Debreceni Egyetem
Informatikai Kar

GRAFIKUS ADATBÁZISKEZELŐ, ÉS ADATFELDOLGOZÓ SZOFTVER NAPFIZIKAI ALKALMAZÁSHOZ

Témavezető:
Dr. Tóth László
egyetemi adjunktus

Készítették:
Molnár Csaba
Siteri László
Weinbach Gábor Áron
Mérnökinformatikus hallgatók

Debrecen
2011

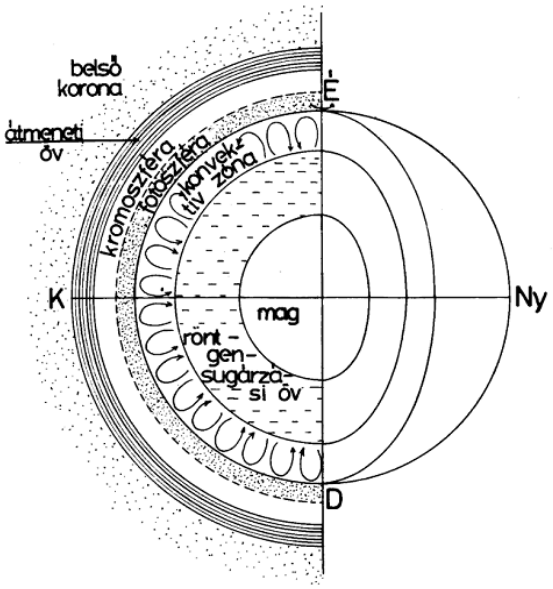
Tartalomjegyzék

1. Bevezetés

A földi élet szempontjából a Nap nélkülözhetetlen égitest. A Föld felszínén megjelenő energiának (hő, szél, víz, stb.) majdnem kizárólagos forrása. A Napból érkező részecske-, és elektromágneses sugárzás energiája hatással van a földi mágneses térre és légkörre. Mozgatórugója az időjárási jelenségeknek. Energiáját hasznosítva építik fel szervezetüket a növények, majd a bennük eltárolt napenergiát használják fel a növényevő állatok, a ragadozók és az emberek. A világ mozgatórugóját jelentő fosszilis tüzelőanyagok is ezekből a növényekből és így a napból nyert energiából jöttek létre. A Nap működésének rövid távú változásai (napkitörések, napszél) hatással vannak a Föld mágneses mezejére és a fejlett infrastruktúrára (rádió, navigáció, finom elektronikai berendezések).

Naprendszerünk központi csillaga. Tömege 1,989 * 10^{30} kg, térfogata 1,4122 * 10^{18} km3, sugara 696 000 km, effektív hőmérséklete 5780 K. Földől való távolsága a Föld ellipszis pályán való keringése miatt 1,521 * 10^{8} km és 1,471 * 10^{8} km között változik. A Nap tömegének jelentős része a középpontja körül, a legbelső 1,5% térfogatban sűrűsödik, mely a sugár 25% -ig terjed. Itt óriási nyomáson (3.4 * 10^{16} Pa) és hőmérsékleten (15 * 10^{6} °K) folynak a fúziós reakciók. Ennek során a hidrogén atomok hélium magokká egyesülnek. Két hidrogén atom hélium atommá való egyesülése során fellépő tömegkülönbség az $E = m*c^2$ képletnek megfelelően energia formájában felszabadul. A Nap az energia kibocsájtás miatt folyamatosan veszít tömegéből. Jelenlegi tudásunk szerint a kora hozzávetőlegesen 4,5 milliárd év és kb. 5 milliárd év múlva válik majd vörös óriássá.[1,2,3]

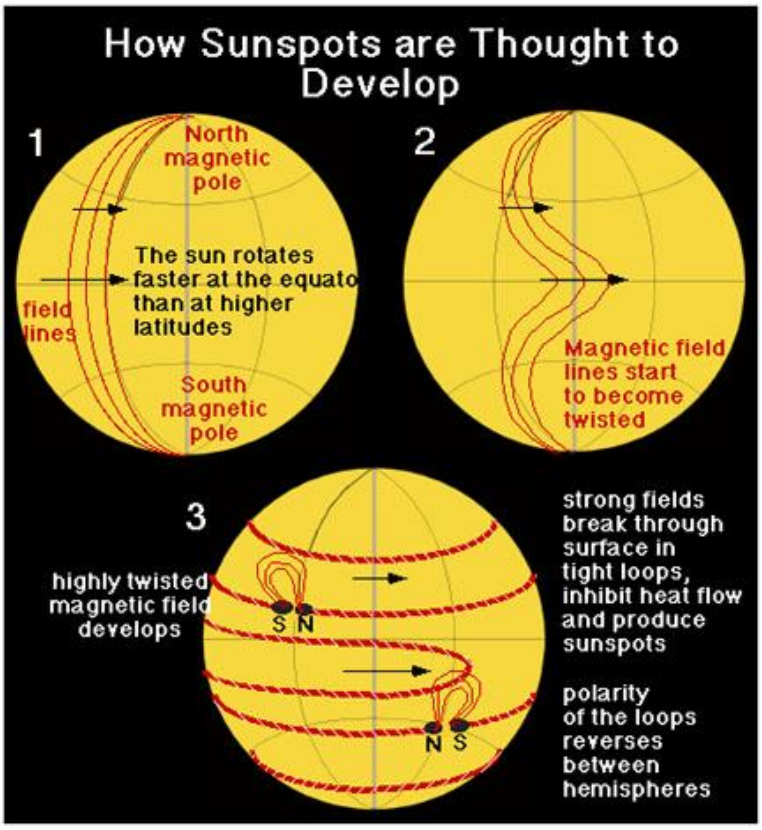
A Nap belsejében történő folyamatok nem figyelhetőek meg közvetlenül, viszont a felszínen játszódó jelenségekből következtetni tudunk azokra. A Nap sugárzási zónája a mag külső részétől a konvektív zóna alatt elhelyezkedő átmeneti rétegig terjed.



1. ábra. A Nap vázlatos szerkezete [1].

Kifelé haladva a röntgensugárzási öv (1. ábra) következik, ami a Nap sugarának közel 70%-áig tart.

Mai ismereteink szerint ennek határán található az úgynevezett napdinamó, ahol a Nap mágneses tere keletkezik. A Nap anyaga plazma halmazállapotú, ami azt jelenti, hogy ionizált atomokból és szabad elektronokból áll. Ezek áramlása mágneses teret hoz létre, amely visszahat a mozgásukra és fordítva, vagyis az anyag és a benne foglalt mágneses tér egymástól elválaszthatatlanok. Ezt „befagyási” jelenségnek nevezzük. További érdekesség, hogy mivel a Nap egy gázgömb, ezért nem merev testként forog, amit differenciális rotációnak neveznek. Ennek és a befagyási jelenségnek köszönhetően a kezdetben poloidális mágneses tér a 11 éves ciklus alatt feltekeredik és toroidálissá válik. A Nap tengely körüli forgásának szögsebessége az egyenlítő felé haladva nő. A ciklus végén a Nap poloidális mágneses tere újra felépül, de ellentétes irányban. Egy-egy ilyen időszak alkotja a 22 éves mágneses ciklust (2. ábra).



2. ábra. A Nap mágneses terének feltekeredése a differenciális rotáció következtében és a napfoltok kialakulása [4].

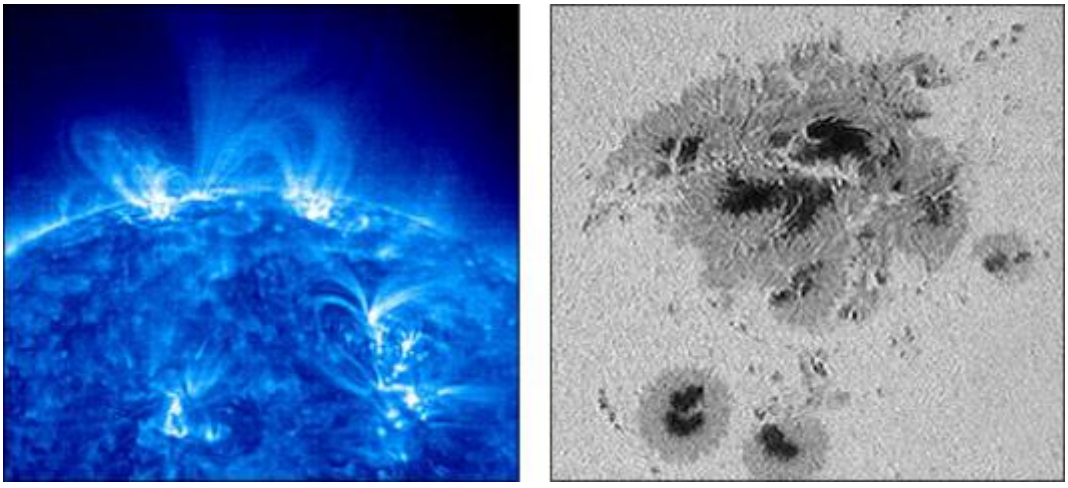
A röntgensugárzási öv fölött a konvektív, vagy más néven hőáramlási zóna (1. ábra), majd a néhány 100 km vastag 5800 ˚K hőmérsékletű fotoszféra következik (1. ábra), ahonnan a látható tartományba eső sugárzás származik. Itt figyelhetők meg a környezetüknél alacsonyabb hőmérsékletű és ezért sötétebbnek tűnő napfolt párok (1. ábra), melyek a konvektív zóna aljáról felfelé induló mágneses erővonal hurkok fotoszférabeli metszetei (2. ábra,). Egy átlagos folt átmérője sokszorosan meghaladja a Föld méretét, és élettartama 1 naptól több hónapig is terjedhet. Szerkezetileg két zónára különíthetőek el, a belső sötétebb kb. 4000 ˚K-os umbrára és a külső világosabb 5500 ˚K-os penumbrára. A napfoltok számának 11 éves periódusú változása a legrégebb idők óta megfigyelt naptevékenységi jelenség. A differenciális rotáció következtében előálló fotoszférabeli szögsebesség (W) a naprajzi szélesség (B) függvényében a következőképpen írható le:

$$\Omega(\mathbf{B}) = \mathbf{14.713} - \mathbf{2.396} * \sin^2(B) - \mathbf{1.787} * sin^4(B) \left[\frac{fok}{nap}\right] \quad [5]$$

A fotoszféra és a korona közötti tartomány a kromoszféra. Vastagsága kb. 10 ezer km, hőmérséklete 4300 ˚K-ről indul, és kifele haladva növekszik.

Felső része az átmeneti réteg, amelyben a hőmérséklet 10 ezer ˚K-ről 1 millió ˚K-re nő. Sűrűsége ritka, anyageloszlása a nagy hőmérséklet-különbségek miatt egyenetlen.

Ezt követi a korona, amely a naprendszer határáig tart. Szerkezete, a Napból kiemelkedő mágneses erővonal hurkoknak köszönhetően (3. ábra), ugyancsak egyenetlen és változékony. A röntgen tartományban készült felvételeken sötét részeket is megfigyelhetünk, melyek a környezetüknél hidegebb, alacsonyabb sűrűségű és nyitott mágneses térrel rendelkező területek. A Nap nyugalmi állapotában ezen úgynevezett koronalyukakból származik a napszél jelentős része, ami a korona anyagának 400-800 km/s sebességgel történő kifelé irányuló áramlása.



3. ábra. A korona képe röntgen tartományban. Megfigyelhetők a mágneses hurkok és a tövüknél a napfoltok (SOHO űrszonda felvétele [6]). (bal) Egy összetett napfolt hátterében a granulációs szerkezet [7] (jobb).

Földünk a Napból szüntelenül áramló, különböző energiájú elektromágneses sugárzások (pl. RTG, -UV, IR, rádióhullámok) és elemi részecskék áradatában, a napszélben kering. A térben és időben változó sebességű és intenzitású napszél a felelős a földi magnetoszféra sajátos alakjáért és egyben ez utóbbi védi meg a földi életet annak pusztító hatásától. A napszél hatását a Föld mágneses terére a felszínen mért térerő nagyságának és irányának megváltozásából érzékeljük. Ez a változás olyan nagymértékű is lehet, hogy a pólusok közelében az iránytűk össze-vissza forognak, amit mágneses viharnak nevezzük. Ilyenkor az is előfordulhat, hogy a mágneses tér változásainak hatására óriási áramok indukálódnak a kőolaj-, és elektromos távvezetékekben.

A napszél inhomogenitásai a Nap aktív vidékeiben gyökereznek, így természetesen összefüggésben van a napciklussal is. Naptevékenységi maximum időszakban több és erősebb, míg minimum időszakban kevesebb és gyengébb mágneses vihar észlelhető.

A mágneses erővonalak tölcsére mentén, még a kisebb energiájú részecskék is lejuthatnak a sarki ionoszférába, sőt az alsóbb légrétegekbe is. Ezeket a tartományokat, sarki sapkáknak (Polar Cusp Area, PCA), másképpen sarki fény zónáknak nevezzük, és kb. a 67. geomágneses szélességek mentén helyezkednek el, és itt alakul ki a sarki fény jelensége. Előfordulhat, hogy az óriási számban lejutó részecskék megnövelik a sarki sapkák ionizációs fokát, amely következtében az Arktisz és az Anktartisz térségében a rövidhullámú rádiózás teljesen lehetetlené válik.

Bár az elektromágneses sugárzások által a Földre szállított energia mennyisége jóval meghaladja a napszél által szállítottat, az utóbbi mégis jóval nagyobb jelentőséggel bír, aminek két fő oka van. Az egyik, hogy az elektromágneses sugárzások összességének napciklus alatti intenzitásváltozása csak mintegy 0.1%, ellentétben a részecske sugárzások 100% intenzitásváltozásával. A másik, hogy míg az elektromágneses sugárzások eloszlása egyenletes a Föld felszínén, addig a részecskék energiájának nagy része viszont csak egy szűk területen (PCA) nyelődik el, megnövelve ezzel az ott lévő gázok hőmérsékletét, ionizáltsági-, és gerjesztési fokát, valamint a disszociált molekulák számát, ami által anomáliákat okoznak a légkörben.

