Eötvös Loránd Tudományegyetem
Informatikai Kar
Térképtudományi és Geoinformatikai Tanszék

Simó Benedek

Földtani adatok kartografált, interaktív megjelenítése a weben open-source eszközök segítségével

doktori disszertáció

Témavezető: Dr. Elek István, egyetemi docens

ELTE TTK Földtudományi Doktori Iskola
Iskolavezető: Dr. Nemes-Nagy József
Térképészet-Geoinformatika Doktori Program
Programvezető: Dr. Zentai László

Budapest, 2017

Tartalomjegyzék

Bevezetés

A térbeli vonatkozású földtudományi adatok bemutatására hosszú évtizedeken keresztül papírtérképeket használtak. A PC-k megjelenése után azonban hamar megjelentek az első digitális technológiával készült térképek, és egyre nagyobb teret nyert az adatbázis alapú feldolgozás, illetve térképkészítés. A 2000-es évekre a térinformatikai eszközök fejlődése, és ezzel párhuzamosan az internet robbanásszerű elterjedése hamarosan lehetőséget teremtett ezeknek az adatoknak a weben való publikálására, mely napjainkra kiemelt fontosságúvá vált az egyes adatgazdák körében.

A Magyar Állami Földtani Intézet (MÁFI) a ’90-es évektől kezdve aktív szerepet vállalt a magyarországi térinformatika kialakulásában és megszilárdulásában, de emellett megtartotta az Intézetre jellemző magas színvonalú tematikus kartográfiai színvonalat is. A MÁFI 2005-ben tette közzé első webes térképi alkalmazását, mely az Egységes Országos Földtani Térképsorozat 1:100 000-es méretarányú térképlapjainak böngészését tette lehetővé (Havas, 2009). Ez az első alkalmazás még a Geomedia szoftver segítségével készült, majd egy szoftverváltást követően az adatok szolgáltatása ezután már ESRI környezetben történt. A 2012-ben Magyar Földtani és Geofizikai Intézet (MFGI) néven tovább működő kutató központ napjainkig ezt a technológiát használja elsődlegesen a földtani térképeinek webes, interaktív publikációjára.

A hardveres eszközök folyamatos fejlődésével és a nyílt forráskódú (open-source) szoftverfejlesztési modell elterjedésével a 2000-es évek közepétől új térinformatikai eszközök jelentek meg a piacon. Ezek az eszközök mára igen jelentős szerepet töltenek be a modern térinformatikában, mivel ingyenességük mellett szabadon testre szabhatóak az adott igényeknek megfelelően.

Doktori kutatásomban ezeknek az eszközöknek a földtani adatokra való alkalmazhatóságát kívántam megvizsgálni a környezet kialakításától a lehetséges migrációs lépéseken át egészen a webkartográfiai megjelenítéséig. Egy ilyen komplex rendszer felépítéséhez számos komponens áll rendelkezésre, és a kívánt céltól függően többféle összeállítás is lehet eredményes, így kutatásomban nemcsak egy általános rendszer kialakítását vizsgáltam, hanem figyelembe vettem a földtani adatok jellegzetességeit is.

Ennek megfelelően a rendszer kialakításakor és vizsgálatakor elsősorban a vektoros adattárolással létrejött térinformatikai adatok szolgáltatására helyeztem a hangsúlyt, így értekezésemben elsősorban az ilyen típusú adatokra épülő adatszolgáltatások részletes vizsgálatára koncentráltam.

A dolgozat első fejezetében azt kívánom bemutatni, hogy milyen alkotóelemekből épülhet fel egy általános webes térinformatikai rendszer (WebGIS rendszer) az adatbázistól a szerver oldali komponenseken át a kliens oldali megjelenítésig.

Ezt követően az MFGI jelenlegi térinformatikai hátteréről, a térképszerkesztés folyamatáról, a használt tematikus kartográfiai módszerekről, valamint a földtani térképek jellegzetességeiről és csoportosítási lehetőségeiről adok leírást.

A kutatási háttér vázolása után a harmadik fejezetben az általam kialakított párhuzamos térképszolgáltatási környezeteket taglalom hardveres, hálózati és adathátter szempontjából, és betekintést nyújtok a rendszerek tervezésekor figyelembe vett alapelvekről, valamint az adattárolási háttér kapcsán részletesen megvizsgálom a szoftverfüggetlen térinformatikai adattárolás lehetőségét.

A nyílt forráskódú rendszer vizsgálata előtti fontos lépésként a zárt forráskódú, kereskedelmi térinformatikai eszközökre épülő rendszert elemzem. Ez viszonyítási alapjául szolgált az új rendszer kialakításának, ezért megvizsgáltam az általános jellemzőit, funkciókészletét, és az egyes elemek feladatait.

Az utolsó fejezetben a kialakítás alkalmas eszközeit, és a rendszerrel szemben támasztott igényeket, és azok kiszolgálását veszem számba. A fejezetben tárgyalom az adatok, valamint a jelkulcsi elemek migrációjára vonatkozó eljárásokat. Végül összefoglalom a kialakított architektúrák előnyeit és hátrányait.

I. Egy webes térinformatikai rendszer építőkövei

A térinformatikai adatok weben való interaktív, kartografált megjelenítéséhez használt rendszerek alapvetően hármas tagozódásúak. Az architektúra első eleme az adattárolásért felelős; ez történhet fájl alapú megoldásokkal, de jelenthet egy olyan adatbázis kezelőt is, mely alkalmas térbeli adatok kezelésére. A rendszer második eleme az adatok szolgáltatására hivatott; ez egy szerver oldali komponens, melynek segítségével az adatok elérhetővé válnak a web irányából, általában valamilyen webszerviz formában. A struktúra harmadik építőeleme az interaktív megjelenésért felelős; ez a böngészőben futó kliens oldali komponens.

1. ábra: A WebGIS rendszerek felépítése

A fejezetben szeretném bemutatni egy általános WebGIS rendszer építőköveit, ezeknek a típusait, jellegzetességeit, előnyeit és hátrányait, korlátait; kitérve az egymással való közös működés lehetőségeire is.

I.1. A WebGIS rendszerek alapja – az adattárolás

A vektoros térinformatikai adatok webre való szolgáltatását alapvetően meghatározza az, hogy milyen módon történik az adatok tárolása, illetve továbbítása a weben keresztül. Ennek a formája lehet fájl alapú, melyen belül megkülönböztethetőek a szöveges, illetve bináris tárolási mechanizmusok, de a kliens-szerver struktúrában működő relációs adatbázis alapú megoldások talán jobban elterjedtek a WebGIS világában. Az

utóbbi években egyre szélesebb körben alkalmazott fájl alapú relációs adatbázisok átmenetet képeznek az egyszerű, fájl alapú, illetve a kliens-szerver struktúrában működő relációs adatbázisok között, így ezeknek a vizsgálata is indokolt lehet.

A disszertációban elsősorban azok a formátumok és adattárolási mechanizmusok kerülnek tárgyalásra, melyek egy WebGIS architektúra megalkotásakor fontos szerepet játszanak, akár az adattárolás, akár az adattovábbítás szempontjából. Ahogyan az a bevezetőben is említésre került, az értekezés elsősorban a vektoros adattípusokra koncentrál, hiszen a földtani adatok esetében ez a térinformatikai adattípus a leginkább jellemző.

I.1.1. Adattárolás fájl alapon

A térinformatikai adatok tárolásának egyik legegyszerűbb és legrégebbi módja a fájl alapú adattárolás, mely történhet szöveges, vagy bináris formában. (Természetesen a "szöveges" adattárolás is alapvetően bináris formában történik, csak egy adott karakterkódolás alapján egyes szövegszerkesztők képesek megjeleníteni az adattartalmat karakteres formában is.) Ennek az adattárolási formának gyakorlatilag végtelen számú variációja létezhet, hiszen amennyiben a fájl tartalmazza az egyes objektumok térbeli pozícióját, illetve tulajdonságait valamilyen strukturált formában, úgy az eleget tesz a vektoros adatmodell elvárásainak, és így felhasználható további feldolgozáshoz, megjelenítéshez vagy adattovábbításhoz. Az idő folyamán mégis létrejöttek olyan szabványossá váló formátumok, amelyek kiemelkedtek a többi közül egy adott szempontnak, vagy felhasználási célnak megfelelően, illetve amelyek széles körben elterjedtek. A fejezetben elsősorban ezeknek a formátumoknak a tárgyalására kerül sor.

I.1.1.1. Szöveges adatformátumok

A GIS világban előforduló szöveges formátumok egyik legnagyobb előnye, hogy az adatokhoz egy egyszerű szövegszerkesztővel is hozzá lehet férni, így annak olvasása vagy módosítása nem jelenthet problémát. A másik nagy előnye, hogy mivel tetszőleges karakter-kódolás állítható be minden egyes fájlhoz, így a speciális karakterek tárolása minden környezetben megoldható, ezért gyakorlatilag operációs rendszer, illetve

szoftverfüggetlenek, és így alkalmazásuk kifejezetten előnyös adatcserével járó folyamatokhoz.

A szöveges adatformátumok közül webes szempontból kiemelkedik a JSON specifikációra épülő GeoJSON formátum, valamint az XML alapokon nyugvó GML formátum. Mindkettő egy fájlban tartalmazza az egyes objektumok geometriai, illetve attribútum adatait, és az egyes elemeket objektum struktúra szerint tárolja el (a GeoJSON formátum egyes elemei például egy lépésben alakíthatóak Javascript objektumokká).

A GeoJSON formátum a Javascript Object Notation (JSON) specifikációján alapszik, és alkalmas pont, vonal, illetve felületek tárolására, de akár egy típusú, több elemből álló elemeket is képes kezelni (multipart, vagyis MultiPoint, MultiLineString, valamint MultiPolygon típusú objektumokat). Az egyik legújabb GIS formátumnak számít, a specifikáció kidolgozása 2007 márciusában kezdődött meg. (Butler et al. 2016) Érdekessége, hogy a formátumra vonatkozó szabványosításért nem valamilyen szervezet, hanem webes fejlesztők munkacsoportja felelős.

A GeoJSON szabvány továbbfejlesztett változata a TopoJSON, mely alkalmas szöveges formában topológiai adattárolásra. A megszokott spagetti tárolás helyett ez a formátum először eltárolja valamennyi egyedülálló törésvonal koordinátáit, majd az egyes objektumokhoz már csak ezeket rendeli hozzá. Ezzel egy két polygon elemet határoló törésvonal koordinátái csak egyszer kerülnek letárolásra, így ez az adattárolás igen kedvező méret-tulajdonságokkal rendelkezik. (Bostock et al. 2016)

A GML (Geography Markup Language) adatformátum az XML (Extensible Markup Language) leíró nyelv kiterjesztése, és alkalmas térinformatikai adatstruktúrák letárolására. Képes pont, vonal és felület struktúrák, illetve a hozzájuk kapcsolódó attribútum adatok tárolására is. A formátumot az Open Geospatial Consortium (OGC) definiálta, és jelenleg a 3-as verziónál jár. (Raimundo et al. 2008)

A szöveges adatformátumokról összefoglalóan megállapítható, hogy már a megalkotásuknál alapvetően az adatcsere folyamatok elősegítése volt a cél, és ezzel együtt kijelenthető, hogy mint klasszikus adattárolási formátum nem előnyös alkalmazásuk a webes adatszolgáltatások adattárolási funkciójára, hiszen a szerver oldali komponensek "gépi kódja" struktúrájukból fakadóan lassabban fér hozzá az adattartalomhoz.

I.1.1.2. Egyszerű bináris adatformátumok

A bináris adatformátumok a szöveges adatformátumokkal szemben elsősorban olyan szempontok figyelembe vételével kerültek kialakításra, amelynél a programok adathozzáférési módja és ideje elsődleges szempont volt. Ennek megfelelően a WebGIS világban is előnyt élveznek az egyszerű szöveges adattárolással szemben.

A bináris formátumok talán legjelentősebb képviselője az ESRI (Environmental Systems Research Institute) cég által kidolgozott shapefile formátum, mely napjainkra szinte szabvánnyá nőtte ki magát a térinformatika világában. Nemcsak az ESRI termékek használják, hanem más kereskedelmi szoftverek és a nyílt forráskódú világ szinte valamennyi térinformatikai szoftvere ismeri, így egy kedvelt formátum a különféle rendszerek közötti adatcsere folyamatok során. (Hall 2008)

Több, bináris fájlt használ egy objektumcsoport leírására; a shp kiterjesztésű fájl tartalmazza az egyes objektumok koordináta adatait, a dbf kiterjesztésű fájl tartalmazza az egyes objektumok attribútumait, valamint az shx fájl tartalmazza a shp fájlban található egyes elemek pozícióját; ez gyakorlatilag egy index fájl. (Zhu 2011) Ehhez a három elemhez további leíró fájlok kapcsolódhatnak; ezek közül talán a legfontosabb a prj kiterjesztésű fájl, mely tartalmazza az adott objektumcsoportot leíró shapefile vetületi jellemzőit. Ez a formátum sokáig kizárólagosan az ESRI cég termékeiben volt elérhető, és zárt forráskóddal rendelkezett, de az 1998-ben kiadott részletes specifikációja alapján mára az open-source világban is szabványos adatcsere formátumként használják (bár a shapefile-ok részeként megjelenő egyes kapcsolt elemek forráskódja a mai napig nem elérhető (Huber et al. 2012)). A shapefile-nak természetesen vannak korlátai, amelyek részint a formátum jellegéből, másrészt pedig magából a specifikációból fakadnak. Az egyik legjelentősebb probléma a formátummal, hogy a 2GB jelenti a formátum felső határát, de sokszor gondot okoz a mezőnevek 10 karakterben rögzített maximális értéke, az Unicode formátumú mezőnevek és értékek támogatásának hiánya, a null értékek kezelésének a következetlen kezelése, de egyes esetekben gondot okozhat a 255-ben maximált mezőnevek száma, vagy a szűkös adattípus támogatás is. (ESRI 1998)

A shapefile formátum mellett még számos, általában zárt forráskódú szoftver rendelkezik még saját önálló bináris formátumokkal, de ezek nem játszanak fontos

szerepet a WebGIS szempontjából, hiszen a kereskedelmi szoftverek közül a webes térképi adatszolgáltatásban kiemelkedő ArcGIS Server, valamint az open-source környezetben alkalmazott hasonló eszközök is a shapefile formátumot támogatják, éppen ezért nem is kerülnek részletezésre ebben a fejezetben.

Összességében tehát megállapítható, hogy a WebGIS szempontjából a bináris, nem adatbázis alapú adattárolási mechanizmusok közül a shapefile formátum játssza a legkiemelkedőbb szerepet, de ismert korlátai miatt alkalmazása számos esetben kedvezőtlen lehet.

I.1.1.3. Adattárolás fájl alapú adatbázisokban

A fájl alapú adattárolási struktúrák mellett a modern GIS adattárolás szempontjából kiemelkedő fontosságúak a szintén fájl alapú, de objektum-orientált, illetve objektum relációs adatmodell szerinti tárolást alkalmazó adattárolási megoldások. Ahogyan a nevük is sugallja, ezek már sokkal magasabb szintű adattárolási-adatrendszerezési képességekkel rendelkeznek, és nagy előnyük, hogy amellett, hogy hordozzák az egyszerű fájl alapú tárolás előnyeit (hordozhatóak, egyszerűen, akár hálózati kapcsolat nélkül is használhatóak), ugyanakkor képesek akár komplexebb adatszerkezetek többrétű adattárolásra is. Segítségükkel akár teljes térképek adatállományai leképezhetőek, de az egyes rétegekhez tartozó klasszikus attribútum tábla mellett akár 1:N típusú adattárolás is megvalósítható bennük kapcsolómezők segítségével. (Nasser 2014) Komplex felhasználhatóságának köszönhetően a modern tériformatikában mára jelentőségük egyre felértékelődik, és használatuk egyre elterjedtebb, hiszen a klasszikus kliens-szerver struktúrában működő relációs adatbázis kezelők több előnyös tulajdonságát is hordozzák, mégsem igényelnek komolyabb telepítést vagy jogosultsági beállításokat. (Tezer 2014)

A fájl alapú térbeli adatbázisok közül két megoldás is az ESRI céghez köthető, az egyik a Microsoft Access adatbázisra épülő Personal Geodatabase, a másik pedig a cég által kifejlesztett File Geodatabase formátum. A harmadik említésre méltó formátum az SQLite fájl alapú relációs adatbázis kezelőt térbeli adattárolási képességekkel kibővítő Spatialite.

Az ESRI-hez kötődő formátumok kapcsán meg kell jegyezni azt, hogy ezeket alapvetően meghatározza a cég központi szoftverének, az ArcGIS for Desktop-nak (és az ArcCatalog-nak) a szoftveres struktúrája. Ennek vezérelve, hogy egy ún. multitier architektúrában működik, melynek a legnagyobb előnye, hogy az asztali szoftverek fejlesztése teljesen elhatárolt módon zajlik a rendszer adattárolási komponensétől, és így a kettőtől elvárt funkcionalitás nem keveredik. Többek között ez teszi lehetővé azt, hogy az ArcGIS szoftverben szinte teljesen egyenértékű módon használhatóak fel asztali, fájl alapú adattárolási formátumok, illetve számos kliens-szerver alapon működő adatbázis kezelő térbeli tárolási formája, így a vektoros adattípusok esetében legáltalánosabban használt feature class (mint legkisebb tárolási egység) minden egyes geoadatbázis fajtában alkalmazható. (ESRI 2016)

Az ESRI szoftverekben felhasználható térbeli adatbázisokra a cég a geoadatbázis (geodatabase) elnevezést használja, mely gyakorlatilag egy objektum-relációs adatbázist jelöl. Ennek az ún. geoadatbázisnak az adattáblái alapvetően két részből állnak. Vannak a felhasználó által definiált adatok, melyek szerkesztéssel illetve betöltéssel jönnek létre, de emellett még vannak az ún. rendszer táblák (system tables), melyek az egész geoadatbázis funkcionalitásának az alapját jelentik. Ezek tárolják gyakorlatilag a relációs adatbázis sémáját, az egyes adattáblák közötti kapcsolatokat, az adatbázis egyéb szabályait és működési sajátosságokat. (Fontos megjegyezni, hogy ezek a rendszer táblák különböznek az egyes geoadatbázis formátumok esetében, mivel részint ezek a táblák is határozzák meg azt, hogy ezek milyen funkcionalitással rendelkeznek, így egy Personal, File, és a majd később tárgyalt Enterprise geoadatbázis segítségével más és más szintű és jellegű feladatok oldhatóak meg.) (Nasser 2014)

Az ArcGIS 10-es verziója előtt ezeknek a rendszer tábláknak a száma olykor elérte a 35-öt is, de mára az adatbázis sémát elsődlegesen leíró táblák száma lecsökkent 4-re, melyek az alábbiak (ESRI 2016b):

- GDB_Items tábla: ebben kerül letárolásra a geoadatbázis valamennyi befoglalt eleme, vagy feature class-ok, feature dataset-ek, raster dataset-ek stb.
- GDB_ItemTypes tábla: tartalmaz egy előre definiált listát azokról az adattípusokról, amelyeket a geoadatbázisban tárolni lehetséges

- GDB_ItemRelationships tábla: a geoadatbázisban tárolt valamennyi felhasználói tábla és adatkör közötti kapcsolatot hivatott rögzíteni, mely jelenthet alá- és fölérendeltségi viszonyt
- GDB_ItemRelationshipTypes tábla: az egyes adattípusok lehetséges kapcsolódási viszonyait tárolja el

Természetesen ezek mellett még számos rendszer tábla előfordulhat, de ezek nagyban függnek a geoadatbázis fajtájától és a benne tárolt adatok típusától. Ezeknek a rendszer tábláknak tehát kettős céljuk van; egyrészt biztosítják a kapcsolódó asztali alkalmazásoknak a szükséges leíró és metaadatokat, másrészről lehetővé teszik a hagyományos egyszerű adattípusokon (pont, vonal, felület) felüli adatstruktúrák tárolását, úgymint mező alias-ok, topológia, vagy annotáció típusok.

A Personal Geodatabase időrendben az első fájl alapú téradat tároló formátum, mely formáját tekintve egy egyetlen fájlból álló Microsoft Access adatbázis. A formátum legnagyobb előnye, hogy a felhasználói és rendszer táblákhoz is lehetséges hozzáférni közvetlenül adatbázis oldalról, vagy akár ODBC kapcsolaton keresztül, így amennyiben szükséges, az ArcGIS környezeten is lehetséges az adatok módosítása, vagy lekérdezése. Azonban annak a ténynek köszönhetően, hogy a Personal Geodatabase egy Microsoft Access adatbázison alapszik, számos olyan hátrányos tulajdonsága van, amely mind az asztali szerkesztés során, mind pedig webes adatszolgáltatásnál hátrányként jelentkezik.

Elsőként kell megemlíteni, hogy a mérete 2 GB-ban van szabályozva, de a tapasztalatok szerint már 250-500 MB-os méretnél is szükséges számolni a teljesítmény jelentős csökkenésével. Az asztali szerkesztés során egyszerre csak egy felhasználható módosíthatja az adatbázist, nem megoldható az térbeli adatok párhuzamos szerkesztése, vagy verziók elmentése, és a párhuzamos olvasási műveletek is lelassulhatnak a felhasználók számának a növekedésével. (ESRI 2016c)

Ez az adattárolási forma tehát alapvetően inkább kevés felhasználónál, kisebb adatszerkezetek szerkesztésekor kaphat szerepet az asztali szerkesztésnél, de a webes környezetben való alkalmazhatósága több szempontból sem kedvező. Egyrészt az egy darab egyidejű felhasználó hozzáférési lehetősége nem kedvezne a sokszor párhuzamos hozzáférést igénylő webes adatszolgáltatáshoz, másrészt mivel az Access formátum csak Windows környezetben érhető el, így ez meglehetősen lekorlátozza az alkal-

mazhatóságát más operációs rendszerek környezetében, így ez az adattárolási típus nem alkalmazható egy nyílt forráskódú környezetben, mivel ezeknek az alapja mindig egy valamilyen Linux alapú környezet.

A File Geodatabase formátumot az ArcGIS 9.2-es verziójánál vezették be azzal a céllal, hogy kiküszöbölje a Personal Geodatabase nyilvánvaló hiányosságait és lecsökkentse annak korlátait. Ez a fejlesztés tehát már célirányosan egy modern, fájl alapú megoldás kialakítására irányult, és nem egy meglévő relációs adatbázis kibővítésére épített. A fejlesztés során elsődleges célul tűzték ki a formátum platform függetlenségét (vagyis hogy Windows és Linux környezetben egyaránt használható legyen), a tárolási kapacitás növelését, többféle komplex adattípus támogatását és a párhuzamos szerkesztés lehetőségét.

Ennek eredményeképpen létrejött egy olyan komplex térbeli adattárolási struktúra, amely fájl alapon, bináris formában tárolja az adatokat egy könyvtárszerkezeten belül, operációs rendszertől függetlenül. Az egy feature class, illetve egy feature dataset mérete egy TB méretben került korlátozásra, de megoldható akár 256 TB-os adatrétegek letárolása is (ez természetesen elsősorban raszteres adattípusok esetében bír nagy jelentőséggel). Emellett a Personal Geodatabase formátumnál lényegesen több komplex adatstruktúra támogatását nyújtja, illetve megoldható az egyes datasetek és feature classok akár párhuzamos szerkesztése is. Habár ebben a formátumban sem megoldott a jogosultságok álltása, illetve a verziók letárolása, mégis egy sokkal fejlettebb adattárolási mechanizmus jött létre, mely mára a lokális asztali szerkesztési környezetben az elsődleges és javasolt formátumnak tekinthető. Kedvező mérettulajdonságainak és a komplex adatszerkezetek támogatásának köszönhetően alkalmazása megfontolandó webes térképszolgáltatási feladatokhoz. (ESRI 2016c)

A Personal és a File Geodatabase formátumokat az alábbi táblázat hasonlítja össze:

	Personal Geodatabase	File Geodatabase
Felhasználás	Windows-os környezetben kisebb adattartalmú, elsősorban vektoros elemek tárolása és szerkesztése	Bármilyen környezetben akár nagyméretű vektoros és raszteres adatok tárolása és szerkesztése
Tárolási forma	Egyetlen mdb kiterjesztésű bináris fájlban	Egy mappában, minden egyedülálló feature class és dataset egy bináris fájl formájában tárolódik
Adattárolási korlátok	Elvben 2 GB, de a gyakorlatban már 250-500 MB-os méretnél is előfordulhatnak lassulások	1 TB feature class-onként vagy dataset-enként, de ez akár 256 TB-ra növelhető
Olvasási hozzáférés	A felhasználók számának növekedésével lassulások tapasztalhatóak	A felhasználók számának növekedésével lassulások tapasztalhatóak
Írási hozzáférés	Egy időben egy felhasználó szerkesztheti az adatbázis valamennyi részét	Egy időben egy felhasználó szerkesztheti az adatbázis egy dataset-ét, (de akár több dataset párhuzamos szerkesztése is lehetséges
Adattartalomhoz való hozzáférés ArcGIS környezeten kívül	Lehetséges, Microsoft Access-en, vagy ODBC kapcsolaton keresztül	Lehetséges, de csak a File Geodatabase API-n keresztül
Platform	Windows	Cross-platform
Webes adatszolgáltatásra való alkalmasság	Csak Windows-os környezetben, korlátozottan	Platform függetlenül, de elsősorban ArcGIS Server segítségével

2. ábra: A Personal és File geoadatbázisok összehasonlítása

A fájl alapú relációs adatbázisok kiemelkedő nyílt forráskódú képviselője az SQLite szoftver, melynek fejlesztéséért egy nemzetközi csapat felelős. A szoftver az adott szegmensben betöltött vezető szerepe egy jó példa arra, hogy a közösségi fejlesztéssel készülő szoftverekben mekkora potenciál rejlik, hiszen kedvező tulajdonságai miatt mára a legelterjedtebb adatbázis-kezelő a világon, hiszen használja:

- minden Android eszköz
- minden iOS és Mac eszköz
- minden Firefox, Chrome és Safari böngésző
- minden Windows 10-es eszköz

Az SQLite relációs adatbázis kezelőt C nyelven fejlesztik, és forráskódja szabadon letölthető. Alapkoncepciója, hogy egy fájlban tömörít egy teljes értékű relációs adatbázist minden elemével együtt, vagyis a táblák, view-k, indexek, és triggerek mind egy bináris fájlban tárolódnak. Az SQLite méretkorlátai rendkívül kedvezőek; egy adatbázis maximális mérete 140 TB lehet, egy rekord mérete akár 1GB is lehet. Emellett minden platformtól és felépítéstő függetlenül használható, vagyis a Windows és Linux 32 és 64 bites környezetét is támogatja. (SQLite 2016)

Erre a megoldásra épülve jött létre a SpatiaLite, mely időrendben a harmadikként jelent meg a piacon a Personal Geodatabase és a File Geodatabase után a 2008-as év folyamán. Ez a kiterjesztett változatú SQLite gyakorlatilag térbeli tulajdonságokkal ruházta fel az SQLite-ot, és nem sokkal ezután már egyre több szoftver implementálta a formátumot, és szerepe az asztali környezetben napjainkban is fokozatosan erősödik. (McInerney 2015) A nagyobb térinformatikai szoftverek közül mind az ArcGIS (a 10.2-es verziótól) mind pedig a nyílt forráskódú QGIS asztali GIS szoftver is támogatja, így szerepe az adatcserék során is jelentős lehet. A webes környezetben való felhasználása is megjelent már, a Geoserver szoftverben egy kiterjesztés (extension) segítségével használható jelenleg. (Geoserver 2016) Azonban az egyetlen fájl alapú szolgáltatás mindig hátrányosabb tulajdonságokkal bír, mint a klasszikus kliens-szerver architektúrában működő rendszerek, ezek tárgyalása a következő fejezetben történik.

I.1.2. Adattárolás relációs adatbázisokban

A fájl alapú térinformatikai adattárolás számos esetben nem elégíti ki az adatgazdák által megfogalmazott igényeket, különösen abban az esetben, ha az adatok felhasználása és szerkesztése több felhasználóhoz is kapcsolódik, ha fontosak a jogosultsági szempontok, vagy amennyiben az adattartalmat további alkalmazások is használni kívánják. Az egyes platformok fájlrendszerében történő tárolással járó korlátok miatt előtérbe kerülnek a kliens-szerver architektúrában működő relációs adatbázis

kezelők, melyek mind megbízhatóságban és teljesítményben, mind testreszabhatóságban felülmúlják a fájl alapú megoldásokat. (ESRI 2016c) A kedvezőbb tulajdonságok egy része magából az adattárolási formából eredeztethető, így a relációs adatbázisok esetében:

- lehetséges az adatok egyidejű elérése, nemcsak olvasási, de akár írási feladatok esetében is
- az adatbiztonság, illetve a jogosultságok beállítása nem a fájlrendszer segítségével, hanem központilag, az adatbázis szinten megoldható
- az adatokhoz való hozzáférés platformtól független módon történhet, nem szükséges helyi vagy hálózati meghajtó csatlakoztatása

A további kedvezőbb tulajdonságok már a relációs adatbázis kezelők működési jellegzetességeiből adódnak, így érdemes áttekinteni, hogy milyen alapelveken nyugszik ez az adattárolási forma.

Az adatok relációs adatbázisokban való tárolásának az alapelve, hogy egy önkényesen meghatározott objektumtípus elemeit táblák tartalmazzák, és az egyes objektumok leíró adatait azonos szerkezetű rekordok alkotják. Az egyes rekordokat mezők, vagy más néven attribútumok építik fel. Ugyanakkor ezek az egyes attribútumok rendelkezhetnek más és más, széles spektrumon mozgó lehetséges adattípussal, mely rendkívül szerteágazó típusú adat tárolását teszi lehetővé az adatbázisban.

A relációs adatbázisok tábláira vonatkozó egyik fontos alapvető kitétel, hogy rendelkezzenek elsődleges kulccsal, hiszen ez biztosítja, hogy az egyes táblák minden rekordja egyedi (vagyis az adattárolás elvi szinten redundancia-mentes) legyen. (Rob 2008) Az egyes adattáblák között definiálhatóak kapcsolatok, melyeket az elsődleges és külső kulcsok alapján való összekapcsolás biztosít. Ez a lehetőség tovább növelheti az adatbázisban esetlegesen előforduló redundancia mértékét.

