EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

Topográfiai térképek webes térképkatalógusa

DIPLOMAMUNKA
TÉRKÉPÉSZ MESTERSZAK

Készítette:
Gyetván Tamás

Témavezető:
Dr. Zentai László
tanszékvezető
ELTE Térképtudományi és Geoinformatikai Intézet

Budapest, 2021

EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR
TÉRKÉPTUDOMÁNYI ÉS GEOINFORMATIKAI TANSZÉK

DIPLOMAMUNKA TÉMABEJELENTŐ

Hallgató adatai:
Név: Gyetván Tamás
Neptun kód: I6OFM4

Képzési adatok:
Szak: térképész, mesterképzés (MA/MSc)
Tagozat: Nappali

Belső témavezetővel rendelkezem

Témavezető neve: *Dr. Zentai László*
munkahelyének neve: *ELTE IK Térképtudományi és Geoinformatikai Tanszék*
munkahelyének címe: *1117, Budapest, Pázmány Péter sétány 1/A.*
beosztás és iskolai végzettsége: *tanszékvezető*

A diplomamunka címe: Magyar állami topográfiai térképek webes adatbázisa

A diplomamunka témája:
(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben diplomamunka témájának leírását)

A diplomamunka témája egy olyan webes megjelenítő alkalmazás létrehozása, amely a népgazdasági célú felmérés 1:10 000-es méretarányú szelvényeit és az Egységes Országos Térképrendszer 1:10 000-es méretarányú szelvényeit tartalmazza, és a térképek által lefedett területet Leaflet alapú webtérképen megjeleníti. Az alkalmazás Leaflet-et, HTML-t és CSS-t használva fog elkészülni.

Budapest, 2020.11.30.

Tartalomjegyzék

1. Bevezetés 4

2. A polgári térképészet és a népgazdasági célú felmérés 5

2.1. Népgazdasági célokat szolgáló felmérés 5

3. Felhasznált programozási nyelvek 8

3.1. HTML 8

3.2. JavaScript 8

3.3. CSS 8

3.4. Leaflet 9

4. A webkatalógus elkészítése 10

4.1. Alapanyagok rendszerezése és kiválogatása 11

4.2. A raszteres állományok fájlneveinek azonossá tétele a szelvényszámmal 11

4.3. Leaflet-környezet kialakítása és a programkód elkészítése 15

4.4. HTML létrehozása és a Leaflet importálása 16

4.5. Négyzetháló kirajzolása 18

4.6. Szelvényszámok hozzárendelése a négyzetháló elemeihez 20

4.7. Adatbázis-ellenőrzés 24

4.8. Oldalsó menü kialakítása 26

4.9. Az adatbázis bővítése 28

4.10. LayerControl és ScaleControl 28

4.11. A katalógus megjelenése 29

5. Összegzés 31

1. Bevezetés

Az egyetemi tanulmányaim során több kurzus is foglalkozott a webkartográfia, a webes GIS és az interaktív térképes weboldalak témakörével. Megismerkedtünk a Leaflet-tel, az OpenLayers-szel, a MapServer különböző lehetőségeivel és a HTML programnyelvvel. Továbbá ezekkel párhuzamosan bővültek a programozási ismereteink is. Ezek a kurzusok számos új ismerettel szolgáltak, a tanagyag bővülésével párhuzamosan nőtt az érdeklődésem a téma iránt. Szabadidőmben is rengeteget foglalkoztam különféle weboldalak létrehozásával akár Leafletet, akár OpenLayerst, esetleg MapServert használva. Ezek ismeretével, és megfelelő programozási ismeretekkel szinte bármilyen térképes weboldal elkészíthető.

A diplomamunkám témájának megválasztásához egy korábbi kurzus projektje adta az ötletet: az Önálló Geoinformatikai Projektfeladat című tárgy keretein belül volt lehetőségem egy topográfiai térképekből álló webes térképkatalógus elkészítésére. Ekkor gondoltam, hogy a diplomamunkámban is egy hasonló feladattal szeretnék foglalkozni.

A szakdolgozat célja egy webes felhasználóbarát térképkatalógus létrehozása, amely a sztereografikus és Gauss-Krüger vetületben készült népgazdasági célú felmérés 1:10000-es méretarányú topográfiai térképeit tartalmazza, és az egyes szelvények területét OpenStreetMap webtérképen megjeleníti és letöltésre elérhetővé teszi.

2. A polgári térképészet és a népgazdasági célú felmérés

Mielőtt ismertetem a webes térképkatalógus elkészítésének menetét, szükségesnek tartom a magyar polgári térképészet rövid történeti áttekintését, valamint a katalógus alapjául szolgáló népgazdasági célú 1:10000-es méretarányú felmérés ismertetését és a felmérés folyamatának leírását.

A II. világháború után a Honvéd Térképészeti Intézettel párhuzamosan, 1952-ben létrehozták az Állami Földmérési és Térképészeti Hivatalt (1967-től Országos Földügyi és Térképészeti Hivatal), melynek feladata a polgári célú földmérési és térképészeti munkák irányítása volt. 1954-ben létrejöttek a nagyobb földmérő vállalatok, a Budapesti Geodéziai és Térképészeti Vállalat (BGTV) és a Pécsi Geodéziai és Térképészeti Vállalat (PGTV), valamint az állami (mai szóhasználattal polgári) térképkiadásért felelős Kartográfiai Vállalat (KV, ma Cartographia Kft.). A KV és a BGTV 1990-ig irányította a hazai kataszteri és polgári topográfiai térképezési munkákat. A Vállalat kezdeti feladata a mindennapi használatra szánt (mai szóhasználattal tömeg-) térképek, oktatási (földrajzi és történelmi) térképek, atlaszok, falitérképek készítése volt. 1959-ben sokszorosítási eljárásokkal bővült a feladatköre. Az 1950-es évek elején a polgári térképészet megkezdte egy topográfiai térképsorozat elkészítését, először 1:5000-es méretarányban, majd 1957-től 1:10000-es méretarányban. (Klinghammer, 1983)

2.1. Népgazdasági célokat szolgáló felmérés

Az 1950-es évekig a polgári és katonai felhasználás igényét az 1:25000-es méretarányú katonai topográfiai térképek elégítették ki. Azonban egyrészt ezek nem voltak alkalmasak a fontosabb polgári célú tervezési és kutatási munkák megvalósításához, másrészt a katonai térképészet szerette volna korlátozni a térképeik polgári felhasználását. Célszerűvé vált egy új, pontosabb és nagyobb méretarányú topográfiai térképrendszer kiadása.

Az Országos Tervhivatal 1951-ben adott utasítást az Országos Földméréstani Intézetnek az 1:5000 méretarányú felmérés megkezdésére. A felmérést eleinte az Országos Földméréstani Intézet Földmérési Iroda végezte. 1954-től a Budapesti Geodéziai és Térképészeti Vállalat és a Pécsi Geodéziai és Térképészeti Vállalat, valamint a Kartográfiai Vállalat megalakulása után a felmérést a BGTV és a PGTV végezte. 1959-ben a BGTV és a Kartográfiai Vállalat egybeolvadása után az utóbbi feladata lett a felmérés folytatása. A felmérés alapja az 1:4000-es méretarányú kataszteri térkép alumínium lemezeinek kicsinyített változatai voltak, ezért

eleinte a kataszteri térkép vetületét és szelvényezését alkalmazták. A szocialista országok 1954-es geodéziai konferenciája után a polgári térképek kataszteri szelvényezés helyett nemzetközi szelvényezést alkalmaztak, valamint a Balti magassági rendszert használták a nadapi helyett. A térképek elkészítéséhez rendelkezésre álltak a honvédség légifényképei, fotótérképek, síkfotogrammetriai és térfotogrammetriai műszerek is, valamint a tapasztalt személyi állomány is jelentős számmal bővült. Azonban a folyamatosan növekvő térképigény miatt az előbb felsorolt alapanyagokkal, eljárásokkal sem tudják a tervezett idő alatt elkészíteni a felmérést. Ezért az Állami Földmérési és Térképészeti Hivatal 1957-től elrendelte a felmérés 1:10000-es méretarányban történő megvalósítását, mert ezen méretarányú térképek elkészítése és sokszorosítása gyorsabb és gazdaságosabb és ugyanúgy pontosak és részletesek. Az Állami Földmérési és Térképészeti Hivatal továbbá utasításba adta, hogy az elkészült 1:5000 méretarányú térképeket 1:10000-es méretarányban kartografálják, majd sokszorosítsák, valamint Gauss-Krüger vetületben, nemzetközi szelvényezéssel, a balti alapszinthez viszonyítva készítsék el. 1960-tól a MN Térképészeti Intézet is csatlakozott a térképezési munkálatokhoz.

