávjában látható menürendszer felépítése funkcionális. Elsőként a „File” menü, ahol a feldolgozni kívánt „csv” fájl hívható be feldolgozásra. *A felület képe a 4. ábrán látható.*

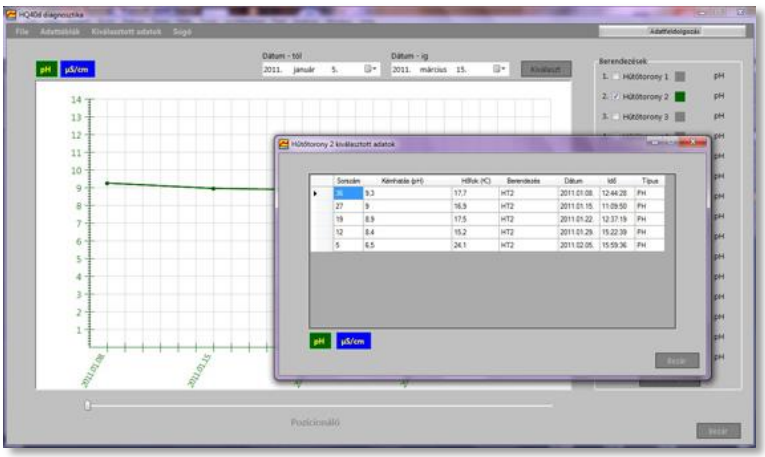


4. ábra - Fájl kiválasztása

Az „Exit” bezárja az alkalmazást.

Az „Adatok” menüpont a már feldolgozott adatok adattábláit mutatja a feldolgozás típusa szerint.

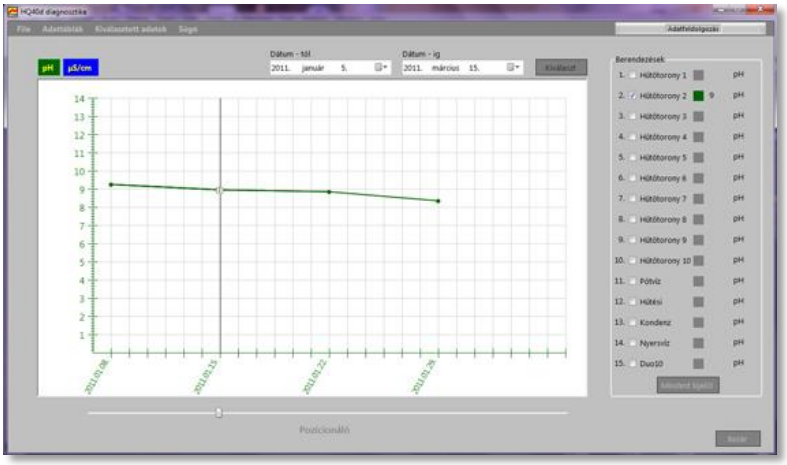
A „Kiválasztott adatok” menüpont a kiválasztott adatokat jeleníti meg berendezésekre lebontva. *A felület képe az 5. ábrán látható.*



5. ábra - A kiválasztott berendezés adatai

Lehetőség van az összes berendezés kiválasztására is a „Mindent kijelöl” gomb megnyomásával.

A „Pozícionáló” a grafikonon történő adatmegjelenítést segíti, mivel az adatpontra állva egy négyzet jelenik meg, jelezve a berendezés sorszámát, valamint megjelenik az adott ponthoz tartozó érték is. *A felület képe a 6. ábrán látható.*



6. ábra - Pozícionáló funkció

Objektumok tervezése

A program osztály szintű megtervezése – ahol a különböző osztály objektumok funkciók szerint lettek megtervezve.

„FileKezelő” osztály:

- Ez a feldolgozó egység végzi a „csv” fájl adatszintű feldolgozását és leválogatja az adatokat adatbázis mezőkre és sorokra.

„Adatkezelő” osztály:

- Ez az egység az adatok mozgatását végzi az adatbázis objektumból, vagy vissza az adatbázis objektumba. Különböző select lekérdezések, insertek, és adatellenőrzések metódusai találhatók meg.

„Grafika” osztály:

- Ez az osztály felel a grafikus megjelenítésekért, a folyamatos renderelésért. Fontos követelmény, hogy a felület nem villoghat a különböző renderelések között, ezért az osztály puffereli a grafikus objektumokat.

„Koordináta” osztály:

- Ebben az osztályban tárolódnak el a különböző mérés adatokhoz tartozó koordináta adatok. Ennek az osztálynak a felhasználásával valósítható meg a pozícionálás funkció.

„Diagnosztika.designer” osztály:

- Ez az osztály felelős az adatbázis kapcsolatokért és adatmozgásokért.
- Elkülöníti az adatbázist a felhasználótól és objektumokra bontja a különböző adatbázis műveleteket.

„Form...” osztályok:

- Ezek az osztályok jelenítik meg a felhasználói felületeket és biztosítják az interakciót a felhasználó és a program funkciói között.

A „Diagnosztika” adatbázis

Az adatbázis célja:

Az adatbázis a HQ40d típusú vízanalitikai műszer adatait tárolja. A mérések rendszeres időpontokban történnek, ahol rögzítik a dátumot, mérés típust (kémhatás / vezetőképesség), hőfokot, a mérés kódját, a berendezés azonosítóját. A műszer adatain kívül rögzítem a fájl nevét, melyből az adatok származnak.