Az így kialakuló hőmérsékleti eltérések előfeltételei a ciklonok kialakulásának, a szabad ionok pedig, mint kondenzációs magvak elősegítik a felhőképződést és jó alapot szolgáltatnak a zivatar-elektromos jelenségek felerősödéséhez. A szabad gyökök fontos szerepet vállalnak a hő háztartásban, az ultraibolya sugárzások által szétbontott oxigén molekulák pedig az ózon

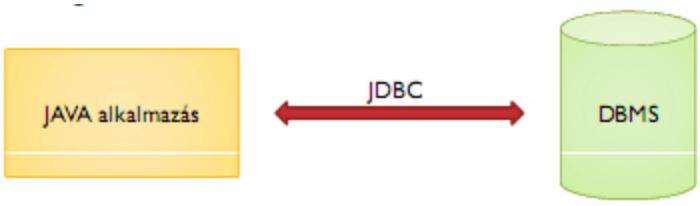
képződésében. Mindebből tehát az következik, hogy a Föld azon féltekéjének időjárása, ahová egy adott napciklus során a napszél részecskéi becsatolódnak az változékonyabb lesz. Ugyanakkor a másik féltekén egyhangúbb időjárás alakul ki, egészen a rá következő napciklusig, amikor is megcserélődnek a viszonyok.

2. Adatok forrása

A National Oceanic and Atmospheric Administration (NOAA) [8] az amerikai Kereskedelmi Minisztérium alá tartozó, tudományos kutatásokat végző ügynökség, mely az óceánokat és a légkört kutatja. A NOAA készít jelentéseket a valós idejű naptevékenységről és a Föld mágneses tevékenységéről, valamint a Nap- Föld kapcsolat kutatásáról, előre jelezve a naptevékenységben és a földi elektromágneses viszonyokban beálló eltéréseket.
Greenwich Photoheliographic Results (GPHR) [9], a fotoszféra-adatbázisok klasszikus tagja, amelynek 1878 és 1976 között évenként megjelent egy-egy kötete tartalmazta a foltok és foltcsoportok pozíció és területadatait minden napra. A sorozat hatása rendkívül jelentősnek bizonyult, reá alapozva nagy mennyiségű publikáció született.

3. Adatbázis- szerver, kliens

A MySQL szerver 5.1-es verzióját használtuk, mely ingyenes, professzionális adattárolást biztosít. Kétrétegű adatbázis-modellt használva épül fel a JAVA SE és az adatbázis között a kapcsolat. Ezek között a kapcsolatot a JDBC valósítja meg, melyet a forráskódba egy mysql connector driver használatával implementáltunk.



4. ábra Kétrétegű adatbázismodell [10].

Az adatbázishoz hozzáférést biztosító ingyenes, SQLyog Community Edition MySQL klienst használtuk, mellyel grafikus felületen keresztül teljes körűen kezelhető a szerveren futó adatbázis.

4. Szerkezet kialakítása

Az adatbázist és a táblát a fent említett SQLyog kliens segítségével hoztuk létre. A tábla attribútumait a forrásállományoknak megfelelő tulajdonságok alapján készítettük el. A tábla attribútumai és jelentésük:

- DATE:A foltcsoport észlelésének dátuma.
- ORA:A foltcsoport észlelésének időpontja. Az értékek 0-1000 közé eső egész számok. Az 500 jelenti a déli 12 órát.
- NUM:Az észlelt foltcsoport csoportszáma.
- SUFGN:A greenwichi csoportazonosítóhoz tartozó szám. A korábbi észlelések között előfordult, hogy két különböző foltot ugyanazzal a csoportazonosítóval láttak el. Ez kiegészítő szám különbözteti meg ezeket a csoportokat.
- MTW:Mount Wilson-i mágneses osztályozás. 0-5 intervallumban vehet fel értékeket. A dolgozat készítése során nem használtuk fel ezeket az adatokat, csak tároltuk az adatbázisban.
- GRTP:Greenwich-i csoportazonosítóhoz tartozó típus. 0-9-ig vehet fel értékeket. Felhasználás szempontjából ugyanaz jellemző rá, mint az előző oszlopra.
- OBSUMB:Észlelt foltcsoport umbrájának mért területe.
- OBSWH:Észlelt foltcsoport whole-jának mért területe.
- CORRUMB:Észlelt foltcsoport umbrájának korrigált területe.
- CORRWH:Észlelt foltcsoport whole-jának korrigált területe.
- DIST:Radiánban megadott érték, mely a napfelszín középpontjától számított távolság.
- ANG:Polárszög mely szögben megadott értékeket tartalmaz 0-360 fokig.
- LNG:(Longitud)A Carrington hosszúsági fokokat tartalmazó oszlop. Az L0-tól vett távolsága a foltcsoportnak. L0 – A heliografikus koordinátarendszer kezdő meridiánjának azt tekintjük, amely 1850. január 1.-jén greenwichi középdélben haladt

át a Nap egyenlítőjének felszálló csomóján. Ezt a hosszúságot a Nap forgásának irányában mérik. Értéke 0-360° terjed.

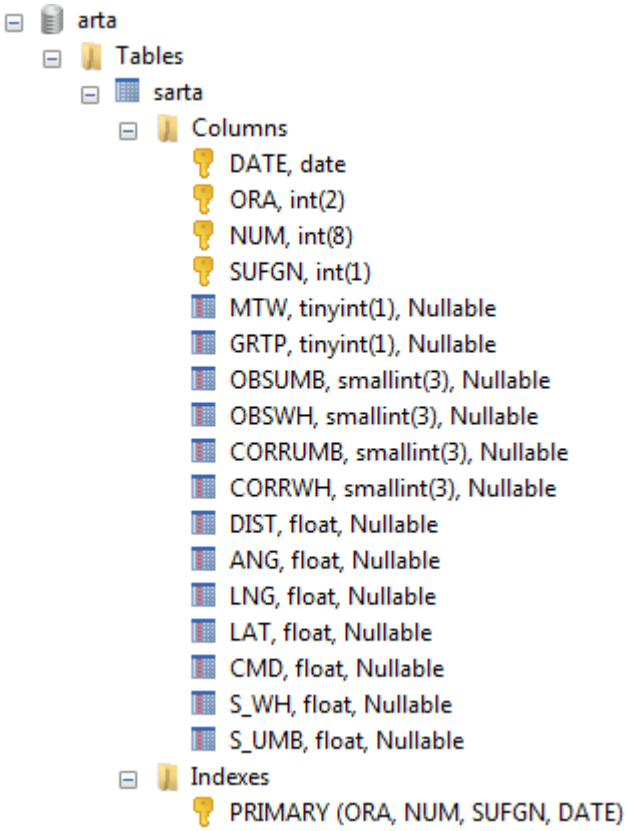
- LAT:A Carrington szélességeket tartalmazó oszlop. Ha ez az érték negatív, akkor a déli, ha pozitív, akkor az északi féltekére eső szélességi körről beszélünk.
- CMD:Centrál Meridián Távolsága, mely a foltcsoport a Centrál mereidiántól vett távolságát adja meg. A Centrál meridián alatt a látható napfelszín középső hosszúsági fokát értjük.

A DATE, ORA, NUM, SUFGN attribútumok együttesen alkotják a tábla elsődleges kulcsát. Ezen tulajdonságok egyértelműen meghatároznak egy foltcsoportot.

Ezután az adatbázist kiegészítettük két oszloppal (S_WH, S_UMB), melyek az UMBRA-hoz és WHOLE-hoz tartozó sugárértékeket tartalmazzák. Ezeket az adatokat közvetlenül nem tartalmazzák a NOAA szerverén hozzáférhető fájlok. A szoftver számolja ki és adja hozzá, a fájlok feltöltése során. Minden sor felvitele után a program kiszámolja az észlelt umbra és whole adatokból a hozzá tartozó sugár értékeket és feltölti az adatbázisba a következő metódus segítségével:

```
int sugar(double r) {
int R = 225;
        double szam = 2 * (((4 * R) / Math.PI) * ((4 * R) / Math.PI));
int sugar = (int) Math.round((Math.sqrt(szam * r / 1000000)));
return sugar;
}
```

Az „r” paraméter az adatbázisból kiolvasott aktuális foltcsoporthoz tartozó összesített terület.



5. ábra A végleges MySQL tábla mezői.

5 JAVA alkalmazás

5.1 JAVA, Integrált Fejlesztői Környezet

Az alkalmazás fejlesztéséhez a Sun Microsystems által kifejlesztett Java SE 6 Update 24 JDK-t használtuk. A Java egy platformfüggetlen programozási nyelv, mely megvalósítja az objektumorientált paradigmákat. Platformfüggetlenségét a Java Virtuális Gép (JVM) által éri el. A JVM byte-kód futtatását teszi lehetővé, mely a java kód fordításakor jön létre. Nyílt forráskódú szabad szoftver. Az integrált fejlesztői környezet (IDE) tekintetében a Netbeans éppen legfrissebb verziójára esett a választás. Jelenleg ez most a Netbeans 6.9.2.

5.2 Grafikus felhasználói Felület

A GUI (Graphical User Interface) komponensekből áll. Téglalap alakú képernyőfelületek, amelyek tulajdonságokkal rendelkeznek és előre megadott szabályok szerint reagálnak különböző eseményekre. Vannak az ún. konténer-komponensek: egy komponensre újabb komponens tehető. Alapvető grafikus elemek, pl. a gombok, listák, legördülő menük, táblázatok stb. A GUI programozására a Java kétféle osztálygyűjteményt (keretrendszert) ad, az AWT-t és a Swinget. Az AWT a régebbi rendszer, amit már nem fejlesztenek tovább. A Java tervezői mindkettőben arra törekedtek, hogy a felhasználói interfész platform független legyen. A két osztálygyűjtemény jellemzői:

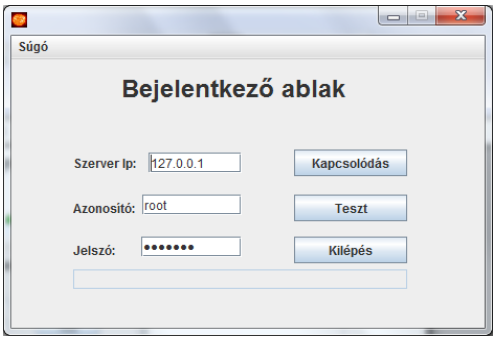
AWT (Abstract Window Toolkit, absztrakt ablakozó eszköztár) a korábban kifejlesztett osztálykönyvtár. Úgy tervezték, hogy az operációs rendszer elemeihez hasonlóak legyenek. Használatuk kissé nehézkes, mivel az AWT osztályok közvetítő feladatot látnak el a felhasználó programja és az operációs rendszer között. Az AWT osztályok használata kevésbé kényelmes, de a futásidejük kisebb.

A Swing az újabb osztálykönyvtár, amely sokkal gazdagabb, mint az AWT, és az előbb említett korlátozás sem létezik. Természetesen a Swing osztály vezérlői sokkal többet tudnak és felépítésük logikusabb. A Swing összetevőket csak a legújabb böngészők támogatják és az áttérés az AWT osztályokról a Swing osztályokra nem bonyolult.

Ahogy a Swing az AWT komponenseit fejleszti tovább, úgy bővítik az AWT grafikus lehetőségeit a Java2D osztályok. Az AWT, a Swing és a Java 2D osztályokat együtt Java Alaposztályoknak (Java Foundation Classes , JFC) szokták nevezni.

5.3 A bejelentkező ablak

A bejelentkező ablak egy javax.swing.JFrame osztály kiterjesztés. Célja a belépéshez szükséges adatok lekérdezése melyet a program működése során adatbázis csatlakozáshoz használ.



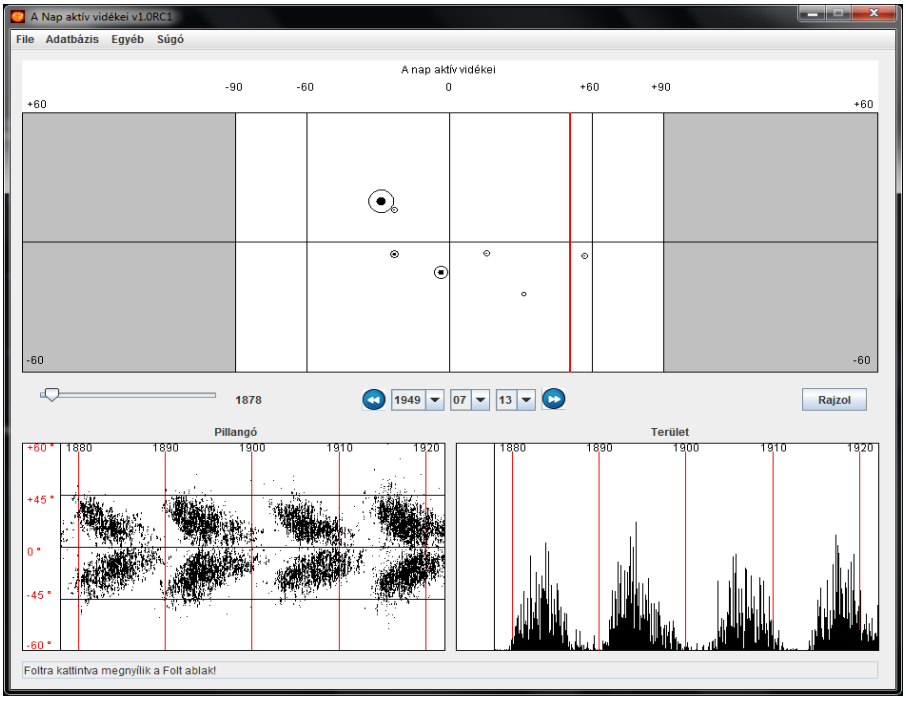
6. ábra Bejelentkező ablak.

Az első szövegmezőben a Mysql szerver hálózati címét adhatjuk meg, ahol mind az IP mind a hostname megadható. Ezután az azonosítót és a jelszót kell megadni. A Teszt gomb segítségével ellenőrizhetjük, hogy helyesek-e az adatok, illetve elérhető-e az adatbázis szerver.

```
private void TestActionPerformed(java.awt.event.ActionEvent evt){
        try {
                Lekerdez();
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                java.sql.Connection conn = DriverManager.getConnection(Acc[0], Acc[1],
                Acc[2]);
                java.sql.Statement s = conn.createStatement();
                conn.close();
        }catch (Exception e) {
                jTextField3.setDisabledTextColor(Color.RED);
                jTextField3.setText("Sikertelen kapcsolódás!!!");
                return;
        }
        jTextField3.setDisabledTextColor(Color.GREEN);
        jTextField3.setText("Kapcsolódás sikeres!!!");
}
```

Sikeres, illetve sikertelen kapcsolódás esetén az ablak alján kapunk erről egy értesítő üzenetet. A kilépés gombbal azonnal kiléphetünk az alkalmazásból. A Kapcsolódás gombra kattintva pedig tovább léphetünk a Főablakhoz. Ha egy új felhasználó indítja el a programot, akkor a menüsorban elérheti a súgót melyben segítséget kaphat.