Az relációs adatbázisokban tárolt adatok lekérdezése a Structured Query Language (SQL) segítségével történhet, mely széleskörű lehetőségeket ad az adott szempontrendszerhez igazodó szűrt információhalmaz leválogatásához. Az adatbázisokban létrehozható indexek segítségével töredékére csökkenthető az adatokhoz való hozzáférés nagy mennyiségű adatot tartalmazó táblák esetében. (Viescas et al. 2009)

Habár az egyes adatbázis kezelők implementációja különbözhet az architektúrájukat és a funkciókészletüket tekintve, mégis általánosan megfogalmazhatóak olyan

tulajdonságok, melyek miatt meggyőző alternatívát jelentenek az egyszerű vagy bináris fájl alapú adattárolással, illetve a fájl alapú adatbázis kezelőkkel szemben. A relációs adatbázis kezelők tehát:

- nagyobb adatbiztonságot nyújtanak, hiszen képesek az egyes adatváltoztatásokat tranzakcióban kezelni, így adatkorrupció nem fordulhat elő (egy tranzakció vagy teljesen végbemegy, vagy egyáltalán nem történik meg)
- párhuzamos, egyidejű adathozzáférést biztosítanak az adatbázis szerveren futó folyamat (process) segítségével, mely képes az igényekhez igazítva kezelni az egyes kéréseket
- alkalmasak az egyes adatbázisok kiterjesztésére rugalmas módon
- mind adattartalom, mind pedig felhasználás szempontjából jól skálázhatóak
- az objektum relációs megközelítésnek köszönhetően alkalmasak az adatokhoz kapcsolódó logika tárolásához; így létrehozhatóak tárolt eljárások, függvények vagy triggerek
- szabványos adatátjárhatósági lehetőségeket biztosítanak
- a normál felhasználás mellett képesek a mentési feladatok megoldására
- komplex lehetőségekkel bírnak az adatbiztonsági feladatok megoldásához; akár tábla szinten lehetséges a felhasználói jogosultságok beállítása a CRUD műveleteknek megfelelően (írás, olvasás, módosítás, törlés korlátozása)
- gyors adathozzáférést nyújtanak az SQL nyelv és az indexelés segítségével

A fent felsorolt szempontok mellett a térinformatika szempontjából a relációs adatbázisok egyik legfontosabb tulajdonsága, hogy bizonyos egyedi mezőtípusoknak köszönhetően lehetséges az egyes térbeli objektumok leképzése adatbázis környezetben. Ezt az egyes relációs adatbázis kezelők más és más megközelítéssel oldják meg, mégis mára szinte mindegyik rendelkezik valamilyen szintű földrajzi információ tárolására alkalmas megoldással.

Az egyes relációs adatbázis kezelők térinformatikai adattárolása azonban különbözik, bár alapvetően mindegyik az Open GIS Consortium (OGC) által kiadott Simple Features Specification-t igyekszik követni. Ez a specifikáció iránymutatást ad, hogy az egyes geometriai formákat illetve földrajzi koordinátákat milyen módon lehet leképezni SQL adatbázis szintjén, ezzel biztosítva a szabványosságot és az adatcserével járó folyamatok egyszerűsödését.

Ennek megfelelően rendelkezik térinformatikai támogatással a kereskedelmi adatbázis kezelők két nagy képviselője, a Microsoft SQL Server és az Oracle is, de az ingyenesen elérhető PostgreSQL és MySQL is tartalmaz térinformatikai támogatást. A natív, adatbázis kezelő szoftver által biztosított képességek (térbeli adatok tárolása és indexelése, elemző függvények biztosítása) figyelembe vételével azonban kijelenthető, hogy a legösszetettebb támogatással a kereskedelmi szoftverek világában az Oracle, a nyílt forráskódú világban pedig a PostgreSQL rendelkezik a PostGIS nevű kiterjesztése segítségével. (Mitchell 2005)

A natív megoldások mellett az ESRI cégnek létezik egy ún. middleware, vagyis közbeékelődő megoldása, mely az egyes adatbázis kezelőkre építve, de szoftveres komponensek segítségével teszi lehetővé térinformatikai adatok tárolását adatbázis környezetben. A beépülő komponens ArcSDE néven érhető el az ESRI cégtől, és a segítségével létrejövő térbeli adatbázis az előző fejezetben már megismert geoadatbázis fogalmi körébe illeszthető Enterprise Geodatabase elnevezést hordozza. Ez a korábban már tárgyalt Personal és File Geodatabase-nál jóval hatékonyabb tárolási forma, és segítségével az ArcGIS szoftver biztosította legmagasabb hatékonyság, testreszabhatóság és konfigurálhatóság jellemző rá. Alkalmas komplex téradatok tárolására. Az ArcGIS környezet az alábbi relációs adatbázisokhoz biztosítja az ArcSDE támogatást (ESRI 2016d):

- IBM DB2
- IBM Informix
- Oracle
- PostgreSQL/PostGIS
- Microsoft SQL Server

A natív és a middleware megoldások közötti különbség abban rejlik, hogy a natív formátumok egyszerűen nem tudnak minden szoftver működéséhez optimalizálni,

és éppen ezért előfordulhat, hogy egyes middleware megoldások egy adott szoftverben jobban teljesítenek. Ez azonban az esetek többségében azt is jelenti viszont, hogy a több szoftverből való elérésük lehetősége ez által csorbul, így ez mindenképpen egy hátrányos tulajdonságuk.

A két vázolt megoldás alkalmazását tehát alapvetően befolyásolja az, hogy melyek a használatukkal kapcsolatos elvárások. Amennyiben a legfontosabb szempont a szoftverfüggetlen adatelérés, úgy mindenképpen a natív formátumok jelentik a legelőnyösebb megoldást. Azonban ha egy adott, kiválasztott szoftverben mutatott teljesítmény a legfontosabb szempont, akkor a middleware megoldások alkalmazása kerülhet előtérbe. A disszertáció későbbi szakaszában még szó esik erről a jelenségről, és a megfelelő megoldásról is, amely az esetek többségében a kétféle módszer ötvözését jelenti.

A webes térinformatikai adatszolgáltatás szempontjából tehát a relációs adatbázis kezelőkben való tárolás lehet a legalkalmasabb megoldás, hiszen ezek számos olyan többletfunkcióval és tulajdonsággal rendelkeznek, amellyel a fájl alapú megoldások nem. Az adattároláshoz szorosan kötődik a WebGIS rendszerek következő eleme, a rendszer szerver oldali komponense, amely alapvetően meghatározza az adatokra épülő szolgáltatások lehetőségeit és minőségét.

I.2. A WebGIS rendszerek szerver oldali komponense

Az előző fejezetben a térinformatikai adatok tárolásáról esett szó, melynek végén megállapításra került az a tény, hogy a webes adatszolgáltatás szempontjából a legalkalmasabb a vektoros térinformatikai adatok relációs adatbázisban való tárolása. De miért is lehet hasznos egy interneten (vagy intraneten) történő webes adatszolgáltatás?

Egy vállalati, szervezeti környezetben alapvető fontosságú a közös felhasználási célú adatok megosztása, mely sok esetben a korábban már tárgyalt adatbázis kezelők, illetve fájlszerverek segítségével megoldható. Azonban ha a belső hálózatokon túl is igény van az adatok megosztására a vállalati egységek földrajzi szempontból szétszabdalt mivolta miatt, akkor a GIS szerverek nyújthatják a megfelelő eszközt erre a feladatra. Használatuk abban az esetben is indokolt lehet, amikor a felhasználóknak sem a

szoftveres, sem a hardveres környezete, sem pedig a hozzáértése nincsen azon a szinten, hogy a térinformatikai adatokat lokálisan, saját környezetében kezelje. Ezen felül jó szolgálatot teljesítenek a GIS szerverek abban az esetben is, hogyha a térinformatikai adatokat a "mindennapi" felhasználóknak szükséges biztosítani.

Technológiai szempontból ezeknek a szolgáltatásoknak az alkalmazott formái olyan webszervizek, melyek fogadják a felhasználók kéréseit, feldolgozzák azokat, majd választ küldenek rá valamilyen formában, futási környezetük pedig egy erős hardveres tulajdonságokkal rendelkező szerver-környezet.

I.2.1. A GIS szerverek hardveres kialakítása

A GIS szerverek általában sokkal erősebb hardvert igényelnek, mint egy általános, GIS szerkesztésre és feldolgozásra használt asztali számítógép. A processzor, a memória (RAM) mennyisége, a grafikus kártya és a hálózati kártya különös hangsúlyt kapnak, mert minőségük vagy mennyiségük komoly hatással lehet a működésre. A stabilitás, a futás közbeni hibajavítások, a folyamatos rendelkezésre állás és a biztonsági elvárások miatt a GIS szerverek kifejezetten szerver típusú környezeteket igényelnek. (Peters 2008)

A fejezetben előbb bemutatásra kerülnek a hardveres egységek annak tükrében, hogy milyen folyamatok, feladatok során játszanak központi szerepet, majd a fejezet második felében szó esik a hardveres környezet kialakításának a tervezéséről, és annak fontosságáról.

I.2.1.1. A hardveres elemek feladatai

A GIS szerverek legmeghatározóbb része a PC-khez hasonlóan a rendszer számításokért felelős egysége, a processzor. Mivel a szerver sokszor rendkívül számításigényes feladatokat lát el, ezért fontos a processzor magasabb órajele, amely megadja az egységnyi idő alatt elvégzett számítások mennyiségét. Az egyes fizikai magok számának, illetve a processzorok virtualizáló képességének is kifejezetten nagy a jelentősége, hiszen így képes a gép több szálon futó feladatokat megoldására. A processzor igénybevétele szolgáltatások elindításakor, vektoros adatokból raszteres kép előállításakor,

térképszervizek cache állományának a generálásakor, illetve speciális számításigényes lekérdezések esetében magas.

Mivel számos kérés kiszolgálása ismétlődhet a GIS szerverek esetében, ezért rentábilis ezeknek a letárolása a gyors elérés érdekében a RAM területeken, így rendkívül magas lehet ennek a mértéke a szerver gépekben, értéke akár a normál asztali gépek akár 10-szerese is lehet. Természetesen ezeknek az írási-olvasási sebessége is meghatározó. Mennyiségének leginkább a szolgáltatások fenntartásánál, raszter-vektor konverzió esetén van döntő jelentősége.

A háttértároló, vagyis a merevlemez legfontosabb tulajdonsága annak mérete, illetve talán még ennél is fontosabb az írás és olvasás sebessége. Újabban a klasszikus merevlemezeket egyre több helyen váltják fel a drágább, mozgó alkatrészt nem tartalmazó SSD (Solid State Drive) meghajtók, melyek sebessége 2-3-szor gyorsabb lehet egy átlagos merevlemeznél, de jelenleg még elterjedtebbek a hagyományos merevlemezeknél. Ezeknél nagy jelentősége van a RAID kötetek kialakításának, melyek alapvetően a sebességet és a biztonságot határozzák meg. Összességében megállapítható, hogy a merevlemezek sajátosságai a pragmatikusan letárolható térinformatikai adatok minőségét és mennyiségét határozzák meg. (Tomlinson 2013)

A térinformatikai adatokból, meghatározott szerkesztési elvek alapján előállítandó grafikus kimenet létrehozásakor nagy jelentősége van a grafikus kártyának is. Ennek szerepe a elsősorban a vektor-raszter konverziók során van.

Mivel a GIS szerverek adatszolgáltatása webes protokollokon keresztül történik, ezért rendkívül fontos ezek hálózati elhelyezése, vagyis a számukra biztosított sávszélesség. Ennek nagysága természetesen függ az elérni kívánt céloktól, de arról semmiképpen nem szabad megfeledkezni, hogy egy hagyományos webszerverhez hasonlítva nagyobb hálózati forgalmat bonyolítanak le. Ennek megfelelően a hálózati kártya adatforgalmi mutatója rendkívül fontos, kulcsszerepe elsősorban a nagy adatmennyiséggel járó párhuzamos lekérdezések esetén van.

Az erősebb hardveres környezet mellet a GIS szerverek esetében természetesen jelentős hangsúlyt kapnak még a szerverek asztali gépekkel szembeni extra tulajdonságai, melyek a stabilitást, a rendelkezésre állást és az adatvesztés minimalizálását biztosítják.

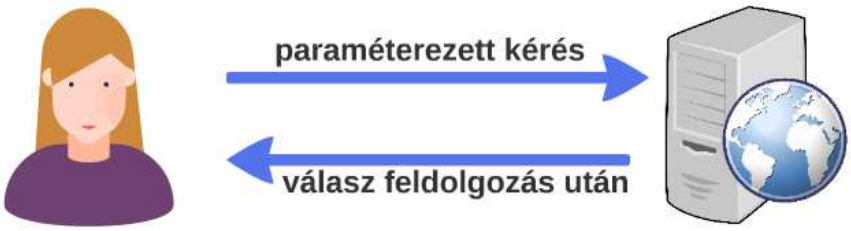
A hardveres tényezők tehát nagyban befolyásolhatják a GIS szerverek működését, viszont az ezekre épülő szoftveres implementációk határozzák meg a GIS szerverek tényleges funkciókészletét és az adatszolgáltatás hatékonyságát, ezért a következő fejezetben ezekről esik majd szó.

I.2.2. A GIS szerverek szoftveres implementációi

A térinformatikai adatszolgáltatás feladatára született megoldások megismeréséhez és összehasonlításához szükség van annak az ismeretére, hogy ezek milyen feladatokat kell, hogy ellássanak. Ahogy az más korábban is említésre került, a GIS szerverek alapvető feladata a térinformatikai adatok többrétű szolgáltatása webes protokollok segítségével, melybe nemcsak a szorosan vett adatszolgáltatás érthető bele, hanem a meglévő adatokon alapuló lekérdezési és feldolgozási feladatok is.

Szoftverfejlesztési szempontból lényeges az a tény, hogy az adatszolgáltatás webes környezetben kell, hogy történjen, hiszen ez meghatározza az alkalmazható technológiai megoldások palettáját. Arra a célra, hogy a felhasználó weben keresztül kapjon választ térinformatikai kérdéseire, a legalkalmasabb eszközök a webszervizek. Ezek a következő módon működnek egy GIS szerver esetében:

- a felhasználó megfogalmaz egy kérést, melyet általában a kliens oldali környezet segítségével paraméterezett formában elküld a szerver irányába HTTP protokoll segítségével
- A GIS szerver fogadja, és formai szempontból megvizsgálja az érkezett kérést
- amennyiben formailag megfelelő a kapott kérés, úgy feldolgozza azt a szerver erőforrásainak használatával, a rajta tárolt adatokat felhasználva
- az elkészült választ HTTP protokoll segítségével visszaküldi a kliens oldalra a felhasználóhoz



3. ábra: Http alapú kommunikáció

Ahogyan általában egy adott cél eléréséhez többféle megoldás is létezhet, úgy természetes az, hogy a webes térinformatikai adatszolgáltatásra is más és más implementációk születtek eddig, hiszen minden fejlesztői csoport a saját szempontrendszerét figyelembe véve igyekezik a legjobbat megalkotni. Felhasználói (illetve kliens oldali fejlesztői) szempontból ugyanakkor jogos elvárás, hogy ezek valamiféle közös nyelvet "beszéljenek".

I.2.2.1. Szabványos webes térinformatikai adatszolgáltatási formák

Az OGC (Open GIS Consortium) szabványosítási szerepe nemcsak az adattárolás esetében jelentős; már a GIS szerverek kialakulásának a kezdetén törekedtek az egységesítésre, és létrehozták a WebGIS mára is legfontosabb szabványait, melyek szoftveres átjárhatóság és a közös felhasználás szempontjából nagy jelentőséggel bírnak. A legfontosabbak ezek közül az alábbiak:

- WMS (Web Map Service): a felhasználó által paraméterek segítségével megfogalmazott kérésre egy georeferált raszteres térkép-részletet küld vissza a kliens oldalra
- WMTS (Web Map Tile Service): egy raszteres georeferált térkép tileosítással (csempézéssel) előállított állományának a szabványosított lekérési formája
- WFS (Web Feature Service): a felhasználó kérésére térbeli objektumokat küld a kliens oldalra geometriai és attribútum adatokkal együtt
- WCS (Web Coverage Service): egy adott területre vonatkozó térbeli objektumok halmazának szabványos lekérdezésére szolgál; ezek sokszor térben és időben változó jelenségeket írnak le

- WPS (Web Processing Sevice): az előbbieknél rugalmasabb szolgáltatási forma, mely egy szerveren végzett számítási, megoldási folyamat elindítását teszi lehetővé a kliens oldalról kezdeményezve

Látható tehát, hogy a szabványosítás igyekszik minden típusú adatszolgáltatást lefedni, ennek ellenére azonban az egyes szoftveres implementációk számos esetben igyekeznek a szabványok nyújtotta lehetőségeket kibővíteni, és így eltérni a szabványtól. Ez két tényezőre vezethető vissza:

- a szabványok lassabban "reagálnak" a felhasználói igényekre, sokszor megfordul a folyamat, és nem a szoftverek követik a szabványokat, hanem a szabványok veszik át a szoftverek megoldásait
- az egyes fejlesztői csoportok előtérbe helyezik a más szoftverükhöz való hatékonyabb kapcsolódást, és így eltérnek a szabványoktól

A következő fejezet mind a szabványok, mind pedig a használt technológia szempontjából vizsgálja meg az egyes szoftveres implemetációkat.

I.2.2.2. A szoftveres implementációk különbségei

A hardveres adottságok kihasználására, a térinformatikai adatszolgáltatás feladatára tehát különféle szoftveres megoldások születtek. Ezek leginkább az implementáció programozási nyelvében és a felhasznált technológiában különböznek, de kialakulásukat, formálódásukat alapvetően a fejlesztési csoportok szem előtt tartott céljai és szempontjai határozták meg. De melyek is lehetnek ezek a fő szempontok?

Elsőként a szorosan vett teljesítmény tekinthető a fejlesztések egyik fő céljának, mely általában az egyes szolgáltatások kvantitatív jellemzőire vonatkozik, ilyen lehet például egy adott lekérdezés feldolgozásához szükséges minimális időintervallumon belüli teljesítése. Mivel a gépi kódhoz a C nyelv áll a legközelebb (a legtöbb programozási nyelvben előforduló absztrakciós réteg hiánya miatt), ezért az ilyen nyelven írt szoftverek tekinthetőek ezen a téren a legelőnyösebbeknek, de természetesen ez csak abban az esetben lehet igaz, hogyha a kód minősége "ideálisnak" tekinthető. A sebességet, a teljesítményt tehát a választott programnyelv, a kód minősége és a futtatási környezet együttesen határozza meg.

A második szempont a szoftver funkciókészlete lehet. Ez talán nem szolgál további magyarázatra; mértéke elsősorban a tervezéssel, a fejlesztés időbeli ráfordításával egyenesen arányos.

A harmadik szempont lehet az egyes szoftverek felhasználói felülete, mely a WebGIS világában talán az egyik legkritikusabb elem. Mivel ezeket a szoftvereket arányaiban sokkal kevesebben használják, mint az asztali GIS eszközöket, így általában ezen a szegmensen igyekeznek a fejlesztési költségeket leginkább csökkenteni. Minősége az előző ponthoz hasonlóan a ráfordítási idővel és az esetlegesen használt webes keretrendszerrel van összefüggésben.

4. ábra: A szerver oldali komponensek három meghatározó tulajdonsága

Mivel a doktori értekezés elsősorban a földtani adatok kartografált megjelenítésére fókuszál, így szükséges még egy negyedik szempontot is kiemelni, ez pedig a GIS szerverek kompatibilitása az asztali GIS szoftverekkel. Ez a tényező kulcsfontosságúnak bizonyulhat, hiszen a bonyolultabb jelkulcsi elemek webre történő átörökítése nehéz feladat lehet, amennyiben a szerver és az asztali szoftver más renderelő egységet használ, illetve ha nincsenek megfelelő konverziót biztosító eszközök. Így az egyes imp-

lementációk összehasonlításakor szükséges megvizsgálni a jelkulcs átörökítésének lehetőségeit is az egyes elemeknél.

Fontos megjegyezni, hogy az implementációk száma rendkívül magas, így most csak a legjelentősebbek, illetve a disszertáció szempontjából a leginkább releváns szereplők kerülnek bemutatásra.

A zárt forráskódú GIS szoftverek piacának vezető szereplője az ArcGIS termékcsalád, mely a GIS szerverek között is erősen képviselteti magát. Alapvetően Windows platformra épít, és működéséhez számos programozási nyelvet használ, így a C#. NET-et, a C++-t és a Python nyelvet is. Jellemző rá a magas teljesítmény, a rendkívül széles funkcionalitási kör, a felhasználói felület kidolgozottsága, és a saját asztali szoftvereivel való szoros együttműködés. Az asztali környezetben kartografált térképek jelkulcsa az desktop szoftver formátumában kerülnek mentésre, melyen alapulva a szerver oldali komponens, vagyis a GIS szerver képes belőle olyan térképes szolgáltatást indítani, mely az asztali megjelenéssel megegyező kinézetet biztosít.

A nyílt forráskódú szerver GIS szoftverek három képviselője kerül rövid bemutatásra; a Mapserver, a Geoserver és QGIS Server.

A Mapserver nevű szerver oldali WebGIS komponens volt az első nyílt forráskódú GIS szerver implementáció. C nyelven íródott, és éppen ezért a korábban említett teljesítmény vonatkozásában az egyik leghatékonyabb eszköz napjainkig is. (Flower 2012) A másik két szempontot figyelembe véve viszont megállapítható, hogy funkcionalitásban szűkösebb lehetőségekkel rendelkezik, mint az ArcGIS Server, és a felhasználói kezelő felület pótlására napjainkig is csak halványabb próbálkozások születtek. Az asztali környezetben kialakított, kartografált térképi állományok stílusainak átörökítése szintén problémás, hiszen a Mapserver által használt mapfile formátum lehetőségei sokszor különböznek az asztali környezetben működő térképszerkesztők funkcionalitásától.

A Geoserver szoftver egy Java alapú megoldás, mely ennek köszönhetően teljes mértékben platform-független, hiszen a Java alapú megoldások mind Windows, mind pedig Linux környezetben egyaránt futtathatóak. A benchmarkok azt mutatják, hogy teljesítményben a Java-ban alkalmazható többszálú futtatásnak köszönhetően általában alig marad el a Mapserver-től. Felhasználói felülete viszont a legkiterjedtebb a nyílt forráskódú megoldások között, melynek komplexitása az ArcGIS Server-rel vetekszik.

Az asztali stílusok örökítésében viszont szintén kevésbé jól teljesít; a weben megjelenítendő stílushoz használt XML alapú SLD (Styled Layer Description) formátum lehetőségei korántsem hasonlíthatóak össze az asztali kartografálás eszközeivel. (Henderson 2014)

A QGIS Server formáját tekintve egy FastCGI/CGI modul, mely az Apache webszerver segítségével válik teljes értékű GIS szerverré; fejlesztése C++ nyelven történik. Teljesítményét tekintve talán elmarad az open-source vetélytársaitól, és funkciókészlete sem olyan széles, mégis van egy hatalmas előnye a riválisaival szemben. Ez pedig az, hogy fejlesztése szoros együttműködésben történik a QGIS Desktop szoftverrel, és emiatt a stílusok webre való örökítése rendkívül egyszerűen történhet, hiszen a QGIS Server ugyanazt a renderelő egységet alkalmazza, mint amelyet a Desktop párja is.

A fent említett szoftveres szerver oldali megoldásokat tekintve látható az, hogy a nyílt forráskódú világ szereplői esetében teljesítmény, a funkcionalitás és a felhasználó-barát megközelítés korántsem azonos szinten mozog. Így a közülük való választás nagyban múlik azon, hogy melyek azok a szempontok, amelyek kiemeltek a feladat megoldása során, így különféle szcenáriók esetén más és más kombinációk lehetnek alkalmasak.

I.3. A kliens oldali megjelenítés lehetőségei

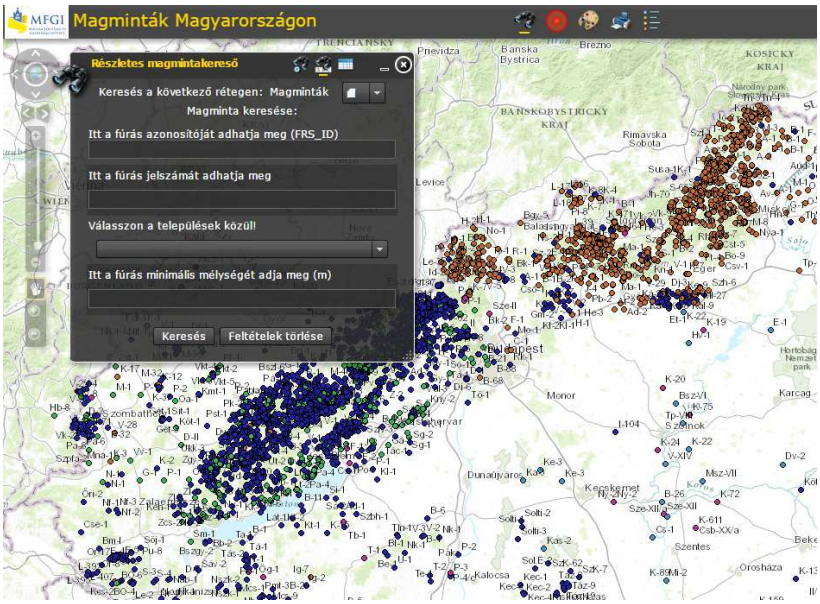
A WebGIS rendszerek eddig tárgyalt összetevői teremtik meg az alapot a webes protokollok segítségével történő térinformatikai adatszolgáltatásra. Az így létrejövő webszervizek alkalmazása ezután többféle módon történhet; akár asztali GIS szoftverekből, akár mobil platformokra (pl. Android, iOS) fejlesztett célalkalmazásokból. Ezekkel ellentétben viszont sokkal népszerűbbek lehetnek az asztali vagy mobil eszközök böngészőjéből történő felhasználási módok, hiszen a korábban említett valamennyi módszer esetén valamilyen külön szoftver telepítésére van szükség, míg egy webböngésző mára minden asztali és mobil eszköz alapvető részeként tekinthető. Éppen ezért ebben a fejezetben a böngészőhöz kötődő megoldások kerülnek áttekintésre, melyekre összefoglalóan kliens oldali megoldásokként történik majd a hivatkozás.

Az egyes kliens oldali megoldásokat alapvetően két csoportba lehet osztani. Az első csoportba a böngészők beépülő moduljaira épülő megoldások, míg a másikba a böngészők natív nyelvét, a Javascript-et használó megoldások tartoznak. Ezekről következik egy rövid áttekintés.

I.3.1. Böngészők beépülő moduljaira épülő megoldások

A böngészők és a web világa jelenleg a számítástechnika egyik leggyorsabban fejlődő szegmense. Ennek oka többek között az, hogy használata mára rendkívül széles körben elterjedt, valamint különféle célok megvalósításához is alkalmas eszköz lehet. Azonban a 2000-es évek végéig fejlettségük és funkciókészletük még nem volt azon a szinten, hogy minden igényt kielégítsen; elmaradottsága többek között a képi és multimédiás formátumok kezelésében mutatkozott meg. Ennek a kiküszöbölésére a böngészők lehetővé tették bizonyos beépülő modulok telepítését a böngésző környezetébe, és így számos célfeladatra alkalmassá váltak a böngészők.

A kliens oldali térinformatikai megjelenítés volt az egyik olyan terület, melyben változást hoztak a plugin alapú megoldások. Ezeket jellemzően az ESRI cég igyekezett minél nagyobb mélységében kihasználni, és így jött létre az ArcGIS Viewer for Silverlight, valamint az ArcGIS Viewer for Flex, melyek természetesen az ArcGIS Server-re építettek működésük során. Előbbi a Microsoft Silverlight, utóbbi az Adobe Flash beépülő modult igényelte a felhasználó böngészőjében.



5. ábra: Adobe Flash alapú kliens oldali megjelenítés

Segítségükkel interaktív alkalmazások formájában vált lehetővé a térinformatikai adatok böngészése és lekérdezése. Egyik legnagyobb előnyük volt, hogy a beépülő modulokon alapulva minden böngészőben egyforma működést és megjelenést garantáltak. Mivel azonban működésükhöz meglehetősen jelentős kliens oldali erőforrás volt szükséges, így az okos mobil eszközök (okostelefonok, tabletek) elterjedésének köszönhetően az így készült alkalmazások felhasználása jelentősen korlátozott volt. A beépülő modulokon alapuló megoldások helyzetén tovább rontott az a tény, hogy a 2010-es éve második felétől az egyes böngészők a plugin technológiákat illetően fokozatos kivezetési stratégiákkal álltak elő, és ennek köszönhetően ezeknek a megoldásoknak a támogatását maga a gyártó is megszüntette.

I.3.2. Javascript alapú megoldások

Az okos mobil eszközök elterjedésével a böngészők fejlődése tovább katalizálódott, és a HTML5 szabványok egyre nagyobb térhódításával a natív Javascript alapú megoldások kerültek előtérbe a webes térinformatikai adatszolgáltatásokon alapuló interaktív alkalmazások esetében is. (Dhakal 2016)

A Javascript alapú megoldások egyik központi elve, hogy a webes alkalmazás a korábbiakhoz képest sokkal nagyobb arányban használja a felhasználó gépének erőforrásait. Ez egy napjainkra kialakult tendencia része, mely a korábbi elsősorban a szerver erőforrásait hasznosító modell helyett egyre több feladat megoldását bízza a kliens oldali gépre. Ennek célja egyrészt a szerverek tehermentesítése, másrészt a kliens oldalon tapasztalható sebesség és interaktivitás növelése. Ezzel a működési elvvel viszont előtérbe került az AJAX (Asynchronous Javascript and XML) technológia, mely a webes alkalmazások az interaktivitást biztosítja. Alapelve, hogy az egyes weboldalak olyan módon képesek adatot cserélni a szerverekkel, hogy azok újratöltődnének.

A Javascript-et hasznosító fejlesztések másik nagy előnye, hogy a megírt alkalmazás minden platformon és eszközön (asztali vagy mobil) megközelítőleg azonos módon futnak. Azért csak megközelítően, mert az egyes böngészők implementációi különbözhetnek, és előfordulhat olyan eset, amikor egy böngésző rendelkezik egy adott képességgel, míg a másik azt nem támogatja. Ez tehát egyúttal a Javascript alapú tech-

nológiák egyik legnagyobb kihívása is; olyan keretrendszert vagy programkönyvtárat fejleszteni, amely minden modern böngésző aktuális verzióját kiszolgálja. Ennek a célnak az eléréséhez természetesen elengedhetetlen a kliens oldali szoftverek folyamatos fejlesztése és karbantartása a legújabb böngésző verziók új megoldásaihoz igazodva.

Az általános jellemzők áttekintése után érdemes megvizsgálni, hogy miként működnek a Javascript alapú webes GIS alkalmazások. Az alkalmazás alapját egy webszerveren tárolt, a böngészőben HTML oldalként renderelt környezet jelentheti. Ez formátumát tekintve lehet egy html kiterjesztésű állomány, de akár valamilyen szerver oldali eszköz segítségével (pl. PHP) is létrehozható. A böngészők lehetővé teszik a HTML oldalakba opcionálisan beágyazott kliens oldali Javascript kódrészletek végrehajtását, mely nagyrészt a webes oldalak interaktivitását biztosítja. Ezek a kódrészletek képesek lehetnek arra, hogy akár a korábban tárgyalt GIS szerverek által biztosított webszervizeket meghívják AJAX kérések segítségével, majd a kérésekre kapott válaszokat fel is tudják dolgozni, és a felhasználó felé közvetíteni. Így a webes alkalmazások gyakorlatilag a szerverrel való szinte folyamatos kapcsolatnak köszönhetően tudják biztosítani a térinformatikai adatok interaktív szolgáltatását.

6. ábra: A kliens oldali eszközök kommunikációja a GIS szerverekkel

A WebGIS-ben kialakult szabványos adatcsere és webszerviz formátumoknak köszönhetően azonban a webes GIS alkalmazások esetében egymáshoz nagyon hasonló kódrészletek szükségesek egy adott feladat (pl. egy térképrészlet lekérése) megoldásához. Ezért létrejöttek olyan nagymennyiségű kódot tartalmazó ún. programkönyvtárak (library-k), melyek lehetővé teszik, hogy a webes alkalmazás fejlesztője csak a konkrét alkalmazáshoz szükséges paraméterek megadásával, egyszerűsített formában programozhassa le a kívánt működést egy interfész (Application Programming Interface, API) segítségével.

