EÖTVÖS LORÁND TUDOMÁNYEGYETEM INFORMATIKAI KAR

TÉRKÉPTUDOMÁNYI ÉS GEOINFORMATIKAI TANSZÉK
TÉRKÉPÉSZ MESTERSZAK

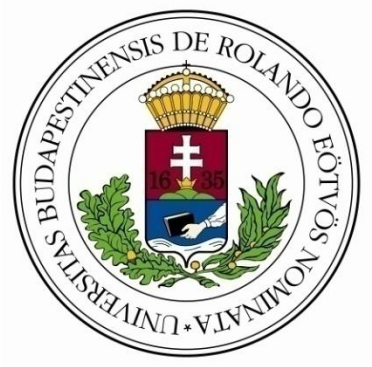
Interaktív térképes földrajzi névmutató készítése

DIPLOMAMUNKA

Készítette:
Patkó Máté
térképész hallgató

Témavezető:
Dr. Gede Mátyás
egyetemi docens

ELTE Térképtudományi és Geoinformatikai Tanszék



Budapest, 2019

Tartalom

Bevezetés

Diplomamunkám célja egy interaktív névmutató elkészítése webes környezetben. Alapját egy kétszáz éves magyarországi földrajzi leírás adja. E könyv címe *Magyar Országnak és a határörző katonaság vidékinek legújabb statistikai és geográphiai leírása (1819)* írta Magda Pál. Az évszám jelzi, hogy a könyv még a nyelvújítás előtt íródott, így nyelvezete eléggé elavult. Olvasását ez egy kissé kényelmetlenné teszi, de odafigyelve könnyen értelmezhető a szöveg. A könyv nem tesz említést az Erdélyi Fejedelemségről, valamint a dalmát térségről. Ahogy a cím is mutatja, csak a Királyi Magyarországról, emellett a katonai határőrvidékekről szól. A könyvben egy átfogó leírást kapunk az országról földrajzi, társadalmi és igazgatási értelemben egyaránt. Az elején olvashatunk az általános természeti állapotról, gazdasági tényezőkről, nemzetekről, vallásokról stb. A munkám szempontjából fontos rész a 180. oldaltól kezdődik. Innentől fogva a könyv átfogóan ismerteti az ország földrajzi állapotát. Megyénként haladva ír a településekről, vizekről, hegységekről, emellett a számomra kevésbé fontos információkat is közöl. A könyv végén található egy névmutató, amelynek feldolgozása a dolgozatom legfontosabb részét képezi.

Az interaktív névmutató létrehozásához a könyvet digitalizálni kellett. Ezt egy többlépcsős folyamattal oldottam meg. Első lépésben el kellett készítenem a könyv digitális képekből álló változatát. Ezt a dolgozat további részében részletesen ismertetem. A továbbiakban a képeket egy karakterfelismerő szoftver segítségével feldolgoztam, hogy a könyv digitális szövegként is elérhető legyen. Ezután a névmutatót készítettem el. Ez egy adattábla, melyet az alapadatokon kívül több fontos információval egészítettem ki. Ahhoz, hogy térképes legyen a névmutató, az elkészült táblázatot adatbázissá kellett alakítanom. A későbbiekben a megfelelő GIS szoftvert és alkalmazást használva az adatokat geokódoltam. Ezután mindezt webes környezetben jelenítettem meg, tehát weblapon. Az oldalon szinkronnézetben megtekinthetők a könyvről készült képek és a szöveg, melyhez egy betűrendben elkészített névmutatót társítottam. A geokódolt adatokat szintén térképen jelenítettem meg interaktív környezetben.

1. Adatnyerés

A könyvből történő adatnyerésnek két lehetősége van: szkennelés vagy fényképezés. A könyv 1819-ben készült el. Az állapota napjainkra eléggé leromlott, így a szkennelés lehetőségét elvetettem. Az észszerű opció a könyv lefényképezése volt. A munkához elengedhetetlen volt egy jó minőségű fényképezőgép, ugyanis képbeállítások finomhangolása miatt ajánlott egy professzionális gép használata. Alkalmas fényképezőgéppel nem rendelkeztem, így Vörös Fannitól kértem kölcsön egy *Panasonic DMC-FZ72* típusú eszközt.

A továbbiakhoz szükségem volt egy speciális fényképező állványra, pontosabban egy könyvszkenner állványra, amelynek segítségével felülnézetből finomhangolt beállításokkal fényképezhettem, az alá helyezett könyvet. Az állvány beállítása lehetővé tette, hogy sorozatszerűen és végig ugyan olyan paraméterekkel készíthessek képeket az állványra helyezett könyvről. A könyvszkenner állványt Nemes Zoltán biztosította számomra.

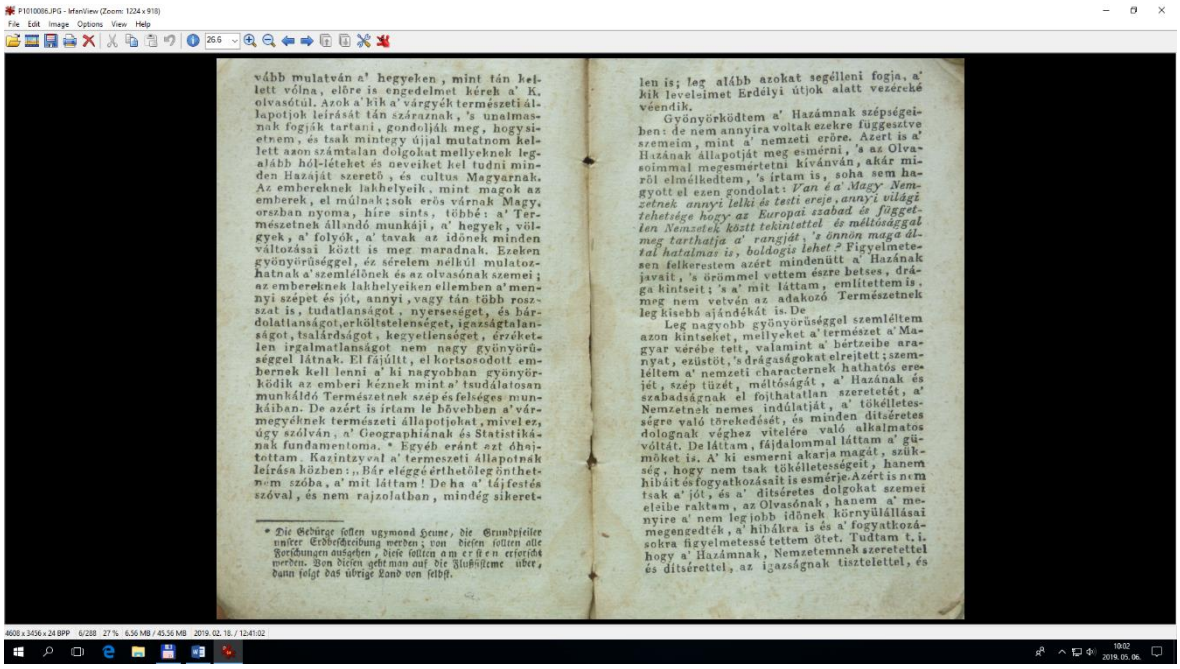
(1. ábra) Az állvány és a fényképező

A fényképzés során fontos volt, hogy egyenletes, lehetőleg természetes fény világítsa meg a felületet, ugyanis a vaku villanó fénye túl erősen világíthatja meg és ez a kép minőségét és olvashatóságát ronthatja. Az oldalakat egyesével lefényképeztem, így összesen 301 képet készítettem. A jó minőség érdekében 16MPx-re állítottam a felbontást. A képeken a szöveg nem mindenütt volt megfelelő, emiatt néhány oldalt utólag finomhangolt beállításokkal újrafényképeztem.

2. OCR lefuttatása

Az optikai karakterfelismerés előtt (Optical Character Recognition - OCR) a létrejött képfájlokat preparálni kellett a folyamathoz.

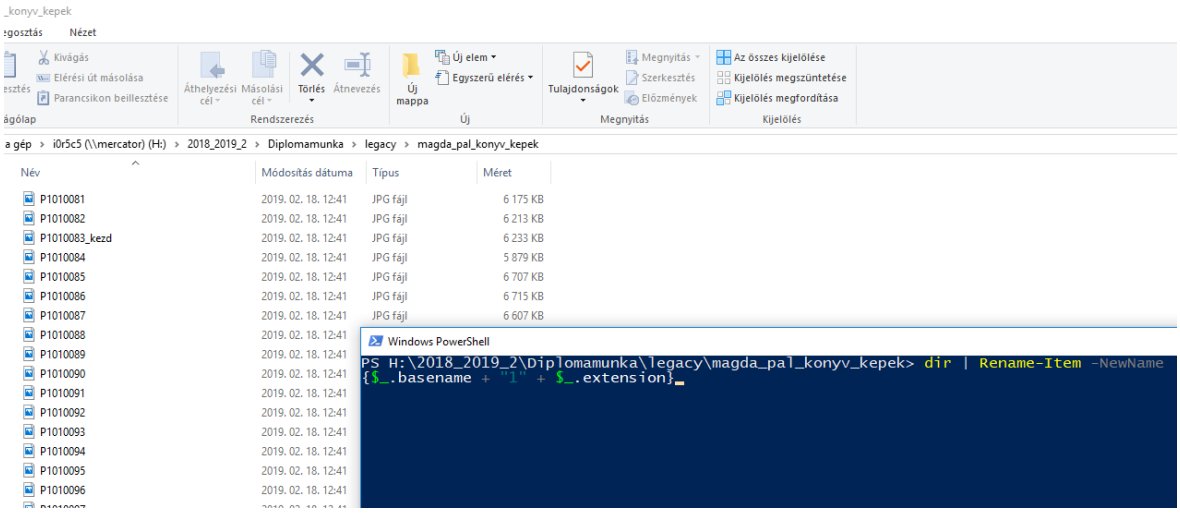
Egy fájl a könyv egy-egy oldalát tartalmazta. A további műveletekhez ezeket ketté kellett vágnom, ehhez az IrfanView képszerkesztő szoftvert alkalmaztam. A program sok hasznos funkcióval rendelkezik, de sajnos több fájlt nem képes egyszerre kezelni, ennek következtében a képeket egyesével választottam szét.



