Meteorológiai Adatbázis

A programunk, egy offline működésre megtervezett meteorológiai adatnaplózási rendszer. Az adatokat többféleképpen tudja feldolgozni a különböző érzékelőktől. Vagy a beimportált adatokat használja, vagy pedig, ha a felhasználó nem rendelkezik bemeneti adattal, akkor az alkalmazás képes véletlenszerűen önmagának generálni környezeti adatokat. Ezeket az adatokat a programunk memóriájában tároljuk.
Képesek vagyunk szűrni az adatok közt, valamint különféle statisztikai elemzések elvégzésre is alkalmas az alkalmazás.
Fontosnak tartottuk a program megvalósítása közben, hogy meghatározzuk a szerepköröket az adatbázisban. Elsődlegesen a rendszert a Felhasználó használja, aki a tárolt adatokat nézni, szűrni és statisztikákat kinyerni tud. Mellette létezik az Admin, aki a Felhasználó szerepkörein felül képes törölni, valamint módosítani az adatok közt (pl. mértékegységeket).

A tervezetünkben öt főbb rétegről beszélünk, amikor az alkalmazást kiépítettük:

1. Model
2. Services
3. Core
4. UI
5. Interfészek

Model réteg:

A réteg központi egysége az ***Adat*** osztályban valósul meg.
Ez az osztály reprezentálja egy időbélyeggel ellátott mérési pontot. Ezek az adatok tartalmaznak egy generált vagy mért mérési értéket (**Ertek**), annak mértékegységét (**MertEgyseg**), a mérés időpontját (**Ido**) és a mérés eredetét (**AdatEredet Eredet**), amelyet az ***AdatEredet*** enum típusú segédosztályunkból kapunk meg, hogy egy adott adat éppen generált vagy importált. Opcionálisan pedig kezeli, hogy milyen nevű szenzoron történt a mérés (**SzenzorNev**) és hogy milyen kategóriába tartozik (**Kategoria**).

Az ***AdatHalmaz*** osztály tárolja el a betölötött adatokat egy *List<Adat>* nevű listában. Ezen felül pedig hozzáadni és törölni lehet több vagy minden adatot az adatbázisból. (**HozzaadTobb, TorolMindent**)

Ebben a rétegben még van két enum típusú segédosztályunk, a ***Role***, ahol a kívánt szerepköröket tudjuk meghatározni (felhasználó és/vagy admin) és a ***MertEgyseg*** osztályok, ahol pedig alapértelmezett mértékegységet tárol.

Services réteg:

Ez a réteg fogja megvalósítani a konkrét funkciókat, mint például az importálás kérdését, hogy a felhasználó az adatok közt tudjon szűrni vagy hogy a program maga generáljon környezeti adatokat a rendszerhez stb.

Az importálás kérdését az ***ImportaloSzolgaltatas*** osztálya dönti el, hogy a felhasználó milyen típusú fájlt akar beolvastatni a programmal. A program megnézni a fájl kiterjesztését,

amelyek kizárólagosan csak CSV, JSON vagy XML formátumban működnek, és meghívja az annak megfelelő importálóját (***CSVImportalo***, ***JSONImportalo*** vagy ***XMLImportalo***).

CSVImportalo fájlbeolvasó osztály pontosvesszőkkel elválasztott fájlokat olvas be. Ha hibásnak lát egy sort, azt kihagyja, de maga a program nem fog leállni, csak számolja hogy hány sort tudott beolvasni, amit a végén a felhasználók számára ki is ír. Ha nem valós CSV fájlt próbálunk vele beolvastatni, akkor szintén jelez a felhasználó fele, hogy nem található ilyen CSV fájl.

JSONImportaló csak JSON fájlokat képes beolvasni. Ezen osztályon belül egy segédosztály is megtalálható (***BiztonsagosDoubleKonverter***), ami arra szolgál, hogy rossz adat esetén ne álljon le a program. Az osztály itt is figyelmezteti a felhasználót, amennyiben nem találja az importálandó fájlt.

