DIPLOMAMUNKA

Mozgásvizsgálati mérések eredményeinek internetes megjelenítése

Zemkó Szonja

Budapesti Műszaki és Gazdaságtudományi Egyetem
Építőmérnöki Kar, Általános- és Felsőgeodézia Tanszék
Építőipari geodéziai szakirány

2011

Tartalomjegyzék

1. Bevezetés....6
2. Mozgásvizsgálatok geodéziai módszerei....8
2.1. A mozgásvizsgálatokról általában....8
2.1.1. A mozgásvizsgálatok szükségessége....8
2.1.2. Geodéziai magasságmérési módszerek....9
2.1.3. Mozgásvizsgálatok GPS-szel....10
2.2. Automatizált mozgásvizsgálati módszerek....10
2.2.1. A CYCLOPS rendszer....11
2.2.2. A CENTAUR rendszer....12
2.2.3. Az automata rendszer elemei....13
2.2.4. Mozgásvizsgálati eredmények megjelenítése....14
3. Informatikai technológiák....17
3.1. Digitális térképek....17
3.1.1. Geoinformációs rendszerek....17
3.1.2. Internetes térképi technológiák....18
3.2. Téradatbázisok....23
3.2.1. PostgreSQL és PostGIS....25
3.3. Web programozási környezetek....27
4. Saját rendszer elkészítése....32
4.1. Rendszerkoncepció....32
4.2. Logikai rendszerterv....33
4.2.1. Felhasznált szoftverkomponensek és telepítésük....36
4.3. Megvalósítás....38
4.3.1. PostgreSQL adattáblák készítése....38
4.3.2. A szerveroldali műveletek végrehajtása....40
4.3.3. Megjelenítés Google Maps API-val....44
4.3.4. Űrlap és táblázat készítése....47
4.3.5. Az OWT Chart alkalmazása....48
4.3.6. Extrák....49
4.4. Tesztelés....50
4.5. Felhasználói felület....51
5. Összefoglalás....53

Köszönetnyilvánítás

Nagyon sok köszönettel tartozom Siki Zoltán tanár úrnak, aki lehetőséget adott nekem arra, hogy bepillantást nyerjek az informatika rejtekeibe és elsajátítsak rengeteg hasznos elméleti és gyakorlati ismeretet. Köszönöm a tanár úrnak a félév során nyújtott áldozatos munkáját, a töretlen lelkesedését és a jó hangulatú konzultációkat, amelyekkel nagyban hozzájárult a feladat sikeréhez.

Nyilatkozat

Alulírott Zemkó Szonja, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy a diplomamunkát meg nem engedett segítség nélkül, saját magam készítettem, és a dolgozatban csak az irodalomjegyzékben megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2011. május 22.

.....

Zemkó Szonja

Abstract

Publication of Settlement Measurement Results Via the Internet

The aim of my diploma work is to create a database to store large amount of data acquired using automated surveying systems and to map these items on Google Maps as well as to create a settlement/time graph. This plan should be carried out with the help of Google Maps API V3.

The construction of Metro 4 in Budapest is accompanied by control surveys done by the SolData-HUNGEOD Consortium. One of the utilized survey methods is the real-time monitoring system. Buildings along the alignment are equipped with reflective targets in order to detect the vertical and horizontal movement of the exterior. The automatic surveying instruments operate 24/7.

I have chosen the CH-Building in the University Campus as the subject of my application. I was given the vertical datas of 38 targets on the facade in a five-year interval, one data per day.

During this project I decided to use open source software exclusively. For data storage I have chosen PostgreSQL with its PostGIS plug-in and the web based application was designed with web programming languages like JavaScript and PHP exploiting some of their up-to-date features (AJAX, JSON).

1. Bevezetés

Az információs rendszerek és a technika rohamos fejlődésének köszönhetően az informatika szerepe a mindennapi életben az utóbbi néhány évben robbanásszerűen megnövekedett. Az Európai Unió is arra törekszik, hogy a polgárok és a vállalkozások széles körben részesüljenek az információs társadalom nyújtotta előnyökből. Szinte már nincs is olyan szakterület, mely ne kapcsolódna valamilyen szinten az informatikához, a mérnöki tudományok esetében pedig különösen fontos szerepük van az informatikai eszközöknek, mivel a műszaki szakmák önmagukban ma már kevésbé állják meg a helyüket. A gyorsabb feldolgozás és a praktikusabb eredmény-megjelenítés érdekében az informatikai eszközök legalább felhasználói szintű ismerete elengedhetetlen, arról nem is beszélve, hogy a széleskörű felhasználói kör ellátása és a távolság leküzdése az interneten keresztül a legegyszerűbb. Mivel az információs társadalom fontos szempontként tartja számon az információ beszerzésére fordított energiát és időt, ezért az alkalmazni kívánt eszközeink kiválasztását ezen paraméterek döntően befolyásolják. A számítástechnika számos megoldást kínál az informatikai jellegű problémáink megoldásához, melyek között léteznek úgynevezett nyílt forráskódú (*open source*) szoftverek, amiket ingyenesen és legálisan letölthetünk, használhatunk és terjeszthetünk.

Szakdolgozatom célkitűzése az, hogy automatikus geodéziai mérőrendszerek mérési eredményeit az interneten keresztül mások számára elérhetővé tegyem. Általában mind a szakmabeli, mind a hétköznapi felhasználók a vizuális és könnyedén kezelhető megoldásokat részesítik előnyben, ezért az információ közlésekor a külsőségek is fontos szereppel bírnak. Az informatikai megoldások segítségével nemcsak a nyers adatok juttathatóak el könnyedén és ráadásul egyszerre több helyre, hanem azok modellezéséhez és elemzéséhez is számos lehetőség adott. A feladatom tehát kibővült az adatgyűjtés helyének térképen való ábrázolásával és a mérési eredményeink grafikus megjelenítésével. A munkám elkészítése során többfajta szoftverrel és módszerrel ismerkedek meg, melyek közül minden esetben az ingyenesen felhasználhatóakat és azok közül is a legkorszerűbbeket részesítem előnyben. Fontosnak tartottam, hogy az általam készítendő alkalmazással azon felhasználóknak is igyekezzek megfelelni, akik hozzászoktak a technológia élvonalában járó megoldásokhoz. A modern és széleskörűen alkalmazható eszközöknek köszönhetően a tervezett programrendszer funkcióját tekintve maximálisan kielégíti a vele szemben támasztott igényeket, mindemellett pedig az esztétikus megjelenítésről sem kell lemondanom.

A geodézia és a térinformatika tudománya szorosan összekapcsolódik, ezért a mérési eredményeink szemléltetése érdekében és további kiértékelések végzéséhez térinformatikai megoldásokat kell igénybe venni. A diplomamunkám lényegét jelentő alkalmazáshoz – a látványos szemléltetés érdekében – nagy mennyiségű eredményt tartalmazó és hosszú időtartamot átfogó adatsorra volt szükség. A jelenleg zajló budapesti 4-es metró projekt kiváló lehetőséget nyújtott arra, hogy megismerkedjek a beruházás során alkalmazott komplex mérőberendezéssel és az általa kinyert adatokra támaszkodva készíthessem el a tervezett programrendszert. Esetünkben tehát a méréskor alkalmazott eszköz egy igen korszerű és automatikus működési elvű geodéziai műszer.

Ugyan a programrendszer megosztása több lehetséges módon is megvalósulhatna, én az interneten elérhető szolgáltatást tűztem ki célnak. Az internetes tartalmak minőségéről az 1994 óta létező W3C (World Wide Web Consortium) gondoskodik, hiszen célja az, hogy a web minél hatékonyabban működjön, ezért fő tevékenysége a webes szabványok kidolgozása. Tagjai között megtalálhatóak a jelentősebb informatikai vállalatok.

A szakdolgozat szöveges része két nagy egységből áll, az első részben a programrendszer elkészítéséhez szükséges ismereteket és eszközöket tárgyalom, a másodikban magát a megvalósítást ismertetem. Az előkészületek bemutatására két fejezetet szentelek, mivel úgy érzem, hogy a geodézia és az informatika szakterületét érdemes külön tárgyalni. A geodézia területén többnyire otthonosan mozgok, az alkalmazandó módszerek alapelveiről már tanultam és gyakorlatban is használtam őket, ezért a korábban megszerzett ismereteimről csak bevezetőként szeretnék említést tenni. A szakdolgozatom feladatához választott adatgyűjtési technológia azonban rengeteg meglepetést tartogat számomra, mivel automatikus mozgásvizsgálati rendszerrel testközelben most találkozok először. A **Mozgásvizsgálatok geodéziai módszerei** című fejezetben tehát főként ezen technológiával kapcsolatos tapasztalataimat szeretném megosztani. Az ismeretek megszerzésének lehetősége és a feladatom inspirációja a 4-es metró nyomvonalán üzemelő, SolData fejlesztésű CYCLOPS és CENTAUR rendszerekről kapott információknak köszönhető, ezért a szakdolgozat keretein belül elsősorban az ő szemszögükből tárgyalom az automatikus mozgásvizsgálati módszereket.

A geodéziához szorosan kapcsolódó térinformatika szakterülete szintén egy jelentős részét képezi az eddigi egyetemi tanulmányaimnak. A szakdolgozatom elkészítésében a térinformatika mint segédtudomány vesz részt, ezért a feladatom ismertetése során sűrűn említést teszek térinformatikai vonatkozású fogalmakról és megoldásokról. Az **Informatikai technológiák** címmel ellátott fejezetet különböző szoftverek és technológiák ismertetésére szentelem. Az informatikának a térinformatikán kívül eső világába idáig csak hétköznapi felhasználóként láttam bele, illetve bizonyos szakmai szoftverek alapszintű ismeretét sajátítottam el. A programírást és a különböző webes technológiákat illetően rengeteg újdonsággal kell megismerkednem. A sikeres feladatmegvalósítás érdekében utánajárok az interneten keresztül elérhető térképes megjelenítési lehetőségeknek, az adattárolási módoknak és annak, hogy hogyan lehet a böngészőn keresztül mindezt megjeleníteni és egyúttal a felhasználó számára interaktívvá tenni.

A gyakorlati munkafolyamatok leírása és a végeredmény bemutatása a **Saját rendszer elkészítése** című fejezetben tartalma. Ahhoz, hogy az olvasó számára jól át tudjam adni a munkám lényegi mozzanatait, a szöveges ismertető mellé igyekszem illusztrációkat is mellékelni; pl. az alkalmazott többféle programozási nyelv egyedi struktúráit a programsorokból kiragadott részletekkel szeretném bemutatni, a munka egyes lépéseinek „kézzelfogható” eredményeit pedig a böngészőben látható képek beillesztésével teszem szemléletessé.

A feladatom elkészítésében segítségemre volt a SolData-HUNGEOD Konzorcium munkatársa, Dömötör Krisztina, aki eljuttatta számomra a mérési eredményeket és megosztotta velem a beruházással kapcsolatos publikus információkat.

A szakdolgozat szöveges része a nyílt forráskódú OpenOffice Writer programban készült.

2. Mozgásvizsgálatok geodéziai módszerei

2.1. A mozgásvizsgálatokról általában

Az építménymozgást két szempontból vizsgáljuk. Elmozduláson értjük az objektum környezetéhez viszonyított helyváltoztatását, alakváltozáson pedig az épület egyes részeinek a többi részhez viszonyított mozgását. Az elmozdulásokat és alakváltozásokat függőleges vagy vízszintes komponensekre bonthatjuk, melyek a mozgás jellegétől függően lehetnek maradandóak vagy rugalmasak.

A mozgások bekövetkezése külső vagy belső hatásokra vezethető vissza. Külső erő alatt értjük a talaj- és rétegvíznyomást, a szélnyomást (tornyok), a használatból következő terhelést (hidak, pillérek, falak) vagy az építmény alatt közvetlenül, illetve közelében zajló földalatti tevékenységeket (alagútépítés, csatornázás). A belső hatások az építmény anyagának fizikai és kémiai tulajdonságainak megváltozásából adódnak (beton zsugorodása, fal ülepedése, stb.).

2.1.1. A mozgásvizsgálatok szükségessége

Ezen folyamatok mérése rendkívüli jelentőséggel bír, ezért mozgásra érzékeny és megóvandó építmények mozgásellenőrzését kötelezően előírják. Az épületek kiválasztását a megrendelő végzi, döntése függ az anyagiaktól, a minőségi- és biztonsági előírásoktól, az épület értékétől, műemlék jellegétől és a károsodás előrelátható mértékétől. A vizsgálatok szolgálhatják a károk előrejelzését, illetve már bekövetkezett károk esetén pedig a mozgás időbeli-, térbeli lefolyásáról adnak információt. Az adatok segítségével előrejelzést, riasztást, kárbecslést, ok kutatást, tovább tervezést tudnak végezni a szakemberek. Továbbá a későbbiekben megépítendő alépítmények, szerkezetek méretezéséhez is elengedhetetlen a már meglévő épület mozgási jellemzőinek ismerete, melyeket az új létesítmény kivitelezésekor folyamatosan ellenőrizni kell.

Geodéziai jellegű vizsgálatok alatt értjük a nagyon kis mértékű (mm vagy annál kisebb nagyságrendű) mozgások meghatározását, valamint a hozzájuk kapcsolódó fizikai mennyiségek (hőmérséklet, páratartalom, feszültség, talajvízszint, stb.) mérését. Mivel az egyes mozgásvizsgálati feladatok különbözőek, ezért minden esetben egyedileg kell megtervezni – az ismert módszerek és irányelvek alapján – a legcélszerűbb megoldást.

A diplomamunkám feladatában elmozdulásméréssel foglalkozom. A létesítmény egyes pontjainak az elmozdulását meghatározhatjuk a tervezett helyzethez vagy az alaphelyzethez képest, az utóbbi pedig lehet abszolút vagy relatív érték. Az elmozdulásmérés geodéziai szempontból egyfajta leegyszerűsítő modellezéssel valósul meg, ami a létesítményen megjelölt pontok ismételt meghatározását jelenti. Az elmozdulásmérés általános elvei:

- Az alaphelyzetet rögzítő pontokat az épület fő statikai elemeibe, stabil, mozgásmentes helyen kell elhelyezni, hogy a mérési eredményeink csak a vizsgálandó mozgásokat tartalmazzák. A felhasznált pontok lehetőleg a mérés teljes ideje alatt felhasználhatóak legyenek, mert egyéb esetben a megbízhatóság erősen csökkenhet.
- A pontossági követelményeket és a mérések gyakoriságát a kivitelezés jellege, annak felszíni és felszín alatti kockázati tényezője határozza meg minden egyes feladat esetén.

• A mérési módot, a mérőfelszerelést és a pontok állandósítási módját az adott körülményeknek és a feladat jellegének megfelelően kell kiválasztani.

• A mérési és számítási módok közül mindig a megkívánt pontosság eléréséhez szükséges legegyszerűbb megoldást kell választani. A táblázatos adatközlés nehezen értelmezhető, ezért javasolt az eredményeket grafikonon vagy térbeli (izovonalas, szintvonalas) rajzon, metszeti ábrán megjeleníteni.

• A kapott eredmények hibákkal terheltek, ezért meg kell adni a mérések megbízhatóságát.

Az elmozdulások lehetnek függőleges vagy vízszintes irányúak, bár a feladatomban csak függőleges irányú mozgásokkal (süllyedésekkel) foglalkozok. A süllyedések oka összefügg a talajszerkezettel, a víztartalommal, a vízmozgásokkal, a kivitelezés közelségével, a kivitelezés mélységével, a víztelenítési eljárással, az épület állagával, stb. A talaj tulajdonságait és a várható süllyedés értékét annál pontosabban lehet meghatározni, minél több vizsgálatot végeznek (természetesen ilyenkor is adódhatnak váratlan helyzetek). A süllyedésmérések célja a süllyedés ellenőrzése, abszolút vagy relatív nagyságának meghatározása és további adatok szolgáltatása (pl. a süllyedések térbeli és időbeli lefolyása).

A folyamatok jellemzésére használható a keresztkorreláció. A keresztkorreláció egy standard módszer ahhoz, hogy megbecsüljük két adatsor korrelációjának mértékét. Vegyünk két adatsort, ***x***(i)-t és ***y***(i)-t, ahol i=1,2,…(n-1). A keresztkorreláció (***r***) ***d*** eltolás függvényében a következőképpen alakul:

$$r(d)=\frac{\sum_i [(x_i-mx)*(y_{(i-d)}-my)]}{(\sqrt{\sum_i (x_i-mx)^2}*\sqrt{\sum_i (y_{(i-d)}-my)^2})}$$

ahol m***x*** és m***y*** az adatsorok várható értékei. Ha ***r*** értékét minden ***d***=1,2,…(n-1) értékre kiszámítjuk, akkor a kapott értéksor kétszer olyan hosszú lesz, mint az eredeti adatsorok. Természetesen a keresztkorreláció kis mértékű eltolás esetén is vizsgálható. A ***d*** értéke −1 és 1 között mozoghat. 1 és −1 esetén maximum korrelációról beszélünk, bár az utóbbi esetén a két adatsor éppen egymás inverze. 0 esetén egyáltalán nincsen korreláció. [BOUR]

A süllyedésvizsgálatok esetén a keresztkorrelációt például arra használhatjuk, hogy az egyes pontok idősorait összehasonlítsuk, és kimutassuk a pontok mozgása közötti összefüggést, vagyis azt, hogy az egyik pont mozgása késleltetéssel megfelel egy másik pont mozgásának. Egy fúrópajzs mozgása esetén várható, hogy késleltetéssel azonos mozgásokat tapasztalunk a vizsgálati pontokban.

2.1.2. Geodéziai magasságmérési módszerek

Süllyedésmérés esetén leggyakrabban alkalmazott geodéziai módszer a *szabatos szintezés*. A szabatos kifejezés arra utal, hogy a méréseket az országos első-, másod- és harmadrendű szintezés pontosságának megfelelően kell végezni. Az észleléshez alkalmazott eszköz általában a libellás vagy kompenzátoros szintezőműszer. Amennyiben abszolút elmozdulásokat is meg szeretnénk határozni, szükséges a mérési helyszín közelében, de mozgásmentesnek tekinthető helyen magassági alappontok létesítése.

Olyan esetekben, mikor a mérési pontok nehezen vagy egyáltalán nem megközelíthetőek, a süllyedésvizsgálatot a *trigonometriai magasságmérés* módszerével is végezhetjük, ez esetben két pont magasságkülönbségét a

$$m = h - H + t * \cot(z)$$

képlettel adhatjuk meg, ahol *h* a műszermagasságot, *H* a jelmagasságot, *t* a pontok közti távolságot, *z* pedig a zenitszöget jelenti. Süllyedésmérés esetén azonban célszerűbb a nagyobb pontosságot biztosító *trigonometriai szintezés* módszerét alkalmazni. Ennek lényege, hogy egy műszerállásból két pont magasságkülönbségét határozzuk meg, ezáltal a műszermagasság és az őt terhelő hiba értéke kiejthető. A trigonometriai szintezés kis *t* távolságok esetén eléri a szabatos szintezés pontosságát. [ÓDOR]

2.1.3. Mozgásvizsgálatok GPS-szel

A műholdas helymeghatározás relatív módszere alkalmas szabatos elmozdulásvizsgálatra, amennyiben a vizsgált pontok nem zárt térben vannak és biztosítható az égboltra való szabad kilátás. A GPS mérések pontossága természetesen elmarad a hagyományos geodéziai módszerekétől, de nagy előnye, hogy a mozdulatlannak tekintett viszonyítási pontban lévő referenciavevő és a mozgó vevő vizsgálati pontjai között nem szükséges semmilyen összeköttetés és az összelátást sem kell biztosítani. Célszerű két referenciapontot kialakítani, hogy ellenőrizhető legyen a mozdulatlanságuk. A mozgásvizsgálatokhoz a legnagyobb pontosságú (statikus) módszerre van szükség, amely magasságmeghatározás esetén a milliméteres pontosságot is elérheti. Nagy kiterjedésű felszín süllyedésének vizsgálatakor a GPS-szel folytatott mérés igen praktikus, mivel a mozdulatlannak tekintett viszonyítási pontok annyira távol kerülnek a vizsgálati pontoktól, hogy a hibahalmozódás miatt bizonytalan eredmény adódna, a GPS-es süllyedésvizsgálat pontossága azonban 10 km-ig független a távolságtól. [KRAU]

2.2. Automatizált mozgásvizsgálati módszerek

A budapesti metróépítési projekt mérési programja az következők szerint zajlik: a 4-es metró beruházásának vezetője a DBR Metró. A Megrendelő és a Mérnök feladata közösen a Vállalkozóval, hogy megrendelje az állomások és környezetük mozgásvizsgálatát. A mozgásvizsgálatok végzésének pontos helyszínét (megfigyelést igénylő területek, épületek) a nyomvonal ismeretében, a kivitelezés helye, ideje, jellege függvényében, valamint előzetes mérési tervek szerint a megrendelő határozza meg. A mérési terveket a SolData-HUNGEOD Konzorcium készíti el a Mérnökkel és a Megrendelővel történő előzetes egyeztetés után. A tervek a Mérnök engedélye után érvényesek és betartandóak. A Konzorcium dolga eldönteni, hogy milyen eszközökkel szándékozik végrehajtani a feladatot, amelyeket a főmérnök és a megrendelő jóváhagyása után telepíthet.

A 2006-ban elkezdett budapesti 4-es metró építését különböző bányászati technológiák segítségével végzik, mely munkafolyamatok a felszíni környezetben is éreztetik hatásukat. Az alagútépítés során keletkező süllyedési horpa ugyanis az épületek elmozdulását okozhatja. A hatások vizsgálata rendkívül nagy jelentőséggel bír, ezért van szükség építménymozgás-megfigyelésre, valamint a zajszint- és a rezgés mérésére. Ezen feladatok elvégzése a metró teljes hosszán a SolData S.A.S. nevű francia cég dolga. A manuális méréseket, vagyis a nyomvonal környezetében lévő épületek, építmények mozgásának vizsgálatát a velük konzorciumban álló

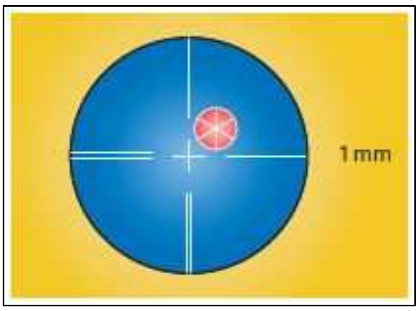
magyar HUNGEOD Kft. végzi. Az állag megóvása a lakóépületek és közművezetékek biztonságán túl rendkívül nagy jelentőséggel bír számos XX. század eleji épületegyüttes és a világörökség részét képező Budapesti Műszaki és Gazdaságtudományi Egyetem CH és Központi épületének esetében is. A felszíni mozgások műszeres mérése által folyamatosan nyomon követhetőek a változások, és korai beavatkozással megelőzhetőek az esetleges károk. A geometriai változásokat az objektum rendszeres megfigyelésével tudjuk meghatározni.

A budapesti 4-es számú metró I. szakaszán lévő épületmozgás-megfigyelő monitoring rendszer egyik eleme az automata geodéziai mérőberendezés, melynek telepítését a SolData-HUNGEOD Konzorcium végezte. Kétféle altípusát alkalmazzák, a CYCLOPS-ot és a CENTAUR-t, melyek összeköthetőek, de egymást kiegészítve is működhetnek. Az automata rendszerrel mért értékeket a Geoscope Web szerver azonnal megkapja, tehát az adatok valós időben elérhetőek. Az automata és manuális módszerek együttes alkalmazása praktikus, gazdaságos és műszakilag is a legjobb. Automata módszerrel főként az épületek homlokzata mérhető, bár pl. a Fővám téri vásárcsarnok esetén a nagy belső tér és a felszín rendszeres forgalma, kitöltöttsége miatt beltérben is gazdaságos a telepítése. Beltérben az épület tulajdonosa, üzemeltetője odafigyel, hogy minden prizma látható legyen a műszer számára. Azokon a helyeken, amelyeket nehéz megközelíteni (pl. udvar, pince, stb.), praktikusabb manuális módszerrel végezni a méréseket. A manuális rendszer nem on-line, azaz a manuális mérések eredményeit nem lehet közvetlenül az interneten publikálni.

. A legtöbb adat (az észlelt adatok 90%-a) az automata és a manuális geodéziai mérésekből származik. Ezen két mérési elem tökéletesen kiegészíti egymást, valamint növeli az eredmények megbízhatóságát, ami még több mérőberendezés hozzáadásával tovább javítható (alkalmaznak még dőlésmérő berendezéseket, rezgőhúros nyúlásmérő szenzorokat, talajmechanikai mozgásokat figyelő műszereket, illetve zaj-és rezgésmérőket). A monitoring rendszernek része továbbá a mérési korrekciók számításához szükséges két (pesti és budai) meteorológiai állomás is.

2.2.1. A CYCLOPS rendszer

A CYCLOPS automata mozgáskövető geodéziai mérőrendszer a SolData és az IGN (Institut Géographique Nationale – a magyar FÖMI-nek megfelelő Francia Földmérési Intézet) közös fejlesztése. A CYCLOPS lényegében egy (vagy több) számítógép által vezérelt motorizált mérőállomásból álló, prizmák elmozdulásait követő rendszer. Egy-egy ciklus során a teodolitok két távcsőállásban végigmérik a hozzájuk rendelt célpontokat, majd elölről kezdik a mérést. A CYCLOPS éjjel-nappal világítás nélkül mér. A mérés automatizálását különböző technológiák, mint pl. az ATR (Automatic Target Recognition) teszik lehetővé.