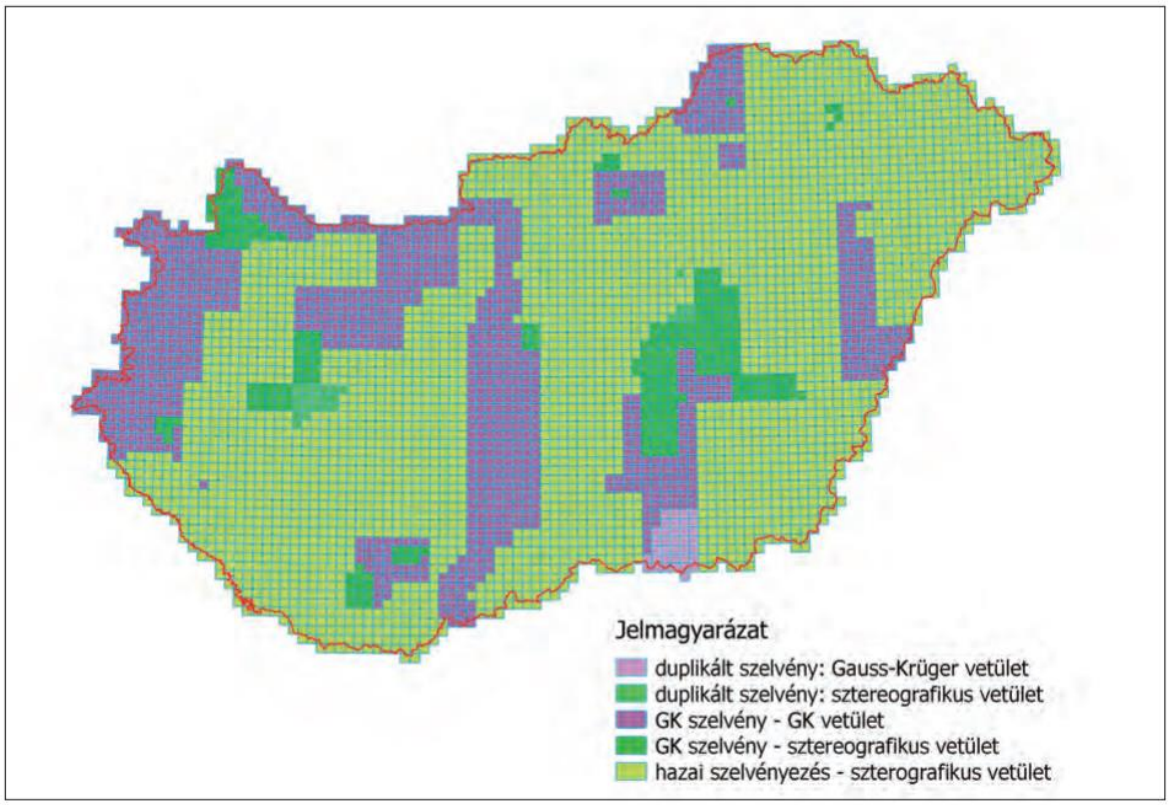
1965-ben titokvédelmi követelményekre hivatkozva megváltoztatták a felmérés alapjait: a Gauss-Krüger vetületi rendszer helyett sztereografikus rendszerben készültek a térképek. A nemzetközi szelvényezés helyett a hazai szelvényezés lépett életbe. Továbbá megkezdték az 1:25000-es és 1:100000-es méretarányú (levezetett) polgári topográfiai térképek szerkesztését is. A felmérés végeztével 4442 darab 1:10000-es méretarányú térképszelvény készült el, melynek felméréseiből

- 3011 darabot a Kartográfiai Vállalat (a BGTV-vel együtt),
- 975 darabot az MN Térképészeti Intézet,
- 456 darabot a Pécsi Geodéziai és Térképészeti Vállalat,
- 17 darabot a Budapesti Műszaki Egyetem

végzett el.

A Földmérési és Távérzékelési Intézet (FÖMI) jogutódja, Budapest Főváros Kormányhivatala Földmérés, Távérzékelési és Földhivatali Főosztálya tárol 4503 darab szkennelt és nyomtatott formátumú térképszelvényt. Ebből 1402 darab Gauss-Krüger, 3101 darab hazai szelvénybeosztású. A Gauss-Krüger szelvényszámmal ellátott térképekből 288 darab sztereografikus, 1114 darab Gauss-Krüger vetületű. A hazai szelvényezésből 61 darab

térképlap Gauss-Krüger szelvényezéssel is elkészült. (Nagy, 1985), (Bene, 1981), (Kratochvilla, 2017)



1. ábra: A Gauss-Krüger és a hazai szelvények eloszlása (Kratochvilla, 2017)

3. Felhasznált programozási nyelvek

3.1. HTML

A HTML (Hypertext Markup Language) weboldalak készítésére használt jelölőnyelv. Meghatározza a jelentését, leírását és a struktúráját a webes tartalomnak. A HTML <elemeket> használ, amit a böngészőprogram értelmezés után az elemek által meghatározott módon megjelenít. (MDN Web Docs), (Stack Overflow)

3.2. JavaScript

A JavaScript (JS) egy objektumorientált, platformfüggetlen programozási nyelv. A nyelv leggyakoribb formája a kliens-szerver oldali JavaScript. A JavaScriptet leginkább weboldalak részeként használják, HTML dokumentumba beágyazva, vagy egy külön fájlra hivatkozva, de nem webböngésző alapú környezetben is alkalmazzák. Elsődleges funkciója a weboldalak interaktívvá, dinamikussá tétele. A JS segítségével képesek vagyunk módosítani a weboldal stílusát és tartalmát, reagálni tudunk a felhasználó interakcióira, le- és feltöltéseket tudunk indítani. Azonban a felhasználó biztonsága érdekében a JavaScript nem képes írni, olvasni, másolni, elindítani fájlokat a felhasználó merevlemezéről. Nem tud hozzáférni a webkamerához, mikrofonhoz, vagy egyéb eszközhöz a felhasználó beleegyezése nélkül. (MDN Web Docs), (javascript.info), (Stack Overflow)

3.3. CSS

A CSS (Cascading Style Sheets) egy programnyelv, amellyel meghatározhatjuk, hogy az általunk leírt HTML dokumentum egyes elemei milyen formátumban legyenek megjelenítve. Segítségével definiálhatjuk az elemek stílusát: betűtípusát, betűméretét, a háttér színét, az elemek elhelyezkedését stb. A CSS nem csak HTML fájlok esetén használható: SVG vagy XML dokumentumok formátumát is meghatározhatjuk vele. A CSS leírását általában külön .css kiterjesztésű fájlban tároljuk, ezzel átláthatóbbá tesszük a weboldal HTML dokumentumát. (MDN Web Docs), (Stack Overflow)

3.4. Leaflet

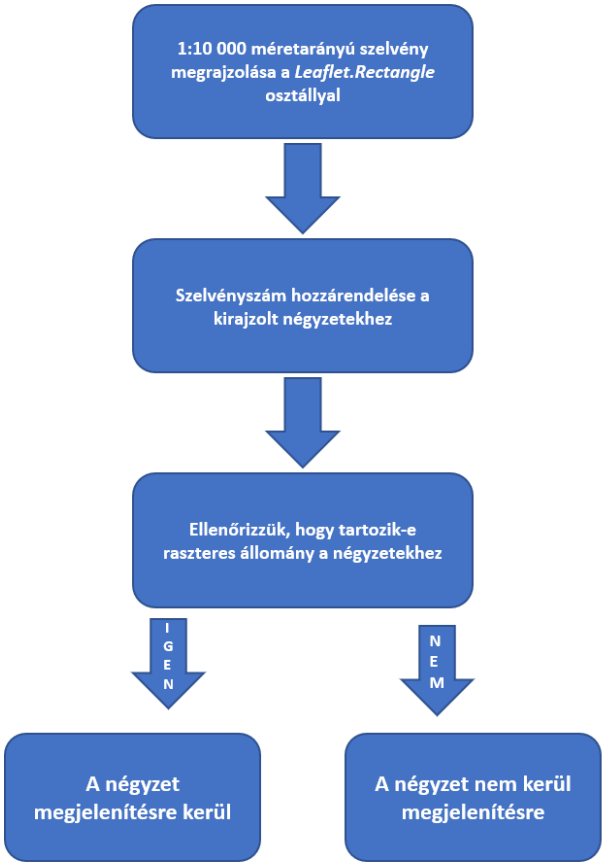
A Leaflet egy nyílt forráskódú JavaScript könyvtár, amellyel interaktív webes térképek készíthetők. A Leaflet kompatibilis minden olyan eszközzel, ami támogatja a HTML5 és CSS3 nyelveket. A Leaflet-tel számos lehetőségünk van a weboldalunk bővítésére, például létrehozhatunk interaktív rétegeket a térképen, adatokat tölthetünk be GeoJSON fájlokból, raszteres adatállományokat jeleníthetünk meg stb. (Leaflet.com)

4. A webkatalógus elkészítése

A webes térképkatalógus célja a népgazdasági célú térképrendszer 1:10000-es méretarányú térképszelvényei által lefedett területeinek a megjelenítése OpenStreetMap alaptérképen, szelvényszámmal ellátva.

A katalógushoz az alapanyagot Zentai László biztosította. Ez tartalmazott 2 darab szkennelt, áttekintő térképet, 4843 darab szkennelt térképszelvényt, és 27 darab Excel állományt, ami a térképekre vonatkozó adatokat tartalmazott.

A program működési elve a következő: két, egymásba ágyazott FOR ciklussal definiáljuk az egyes térképszelvények által lefedett területet reprezentáló négyzeteket (ezek a valóságban változó méretű foktrapézok), majd ezeket olyan azonosítóval látjuk el, ami megegyezik az oda tartozó térkép szelvényazonosítójával. Ezt követően megkeressük azokat a szelvényeket, amelyek megtalálhatók az adatbázisban, ha van, akkor a négyzet kirajzolásra kerül, ha nincs, akkor nem kerül megrajzolásra.



2. ábra: A webkatalógus működési elve (saját ábra)

A működési elv alapján a térképkatalógus elkészítése 3 munkafázisra osztható, ezek:

1. Alapanyagok rendszerezése és kiválogatása
2. A raszteres állományok fájlneveinek azonossá tétele a szelvényszámmal
3. Leaflet-környezet kialakítása és a program elkészítése

Az egyes munkafázisok külön-külön kerülnek részletesebb bemutatásra.

4.1. Alapanyagok rendszerezése és kiválogatása

A folyamat első része a raszteres állományok válogatása és rendszerezése volt. Csak azok a térképek kerültek be a térképkatalógusba, amelyek egy teljes egész térképszelvényt tartalmaznak, valamint Gauss-Krüger szelvényazonosítóval vannak ellátva. Ez alapján 4831 darab fájlt különítettem el.