Végezetül az ***XMLImportalo*** osztály csak XML típusú fájlokat képes importálni. A felhasználót, mint korábban a két másik importáló osztályunk, szintén figyelmezteti nem létező fájlok importálása esetén.

A ***Generalo*** osztályunk véletlenszerű tesztadatokat képes generálni. Ehhez meg kell adni hogy hány adatot szeretnénk, milyen annak az értéktartománya és milyen időintervallumban és mértékegységben generálja le az adatot. Ő egy megadott kategória lista alapján pedig legenerálja a tesztadatokat nekünk.

Az ***MemoriaAdatforras*** osztály egy darab ***AdatHalmaz*** objektumot tárol és kezel.

Adatokat szűrni különféle feltételek alapján a ***Szuro*** osztályban kerül sor. A feltételek közt nekünk felhasználóknak meg kell adni, hogy milyen időintervallumban szeretnénk szűrni és mi(k) legyen(ek) a minimum és maximum értéke(i) az adatoknak, amiket szűrnénk.

StatisztikaElemzo osztály végzi a statisztikai számításokat. Ezek lehetnek minimum, maximum értékek, darabszám, átlag, napi lebontású statisztika, amely minden napra külön bontja le.

Végül pedig a ***MertekegysegKezelo*** osztályunk kezeli az alapértelmezett mértékegységet, amely alapból C°-on méri az adatokat. Itt történik az, amikor az Admin meg szeretné változtatni az adatok mértékegységét valamilyen új, másik mértékegységre.

Core réteg:

A réteg központi vezérlője az ***AlkalmazasVezerlo*** osztályunk, amely irányítja az egész alkalmazást. Főbb vezérlési műveletei közé tartozik a **Futtat(), Importalas(), Generalas(), Megtekintes(), Szures(), Statisztika(), ModositMertekegyseg()** és a **TorolMindent()** függvények.

Futtat() függvényben van a programunk fő ciklusa. Ez a függvény jeleníti meg a menüt, ahol bekéri a felhasználótól, hogy milyen feladatokat szeretne csinálni és mindent végre is hajt.

Importalas() függvény kéri be a felhasználótól a fájl elérési útját, ellenőrzi és meghívja a megfelelő importálóját, majd az eredményt az adathalmazba hozzáadja.

Generalas() függvényünk kéri be a megfelelő paramétereket, amennyiben mi nem rendelkezünk valós mért adatokkal, amelyet beimportálhatunk. Ellenőrzi a megadott paramétereket majd legenerálja az adatokat.

Megtekintes() függvény, lapozható formában jeleníti meg a felhasználó számára az adatot. Ezek közt lehet előre-hátra lapozgatni vagy pedig visszalépni a menübe. Egyszerre 10 adatot jelenít meg/oldal.

Szures() függvény fogja a felhasználótól bekérni a szűrési feltételeket, majd elvégzi a kért szűrést és megjeleníti az adatokat.

Statisztika() függvény megjeleníti az összesített statisztikákat.

ModositMertekegyseg() függvény használja az Adminnak a funkcióját. Ezt a Felhasználó szerepkör nem éri el, de az Admin képes ezzel a függvénnyel módosítani az adatokat, mértékegységeket stb.

TorolMindent() függvény pedig szintén egy Adminhoz tartozó függvény. Ezzel fogja tudni törölni az összes eddig betöltött adatot.

Validator osztály egy egyszerű segédosztály. Ellenőrzi, hogy nem üresek vagy null-ok a stringek.

Végül a ***ParancsKezelo*** segédosztályunk felel az inputok bekéréséhez.

UI réteg:

Ez a réteg fogja a felhasználó számára megjeleníteni konzolosan az adatokat és a kiírásokat.

Megjelenito osztály jeleníti meg a konzolon a kiírásokat és a program menüpontjait. Ehhez használjuk a **Menu(), Szoveg(), Siker(), Hiba(), MegjelenitListat()** és **MegjelenitNapiStat()** függvényeket.