A különböző API-k hasonló célt szolgálnak: olyan eszközzel látni el a webes GIS alkalmazások fejlesztőit, hogy azoknak ne teljesen az alapoktól kelljen megoldani a különféle forrásból származó térképi rétegek megjelenítését vagy egyes alapvető térinformatikai funkciók integrálását, hanem már egy induló eszközkészlettel ellátott keretet kelljen csak továbbfejleszteniük. Hozzá kell tenni azonban, hogy a felhasználói felületet az esetek többségében még a fejlesztőnek kell megalkotnia, melynek mára elfogadott alapelve és egyben elvárása, hogy "minden eszközön jól mutasson", vagyis reszponzív legyen (minden eszköz saját méreteihez igazodó megjelenés). Ennek a célnak fontos eszköze a legújabb webes szabványok, vagyis a HTML5 és a CSS3 használata.

A kliens oldali eszközökkel megvalósul a GIS adatok interaktív elérésének biztosítása az adattárolástól egészen a weben való megjelenítésig. A WebGIS rendszerek kialakítását azonban befolyásolhatja az egyes adatszolgáltatások tartalmi jellege és tematikája. A következő fejezet ennek megfelelően a kutatás adathátterét, vagyis a földtani adatokat, és az azokon alapuló térképszerkesztési folyamatokat tárgyalja.

II. A földtani térképszerkesztés folyamata

Az előző fejezetekben áttekintésre kerültek a WebGIS elemek főbb alkotóelemei. Alkalmazásukat azonban alapvetően befolyásolják a szolgáltatni kívánt adatok tartalmi és térinformatikai jellegzetességei, illetve a földtani térképkészítés folyamata, ezért ebben a fejezetben ezekről esik majd szó. Bemutatásra kerülnek a földtani adatok főbb jellegzetességei kartográfiai és térinformatikai szempontból, a digitális térképszerkesztés környezete, illetve mindaz a hardveres, hálózati és szoftveres háttér, ami a térképszerkesztési feladatok elvégzését lehetővé teszi.

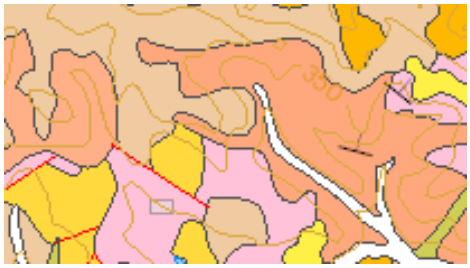
II.1. A földtani térinformatikai adatok jellegzetességei

A webes térinformatikai adatszolgáltatás egyik központi kérdése, hogy miként lehetséges az asztali környezetben létrehozott kartografált térképek megjelenítése a weben. Emiatt érdemes azt megvizsgálni, hogy az Intézetben folyó kartográfiai munka milyen típusú adatokat használ, vagyis hogy milyen a földtani adatok jellege térinformatikai szempontból.

A földtani térinformatikai adatok egyes esetekben terepi felvételezés vagy mérés eredményeként keletkeznek, máskor viszont mért adatok további értelmezéséből származnak. Mivel a földtani térképezés feladata elsősorban az, hogy meghatározza az adott geológiai képződmények földtani jellemzői mellett ezek térbeli elterjedését, ezért ezek az adatok jellegüket tekintve alapvetően vektoros adatok, (ritkábban 3 dimenziós állományok, de ezek nem képezik a dolgozat témáját). Tárolásuk és szervezésük során alapvetően a felület típusú elemek a leginkább jellemzőek, de némely esetben előfordulnak a vonal, illetve pont típusú geometriával rendelkező elemek is.

A földtani térképek legjellemzőbb adattípusa tehát a felületi típusú elemek, melyekkel a geológiai képződmények elterjedését lehet ábrázolni. Ennek a jelölésére többféle tematikus kartográfiai módszer is használható.

A leginkább kézenfekvő, és a legelterjedtebben alkalmazott megoldás a felületi színezés használata.



7. ábra: Egyszerű felületi színezés

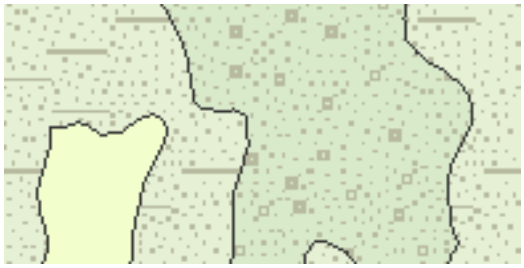
Ennek használata során a kartográfust nemzetközi, konvencionális színskála segíti, mely a főbb geológiai korok színezési módjára tartalmaz utasításokat. Éppen ezért a földtani térképek szerkesztésekor kiemelt fontosságú geológus szakértők bevonása a munkába, hiszen csak ők rendelkezhetnek azzal a szükséges szaktudással, amely alapján az egyes földtani képződmények színskálája logikusan kialakítható.

Földtani kor	Szín
neogén	sárga
paleogén	narancssárga
kréta	zöld
jura	kék
triász	lila
perm	sárgásbarna
karbon	szürke
devon	barna
szilur	világos szürkészöld
ordovícium	olajzöld
kambrium	sötét kékeszöld
proterozoikum	lilás rózsaszín
archaikum	rózsaszín

8. ábra: Földtani időszakok és konvencionális színeik

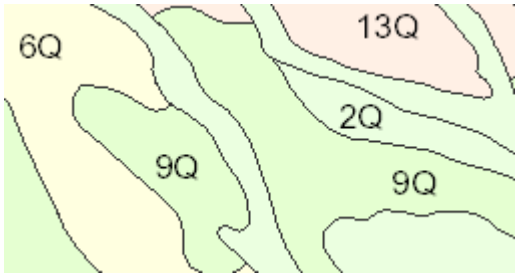
A felületi színezés önmagában azonban sokszor nem elégíti ki a térképolvasói vagy felhasználói igényeket. Ez elsősorban a földtani térképek jelkulcsi elemeinek magas számából adódik, mely tematikánként akár a több százas nagyságrendet is elérheti, sokszor figyelmen kívül hagyva a térkép méretarányából fakadó ábrázolási nehézségeket.

A felületi színezést így kiegészítheti a felületi jelek módszere, amely valamilyen geometriai minta ismétlését jelenti a diszkréten lehatárolt objektumok egészén megjelenítve. Ennek a módszernek a használata azonban a térinformatikai szoftverekben vektoros, vagy raszteres állományok ismétlésével lehetséges, és sokszor rendkívüli mértékben megnöveli az állományok méretét és használhatóságát, ezért csak korlátozott mértékben alkalmazzák.



9. ábra: Felületi színezés és felületi jelek együttes alkalmazása

A felületi színezés értelmezhetőségének a javítására a földtani térképek esetében elterjedten alkalmaznak felületi elemekre vonatkozó megírásokat, vagyis földtani indexeket. Ezek az összetett tartalommal bíró szimbolikus jelek iránymutatást adnak elsősorban az adott földtani képződmény korára, de a genetikára, a kőzetrétegtani beorolásra (formációkra), és kőzettani összetételre is tartalmazhatnak adatot.



10. ábra: Jellemző ábrázolásmód, felületi színezés és indexek használata

A térképszerkesztési és térképolvasási szempontok figyelembe vételével kijelenthető, hogy a felületi színezés földtani indexekkel való közös használata, alkalmanként felületi jelekkel kombinálva a legalkalmasabb a földtani tematikus térképek ábrázolásakor.

Mivel a térinformatikai rendszerekben való adattárolás során lehetséges az egyes földtani objektumokhoz szimbolikus információkat rendelni, ezért az imént említett kartográfiai módszer megvalósításához minden földtani polygon típusú elemhez

szükséges tárolni az adott földtani képződmény indexét. Amennyiben ez teljesül, úgy az alkalmazott színek, illetve földtani indexek egy központi adatbázisból hozzárendelhetőek a polygonokhoz, minden egyes térképre vonatkozóan.

II.2. A földtani térképek típusai

Magyarország földtani jellemzőinek a bemutatására számos térképmű született, melyeket többféleképpen is lehetséges csoportosítani. Az első kézenfekvő csoportosítás lehet a méretarány alapján való elkülönítés. Földtani értelemben ez szoros összefüggésben van az adatsűrűséggel, vagyis ebben az esetben a méretarány azt mutatja meg, hogy egy adott tematika kidolgozottsága milyen nyomtatott méretarányban tekinthető ideálisan olvashatónak. Eszerint megkülönböztethetőek a milliós nagyságrendű áttekintő térképek, az általában egy adott tematikát országosan bemutató közepes méretarányú térképek, valamint a regionális vagy lokális földtani jellemzőket bemutató nagy méretarányú tájegységi és szatelit térképek.

A földtani térképeknek adattartalmuk alapján is többféle verziója lehet: észlelési (a geológus által terepen észlelt képződmények), fedett (negyedidőszakot is ábrázoló), fedetlen (negyedidőszakot nem ábrázoló), mélyföldtani (tetszőleges földtani képződmény, vagy időszak képződményeinek elhagyásával).

Pusztán a tematikát alapul véve is lehetséges az osztályozás: így léteznek szerkezetföldtani, építésföldtani, mérnökgeológiai, hidrogeológiai, földtani veszélyeztetettségi stb. térképek.

A területi felosztás alapján megkülönböztethetőek a lokális, a regionális, az országos vagy a nemzetközi területeket lefedő földtani térképek.

A fenti szempontoknak megfelelően a Magyar Földtani és Geofizikai Intézetben számos földtani térkép készült el az évek során, melyeknek adatbázisai és jelkulcsai rendszerezett formában tárolódnak. Bár a felvételezés és a térképek nyomtatott megjelenése hagyományos, a korszerű földtani térképek valójában olyan adatrendszerek, és több funkciós információs adatbázisok, amelyekből tetszőleges digitális térképi tartalom generálható (Turczi 2010, Kercsmár et al. 2014).

II.3. A földtani térképszerkesztés technológiai háttere

A földtani adatok jellegének, és az alkalmazott kartográfiai megoldásoknak az áttekintése után ebben a fejezetben bemutatásra kerülnek a jelenlegi térképszerkesztési folyamatoknak.

A Magyar Földtani és Geofizikai Intézetben a térinformatikai adatok kezelése, és a kartográfiai munka az ArcGIS termékcsalád segítségével zajlik.

Az Intézetben a felhasználók számítógépei egy Windows hálózat részei, a felhasználók jogosultságainak a kezelése a Windows Active Directory segítségével történik, mely egy dedikált szerveren fut. A belső hálózat minden kapcsolódási pontján garantált a 100 MBbit-es kapcsolati sebesség, de a dedikáltan GIS feladatokkal ellátott szervezeti egységek irányában ez eléri az 1GBit-es sebességet. Az általános adatok központi elérését egy minden felhasználó által elérhető Windows alapú fájlszerver biztosítja.

Az Intézeti adatstruktúra fontos szereplője a földtani objektumok központi adatbázisa, a Geobank. Ez egy kiterjedt relációs adatbázis, mely tartalmaz adatokat földtani célú fúrások rétegsorairól, vízföldtani kutak idősorairól, de megtalálhatóak benne az egyes földtani egységek valamennyi nyilvántartott metaadata is. Ez a relációs adatbázis egy Microsoft SQL Server adatbázis kezelő segítségével elérhető a hálózaton belülről, így akár GIS alkalmazások is hozzá tudnak férni ezekhez az adatokhoz.

Az Intézeti GIS adattárolás egy tetemes része adatbázis környezetben történik, mely az első fejezetben bemutatott ArcSDE tárolási mechanizmussal történik, egy dedikált Microsoft SQL Server elem segítségével. A geometria egy bináris mezőben tárolódik, sdebinary formátumban. A hozzáférés szabályozása Windows felhasználók és felhasználói csoportok alapján történik, és ilyen formán rugalmasan kezelhető az adatokhoz való hozzáférés. Erre az adatbázis kezelőre csak kész, lezárt térbeli adatbázisok, illetve projekt eredmények kerülnek.

A térbeli feldolgozás és projektmunka során a szerkesztéshez használt adatok Personal, illetve File Geodatabase-ekben tárolódnak, melyek a korábban említett Windows fájlszerver megfelelő könyvtáraiból érhetőek el.

A GIS adatokhoz való hozzáféréshez minden felhasználó rendelkezik az ArcGIS szoftvereivel, melyeknek licenceit egy lebegő licences kiszolgáló rendszer osztja ki. A

térinformatikai adatokhoz való hozzáféréshez az ArcCatalog, míg a GIS szerkesztéshez és elemzéshez, illetve a kartográfiai munkához az ArcMap áll rendelkezésre a felhasználóknak.

11. ábra: Lezárt adatok mozgatása ArcSDE környezetbe

A térinformatikai adatok életciklusa a projekt kezdetétől indul, és a feldolgozás és szerkesztés során Personal, illetve File Geodatabase formában tárolódik. Az adatok a lezárást követően kerülnek ArcSDE környezetbe, ahonnan az adattartalom beemelése egy később időpontban is lehetséges.

Az Intézet webes térinformatikai szolgáltatásai egy-két kivételtől eltekintve ilyen lezárt adatállományokon alapszanak. A következő fejezetekben bemutatásra kerül a szolgáltatási háttér és a kétféle szolgáltatási módszer, az ArcGIS alapú, valamint az open-source eszközökön alapuló adatszolgáltatás.

III. Térképszolgáltatási környezetek kialakítása

Doktori kutatásom egyik fő célja az volt, hogy megvizsgáljam a nyílt forráskódú eszközök alkalmazhatóságát webes térinformatikai adatszolgáltatásokra. A kutatás során alkalmam volt párhuzamosan, azonos hardveres környezetben alapjaitól felépíteni egy ArcGIS szoftvereken, illetve egy open-source eszközökön alapuló rendszert. Ez lehetővé tette, hogy megvizsgáljam mindkettő jellegzetességei, és összehasonlítsam őket többféle szempontrendszer alapján.

III.1. A hardveres környezet

A disszertáció bevezető fejezetében tárgyalásra került a GIS szerverek hardveres igénye. Ennek megfelelően a környezet kialakításakor olyan hardveres eszközöket választottunk, melyek ezeknek a kívánalmaknak megfelelnek. A tervezési alapelvek tárgyalása után a kialakított hardveres háttér kerül bemutatásra.

III.1.1. Tervezési alapelvek

A GIS szerverek hardveres kialakítása alapvetően befolyásolja a későbbi teljesítményt, ezért rendkívüli módon fontos az alapos tervezésük. A kialakításkor a következő tényeket fontos figyelembe venni:

- szolgáltatni kívánt adatok típusa és mennyisége
- párhuzamosan szolgáltatott szervizek száma
- várható maximális egyidejű terhelés
- hálózati sávszélesség
- szerveren futó számításigényes feladatok futtatása

A fenti lista bármelyik elemének a figyelmen kívül hagyása a rendszer kedvezőtlen működését okozhatja, ezért érdemes ezeket mélyebb részletességgel is megvizsgálni.

Amennyiben előzetesen nem történik meg annak a részletes felmérése, hogy milyenek lesznek azok a típusú adatok, amelyeket a GIS szerver majd szolgáltat, úgy a szerver túlterhelése léphet fel problémaként. Így például a raszteres vagy vektoros adatszolgáltatások nemcsak formájukban különbözhetnek, hanem memória-felhasználásukban is. Míg a raszteres adathalmazok esetében a méret mellett a felbontás lehet a legfontosabb tényező, úgy a vektoros adatoknál mind az elemek száma, mind pedig az összetettsége döntő tényező lehet.

A szolgáltatni kívánt adatok minőségi jellemzői mellett figyelembe kell venni azok mennyiségét is. Mivel a GIS szerverek az egyes szolgáltatások hatékony működéséhez egy adott mennyiségű memória területet (RAM) dedikálnak, így fontos az, hogy becslés készüljön a szolgáltatások számát illetően is.

A várható maximális egyidejű terhelés az a tényező, amelyet talán a legnehezebb pontosan becsülni. A felhasználók "viselkedési szokásai" sokszor kiszámíthatatlanok, és képesek a legkevésbé várt területeken problémát okozni. Éppen ezért érdemes abból a feltételezésből kiindulni, hogy a szoftveres kialakítás, illetve a megfelelő jogosultsági szintek kiosztása a kiugró felhasználói tevékenységeket kiszűri, és így a "normál tevékenységgel" lehetséges a kalkuláció. (Ez a gyakorlatban azt jelentheti, hogy a felhasználó az adott jogosultsági szintjének megfelelő minimális tevékenységei jogkört kapja meg, és a szerver erőforrásaihoz való hozzáférést a lehető legnagyobb mértékben korlátozni szükséges.)

Amennyiben mind a szolgáltatások minősége és mennyisége, és mind a jogosultsági beállítások és a felhasználók megfelelő szoftveres "korlátozása" megtörténnek a GIS szerveren, egy tényező még mindig okozhat problémát: a szerver hálózati adottságai. Hiába bírja a szerver processzora és memóriája a szolgáltatásokat, hogyha túl sok adatot kell egy szűk csatornán közvetítenie a felhasználók irányába, akkor ennek előbb vagy utóbb a felhasználók látják kárát: hosszabb várakozási idővel kell, hogy számoljanak, és a szolgáltatások kiesését tapasztalhatják. (A hálózati megfontolásokról a következő fejezet ad részletesebb képet.)

A GIS szerverek hardveres terhelését befolyásolhatja az általuk elvégzendő számításigényes feladatok mennyisége is. Ezek lehetnek elsősorban a klasszikus adatszolgáltatások előállításához szükséges háttérfeladatok (például az egyes térképszolgáltatásokhoz szükséges raszteres piramisépítése), de a szerverre érkező adatbázis

lekérdezések is ide sorolhatóak (például vektoros elemek attribútum vagy hely alapján való leválogatása egy webes alkalmazáson keresztül). A maximális rendelkezésre állás biztosítása érdekében lehetséges ezeknek a feladatoknak egyrészről az időzítése, másrészről az átszervezése más szerverre.

A fent jelölt problémák kezelésére természetesen a legjobb a GIS szerver feladatainak és felhasználásának alapos tervezése, mely biztosítékot jelenthet a stabil működésre és a kedvező rendelkezésre állásra is. Azonban sokszor előfordul az a helyzet, hogy az kiterjedt tervezés ellenére is akadnak problémák a rendszerrel. Ennek oka sokszor az, hogy előre nem látható és jelentős mértékű igények merülnek fel a GIS szerverrel kapcsolatban. Ennek a megoldására többféle opció is kínálkozik.

12. ábra: Fizikai gép virtualizációjával létrejövő konfigurálható környezetek

A webes világban a hardveres tényezők kiküszöbölésére jó megoldás lehet a virtuális környezetek a használata, mely lehet lokális környezetben, vagy működhet felhő alapú szolgáltatás formájában. Ezeknek a módszereknek a lényege, hogy a GIS szerverhez lokális környezetben korlátolt, de felhő alapú felhasználás esetében szinte korlátlan mennyiségű erőforrás rendelhető hozzá. Így akár a processzor, akár a memória (RAM), akár a háttértároló (HDD) mennyisége nem megfelelő, úgy azt utólagosan lehetséges változtatni az igényeknek megfelelően.

Amennyiben nincsen mód a hardveres környezet virtuális konfigurációjára, úgy opcionálisan kialakíthatóak ún. klaszterezett megoldások. Ezeknek az alapelve, hogy elosztják a rendszer belépési pontjára nehezedő terhelést. Ez történhet úgy, hogy a rendszer egyes funkciókat ellátó elemei dedikált környezetbe kerülnek (vagyis az adattárolás, a webszerver, illetve a szorosan vett GIS szolgáltató egység), de történhet több azonos paraméterekkel rendelkező teljes értékű GIS szerver összefűzésével, és a belépési ponton (általában a webszerveren) alkalmazott terhelés-elosztó modul segítségé-

vel.

13. ábra: A terheléselosztás sematikus ábrája

Utóbbi esetében az egyes egységekre nehezedő nyomás annak a függvényében változik, hogy miképpen alakul a felhasználók által támasztott igény a GIS szerver szolgáltatásaira.

III.1.2. A kialakított hardveres környezet

Az előző fejezetben megfogalmazott alapelvek azt mutatják, hogy a környezet kialakításakor fontos figyelembe venni:

- a szolgáltatott adatok típusát (vektoros vs. raszteres)
- a szolgáltatott adatok mennyiségét
- az egyidejű maximális felhasználást
- a felhasználás módjait, típusát
- számításigényes feladatok futtatásának az igényét

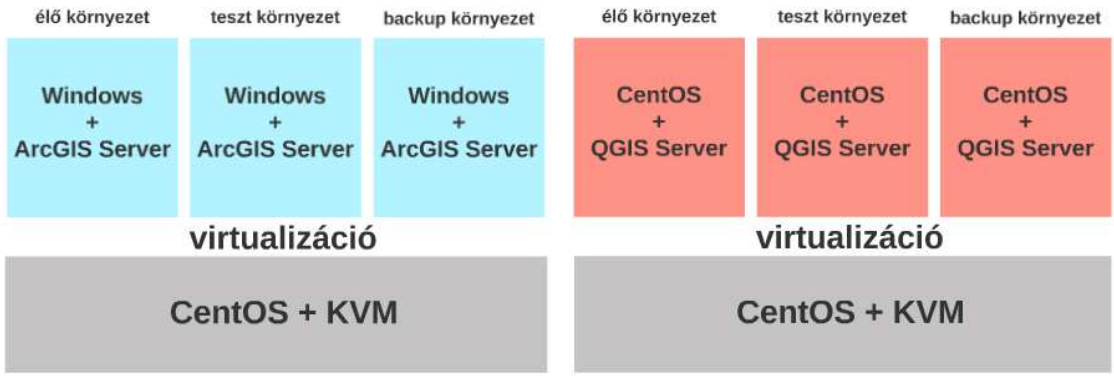
A fenti szempontokat figyelembe véve az MFGI által támasztott igények a következőképpen foglalhatóak össze:

- a szolgáltatni kívánt térinformatikai adatok jellegüket tekintve elsősorban vektorosak
- mennyiségüket jól illusztrálja, hogy a legtöbb tematika országos jellegű, a szolgáltatott tematikák száma 20 és 30 közötti
- az egyidejű használatot illetően nem jellemző a magas párhuzamos felhasználó szám
- a felhasználás típusa elsősorban az adatok statikus megjelenítésére irányul, ritkábban leválogatási feladatokat is magában foglalhat
- a szerverek legfőbb számításigényes feladatai a cache-elési feladatok

Mivel a feladatkörök változására való felkészültség fontos szempont volt a környezet kialakítása során, így a korábban említett rugalmasság eléréséhez vagy felhő alapú megoldásra, vagy egy fizikai szerverre volt szükség. Mivel a GIS szerverek paramétereihez illeszkedő felhő alapú szolgáltatások árban semmiképpen nem bizonyultak megfizethető megoldásnak, így ennek tudatában indult el a fizikai hardveres eszközök beszerzése, mely során két minden paraméterében megegyező IBM System x3650 M4 típusú szerver került az MFGI tulajdonába. A szerverek az alábbi főbb paraméterekkel rendelkeztek:

- 2×8 core Xeon 2 GHz CPU
- 64 GB RAM
- 16 TB HDD

A háttértárolók RAID10-es tömbökkel kerültek kialakításra, és ezen alapulva mindkettőn CentOS operációs rendszer lett telepítve. Ezekre azután KVM virtualizációs környezet került, mely mindkét környezet teszt és éles környezetét, valamint az éles rendszer folyamatosan felülíródó mentett állapotát tartalmazza.



14. ábra: A felépített hardveres környezetek és az alkalmazott GIS szerverek

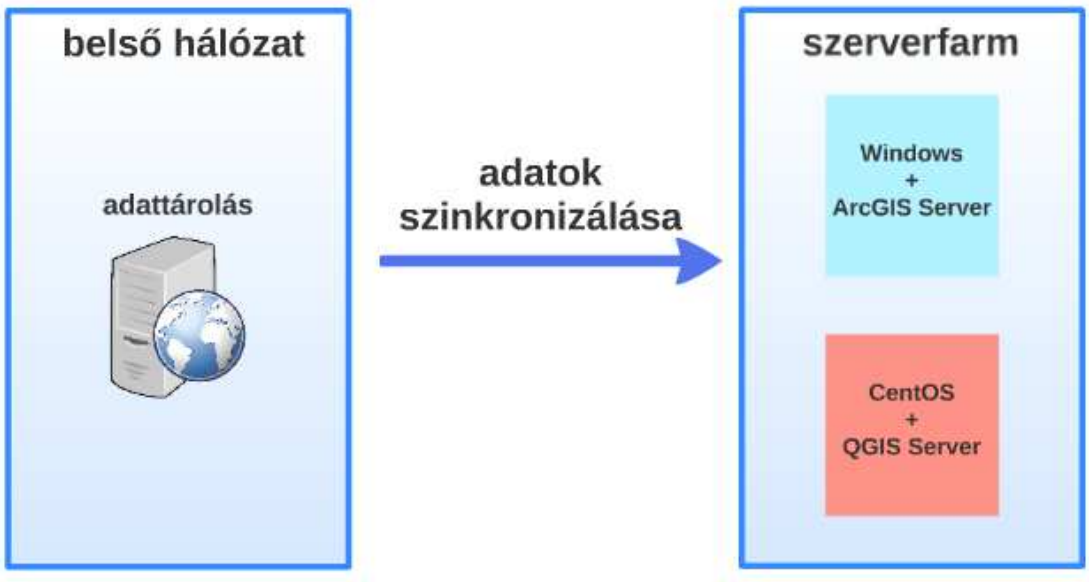
A virtuális környezetek kialakításának köszönhetően a rendszer nagyfokú paraméterezhetőséggel rendelkezik, hiszen szabadon változtatható az egyes gépekhez rendelt CPU erőforrás, vagyis a magok száma, de ugyanígy állítható az egyes gépek memória, illetve háttértároló igénye is.

III.2. Hálózati megfontolások

A két párhuzamos környezet kialakításakor fontos szempont volt, hogy a webes térképi adatszolgáltatás ne befolyásolja az Intézet általános webes adatforgalmát, vagyis hogy zavartalan legyen minden más internet felé való kapcsolódás. Ennek érdekében a két külön dedikált szerver környezet kialakítása az intézeti tűzfalon kívül, egy webes adatszolgáltató központban (szerverfarmon) történt.

Ennek legnagyobb előnye, hogy a szolgáltatás sávszélessége egy előre rögzített, garantált sebességgel történik, mely mind a letöltési, mind pedig a feltöltési sebességre vonatkozik (és ebben az esetben a feltöltési sebesség az, amely főként releváns, hiszen a párhuzamos felhasználók letöltése a szerver szempontjából feltöltésként értelmezhető).

A szerverfarmok másik nagy előnye a magas szintű rendelkezésre állás. Ez azt mutatja meg, hogy egy adott webes szolgáltatás az idő mekkora részében érhető el, legtöbbször egy százalékértékkel adják meg. Egy szerverfarmon ez a paraméter sokkal kedvezőbb lehet, hiszen esetükben sokkal nagyobb erőforrásokkal rendelkeznek váratlan események kezelésére, így például egy esetleges áramszünet idején is folyamatosan tudják biztosítani a szolgáltatást komoly kapacitással rendelkező generátorok segítségével, míg ez lokális környezetben a kisebb kapacitású szünetmentes egységek használatával kevésbé hatékony lehet.



15. ábra: A térképszolgáltatásért felelős szerverek elhelyezése

Emellett még fontos lehet az is, hogy egy szolgáltatónál elhelyezett architektúra egy sokkal izoláltabb adatkezelést tesz lehetővé, és így nem szükséges az esetleges bizalmas, belső használatú adatok kezelésére és védelmére többlet energiát fordítani.

A szolgáltatónál való elhelyezés azonban természetesen nem csak előnyökkel jár, hiszen a megfelelő sebesség biztosításához szükséges az adatok mozgatása, és ez természetesen lassabb lehet, mint egy belső környezetben. Ezen kívül szükséges lehet a belső, és a külső adattárolás összhangban tartása, hogy elkerülhetőek legyenek a tartalmi különbségekből adódó problémák. További nehézségeket okozhat, hogy a fizikailag a belső hálózattól távolabb eső szerverfarmokon egy esetleges leállás esetén az újraindítás csak személyes megjelenéssel, időkieséssel oldható meg.

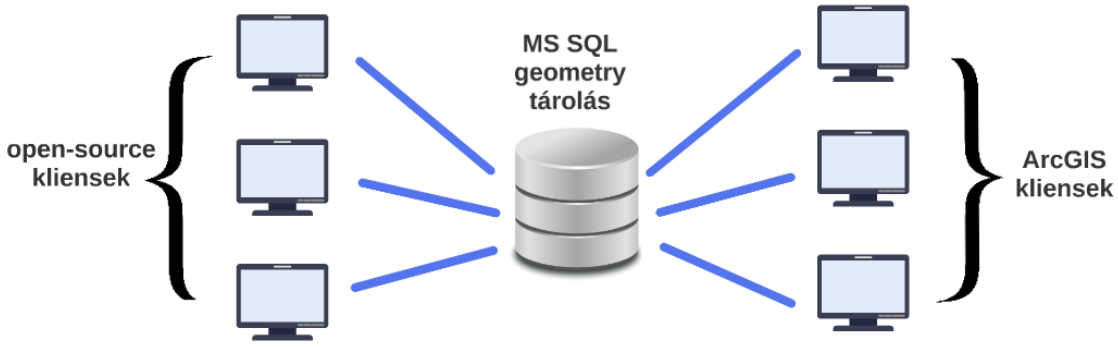
III.3. Adattárolási háttér

Az előző fejezetben vázolt hálózati megfontolásokból adódóan tehát látszik, hogy az egyes térképszolgáltatási feladatokat ellátó GIS szerverek elhelyezése külön hálózatban történt. Ennek az a következménye, hogy belső hálózatban tárolt, lezárt térinformatikai adatok mozgatását és szinkronizálását szükséges volt megoldani valamilyen formában. Ennek a módját és lehetőségeit azonban alapvetően befolyásolja a belső adattárolás módja, így most ez kerül bemutatásra részletesebben. A dolgozat második fejezetének végén nagy vonalakban már bemutatásra került a belső térinformatikai adattárolás módja, de mivel a webes térinformatikai adatszolgáltatás lezárt adatkörökön alapszik, ezért ebben a fejezetben hangsúlyozottan ennek a módszertanáról esik majd szó.

III.3.1. A geometriai adatok tárolására vonatkozó megfontolások

Mivel az Intézetben az alapértelmezett relációs adatbázis kezelő a Microsoft SQL Server, így az ArcSDE eszköz is ezt használja a térinformatikai adattároláshoz. Ennek megfelelően tehát két lehetőség adódik a geometriai adatok tárolására egy mezőben. Az egyik forma a Microsoft SQL Server natív geometria tárolása, mely egy geometry mezőtípust használ, a másik pedig az ArcSDE tárolási mechanizmusa, amely egy bináris típusú mezőt igényel, és sdebinary néven ismert.

Az geometriai információk adattárolási módszerének kiválasztásakor alapvetően két tényező játszik fontos szerepet. Az egyik az asztali kliensek esetében tapasztalható sebesség, vagyis az, hogy egy adott térinformatikai adatmennyiség megjelenítése az adatbázisból lekérve mennyi időt vesz igénybe a szoftvernek. A másik tényező az adatokhoz való hozzáférés lehetősége különféle más, nem ESRI alapú, vagy nyílt forráskódú szoftverekből.