(2. ábra) IrfanView

A további műveletekhez a fájlok neveit egyszerűsíteni és egységesíteni kellett. Mivel több, mint hatszáz kép készült a művelet során, a manuális átnevezés hosszas és körülményes munka lett volna. A szétválasztás során a képek sorrendisége sérült, ugyanis a vágás minden második kép fájlnevét felcserélte az előzővel. A felmerült problémát a Windows

PowerShell-el és a Windows parancssorral oldottam meg. Néhány parancs alkalmazásával a sorrend helyre állt, a fájlneveket pedig rendre (1.jpg, 2.jpg…) neveztem át. A fájlok így már könnyen kezelhetők lettek és egyértelműen megőrzik sorrendjüket.



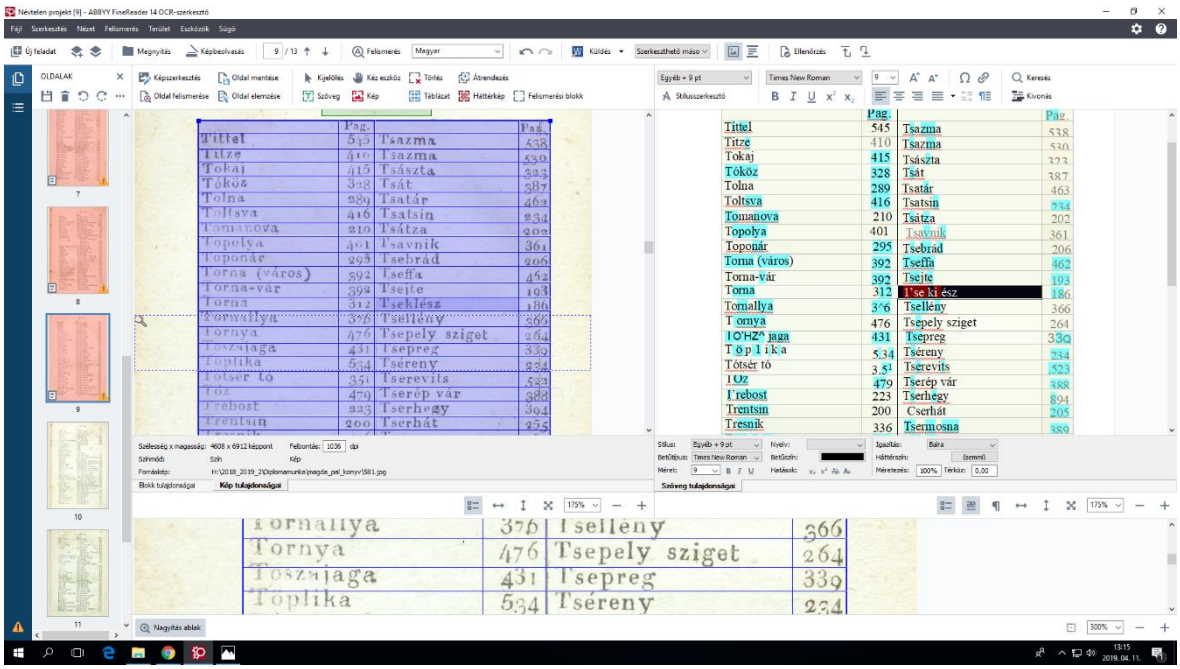
(3. ábra) Fájlok átnevezése

Ezután már elkezdhettem az optikai karakterfelismerést (továbbiakban OCR). Az OCR segítségével raszteres fájlok, így a fényképek, szkennelt lapok tartalmát szövegfájlokká alakíthatjuk (Finereader, 2017). A könyv oldalait vizsgálva, több kritériumnak is meg kellett felelnie a kiválasztott programnak. Legfontosabb, hogy magyar karaktereket képes legyen felismerni, az oldalbeállítások tükrözzék az eredeti formátumot és szinkron nézetben javítható legyen a felismert szöveg. Több program használatát ezek hiányában el kellett vetnem. A legoptimálisabbnak az Abbyy Finereader nevű szoftver bizonyult. A program a fent említett kritériumoknak megfelel és több hasznos funkcióval is rendelkezik, ezen felül a kezelőfelülete egyszerű és könnyen tanulható.

A készített képek 16MPx-esek, a könyv oldalai 18 x 11,3 cm-esek. Ahhoz, hogy a virtuális térben a lapok megőrizzék méretüket, át kellett állítanom az oldalak felbontását. A fent említett adatok alapján átszámolva az optimális képfelbontás nagyjából 518 DPI-nek (dot per inch) felelt meg. Néhány tesztolvasást követően kiderült, hogy ez a szám túl alacsonynak bizonyult. A megfelelő ennek duplája, 1036 DPI lett. A tesztlapok mérete így már megegyezett az eredetivel.

Elsőként a könyv legvégét, vagyis a névmutatót olvastattam be. A nagyjából 27 oldalas *névmutató* beolvasását a lehető legpontosabban kellett elvégeznem, ugyanis ez adta a

készülő adatbázis gerincét. Az OCR táblázatos felismerést is támogat, így az adatokat már táblázatszerűen kezelhettem a karakterfelismerés után.



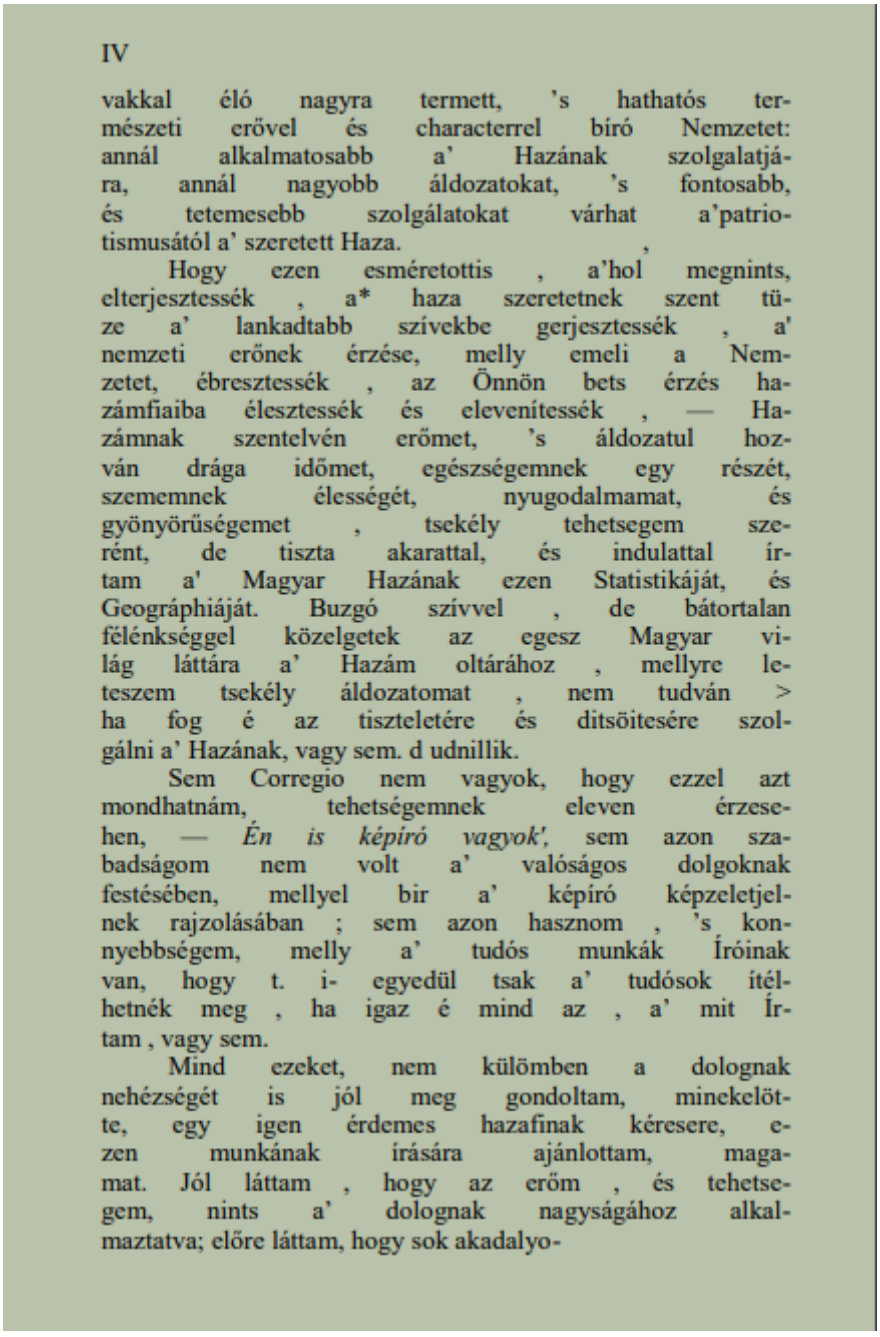
(4. ábra) Abbyy Finereader

A táblázat duplahasábos szerkezete miatt sajnos nem exportálhattam Excel formátumba az adatokat. A rendezés egyszerűbbnek bizonyult, ha szövegfájlként kezeltem, majd Excelbe helyeztem már rendezett alakban. Ez későbbiekben az interaktív névmutató alapja lett, ezért átkonvertáltam CSV formátumra UTF-8-as karakterkódolással. A fájl neve *nevm.csv*. A nevek a *nev_archaikus*, a hozzájuk tartozó oldalszám pedig az *oldal* mezőben jelenik meg. Ezzel a lépéssel a névmutató törzse kialakult.