5.4 Főablak



7. ábra Főablak.

5.4.1.1 A festővászon (Canvas) osztály

Az AWT Canvas („festővászon”) nevű grafikus komponense lehetőséget ad arra, hogy a felhasználói felület egy részén rajzolni lehessen (Swing megfelelője nincs a Canvas-nak). Egy rajz elkészítéséhez örökölni kell a Canvas osztályt, majd felül kell definiálni a paint (Graphics g) metódusát azzal a kóddal, amely tartalmazza a rajzolási feladatokat (pl.: drawLine(…), stb.). Rajzolni a paraméterként kapott Graphics típusú objektumra lehet a Graphics osztály által biztosított metódusokkal (pl.: drawLine(…), stb.).

Alapesetben a paint(Graphics g) meghívása az update(Graphics g) metóduson keresztül történik. Az update(Graphics g)-t a JVM akkor hívja meg, amikor úgy érzi, hogy szükséges újrarajzolni a komponenst. Ha indirekt módon szeretnénk újrarajzoltatni a komponenst, akkor

meghívjuk a repaint() metódusát a komponensnek. Az update(Graphics g) alapesetben a komponens alapszínnel történő letörlésért, és a paint(Graphics g) meghívásáért végzi, ami (felüldefiniálható indokolt esetben).
A MyCanvas_OS (ősosztály) a Canvas osztály felüldefiniálása mely a koordináta rendszerhez kirajzolásához szükséges metódusokat tartalmazza. Ezt egy BufferedImage-re rajzoljuk mely a későbbi ábrázolás hátteréül szolgál. Új nap megjelenítése esetén nem rajzolja újra a hátteret. Ezt az osztályt használja a Főablak, illetve az Animációs ablak az ábrázoláshoz. Az update() metódusa úgy lett felüldefiniálva, hogy a rajzfelületet nem törli, így újrarajzolás esetén az nem villan fel fehéren. A koordináta rendszer kirajzolása konstansok arányításával történik, így a rajzfelület átméretezése esetén is méretarányos lesz (Függelék: 12.1).

5.4.1.2 Adatok osztály

Az adatbázisban előforduló attribútumoknak megfelelően tartalmaz egy változót a hozzá tartozó set,get illetve toStringet() metódusokkal az adatok kiíratásához. Mivel a programban esetenként változó mennyiségű attribútumot használunk így az annyi fajta konstruktort tartalmaz a minimális memória felhasználás érdekében. Ezt szakszóval konstruktortúlterhelésnek nevezzük. Az adatok feldolgozása során a láncolt listában ilyen típusú adatelemeket használunk.

5.4.1.3 Esemény osztály

A Főablak példányosítása után létrejön belőle egy példány. Ez Főablakon történő eseményeket megvalósító metódusokat tartalmazza. Ilyen például a dátum kiválasztása melynél a szökőévnek, illetve a hónapoknak megfelelően generálódik le a napokat tartalmazó legördülő menü tartalma.

5.4.1.4 SQL osztály

A MySQL connector segítségével hoz létre kapcsolatot az adatbázissal. Három metódust tartalmaz a program működése során végrehajtott adatbázis műveleteknek megfelelően. Az SQL parancsokat két csoportba lehet rendezni. Valamilyen adatbázis módosító művelet(törlés, felvitel, módosítás), illetve adatbázis lekérdezés (select). Ennek megfelelően tartalmaz egy

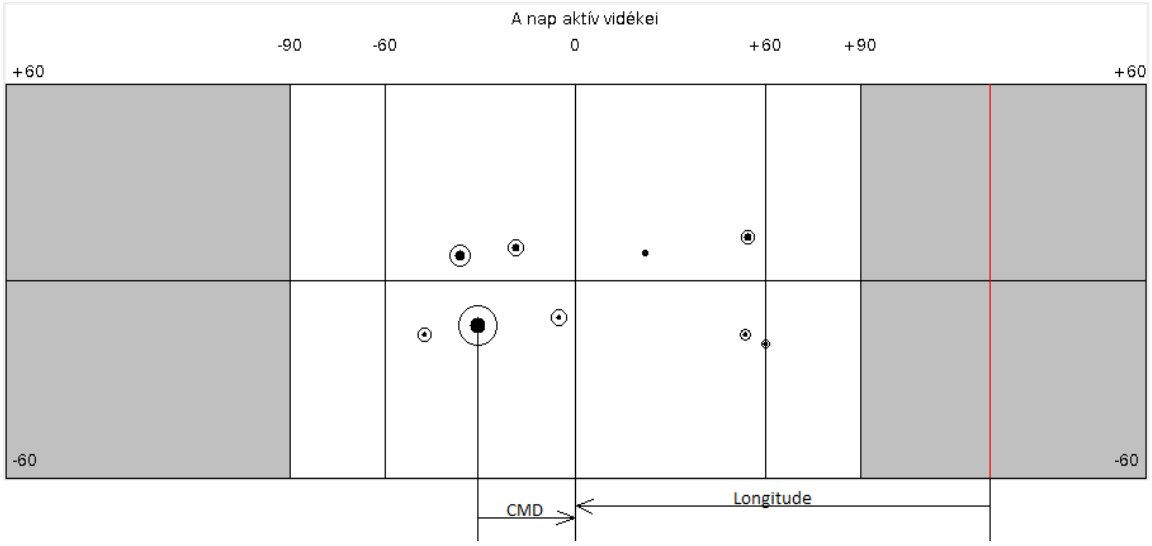
végrehajtó metódust, egy lekérdező metódust és egy kötegelt feldolgozást biztosító metódust a nagy mennyiségű adat feltöltéséhez.

```
public Object vegrehajt(String parancs) {     //delete,add,modify
        PreparedStatement pstmt = null;
        int eredmeny = 0;
        conn = null;
        try {
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                conn = DriverManager.getConnection(url, userName, password);
                pstmt = conn.prepareStatement(parancs);
                eredmeny = pstmt.executeUpdate();
        }catch (Exception e) {
                return (e.getMessage());
        }
        finally{
                try{
                        pstmt.close();
                        conn.close();
                }catch (SQLException ex) {
                        return ex.getMessage();
                }
                }
        return eredmeny;
}
```

5.4.1.5 Főablak foltábrázoló vászon

A foltok/foltcsoportok, illetve az kezdő meridián (L0) ábrázolása történik itt. A foltok foltcsoportok ábrázolása egy fekete körrel (whole area) illetve egy kitöltött fekete körrel (umbra) történik. Az L0-t egy függőleges piros egyenes jelöli. A vászon a nap felszínének kétdimenziós leképezése. A jobb és bal szélén szürkével jelöltük a nap nem látható részét. Vertikális irányban -60 és +60 fok között ábrázolunk mivel nagyobb szélességen már nincs észlelhető nap aktivitás. A rajzolás elindítása a dátum léptető, illetve a rajzol gomb használatával váltható ki. Ekkor az Esemény osztály MyCanvas_Action() metódusa hívódik meg. Ez létrehoz egy új MySQL kapcsolatot majd a paraméterül kapott dátumot beillesztve az SQL parancsba lekérdezi az ábrázoláshoz szükséges adatokat. Az adatokat a szintén paraméterül kapott vászon láncolt listájába tölti, majd a vászon repaint() metódusát meghívva kirajzolja a kívánt képet. A kép kirajzolása a MyCanvas_FA osztályban történik. Először a

hátteret teszi ki a vászonra, mely tartalmazza a koordináta rendszert és ezután hívja a Foltok() metódust, mely a láncolt listát feldolgozva rárajzolja a foltokat és az L0-t. A vászonnál 2,5-szeres nagyítást használunk. Azaz 1° 2,5 pixelnek felel meg.



8. ábra Vászon foltokkal.

A Central Mereidian-t (CM) megfeleltetjük a koordináta rendszer y tengelyének. Ezután a foltcsoport koordinátáját hozzá adjuk a CM koordinátájához. Így a kapott értéket azonnal tudjuk ábrázolni. A vászon 0,0 pontja a bal felső sarokban van. Az L0 ábrázolása esetén a folt Lng értékéből kivonjuk a folt CMD értékét. Az így kapott értéket vizsgáljuk. Ha nagyobb mint 180° akkor az L0 a CM-től balra esik és ki kell vonni belőle 540°-t. Ezután már ábrázolható az L0. Minden folt értékhez kirajzolja az L0-t így hibás adatbázis adatok esetén előfordulhat, hogy több L0 látható.

```
void foltok(Graphics g) {
        double CMD, L, B, FP = 2.5;
        for (int i = 0; Foltok.size() > i; i++) {
                CMD = FP * Foltok.get(i).getCmd();
                B = FP * Foltok.get(i).getLat();
                L = FP * Foltok.get(i).getLng();
                int R1 = (int) Math.round(Foltok.get(i).getS_umb());//belső
                int R2 = (int) Math.round(Foltok.get(i).getS_wh());//külső
                g.setColor(Color.red);
                if (Math.abs(CMD - L) > 180 * FP) {
                        g.drawLine((int) Math.round(540 * FP + (CMD - L)), YMIN, (int)
                        Math.round(540 * FP + (CMD - L)), YMAX);
                } else {
                        g.drawLine((int) Math.round(XMAX / 2 + CMD - L), YMIN, (int)
```

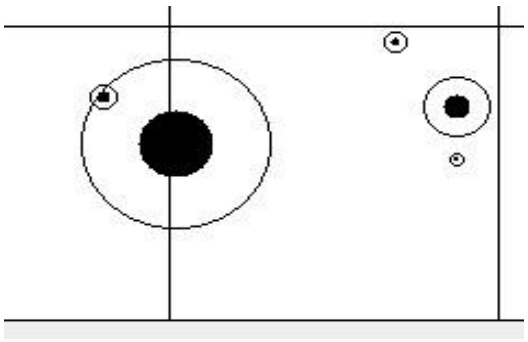
```
                        Math.round(XMAX / 2 + CMD - L), YMAX);
                }
                g.setColor(Color.black);
                g.drawOval((int) Math.round(XMAX / 2 + CMD) - R2, (int)
                Math.round((YMAX + YMIN) / 2 - B) - R2, 2 * R2, 2 * R2);    //külsö
                g.fillOval((int) Math.round(XMAX / 2 + CMD) - R1, (int)
                Math.round((YMAX + YMIN) / 2 - B) - R1, 2 * R1, 2 * R1);    //belső
        } // end for
}
```

A foltokra jobb egérgombbal kattintva egy Folt ablak nyílik. Erről értesítést is kapunk, ha az egeret a vászonra mozgatjuk. A kattintás koordinátájának lekérdezése után az alábbi folyamatok zajlanak le. A láncolt listából vesszük növekvő sorrendben a foltokat és a kattintás koordinátáját és a kör egyenletének segítségével meghatározzuk, hogy az adott foltra kattintottunk-e.

```
while (canvas1.Foltok.size() > ++i) {
        CMD = 2.5 * canvas1.Foltok.get(i).getCmd();
        B = 2.5 * canvas1.Foltok.get(i).getLat();
        double k = x - (canvas1.XMAX / 2 + CMD);
        double l = y - ((canvas1.YMAX + canvas1.YMIN) / 2 - B);
        double r = canvas1.Foltok.get(i).getS_wh();
        if ((k * k) + (l * l) <= r * r) {
                Folt_window = new Folt_ablak();
                Folt_window.setVisible(true);
                try {
                        ResultSet rs = null;
                        StringBuilder str = new StringBuilder("SELECT * FROM test.arta
                        WHERE num=");
//SQL parancs létrehozása
                        Object eredmeny = lekerdez.lekerdez(str.toString());
                        if (eredmeny instanceof String) {
                                return ((String) eredmeny);
                        } else {
                                rs = (ResultSet) eredmeny;
                                while (rs.next()) {
                                        Folt_window.setAdatok(new Adatok(
                                        rs.getDate(1), //dátum
                                        //adatok feltöltése
                                        rs.getDouble(17) //S_WH
                                        ));
                                }  // While vége
                                rs.close();
                        } // else vége
                } catch (Exception e) {
                        return ("Hiba a kattintás metódusban" + e.getMessage());
```

```
                }
                Folt_window.setTitle("A folt azonosítója :" +
                canvas1.Foltok.get(i).getAzonosito().toString());
                Folt_window.Action();
                break;
        } // if a benne a körben vizsgálat vége
} //while vége
```

A növekvő sorrendben történő feldolgozás azért szükséges, mert a foltok ábrázolása során előfordul, hogy átfedik egymást. Így nem fordulhat olyan elő, hogy nem tudunk úgy kattintani egy kis foltra, hogy mindig a nagy folthoz tartozó Folt ablak jelenjen meg.



9. ábra Foltok átfedése.

5.4.1.6 Foltablak, diagramok, táblázat

Egy foltcsoportra kattintás hatására megnyíló Foltablak az aktuális foltcsoporthoz tartozó adatokat tartalmazza táblázatos formában, valamint a foltcsoport területének változását oszlopdiagram formájában. Az oszlopdiagram piros része ábrázolja a folt külső részének (Whole) területét, a fekete rész pedig ennek arányában a belső rész (Umbra) területét. Az x tengelyen a folt megfigyelésének dátuma van megjelenítve, az y tengelyen pedig a hozzájuk tartozó értékskála, mely a Nap félgömb felszínének milliomod részében van megadva. A pontos értékek az ablakban lévő táblázatból olvashatóak le. A két diagram rajzolása a már fent említett vászon paint() metódusának felüldefiniálásával történt. Az adatbázisban két értékpáros van megadva egy folt területére. Az egyik a mért-umbra, és a mért-whole, a másik a korrigált-umbra és a korrigált-whole érték. A két oszlopdiagram ezen adatok eltérését jeleníti meg. Az alattuk lévő táblázatban található az adatbázisból kinyert adatok számszerű értéke, és a hozzájuk tartozó egyéb adatbázisbeli tulajdonság.

Dátum	Óra	ID	SufgN	MTW	GRTP	ObsUmb	ObsWh	CorrUmb	CorrWh	Dist	Ang	Lng	Lat	CMD
1946-01-13	497	14408	0	2	6	47	248	28	148	0,542	7,4	238,7	28	-4,5
1946-01-14	429	14408	0	2	6	185	1014	112	612	0,559	349,8	237,4	28,8	6,5
1946-01-15	439	14408	0	2	6	209	1321	132	834	0,61	333,1	235,8	28,4	18,2
1946-01-16	490	14408	0	2	6	189	1086	134	769	0,708	319,2	235,4	28,3	31,6
1946-01-17	403	14408	0	2	6	139	914	120	787	0,814	310,8	236,3	28,7	44,5
1946-01-18	417	14408	0	2	6	92	630	110	752	0,908	305,2	236,2	28,9	57,8
1946-01-19	415	14408	0	2	6	40	309	84	647	0,971	301,7	236,2	29,1	70,9

10. ábra Kiválasztott folthoz tartozó adatok.