Az adatbázisba kívülről csak egyetlen táblához férhetünk hozzá, ahol a berendezések nevét (azonosítóját) tároljuk. *Az adatbázishoz társított program ezen adatok nélkül nem rögzít semmit az adatbázisban. Ezek az adatok meg kell, hogy egyezzenek a műszerbe rögzített berendezés azonosítókkal!*

Az adatok feldolgozását egy „csv” fájlt feldolgozó algoritmus végzi, ahol az adatok leválogatódnak táblaszintű adatokká. Minden adatsor egy mérés adattípusnak felel meg. Minden adatellenőrzést az algoritmusra bíztam. Így nem történhet meg - azonos adatok felvitele, hibásan rögzített adatok tárolása.

Az adatbázis funkciói:

<u>Fájlnév</u>

- A már feldolgozott fájl neve felkerül az adatbázis táblába.

<u>Berendezés neve</u>

- A berendezések nevének tárolása.

Dátum adatok

- A mérés dátumának és időpontjának tárolása későbbi lekérdezések miatt.

Mérés típus adatok

- Itt tároljuk a mérés típus adatait, CD, vagy PH.

Személyek adatai

- A mérést végző személyek nevének tárolása.

Vezetőképesség adatai

- Típus szerint itt tároljuk a vezetőképesség mérés adatait (vezetőképesség, hőfok, mérés kódja).

Kémhatás adatai

- Típus szerint itt tároljuk a kémhatás mérés adatait (kémhatás, hőfok, mérés kódja).

Az adatbázis táblák szerkezete:

Fájlnév: Nyilvántartási célra – a program ellenőrzi, hogy történt-e adatfeldolgozás.

Fajlnev {fajlID, fajlnev}

fajlID (int);

fajlnev (nvarchar)

Berendezés: A feldolgozás során csak a táblában tárolt berendezésekhez társulhat bármilyen adat. Az adatokat a programból manuálisan kell feltölteni!

Berendezesek {berendezesID, berendezes_nev}

berendezesID (int);

berendezes_nev (nvarchar)

Dátum: A mérések dátumát és idejét rögzíti a feldolgozó algoritmus

Mikor {datumID, datum, ido, szemely}

datumID (int);

datum (date);

ido (nvarchar);

szemely (int)

Mérés típus: A mérés kétféle típusát és a hozzá tartozó azonosítót rögzíti a feldolgozó algoritmus. A feldolgozás során csak egy-egy sornyi bejegyzés keletkezik.

Tipus {typeID, tipus, azonosito}

typeID (int);

tipus (nvarchar);

azonosito (nvarchar)

Személyek: A mérést végző személyek neve és beosztása kerül letárolásra. Minden személynév csak egyszer szerepelhet! A beosztás megadása nem kötelező.

Szemelyek {szemelyID, nev, beosztas}

szemelyID (int);

nev (nvarchar);

beosztas (nvarchar)

Vezetőképesség: A vezetőképesség típusú adatok kerülnek letárolásra.

Vezetokepesseg {vezID, vezetokepesseg, hofok, tipus, mikor, meres_kod, berendezes_azonosito}

vezID (int);

vezetokepesseg (float);

hofok (float);

tipus (int);

mikor (int);

meres_kod (int);

berendezes_azonosito (int)

Kémhatás: A kémhatás típusú adatok kerülnek letárolásra.

Kemhatas {phID, kemhatas, hofok, tipus, mikor, meres_kod, berendezes_azonosito}

phID (int);

kemhatas (float);

hofok (float);

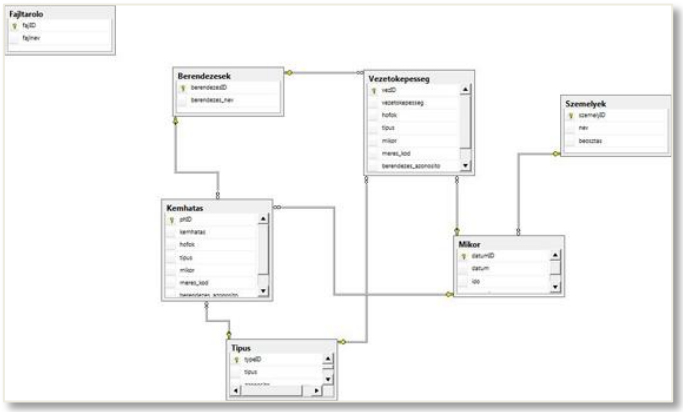
tipus (int);

mikor (int);

meres_kod (int);

berendezes_azonosito (int)

