ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Ács Péter**

2022 Hallgató törzskönyvi száma: **T/007001/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Ács Péter**
Törzskönyvi száma: T/007001/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Lemez felhasználás analizáló vizualizácós szoftver
Disk utilization visualization software

Intézményi konzulens: Sipos Miklós László
Külső konzulens:

Beadási határidő: 2022. május 15.

A záróvizsga tárgyai: Szoftvertechnológia és grafikus felhasználói interfész tervezése
Szoftvertervezés és -fejlesztés specializáció

A feladat

A feladat egy olyan asztali alkalmazás készítése mely egy adott mappát és almappáit, teljes meghajtót, esetleg külső eszközt (pendrive-ot, külső hdd-t) képes analizálni a foglalt tárterület szerint. Készítsen egy olyan felhasználói felületet, melyben az így kapott adatok között a felhasználó képes böngészni, területet felszabadítani, fájl útvonalat másolni, stb. Vizsgálja meg milyen adatvizualizálási formákkal lehet az így kapott adatokat olvasási szempontból felhasználóbaráttá tenni. Az alkalmazás biztosítson regisztrációs, valamint bejelentkezési felületet. A bejelentkezett felhasználó számára adjon lehetőséget az analitika naplózására. Biztosítson elérést az így naplózott fájl információkra, ezeket kizárólagosan csak a létrehozója érje el. Ehhez használjon ASP.NET Identity-t, amely a hitelesítést végzi, valamint az engedélyezési folyamatot kezeli. Az alkalmazás készüljön Windows rendszerhez, valamint vizsgálja meg a cross-platform lehetőségeit. A szoftver kezelőfelületét tekintve vizsgálja meg az aktuális trendeket és technológiákat, amelyek alapján hozza meg döntését a felhasznált technológiát illetően. Végezzen továbbá elemzést a tesztelési lehetőségekről és készítse is el az egyes teszt típusokat és teszteseteket.

A dolgozatnak tartalmaznia kell:

- a megoldandó feladat pontos leírását és részletes elemzését,
- magas szintű szerkezeti mintát az alkalmazásról,
- az egyes funkciók megvalósításához felhasználható lehetséges technológiákat,
- az elkészített rendszer tervét, blokkdiagramját,
- a megvalósítás menetét, felmerülő problémákat és az azokra adott megoldást,
- a tesztelési tervet, a tesztelés módszertanát és a tesztelés eredményeit,
- az elkészült szoftver felhasználási lehetőségeit.

Ph.

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. május 13.

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
Neumann Faculty of Informatics

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun kód:	Tagozat:
ÁCS PÉTER	[redacted]	NAPPALI

Telefon: Levelezési cím (pl: lakcím):

[redacted]

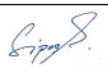
Szakdolgozat címe magyarul:

LEMEZ FELHASZNÁLÁS ANALIZÁLÓ VIZUALIZÁCIÓS SZOFTVER

Szakdolgozat címe angolul:

DISK UTILIZATION VISUALIZATION SOFTWARE

Intézményi konzulens:	Külső konzulens:
SIPOS MIKLÓS LÁSZLÓ	---

Alk.	Dátum	Tartalom	Aláírás
1.	2021.09.10.	Félév teendőinek, menetrendjének és mérföldköveinek ismertetése.	
2.	2021.10.18.	Hasonló rendszerek elemzése.	
3.	2021.11.22.	Irodalomkutatás, annak összegzése valamint konzekvenciák meghatározása.	
4.	2021.12.06.	Tervezés, követelmények specifikációja. Véglegesítés. Leadással kapcsolatos teendők egyeztetése.	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. (BSc) tantárgy követelményét teljesítette, beszámolóra bocsátható.

Budapest, 2021.12.06.

Sipos Miklós

intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun Kód:	Tagozat:
ÁCS PÉTER	[redacted]	NAPPALI TAGOZAT

Telefon: Levelezési cím (pl.: lakcím):

[redacted]

Szakdolgozat címe magyarul:

LEMEZ FELHASZNÁLÁS ANALIZÁLÓ VIZUALIZÁCIÓS SZOFTVER

Szakdolgozat címe angolul:

DISK UTILIZATION VISUALIZATION SOFTWARE

Intézményi konzulens:	Külső konzulens:
SIPOS MIKLÓS LÁSZLÓ	-

Alk.	Dátum	Tartalom	Aláírás
1.	2022.03.04.	Fejlesztés tervezett menetének revíziója, szakmai egyeztetés, fejlesztési teendők megkezdése.	
2.	2022.04.01.	Fejlesztés állapotának és haladásának áttekintése. Technikai problémák és módosítások egyeztetése.	
3.	2022.04.15.	Felmerülő problémák egyeztetése és megoldása. Tesztelési fázis megindítása.	
4.	2022.05.06.	Dokumentáció véglegesítése. Leadással kapcsolatos teendők egyeztetése.	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat II. tantárgy követelményét teljesítette, védésre bocsátható.

.....

Intézményi konzulens

Budapest, 2022.05.10.

Absztrakt

Mindenkivel előfordult az mikor a számítógépén helyet kell felszabadítani. Rengeteg szoftver létezik, amely az operációs rendszer és a gyakran használt szoftverek által ideiglenesen létrehozott fájljait tudja automatikusan törölni, ezzel helyet felszabadítva.

Sokszor azonban a legnagyobb helyet nem ezek a fájlok foglalják, hanem az adott felhasználó által mentett, letöltött fájlok. Az ilyen fájlok rendszerezése a felhasználó feladata, azonban egy nagyobb háttértárral rendelkező gép esetében ennek manuális végzése felettébb időigényessé vállhat.

Célom egy olyan szoftvert készíteni mely egy adott mappának vagy meghajtónak képes feltérképezni a foglaltsági szintjét és ezt vizualizálni a felhasználó számára. Így a manuális törlés, valamint ezeknek a fájloknak a rendszeresítési folyamata kevesebb időbe kerül. A szoftver regisztrációhoz kötötten naplóz, ezzel felhasználóhoz kötötten le lehet kérni korábbi feltérképezés során kiadott műveleteket.

Ennek segítésével visszamenőleg látható milyen volt a fájlrendszer mielőtt a felhasználó elvégezte a szükséges helyfelszabadítást és/vagy rendszerezést. A célok között szerepel a cross-platform támogatás emiatt választott keretrendszer a flutter. A flutter egy kódbázis alapján képes a legelterjedtebb webes, asztali, valamint mobiltelefonos környezetre készíteni alkalmazásokat.

A fejlesztés során elkészült a feltérképezés, majd a különböző megjelenítési adatstruktúrákban történő vizualizáció. Az elvárt fájlműveletek Flutter nyelv specifikus cross-platform elven keresztül készültek. Windows specifikus környezetben az alkalmazás képes fájlok vagy mappák helyét megnyitni, útvonalukat másolni, tulajdonságait lekérni, lomtárba helyezni vagy akár véglegesen is törölni. Az alkalmazás 3170 különböző fájl típust ismer, ezekről a típusokról konkretizált tőmondatokkal képes tájékoztatni a felhasználót az adott állomány általános használatimódjáról.

Abstract

Everyone has experienced running out of storage on a computer. There are several pieces of software which can help in that case. These kinds of programs usually delete temporary files generated by the operating system or by frequently used pieces of software. Most of the time, these files aren't the ones that take up a lot of space. It's the files saved or downloaded by the user. Managing these files should be done manually by the user, but if we consider a computer with a large storage system, this task can take up a lot of time.

My goal is to create a software which can help by mapping up a folder or a mounted disk. This mapped up image will help to visualize which files take up huge chunks of the storage. This way, the time to delete and/or manage these files will take less time to do. After registration, the software can log former operations. These operations are only accessible to the owner of the account. This feature helps to backtrack the managements.

My goals contain cross-platform support, for this reason the chosen framework is flutter. Flutter can generate for the most popular platforms such as web, desktop also mobile phones. The biggest advantage of using flutter is that only one codebase is needed to generate for multiple platforms.

During development, the specific filesystem was processed in different data visualization structures. The required file operations were performed using the Flutter language specific cross-platform principle. In a Windows specific environment, the application can open file or folder locations, copy their paths, retrieve their properties, place them in the recycle bin or even delete them permanently. The application can recognize 3170 different file types and can inform the user about the general usage of a file by providing specific descriptions of these types.

Tartalomjegyzék

1. Bevezetés

Az informatikában az adatok kezelése kulcsfontosságú, azonban ehhez szükséges az adatok valamilyen formában történő tárolása. A háttértár kapacitások méretének mértéke az elmúlt évtizedek során rengeteg technológiai mérföldkövet hódított meg. Több száz gigabyte-nyi háttértár gyors eléréssel már szinte az olcsó kategóriás laptopokban is alapvető. Összehasonlításképpen a világ első merevlemeze megközelítőleg 5 megabyte-nyi kapacitással rendelkezett. Külön érdekesség, hogy ennél a háttértárkapacitásnál egy megabyte-ra jutó költség megközelítőleg 10 000 dollárba került. Mai árfolyamot tekintve egy 500 gigabyte-os winchester megközelítőleg 10 000 forintba kerül. Tehát az egy megabyte-ra jutó költség körülbelül 0,00006 dollárra csökkent. Természetesen nagyobb háttértárat választva ezt a számot még kevesebbre lehetne csökkenteni.

Egy dolog nem változott, a tárkapacitás limitált, ennek beosztása a felhasználó felelősségi köre.

A szakdolgozatom célja egy olyan vizualizációs szoftver elkészítése, mely ennek a felelősségi körnek a megkönnyítésest fogja szolgálni. A készítendő szoftver ezt olyan módon teszi, hogy a kulcsfontosságú tárhelyfoglaltság mértékének felmérését, egy vizualizációs modellformával fogja könnyíteni a felhasználó számára.

A dolgozatomban megvizsgálom a jelenleg elérhető szoftver alternatívákat, valamint felmérem ezek funkcióit, elérhetőségeiket, a mai elterjedt operációs rendszer lehetőségek között.

A dolgozat legnagyobb inspirációja egy általam rendszeresen használt szoftver, a WinDirStat. Mely funkcióköre és relevanciája egy rendszergazdai körben egy kifejezetten elterjedt szoftverlehetőség. Ez a szoftver kevés helyet foglal, azonban egy rettentően hasznos tárhelyfelszabadító eszköz, melyet rendszeresen ajánlok az ismeretségi körömben.