16. ábra: Az ideális tárolás, egy értelmezhető formátum minden kliensnek

Az Intézeti szinten használt központi geometriai adattárolási forma kiválasztásakor fontos szempont volt, hogy az időben később kialakítandó nyílt forráskódú rendszer is ugyanabból az adatforrásból "dolgozhasson", mint amiből az ArcGIS kliensek és az ArcGIS for Server. Ennél fogva jó választásnak tűnhetett a Microsoft SQL Server natív geometria tárolása, hiszen használatával mind az ArcGIS termékeiből elérhető az adattartalom, mind pedig a nyílt forráskódú világ szoftvereiből (az ogr2ogr eszközön keresztül).

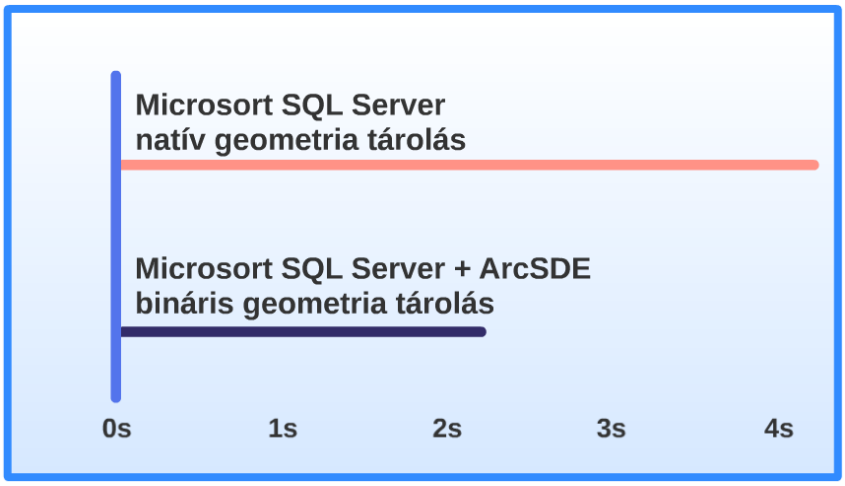
III.3.2. Tesztelési eredmények

Az elvi indokok tehát alapvetően az SQL Server natív geometria tárolása mellett szóltak, de mivel a kirajzolás sebessége egy fontos szempont a bonyolult geometriával rendelkező és olykor nagy adatsűrűséggel rendelkező földtani adatok esetében, így következett egy sebesség teszt ArcGIS környezetben, hiszen ez az Intézet alapértelmezett GIS és kartográfiai munkákra használt eszköze. Ennek során a teljes magyarországi felszíni földtani elemeket tartalmazó polygon állomány adattartalma került megjelenítésre. A minta állomány mind a natív geometriai, mind pedig sdebinary tárolási mód-

szerrel behívásra került. A teszt során azonosak voltak a környezeti jellemzők, így ugyanaz volt:

- az adattartalom
- az adatbázis kezelő
- a kliens gép
- a hálózati adottságok

Ezekre azért volt szükség, hogy semmiképpen ne befolyásolja az eredményt egyik tényező sem, tehát a sebesség vizsgálatakor ezek kizárható tényezőknek tekinthetőek legyenek. Az egyes kirajzolások emellett többször is mérésre kerültek, így egy átlagolt eredmény került rögzítésre.



17. ábra: Geometria tárolások kirajzolásának ideje

A sebességre vonatkozó teszt némiképpen meglepő eredményt hozott, hiszen az MS SQL Server natív geometria tárolásával megjelenített állomány kirajzolása szinte kétszer annyi időt vett igénybe, mint az ESRI alapú, sdebinary típusú geometria tárolást használó állományé. Ezzel bizonyításra került az első fejezetben megfogalmazott állítás, miszerint egy szoftverre optimalizált adattárolási formával az esetek döntő többségében nem veheti fel a versenyt valamilyen szabványos adattárolási struktúra.

Mivel az Intézetben irányelv, hogy a térinformatikai elemzéseket és kartográfiai szerkesztéseket ESRI környezetben szükséges végrehajtani, így a sebességre vonatkozó eredmények tükrében az sdebinary tárolási forma mellett kellett dönteni. Egy ilyen döntés alapvetően negatívan befolyásolhatja a nyílt forráskódú rendszerekkel való együttműködést, és némiképpen megnehezíti, de korántsem lehetetleníti el azt.

III.3.3. Az alkalmazott tárolási forma következményei

Az Intézet belső, központi adattárolása tehát Microsoft SQL Server környezetben, ArcSDE segítségével, és sdebinary geometriai adattárolással történik. Azonban ahogyan arról az előző fejezetben is szó esett, a webes térképi adatszolgáltatás nem közvetlenül a belső hálózatból történik, hanem két szolgáltatónál elhelyezett dedikált szerver segítségével, így az adatok kimásolásának és összhangban tartásának a lehetősége fontos eleme a rendszernek.

Mivel a központi adattárolás csak lezárt állományokat tartalmaz, így annak adott időközönként való másolása a "kinti" környezetbe jó megoldás lehet, ráadásul így csak a webes adatszolgáltatáshoz szükséges adattartalmat szükséges átmozgatni. Ez az ArcGIS alapú környezet esetében ennek az adattartalomnak az azonos formában történő kimásolását jelenti, míg a nyílt forráskódú eszközök esetében ez valamilyen adatmigrációs eljárás kidolgozását követeli meg.

IV. Kereskedelmi szoftverekkel kialakított WebGIS rendszer

Ebben a fejezetben az általam létrehozott, ESRI termékcsalád elemeivel kialakított rendszert fogom bemutatni, az adatháttértől egészen a kliens oldali megjelenítésig. Az ESRI termékcsalád asztali környezetben betöltött szerepe az MFGI-ben rendkívül jelentős, szinte kizárólagosnak mondható, így kézenfekvő megoldásnak tűnt elsőként ennek a szoftvercsaládnak a webes térképszolgáltatási rendszerét alkalmazni a feladatra.

A kialakításra kerülő rendszer adathátterét Microsoft SQL Server alapú ArcSDE adatbázisok biztosítják, melyek időszakosan kerülnek ki és frissülnek a belső hálózat adatai alapján. Erre az adattartalomra épülve a webes térinformatikai adatszolgáltatásról az ArcGIS Server gondoskodik. A kliens oldalon használt keretrendszer, a WebApp Builder for ArcGIS és az ArcGIS for Server között a Portal for ArcGIS teremti meg a kapcsolatot. A rendszer kialakításának az alapja Windows Server operációs rendszer, mely a korábban említett dedikált virtuális környezetben került kialakításra.

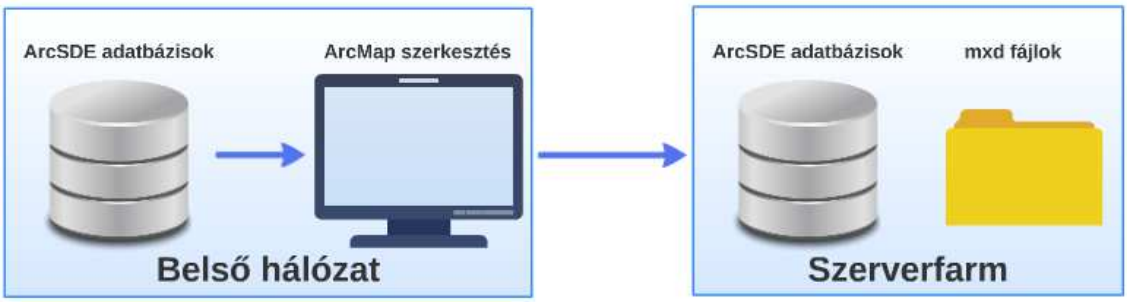
IV.1. Az ArcGIS for Server

Az ArcGIS for Server a piac egyik legjelentősebb és legkomplexebb szerver oldali komponense, mely a térinformatikai adatokkal kapcsolatos adatszolgáltatási feladatok széles spektrumát kínálja. Alkalmas mind raszteres, mind vektoros adatkörök webes szolgáltatására, de emellett használható még paraméterezhető GIS feladatok publikálására, (mely a WPS szervizhez hasonló szolgáltatás indítását teszi lehetővé) vagy akár geokódoló szolgáltatások indítására is. (A fejezetben természetesen elsősorban a webes térképszolgáltatási jellegzetességeiről lesz szó.)

Mivel a szoftver az ArcGIS termékcsalád (platform) szerves része, így jól tudja kezelni a kapcsolódó formátumokat mind adatszinten (különféle típusú geoadatbázisok), mind pedig elkészült térképi állományok szintjén (mxd kiterjesztésű állományok).

IV.1.1. A térképszolgáltatások kialakítása

A térképszolgáltatások kialakításának és publikálásának a menete a következőképpen alakul az ArcGIS for Server segítségével. Asztali környezetben elkészül egy térképes állomány mxd formátumban, mely adat szinten hivatkozza az MS SQL Server-en található ArcSDE adatbázisok valamely elemeit. Erre az adattartalomra építve ArcMap környezetben megtörténik a kartografálási folyamat, és a megfelelő jelkulcsi elemek alkalmazásával elkészül a kész digitális térkép.



18. ábra: Adattartalom és kész térképek másolása külső környezetbe

Ezután mind az adattartalom, mind pedig a térképes állomány, vagyis az mxd fájl kikerül a szerverfarmra, és az adatkapcsolatok beállítása után teljes értékűen használhatóvá válik. Az ArcGIS for Server asztali környezetbe való integrálhatóságának köszönhetően a térképszolgáltatások indítása egyenesen az ArcMap környezetből történhet, melynek során paraméterezhető formában publikálható a térképi állomány, és annak adatrétegei.

A publikálási folyamat során az alapadatok (név, elérés) mellet további adatok is beállíthatóak, így akár megadható az adott szolgáltatás timeout értékei (mennyi idő után ne válaszoljon a szerver, ha nem érhető el a szolgáltatás éppen), illetve az is, hogy milyen időközönként indítsa újra a szerver az adott térképszolgáltatás folyamatait.

Mivel az ArcGIS for Server térképszolgáltatásai nem OGC szabványos formában szolgáltatják az adatokat, hanem egy saját, egyedi formát használnak (a szolgáltatások REST, azaz Representational State Transfer protokoll segítségével érhetőek el), ezért lehetőség van a publikálási folyamat során OGC szabványos szolgáltatások párhuzamos indítására is, így lehetőség van az adott térképes állomány rétegein alapulva WMS, vagy WFS szolgáltatásokat is indítani. Szükséges azonban megemlíteni, hogy az ArcGIS

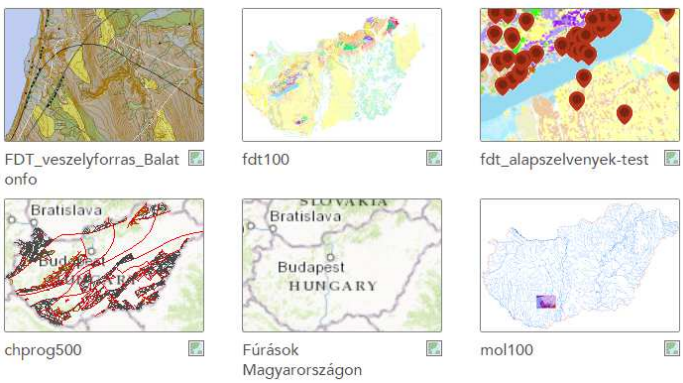
for Server segítségével publikált WMS és WFS szolgáltatások az OGC által előírt minimum szintet teljesítik, és ezek kiterjesztésére rendkívül szűk lehetőségekkel bírnak.

A REST protokoll segítségével a szolgáltatások elérése ezután mind asztali ArcGIS környezetből, mind pedig webes alkalmazások vagy mobil platformok irányából megoldható.

Ahogyan az a dolgozat bevezető fejezetében bemutatásra került, a térképszolgáltatások gyakorlatilag a kliens irányából érkező kérések alapján állítják elő a megfelelő térképet, raszteres formában. Azonban ez a módszer egyrészt meglehetősen számításigényes, és emellett még a hálózati forgalmat is jelentősen terheli. Ennek a jelenségnek a kiküszöbölésére általában olyan megoldást alkalmaznak, mely során az adott területet egy négyzetesen növekvő nagyságú rácshálóval fedik le, és minden egyes négyzetes elemre előállítják a megfelelő raszteres elemet. Az eljárás a csempézéshez hasonlít, így többek között ilyen néven is hivatkoznak rá. Az ArcGIS for Server egyik nagy előnye, hogy tartalmazza ezt a lehetőséget, és a szolgáltatás elindításakor akár automatikusan is elindítható a csempekészítés, és így a térképszolgáltatások megfelelő sebességgel elérhetőek kliens oldalról.

IV.1.2. A Portal for ArcGIS szerepe

Az ArcGIS for Server segítségével elkészült térképszolgáltatásokat használva alapvető funkcionalitású alkalmazásokat lehet létrehozni a Portal for ArcGIS segítségével, amelyekre webmap néven hivatkozik a szaknyelv. Ezeket a webmap-eket azután a WebApp Builder for ArcGIS segítségével lehetséges egy sokkal szélesebb funkcionalitású alkalmazássá alakítani.



19. ábra: A Portal for ArcGIS grafikus felületének részlete

Az eszköz ezen felül lehetőséget kínál egy webes környezetben, mintegy listaként megjeleníteni valamennyi elkészült webes alkalmazást.

IV.2. Az ArcGIS kliens oldali komponense

A dolgozat bevezető fejezetében bemutatott kliens oldali megjelenítés feladatára az ESRI termékcsalád az ArcGIS for Javascript API-t biztosítja, mely azzal a céllal jött létre, hogy az ArcGIS for Server segítségével létrehozott webszervizeken alapulva reszponzív, interaktív webes térképes alkalmazásokat lehessen létrehozni Javascript nyelv segítségével. Az API létrehozásakor egy olyan fejlesztési mechanizmust alkalmaztak, mely az egyébként objektumalapú, prototípus alapú Javascript nyelvet számos az objektum-orientált programnyelvekre jellemző tulajdonsággal ruházza fel, mint az egységbezárás, az öröklődés vagy a polimorfizmus.

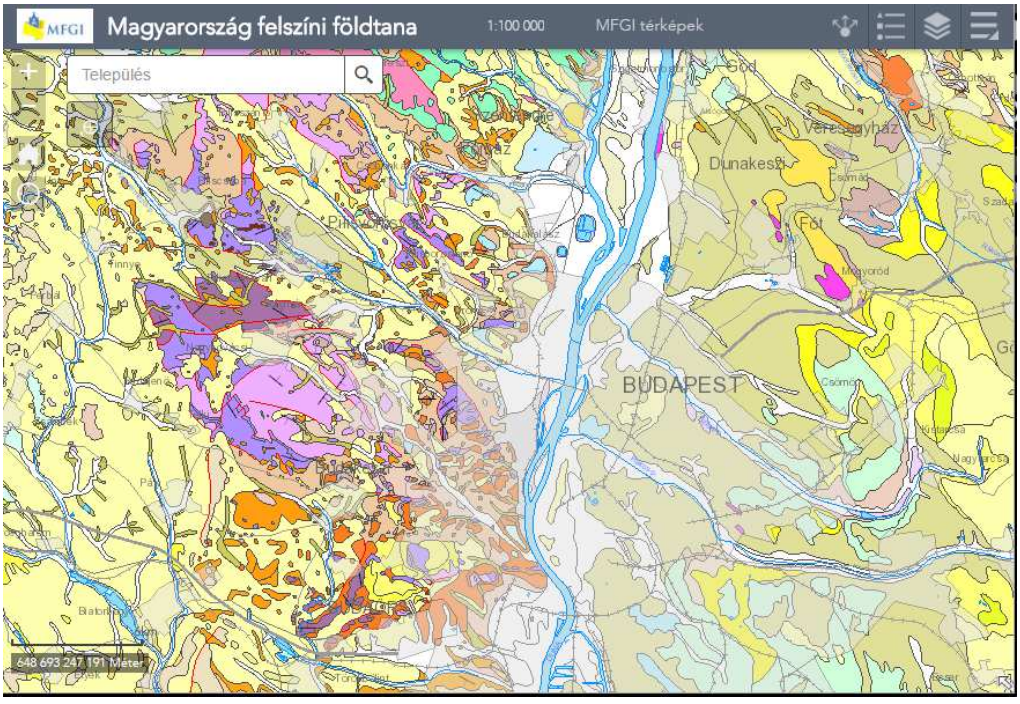
Ennek a megvalósításához a Dojo Toolkit nevű eszközt alkalmazták, melynek lényegi eleme, hogy a programkód egyes elemei modulárisan használhatóak fel, és így csökkenthető a kódrészletek ismétlése, és lehetséges az egyes "osztályok" tulajdonságainak az örökítése. Emellett a programkönyvtár képes az egyes böngészők közötti Javascript implementációk különbségeinek kiküszöbölésére, ezzel biztosítva azt, hogy az adott kódrészletek egyformán fussanak minden böngészőben. A Dojo Toolkit viszont a kódszervezési mechanizmusa mellett még egy fontos tulajdonsággal bír: tartalmaz ún. dijit elemeket, melyek olyan funkcionális programkód részletek, melyeknek egyben vizuális megjelenése is van. Ezek lehetnek akár egyszerű elemek, mint egy gomb, de akár komplexebbek, mint egy oszlopai alapján sorba rendezhető táblázat. Mindkettő esetében az egyes dijit-hez társul egyfajta működési elv, de emellett az ilyen elemeknek vizuális megjelenésük is van. Ez fejlesztői oldalról azért lehet előnyös, mert így nem szükséges minden egyes elemet HTML és CSS segítségével megalkotni, Javascript-tel a működéi hátterét lefejleszteni, hanem a már kész elemeket lehetséges testre szabni a Dojo Toolkit által biztosított öröklődés felhasználásával.

A Dojo Toolkit segítségével előállított már komplexebb funkcionalitással rendelkező elemeket widget-eknek hívják, és végső soron ezekkel lehetséges az egyes alkalmazások funkcionalitásának a kibővítése. A működésükre és a megjelenésükre vonatkozó kódrészlet egy-egy dedikált fájlban tárolódik (egy fő Javascript állomány, egy

fő HTML állomány), és egy konfigurációs fájl segítségével lehet testre szabni a paraméterezhető tulajdonságaikat. Az, hogy maga az alkalmazás tartalmazza-e az adott widget-et, azt az alkalmazáshoz tartozó fő konfigurációs állomány dönti el.

A Dojo segítségével kialakított alkalmazások felépítésének a megértéséhez szükséges lehet egy példa bemutatása. Legyen a cél egy adott alkalmazásban egy webszervizre alapozva térbeli objektumok leválogatásának a lehetősége egy attribútumuk alapján. Ennek a megvalósítása során első lépésben egy olyan widget-et szükséges fejleszteni, mely a felhasználói felületen tartalmaz egy szöveges kereső mezőt és egy gombot. Mindkettő elem egy dijit elem segítségével kerül bele a widget-be, melyeknek működését a dedikált Javascript állományban lehet megadni. Viszont lehetséges, hogy ezt a widget-et több alkalommal is szükséges a jövőben felhasználni, ezért indokolt lehet a változó paraméterek "kiemelése" a kódból, és a paraméter elhelyezése a dedikált konfigurációs állományban (ilyen lehet például a webszerviz URL-je). Az, hogy az alkalmazásba végül bekerül-e ez a widget, azt az alkalmazás fő konfigurációs állománya dönti el.

Erre a keretrendszerre épül egy további elem, az ArcGIS for WebApp Builder, mely egy webes felület segítségével teszi lehetővé az alkalmazások összeállítását, így akár programozási tudás nélkül is összeállíthatóak webes térképi alkalmazások.



20. ábra: WebApp Buider segítségével összeállított webes térképi alkalmazás

IV.3. Előnyök és hátrányok

A fejezetben bemutatásra került egy ArcGIS alapú WebGIS rendszer. Ennek legfontosabb elemei az adattárolásért felelős MS SQL Server alapokon nyugvó ArcSDE adatbázisok, az erre épülő ArcGIS Server, valamint a kliens oldali megjelenítésért felelős ArcGIS for Javascript API.

Mivel az egyes építőelemek egy szoftvergyártótól származnak, így elmondható, hogy az egyes komponensek jól kommunikálnak egymással, és az asztali környezetben kialakított, kartografált térképekből a térképszervizek kialakítása, majd azok weben való megjelenítése nem jelent problémát.

A rendszer adattárolásért felelős komponensét vizsgálva az látszik, hogy az sdebinary tárolási mechanizmus használata mindenképpen szükséges a gyors működéshez, mivel az MS SQL Server natív geometriai tárolása a tesztelések alapján roszszabbul teljesített. Ezt kétféle nézőpontból is lehet vizsgálni:

- egyrészt pozitívumként lehet értékelni azt, hogy az ESRI cég által kifejlesztett formátum gyorsabb működést eredményez
- másrészről viszont erősen sérül a más rendszerekkel való átjárhatóság követelménye, melynek fontossága napjainkra egyre fontosabb tényezővé válik

Utóbbi szempont elsősorban az ugyanazokon az adatokon alapuló nyílt forráskódú szoftverekkel való hozzáférést nehezíti meg, de erről az aspektusról a dolgozat következő nagy fejezetében megoldást kínálok majd.

Az ArcGIS for Server, mint a rendszer szerver oldali komponense az ESRI asztali szoftvereibe illeszkedve jól látja el a térképszolgáltatási feladatokat, és legnagyobb előnye, hogy az ArcMap-ben definiált jelkulcsi elemeket és stílusokat hatékonyan és változtatás nélkül örökíti a webes szolgáltatáshoz. Hiányossága viszont, hogy az OGC által előírt szabványos adatszolgáltatási struktúrákat (pl. WMS, WFS, WMTS) olykor nehezen lehet előállítani, és a kibővítésére vonatkozó lehetőségek is szűk körűek. Az ArcGIS for Server-hez kapcsolódó Portal for ArcGIS funkciója a rendszer szempontjából nem lenne feltétlenül szükséges, hiszen célja elsősorban gyorsan összeállítható alkalmazások lehetőségének a biztosítása, és erre a kliens oldali eszközök sokkal alkalmasabbak.

A kliens oldali eszközök, vagyis az ArcGIS for Javascript és a WebApp Builder for ArcGIS a Dojo Toolkit segítségével moduláris fejlesztést tesznek lehetővé, és használatukkal az ArcGIS térképszervizek hatékonyan építhetőek be kliens oldali alkalmazásokba. Ezek számos előnyös tulajdonsága a Dojo Toolkit-re vezethető vissza:

- böngésző implementációk különbségeinek kiküszöbölése
- logikus kódszervezési mechanizmus
- vizuális építőelemek megléte (dijit-ek)
- paraméterezhető, célfunkciót ellátó egységek (widget-ek) elkülönítése
- többnyelvűség támogatása

Ezek az előnyök mellett azonban természetesen hátrányokkal is járhat a keretrendszerek használata. Mivel alapvetően azzal a céllal jött létre, hogy akár nagyméretű kliens oldali alkalmazások fejlesztését is lehetővé tegye, így kisebb projektek esetében talán épp nehézséget jelent a programkód széttagoltsága, és nehézkes lehet akár egy működéshez kapcsolódó apró módosítás végrehajtása is. Ezen kívül a kódok debugolása (hibakeresése) is nehézkes lehet, és komoly energiákat felemészthet.

A kliens oldali komponensek további hátrányos tulajdonsága lehet még, hogy mivel az API fejlesztése az ESRI céghez köthető, ezért egyértelműen a saját szerver oldali szolgáltatásaira van optimalizálva, és így a támogatott formátumok és szolgáltatások köre viszonylag szűknek tekinthető. Emellett az egyes widget-ek funkcionalitása is többnyire az ArcGIS típusú webszervizekkel működik csak teljes értékűen.

Az adattárolás, a szerver és kliens oldali komponensek kedvező és hátrányos tulajdonságai mellett mindenképpen szükséges megemlíteni még egy tényezőt, amely befolyásolhatja a használatot, ez pedig a termék árazása. Habár ez nem tartozik a szűken vett szakmai szempontok közé, mégis megfontolandó, hogy az egyszeri megvásároláshoz, illetve szoftverkövetéshez szükséges anyagi ráfordítás mennyiben befolyásolhatja a más térinformatikai fejlesztések lehetőségét. A nyílt forráskódú eszközök esetében ezzel a tényezővel nem szükséges számolni, így a következő fejezetben egy ilyen szoftvereken alapuló rendszerről lesz szó.

V. Open-source eszközökkel kialakítható WebGIS rendszer

Kutatásom során alkalmam volt az ArcGIS szoftverekkel kialakított WebGIS rendszer hardveres paramétereivel megegyező környezetben egy nyílt forráskódú rendszert kialakítani. Ennek során megvizsgáltam az alkalmazható eszközöket, felderítettem a rendszer hiányzó elemeit, illetve kidolgoztam azokat az eljárásokat, amelyek az MFGI belső adatrendszerének a használatát lehetővé tették. A fejezetet az adattárolás lehetőségeivel és kihívásaival kezdem, ezt követően a szerver oldali komponensek alkalmazhatóságáról esik majd szó, és végül a fejezetet a lehetséges kliens oldali megoldásokkal zárom.

A nyílt forráskódú eszközökre és szabványokra épülő rendszer kidolgozását a dolgozat harmadik fejezetében ismertetett hardveres háttéren alapulva kezdtem meg. Ennek során egy virtuális gépen CentOS alapú operációs rendszert telepítettem és konfiguráltam, majd ezután teszteltem és kialakítottam az architektúra többi elemét is.

V.1. Az adatháttér

A dolgozat első fejezetében tárgyalt térinformatikai adattárolásra alkalmas nyílt forráskóddal rendelkező relációs adatbázis kezelők közül a rendszerhez a PostgreSQL-t választottam PostGIS kiterjesztéssel. A döntés okának a hátterét a következő részben mutatom be.

V.1.1. Adatbázis környezet kiválasztása

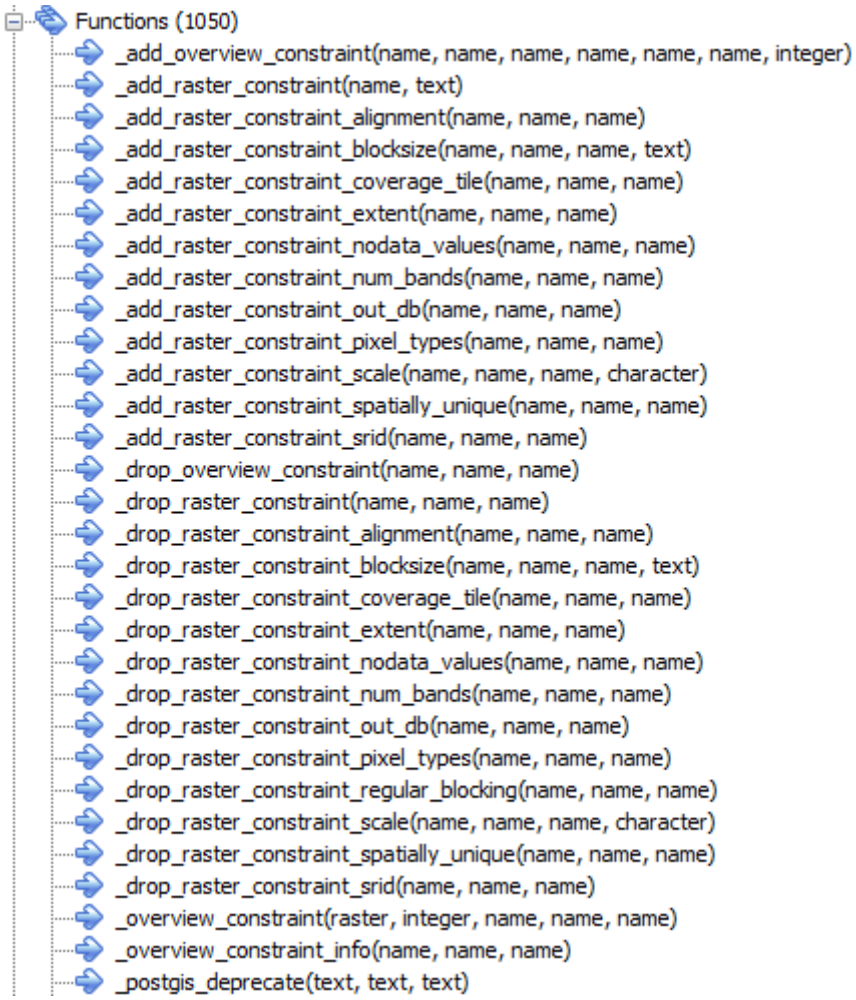
A PostGIS, mint kiterjesztés maga is számos előnyös tulajdonsággal rendelkezik, azonban ezek tárgyalása előtt szükséges felhívni a figyelmet a PostgreSQL adatbázis kezelő két kedvező tulajdonságára.

A PostgreSQL egyik legnagyobb különbsége a nyílt forráskódú adatbázis kezelők másik képviselőjével, a MySQL-el szemben, hogy alapvetően egy olyan objektum relációs adatbázis (Object-Relational Database Management System), amely a felhaszná-

lók, illetve fejlesztők adattípusaival szabadon bővíthető. Ez teszi lehetővé azt, hogy a térbeli adatkezelés problémájára fejlesztett PostGIS eszköz nem a PostgreSQL fejlesztői csoportja, hanem egy erre szakosodott másik fejlesztői csapat hatáskörébe tartozik. (Természetesen szoros az együttműködés, de egymástól akár függetlenül is folyhat a két komponens fejlesztése.)

A később tárgyalásra kerülő kvantitatív paraméterekkel rögzíthető előnyök mellett fontos még kiemelni a PostgreSQL környezet másik nagy előnyét, amely a fejlesztői háttérből eredeztethető. Mivel az évek során a PostgreSQL gyakorlatilag szabvánnyá és meghatározóvá vált a GIS szoftverek adattárolási hátttereként, így a nyílt forráskódú szoftverek programozói hatékonyan tudnak együttműködni az eszköz fejlesztésében. (Például a szerver oldali komponensek sokszínűségét szemlélve azonnal felfedezhető, hogy az egyes eszközök sokszor ugyanazokat a feladatokat látják el, és az összes belé-jük fordított energia gyakran elaprózódik ahelyett, hogy egy olyan szoftver alakulna ki, amely egyesíti valamennyi előnyös tulajdonságát.) A fejlesztői irányvonalakat az is pozitív irányban befolyásolja, hogy a PostgreSQL mögött nem található olyan üzleti érdekcsoport, amely például a MySQL mögött. (A MySQL felvásárlását követően az Oracle folyamatosan igyekszik a MySQL pozícióját gyengíteni, és a fejlesztést megnehezíteni (Williams 2012)).

A PostgreSQL előnyei mellett a PostGIS választásának az oka, hogy a többi szereplőhöz képest sokkal szélesebb funkciókészlettel rendelkezik. A nyílt forráskódú környezetbe telepített PostgreSQL a PostGIS kiterjesztéssel jelenleg 1050 függvényt tartalmaz, míg például a MySQL-ben elérhető függvények száma jelenleg 128, de az elérhető függvények számában a Microsoft SQL Server Spatial Extension-jét is felülmúlja. (Martinez 2015)



21. ábra: A PostgreSQL-ben elérhető függvények PostGIS kiterjesztéssel

Emellett a PostGIS a MySQL-el szemben képes akár háromdimenziós adatok tárolására is, míg a MySQL bemenő adatként elsősorban a Well-Known Text (WKT) és a Well-Known Binary (WKB) adattípusokat támogatja, melyek a 2 dimenziós adatok tárolására szolgálnak. (MySQL 2016).