A táblák kapcsolati ábrája



7. ábra - Adatbázis diagram képe

Adatbázis scriptje:

```
USE [master]
GO
/****** Object:  Database [Diagnosztika]    Script Date: 2015.02.20. 19:16:14 ******/
CREATE DATABASE [Diagnosztika]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'Diagnosztika', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\MSSQL\DATA\Diagnosztika.mdf' , SIZE = 10240KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'Diagnosztika_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\MSSQL\DATA\Diagnosztika_log.ldf' , SIZE = 768KB , MAXSIZE = UNLIMITED, FILEGROWTH = 10%)
GO
ALTER DATABASE [Diagnosztika] SET COMPATIBILITY_LEVEL = 110
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Diagnosztika].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
```

```
ALTER DATABASE [Diagnosztika] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Diagnosztika] SET ANSI_NULLS OFF
GO
ALTER DATABASE [Diagnosztika] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Diagnosztika] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [Diagnosztika] SET ARITHABORT OFF
GO
ALTER DATABASE [Diagnosztika] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [Diagnosztika] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [Diagnosztika] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [Diagnosztika] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [Diagnosztika] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [Diagnosztika] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [Diagnosztika] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [Diagnosztika] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [Diagnosztika] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [Diagnosztika] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [Diagnosztika] SET  DISABLE_BROKER
GO
ALTER DATABASE [Diagnosztika] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [Diagnosztika] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [Diagnosztika] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [Diagnosztika] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [Diagnosztika] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [Diagnosztika] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [Diagnosztika] SET HONOR_BROKER_PRIORITY OFF
GO
```

```
ALTER DATABASE [Diagnosztika] SET RECOVERY SIMPLE
GO
ALTER DATABASE [Diagnosztika] SET  MULTI_USER
GO
ALTER DATABASE [Diagnosztika] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [Diagnosztika] SET DB_CHAINING OFF
GO
ALTER DATABASE [Diagnosztika] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [Diagnosztika] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
USE [Diagnosztika]
GO
/****** Object:  StoredProcedure [dbo].[ResetIdentity]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:<Author,,Name>
-- Create date: <Create Date,,>
-- Description:<Description,,>
-- =============================================

CREATE PROCEDURE [dbo].[ResetIdentity] AS
  DBCC CHECKIDENT (Vezetokepesseg, RESEED, 0)
GO
/****** Object:  StoredProcedure [dbo].[ResetIdentity1]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:<Author,,Name>
-- Create date: <Create Date,,>
-- Description:<Description,,>
-- =============================================

CREATE PROCEDURE [dbo].[ResetIdentity1] AS
  DBCC CHECKIDENT (Mikor, RESEED, 0)
GO
```

```
/****** Object:  StoredProcedure [dbo].[ResetIdentity2]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:<Author,,Name>
-- Create date: <Create Date,,>
-- Description:<Description,,>
-- =============================================

CREATE PROCEDURE [dbo].[ResetIdentity2] AS
  DBCC CHECKIDENT (Kemhatas, RESEED, 0)
GO
/****** Object:  StoredProcedure [dbo].[ResetIdentity3]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:<Author,,Name>
-- Create date: <Create Date,,>
-- Description:<Description,,>
-- =============================================

CREATE PROCEDURE [dbo].[ResetIdentity3] AS
  DBCC CHECKIDENT (Szemelyek, RESEED, 0)
GO
/****** Object:  StoredProcedure [dbo].[ResetIdentity4]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:<Author,,Name>
-- Create date: <Create Date,,>
-- Description:<Description,,>
-- =============================================

CREATE PROCEDURE [dbo].[ResetIdentity4] AS
```

```
 DBCC CHECKIDENT (Fajltarolo, RESEED, 0)
GO
/****** Object:  StoredProcedure [dbo].[ResetIdentity5]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO



-- =============================================
-- Author:<Author,,Name>
-- Create date: <Create Date,,>
-- Description:<Description,,>
-- =============================================

CREATE PROCEDURE [dbo].[ResetIdentity5] AS
 DBCC CHECKIDENT (Berendezesek, RESEED, 0)

GO
/****** Object:  Table [dbo].[Berendezesek]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Berendezesek](
[berendezesID] [int] IDENTITY(1,1) NOT NULL,
[berendezes_nev] [nvarchar](20) NOT NULL,
 CONSTRAINT [PK_Berendezesek] PRIMARY KEY CLUSTERED
(
[berendezesID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Fajltarolo]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Fajltarolo](
[fajlID] [int] IDENTITY(1,1) NOT NULL,
[fajlnev] [nvarchar](100) NOT NULL,
 CONSTRAINT [PK_Fajltarolo] PRIMARY KEY CLUSTERED
(
```

```
[fajlID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Kemhatas]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Kemhatas](
[phID] [int] IDENTITY(1,1) NOT NULL,
[kemhatas] [float] NOT NULL,
[hofok] [float] NOT NULL,
[tipus] [int] NOT NULL,
[mikor] [int] NOT NULL,
[meres_kod] [int] NOT NULL,
[berendezes_azonosito] [int] NOT NULL,
 CONSTRAINT [PK_Kemhatas] PRIMARY KEY CLUSTERED
(
[phID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Mikor]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Mikor](
[datumID] [int] IDENTITY(1,1) NOT NULL,
[datum] [date] NOT NULL,
[ido] [nvarchar](50) NOT NULL,
[szemely] [int] NOT NULL,
 CONSTRAINT [PK_Mikor] PRIMARY KEY CLUSTERED
(
[datumID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Szemelyek]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Szemelyek](
[szemelyID] [int] IDENTITY(1,1) NOT NULL,
[nev] [nvarchar](50) NOT NULL,
[beosztas] [nvarchar](50) NULL,
 CONSTRAINT [PK_Szemelyek] PRIMARY KEY CLUSTERED
(
[szemelyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Tipus]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Tipus](
[typeID] [int] IDENTITY(1,1) NOT NULL,
[tipus] [nvarchar](2) NOT NULL,
[azonosito] [nvarchar](12) NOT NULL,
 CONSTRAINT [PK_Tipus] PRIMARY KEY CLUSTERED
(
[typeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Vezetokepesseg]    Script Date: 2015.02.20. 19:16:14 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Vezetokepesseg](
[vezID] [int] IDENTITY(1,1) NOT NULL,
[vezetokepesseg] [float] NOT NULL,
[hofok] [float] NOT NULL,
[tipus] [int] NOT NULL,
[mikor] [int] NOT NULL,
[meres_kod] [int] NOT NULL,
[berendezes_azonosito] [int] NOT NULL,
 CONSTRAINT [PK_Vezetokepesseg] PRIMARY KEY CLUSTERED
(
```

```
[vezID] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]


GO

ALTER TABLE [dbo].[Kemhatas]  WITH CHECK ADD  CONSTRAINT [FK_Kemhatas_Berendezesek] FOREIGN KEY([berendezes_azonosito])

REFERENCES [dbo].[Berendezesek] ([berendezesID])

GO

ALTER TABLE [dbo].[Kemhatas] CHECK CONSTRAINT [FK_Kemhatas_Berendezesek]

GO

ALTER TABLE [dbo].[Kemhatas]  WITH CHECK ADD  CONSTRAINT [FK_Kemhatas_Mikor] FOREIGN KEY([mikor])

REFERENCES [dbo].[Mikor] ([datumID])

GO

ALTER TABLE [dbo].[Kemhatas] CHECK CONSTRAINT [FK_Kemhatas_Mikor]

GO

ALTER TABLE [dbo].[Kemhatas]  WITH CHECK ADD  CONSTRAINT [FK_Kemhatas_Tipus] FOREIGN KEY([tipus])

REFERENCES [dbo].[Tipus] ([typeID])

GO

ALTER TABLE [dbo].[Kemhatas] CHECK CONSTRAINT [FK_Kemhatas_Tipus]

GO

ALTER TABLE [dbo].[Mikor]  WITH CHECK ADD  CONSTRAINT [FK_Mikor_Szemelyek] FOREIGN KEY([szemely])

REFERENCES [dbo].[Szemelyek] ([szemelyID])

GO

ALTER TABLE [dbo].[Mikor] CHECK CONSTRAINT [FK_Mikor_Szemelyek]

GO

ALTER TABLE [dbo].[Vezetokepesseg]  WITH CHECK ADD  CONSTRAINT [FK_Vezetokepesseg_Berendezesek] FOREIGN KEY([berendezes_azonosito])

REFERENCES [dbo].[Berendezesek] ([berendezesID])

GO

ALTER TABLE [dbo].[Vezetokepesseg] CHECK CONSTRAINT [FK_Vezetokepesseg_Berendezesek]

GO

ALTER TABLE [dbo].[Vezetokepesseg]  WITH CHECK ADD  CONSTRAINT [FK_Vezetokepesseg_Mikor] FOREIGN KEY([mikor])

REFERENCES [dbo].[Mikor] ([datumID])

GO

ALTER TABLE [dbo].[Vezetokepesseg] CHECK CONSTRAINT [FK_Vezetokepesseg_Mikor]

GO

ALTER TABLE [dbo].[Vezetokepesseg]  WITH CHECK ADD  CONSTRAINT [FK_Vezetokepesseg_Tipus1] FOREIGN KEY([tipus])

REFERENCES [dbo].[Tipus] ([typeID])

GO

ALTER TABLE [dbo].[Vezetokepesseg] CHECK CONSTRAINT [FK_Vezetokepesseg_Tipus1]

GO

USE [master]

GO

ALTER DATABASE [Diagnosztika] SET  READ_WRITE

GO
```