2.1. ábra: ATR (*forrás*: [LEIC])

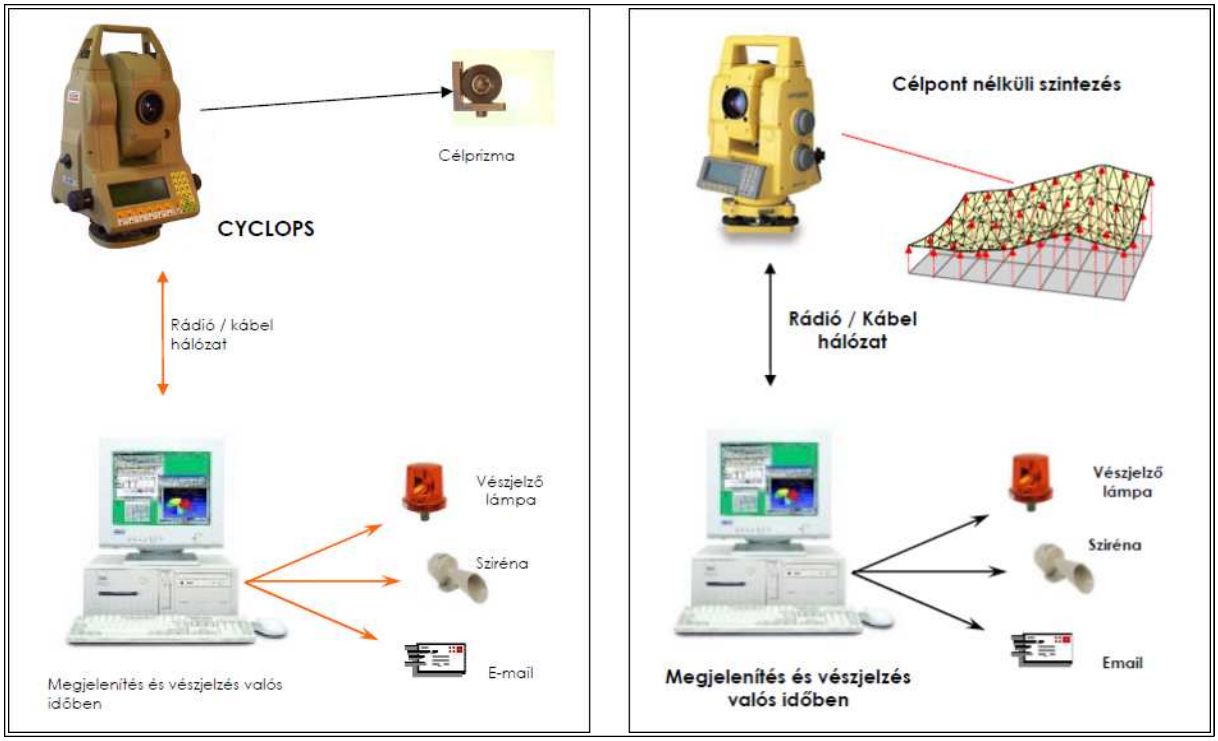
Használatának előnye a rövid mérési idő, precizitás, könnyű kezelhetőség és a munka hatékonyabbá tétele. Az ATR alkalmazása rutinszerű (monitoring, két távcsőállásban végzett) munkák végzésénél igen célszerű. Mivel a célpontfelismerő funkciót egybeépítették a távolságmérő rendszerrel, a felhasználónak csak a célpont közelébe kell irányoznia a távcsövet, a prizma közepét a műszer magától megtalálja. Az objektíven át vetített infravörös fénynyaláb visszaverődésének kiértékelése után, a távcső automatikus középre irányzását végzi a mérőállomás [LEIC]. A mérés ideje körülbelül 3 másodperc prizmánként (amennyiben elsőre sikerül megmérni). A prizmák abszolút helyzetét X, Y és Z irányokban adja meg, melyek közül az X és Y koordináták a

vizsgálati pontok vízszintes-, a Z pedig a függőleges változásait jellemzik. Az egy teodolitra eső vizsgálati pontok megszokott száma 50 prizma, de a 4-es metró állomásainak környezetében ennél jóval több prizmát irányoznak, ezért a prizmákról 30 percenként kapunk új adatokat. A CYCLOPS-ot fedél és védőrács óvja a sérülésektől. A CYCLOPS elviekben abszolút értékeket szolgáltat (de a feladat jellegétől is függ, pl. esetünkben relatív). A referencia-prizmák mért helyzete mellett a – szintén a konzorcium által – budai vagy pesti oldalon telepített automatikus meteorológiai állomások aktuális adatait is felhasználja. Ezáltal nem csak az esetleges saját mozgás torzító hatását, hanem a meteorológiai paraméterek változását is figyelembe veszi a mérési pontosság fokozása érdekében. A CYCLOPS-okat a legkülönfélébb szerkezetek alakváltozásainak ellenőrzésére használják. Legelterjedtebb alkalmazásai között szerepel az alábbiak megfigyelése: épületsüllyedések, kompenzáló injektálások, műtárgyak, gátak, hidak alakváltozásai, alagút-konvergencia. Az információk valós időben megjeleníthetők a GEOSCOPE Web-en.

A 4-es metró CYCLOPS rendszere háromféle mérőállomást alkalmaz: Leica TCA 1800, Leica TCA 2003, valamint Leica TCRA 1201 típusokat. A különbség a hatótávolságban figyelhető meg. A (leggyakrabban alkalmazott) TCA 1800 körülbelül 100 méterig biztosítja a megkívánt pontosságot, míg a TCA 2003-mal 120 méter távolságban lévő prizmákra is kellő pontossággal végezhetünk mérést (ritkábban előforduló típus). Egy ilyen típusú műszer található pl. a Baross téren. A CYCLOPS rendszer szokásos pontossága 60 m-en ±0,5 mm, de függ a referenciapontok stabilitásától is.

2.2.2. A CENTAUR rendszer

A CENTAUR névre hallgató rendszert szintén alkalmazzák a 4-es metró beruházás keretein belül. A CYCLOPS-hoz hasonlóan ez is a SolData és az IGN közös fejlesztése. A CENTAUR egy motorizált, számítógép által vezérelt mérőállomásból áll, amelynek lényege, hogy állandósított jelek nélküli felületek automatikus szintezését végzi, a mért süllyedések és emelkedések pedig valós időben elérhetőek. Gyakori alkalmazási területek: autópályák, utak burkolatai, repülőtéri fel- és leszálló pályák, hidak, instabil lejtők. A műszer a tájékozását referenciapontokra való méréssel végzi. Az abszolút függőleges mozgás szempontjából vizsgált pontok egy virtuális vízszintes hálón vannak elhelyezve. A műszer a méréseket a vízszintestől számított 10-60°-os tartományban képes végrehajtani. A mérési hatótávolsága általában 50 m, szokásos pontossága 40 m-en ±0,5 mm (ez az érték aszfalton a 2 mm-t is meghaladhatja). A mérési pontosság a távolságon kívül a felület minőségétől (beton, aszfalt, homok, talaj, stb.) és a referenciapontok stabilitásától is függ. A külső körülmények, mint pl. járműforgalom gyakran okoznak jelvesztést. A hőmérséklet-változás okozta korrekciókat valós időben számítja. A CYCLOPS-hoz hasonlóan a CENTAUR is éjjel-nappal, a felület megvilágítása nélkül működik, valamint a műszert fedél és védőrács óvja a sérülésektől.

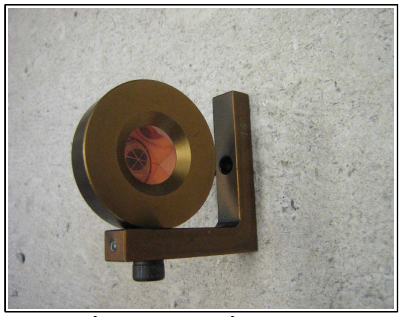


2.2. ábra: CYCLOPS és CENTAUR (*forrás*: [SOLD])

2.2.3. Az automata rendszer elemei

Referencia-prizmák: a műszer tájékozásához szükséges pontokat jelölő prizmák, a mérési eredményeket hozzájuk viszonyítva kapjuk meg. Nagyon fontos, hogy mozdulatlannak tekinthessük őket, ezért elhelyezésük a süllyedési zónán kívül történik. A valóságban előfordul, hogy ezek a pontok is mozognak, pl. a Szent Gellért téren olyan épületekre telepítettek referenciapontokat, melyekre hatást gyakorol a Duna, a talajvizek és maga az építkezés is. Az ilyen esetek miatt időnként a referencia-prizmákat is ellenőrzik. Az olyan prizma, amelyen elmozdulást mérnek nem használható a továbbiakban referenciapontként, vagy folyamatosan újra meg kell határozni a helyzetét és átállítani az értékét az érintett műszerekben. Amennyiben kivonják a pontot és nincsen már elegendő fix referencia-prizma a tájékozáshoz, újakat kell telepíteni.

Vizsgálati (cél-) prizmák: a mérés szempontjából fontos, hogy olyan helyekre kerüljenek fel (lehetőleg egyenletesen elosztva), ahol a műszer akadálymentesen meg tudja őket irányozni. A szakértők térkép és az épületek homlokzatát ábrázoló digitális fényképek segítségével, valamint a környezeti adottságok figyelembevételével állapítják meg a prizmák helyét és sűrűségét. A sűrűség függ attól, hogy az adott épület mennyire van közel a mélyépítési munkaterülethez (egy közvetlen az alagútépítés felett lévő épület homlokzatára több prizma kerül, míg egy távolabbi, de még a süllyedési zónában lévő épületére kevesebb). A végleges helyükre csavarozással vagy rögzítő kapcsolatokkal helyezik fel őket.



2.3. ábra: Vizsgálati prizma

CENTAUR pontok: a felületpásztázó műszerrel végzett mérésekhez szükséges pontokat az aszfaltra festik fel színtelen festékkel, vagyis a pontok külső szemlélő számára láthatatlanok. Olyan területeken alkalmazzák, ahol nagyobb mértékű süllyedésre lehet számítani, pl. a Műszaki Egyetem CH épülete alatt épített nyugati szellőző és környezete, illetve két állomás között épülő szellőző műtárgy esetében, ahol a műtárgy forgalmasabb út alatt helyezkedik el (Vámház körút).

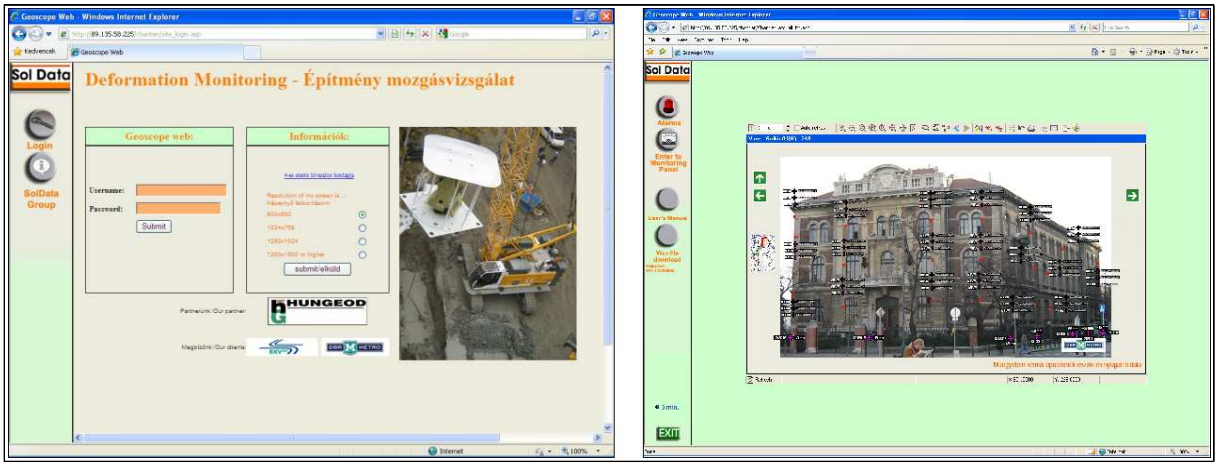
2.4. ábra: Leica TC1800

Mérőállomás (elektronikus tahiméter): a mérőállomások helyének tervezésekor a legfontosabb szempont az, hogy a lehető legnagyobb területet belátható legyen, vagyis hogy minél több prizmát meg lehessen irányozni. A metróállomások környezetében általában nem egy műszert helyeznek ki, a Szent Gellért téren pl. három található. Miután eldöntötték a műszer pontos helyét, először a műszert tartó szerkezetet (tálcát) fúrják fel a kijelölt helyen, melyen ezután rögzítik műszertalpat és végezetül a ráhelyezik a mérőműszert is. Ha ezzel készen vannak, akkor következik a műszer „betanítása”, vagyis négy fordulóban, két távcsőállásból bemérik az összes referencia-prizmát, vizsgálati prizmát és CENTAUR pontot (ha van ilyen). Ezek után a nullmérési eredményt minimum egy nap méréseinek kiátlagolásával kapják meg. Mindemellett párhuzamosan végezhető a hálózatmérés, mely során valamennyi mérőállomás, illetve néhány referencia- és célprizma abszolút koordinátáit határozzák meg. A műszertalpat a műszer esetleges meghibásodása esetén sem veszik le, csak az alhidádét cserélik ki, ha szükséges.

Minden prizma, CENTAUR pont és mérőállomás egyedi névvel rendelkezik. A nevek „beszédesek”, ugyanis rengeteg kódolt információt tartalmaznak az adott objektumról. Ezt egy példán keresztül a legegyszerűbb bemutatni, vagyis egy Szent Gellért téren elhelyezett referencia-prizma elnevezése például: **RGL31.01.02.**, amelyben az **R** jelenti a referenciát (angolul: *reference*), a **GL** a Gellért tér rövidítése, a **31**-es szám az épületre utal, a **01** és a **02** pedig arra, hogy az épület homlokzatán raszter-szerűen elhelyezett pontok közül az adott prizma hányadik sorban és hányadik oszlopban található.

2.2.4. Mozgásvizsgálati eredmények megjelenítése

A már említett automata mérőrendszerek a 4-es metró teljes szakaszáról kinyert, még feldolgozatlan adatokat rádió kapcsolat segítségével továbbítják a SolData-HUNGEOD Konzorcium Móricz Zsigmond körtéren lévő irodájában található központi szerver adatbázisába. Ezáltal megkezdődhet a kiértékelés. Az automata geodéziai rendszertől érkező nyers adatokon első lépésben egy összetett kiegyenlítő szoftver számításokat végez, mivel valós eredményeket csak a különböző külső hatások figyelembevételével (például hőmérséklet) kaphatunk. A már kiegyenlített értékek egy egységes, szintén a SolData által kifejlesztett felületen, a Geoscope Weben jelennek meg. A szoftver elérését és az adatokhoz való hozzáférést internetkapcsolattal rendelkező, Windows operációs rendszerrel működő számítógépen lehet biztosítani. A Geoscope Web azon személyek PC-jére kerül fel, akiket a megrendelő és a kivitelező erre megbízott, hozzáférni pedig a SolData-HUNGEOD Konzorcium honlapján lehetséges.



2.5. ábra: A Geoscope Web belépési felülete és a CH épület megfigyelése (*forrás*: SolData)

A felhasználó a belépés után a jogosultságaitól függően kiválaszthatja a számára érdekes területet, melyhez elegendő a város megfelelő területére kattintani. A megbízó és a mérnök számára minden adat elérhető, a kivitelezők azonban csak a saját építési területük adataihoz rendelkeznek hozzáférési jogokkal. Az adott terület betöltődése után kiválaszthatjuk, hogy melyik mérőműszer adatait kívánjuk megtekinteni, de megjeleníthetőek egyes épületek és a rajtuk lévő mérési pontok, illetve a manuális méréshez használt csapok is. Ezen pontokra kattintva előhívhatók a rájuk vonatkozó aktuális adatok. A változások időbeli tanulmányozásához grafikon formájában is lekérhetőek az adatok, ehhez az időintervallum, lépték, stb. paraméterek beállítása szükséges. [MTMM]

Mivel a felhasználók többsége vizuális típus, a mozgásokról kapott numerikus adatokat célszerű grafikusan is megjeleníteni, amit többféle módon tehetünk meg:

- izovonalas mozgástérkép,
- mozgásdiagram,
- terepmetszet,
- mozgások ábrázolása két dimenzióban mozgásvektorokkal,
- mozgások axonometrikus ábrázolása.

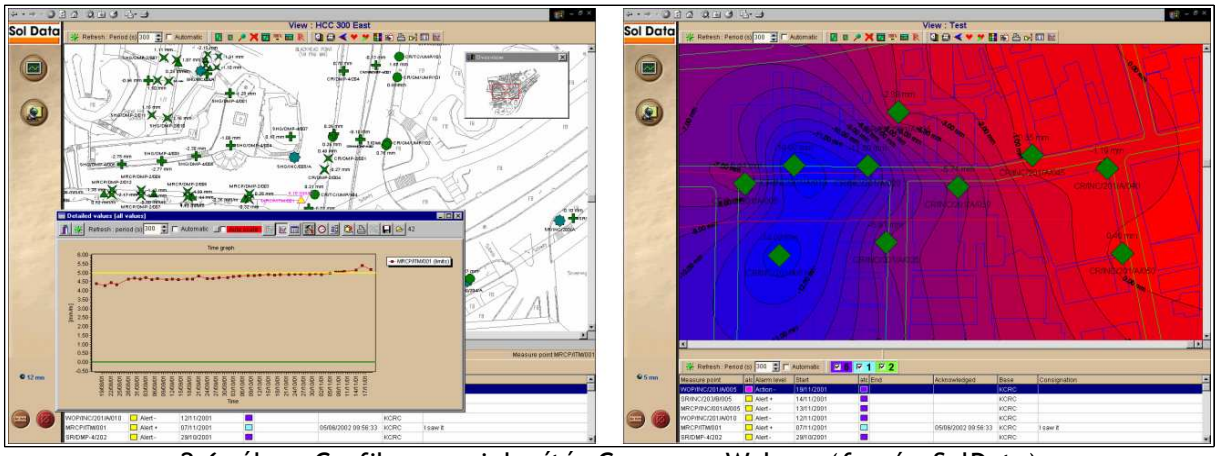
A fentiek közül a Geoscope Web a megbízó kérésére a 4-es metró projekt keretében az első kettőt alkalmazza. Az adatok valós idejű térinformatikai megjelenítésén túl a Geoscope Web legfontosabb szolgáltatásai közé tartozik a riasztás, ami akkor aktiválódik, ha egy mért érték átlépi azt a küszöböt, amelyet a szakértők korábban az adott pontra meghatároztak. Riasztáskor az erre kijelölt személyek e-mailben és/vagy SMS üzenetben kapnak értesítést az érintett mérési pontról és az általa elért határértékről (természetesen valós időben). Három riasztási szintet különböztetnek meg:

- Elsőfokú riasztás (szakértői figyelmeztetés): az adott szakértői kör figyelmét hívja fel.
- Másodfokú riasztás (készültség): a ponton regisztrált változások olyan mértékűek, hogy valószínűleg mesterséges hatás eredményeképpen adódtak, ezért a kivitelezőt biztonsági intézkedések megtervezésére szólítja fel.
- Harmadfokú riasztás (vészjelzés): az automatikus első- és másodfokú riasztásokat követően kerülhet rá sor, sürgős szakértői beavatkozás igénylő helyzet, amely akár evakuálást is jelenthet.

Az összes mérési adat két Geoscope Web felületen érhető el:

• hidrogeológiai adatok megjelenítése (a SolData-Mecsekérc-Mélyépterv Konzorciumban szolgáltatása),

• deformációs adatok megjelenítése (a SolData-HUNGEOD Konzorcium szolgáltatása). [forrás: SolData]



2.6. ábra: Grafikus megjelenítés Geoscope Web-en (*forrás*: SolData)

3. Informatikai technológiák

3.1. Digitális térképek

A digitális térképek előnye a hagyományosokkal szemben vitathatatlan. A papíralapú térképeken rendkívül nehézkes a változások vezetése, a papír mérete korlátozza a látható (tárolható) információk mennyiségét, ráadásul az állaga az idő előrehaladtával romlik. A digitális térképen az elemeket nem csak láthatjuk, hanem interaktívan kezelhetjük is őket. A strukturált adatrendszer (vagyis egy fájlon belül több elkülönített réteg alkalmazása) lehetővé teszi, hogy módosítás vagy törlés esetén kizárólag az érintett objektumok változzanak meg. Továbbá a rétegeknek köszönhetően a felhasználó döntheti el, hogy éppen mit szeretne látni a tárolt információk közül, melynek mennyiségét csak a rendelkezésre álló tárhely mérete korlátozza. Digitális adattárolásnál ráadásul szelvényezésre sincsen szükség. A térképi objektumokhoz a megjelenítéshez szükséges adatokon kívül úgynevezett nem térképi adatok (attribútumok) is hozzárendelhetőek, melyekkel megkönnyíthető az elemzés. Ezen lehetőségek kihasználásával a térkép többféle szakterület igényeihez is igazítható, hiszen a virtuális térképek készítőinek elsődleges célja inkább az, hogy a felhasználó könnyedén megtalálhassa a számára hasznos információkat, mintsem az, hogy minél több statikus adatot láthatóvá tegyenek. A digitális technológiának köszönhetően automatizálhatóak az egyes folyamatok (generalizálás, elemzés). [MITC]

Attól függően, hogy a számítógéppel kezelhető térkép rajzi elemei között van-e kapcsolatleírás vagy nincs, beszélhetünk digitális másolattérképekről vagy digitális térképi adatbázisról. Az ábrázolás szempontjából megkülönböztetünk raszteres vagy vektoros formátumot. A vektoros térkép előnye a raszteressel szemben, hogy kisebb méretű a fájl és tetszőleges felbontásban megjeleníthető, hátránya viszont, hogy sokkal összetettebb adatstruktúrája van.

Raszteres adatokat mátrixos, ill. hálós struktúrában tárolunk, ahol a sorok és oszlopok metszéspontjában található pixelek (cellák) értékkel rendelkeznek. Rasztertérképet fotogrammetriai felvétellel vagy távérzékeléssel nyerhetünk, illetve már meglévő hagyományos térkép digitalizálásával (szkenneléssel).

Vektoros állomány esetén koordinátákkal definiáljuk a pontokat és a pontokat összekötő vonalakat, poligonokat. Vektorelemekből álló térképet geodéziai méréssel nyert adatokból (pl. GPS mérési eredmények), digitalizálással és vektorizálással, vagy már meglévő adatbázisok átvételével készíthetünk. Az egyes elemek (vonalak, pontok) egyedi azonosítóval rendelkeznek és a környezetükhöz való kapcsolatuk ismert.

Amennyiben a digitális térkép a raszteres és vektoros adatokat kombinálja, akkor hibrid rendszerről beszélünk. A geometriai adatok kapcsolatát topológiai modellel írhatjuk le. (forrás: [AG05][AT10])

3.1.1. Geoinformációs rendszerek

A számítástechnika fejlődése, a számítógépek tökéletesítése lehetővé tette olyan információs rendszerek létrehozását, melyek a térképi információszerzés, adattárolás, elemzés és megjelenítés feladatainak ellátására egyaránt alkalmasak. Ezen rendszereket az angol elnevezésből eredően leggyakrabban GIS-nek (Geographical Information System) hívják, de magyar szinonimái is

léteznek, például: földrajzi információs rendszer, geoinformációs rendszer, térinformációs rendszer, térképalapú információs rendszer. Segítségével már meglévő adatokból tudunk új adatokat levezetni, vagyis elemezni. Az elemzés során egy komplex problémát tagolunk részekre (analízis), majd a kapott részeredményeket egyesítjük (szintézis). A térinformatikai rendszerek fejlesztői az informatikusok és a térinformatikusok, szakmai jellegű felhasználásának célja lehet műszaki tervezés, nemzetbiztonsági intézkedés, piackutatás, társadalmi felmérés, stb.

A rendszer elemei a valós világot modellező entitások digitális változatai (objektumok). A térinformatika lényege, hogy az objektumokhoz nem csak geometriai-, hanem szakadat (attribútum) is tartozik, valamint definiálják az egymás közötti kapcsolatukat is. Az adatokat a logikai modellben meghatározott elvek szerint tárolja egy fizikai modell (adatbázis). A geometriai tartalom szerint megkülönböztetünk pont (0D), vonal (1D), felület (2D) és test (3D) típusú objektumokat, továbbá a mozgásvizsgálat szempontjából lényeges, hogy a geometriai adatokhoz időpontokat is hozzá lehessen rendelni, (4D). A testek modellezése történhet 3D-ben (felületekkel, testekkel), 2+1D-ben (ha szintvonalakkal jellemezzük a magasságot) és 2,5D-ben (ha a pontok attribútumaként szerepelnek a magassági adatok). Az alakzatok megadása vektor, ill. raszter alapú rendszerben lehetséges. Az attribútumok típusa az adott feladat jellegétől függ, hagyományosan táblázatos formában tárolják.

A helymeghatározást általában koordináták alapján végezzük, de diszkrét jellemzők alkalmazása is célszerű (pl. postai irányítószám, intézménynév, cím, stb.). A koordináták vonatkozási rendszere és a vetületi rendszer előre meghatározott.

Az elemzéshez számtalan művelet végrehajtható, melyhez segítségül hívhatóak a kereső nyelvek (pl. SQL). Néhány alapvető elemzést segítő funkció:

- mérések, számlálás (pontok száma, távolság, szög, kerület, terület),
- metszések (pont-poligon, vonal-poligon, poligon-poligon),
- hálózatelemzés (legkedvezőbb útvonal),
- statisztikai (eloszlás, sűrűség, korreláció, többváltozós analízis),
- magassági modellezés (lejtés, vízgyűjtő terület, stb.).

(forrás: [DET1] [ASJ1] [AT10])

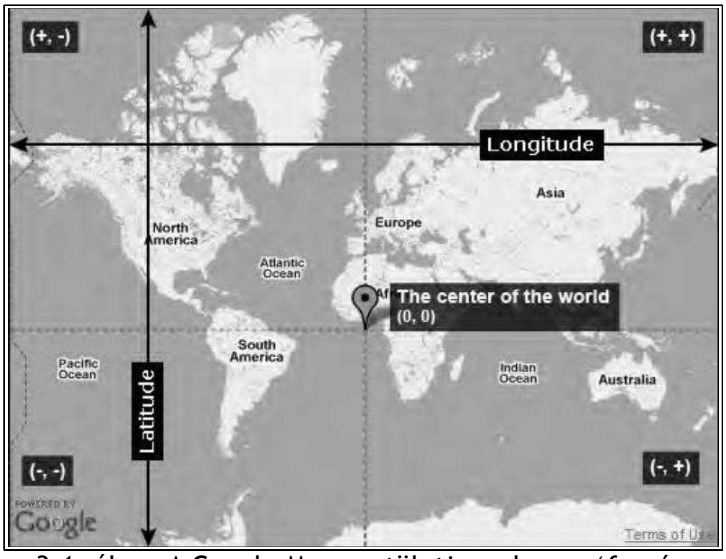
3.1.2. Internetes térképi technológiák

A térképek internetes közzétételéhez webszerver szükséges, melyek által kínált tartalmat a kliens oldalán egy böngésző segítségével jeleníthetjük meg. A technikai megoldások szempontjából megkülönböztetjük a szerveroldali, a kliensoldali és a kombinált alkalmazásokat. Amennyiben a munka nagyobbik része a kliens számítógépén hajtódik végre, vastag kliensről beszélünk. A vastag kliens esetén ritkábban van szükség a szerverrel való kommunikációra (adatok, kliensoldalon futtatott programkomponensek letöltése). A munkavégzés hatékonysága elsősorban a kliens gépének teljesítményétől függ. A szerverrel végrehajtott feladatok csökkentésével a leterheltség is csökken, ezáltal növelhető a kiszolgálható felhasználók száma. A vastag kliens előnye, hogy a munkafolyamatok nem igényelnek nagy kapacitású szervert, tehát a szerver szempontjából költségkímélő megoldás. Ezzel szemben a kliensoldali gépek drágábbak, mivel az ő oldalán általában nagyobb teljesítményre van szükség, illetve minden egyes kliensre szoftver licenc szükséges.