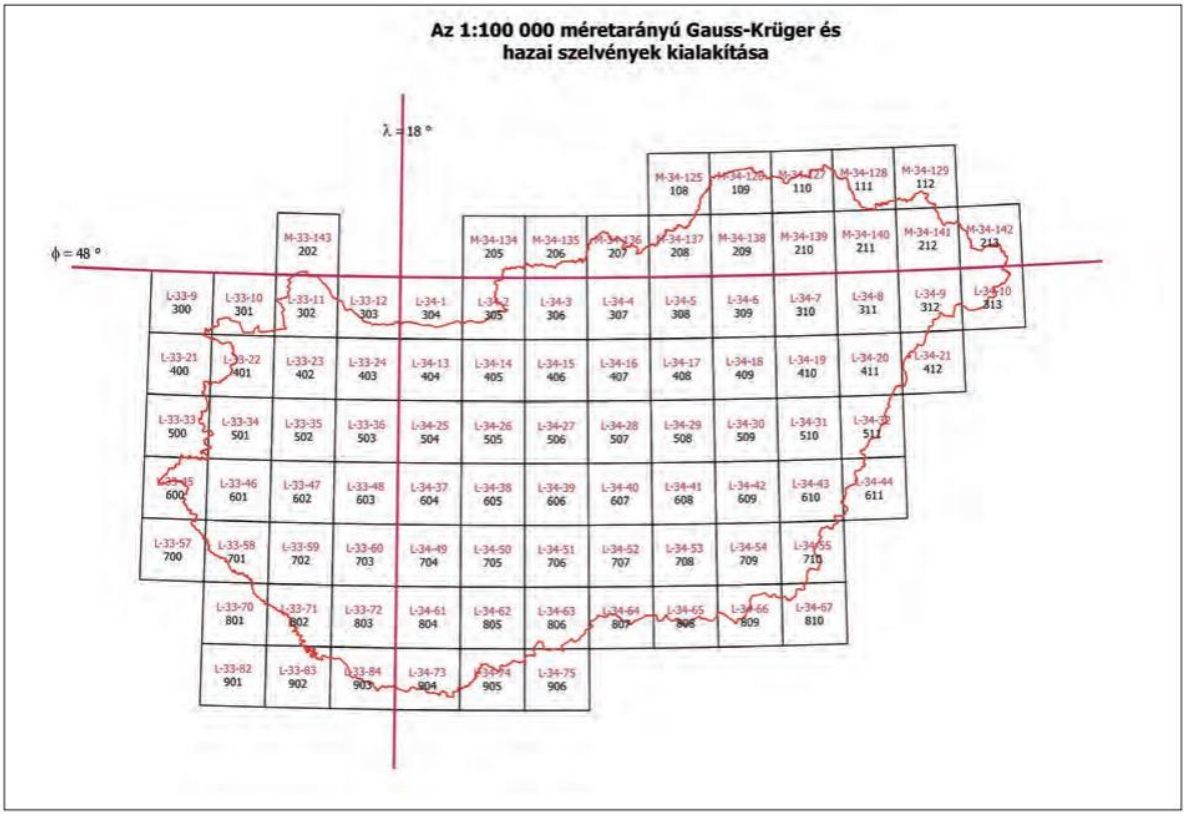
4.2. A raszteres állományok fájlneveinek azonossá tétele a szelvényszámmal

Ebben a munkafázisban a kiválogatott állományok fájlneveit a megfelelő formátumúvá alakítottam.

Gauss-Krüger szelvényezés esetén egy 1:10000-es térképszelvény a következőképp jön létre: egy teljes, 4x6 fokos zónát 12x12-részre osztunk, ezzel létrehozzuk az 1:100000-es méretarányú szelvényeket. A szelvényazonosító itt három tagból áll: a zónát meghatározó betűből és számból, illetve egy harmadik, egy 1 és 144 közötti számból, például L-34-29. Az 1:100000-es szelvényt 4 felé osztva megkapjuk az 1:50000-es térképeket, a szelvényazonosító ebben az esetben megegyezik a százezres méretarány azonosítójával és egy kiegészítő A-B-C-D karakterrel, például L-34-29-A. Ezt tovább osztva, szintén 4 részre, megkapjuk az 1:25000-es méretarányú térképeket: a szelvényszám egy újabb a-b-c-d taggal bővül, például L-34-29-A-b. Végül ha további 4 részre osztunk egy 1:25000-es szelvényt, megkapjuk az 1:10000-es térképeket, ahol a szelvényazonosító az 1-2-3-4 számokkal bővül, például L-34-29-A-b-4. (Nagy, 1985)

A sztereografikus vetületbe való áttéréssel együtt vezették be a térképek hazai szelvényezését (másnéven telefonszámos szelvényezés) is. Ezt úgy érték el, hogy a Gauss-Krüger 1:100000-es méretarányú térképeket megfeleltették az azonos méretarányú hazai szelvényezésben készült térképlapnak. A hazai szelvények további felosztása ugyanúgy történik, mint a Gauss-Krüger szelvényezésénél azzal a különbséggel, hogy a további felosztásokból létrejövő karakterek számok. Egy 1:10000-es szelvény azonosítója tehát 6 darab számból áll (például 609-123), ami

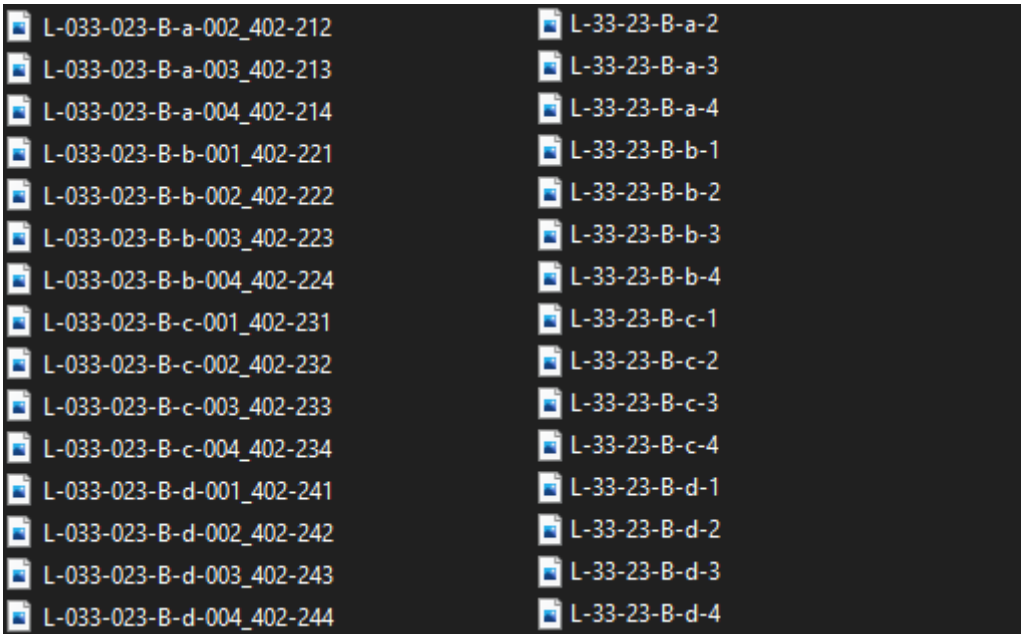
az 1960-as években használatos budapesti telefonszámokhoz volt hasonló. Innen kapta ez a fajta szelvényezés a „telefonszámos” kifejezést. (Kratochvilla, 2017)



3. ábra: A közös szelvényezés (Bene, 1981)

Ebben a munkafázisban a cél tehát, hogy az adatbázisban szereplő térképek fájlneveit a Gauss-Krüger szelvényezés formátumára hozzuk. A fájlnevek a legtöbbször számunkra nem megfelelő formátumban szerepeltek. több mint 4000 fájl esetén egyesével átnevezni minden állományt túl sok időt vett volna igénybe, ezért a névmódosításokat a Bulk Rename Utility programmal végeztem el. Ez a program lehetőséget adott arra, hogy előre meghatározott műveletekkel és a hozzájuk tartozó paraméterek megadásával akár a teljes adatbázisban szereplő fájlok neveit módosítsam.