A „csv” fájl

A műszer által szolgáltatott fájl egy .csv kiterjesztésű adattömeg. Az adatokat sorokra bontva tartalmazza típus és dátum szerint. A sorok „RD” jelöléssel kezdődnek és „EndOfRecord” jelöléssel végződnek. A fájl első része az adatok típusait mutatja. Az adatok egy vesszővel vannak elválasztva (ezt hívjuk szeparátornak). Ezek a jelölések biztosítják, hogy a különböző adatokat el lehessen különíteni a feldolgozás során. A programom az adatokat csak a hőmérséklet adatokig vizsgálja, a többi adat a program funkciói szempontjából közömbös.

A .csv fájl egy részlete:

.....

RD,CD,2011-2-5,16:02:49,FAZOLD CS,CDC401,>080242580002,A,HT1 (047),857,µS/cm,19.1,°C,,,,,NaCl,Reference Temperature = 25 °C,,,,,,,,,,OK,2010-11-22,13:40:29,TOTH CS,Cell constant,0.306,,cm-1,,,,1,1000,µS/cm,988,µS/cm,24.4,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,°C,,,,,1296921769,1290433229,EndOfRecord

RD,pH,2011-2-5,16:02:49,FAZOLD CS,pHC101,<093002561024,A,HT1 (047),6.5,pH,20.5,°C,37.0,mV,,,,,,,,,,,,,,OK,2010-11-22,13:37:47,TOTH CS,Slope,-57.27,(97%),mV/pH,9.7,mV,,2,4.01,pH,181.8,mV,26.1,7.00,pH,9.9,mV,26.1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,°C,,,,,1296921769,1290433067,EndOfRecord

.....

Programozás

A .csv fájl feldolgozása

A programozás fő osztályait már bemutattam. Most az osztályokat programozási oldalról vizsgálom és mutatom be.

A programom egyik kulcsfontosságú egysége a .csv fájlt feldolgozó osztály.

Az osztály célja, hogy a .csv fájl tartalmát jól elkülöníthető adattípusokra bontsa fel, mégpedig nagy biztonsággal és hibák nélkül. Fontos tudni, hogy a .csv fájl önmagában nem egy hibamentes adattömeg. A fejlesztés során tehát különösen fontos, ezen hibák feltárása és kezelése.