Sajnos ezen szoftver nem minden esetben működik tökéletesen, van néhány hiba, összeomlás, mely előidézhető. Azonban mai napig kijelenthetően a legelterjedtebb Windows alternatíva.

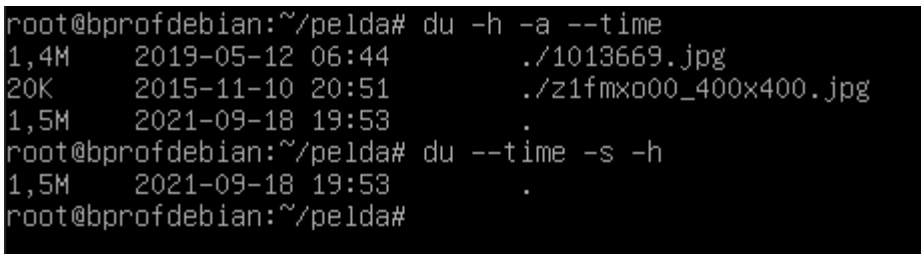
Célom e szoftver funkciótárát kiegészíteni, valamint modernizálni mai fejlesztési- és grafikus trendek alapján.

2. Irodalomkutatás

2.1 Hasonló rendszerek kutatása

2.1.1 Lemez kihasználtság megtekintése a mai grafikus szoftverlehetőségek előtt

A mai grafikus szoftver lehetőségeket megelőzően már a linux alapú rendszerekben is megtalálható volt beépített terminal parancs a lemez foglaltság vizsgálatára. Ez a parancsa a du karaktersorozattal elérhető. Ezt különböző paraméterek segítségével lehet használni. Ahogy ezt a 2.1-es ábra is mutatja.



2.1. ábra du parancs debian környezetben

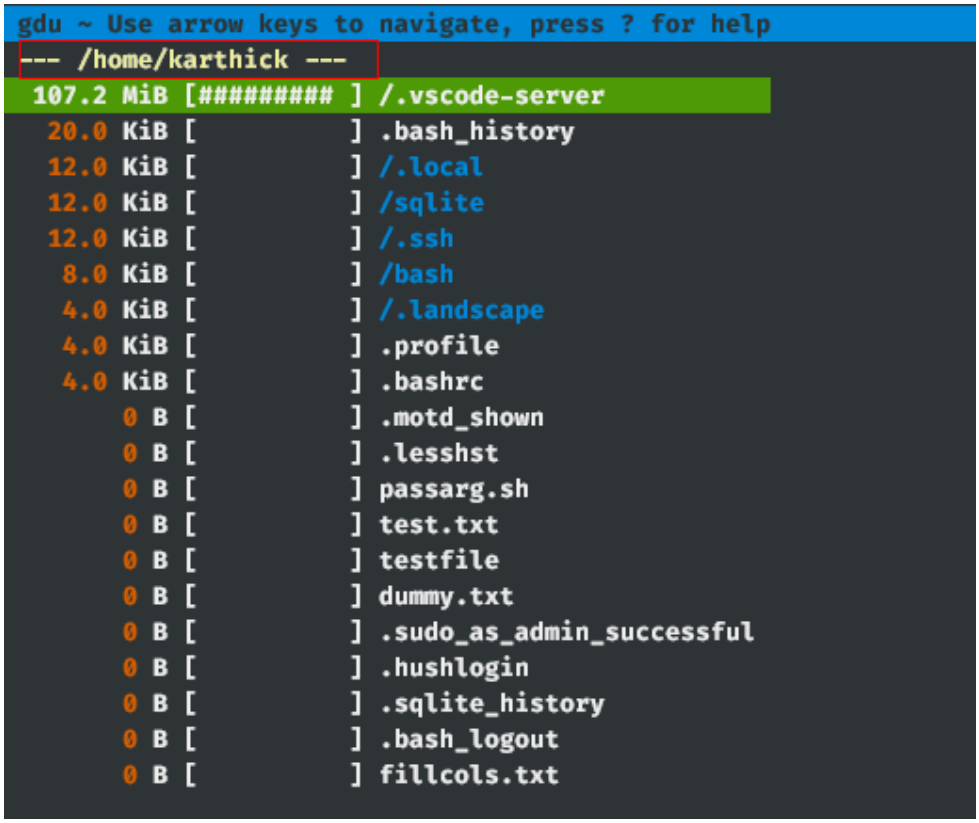
Ebben a „pelda” elnevezésű mappában megfigyelhető, hogy kettő fájl található, bemeneti paraméterek pedig:

- *-a* : (all) listázza az összes fájlt nem csak a mappákat, mint eredetileg
- *-h* : (human readable) ember számára olvasható formátumban (Jelen esetben K kilobyte, valamint M megabyte-ként megjelenítve)
- *--time* : Megjeleníti az utolsó módosítási dátumot
- *-s* : (summerize) összesíti és megjeleníti a teljes mappa tartalmát

A du parancs előnyei, hogy bash nyelv segítségével könnyen lehet akár naplózási scriptet írni, amelyet cronjob scriptként akár automatikusan is lehet futtatni. Linux rendszerben alapértelmezett, tehát nincs szükség semmit telepíteni.

Hátránya, hogy teljes háttértár vizsgálat esetében nem nyújt fájlonkénti lebontást. Mappa tartalom szerinti megjelenítést végez, ahhoz, hogy a mappán belüli tartalom vizsgálásra kerüljön az adott mappában kell kiadni a parancsot. Szintén hátrány, hogy a grafikai megjelenítés mai szemmel nézve kevés. Nem nyújt vizuális információt a fájl, vagy a mappák méretéről, ami könnyen feldolgozható az ember számára. Csak a méretegység számot jeleníti meg. [1]

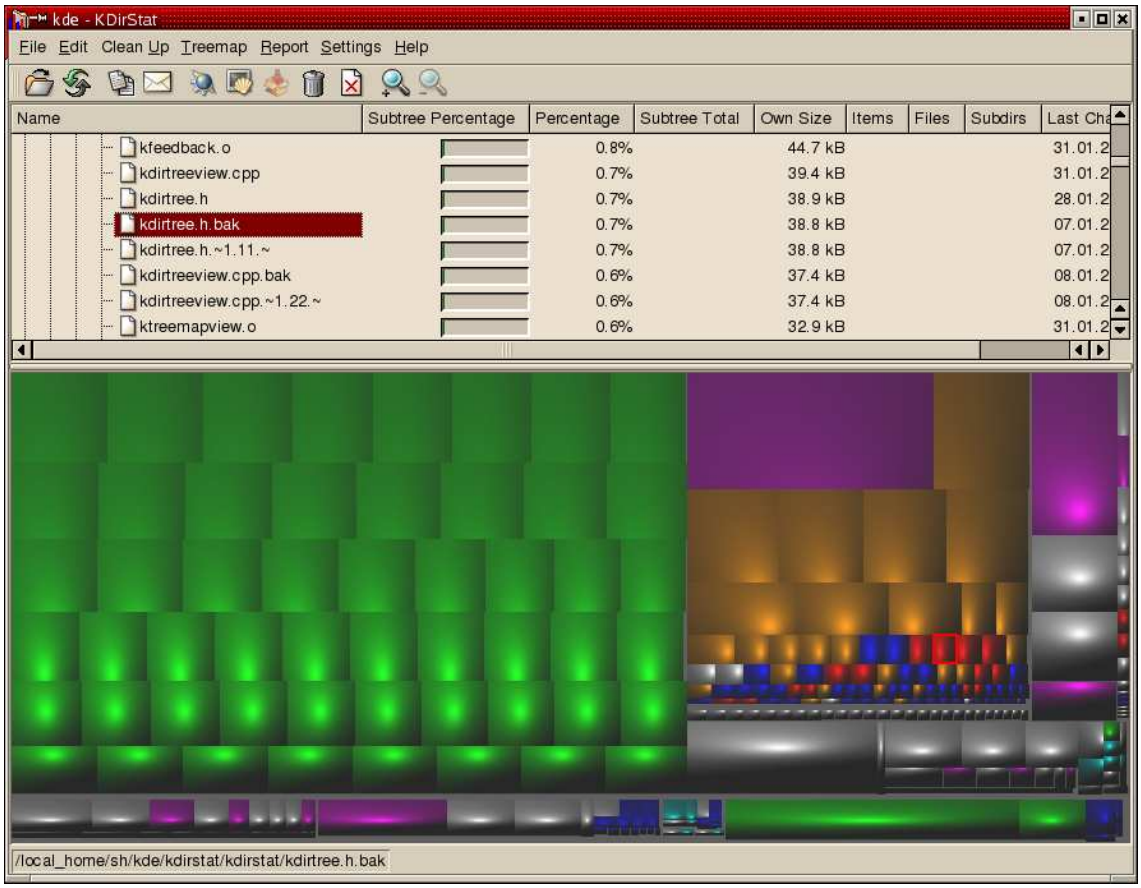
Linuxra telepíthető szoftver azonban a *gdu* parancs, azaz „graphical disk usage”. Ez a szoftver *du* parancs hiányosságait próbálja pótolni. Ez a szoftver már nyújt vizuális információt, valamint grafikus környezetet alakít ki egy parancssoros felületen. Használ színeket, amely a lekért információ feldolgozását könnyíti az ember számára. Ez a kezdetleges vizualizációs visszajelzés gyorsítja az információ leolvasását. Valamint új parancsok segítségével bővíti az alapértelmezett parancs készletét. Lehetőséget nyújt adott karakteregyezést kizárni a *-i [egyezési szöveg]* vagy *–ignore-dirs [egyezési szöveg]* segítségével. Képes naplózni is a *-l [elérési út]* vagy *–log-file [elérési út]* parancs használatával. [2] A 2.2-es ábra jól mutatja a foglaltsági sávot a szoftverben.