Ezután az OCR-t a könyv megmaradt részeire alkalmaztam. A megmaradt 550 oldalt a lehető legjobban automatizált módon ismertettem fel. A képek beolvasása után digitális korrekciót hajtottam végre minden egyes lapon, ezzel kiegyenesítettem a sorokat, a domború lapokat kisimítottam és a deformációkat kiegyenlítettem. Az Abbyy program képszerkesztő részében az ajánlott automatikus kezdőfeldolgozást alkalmazva az előbb említett problémák jelentősen javultak. A program ezután már elég pontosan meg tudta határozni a felismerési blokkokat, így elkezdhettem a karakterfelismerést. A végeredmény magas százalékban megbízhatónak bizonyult.

A felismert szöveget szövegfájlként mentettem el, majd PDF fájlformátummá konvertáltam. Mivel minden egyes oldal külön képfájlként létezik, így a szöveges dokumentumot is fel

kellett darabolni külön oldalakra. Ezt a *www.ilovepdf.com* oldal segítségével oldottam meg. A fájl feltöltése után a szétválasztó műveletet lefuttattam, majd az elkészült fájlokat egy tömörített állományban letöltöttem. (iLovePDF, 2019)



IV

vakkal éló nagyra termett, 's hathatós természeti erővel és characterrel bíró Nemzetet: annál alkalmatosabb a' Hazának szolgalatjára, annál nagyobb áldozatokat, 's fontosabb, és tetemesebb szolgálatokat várhat a'patriotismusától a' szeretett Haza.

Hogy ezen esméretottis , a'hol megnints, elterjesztessék , a* haza szeretetnek szent tüze a' lankadtabb szívekbe gerjesztessék , a' nemzeti erőnek érzése, melly emeli a Nemzetet, ébresztessék , az Önnön bets érzés hazámfiaiba élesztessék és elevenítessék , — Hazámnak szentelvén erőmet, 's áldozatul hozván drága időmet, egészségemnek egy részét, szememnek élességét, nyugodalmamat, és gyönyörüségemet , tsekély tehetsegem szerént, de tiszta akarattal, és indulattal írtam a' Magyar Hazának ezen Statistikáját, és Geográphiáját. Buzgó szívvel , de bátortalan félénkséggel közelgetek az egesz Magyar világ láttára a' Hazám oltárához , mellyre leteszem tsekély áldozatomat , nem tudván > ha fog é az tiszteletére és ditsöitesére szolgálni a' Hazának, vagy sem. d udnillik.

Sem Corregio nem vagyok, hogy ezzel azt mondhatnám, tehetségemnek eleven érzesehen, — *Én is képíró vagyok',* sem azon szabadságom nem volt a' valóságos dolgoknak festésében, mellyel bir a' képíró képzeletjelnek rajzolásában ; sem azon hasznom , 's konnyebbségem, melly a' tudós munkák Íróinak van, hogy t. i- egyedül tsak a' tudósok ítélhetnék meg , ha igaz é mind az , a' mit Írtam , vagy sem.

Mind ezeket, nem külömben a dolognak nehézségét is jól meg gondoltam, minekelötte, egy igen érdemes hazafinak kéresere, ezen munkának írására ajánlottam, magamat. Jól láttam , hogy az erőm , és tehetsegem, nints a' dolognak nagyságához alkalmaztatva; előre láttam, hogy sok akadalyo-

(5. ábra) Beolvasott PDF

3. Adatbázis előkésztése

A táblázat törzse létrejött a névmutató beolvasásával. A táblázat további mezőit manuálisan létrehoztam, amelyek az alábbi ábrán láthatók.

MEZŐK	PÉLDA
id	*309*
nev_archaikus	*Drenovátz*
nev_modern	*Darnóc vára*
nev_modern_idegen	*SlatinskiDrenovac*
geoc	*SlatinskiDrenovac*
kozig	*Verőce vármegye*
mai_kozig	*Eszék-Baranya megye*
orszag	*Horvátország*
oldal	*523*
tipus	*var*
valtozas	*kieg*
megjegyzes	*A névalak kiegészült*
hiba	*Nincs a megjelölt oldalon.*
manual_geoc	*X*
geoc_megjegy	*Darnóc mellett.*

(6. ábra) A táblázat szerkezete

Legelőször a kulcsmezőt, vagyis az *id* mezőt alkottam meg. Ez a mező biztosítja minden egyes rekord egyediségét, azonosíthatóságát és sorrendjét a táblázaton belül. A *nev_archaikus* a könyvben megtalálható eredeti névalakot mutatja. Ez a mező a névmutatóban megjelölt névalakra hivatkozik. Több esetben sajnos ez nem minősült megbízhatónak, ugyanis a törzsszöveg és a névmutatóban található névalakok közt gyakran eltérést figyeltem meg. A *nev_modern* mezőben az adott névalak mai, modern, magyar nyelvű névalakját jelöltem meg. Magyarország területe a trianoni békediktátum életbe lépését követően jelentősen lecsökkent. A hivatalos névalakok megváltoztak, ezért a *nev_modern_idegen* mezőben megjelöltem az adott objektum jelenlegi államnyelvi névalakját. A *geoc* a legfontosabb, ugyanis ez volt az alapja a munka további részének. A

mező tartalmazhatja a *nev_modern*, illetve a *nev_modern_idegen* alakokat, viszont több esetben ezektől teljesen eltérő is lehet. A geokódolás e mező alapján történik meg, ezt a következő fejezetben részletezem. A *kozig* tartalmazza a korabeli közigazgatási egység nevét, mely az objektumot tartalmazza. Ekkor vármegyék, kiváltságos kerületek és határőrvidékek alkották a magyar közigazgatási egységeket (Faragó, 2014). A *mai_kozig*-en belül a jelenlegi közigazgatási egységek nevei találhatók. Az Európai Unió tagállamaiban, valamint Szerbiában a NUTS3-as igazgatási szintet jelöltem meg. Horvátország, Magyarország és Románia esetében ez a megye, Szlovákiában és Szerbiában a kerületek/körzetek, Szlovéniában a statisztikai régiók. Ausztria esetében NUTS2-es szintet használtam, vagyis a tartományokat, ugyanis területileg ezek közelebb állnak az előbb említett egységek méreteihez (Eurostat, 2019). A NUTS rendszer alól Ukrajna kivétel, így az „oblaszty" közigazgatási egységet jelöltem meg, ami a terület esetében csak Kárpátalját jelentette. Az *orszag* mezőben láthatjuk, hogy az adott objektum mely ország határain belül található. Az *oldal* mutatja az adott név helyzetét a könyvben. A *tipus* mező az objektum tulajdonságát mutatja. Ez lehet: *telepules, var, hegy, mocsar* stb. A *valtozas* mező a névalak megváltozásáról tájékoztat. A sorok itt is többféle értéket tartalmazhatnak pl.: kieg – kiegészült, atalakult – átalakult stb. A *megjegyzes* mezőben a névalak változást részletezem.

A táblázat készítése során többször is problémába ütköztem, ugyanis a névmutató és a törzsszöveg között rengeteg eltérést tapasztaltam. Több esetben egy adott név nem volt található a hozzá kapcsolt oldalon, vagy más névalakban fordult elő a törzsszövegben. Helyesírási hiba, redundancia egyaránt felmerült. E problémák megjelölésére hoztam létre a *hiba* mezőt, melyben megjelöltem az adott rekordhoz tartozó eltéréseket. Néhány esetben úgy véltem, hogy szükség lehet manuális igazításokra a geokódolás után. Ezt egy X-szel jelöltem meg a *manualgeoc* mezőben. A *geoc_megjegy* az esetleges manuális elhelyezéshez nyújt segítséget. Az elkészült fájlt *db.xslx* néven mentettem.