1. A .csv fájl sorai a „BD” jelöléssel kezdődnek. Tehát a sorok vizsgálata csak ezen adat megléte esetén értelmes. *A kód a 8. ábrán látható.*

```
//Adatok vizsgálata, hogy létezik e az RD sor
if (adatok[0] == "RD")
{
for (int j = 0; j < 13; j++)
{
meres = adatok[10].ToString();
meres = meres.Replace(".", ",");
berendezes = adatok[8].ToString();
berendezes = berendezes.Remove(berendezes.Length-1);
........
```

8. ábra - Adatsorok vizsgálata

2. Következő szempont, hogy csak akkor helyes a sor kiértékelése, ha a berendezés létezik az adatbázis táblánkban és a mérés adatai számok és nem vegyülnek más karakterekkel. *A kód a 9. ábrán látható.*

```
//Ha a berendezés létezik és a mérés adatai számok
if (ak.berendezesEllenor(berendezes) && stringFigyelo(meres))
{
  tipus = adatok[1].ToString();
   datum = adatok[2].ToString();
   ido = adatok[3].ToString();
  felhasznalo = adatok[4].ToString();
   mero_kod = Convert.ToInt32(adatok[9].ToString());
   hofok = adatok[12].ToString();
   hofok = hofok.Replace(".", ",");
   hofok = hofok.Substring(0, 4);
   .....
```

9. ábra - Helyes adatok

3. További feltétel, hogy szétválasszam a különböző mérés típusokat. Lehet „CD” vagy „PH” típus. Az adatok rögzítése csak a típus szerinti leválogatásban kerül letárolásra az adatbázisban. *A kód a 10. ábrán látható.*

```
if (tipus == "CD")
{
        vezetokepesseg = meres;
         ak.alapadatRogzites(tipus, datum, ido, felhasznalo, berendezes, mero_kod, vezetokepesseg, hofok);
 }
 else
{
        kemhatas =  meres;
        ak.alapadatRogzites(tipus, datum, ido, felhasznalo, berendezes, mero_kod, kemhatas, hofok);
 }
```

10. ábra - Adattípusok elkülönítése

Grafikus megjelenítés

A programozás egyik legnehezebb része az adatok grafikus megjelenítése volt. Itt nem csak a különböző Form-ok megtervezéséről van szó. A grafikon elkészítése és működtetése ettől teljesen más programozási gondolkodást kíván meg. Nézzük a lépéseket.

1. A két adattípusnak két különböző koordináta panel lett létrehozva. Ezen panelek kapcsolók segítségével jeleníthetők meg. Mindig csak egy panel aktív.
2. Minden panelen a maga típusa szerint lett kialakítva a koordináta rendszer. Az x koordináta a dátumokat mutatja, míg az y koordináta az adat értékeket. A koordinátára a jobb láthatóság érdekében egy rács lett készítve.
3. Az adatok megjelenése grafikus pontokból áll és ezeket a pontokat vonal köti össze. Kirajzolásuk attól függ, hogy a berendezés ki lett e választva, valamint, hogy a dátum intervallum meg lett e adva. *A kód a 11-12. ábrán látható.*

```
//HT1 vezetőképesség adatpontok kirajzolása
 private void grafHT1VK(Graphics g, int leptek, int index)
 {
    g.SmoothingMode = SmoothingMode.HighQuality;
    g.SmoothingMode = SmoothingMode.AntiAlias;
    Brush ht1Brush = new SolidBrush(Color.Magenta);
    if (ht1VKLista.Count != 0)
    {
       int ertek = 450 - Convert.ToInt32(Math.Round((ht1VKLista[index].vezetokepesseg1 * 0.150), 0, MidpointRounding.ToEven));
       g.FillEllipse(ht1Brush, 86 + leptek, ertek - 2, 7, 7);
       ht1KoVezk.Add(new Koordinata() { ht1VX = 86 + leptek, ht1VY = ertek });
    }
 }
```

11. ábra - Adatpontok megjelenítése

```
   //HT1 vezetőképesség vonal kirajzolása
private void grafVonalHT1VK(Graphics g)
{
   g.SmoothingMode = SmoothingMode.HighQuality;
   g.SmoothingMode = SmoothingMode.AntiAlias;
   Brush ht1Brush = new SolidBrush(Color.Magenta);
   Pen ht1Pen = new Pen(ht1Brush, 2);
   if (ht1KoVezk.Count != 0)
   {
      for (int i = 0; i < ht1KoVezk.Count - 1; i++)
      {
            g.DrawLine(ht1Pen, (ht1KoVezk[i].ht1VX) + 3, (ht1KoVezk[i].ht1VY) + 1, (ht1KoVezk[i + 1].ht1VX) + 3,  (ht1KoVezk[i + 1].ht1VY) + 1);
      }
   }
}
```

12. ábra - Adatvonalak megjelenítése

Azért, hogy a kép kirajzolása folyamatos legyen és akadás ne legyen látható a megjelenő kép pufferelésére van szükség. Tehát a különböző grafikus objektumoknak paraméterként átadott Graphics típust pufferelni kell. Ezt a műveletet központilag kell végezni, hiszen több grafikus objektumot kell megjeleníteni – koordináták, rácsok, grafikus pontok és vonalak. *A kód a 13. ábrán látható.*

```
//Pufferelt vezetőképesség grafika
 public void DrawToBufferVK(BufferedGraphics vg)
 {
        grafika.tBPoz = trackBarPozicio.Value;
        vg.Graphics.FillRectangle(Brushes.White, 0, 0, 908, 525);
        grafika.vkGraf(vg.Graphics);
 }
```

13. ábra - Pufferelt grafika

Minden pillanatban amikor a teljes grafika kirajzolásra kerül pld. pozícionáláskor, vagy a koordináta görgetésekor stb. a felületet újra és újra meg kell rajzoltatni. A rajzolás után a grafikát eltároljuk a számítógép memóriájában, majd az új rajzoláskor megjelenítjük. Így biztosítjuk a villogásmentes megjelenést.