2.2. ábra GDU futás közben

2.1.2 Az első meghatározó grafikus szoftver lehetőség

Az első meghatározó grafikus szoftverek egyike, amelyben a főbb szempont a könnyű vizualizálás volt az a KDirStat. Ez a KDE Unix alapú grafikus operációs rendszer 3-as verziójához készült 2006-ban. Az operációs rendszer szoftvercsomagjának része volt. A szoftver teljesen nyílt forráskódú és ingyenes. A legnagyobb sajátossága a grafikus treemap panel. Ennek felépítése a mappa rendszerhez igazított fájlméret alapján történik. Ezen a felületen történő kattintásra a megfelelő fájlhoz vezet a szoftveren belül így könnyen kezelhető, megtalálható mely fájlok rendelkeznek a legnagyobb területfoglaltsággal. A treemap-ban szereplő blokkok különböző színekre vannak bontva, fájl típus csoportosítás szerint. Ezzel is segítve az áttekinthetőséget, mint ahogy ez látszik a 2.3-as ábrán. [3]



2.3. ábra KDirStat futás közben

Mivel ez egy Linuxra készült szoftver így Windows rendszeren nem futtatható.

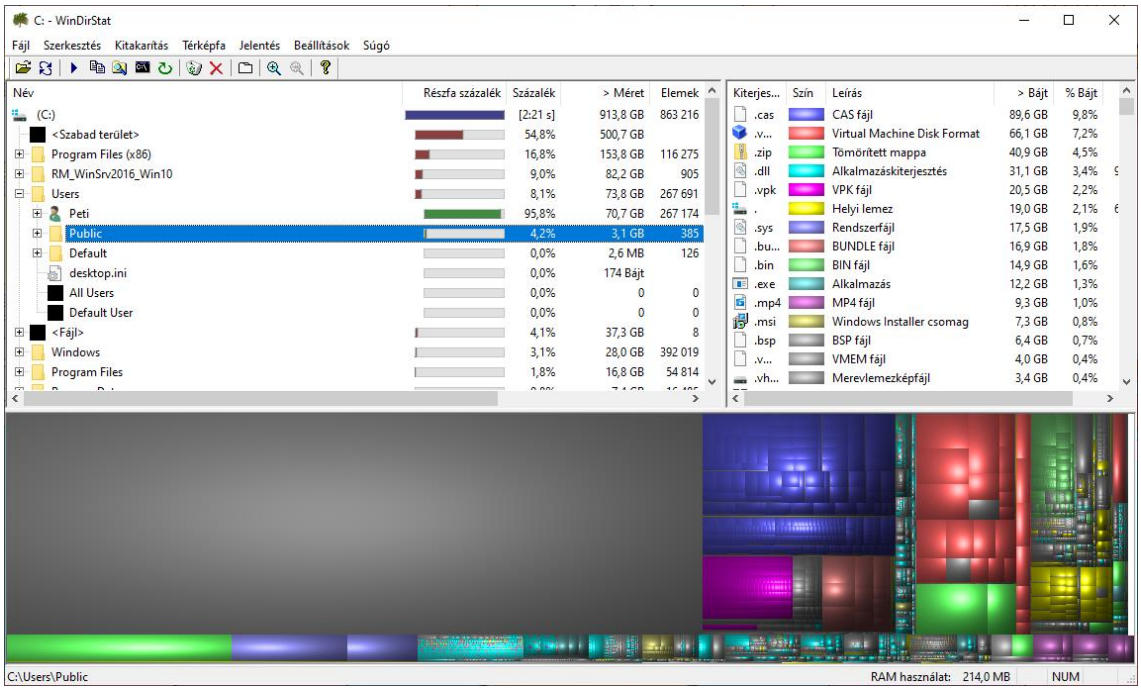
Ezt a KDirStat projektet a fejlesztők újra feldolgozták, jelenleg QDirStat néven elérhető. Ezen dokumentum írásakor még támogatott, az 1.8-as verzió a legfrissebb. [4]

2.1.3 Windows alternatívák

A KDirStat hatalmas sikere által több fejlesztő inspirációt kapott egy Windowsos szoftveralternatíva készítésére. Ez a ma ismert WinDirStat nevű szoftver.

A WinDirStat 2003-ban jelent meg, jelenleg több mint 17 millió letöltéssel rendelkezik a sourceforge oldalán, a fosshub oldalán pedig 10 millióval. A szoftver teljesen nyílt forráskódú, ingyenes, valamint teljes mértékben transzparens a tényről, hogy a KDirStat Windows-os klónja. [5] Ezt a hivatalos főoldalán észre is lehet venni, mivel kiemelték, hogy amennyiben Linux rendszerre keres szoftvert használja a QDirStat-ot. Azonos módon treemap megjelenítést használ, valamint alkalmazza fájlok színezéses szerinti csoportosítást is. Nyelvet tekintve több különböző integrációt tartalmaz, ezek között a magyar nyelvet is. Ezt a fordítást Leonard Nimoy készítette az 1.1.2-es változathoz 2006-ban. WinDirStat fejlesztését nyílt forráskódú szoftver elven teszik. A GitHub oldalán mai napig követhetőek módosítások és hozzájárulások a szoftverhez. [6]

Az alábbi 2.4-es ábrán egy saját környezetben futó WinDirStat látszik, jól látható a különböző színezés, valamint a fájl kiterjesztés lista (jobb felső sarok).



2.4. ábra WinDirStat futás közben

A WinDirStat a KDirStat alapjai mellé kiegészítette a szoftvert egy kiterjesztési listával, mely a beszúrt kép jobb oldalán található. Ez a lista minden különböző kiterjesztési típust beazonosít, valamint különböző színt rendel ezekhez, ezzel is növelve a könnyed kezelést.

A szakdolgozat ideje alatt aktuális Windows 10-es környezetben a szoftver működőképes, azonban különböző át méretezések, nagyítások szoftverösszeomláshoz vezethetnek.

A WinDirStat-on kívül több Windows specifikus szoftver elérhető említésképp néhány:

- FolderSize,
- Scanner
- SpaceSniffer

Megfigyelhető azonban, hogy ezen szoftverek sikere közel sem akkora, mint a WinDirStat-é. Az egyetlen elérhető analitika a SpaceSniffer-ről található, 3,5 millió letöltéssel rendelkezik, ami töredéke a konkurensének. [7]

2.1.4 Mac OS alternatívák

Egy Mac OS X-re készült lemez analizációs, valamint vizualizációs szoftver a Disk Inventory X. Ez ismételten egy KDirStat által inspirált szoftver. Szintén nyílt forráskódú, ingyenes és treemap megjelenítést használ. A korábban említett WinDirStat fő oldalán ez a szoftver is kiemelt alternatíva amennyiben Mac OS X-re keres szoftvert. [8] Ez a szoftver már nem támogatott, valamint az újabb Mac OS X-es rendszereken kompatibilitási okok miatt nem futtatható.

A szoftver fejlesztője a Disk Inventory X GitHub oldaláról egy másik szoftvert ajánl ez a GrandPerspective.

A GrandPerspective nyílt forráskódú és szintén treemap megjelenítési elvet használ. Jelenleg több mint 2 millió letöltéssel rendelkezik a sourceforge oldalán. [9]

Érdekesség, hogy a szoftver hivatalos oldalán a következő sorok találhatóak:

„Download GrandPerspective from Sourceforge for free or from the App Store for $2.99. You'll get the same app either way.” [10]

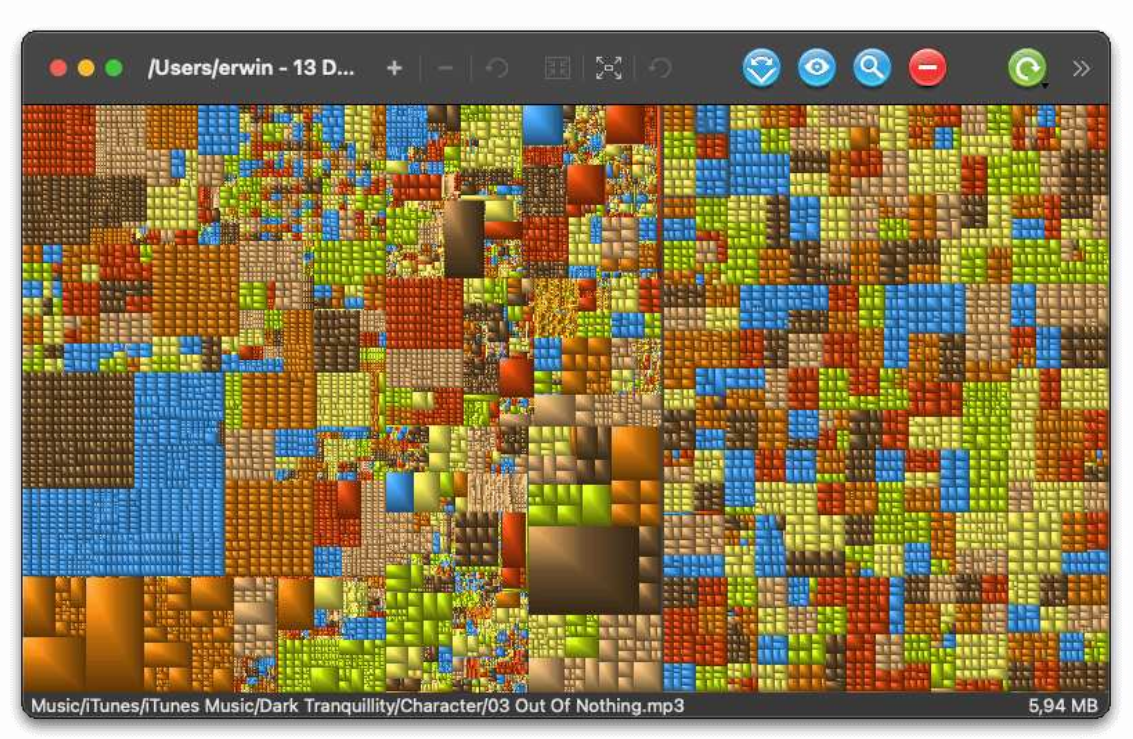
A fejlesztő kiemeli, hogy a program ingyenes azonban az App Storeból 2,99 dollárért is meg lehet vásárolni.

Azt, hogy a kettő között mi a különbség, a következő sorokban fejti ki:

„So what's the difference? When you get it from the App Store you know that it has passed Apple's review and quality control, you receive automatic updates, you help cover distribution costs and you support further development.” [10]

Tehát a legfőbb különbség az, hogy az Apple felülvizsgálta a szoftver adott verzióját, valamint automatikus frissítéseket kap, aki fizetett a szoftverért. Szintén kiemelt, hogy a vásárló ezzel segíti a fejlesztőt a szoftver terjesztésében, valamint támogatja a további fejlesztését. Ezen a platformon keresztül a szoftver egyszeri vásárlást igényel.