Az eredmények:

Összes név	2055
Megtalált	1642
Ebből redundancia	26
Nem található	414
Hibák	
Különbség a szövegben és a névmutatóban.	97
Nem volt a megjelölt oldalon.	79

(7. ábra) Eredmények

4. Geokódolás

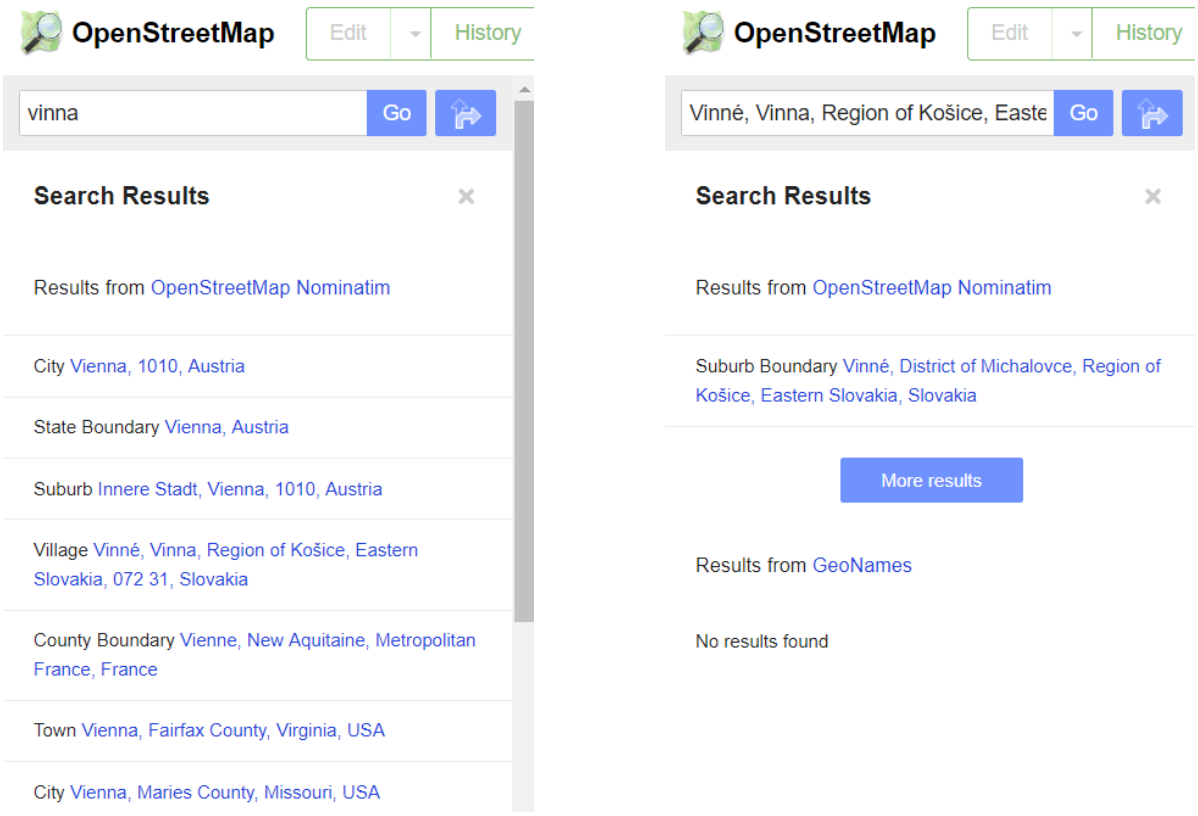
A geokódolás során egy adatbázis adott mezőjéhez térbeli adatokat rendelünk, így azokat térképen meg tudjuk jeleníteni. Legoptimálisabb esetben az adatbázisunk már rendelkezik egy olyan mezővel, mely a geometriát tartalmazza valamilyen szám formátumban. Ha nem áll rendelkezésre adat, akkor string típussal is lehet geokódolni a megfelelő módszert alkalmazva. (Gede, 2019) Esetemben a geokódolást a *geoc* mező fogja szolgáltatni.

A folyamat elkezdéshez az adatokat át kellett konvertálnom CSV fájlformátumba. Fontos, hogy a karakterkódolás UTF-8-as legyen, ugyanis a Microsoft programcsaládnál nem ez az alapértelmezett. Számomra fontos az UTF-8, ugyanis cirill és a speciális latin karakterek is szerepelnek a táblázat szövegében. A továbbiakban minden esetben ezt a karakterkódolást használom (Egmont, 2005). A fentebb említett *id* mezőt megszüntettem, ugyanis a mezőnek csak a táblázat szerkesztésénél volt jelenősége.

A geokódolást a QGIS (Quantum GIS) szoftverrel készítem el. A program alapvető alkalmazásaival a geokódolás nem lehetséges, de a *Geocode Sqlite table* nevű bővítménnyel ez megoldható. Az alkalmazás SQLite kapcsolatot igényel. Az SQLite egy kliensoldali, szervert nem igénylő SQL (StructuredQueryLanguage) adatbázis motor (Hipp, 2019). A QGIS kezelőfelületén létrehoztam az adatbázis–kapcsolatot, melyhez hozzárendeltem a korábban elkészített CSV fájlt. A kapcsolat kiépítésekor fontos, hogy létrehozzon egy új *id*

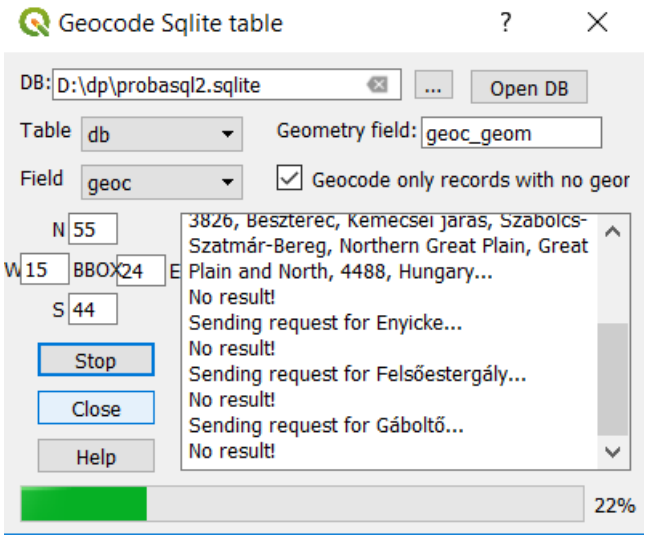
mezőt. A korábbival ellentétben ez már egészszám formátumú lesz, így a tábla minden egyes rekordjához hozzárendel egy spatial indexet, ami térbeli egyedi azonosítóként szolgál. A többi mező továbbra is szöveg típusú maradt.

A táblázat így már adatbázissá alakult. Térbeli azonosítóval rendelkezik, de még nem tartalmaz térbeli adatot. A fentebb említett bővítmény szöveg típusú adatbázis mezők alapján kérdez le térbeli adatokat az OSM-től (OpenStreetMap) (Gede, 2019). A táblázatom *geoc* mezőjével egy lekérdezést indítunk el. Ahogy korábban említettem a *geoc* nem csak földrajzi neveket tartalmaz, hanem komplex megjelöléseket is az esetleges hibák kiküszöbölése végett. Például Vinna(HU)/Vinné(SK) egy kis falu Szlovákiában a Kassai kerületben. Ha az OSM keresőbe beírjuk ezt a nevet első találatként Bécset adja meg. Ez abból adódik, hogy Bécsnek az angol névalakja Vienna és ez nagyon közel áll a Vinna/Vinné névalakhoz. Az OSM közel azonos névalakok miatt, így mindig Bécset fogja azonosítani a keresés során. Ez súlyos hibát eredményez a geokódolásban, viszont ezt egyszerűen ki lehet küszöbölni. A település neve mellett közöljük az ország nevét, valamint a bennfoglaló különböző szintű közigazgatási egységek neveit. Ez térben korlátozza a keresést és csökkenti a találatok számát. E módszert alkalmazva a kétes helyzetű települések is könnyen lokalizálhatók. *(8. ábra)*



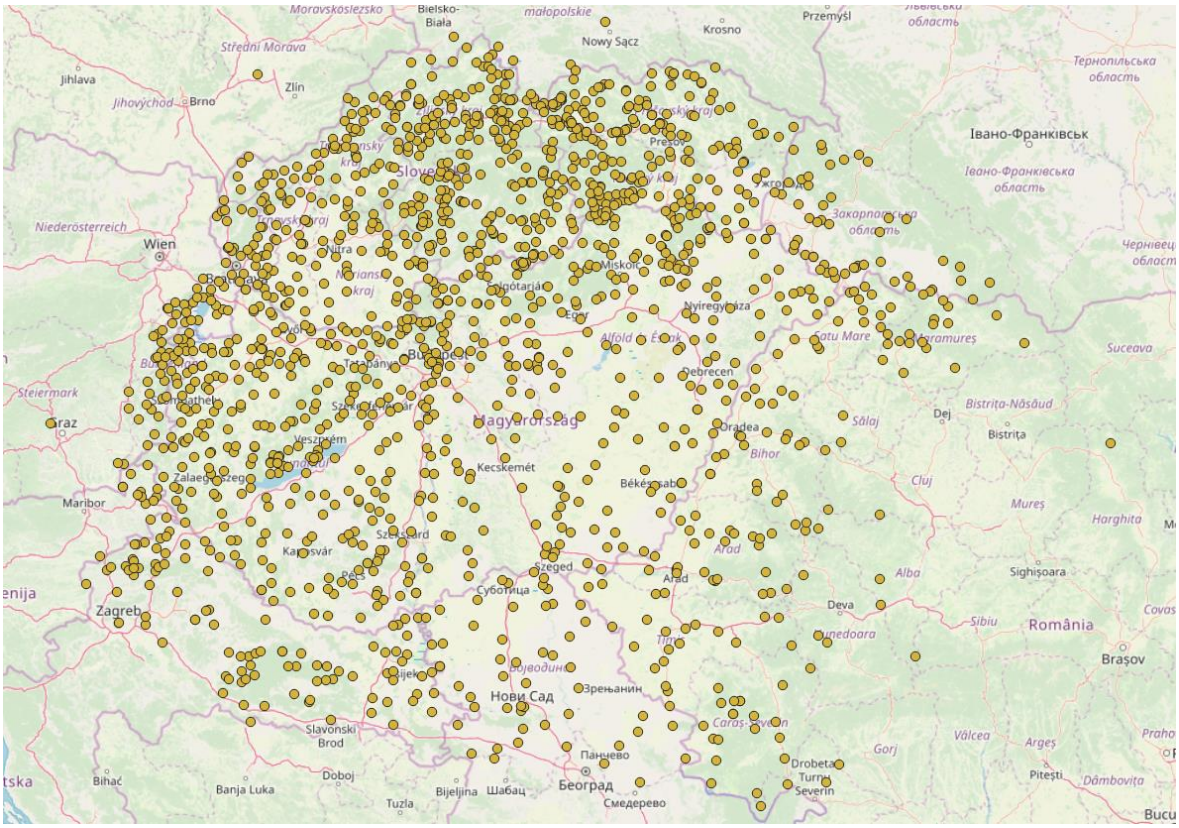
(8. ábra) OSM keresés

Így elkezdhettem a geokódolást. Az alkalmazásban először ki kellett választanom az adatbázis-kapcsolatot, majd azon belül a táblát. Meg kellett adnom azt a mezőt, amely alapján a program a geokódolást végzi. A területet lehatároltam földrajzi koordináták megadásával (Gede, 2019).