5.4.1.6.1 Statisztikai diagramok

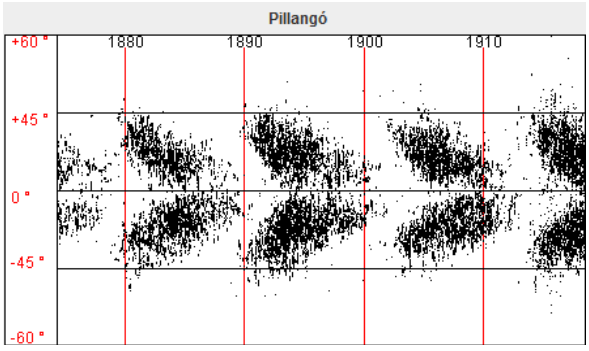
A Napfoltok száma a napciklussal együtt erőteljesen növekszik, majd erőteljesen csökken. Ezt jeleníti meg a két diagram a főablak alján. A statisztikai diagramok a foltablak alatt található csúszó-gomb értékének beállításával jeleníthetőek meg. A csúszka a javax.swing osztály beépített komponense. Kattintás, illetve a csúszka mozgatás segítségével állítható be a diagramokhoz tartozó aktuális év. A beállított dátumot a csúszka mellett található „jLabel” Swing komponensen látjuk. Amint beállítottuk a csúszka értékét, két beágyazott osztály példánya jön létre – egyik a pillangódiagramhoz, a másik a területdiagramhoz. Ezek az osztályok a „Thread” (szál) osztályból vannak származtatva és a run() metódusaik felül vannak definiálva oly módon, hogy a metódusok az esemény osztályban lévő adatlekérdezést, feldolgozást majd rajzolást végző metódusokat hívják. A folyamatot hosszabb futásideje miatt kellett külön szálra tenni.

```
public class PillangoSzal extends Thread {
        @Override
        public void run() {
                try {
                        sleep(100);
                        esemeny.Pillango(jSlider1.getValue(), 410, canvas2);
```

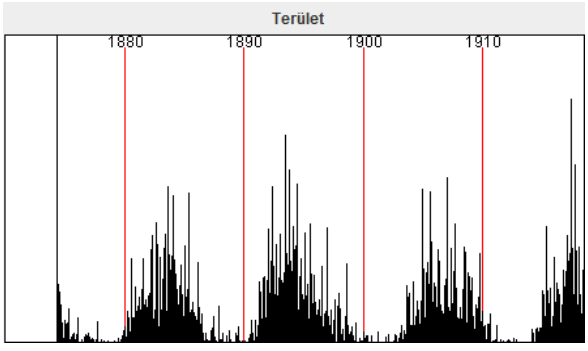
```
                } catch (InterruptedException ex) {
                        Logger.getLogger(FoAblak.class.getName()).log(Level.SEVERE,
                        null, ex);
                }
        }
}
```

```
public class TerületSzál extends Thread {

        @Override
        public void run() {
                try {
                        sleep(100);
                        esemeny.Terület(jSlider1.getValue(), 450, canvas3);
                } catch (InterruptedException ex) {
                        Logger.getLogger(FoAblak.class.getName()).log(Level.SEVERE,
                        null, ex);
                }
        }
}
```

A csúszka kezdőállapota az adatbázis kezdődátuma, a végállapota az az évszám, amelynél a rajzolt diagram vége a vászon végéig tart. Egyik statisztikai diagram a pillangó diagram. A Nap felszínén a napfoltok megjelenési helyének változása a ciklikusságnak megfelelően változik. Ezt ábrázolva egy úgynevezett pillangó diagramot kapunk. A diagram y tengelyének értékei a foltok szélességi körön való elhelyezkedésének koordinátái. Az x tengelyen a foltok dátum szerinti elhelyezkedésüknek megfelelően jelennek meg. Minden egyes pixeloszlopra 40 naphoz tartozó folt van ábrázolva.



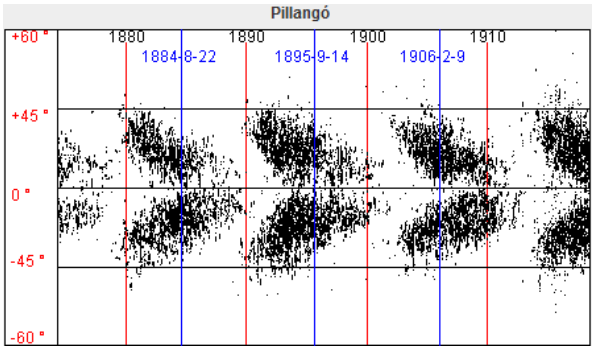
11. ábra Pillangó diagram.



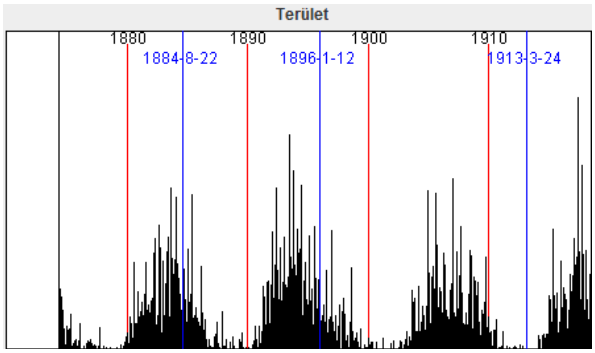
12. ábra Terület diagram.

Mind a két diagramnál 10 éves beosztás található. Ennek segítségével nagyjából beazonosítható egy ciklus kezdete valamint vége. A területdiagramon 40 napra eső foltok

területének összegét ábrázolja egy pixeloszlop. A statisztikai diagramok fölött tartva az egérkurzort, az főablak alsó sorában megjelenik az aktuális oszlophoz tartozó dátum. Bal egérgombbal kattintva meg lehet jelölni egy függőleges vonallal az aktuális helyen lévő értéket. A diagramon megjelenik egy függőleges kék vonal, valamint a hozzá tartozó dátumérték. Ha nincs szükségünk a vonalakra, akkor jobb egérgombbal a diagram felületén kattintva eltüntethetjük őket. Ehhez felül kellett definiálnunk a java beépített egérkezelő függvényeit. Ugyanezt alkalmazhatjuk a Terület diagramra is.



13. ábra Pillangó diagram területbélyegekkel ellátva.



14. ábra Terület diagram területbélyegekkel ellátva.

5.4.2 Kezelő eszközök

A főablakon elhelyeztünk három legördülő menüt. Ezzel állíthatjuk mely dátumhoz tartozó foltokat szeretnénk megtekinteni a felső vásznon. A rajzolás maga a Rajzol gomb lenyomása után történik. A jobb és bal oldalán elhelyeztünk két ikont mely segítségével előre, illetve hátra lehet léptetni a dátumot egy nappal. Ilyenkor a rajzolás automatikusan megtörténik.

5.4.3 Menüpontok

A program főablakában megjelenített menüsáv a java beépített jMenuBar komponense, mely a Netbeans Design nézetében is szerkeszthető és tetszőlegesen testre szabható. A következő menüpontokat vettük fel a program Főablakába: Fájl, Adatbázis, Egyéb és, Súgó.

5.4.3.1 Kép mentése

A Fájl ablakban a Save menüpont segítségével a vászon tartalmát menthetjük el JPG fájlba. A menüből aktivizálhatjuk vagy a CTRL-S gombnyomás segítségével. Először egy fájlválasztó ablak jelenik meg, ahol megadhatjuk a kívánt fájl nevet és útvonalat. Ezután a vászon SaveToFile() metódusa hívódik meg. Itt létrejön egy szál, melyben létrehozunk egy BufferedImage objektumot. Erre rámásoljuk a vászon hátterét és ezután meghívjuk a kép Graphics objektumára a foltrajzoló metódust. Ez után történik meg a kép fájlba írása. Ha mentés közben hiba lép fel vagy felhasználói megszakítás történik, akkor erről egy felugró ablakban kapunk értesítést (Függelék 12.3.).

5.4.3.2 Adatbázis menüpontok

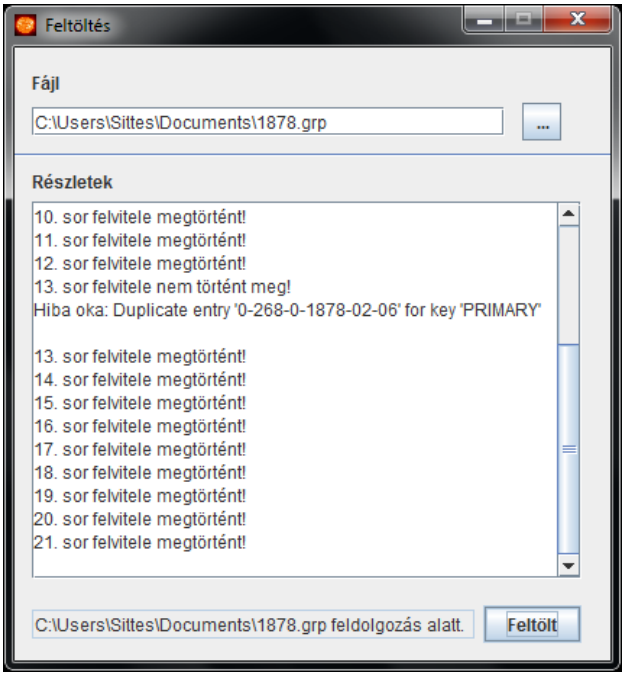
Az Adatbázis menüpontban az adatbázissal végezhető műveletek találhatók. Itt alakíthatjuk ki a fent említett adatbázist és a tábla szerkezetét (Függelék: 12.4.). Az esemény osztály szerkezet_kialakító() metódusa hívódik, mely egy SQL osztály példányán keresztül kapcsolódik az adatbázishoz és a megfelelő parancsot kiadva létrehozza az adatbázist, és a hozzá tartozó táblát. A művelet sikerességéről visszajelzést kapunk, mely egy információs ablakban olvasható. Adatbázis vagy tábla létezése esetén is ebbe az információs ablakba kapunk hibajelzést. Az asztali alkalmazás hibakezelését nagyban segíti a javax.swing osztály JOptionPane komponense, mellyel többféle felugró ablak jeleníthető meg egy művelet végrehajtásának sikerességéről, vagy sikertelenségéről. Felhasználóbarát alkalmazást kapunk használata segítségével. Lehetőség van már meglévő adatbázis importálására az Adatbázis import almenüpontja alól, melyben egy „csv” kiterjesztésű fájl kiválasztásával az adott fájlban lévő összes adatot feltölti az adatbázisba. A „csv” kiterjesztés egy szabvány fájlformátum. Vesszővel tagolt értékek tárolására használatos formátum. Ilyen fájlt a programmal is készíthetünk az export menüpont használatával. A szoftver importálás során csak a „csv” fájlt engedi kiválasztani. Ennek implementálása a kódba egy FileFilter osztályból való örököltetéssel lehetséges.

```
//Kitallózható fájlok típusának szűkítése
class MyFilter extends javax.swing.filechooser.FileFilter {

	public boolean accept(File file) {
		String filename = file.getName();
		return filename.endsWith(".csv") || file.isDirectory();
	}

	public String getDescription() {
		return "*.csv";
	}
}
```

Importáláskor a program egy külön Import osztályt példányosít, mely a Thread osztályból származtatott tulajdonságokkal rendelkezik. A run() metódus felüldefiniálásával tettük ily módon külön futó folyamattá az importálást. Ez a tulajdonság igaz az exportálásra is. Importáláskor a program leellenőrzi az adatbázis és tábla meglétét és valamelyik hiányában tájékoztatja a felhasználót erről. Importálás során %-os arányban jelzi egy felugró ablak, a folyamat állapotát. A feldolgozás előtt végigolvassa a program a feltöltendő fájl sorait. Innen tudja, hogy hány %-nál jár éppen az importálás állapota.

Exportálásnál is ugyanígy %-osan kap a felhasználó információt a folyamatról. Itt az adatbázisban lévő sorok száma adja meg a teljes 100%-ot.

Ezen a menüponton belül az Upload menüpontra kattintva előugrik egy ablak, melyben van egy beviteli mező, ahova be lehet írni a feltöltendő fájl elérési útját, vagy a mellette lévő gomb segítségével ki lehet tallózni. Itt csak a NOAA által közzétett „.grp” kiterjesztésű fájlokat fogja elfogadni, és helyesen kezelni. Ha nem ilyen kiterjesztésű fájlt adunk meg, akkor hibaüzenettel jelzi a program. Ha a megfelelő fájlt választottuk ki, akkor a Feltölt gombra kattintva elindul a feltöltés. A gomb megnyomásának hatására létrejön egy külön szál. Ez végzi a feltöltés folyamatát és közben hozzáférünk a program más komponenseihez is. Az ablak szövegdobozában látjuk a folyamat részleteit, minden egyes sorról visszajelzést kapunk, hogy a feltöltés sikeresen megtörtént vagy sem.



15. ábra Feltöltés folyamata.

A folyamat befejezését az jelzi, hogy az alsó sávban lévő „fájl feldolgozás alatt” felirat átvált egy összesítő információra, amiből megtudjuk, hogy a fájl hány sora került fel az adatbázisba, és hány sor volt a fájlban.