4. Pozícionáláskor az adatpontra állítva a pozícionáló csúszkáját egy kis négyzet jelenik meg, mutatva a berendezés sorszámát és kiíratva a pozícióhoz tartozó adatot.

Ehhez egy négyzet grafikus objektumot hoztam létre, mely tartalmazza a berendezés sorszámát is. *A kód a 14. ábrán látható.*

```
//Pozicionáló négyzet objektuma
private void negyzet_1(Graphics g, int pozX, int pozY)
{
    Brush szamBrush = new SolidBrush(Color.Black);
    Pen szamPen = new Pen(szamBrush, 1);
    Brush negyzetBrush = new SolidBrush(Color.White);
    Pen negyzetPen = new Pen(negyzetBrush, 1);
    g.DrawRectangle(szamPen, pozX-2, pozY - 3, 10, 10);
    g.FillRectangle(negyzetBrush, pozX-2, pozY - 3, 10, 10);
    g.DrawString("1", new Font("Arial", 6), szamBrush, pozX, pozY - 3);
}
```

14. ábra - Négyzet objektum

Kirajzolásának feltétele, hogy a letárolt adatpont x koordináta adatai megegyezzenek a csúszka állásának x koordináta értékével és mind a berendezés, mind a panel kiválasztása igaz legyen. *A kód a 15. ábrán látható.*

```
if (ht1)
{
    if (_panelVezk)
    {
        ertek1 = null;
        for (int i = 0; i < ht1KoVezk.Count; i++)
        {
            if (ht1KoVezk[i].ht1VX == (90 + trackBarpozicio) - 4)
            {
                negyzet_1(g, ht1KoVezk[i].ht1VX, ht1KoVezk[i].ht1VY);
                ertek1 = ht1VKLista[i+ lepteto].vezetokepesseg1.ToString();
            }
        }
    }
```

15. ábra - Pozíció kiértékelése

5. A grafikonok léptéke annak megfelelően változik, hogy a kiválasztott dátum intervallum adatai belefférnek e a megrajzolt koordinátába. Ha nem férnek el, akkor két görgető gomb jelenik meg a felületen. Ez a funkció szintén a léptetések ütemében újrarajzolja a koordinátát és pufferelten jeleníti meg ismét. *A kód a 16. ábrán látható.*

```
//Görgetés jobbra
private void btnJobb_Click(object sender, EventArgs e)
{
    koordinatakTorlese();
    labelErtekNullazo();
    if (grafika.leptetoIndex < 1)
    {
        grafika.leptetoIndex = 0;
        if (rbtnVezetokepesseg.Checked)
        {
            panelDiagram.Refresh();
            grafxVK.Render(Graphics.FromHwnd(panelDiagram.Handle));
            DrawToBufferVK(grafxVK);
        }
        if (rbtnKemhatas.Checked)
        {
            panelKemhatas.Refresh();
            grafxKH.Render(Graphics.FromHwnd(panelKemhatas.Handle));
            DrawToBufferKH(grafxKH);
        }
    }
    else
    {
        grafika.leptetoIndex--;
        if (rbtnVezetokepesseg.Checked)
        {
            panelDiagram.Refresh();
            grafxVK.Render(Graphics.FromHwnd(panelDiagram.Handle));
            DrawToBufferVK(grafxVK);
        }
        if (rbtnKemhatas.Checked)
        {
            panelKemhatas.Refresh();
            grafxKH.Render(Graphics.FromHwnd(panelKemhatas.Handle));
            DrawToBufferKH(grafxKH);
        }
    }
}
```

16. ábra - Görgetés funkció

Más esetben az x koordinátában arányosan elossza a dátum adatokat a program.

Minden berendezés grafikon más és más színnel mutatja az adatokat, így a különböző mérési adatok berendezés-szinten összehasonlíthatóak lesznek.

Adatbázis műveletek

Az adatbázissal kapcsolatos műveleteket két csoportra lehet bontani.

- Az adatbázissal közvetlen kapcsolat kezelése
- Az adatok átadása-átvétele az adatbázis objektumtól

Ezt a két funkciót két különböző osztály végzi el.