(9. ábra) Geokódolás

A folyamat végeztével az adatbázis tartalmazta a térbeli adatokat és vektoros layerként jelenik meg, s így behívható a QGIS kezelőfelületébe. Az adatok pontszerűen jelennek meg. Az elkészült layert *adatok.shp* néven mentettem el. *(10. ábra)*



(10. ábra) A geokódolt adatok

5. A közigazgatás és a folyóvizek layerek

A geokódolással elkészült térkép fő tematikája elkészült. Ahhoz, hogy szemléletes legyen a megjelenítés, digitalizáltam a korabeli közigazgatási határokat és a vízrajzot.

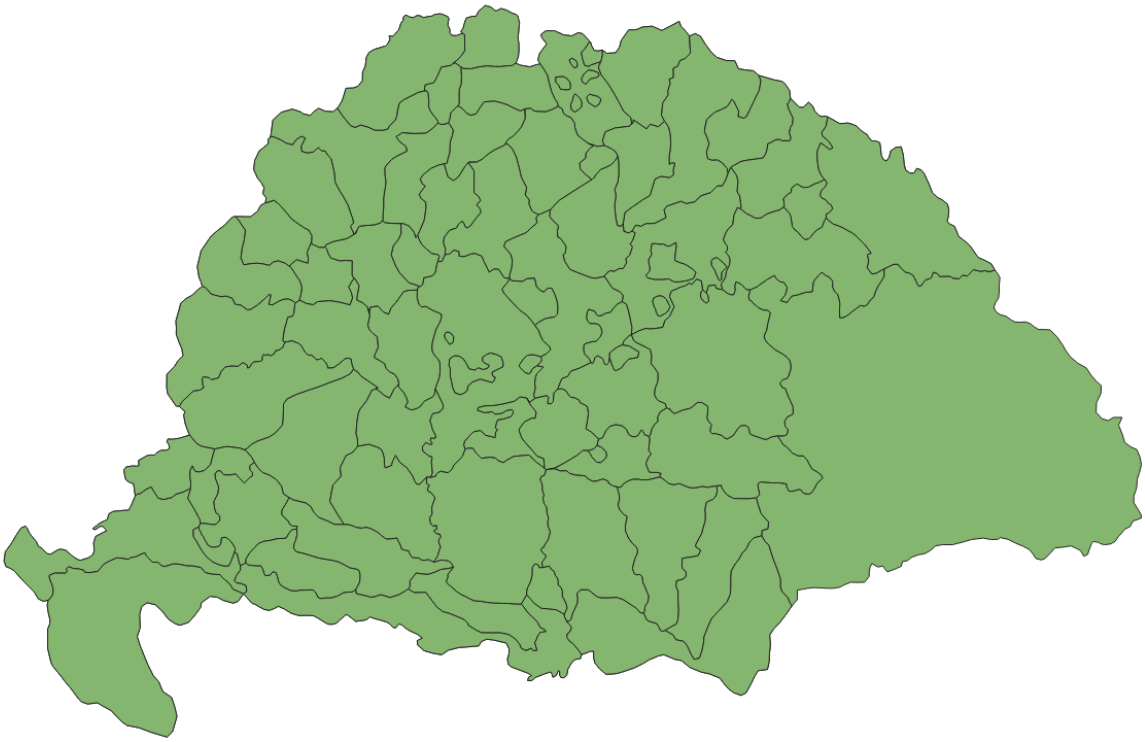
A XIX. századi közigazgatás jelentősen eltért a maitól. A törökök kiűzése után a hódoltság területén a magyar igazgatási rendszert újjászervezték. A Királyi Magyarország közigazgatási egységei a vármegyékből, kiváltságos területekből és a katonai őrvidékekből tevődtek össze. Az egyes egységek egymástól függetlenül működtek, csak a magyar államiság kötötte össze őket. A közigazgatási egységek megújulása a területi viszonyokat nézve igen érdekes formákat mutatott. Az átszervezés során több megye megszűnt, egyesült, vagy korábbi területi viszonyai megváltoztak. Több vármegyének is keletkezett exklávéja,

így például a korábbi Külső-Szolnok vármegye egy a Berettyó menti darabja Heves vármegye exklávéjaként működött tovább. (Faragó, 2014)

A digitalizálást QGIS szoftverrel végeztem el. A munka alapját a Cartographia Középiskolai Történelmi atlaszából szkenneltem be (Cartographia, 2009). A képet georeferáltam QGIS-en belül. A folyamat elkezdéséhez egy új shapefájl layert készítettem, amit *kozig.shp*-nek neveztem el. A geometriáját poligonok adják. A layerhez tartozó adatbázisban megalkottam az egyedi azonosításra szolgáló *id*-t, a *kozige*-t (az egyes egységek neveit jelöli) és a *tipus* mezőt, mely az adott közigazgatási egység jellegét adja meg. Mivel poligonokról van szó, a digitalizálás során ügyelnem kellett arra, hogy ne lépjenek fel topológiai problémák, tehát hézag és átfedésmentes felületeket kellett alkotnom. A QGIS fejlesztett digitalizáló eszköztárával ez könnyen megvalósítható, ugyanis ezzel helyes topológiájú, tehát hézag és átfedésmentes poligonokat lehet készíteni.

Az elkészült layert felhasználva kialakítottam az országhatárt és a közigazgatási egységeket határoló vonalas réteget. Ezt a *geoprocsessing tools* alkalmazásával oldottam meg. A *dissolve*-val egyesítettem a Magyar Királyság területére vonatkozó poligonokat. A *dissolve* egyesíti egy layer azon elemeit, melyek attribútum adatai megegyeznek. A *geometry tools*-on belül a *Polygons to lines*-szal konvertáltam a poligonokat vonalas réteggé, így kialakult az országhatár réteg, amit ideiglenes fájlként mentettem el. A megyék körvonalaihoz, *kozig.shp*-ból újra vonalas réteget készítettem. A kialakult két ideiglenes rétegen *difference*-szel megszüntettem a két réteg átfedéseit. A rétegeket ezután *merge*-dzsel egyesítettem, majd *kozighat.shp*-nek neveztem el.

A vízfolyásokat vonalas elemként jelenítettem meg a térképen. Ehhez a *www.geofabrik.de* weboldalról szereztem be a megfelelő adatokat. Az oldalon shapefájlként országokra lebontva találhatók az adatok, így minden érintett országhoz tartozó állományt le kellett töltenem. A munkámhoz szükséges adatokat manuálisan válogattam ki a layerekből, ugyanis az egyes vízfolyások különböző névváltozatai miatt az automatikus keresés nem volt megoldható. Az adatnyerés után a darabokban lévő vonalakat neveik alapján egyesítettem, majd hozzákapcsoltam a korábban elkészült adattábla megfelelő elemeit, létrehozva ezzel a *vizfolyas.shp* vonalas réteget (Geofabrik, 2019).



(11. ábra) Az elkészült felület

6. Mapfájl

Munkám egyik fő célja az adatok térképes ábrázolása. A webes megjelenítéshez a MapServer programot használtam WMS (Web Map Service) adatszolgáltatási protokollon keresztül. A WMS egy szerver-kliens oldali kapcsolatot biztosít, aminek lényege, hogy a szolgáltatás a szerverről hálózaton keresztül közöl térbeli raszteres adatokat a klienssel, melyből lekérhetjük a szerveroldali vektoros adatokat. (AdatTárKép, 2017). A MapServer számára a térkép leírását egy MAP fájlban adtam meg. Mivel az egyes layereken belül az adatok már kategorizálva vannak, így csak azok megjelenését szükséges definiálni. A mapfájl egy szövegfájl, aminek a szerkezetét objektumok adják. Minden objektum elejét a megfelelő kulcsszó jelöli, a végüket pedig az END mutatja. A legfontosabb objektum a MAP, mely a szerkezet törzséül szolgál, minden más ezen belül található (Gede, 2017).