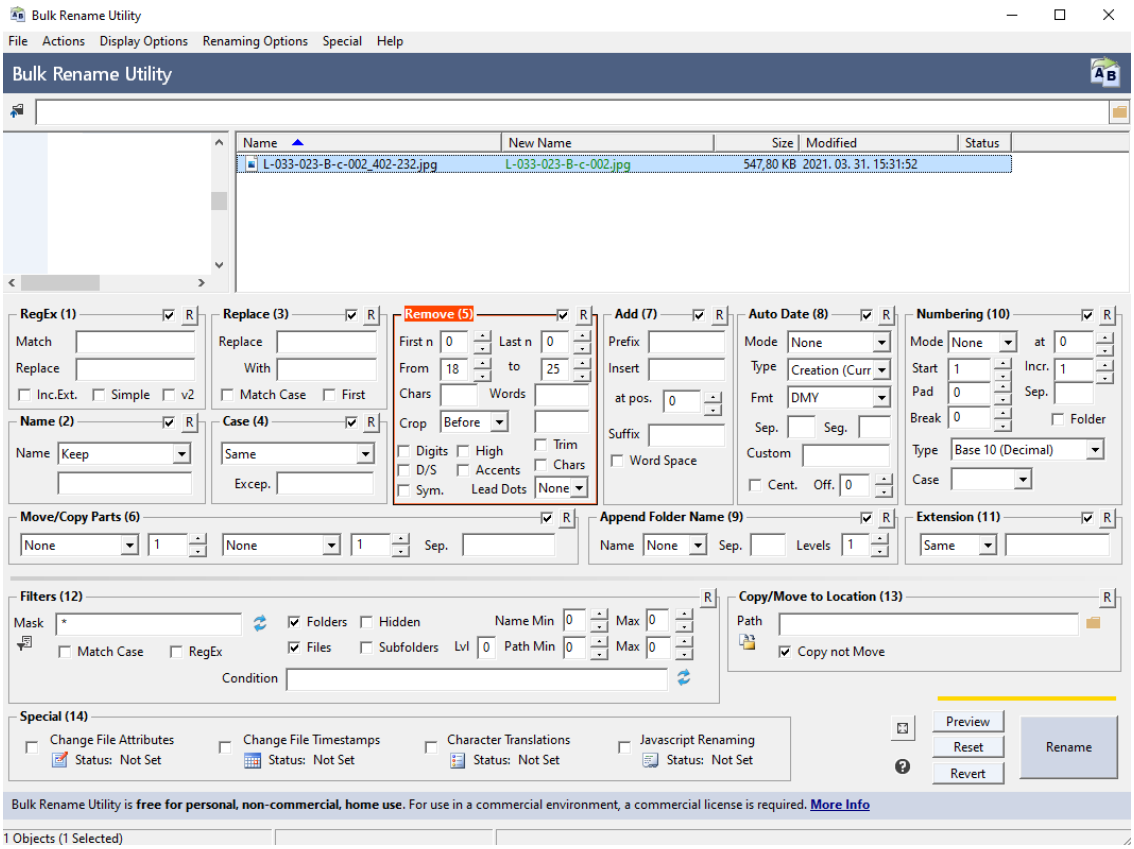
A fájlnevek átnevezésénél kétféle probléma lépett fel: az egyik az, hogy a szelvényszám felesleges „0” karaktereket tartalmaz, valamint „_” karakterrel elválasztva, a fájlnév tartalmazza a sztereografikus rendszeri szelvényszámát, ami felesleges információ.



4. ábra: A formázatlan és a formázott szelvényszámok (Saját ábra)

A két problémát jól szemlélteti a 4. ábra, ahol egymás mellett látható az eredeti és a további feldolgozáshoz szükséges szelvényszám. Az átnevezés folyamatát a listában található „L-033-023-B-c-002_402-232” szelvényszám formázásával mutatom be.

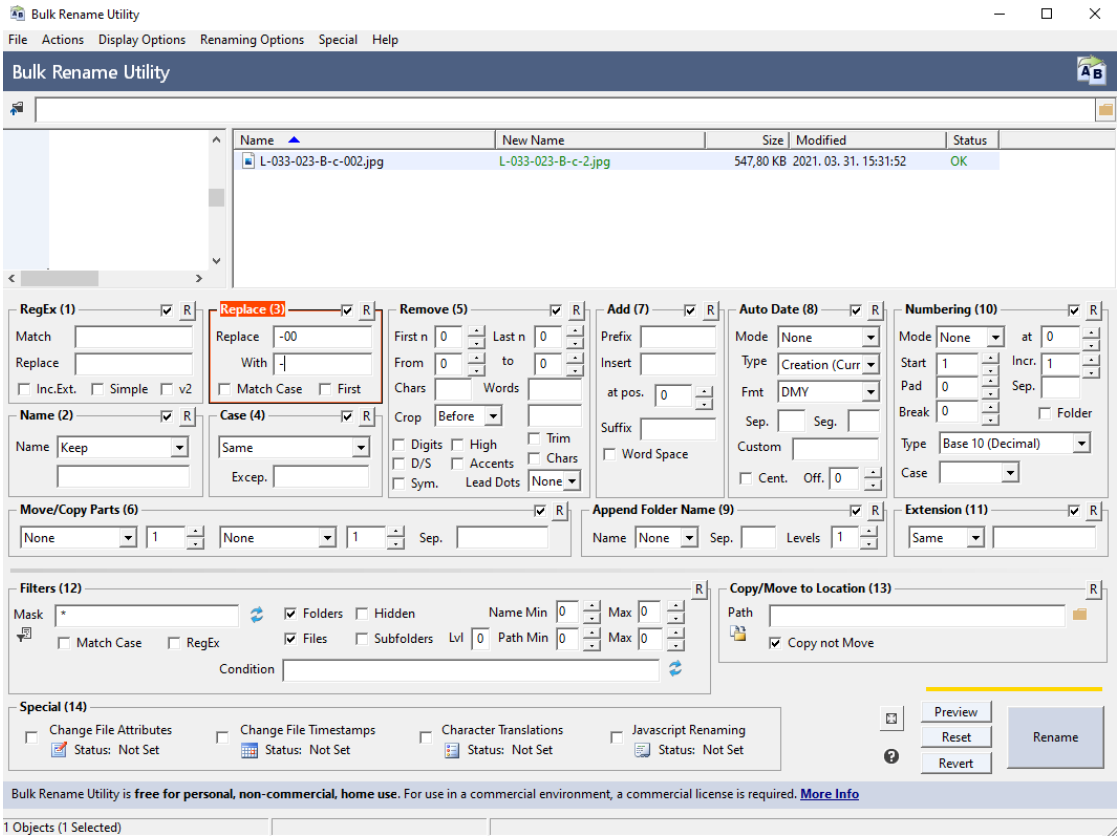
A szelvényszám formázását a „_” karakter és az utána következő sztereografikus szelvényszámok eltávolításával kell kezdeni. Ezt a Remove paranccsal végezzük, amellyel megadhatjuk, hogy a fájlnév hányadik karakterétől számítva töröljük a felesleges karaktereket. Mivel az összes Gaus Krüger szelvényszám azonos számú karakterből áll, így egy lépésben elvégezhető a sztereografikus szelvényszámok törlése. Ha először a „00” karaktereket távolítanánk el, akkor attól függően, hogy a Gauss Krüger szelvényszám második és harmadik blokkja hány karakterből áll, úgy változna fájlonként a „_” karakter sorszáma is. Emiatt nem tudnánk egyszerre az összes sztereografikus szelvényszámot alkotó karaktert eltávolítani. Tehát először távolítsuk el a Remove paranccsal a „_” után következő összes karaktert az 5. ábrán látható beállításokkal:



5. ábra: A sztereografikus azonosítót eltávolító beállítások (saját ábra)

Ekkor a szelvényszámunk „L-033-023-B-c-002_402-232”-ről „L-033-023-B-c-002”-re változik.

A következő lépésben cseréljük ki a „-00” karaktereket kötőjelekre a következő beállításokkal:



6. ábra: A "-00" karaktereket eltávolító beállítások (saját ábra)

Ekkor a szelvényszámunk „L-033-023-B-c-002” -ről „L-033-023-B-c-2” -re változik.

Végül hajtsuk végre ugyanezt a parancsot, de a „-00” helyett a „-0” karaktert cseréljük kötőjelekre. Ekkor a szelvényszám „L-033-023-B-c-2” -ről „L-33-23-B-c-2” -re változik. Ez a szelvényszám-formátum további felhasználásra már megfelelő. Az egész adatbázisra lefuttatva az előbb ismertetett lépéseket a megfelelő sorrendben, át tudjuk nevezni mind a 4831 fájlt egyszerre.

4.3. Leaflet-környezet kialakítása és a programkód elkészítése

A harmadik munkafázisban az alkalmazás „kódját” (programlistáját) hoztam létre. A Magyarországot lefedő négy darab zóna térképeit tartalmazó webkatalógusokat külön HTML fájlban hoztam létre, ezzel kicsit meggyorsítva a weboldalak töltési idejét. Továbbá nem a teljes zónát hozzuk létre, hanem csak azt a részét, ami Magyarországot fedi, szintén azzal a céllal,

hogy csökkentsük a töltési időt. A program működését az L-34-es zónához tartozó HTML fájl elkészítésével mutatom be.

4.4. HTML létrehozása és a Leaflet importálása

Első lépésben létrehoztam a HTML fájlt, majd hozzáadtam a Leaflet működéséhez szükséges script és css fájlok hivatkozásait. A hivatkozások megadásával nem kell letölteni a tárhelyünkre a fájlokat, viszont, ha a Leaflet szerverei átmenetileg nem működnek, a térképünk sem működik. (Leaflet.com)

```
...
      <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
        integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4PsZChSR7A=="
        crossorigin=""/>
      <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
        integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA=="
        crossorigin=""></script>
...
```

Létrehoztam egy elemet map névvel, ami a térképünk tárolására szolgál:

```
...
<div id="map"></div>
...
```

A térkép réteget a map változóban tárolom. Ennek a setView paramétere a térkép nézetét határozzák meg szélességi és hosszúsági koordinátákkal, a vessző utáni szám pedig a nagyítás mértékét határozza meg.

```
...
let map = L.map('map').setView([47, 19], 7);
...
```

A Leaflet úgynevezett zoom-szinteket határoz meg a térkép különböző részletességgel történő megjelenítésére: a kezdő, nulladik szint az egész világot egy 256x256 pixel méretű képpel ábrázolja. Az első zoom-szinten a térképünk mérete 512x512 pixelméretű lesz, így 4 darab 256x256-os raszterrel (a Leaflet ezeket a képállományokat csempének hívja) lehet ábrázolni azt. Minden további zoom-szinten az egyes rasztereket 4 részre osztjuk, és az egyes részek

mérete a duplájára nő. Tehát a térkép felbontása az egyes zoom-szintekben $256 \cdot 2^{\text{zoom-szint}}$ lesz. Ez a folyamat addig ismétlődik, amíg el nem érjük a legnagyobb nagyítási lehetőséget. Ez alaptérképenként változó, általában a 18. szintig tudunk nagyítani. (Leaflet.com)

7. ábra: A Leaflet zoom funkciójának működése (Leaflet.com)