Nagyobb méretű térképes adatbázisok esetén azonban jobb teljesítményt lehet elérni az ún. vékony klienssel, amely a feladatainak elvégzéséhez más gépek segítségét veszi igénybe. Sem a szoftver, sem az adatbázis tartalma nem kerül át a kliens oldalára, csak a szerver által előállított „fényképek” a térinformatikai adatbázisról. A térképszerver használatához tehát nincs feltétlenül szükség a kliensoldalon telepített bővítményekre (plug-in), az aktuális művelet eredményét a szerver képfájl (jpg, png) formájában küldi el a kliens böngészőjének. A néhány 10-100 KByte méretűre tömörített képek küldése kisebb átviteli sebesség mellett sem okoz gondot. Vékony klienssel működő alkalmazás esetén a kliensoldali számítógép sebessége nincs nagy befolyással a munka teljesítményére, azonban a szerver könnyen túlterheltté válhat, hiszen a munka minden egyes részfeladatánál szükség van rá. A megfelelően méretezett szerver költsége egy nagy hálózat esetén igen jelentős összeg. [SZ01]

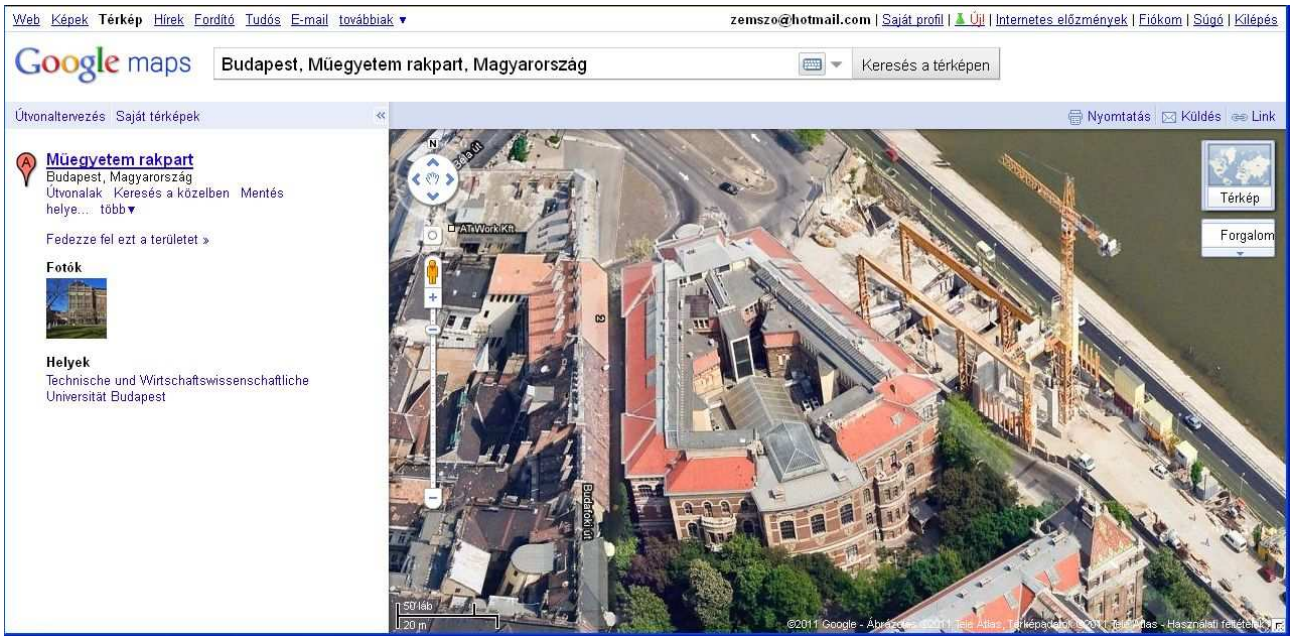
Szerver-kliens struktúrájú programrendszerek készítésekor mindig előre meg kell tervezni, hogy az egyes feladatrészek hogyan oszlanak meg a szerver és a kliens között. A tervezésnek nagy jelentősége van, hiszen ekkor hozunk döntést a rendszer költségéről, teljesítményéről, biztonságáról és a későbbi módosításokhoz való alkalmazkodási képességéről is. [*forrás*: en.wikipedia.org/wiki/Fat_client]

Manapság az internet szerves részét képezik a térképes szolgáltatások, melyeket nap mint nap használunk címek keresésére, útvonaltervezésre, stb. A helyhez köthető információk nagy előnye a megjeleníthetőség, a vizuális megoldások pedig vonzóak a felhasználók számára. Az interneten számos cég kínál bárki számára elérhető térképes szolgáltatásokat, pl. Yahoo! Maps, Bing Maps, de a legnépszerűbb termék a Google Maps. A 2005 februárja óta működő szolgáltatás nagy előnye az oldal gyors betöltődése és a számos felhasználóbarát tulajdonsága.



3.1. ábra: A Google Maps vetületi rendszere (*forrás*: [SVEN])

A Google Maps által alkalmazott koordináta-rendszer alapja a WGS 84 ellipszoid, amely megegyezik a GPS rendszerben használttal. A koordináták megadásakor gyakorlatilag szélességet és hosszúságot definiálunk. A szélesség az Egyenlítő mentén nulla és északi irányban pozitív. A hosszúság kezdőértéke a Greenwich-i meridián és kelet felé pozitív.



3.2. ábra: a CH épület ferde műholdfelvételen (*forrás*: Google Maps)

A Google Maps és a Google Earth műholdfelvételeinek nagy részét a Landsat 7 készítette. Ezzel a műholddal 30 m-es felbontás érhető el. Budapest területén a GeoEye-1 műhold 50 cm-es felbontású képeit láthatjuk. Nagy felbontású képek készültek az Egyesült Államok, Kanada és az Egyesült Királyság lakott részeiről, melyek nem műholdképek, hanem légifényképekből előállított ortofotók. A Google felvételein nem valós időben látjuk a helyszíneket, a képek általában több hónappal (esetleg évvel) korábbi állapotot mutatnak.

A keresés szempontjából a Google Maps igen rugalmas, értelmezni tud pl. földrajzi szélesség-hosszúság koordinátákat, teljes címeket, rövidítéseket, postai irányítószámokat, intézményneveket, és egyszerű lekérdezéseket is végrehajthatunk vele (két cím közti útvonal tervezése, közeli szolgáltatások keresése, stb.). [GIBS]

Az internet elterjedésével megjelent a WebGIS fogalma, vagyis olyan térinformatikai alkalmazások, melyeket az interneten keresztül érhetünk el. Nagy előnyük, hogy a kezelőfelületet nem kell minden egyes felhasználó gépén telepíteni. Ez idő- és tárhely-megtakarítást jelent, ezen kívül pedig további lehetőségeket is kínál a felhasználóknak:

- különböző GIS-rendszerek összekapcsolása és adatcseréje,
- újfajta tartalmak (például űrfelvételek) elérése,
- megjelentek az ún. digitális földgömbök (pl. Google Earth, Microsoft Bing Maps),
- a digitális földgömbök kiegészültek a városokról készített 3D modellekkel.

Az internetnek köszönhetően a speciális szakképzettséggel nem rendelkező, hétköznapi emberek is hozzáférhetnek az adatokhoz és csatlakozhatnak a térinformatikai rendszerek felhasználói köréhez (pl. útvonalkeresés). A helymeghatározásra használt mobil eszközök elterjedésével a WebGIS népszerűsége még tovább növekszik.

Az internetes térinformatikai eszközök alkalmazásához leíró nyelvek ismerete szükséges. Ilyen nyelv pl. az SGML (*Standard Generalized Markup Language*), amely nemzetközileg elismert szabvány. A legismertebb belőle származtatott leíró nyelv a HTML és az XML. A HTML

(*Hypertext Markup Language*) weboldalak készítéséhez kifejlesztett nyelv; értelmezését a böngésző hajtja végre. A HTML általános jellege miatt az egyes szakterületek sajátosságaira nem terjed ki. Az XML (*Extensible Markup Language*) szintén általános jellegű leírónyelv, amelyből speciális célú leíró nyelvek fejlődtek ki, pl. a GML (*Geography ML*), amely földrajzi objektumok és tulajdonságaik leírására kifejlesztett nyelv, a GeoWeb infrastruktúra leíró nyelve. A Google térkép alapú szolgáltatásainak XML alapú nyelve a KML (*Keyhole ML*), elsődleges célja a földrajzi jelenségek megjelenítése és kiegészítése. 3D városmodellek reprezentációjához a CityGML szolgáltatja az informatikai modellt, amelynek különlegessége, hogy a megjelenítés mellett elemzésre is alkalmas.

A Digitális Föld projektet Al Gore amerikai alelnök 1998-as kezdeményezésére hozták létre, amely gyakorlatilag a Föld egészére kiterjedő térbeli adatok 3D-s internetes megjelenítését szolgálja. A legismertebb virtuális földgömbök (melyeken kívül még számos, kevésbé népszerű is létezik):

- NASA World Wind (2004): topográfiai térképeken, űr- és légifelvételeken alapul, szabad szoftver ingyenes adatokkal.
- Google Earth (2005): légi- és űrfelvételeken alapszik, tartalmazza az úthálózatot. Szabadon letölthető, de csak az alapadatok ingyenesek.
- Bing Maps (korábban: Windows Live Search Maps, 2006): úttérkép az egész Földről, helyenként űrfelvételekkel. Speciális funkciók: madártávlat, 3D városmodellek. Szabad szoftver ingyenes adatokkal.

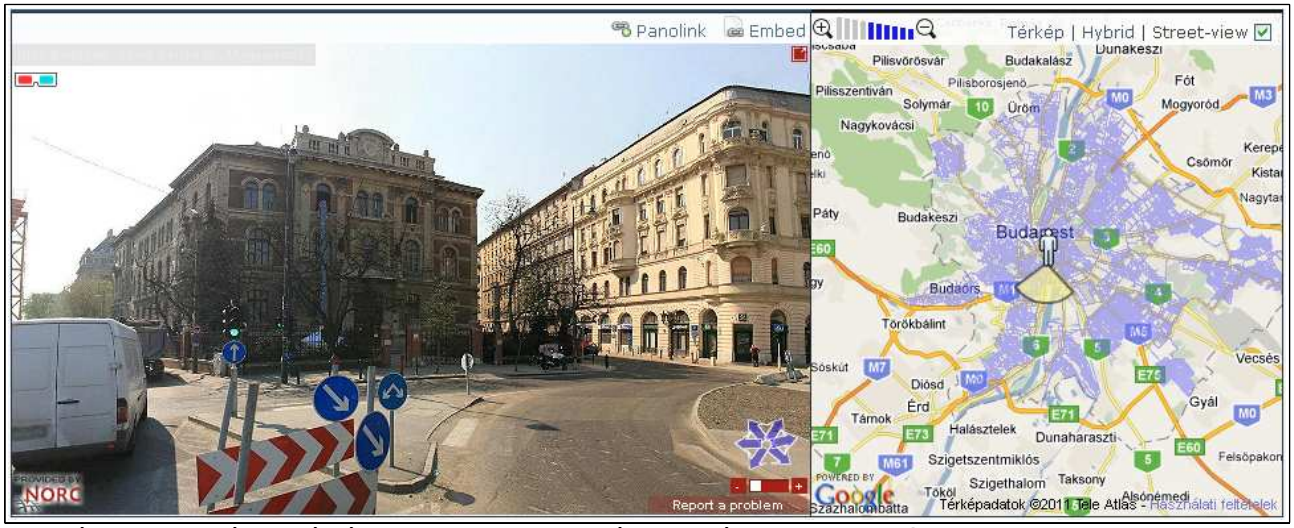
A virtuális földgömb szoftverek 3D-ben jelenítik meg a Földet (vagy egy másik égitestet) és a felhasználó számára lehetővé teszik a virtuális térben való szabad mozgást. A mozgás illúzióját azzal érik el, hogy a megfigyelési szög, a helyzet és a vizsgált tartalom változtatható. A térképhez hasonlóan a terület egyaránt megjeleníthető műholdképi, térképi vagy hibrid formában, továbbá kicsinyíthető, nagyítható, forgatható és dönthető, saját fényképpel kiegészíthető.

A 3D városmodellek a virtuális földgömbök speciális kiegészítői, a jelentősebb virtuális földgömbök mind rendelkeznek ilyen modellekkel. Ugyan a virtuális földgömbök elemzésre csak korlátozottan alkalmasak – mivel elsődleges funkciójuk a megjelenés –, a 3D városmodellek segítségével már elemzéseket is végezhetünk. Ehhez az szükséges, hogy az objektumok ne csak geometriai, hanem tartalmi (szemantikai) adatokat is tartalmazzanak. A térbeli modellezés fontos jellemzője a részletek szintje (*Level of Detail, LOD*). [DET2]

3.3. ábra: LOD 0:regionális modell, LOD 1: blokk modell, LOD 2: részletes tetőszerkezet, LOD 3: részletes építészeti modell (*forrás*: Google Earth)

A Google néhány további szolgáltatása:

Google Street View: Az adatgyűjtést egy számítógéppel, 360°-os kamerarendszerrel, GPS-szel és távolság meghatározására szolgáló lézer berendezéssel felszerelt gépkocsival (parkokban kerékpárral, beltérben kézikocsival) végzik. A képek pozicionálásához GPS mérési eredményeket, illetve – ott, ahol az épületek leárnyékolják a műholdakat – inerciális navigációs rendszerekkel nyert adatokat használnak fel. A kamerák által készített képek átfedésben vannak, melyeket a 360°-os látvány eléréséhez „összefűznek”. Ekkor a képek torzítottnak tűnnek, mivel az összeillesztéshez síkba transzformálják őket, ezért a végleges eredményhez vissza kell vetíteni egy gömbre. Az alkalmazásban való mozgáskor a soron következő megjelenített panoráma kiválasztása a lézerrel mért távolsági adatok segítségével lehetséges.



3.4. ábra: NORC által készített budapesti utcakép a CH épülettel (*forrás*: www.norc.hu/street-view)

A Google Street View alkalmazással az Egyesült Államokban rengeteg helyen, továbbá számos európai és a világ más táján lévő nagyvárosokban „kalandozhatunk”. Budapest fotózását a hatóságok leállították, de a romániai NORC rendszer üzemeltetői elkészítették a panorámaképeket a főváros utcáiról. Az általuk készített (Google Street View-hez hasonló) on-line alkalmazással virtuálisan körbejárható Budapest.

Google SketchUp: egyszerűen kezelhető 3D-s szerkesztőprogram, elsősorban épületek modellezésére tervezték, de bármilyen egyéb objektum vagy enteriőr modellezésére is alkalmas. Épület-modellezési lehetőségek: a modellezni kívánt épület pozíciója és méretei Google Earth-ből átvehetőek, a kész alakzat felületeire fénykép illeszthető, mely lehet saját fotó, de lehet a Google Maps utcaképeiből átvett képrészlet is. Az elkészült modellünket a Google Earth-be is beilleszthetjük. Maga a SketchUp program ingyenes, csak a professzionális változatért kell fizetni (SketchUp Pro). [*forrás*: Google]

3.5. ábra: A CH épület modellje Google SketchUp-ban (*forrás*: Google)

3.2. Téradatbázisok

Az adatbázis célja az adatok rendezett tárolása valamilyen előre meghatározott, leggyakrabban előforduló igényeket kielégítő szisztéma szerint. A számítógépes adatbázis az adatbázis-kezelő program része, amely az adattárolás mellett lehetővé teszi a tárolt adatok visszakeresését, illetve kezelését is. Az adatbázis-kezelők és a műveletek végzésére szolgáló programnyelvek abból a célból jöttek létre, hogy az adatokkal való munkavégzést megkönnyítsék és egységessé tegyék. A korszerű adatbázis-kezelők az adatokon kívül a rájuk vonatkozó konzisztencia-szabályokat és az egyes felhasználók jogosultságait is tárolják. Az adatok közti kapcsolatok tárolása szempontjából többféle logikai modell létezik, azonban a ma használt adatbázis-kezelők leginkább az ún. relációs modellre épülnek. A relációs adatmodell kidolgozója E. F. Codd (1970), aki egyúttal számos feltételt fogalmazott meg annak érdekében, hogy egy adatbázist valóban relációsnak tekinthessünk.

A reláció gyakorlatilag egy táblázat és a benne lévő sorokban tárolt adatok összessége. A táblázat (más néven *entitás*) felépítése két dimenziós, soraiban a logikailag összetartozó egyedek találhatóak, az oszlopokban pedig a hozzájuk tartozó különböző mennyiségek (*attribútumok*) jelennek meg. A sorok (*rekordok*) sorrendje közömbös, azonban két teljesen azonos tartalommal bíró rekord nem létezhet egy táblán belül, továbbá az egyes attribútumoknak közös adattípussal kell rendelkezniük (pl. szöveges, dátum, logikai, stb.). Mivel nem lehet két teljesen azonos sor, ezért minden relációban létezik az attribútumoknak egy olyan csoportja, melynek elemei egyértelműen azonosítanak egy sort (reláció *kulcs*). A sorok és oszlopok metszéspontjában található *mezők* kizárólag egy értéket tartalmazhatnak. A *NULL* érték annak a jelölésére szolgál, hogy a cella értékét nem ismerjük vagy nem határoztuk meg (alkalmazása a kulcs oszlopokban nem megengedett). Az adatkezelést meggyorsíthatjuk, ha a táblák oszlopaihoz ún. *indexeket* rendelünk.

A relációs modell elvén működő relációs adatbázis-kezelő rendszer (*Relational Database Management* System, RDBMS) az adatokat egymással kapcsolatban álló táblákba rendezve tárolja. Ezen RDBMS-ek az elsődleges működtetői az információs rendszereknek világszerte, különösképpen az Internet/Intranet alapú alkalmazásoknak és az osztott szerver/kliens

számítógéprendszereknek. A tárolt adatok elérhetőségét szabályozzák, azaz az egyes adattáblákhoz csak bizonyos felhasználók férhetnek hozzá. Független adatokat tartalmazó táblák összekapcsolása abban az esetben lehetséges, ha mindkettő rendelkezik azonos értékeket tartalmazó oszlopokkal. Az adatbázis szerkezetét leíró adatokat *metaadatoknak* nevezzük.

A legnépszerűbb relációs adatbázis-kezelő szoftverek: Microsoft SQL Server (kereskedelmi), MySQL (nyílt forráskódú), Oracle (kereskedelmi) és PostgreSQL (nyílt forráskódú).

A relációs adatbázis-kezelők lekérdezési nyelve az SQL (*Structured Query Language*), amelynek története az 1970-es években kezdődött az IBM cég köreiben. A kutatás egyike volt azon törekvéseknek, melyek Codd elméletein alapuló programnyelv kidolgozását tűzték ki célul. Az évtizedes fejlesztések eredményeképp egy szabványosított nyelv alakult ki, melyet számos relációs adatbázis-kezelő program esetében alkalmazhatunk, ráadásul az operációs rendszer környezettől függetlenül (bár a későbbi fejlesztői bővítések miatt többféle dialektusa létezik).

Az SQL egyszerűsége abban rejlik, hogy nem tartalmaz algoritmikus szerkezeteket (elágazás, ciklus), vagyis a felhasználónak elég csak azt megadnia, hogy mire van szüksége, a végrehajtás lépéseit nem kell definiálnia. Az ilyen típusú megoldást halmaz-orientáltnak nevezik. Az SQL alapú programok úgynevezett logikai adatcsomagokkal dolgoznak. Adatcsomag alatt táblákat értünk, melyeknek kezelése sokkal hatékonyabbá teszi a munkát, mintha egyedi sorokkal dolgoznánk. Az SQL nyelv utasításai az angol mondat szerkezetéhez hasonlóan épülnek fel. A bevitt parancsok végrehajtásakor egy optimalizáló algoritmus dönti el, hogy milyen elemi lépések sorozatával állítja elő a kívánt eredményt. Az SQL nyelv tartalmazza mindazon parancsokat, melyek ahhoz szükségesek, hogy létrehozzunk, módosítsunk, lekérdezzünk, vagy töröljünk adatokat. Továbbá más programnyelvekhez hasonlóan az SQL is kezel beépített függvényeket, melyek egy konkrét értéket adnak vissza (pontos idő, átlagszámítás, stb.). (*forrás*: [KLIN] [SZ02])

Az SQL nyelvi elemei 4 csoportba különíthetőek el:

- adatdefiníciós (reláció, index, nézeti tábla létrehozása és megszüntetése),
- adatmanipulációs (relációk feltöltése, sorok törlése, attribútumok módosítása),
- lekérdező,
- adatvezérlő.

```
SELECT z,datum FROM meresek WHERE pontnev='prizma23' ORDER BY datum DESC;
```

A fenti példa egy egyszerű lekérdező típusú parancs, amely egy mérési adatokat (pontnév, dátum, z koordináta) tartalmazó táblára vonatkozik. A FROM jelöli ki azt a táblát, melyben a SELECT után felsorolt attribútumok megtalálhatóak, a WHERE után pedig az a feltétel áll, ami szerint a kiválasztott sorokat szűkítjük. A rekordok valamely attribútum szerinti sorba rendezése az ORDER BY utasítással lehetséges. A fenti utasítással tehát dátum szerint csökkenő (*descending*) sorrendben kapjuk meg a 23-as számú prizmára végzett „z” mérési eredményeket.

A fenti utasítás esetében az SQL nyelv önálló felhasználására látható példa, amely űrlap vagy jelentés készítésekor fordul elő. Egyébként úgynevezett beágyazott SQL-ről beszélünk, amikor az SQL utasítást algoritmikus nyelven írt parancsok soraiba illesztjük be. (további részletek a 4. fejezetben).

3.2.1. PostgreSQL és PostGIS

Az relációs adatbázisok speciális, továbbfejlesztett változata az objektum relációs rendszer (*Object Relational Database Management System*, ORDBMS), amelynek lényege az, hogy különböző, az objektum orientált programozási nyelvekben megszokott lehetőségek jelennek meg a relációs adatbázisokban. Az objektum orientált programozás lényege, hogy maga az objektum mindig egy osztályhoz tartozik, amely osztály függvényeket és változókat tartalmaz, de azoknak elérése csak az objektumon keresztül lehetséges. Új osztály létrehozásakor előfordulhat, hogy a szükséges függvénykódok már szerepelnek egy meglévő osztályban, ekkor az új osztályt származtathatjuk egy már meglévőből (öröklődés). A PostgreSQL adatbázis-kezelő rendszer objektum-relációs jellege többek között az öröklés lehetőségének támogatásával valósul meg. Öröklés alatt ez esetben azt értjük, hogy ha két tábla szerkezete nagyon hasonló (oszlopaik között nagy az átfedés), akkor a táblákat származtathatjuk egymásból. Objektum-orientált tulajdonsága továbbá a komplex adatok tárolása, vagyis pl. IP cím, tömbök, grafikus adatok (pont, szakasz, törtvonal, stb.) és lehetőség van saját adattípus létrehozására is.

A PostgreSQL a legkiválóbb nyílt forráskódú adatbázis-kezelő rendszer, mivel képes arra a teljesítményre és funkcionalitásra, mint kereskedelmi társai, ráadásul terabyte méretű adatbázisok kezelésére is alkalmas. A többi relációs adatbázishoz hasonlóan a PostgreSQL esetében is teljesülnek a következő tulajdonságok:

- a szerver-kliens struktúrájú adatbázis-kezelés fenntartása kis költségű,
- az író és olvasó felhasználók használat közben nem akadályozzák egymást,
- több alkalmazási típus számára konfigurálható és kiterjeszthető,
- méretezhetőségének és alkalmazkodási képességének köszönhetően működése hatékony.

A PostgreSQL általános célú szoftver, azonban megfelelő hozzáértéssel saját igényekhez igazítható, illetve kiegészíthető a mások által készített bővítmények kínálatából. Biztonsági szempontból is kiváló szoftver, mely jellemzői alapján inkább az Oracle-höz ill. az SQL Server-hez hasonló, mintsem a MySQL-hez. A PostgreSQL kiegészíthető grafikus kezelőeszközökkel mint pl. pgAdmin3 (mely a PostgreSQL Windows alapú verzióját használva alapértelmezett módban telepítése kerül) és PHPPgAdmin (amely egy böngészőben elérhető webes alkalmazás). Ezen szoftverek kliensoldali alkalmazások, melyek SQL utasításokat tudnak küldeni a PostgreSQL irányába, és megjelenítik az eredményeket a felhasználó számára. Egy Admin kliens számos PostgreSQL szervert képes elérni, és fordítva: egy PostgreSQL szerver több kliens oldaláról is elérhető. A PostgreSQL kezelését lekérdező nyelvvel is lehet végezni (psql), amennyiben professzionálisabb felhasználói ismeretekkel rendelkezünk. [RIGG]

A téradatbázisok megjelenése szorosan összefügg a térinformatikai elemzések fejlődésével. A fénykorukat élő digitális és webes típusú térképek – melyeket az átlagfelhasználó főként helymeghatározásra vesz igénybe – megfelelő statisztikai adatokkal kiegészítve vizsgálati, mintavételezési munkákra is alkalmasak. Az elemzési szempontok könnyedén megjeleníthetők a térképen geometriai alakzatok és különböző jelentéssel bíró színek, minták, stb. segítségével, azonban nagy mennyiségű információ esetén a vizuális adatok alapján végzett vizsgálat nem hatékony. A térbeli objektumokra specializálódott adatbázisok abból a célból jöttek létre, hogy megkönnyítsék ezt a fajta munkát és egyúttal növeljék annak teljesítményét. Kiváló ilyen példa a PostgreSQL kiegészítő alkalmazása, a PostGIS névre hallgató térinformatikai bővítmény, amely kifejezetten a térbeli adatok tárolását és kezelését biztosítja.

A térbeli adatbázis (*spatial database*) speciális adattípusokkal képes geometriai adatok létrehozására és tárolására, ezen kívül speciális függvényei segítségével az SQL lekérdezések alkalmassá válnak térinformatikai elemzések végrehajtására is. Egyszerűbben mondva a térbeli adatbázis egyszerre adattároló és vizsgálati eszköz, mellyel a vizsgálat elvégzésekor megjelenítésre sincs szükség. Minél összetettebb eredmény érdekében igen komplex lekérdezésekhez is megteremthetjük a feltételeket anélkül, hogy az adatmennyiségek növelésével a teljesítmény csökkenne. A térbeli tulajdonságokkal kapcsolatos információk hagyományos lekérdezésekkel kinyerhetőek, tehát azon felhasználók számára is előnyös, akik nincsenek hozzászokva a geometriai, térképi adatok elemzéséhez. Ez annak a különleges képességnek köszönhető, hogy a térbeli adatbázis képes vonalláncokkal, övezetekkel (zónákkal) és további geometriai objektumokkal modellezni a valóságot.