Az alábbi képen a GrandPerspective látható futás közben, túlnyomóan zsúfolt, azonban a főbb funkciókat képes betölteni. Felül látszódnak a különböző opciók. Ilyen például a közelítés lehetősége is.



2.5. ábra GrandPerspective futás közben

Mac OS X-re egy fizetős jólismert alternatíva a DaisyDisk nevű alkalmazás.

A DaisyDisk ellenben a legtöbb korábban említett szoftverrel nem nyílt forráskódú. Az alkalmazás teljes használata fizetést igényel. Ez egyszeri vásárlással 9,99 dollárba kerül, azonban trial verzió is elérhető javarészt korlátolt funkcionalitásokkal. A korábban említett szoftverekkel ellenben, a DaisyDisk képes felhőtárhely analizálására is. [11] Ezt a három legelterjedtebb felhőtárhelyben lehet végezni, név szerint:

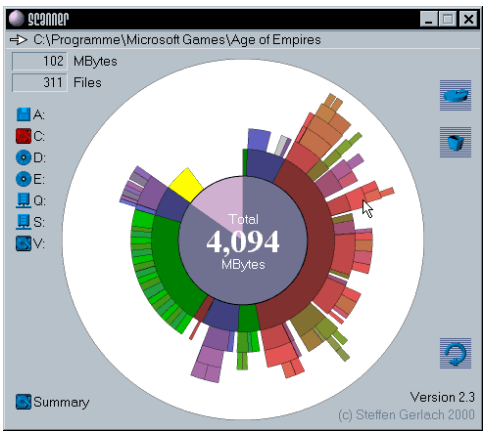
- Dropbox
- Google Drive
- Microsoft One Drive

Az ingyenes trial verzióban korlátolt funkciókra példák:

- Nem engedélyezett a rendszergazdai fájl vizsgálás.
- Nem engedélyezett a felhő tárhely vizsgálat.
- Nem engedélyezett a talált fájlok alkalmazáson belüli törlése.
- Nem engedélyezett a talált fájlok intézőben történő megjelenítése.

Különbség még, hogy nem treemap megjelenítést használ, hanem egy tovább fejlesztett kördiagrammot un. Sunburst diagramot használ, melyet egy viszonylag modernizált dizájnnal bont szét. A szoftver 2008-ban jelent meg, azonban ez a sunburst megjelenítés mód korántsem új ötlet a lemezanalizáló szoftverek terében. A korábban említett Scanner nevű Windows specifikus, nyílt forráskódú szoftver ugyanezt a kördiagramm dizájnt fejlesztette már 1999-ben a Wikipédia oldala szerint. [12]

A megjelenítési módok összevetése az alábbi 2.6-os és 2.7-es ábrán láthatóak.



2.6. ábra Scanner megjelenítés mód

2.7. ábra DaisyDisk megjelenítés mód

A rendszerek összefoglalásához elkészítettem az alábbi táblázatot, kiemelve a legfőbb szempontokkal:

Vizsgált rendszer	Operációs rendszer	Ingyenes	Támogatott	Pozitív	Negatív
Du Parancs	Linux	Igen	Igen	Alapértelmezetten elérhető Linux operációs rendszerekben	Nincs vizualizáció és funkció lehetőség
GDU szoftver	Linux	Igen	Igen	Jól kiegészíti az alapértelmezett parancs hiányosságait	Kevés vizualizáció és funkció lehetőség
KDirStat	KDE	Igen	Nem	Felettébb meghatározó szoftverújításokat és trendeket hozott magával	Már nem támogatott, azonban új verziója a QDirStat elérhető
WinDirStat	Windows	Igen	Igen	Legelterjedtebb Windows alternatíva, lebontja fájl szerinti terület foglaltságra is az érintett mappát/meghajtót	Összeomlás lehetősége fennáll, átméretezés hosszabb töltések esetében.
GrandPerspective	Mac OS	Igen	Igen	Ingyenes könnyen elérhető alternatíva.	Áttekinthetőségét tekintve kifejezetten nehéz a többi alternatívával összevetve.
Daisy Disk	Mac OS	Igen (túlnyomó részt korlátoltan)	Igen	Az első szoftver, melyben lehetőség van felhőtárhely feldolgozásra. Kifejezetten modern grafikus dizájn.	Ingyenes verzió esetében a szoftver túlságosan korlátolt.

2.1. táblázat Hasonló rendszerek összegzése

2.2 Kapcsolódó témakörök

2.2.1 Az adattárolás

Az adatok tárolása bináris formában, minden fájltípus konvertálható egyesekre és nullákra. Ennek a bináris számsorozatnak az értéke, terjedelme határozza meg egy fájl méretét és tartalmát. A tárolás fizikális módját, valamint az adatok, fizikális olvasásának a menetét az adott tároló hardver határozza meg. A fájlok tárolására szolgáló hardvereket közvetlen elérésű adattárólóknak nevezik. Ezekről elmondható, hogy nincs szükségük tápfeszültségre ahhoz, hogy a tárolt adatot megőrizzek.

Ide sorolható példák:

- HDD (Hard disk Drive)
- SSD (Solid State Drive)
- Pendrive (USB Flash Drive)
- SD Card

Az előbb felsorolt példák mellett még rengeteg tárolási módszer ismert. E néhány példa azon a tárolásimódszereket összegzi, amivel szinte mindenki találkozhat a mindennapokban. A jelenlegi direktívák szerint lettek felvezetve. Ebben az iparterületben is hatalmas fejlődések figyelhetőek meg az elmúlt évtizedek során.

Megfigyelhető, hogy HDD-n kívül az összes többi tároló flash alapú. Ezt flashmemóriának nevezik. A flashmemória az EEPROM egy továbbfejlesztett változatának nevezett Flash EEPROM. A mai Flash alapú tárolók elterjedéséhez a korábbi lehetőségek vizsgálata szükséges.

Időrendi sorrend szerint technológiát tekintve a következő írható fel néhány ROM típusról, név szerint:

- ROM (Read-only Memory)
- EPROM (Erasable Programmable Read-only Memory)
- EEPROM (Electrically Erasable Programmable ROM)
- Flash EEPROM

Ezekről a változatokról tárolást tekintve kijelenthető, hogy ugyanazt a célt szolgálják. Azonban a különböző típusok különböző újításokat engedtek meg a funkcionalitás szempontjából.

Egy szimpla ROM-ban az általa hordozott adat egyszeri statikus eltárolását engedi, ezen adat újra írásához változtatásához teljes ROM chip cseréréjére szükség van.

A későbbiekben EPROM esetében már törölhetőséget, és újra írást enged meg, erre több módszer is ismert. Ide sorolható példa az UV-EPROM. UV-EPROM során a chip törlési folyamata UV sugárzással történik. Hosszabb folyamatnak tekinthető, mivel a chip alaplapról történő leforrasztása után csak az UV sugárzásnak kitett időtartam 5-20 perc közé esik. Ezen törlés után az UV-EPROM újra módosítható, programozható. Ezt a programozási folyamatot egy EPROM újra égetésének nevezik. A folyamat során a chip gyártója által meghatározott feszültség (ez az úgynevezett VPP) szerint történik az újra égetési folyamat. Ehhez folyamathoz céleszköz szükséges.

EEPROM során legfőbb különbség, az, hogy az újra programozhatósághoz nincsen szükség a nyáklapról történő leforrasztásra, valamint a teljes adattartam törlése sem szükséges (a korábban említett EPROM-hoz képest). Adat törlés során meg kell határozni azt, hogy melyik byte törlése hajtódjon végre. Ahhoz, hogy az újra programozhatóság megtörténhessen, a teljes áramkörtervben szerepelnie kell egy olyan vezérlőnek, amivel ez a folyamat lebonyolítható. Ilyen esetben nincs szükség az újra programozáshoz céleszközre, ilyenkor ez egy rendszerben integrált. Ez indokolja azt, hogy az EEPROM használata a korábban említett UV-EPROM-mal összevetve egy relatív drágább módszertan.

A Flash EEPROM esetében a flash szó angolról lefordítva villanás. Ez az angol nyelvhasználat során „in a flash” szavakba foglalható. Magyarra fordítva azonnal, egy szempillantás alatt szavaknak, kifejezésnek felel meg. Ez a korábbi módszerekkel összevetett adatváltoztatási előrelépést foglalja önmagában. A Flash EEPROM-ra lehet utalni más néven, rövidebben ez a szóhasználatban Flash Memory-ként terjedt el (magyarra fordítva flash memória). A Flash EEPROM során a törlési folyamat elektronikus, legfőbb különbség EEPROM és Flash EEPROM között az, hogy Flash esetében ez nem egy meghatározott byte szerint történik, hanem teljes tartalom szerint. Ez az idő során úgy változott, hogy Flash esetén blokkokra osztott részekről lehet beszélni. Egy ilyen blokk törlése lehetséges. SLC (Single Level Cell) esetében a memória cella mérete 1 bit. Az újra írhatóság, törlés, valamint a költséghatékonyság előrelépése tette lehetővé azt, hogy a BIOSROM tárolása flashmemoriában történik, lehetővé téve a BIOS frissítést. [13]

Negatívum az, hogy a flash memóriáknál a programozási és törlési ciklusok száma (un. P/E cycle) véges. Blokkosított eszközök esetében (mint például egy Pendrive) ezt egy úgy nevezett FTL (Flash Transition Layer) segíti. Ez olyan szoftver vezérlés melynek során a különböző blokkok kihasználása egységesíti a degradálást a blokkok között.

Az SSD-k (Solid State Drive-ok) elterjedése nagymértékben a flash memóriák utóbbi időben történt fejlődésének köszönhető. Valamelyest elkezdte leváltani a HDD-ket adott kereteken belül.