Bár a háromdimenziós adatok kezeléséhez hasonlóan a raszteres adattípusok sem képezik részét a dolgozat témájának, mindenképpen fontos megemlíteni, hogy a PostGIS kiterjesztés segítségével akár ez is megoldható feladat, és ezek kezelésére számos adatbázis függvény áll rendelkezésre, így az ilyen adattípusokon alapuló elemzések akár további szoftverek alkalmazása nélkül is lehetségesek.

V.1.2. Adatbázis környezet kiválasztása

A telepített CentOS környezetben a Linux környezetre jellemző csomagkezelés a YUM program segítségével, parancssoros környezetből valósítható meg. Ennek megfe-

lelően installáltam és konfiguráltam a PostgreSQL adatbázis kezelőt a PostGIS kiterjesztéssel együtt.

```
yum install postgresql94-server postgresql94-contrib
/usr/pgsql-9.4/bin/postgresql94-setup initdb
systemctl enable postgresql-9.4.service
systemctl start postgresql-9.4.service
```

22. ábra: A PostgreSQL telepítése

Elsőként a PostgreSQL telepítése történik meg, mely során szükséges megadni az alapértelmezett adatbázist, valamint szükséges aktiválni, majd elindítani az adatbázis szerver szolgáltatásait, melyet azután a rendszer egy adott portjára hivatkozva lehetséges elérni (ez a PostgreSQL esetében alapértelmezetten az 5432-es port, de természetesen ez konfigurálható minden olyan portra, amely nincsen használatban más programok és eszközök által).

A PostgreSQL telepítése utána a hozzá illeszkedő PostGIS kiterjesztés telepítése következik, ezt a következő ábra mutatja.

```
yum install postgis2_94
```

23. ábra: A PostGIS telepítése

A telepítés után a legfontosabb lépés az adatbázis kezelő helyes konfigurációja. A beállításokat tartalmazó fájlok egyike a pg_hba.conf állomány, melyben megadhatóak az adatbázishoz való hozzáférések paraméterei többféleképpen is.

```
local      database  user  auth-method  [auth-options]
host       database  user  address  auth-method  [auth-options]
hostssl    database  user  address  auth-method  [auth-options]
hostnossl  database  user  address  auth-method  [auth-options]
host       database  user  IP-address  IP-mask  auth-method  [auth-options]
hostssl    database  user  IP-address  IP-mask  auth-method  [auth-options]
hostnossl  database  user  IP-address  IP-mask  auth-method  [auth-options]
```

24. ábra: Kliensek autentikációjának megadása

Ez a fajta definiálás akár adatbázis és felhasználó szinten megadható, valamint tartalmazhatja a távoli hozzáférés IP tartományát vagy egy dedikált IP címét, illetve az azonosítás módját is. (PostgreSQL 2016)

Az adatbázis szerver másik konfigurációs állománya a postgresql.conf állomány, melyben lehetséges megadni (zárójelben a vonatkozó paraméterek):

- a kapcsolódó távoli gépek IP címeit, mely lehet egy érték vagy egy vesszővel elválasztott lista (listen_adresses)
- a szolgáltatás által használt port száma (port)
- az adatbázishoz való kapcsolódások maximális száma (max_connections)

Ezeken a paramétereken túl természetesen még több tucat paraméter beállítására van lehetőség, melynek segítségével részletekbe menően állíthatóak a szerver erőforrás felhasználási és naplózási tulajdonságai, valamint a lekérdezésekre és replikációra vonatkozó valamennyi paraméter.

A konfiguráció záró lépéseként szükséges lehet a hálózaton kívülről való elérés biztosítása, melyet a tűzfalra vonatkozó szabályok módosításával lehet megoldani. Ennek során a szolgáltatási port számát szükséges kivételként felvenni a tűzfalat biztosító eszköz listájára.

A PostgreSQL adatbázis kezelő PostGIS kiterjesztéssel való telepítése és konfigurálása, valamint a tűzfalszabályok módosítása után a környezet készen áll arra, hogy új adatbázisokat lehessen kialakítani rajta, és segítségével WebGIS rendszerek hatékony alapját képezze. Mivel a Magyar Földtani és Geofizikai Intézetben a térinformatikai adattárolás nem ilyen környezetben történt, ezért szükséges volt a megfelelő migrációs folyamatok lépéseinek a kidolgozása, majd kialakítása. Ennek részleteiről a következő fejezet nyújt részletes képet.

V.1.3. Adatmigráció megoldása

A felépítendő WebGIS rendszer szempontjából gyakorlatilag adottságként jelentkezett az, hogy a publikálásra szánt térinformatikai adatok az MFGI belső hálózatában, Microsoft SQL Server környezetben, és az ArcSDE sdebinary szabványát használva voltak tárolva. Ez gyakorlatilag azt a feladatot generálta, hogy valahogyan meg kellett oldani az adatok migrálását egy zárt formátumból a PostgreSQL, illetve a PostGIS geometry típusú adattárolási formájába.

25. ábra: Az MS SQL ArcSDE és a PostGIS közötti migráció kérdése

A megoldásra elsőként az ogr2ogr nevű eszköz kínálkozott, mely a GDAL programkönyvtárral együtt az Open Geospatial Consortium egyik legfontosabb projektje. Az elnevezések olykor félrevezetőek, hiszen hivatalosan a GDAL elnevezést a kettő eszközre együttesen használják, de a felosztás a következőképpen alakul: míg a GDAL eszköz elsősorban a raszteres formátumokhoz kapcsolódó eljárásokat és konverziós lehetőségeket biztosítja, addig az ogr2ogr eszköz a térinformatika vektoros adattípusai között teremti meg a konverzió lehetőségét. Ez a két eszköz gyakorlatilag valamennyi nyílt forráskódú térinformatikai szoftver alapja, és a felhasználói felülettől függetlenül a háttérben az egyes szoftverek mind ezeknek a függvénykönyvtáraknak az eljárásait hívják meg.

Az ogr2ogr eszközt a GDAL-hoz hasonlóan C programnyelvben fejlesztik, és a nyílt szabványokra épülő formátumok mellett némely zárt formátumhoz való hozzáférést is biztosít azok API-ja segítségével. Ez azt jelenti, hogy ha nem nyilvános az adattárolás pontos formája és algoritmusa, akkor is lehetséges az adatok kinyerése bizonyos adattípusokból.

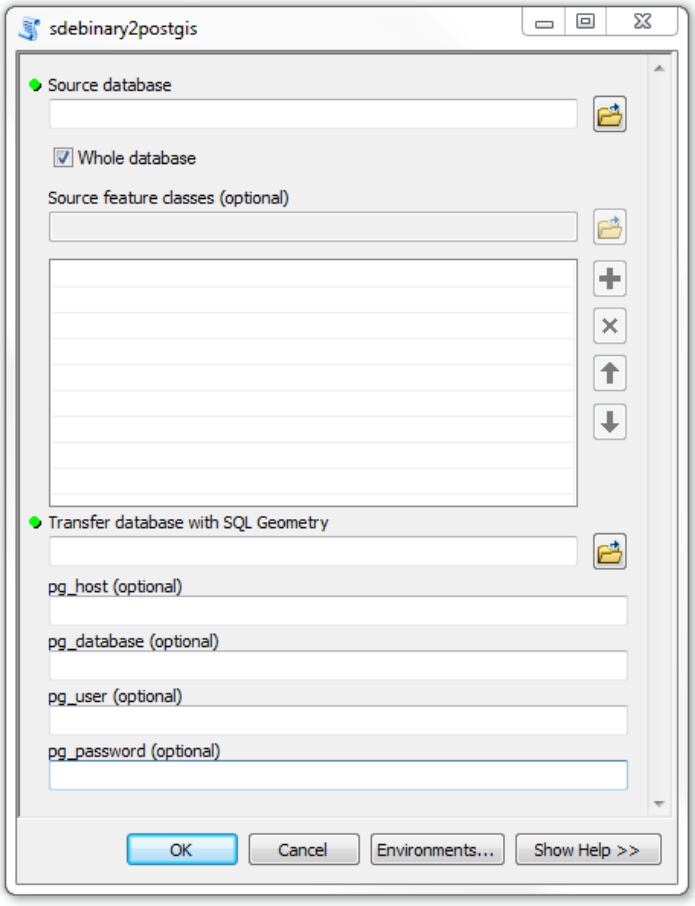
Korábban az ArcSDE adattárolási forma másként működött, mint a jelenlegi verziókkal. A rendszer szerves része volt egy alkalmazás szerver (application server), mely mintegy beékelődött az adatbázis, és a kliensek közé, így nem egy direkt kapcsolat alakult ki, hanem ez az alkalmazás-szerver fogadta a kéréseket, továbbította azokat az adatbázis irányába, majd az érkező választ visszaküldte a felhasználóhoz.

Ehhez az alkalmazás szerverhez az ESRI biztosított egy API-t melynek segítségével akár a klasszikus asztali kliens szoftverek (ArcCatalog, ArcMap) nélkül is lehetséges volt hozzáférni az adatokhoz. Az ogr2ogr függvénykönyvtár ezt a lehetőséget használta ki, és így biztosította az ArcSDE technológiával tárolt térinformatikai adatok elérését. Mivel ez a technológia már elavultnak számít, és az MFGI-ben sincsen jelen évek óta,

így a direkt konverzió nem volt lehetséges az ogr2ogr eszközzel, ennek a régebbi technológiának csak a migráció céljából való fenntartása pedig nem bizonyult egy rentábilis megoldásnak.

A fentiek alapján látható volt, hogy a migrációhoz szükséges még egy közbeépülő lépcső beiktatása. Ahogyan az a dolgozatban már többször is említésre került, az ArcSDE az MS SQL-el együtt kétféle geometriai adattárolást tesz lehetővé, az ArcSDE sdebinary-t és az MS SQL natív tárolását, a geometry-t. Mivel az ogr2ogr eszköz ismeri az MS SQL natív tárolási módját, ezért ez bizonyult a megoldás kulcsának.

Így a folyamat a következőképpen alakult. Az ArcGIS környezeten belül meg kellett oldani a térképszerkesztés miatt kedvezőbb sdebinary tárolásból való migrációt MS SQL geometry formátumba, majd ebből egy következő lépcsőben a PostGIS geometry mezőjébe való konverziót. (A harmadik fejezetben a Microsoft SQL Server natív geometria tárolása már ki lett zárva a kliensekben tapasztalható kedvezőtlen sebesség miatt.)

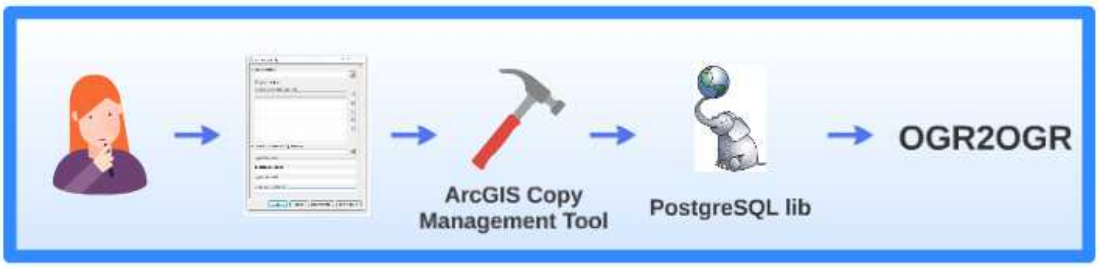


26. ábra: A fejlesztett eszköz felhasználói felülete

ArcGIS környezetben biztosított annak a lehetősége, hogy a felhasználók, illetve fejlesztők olyan eszközöket, más néven Tool-okat fejleszthetnek, amelyeket aztán az asztali kliens szoftverekből (ArcCatalog, ArcMap) lehetséges meghívni akár egy gombnyomás segítségével. Ezek az eszközök általában egy felhasználói felületből, valamint az alkalmazás logikáját tartalmazó kódból állnak össze. A felhasználói felület funkciója jellemzően az eszköz végrehajtásakor változóként jelentkező paraméterek a bekérése, a kód pedig a különféle API-k meghívásának segítségével a biztosított paraméterekkel hajtja végre a programozott funkcionalitást.

Az adatkonverzió automatizálására kifejlesztettem egy olyan ArcGIS Script Tool-t, mely a következőképpen működik:

- a felhasználótól egy grafikus felület segítségével bekéri az MS SQL és a PostgreSQL kapcsolódási paramétereit, valamint a migrálni kívánt adatbázis nevét
- Ezek után az sdebinary formátumban tárolt, megadott adatbázis valamennyi vektoros tábláját (feature class-át) átmásolja egy ideiglenes, MS SQL geometry adattárolású adatbázisba az ArcGIS Copy Management Tool függvényének a meghívásával
- az eszköz ezután a forrás adatbázis nevével megegyező adatbázist hoz létre a PostgreSQL API segítségével, és aktiválja a PostGIS kiterjesztést
- Ezek után az ogr2ogr eszköz eljárásainak a segítségével átmásolja az adatokat PostGIS környezetbe, majd létrehozza a szükséges térbeli indexeket
- Utolsó lépésként kitörli az ideiglenes, MS SQL geometry adattárolással rendelkező köztes táblákat



27. ábra: Az adatbázis konverzió folyamata

Az eszköz kifejlesztéséhez Python nyelvet használtam, és a kódban az ArcGIS, a PostgreSQL és az ogr2ogr API-kat, illetve könyvtárakat hívtam meg. Az eszköz segítségével lehetővé vált valamennyi sdebinary adattárolást alkalmazó térbeli adatbázis migrálása egyenként, egy lépésben, az ArcGIS felületről PostgreSQL, illetve PostGIS formátumba.

Az eszköz kialakítása során úgy alakítottam ki a háttérben futó kódot, hogy az a felhasználói felületről paraméterezhető legyen. A kód azonban könnyen alakítható olyan formába, hogy azt akár ütemezett feladatként lehessen futtatni, ezáltal biztosítva az adatok rendszeres időközönként való szinkronizálását. Mivel az elvárások között nem szerepel a kétirányú szinkronizálás (vagyis a PostGIS oldalról nem történik változtatás az adattartalomban), így a migráció során valamennyi adattábla felülíródik.

A CentOS alapon kialakított PostgreSQL és PostGIS adatbázis környezet megteremtette az alapot egy WebGIS rendszer adattárolási komponenséhez. Az adatok szinkronban tartása azonban nem lett volna lehetséges az általam kidolgozott adatmigrációs eljárás nélkül, melynek segítségével így lehetővé vált az intézeti térinformatikai adatokhoz való hozzáférés szoftverfüggetlenül, vagyis mind ArcGIS oldalról, mint pedig valamennyi open-source eszközből. A tervezett rendszer kialakításához ezután a WebGIS rendszerek egyik legsarkalatosabb pontját, a szerver oldali eszközt volt szükséges megválasztani, telepíteni és konfigurálni.

V.2. A szerver oldali komponens

A nyílt forráskódú térinformatikai eszközök között talán a legnagyobb sokféleséget a szerver oldali adatszolgáltatásra alkalmas eszközök mutatnak. Ezek feladata a meglévő adatokból, a megfelelő kartográfiai szabályok alkalmazásával minél gyorsabb szebb és sokoldalúbb térképszolgáltatásokat előállítani. Könnyen belátható, hogy erre a feladatra a sokféle szempontrendszer alapján több megoldás is kínálkozik, és a képet még tovább árnyalhatja, hogy a megvalósításhoz használt programnyelv is különbözhet.

28. ábra: Open-source szerver oldali megoldások

Az adattárolási fejezetben már szó esett arról, hogy a PostgreSQL és a PostGIS vezető és egyértelmű szerepe mellett a szerver oldali komponensek jelentősen nagyobb sokféleséget mutatnak. A fenti ábrán látható négy eszköz talán a legnagyobb szereplőket mutatja, de ezek mellett még számos más, kisebb funkciókészletű eszköz érhető el. Ezek mind a fejlesztés kezdetének időpontjában, mind a használt technológiákban és programnyelvekben, mind pedig természetesen a fejlesztői csapatban különböznek.

Ahogyan ezek a fejlesztések időben és térben is izoláltan haladtak évekig, úgy vonzották be a támogatókat és felhasználókat. Az évek során azonban egyre inkább körvonalazódott az a jelenség, hogy valójában mindegyiknek vannak erősségei, de talán más és más ponton. Mikor azonban ezt belátták az egyes fejlesztői csoportok, és egyértelművé vált, hogy sokkal "jövedelmezőbb" lenne a felhasználók szempontjából ezeknek a fejlesztéseknek az összevonása, addigra már annyi befektetett energia és felhasználó állt az egyes szoftverek mögött, hogy ezek egyesülése nem történt meg, és várhatóan még egy ideig nem is fog megtörténni.

V.2.1. Szerver oldali eszköz kiválasztása

Felmerül a kérdés, hogy a sokféle eszköz közül hogyan lehetséges mégis választani. A disszertáció bevezető fejezetében már tárgyalásra került az a szempontrendszer, amely alapján egy adott feladatra, illetve körülményre ki lehet választani a megfe-

lelő térképszolgáltatás előállítására alkalmas eszközt. Ezek a szempontok a következőek lehetnek:

- a teljesítmény, mely általában az egyes szolgáltatások kvantitatív jellemzőire vonatkozik
- az eszköz funkciókészlete
- az eszköz felhasználói felülete

Ideális esetben a fenti három feltétel egyszerre teljesülne, vagyis létezne olyan eszköz, ami nemcsak gyors sebességgel, de széles funkciókészlettel és felhasználóbarát felülettel rendelkezne. Mivel talán kijelenthető, hogy ilyen eszköz jelenleg nem létezik, így a felhasználás célja és módja szerint szükséges megválasztani a legalkalmasabb eszközt.

29. ábra: A szerver oldali komponens választásának szempontjai

A Magyar Földtani és Geofizikai Intézet szempontjából a térképszolgáltatások sebessége nem a legfontosabb követelmény, de annál fontosabb a széles funkciókészlet és a sokrétű lehetőséget biztosító felhasználói felület. Ezeknek a szempontoknak a figyelembe vételével a meglévő eszközök közül a Geoserver és a QGIS Server került vizsgálatra.

A dolgozat második fejezetében bemutatásra kerültek a földtani térképek jellegzetességei kartográfiai szempontból. Mivel az egyes földtani előfordulások elkülönítése sokszor komplex tematikus kartográfiai eszközöket kíván, ezért a felmerülő két eszköz közötti döntést az befolyásolta elsősorban, hogy ennek a feladatnak hogyan tudnak eleget tenni.

A Geoserver a térképszolgáltatások stílusainak a megadásához többféle megoldást is kínál. Ilyen az OGC szabványra épülő XML alapú SLD (Styled Layer Description) fájl vagy a Geoserver által kifejlesztett CSS szabványra építő megoldás. Ezek gyakorlatilag olyan leíró állományok, amelyek tartalmazzák az egyes rétegek megjelenésére vonatkozó szabályrendszert. A legnagyobb probléma viszont az velük, hogy nincsen olyan vizuális tervező eszköz, amellyel ezeket elő lehetne állítani, így egy bonyolultabb jelkulccsal rendelkező térkép publikálása rendkívül sok manuális munkát igényelhet, amely a gyakorlatban hosszú és sokszor átláthatatlan XML fájlok szerkesztésével járhat, és a kívánt eredmény elmaradását akár egy elgépelt karakter is okozhatja.

```
<FeatureTypeStyle>
  <Rule>
    <PolygonSymbolizer>
      <Fill>
        <CssParameter name="fill">#40FF40</CssParameter>
      </Fill>
      <Stroke>
        <CssParameter name="stroke">#FFFFFF</CssParameter>
        <CssParameter name="stroke-width">2</CssParameter>
      </Stroke>
    </PolygonSymbolizer>
    <TextSymbolizer>
      <Label>
        <ogc:PropertyName>name</ogc:PropertyName>
      </Label>
      <Halo>
        <Radius>3</Radius>
        <Fill>
          <CssParameter name="fill">#FFFFFF</CssParameter>
        </Fill>
      </Halo>
    </TextSymbolizer>
  </Rule>
</FeatureTypeStyle>
```

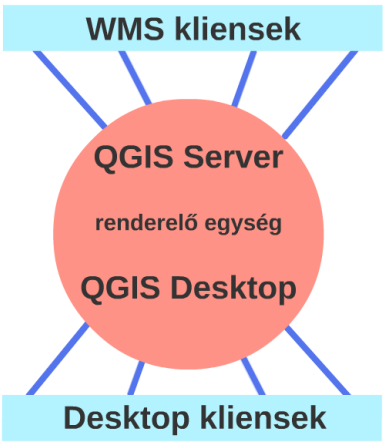
30. ábra: Egyszerű polygon feliratozás megadása Geoserver segítségével

Az idők folyamán több olyan kísérlet is történt, amely valamilyen asztali szoftverben történő "kartografálás" után tette lehetővé az ezekbe a formátumokba való konverziókat, egy export segítségével. A tapasztalatok azonban azt mutatták, hogy ezek az exportok egyrészt a séma definíciójának a különböző jellege miatt nem hozták meg a várt eredményeket, de előfordult az is, hogy egy "bonyolultabb" kartográfiai megoldás (például sraffozás és színezés együttes használata polygon kitöltésére) okozott gondot az átörökítés során.

A fenti szempontokat összefoglalva megállapítható, hogy a Geoserver nem alkalmas egy asztali környezetben elkészült kész, kartografált állománynak a weben való

szolgáltatására, hacsak a megjelenítés igénye nem szorítkozik egyszerű megoldásokra, vagy ha az adatgazdák készek a manuális munka alkalmazására. Mivel a földtani térképek sokszor bonyolultabb ábrázolásokat használnak, ezért kijelenthető az, hogy a Geoserver nem használható hatékonyan erre a feladatra, melynek eredendő oka abban keresendő, hogy a térképek megjelenítéséhez használt renderelő, vagyis megjelenítő motorja egyedi, és nincsen kapcsolatban semmilyen asztali térinformatikai szoftverrel.

A másik tárgyalásra kerülő eszköz, a QGIS Server talán pont ebből a szempontból rendelkezik sokkal kedvezőbb tulajdonságokkal, hiszen szorosan kapcsolódik a nyílt forráskódú térinformatikai világ elsődleges szoftveréhez, a QGIS Desktop-hoz, és az abban definiált stílusokat képes átörökíteni a webes térképszolgáltatásaiba.



30. ábra: A QGIS eszközök közös renderelő egysége

Ennek az oka, hogy a webes térképi adatszolgáltatás során ugyanazt a renderelő motort használja, mint az asztali GIS szoftver, a QGIS Desktop.

A QGIS asztali szoftver sikerét, és rohamos elterjedését kihasználva a projekt fejlesztői úgy látták, hogy érdemes a szerver oldali komponensek világában is megjelenni az eszközzel. Egy főként C++-ban programozott eszközt fejlesztettek ki, amely az asztali szoftverben elérhető stílusokat és megjelenítési szabályokat használva közelítette meg a webes megjelenítés lehetőségét. Konfigurálhatóságát, illetve a webes térinformatikai adatszolgáltatások indítását az asztali szoftverből tették elérhetővé.

Az eszköz segítségével ez alapján elméleti szinten lehetővé válik a földtani térképek asztali környezetben való kartografálása, majd weben való publikálása, akár bonyolultabb térképszerkesztési eljárások alkalmazását követően is.

Ez természetesen egy ígéretes eljárás, azonban az ArcGIS környezetben kialakított több tucat térkép földtani jelkulcsát legalább részben szükséges a QGIS környezetbe örökíteni, hogy azután majd térképszolgáltatásokat lehessen indítani belőlük a QGIS Server segítségével.

V.2.2. Jelkulcsok migrációja nyílt környezetbe

A földtani térképek talán legfontosabb és legbonyolultabb eleme a földtani képződmények elterjedését tartalmazó térinformatikai réteg (feature class, vagy tábla). A második fejezetben bemutatásra kerültek a térképszerkesztés során alkalmazott földtani tematikus kartográfiai megoldások, melyek közül mindnek elhagyhatatlan eleme a földtani képződményekre vonatkozó színezés.

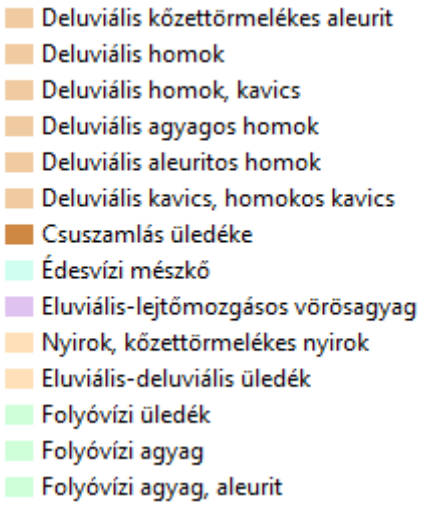
A színezés mellett akár valamilyen geometriai kitöltés alkalmazása is indokolt lehet némely földtani téma esetében, de ezek migrálása semmiképpen nem automatizálható, hiszen az egyes szoftverekben más és más formában és palettával érhetőek el. Az egyes földtani elterjedésekre vonatkozó megírások migrálása szintén nem képezheti a migrálás érdemi részét, hiszen az egyes térbeli geometriai elemek feliratozása minden szoftverben elérhető valamilyen formában.

A fentiek tudatában kijelenthető, hogy a földtani térkép jelkulcsi elemeinek a migrálásakor a legfontosabb feladat a felületi színezés szabályainak az átörökítése, hiszen a jelkulcs többi eleme csak manuális utómunkával oldható meg teljes mértékben. Az általam kidolgozott eljárás tehát a geológiai térképeken ábrázolt földtani elterjedés színkulcsának a migrációját foglalja magában.

Mivel az ArcGIS szoftver semmilyen API segítségével nem teszi lehetővé egy adott réteg attribútum listájához tartozó jelkulcsának exportálását (a földtani tematika fő eleme, a földtani felületi elemek ábrázolása a fentiek szerint így történik), ezért egy olyan eljárást fejlesztettem ki, amely az ArcGIS egy exportálható formátumából nyeri ki ezeket az információkat.

OBJECTID *	geo_ndx	centroid_tipus	GEO_ID	GEO_ID_txt	ID	GEO	NEV
49	pd_Qp3-h	2	2790	2790	279	pd_Qp3-h	Proluviális-deluviális üledék
50	fd_Qh_k	2	1762	1762	176	fd_Qh_k	Folyóvízi-deluviális kavics, homokos kavics
51	pd_Qp3-h	2	2790	2790	279	pd_Qp3-h	Proluviális-deluviális üledék
52	d_Qp3-h	2	801	801	801	d_Qp3-h	Deluviális üledék
53	pd_Qp3-h	2	2790	2790	279	pd_Qp3-h	Proluviális-deluviális üledék
54	b_Qh_al	2	999	999	999	b_Qh_al	Mocsári aleurit
55	e_Qp3_bl	2	3078	3078	307	e_Qp3_bl	Barna lösz
56	pd_Qp3-h	2	2790	2790	279	pd_Qp3-h	Proluviális-deluviális üledék
57	fT3	2	2302	2302	230	fT3	Fődolomit Formáció
58	f_Qh1_al	2	3276	3276	327	f_Qh1_al	Folyóvízi aleurit
59	f_Qh2_k	2	3300	3300	330	f_Qh2_k	Folyóvízi kavics, homokos kavics
60	pd_Qp3-h	2	2790	2790	279	pd_Qp3-h	Proluviális-deluviális üledék
61	f_Qh2_a	2	3284	3284	328	f_Qh2_a	Folyóvízi agyag
62	lb_Qh	2	1741	1741	174	lb_Qh	Tavi-mocsári üledék
63	f_Qp3_k	2	3358	3358	335	f_Qp3_k	Folyóvízi kavics, homokos kavics
64	soPa2	2	59	59	59	soM3	Somlói Formáció
65	f_Qp3_k	2	3358	3358	335	f_Qp3_k	Folyóvízi kavics, homokos kavics

31. ábra: Földtani elemeket tartalmazó feature class ArcGIS környezetben



32. ábra: Térképi megjelenítés ArcGIS-ben attribútumok alapján

Az ArcGIS for Server korábbi verziói egy msd nevű formátumot használtak a webes publikálás folyamatának a során. (Mára ez már csak ún. legacy formátum, ami azt jelenti, hogy csak azért biztosítják még ezt, hogy a korábbi verziókkal rendelkező felhasználók tudják használni.) Ez a fájl gyakorlatilag egy olyan tömörített állomány, amely tartalmazza valamennyi térképi réteg stílusának leírását, külön XML állományok formájában. (Amennyiben az mxd állomány tehát 10 réteget tartalmaz, akkor 10 XML állomány jön létre, melyek elnevezése igazodik az mxd fájl rétegeinek az elnevezéséhez, és mindegyik az adott rétegre vonatkozó megjelenítési szabályokat és információkat tartalmazza.)

Minden egyes rétegre vonatkozó állományban valamennyi stílusinformáció megtalálható az adott geometriai elemek körvonalától, kitöltésétől vagy pontszerű jelétől egészen a méretarányhoz kötött megjelenésig. Így a földtani elterjedést tartalmazó réteg megjelenítésére vonatkozó szabályok is kinyerhetőek, amelynek egyes

elemei attribútumai alapján rendelkeznek különféle színnel, ahogyan ezt a fenti két ábra is mutatja.

Ennek a rétegekre bontott XML állományoknak az elérhetőségével köszönhetően egy programmal végigiterálva az állományokon kinyerhetők az attribútum alapú kartografálásra vonatkozó információk.

```
<ICIMUniqueValueClass xsi:type="typens:CIMUniqueValueClass">
    <Description/>
    <Editable>true</Editable>
    <Label>tektonika</Label>
    <Symbol xsi:type="typens:CIMSymbolReference">
        <StylePath/>
        <Symbol xsi:type="typens:CIMLineSymbol">
            <LineDashEnding>NoConstraint</LineDashEnding>
            <Effects xsi:type="typens:ArrayOfICIMGeometricEffect"/>
            <SymbolLayers xsi:type="typens:ArrayOfICIMSymbolLayer">
                <ICIMSymbolLayer xsi:type="typens:CIMFilledStroke">
                    <JoinStyle>Round</JoinStyle>
                    <MiterLimit>10</MiterLimit>
                    <Alignment>Center</Alignment>
                    <PatternFollowsStroke>true</PatternFollowsStroke>
                    <Effects xsi:type="typens:ArrayOfICIMGeometricEffect"/>
                    <Enable>true</Enable>
                    <Name/>
                    <ColorLocked>false</ColorLocked>
                    <Width>1</Width>
                    <Pattern xsi:type="typens:CIMSolidPattern">
                        <Color xsi:type="typens:CIMRGBColor">
                            <R>250</R>
                            <G>52</G>
                            <B>17</B>
                            <Alpha>100</Alpha>
                        </Color>
                    </Pattern>
                    <CapStyle>Butt</CapStyle>
                </ICIMSymbolLayer>
            </SymbolLayers>
        </Symbol>
    </Symbol>
    <Values xsi:type="typens:ArrayOfICIMUniqueValue">
        <ICIMUniqueValue xsi:type="typens:CIMUniqueValue">
            <FieldValues xsi:type="typens:ArrayOfString">
                <String>11</String>
            </FieldValues>
        </ICIMUniqueValue>
    </Values>
    <Visible>true</Visible>
</ICIMUniqueValueClass>
```

33. ábra: Msd fájl kibontása után keletkező XML fájl részlete

A legfontosabb, földtani elterjedésre vonatkozó színkulcsi elemekhez tartozó színeket az alkalmazott színmodell formájában lehetséges kinyerni; így az RGB, a CMYK, vagy HSV modellhez tartozó számértékek rögzítésre kerülhetnek egy listában.