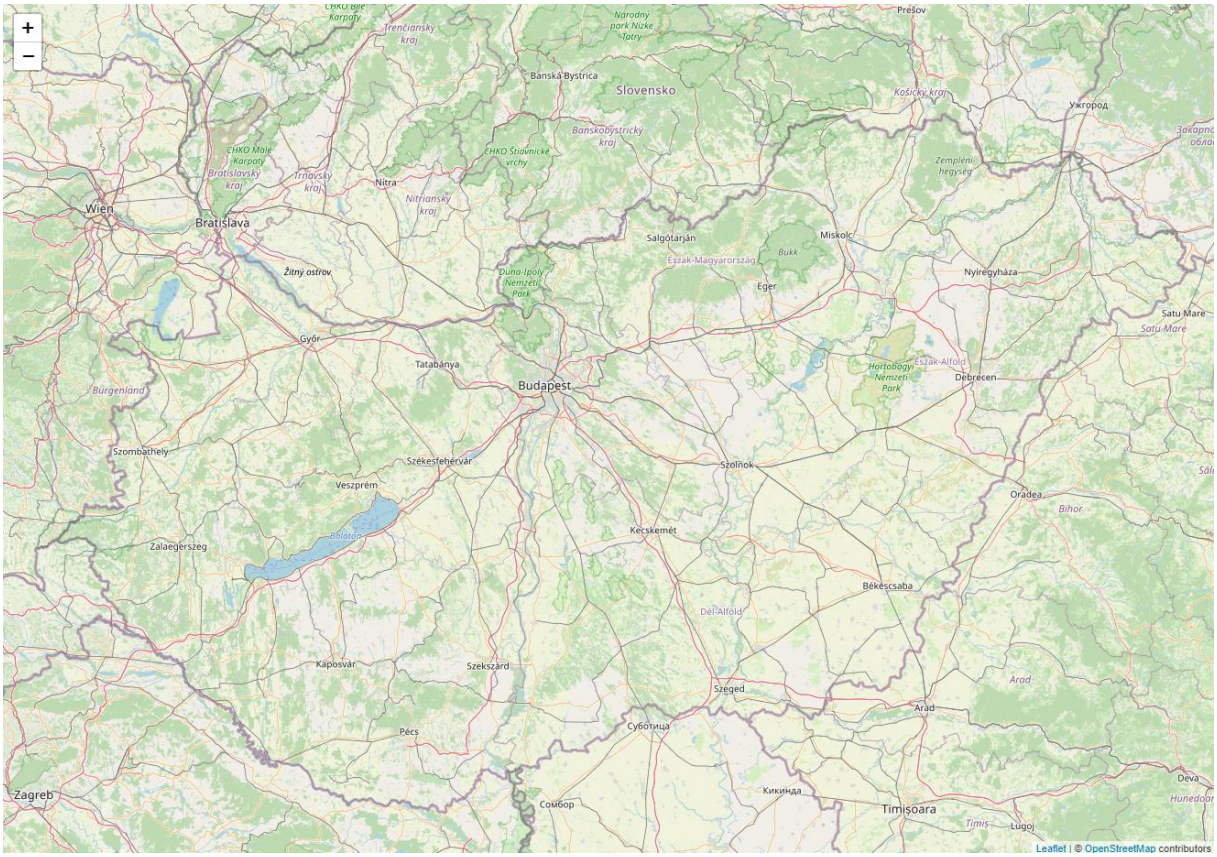
Végül létrehozom az OpenStreetMap változót, amelyben az OSM alaptérképet tárolom, ezt hozzáadom a map elemhez.

```
...
let OpenStreetMap = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y
}.png', {attribution: '&copy; <a href="https://www.openstreetmap.org/copyright
">OpenStreetMap</a> contributors'
        }).addTo(map);
...
```

A <style> elemen belül létrehoztam a #map CSS stílust, amivel a map elem határait adom meg. Ezekkel a beállításokkal a térképünk teljes képernyős módban lesz látható.

```
...
#map {position: absolute; top: 0; bottom: 0; left: 0; right: 0;

        }
...
```

Ha ebben az állapotban lefuttatjuk az elkészült HTML fájlt, a következőt láthatjuk:



8. ábra: Az üres Leaflet térkép (saját ábra)

4.5. Négyzetháló kirajzolása

A térképeket helyettesítő négyzetek a Leaflet Rectangle osztályával kerültek megrajzolásra. Az L.Rectangle osztálynak két ellentétes sarokpont-koordinátára van szüksége. A négyzetet meghatározó jobb felső és bal alsó sarokpont koordinátáit két egymásba ágyazott FOR ciklussal határozzuk meg. A négyzeteket balról kezdve, oldalirányban, soronként rajzoljuk.

Létrehozzuk a startLat és startLon változókat, amelyek a FOR ciklusok kezdő értékeit adják. Az L-34-es zóna Magyarországot lefedő részének kezdő szélességi *(startLat)* és hosszúsági *(startLon)* értékei. Ezek az L-34-es zóna esetében a következők:

```
startLat = 48;
startLon = 18;
```

Létrehozzuk a FOR ciklusokat:

```
for(let lat = startLat; lat >= 45.625; lat = lat - 0.0416666667) {
	for(let lon = startLon; lon < 23; lon = lon + 0.0625) {
	}
}
```

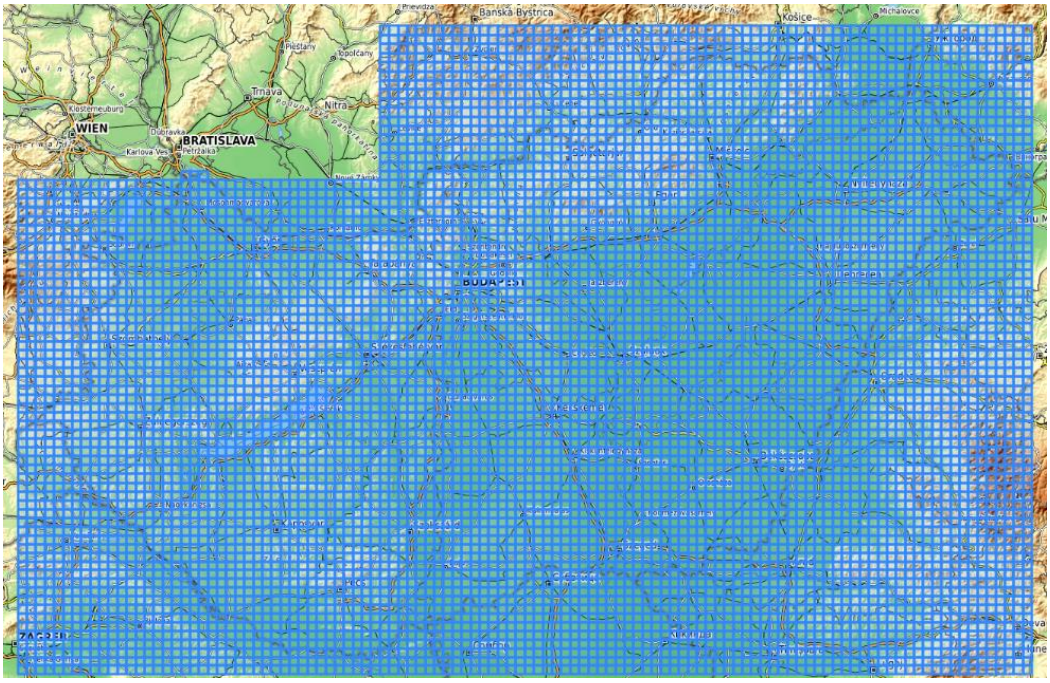
A külső FOR ciklus az egyes kockák megrajzolásához szükséges sarokpontok szélességi értékeit határozza meg: az első paraméter, tehát a ciklusváltozó *(lat2)* kezdő értéke az L-34-es zóna Magyarországot lefedő részének szélességi értéke, 48 fok. A futási feltétellel meghatározzuk, hogy a ciklus milyen szabály teljesüléséig fut: ez az érték az L-34-es zóna Magyarországot lefedő részének a másik határa, 45 fok. A ciklus tehát addig fut, amíg el nem érjük az L-34-es zóna határát, 45 fokot. A harmadik paraméter egy művelet, ami minden egyes ciklusban lefut, amíg el nem érjük a zóna végét. A ciklusváltozóból azt az értéket vonjuk le, amekkora egy 1:10000-es térképszelvény által lefedett szélességi tartomány, tehát 2 perc 30 másodpercet, azaz 0.0416666667 értéket.

A belső FOR ciklus a sarokpontok hosszúsági koordinátáit számolja: az első paraméter itt is a kezdő hosszúsági érték, az L-34-es zóna esetében 18 fok. Ez a ciklus is addig fut, amíg el nem érjük a zóna végpontjának hosszúsági értékét, tehát a 23 fokot. A ciklusváltozót itt 3 perc 45 másodperccel növeljük, azaz hozzáadunk 0.0625-öt, mert egy szelvény ekkora hosszúsági tartományt fed le.

Tehát a koordinátapárok meghatározásakor tulajdonképpen a kezdő szélességi és hosszúsági koordinátákkal meghatározott kezdőpontot toljuk el horizontálisan 2 perc 30 másodperccel, és vertikálisan 3 perc 45 másodperccel, majd az így létrejött koordinátapár értékeit a *bounds* változóban tároljuk el a következőképpen:

```
...
let bounds = [[lat, lon], [lat + 0.0416666667, lon + 0.0625]];
...
```

Ha a fent leírtak alapján létrehozzuk mind a négy, Magyarországot lefedő zóna FOR ciklusait, és feltétel nélkül minden négyzetet kirajzolnánk, a következő eredményt kapjuk:



9. ábra: A teljes, Magyarországot lefedő négyzetháló (saját ábra)

4.6. Szelvényszámok hozzárendelése a négyzetháló elemeihez

A szelvényszám generálását a négyzetháló kirajzolásával párhuzamosan végzi a program, tehát az erre vonatkozó programkód a FOR ciklusokon belül van. Fontos megjegyezni, hogy a programkód globálisan nem alkalmazható Gauss-Krüger szelvényszámok generálásra. Kizárólag a Magyarországot metsző 4 zónára, azon belül is csak az országot ténylegesen lefedő részeire alkalmazható.