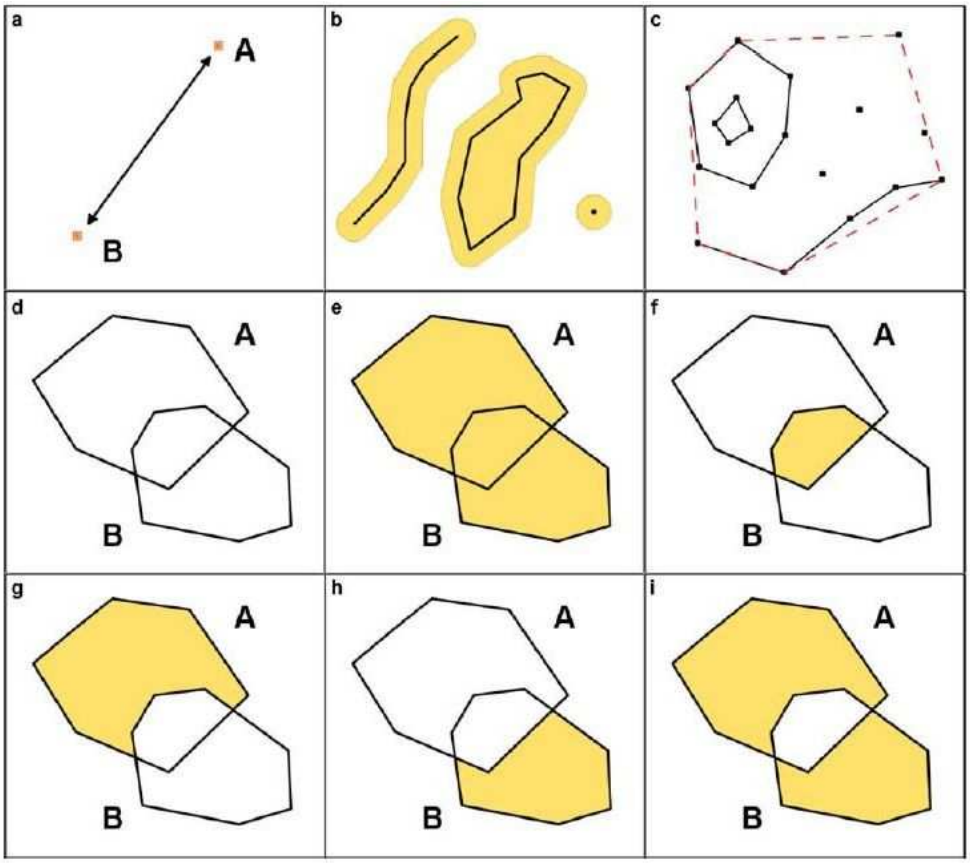
A térinformatikából már ismert módon, kétdimenziós térképek esetén háromféle alap geometriai adattípus létezik: pont, törtvonal és poligon. Az elemzés ott kezdődik, amikor ezen objektum-modellekkel műveleteket is végre tudunk hajtani. Ennek az eszköze a speciális térbeli lekérdezés, amely geometriai függvényeket használ térbeli objektumokkal kapcsolatos kérdésekre adandó válaszokhoz. A PostGIS alkalmazásával a hagyományos SQL nyelvhez számos geometriai objektumot kezelő függvényt adhatunk hozzá, melyeknek működési elve hasonló a dátum/idő típusú adatokkal kapcsolatos lekérdezésekéhez (két időpont között lévő időtartam, a kérdéses időpont múlt- vagy jövőbeli esemény-e, stb.). A függvényeket csoportosíthatjuk típusaik szerint:

- mérési műveletek (hossz, kerület, terület, távolság, stb.),
- térbeli műveletek (unió, különbség, buffer, stb.),
- topológiai kapcsolatok vizsgálata (érinti, metszi, tartalmazza, stb.).

A lekérdezések mellett a természetesen lehetőségünk van térbeli objektumok létrehozására és módosításra is. A PostGIS/PostgreSQL esetén az alapvető 3 fajta geometriai adattípus kiegészül komplexebb objektumokkal, mint a multipoint, multilinestring, multipolygon, geometriai adatcsoport és íves alakzatok. A 2 dimenziós alakzatainkat elhelyezhetjük 3 dimenziós modelltérben, azáltal, hogy magassági adatokat rendelünk hozzájuk (2,5D). A PostGIS nagyon lényeges szolgáltatása továbbá a vetületi rendszert definiálására szolgáló adattípus. Az alapértelmezett vetület a Descartes-féle koordináta-rendszer, azonban nagyjából 3000 különböző, beépített vetületi rendszer áll rendelkezésünkre, amelyekre egy-egy számkóddal lehet hivatkozni (*SRID*). Várhatóan a PostGIS következő lépése a raszter funkció bevezetése lesz, amely valójában pixelek numerikus tárolását és azok térbeli helyszínekkel való korrelációját jelentené. A PostGIS a PostgreSQL-hez hasonlóan nyílt forráskódú. [OBEH]

```
SELECT AddGeometryColumn ('epulet', 'geom', 4326, 'POLYGON', 2);
```

A fenti SQL utasítással egy térbeli adatok tárolására alkalmas oszlopot adhatunk hozzá a már meglévő táblához. A speciális PostGIS parancs az AddGeometryColumn (a többi térbeli függvény listája a PostGIS honlapján elérhető). A zárójelben lévő elemek sorrendje kötött, éspedig: tábla neve (epulet), oszlop neve (geom), vetületi rendszer kódja (4326=WGS84), geometriai elem típusa (poligon), dimenzió (2). További példák bemutatása a 4. fejezetben.



3.6. ábra: A PostGIS által támogatott térbeli műveletek: a) távolság b) övezet c) konvex befoglaló hurok e) unió f) metszet g) A-B h) B-A i) szimmetrikus különbség d) a bemutatáshoz használt alakzatok (*forrás:* [SHEK])

3.3. Web programozási környezetek

Az internetes tartalmak megjelenítéséhez a böngésző programkódokat interpretál. Ezek a kódok speciális, web programozáshoz használt programnyelveken íródnak és az őket tartalmazó fájlokra hivatkozva érhetőek el. A fájlok eléréséhez egy *URL* cím szükséges. Az URL (*Uniform Resource Locator*) egy fájl egyedi címe, amely legtöbbször az interneten található fájlokra hivatkozik, de lokális hálózaton belül (*intranet*) is alkalmazható.

A webes megoldások egyik alapfogalma a CGI (*Common Gateway Interface*) azaz egy eljárási interfész, melyen keresztül a szerver továbbküldi a felhasználói kéréseket a címzett alkalmazás felé és onnan visszajelzéseket kap, melyeket aztán visszaküld a kliens felé. A CGI-t azzal a céllal hozták létre, hogy ezen keresztül bővíthető legyen a webszerver funkcionalitása. Példa erre az OWT Chart (bővebben a 4. fejezetben), amely egy szerveren futtatott, de a webszervertől független program. A CGI gyakorlatilag programnyelvi fordítóként működik, vagyis a kliens kéréseit a szerver számára érthető módon interpretálja és értelmes válaszokat ad vissza. A CGI-nek a lehető legtöbbféle kliens és szerver számára meg kell teremtenie az átjárót. Mivel a kliensoldali böngészők, a szervertípusok és az operációs rendszerek esetenként különbözőek lehetnek, ezért egy „több nyelven értő” interfész elkészítésére van szükség, amely bármilyen programozási nyelven megírható. [SOLB]

A web kialakulásának kezdetén mindössze a *HTML* és a *CGI* létezett. A HTML (*Hypertext Markup Language*) definiálja a szöveges dokumentum részeit, és utasítja a böngészőt arra, hogy hogyan jelenítse meg őket. A HTML utasítások a megjelenítendő tartalom köré kerülnek, pl. a … közti szöveg félkövér (*bold*) karakterekkel jelenik meg. Az oldal HTML elemeinek kiválasztása és kezelése az ún. *DOM* (*Document Object* Model) szabvány szerint lehetséges. A DOM a HTML dokumentum felépítését definiálja, amely hierarchikus jellegű, a csomópontok maguk az egyes elemek és azok tulajdonságai. A megjelenő elemek stílusának leírására a CSS (*Cascading Style Sheets*) stílusleíró nyelv szolgál. A HTML nagy hátránya, hogy statikus, vagyis a megjelenített adatok nem változtathatóak, bármilyen művelet elvégzéséhez újra be kell tölteni az egész lapot, ami egyáltalán nem tetszetős megoldás, mivel lassú. Igen kellemetlen, ha egy űrlapon a felhasználó által megadott adatok ellenőrzését csak a szerveren tudjuk elvégezni – ehelyett arra van szükség, hogy az adatokat egy kliens oldali alkalmazás ellenőrizze, mielőtt a szerver felé elküldenénk őket. Az ilyen és ehhez hasonló problémák kezelésére kiváló eszköz a JavaScript, melynek műveletei a kliensoldali böngésző által is végrehajthatóak, tehát kevesebb kapcsolatot igényel a szerverrel.

Az internetes böngészők fejlődése a JavaScript programnyelvet teljes értékű webfejlesztői eszközzé tette, mivel logikus felépítésének, széleskörű alkalmazási lehetőségének és rugalmas jellegének köszönhetően igazán modern webes megoldásnak tekinthető. A HTML, CSS technológiákkal megfelelően ötvözött JavaScript tökéletesen kiegészíti azok funkcióját. A JavaScript a népszerűbb böngészők mindegyikén futtatható, azonban néhány esetben az utasításokat más módon kell megfogalmazni ahhoz, hogy működjenek.

A statikus HTML-lel ellentétben a JavaScript egy dinamikus megoldásokra specializálódott programnyelv. Miközben válaszokat adunk, a program folyamatos reakcióra képes, vagyis pl. a kurzor alakot változtat, gombok felvillannak mihelyst elhúzzuk felettük az egeret, a kijelölt szöveg színt vált, párbeszédablakok jelennek meg, stb.- mindez bőséges interaktív jellegről tanúskodik. Azonban ez nem sokat ér, ha a kliensünk a szervertől elzártan tevékenykedik, hiszen a szerveren lévő különböző, független folyamatok elérése jelentősen megnövelhetné az alkalmazás hatékonyságát. A szerver általában nagyobb teljesítményű, mint a kliens, és nagy előnye, hogy óriási információs adathalmazt tárol. A szerveren lévő adatok eléréséhez kell egy olyan program, amely kifejezetten erre alkalmas. Ilyen program pl. PHP nyelven írható. [HEIL]

A PHP egy széles körben alkalmazott, nyílt forráskódú programnyelv, melyet kezdetben webfejlesztéshez készítettek, innen ered az eredeti elnevezése (*Personal Home Page*). A PHP jól használhatónak és népszerűnek bizonyult, ezért idővel teljes értékű programnyelvvé nőtte ki magát. A mai elnevezése (*PHP Hypertext Preprocessor*) arra utal, hogy a PHP-ben megírt weboldal eléréséhez először a szerver feldolgozza a kódokat, majd csak a kész eredményt küldi el a szerver felé, amit a böngésző is tud értelmezni. Maga a PHP kód a kliens számára láthatatlan marad, ezért biztonságos. Számos további előnye:

- feldolgozása gyors, használata egyszerű (nem szakmabeliek számára is);
- számos népszerű operációs rendszer támogatja (Windows, Linux, Mac OS, UNIX változatok);
- bőséges technikai támogatás érhető el hozzá;
- a nyílt forráskód miatt saját igényekhez igazítható.

A PHP általános célú programnyelv, amellyel általános célú szkriptek készíthetőek, továbbá az egyik legnépszerűbb azon nyelvek közül, amelyeket interaktív weboldalak készítésére fejlesztettek ki. A PHP szerveroldali szkript, tehát a JavaScripttel ellentétben nem a böngésző értelmezi, hanem egy szerveroldali modul. Adatokat tud elhelyezni és lekérdezni adatbázisokból. Nem képes viszont a kliensoldali géppel kölcsönhatásba kerülni, azaz eseményekre reagálni. Ezzel szemben például a JavaScripttel nem lehet adatbázis műveleteket végrehajtani, ezért a legeredményesebb megoldás a két programnyelv kombinációja.

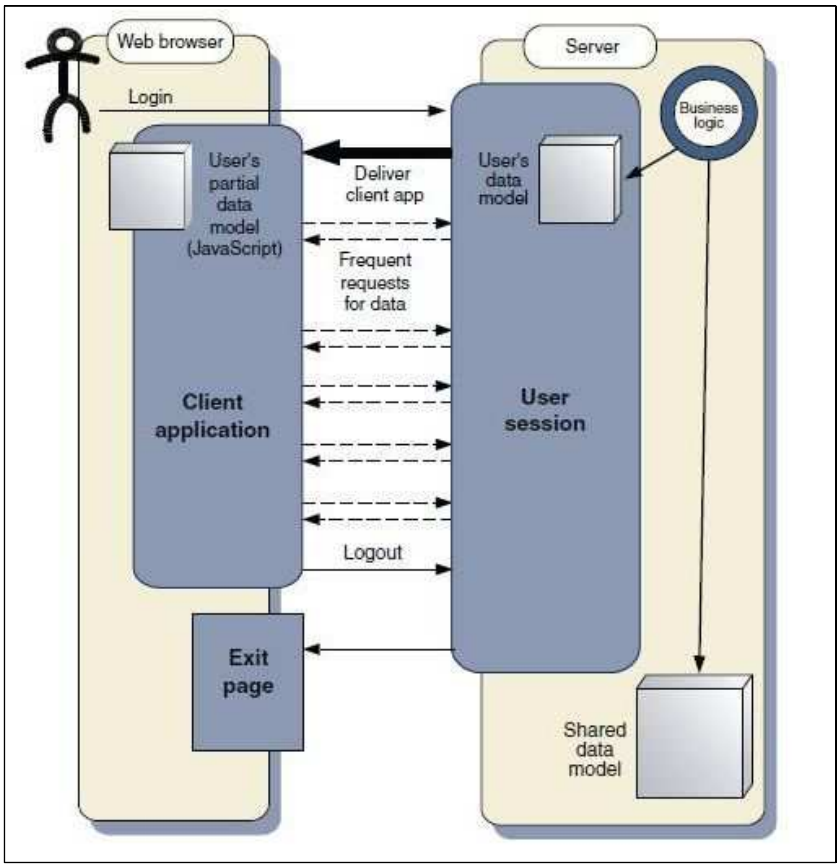
A PHP számos adatbázis-kezelő szoftvert támogat, továbbá a merevlemezen lévő fájlokat is képes manipulálni és az operációs rendszernek is tud parancsokat adni. [VALA]

A JavaScript és PHP hatékony együttműködése esetén a felhasználó nagy mennyiségű adathoz férhet hozzá, módosíthatja őket, több kliens együttes kapcsolódása esetén pedig egyszerre több felhasználó számára megoszthatóvá válnak a szerveroldali tartalmak. A rugalmas kapcsolat érdekében hozták létre a JavaScript *AJAX* elnevezésű, aszinkron működő eljárását.

Az AJAX (*Asynchronous JavaScript And Xml*) egy forradalminak mondható eljárás, amely egy speciális JavaScript objektum (*XMLHttpRequest*) segítségével lehetővé teszi a szerverrel való kommunikációt anélkül, hogy a böngészőben újra kellene tölteni az egész oldalt. Az aszinkron tulajdonság miatt egyszerre több AJAX típusú kapcsolat is létesíthető, miközben a weboldal funkciói a parancsok végrehajtása alatt ugyanúgy használhatóak maradnak. Korábban a webes technológiák egyik legkomolyabb korlátja az volt, hogy a szerver és felhasználó közötti adatcseréhez általában szükségünk volt egy új oldal betöltéséhez és megjelenítéséhez. Az AJAX alkalmazást a népszerű böngészőprogramok jelenlegi verziói mind támogatják. [HEIL]

Az AJAX leginkább a felhasználói interfészekhez (*User Interface*, *UI*) hasonlítható, amelyek az utasításaink végrehajtásához nélkülözhetetlenek, de működésük a felhasználó számára láthatatlan, vagyis nem akadályoznak az éppen aktuális munkában. Az eljárást elsősorban kis mennyiségű adatcsomagok küldésére alkalmazzák. Az XMLHttpRequest objektummal küldött kérésben egy (URL) hivatkozást adunk meg, amely az aktuális feladatot végrehajtó szerveroldali szkriptet/programot azonosítja. Az XMLHttpRequest esetén a felhasználó számára láthatatlan XML adatcsere történik a kliens és a szerver között. Az objektum az eredményt bármilyen szöveges formátumban képes fogadni, azonban a tartalom általában JSON alakjában érkezik vissza a kliens felé, amelyet a JavaScript dolgoz fel. A gyakorlatban ez egy igen gyors folyamat, és lassú szerver esetén is hatékony. Az AJAX működésére jó példa a Gmail levelező kliens, amelyben leveleket törölhetünk, stb. anélkül hogy egy új oldal betöltődésére kellene várnunk.

Az XML egy általános számítástechnikai adatformátum, webes környezetben igen jól feldolgozható, mivel az XML formában érkező választ a DOM szerint lehet kezelni. A JSON (*JavaScript Object Notation*) adatcsere formátum, melynek előnye, hogy a JavaScript könnyedén kezeli. Szöveges jellegük miatt mindkét adatformátumra igaz, hogy az ember számára is értelmezhetőek. A JSON-ben az adatokat rendezetten (tömb) és rendezetlenül (objektum) is tárolhatjuk. Az objektumot vesszővel elválasztott név-értékpárok alkotják, amelyeknek a sorrendje tetszőleges. Az objektum elemei kapcsos {}, a tömb elemei szögletes [] zárójelek közé kerülnek. [www.json.org]



3.7. ábra: Egy AJAX-szal működő alkalmazás ciklusa (*forrás:* [CRAN])

Az ábra azt szemlélteti, hogy AJAX-szal működő alkalmazás esetén a felhasználó böngészője letölti a szerverről a kliensoldali tartalmat (nagyrészt JavaScript), amely képes önállóan kezelni a felhasználó kéréseit, és ha szükséges, aszinkron módon lekér adatokat a szerverről. Ennek a megoldásnak nagy előnye, hogy a munkafolyamat során az alkalmazás nem töltődik újra, ezért a böngészőben tárolni tud bizonyos állapotparamétereket (pl. internetes vásárláskor a bevásárlókosár tartalmát).

Az AJAX használatát a Google az elsők között vezette be, és nemcsak a GMail esetében alkalmazza már 2004 óta, de a Google Suggest is ezzel az eljárással ad javaslatokat a keresendő kifejezésünkre, valamint a Google Maps is ezzel a módszerrel végzi a helyalapú kereséseket. Természetesen a Yahoo! cég is hamar felismerte az AJAX-ban rejlő előnyöket és alkalmazza is pl. Flickr nevű online fénykép-megosztó szolgáltatásánál. Ezek után belátható, hogy üzleti szempontból is igen jelentős megoldásról beszélünk.

A Google Maps-ről külön érdemes néhány szót ejteni. Ez a térképes szolgáltatás gyakorlatilag egy térképnézegető és egy kereső ötvözete. A keresés maga hagyományos webes megoldású, de a térkép AJAX módszerrel üzemel. A fontos helyszíneket jelölő *markerekre* kattintva felugró információs ablakok jelennek meg, illetve a térkép az egér segítségével „megragadható”. A térkép maga mozaikokból áll, ezért amikor elnavigálunk az aktuális nézetről, a szomszédos mozaikkockák aszinkron módon, automatikusan töltődnek be. Az új térképrészletek megjelenéséig a mozaikkocka üres, azonban ez nem akadályoz minket abban, hogy addig tovább mozogjunk a térképen, és még újabb területek megjelenítésére utasítsuk a programot. Az aszinkron folyamatnak köszönhetően a

linkek, zoom görgők és egyéb, az oldalon található opciók is elérhetőek maradnak. Használat közben a böngésző átmenetileg eltárolja a térkép azon szegmenseit, melyek már egyszer betöltődtek, ezáltal a már korábban meglátogatott területek gyorsan újratölthetőek.

A web alapú térképes alkalmazások mögött komoly térképészeti megoldásokkal felszerelt szerver áll, amelynek minősége természetesen pozitívan befolyásolja az oldal iránti érdeklődők számát, de emellett a könnyen kezelhetőség is fontos szempont. A közönség érdeklődése egyre inkább eltolódik az interaktivitásban gazdag weboldalak felé, tehát a rugalmas megoldást biztosító AJAX fényes jövő elé néz. [CRAN]

A weboldal fejlesztés egy további lehetősége a számos nagyvállalat által kínált alkalmazásprogramozási felület (*Application Programming Interface, API*), amely szabályok és specifikációk alkalmazásával lehetővé teszi egy adott szoftver szolgáltatásainak és forrásainak elérését egy másik program számára. Előnye abban rejlik, hogy leegyszerűsíti a kommunikációt két szoftver között, vagyis nem szükséges az alkalmazni kívánt programrendszer belső működésének ismerete ahhoz, hogy a felhasználóivá váljunk. Működési elve a hasonló a felhasználói interfészhez, mely a felhasználó és a számítógép közötti kapcsolatteremtést teszi lehetővé, azzal a különbséggel, hogy az API ugyanezt két szoftver között biztosítja.

API-t készíthetünk alkalmazások, könyvtárak és operációs rendszerek számára is. Az API tulajdonképpen a szükséges programnyelvek és szabványos parancsok definiálását jelenti, továbbá tartalmazhatnak eljárási-, valamint adatrendszerrel kapcsolatos specifikációkat és protokollokat, melyek a végrehajtó és felhasználó programok közötti kapcsolatteremtéshez szükségesek.

A legnépszerűbb hely alapú megjelenítésre alkalmas internetes szolgáltatás, a Google Maps a legtöbbet használt API-típusú webes termék címét is kiérdemelte. A Google Maps API lehetővé teszi azt, hogy a Google Maps-t saját magunk által készített alkalmazásokban is felhasználjuk. Az ilyen alkalmazásokat és weboldalakat, melyek kettő vagy több forrásból származó adatok, funkciók kombinálásával jönnek létre, *mashup*oknak nevezzük. A mashupok manapság egyre elterjedtebbek, és gyakorlatilag forradalmasították az információ felhasználásának és megjelenítésének módszerét. Legtöbbjük esetében a térképes alkalmazások különösen fontos összetevőknek számítanak. A Google Maps dinamikus jellemzői mögött HTML, JavaScript és CSS együttes működése áll.

A Google Maps API V3 a korábbi verzióhoz képes jelentős előrelépést jelent. A régi API szinkron modell elven működött, ezért a térkép megjelenítéséhez először rengeteg szkript egymásutánját kellett letölteni és futtatni. A webes technológiák fejlődésének köszönhetően a folyamatok modularizálhatóvá váltak, ezáltal megoldható, hogy a térkép megjelenítéshez szükséges kód töltődjön be elsőként. Az API teljesítményének növelésében fontos szerepe volt annak a törekvésnek, hogy a szolgáltatás mobiltelefonokon is használható legyen. [SVEN]

4. Saját rendszer elkészítése

4.1. Rendszerkoncepció

A rendszer elkészítésének célja az, hogy bemutassam az informatikai megoldások kiaknázási lehetőségeit a geodéziai tudományok egy adott részterületén. Az informatika fejlődése jelenleg olyan szakaszban tart, hogy bármilyen természetű feladathoz stabilan működő és a felhasználói igénynek maximálisan megfelelő alkalmazást készíthetünk. Ez alól nem képez kivételt a – kevésbé tömeges érdeklődésre számottevő és általában egyedi helyzetek megoldásával foglalkozó – geodézia tudománya sem. A világszerte hosszú évek óta zajló teszteléseknek és fejlesztéseknek hála manapság többféle szoftver és eljárási módszer közül is választhatunk, ezáltal a kihívás nem a megvalósításban rejlik, hanem a leghatékonyabb megoldás megtalálásában. Az internetes fájlmegosztásnak köszönhetően manapság már könnyedén és ingyenesen hozzá lehet jutni bármilyen szoftverhez, azonban ez nem feltétlenül jelenti azt, hogy a szoftver tulajdonosa előzőleg hozzájárult a program ily módon történő felhasználásához. 2011-es források szerint a globális szintű fájlmegosztás közel egynegyede kalózkodás. Legális tevékenységek folytatásához nem árt tisztában lenni a szoftverek felhasználási jogosultságaival, melyeknek négy kategóriája különböztethető meg:

- *korlátlan ideig ingyenesen használható*: nyílt forráskódú és ingyenes szoftverek;
- *megszűnéséig ingyenesen használható*: nyilvános szolgáltatás (pl. az Általános- és Felsőgeodézia Tanszék honlapján lévő vetületi átszámító program);
- *korlátolt ideig ingyenesen használható*: shareware, demo változatok;
- *nem használható ingyenesen*: kereskedelmi szoftverek.

A rendszer tervezésekor szem előtt tartottam a költségek minimalizálását, ezért az ingyenesen használható programok kizárólagos alkalmazása mellett döntöttem, melyeknek nagy része nyílt forráskódú szoftver. Nyílt forráskódú szoftver (*Open Source Software*) alatt azon programokat értjük, melyeknek a programozási forráskódja elérhető. Richard M. Stallman koncepciója szerint az ilyen típusú programok négy szabadságot biztosítanak a felhasználó számára:

- a program futtatása tetszőleges célból;
- a program működésének tanulmányozása és adaptálása az igényeinknek megfelelően (forráskód);
- a program továbbadásának szabadsága;
- a program továbbfejlesztési lehetősége és a fejlesztések visszajuttatása a fejlesztői és felhasználói közösségnek (forráskód).

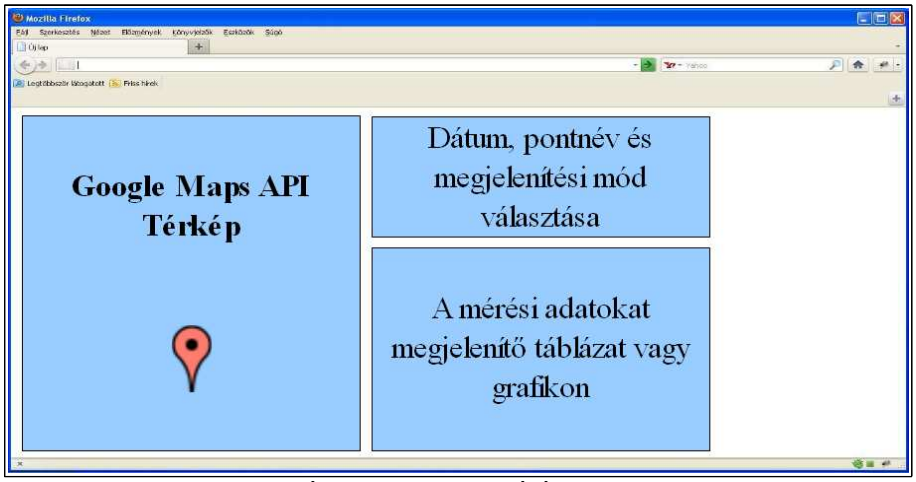
Azok a fejlesztők, akik a munkájukat nyílt forráskóddal szeretnék közzétenni, nyílt forráskódú licence-t csatolnak hozzá (ilyen például a *General Public License*, GPL). Ez a licence biztosítja a fenti szabadságokat, illetve a védelmüket is szolgálja, hiszen az ily módon közzétett szoftvereket megfelelő hozzáértéssel saját igényeinkhez igazíthatjuk, de a módosításoknak szintén GPL licenc

alatt kell megjelenniük. Ezáltal a szabad tartalmakból készült származékos mű is szabad marad. [SZ03]

A diplomaterv készítése során Windows operációs rendszert használtam, de több operációs rendszeren (Linux, OS-X, Unix) is működtethető az elkészített rendszer.