A közvetlen kapcsolatot és műveleti objektumokat a Diagnosztika.designer.cs osztály valósítja meg. Ennek az osztálynak a létrehozását a Visual Studio biztosítja, miután létre lett hozva maga az MS SQL Diagnosztika nevű adatbázis. A különböző műveleteket objektumokra bontja, melyeket a C# programnyelv LINQ névterének használatával érhetünk el. Így biztosított az adatbázis skálázhatósága, valamint elszeparálása a program többi adatától.

```
//A mérés adatainak rögzítése
public void alapadatRogzites(string tipus, string d, string idoAdat, string felhasznalo, string berendezes, int mero_kod, string meresAdat, string hofokAdat)
{
int tipus_id = 0;
int szemely_id = 0;
int datum_id = 0;
DateTime dat = Convert.ToDateTime(d);
tipus_id = tipusAdat(tipus);
int berendezes_id = berendezesIndex(berendezes);
//Ha a felhasználó még nincs rögzítve
if (szemelyEllenor(felhasznalo) == false)
{
Szemelyek szemely = new Szemelyek { nev = felhasznalo };
db.Szemelyeks.InsertOnSubmit(szemely);
db.SubmitChanges();
}
szemely_id = szemelyAdat(felhasznalo);
Mikor mikor = new Mikor { datum = dat, ido = idoAdat, szemely = szemely_id };
db.Mikors.InsertOnSubmit(mikor);
db.SubmitChanges();
datum_id = datumAdat(dat);
switch (tipus)
{
case "CD":
                    Vezetokepesseg vezK = new Vezetokepesseg { tipus = tipus_id, berendezes_azonosito =     berendezes_id,  hofok = Double.Parse(hofokAdat),
 vezetokepesseg1 = Double.Parse(meresAdat), mikor = datum_id, meres_kod = mero_kod };
db.Vezetokepessegs.InsertOnSubmit(vezK);
db.SubmitChanges();
break;
case "pH":
                    Kemhata kemH = new Kemhata { tipus = tipus_id, berendezes_azonosito = berendezes_id, hofok = Double.Parse(hofokAdat), kemhatas =
                    Double.Parse(meresAdat), mikor = datum_id, meres_kod = mero_kod };
db.Kemhatas.InsertOnSubmit(kemH);
db.SubmitChanges();
break;
}
}
```

17. ábra - LINQ adatbázis műveletek

Az adatok mozgatását pedig az Adatkezelő.cs osztály végzi. Ebben az osztályban található minden olyan művelet, ami összefügg az adatbázisban tárolt adatokkal. (Select, Insert, Update, Delete) A gyakorlatban itt nem minden funkcióját használtam. *A kód a 17. ábrán látható.*

Mivel relációs adatbázis kezelésről van szó, a különböző adattáblák szoros kapcsolatban állnak egymással. Kivétel ettől a magányosan álló Fájltároló tábla.

A kialakított módszerrel bármilyen adat-összefüggést lekérdezhetünk az adatbázisból.

Mivel a programom funkciója az adatok feldolgozás utáni tárolása és lekérdezése, itt most más adatbázis műveletet nem végzek.

Tesztelés

Adat tesztelés

A programozás során a különböző logikai egységek folyamatosan tesztelve lettek. Más fázisokra csak a hibátlan művelet végzés után tértem át.

A tesztelést a .csv fájl feldolgozásán kezdtem. Mint fentebb írtam, itt különböző feltételek meglétének vizsgálatával lehetett csak elérni, hogy a fájl minden sora, minden típusa, valamint minden típushoz tartozó adat a megfelelő adatbázis mezőbe kerülhessen. Ez annál is

inkább nehéz feladat volt, mivel a .csv fájl egyáltalán nem hibamentes és voltak olyan adatok, melyekben oda nem való karakterek jelentek meg. Ez nemcsak nehezítette a feldolgozást, hanem a számítógépet is jelentősen leterhelte és ezért alkalmazni kellett a többszálú feldolgozást is. Annak vizsgálata, hogy egy bizonyos adat szám érték-e, egy külön kiértékelő függvénnyel lett ellenőrizve. Ez csak a .csv fájl beható vizsgálata során lett felderítve, manuálisan sorról sorra kiértékelve a fájl szövegét. A tesztelést mindaddig végeztem ezen a modulon, amíg az adatbázisban csak helyesen bejegyzett adatsorok jelentek meg.

A tesztelés másik nagy egysége a grafika megjelenésének vizsgálata. Nem engedhető meg, hogy a grafikonok értéke eltérjen az adatok értékétől, hiszen ekkor nem valós a mérés értékeinek vizsgálata. Tehát a mérés számszerű adatait át kellett alakítani a legpontosabb mértékben koordináta adatokká. Ennek a feladatnak a banki kerekítések használata felel meg a legjobban. *A kód a 18. ábrán látható.*

```
1.  int ertek = 450 - Convert.ToInt32(Math.Round((ht1KHLista[index].kemhatas * 30), 0, MidpointRounding.ToEven));
2.  int ertek = 450 - Convert.ToInt32(Math.Round((ht1VKLista[index].vezetokepesseg1 * 0.150), 0, MidpointRounding.ToEven));
```

18. ábra - Adatok pontos átalakítása koordináta értékekké

Itt a kétféle mérés adat lett átalakítva a grafikon léptékének megfelelően koordináta értékké.

A tesztelés során kiderült, hogy pontos értékeken dolgozik a program és valóban a valós adatokat mutatja.

További tesztelés lett végrehajtva amikor a kiválasztott dátum intervallum alapján a grafikonból kicsúsznak az értékek és görgetésre van szükség.

Ebben az esetben a léptéknek megfelelő értékkel kellett a koordináta pontokat eltolni, vagy jobbra, vagy balra és ebben az esetben is meg kellett egyezni a dátum érték adatának a grafikonon látható adattal.