A HDD másnéven winchester egy elektromechanikus mágneses elven működő adattároló. Az első HDD 1957-ben készült, az IBM 305 RAMAC számítógéphez. Az eszköz hatalmas fejlődéseken ment keresztül, felettébb érdekes, hogy a mai napig mindennapi használatban lévő elterjedt eszköznek számít. [14]

Az alábbi táblázat a fejlődést összefoglalja és számszerűsíti ezt:

Parameter	Started with (1957)	Improved to	Improvement
Capacity (formatted)	3.75 megabytes[18]	18 terabytes (as of 2020)[19]	4.8-million-to-one[20]
Physical volume	68 cubic feet (1.9 m^3)[c][6]	2.1 cubic inches (34 cm^3)[21][d]	56,000-to-one[22]
Weight	2,000 pounds (910 kg)[6]	2.2 ounces (62 g)[21]	15,000-to-one[23]
Average access time	approx. 600 milliseconds[6]	2.5 ms to 10 ms; RW RAM dependent	about 200-to-one[24]
Price	US$9,200 per megabyte (1961; US$83,107 in 2021)[25]	US$0.024 per gigabyte by 2020[26][27][28]	3.46-billion-to-one[29]
Data density	2,000 bits per square inch[30]	1.3 terabits per square inch in 2015[31]	650-million-to-one[32]
Average lifespan	c. 2000 hrs MTBF[*citation needed*]	c. 2,500,000 hrs (~285 years) MTBF[33]	1250-to-one[34]

2.8. ábra HDD-k fejlődése a megjelenésük óta

Kijelenthető, hogy a flash memóriák újabb technológiát képviselnek, SSD-k szempontjából vizsgálva az első flash memória alapú számítógép 2006-ban jelent meg. Ez a Sony Vaio UX90 volt 16 GB-nyi flash memóriával. [15] (ezt a későbbiekben 32GB-ra váltották ebben az eszközben). Az Apple a Macbook Air-rel 2008-ban mutatta be, és nyújtotta lehetőségként az SSD-ket, 2010-től kezdve pedig minden modellhez SSD vagy Hybrid Drive lehetőséget ajánl.

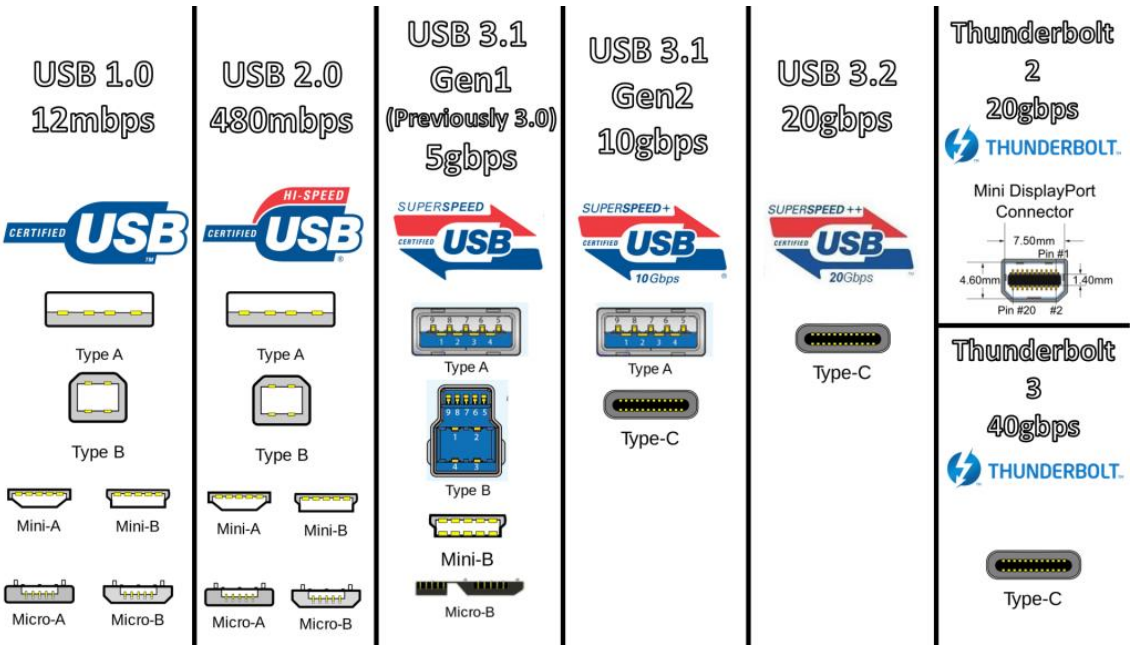
A Hybrid Drive az SSD-k és a HDD-k által előnyben nyújtott lehetőségeket és technológiákat próbálja egységesíteni egy kevert más szóban „hibrid" eszközben. Nagymértékben a gyorsítótáron alapszik, azonban kevésbé lett elterjedt lehetőség a gyártók által.

Mivel az SSD-ékben nincsen mechanikus rész, ellenben a HDD-kel melyben az olvasófej forgó lemezről olvas, így megfelelő használat mellett nincs kitéve mechanikus meghibásodásnak. Szintén emiatt nincs alapzaja, ellenben egy HDD-hez képest, melyben a fordulatszám akár 7200 RPM (fordulatszám/perc) is lehet. Olvasást, valamint írást tekintve, kollektív kijelenthető, hogy gyorsabbak, mint a HDD-k. Költséghatékonyság szempontjából az SSD-k tárkapacitást tekintve drágábbak. A korábban említett P/E(program/erase) ciklus limitálja az SSD-k élettartamát, mivel ezen limitek elérése hibás blokkhoz vezetnek. A hibás szektorok kezelése HDD-kben SSD-ékben azonos, amennyiben hibás szektor problémája elő áll ezt képesek feltérképzeni, és kizárni a használatból. Rengeteg kutatás készült az SSD-ék élettartamáról, kijelenthető, hogy élettartamot tekintve az SSD-k képesek leváltani a HDD-ket, ez szerverkörnyezetben is teljes mértékben előrelépést jelenthet az írási és az olvasási sebességek miatt. (Figyelembe véve a költséghatékonysági szempontokat.) [16]

Egy kutatásban 2015-ben teszt alá vetettek különböző gyártók által készült 240/256GB-os SSD-éket. A célja az volt, hogy megvizsgálják, hogy mekkora adatforgalom után hal meg valójában egy ekkora SSD. Átlagosan megközelítőleg 2 PB-nyi adatforgalom után mentek tönkre ezek a 240-256 GB-os SSD-k. [17] Személyes tapasztalatom, egy 120 GB-os Kingston márkájú SSD-vel az, hogy 6 év alatt 9 TB-nyi adatforgalom után még tökéletesen használható állapotban van.

Rengeteg tárolásra használt módszer már elavultnak tekinthető. Ez abból a perspektívából igaz, hogy az újabb eszközök gyorsabb adatátvitelt, nagyobb átviteli sávszélességet, kapacitást, valamint átlagosan nagyobb élettartamot is nyújtanak, mint az elődjeik. Az adat szemszögből megközelítve ezt kijelenthető, hogy amennyiben egy felhasználó hozzáfér a szükséges hardver követelményeihez (például egy bakelit lemez esetében egy bakelit lejátszóhoz), ilyen esetben maga az adat az nem avul el. Egy tárolási mód akkor nevezhető elavultnak, ha a hozzá szükséges hardver elérhetőség mértéke a piacban relatív kevés, valamint a hardver piac kollektív már más irányzat felé megy. Például az operációs rendszer telepítési folyamata során megfigyelhető a használt lemezképfájl. Régebben Floppy lemezről majd CD-ről későbbiekben pedig DVD-ről volt ez megszokott. Jelenlegi irányzat a Pendrive-ról történő telepítés. Ilyenkor az legújabb USB port szabványa által nyújtott sebességi előrelépések kihasználásával a telepítési gyorsabb. Ezt az tároló és a gép által nyújtott port verzió szabja meg.

Az USB típusok előrehaladásról, sávszélességet tekintve az alábbi 2.9-es ábra sok információt nyújt:



2.9. ábra USB adatcsatorna sávszélesség fejlődése

Az előbb említett telepítési módok és még más trendek (például laptopoknál a könnyítés vékony dizájn) miatt, jelenleg nagyon jó példa erre a DVD olvasók szerepe. Ez a laptopokban, olyan szinten változott, hogy korábban a telepítés miatt alapszükséglet volt, jelenleg pedig nagyobb mértékben opcionális bővítőkártyaként elérhető. Asztali számítógépek esetében pedig, a lemezolvasó gépház integrációt tekintve, jelenlegi piacot tekintve nem minden gépházban elérhető. (Néhány példa ilyen mai gépházra: Corsair D5000, Fractal Torrent, Fractal Meshify, NZXT h510, Lian Li o11)

Portok szempontjából érdekesség az Európai Unió bizottsága az IP/21/4613 számú határozatban a portok egységesítéséről döntött. Az USB-C típusú port egységesítése a környezeti kihatásokról, elektronikus hulladék teremélési szempontból kifejezetten érdekes probléma kör. [18]

2.2.2 Fájlrendszerek

Fájlrendszernek azt nevezik, ami azt metodológiát és adatstruktúrát határozza meg, mely egy operációs rendszer számára lehetővé teszi az adatok vezérlésének a menetét. Fájlrendszer nélkül egy háttértáron lévő adat az valójában csak egy hosszú bináris karaktersorozat lenne, nem lenne megállapítható az, hogy hol kezdődik egy adat, és hol fejeződik be. A fájlrendszer legfőbb célja ezt izolálni, és azonosíthatóvá tenni. Ez az azonosíthatósági struktúra, valamint a művelet logika csoportosítja a nyers bináris adatot, ezt nevezik fájlrendszernek.