A szelvényazonosítók létrehozásához öt változót előre definiálunk az L-34-es zóna esetében következő értékekkel:

- current100K = 1;
- numberOf100KCols = 10;
- rowStepAt100K = 3
- rowCounterfor100K = 8
- columnCounterFor100K = 8

Az egyes négyzetekhez tartozó azonosítót a *label* változóban tároljuk:

```
...
let label = `${startingSlice}-${current100K}-
${labelMatrixFor50K25K10K[matrix_i][matrix_j]}`
...
```

A *label* változó 3 tagból áll: a *startingSlice* tagból, ami a zónát meghatározó betűt és számot tárolja, a *current100K* azonosítóból, ami az 1-144-ig terjedő, 1:100000-es térképek számát tárolja, és a *labelMatrixFor50K25K10K* változóból, ami pedig az 1:50000-es, az 1:25000-es és az 1:10000-es méretarányhoz tartozó három darab (például: A-a-1) szelvényazonosítókat tartalmazza.

A current100K változó tehát az első 1:100000-es zóna azonosítóját tartalmazza. A változó kezdeti értéke az L-34-es zóna estében 1, mivel ez az első zóna, amiben négyzetet megrajzolunk. Az L-33-as zóna esetében ennek a változónak az értéke 9 lesz, mivel ott a 9-es zónában kezdjük a kirajzolást és a szelvényszám-hozzárendelést.

A current100K változását a columnCounterFor100K és a rowCounterFor100K változó értékeitől tesszük függővé. A rowCounterFor100K és columnCounterFor100K változókat a kirajzolás haladtával folyamatosan változtatjuk, egyfajta visszaszámlálóként fogjuk őket használni. Mindkét változó kezdeti értéke 8, mivel egy 1:100000-es szelvény 8x8 darab 1:10000-es kisebb négyzetből áll.

```
...
if (columnCounterFor100K == 1) {
                    current100K++;
                    columnCounterFor100K = 8;
                } else {
                    columnCounterFor100K--;
                }
...
```

A fenti kódrészlet a külső FOR ciklus törzsében található. Ha a columnCounterFor100K változó értéke 1 lesz, tehát oldalirányban kirajzoltunk 8 négyzetet, és ezzel zónát lépünk, akkor a current100K értékét 1-gyel növeljük és a columnCounterFor100K visszaszámláló értékét visszaállítjuk 8-ra, ezzel elölről kezdjük a visszaszámlálást. Ameddig a columnCounterFor100K értéke nem 1, addig minden kirajzolt négyzet után csökkentsük az értékét 1-gyel.

A current100K értékét akkor is változtatni kell, hogyha horizontálisan lépünk 8-at, mert ekkor szintén új zónába kerülünk. A horizontális lépésszámláló a rowCounterFor100K változó.

```
...
if (rowCounterFor100K == 1) {
                current100K = (current100K - 1) + rowStepAt100K;
                rowCounterFor100K = 8;
            } else {
                current100K = current100K - numberOf100KCols;
                rowCounterFor100K--;
            }
...
```

Ha a rowCounterFor100K értéke 1, tehát új szelvényhez értünk, akkor a current100k értékéhez hozzá kell adni a rowStepAt100K értékét (ami az L-34-es zóna esetében 3), mivel az országot lefedő zóna a 10-es számú 1:100000-es szelvényig tart, és a következő szelvény a 13-mas lesz. Továbbá a current100K értékéből 1-et le kell vonnunk, mivel a belső ciklusban azt már 1-gyel tovább növeltük. Ha ezt nem tennénk meg, akkor a következő kezdő szelvényszám 13 helyett hibásan 14 lenne.

Ha nem lépünk 1:100000-es szelvényt horizontálisan, akkor vissza kell ugrani a kezdő 1:100000-es szelvényünkre úgy, hogy kivonjuk a numberOf100KCols változó értékét a current100K aktuális értékéből, és csökkentjük a rowCounterFor100K visszaszámláló értékét 1-gyel. A numberOf100KCols változó a zónában található 1:100000-es oszlopok számát adja meg. L-34-es zóna esetén ez 10 darab.

A két elágazással és a hozzájuk tartozó változókkal könnyen meg tudjuk határozni az egyes zónákra az 1:100000-es szelvényazonosítókat. Mind a négy zóna 1:100000-es azonosítóit ugyanezen logika mentén hozom létre, de a current100K, a numberOf100KCols és a rowStepAt100K azonosítók kezdeti értékeit zónákra szabottan változtatom attól függően, hogy az adott zóna mekkora, és melyik része fedi az országot.

A labelMatrixFor50K25K10K változó egyszerre tárolja az 1:50000-es, az 1:25000-es és az 1:10000-es azonosítókat. A lehetséges értékeket egy mátrixban tárolom, mivel ezek egy 1:100000-es szelvényen belül fixen helyezkednek el, így felesleges lett volna őket dinamikusan generálni.

```
const labelMatrixFor50K25K10K = [
   ['A-a-1', 'A-a-2', 'A-b-1', 'A-b-2', 'B-a-1', 'B-a-2', 'B-b-1', 'B-b-2'],
   ['A-a-3', 'A-a-4', 'A-b-3', 'A-b-4', 'B-a-3', 'B-a-4', 'B-b-3', 'B-b-4'],
   ['A-c-1', 'A-c-2', 'A-d-1', 'A-d-2', 'B-c-1', 'B-c-2', 'B-d-1', 'B-d-2'],
   ['A-c-3', 'A-c-4', 'A-d-3', 'A-d-4', 'B-c-3', 'B-c-4', 'B-d-3', 'B-d-4'],
   ['C-a-1', 'C-a-2', 'C-b-1', 'C-b-2', 'D-a-1', 'D-a-2', 'D-b-1', 'D-b-2'],
   ['C-a-3', 'C-a-4', 'C-b-3', 'C-b-4', 'D-a-3', 'D-a-4', 'D-b-3', 'D-b-4'],
   ['C-c-1', 'C-c-2', 'C-d-1', 'C-d-2', 'D-c-1', 'D-c-2', 'D-d-1', 'D-d-2'],
   ['C-c-3', 'C-c-4', 'C-d-3', 'C-d-4', 'D-c-3', 'D-c-4', 'D-d-3', 'D-d-4'],
];
```

A mátrixból két elágazással adjuk meg a megfelelő helyen levő értékeket. A belső FOR ciklusban értelemszerűen a mátrix megfelelő oszlopának az indexét határozzuk meg:

```
...
if (matrix_j == 7) {
    matrix_j = 0;
} else {
    matrix_j++;
}
...
```

A fentiekhez hasonlóan itt is a kirajzolással párhuzamosan léptetjük a matrix_j változót. Ha a változó értéke 7-et vesz fel, tehát az oszlop végére értünk, akkor elölről kezdjük az indexelést, minden más esetben tovább léptetjük a változó értékét.

A sorok indexelését ugyanezen logika mentén végezzük:

```
...
if (matrix_j == 7) {
    matrix_j = 0;
} else {
    matrix_j++;
}
...
```

Ha a változó értéke 7, azaz a sor végére értünk, akkor nullára állítjuk a változó értékét, ezzel itt is elölről kezdjük az indexelést. Minden más esetben itt is tovább léptetjük a változó értékét.

A matrix_i és matrix_j változók együttes értékei meghatározzuk, hogy az adott négyzetünkhöz hányadik indexen levő tagot kell hozzárendelni. A négyzetenként meghatározott startingSlice, current100K és labelMatrixFor50K25K10K változók értékeinek az összefűzésével a label változóban tároljuk a négyzetek szelvényszámait.

```
...
let label = `${startingSlice}-${current100K}-
${labelMatrixFor50K25K10K[matrix_i][matrix_j]}`
...
```

4.7. Adatbázis-ellenőrzés

Mielőtt kirajzolnánk a térképeket reprezentáló, szelvényszámmal ellátott négyzeteket, ellenőriznünk kell, hogy az adatbázisban megtalálható-e a hozzájuk tartozó raszteres állomány. Ezt egy egyszerű feltételes elágazással tudjuk megvizsgálni.