4.2. Logikai rendszerterv

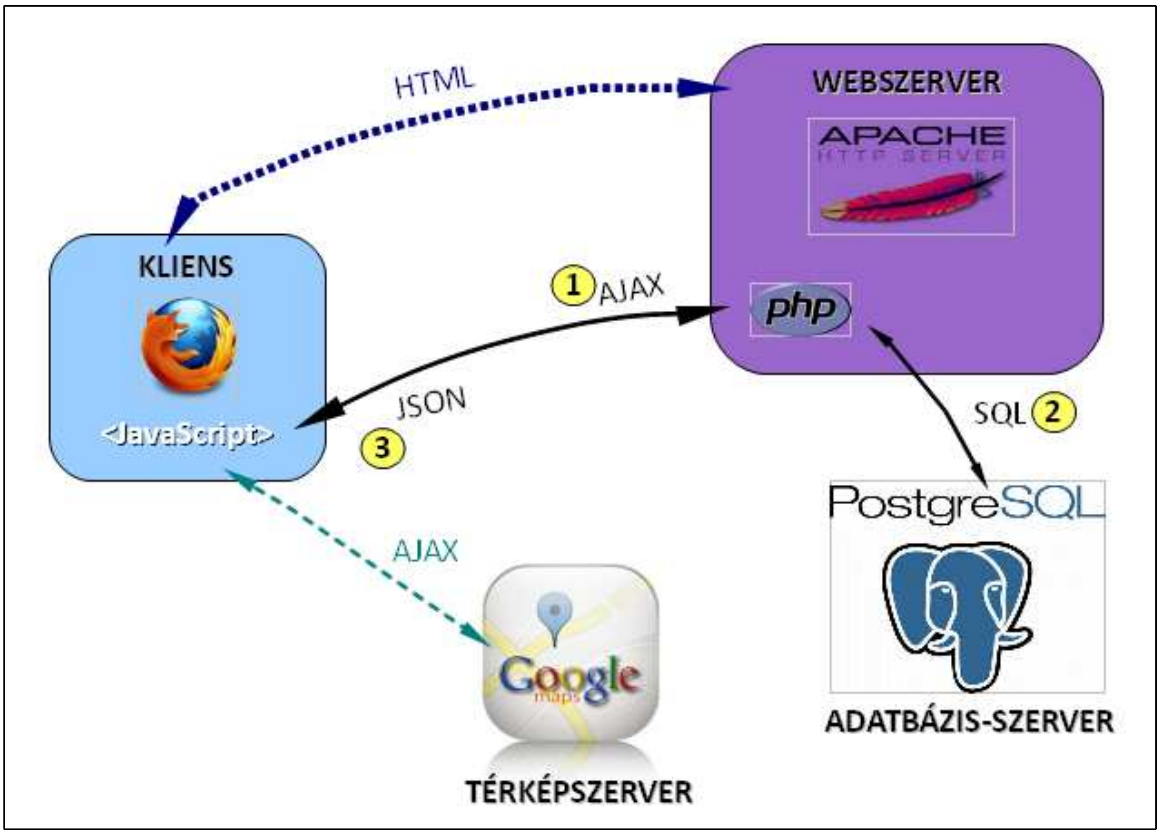
A tervezett programrendszer a 4-es metró építésével kapcsolatos mozgásvizsgálati munkák során alkalmazott CYCLOPS automata mozgásvizsgálati rendszerrel kinyert eredményeket dolgozza fel a CH épületre vonatkozóan (a SolData-HUNGEOD Konzorcium által elérhetővé tett adatokra korlátozva). A program típusa internetes alkalmazás, melynek használatához mindössze egy böngészőre van szükség a kliens oldalon. Az alkalmazás célja a mérés pontjainak hely alapú megjelenítése, valamint a mérési eredmények lekérdezése. Ehhez a felhasználó rendelkezésére bocsájtok egy térképes megjelenítő ablakot, a lekérdezéshez szükséges paraméterek bevitelére szolgáló űrlapot és a mérési adatokat szemléltető táblázatot vagy grafikont. A felhasználói felület tervezésekor figyelembe veszem, hogy 1024 pixel széles képernyőn az eredeti elrendezésben legyen látható a tartalom.



4.1. ábra: A megjelenítési elemek

A nagy mennyiségű adat hatékony feldolgozásához (38 pont mindegyikén napi egy mérési eredmény 2006-07-17-től 2011-02-22-ig tartó intervallumban), célszerű egy adatbázis-kezelő program segítségét igénybe venni a rendszerezésükhöz. A térképes megjelenítés megoldható lenne saját térkép alkalmazásával is, de jelen megoldásomhoz a Google Maps szolgáltatását használom fel. Az interneten keresztül elérhető programrendszerek esetén szerver és kliens oldali megoldásokra egyaránt szükség van. A szerveren futó szoftver komponensek esetünkben három alegységre bonthatók:

- webszerver (Apache),
- térképszerver (Google Maps),
- adatbázis-szerver (PostgreSQL).



4.2. ábra: Az alkalmazás működésének folyamata

A kliens oldalán megjelenő felület gyakorlatilag egy HTML oldal és az abba beágyazott JavaScript parancsok eredménye, amit az oldal forráskódjának neveznek. A forráskód nyilvános, teljes egészében letöltődik a kliens oldalára. A szerveren tárolt adatok eléréséhez és a velük való műveletek végzéséhez azonban a szerver oldalán is szükséges egy végrehajtó programsor, amit PHP programnyelven készítek el. Annak érdekében, hogy a kliens-szerver közti kapcsolat rugalmasan és hatékonyan valósuljon meg, aszinkron módon működő megoldást alkalmazok (AJAX).

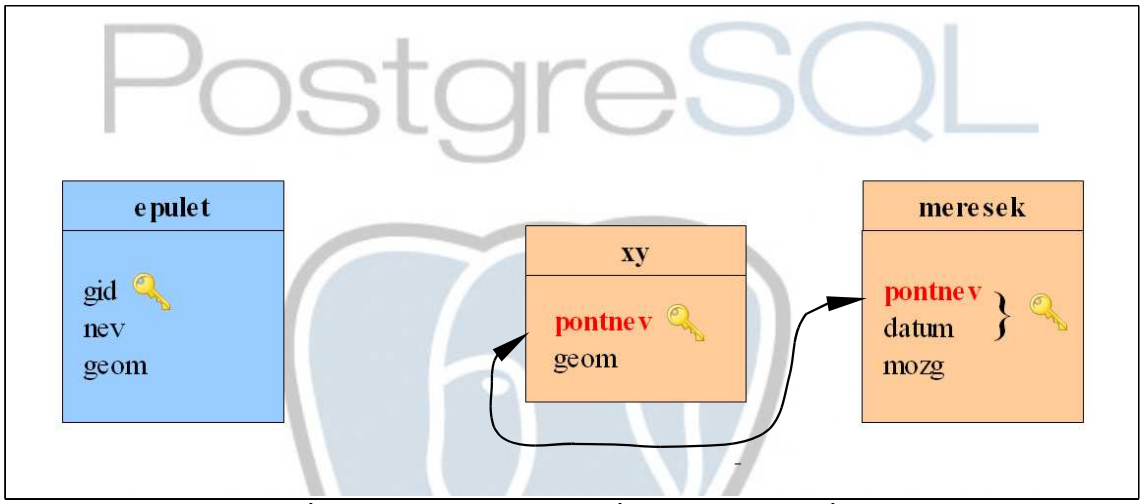
Az adatok tárolásához több adattábla szükséges, mivel csak a logikailag összetartozó adatok kerülnek ugyanabba a táblába. Összesen három adattáblát alakítok ki, melyekbe a következő adatok kerülnek:

- a mérési eredmények (pontnév, dátum, relatív z koordináta),
- a vizsgálati prizmák pontjai (pontnév, geometria),
- az épületek alaprajzi poligonjai (ID, épületnév, geometria).

Az épületek feltüntetését azért választottam, mert célprizmák a CH épület mind a négy homlokzatán vannak, de a Google Maps ferde műholdfelvételein lehetetlen úgy ábrázolni őket, hogy a pontok mind a négy irányú felvételen a helyükön látszódjanak. Az egyszerűség kedvéért a műholdfelvételek helyett az úthálózati térképet veszem igénybe, amelyre az API-val felhelyezem az épületek alaprajzát. Az adattárolást a PostgreSQL-lel valósítom meg, melynek a térképi koordináták hatékony tárolásához a PostGIS kiegészítő szolgáltatásra van szüksége. A PostGIS a koordinátákat speciális WKB (*Well-Known Binary*) adatformátumban tárolja. A WKB gyakorlatilag egy hexadecimális kód, amely az alakzatok síkbeli (térbeli) elhelyezkedése mellett az alakzat (pont,

vonal, poligon, stb.) típusát és a vetületi rendszerét is tárolja. WKB típusú kódot a felhasználó nem tud létrehozni, ilyen formátumot csak az arra alkalmas szoftverrel generálhatunk.

Az épületek alaprajzának táblázata független a másik kettőtől, nincs köztük attribútum kapcsolat. A prizmák helyzete (xy) és a prizmákra végzett mérések eredményét tartalmazó táblázatok között kapcsolat létesíthető a prizmák nevei alapján. A táblázatok sorainak egyértelműségéről a kulcsként definiált attribútumok gondoskodnak. A mérési adatok esetén az azonosításhoz szükséges kulcsot a pontnév és a dátum együtt biztosítja, mivel minden ponthoz több napi mérési eredmény tartozik, és minden egyes napon több (szerencsés esetben mindegyik) prizmán történt mérés. Az epulet tábla esetén nincs mindegyik épülethez név rendelve, ezért a poligonok azonosítását sorszám (gid) attribútum hozzáadásával oldom meg.



4.3. ábra: A PostgreSQL-ben létrehozott adatbázisok

Az adatbázis adatainak kinyeréséről PHP-ban megírt utasítások gondoskodnak. A PHP szkript biztosítja a PostgreSQL-be való belépést, ezen túl pedig beágyazott SQL utasítások vannak benne, amelyekkel lekérdezi a térképi megjelenítéshez szükséges adatokat, illetve a felhasználó által kért mérési eredményeket. A PHP kód végrehajtására JavaScript függvények adják ki a parancsot, és a visszaérkező eredményt is a JavaScript dolgozza fel. A hatékonyság növelése érdekében ezek a folyamatok aszinkron módon zajlanak, a PostgreSQL-ből lekérdezett nagy mennyiségű adatokat pedig ún. JSON formátumban kapja meg a JavaScript.

A JavaScript további feladatai közé tartozik a Google Maps API meghívása, amely szintén aszinkron módon tölti be az oldalra a megfelelő térképrészletet, de erről nem kell külön gondoskodnunk, mert az API V3-at alapból aszinkron működő alkalmazásként alkották meg a készítői. Ezen kívül a JavaScript függvények dinamikus jellegének köszönhető, hogy ellenőrizhetjük a felhasználó által megadott paraméterek helyességét, még mielőtt elküldenénk őket a szerver felé.

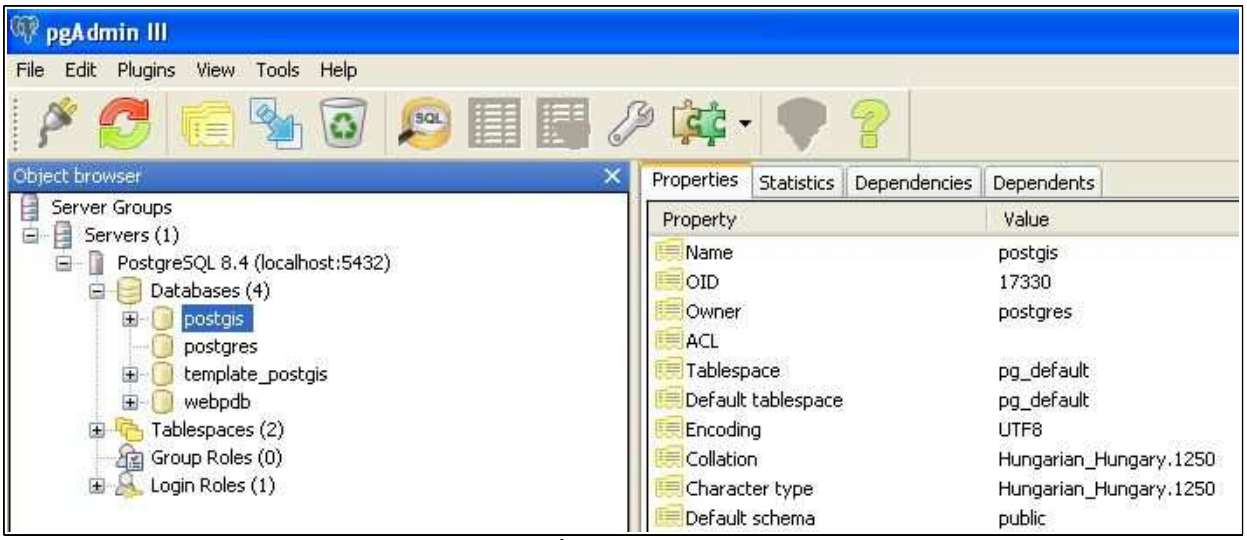
A HTML kódok a rendszer statikus megjelenítési elemeit definiálják, tehát pl. a böngésző felületének felosztását, az űrlap elemeit, és a megjelenítés stílusának paramétereit.

4.2.1. Felhasznált szoftverkomponensek és telepítésük

A munkám során a web- és az adatbázis-szerver, valamint a kliens ugyanazon a gépen valósul meg, tehát nincs szükség külső URL hivatkozásra, a szerver egyszerűen a *localhost* névvel érhető el. Mivel a használt szerver komponensek nem szerves részei az operációs rendszernek, ezért ezeket külön telepíteni kell. A munkámhoz a legnépszerűbb nyílt forráskódú webszervert, az *Apache*-t használom. Az ingyenes szoftvereket tartalmazó ms4w nevű térképészeti eszközcsomag telepítésével egy mozdulattal felkerül a gépre az Apache, a PHP programnyelv, továbbá egy térképszerver is, melyre esetünkben nincsen szükség, mivel Google Maps API-val biztosítom a térképi hátteret. Az Apache szerver beállítható, hogy az operációs rendszer indulásakor automatikusan elinduljon, ehhez a Vezérlőpult – Felügyeleti eszközök – Szolgáltatások modult kell elindítani, amely a háttérben futó alkalmazások kezelését teszi lehetővé. Az internetes oldalunkat definiáló programsorokat az Apache *htdocs* könyvtárába kell elhelyezni annak érdekében, hogy megfelelően működjenek.

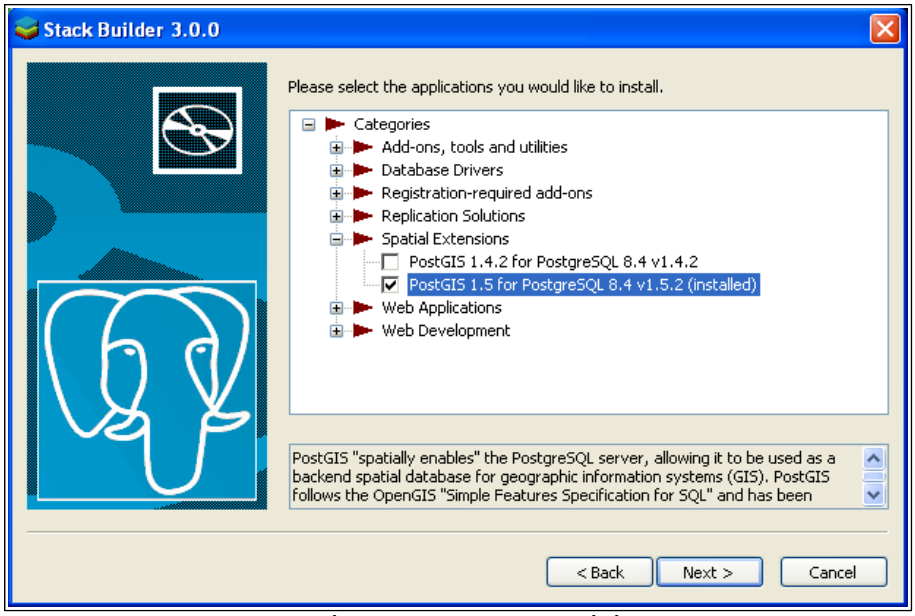
A PostgreSQL szerver egy kliens-szerver típusú adatbázis, melynek aktuális, 8.4-es verzióját telepítem. A rendszer futtatása a *host* (gazdagép) feladata, amely hálózaton keresztül érhető el. A host definiálásakor szükséges egy név és egy cím (IP cím) megadása. Esetünkben a TCP/IP kapcsolat ugyanazon a gépen valósul meg, tehát a host megadása a „localhost” névvel történik. A kliens kapcsolódása a hálózati *porton* keresztül történik, amely egy azonosító szám (PostgreSQL esetén 5432 az alapértelmezett érték). Az adatbázis-szerver kapcsolódáshoz a kliensnek meg kell adnia a hálózati portot. Amennyiben a szerveren több PostgreSQL adatbázis-kezelő is fut, mindegyik egy külön hálózati porton keresztül érhető el. Az adatbázisszerverhez való kapcsolódáskor létrejön egy munkamenet (*session*), amely mindaddig fennáll, amíg az adatbázis-műveletek befejeztével meg nem szüntetjük. Minden egyes kapcsolat egyedi azonosítóval rendelkezik.

A PostgreSQL-hez alkalmazható grafikus felhasználói interfészek közül először a PHPPgAdmin-t telepítettem, amelyet nem találtam eléggé felhasználóbarátnak, ezért inkább a PgAdmin3 nevű szoftver használatára váltottam. A pgAdmin3 egy kliens oldali program, ezért használatakor első lépésként a szerver regisztrációja szükséges. A definiálandó adatok között szerepel a host (localhost), a port (5432), az alapértelmezett adatbázis (postgres), a felhasználónév (postgres) és a jelszó.



4.4. ábra: pgAdmin3

A Windows Intézőhöz hasonló felépítésű pgAdmin3 kezelése egyszerű, a kiválasztott objektummal kapcsolatos tevékenységek a jobb egérgombbal való kattintásra legördülő menüben találhatóak. A kiválasztott parancsokat az interfész SQL utasítások formájában küldi a PostgreSQL szerver felé, amely végrehajtja őket. A felhasználónak lehetősége van arra is, hogy közvetlenül SQL utasításokkal adja meg a kívánt műveletet. Ezt a fajta megoldást akkor találtam hasznosnak, amikor nagy mennyiségű adatot kellett bevinnem az adatbázisba.



4.5. ábra: A PostGIS telepítése

A PostGIS telepítése a PostgreSQL Stack Builder alkalmazásával lehetséges. Ez egy egyszerűen kezelhető letöltéseket és telepítéseket vezérlő varázsló, amelyben néhány lépéssel elvégezhető a kívánt művelet. A telepítés során a PostGIS 1.5-ös verzióját választom. A PostGIS kétféleképpen tud WKB típusú geometriai (GIS) adatokat létrehozni, az egyik megoldás a speciális SQL függvény (ST_GeomFromText), amely WKT (*Well-Known Text*) vagyis szöveges formátumban megadott vektor típusú elemből generál WKB kódot; a másik lehetőség pedig az

shp2pgsql modul, amellyel ESRI Shape fájlból tudunk PostgreSQL/PostGIS számára megfelelő adatformátumot előállítani.

A térinformatikai jellegű feladatokhoz szükséges szoftverek az OSGeo4W programcsomag telepítésével kerülnek fel a gépre. Ez a csomag olyan Win32 alapú nyílt forráskódú térinformatika szoftverek gyűjteménye, melyeket az OSGeo alapítvány jóváhagyott. Az OSGeo (*Open Source Geospatial Foundation*) non-profit szervezet célja a nyílt forráskódú térinformatikai szoftverek és adatok fejlesztésének támogatása, tehát az általuk ajánlott programok alkalmazása minőségi munkát garantál. Az ms4w telepítésekor már említett térképszerver, a PostGIS, valamint az OSGeo4w-ben található QGIS és GRASS programok mind OSGeo által támogatott termékek. A GRASS program a QGIS-szel együtt kerül fel a gépre, azonban a munkám elkészítéséhez csak a QGIS-re van szükségem.

4.3. Megvalósítás

A programrendszer megalkotása három lényeges munkafolyamatra bontható: adatbázis-készítés, szerveroldali szkriptek és kliensoldali programsorok megírása.

4.3.1. PostgreSQL adattáblák készítése

A mérési eredmények táblázata: a SolData-HUNGEOD Konzorcium munkatársától a prizmákra végzett mérési adatokat egy XLS formátumú Excel táblázatba rendezve kaptam meg. Az XLS fájlt az Open Office program Calc komponensével kezelem. A táblázat elvben 2006-07-17 és 2011-02-22 intervallumban napi szinten egy mérési adatot tartalmaz minden prizmára. A valóságban azonban az adatsorok időben nem mindenütt folytonosak, mivel akadnak napok, melyeken bizonyos prizmákhoz nem tartoznak mérési adatok. A táblázatban lévő adatokat ún. strukturált szövegfájlként (comma-separated values, CSV) exportálom. A CSV formátum olyan fájltípusok összessége, amelyek táblázatos adatokat egyszerű szöveges dokumentumként képesek tárolni. A CSV szövegben a sorok felelnek meg az oszlop rekordjainak, a mezőket pedig a sorok elemeit elválasztó vesszők (vagy esetenként más karakterek) jelzik. Az alábbiakban látható néhány elem a létrehozott CSV fájlból, melyeknek csak 3 attribútuma (pontnév, dátum, süllyedési érték) tartalmaz számomra hasznos információkat.

```
TGLCHWE0303Z;2011.02.22.;-41
TGLCHWE0303Z;2011.02.21.;-41
TGLCHWE0303Z;2011.02.19.;-42
          [ … ]
```

A táblázat létrehozása és az adatok bevitele két SQL utasítással valósult meg:

```
DROP TABLE IF EXISTS meresek;
CREATE TABLE meresek (
      pontnev CHAR (12) NOT NULL,
      datum DATE NOT NULL,
      mozg SMALLINT NOT NULL,
      PRIMARY KEY (pontnev,datum)
);

CREATE INDEX meresek_pontnev ON meresek (pontnev);
CREATE INDEX meresek_datum ON meresek (datum);
```

Az oszlopok megadásakor szükséges megadni a bennük tárolt adatok típusát (és esetenként azok hosszát), valamint kikötést tenni arra, hogy a cellák értéke nem lehet üres (NULL) érték. Az azonosításhoz szükséges kulcs(ok) definiálása szintén a tábla létrehozásakor történik.

```
COPY meresek (pontnev,datum,mozg)
    FROM 'CH.csv'
    WITH DELIMITER AS ';';
```

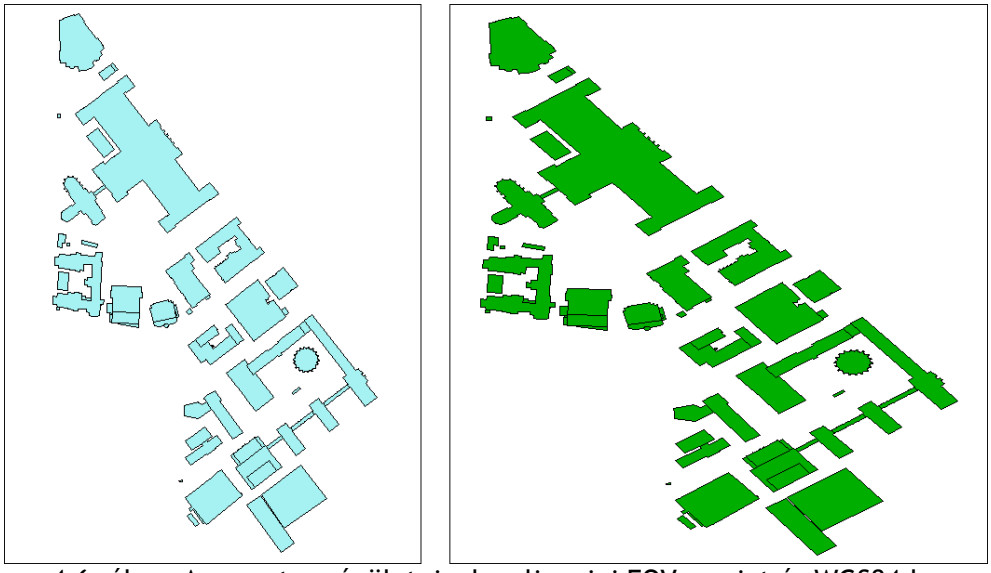
A COPY … FROM paranccsal adatokat másolhatunk egy fájlból a PostgreSQL adattábláiba. Meg kell adni a tábla azon oszlopait, ahová az adatokat szeretnénk helyezni, az adatokat tartalmazó fájl elérhetőségét, illetve CSV fájl esetén az elválasztójel típusát. A táblázat feltöltése után találtam néhány hibás adatot, melyek 10 méter nagyságrendű értékeikkel kilógtak a mm-es nagyságrendű süllyedések sorából. Ezeket a számadatokat SQL lekérdezésekkel hamar kiszűrtem és eltávolítottam a halmazból.

A célprizmák pontjainak táblázata: a mérési adatok táblázatában lévő pontnevek (kódok) alapján nem egészen egyértelmű a prizmák pontos helye az épületen. A kérésemre a SolData-HUNGEOD Konzorcium rendelkezésemre bocsájtott néhány fényképeket az épület minden oldaláról, melyeken látható a prizmák pontos helye, így tehát be tudom azonosítani a pontok vízszintes helyzetét és el tudom őket helyezni az épület alaprajzán. A Google Maps internetes oldalán a „Mi van itt?” funkcióval lehet megszerezni egy adott pont koordinátáit. Ahhoz, hogy az összegyűjtött adatok bekerüljenek a PostgreSQL rendszerébe, először egy két attribútummal (pontnév, geometria) rendelkező táblát kell létrehozni. A tábla feltöltését SQL utasítással végzem. A diszkrét koordinátákat egy speciális PostGIS függvény (ST_GeomFromText) alakítja át WKT formátumból WKB kódokká. Az egyes pontok tárolásához a következő SQL utasításra van szükség:

```
INSERT INTO xy (pontnev, geom) VALUES
('TGLCHSD0307Z',ST_GeomFromText('POINT(19.054349 47.482460)',4326));
```

Ezek után az **xy** nevű tábla *geom* oszlopában már csak egy szoftver által értelmezhető hexadecimális kódot láthatunk.

Az épületek alaprajzi poligonjainak táblázata: mivel a Google Maps API lehetővé teszi a felhasználó számára a poligonok ábrázolását, ezért egy alakzat geometriájának ismeretében írható olyan szkript, amely megjeleníti a térképen az épület körvonalait. A CH épület alaprajzi koordinátáit a tanszéki szerveren található modell feldolgozásából nyertem ki. A Shape fájl típusú ingatlannyilvántartási alaprajz koordinátáinak felhasználásához először egy vetületi transzformációra volt szükségem, mivel az eredeti, EOV rendszerbeli adatokat a Google Maps API nem tudja értelmezni. A Quantum GIS (QGIS) térinformatikai szoftver segítségével WGS 84 koordinátákká alakíthatóak az eredeti EOV adatok. Mivel ez a transzformáció méteres pontosságot biztosít, ezért a kapott koordináták geodéziai célra nem alkalmasak, de a feladatomhoz szükséges alaprajz ábrázolásához megfelelőek.