Amennyiben sikerül ArcGIS környezetből az msd fájl segítségével kinyerni egy adott rétegre vonatkozó stílust, úgy a következő lépés ennek az átörökítése QGIS környezetbe. A QGIS Desktop-ban lehetséges egy réteg stílusát egy beépülő fájl segítségével megadni, mely egy szintén XML típusú fájl, és a qml kiterjesztéssel rendelkezik. A feladat tehát a két XML fájl közötti konverzió megoldása, valamilyen programozási eszköz segítségével.

```
<symbol alpha="1" name="3" type="fill">
  <layer class="SimpleFill" locked="0" pass="0">
    <prop k="border_width_unit" v="MM"/>
    <prop k="color" v="224,224,224,255"/>
    <prop k="color_border" v="110,110,110,255"/>
    <prop k="offset" v="0,0"/>
    <prop k="offset unit" v="MM"/>
    <prop k="style" v="solid"/>
    <prop k="style_border" v="no"/>
    <prop k="width_border" v="0.0"/>
  </layer>
</symbol>
<symbol alpha="1" name="4" type="fill">
  <layer class="SimpleFill" locked="0" pass="0">
    <prop k="border_width_unit" v="MM"/>
    <prop k="color" v="224,224,224,255"/>
    <prop k="color_border" v="110,110,110,255"/>
    <prop k="offset" v="0,0"/>
    <prop k="offset unit" v="MM"/>
    <prop k="style" v="solid"/>
    <prop k="style_border" v="no"/>
    <prop k="width_border" v="0.0"/>
  </layer>
</symbol>
```

34. ábra: QML fájl részlete

A kétféle XML struktúra alapján kifejlesztettem egy módszertant, amely alapján lehetséges a földtani elterjedésre vonatkozó színkulcsok migrációja. Ennek lépései a következőek:

- az alkalmazás egy mxd állományból generál egy msd fájlt a ConvertToMSD ArcGIS Tool segítségével
- átnevezés után kibontásra kerül az msd fájl
- a rétegeken való iterálás során kiolvasásra kerülnek a színkulcsra vonatkozó információk
- a QGIS stílusoknak való megfeleltetéssel generálódik egy qml kiterjesztésű fájl, melyet később lehetséges használni

Az eszköz létrehozása Python nyelv segítségével történt meg, melynek kialakításában rajtam kívül az MFGI Geoinformatikai Főosztályáról Pogácsás Réka, valamint gyakornok hallgatóként Gombos Levente vett részt.

A migrációs eszköz kialakításával a földtani felületi elemek attribútum alapú színezések átörökítése egyszerűsödött le, de ahogyan az korábban már említésre került, a földtani térképek esetenként felületi jeleket és földtani indexeket is használnak a jobb elkülöníthetőség és olvashatóság miatt, így ezek kiegészítése mindenképpen manuális munkát igényel még.

Összességében kijelenthető, hogy a stílusok migrációját biztosító eszköz kialakítása nagymértékben leegyszerűsítette az ArcGIS-ben megalkotott jelkulcsok migrációját QGIS környezetbe, és csak kiegészítő jellegű manuális munka maradt a teljes migrálás befejezéséhez.

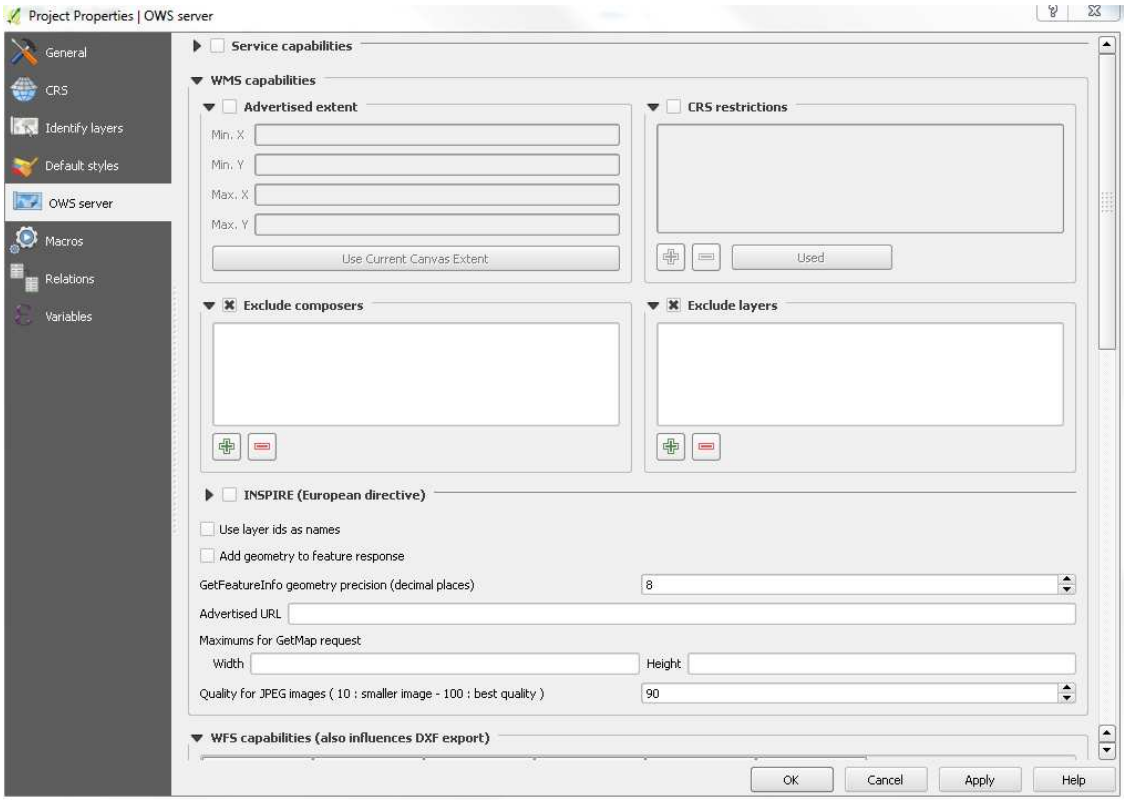
V.2.3. Webes térképszolgáltatás kialakítása QGIS Server segítségével

A migrált adattartalom, és a részlegesen migrált földtani színkulcs segítségével így már létrehozhatóak QGIS Desktop környezetben azok az állományok, amelyek alapján a földtani térképek szolgáltatása lehetővé válik a QGIS Server segítségével, WMS szolgáltatások formájában.

A QGIS Server az Apache webszerver egy ún. FastCGI moduljaként telepíthető, és CentOS környezetben a QGIS-szel együtt telepíthető fel a szerver környezetébe. A szerkesztés történhet asztali környezetben, majd az elkészült állományok mozgathatóak a szerverre. Ehhez természetesen az szükséges, hogy a térképek adattartalmát mind az asztali kliens, mind pedig a szerver elérhesse.

A QGIS Server nem rendelkezik dedikált webes felülettel, hanem az egyes térképszolgáltatások konfigurációs lehetőségei a QGIS Desktop-ba beépülve, a projekt tulajdonságaiként vannak jelen. Itt beállíthatóak a látható rétegek, az elérhető attribútum mezők, valamint megadhatóak a WMS szolgáltatás metaadatai is. Az állomány adattartalmával akár WFS szolgáltatás is indítható a WMS szolgáltatással egy időben.

A grafikus felületen megadott paraméterek végső soron a QGIS Desktop projekt fájljában kerülnek tárolásra, vagyis az ArcGIS környezetre jellemző msd fájlhoz hasonlóan ez a projekt fájl tartalmazza az adattartalomra és a megjelenítésre vonatkozó valamennyi információt, csak az msd fájllal ellentétben ez egy XML fájlt jelent.

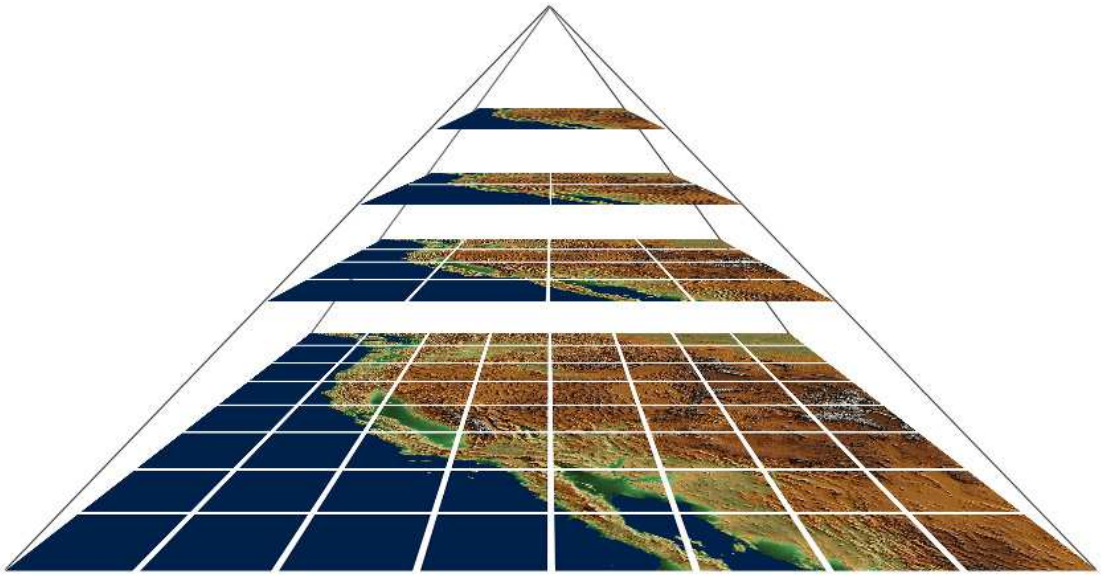


35. ábra: A QGIS Server beállításai a QGIS Desktop-ban

V.2.4. Térképszolgáltatások teljesítmény optimalizációja

A hagyományos WMS szolgáltatások sebességéről és CPU terheléséről már esett szó a korábbi fejezetekben. A bonyolultabb vonalművel, és komplex jelkulccsal rendelkező WMS rétegek szolgáltatása nagyban terhelheti a térképszolgáltatás feladatát ellátó szerver hardverét, és már kisebb felhasználói felhasználás mellett is problémákat okozhat. A terhelés elsősorban a CPU egységet érintheti, melynek nagyfokú igénybevétele a szerver alapvető funkcióinak az ellátását is ellehetetleníti.

Az ArcGIS szoftverekkel kialakított rendszerben erre egy beépített cache-elési (csempézési) mechanizmus kínál megoldást, melyhez hasonló természetesen a nyílt forráskódú világban is megtalálható.



36. ábra: A cache-elés (csempézés) működési elve

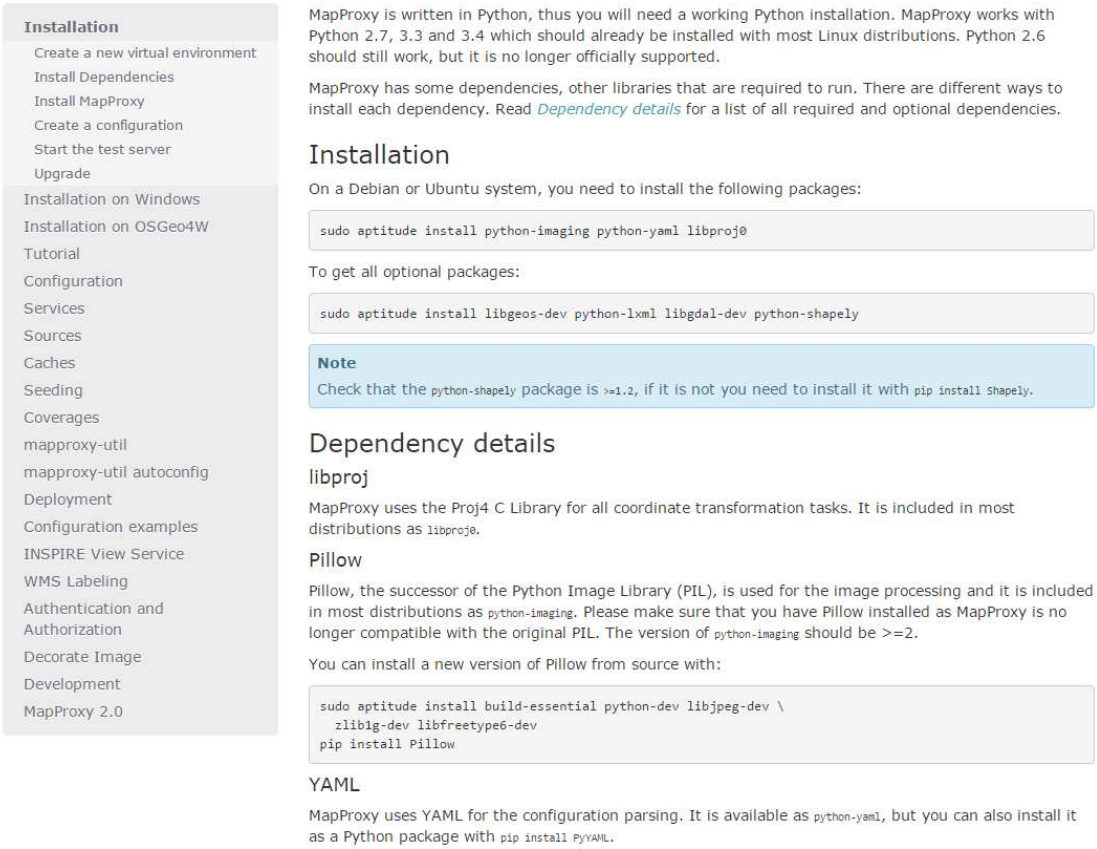
Az elérhető eszközök működési elve, hogy egy előre definiált, koordinátákkal lehatárolt területet négyzetesen növekvő rácshálókkal fed le, és a rácsháló által lehatárolt négyzetes területet kéri le a WMS szolgáltatástól. Ennek a technológiának a segítségével a webes alkalmazások egy WMTS szolgáltatásként hívhatják be az adott forrás WMS réteget.

Az elérhető eszközök többsége ezt a feladatot kizárólag a Google Web Mercator néven közismertté vált módosított Mercator-féle vetületben tárolt adatrétegek esetén képesek megoldani. Mivel Magyarország területét a leginkább kedvező torzulási értékekkel az EOV vetület biztosítja, így a legtöbb magyarországi webes földtani térkép ábrázolása is ebben a vetületben kívánatos, ezért egy olyan eszközre van szükség, mely képes ezt a vetületet is támogatni.

Az elérhető eszközök közül a Mapproxy eszköz került kiválasztásra az alábbi szempontok miatt:

- EOV vetület támogatása
- kedvező csempézési sebesség
- nagyfokú konfigurálhatóság
- átlátható működés
- kiterjedt és folyamatosan frissülő dokumentáció
- proxy-ként való működés lehetősége

Természetesen az eszköz mellett való döntést mindegyik tényező befolyásolta, de mindenképpen szükséges kiemelni az eszközhöz kapcsolódó dokumentáció minőségét, mely mind tartalmilag kiterjedt, mind a tartalmazó funkciókat illetően minden igényt kielégítőnek bizonyult.



37. ábra: A Mapproxy eszköz dokumentációjának részlete

Számos olyan nyílt forráskódú eszköz létezik, amely funkciójában rendkívül kiterjedt, mégis használhatatlan, a felhasználói felület vagy a megfelelő dokumentáció hiányában. A szerver oldali eszközök kiválasztásánál fontos szempontként merült fel a felhasználói felület megléte, hiszen nem garantálható, hogy a működtetés során mindig a kódot is értelmezni tudó felelős személy kerül kijelölésre a működtetésre. Azonban a felhasználói felülettől abban az esetben akár el lehet tekinteni, ha a dokumentáció alapján könnyen kezelhető az adott eszköz.

A Mapproxy tehát egy Python-ban fejlesztett parancssorból futtatható eszköz, amely konfigurációs fájlok segítségével paraméterezhető, és telepítése a Python csomagok kezelésére szolgáló pip eszköz segítségével történhet. A parancssorban meg-

adott konfigurációs állományokat YAML formátumban szükséges megadni a dokumentációban megadható paraméterek alapján.

```
layers:
  - name: root
    title: Root Layer
    layers:
      - name: layer1
        title: Title of Layer 1
        layers:
          - name: layer1a
            title: Title of Layer 1a
            sources: [source1a]
          - name: layer1b
            title: Title of Layer 1b
            sources: [source1b]
      - name: layer2
        title: Title of Layer 2
        sources: [cache2]
```

38. ábra: A Mapproxy eszköz konfigurálása YAML formátum segítségével

Az eszköz képes a QGIS Server által szolgáltatott WMS szolgáltatást felhasználni bemenő adatforrásként, és egy definiált rácsháló (grid) alapján megoldja a csempézés feladatát.

```
grids:
    webmercator:
        base: GLOBAL_WEBMERCATOR
    eov:
      srs: EPSG:23700
      bbox: [400000,0,1000000,400000]
      res: [423.3333333333,211.6666666667,105.8333333333,52.9166666667,26.4583333333,13.2291666667]
```

39. ábra: Grid megadása EOV vetülethez

Az eszköz egyik legnagyobb előnye, hogy nemcsak statikus, könyvtárszerkezetben való csempék generálására képes, hanem tud önálló WMS szolgáltatásként is funkcionálni, és így továbbítja a kéréseket az eredeti, a csempézés alapját jelentő WMS szolgáltatás irányába. Az eredeti WMS-től származó válaszokat pedig természetesen továbbítja a felhasználó irányába.

A QGIS Desktop-ban kartografált, PostgreSQL és PostGIS adattartalommal rendelkező, QGIS Server által szolgáltatott WMS szolgáltatásokat tehát a Mapproxy eszköz segítségével lehetséges csempézett formában, de mégis teljes értékű WMS-ként szolgáltatni. Ezzel majdnem teljessé válik a kialakítandó WebGIS rendszer, melynek így már csak a kliens oldali komponense hiányzik.

V.3. A kliens oldali megjelenítés

A földtani adatok kartografált, interaktív megjelenítéséhez szükséges eszközök és folyamatok már bemutatásra kerültek az előző fejezetekben. Az ArcGIS környezetből való adatkörök és stílusok migrációja után a WebGIS rendszer kliens oldali kialakítása következett. Ennek során olyan rendszer kialakítását tűztem ki célul, amely egy konfigurációs állomány változtatásával képes más adattartalmat és funkcionalitást biztosítani a felhasználóknak.

A fejlesztés térképes feladataihoz a Javascript alapú Openlayers könyvtárat választottam, hiszen a kliens oldali eszközök között ez rendelkezik messze a legjobb sebességmutatókkal és funkciókészlettel, a felhasználói felület kialakításához pedig a széles körben elterjedt, reszponzív megjelenítést biztosító Bootstrap eszköz egyes elemeit használtam fel.

V.3.1. A megvalósításhoz felhasznált eszközök

Az Openlayers programkönyvtár fejlesztése 2005-ben indult meg azzal a céllal, hogy formátumoktól és szerver oldali komponensektől függetlenül egy egységes kliens oldali rendszerben lehessen kezelni számos webes térinformatikai adatformátumot. A fejlesztés 2014-től nagy változáson esett keresztül, és alapjaiban újraírták, de a fejlesztés és a hibajavítás a mai napig is folyamatosan zajlik.

40. ábra: Az Openlayers 3-as verziójának bejelentése Twitter-en

A kód szervezéséhez és tömörítéséhez fejlesztők a Google Closure Compiler használják, de a korábban tárgyalt Dojo Toolkit-tel szemben ennek az ismerete nem szükséges az Openlayers API-t használó fejlesztőknek. Ennek célja elsősorban a kód

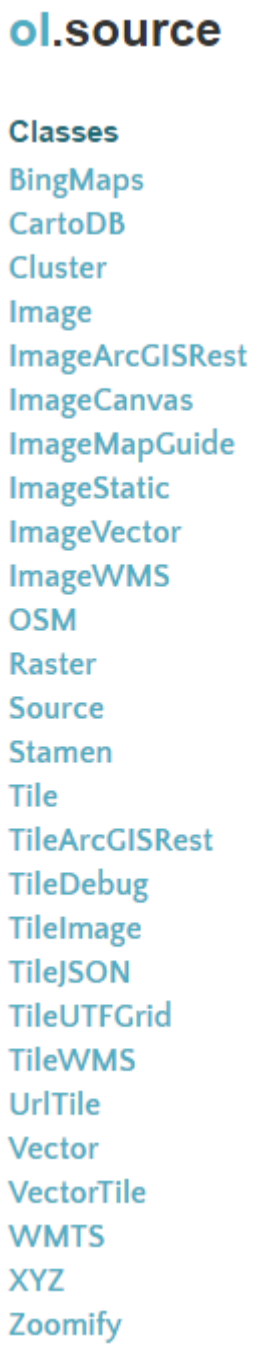
tömörítése a változók nevének, és azok referenciájának a módosításával. Ezzel mind a sebesség, mind a mérettulajdonságok kedvezően alakulnak.

A programkönyvtár fejlesztői azt a célt tartják maguk előtt, hogy használatával a programozók az elérhető maximális rugalmasságot kapják meg; azaz a saját kód szervezése, a felhasználói felület választása egyaránt a céljaiknak és szempontjaiknak leginkább megfelelő lehessen. Ez a részint alacsonyabb kidolgozottságnak is tekinthető megközelítés két következménnyel jár:

- a különféle célfeladatok megoldására a fejlesztőknek sokkal nagyobb rugalmasságuk van, egyszerűbb lehet a meglévő rendszerekbe való integráció
- a fejlesztések kezdeti fázisában több ráfordított energiát igényel a minimális igényeket kiszolgáló működés elérése

Az ArcGIS kliens oldali megoldásai kapcsán említett widget alapú fejlesztési mechanizmus tehát itt nincsen jelen, a fejlesztőnek magának kell gondoskodnia az esetlegesen elvárt modularitásról, a paraméterezhetőségről és a felhasználói felületről. Mindezek ellenére azonban az elkészült alkalmazások kódjainak hatékonysága magasabb lesz (az egyszerűbb szerkezet miatt), és a kódbázis karbantartása, illetve hibajavítása is egyszerűbb feladat lesz.

A konkrét célfeladatok megoldásához ebben az esetben is a programkönyvtár osztályai használhatóak fel, melyek számos adatforrás felhasználását biztosítják általában szabványos nyílt formátumok formájában, de előfordulnak a kereskedelmi szoftverekhez kapcsolódó adatforrások is. Az egyes forrásokból származó adatok rétegekbe szervezhetőek, és az API ezeket a meglévő szerver, vagy kliens oldali szabályok alapján egy közös felületen jeleníti meg.



41. ábra: Az Openlayers támogatott formátumai a dokumentációban

A programkönyvtár mind raszteres, mind vektoros adatforrásokat támogat, utóbbiak megjelenítésére fejlett lehetőségekkel rendelkezik. Szükséges kiemelni, hogy a 2014-től elérhető 3-as verzió sokkal nagyobb teljesítményt képes elérni a renderelés terén, mint amilyet a megelőző verzió nyújtott. Ez nagyban annak köszönhető, hogy a böngészők egyre szélesebb körben támogatják a WebGL renderelés használatát, melynek lényege, hogy a korábbiakhoz képest a böngésző sokkal nagyobb részét képes kihasználni a felhasználó grafikus kártyájának, mint korábban. Ennek az újfajta megjele-

nítő módszernek az alkalmazása egyre inkább egyfajta paradigmaváltás előszeleként is felfogható, hiszen várhatóan tovább nő majd a kliens oldali erőforrások kihasználásának a mértéke, valamint változni fog a megjelenítés definiálásának a helye is.

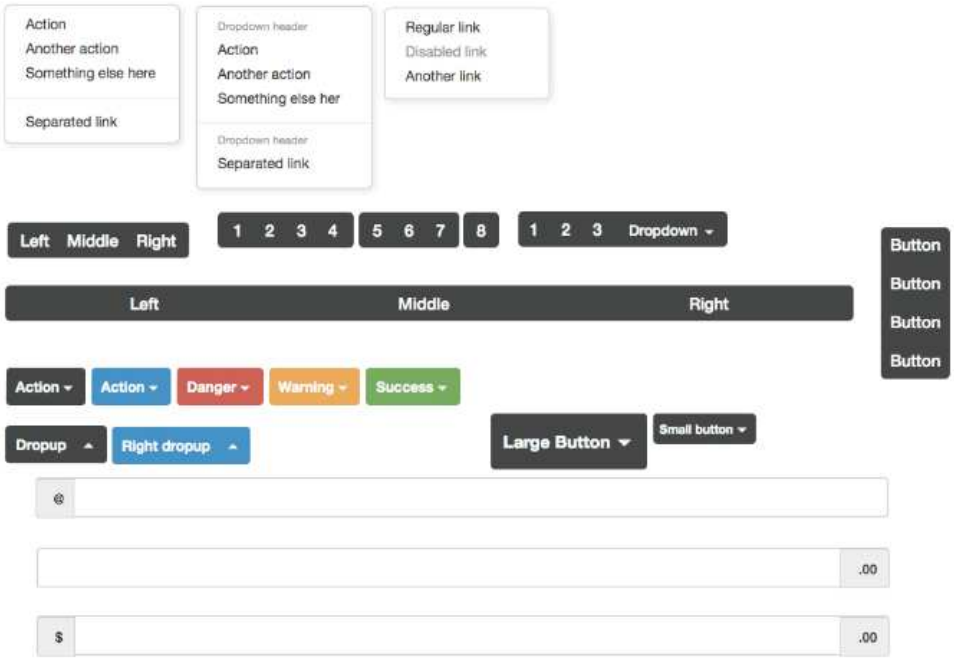
A programkönyvtár támogatja a HTML5 és a CSS3 szabványokat is, így használatával hasznosítható a korábban már tárgyalt, minden eszközön azonos megjelenítést biztosító reszponzív fejlesztési irányelv.

Az Openlayers-ről végeredményben a következők állapíthatóak meg:

- szabványos webszervizek és formátumok széleskörű támogatása
- vetületek széleskörű támogatása
- fejlett renderelési mechanizmus
- egyszerű hibakeresési lehetőségek
- rugalmas fejlesztési módszerek használata
- legújabb böngésző szabványok (HTML5, CSS3) támogatása
- gyakori hibajavítások
- nyílt forráskód, könnyű ráfejlesztési lehetőségek

A fentiek tükrében kijelenthető, hogy a webes térinformatikai adatszolgáltatás céljaira, és a földtani adatok interaktív bemutatására is alkalmas lehet az Openlayers eszköz.

A kliens oldali megjelenítést alapvetően az egyes HTML elemekhez tartozó stílusinformációk adják meg. Ezeknek a stílusinformációknak a megadására a böngészők a CSS formátumot használják, mely jelenleg a 3-as verziónál jár (A HTML 5-ös szabványaihoz igazodva). A különböző böngészők azonban sokszor másképp értelmezik az egyes CSS szabványokat, ezért sokszor nem garantált az egységes megjelenítés, és előfordulhat, hogy egy adott látvány elérését másképpen kell definiálni az egyes böngészőknek. Mivel ezeknek a különbözőségeknek az állandó fejben tartása nem várható el a fejlesztőktől, így ezért az általános funkciókat magukban foglaló elemek kialakításához keretrendszerek állnak rendelkezésre.



42. ábra: A Bootstrap néhány grafikus eleme

Az egyik ilyen, talán legelterjedtebb keretrendszer a Twitter Bootstrap, vagy rövidebb nevén csak Bootstrap, mely a legjellemzőbb felhasználói felületek építőelemeit tartalmazza, teljes funkcionalitással. Így ez nem csak a megjelenítést, hanem a működés alapvető elemeit is nyújtja a fejlesztőknek. Az eszköz tehát lehetővé teszi, hogy weboldalakba, illetve webes alkalmazásokba a fejlesztő integrálhasson olyan komplex eszközöket, amelyeknek mind a megjelenése, mind a működése később befolyásolható. Így a fejlesztőknek nem szükséges teljesen az alapoktól lefejleszteni minden egyes részletét egy adott eszköznek (pl. egy oszlopai alapján sorba rendezhető táblázatnak), hanem azok már készen felhasználhatóak és konfigurálhatóak a Bootstrap keretrendszer használatával.

Ezzel az eszközzel kiegészülve az Openlayers közösen egy olyan keretrendszer kialakítását teszi lehetővé, amely alkalmas lehet egy konfigurálható webes térképi eszköz kialakításához.

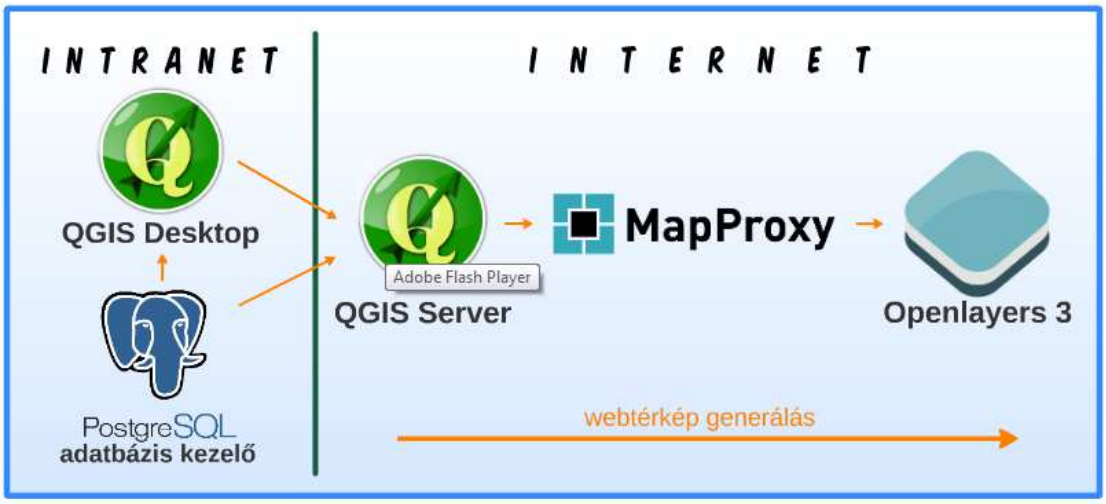
V.3.2. Webes keretrendszer fejlesztése

Az ArcGIS alapú WebGIS rendszer a WebApp Builder for ArcGIS szoftvert biztosítja a webes térképi alkalmazások elkészítéséhez. Az ennek az eszköznek a segítségével elkészült alkalmazásokat ezután testre lehet szabni, tovább lehet fejleszteni. Kuta-

tásom során egy ehhez hasonló rendszert fejlesztettem ki, melynek a fő célja a jól használható felület és a gyorsaság mellett a könnyű konfigurálhatóság volt.

V.3.2.1. A fejlesztés háttere

A fejlesztés során a bemenő alapadatokat a korábban bemutatott PostgreSQL alapú adatbázisok, és az arra épülő QGIS Server-rel elkészített, Mapproxy eszközzel cache-elt webes térképszolgáltatások alkották. A feladat tehát egy interaktív rendszer létrehozása volt ezeknek a webszervizeknek a felhasználásával.