Minden egyes HTML fájlban található egy labellist változó: ez tárolja az adatbázisunkban az adott zónához tartozó raszteres állományok helyes formátumú fájlneveit. Az egyenként elvégzett kirajzolás előtt, az adott négyzethez tartozó label változó értékét, tehát a szelvényszámot, ellenőrizni kell, hogy megtalálható-e a labellist változó értékei közt. Ha igen, azaz van ilyen nevű fájlunk az adatbázisban, akkor kirajzoljuk a négyzetet, de ha nincs ilyen érték a labellist változóban tárolva, akkor nem rajzoljuk ki. A programkód az egymásba ágyazott FOR ciklusokon belül a következőképp néz ki:

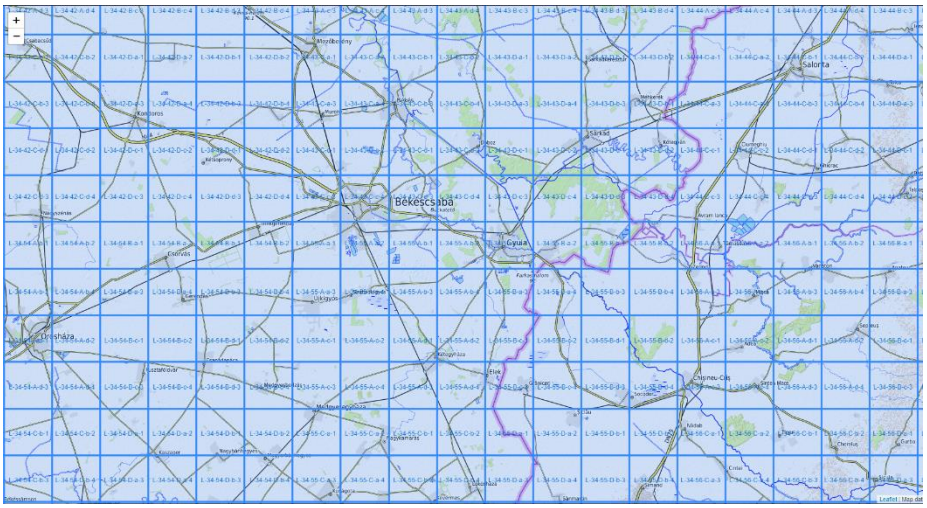
```
...
if (labellist.includes(label)) {
                    let rect = L.rectangle(bounds);
                    rect.bindTooltip(label, {permanent: true, direction:"cente
r", className: 'labelstyle'}).openTooltip()
                    rect.addTo(map)

                    }
...
```

Tehát, ha a labellistünk tartalmazza az label változó aktuális értékét, akkor a következőket tesszük: létrehozzuk a rect változót, amely az aktuális négyzetet jelöli. Ehhez a bindTooltip és az openTooltip paranccsal hozzáadjuk a szelvényszámot feliratként. Ezután a négyzetet hozzáadjuk a térképünkhöz az addTo paranccsal.

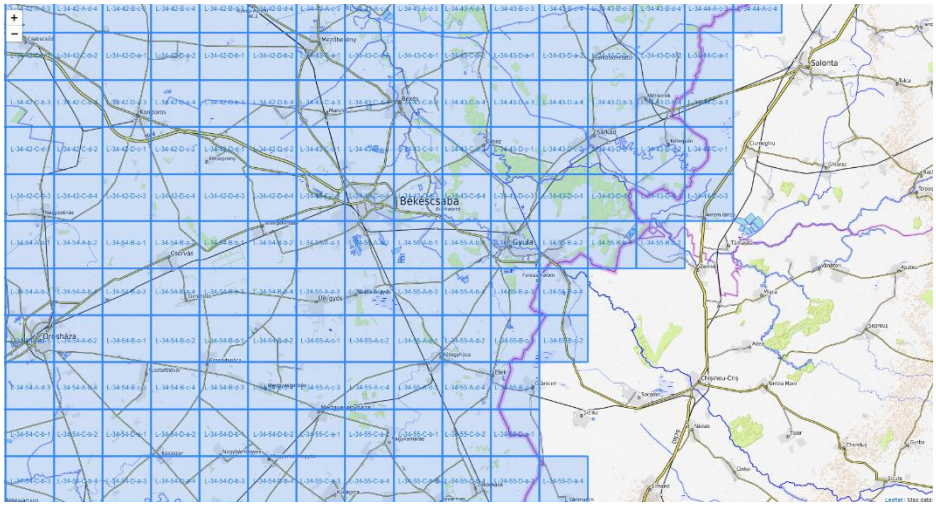
Ahhoz, hogy a négyzetre kattintva el tudjuk érni az adatbázisból az adott raszteres állományt, létre kell hoznunk egy eseményt, amely a négyzetekre történő kattintást kezeli, majd a kattintási esemény lefutása után új ablakban megjeleníti a kívánt fájlt.

```
...
rect.addTo(map).on("click", function (e) {
                    let clickedRectangle = e.target;
                    window.open('../raster_database/L34/'+label+'.jpg');
...
```

Egy négyzetre történő kattintáskor a program a window.open paranccsal megnyitja a raster_database/L34 mappában található fájlt, úgy, hogy az általunk kattintott négyzet label értékét (szelvényszámát) összefűzi a .jpg fájlkiterjesztéssel, ezzel megadva a helyes elérési útvonalat. Így a böngésző új ablakban jeleníti meg a térképet.



10. ábra: A katalógus adatbázis-ellenőrzés nélkül (saját ábra)



11. ábra: A katalógus adatbázis-ellenőrzéssel (saját ábra)

4.8. Oldalsó menü kialakítása

Miután elkészült mind a négy zóna HTML állománya, szükséges volt a könnyebb kezelhetőség érdekében egy felhasználóbarát kezelőfelület kialakítása. Létrehoztam egy oldalsó menüt, amelyből egy kattintással el tudjuk érni a 4 zóna HTML fájljait.

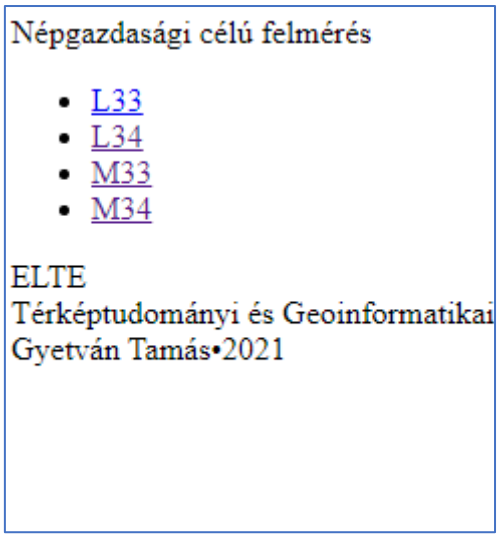
Első lépésként módosítottam a map elemhez tartozó stílusbeállításokat:

```
#map {position: absolute; top: 0; bottom: 0; left: 250px; right: 0;
}
```

A map elem bal oldali határa 250 pixelre került a képernyő bal szélétől, ezzel létrehoztam a helyet az oldalsó menü kialakításához.

```
...
<nav class = "sidebar">
            <div class = "text">Népgazdasági célú felmérés</div>
            <ul>
                <li><a href = "html/L33.html">L33</a></li>
                <li><a href = "html/L34.html">L34</a></li>
                <li><a href = "html/M33.html">M33</a></li>
                <li><a href = "html/M34.html">M34</a></li>
            </ul>

            <div class="kolofon">ELTE<br>Térképtudományi és Geoinformatikai Ta
nszék<br>Gyetván Tamás•2021</div>
        </nav>
...
```

Az oldalsó menühöz létrehoztam egy <nav> elemet. Ezek a típusú elemek általában olyan linkeket tartalmaznak, amelyek a weboldalon belüli, vagy más weboldalakra történő navigációt végzik. Az elemen belül létrehoztam egy <ul> elemet, amely egy sorrend nélküli listát jelöl. Ebben a listában pedig 4 <li> elemet helyeztem el, amelyek a webkatalógus egyes HTML állományainak hivatkozásai. Az oldalsó menü használatakor ezeket a linkeket jelenítjünk meg „gombokként”, amikre kattintva elérjük az egyes weboldakat. Továbbá létrehoztam egy elemet, amely a kolofon tárolására szolgál. (MSDN web docs)



12. ábra: A formázatlan oldalsó menü (saját ábra)

A listát felépítő elemekre, illetve a teljes listára vonatkozó stílusbeállításokat a css mappán belül található style.css lépcsőzetes stílusalap-dokumentumban tárolom. A .css állomány elérési útvonalát az egyes HTML fájlok elején, a Leaflet működéséhez szükséges hivatkozásokkal együtt adom meg.

```
...
<link rel="stylesheet" href="css/style.css">
...
```

Minden egyes elemet külön-külön formázunk meg. Például, az oldalsó menü címére vonatkozó formázási beállítások a következőképpen néznek ki:

```
.sidebar .text{
            color: white;
            font-size: 20px;
            font-weight: 600;
            line-height: 30px;
            text-align: center;
            background: #1e1e1e;
            letter-spacing: 1px;
        }
```

A color paranccsal a betű színét, a font-size paranccsal a betű méretét, a font-weight beállítással félkövér típusú betűt hoztunk létre. A line-height beállítással a cím és a menüt alkotó többi elem közti helyet határoztam meg. A text-align parancs a szöveg középre igazítását végzi, a

background paranccsal a háttér színét adom meg, a letter-spacing paranccsal pedig a betűk közti távolságot adom meg.

4.9. Az adatbázis bővítése

A webkatalógus nagyon egyszerűen bővíthető további raszterekkel: a fájl nevét egyenlővé kell tenni az ábrázolt térkép szelvényszámával, természetesen a megfelelő formátumban. Ez után a fájlunkat a raster_database mappán belül a megfelelő zóna mappájába kell másolni. Ha a fájlnév megadásakor nem vétettünk hibát, a térképet reprezentáló négyzetnek meg kell jelennie a webes katalógusban.