4.6. ábra: Az egyetem épületeinek poligonjai EOV szerint és WGS84-be transzformálva (QGIS ábrák)

Természetesen az internetes programrendszerhez az ESRI Shape formátum nem megfelelő, hiszen ahhoz, hogy a Google Maps API-val megjeleníthetőek legyenek a poligonok, diszkrét Y és X koordinátákra (egészen pontosan szélességi és hosszúsági adatokra) van szükség. A QGIS-ből a Shape fájl tartalmát a PostgreSQL-be importálom a fentebb már említett shp2pgsql paranccsal. Az eredmény egy WKB formátumú geometriai adat, amellyel a poligonok sarokpontjainak koordinátái, a geometriai alakzat típusa (poligon) és a vetületi rendszer egy attribútumként tárolható. Mivel a Shape fájlban nem mindegyik épülethez volt hozzárendelve név, ezért az egyértelmű azonosításhoz egy sorszámmal látom el őket, így végül három oszlop kerül az **epulet** adattáblába.

```
DROP TABLE IF EXISTS epulet;
CREATE TABLE epulet (
      gid SERIAL NOT NULL,
      nev VARCHAR (10),
      geom GEOMETRY,
      PRIMARY KEY (gid)
);
```

4.3.2. A szerveroldali műveletek végrehajtása

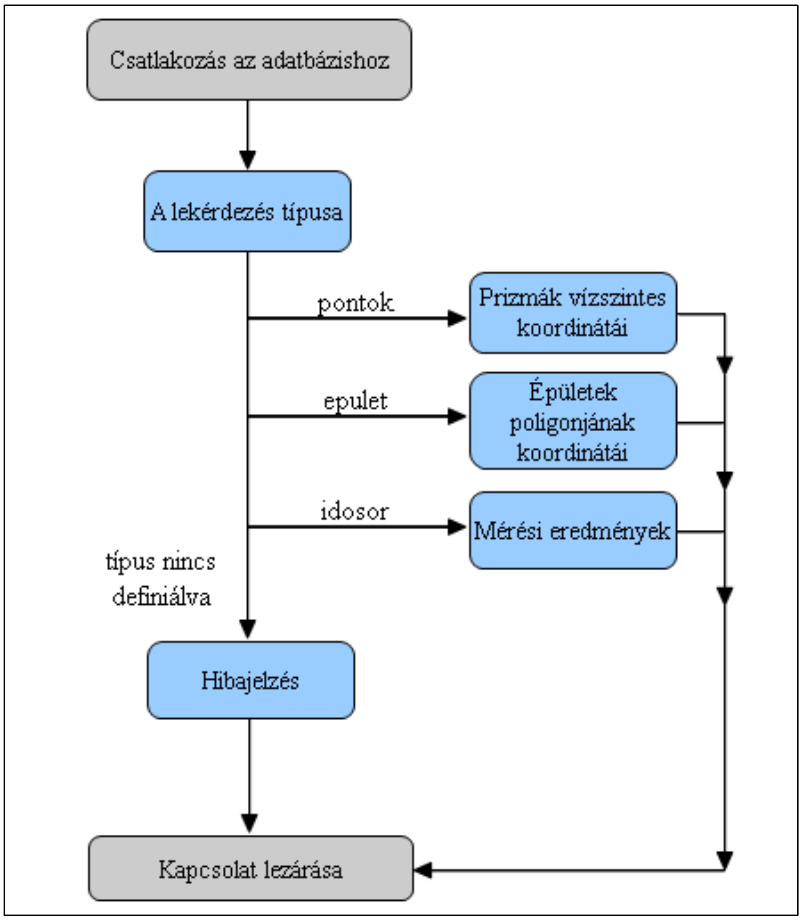
A szerveroldali műveletek alatt az adatbázis tartalmának kinyerését és azok kliens felé történő elküldését értjük. Ehhez egy programsort kell írni, amit PHP nyelven készítek el. Ahhoz, hogy a PHP fel tudja dolgozni az adatbázis-kezelés SQL utasításait, az Apache szerver PHP értelmezőjében aktívvá kell tenni a PostgreSQL függvényeket tartalmazó modult. Az ehhez szükséges php.ini fájl az Apache-on belül lévő cgi-bin könyvtárban található. A Windows alapú bővítmények felsorolásában egy bizonyos **extension=PHP_pgsql.dll** sort kell megkeresni és törölni a sor eleji komment jelet (#). A DLL (*Dynamic Link Library*) fájlok a Windows operációs rendszer alkalmazásainak segédfájljai, tartalmuk csak akkor töltődik be a memóriába, amikor egy programnak szüksége van rájuk és meghívja őket. Amennyiben több program akar egyszerre használni egy DLL fájlt, akkor is csak egyszer kell benne lennie a memóriában.

A PHP-val elsőként gondoskodni kell az adatbázishoz való csatlakozásról. Hogy minél univerzálisabban felhasználható programkódom legyen, a csatlakozáshoz szükséges adatokat egy külön PHP fájlban tárolom, ami csak a konfigurációs változókat tartalmazza. Előnye, hogy az adatok megváltozása esetén nem magában a műveleteket tartalmazó PHP szkriptben keresgetni és javítani a paramétereket, amely hibára ad lehetőséget. Jelen esetben ez a fájl a következőképpen néz ki:

```
<?php
$host="localhost";
$dbname="postgis";
$user="postgres";
$pass="******";
?>
```

A szkriptben erre a fájlra hivatkozok amikor a kapcsolatot létesítem a PostgreSQL-lel.

A PHP szkript lényege egy háromágú elágazás, amelyek különféle adatok megjelenítéséhez szükséges értékek lekérését hajtják végre, de csak abban az esetben, ha az adott feladat meghívásra kerül. Mindhárom ág egy-egy AJAX típusú JavaScript hívással lesz elérhető, és mindhárom egy-egy JSON stringet ad vissza. Az egyes ágakon lévő utasítások tehát a lekérdezéshez szükséges SQL parancsokból és a JSON string előállításából állnak. Az első kettő ág a térképes megjelenítéshez szükséges (épületeket ábrázoló poligonok, prizmák helyét jelző pontok), a harmadik ág pedig a felhasználó által kért mérési eredményeket továbbítja a kliens felé.



4.7. ábra: A PHP programsor folyamatábrája

Ugyanez kód formájában a következőképpen alakul:

```
<?php
include_once("config.php");
$conn=@pg_connect ("host=$host dbname=$dbname user=$user password=$pass");
[ … ]

switch ($_REQUEST["tipus"]) {
case "pontok" : […]
      break;
case "epulet" : […]
      break;
case "idosor": […]
      break;
[ … ]

}
pg_close();
?>
```

A PostgreSQL a koordinátákat WKB formátumban tárolja, azonban a Google Maps API csak a szöveges formátumú, konkrét szélesség és hosszúság értékeket tudja kezelni. Szükséges tehát, hogy a PHP szkriptben foglalt SQL utasítással a konkrét koordinátákat kérjük le. A prizmák pontjai esetén a JSON string szerkezete egyszerűbb, mert nincsenek benne altömbök, hanem csak egy nagy tömböt tartalmaz, benne elemenként a pontok nevével és vízszintes koordinátáikkal.

```
{"xy":

      [
      {"pontnev": "TGLCHES0201Z","lng": "19.054713","lat": "47.482812"},
      {"pontnev": "TGLCHES0203Z","lng": "19.054496","lat": "47.483087"},
      [ … ]
      {"pontnev": "TGLCHWE0303Z","lng": "19.053997","lat": "47.482596"}
      ]

}
```

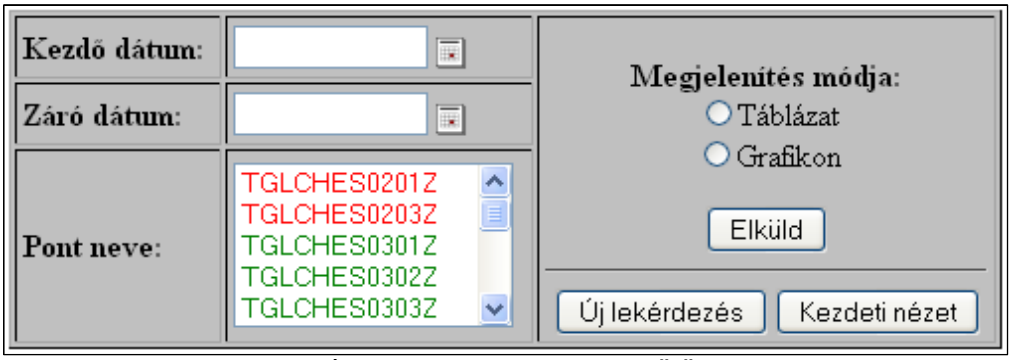
Az épületek poligonjaihoz azonban olyan JSON string szükséges, melyben a külső tömb több altömböt is tartalmaz, hiszen az egyes épületek különálló egységek, ezért a sarokponti koordinátákat poligononként elkülönítve célszerű tartani. A külső tömb elemei az épületek, a belső tömb elemei pedig a töréspontok WGS84 koordinátái.

```
{"epuletek":
        [{"nev": "MT épület", "pontok":
                                [{"lng": "19.0564284579814","lat": "47.4796396443836"},
                                {"lng": "19.0563529571093","lat": "47.4796008817287"},
                                […]
                                {"lng": "19.0548947932599","lat": "47.4795453827175"}]
        },{"nev": "L épület", "pontok":
                                [...]
        },{"nev": "K épület", "pontok":
                                [...]
        },{...
        }]
}
```


Az altömbök elemeinek kiválasztása minden épület esetén ugyanaz a folyamat, ezért az ilyen esetekben a hatékony munkavégzéshez ciklust kell beépíteni a programba. A ciklusmag egyes feldolgozása során az **epulet** tábla egy sorát olvasom ki az adatbázisból és átalakítom JSON formátumba.

```
case "epulet" : $sql="select nev, ST_AsText(geom) as g, gid,
ST_NPoints(geom) as n from epulet";
      $resource=@pg_query ($sql);
      $buf = "";
      $buf .= "{'epuletek':\n[";
      while (($r=pg_fetch_array($resource,NULL,PGSQL_ASSOC))) {
            $p = preg_split('/,/',trim(substr($r["g"], 7), "()"));
            $buf .= "{'nev': '" . $r["nev"] . "' , 'pontok': [";
            for ($i=0; $i < $r["n"]; $i++) {
                  $pi = preg_split('/ /', $p[$i]);
                  $buf .= "{'lng': '" . $pi[0] . "' , 'lat': '" . $pi[1] .
"'},\n";
            }
            $buf = trim($buf, ",\n");
            $buf .= "\n]\n},";
      }
      echo trim($buf, ",\n");
      echo "\n]\n}";
      break;
```

Ahhoz, hogy a WKB formátumban tárolt geometriai adatokat a Google Maps API értelmezni tudja, WKT formátumú diszkrét szélesség/hosszúság értékekben kell megadni a pontok helyzetét. Az átalakítást a speciális ST_AsText() nevű PostGIS paranccsal végeztem, amely metaadatok nélkül adja vissza a koordinátákat. Az ST_NPoints() megadja, hogy a WKB formában tárolt geometriai adat (esetünkben poligon) hány darab pontot tartalmaz. A preg_split utasítással szétdarabolható az adott szöveg, a paraméterben megadott elválasztó karakterekkel.



4.8. ábra: Az oldalon megjelenő űrlap

A mérési eredmények megjelenítéséhez szükséges ág bonyolultabb abból a szempontból, hogy a felhasználó által bevitt paraméterek függvényében kell lekérni az adatokat.

A felhasználó a HTML-ben található űrlap mezőibe viszi be a kiválasztott pontnevet és időintervallumot, tehát ezen mezők tartalmára kell hivatkozni a PHP-ben. A JSON string tömbje szintén altömbökből épül fel, mégpedig annyiból, ahány pont mérési adatait lekérjük. Mivel a pontok száma a felhasználótól függ, a lekérdezéshez szükséges ciklus végértéke mindig az éppen kért pontok számával egyenlő, ezért ezzel a feltétellel adható meg.

A PHP szkript végén szükséges az adatbázissal való kapcsolat lezárása. A szkriptben foglalkoztam hibakezeléssel is, tehát olyan esetekben, mikor valamilyen oknál fogva nem hajtható végre az utasítás (pl. nem lehet kapcsolódni a PostgreSQL-hez, vagy nem található a HTML fájlban a keresett hivatkozás) a PHP a böngésző felületén a konkrét hibával kapcsolatos üzenetet ad vissza. Abban az esetben, ha nincs konkrét hibaüzenet definiálva, hiba esetén egy általános PHP hibaüzenet jelenik meg, amelyből nehéz megállapítani, hogy hol lehet a gond.

4.3.3. Megjelenítés Google Maps API-val

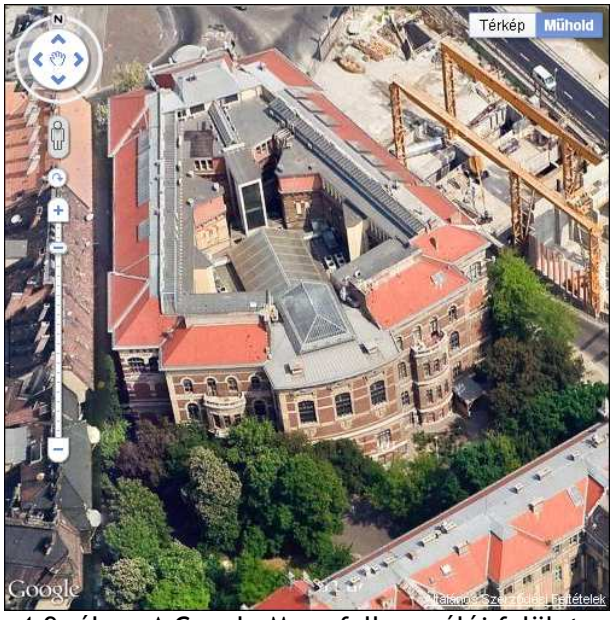
A HTML dokumentum két fő részből áll, a fej (*head*) és a törzs (*body*), amelyek közül a törzs tartalmaz a felhasználó számára is megjelenő elemeket. Kivétel egyedül az oldal címe (*title*), amelynek a header részbe kell kerülnie. A felhasználó a dokumentum címét a böngésző címsorában fogja látni, a fájl nevét pedig az oldal hivatkozásakor szükséges megadni, a két fogalom tehát független egymástól. A header tartalmazza továbbá a megjelenítés stílusát definiáló sorokat és általában a JavaScript kódokat.

A programkódban a JavaScript függvényeknek meg kell előzniük az őket meghívó elemek definiálását, ezért a legcélszerűbb a header részbe elhelyezni őket. A Google Maps API V3 alkalmazásához szükséges kódokat a Google által készített hivatalos segédletben (code.google.com/intl/hu-HU/apis/maps/documentation/javascript) lehet megtalálni, amelyben az API által kínált egyéb lehetőségekről is leírást találhatunk.

Ahhoz, hogy a térkép megjelenjen a saját oldalunkon, a Google Maps API V3 JavaScript rutinjainak elérhetőségét és az alábbi paramétereket kötelező megadni:

- a nagyítás (*zoom*) mértéke,
- a kép közepének (*center*) koordinátái,
- a megjelenített térkép típusa (térkép, műholdfelvétel, hibrid).

Ezen kívül HTML tagekkel meg kell adni az API elhelyezésére szolgáló mezőt és annak méretét. Természetesen számtalan további eszköz áll rendelkezésünkre, amellyel tartalmasabbá tehetjük az alkalmazást, de az alábbiakban a létező legegyszerűbb Google Maps API V3 alkalmazást szeretném bemutatni, amelyhez a fent említett kódok elegendőek. A megjelenítéssel kapcsolatos paramétereket egy „initialize” nevű függvény tartalmazza. A térkép számára létrehozott „map canvas” azonosítójú HTML oldalelembe a DOM segítségével helyezhető el a térkép. Maga az API az oldal betöltésével (*onload*) egyidőben jelenik meg.



4.9. ábra: A Google Maps felhasználói felülete

```
<html>
<head>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?
sensor=false">

function initialize() {
    var latlng = new google.maps.LatLng(47.482838,19.054386);
    var myOptions = {
      zoom: 19,
      center: latlng,
      mapTypeId: google.maps.MapTypeId.SATELLITE
    };

    var map = new google.maps.Map(document.getElementById("map_canvas"),
myOptions);
}

</script>
</head>
<body onload="initialize()">
      <div id="map_canvas" style="width:500; height:500"></div>
</body>
</html>
```

Budapestre nincsen Street View szolgáltatás, ezért az ikonját a későbbiekben megszüntetem. Mint ahogy már említettem, műholdkép helyett az úthálózati térképet fogom alapértelmezettként használni és a 45°-os műholdnézetet is kikapcsolom.

A feladatomban jelölni szeretném a vizsgálati prizmák helyét és az épületek alaprajzát, amihez először a PostgreSQL adatbázisból kell kigyűjteni a szükséges adatokat a PHP szkript segítségével. A rugalmas kliens-szerver kommunikációhoz az AJAX eljárást használom. Mivel három logikailag elkülöníthető adattáblám van (amelyeket a PHP is külön kezel), ezért összesen három különböző típusú AJAX kérést kell küldeni a szerver felé. A működésükhöz egy speciális, JavaScript objektumot hozok létre. A gond az, hogy az Internet Explorer esetén az objektum neve más. Ahhoz, hogy böngésző-függetlenné tegyem az alkalmazást, az Internet Explorerrel kompatibilis objektum használatát is lehetővé kell tenni. Erre az alábbiakban bemutatott speciális programnyelvi szerkezet ad módot, amely arra szolgál, hogy végigpróbáljuk a létező objektumokat és kiválasszuk azt, amelyik az adott feltételek mellett működik.

```
function GetXmlHttpObject() {
      try {
            xmlHttp=new XMLHttpRequest();
      }
      catch (e) {
                  try {
                        xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
                  }
                  catch (e) {
                        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
                  }
      return xmlHttp;
}
```

Böngészőfüggetlen kódok készítésére elsősorban az Internet Explorer szabványtól eltérő megoldásai miatt van szükség. Az ilyen kódok komplikáltak, ezért megírásuk alapos odafigyelést igényel.

Esetünkben a változó azt az értéket kapja, amelyiket az aktuális böngésző értelmezni tud. Nagyon fontos, hogy globális változóként kezeljük az objektumot, mivel a függvényen kívül is szeretnénk használni. Az AJAX utasításokban definiálni kell azt az URL-t, amely a PHP szkript megfelelő soraira hivatkozik. Attól függően, hogy a PHP fájlban lévő elágazások közül melyiket hívjuk meg, az URL-ek a következőképpen alakulhatnak:

```
chepulet.php?tipus=pontok
chepulet.php?tipus=epulet
chepulet.php?tipus=idosor
```

Az AJAX XMLHttpRequest objektumának jellemzője a readyState, vagyis az állapotjelző. Ez a paraméter megadja, hogy az adatletöltés melyik szakaszában vagyunk. Az AJAX kérésünk eredménye akkor válik elérhetővé, amikor a readyState értéke 4-gyel egyenlő.

0-4-ig öt különböző állapotot különböztetünk meg:

- 0 - nincs inicializálva, megfelelő inicializálás nélkül az objektum nem működőképes;
- 1 - letöltés, a kért adatok éppen töltődnek a szerverről;

• 2 - letöltés vége;

• 3 - interaktív állapot, ekkor a felhasználó kapcsolatba léphez az objektummal akkor is, ha a letöltődés nem fejeződött be;

• 4 - inicializálva.

A feladatomban a kérés eredményét minden esetben JSON formátumban állítom elő, melyet egyetlen JavaScript utasítással objektummá alakítok át. Az egyes objektumok elemeivel minden esetben ugyanaz a teendő, ezért ciklus utasítással adom meg a feladatot. A **pontok** és az **epulet** tábla tartalma a térképes megjelenítéshez szükséges. A pontszerű objektumok jelölésére a Google Maps ún. *markereket* használ. A háromféle magasságban elhelyezett prizmákat különböző színű markerekkel különböztetem meg. Ezt könnyedén meg tudom valósítani, mivel a pontnevekbe bele van kódolva a pontok magassági helyzete, azaz az első kettő számjegy 01, 02 vagy 03 lehet aszerint, hogy melyik szinten található az adott prizma. A különböző színű markerek Benjamin Keen honlapjáról származnak.

A markerekhez információs ablakokat (*infowindow*) is hozzáadhatunk, amelyek kattintásra jelennek meg. Ez programnyelven úgy néz ki, hogy egy eseményfigyelő objektumot hozok létre, amely a kattintás eseményére megjeleníti az ablakot. Ezek az ablakok tartalmazhatnak szöveget, képet, URL címet, stb. A feladatomban arról adnak információt, hogy az adott prizma az épület melyik homlokzatán található. A pontnevek 5. betűje mindig az égtáj rövidítését tartalmazza, ezért elegendő megvizsgálni a JSON megfelelő helyén található pontnév megfelelő karakterét. A lehetséges esetek vizsgálatához a PHP szkriptben már bemutatott switch-case szerkezet az ideális, amely JavaScriptben is működik.

Az épületek poligonjainak elhelyezése hasonlóképpen történik, mint a markereké azzal a különbséggel, hogy az épületek adatait tartalmazó tömb minden eleme egy további tömböt tartalmaz: az épületek töréspontjainak koordinátáit.

A markerek és a poligonok elhelyezéséhez a Google Maps API V3 hivatalos használati útmutatójában talált sémákat használtam fel.

A markerek és a poligonok nincsenek felhasználói utasításhoz kötve, a térkép inicializálásával együtt automatikusan töltődnek be.

4.3.4. Űrlap és táblázat készítése

A felhasználó számára megjelenő űrlapokat táblák formájában adom meg, amelyek oszlopokból és sorokból állnak, továbbá ezek az elemek – ugyanúgy mint az Excel táblák esetén – helyenként összevonhatóak. A tábla felosztását és a benne megjelenő statikus tartalmakat HTML kódokkal definiálhatjuk. A mozgásvizsgálati paraméterekhez egy 3×3-as táblát készítek, amelynek az utolsó oszlopa összevont sorokból áll. Az időintervallumhoz szükséges dátumok megadásához egyszerű szöveges mezőt (*text*) hozok létre, a pontneveket pedig egy többválasztós legördülő menüben lehet kijelölni.

Azok számára, akik nem ismerik a pontok nevében kódolt információkat, nem egyértelmű, hogy az épületen hol helyezkednek el a listában megadott pontok. Meg kell adni a lehetőséget arra, hogy a térképen megjelenő markerek alapján is ki lehessen jelölni a pontot. Ehhez felhasználható az információs ablakok megjelenítéséhez létrehozott eseményfigyelő. Amikor a markerre rákattintunk,

az ablak megjelenése mellett a program kijelöli azt a pontnevet a menüben, amelyik megegyezik a jelölt pont nevével.

Az adatsorok megjelenítési módját ún. rádió gombokkal választhatjuk ki, amelynek lényege, hogy az opciók közül mindig csak egy lehet kijelölve. Az eredmények lekérdezéséhez szükséges utasítások végrehajtására az „Elküld” feliratú gombbal lehet kiadni a parancsot. A kattintásra reagáló függvény ellenőrzi, hogy minden mező ki van-e töltve és hogy kiválasztottuk-e a megjelenítés módját. Ha mégsem, akkor figyelmeztetést küld, egyébként pedig egy AJAX kapcsolat során a PHP kód **idosor** nevű feladatának az eredményeit kapja meg. Mivel az adatsorok tartalma függ a felhasználó által bevitt paraméterektől, ezért ezeket a feltételeket a PHP-nek el kell juttatni. Az objektummá alakított JSON formátumú adatsorral kétféle célunk lehet: táblázat vagy grafikon készítése.

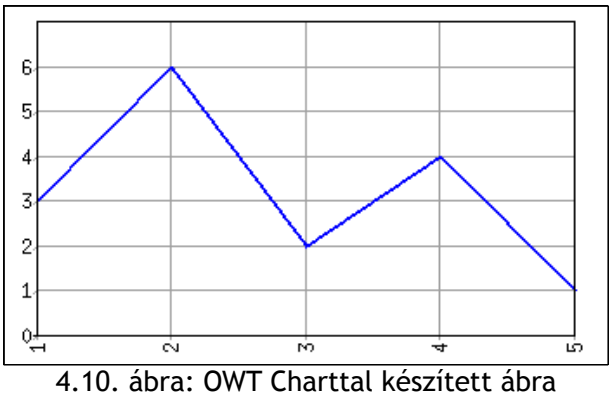
A táblázathoz HTML kódokkal definiálni kell a tábla méretét, amely esetenként eltér. A tábla fejlécének hossza megegyezik a pontok számával, a sorok hossza pedig az első pont adatsorának hosszával (mivel a helyenként hiányzó napi mérési adatok miatt az adatsorok hossza eltérő). A legelső oszlopba a dátumok kerülnek, a többi oszlopba pedig időrendben a mérési adatok.

4.3.5. Az OWT Chart alkalmazása

A szakdolgozat szövegében többször említést tettem a vizuális megoldások erejéről. A mérési eredményeket is akkor lehet igazán szemléletessé tenni, ha pl. grafikonon ábrázoljuk őket. Az OWT Chart egy egyszerűen működő, nyílt forráskódú szerveroldali alkalmazás, amely a bevitt paraméterek segítségével többféle grafikont képest előállítani, majd GIF típusú képfájl formájában küldi el a kliens felé. Mivel az OWT Chart egy CGI típusú alkalmazás, ezért használatához a letöltött végrehajtható (EXE) fájt az Apache szerver cgi-bin könyvtárába kell másolni. A grafikon elkészíthető oly módon is, ha a böngésző címsorába visszük be a kívánt értékeket, azonban a hivatkozást mindig egy URL-ként kell megadni, amely egy vonalas grafikon esetén a következőképpen néz ki:

```
localhost/cgi-bin/owtchart.exe?
Type=Line&W=400&H=200&NumSets=1&NumPts=5&Vals=3!6!2!4!1
```

Ez a kód az oldalt látható 400×200 méretű, 1 darab 5 értékkel rendelkező grafikont rajzolja meg. Ez egy egyszerű vonalas grafikon, mely számos egyéb paraméterrel kiegészíthető. Az extra paraméterek listája az OWT Chart hivatalos segédletében található, azonban számomra csak annyi fog kelleni, hogy a vízszintes (X) tengelyen lévő feliratokat módosítani tudjam a dátumok értékére, illetve hogy több pont esetén különböző színnel jelenjenek meg az egyes adatsorok.



4.10. ábra: OWT Charttal készített ábra

Mivel a **NumSets** paraméter a pontok számától, a **NumPts** pedig a kiválasztott időintervallumban lévő napok számától függ, ezért ezeket az értékeket a szervertől visszakapott JSON objektum *idosor,* illetve *adatok* tömbjeinek hosszával lehet megadni. Az X tengelyen megjelenő dátumokat definiáló **XLabels** paraméter értékeit és a **Vals** után álló mérési eredményeket szintén a JSON megfelelő tömbjeinek elemeivel lehet feltölteni. Ezek a megoldások két, egymásba

ágyazott ciklussal valósulnak meg, mivel az egyes elemek közé mindig elválasztójelnek kell kerülnie. A programsornak ez a része a következőképpen alakul:

```
var v = "Vals=";
var l = "Xlabels=";
for (var j=0 ; j < tabla.idosor.length ; j++) {
     for (var i=0 ; i < tabla.idosor[0].adatok.length ; i++) {
          if (j == 0) {
               l += tabla.idosor[0].adatok[i].datum + ";";
          }
          v += tabla.idosor[j].adatok[i].mozg + "!";
     }
}
p += v.trim("!") + "&" + l.trim(";");
```

Fontos, hogy az URL-ben a paraméter legutolsó értéke után nem állhat elválasztójel, ezért az adatsor végéről a „trim” nevű JavaScript utasítással le kell szedni őket.

Színezés: az alkalmazás csak a HTML színkóddal megadott árnyalatokat tudja értelmezni (színek neveit nem). A kiválasztott 8-féle szín egy JavaScript tömbbe kerül, melyekből mindig annyi elemet sorolunk fel a **SetColors** értékeiként, ahány pont adatsorát ábrázolni szeretnénk (az OWT Chart ugyanis elvárja, hogy pontosan annyi szín legyen definiálva, ahány gráfot tartalmaz a tábla).