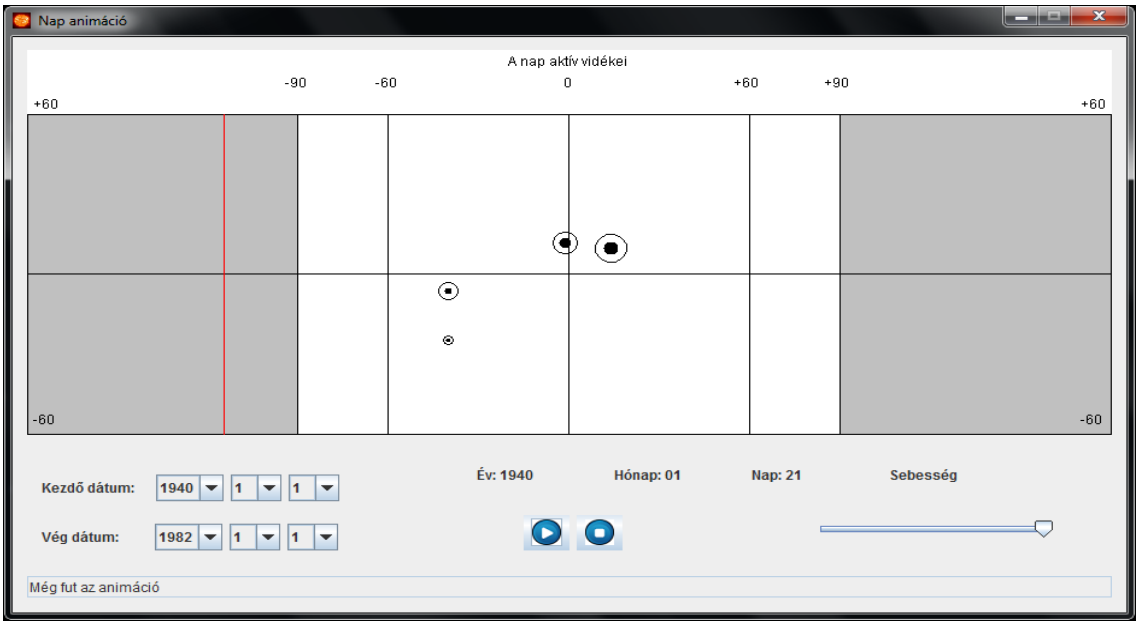
16. ábra A feltöltés befejeződött.

Üres elérési út esetén kattintva a Feltölt gombra, figyelmeztet minket a program, hogy nem választottunk ki fájlt.

5.4.3.3 Egyéb menüpont

5.4.3.3.1 Animációs ablak

Célja a napfoltok megjelenítése filmszerűen a legördülő menükkel kiválasztott kezdő és vég dátum között. A lejátszás csak akkor indul el, ha a vég dátum nagyobb. A lejátszás a „Play” és a „Stop” gombok segítségével vezérelhető. Az éppen megjelenő nap dátuma a vászon alatt olvasható. A foltok megjelenítése egy adott napon megegyezik a Főablakon láthatóval.



17. ábra Animációs ablak.

Maga a megjelenítés az alábbi módon történik. Az animációs ablak vászonja szintén a MyCanvas_Os osztályt definiálja felül. A „Play” gomb lenyomása után a beállított dátumoknak megfelelően létrejön egy SQL parancs mely lekérdezi a teljes intervallumhoz tartozó adatokat. Szintén láncolt listába tölti az adatokat melyből feldolgozás során folyamatosan törli is azokat. A Főablakkal ellentétben itt az ábrázolás képre történik. Az animáció az úgynevezett termelő-fogyasztó algoritmus alapján működik. Két szálat hozunk létre melyből az egyik a megjelenítendő képek rajzolását végzi és ezt egy pufferbe helyezi. A megjelenítést a másik szál végzi, mely a pufferből kiveszi a kész képeket és a vászonra helyezi adott időközönként. Ezt az időközt a vászon alatti csúszkával állíthatjuk. Ha a puffer üres vagy éppen tele van, akkor az adott szál várakozik. Ebből a várakozásból a másik szál

ébreszti fel (már nem üres vagy teli a puffer). A „Stop” gomb lenyomásával a leállítható a lejátszás. Ilyenkor ugyanaz a folyamat zajlik le, mint mikor lejátszás során elérjük a beállított vég dátumot. A képrajzoló metódus egy null értékű képet helyez a pufferbe, mellyel majd leáll a rajzoló szál. Így jelzi a kép a megjelenítő szálnak, hogy fejezze be a futást (Függelék 12.2.).

5.4.3.3.2 Teljes menüpont

Az Egyéb menü Teljes menüpontjában megnyílik egy új ablak, melyben a fent már említett és a főablakban látható pillangó és terület diagramok jeleníthetőek meg a teljes adatbázis adatait feldolgozva és megjelenítve a rajz vásznon.

5.4.3.4 Súgó menüpont

Segítségnyújtást kaphatunk a program használatához. Létrehoztunk egy új felületet, melyben elhelyeztünk egy JTree objektumot. Ez a program főbb funkcióit tartalmazza témakörökre osztva. A témakörökre kattintva az ablak jobb oldalán megjelenik a hozzájuk tartozó html alapú oldal egy jEditorPane objektumon. Ez egy olyan speciális panel, amely képes megjeleníteni a html oldalakat. A html alapú súgófájlok megírásához a Notepad++ programot használtuk.

5.5 Alkalmazás fordítása

A program fordítását a NetBeans segítségével végeztük. A beépített fordító alapesetben létrehozna a projekt könyvtárában egy /dist könyvtárat melyben létrejön a lefordított program .jar kiterjesztésel és létrejön egy /lib könyvtár melyben további szükséges osztályokat találjuk(esetünkben swing-layout-1.0.4.jar és a mysql-connector-java-5.1.15-bin.jar fájlokat). Ezt nem tartottuk biztonságos és szép megoldásnak. Szerettük volna, hogy egy jar, vagy exe fájl legyen az elkészített program. Végül a build.xml fájl kiegészítve egy úgynevezett ANT nyelven íródott script-tel sikerült is elérni, hogy a fordító kimenetele egyetlen jar fájl legyen.

6 Web-alkalmazás

6.1 Alapismeretek megszerzése:

A szakdolgozat megértéséhez elengedhetetlen volt a HTML, PHP, MYSQL, JSP programozási ismeretek, illetve a Drupal tartalomkezelő ismeretének elsajátítása. Az alkalmazás fejlesztésének, dolgozat elkészítésének főbb lépései:

- A dolgozat témájának megértése, elsajátítása
- Web-alkalmazás tervezése
- Adatbázis megtervezése
- Alkalmazás elkészítése

6.2 CMS, Drupal

6.2.1 A CMS bemutatása

Content managemet system, azaz tartalomkezelő rendszer a weboldal tartalmának építéséhez, karbantartásához biztosít kényelmes és gyors munkavégzést. Egy olyan adminisztrációs felület, melynek segítségével változtatni, frissíteni, módosítani lehet a weboldal tartalmát (pl.: képfeltöltés, szöveges tartalomfrissítés, stb.). Egy statikus honlap esetén akármilyen változtatásnál szükséges egy web-programozó, míg dinamikus honlapoknál a felhasználó akár szaktudás nélkül is bármikor elvégezheti a tartalmak szerkesztését. Egyszerűen és költséghatékonyan lehet az oldal kinézetét megváltoztatni anélkül, hogy a futó program módosítása során elkerüljük a hibák lehetőségét.

6.2.2 Drupal, mint webes tartalomkezelő rendszer

A Drupal egy webes tartalomkezelő rendszer, mellyel könnyedén lehet szövegeket, képeket, csatolmányokat feltölteni és a felhasználók számára közzétenni. Nem kell technikai háttérrel foglalkozni, csak a tartalmakra összpontosítani, így rengeteg idő spórolható meg. A tartalmakat egy adatbázisban tárolja. Lehetőséget ad arra, hogy a látogatók különböző szerepkörökben és más-más jogosultsággal használják a weboldalt. Továbbá lehetőségünk

van blogok és fórumok megvalósítására is. Kiváló a felhasználó-, és jogosultság kezelés, a felület megjelenés és testre szabhatóság tekintetében.
Csak fel kell telepítenünk a rendszert egy tárhely szolgáltatóhoz, és máris kezdhetjük weboldalunk kialakítását. Továbbá hozzáfűzhetünk kiegészítő modulokat, sminkeket, aktuális fordításokat.

6.3 Greenwich daily total sunspot area adatbázis fejlesztése

6.3.1 Fejlesztői környezet, Platform

A weboldal fejlesztése a Drupal 6.20 -as verziójával, illetve Notepad++ szövegszerkesztő segítségével történt Microsoft Windows 7 operációs rendszer alatt.
Szerver oldalon a program Apache 2.2-es webszervert, PHP 4 értelmezőt és MySQL 5.1 adatbázis szervert igényel, a kliens oldalon pedig Google Chrome 10.0 vagy Mozilla Firefox 3.0 változata szükséges az alkalmazás megfelelő működéséhez.

6.3.2 Adatbázis szerkezet kialakítása

Az itt használt adatbázis felépítése, szerkezete megegyezik a fentebb kifejtettel.

6.4 Felhasználói dokumentáció

6.4.1 A rendszer követelményei

A Web-alkalmazás szükséges és elégséges rendszerkomponensei a következők:

- Széles sávú internetkapcsolat
- Legalább 1024x768 felbontás, 16 bites színmélység
- Google Chrome 10.x böngésző (ajánlott)

6.4.2 Felhasználói felület

A felhasználói felület a következő komponensekből áll: (19. ábra)

- Fejléc, melyen az oldal címe található.
- Navigációs menüsor, melynek elemei következők: Kezdőlap, Grafikon, Adatbázis, Fórum, Letöltések, Kapcsolat, Linkek, Súgó.
- Tartalmi rész. Jelen esetben az aktuális híreket, információkat és további fejlesztéseket jelenítjük meg a felhasználók, vendégek számára. Itt olvashatunk arról, hogy miért jött létre, és mi a célja oldalunknak.
- Blokkok. Navigációs blokk ahol, megtekinthetőek a felhasználó adatai (regisztrálás után). Az utolsó tíz újonnan regisztrált felhasználó névsorát láthatjuk, illetve egy statisztikai részt az oldal használatával kapcsolatban.
- Lábléc, ahol elérhetőek a menüpontok.

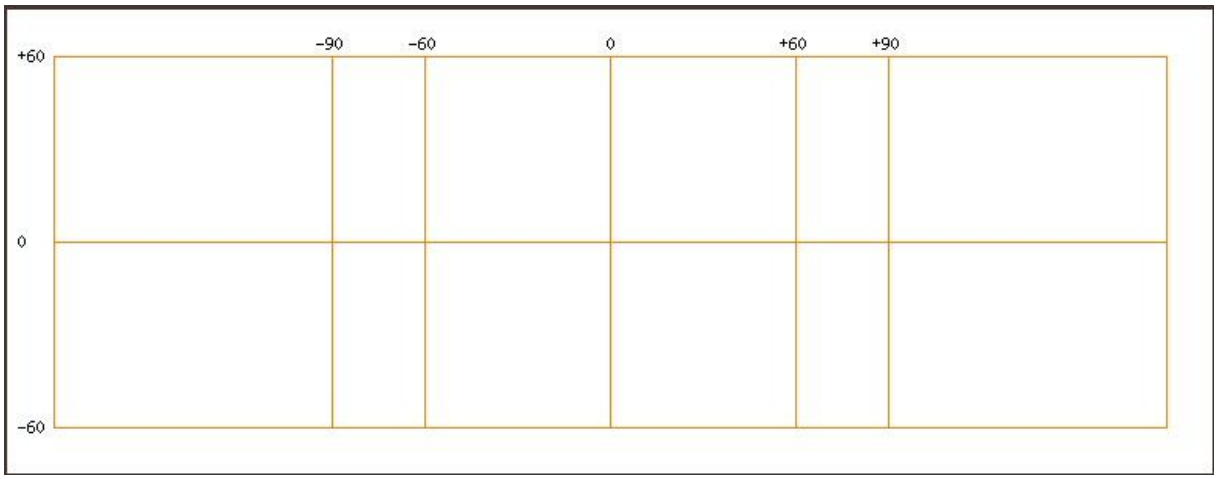
19. ábra A weboldal kezdőlapja.

6.4.2.1 Grafikon

Ezen az oldalon található a Napfoltok ábrázolása, mellyel naponta megtekinthető a napfoltok alakulása, egy pillangó diagram, ahol a Napfoltok megjelenési helyének változását, illetve egy terület diagram, mely a Nap aktív területének változását mutatja.

6.4.2.1.1 Napfoltok ábrázolása, foltadatok, statisztikai ábrák

Kezdetben egy üres koordináta rendszer jelenik meg. (20. ábra).



20. ábra Üres állapotban a napfoltok ábrája.

Az ábra alatt található a dátum kiválasztása opció, ahol kiválasztva az évet, hónapot és napot, a program az aktuális dátumhoz tartozó napfoltokat és az L0-t rajzolja ki. Egy genDateSelector() függvény segítségével beállítjuk az évet, hónapot és napot.

```
function genDateSelector($year, $month, $day, $prefix = "")
{
// Év beállitása
        echo"<select name=\"${prefix}year\"
        onchange=\"ChangeOptionDays(this.form,'$prefix')\">";
//Első választható év = 1874
        $min = 1874;
//Az utolsó választható év pedig a jelenlegi év ( jelen esetben: 2011)
        $max = max($min, date("Y"));
//Bejárjuk egy ciklussal a kezdeti évtől a jelenlegi évig
        for ($i = $min; $i <= $max; $i++)
        {// értékül adjuk az éveket
                print "<option value=\"$i\"" . ($i == $year ? " selected=\"selected\"" : "") .
">$i</option>";
```

```
        }
        echo "</select>";
// Hónap beállítása
        echo "<select name=\"${prefix}month\"
        onchange=\"ChangeOptionDays(this.form,'$prefix')\">";
        $honapnevek = array("01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
"12");
        $napok_szam = array("01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
"12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27",
"28", "29", "30", "31");
        for ($i = 1; $i <= 12; $i++)
        { print "<option value=\"$i\"" . ($i == $month ? " selected=\"selected\"" : "") . ">".
        $honapnevek[$i-1]."</option>";  }
        echo"</select>";
// Nap beállítása
        echo "<select name=\"${prefix}day\">";       //Választási lehetoség
        for ($i = 1; $i <= 31; $i++)
        {
                echo "<option" . ($i == $day ? " selected=\"selected\"" : "") . ">".
                $napok_szam[$i-1]."</option>";       //Milyen értékei lehetnek
        } echo "</select>";  }
```

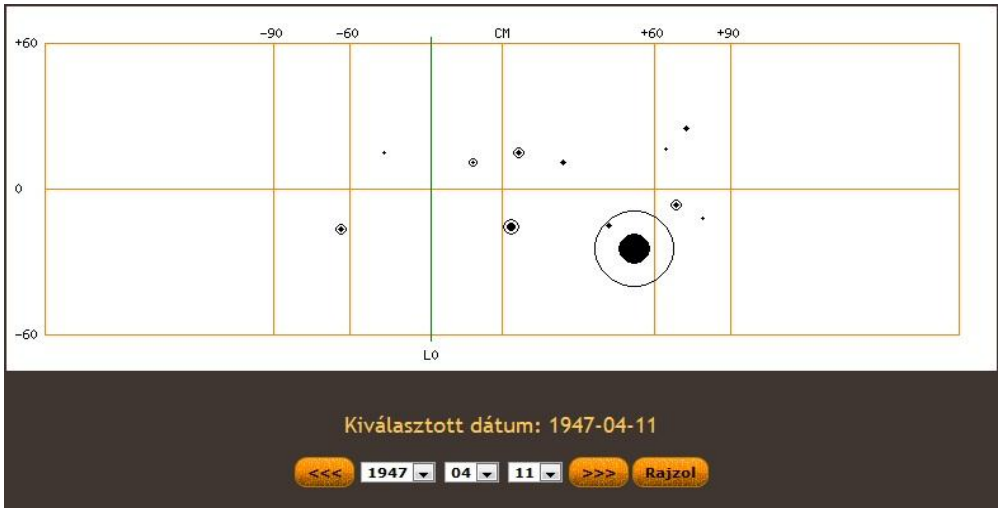
Megvizsgáljuk, hogy a kiválasztott év szökőév vagy sem.

```
function daysInFebruary (year)
        { return (((year % 4 == 0) && ( (!(year % 100 == 0)) || (year % 400 == 0))) ? 29 : 28
); }
```

Beállítjuk, hogy melyek a 30, illetve 31 napos hónapok.

```
function DaysInMonth(WhichMonth, WhichYear)
        { var DaysInMonth = 31;
        if (WhichMonth == "4" || WhichMonth == "6" || WhichMonth == "9" || WhichMonth
== "11")
        { DaysInMonth = 30; }
        if (WhichMonth == "2")
        { DaysInMonth = daysInFebruary( WhichYear ); }
        return DaysInMonth; }
```

Miután kiválasztottuk a dátumot, a Rajzol gomb segítségével kirajzoltatjuk a napfoltokat - (21. ábra).