6.1 Kliens oldal

Első lépésben létrehoztam a MAP objektumot, ezután megadtam a címet. Mivel a mapfile raszteres térképet hoz létre, ezért a kiterjesztést és annak méretét pixelben meg kellett adnom. Az adatok térbeli megjelenésének határait földrajzi koordinátákkal adtam meg, ezek rendre nyugati, déli, keleti és északi koordináták. A raszteres fájlt a program a felhasznált vektoros layerek alapján generálja, így azok elérési útját meg kellett adnom. Fontos, hogy olyan helyen legyenek eltárolva, aminek olvashatósága nincs korlátozva, az adatok így távoli eszközökön is megtekinthetők. Az alkalmazott vetület EPSG számát jelöltem meg, mely jelen esetben a 3857, ez a Pszeudo Mercator vetület kódja.

```
MAP
 NAME "Magyar Országnak és a határörzö katonaság vidékinek legújabb statistikai és geográphíai leirása"
 SIZE 800 600
 IMAGETYPE png
 EXTENT 15 44 25 50
 SHAPEPATH "/home/hal2015/i0r5c5/public_html/DIPLOMAMUNKA/layers"
 PROJECTION
  "init=epsg:3857"
 END
…
END
```

A SYMBOL objektum segítségével jelek sémáit hozhatjuk létre, melyek megkönnyítik a grafika elkészítését. Egy szimbólumot készítettem, amit „pont”-nak neveztek el, így a későbbiekben ez alapján hivatkozhatok rá. Megadtam az objektum típusát, ami jelen esetben ellipszis. Ennek méretét a tengelyek végeinek megadásával állítottam be, mivel ugyanazon két számot írtam, ezért egy szabályos ellipszist, tehát kört hoztam létre. Ahhoz, hogy a létrejött kör kitöltését a továbbiakban jelkategóriák szerint színezhessem, a FILLED tulajdonságot „true”-ra kellett beállítanom. A SYMBOL segítségével képfájlokra is lehet hivatkozni, így a már előre megrajzott jeleket is megjeleníthetjük. Munkámban több térképi elemet is e módon ábrázoltam.

```
…
SYMBOL
  NAME "pont"
  TYPE ellipse
  POINTS 1 1 END
  FILLED true
 END
...
```

A LEGEND objektum segítségével jelmagyarázatot generáltam a megjelenített adatokból. Ez szintén képként jelenik meg, melynek háttérszínét úgy állítottam be, hogy azonos legyen a weblapéval. A jelek kis téglalapokban tűnnek fel, melyek méretét 40x20 pixelre állítottam.

```
…
LEGEND
IMAGECOLOR "#ffe4c4"
 LABEL
TYPE truetype
FONT sans
SIZE 12
COLOR 0 0 0
OFFSET 0 -5
END
  KEYSIZE 40 20
 END
...
```

Ahhoz, hogy a térbeli adatokat megjelenítsem, legalább egy LAYER-t definiálnom kellett. A nevét, a típusát, a vetületét, az adatfájl helyét és a szolgáltatásokat engedélyező utasításokat adtam meg.

```
…
LAYER
 NAME "Jelek"
 TYPE annotation
 PROJECTION
  "init=epsg:3857"
 END
 DATA "adatok.shp"
 STATUS on
FONTSET "libfonts.list"
….
```

A LAYER-eken belül a CLASSITEM adja meg, melyik attribútum alapján akarjuk osztályozni az elemeket. A layeren belül osztályokat (CLASS) hoztam létre, az EXPRESSION-nel az értéküket adtam meg. A térképi névrajzot a LABEL objektum segítségével jelenítettem meg, majd a szöveg stílusbeállításit definiáltam.

```
…
CLASSITEM "tipus"
LABELITEM "nev_archai"
CLASS
 NAME "Település"
 EXPRESSION "telepules"
 STYLE
  SYMBOL "pont"
  SIZE 10
  COLOR 255 255 255
  OUTLINECOLOR 30 30 30
 END
 LABEL
  TYPE truetype
   FONT sans
  (…)
 END
END
…
```

Az átláthatóság végett, a *mapfájl*-ban az objektumok szintjét az előttük álló *szóköz* leütések száma jelzi, valamint a fontosabb END-ek mögött *hashmark*-kal (#) megjegyzésben megjelöltem az objektum nevét.

6.2 WMS

A webes megjelenítés során az adatokat interaktív közegben ábrázoltam. A WMS rendelkezik a GetFeatureInfo funkcióval, mely alkalmazással a kliensoldalra küldött raszteres állományra kattintva egy kérést küld a szerveroldalra, beazonosítva ezzel a képkoordináták alapján a vektoros állományt, aminek az adatait megjeleníti a kliensoldalon.

Ahhoz, hogy ez működjön, a mapfájlon belül kellett meghívnom a WMS szolgáltatást. A MAP objektumban a *"wms_enable_request" "*"*-al aktiváltam, e nélkül nem működne a WMS szolgáltatás. A *"wms_feature_info_mime_type" "text/html"* biztosítja, hogy a kérések szövege HTML formátumban jelenjen meg.

```
…
WEB
 METADATA
  "wms_enable_request" "*"
  "wms_srs" "epsg:3857"
  "wms_feature_info_mime_type" "text/html"
 END
END
...
```

A WMS teljeskörű eléréshez az összes layeren belül engedélyeztettem a szolgáltatást. A megfelelő paraméterek megadásával, a kívánt adatokat lekérhetjük HTML formátumban, melyek a *wmstemplate.html/wmstemplateviz.html* fájlok segítségével jelennek meg. E fájlokban a megjelenítendő layer mezőneveit adtam meg, a kérés ezek alapján mutatja meg a lekért adatokat. A mapfájlon belül a WMS szolgáltatások hozzáadásával elkészültem, viszont ez még nem elég funkciója teljes körű betöltéséhez. Ahhoz, hogy működjön, a JavaScript kódban is aktiválnom kellett.

```
…
METADATA
  "wms_enable_request" "getmap"
  "wms_include_items" "all"
 END
 TEMPLATE "wmstemplate.html"
...
```

7. Weblap elkészítése

A weblap elkészítéshez a Notepad++ szöveg- és forráskódszerkesztő programot használtam. A program ismeri és kezelni is tudja a számomra fontos programnyelveket, a HTML-t, CSS-t és a JavaScriptet egyaránt.

7.1 HTML

A HTML (HyperText Markup Langauge) a weblapok fejlesztésben használatos elsődleges leíró nyelv. Legfrissebb változata a HTML5 (Nonsense, 2019).

A weblapot *index.html*-nek neveztem el. A HTML elemek legtöbbje egy kezdő és végelemből tevődik össze. A HTML törzsét a <html>…</html> adja, ezelőtt szerepel <!DOCTYPEhtml>, ami beállítja, hogy a böngésző a fájlt HTML dokumentumként kezelje. A <head>… </head> elem a metaadatokat, címet, elérési utakat, a beépülő modulokat és technikai információt tartalmaz. Minden, ami a <head>-en belül található, a böngészőablakban nem jelenik meg. A webböngésző ablakában csak az jelenik meg, ami a <body>…</body> elemen belül található. A HTML fájlon belül az egyes szinteket a gyakorlatnak megfelelően behúzásokkal jelöltem meg (Gede, 2017).

A megjelenítést a legtöbb esetben elemekben végeztem el. A -ek olyan HTML tárolóelemek melyeken nagyobb blokkokat foglalhatunk csoportba. A -eket egymásba ágyazva könnyen felépíthettem a weblap struktúráját, ezek stílusbeállításait CSS-ben írtam meg. A -eket egyedi azonosítóval láttam el, így minden egyes elemet külön definiálhattam.

A <head>-en belül állítottam be az oldal technikai adatait. A <metacontent>-ben a HTML szöveg karakterkódolását állítottam át UTF-8-ra. Hivatkoztam a stílusbeállítást tartalmazó CSS-re, melyet külön fájlban írtam meg. A <title> elemben a weblap címét, végül pedig a beépülő JavaScript modulokat adtam meg.

```
…
<head>
	<meta content="text/html; charset=utf-8" http-equiv="content-type">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<link rel="stylesheet" type="text/css"   href="css/style.css" />
	<title>Magyar Országnak és a határörzö katonaság (…)</title>
	<script src="js/pdf.js" type="text/javascript"></script>
	<script src="http://openlayers.org/api/OpenLayers.js" type="text/javascript"></script>
</head>
…
```

A <body> első eleme a , melyen belül további három a „header”, a „show” és a „footer” található. A „header"-ben leírtam a könyv és egyben az oldal címét. A „show”-ban több elemet egymásba ágyazva, az weboldal legfontosabb része, a könyv, a névmutató és a térkép jelenik meg. A „footer”-ben kolofont helyeztem el.

```
…
<body>
	<div id="main">
		<div id="header">
		…
		</div>
		<div id="show">
		…
		</div>
		<div id="footer">
		…
		</div>
	</div>
…
</body>
```

A „show”-n belül elsőként a „text” található, mely az oldal rövid leírását tartalmazza. A „book”-on belül több elemet ágyaztam egymásba. A weblap bal oldalán a könyvről készített PDF állomány található. A fájlok HTML szövegként egy , képként pedig egy <canvas> elemben jennek meg. A <canvas> egy olyan HTML elem, amiben JavaScript utasítások segítségével raszteres rajzot lehet létrehozni. Az oldal közepén a könyvről készült fényképek találhatók egy <img> elemben. Az <img> elem a HTML képtároló objektuma, melyben hivatkozással jeleníthetjük meg a fájlokat. A jobb oldalon a névmutató helyezkedik el, a betűk és az alatta listázott nevek külön elemben jelennek meg. A könyv alatt a lapozó funkciót helyeztem el. A gombokkal, egyesével lehet előre illetve hátra lapozni,

valamint az első és utolsó oldalakra ugrani. A gombok között található egy elem, melyben feltűntettem aktuális oldal számát és emellett keresőmezőként is üzemel. A , hasonlóan a -hez egy tárolóelem, azonban ezt csak sorközi megjelenítésre használható (w3schools, 2019).

```
…
<div id="book">
	<div id="page">
		<div id="pdftarto">
			<canvas id="pdfcanvas">
			</canvas>
			<div id="pdf">
			</div>
		</div>
		<img id="pic">
		<div class="buttons">
			<button (…) onclick="showPage(1)">(…)</button>
			<button (…) onclick="if (oldal>1) showPage(--oldal)">(…)</button>
			<span id="searchbar"><input id="keres" type="text" maxlength="3"> / 602</span>
			<button (…) onclick="if (oldal<10) showPage(++oldal)">(…)</button>
			<button (…) onclick="showPage(602)"> (…)</button>
		</div>
	</div>
	<div id="ind">
		<div id="letter">
		</div>
		<div id="disp">
		</div>
	</div>
</div>
…
```

Az oldal alján „map”-on belül jelenítettem meg a térképet és mellette a jelmagyarázatot.

```
…
<div id="map">
	<div id="map_show">
	</div>
	<div id="legend" tabindex="1">
	<h3>Jelmagyarázat</h3>
	<img src="(…)">
	</div>
</div>
…
```

A HTML dokumentum végén két <script> elemet helyeztem el. Az elsőben a névmutatóhoz, a másodikban a térképhez tartozó JavaScript kód található.