Rengeteg különböző fájlrendszer ismert. Ezek tulajdonságai meghatározható flexibilitás, méret, biztonság, sebesség és még sok különböző tulajdonság szerint. A fájlrendszerek sokszor egy adott hardver működési logika szerint vannak meghatározva. Például az ISO 9660 egy optikai lemezekhez készült fájlrendszer. [19]

Microsoft Windows specifikus fájlrendszerek:

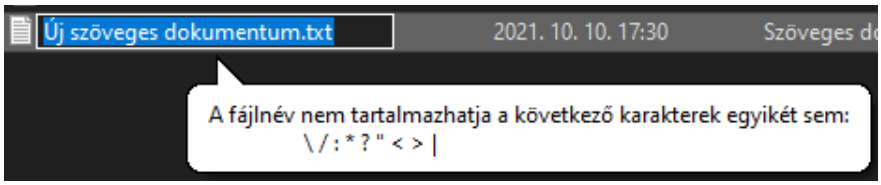
- NTFS
- FAT
- exFAT
- Live File System
- ReFs (Windows Server 2012-től, valamint, Windows 8-tól)

Windows specifikus fájlrendszer esetében a meghajtók, vagy partíciók az elérési út elejében tárolt betűvel jelzett érték. (Például C:\Users a C meghajtón, vagy partíción tárolt Users rendszer mappa). A C konvencionálisan az elsődleges partíciót jelöli, a rendszer innen tölt, más szóval bootol be az indítás során. Telepítés során a C meghajtón a rendszerspecifikus elérési útvonalak feltöltésre kerülnek, ezen fájlok törlése módosítása sokszor korlátolt a rendszergazdák számára is. A C elnevezéskonvenció az MS-DOS alapokhoz visszavezethető, ahol az A és a B betű foglalt volt a floppy meghajtók számára. Partíciónálás vagy új lemez rendszerbeléptetése után a lemezjelölő betű a disk mannagement (lemezkezelő) rendszerszoftverben módosítható más betűre az angol abc szerint, egyénileg azonosítható betűnek kell lennie. [20]

A Windows-on belül egy fájl útvonala fa szerkezetet alkot. A gyökérpont maga a korábban meghatározott meghajtó betűjele. A legalsó gyermek, másnéven levél a fájl. Ennek a szülői a gyökérpont kivételével azon mappák, melyben az adott fájl szerepel.

(Elképzelhető olyan eset is, hogy a meghajtó gyökérpontja alatt egy levél elem van, ilyenkor ez maga a fájl, ebben az esetben egy két elemű fa szerkezetről van szó.)

Elnevezési konvenciókat követve fájl és mappa elnevezések szempontjából a Windows NTFS-ben kis-nagy betű érzékeny rendszer. Ami azt jelenti, hogy megkülönböztetést tesz ezek között. (Például az file.html és File.html egy mappa alatt két különböző fájlként van azonosítva, ez a mappákra is igaz.) Mai Unicode karakterkészlet szerint több különböző karaktert használhatunk, lehet például ékezetes karakter is egy fájl, vagy mappa nevében. Azonban vannak olyan speciális karakterek, amiket nem lehet használni, ilyen fájlok létrehozása során az alábbi 2.10-es ábrán megjelenített üzenettel találkozhatunk:



2.10. ábra Windows 10-es környezetben lévő kizárt karakterek

Az így kizárt karaktereken kívül még néhány konvenció fent maradt, nem használhatóak speciális karakter sorozatok sem az elnevezés során ilyen például a „con” fájl elnevezés. Ezt Windows rendszerekben nem lehet létrehozni. Több ilyen konvenció létezik mai napig a Windows rendszerekben. [21]

Szintén igaz, egy Windows operációs rendszerben a fájlok kiterjesztése pont karakter elválasztással történik. Ez határozza meg a fájl típusát a rendszer számára, melyből kiválasztja a rendszer által, vagy alapértelmezetten társított szoftver által megadott ikont. Ez az un. suffix, a fájlhoz tartozó utótag három vagy négy karakter lehet. (Konvenció szerint nev.exe esetében a nev a fájl neve, az exe, executable futtatható állomány pedig a kiterjesztés)

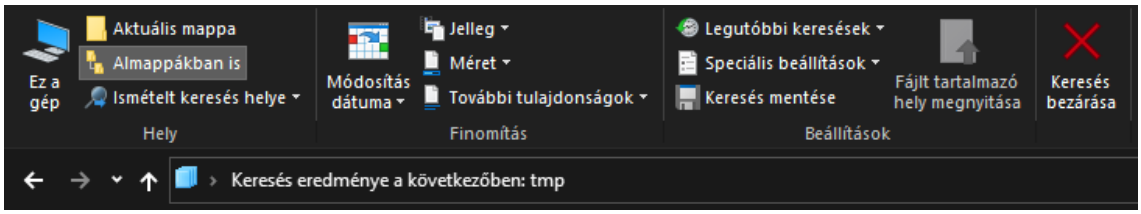
Korábban említett tároló hardver felvezetésben olvasható, hogy a hardverek különböző szektorokba teszik az adatot. A fájl rendszer felelősségi köre az, hogy az így elszórtan eltárolt adatokat rendszerezze, valamint számontartsa, azt, hogy melyik szektoron belüli adatfolyamhoz melyik fájl társítható, valamint a jelenlegi üres hely allokációja és vezérlése is a feladatköre.

2.2.3 Metaadat

Metaadatnak (angolul metadata-nak) nevezik azt az adatot, mely egy adatról szolgál információt, viszont ez nem ad vissza tartalmi adatot. ”Data about data” az informatika tudományágában elfogadott ez az elfogadott definíció. [22] Köznyelvben a definiálása problémás, más tudományágakban, más alkalmazás terület miatt más, pontosítottabb definíciót nyerhet az adott területágban használt felhasználási kör miatt.

Fájlrendszerekben a legfőbb szerepköre fájlhoz kötött adatok egységbe foglalása. Windows rendszerben az ilyen adatok tárolásához a fájlrendszer biztosítja az allokációt. Megfigyelhető, hogy amennyiben a felhasználó létrehoz egy üres fájlt, ilyenkor ez létrejön, allokálva van, és megmarad a rendszerben. Azonban a fájl mérete például üres txt esetén 0 Kb jelzéssel van ellátva. El van nevezve, és a rendszer allokálást köszönhetően, az elhelyezkedése is biztosított. Ezek az adatok ilyenkor fájlon kívül adatok az adott fájlról. [23] Ilyen metaadatokkal a mindennapok során is lehet találkozni, egyszerű példa erre egy film esetében a film kategóriája. A korábbi definíciót használva maga film videó az adat, ehhez tartozó tartalmon kívüli adat a kategória megjelölés. Az ilyen metaadatoknak hatalmas szerepe van SEO (Search Engine Optimization) keresőoptimalizálás során. Windows-os rendszerben a beépített fájl kezelő keresőjében felhasználhatóak az így eltárolt metaadatok a keresési eredmény szűkítésére.

Erre az alábbi 2.11-es ábrán látható menüpontok által meghatározott lehetőségek érhetőek el:

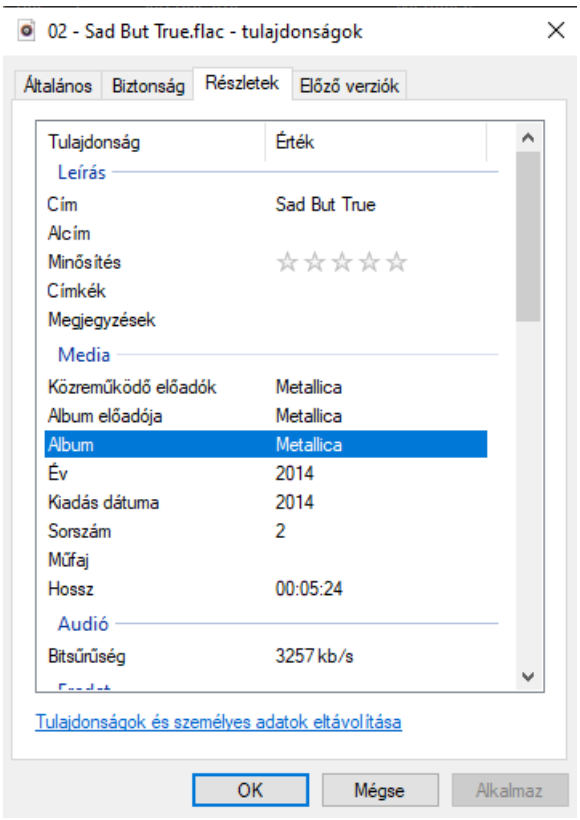


2.11. ábra Metaadat alapjáni szűrés Windows 10-en

A legtöbb tudományágban a metaadatok meghatározottak, egységesítettek. Ilyenkor az adat tartalmát és kontextusát körülíró adat szerepét tölti be. Az egységesítés miatt köztudatban ez mindenki számára érthető információt szolgáltat anélkül, hogy a teljes új információ feldolgozására szükség legyen. Ez az egységesítés a számítógépes rendszerekben, a tudományág által meghatározott, megszokott adathalmazt használja. Digitális audio lemezek esetében eltárolt adat az előadó, közreműködő, műfaj stb. Jelenleg ugyanez konvenció használt Windows rendszerekben is. A metaadat fájlon kívül, de fájlhoz tartozó, így a fájl áthelyezésével ez az adat nem veszik el. Zenéhez tartozó lemezborító esetén a zeneszám mobiltelefonra történő másolása után ezt az adatot a

lejátszó alkalmazás meg tudja jeleníteni fel tudja használni, még akkor is, ha ez a kép a telefonon soha nem volt elérhető. A fájl magában hordozta ezt az adatot.

Windows 10-es környezetben lehetőség van egy audio fájlnak, több ilyen adatát beállítani, módosítani ahogy ez a 2.12-es ábrán látható.



2.12. ábra Egy zene szám metaadatai.

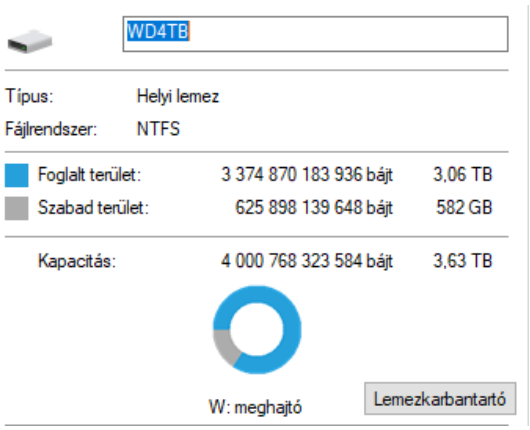
Minden fájl magában hordoz metaadatot, a méret is ilyen. Ez a tartalom szerint dinamikusan változhat. Ugyanez elmondható egy audio, videó hosszáról is. (Audio esetében az ID3 metaadat címkében eltárolt hossz érték pontatlannak nevezhető, zenehossz mérő algoritmusok során ezért ezt framekből szokás számolni)

A legtöbb fájlrendszerről elmondható, hogy nem csak a fájlokról tárol metaadatot hanem a nem használt területekről, valamint a hibás szektorokról is.