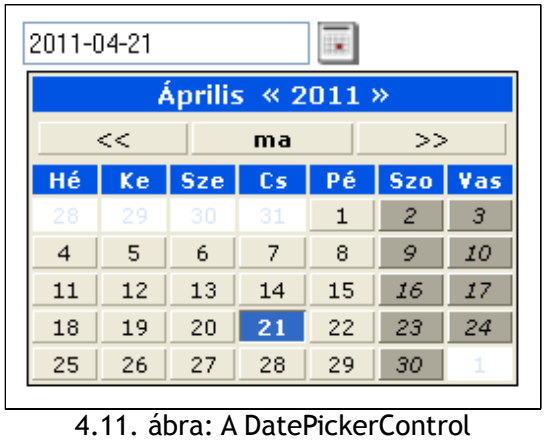
Az eredményt tartalmazó GIF képet a számára már HTML-ben létrehozott helyen helyezem el, mégpedig úgy, hogy az előre definiált 500×250 méretű cella *innerHTML* tulajdonságához hozzárendelem a képet. A kép forrása lesz maga az összeállított OWT Chart URL:

```
document.getElementById("...").innerHTML="<img src=" ... ">";
```

Az URL-be a **Title** paramétert hozzáadva a grafikonnak címet is tudok adni.

4.3.6. Extrák

Naptár beillesztése: azért, hogy megkönnyítsem a lekérdezéshez szükséges dátumok kiválasztását, *interaktív naptárat* csatolok az űrlaphoz. Ez egy kliens oldali JavaScript program, amelyet külső könyvárakból érhetünk el. Számos ingyenesen felhasználható megoldást találhatunk az interneten, melyek közül én a DatePickerControl nevű alkalmazást választom.



4.11. ábra: A DatePickerControl alkalmazása

A szkripteket és ikonokat tartalmazó könyvtárat le kell tölteni és el kell helyezni abban a mappában, amelyben az oldalunk programkódja található. Ezután a program készítőjének utasításait követve könnyedén beilleszthetjük a naptárat a saját oldalunkra. A naptár működéséhez a saját oldalunk programkódjában hivatkozni kell a fent említett könyvtárak tartalmára és egy speciális HTML szövegdobozt kell létrehozni, amely engedélyezi az alkalmazás használatát. A naptárat a szövegmező melletti ikonra kattintva lehet megjeleníteni.

Számos lehetőség kínálkozik arra, hogy testreszabjuk az alkalmazást, pl.: a feliratok lefordíthatóak magyar nyelvre, továbbá lehetőség van a dátum formátumának beállítására, a kiválasztható időintervallum szűkítésére, és számos további paraméter megváltoztatására is.

Képgaléria: a térképi ábrázolás mellett néhány helyszíni felvételt is szeretnék mellékelni a CH épületről. A fényképek megjelenítéséhez többféle lehetőség is kínálkozik, pl. a képeket be lehet illeszteni az oldal tartalmába vagy felugró ablakban is előbukkanhatnak, azonban ezeknél sokkal hatékonyabb és korszerűbb megoldás az ún. *képgaléria készítő* program.

A megvalósításhoz a népszerű Lightbox 2 szkriptet használom. Ezt az alkalmazást sokan kedvelik, mert esztétikus és könnyedén kezelhető. Az egér mellett a billentyűzet utasításaira is reagál, és teljes méretükben élvezhetőek a képek anélkül, hogy az oldal szerkezetét módosítani kellene. Több kép esetén egymás után fűzhetjük őket és album-szerűen végigléptethetünk rajtuk.

A működési elv hasonló a naptáréhoz, azzal a különbséggel, hogy itt egy hivatkozásra kattintva lehet indítani a programot. A külön galériába tartozó képeknek külön csoportnevet kell adni.

4.12. ábra: A Lightbox 2 működése (*forrás:* [LBOX])

4.4. Tesztelés

A HTML és JavaScript programsorok ellenőrzését a Firefox Firebug nevű bővítményével végeztem. Ugyanezt a PHP esetén a NetBeans alkalmazás teszi lehetővé. Ezekkel a programokkal soronként végrehajthatóak az utasítások, nyomon követhetők a változók aktuális értékei, ezáltal könnyebben fedezhetjük fel a programhibákat.

A Google térképen a műholdfelvételhez igazított marker pontok sajnos nem estek egybe az alaprajz poligonján lévő helyükkel, ugyanis a műholdkép és a rajzolt alakzat nincs teljesen fedésben. Mivel alapértelmezett nézetként az úttérkép és a felszerkesztett poligon jelenik meg, célszerűnek láttam a markerek helyét a poligonhoz igazítani. Ehhez ismét elővettem az épületek alaprajzait tartalmazó Shape fájlt és ezúttal a QGIS-szel gyűjtöttem ki a pontok koordinátáit. A pontok adatbázisba való betöltéséhez az egyes pontokhoz szükséges SQL utasításokat egy xy.sql nevű szöveges fájlban tároltam, ezért csak a koordináták kijavítására volt szükség, majd az utasítások újbóli végrehajtására a PostgreSQL PgAdmin3-ban.

A munkám során alkalmazott Google Maps API egy nyilvános szolgáltatás, azaz saját környezetében alkalmazva ingyenesen és korlátlanul felhasználható. Az ilyen jellegű szoftverek mindaddig elérhetőek, ameddig a tulajdonosa meg nem szünteti őket, vagy csak díj ellenében engedélyezi a használatát. Ha ilyen alkalmazásokat veszünk igénybe, számolni kell azzal, hogy

nekünk, mint felhasználóknak egyáltalán nincsen garantálva az, hogy a termék mindig (ingyenesen) elérhető marad.

A Google Maps API miatt a programrendszer internet nélkül nem működik. Ebből az következik, hogy az internetkapcsolatot abban az esetben sem lehet mellőzni, hogyha az alkalmazást egy olyan lokális hálózaton használjuk, ahol a webszerver és az adatbázis-szerver a hálózat része.

A kapott mérési adatsorok sajnos néhol foghíjasak, azaz vannak olyan napok, amelyeken nem mindegyik prizmához tartozik eredmény. Mind a táblázatos mind a grafikus megjelenítésnél problémát okozott, hogy az adat nélküli napok rekordjai teljesen hiányoznak az adatbázisból. Az OWT Chart a függvényváltozóként megadott adatokat (esetünkben: dátum) diszkrét értékként kezeli, vagyis nem értelmezi a két érték közti különbséget, a lépésköz mindenütt egyforma az X tengelyen. A megoldásban ezért azt választottuk, hogy az X tengelyre az elsőként kiválasztott pont adatsorából töltődnek be a dátumok. Ennek a módszernek az az eredménye, hogy ha a többi eredménysor idősora nem azonos az elsővel, akkor elcsúszások vannak az adatsorokban. A hosszú időintervallum és a kis mértékű mozgások miatt ez nem feltűnő, de a feladat precíz megoldására az OWT Chart nem alkalmas.

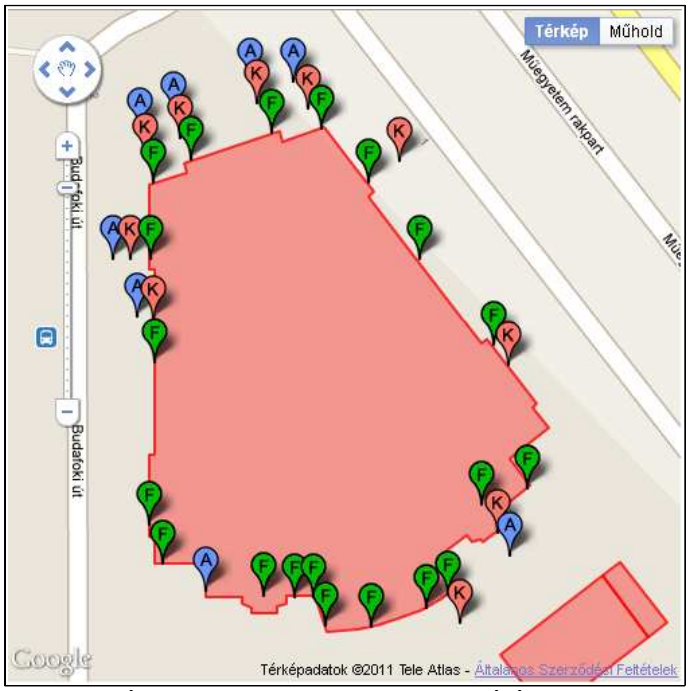
4.5. Felhasználói felület

A programrendszer kezeléséhez a felhasználónak egy böngészőre és internetkapcsolatra van szüksége. Az operációs rendszer szempontjából Windows, LINUX és Mac OS környezetben is futtatható. A megjelenítés ugyanúgy működik hagyományos 4:3 arányú (1600×1200, 1024×768 felbontású) képernyőkön is, mint a 16:9 (1024×576, 1280×720) típusú monitorok esetén. A böngészőt illetően Mozilla Firefox, Windows Internet Explorer vagy Opera 8.0-nál későbbi verzióival kompatibilis. Az URL hívása után a böngészőben egy térkép és egy űrlap jelenik meg. A térképet a Google Maps szolgáltatja, ezért a betöltődés sebessége a Google Maps szerverétől függ. A Google Maps-ben többféle megjelenítési opció közül választhatunk, de esetünkben alapértelmezettként az úthálózati térkép jelenik meg, mely a Budafoki út és a Műegyetem rakpart sarokpontjára van fókuszálva. A Google úthálózati térképén az épületek nem jelennek meg, azonban az alkalmazáshoz a BME néhány épületének alaprajzát megjelenítem. A szolgáltató szerverének sebességétől függően a térképen néhány másodperc elteltével a BME Műegyetem rakpart felőli kampuszának épületei jelennek meg. A programrendszer tárgya a CH épület, ezért alapértelmezettként a térképen a CH épület alaprajzát láthatjuk. A többi épület megtekintéséhez nézetet kell váltani, ami a térkép interaktív tulajdonságának köszönhetően kényelmes. A térképhez három kezelő tartozik:

- mozgatás (bal felső sarok),
- nagyítás (bal oldali skála),
- nézet választása (jobb felső sarok).

A térképen a kezelő nélkül is mozoghatunk, ha az egérrel „megragadjuk” egy tetszőleges pontját. Amennyiben a Műegyetem rakpart mentén mozgunk, vagy csökkentjük a nagyítás mértékét, láthatóvá válik a többi egyetemi épület alaprajza is. A „Műhold” feliratú gombra kattintva az úthálózati nézetről műholdfelvételre válthatunk, a „Térkép” gombbal pedig vissza. Szintén automatikusan töltődő elemek a mozgásvizsgálati prizmákat jelölő markerek. A markerek szín- és

betűkóddal különböztetik meg a prizmák homlokzaton elfoglalt helyét. Ha az egér kurzorral elhaladunk az egyes markerek felett, felbukkan a jelölt prizma neve.



4.13. ábra: A Google Maps felhasználói felülete

A lekérdezéshez szükséges paramétereket a térkép melletti űrlapon állíthatjuk be. Elsőként jelöljük ki a vizsgálati ponto(ka)t és az időintervallumot. A dátumok kiválasztásához segítséget nyújt az interaktív naptár, amelyet a mező melletti ikonnal hívhatunk elő. Az intervallum 2006. 07. 17. és 2011. 02. 22. közötti időszak lehet, bár ezt a naptár korlátozza. A pontneveket tartalmazó listában az egyes elemek betűszínei ugyanazt a logikát követik, mint a markerek esetén. A markerekre rákattintva a listában lévő elemek kijelölhetőek, de természetesen a listából is ki lehet őket választani. Az eredmények lekérdezéséhez ki kell választani a megjelenítés módját.

Egyebek:

- az űrlap alján lévő hivatkozásokat követve néhány helyszíni fényképet tekinthetünk meg az épületről;
- az „Új lekérdezés” gombbal törölhetőek a bevitt adatok és a megjelenített eredmények;
- a „Kezdeti nézet” gombbal visszanavigálhatunk a kiinduló helyzetre a térképen.

5. Összefoglalás

A programrendszer elkészítésének célja egy interaktív, interneten keresztül elérhető kezelőfelület volt, amellyel nagy mennyiségű süllyedésmérési eredményt lehet szemléltetni. A Google Maps API rutinjainak köszönhetően a vizsgálati pontokat és az épület alaprajzát is sikerült – a feladathoz elegendő pontossággal – feltüntetni. A megvalósított alkalmazás bizonyíték arra, hogy ingyenes eszközökkel is profi megoldást lehet kínálni a felhasználó számára. A különböző informatikai fejlesztésekre építve (AJAX, JSON, Google Maps API) mindössze néhány száz soros programmal megvalósítható egy korszerű, felhasználóbarát környezet.

A feladat elkészítése során a modern geodéziai technológiák megismerése mellett nagy adag informatikai tudásra is sikerült szert tennem. A munkához folyamatos inspirációt nyújtott annak a ténynek a felismerése, hogy az informatika (a geodéziához hasonlóan) nagyon széles körben alkalmazható tudomány. Az informatika segítségével nem csak a logikai-, és a vizuális érzék fejleszthető, hanem számos egyéb szakterülettel is megismerkedhetünk és rengeteg hasznos tapasztalatot gyűjthetünk munkánk során. A feladatom elkészítéséhez szükséges alapismeretek megszerzése kiváló lehetőséget nyújtott arra, hogy bepillantást nyerjek egy számomra eddig ismeretlen világba, amely tele van lehetőségekkel, érdeklődőkkel és szakértőkkel, akik segítséget nyújtanak. Az érintett területek (számítástechnikai alapismeretek, térinformatikai módszerek, adatbázis-kezelés, web programozási nyelvek) tanulmányozása során egytől-egyig olyan hasznos tapasztalatokat gyűjtöttem, amelyekből hosszú távon profitálhatok.

Az diplomamunka részeként elkészített programrendszer egyelőre egy alapszintű megoldást kínál, melyet számos egyéb funkcióval lehetne továbbfejleszteni. Néhány ötlet a további munkákhoz:

- Google Maps API helyett saját térképszerverről elérhető műholdfelvétel, ortofotó és/vagy térkép alkalmazása, amelynek nagy előnye, hogy naprakészebb adatokat és jobb felbontású képeket biztosíthatunk, mint a Google; továbbá megszűnik a külső szolgáltatótól való függés is;

- az OWT Chart helyett komplexebb megoldásokra alkalmas grafikonszerkesztő szoftver alkalmazása, pl. gnuplot (szerver oldali) vagy dygraphs (kliens oldali);

- változatosabb mozgásvizsgálati lekérdezések, pl. a 2 cm-nél többet süllyedt pontok neve;

- a süllyedések komplexebb vizsgálatához térinformatikai elemzésre alkalmas szoftverrel izovonalak rajzolása a térképen (pl. a CGI-ként használható *Generic Mapping Tools, GMT*);

- a CYCLOPS rendszer műszerei három dimenzióban mérik a prizmák helyét. Ha hozzáférhetnék a vízszintes adatokhoz is, akkor lehetőség nyílna az egydimenziós süllyedésvizsgálatok mellett deformációk kimutatására is;

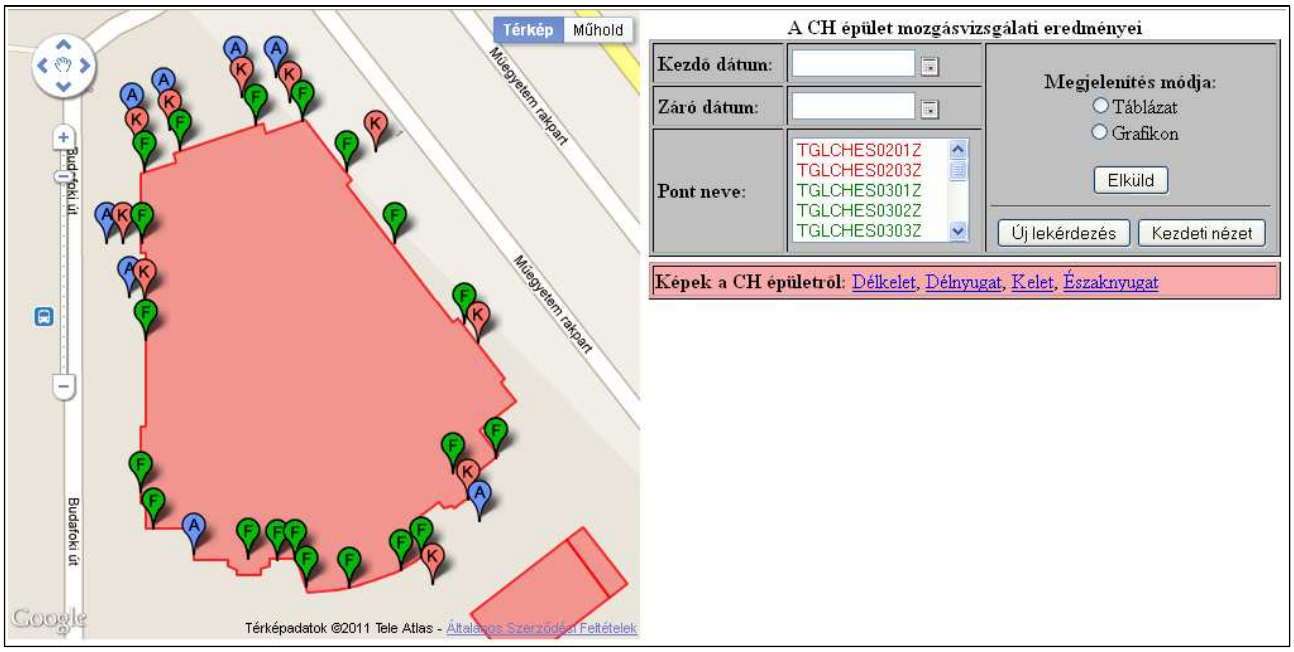
- az adatsorok közti összefüggések vizsgálatához keresztkorreláció számítására alkalmas kiegészítő program írása.

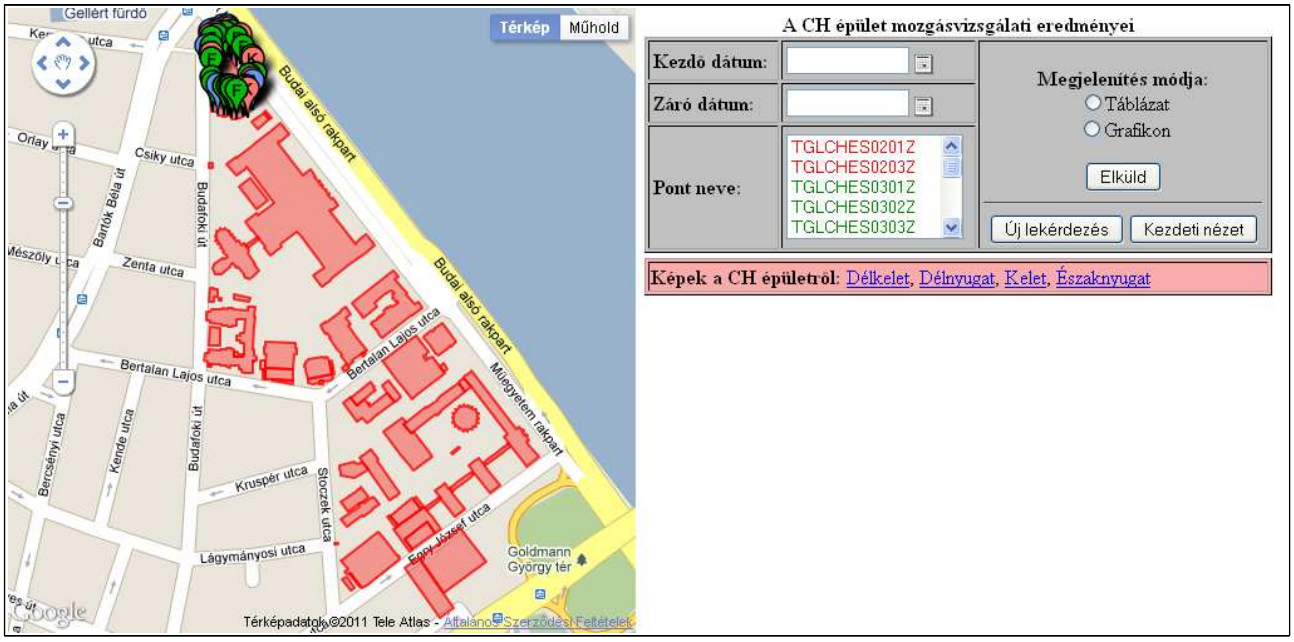
Irodalomjegyzék

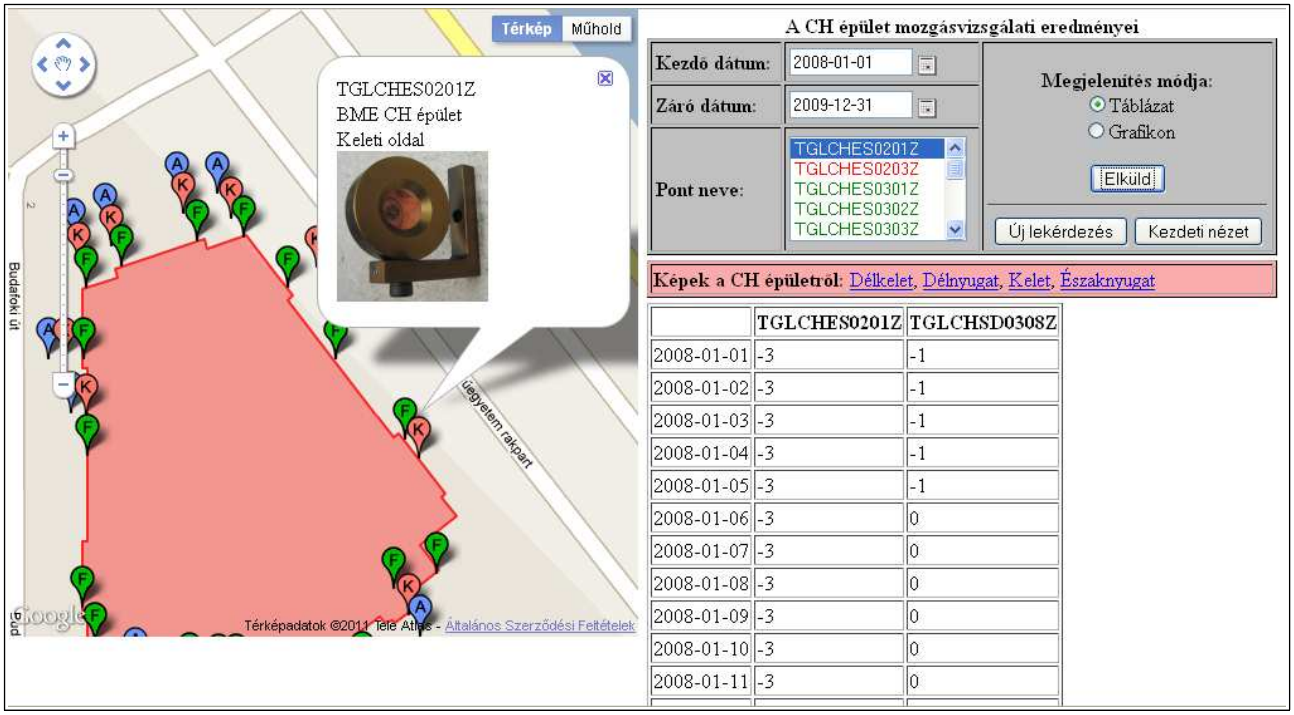
- [ÓDOR] *Dr. Ódor Károly*: **Ipari geodézia I.** (404 oldal, Tankönyvkiadó, 1976)
- [BOUR] *Paul Bourke*: **Cross Correlation** (www.paulbourke.net/miscellaneous/correlate/)
- [KRAU] *Krauter András*: **Geodézia** (Műegyetemi Kiadó, 1995)
- [LEIC] Leica T1800, TC1800, TCA1800 ismertető (6 oldal, Leica Geosystems AG, 1999)
- [SOLD] www.soldata.hu/doc/Cyclops.pdf és www.soldata.hu/doc/Centaur.pdf
- [MTMM] www.mtm-magazin.hu/cikk.PHP ? cikk_id=502&PHPSESSID=b3e0fb4e3ac6390d217b17059fbb9419
- [MITC] *Tyler Mitchell*: **Web Mapping Illustrated** (368 oldal, O'Reilly, 2005)
- [AG05] Nagyméretarányú digitális térképezés órai segédlet (www.agt.bme.hu)
- [AT10] Térinformatika alapjai HEFOP jegyzet (www.fmt.bme.hu/fmt/oktatas/feltoltesek/BMEEOFTAT10/at10segedlet.pdf)
- [DET1] *Detrekői Ákos – Szabó György*: **A helyhez kapcsolódó információk jelentősége** (www.matud.iif.hu/2010/09/04.htm)
- [ASJ1] Térinformatikai elemzések HEFOP jegyzet (www.fmt.bme.hu/fmt/oktatas/feltoltesek/BMEEOFTASJ1/asj1segedlet.pdf)
- [SZ01] *dr. Siki Zoltán*: **Térképek internetes publikálása** (www.agt.bme.hu/maps/im.html)
- [DET2] *Detrekői Ákos*: **Virtuális földgömbök – 3D városmodellek** (www.fomi.hu/honlap/magyar/szaklap/2010/01/2.pdf)
- [GIBS] *Rich Gibson, Schuyler Erle*: **Google Maps Hacks** (337 oldal, O'Reilly, 2006)
- [SZ02] *dr. Siki Zoltán*: **Adatbáziskezelés és szervezés** (www.agt.bme.hu/szakm/adatb/adatb.htm)
- [KLIN] *Kevin Kline*: **SQL in a Nutshell** (700 oldal, O'Reilly, 2004)
- [RIGG] *Simon Riggs, Hannu Krosing*: **PostgreSQL 9 Administration Cookbook** (360 oldal, Packt Publishing, 2010)
- [OBEH] *Regina O. Obe, Leo S. Hsu*: **PostGIS In Action** (425 oldal, Manning Publications Co., 2009)
- [SHEK] *Shashi Shekhar, Hui Xiong*: **Encyclopedia of GIS** (1370 oldal, Springer-Verlag N.Y., 2008)
- [SOLB] *Selena Sol, Gunther Berznieks*: **Instant Web Scripts with CGI/PERL** (809 oldal, M&T Books, 1996)

- [HEIL] *Christian Heilmann*: **Beginning JavaScript with DOM Scripting and Ajax** (512 oldal, Apress, 2006)
- [VALA] *Janet Valade*: **PHP 5 for Dummies** (408 oldal, Wiley, 2004)
- [CRAN] *Dave Crane, Eric Pascarello, Darren James*: **AJAX In Action** (680 oldal, Manning Publications Co., 2006)
- [SVEN] *Gabriel Svennerberg*: **Beginning Google Maps API 3** (328 oldal, Apress, 2010)
- [SZ03] *dr. Siki Zoltán*: **Magyar nyelvű nyíltforrású programok a geoinformatika területén** (www.agt.bme.hu/gis/eloadasok/szolnok2009.pdf)
- [LBOX] www.huddletogether.com/projects/lightbox2/