7.2 CSS

A CSS (Cascading Style Sheets) egy stítlusleíró nyelv, mellyel a HTML fájlok megjelenését szabályozzuk. CSS-t be lehet ágyazni a HTML kódba a <style> elemek közé, de CSS fájlként is tárolhatjuk. Én az utóbbi megoldást választottam. A kód használatánál először a kívánt HTML elemre kell hivatkoznunk, melyet testre akarunk szabni. A deklarálás során kapcsos zárójelek között megjelöljük a tulajdonságot, majd egy kettőspont után megadjuk annak értékét. A deklarációkat pontosvesszővel választjuk el. (Gede, 2017)

Mivel több HTML elem stílusjegyeit definiáltam, így megjegyzésben megneveztem a különálló részeket a könnyebb átláthatóság végett. CSS-ben a megjegyzéseket „/*…*/” közzé tesszük.

A weblap stílusszerkesztésénél az egyes tárolóelemek helyzetét, valamint a színek értékeit állítottam be. Főleg barnás, sárgás színeket használtam, ezeket hexadecimális kóddal jelöltem meg. Az oldalon megjelenő HTML szövegeket is CSS-sel formáztam.

A stílusbeállításokat a *stlye.css* fájlba rögzítettem.

```
body {
	background-color: #CCAA87;
}
(…)
#pdfspan {
	position: absolute;
	transform-origin: 0% 0%;
	overflow: hidden;
	line-height: 1.0;
	}
#pdf {color : #FFFFFF;
	position: absolute;
	display: inline-block;
	height: 100%;
	width: 100%;
	line-height: 1.0;
	white-space: pre;
	transform-origin: 0% 0%;
	left: 0;
	font-family: times;
	overflow: hidden;
	opacity: 0.2;}
(…)
#letter {
	border-style: solid;
	right: 10px;
	position: relative;
	padding: 10px;
	padding-left: 17px;
	height: 15px;
	border-width: 2px;
	border-color: #634640;
	color: #634640;
}
…
```

7.3 JavaScript

A JavaScript egy objektumalapú programozási nyelv, a leggyakrabban a weblapok interaktív részeinek megírására használják. A program szintaxisa (nyelvezete) hasonlít a C-re. A nyelv maga gyengén típusos, tehát változókat definiálhatunk, de típusukat nem adhatjuk meg. HTML-be a <script> elem segítségével szúrhatunk be JavaScript kódot (Gede, 2017).

7.3.1 Névmutató

Az első <script> elem betölti a könyvről készölt képeket, PDF-eket és névmutatót a helyükre, valamint megadja gombok és a keresőmező funkcióját.

A gombok közötti kereső az aktuális oldal számát mutatja az ’osz’ változó alapján. Ezt a változót egy olyan értékként kezeltem, ami az enter lenyomására keresőmezőben található

szám értékére (oldalra) lapozza a könyvet. Ez egy 0 és 602 közötti szám lehet. A lapozó gombok ehhez a számhoz viszonyítva látják el az adott funkciójukat.

```
…
var osz=document.getElementById('keres');
	osz.onkeyup=function(e) {
		if (e.keyCode==13) {
		var old=parseInt(osz.value);
		if (old>0&&old<603)
			showPage(oldal=old);
		else{
		}
	}
}
…
```

A fájlokat a program nevük alapján kezeli, amiket az oldalszámok alapján adtam meg. Az ’oldal’ változó értéke 1, tehát alaphelyzetben az 1 nevű fájlokat tölti be a program. A ’pdfdiv’ változó, amely a *document.getElementById('pdf')* függvény segítségével fájlokat tölt be. A fájlok nevét *functionshowPage(p)* ’p’ paraméterével jelöltem, így a program ez alapján a JPG kiterjesztésű állományokat betölti a megadott helyre. Ezután az ’osz’ és az ’oldal’ értékét azonosnak tekintettem a ’p’ értékével. A ’loadingTask’ néven új változót készítettem, mely a *pdf.js* modul segítségével a képekhez hasonlóan tölti be a PDF fájlokat. Ezután a program nézetablak ’viewport’ nagyságát adja meg, majd ezt a <canvas> méretével szinkronizálja, e nélkül a behívott adatok elcsúsznak. Ezután képként jeleníti meg a behívott PDF-et canvas-ben a *page.render(renderContext).then(function()* segítségével. Végezetül megjeleníti a PDF fájl karaktereit a canvas felett HTML szövegként (Useful Angle, 2018).

```
…
var oldal=1;
var pdfdiv=document.getElementById('pdf');
	function showPage(p) {
	document.getElementById('pic').src='kepek/'+p+'.jpg';
	osz.value=p;oldal=p;
	var loadingTask = pdfjsLib.getDocument('pdf/'+p+'.pdf');
	loadingTask.promise.then(function(pdf) {
		pdf.getPage(1).then(function(page) {
			var scale = 1.5;
			var viewport = page.getViewport({ scale: scale, });

			var canvas = document.getElementById('pdfcanvas');
			console.log(canvas)
			var context = canvas.getContext('2d');
			canvas.height = viewport.height;
			canvas.width = viewport.width;

			var renderContext = {
				canvasContext: context,
				viewport: viewport
			};
			page.render(renderContext).then(function(){
				return page.getTextContent();
			}).then(function(tc){
				pdfdiv.innerHTML='';
				pdfjsLib.renderTextLayer({
					textContent: tc,
					container: pdfdiv,
					viewport: viewport,
					textDivs: []
				});
			});
		});
	});
}
showPage(oldal);
…
```

A „letter” -ben a névmutató a nevek kezdőbetűi alapján jelenik meg, melyekre kattintva listázza az adott betűkhöz tartozó neveket az alatta elhelyezkedő „disp”-ben. Az itt megjelenő nevekre kattintva a program a könyv kijelölt oldalára lapoz.

A kezdőbetűk fontos, hogy ábécé szerinti sorrendben helyezkedjenek el a „letter”-ben. A betöltött betűk már HTML formátumúak, így a JavaScript ’sort’ függvényével rendezhetők, viszont ez elsősoron a normál latin ábécé betűit rendezi és csak azután minden egyéb speciális latin karaktert, pl.: ö, ő, ü, ű. Mivel több kezdőbetű is ékezetes a ’sort’ függvény így rossz sorrendet és többször megjelenő betűket készít. A megoldás, hogy betöltött ékezetes betűket az ékezet nélküli párjukra kell cserélni, így az összes kezdőbetű sorrendben fog megjelenni (MDN, 2019).

Első lépésben definiáltam az ’ekes’ tömböt, mely tárolja a speciális latin karaktereket, majd az ’ektelen’-t, amely a normál párjukat rendre tartalmazza. Az ’ektelenit’ függvénnyel a

program végigmegy az ’ekes’ tömbön, majd visszatérési értékként megadja az ’ektelen’ párját. A program így már az ékezetes betűket ékezet nélküliként fogja kezelni, viszont ez csak a kisbetűkre érvényes.

```
…
var ekes    =['á','ä','â','ą','ç','č','ď','é','ę','ğ','ı','í','ĩ','ľ','ň','ó','ö','ő','ô','ş','š','ť','ţ','ú','ü','ű','ů','ž'];
var ektelen =['a','a','a','a','c','c','d','e','e','g','i','i','i','l','n','o','o','o','o','s','s','t','t','u','u','u','u','z'];
function ektelenit(b) {
        var i=ekes.indexOf(b);
        if (i>=0)
                return ektelen[i];
        else
                return b;
}
…
```