21. ábra Az 1947-04-11 dátumhoz tartozó napfolt csoportok.

Egy Select lekérdezés végrehajtásával, lekérdezzük a foltok rajzolásához szükséges adatokat (Num, Obsumb, Obswh, Lng, Lat, Cmd, S_Umb, S_Wh), majd az adatok feldolgozása után megjelenítjük az L0 -t és a foltokat a következőképpen:

A rajzoláshoz kétszeres nagyítást használunk, tehát 1° 2 pixelt jelent. Vesszük a CMD és az Lng különbségét. Vizsgáljuk az értékét, ha nagyobb, mint 180° (mivel 0°-tól ábrázolunk, ami a Central Meridian) kivonunk belőle 540°-t és az így kapott értéket jelenítjük meg. Ez azért szükséges, mert a CMD +/-90°, az Lng pedig 0° és 360° között vehet fel értékeket. Minden folt értékéhez kirajzolja így az L0 -t, melyet egy függőleges zöld egyenes jelöl. Rossz adatbázis adatok esetén előfordulhat, hogy több L0 látható az ábrán.

```
//L0 számítás
$FP = 2; // kétszeres nagyítás
$B = $FP * $lat; //szélesség (latitude)
$L = $FP * $lng; //hosszúság (longitude)
$CMD_szorzott = $FP * $cmd;
if (abs($CMD_szorzott - $L) > (180 * $FP))
{
        imageline($img["foltok"], round(540*$FP+$CMD_szorzott-$L)+FOLT_SZEGELY,
FOLT_SZEGELY-5, round(540 * $FP + $CMD_szorzott - $L)+FOLT_SZEGELY, YMAX
- FOLT_SZEGELY+5, $cmd_color);
}else {
        imageline($img["foltok"], round(XMAX/2 + $CMD_szorzott -
$L)+FOLT_SZEGELY/2, FOLT_SZEGELY-5, round(XMAX / 2 + $CMD_szorzott -
$L)+FOLT_SZEGELY/2, YMAX - FOLT_SZEGELY+5, $cmd_color);  }
```

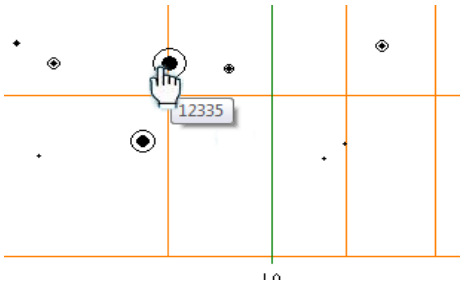
A foltokat az alábbi kód segítségével ábrázoljuk.

Az (XMAX/2) jelenti a 0° szélességi kört, az ((YMAX+YMIN)/2) pedig a 0° hosszúsági kört.

```
//Umbral rajzolása
        imagefilledellipse($img["foltok"], round(XMAX / 2 +
$CMD_szorzott)+FOLT_SZEGELY/2, round((YMAX + YMIN) / 2 - $B),
2*(round($sugar_UMB)), 2*(round($sugar_UMB)), $obsumb_color);
//Whole rajzolása
        imageellipse($img["foltok"], round(XMAX / 2 +
$CMD_szorzott)+FOLT_SZEGELY/2, round((YMAX + YMIN) / 2 - $B),
2*(round($sugar_WH)), 2*(round($sugar_WH)), $obswh_color);
```

További vezérlő gombok találhatóak a dátum kiválasztásánál (<<<, >>>), melyekkel a dátumot léptethetjük naponként.

A foltokra mozgatva az egérkurzort, úgynevezett tooltip -ben kiírja az aktuális folt csoportszámát (22. ábra).



22. ábra A 12335 számú foltcsoport.

A foltcsoportokra az egér bal gombjával kattintva megtekinthetőek annak adatai egy táblában. Ábrázolásnál előfordulhat, hogy átfedik egymást a foltcsoportok, ezért növekvő sorrendben ábrázoljuk a foltokat, így elkerüljük azt, hogy nem tudunk úgy kattintani a kisebb foltra, hogy a nagyobb folt jelenjen meg mindig. Egy Select lekérdezéssel a kiválasztott folt csoportszámához tartozó adatokat lekérdezzük és megjelenítjük a látogatók számára (23. ábra).

Kiválasztott folt csoportszáma: 14900

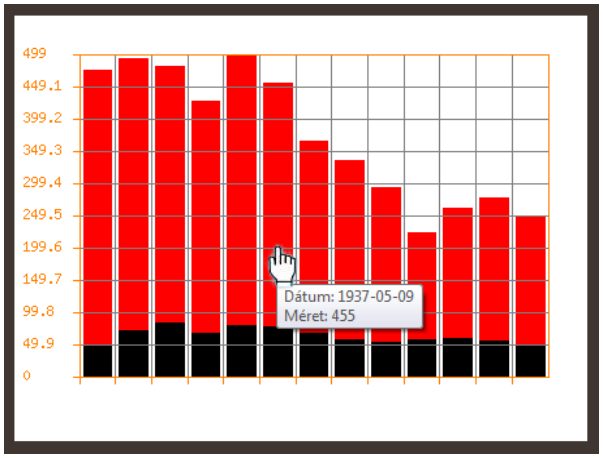
Date	Time	Group number	SUFGN	MT Wilson	Group type	Observed Umbral	Observed Whole	Corrigalt Umbral	Corrigalt Whole	Distance	Angle	Longitude	Latitude	Central meridian distance
1947-04-10	342	14900	0	5	2	11	49	10	43	0.82	295	92	16.5	50.6
1947-04-11	346	14900	0	5	2	2	11	3	15	0.927	290.3	92.8	16.3	64.7
1947-04-12	311	14900	0	5	0	0	21	0	57	0.982	289.5	91.6	17.8	76.2

23. ábra A 14900 -as csoportszámú folt adatai.

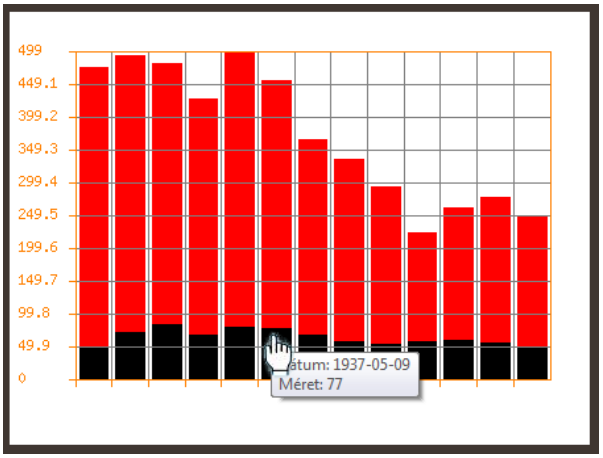
Az attribútumokra mozgatva az egérkurzort, kiírja egy tooltip -ben a kapcsolatos információkat.

Két további statisztikai ábrát tartalmaz az oldal. Egy Korrigált területi diagramot, ahol számunkra ismeretlen módszer alapján a Corrigalt Umbral és Corrigalt Whole értékek alapján rajzolt diagram látható. A Mért területi diagramon pedig az Obsever Umbral és Observed Whole értékeket ábrázolja.

Piros oszlopok a folt külső részének (Whole) területét, fekete oszlopok pedig a (Whole) arányában a belső rész (Umbral) területét jelölik. A diagramokon lévő oszlopokra mozgatva a kurzort, kiírja a hozzá tartozó dátumot és a folt méretét színtől különbözően (24-, 25. ábra).



24. ábra Korrigált területi diagram.



25. ábra Mért területi diagram.

A továbbiakban a Korrigált területi diagram elkészítését részletezzük, a Mért területi ábránál hasonlóképpen jártunk el.

Végigmegyünk a lekérdezett adatokon egy while ciklus segítségével, majd megkeressük a legnagyobb Corrigalt Umbral, Corrigalt Whole, Obsever Umbral, és Observed Whole

értékeket, melyeket letároljuk egy-egy tömbben. Ezek az értékek lesznek az oszlopok méretei. Megszámoljuk egy sizeof() függvény segítségével, hogy a táblában mennyi rekordunk van, majd a kapott értékkel elosztjuk a rajzfelület szélességét, így megkapjuk az X tengely beosztását.

```
$meretwh = sizeof($corrwhtomb);
$szel = 360/($meretwh);
```

A lentebb látható kódrészletben egy for ciklussal végigmegyünk az adatsorokon, majd megrajzoljuk egy imagefilledrectangle() függvénnyel a diagram oszlopait DEFINE változók segítségével.

```
for($i=0; $i<($meretwh); $i++)
        {
        ...
//Piros oszlop (Corrigalt Whole)
        imagefilledrectangle ($img["korrigalt"], (($i * $szel) +
KORRIGALT_SZEGELY_j_f)+$szel+17 , KORRIGALT_HEIGHT -
KORRIGALT_SZEGELY_b_l, (($i * $szel+3) + KORRIGALT_SZEGELY_b_l),
KORRIGALT_HEIGHT - KORRIGALT_SZEGELY_b_l -
$corrwhtomb[$i]/($legnagycorrwh/250), $wh_color);
//Fekete oszlop (Corrigalt Umbral)
        imagefilledrectangle ($img["korrigalt"], (($i * $szel) +
KORRIGALT_SZEGELY_j_f)+$szel+17 , KORRIGALT_HEIGHT -
KORRIGALT_SZEGELY_b_l,   (($i * $szel+3) + KORRIGALT_SZEGELY_b_l),
KORRIGALT_HEIGHT - KORRIGALT_SZEGELY_b_l -
(250/$legnagycorrwh)*$corrumbtomb[$i], $umb_color);      }
```

Az Y tengelyt felosztjuk tíz egyenlő részre, majd egy imagestring() függvénnyel kiíratjuk az Y tengely értékskáláját, melyek a Nap félgömb felszínének milliomod részében vannak megadva.

```
for($i=0; $i<11; $i++)
        {
// Y tengely értékei:
        imagestring($img["korrigalt"], 2, KORRIGALT_SZEGELY_b_l - 43,
(KORRIGALT_HEIGHT - KORRIGALT_SZEGELY_b_l-($i * 25)) - 8,
($legnagycorrwh)/10 * $i, $line_color);
        … }
```

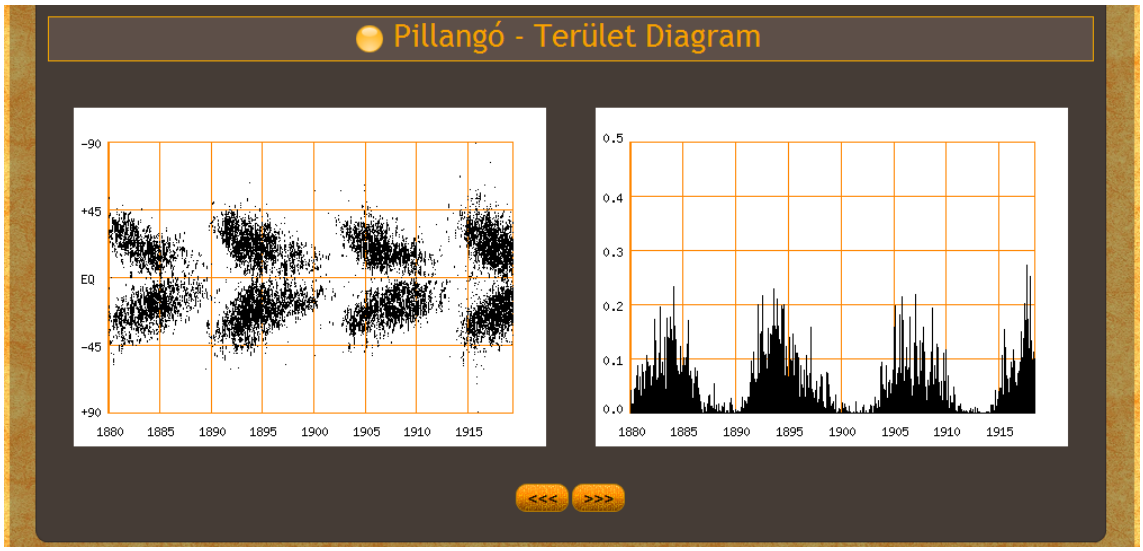
Az elkészült diagramot az imagepng() függvény segítségével megjelenítjük az oldalon, az imagedestroy() -val pedig felszabadítjuk a memóriát.

```
imagepng($img["korrigalt"], "images/korrigalt_alapkep.png");
imagedestroy($img["korrigalt"]);
```

Az Grafikon oldalra a Backspace billentyű gombbal, vagy a böngésző vissza gombjával léphetünk vissza.

6.4.2.1.2 Pillangó-, Terület diagram

A Napfoltábrázolás alatti részben található a Pillangó-, és Terület diagram (26. ábra).



26. ábra 1880 és 1919 közötti évek Pillangó-, és Területi diagramja.

Az ábrák alján található két vezérlőgomb (26. ábra), melyek segítségével léptethetjük kb. 20 évenként a dátumot előre és vissza.