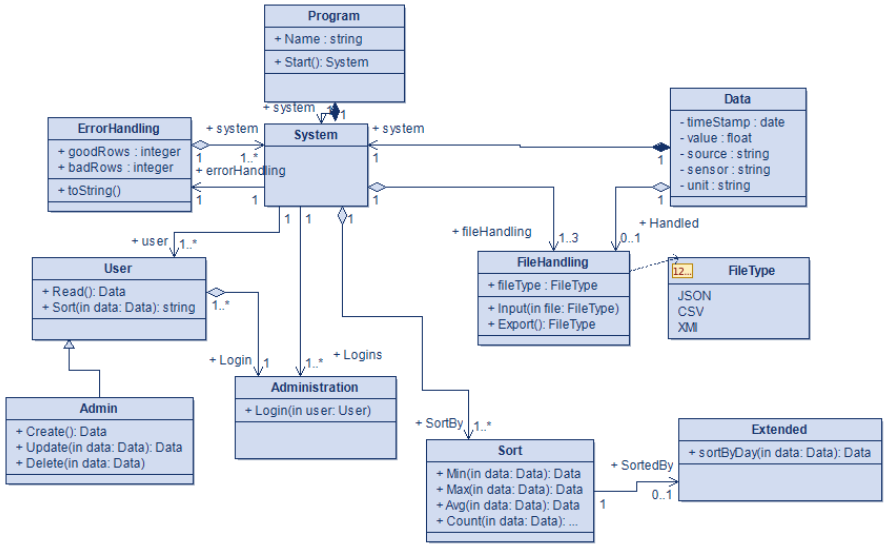
A ***MenuRenderer*** osztály pedig kirajzolja a felhasználónak a menüt.

Interfész réteg:

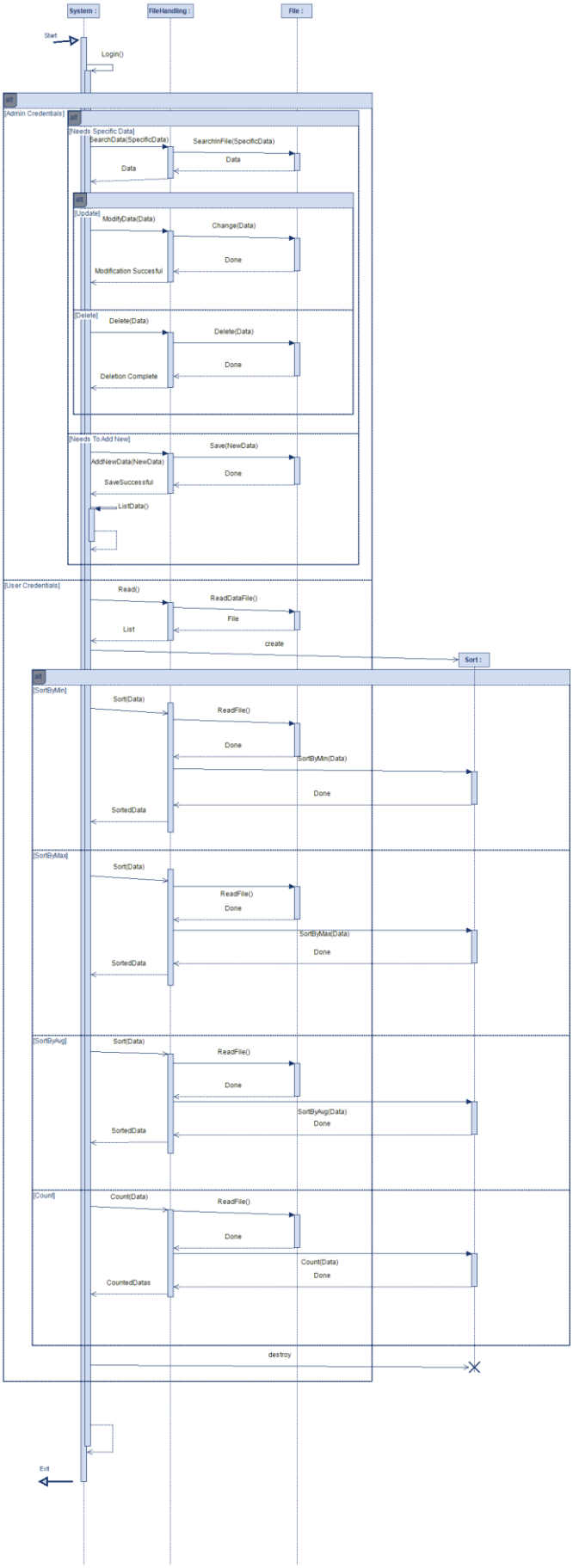
A rétegeken felül még interfészeket is hozzáadtunk, hogy a tesztelések egyszerűbb legyenek, illetve az implementációt is meg tudja könnyíteni. Ezeken keresztül kommunikálnak a modulok egymással. Könnyen bővíthetőek.