42. ábra: A létrehozott architektúra kiegészülve az Openlayers-szel

A rendszer kliens oldali térinformatikai komponenseként a korábban bemutatott Openlayers függvénykönyvtárat választottam, míg a felhasználói felület kialakításához a Twitter Bootstrap kliens oldali eszközt használtam. A Javascript nyelv egyszerűbb használatához ezen kívül felhasználásra került a jQuery eszköz, melynek segítségével gyorsabban és hatékonyan lehet kliens oldali kódot készíteni. A felhasznált függvénykönyvtárak mellett a kifejlesztett keretrendszer ezekre épülő kódjának a szervezése azonban kulcsfontosságú volt az átláthatóság, a kiterjeszthetőség, és a skálázhatóság szempontjából.

V.3.2.2. A kliens oldali kód strukturálása

A kód strukturálása olyan formában történt, hogy az egyes célfeladatok kódja külön Javascript objektum részeként került elhelyezésre, külön fájlban, és ezeket az alkalmazás vázát jelentő HTML oldal fogja össze.

A Javascript nyelvben az egyes változók deklarálása sokszor "globális" szinten történik, amivel az a probléma, hogy átláthatatlanná teszik a kódot, és így nincsen lehetőség egy valamilyen szervezési elv alapján csoportosítani a kódot. Ezt a jelenséget tovább súlyosbítja, ha a kliens oldali kód a HTML fájlba közvetlenül beágyazva, script tag-ek közé kerül, hiszen ez alapvetően ellentmond annak a manapság egyre inkább méltán népszerű fejlesztési alapkoncepciónak, hogy a felhasználói felület vázát, és annak működési módját minél inkább szükséges elválasztani.

A fent említett problémákra az jelenthet megoldást, ha minden egyes funkciókészletet ellátó programrészlet külön objektumba (és fájlba) kerül, erre az alábbi kódrészlet mutatja a példát.

```
var Gui = {};

Gui.updateLayout = function () {
  $('#map').height(window.innerHeight - 40);
  $('#map').width(window.innerWidth);
  $('#headerController').width(window.innerWidth);
  $('.panel-body').css('height', window.innerHeight - 110);
};

Gui.addEventHandlers = function () {
  $(window).on('resize', function () {
    Gui.updateLayout();
  });
  $('.icon-container').each(function(){
    this.onclick = function() {};
  });
```

43. ábra: A Gui nevű globális objektum

A GUI (Graphical User Interface, vagyis felhasználói felület) összes funkcionalitását láthatóan ez az objektum (Gui), és a gui.js nevű fájl biztosítja. A felhasználói felület rögzített állapotait a Gui objektum tulajdonságaként lehet tárolni, míg a meghívható függvények szintén a Gui objektumhoz kapcsolódnak, ahogyan ezt a fenti ábra is mutatja.

Ahogyan bővül a programozott funkcionalitás, úgy lesz egyre több globális objektum, illetve fájl, de ezek száma mégiscsak egy racionális értéken belül mozog. Természetesen ez a megközelítés nem veti el teljesen a helyi használatú változók használatát (például iterációk esetében), de ezek ily módon nem szennyezik be a globális névteret.

A megközelítés másik előnye, hogy a globális objektumok hozzáférnek egymás tulajdonságaihoz és eljárásaihoz, így a köztük való kommunikáció is megoldott.

V.3.2.3. A kliens oldali kód főbb elemei

Az általam kialakított alkalmazás a fent vázolt programszervezési eljárásnak köszönhetően az alábbi fájlokat foglalta magában, melyeket az oldal vázát jelentő HTML oldal fogja össze.

```
<script src="src/config.js"></script>
<script src="src/map.js"></script>
<script src="src/info.js"></script>
<script src="src/print.js"></script>
<script src="src/gui.js"></script>
```

44. ábra: A keretrendszer saját fájljai

Az egyes fájlok objektumokat is jelentenek, így a kódban globális változóként jelenik meg:

- a Config objektum, mely az alkalmazás konfigurációját hordozza
- a Map objektum, mely az alkalmazás térképi vonatkozását kezeli
- Az Info objektum, amely az egyes elemekre való kattintásra válaszul információt ad a felhasználónak az adott pontban található lekérdezhető objektumról
- a Print objektum, mely értelemszerűen a nyomtatási feladatokat kezeli az alkalmazáson belül
- valamint a korábban már említett Gui objektum, mely a felhasználói felület működését szabályozza

A konfigurálás céljára létrehozott fájl és objektum az alkalmazás valamennyi változó, és változtatható paraméterét tartalmazza.

```
var Config = {
  map: {
    viewOptions: {
      zoom: 3,
      projection: 'EPSG:23700',
      resolutions: [423.3333333333, 211.6666666667, 105.8333333333, 52.9166666667,26.4583333333, 13.2291666667],
      center: [653224.1858820765, 241128.1638166387]
    },
    queryLayers: [
      {name: 'fdt100.DBO.fdt_pg',
        template: {
          'nev': 'Név',
          'geo_ndx': 'Földtani index'
        }
      }
    ],
    baseLayers: {
      stamenTerrain: {
        title: 'Stamen Terrain'
      },
      osm: {
        title: 'OpenStreetMap'
      }
    },
    defaultBaseLayer: 'osm',
    mapproxymap:'fdt100'
  }
};
```

45. ábra: Az alkalmazás konfigurálható paraméterei

Ebben a térképes alkalmazás általános működésére vonatkozóan a következő paramétereket lehetséges megadni:

- az alkalmazás induló középpontja
- az induló zoom szint
- a használt vetületi rendszer kódja
- az elérhető felbontási szintek (ezek megegyeznek a Mapproxy eszközben definiált grid értékeivel)

A térkép fő tematikáját jelentő réteget a Mapproxymap paraméter segítségével lehet megadni, és emellett definiálható, hogy mely alrétegekről lehessen információt kérni. Ezt a queryLayers paraméterezhető objektum feltöltése szabályozza.

```
queryLayers: [
  {name: 'fdt100.DBO.fdt_pg',
    template: {
      'nev': 'Név',
      'geo_ndx': 'Földtani index'
    }
  }
],
```

46. ábra: Szabályozható információs rétegek

Az információs ablak egy template, azaz minta alapján szabályozható, így megadhatóak azoki az attribútumok, és azok alias-ai, amelyek egy adott rétegre vonatkozóan a kattintás hatására megjelennek.

47. ábra: Elem információ megjelenítése template alapján

Amennyiben a térképi alkalmazás nem EOV-s vetületet használ, úgy lehetséges megadni az alapértelmezett háttér réteget (jelenleg ez lehet OpenStreetMap vagy Stamen réteg, mindkettő szabadon elérhető).

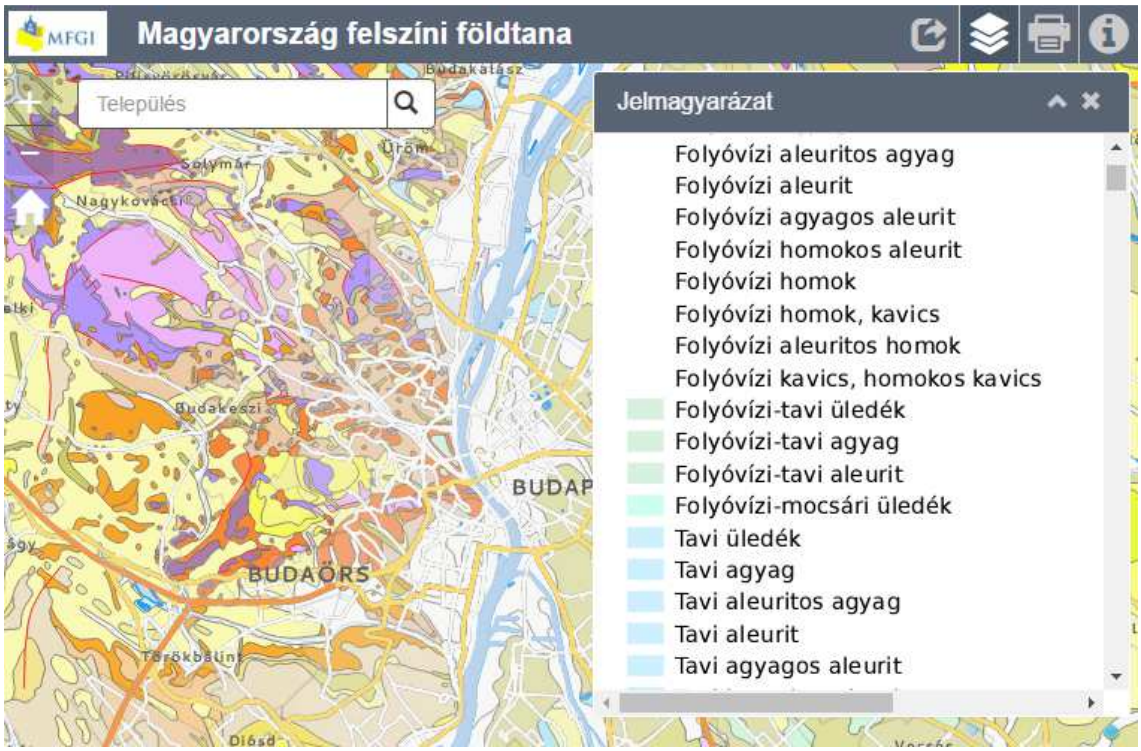
```
baseLayers: {
  stamenTerrain: {
    title: 'Stamen Terrain'
  },
  osm: {
    title: 'OpenStreetMap'
  }
},
defaultBaseLayer: 'osm',
```

48. ábra: Alaprétegek beállítása a konfigurációs fájlban

V.3.2.4. A felhasználói felület kialakítása

A kialakított kódszervezési struktúra nagyban hozzájárult az alkalmazás hatékony, skálázható működéshez, és lehetővé teszi a jövő folyamán további funkciókészletek integrálását a rendszerbe. Azonban a felhasználói felület kialakításakor is figyelembe kellett venni a bővíthetőség igényét, ezért a felületen az egyes funkciók elérését az alkalmazás felső részébe integrált gombsor biztosítja, míg a részletes funkciókészletek panelekben jelennek meg.

49. ábra: A jobb felső sorban elérhető gombsor



42. ábra: A panelben megjelenő funkciókészlet

A felhasználói felület kialakításakor szándékosan olyan megjelenítést választottam, amely hasonló az ArcGIS rendszer által biztosított felhasználói felületnek. Ennek két fő oka volt. Egyrészt az vezérelt, hogy megmutassam, hogy nyílt forráskódú eszközök felhasználásával is létre lehet hozni jól működő, konfigurálható felületet. A másik tényező az volt, hogy egy egységes felülettel a felhasználónak nem tűnik szembe, hogy a használt technológia esetleg más a háttérben, és így az egyes fejlesztésekre a legalkalmasabb eszköz választható, de mégis megmarad az egységes kliens oldali megjelenés.

A keretrendszer fejlesztésével egy olyan eszközt hoztam létre, amely alkalmas a földtani adatok kartografált, interaktív megjelenítésére, és emellett kódbázisa átlátható, jól szervezett és tovább bővíthető.

V.4. Előnyök és hátrányok

A fejezetben bemutattam az általam kidogozott nyílt forráskódú WebGIS rendszert, mely alkalmas a földtani adatok kartografált, interaktív megjelenítésére webes környezetben.

A rendszer alapját egy PostgreSQL adatbázis kezelő jelenti PostGIS kiterjesztéssel, melyen alapulva a kartografálás QGIS Desktop-ban történhet egy ArcGIS környezetből történő jelkulcs migráció után. A kartografált földtani térképből ezután a QGIS Server állítja elő a webes térképszolgáltatást, melynek cache-elése, csempézése a kedvezőbb sebesség elérése miatt a Mapproxy eszközzel történik. Ezt a forrást felhasználva a böngészőben való megjelenítést az Openlayers, illetve a Bootstrap függvénykönyvtárak segítségével egy konfigurálható keretrendszer oldja meg.

A rendszer adattárolásra hivatott eleme, a PostgreSQL a PostGIS kiterjesztéssel jelenleg a térinformatikai adattárolás egyik legfejlettebb eszköze, és használatával nemcsak a webes publikáció válik lehetségessé, hanem megnyílik vele a további nyílt forráskódú szoftverekből való elérés lehetősége. Az kidolgozott adatmigrációs eszköz segítségével ráadásul lehetséges a két adattárolás akár párhuzamos használata, és így a többféle szoftveres eszközből való elérés megteremti a lehetőséget az adatokhoz való szoftverfüggetlen elérésre az Intézeten belül.

Az open-source eszközökkel való webes adatszolgáltatás eléréséhez szükséges volt az ArcGIS környezetben kialakított stílusok migrálása. A probléma megoldása során megmutatkozott a nyílt forráskódú szoftverekben rejlő erő, hiszen a stílusok migrációja elsősorban a QGIS Desktop által használt qml formátum nyitottsága miatt vált lehetővé. Ez a nyitottság a továbbiakban is számos lehetőséget rejt.

A webes térképszolgáltatás elindításáért felelős QGIS Server egyik hátránya lehet, hogy nem rendelkezik egy webes felhasználói felülettel, viszont ennél sokkal fontosabb tulajdonsága, hogy képes egy asztali környezetben kartografált földtani térképből térképszolgáltatást indítani, a jelkulcs megtartásával. A csempézés folyamatára használt Mapproxy eszköz nagyfokú konfigurálhatóságával a rendszer egyik legerősebb komponense, és az ArcGIS környezetben biztosított eszközöknél kedvezőbb sebességgel rendelkezik.

A kliens oldali keretrendszer két, aktív fejlesztési háttérrel rendelkező eszközre, az Openlayers-re és a Bootstrap-re épül. A térinformatikai szempontból fontosabb Openlayers rugalmasságával és sokrétű adattípus támogatásával a keretrendszer további kiterjesztését teszi lehetővé. A fejlesztett webes keretrendszer további előnye, hogy átlátható kódszervezési mechanizmust követ, és így az esetlegesen igényelt to-

vábbi funkciókkal könnyen kiterjeszthető, de ebből az okból kifolyólag sokkal egyszerűbb a kód debug-olása (hibakeresése), mint az ArcGIS környezetben.

A felépített rendszerről összességében kijelenthető, hogy képes kiszolgálni az MFGI webes térképszolgáltatási igényeit, és jó alternatívája lehet az ArcGIS alapú WebGIS rendszernek. A nyílt forráskódú rendszerbe való adat, illetve jelkulcs migrációnak köszönhetően akár a teljes földtani térképkészítés folyamata átkerülhetne open-source környezetbe. Mivel azonban az ArcGIS asztali termékeiben az elemzésekhez és adatfeldolgozáshoz biztosított eszközök jelenleg még erősebbek, mint amit a QGIS Desktop tud nyújtani, ezért a nyílt forráskódú eszközök elsősorban a webes publikálás során használhatóak hatékonyan a Magyar Földtani és Geofizikai Intézetben.

Végül, de nem utolsósorban az előnyös tulajdonságok között szükséges megemlíteni az open-source eszközök ingyenességét, illetve azt, hogy fejlesztési irányukat lehetséges befolyásolni meglévő programozási tudás, vagy anyagi tőke segítségével. Így amennyiben a bemutatásra kerülő rendszer kiváltja az ESRI alapú technológiát, úgy vagy anyagi forrás szabadulhat fel, vagy egy még inkább testreszabott megoldás kifejlesztése válik lehetővé.

Összefoglalás

A Magyar Földtani és Geofizikai Intézet által elkészített földtani térképek a kutatóintézet egyik legfontosabb végtermékei. A publikáció korábban papírtérképek segítségével történt, mára azonban a modern kor elvárásainak megfelelően már webes környezetben zajlik. Az asztali szerkesztéshez használt szoftverekhez illeszkedve az Intézet korábban egy zárt forráskódú környezet használata mellett döntött. Doktori kutatásomban ezért azt kívántam megvizsgálni, hogy lehetséges-e a földtani adatok weben való interaktív megjelenítése open-source eszközök segítségével.

A dolgozat bevezető fejezetében áttekintést nyújtottam a WebGIS rendszerek felépítéséről, megvizsgáltam a webes adatszolgáltatáshoz legalkalmasabb adattárolási mechanizmusokat, áttekintettem a WebGIS rendszerek szerver oldali komponenseinek feladatait, valamint bemutattam az interaktív kliens oldali megjelenítéshez használt eszközök típusait.

A második fejezetben a szorosan vett szakmai keretek közül kilépve ismertettem a földtani térképek jellegzetességeit és típusait, azok osztályozási lehetőségeit, a földtani térképszerkesztés folyamatát, valamint az alkalmazott tematikus kartográfiai módszereket.

A kutatásom részeként párhuzamosan, egymással megegyező hardveres környezetben teszteltem a kétféle térképszolgáltatási módszert, és áttekintést nyújtottam a kialakításuk során felmerülő hardveres és hálózati megfontolásokról, majd bemutattam földtani adatokat tartalmazó adatbázisok geometriai adattárolását.

Az értekezés negyedik fejezetében a kereskedelmi, zárt forráskódú szoftverekkel kialakított WebGIS rendszer sarkalatos pontjait mutattam be az adattárolástól a szerver oldali komponensen keresztül egészen a kliens oldali megjelenítésig, vázolva a rendszer előnyös és hátrányos tulajdonságait.

Doktori munkám eredményeképpen a földtani adatok kartografált, interaktív megjelenését biztosító WebGIS rendszert hoztam létre nyílt forráskódú eszközök segítségével. A kialakítás részleteit és eredményeit a disszertáció utolsó fejezete tartalmazza. Ebben bemutatom az általam kidolgozott rendszer egyes alkotóelemeit valamint az adatbázisok és a jelkucsok migrációjára kifejlesztett eszközöket.

Summary

The geologic maps made by the Geological and Geophysical Institute of Hungary are currently published with the use of commercial GIS products. My objective was to investigate the potential use of open-source software products for the same purpose, and to create an architecture that could facilitate the publishing of geology related GIS data on the web in the form of interactive web mapping applications.

In the first chapter of my dissertation a general overview of WebGIS architectures was provided, outlining the potential and most suitable GIS data storage methods, the role of server-side components in the architecture as well as the possible forms of interactive client-side visualizations in web browsers.

Followed by the presentation of general architectures, I provide details about the characteristics of geologic GIS data, and present the typical classification of geologic maps. I also outline the map production workflow and the applied potential cartographic methods related to geologic maps.

As an important part of my research, I have implemented and tested two WebGIS solutions with the same hardware tools, one with commercial software products and another with an open-source stack. In my essay I have provided the necessary considerations concerning the hardware and network related aspects of the systems, and I also investigated the consequences of the applied GIS data storage models.

The fourth part of the essay contains observations about the implemented commercial WebGIS system, from the data storage to the client-side visualizations, focusing on the advantages and disadvantages of such a system.

In the closing chapter of my dissertation the proposed open-source software architecture is presented as a core result of my research, which is capable of providing a web based solution for interactively viewing geologic maps with high standard cartographic design. As a result, I also provide the required solutions for the necessary data and cartographic design migration, outlining the software tools I developed for those purposes.

Köszönetnyilvánítás

Szeretnék köszönetet mondani mindazoknak, akik doktori kutatásomat segítették valamilyen formában, és akik nélkül jelen disszertáció nem jött volna létre.

Elsősorban Elek István konzulensemet, és Orosz László felettesemet szeretném kiemelni, akik baráti biztatásukkal és értékes tanácsaikkal folyamatosan segítették munkámat, és hasznos útmutatásaikkal nagyban hozzájárultak a disszertáció színvonalának emeléséhez. Hálás vagyok azért, hogy hittek és bíztak bennem, és hogy mindig a segítségemre voltak.

Köszönetet mondok Gede Mátyásnak, aki bevezetett a szerver és kliens oldali nyelvek világába, és akinek a segítségével megismerhettem a nyílt forráskódú világban rejlő lehetőségeket.

Hálás köszönet az ELTE Térképtudományi és Geoinformatikai Tanszékének, akik segítségével megszerettem a térképészet és a térinformatika tudományát, és akik mindig a legmagasabb színvonalon igyekeztek a szakma tudásanyagát átadni.

Köszönet illeti az MFGI Geoinformatikai Főosztályának teljes csapatát, akik inspiráló környezetet biztosítottak a doktori kutatáshoz. Külön köszönetet szeretnék mondani Turczi Gábornak, aki bevezetett a térinformatika földtani alkalmazásának világába, és aki a doktori kutatást illetően is hasznos tanácsokkal látott el. A hardveres környezetek kialakításában és megvalósításában elévülhetetlen érdeme van Zelei Tamásnak és Rezessy Attilának, ezúton köszönöm a segítségükért. Pogácsás Rékának a jelkulcsi konverziók során nyújtott hasznos segítséget szeretném megköszönni. Vikor Zsuzsának, Varga Bálintnak, Maigut Verának és Popovics Istvánnak hálás vagyok a baráti bíztatásokért, amivel a doktori során kísértek.

A földtani térképezésre vonatkozó részek lektorálásáért Kercsmár Zsoltnak jár köszönet, aki önzetlen segítséget biztosított a felmerülő kérdések megválaszolásában.

Végül, de nem utolsó sorban szeretném megköszönni családomnak azt a biztatást, amit nyújtottak, és akik arra tanítottak, hogy ha valamit csinálok, azt mindig csak a legjobbra törekedve érdemes tenni.

Irodalomjegyzék

Bostock, M., Metcalf C. (2016): The TopoJSON Format Specification URL: https://github.com/topojson/topojson-specification/ (Elérve: 2017. február 15.)

Burhum R. Y., Kousgaard, U. (2012): Are there any attempts to replace the shapefile? URL: http://gis.stackexchange.com/questions/20613/are-there-any-attempts-to-replace-the-shapefile? (Elérve: 2017. február 15.)

Butler,H., Daly, M., Doyle, A., Gillies, S., Schaub, T., Schmidt C. (2016): The GeoJSON format Specification http://geojson.org/geojson-spec.html URL: http://geojson.org/geojson-spec.html (Elérve: 2017. február 15.)

Chen R., Xie J. (2008): Open Source Databases and Their Spatial Extensions. In: Advances in Geographic Information Science, Springer, pp. 105-129.

Dhakal, S. (2016): Why should GIS professionals make a switch to javascript based Web Appbuilder (WAB) from Flex/Silverlight based Map Viewer? https://www.linkedin.com/pulse/why-should-gis-professionals-make-switch-javascript-based-dhakal (Elérve: 2017. február 15.)

Elek, I (2015).: Topologikus térbeli adatstruktúrák. Typotex kiadó, Budapest

Elek, I. (2014): Redundancia-mentes topologikus adatszerkezetek, Geodézia és Kartográfia LXVI:(11-12) pp. 19-24.

Elek, I. – Gercsák, G. (2011): Developing map databases: problems and solutions. Joint ICA Symposium. Orleans

ESRI Inc. (1998): ESRI Shapefile Technical Description - An ESRI White Paper URL: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf (Elérve: 2017. február 15.)

ESRI Inc. (2016): The architecture of a geodatabase. URL: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/the-architecture-of-a-geodatabase.htm (Elérve: 2017. február 15.)

ESRI Inc. (2016b): An overview of geodatabase system tables. URL: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/using-sql-with-gdbs/overview-geodatabase-system-tables.htm (Elérve: 2017. február 15.)

ESRI Inc. (2016c): Types of geodatabases. URL: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/geodatabases/types-of-geodatabases.htm (Elérve: 2017. február 15.)

ESRI Inc. (2016d): Geodatabase storage in relational databases. URL: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/gdb-architecture/geodatabase-storage-in-relational-databases.htm (Elérve: 2017. február 15.)

Flower, C. (2012): MapServer and GeoServer (and tilecache) comparison serving Ordnance Survey raster maps. URL: http://www.esdm.co.uk/mapserver-and-geoserver-and-tilecache-comparison-serving-ordnance-survey-raster-maps (Elérve: 2017. február 15.)

Galambos Cs. (2004): Földtani térképek felületi jelei. Geodézia és Kartográfia 56(7): 16-21.

Galambos Cs., Simonyi D. (2006): Földtani térképeken alkalmazható színadatbázis és felületijel-készlet. In: Balla Z. (ed.): A Magyar Állami Földtani Intézet Évi Jelentése 2005, Magyar Állami Földtani Intézet, Budapest, pp. 193-198.

Geoserver (2016): Userv manual: SpatiLite. URL: http://docs.geoserver.org/latest/en/user/community/spatialite/index.html (Elérve: 2017. február 15.)

Hall, B., Leahy M. (2008): Open Source Approaches in Spatial Data Handling. Springer-Verlag Berlin Heidelberg. pp. 10-11.

Huber W. A., Lawhead, J. (2012): “Oddities” in the Shapefile technical specification URL: http://gis.stackexchange.com/questions/18969/oddities-in-the-shapefile-technical-specification (Elérve: 2017. február 15.)

Henderson, C. (2014): Mastering Geoserver. Packt Publishing.

Kercsmár, Zs., Budai T., Csillag G., Selmeczi I., Lantos Z., Babinszki E., Maros Gy. (2014): A klasszikus földtani térképezés gazdasági, társadalmi és tudományos jelentősége. MFGI Évi Jelentése 2012-2013-ról, MFGI, Budapest pp. 167-178.

Loechel, A. J., Schmid, S. (2012): Caching techniques for high-performance Web Map Services. Proceedings of the AGILE'2012 International Conference on Geographic Information Science, pp. 52-57.

Maigut V. (2005): Földtani térképek kartografálásának segítése térinformatikai módszerekkel. In: Balla Z.: A Magyar Állami Földtani Intézet Évi Jelentése 2004, Magyar Állami Földtani Intézet, Budapest pp. 139-144.

Maigut V. (2004): Új, digitális földtani alapmű a MÁFI-ban. Geodézia és Kartográfia 56(7): 22-26.

Martinez A. (2015): How to choose a geological relational database management system. URL: http://opengeostat.com/how-to-choose-geological-database-management-system/ (Elérve: 2017. február 15.)

McInerney, D., Kempeneers, P. (2015): Open Source Geospatial Tools, Applications in Earth Observation. Springer, pp. 46-49.

Mearns B. (2015): QGIS Blueprints. Packt Publishing. pp.138-139

Mitchell T. (2005): Web Mapping Illustrated. O'Reilly, pp. 241-275.

MySQL (2016): Supported Spatial Data Formats. URL: https://dev.mysql.com/doc/refman/5.7/en/gis-data-formats.html (Elérve: 2017. február 15.)

Nasser, H. (2014): Learning ArcGIS Geodatabases. PacktPublishing Ltd., Birmingham pp. 33-42.

Neteler, M., Mitášová, H. (2008): Open Source GIS: a GRASS GIS approach. Springer, New York, 424 p. URL: http://www.grassbook.org (Elérve: 2017. február 15.)

Peters, D. (2008): Building a GIS: System Architecture Design Strategies for Managers. pp. 5-13.

PostgreSQL (2016): The pg_hba.conf File. URL: https://www.postgresql.org/docs/9.4/static/auth-pg-hba-conf.html (Elérve: 2017. február 15.)

Quinn S., Dutton, J. A. (2016) Some common open formats for spatial data. The Pennsylvania State University. URL: https://www.e-education.psu.edu/geog585/node/691 (Elérve: 2017. február 15.)

Raimundo, S., Chang-Tien L. (2008): Geography Markup Language (GML). In: Shekhar, S. : Encyclopedia of GIS. Springer, pp. 364-369.

Rob, P., Coronel C., Crockett K. (2008): Database Systems: Design, Implementation & Management. Cengage Learning EMEA. pp. 223-224

Síki Z. (2009): Produktív környezetben használt, nyílt forráskódú komplex térinformatikai megoldások Prezentáció, CASCADOSS műhelymunka-tanácskozás és GRASS tanfolyam, Szeged 2009. január 27-30.
URL: http://www.agt.bme.hu/gis/eloadasok/siki_szeged2009sz.pdf (Elérve: 2017. február 15.)

Santiago, A. (2015): The book of Openlayers. Leanpub

SQLite (2016): About SQLite URL: https://www.sqlite.org/about.html (Elérve: 2017. február 15.)

Tezer, O.S.,(2014): SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. URL: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems (Elérve: 2016 február 15.)

Thakur, J.K., Singh, S.K., Ramanathan, A., Prasad, M.B.K., Gossel, W. (2012): Geospatial Techniques for Managing Environmental Resources, Springer, pp. 105-111.

Tomlinson R. (2013): Thinking about GIS: Geographic Information System Planning for Managers. ESRI Press. pp. 109-150.

Tóth K. (2014): SQL Server Database design. CreateSpace Independent Publishing Platform

Turczi G. (2005): Földtani térmodell építése – adatbázisok az intra- és interneten. In: Balla Z.: A Magyar Állami Földtani Intézet Évi Jelentése 2004, Magyar Állami Földtani Intézet, Budapest pp. 125-130.

Turczi G. 2012: A digitális térképkészítés korszaka a Magyar Állami Földtani Intézetben. A Magyar Állami Földtani Intézet Évi Jelentése 2010, Magyar Állami Földtani Intézet, Budapest pp. 97-101.

Viescas, J. L., Hernandez M.J. (2009): SQL-lekérdezések földi halandóknak. Addison-Wesley, pp. 4-16.

Williams A. (2012): Oracle Makes More Moves To Kill Open Source MySQL. URL: https://techcrunch.com/2012/08/18/oracle-makes-more-moves-to-kill-open-source-mysql/ (Elérve: 2016 február 15.)