4.10. LayerControl és ScaleControl

A térképkatalógust két funkcióval bővítettem. Az egyik a Leaflet LayerSwitcher nevű pluginja, amely lehetővé teszi több térképi alapréteg használatát a katalógusban, és azokat egy kattintással változtatni tudjuk. Az OpenStreetMap réteg mellé bekerült egy OpenTopoMap réteg is a programba. A másik a ScaleControl, ami egy dinamikusan változó méretarány skála.

```
...
let OpenTopoMap = L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.pn
g', {
            attribution: 'Map data: &copy; <a href="https://www.openstreetmap.
org/copyright">OpenStreetMap</a> contributors, <a href="http://viewfinderpanor
amas.org">SRTM</a> | Map style: &copy; <a href="https://opentopomap.org">OpenT
opoMap</a> (<a href="https://creativecommons.org/licenses/by-sa/3.0/">CC-BY-
SA</a>)'
...
        }).addTo(map);
```

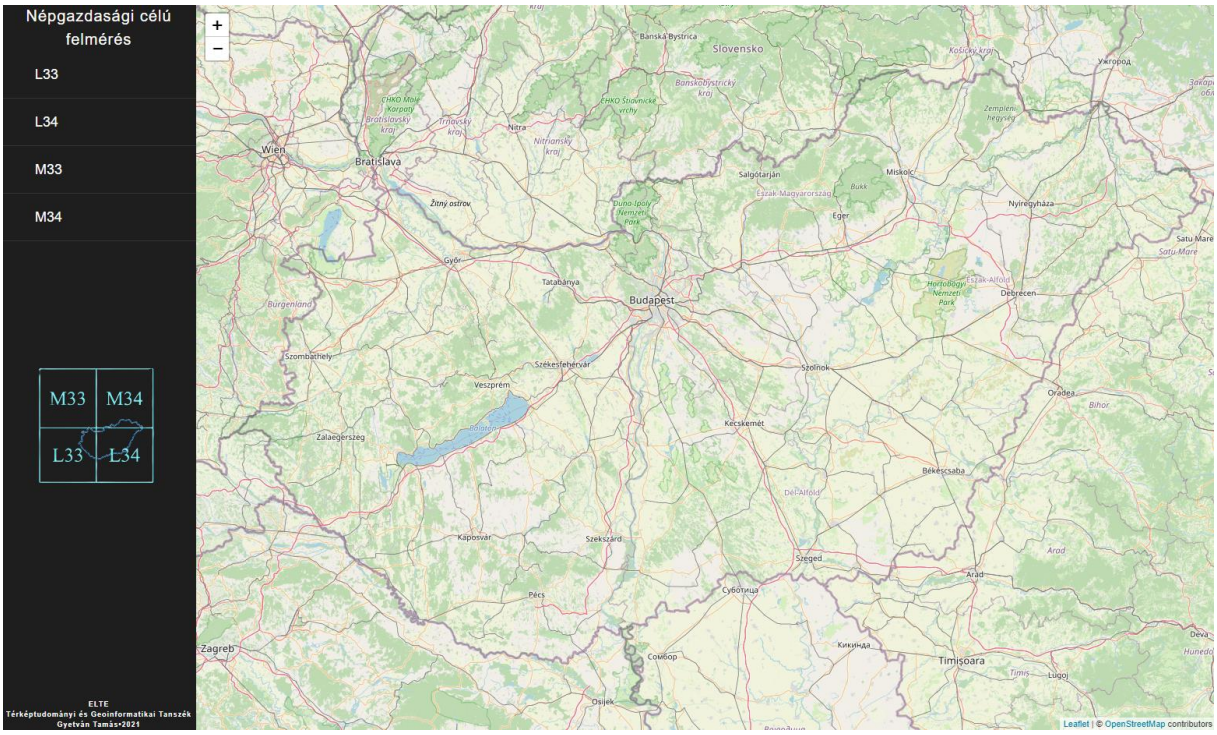
A LayerControl és a ScaleControl kódja a következő:

```
...
let baseMaps = {
            OpenStreetMap,
            OpenTopoMap
        };
        let overlayMaps = {};
        L.control.layers(baseMaps, overlayMaps).addTo(map);
        L.control.scale().addTo(map);
...
```

13. ábra: A LayerSwitcher és a dinamikusan változó méretarány-skála (saját ábra)

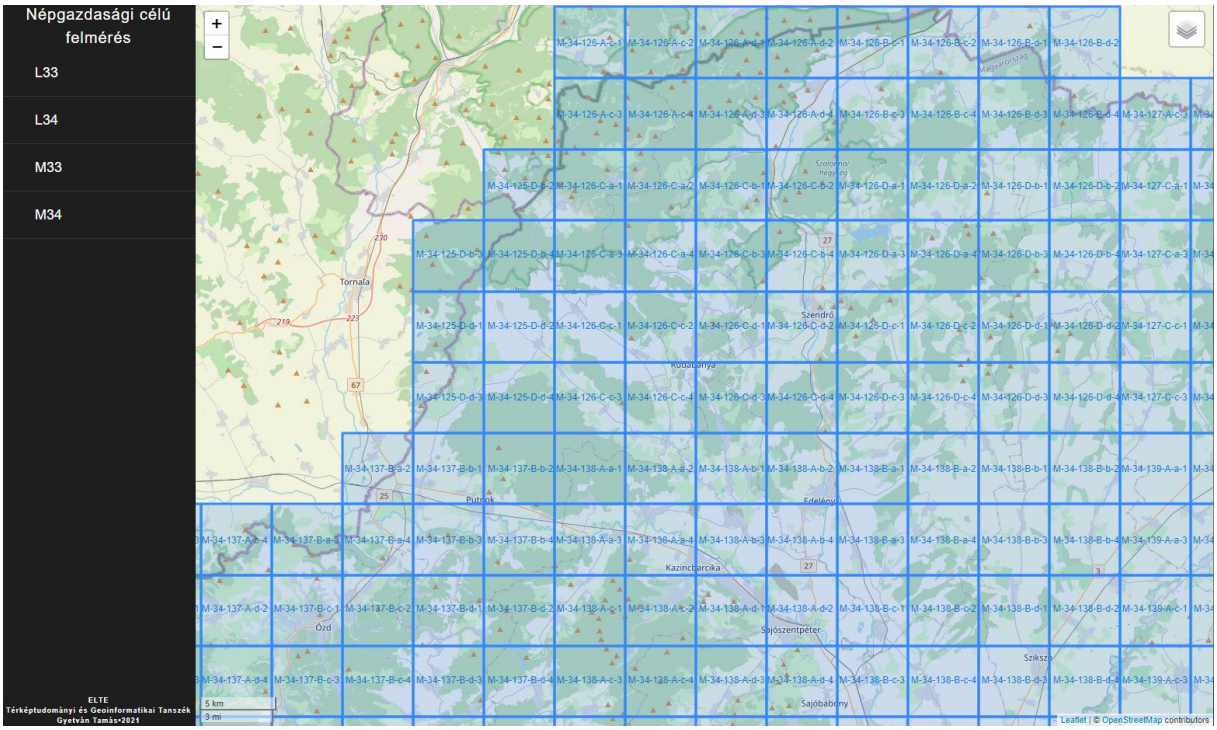
4.11. A katalógus megjelenése

A webkatalógus a http://mercator.elte.hu/~i6ofm4/index.html hivatkozáson érhető el. A katalógus kezdőoldalán tudjuk kiválasztani azt a zónát, amelyben megtalálható az általunk keresett térkép.



14. ábra: A katalógus kezdőoldala (saját ábra)

A megfelelő zóna gombjára kattintva nyílnak meg a katalógus különböző oldalai. A webkatalógus oldalai (főleg az L34-es zóna) néha kissé nehézkesen töltődnek be a számolások és a szelvényezés elkészítése miatt. Ez sajnos a böngésző kapacitásaitól függ. Miután betöltött az oldal, egyszerűen keressük meg az általunk megtekinteni kívánt térképszelvényt, és kattintással válasszuk ki. Ekkor a böngésző új ablakában megnyílik a keresett térképszelvény raszteres állománya.



15. ábra: Az M33-as zóna katalógusa (saját ábra)

5. Összegzés

A dolgozat elején kitűzött célt sikerült teljesíteni. Létrehoztam egy egyszerű, kényelmesen használható webes térképkatalógust, amely könnyen bővíthető az esetleges hiányzó szkennelt térképek állományaival. A diplomamunka első részében igyekeztem egy minél alaposabb történeti áttekintést adni az olvasónak a hazai polgári térképészetről, valamint a katalógus alapját szolgáló népgazdasági célokat szolgáló felmérésről. A második részben ismertettem a katalógus elkészítéséhez felhasznált programnyelveket. A harmadik részben lépésről lépésre bemutattam a webes katalógus egy oldalának az elkészítését.

A diplomamunkámban részletesen leírt módszerrel bármilyen felmérés vagy térképrendszer webes katalógusát el lehet készíteni, méretaránytól, állománytípustól függetlenül. A katalógus forráskódja megtekinthető, igyekeztem a katalógus elkészítésének a lépéseit a lehető legérthetőbb módon dokumentálni.

Irodalomjegyzék

Bene András (1974): Polgári topográfiai térképezésünk további feladatai. Geodézia és Kartográfia, 1974/1, 39-48.

Bene András (1981): A magyar polgári topográfia történetének áttekintése. Geodézia és Kartográfia, 1981/5, 320-324.

Klinghammer (1983): Klinghammer István, Papp-Váry Árpád: Földünk tükre, a térkép

Kratochvilla Krisztina (2017): Hazai szelvényezésű archív topográfiai térképek digitális feldoldozása. Catastrum, 4. évfolyam (2017), 1. szám

Nagy Zoltán (1985): Magyar topográfiai alaptérképművek, egyetemi doktori értekezés

Internetes hivatkozások:

Leaflet:

https://leafletjs.com/reference-1.7.1.html

https://leafletjs.com/examples/zoom-levels/

MDN Web Docs:

https://developer.mozilla.org/en-US/docs/Web/HTML

https://developer.mozilla.org/en-US/docs/Web/CSS

https://developer.mozilla.org/en-US/docs/Web/JavaScript

Stack Overflow:

https://stackoverflow.com/tags/html/info

https://stackoverflow.com/tags/javascript/info

https://stackoverflow.com/tags/css/info

Javascript.info

https://javascript.info/intro

SZAKDOLGOZAT / DIPLOMAMUNKA

EREDETISÉG NYILATKOZAT

Alulírott ……Gyetván Tamás…….Neptun-kód:….I6OFM4…...

ezennel kijelentem és aláírásommal megerősítem, hogy az Eötvös Loránd Tudományegyetem Informatikai Karának, Térképtudományi és Geoinformatikai Intézetében írt,

……Topográfiai térképek webes térképkatalógusa……..

…….....

című 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,
2021.05.14

……Gyetván Tamás s.k.…….
hallgató aláírása