A ***Program.cs*** elindításakor meg tudjuk határozni, hogy Admin vagy Felhasználóként szeretne majd a felhasználó belépni. Ilyenkor a menü megjelenik a fő ciklusban és választhatunk 0-7 között valamilyen menüpontot, attól függően, hogy mit szeretnénk csinálni az alkalamazáson belül. Ezeket a program végrehajtja.
A hibakezelést úgy oldja meg, hogy nem áll de, csak kiírja a megfelelő hibakódot a felhasználó számára.

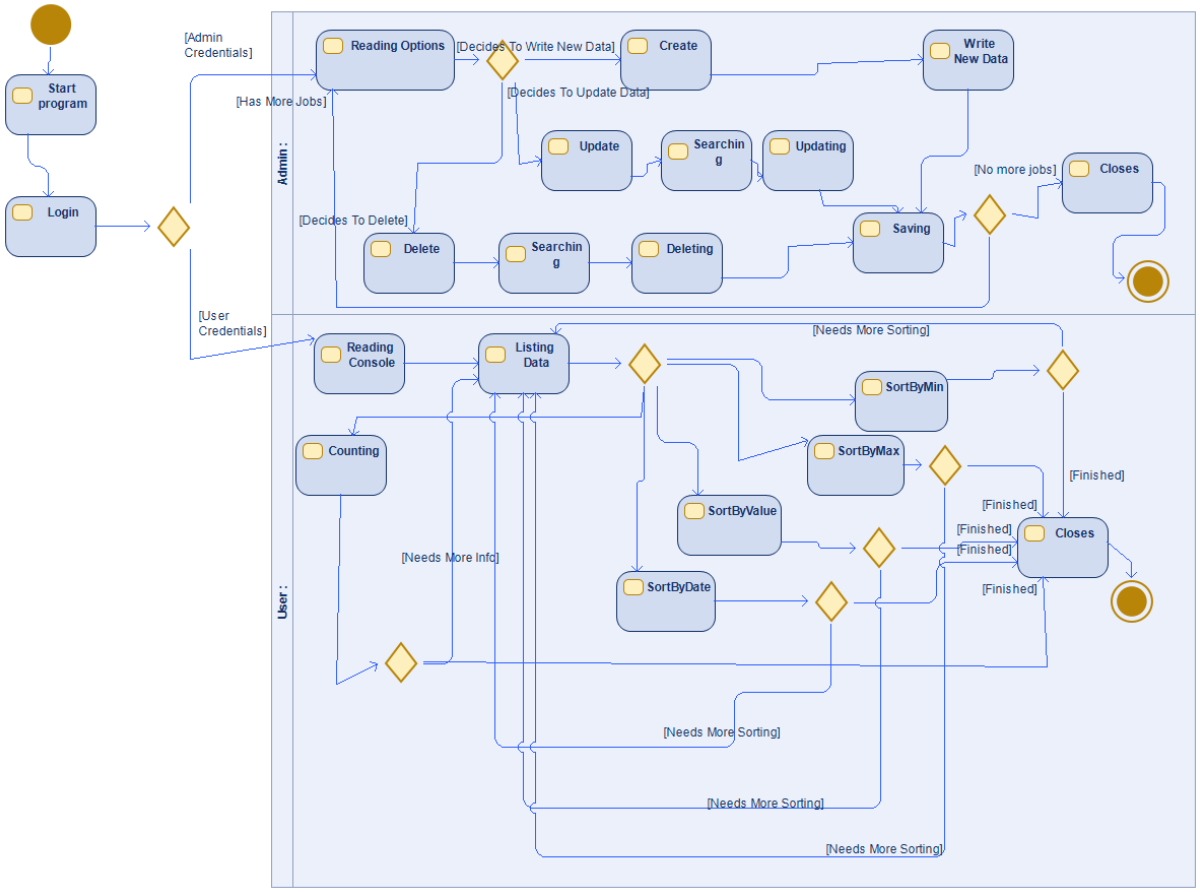
Az alkalmazás osztálydiagramja



Az alkalmazásban egy tipikus lefutást rögzítő szekvencia diagram



Az alkalmazás áttekintő jellegű Aktivitás diagramja



Modulok leírása

A program írásakor többféle modult alkalmaztunk.

Voltak **adatkezelő modulok**, amiknek a céljuk az volt, hogy az adatokat eltárolják a program futása során. Az adatok főbb funkciói közé tartoztak a hozzáadások, törlések, módosítások, illetve az ellenőrzés, hogy egyáltalán létezik az inputban valami adat?

Ezt a modult minden más modul akkor használta, amikor adatokat szerettek volna elérni. Ezen felül használja még az ***Adat*** osztály az egyes mérési pontoknak eltárolására.

Használtunk **importáló modulokat** is egyes osztályokban (pl. ***XMLImportáló*** vagy ***ImportaloSzolgaltatas***), hogy külső fájlokból tudjunk beolvasni adatokat. A modult meghívtuk pl. az ***Adat*** osztályban, amikor el akartuk dönteni, hogy az adatokat importáljuk vagy generáljuk.

Generáló modult azért hoztuk létre, hogy véletlenszerű tesztadatokat tudjunk generáltatni abban az esetben, ha nincs semmilyen valós adatunk amit importálhatnánk. Ezt a **Vezérlő modul** tudja meghívni és az Adatkezelő modulba generálja az adatot. Szintén az ***Adat*** osztályban hívtuk elő, amikor generált adatokkal akartunk dolgozni.