A folyamat a továbbiakban a névmutatót tölti be. Definiáltam az ’x’ változót az ’XMLHttpRequest’ függvénnyel, így a JavaScript kérések során a program a szerverrel közvetlenül tud kommunikálni az oldal újratöltése nélkül. Ez azért fontos, mert a betűk a szerveren található *nevm.csv*-re hivatkoznak. Az adatok kinyeréséhez meghatározza a CSV fájl sorai végén az elválasztást, így egyenként tudja kezelni azokat. Ezután egy cikluson belül megvizsgálja a sorok hosszát, majd elválassza őket minden sor legelső ';' karakterénél, ugyanis a CSV fájlokban az oszlopok közötti elválasztást ';' jelöli. Ezzel a táblázat minden sorát és első oszlopának adatát külön értékként kezeli a program. Az egyes nevek kezdőbetűit kinyeri a szavakból, majd kisbetűvé alakítja át. Ez azért fontos, mert a korábban említett rendező eljárás, csak a kisbetűket vizsgálja. A következő lépésben lecseréli az ékezetes betűket ékezet nélkülire, majd visszaalakítja nagybetűvé. Végezetül a kezdőbetűket elhelyezi és a közben hivatkozássá alakítja őket, melyek a hozzá kapcsolódó nevekre mutatnak.

```
…
var x=new XMLHttpRequest();
x.open('get','nevm_uj.csv',false);
x.send();
var nevek=x.responseText.split('\n');
var kb="",nevm=[];
for (var i=0;i<nevek.length;i++) {
        var d=nevek[i].split(';');
        if (d.length>1) {
                nevm.push({nev:d[0],oldal:parseInt(d[2])});
                if (d[0][0]!='('&&ektelenit(d[0][0].toLowerCase())!=kb.toLowerCase()) {
                kb=ektelenit(d[0][0].toLowerCase()).toUpperCase();
        document.getElementById('letter').innerHTML+='<a href="javascript:betumutat(\"+kb+'\')">'+kb+'</a> ';
                }
        }
}
…
```

Ahhoz, hogy a nevek megjelenjenek a „disp”-ben létrehoztam a ’betumutat’ függvényt, ezzel a névmutató kezdőbetűi a hozzájuk tartorzó nevekre hivatkoznak. Ezután egy ciklusban a program megvizsgálja a ’nevm’ tömböt, majd a kezdőbetűkön folytatott kisbetű-nagybetű metódust alkalmazva kiválogatja a neveket, majd a tárolót megtölti az oldalhivatkozással ellátott nevekkel.

```
function betumutat(b) {
	var nd=document.getElementById('disp');
	nd.innerHTML='';
	for (var i=0;i<nevm.length;i++) {
		if (ektelenit(nevm[i].nev[0].toLowerCase())==b.toLowerCase())
		nd.innerHTML+='<a href="javascript:showPage('+nevm[i].oldal+')">'+nevm[i].nev+'</a><br/>';
	}
}
```

7.3.2 Térkép

A következő <script> elemen belül a térkép megjelenítő funkcióit írtam meg.

Az első lépésben a program megjeleníti a háttértérképként működő OSM-et. A ’wmsquery’ változót létrehoztam, ami az *OpenLayers.Layer.WMS(…)* függvény felhasználásával, a korábban begyázott ’*OpenLayers.js*’ segítségével WMSlayert hoz létre a megadott URL alapján. Ezen belül a ’layers’-szel megjelöltem azokat a layereket, amik képkoordinátájáról le lehet kérni a kívánt adatokat. A ’format’-tal megadtam a raszteres térkép formátumát. Az ’isBaseLayer’-t *false*-ra, a ’transparent’-et pedig *true*-ra állítottam, így a térkép az OSM felett fog elhelyezkedni (Gede, 2017).

```
…
var map = newOpenLayers.Map('map_show', //OpenLayers térképet behívjuk a <div id="map_show">-ba
	{
	projection: "EPSG:3857",
	} );
var wmsquery = newOpenLayers.Layer.WMS( "MapServer",
			'http://mercator.elte.hu/(…)/map.map',
			{
				layers: ['Jelek','Vizfolyas'],
				format: 'image/png',
				isBaseLayer: false,
				transparent: true
			},
	{ singleTile: true }
);
map.addLayer(wmsquery);
map.addLayer(newOpenLayers.Layer.OSM());
…
```

A WMS GetFeatureInfo szolgáltatást már a mapfájlon belül elkezdem kialakítani. Itt a ’*gf’* változót létrehoztam, majd az *OpenLayers.Control.WMSGetFeatureInfo()* metódussal megadtam a funkcióját. Beállítottam, hogy a kérés szövege HTML formátumban jelenjen meg. A ’p’ deklarálása után az *OpenLayers.Popup.FramedCloud(…)* függvény megmondja, hogy a kérés egy szöveg buborékban jelenjen meg a hivatkozott x és y képi és térképi koordináták alapján, majd megadtam a megjelenő ennek méretét. Végezetül engedélyeztem a GetFeature szolgáltatást, valamint az OpenLayers kicsinyítő és nagyító gombjait megjelenítettem térképen és megadtam alapnézet szélső koordinátáit.

```
…
var gf=newOpenLayers.Control.WMSGetFeatureInfo(
        {
        infoFormat: "text/html",
        layers: [wmsquery],
        eventListeners:
                {
                        getfeatureinfo: function(e)
                        {
                        if (e.text!="")
                                {
                                var p=new OpenLayers.Popup.FramedCloud(…);
                                p.maxSize=newOpenLayers.Size(300,300);
                                map.addPopup(p);
                                }
                        }
                }
        });
map.addControl(gf);
gf.activate();
map.addControl(newOpenLayers.Control.Scale());
map.zoomToExtent(newOpenLayers.Bounds(1800000,5700000,2600000,6300000));
…
```

Összegzés

Diplomamunkám egy soklépcsős feladatnak bizonyult. Munkám során fényképezéssel, szövegfelismeréssel, fájlkezeléssel, táblázatkezeléssel, térbeli adatbázis kezeléssel, weblap és mapszerver készítésével foglalkoztam. Sok dologgal találkoztam és rengeteg új ismerettel gazdagodtam.

A fényképezés egy viszonylag egyszerű, de időigényes feladat volt. Az elkészült képek feldolgozása során több probléma is felmerült, főleg az OCR alkalmazásával. A szövegfelismerés nem sikerült tökéletesen, de a lényegi részét a névmutatót kis javításokkal korrigáltam. A táblázat mezőit többször átalakítottam, mígnem létrehoztam egy jó struktúrát.

A táblázat kitöltése a bizonyult leghosszabbnak a munka során. Több mint 2000 név adatait manuálisan tápláltam be a táblába. A felkutatásuknál az archaikus névalakok miatt többször is problémába ütköztem, de idővel ezt is meg tudtam oldani. A térképhez készítettem egy tematikus hátteret a szemléletesség végett. A mapfájlt a korábbi ismereteim alapján készítettem el. Webes környezet kialakításánál a CSS, HTML részeket könnyen meg tudtam oldani. A JavaScript esetében segítségre szorultam, de végül ez is elkészült. Összességében elmondhatom, hogy a célomat elértem és egy könnyen értelmezhető webes interaktív térképes névmutatót készítettem.

Irodalomjegyzék

AdatTárKép. (2017). *AdatTárKép*. Forrás: https://adatterkep.com/qgis-wms-reteg-hasznalata

Cartographia. (2009). *Középiskolai Történelmi Atlasz.*

Egmont, K. (2005). *Unicode, UTF-8*. Forrás: http://www.cs.bme.hu/~egmont/utf8/

Eurostat. (2019). *Eurostat*. Forrás: https://ec.europa.eu/eurostat/web/nuts/background

Faragó, I. (2014). *A Magyar közigazgatás változásai.* Faragó Imre.

Finereader, A. (2017). Forrás: https://help.abbyy.com/hu-hu/finereader/12/overview

Gede, M. (2017). Forrás: http://mercator.elte.hu/~saman/hu/okt/webgis.html

Gede, M. (2019). *Geocode Sqlite table*. Forrás: https://samanbey.github.io/geocode_sqlite/

Geofabrik. (2019). Forrás: http://download.geofabrik.de/europe.html

Hipp, D. R. (2019). *SQLite homepage*. Forrás: https://www.sqlite.org/index.html

iLovePDF. (2019). *iLovePDF*. Forrás: www.ilovepdf.com

MDN. (2019). Forrás: MDN web docs: https://developer.mozilla.org/hu/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

Nonsense, Q. (2019). *HTML.com*. Forrás: https://html.com/#What_is_HTML

Useful Angle. (2018). Forrás: https://usefulangle.com/post/90/javascript-pdfjs-enable-text-layer

w3schools. (2019). Forrás: w3schools.com: https://www.w3schools.com/html/

Köszönetnyilvánítás

Köszönöm Gede Mátyás az útmutatást, a munka során nyújtott segítségét és témaötletet.

Köszönöm, Vörös Fanninak, hogy kölcsönadta a fényképezőgépét.

Köszönöm Nemes Zoltánnak, hogy biztosította számomra a fényképező állványt.

És mindazoknak, akik bármilyen segítséget nyújtottak munkám során.

DIPLOMAMUNKA LEADÁSI

és

EREDETISÉG NYILATKOZAT

Alulírott ...……..……Neptun-kód: ……..……..

az Eötvös Loránd Tudományegyetem Informatikai Karának, Térképtudományi és Geoinformatikai Tanszékén

……..

……..... című diplomamunkámat a mai napon leadtam.

Témavezetőm neve: ……

CD-t / DVD-t mellékelek *(aláhúzandó)*: igen nem

Büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom, hogy jelen szakdolgozatom/diplomamunkám saját, önálló szellemi termékem; az abban hivatkozott szakirodalom felhasználása a szerzői jogok általános szabályainak megfelelően történt.

Tudomásul veszem, hogy szakdolgozat/diplomamunka esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

A témavezető által benyújtásra elfogadott szakdolgozat PDF formátumban való elektronikus publikálásához a tanszéki honlapon

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2019. december 16.

……..

hallgató aláírása