A pillangó diagram az alábbi módon lett megvalósítva. A Napfelszínen a foltok megjelenési helyzete a ciklikusságnak megfelelően változik. A megadott dátum mindig az ábra közepén helyezkedik el, így mindig kb. +/-20 év közötti intervallumban ábrázoljuk. Az X tengelyen a foltok dátum szerinti elhelyezkedésüknek megfelelően jelennek meg ötévenként, az Y tengely értékei pedig a foltok szélességi körön való elhelyezkedésének koordinátái.

Minden egyes pixeloszlopra 40 naphoz tartozó adat van ábrázolva. Két értékre van szükségünk, a szélességre és a dátumra. Megvizsgáljuk, hogy ha a latitude kisebb mint 0°, akkor a 0° hosszúsági kör (EQ) alá fog esni,

```
$lat = round($lat *(-2));
imageline( ($img["pillango"]), (x1 + $i), (y1 + 120 + $lat), (x1 + $i), (y1 + 120 + $lat),
$pillango_line_color);
```

különben fölé esik. A pontokat egy imageline() függvény segítségével rajzoltuk, melynek kezdő-, és vég koordinátái ugyanazok.

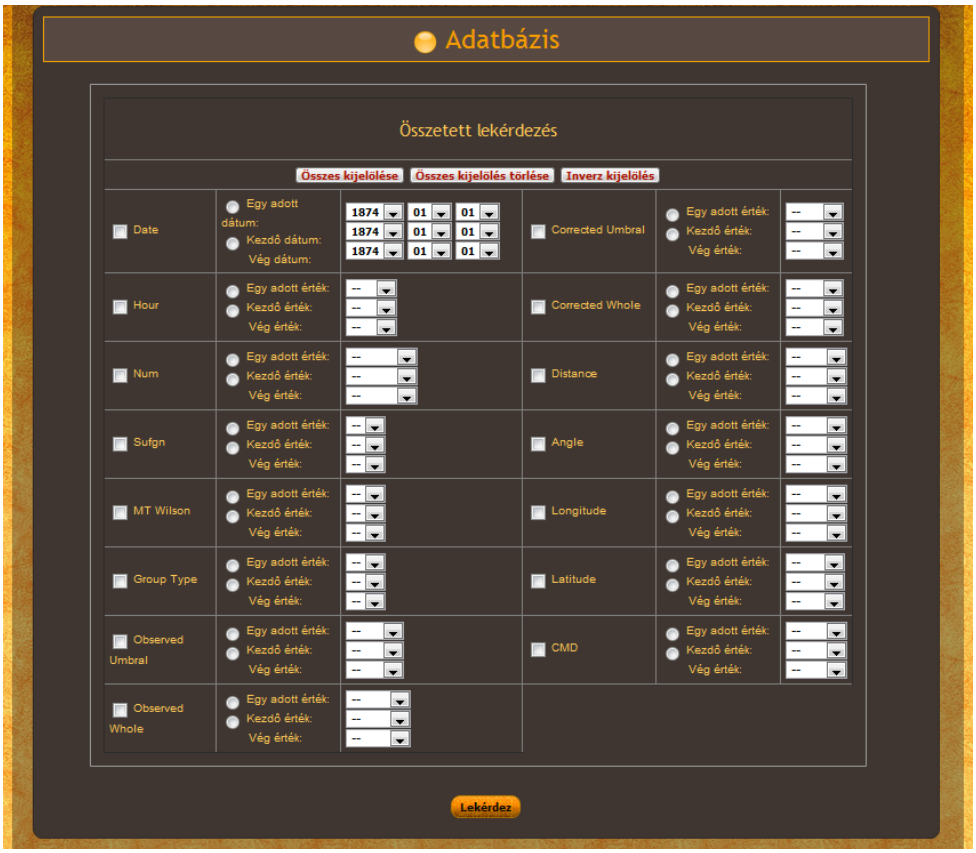
```
$lat = round($lat *(2));
imageline( ($img["pillango"]), (x1 + $i), (y1 + 120 - $lat), (x1 + $i), (y1 + 120 - $lat),
$pillango_line_color);
```

A terület diagram a következőképpen működik. Napciklussal együtt változik a foltok száma is. Kezdetben növekszik, majd lecsökken a foltok száma. Ezt ábrázolja a terület diagram (26.ábra). Az X tengely hasonlóképpen van beosztva, mint a pillangó diagramnál, az Y tengely értékei pedig a Nap felszínének milliomod részét mutatja. Minden egyes pixeloszlopra kirajzolunk 40 napra eső folt területének összegét.

```
for($i=1; $i<361; $i++)
        {
        $terulet = $obswh40[$i] / 1000;
        imageline ($img["terulet"], x1 + $i, TERULET_HEIGHT - y1, x1 + $i,
(TERULET_HEIGHT - y1 - round($terulet)), $terulet_line_color);
        }
```

6.4.2.2 Paraméterezhető adatbázis lekérdezések

Az Adatbázis menüpontban található egy összetett paraméterezhető adatbázis lekérdezés, ahol megtekintheti az adatokat a megadott feltételek alapján (27. ábra).



27. ábra Paraméterezhető adatbázis lekérdezés felülete.

A jelölőnégyzettel választhatóak ki az attribútumok. Rádiógombok segítségével választhatunk 'Egy adott érték', vagy 'Kezdő érték' és a 'Vég érték' között. Ha egy adott értéket választunk, akkor csak a kiválasztott értékre kereshetünk rá, ha csak kezdő értéket választunk, akkor a megadott érték és az ettől nagyobb értékekre keres, ha pedig Végértéket adunk meg, akkor a megadott érték, és az ettől kisebb értékeket listázza ki. Mindkét értéket kiválasztva az érték páros között kérdezi le.

További három funkció található az oldalon, melyek segítenek a gyors kiválasztásban. Az 'Összes kijelölés' gomb használatával kijelölhetjük az összes attribútumokat. Az 'Összes kijelölés törlése' gombra való kattintás után megszüntethetjük a kijelöléseket, alaphelyzetbe állíthatjuk az oldalt. Az 'Inverz kijelölés' -el pedig ellentett kijelölést hajthatunk végre. A

számunkra megfelelő érték kiválasztása és a 'Lekérdezés' gomb lenyomása után egy táblázatban megjelenik a lekérdezés eredménye (28. ábra).

28. ábra 1874-05-10 és 1875-01-01 dátum közötti adatok lekérdezése.

Egy oldalra 30 rekord adatot listázunk. Ha a lekérdezett adatmennyiség sora meghaladja ezt az értéket, akkor a táblázat alján megjelenik egy oldallapozás funkció, mellyel könnyedén tudunk az oldalak közt lapozni. Az 'Előző' és 'Következő' linkre kattintva egyesével lapozhatunk az oldalak között. Ha az oldalak száma meghaladja a 13-at, akkor megjelenik egy választó gomb, mellyel még könnyebben ugorhatunk a kívánt oldalra. Ennek azért van jelentősége, mert az oldalak száma akár az 100 -at is elérheti.
Ha a megadott feltételek nem teljesülnek a lekérdezés eredménye sikertelen lesz, ekkor egy hibajelzés jelenik meg (29. ábra).

29. ábra Az 1874-01-01 dátum lekérdezés eredménye.

Az előző oldalra a Backspace billentyű gombbal, vagy a böngésző vissza gombjával léphetünk vissza.

6.4.2.3 Fórum

Ezen az oldalon a fórumok listája található. Létrehoztunk egy társalgót, ahol a regisztrált felhasználó véleményt oszthat meg, vagy társaloghat a többi felhasználóval. A fórumtémákra kattintva megtekintheti a többi felhasználó hozzászólásait, illetve hozzászólhat egy már meglévő témához (30. ábra).

30. ábra Fórum kinézete.

Továbbá a felhasználók új fórumtémát küldhetnek be. Ez az opció elérhető az 'Új Fórumtéma beküldése' linkre kattintva, vagy a navigációs blokkban lévő Tartalom beküldése → Fórum topic menüponttal.

31. ábra Új fórumtéma hozzáadása.

A Tárgy címszónál megadjuk a beküldendő új téma nevét. Kiválasztjuk egy választó gomb segítségével a Fórum címszónál, hogy mely fórumba szeretnénk közé tenni. Megírjuk a beküldendő szöveget. Legvégül a Mentés gombra kattintva beküldjük (31. ábra).

6.4.2.4 Letöltések

Itt találhatjuk meg a letölthető dokumentumokat.

32. ábra Letölthető dokumentumok.

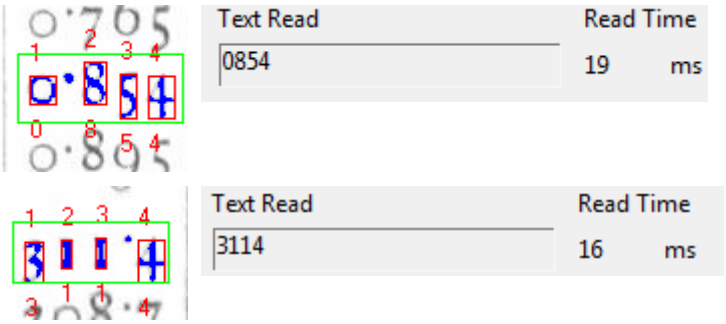
Letölthető a JAVA alkalmazás, amely internetkapcsolat nélkül is használható. Másrészt a program használatához letölthető a teljes adatbázis. Ha csak egy adott év adataira van szükség, akkor egy választó gomb segítségével kiválasztjuk az évet és a Letöltés gomb segítségével letöltjük az adatbázisból. A program használatához egy MySQL szerverre van szükségünk. Ha ezzel nem rendelkezünk, akkor egy link segítségével letölthetjük és az általunk készített videó segít a telepítésében mely a weblapba ágyazva megtekinthető (32. ábra).

6.4.2.5 Kapcsolat, Súgó

A Kapcsolat menüpontban található a készítők nevei, elérhetőségeik. Továbbá létrehoztunk egy Súgó menüpontot, ahol az oldal működését részletezzük.

7. Jövőbeli tervek

Tervben van egy részletesebb, 1800-as évekből származó adatbázis digitalizálása, mellyel, sokkal több információt kaphatunk a napfoltokról. A National Instruments Labview 8.6-os verziójával próbáltuk a képeket digitalizálni és az értékeket feldolgozni. Digitalizálás előtt szintre vágást használtunk, a Labview-val, hogy fekete-fehér képet kapjunk. A képek olyan speciális karakterkészlettel készültek, hogy a beolvasáshoz saját karakterkészletet készítettünk a Labview NI OCR nevű segédprogramjával.



33. ábra Labview OCR segédprogrammal készített karakterfelismerés

A fő problémát a tizedes pont okozta. A Labview-val nem tudtuk megoldani, hogy felismerje helyesen a számokat. A számjegyek beolvasása sikeres volt. A fő problémát a régies stílusú tizedes pont okozta, mivel azt a program figyelmen kívül hagyja. Ezt a gondot egyelőre nem sikerült kiküszöbölnünk. Ezen kívül több professzionális szoftverrel is próbálkoztunk feldolgozni a képeket, de eddig sikertelenül.

Folyamatban van egy speciális foltkereső elkészítése és a programba integrálása. Ezzel párhuzamosan a már meglévő kódot próbáljuk optimalizálni (memóriahasználat-, futási idő csökkentése). Szeretnénk több nyelvre (angol, német) is lefordítani a szoftvereket, ezáltal szélesebb körben használhatóvá tenni.

8. Összefoglalás

Munkánkkal a napfizikusok munkáját szerettük volna megkönnyíteni, elősegíteni. Reméljük, hogy ezzel hozzájárulunk a Nap mélyebb megismeréséhez és kutatásához. Egy elavult adathalmaz helyett létrehoztunk egy digitális formában tárolt és könnyen felhasználható modern adatbázist. Ennek során például több hibásan rögzített, mért értéket találtunk. Továbbá készítettünk egy olyan modern igényeknek megfelelő grafikus felületű alkalmazást mely az adatok megtekintését és megjelenítését segíti elő. Mindez online is megtekinthető a webes alkalmazás segítségével. A részletes adatbázis digitalizálása után további lehetőségek rejlenek a szoftverekben melyeket a későbbikben szeretnénk lehetőségeinkhez mérten kiaknázni.

9. Köszönetnyilvánítás

Ezúton szeretnénk köszönetet mondani Dr. Tóth László egyetemi adjunktusnak a munka során nyújtott folyamatos támogatásáért és segítőkészségéért.

Ezen kívül szeretnénk még megköszönni a www.prog.hu fórum aktív közösségének, akik szoftver fejlesztése közben felmerülő kérdésekben szakmai tudásukkal és segítőkészségükkel elősegítették a program sikerességét.

10. Munkamegosztás ismertetése

Molnár Csaba:

- Login
- Canvas osztályok (MyCanvas_FA, MyCanvas_OS, MyCanvas_ANI)
- Adatok osztály
- Esemény osztály (MyCanvaas_Action, Canvas_Klikk, Animáció) metódusai
- SQL osztály
- Főablak folt ábrázoló vászon
- Kép mentése
- Animáció
- Alkalmazás fordítása
- Súgó - az általam elkészített részekhez tartozó html oldalak

Siteri László:

- Adatbázis- szerver, kliens telepítése, beállítása
- Adatbázisszerkezet kialakítása
- Canvas osztályok (MyCanvas_Terulet, MyCanvas_Pillango, MyCanvasCorrTer, MyCanvasObsTer)
- Esemény osztály (Terület, Pillangó, Feltölt, Szerkezet_kialakító, Teljes) metódusai
- SQL osztály batchStatement metódusa
- Folt ablak
- Statisztikai diagramok
- Adatbázis menüpontok (Import, Export, Upload)
- Teljes diagram
- Súgó

Weinbach Gábor Áron:

- Drupal telepítése, konfigurálása.
- Web-alkalmazás megtervezése, elkészítése.

11. Irodalomjegyzék

[1] Tippan Erika, A naptevékenység, és földi hatásai Földrajzi szemináriumi dolgozat.