2.2.4 Adat vizualizációs módszerek

Kör és fánk diagrammok (Pie and donut charts)

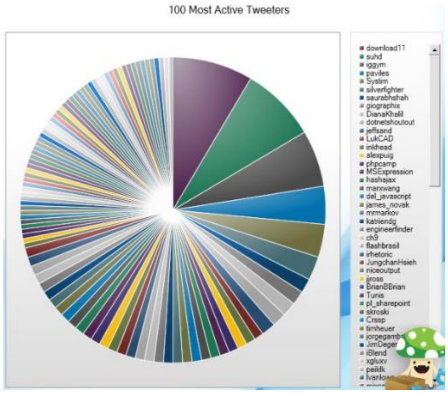
A kör és fánk diagram a nevét a vizualizáció nevéről kapta, az alakjára referál. Kifejezetten jól működik relatív kevés szegmensre történő bontás esetében. Windows 10-es operációs rendszerben alapértelmezetten a szabad területet fánk diagram módon jeleníti meg a rendszer, ez a 2.13 ábrán látható.



2.13. ábra Windows 10 tárhely foglaltság jelzése fánk diagrammal

Szintén leolvasható az ábráról a színmegjelölés, mely egy kör vagy fánk diagram esetében kifejezetten fontos. Ez definiálja azt, hogy egy adott szegmens mit jelent a diagramban. Korábbi Windows rendszerekben a jelölés mód kör, vagy henger megjelenítéssel működött, ilyenkor is a szabad tárterületet jelölte. Jelölést tekintve komplex diagramok estében lehet százalékosan is jelölni az adott szegmenseket.

A kördiagram relatív kevés osztás esetében tökéletesen jól működik, azonban ez a megjelenítésmód relatív nagyobb adathalmaz esetében kifejezetten túlkomplikálttá válhat. Ezt az alábbi 2.14-es ábra tökéletesen szemlélteti:



2.14. ábra Egy túlzsúfolt kördiagram

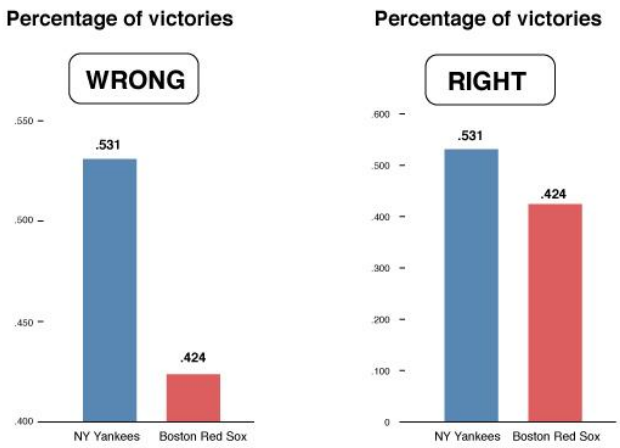
Az előbbi 2.14-es ábrán 100 különböző adathalmaz megjelenítése látható, ugyan leolvasható, de kijelenthető, hogy nagy adathalmazokra nem optimális megoldást nyújt.

A kördiagram egy továbbfejlesztett változata a sunburst diagram. Ez egy fa adatstruktúra lebontása kördiagramban. A kör közepe a gyökérpont (root), kifelé pedig az ágakon (node) keresztül lehet elérni a levél (leaf) elemekhez. Ez egy relatív nagyobb adathalmazt képes összefoglalni a hagyományos kördiagramhoz képest. Sok esetben azonban kifejezetten vékony ágakhoz vezethet, valamint egy nagy adathalmaz esetében még mindig kifejezetten nehéz a leolvasása.

Oszlop Diagram (Bar chart)

Az oszlop diagrammok különböző adathalmazokon definiált értékek megjelenítésére használt. Egy adott oszlop csúcsa definiálja az értéket, valamint tartalmazhat szín jelölést is. Az egyik tengelyen szokás megjelölni az érték lépcsőt, a másik tengelyen pedig az oszlop címeket. A tengelyek felcserélése megszokott, valamint lehetőség van ezeket csoportosítottan is megjeleníteni. Ilyen esetben egy oszlopon több érték is szerepelhet.

Kifejezetten rossz módszertan hatalmas érték különbségek, között több adathalmaz estében nem optimális. Az értéklépcső definiálása félrevezető tud lenni, ezt az alábbi 2.15-ös ábra tökéletesen ábrázolja.



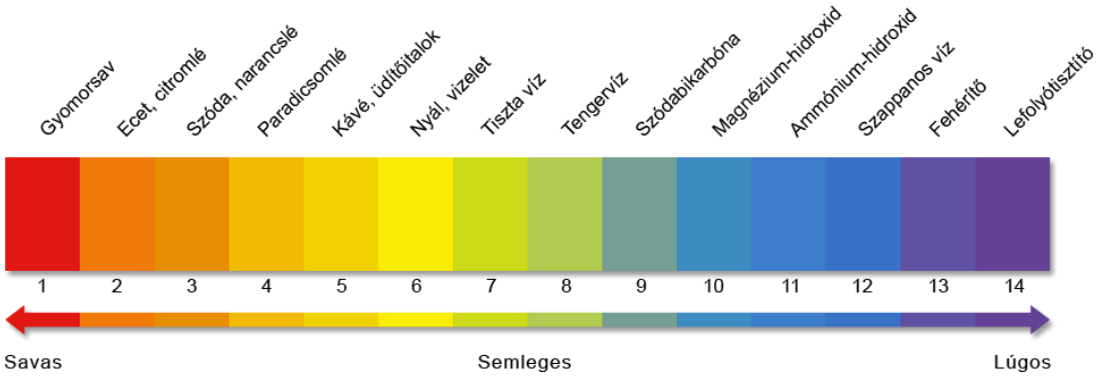
2.15. ábra Oszlop diagram félrevezető értéklépcsővel

Több adathalmaz esetében kifejezetten nehéz a leolvasása. Valamint az értéklépcső definiálása során a nagy kilengés mértéke szintén nehezíti a vizualizációs lehetőségeit.

Hőtérkép (Heatmap)

A hőtérképek kifejezetten kilengés mértékre használt adatstruktúra modellek. Az ilyen kilengés mérték meghatározásához, konvencionálisan egy vagy több szín használata szükséges, melyek árnyalatai határozzák meg a kilengés mértékét. Általánosságban egy világosabb árnyalat kisebb kilengést jelöl, ameddig a sötétebb pedig egy relatív nagyobb kilengést.

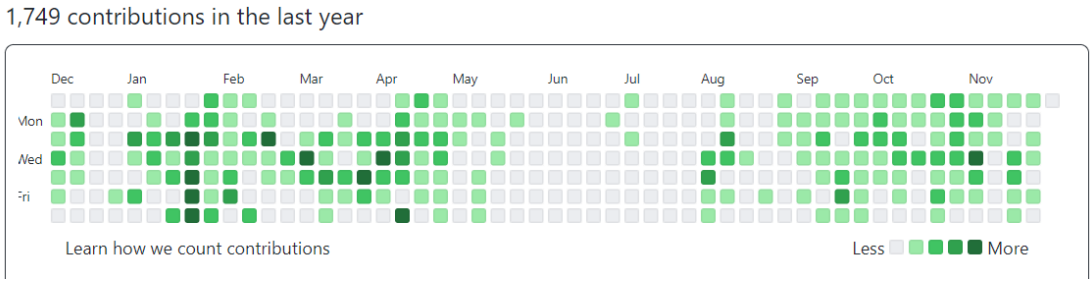
Tartalmazhat középértéket, ilyenkor a középértéktől kifelé más színek határozzák meg azt, hogy a kilengés mértéke a középértékhez viszonyítva mekkora. Kémia tudományában ilyen skála a pH érték mérésére használt indikátor papír melynek színbeosztása a 2.16-os ábrán látható.



2.16. ábra pH skála

Az 2.16-os ábrán jól látható, hogy a középérték a semleges, a két szélsőérték pedig a savas, valamint a lúgos.

Szintén hőtérkép használt idővel összevetett sűrűség esetében is. Ilyen például, a GitHub kontribúció összevetve a napok számával, a 2.17-es ábra jobb alsó sarkában látható ennek a sűrűségnek a jelölése.



2.17. ábra GitHub kontribúciók megjelenítésmódja

Ez a kép Sipos Miklós oktatóm GitHub profilján elérhető kontribúciók sűrűségi gyakoriságának hőtérképe.

A hőtérkép térképen történő megjelenítése nevéből eredően jelölhet hőfokot, ilyen esetben általánosságban piros/narancs sötét színnel szokás jelölni a magas hőfokot. Ennek halványabb árnyalatai pedig jelölik az ennél kisebbeket. Másik alternatíva hidegebb évszakok esetében a kék szín használat ilyenkor a skála az előbb említett ellentétje. Tehát sötétkék a hideg, a világosabb árnyalatok pedig az ennél magasabb hőfokot jelentik.

Mivel az adat mértéke/kilengése ebben a megjelenítési struktúrában árnyalatokon keresztül történik, ezért ez sokszor inkább becsült értékeket takar, mint pontos egységet.