Szűrő modulokkal az adatok között tudunk szűrni valamilyen paraméternek megfelelően, legyen az időintervallum szerint, vagy érték szerint. Ezt a **vezérlő modul** hívja meg, majd az **adatkezelő modulból** olvassa be az adatokat. Az eredeti adathalmaz változatlan marad, abban nem módosít. A szűrések csak új listát írnak ki a felhasználónak, ezáltal a paraméterek közt többször lehet szűrni, anélkül, hogy az eredeti adatokat megváltoztatnánk.

Statisztikai modulban az adatokon végzünk különféle statisztikai számításokat értékek szerint akár napi bontású statisztikai adatként. Ezt is a **vezérlő modul** hívja elő és ugyanúgy, mint a **szűrő modulnál**, csak olvassa az eredeti adathalmazt és új listát ír ki a felhasználónak, bármiféle változtatás nélkül az eredeti adathalmazból.

Jogosultság kezelő modulban ellenőrizzük, hogy a felhasználónak joga van-e egy adott művelethez, ehhez a ***Role*** enum segédosztályt használja a szerepköröknek a megkülönböztetéséhez. Ezt is a **vezérlő modul** hívja meg.

Megjelenítő modul felel a konzolos felületért, hogy kiírhassa a felhasználónak a különféle outputokat. A **vezérlő modul** hívja meg az összes kiíráshoz és kirajzoláshoz, amihez a ***MenuRenderer***-t használja. Nem módosít, csak kiír.

Vezérlő modul a program agya, innen irányítódik az egész alkalmazás, ezért hívja meg az összes többi modult attól függően, mit szeretnénk az alkalmazáson belül elvégezni. Ez a modul kezeli az inputokat a felhasználótól és ellenőrzi, hogy nem üresek-e a stringek. Az egész programot ez a modul köti össze. Ő futtat, ő szűr, ő jelenít meg, ő felel a jogosultságokért.