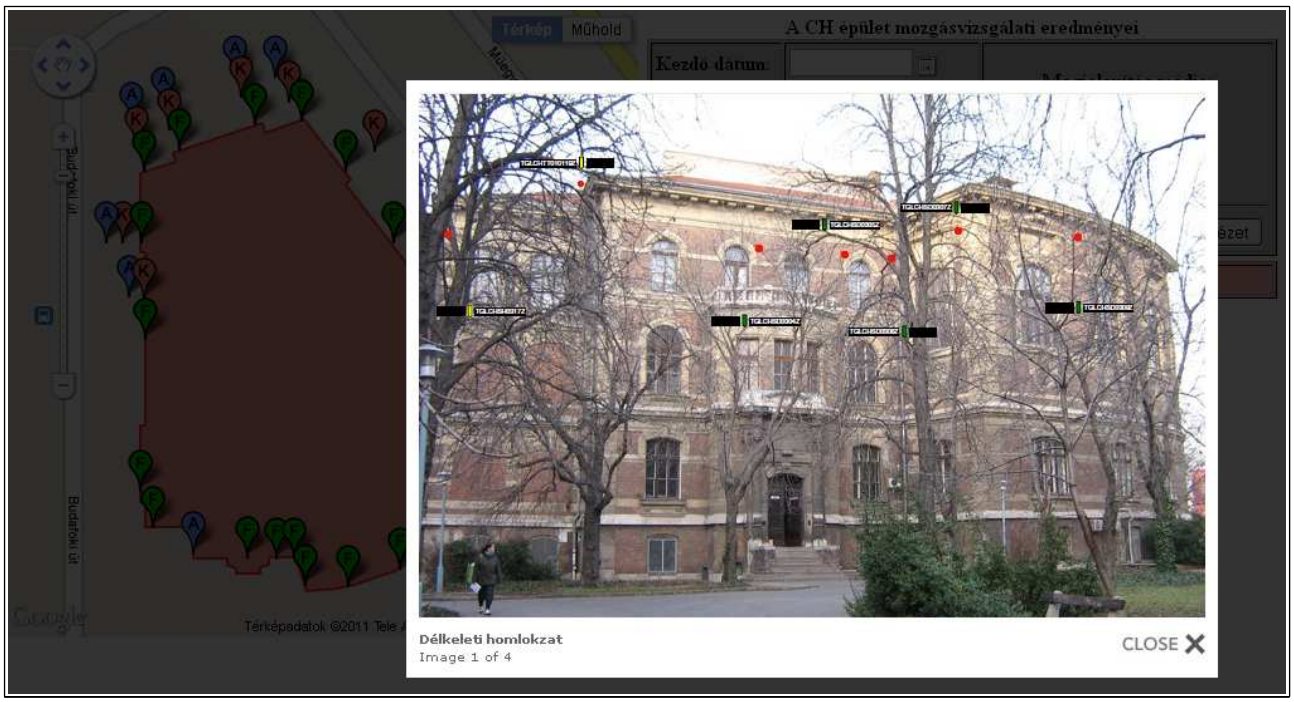
Mellékletek

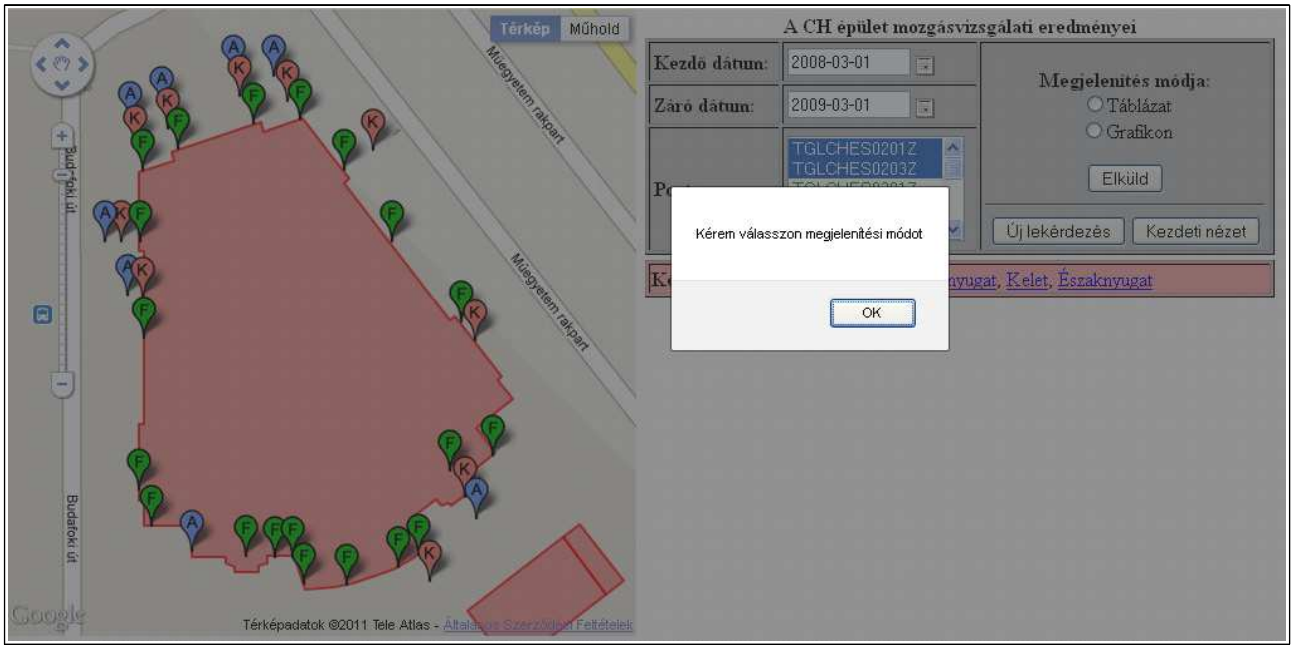

Térkép
Műhold
Műegyetem rakpart
Budafoki út
Térképadatok ©2011 Tele Atlas - Általános Szerződési Feltételek
A CH épület mozgásvizsgálati eredményei
Kezdő dátum:
Záró dátum:
Pont neve:
TGLCHES0201Z
TGLCHES0203Z
TGLCHES0301Z
TGLCHES0302Z
TGLCHES0303Z
Megjelenítés módja:
Táblázat
Grafikon
Elküld
Új lekérdezés
Kezdeti nézet
Képek a CH épületről: Délkelet, Délnyugat, Kelet, Északnyugat


Gellért fürdő
Térkép
Műhold
Budai alsó rakpart
Műegyetem rakpart
Csiky utca
Bartók Béla út
Budafoki út
Zenta utca
Bertalan Lajos utca
Bercsényi utca
Kende utca
Kruspér utca
Stoczek utca
Lágymányosi utca
Egry József utca
Goldmann György tér
Térképadatok ©2011 Tele Atlas - Általános Szerződési Feltételek
A CH épület mozgásvizsgálati eredményei
Kezdő dátum:
Záró dátum:
Pont neve:
TGLCHES0201Z
TGLCHES0203Z
TGLCHES0301Z
TGLCHES0302Z
TGLCHES0303Z
Megjelenítés módja:
Táblázat
Grafikon
Elküld
Új lekérdezés
Kezdeti nézet
Képek a CH épületről: Délkelet, Délnyugat, Kelet, Északnyugat



	TGLCHES0201Z	TGLCHSD0308Z
2008-01-01	-3	-1
2008-01-02	-3	-1
2008-01-03	-3	-1
2008-01-04	-3	-1
2008-01-05	-3	-1
2008-01-06	-3	0
2008-01-07	-3	0
2008-01-08	-3	0
2008-01-09	-3	0
2008-01-10	-3	0
2008-01-11	-3	0


Térkép
Műhold
A CH épület mozgásvizsgálati eredményei
Kezdő dátum:
Budafoki út
Térképadatok ©2011 Tele
Délkeleti homlokzat
Image 1 of 4
CLOSE


Térkép
Műhold
A CH épület mozgásvizsgálati eredményei
Kezdő dátum:
2008-03-01
Záró dátum:
2009-03-01
TGLCHES0201Z
TGLCHES0203Z
Megjelenítés módja:
Táblázat
Grafikon
Elküld
Új lekérdezés
Kezdeti nézet
Kérem válasszon megjelenítési módot
OK
Műegyetem rakpart
Budafoki út
Térképadatok ©2011 Tele Atlas -

```
<!DOCTYPE html>
<html>
        <head>

<meta name="viewport" content="initial-scale=1.0, user-scalable=no" /><!--mobiltelef
onhoz-->
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-2"/>

<title>A CH épület mozgásvizsgálati eredményei (2006/07/17 - 2011/02/22)</title>

<link rel="shortcut icon" href="favicon.ico" type="image/x-icon" /><!--BME logo-->
<link href="http://code.google.com/apis/maps/documentation/javascript/examples/defau
lt.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" href="css/lightbox.css" type="text/css" media="screen" />
<link type="text/css" rel="stylesheet" href="datepickercontrol/datepickercontrol.css
">

<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"
>//API</script>
<script type="text/javascript" src="js/prototype.js"></script>
<script type="text/javascript" src="js/scriptaculous.js?load=effects,builder"></scri
pt>
<script type="text/javascript" src="js/lightbox.js">//képgaléria</script>
<script type="text/javascript" src="datepickercontrol/datepickercontrol.js">//naptár
</script>

<script language="JavaScript" type="text/javascript">

String.prototype.trim = function (c) {
        /* kibõvíti a string objektumot a trim függvénnyel
           levágja az elején és a végén az adott karaktereket */
        var r = new RegExp("^"+c+"+|"+c+"+$");
        return this.replace(r, "");
}

function loopSelected() {
        /* kiszedi a pontneveket a multiselect elembõl
           az idosor php url-hez */
        var sel = '';
        var selobj = document.getElementById("pn");
        for (var i=0 ; i < selobj.options.length ; i++) {
                if (selobj.options[i].selected) {
                        if (sel.length)
                                sel += ',';
                                sel += selobj.options[i].value;
                }
        }
        return sel;
}

function dateCheck(datum) { //datum: formális paraméter
        var r = new RegExp('^20[0-9]{2}-[0-1][0-9]-[0-3][0-9]$');

        if (! r.test(datum)) { //dátumformátum-ellenõrzés
                alert ('Nem megfelelõ dátum ' + datum);
                return false;
        }
        return true;
}

function GetXmlHttpObject() { // AJAX objektum létrehozása
        try {   // Firefox, Opera 8.0+, Safari
                xmlHttp=new XMLHttpRequest();
        }
        catch (e) {     // Internet Explorer
                try {
                        xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
                } catch (e) {
                        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
```

```
                }
        }
        return xmlHttp;
}

function GetPoints() { //pontok lekérése
        var url = "chepulet.php?tipus=pontok";

        if (xmlHttp==null) {
                alert ("Nincs AJAX!");
                return;
        }
        xmlHttp=GetXmlHttpObject();
        xmlHttp.onreadystatechange = StatePChanged;
        xmlHttp.open("GET", url);
        xmlHttp.send(null);
}

function StatePChanged() {
        if (xmlHttp.readyState==4) {
                AddPContent(xmlHttp.responseText);
        }
}

function AddPContent(json) { //markerek hozzáadása a térképhez
        var pontok = eval("(" + json + ")"); //json string objektummá alakítása
        var blueIcon = "blue_MarkerA.png";
        var redIcon = "red_MarkerK.png";
        var greenIcon = "darkgreen_MarkerF.png";
        var cols = ["", "blue", "red", "green"];
        var shadow = new google.maps.MarkerImage('marker-shadow.png',
              new google.maps.Size(37, 32),
              new google.maps.Point(10,0),
              new google.maps.Point(0,32));

        var lsel = document.getElementById("pn");
        while (lsel.length > 1) {
                lsel.remove(lsel.length - 1);
        }

        for (var i=0 ; i < pontok.xy.length ; i++) {
                appendOption(lsel, pontok.xy[i].pontnev, pontok.xy[i].pontnev,
                                cols[pontok.xy[i].pontnev.charAt(8)]);
                                //pontnevek feltöltése az ûrlapra

                var latlng = new google.maps.LatLng(parseFloat(pontok.xy[i].lat),par
seFloat(pontok.xy[i].lng));
                var marker = new google.maps.Marker({ //API rutin markerekhez
                        position: latlng,
                        map: map,
                        shadow: shadow,
                        title: pontok.xy[i].pontnev,
                        icon: blueIcon
                        });

                if (pontok.xy[i].pontnev.charAt(8) == 2) {
                        marker.setIcon(redIcon); //marker ikoncsere
                }
                else if (pontok.xy[i].pontnev.charAt(8) == 3) {
                        marker.setIcon(greenIcon); //marker ikoncsere
                }

                attachInfo(marker, pontok.xy[i].pontnev); //infoablak létrehozása
        }
}

function attachInfo (marker, txt) {
        var wstr = txt + "<br>BME CH épület<br>";
        // infoablak tartalma
```

```
        switch (txt.charAt(5)) {
                case "N": wstr += "Északi oldal<br>";
                          break;
                case "E": wstr += "Keleti oldal<br>";
                          break;
                case "W": wstr += "Nyugati oldal<br>";
                          break;
                case "S": wstr += "Déli oldal<br>";
                          break;
        }
        wstr += "<img src='prizma.png' width=120 height=120>";
        var infowindow = new google.maps.InfoWindow({
                content: wstr,
                position: marker.getPosition()
        });
        google.maps.event.addListener(marker, 'click', function() {
                infowindow.open(map, marker);

                var lsel = document.getElementById("pn");
                for (var i = 0; i < lsel.length; i++) {
                        if (lsel.options[i].value == txt) {
                        lsel.options[i].selected = true;
                        /* kijelöli a táblában azt a pontnevet,
                           amelyiknek a markerére kattintunk */
                        }
                }

        });
}

function GetBuildings() { //épület poligonok lekérése
        xmlBHttp=GetXmlHttpObject();

        if (xmlBHttp==null) {
                alert ("Nincs AJAX!");
                return;
        }
        xmlBHttp.onreadystatechange = StateBChanged;
        var url = "chepulet.php?tipus=epulet";
        xmlBHttp.open("GET", url);
        xmlBHttp.send(null);
}

function StateBChanged() {
        if (xmlBHttp.readyState==4) {
                AddBContent(xmlBHttp.responseText);
        }
}

function AddBContent(json) { //poligonok hozzáadása a térképhez
        var buildings = eval("(" + json + ")"); //json string objektummá alakítása

        for (var j=0 ; j < buildings.epuletek.length ; j++) {
                var bCoords = new Array(); //üres tömb;

                for (var i=0 ; i < buildings.epuletek[j].pontok.length ; i++) {
                        bCoords[i] = new google.maps.LatLng(
                                parseFloat(buildings.epuletek[j].pontok[i].lat),
                                parseFloat(buildings.epuletek[j].pontok[i].lng));
                }

                var building = new google.maps.Polygon({ //API rutin poligonhoz
                        paths: bCoords,
                        strokeColor: "#FF0000",
                        strokeOpacity: 0.8,
                        strokeWeight: 2,
                        fillColor: "#FF0000",
                        fillOpacity: 0.35
                });
```

```
                building.setMap(map);
        }
}

function GetTable() { //mérési eredmények lekérése
        xmlTHttp=GetXmlHttpObject();

        if (xmlTHttp==null) {
                alert ('Nincs AJAX!');
                return;
        }
        xmlTHttp.onreadystatechange = StateTChanged;
        var pontnev = loopSelected();

        if (pontnev.length == 0) {
                //pontnév ellenõrzés
                alert ('Nincs pontnév megadva');
                return;
        }
        var kezdo = document.getElementById("kez").value;

        if (! dateCheck(kezdo)) {
                //kezdõdátum ellenõrzés
                return;
        }
        var zaro = document.getElementById("zar").value;

        if (! dateCheck(zaro)) {
                //záródátum ellenõrzés
                return;
        }
        var url="chepulet.php?tipus=idosor&pontnevek=" + pontnev + "&kezdo_datum=" +
 kezdo + "&zaro_datum=" + zaro;
        xmlTHttp.open("GET",url);
        xmlTHttp.send(null);
}

function StateTChanged() {
        if (xmlTHttp.readyState==4) {
                AddTContent(xmlTHttp.responseText);
        }
}

function AddTContent(json) { //tábla vagy grafikon hozzáadása
        var tabla = eval ("(" + json + ")"); //json string objektummá alakítása
        if (document.getElementById("tab").checked) {
                //tábla
                var p = "<table border='1'><tr><th></th>";
                for (var j=0 ; j < tabla.idosor.length ; j++) {
                        p += "<th>" + tabla.idosor[j].pontnev + "</th>";
                }
                p += "</tr><center>";

                for (var i=0 ; i < tabla.idosor[0].adatok.length ; i++) {
                        p += "<tr><td>" + tabla.idosor[0].adatok[i].datum + "</td>";
                        for (var j=0 ; j < tabla.idosor.length ; j++) {
                                p += "<td>" + tabla.idosor[j].adatok[i].mozg + "</td
>";
                        }
                        p += "</tr></center>";
                }
                p += "</table>";
                document.getElementById("tbl").innerHTML=p;
        }
        else if (document.getElementById("gra").checked) {
                //grafikon
                var p = "/cgi-bin/owtchart.exe?Type=Line&W=500&H=300&NumSets=" +
                        tabla.idosor.length + "&NumPts=" +
                        tabla.idosor[0].adatok.length + "&";
```

```
                //jelmagyarázat a görbékre

                var v = "Vals=";
                var l = "XLabels=";
                for (var j=0 ; j < tabla.idosor.length ; j++) {
                        for (var i=0 ; i < tabla.idosor[0].adatok.length ; i++) {
                                if (j == 0) {
                                        l += tabla.idosor[0].adatok[i].datum + ";";
                                }
                                v += tabla.idosor[j].adatok[i].mozg + "!";
                        }
                }
                p += v.trim("!") + "&" + l.trim(";");

                var c = "&SetColors=";
                var clr = new Array();
                clr[0]="8000FF";
                clr[1]="01DFD7";
                clr[2]="FF8000";
                clr[3]="01DF01";
                clr[4]="0000FF";
                clr[5]="FF0000";
                clr[6]="FF00FF";
                clr[7]="61380B";
                for (var k=0 ; k < tabla.idosor.length ; k++) {
                        //színezi a grafikonokat
                        c += clr[k] + "!";
                }
                c = c.trim("!");
                p += c;

                var leg = "";
                for (var l=0 ; l < tabla.idosor.length ; l++) {
                        //kiírja, hogy a megjelenített grafikonok melyik pontokhoz t
artoznak
                        leg += "<font size='2' color=" + clr[l] + ">" + tabla.idosor
[l].pontnev + "</font>, ";
                }
                leg = leg.trim(", ");

                document.getElementById("tbl").innerHTML="<b>" + leg + "</b><br><img
 src=" + p + ">";
        }
        else {
                alert ("Kérem válasszon megjelenítési módot");
        }
}

function appendOption(sel, val, txt, col) {
// html (multi)select-hez hozzátesz egy elemet
                //sel: html select objektum
                //val: az elem értéke
                //txt: a kiírt szöveg
                //col: elem színe
        var newOpt = document.createElement('option');
        newOpt.text = txt;
        newOpt.value = val;
        newOpt.style.color = col;
        try {
                sel.add(newOpt, null);
        } catch(ex) {
                sel.add(newOpt);        // Internet Explorer
        }
}



function initialize() { //térkép megjelenítése
```

```
        myLatlng = new google.maps.LatLng(47.482838,19.054386);
        zoomLevel = 19;
        var myOptions = {
                zoom: zoomLevel,
                center: myLatlng,
                streetViewControl: false,
                rotateControl: false,
                mapTypeId: google.maps.MapTypeId.ROADMAP
        }
        map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
        map.setTilt(0);

        GetBuildings(); //épület poligonok AJAX
        GetPoints(); //marker pontok AJAX
}

function startZoom() {
        map.setCenter(myLatlng); //visszaállítja a térkép kezdeti középpontját
        map.setZoom(zoomLevel); //visszaállítja a térkép kezdeti nagyítását
}

function ClearTable() {
        document.getElementById("tbl").innerHTML = ""; //törli a táblázatot vagy gra
fikont
        document.form1.reset(); //törli az ûrlapba bevitt adatokat

}

</script>

</head>

<body onload="initialize()">
        <!-- Hungarian -->
        <input type="hidden" id="DPC_TODAY_TEXT" value="ma">
        <input type="hidden" id="DPC_BUTTON_TITLE" value="Naptár nyitása...">
        <input type="hidden" id="DPC_MONTH_NAMES" value="['Január', 'Február', 'Márc
ius', 'Április', 'Május', 'Június', 'Július', 'Augusztus', 'Szeptember', 'Október',
'November', 'December']">
        <input type="hidden" id="DPC_DAY_NAMES" value="['Vas', 'Hé', 'Ke', 'Sze', 'C
s', 'Pé', 'Szo']">
        <input type="hidden" id="DPC_FIRST_WEEK_DAY" value="1">

<table border="0">

        <tr><td width="500" rowspan="3"><div id="map_canvas" style="width:500px; hei
ght:500px"></div></td>
        <td>
        <form name="form1">
                <table border='1' width='500' bgcolor = 'silver' cellpadding='3'>
                        <capture><center><b>A CH épület mozgásvizsgálati eredményei<
/b></center></capture>
                        <tr><td><b>Kezdõ dátum: </b></td>
                        <td><input type="text" id="kez" name="kez" size="12" value="
" datepicker="true" datepicker_format="YYYY-MM-DD"><!--datepicker_min="2006-07-17" d
atepicker_max="2011-02-03"--></td>
                        <td rowspan='3'><center><p><b>Megjelenítés módja:</b><br>
                        <input type="radio" name="rad" id="tab" value="tab">Táblázat
<br>
                        <input type="radio" name="rad" id="gra" value="gra">Grafikon
<br></p>
                        <input type="button" id="but" value="Elküld" onClick="GetTab
le()"><hr>
                        <input type="button" id="res" value="Új lekérdezés" onClick=
"ClearTable()">
                        <input type="button" value="Kezdeti nézet" onClick="startZoo
m()">
        </td></tr>
        <tr><td><b>Záró dátum: </b></td>
```

```
                        <td><input type="text" id="zar" name="zar" size="12" value="
" datepicker="true" datepicker_format="YYYY-MM-DD"><!--datepicker_min="2006-07-17" d
atepicker_max="2011-02-03"--></td></tr>
                <tr><td><b>Pont neve: </b></td>
                        <td><select name="pn" id="pn" multiple size="5"></select></t
d>
                </tr>
</table>
        </form>
</td></tr>
<tr><td>
        <table border='1' width='500' bgcolor = 'FAABAB' cellpadding='1'>
                <tr><td>
                                        <b>Képek a CH épületrõl: </b>
                <a href="images/dk.png" rel="lightbox[gallery]" title="Délkeleti hom
lokzat">Délkelet</a>,
                <a href="images/dny.png" rel="lightbox[gallery]" title="Délnyugati h
omlokzat">Délnyugat</a>,
                <a href="images/k.png" rel="lightbox[gallery]" title="Keleti homlokz
at">Kelet</a>,
                <a href="images/eny.png" rel="lightbox[gallery]" title="Északnyugati
 homlokzat">Északnyugat</a>
                </td></tr>
        </table>
        </td></tr>
        <tr><td><div id="tbl" style="width: 500px; height: 250px"></div>
        </td></tr>
        </table>
</body>

</html>
```

```php
<?php
/* paraméterek:
 * tipus=pontok (prizma vízszintes koordinátái)
 * tipus=epulet (épület poligon)
 * tipus=idosor (süllyedés egy idõintervallumban)
 *      pontnevek
 *      kezdo_datum
 *      zaro_datum
 */


include_once("config.php"); //külsõ konfigurációs file
$conn=@pg_connect ("host=$host dbname=$dbname user=$user password=$pass");
//kapcsolódás az adatbázishoz
if (! $conn)
        //hibajelzés ha nem sikerült a kapcsolódás
        die ("Nem sikerült a csatlakozás az adatbázishoz.");

switch ($_REQUEST["tipus"]) {

case "pontok" : $sql="select pontnev,ST_X(geom) as lng,ST_Y(geom) as lat FROM xy ord
er by pontnev";
        $resource = @pg_query ($sql); // az sql utasítás végrehajtása
        if (! $resource) die ("Hiba");
        $buf = "";
        $buf .= "{'xy':\n["; //json eleje
        while (($r=pg_fetch_array($resource,NULL,PGSQL_ASSOC))) {
                /* átvesz 1 sort a lekérdezés eredményébõl
                 * a null a következõt veszi
                 * pgsql assoc: az oszlopnév az index, az elem értéke pedig az érték
 */
                $buf .= "{'pontnev': '" . $r["pontnev"] . "' , 'lng': '" . $r["lng"]
 . "' , 'lat': '" . $r["lat"] . "'},\n";
        }
        echo trim($buf, ",\n");
        echo "\n]\n}"; //json vége
        break;

case "epulet" : $sql="select nev, ST_AsText(geom) as g, gid, ST_NPoints(geom) as n f
rom epulet";
        //geometria: g ; pontok száma: n
        $resource=@pg_query ($sql); // az sql utasítás végrehajtása
        if (! $resource) die ("Hiba");
        $buf = "";
        $buf .= "{'epuletek':\n["; //json eleje
        while (($r=pg_fetch_array($resource,NULL,PGSQL_ASSOC))) {
                /* átvesz 1 sort a lekérdezés eredményébõl
                 * a null a következõt veszi
                 * pgsql assoc: az oszlopnév az index, az elem értéke pedig az érték
                 */
                $p = preg_split('/,/',trim(substr($r["g"], 7), "()"));
                /* leszedi a nyitó és záró zárójeleket
                 * és az elejérõl a polygon szót is
                 * majd tömbbe teszi a vesszõvel elválasztott tagokat
                 */
                $buf .= "{'nev': '" . $r["nev"] . "' , 'pontok': [";
                for ($i=0; $i < $r["n"]; $i++) {
                        $pi = preg_split('/ /', $p[$i]);
                        //WKT-ból szóközzel elválasztva
                        $buf .= "{'lng': '" . $pi[0] . "' , 'lat': '" . $pi[1] . "'}
,\n";
                }
                $buf = trim($buf, ",\n");
                $buf .= "\n]\n},";
        }
        echo trim($buf, ",\n");
        echo "\n]\n}"; //json vége
        break;
```

```
case "idosor": $pontnevek = preg_split('/,/', $_REQUEST["pontnevek"]);
        $buf = "";
        $buf .= "{'idosor':\n["; //json eleje
        for ($i=0 ; $i < count($pontnevek) ; $i++) {
                $sql="select pontnev,datum, mozg from meresek where pontnev='" . $po
ntnevek[$i] . "' and datum between '" . $_REQUEST["kezdo_datum"] . "' and '" . $_REQ
UEST["zaro_datum"] . "' order by datum"; //string-et keres az sql nem oszlopot
                $resource = @pg_query($sql);
                if (! $resource) die ("Hiba");
                $buf .= "{'pontnev': '" . $pontnevek[$i] . "' , 'adatok': [\n";
                while (($r=pg_fetch_array($resource,NULL,PGSQL_ASSOC))) {
                /* átvesz 1 sort a lekérdezés eredményébõl
                 * a null a következõt veszi
                 * pgsql assoc: az oszlopnév az index, az elem értéke pedig az érték
                 */
                        $buf .= "{'datum': '" . $r["datum"] . "' , 'mozg': '" . $r["
mozg"] . "'},\n";
                }
                $buf = trim($buf, ",\n");
                $buf .= "\n]\n},\n";
        }
        $buf = trim($buf, ",\n");
        $buf .= "\n]\n}"; //json vége
        echo $buf;
        break;

default: die ("Nem megfelelõ paraméterezés");
}

pg_close();
?>
```