[2] Marik Miklós, Csillagászat, Akadémiai Kiadó, Budapest 1989

[3] Hédervári Péter, Csillagunk: a Nap (Magvető, 1980)

[4] http://www.windows2universe.org/spaceweather/images/sunspot_form.jpg

[5] http://en.wikipedia.org/wiki/Solar_rotation

[6] http://polaris.mcse.hu/2005._ev_archivuma/2005-oktober-kozponti-csillagunk-a-nap.html

[7] http://polaris.mcse.hu/2005._ev_archivuma/2005-oktober-kozponti-csillagunk-a-nap.html

[8] http://www.noaa.gov/

[9] ftp://ftp.ngdc.noaa.gov/STP/SOLAR_DATA/SUNSPOT_REGIONS/Greenwich/

[10] http://www.iit.uni-miskolc.hu/iitweb/export/sites/default/users/barabas/Targyak/db2/ea5.pdf

[11] Nagy Gusztáv: Drupal 6 alapismeretek Ad Librum Kiado 2010

[12] PHP: Documentation: http://www.php.net/docs.php

12. Függelék

12.1 MyCanvas absztrakt osztály

```
import java.awt.*;
import java.util.LinkedList;
import javax.swing.JTextField;

public abstract class MyCanvas_Os extends Canvas {

    int x, y, XMAX, YMAX, XMIN, YMIN;
    LinkedList<Adatok> Foltok = new LinkedList<Adatok>();
    JTextField hibasav;
    BufferedImage alap;

    public MyCanvas_Os() {
    }

    public MyCanvas_Os(int x, int y, JTextField jTextField1) {
        this.setSize(x, y);
        this.x = x;
        this.y = y;
        hibasav = jTextField1;
        alap = new BufferedImage(x, y, BufferedImage.TYPE_INT_RGB);
        rajzol(alap.createGraphics());
    }

    @Override
    public void update(Graphics graphics) {
        paint(graphics);
    }

    void setAdatok(Adatok adatok) {
        Foltok.add(adatok);
    }
...
    void rajzol(Graphics graphics) {
        XMIN = 0;
        XMAX = x - 1;
        YMAX = y - 1;
        YMIN = 60;

        graphics.setColor(Color.white);
        graphics.fillRect(0, 0, x, y);

        graphics.setColor(Color.LIGHT_GRAY);
        graphics.fillRect(XMIN, YMIN, XMAX / 4, YMAX);
```

```
        graphics.fillRect(XMAX * 3 / 4, YMIN, XMAX, YMAX);

        graphics.setColor(Color.BLACK);
        graphics.drawLine(XMIN, (YMIN + YMAX) / 2, XMAX, (YMIN + YMAX) / 2);   // x
tengely
        graphics.drawLine(XMAX / 2, YMIN, XMAX / 2, YMAX);         // y tengely

        graphics.drawLine(XMAX / 4, YMIN, XMAX / 4, YMAX);     // -90 fok
        graphics.drawLine(XMAX / 3, YMIN, XMAX / 3, YMAX);    // -60 fok

        graphics.drawLine(XMAX * 2 / 3, YMIN, XMAX * 2 / 3, YMAX); // +60 fok
        graphics.drawLine(XMAX * 3 / 4, YMIN, XMAX * 3 / 4, YMAX); // +90 fok

        graphics.drawString("A nap aktív vidékei", (XMAX / 2) - 50, 15);
        graphics.drawString("-90", XMAX / 4 - 10, 35);   //felül
        graphics.drawString("+90", XMAX * 3 / 4 - 13, 35);  //felül
        graphics.drawString("0", XMAX / 2 - 4, 35);
        graphics.drawString("-60", XMAX / 3 - 10, 35);    //felül
        graphics.drawString("+60", XMAX * 2 / 3 - 13, 35);  //felül
        graphics.drawString("+60", XMIN + 5, 55);     //felül
        graphics.drawString("-60", XMIN + 5, 350);    //alul
        graphics.drawString("+60", XMAX - 25, 55);    //felül
        graphics.drawString("-60", XMAX - 25, 350);   //alul


        /* keret kirajzolása */
        graphics.drawLine(XMIN, YMAX, XMAX, YMAX);
        graphics.drawLine(XMIN, YMIN, XMAX, YMIN);
        graphics.drawLine(XMIN, YMAX, XMIN, YMIN);
        graphics.drawLine(XMAX, YMIN, XMAX, YMAX);
    }

    public abstract void paint(Graphics graphics);
}
```

12.2 Animáció osztály fontosabb részei

```
int DELAY = 500;
    MyCanvas_Ani ani_canvas;
    Alkot alkot;
    Kitesz kitesz;
    LinkedList<BufferedImage> puffer = new LinkedList<BufferedImage>();
    int hossz = 5;
    String[] acc;
```

```
boolean stop_state = false;
Calendar kezdo, veg;

public Animation(String[] acc) {
   initComponents();
   this.acc = acc;
}

public Calendar StringToCal(String datum) {
   Calendar c = null;
   SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
   try {
      Date d = sdf.parse(datum);
      c = Calendar.getInstance();
      c.setTime(d);
   } catch (ParseException e) {
      jTextField1.setText("Hibas dátum!");
   }
   return c;
}

public String CalToStr(Calendar c) {
   SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
   return sdf.format(c.getTime());
}

public void kiir_datum(Calendar datum) {
   String Str = CalToStr(datum);
   jLabel1.setText("Év: " + Str.substring(0, 4));
   jLabel2.setText("Hónap: " + Str.substring(5, 7));
   jLabel3.setText("Nap: " + Str.substring(8, 10));
}

public synchronized BufferedImage getAdat(Thread t1) {
   while (puffer.size() == 0) {
      try {
         wait();
      } catch (InterruptedException e) {
      }
   }
   BufferedImage image = puffer.pollFirst();
   notify();
   return image;
}

public synchronized void putAdat(Thread t2, BufferedImage kep) {
   while (puffer.size() == hossz) {
      try {
```

```
            wait();
        } catch (InterruptedException e) {
        }
    }
    puffer.addLast(kep);
    notify();
}

private void Close(java.awt.event.WindowEvent evt) {
    stop_state = true;
}

private void PlayActionPerformed(java.awt.event.ActionEvent evt) {
    String lekerdezes_datum;
    String veg_datum;
    lekerdezes_datum = jComboBox1.getSelectedItem().toString() + "-";
    lekerdezes_datum += jComboBox2.getSelectedItem().toString() + "-";
    lekerdezes_datum += jComboBox3.getSelectedItem().toString();
    veg_datum = jComboBox4.getSelectedItem().toString() + "-";
    veg_datum += jComboBox5.getSelectedItem().toString() + "-";
    veg_datum += jComboBox6.getSelectedItem().toString();
    kezdo = StringToCal(lekerdezes_datum);
    veg = StringToCal(veg_datum);
    if (kezdo.compareTo(veg) <= 0) {
        if ((alkot == null) || (!alkot.isAlive())) {
            puffer.clear();
            alkot = new Alkot("Termelo", this, acc);
            kitesz = new Kitesz("Fogyaszto", this, CalToStr(kezdo));
            stop_state = false;
            alkot.start();
            kitesz.start();
        } else {
            jTextField1.setText("Még fut az animáció");
        }
    }
}

private void StopActionPerformed(java.awt.event.ActionEvent evt) {
    stop_state = true;
}

private void jSlider1MouseMoved(java.awt.event.MouseEvent evt) {
    DELAY = jSlider1.getValue();
}

class Alkot extends Thread {

    Animation a;
```

```
    SQL sql;
    int x = 900;
    int y = 360;
    BufferedImage rajz;

    private Alkot(String name, Animation aThis, String[] acc) {
       super(name);
       sql = new SQL(acc);
       a = aThis;
    }

    @Override
    public void run() {
       lekerdez(CalToStr(kezdo), CalToStr(veg));
       while (kezdo.compareTo(veg) <= 0) {
          if (stop_state) {
             break;
          }
          BufferedImage picture = new_elem(kezdo);
          a.putAdat(this, picture);
          kezdo.add(Calendar.HOUR, 24);
       } //while vége

       a.putAdat(this, null);
    }

    private void lekerdez(String kezdo, String vegso) {
       StringBuilder str = new StringBuilder("SELECT
DATE,num,lng,lat,cmd,s_umb,s_wh FROM test.arta WHERE DATE BETWEEN '");
       str.append(kezdo);
       str.append("'  AND '");
       str.append(vegso);
       str.append("' order by date asc");
       try {
          ResultSet rs = null;
          Object eredmeny = sql.lekerdez(str.toString());
          if (eredmeny instanceof String) {
             jTextField1.setText((String) eredmeny);
          } else {
             rs = (ResultSet) eredmeny;
             ani_canvas.clearAdatok();
             while (rs.next()) {
                ani_canvas.setAdatok(new Adatok(
                      rs.getDate(1), //dátum
                      rs.getInt(2), //folt azonosító
                      rs.getDouble(3), //Longitude
                      rs.getDouble(4), //Latitude
                      rs.getDouble(5), //CMD
```

```
                        rs.getDouble(6), //S_UMB
                        rs.getDouble(7) //S_WH
                        ));
                }
            } // else vége
        } catch (Exception e) {
            jTextField1.setText("Hiba az animaciós lekerdezésnél: " + e.toString());

        }
    }

    private BufferedImage new_elem(Calendar vegso) {
        rajz = new BufferedImage(x, y, BufferedImage.TYPE_INT_RGB);
        rajz.createGraphics();
        ani_canvas.foltok(rajz.getGraphics(), kezdo);
        return rajz;
    }
}

class Kitesz extends Thread {

    Animation a;
    BufferedImage pic;
    Graphics graphics;
    Calendar kitesz_datum;

    Kitesz(String name, Animation aThis, String datum) {
        super(name);
        a = aThis;
        graphics = ani_canvas.getGraphics();
        kitesz_datum.getInstance();
        kitesz_datum = StringToCal(datum);
    }

    @Override
    public void run() {
        for (pic = a.getAdat(this); pic != null; pic = a.getAdat(this)) {
            try {
                sleep(DELAY);
            } catch (InterruptedException e) {
            }
            graphics.drawImage(pic, 0, 0, null);
            a.kiir_datum(kitesz_datum);
            kitesz_datum.add(Calendar.HOUR, 24);
        }
    }//run metódus vége
}
```

12.3 Fájl mentése

```
try {
        JFileChooser SF = new JFileChooser();
        SF.setCurrentDirectory(new File("."));
        SF.setSelectedFile(new File(jComboBox1.getSelectedItem().toString()));
        SF.setFileFilter(new FileFilter() {

           public boolean accept(File f) {
              return f.getName().toLowerCase().endsWith(".jpg")
                    || f.isDirectory();
           }

           public String getDescription() {
              return "JPG Files";
           }
        });

        SF.showSaveDialog(this);
        String path = SF.getSelectedFile().getPath();
        if ((path.substring(path.length() - 3, path.length()).compareTo("jpg")) != 0) {
           canvas1.saveToFile(SF.getSelectedFile().getPath() + ".jpg");
        } else {
           canvas1.saveToFile(SF.getSelectedFile().getPath());
        }
     } catch (Exception e) {
        JOptionPane.showMessageDialog(this,
              "Mentés megszakítva!",
              "Hiba",
              JOptionPane.ERROR_MESSAGE);
     }
BufferedImage makeImage() {
      int type = BufferedImage.TYPE_INT_RGB;
      BufferedImage image = new BufferedImage(x, y, type);
      Graphics2D g2 = image.createGraphics();
      g2.drawImage(alap, 0, 0, null);
      this.paint(g2);
      g2.dispose();
      return image;
   }

   private void save(BufferedImage image, String utvonal) {
      File file = new File(utvonal);
      try {
         ImageIO.write(image, "jpg", file);
      } catch (IOException e) {
         hibasav.setText("Írás hiba: " + e.getMessage());
      }
```

```
    }

    void saveToFile(final String utvonal) {
        new Thread(new Runnable() {

            public void run() {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    hibasav.setText("File mentés interrupted:" + e.getMessage());
                }
                save(makeImage(), utvonal);
            }
        }).start();
    }
```

12.4 Adatbázis szerkezet kialakítása

```
public String szerkezet_kialakito() {
        String vissza = "";
        //SZERKEZET KIALAKÍTÁSA
        SQL kapcsolat = new SQL(acc);
        //ADATBÁZIS LÉTREHOZÁSA
        Object result = null;
        result = kapcsolat.vegrehajt("CREATE DATABASE test;");
        if (result instanceof String) {
            if (result.equals("Can't create database 'test'; database exists")) {
                vissza += "Az adatbázis már létezik! ";
            }
        } else {
            vissza += "Adatbázis létrehozva! ";
        }
        //TÁBLA LÉTREHOZÁSA
        result = null;
        result = kapcsolat.vegrehajt("CREATE TABLE sarta.arta("
                + "DATE DATE NOT NULL,"
                + "ORA INT(2) NOT NULL,"
                + "NUM INT(8) NOT NULL,"
                + "SUFGN INT(1) NOT NULL,"
                + "MTW TINYINT(1) DEFAULT NULL,"
                + "GRTP TINYINT(1) DEFAULT NULL,"
                + "OBSUMB SMALLINT(3) DEFAULT NULL,"
                + "OBSWH SMALLINT(3) DEFAULT NULL,"
                + "CORRUMB SMALLINT(3) DEFAULT NULL,"
                + "CORRWH SMALLINT(3) DEFAULT NULL,"
                + "DIST FLOAT DEFAULT NULL,"
                + "ANG FLOAT DEFAULT NULL,"
```

```
            + "LNG FLOAT DEFAULT NULL,"
            + "LAT FLOAT DEFAULT NULL,"
            + "CMD FLOAT DEFAULT NULL,"
            + "S_UMB FLOAT DEFAULT NULL,"
            + "S_WH FLOAT DEFAULT NULL,"
            + "PRIMARY KEY (ORA,NUM,SUFGN,DATE));");
    if (result instanceof String) {
        if (result.toString().compareTo("Table 'arta' already exists") == 0) {
            vissza += " A tábla már létezik!";
        } else { vissza += result.toString(); }
    } else { vissza += " Tábla létrehozva!"; }
    return vissza;
}
```

Nyilatkozat

Alulírott Molnár Csaba, Siteri László, Weinbach Gábor Áron, a Debreceni Egyetem Informatikai Karának hallgatói ezennel büntetőjogi felelősségünk tudatában nyilatkozunk és aláírásunkkal igazoljuk, hogy Grafikus adatbázis-kezelő és adatfeldolgozó szoftver napfizikai alkalmazáshoz című dolgozatunk saját, önálló munkánk; az abban hivatkozott nyomtatott és elektronikus szakirodalom felhasználása a szerzői jogok nemzetközi szabályainak megfelelően készült.

Tudomásul vesszük, hogy dolgozat esetén plágiumnak számít:

• szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;

• tartalmi idézet hivatkozás megjelölése nélkül;

• más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírottak kijelentjük, hogy a plágium fogalmát megismertük, és tudomásul vesszük, hogy plágium esetén dolgozatunk visszautasításra kerül.

Debrecen, 2011. április 27.

....

Molnár Csaba Siteri László Weinbach Gábor Áron