A tesztelés következő pontja a pozícionáló használatakor megjelenő értékek vizsgálata. Amennyiben a pozícionáló függőleges vonala metszi a grafikon adatpontját akkor megjelenik egy kis négyzet, valamint az adott érték megjelenik a berendezés neve melletti sávban.

Itt, mivel két mérés típusról van szó, mindkét eset vizsgálva lett. A pozícionált adatok nem térhettek el az adatbázisban tárolt adatoktól.

Felület tesztelés

A tesztelés könnyebb része a felület és a rajta található kontrolok (gombok, menük, kiválasztók, stb.) tesztje. Itt az lett vizsgálva, hogy az adott kontrol valóban ellátja e feladatát.

A különböző menük megnyitásával, azaz használatával végig lett tesztelve, hogy a fájl megnyitása és behívása működik, azt a fájlt nyitja meg amelyik ki lett jelölve, és működik-e a funkció ami meggátolja más fájltípusok megnyitás-kísérletét.

Az adattáblák menü minden során végiglépve teszteltem, hogy minden adattábla a megnevezésnek megfelelő adattokat, melyeket már feldolgoztam.

A kiválasztott adatok menüpont tesztelése során egyrészt azt kellett figyelni, hogy a programozásnak megfelelően csak akkor legyen aktív ez a menü, ha már ki lett választva a vizsgálandó dátum intervallum. Ha ez megtörtént a menü aktívvá válik és minden berendezés adattáblája vizsgálható. Itt is minden menüsoron végig kellett menni és már berendezés szinten vizsgálni az adattáblákban levő adatok helyességét.

A súgó funkció hasonlóképp lett vizsgálva, valamint meg kellett felelni az átláthatóság és értelmezhetőség kritériumainak is.

A dátum választók esetében azt kellett vizsgálni, hogy a programozás során beállított védelmi funkció működik-e. Itt egy olyan funkció lett készítve, ami nem engedi meg, hogy a „–tól” érték nagyobb legyen az „–ig” értéknél, hiszen csak olyan időszak vizsgálható ahol a kezdeti dátum kisebb a végső dátumnál. További megszorítás, hogy csak akkor lehet aktív a kiválasztott adatok menü, valamint a kiválaszt gomb, ha a dátum intervallumhoz valóban tartozik eltárolt érték. Különben felesleges a továbblépés engedélyezése. A program minden esetről információs ablakot jelenít meg, evvel segítve a felhasználót. Ha nincs adat, akkor nincs a koordinátában sem semmi egy egyszerű felületnél.

Mivel nincs sehol manuális adatbeviteli mező, így a bevitt adatok helyességét nem kellett vizsgálni.

Kézi navigáció vizsgálata során az lett figyelve, hogy a „Tab” billentyűvel milyen logikus elgondolás során lehet navigálni.

További tesztelés történt az adatbázis kapcsolat és adatelérések terén is. Itt olyan funkciók lettek beépítve, (try – catch – SqlException) amelyek figyelik az adott adatbázis hívási műveletet és bármi váratlan esetben hibaüzenetet adnak. Így nem szakad meg a program futása.

Fejlesztési lehetőségek

Design patternek

Egy program „élete” során folyamatos változásokon megy át. A fejlesztés kezdetén egy jól átgondolt, vagy csak annak tartott koncepció mentén folyik a fejlesztés. Mivel a programozás alapvető koncepciója az OOP programozás, ezért különböző objektumokkal kell dolgozni.

Vannak alapvető programozási tételek, melyeknek minden programnak meg kell felelnie. Azonban a mai kor követelményei folyamatosan fejlődnek és jelenleg az MVC és a Design Pattern programfejlesztés jelenti a modern gondolkodás alapját. Programom fejlesztése során az MVC technológiát vettem alapul, ahol követelmény, hogy a vezérlő (Control) réteg a modul részen keresztül mozgasson adatokat a nézet azaz a View réteg felé és onnan vissza.

Ezt az elvárást úgy valósítottam meg, hogy a felületek (Formok) sehol sem végeznek számítási feladatokat, hanem osztályokon keresztül kommunikálnak az adatbázissal és mozgatják a grafikai elemeket.

A program további fejlesztése tehát egyrészt lehet programozás-technikai fejlesztés a Design Patternek felhasználása irányába, másrészt lehet funkcionális is, ahol a program további funkciókat kaphat a jobb felhasználhatóság érdekében.

Funkcionális fejlesztés

Lehetőségként felmerülhet, hogy a programot nem csak egy adott berendezés csoportra lehessen alkalmazni „beégetett” módon, hanem kibővített módon bármilyen berendezés rugalmasan vizsgálható lehessen, ha a HQ40d műszert éppen máshol használják. Ehhez fontos, hogy az alap adatbázis ne előre deklaráltan jöjjön létre, hanem rugalmasan a műszer alap elnevezéseihez igazodva hozza létre adattábláit, mezőit.

Ehhez igazítható a felületen található elnevezések sora, valamint azok a menük is ahol megjelennek majd az adatok.

Lehet fejleszteni a grafikai megjelenést is. Olyan funkciók valósíthatók meg, melyek a jobb kezelhetőséget, valamint az adatok pontosabb megjelenését szolgálják. Alkalmazható például egy rugalmas nagyítási funkció, vagy a koordináták értékeinek beállítása tetszés szerint. A felület színe, betűk nagysága, stílusa szintén rugalmasan változtatható lehetne.

Számos lehetőség merülhet fel az innovatív gondolkodású emberek fejében. A lehetőségek szinte korlátlanok.

Ábra jegyzék