Zhu B., Wanf A. (2011): The Storage Technology for GIS Data Realization. Journal of Computers. 2011/6. pp. 2229-2236

Ábrajegyzék

Mellékletek

Az adatbázis migrációhoz fejlesztett eszköz kódja

sdebinary2postgis.py

```
# -*- coding: utf-8 -*-
#The following are required:
#-ArcGIS 10.3, arcpy installed
#-python 2.7.5 installed
#-ogr2ogr 2.0.2 installed and in the PATH variable of the user
#-psycopg installed with pip
#-all feature classes are stored in feature datasets
#-all feature classes have(!), and have the same EPSG code

import sys
reload(sys)
sys.setdefaultencoding('utf-8')
import arcpy
import os
from psycopg2 import connect
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
import ogr
import subprocess


source_workspace = arcpy.GetParameterAsText(0)
sql_geometry_workspace = arcpy.GetParameterAsText(3)
pg_host = arcpy.GetParameterAsText(4)
pg_conn_dbname = arcpy.GetParameterAsText(5)
```

```
pg_user = arcpy.GetParameterAsText(6)
pg_password = arcpy.GetParameterAsText(7)

wsd = arcpy.Describe(source_workspace)
pg_target_dbname = wsd.connectionProperties.database
mssql_source_servername = wsd.connectionProperties.server
mssql_source_dbname =
arcpy.Describe(sql_geometry_workspace).connectionProperties.database

arcpy.env.overwriteOutput = True

#copy feature classes from source workspace to database with sql geometry
arcpy.env.workspace = source_workspace
fdlist = arcpy.ListDatasets()
for fd in fdlist:
  arcpy.AddMessage(unicode("Copy feature classes from the following feature dataset:
" + fd))
  fclist = arcpy.ListFeatureClasses("*","",fd)
  for fc in fclist:
    arcpy.AddMessage(fc)
    outfc = os.path.join(sql_geometry_workspace,fc)
    arcpy.CopyFeatures_management(fc, outfc)
arcpy.AddMessage("Successfully copied all feature classes to a database with sql
geometry")

#create empty postgis database
try:
  conn = connect("user="+pg_user+" dbname="+pg_conn_dbname+"
host="+pg_host+" password="+pg_password)
  conn.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
  arcpy.AddMessage("Successfully established connection to PostGIS server")
  cur = conn.cursor()
```

```
  cur.execute("CREATE DATABASE "+pg_target_dbname+" WITH OWNER="+pg_user+"
ENCODING='UTF8' TABLESPACE = pg_default LC_COLLATE='en_US.UTF-8' LC_CTYPE =
'en_US.UTF-8' CONNECTION LIMIT=-1;")
  conn.commit()
  cur.close()
  conn.close()
  arcpy.AddMessage("Successfully created empty database")
except:
  arcpy.AddError(arcpy.GetMessages(2))
#add PostGIS extension
try:
  conn = connect("user="+pg_user+" dbname="+pg_target_dbname+"
host="+pg_host+" password="+pg_password)
  arcpy.AddMessage("Successfully established connection to PostGIS server")
  cur = conn.cursor()
  cur.execute("CREATE EXTENSION PostGIS;CREATE SCHEMA dbo AUTHORIZATION
simobenedek;")
  conn.commit()
  cur.close()
  conn.close()
  arcpy.AddMessage("Successfully added PostGIS extension")
except:
  arcpy.AddError(arcpy.GetMessages(2))

import _subprocess
startupinfo = subprocess.STARTUPINFO()
startupinfo.dwFlags |= _subprocess.STARTF_USESHOWWINDOW
startupinfo.wShowWindow = _subprocess.SW_HIDE

pg_conn_string = "PG: host="+pg_host+" user="+pg_user+"
dbname="+pg_target_dbname+" password="+pg_password
```

```
mssql_conn_string =
"MSSQL:server="+mssql_source_servername+";database="+mssql_source_dbname+";
trusted_connection=yes"
epsg_code = "EPSG:" + str(arcpy.Describe(fc).spatialReference.PCSCode)


subprocess.call(["ogr2ogr.exe", "-overwrite", "--config", "PGCLIENTCODING", "ISO-
8859-2", "-f", "PostgreSQL", pg_conn_string, mssql_conn_string, "-a_srs", epsg_code,
"-lco", "GEOMETRY_NAME=geom"],startupinfo=startupinfo)
arcpy.AddMessage("Successfully copied all tables to PostGIS")

arcpy.env.workspace = sql_geometry_workspace
fclasses = arcpy.ListFeatureClasses()
for dfc in fclasses:
  arcpy.Delete_management(dfc)
```

A fejlesztett konfigurálható webes keretrendszer kódja

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Magyarország felszíni földtani térképe - 1:100 000</title>

    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <!-- Optional theme -->
```

```
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css">
  <!--fontawesome-->
  <link href="http://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-
awesome.min.css" rel="stylesheet">
  <!--Openlayers style-->
  <link rel="stylesheet" href="https://Openlayers.org/en/v3.19.1/css/ol.css"
type="text/css">
  <!--map related icons-->
  <link href="https://mapskincdn.appspot.com/1.0/mapskin.min.css"
rel="stylesheet">
  <!--Own styles-->
  <link rel="stylesheet" href="site.css">
 </head>
 <body>
  <div id="headerController">
   <a href="http://map.mfgi.hu" target="_blank"><img class ="floating-left logo"
src="logo.jpeg"></a>
   <div class="floating-left title hidden-xs">Magyarország felszíni földtana</div>
   <div id="information" class="icon-container floating-rigth"><i class="ms ms-
information ms-2x" title="Információ"></i></div>
   <div id="print" class="icon-container floating-rigth hidden-xs"><i class="ms ms-
printer ms-2x" title="Nyomtatás"></i></div>
   <!--<div id = "search" class="icon-container floating-rigth"><i class="ms ms-zoom
ms-2x" title="Keresés"></i></div>-->
   <div id="legend" class="icon-container floating-rigth"><i class="ms ms-layers ms-
2x" title="Jelmagyarázat"></i></div>
   <div id ="share" class="icon-container floating-rigth"><i class="glyphicon glyphicon-
share" title="Megosztás" style="font-size:24px;"></i></div>
  </div>
  <!--information panel-->
```

```
  <div id="panelinformation" class="panelBase panel panel-default" style="display:
none;">
    <div class="panel-heading panel-heading-custom">
      <h3 class="panel-title">Infó az alkalmazásról</h3>
      <span class="pull-right">
        <!--<i class="glyphicon glyphicon-fullscreen" data-placement="left" title="Show
Full Screen"></i> -->
        <i class="glyphicon glyphicon-chevron-up minimizer" data-toggle="collapse"
href="#panelinformationContent" data-placement="left" title="Panel lekicsinyíté-
se"></i> 
        <i class="glyphicon glyphicon-remove" data-toggle="tooltip" data-
placement="left" title="Bezár"></i> 
      </span>
    </div>
    <div id="panelinformationContent" class="panelContent panel-body collapse in">
      Üdvözöljük az MFGI interaktív alkalmazásában!<br/><br/>

      Reméljük örömmel használja az alkalmazást!
      Amennyiben kérdése lenne,
      kérjük látogasson el a <a href="http://map.mfgi.hu"
target="_blank">http://map.mfgi.hu</a> oldalra,
      ahol megtalálja elérhetőségeinket!<br/><br/>
      <img src="logo.jpeg">
    </div>
  </div>
  <!--search panel-->
  <div id="panelsearch" class="panelBase panel panel-default" style="display: none;">
    <div class="panel-heading panel-heading-custom">
      <h3 class="panel-title">Keresés</h3>
      <span class="pull-right">
        <!--<i class="glyphicon glyphicon-fullscreen" data-placement="left" title="Show
Full Screen"></i> -->
```

```
        <i class="glyphicon glyphicon-chevron-up minimizer" data-toggle="collapse"
href="#panelsearchContent" data-placement="left" title="Panel lekicsinyíté-
se"></i> 
        <i class="glyphicon glyphicon-remove" data-toggle="tooltip" data-
placement="left" title="Bezár"></i> 
      </span>
    </div>
   </div>
   <!--legend panel-->
   <div id="panellegend" class="panelBase panel panel-default" style="display: none;">
    <div class="panel-heading panel-heading-custom">
     <h3 class="panel-title">Jelmagyarázat</h3>
     <span class="pull-right">
      <!--<i class="glyphicon glyphicon-fullscreen" data-placement="left" title="Show
Full Screen"></i> -->
      <i class="glyphicon glyphicon-chevron-up minimizer" data-toggle="collapse"
href="#panellegendContent" data-placement="left" title="Panel lekicsinyíté-
se"></i> 
      <i class="glyphicon glyphicon-remove" data-toggle="tooltip" data-
placement="left" title="Bezár"></i> 
     </span>
    </div>
    <div id="panellegendContent" class="panelContent panel-body collapse in">
     <img
src="http://openmap.mfgi.hu/wms/fdt100_new/fdt100_new?SERVICE=WMS&VERSIO
N=1.3.0&REQUEST=GetLegendGraphic&LAYER=fdt100.DBO.fdt_pg&FORMAT=image/p
ng&STYLE=default&SLD_VERSION=1.1.0">
    </div>
   </div>
   <!--print panel-->
   <div id="panelprint" class="panelBase panel panel-default" style="display: none;">
    <div class="panel-heading panel-heading-custom">
```

```
      <h3 class="panel-title">Jelmagyarázat</h3>
      <span class="pull-right">
       <!--<i class="glyphicon glyphicon-fullscreen" data-placement="left" title="Show
Full Screen"></i> -->
       <i class="glyphicon glyphicon-chevron-up minimizer" data-toggle="collapse"
href="#panelprintContent" data-placement="left" title="Panel lekicsinyíté-
se"></i> 
       <i class="glyphicon glyphicon-remove" data-toggle="tooltip" data-
placement="left" title="Bezár"></i> 
      </span>
    </div>
    <div id="panelprintContent" class="panelContent panel-body collapse in">
     Ezzel a funkcióval térkép aktuális kivágatát tudja kinyomtatni az alábbi gombra
való kattintással!
      <button id="printButton" type="button" class="btn btn-
success">Nyomtatás</button>
    </div>
   </div>
   <!--share panel-->
   <div id="panelshare" class="panelBase panel panel-default" style="display: none;">
    <div class="panel-heading panel-heading-custom">
     <h3 class="panel-title">Megosztás</h3>
     <span class="pull-right">
      <!--<i class="glyphicon glyphicon-fullscreen" data-placement="left" title="Show
Full Screen"></i> -->
       <i class="glyphicon glyphicon-chevron-up minimizer" data-toggle="collapse"
href="#panelshareContent" data-placement="left" title="Panel lekicsinyíté-
se"></i> 
       <i class="glyphicon glyphicon-remove" data-toggle="tooltip" data-
placement="left" title="Bezár"></i> 
      </span>
    </div>
```

```
    <div id="panelshareContent" class="panelContent panel-body collapse in">
      Amennyiben tetszik az alkalmazás, itt megoszthatja ismerőseivel!
      <br/> <br/>
      <div class="btn-group">
       <a class="btn btn-default btn-lg" target="_blank" title="Megosztás Facebook-on"
href="https://www.facebook.com/sharer/sharer.php?u=openmap.mfgi.hu"><i
class="fa fa-facebook fa-lg fb"></i></a>
       <a class="btn btn-default btn-lg" target="_blank" title="Megosztás Twitter-en"
href="https://twitter.com/home?status=tetszik%20ez%20a%20t%C3%A9rk%C3%A9p!
%20http%3A//openmap.mfgi.hu"><i class="fa fa-twitter fa-lg tw"></i></a>
       <a class="btn btn-default btn-lg" target="_blank" title="Megosztás Google Plus-
on" href="https://plus.google.com/share?url=openmap.mfgi.hu"><i class="fa fa-
google-plus fa-lg google"></i></a>
       <a class="btn btn-default btn-lg" target="_blank" title="Megosztás LinkedIn-en"
href="https://www.linkedin.com/shareArticle?mini=true&url=openmap.mfgi.hu&title=
Magyarorsz%C3%A1g%20f%C3%B6ldtani%20atlasza&summary=&source="><i
class="fa fa-linkedin fa-lg linkin"></i></a>
      </div>
    </div>
   </div>
   <div id ="popover"></div>
   <div id="map" class="container"></div>
   <div class="zoombuttons">
    <div class="zoom-in">+</div>
    <div class="zoom-out">–</div>
   </div>
   <div class="home"><span class="glyphicon glyphicon-home" aria-
hidden="true"></span></div>
   <div class="searchForm">
    <div class="form-group">
     <input type="text" class="form-control" id="usr" placeholder="Település">
     <span class="searchIcon glyphicon glyphicon-search" aria-hidden="true"></span>
```

```
    </div>
  </div>

  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
  <!-- Latest compiled and minified JavaScript -->
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script
>
  <!--proj4-->
  <script src="lib/proj4/proj4.js"></script>
  <!--Openlayers 3-->
  <script src="https://Openlayers.org/en/v3.19.1/build/ol-debug.js"
type="text/javascript"></script>
  <!--Own scripts-->
  <script src="src/config.js"></script>
  <script src="src/map.js"></script>
  <script src="src/info.js"></script>
  <script src="src/print.js"></script>
  <script src="src/gui.js"></script>
  <!--<script src="src/all.js"></script>-->
 </body>
</html>
```

gui.js

```
var Gui = {};

Gui.updateLayout = function () {
 $('#map').height(window.innerHeight - 40);
 $('#map').width(window.innerWidth);
```

```
  $('#headerController').width(window.innerWidth);
  $('.panel-body').css('height', window.innerHeight - 110);
};

Gui.addEventHandlers = function () {
  $(window).on('resize', function () {
    Gui.updateLayout();
  });
  $('.icon-container').each(function(){
    this.onclick = function() {};
  });
  //show panels on icon click
  $('.icon-container').on('click', function (e) {
    if (!$(e.currentTarget).hasClass('active')){
      Gui.hidePanels();
      $('.icon-container').removeClass('active');
    }
    var id = e.currentTarget.getAttribute('id');
    var sel = '#panel' + id;
    $(sel).toggle();
    $(e.currentTarget).toggleClass('active');
  });
  //hide panels on x click
  $('.panel-heading-custom .glyphicon-remove').on('click', function (e) {
    var id =
e.currentTarget.parentElement.parentElement.parentElement.getAttribute('id');
    sel = '#' + id;
    $('.icon-container').removeClass('active');
    $(sel).toggle();
  });

  $('.panelContent').on('show.bs.collapse',function(){
```

```
  $('.panel-heading-custom .minimizer').removeClass('glyphicon-chevron-
down').addClass('glyphicon-chevron-up');
 });
 $('.panelContent').on('hide.bs.collapse',function(){
  $('.panel-heading-custom .minimizer').removeClass('glyphicon-chevron-
up').addClass('glyphicon-chevron-down');
 });
 $('.zoom-in').on('click', function () {
  var zoom = ol.animation.zoom({
   resolution:Map.map.getView().getResolution(),
   duration:200,
   easing:ol.easing.inAndOut
  });
  Map.map.beforeRender(zoom);
  Map.map.getView().setZoom(Map.map.getView().getZoom()+ 1);
 });
 $('.zoom-out').on('click', function () {
  var zoom = ol.animation.zoom({
   resolution:Map.map.getView().getResolution(),
   duration:200,
   easing:ol.easing.inAndOut
  });
  Map.map.beforeRender(zoom);
  Map.map.getView().setZoom(Map.map.getView().getZoom()- 1);
 });
 $('.home').on('click',function(){
  var zoom = ol.animation.zoom({
   resolution:Map.map.getView().getResolution(),
   duration:1000,
   easing:ol.easing.inAndOut
  });
  var pan = ol.animation.pan({
```

```
    source:Map.map.getView().getCenter(),
    easing:ol.easing.inAndOut
  });
  Map.map.beforeRender(zoom,pan);
  Map.map.getView().setZoom(Config.map.viewOptions.zoom);
  Map.map.getView().setCenter(Config.map.viewOptions.center);
 });

 $('#printButton').click('click',Print.addLink);
};

Gui.hidePanels = function(){
 $('.panelBase').hide();
};

$('document').ready(function () {
 Gui.updateLayout();
 Gui.addEventHandlers();
 Map.create();
 Map.addCoordinateWidget();
 Map.addMapproxyLayer();

 Info.init();
});
```

map.js

```
var Map = {};

Map.create = function () {
```

```
  proj4.defs("EPSG:23700", "+proj=somerc +lat_0=47.14439372222222
+lon_0=19.04857177777778 +k_0=0.99993 +x_0=650000 +y_0=200000 +ellps=GRS67
+towgs84=52.17,-71.82,-14.9,0,0,0,0 +units=m +no_defs");
  Map.map = new ol.Map({
    layers: [],
    target: 'map',
    view: new ol.View(Config.map.viewOptions),
    renderer: 'canvas'
  });
};

Map.defineBaseLayers = function () {
  var osm = new ol.layer.Tile({
    source: new ol.source.OSM(),
    preload: Infinity,
    title: Config.map.baseLayers.osm.title,
    name: 'osm'
  });

  var stamenTerrain = new ol.layer.Tile({
    source: new ol.source.Stamen({layer: 'terrain'}),
    title: Config.map.baseLayers.stamenTerrain.title,
    name: 'stamenTerrain'
  });

  Map.allBaseLayers = [osm, stamenTerrain];
};

Map.setBaseLayer = function (blname) {
  if (Map.baseLayer) {
    Map.clearBaseLayer();
  }
```

```
  for (var i = 0; i < Map.allBaseLayers.length; i++) {
    if (Map.allBaseLayers[i].get('name') == blname) {
      Map.baseLayer = Map.allBaseLayers[i];
    }
  }

  Map.map.getLayers().insertAt(0, Map.baseLayer);
};

Map.clearBaseLayer = function () {
  Map.baseLayer = null;
  Map.map.getLayers().removeAt(0);
};

Map.addTopicLayer = function () {
  var url =
'http://89.135.122.205:8081/geoserver/budatest/wfs?service=wfs&version=2.0.0&req
uest=getfeature&outputformat=application/json&typenames=budatest:jarda_5_wm&
count=400';

  var vectorSource = new ol.source.Vector({
    url: url,
    format: new ol.format.GeoJSON()
  });

  Map.resultLayer = new ol.layer.Vector({
    source: vectorSource,
    renderOrder: null,
    style: function (feature, resolution) {
      var defaultStyle = {
        'Point': [new ol.style.Style({
           image: new ol.style.Circle({
```

```
      fill: new ol.style.Fill({
        color: 'rgba(255,255,0,0.5)'
      }),
      radius: 10,
      stroke: new ol.style.Stroke({
        color: '#ff0',
        width: 1
      })
    })
  })],
'LineString': [new ol.style.Style({
    stroke: new ol.style.Stroke({
      color: '#f00',
      width: 3
    })
  })],
'Polygon': [new ol.style.Style({
    fill: new ol.style.Fill({
      color: 'rgba(0,0,255,0)'
    }),
    stroke: new ol.style.Stroke({
      color: '#0ff',
      width: 2
    })
  })],
'MultiPoint': [new ol.style.Style({
    image: new ol.style.Circle({
      fill: new ol.style.Fill({
        color: 'rgba(255,0,255,0.5)'
      }),
      radius: 5,
      stroke: new ol.style.Stroke({
```

```
            color: '#f0f',
            width: 1
          })
        })
      })],
    'MultiLineString': [new ol.style.Style({
        stroke: new ol.style.Stroke({
          color: '#0f0',
          width: 3
        })
      })],
    'MultiPolygon': [new ol.style.Style({
        fill: new ol.style.Fill({
          color: 'rgba(0,0,255,0)'
        }),
        stroke: new ol.style.Stroke({
          color: '#00f',
          width: 2
        })
      })]
  };
  return defaultStyle[feature.getGeometry().getType()];
 }
});

 Map.map.getLayers().insertAt(2, Map.resultLayer);
};

Map.addTiledLayer = function () {
 var url =
'http://map.mfgi.hu/arcgis/rest/services/public/map_furaspont_wm/MapServer/';
```

```
  var tiledlayer = new ol.layer.Tile({
    source: new ol.source.TileArcGISRest({
      url: url
    })
  });
  Map.map.getLayers().insertAt(1, tiledlayer);

};

Map.addMapproxyLayer = function () {
  var MapproxyLayer = new ol.layer.Tile({
    preload: Infinity,
    opacity: 0.9,
    source: new ol.source.TileWMS({
      url: 'http://openmap.mfgi.hu/Mapproxy/service',
      params: {
        layers: Config.map.Mapproxymap
      },
      tileGrid: new ol.tilegrid.TileGrid({
        extent: [400000, 0, 1000000, 400000],
        resolutions: [423.3333333333, 211.6666666667, 105.8333333333,
52.9166666667, 26.4583333333, 13.2291666667]
      }),
      projection: 'EPSG:23700'
    })
  });
    Map.map.getLayers().insertAt(1, MapproxyLayer);
};

Map.addCoordinateWidget = function(){
```

```
  Map.map.addControl(new ol.control.MousePosition({projection: 'EPSG:23700',
coordinateFormat: ol.coordinate.createStringXY(), undefinedHTML: '', className: 'eov-
mouse-position'}));
}

Map.getCurrentScale = function ()
{
  var map = Map.map;
  var view = map.getView();
  var resolution = view.getResolution();
  var units = map.getView().getProjection().getUnits();
  var dpi = 25.4 / 0.28;
  var mpu = ol.proj.METERS_PER_UNIT[units];
  var scale = resolution * mpu * 39.37 * dpi;
  return scale;

}
```

info.js

```
var Info = {};

Info.init = function () {
  //setting the Info.popover value
  Info.popover = new ol.Overlay({
    element: document.getElementById('popover'),
    autoPan: true,
    positioning: 'center-center'
  });
  Map.map.addOverlay(Info.popover);
  //adding close to x button
```

```
  //setting the global Info.coordinate value
  if (ol.has.TOUCH) {
    Map.map.getViewport().addEventListener('touchstart', function (e) {
      Info.touchStartPixel = Map.map.getEventPixel(e);
    });
    Map.map.getViewport().addEventListener('touchend', function (e) {
      Info.touchEndPixel = Map.map.getEventPixel(e);
      var deltaX = Math.abs(Info.touchStartPixel[0] - Info.touchEndPixel[0]);
      var deltaY = Math.abs(Info.touchStartPixel[1] - Info.touchEndPixel[1]);
      if (deltaX + deltaY < 6) {
        Info.coordinate = Map.map.getEventCoordinate(e);
        Info.renderResult(Info.coordinate);
      }
    });
  } else {
    Map.map.on('singleclick', function (e) {
      Info.coordinate = e.coordinate;
      Info.renderResult(Info.coordinate);
    });
  }

};

Info.renderResult = function (coordinate) {
  Info.popover.setPosition(coordinate);
  console.log(coordinate);
  $('#popover').popover('destroy');
  var title = 'Elem információ';
  var items = '';
  var carousel = '<ol class="carousel-indicators">';

  //returnedFeatures is an object, containing named arrays containing the features
```

```
  Info.getFormattedFeatureInfo(Info.getFeatureInfoURLforCoordinate(coordinate),
function (returnedFeatures) {
    console.log(returnedFeatures);
    var count = 0;
//key is the name of the layer
    for (key in returnedFeatures) {
      if (returnedFeatures.hasOwnProperty(key)) {
        //array of features from one layer
        var featuresFromOneLayer = returnedFeatures[key];
        //for each feature get Aliased fields based on Config.map.queryLayers array
        for (var i = 0; i < featuresFromOneLayer.length; i++) {
          var feature = featuresFromOneLayer[i];
          for (var i = 0; i < Config.map.queryLayers.length; i++) {
            //if this feature is part of a configured queryLayer
            if (Config.map.queryLayers[i].name == key) {
              //object containing alias/name pairs
              var fieldsWithAlias = Config.map.queryLayers[i].template;
              //add key/value pairs to featureTable
              var featureTable = '';
              for (keyname in fieldsWithAlias) {
                if (fieldsWithAlias.hasOwnProperty(keyname)) {
                  var fieldAlias = fieldsWithAlias[keyname];
                  var fieldValue = feature[keyname];
                  var row = '<b>' + fieldAlias + '</b>' + ' : ' + fieldValue + '<br>';
                  featureTable += row;
                }
              }
              if (count == 0) {
                items += '<div class="item active">' + featureTable + '</div>';
                carousel += '<li data-target="#myCarousel"
onclick="$(&amp;quot;#myCarousel&amp;quot;).carousel(' + count + ');" data-slide-to="0"
class="active"></li>';
```

```
            } else {
              items += '<div class="item">' + featureTable + '</div>';
              carousel += '<li data-target="#myCarousel"
onclick="$(&quot;#myCarousel&quot;).carousel(' + count + ');" data-slide-to="' + count
+ '"></li>';
            }

          }
        }
        count++;
      }
    }
  }

  carousel += '</ol>';

  var content = '<div class="container"><div id="myCarousel" data-interval="false"
class="carousel slide"><div class="carousel-inner" role="listbox">'
        + items + '</div>' + carousel + '</div>\n\
<a class="car-control left-control" data-target="#myCarousel" role="button"
onclick="$(&quot;#myCarousel&quot;).carousel(&quot;prev&quot;);"><span
class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span><span class="sr-
only">Previous</span></a>\n\
<a class="car-control right-control" data-target="#myCarousel" role="button"
onclick="$(&quot;#myCarousel&quot;).carousel(&quot;next&quot;);"><span
class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span><span class="sr-
only">Next</span></a>\n\
</div>';

  $('#popover').popover({
    'placement': 'auto',
    'animation': false,
```

```
    'html': true,
    'title': title + '<button type="button" id="close" class="close"
onclick="$(&amp;quot;#popover&amp;quot;).popover(&amp;quot;hide&amp;quot;);">&amp;times;</button>',
    'viewport': '#map',
    'content': content
   });
   if (count > 0) {
    $('#popover').popover('show');
   }

   if (count > 1) {
    $('.car-control').show();
    $('.carousel-indicators').show();
   } else {
    $('.car-control').hide();
    $('.carousel-indicators').hide();
   }
  });
};

Info.getFeatureInfoURLforCoordinate = function (coordinate) {
  var url = Map.map.getLayers().getArray()[0].getSource().getGetFeatureInfoUrl(
        Info.coordinate,
        Map.map.getView().getResolution(),
        Map.map.getView().getProjection(),
        {'INFO_FORMAT': 'text/xml', 'query_layers': 'fdt100'}
  );
  return url;
};

Info.getFormattedFeatureInfo = function (url, callback) {
  var returnedFeatures = {};
```

```
  $.ajax(url, {dataType: 'xml'}).done(function (data) {
//    console.log(data);
    $xml = $(data);
    $xml.find('Layer').each(function () {
      var layername = $(this).attr('name');
      if (Info.getQueryLayers().indexOf(layername) == -1) {
        return;
      }
      $(this).find('Feature').each(function () {
        var feature = [];
        $(this).find('Attribute').each(function () {
          feature[$(this).attr('name')] = $(this).attr('value');
        });
        if (!returnedFeatures[layername]) {
          returnedFeatures[layername] = [];
        }
        returnedFeatures[layername].push(feature);
      });
    });
    callback(returnedFeatures);
  });
};

Info.getAliasFieldsForQueryLayer = function (layername) {
  var fields = [];
  for (var i = 0; i < Config.map.queryLayers.length; i++) {
    var ql = Config.map.queryLayers[i];
    if (ql.name !== layername) {
      continue;
    } else {
      fields = Object.keys(ql.template);
    }
```

```
  }
  return fields;
};

Info.getQueryLayers = function () {
  layers = [];
  for (var i = 0; i < Config.map.queryLayers.length; i++) {
    layers.push(Config.map.queryLayers[i].name);
  }
  return layers;
};
```

print.js

```
var Print = {};

Print.generateURL = function () {
  var url =
'http://openmap.mfgi.hu/wms/fdt100_new/fdt100_new?request=GetPrint&SERVICE=
WMS&version=1.3.0&TEMPLATE=Fdt%20100&format=pdf&CRS=EPSG:23700&WIDTH
=1467&HEIGHT=729&LAYERS=fdt100.DBO.fdt_pg,fdt100_szinek,fdt100.DBO.szik_pg,fd
t100.DBO.tekt_szveg_l,fdt100.DBO.fdt_pgl,fdt100.DBO.teler_l,viz_pg,telepules_kat2,t
elepules_kat1a,telepules_kat1,ut,viz_l,hatar&STYLES=,&DPI=300&TRANSPARENT=true'
;

  var bbox = Map.map.getView().calculateExtent(Map.map.getSize());

  var bboxstring = bbox[0].toString() + ',' + bbox[1].toString() + ',' + bbox[2].toString() +
',' + bbox[3].toString();

  url+= '&map0:EXTENT='+bboxstring+'&BBOX=' + bboxstring;
  return url;
```

```
};

Print.addLink = function(){
  var url = Print.generateURL();
  $('#panelprintContent').append('<br/><a href="'+ url +'" target="_blank">Térkép le-
töltése</a>');
};
```

config.js

```
var Config = {
  map: {
    viewOptions: {
      zoom: 3,
      projection: 'EPSG:23700',
      resolutions: [423.3333333333, 211.6666666667, 105.8333333333,
52.9166666667,26.4583333333, 13.2291666667],
      center: [653224.1858820765, 241128.1638166387]
//      center: [2164580, 5971464],
//      zoom: 2,
//      resolutions: [2445.98490512564, 1222.99245256282, 611.49622628141,
305.748113140705, 152.8740565703525, 76.43702828517625, 38.21851414258813,
19.109257071294063, 9.554628535647032, 4.777314267823516]
    },
    Mapproxymap:'fdt100_new',
    queryLayers: [
      {name: 'fdt100.DBO.fdt_pg',
        template: {
          'nev': 'Név',
          'geo_ndx': 'Földtani index'
        }
      }
```

```
    ],
    baseLayers: {
      stamenTerrain: {
        title: 'Stamen Terrain'
      },
      osm: {
        title: 'OpenStreetMap'
      }
    },
    defaultBaseLayer: 'osm'
  }
};
```

[38]ADATLAP
a doktori értekezés nyilvánosságra hozatalához

I. A doktori értekezés adatai

A szerző neve: Simó Benedek
MTMT-azonosító: 10054004
A doktori értekezés címe és alcíme: Földtani adatok kartografált, interaktív megjelenítése a weben open-source eszközök segítségével
DOI-azonosító[39]
A doktori iskola neve: Földtudományi Doktori Iskola
A doktori iskolán belüli doktori program neve: Térképész program
A témavezető neve és tudományos fokozata: Elek István, PhD
A témavezető munkahelye: ELTE, Térképtudományi és Geoinformatikai Tanszék

II. Nyilatkozatok

A doktori értekezés szerzőjeként[40]
a) hozzájárulok, hogy a doktori fokozat megszerzését követően a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az ELTE Digitális Intézményi Tudástárban. Felhatalmazom a Természettudományi Kar Dékáni Hivatalának Doktori, Habilitációs és Nemzetközi Ügyek Csoportja ügyintézőjét, hogy az értekezést és a téziseket feltöltse az ELTE Digitális Intézményi Tudástárba, és ennek során kitöltse a feltöltéshez szükséges nyilatkozatokat.
b) kérem, hogy a mellékelt kérelemben részletezett szabadalmi, illetőleg oltalmi bejelentés közzétételéig a doktori értekezést ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[41]
c) kérem, hogy a nemzetbiztonsági okból minősített adatot tartalmazó doktori értekezést a minősítés *(dátum)*-ig tartó időtartama alatt ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[42]
d) kérem, hogy a mű kiadására vonatkozó mellékelt kiadó szerződésre tekintettel a doktori értekezést a könyv megjelenéséig ne bocsássák nyilvánosságra az Egyetemi Könyvtárban, és az ELTE Digitális Intézményi Tudástárban csak a könyv bibliográfiai adatait tegyék közzé. Ha a könyv a fokozatszerzést követőn egy évig nem jelenik meg, hozzájárulok, hogy a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban.[43]
2. A doktori értekezés szerzőjeként kijelentem, hogy
a) az ELTE Digitális Intézményi Tudástárba feltöltendő doktori értekezés és a tézisek saját eredeti , önálló szellemi munkám és legjobb tudomásom szerint nem sértem vele senki szerzői jogait;
b) a doktori értekezés és a tézisek nyomtatott változatai és az elektronikus adathordozón benyújtott tartalmak (szöveg és ábrák) mindenben megegyeznek.
3. A doktori értekezés szerzőjeként hozzájárulok a doktori értekezés és a tézisek szövegének plágiumkereső adatbázisba helyezéséhez és plágiumellenőrző vizsgálatok lefuttatásához.

Kelt: 2017.03.09.

.....
a doktori értekezés szerzőjének aláírása

[38] Beiktatta az Egyetemi Doktori Szabályzat módosításáról szóló CXXXIX/2014. (VI. 30.) Szen. sz. határozat. Hatályos: 2014. VII.1. napjától.
[39] A kari hivatal ügyintézője tölti ki.
[40] A megfelelő szöveg aláhúzandó.
[41] A doktori értekezés benyújtásával egyidejűleg be kell adni a tudományági doktori tanácshoz a szabadalmi, illetőleg oltalmi bejelentést tanúsító okiratot és a nyilvánosságra hozatal elhalasztása iránti kérelmet.
[42] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a minősített adatra vonatkozó közokiratot.
[43] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a mű kiadásáról szóló kiadói szerződést.