Treemap

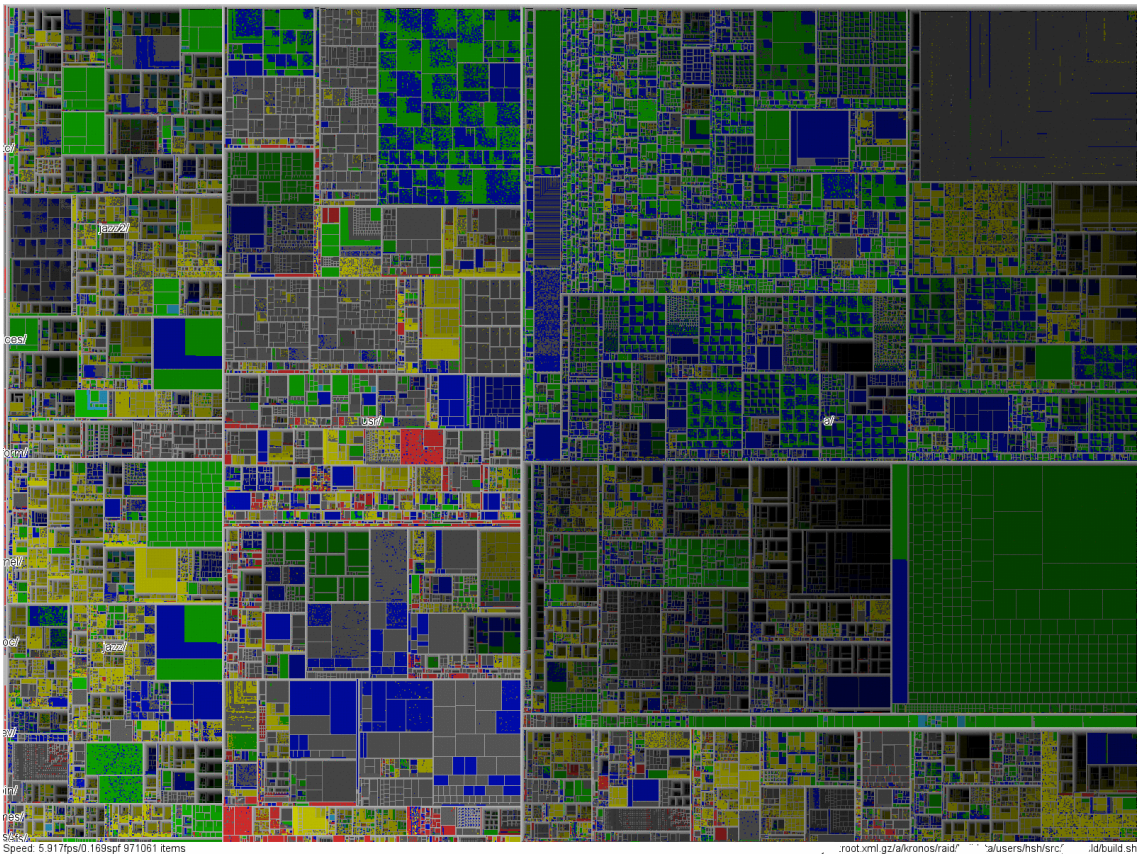
A treemap egy olyan vizualizációs leíró metodológia, melynek a során az adatok hierarchiája egymásba ágyazott téglalappal történik. Célja egy fa szerkezet vizualizálása minden egyes téglalap egy ága a fa szerkezetnek. Egy téglalapban beágyazott téglalap esetében a külső nagyobban jelölt téglalap az ős (parent) a benne szereplő téglalap vagy téglalapok pedig a gyermek/gyermekei. Treemap vizualizáció során különböző, kontrasztos színek használata ajánlott ezzel is segítve az átláthatóságot. Hatalmas előnye a treemap-nak, hogy a gyökér elem méretétől függ a benne ágyazott elemek mérete. Ez vizualizációs szoftverek során lehet kívülről érkező konstans érték. A gyermekelemek, az ebben szereplő téglalapok pedig ehhez a konstans értékhez igazodva meghatározhatók, az egymásba ágyazást tekintve. A treemap remekül használható mikor rengeteg adatot kell megjeleníteni a képernyőn.

A treemap megjelenítési mód feltalálása 90es évek első felében történt, Ben Shneiderman professzor nevéhez köthető. Célja a fa szerkezetű rendszerek korlátozott módú megjelenítése volt. Erre egy rekurzív algoritmust talált ki mely bármekkora mélységig kiterjeszthető.

„Tree structured node-link diagrams grew too large to be useful, so I explored ways to show a tree in a space-constrained layout. I rejected strategies that left blank spaces or those that dealt with only fixed levels or fixed branching factors. Showing file size by area coding seemed appealing, but various rectangular, triangular, and circular strategies all had problems. Then while puzzling about this in the faculty lounge, I had the Aha! experience of splitting the screen into rectangles in alternating horizontal and vertical directions as you traverse down the levels. This recursive algorithm seemed attractive, but it took me a few days to convince myself that it would always work and to write a six-line algorithm” [24]

Ben Shneiderman oldalán megtalálható több korábban említett szoftver, a KDirStat, WinDirstat valamint a Microsoft Excel is.

Treemap előnyét a legegyszerűbb egy vizualizációs példával bemutatni, Shneiderman ezt több előadása során különböző módszerekkel tette, az alábbi 2.18-as képen például egy treemap-ban egy millió egységnyi megjelenített elem látható. [25]



2.18. ábra Egy millió elem egy treemapban

Shneiderman és munkatársai rengeteg időt fektettek az interakció, (Például a nagyítás lehetősége) szűrés, valamint optimalizálási részével a treemapok-nak.

Ben Shneiderman ezzel a vizualizációs módszertannal beírta magát az informatika történelmébe. Nevéhez köthető még Human-Computer Interaction (HCI) Pioneers Project melynek a célja a legnagyobb technológiai forradalmárokra felhívni a figyelmet. Ehhez Shneiderman egy publikusan elérhető portréalbumot készített. Számomra talán legismertebb név Tim Berners-Lee, viszont sok más név mellett az itt szereplő összes név megérdemelné az említést a szakdolgozatban. [26]

2.2.5 Az asztali alkalmazás fejlesztésére használandó keretrendszer lehetőségek megtekintése

Electron

Ingyenes nyílt forráskódú cross-platform grafikus felhasználói felület készítő keretrendszer. Eredetileg a GitHub fejlesztői csoporthoz köthető azonban, jelenleg az OpenJs Foundation fejlesztése alatt áll. Node.js-en alapszik, npm csomagként telepíthető a következő paranccsal: *npm install electron*

Chromium alapú render motorral dolgozik. Rengeteg mai grafikus szoftver alapja név szerint néhány említés: [27]

- Visual Studio Code
- GitHub Destkop
- Figma
- Microsoft Teams

Flutter

Ingyenes nyílt forráskódú cross-platform grafikus felhasználói felület készítő szoftverfejlesztő eszköztár (SDK). A Google fejlesztette, valamint a Google Community felület biztosít dokumentációs, valamint közösségi támogatást is a rendszerhez. Dart alapú típusos nyelv. Kifejezetten jó Mobile First fejlesztési filozófiához. Egy kódbázison alapszik, támogatott rendszerek közé sorolható az Android, iOS, Linux, Mac, Windows, valamint a Google Fuchsia (Google saját fejlesztésű operációs rendszere). Az asztali alkalmazások terén a szoftver 2021 március 3-án jött ki a Flutter 2, ebben jelent meg az asztali alkalmazás fejlesztésének lehetősége korai hozzáférés formájában.

Electron vs. Flutter

Mindkét szoftver legnagyobb előnye az, hogy az a tudás melyet webfejlesztési környezetben épít egy fejlesztő az felhasználható. Ilyen például a HTML, valamint a JavaScript nyelv használata és a DOM manipuláció elve.

Electron segítségével relatív könnyű a desktop alkalmazás készítése szimpla projektek esetében. Azonban amint egy komplikáltabb alkalmazást kíván készíteni nagymértékben harmadik féltől származó JavaScript keretrendszerek függőségére lehet szükség. Ilyen esetben szükséges webpack-et használni ahhoz, hogy az Electron számára értelmezhető és használható legyen az ilyen forrásból származó kód. Az így készített webpack-ek komplexitása nagy, amennyiben egy webpack konfigurálása nem sikerül a teljes alkalmazás futtathatatlan állapotba kerül. Ez kifejezetten nagy problémát jelent régebbi, már elavult, nem támogatott szoftver csomag függőségek esetében.

Az Electron HTML-en alapszik ez a mai desktop alkalmazások számára sokszor már nem elegendő. A HTML webes környezetre készült, nem asztali alkalmazások leírására, ez lehetséges, viszont nem ez volt az intuitív szándék a nyelvnek. Jól megfigyelhető, hogy HTML oldalak statikusak, amennyiben bármiféle módosítás szükséges ez DOM manipuláció segítségével történik ehhez JavaScriptre van szükség. Teljesítmény szempontjából az Electron-ban készített alkalmazásokról elmondható, hogy native alkalmazásokkal összevetve inkonzisztens a teljesítményük nagyobb memóriaigénnyel. (native alkalmazásnak azt nevezik mikor egy adott operációs rendszerre készült egy alkalmazás) Az ilyen inkonzisztenciára jó példa a Microsoft Teams alkalmazás több ilyen teljesítmény probléma található ebben az alkalmazásban, a nagy memória fogyasztás is jelen van. Összevetve a Flutter-rel a flutter telefonon gépikóddá fordítja a kódot, amely teljesítményt tekintve nagyon jó. Ugyanez elmondható az asztali környezetről is. Az Electron JIT (Just In Time) elvet követi mely összevetve a Flutter-rel, ami AOT (Ahead Of Time) elvet követ. A flutter képes komplex grafikai műveleteket is elvégezni 60 FPS (képkocka per másodperc) tartással. A flutter nem Chromium alapú így a render folyamata az eszközön történik hasonlóan a natív alkalmazásokhoz. Memóriahasználatot tekintve kevesebb memóriát igényel, valamint reszponzívabb érzetet kelt a felhasználóban.

Mind a kettő alkalmazásban az üzletilogika a JavaScript alapon történik, azonban a Flutter Dart alapúsága miatt egy típusos nyelve mely segíti ezt a folyamatot.

Tárhely igényt tekintve az Electron 50%-al több tárhelyigénnyel rendelkezik egy szimpla „Hello World” alkalmazás esetében is. Ez annak tudható be, hogy egy Electron alkalmazás magában foglalja nagy részben a Chromium böngésző részeit, valamint a Node-ot. A Flutter jobb tárhely kezeléssel rendelkezik annak köszönhetően, hogy nem szerepel benne ekkora függőség.

A Flutter-ben egyszerű egy alkalmazást mobilos környezetre átvinni. Az Electron-ban ugyan megvan erre a lehetőség, azonban ez nem egy támogatott út. Ilyen esetben ez az átírási folyamat a fejlesztőre van hagyva, mely rengeteg stabilitási és funkcionalitási problémát jelenthet. Nagy hátránya a Flutter-nek azonban az, hogy az asztali környezetre való fejlesztés jelenleg béta verzióban van, viszont a jövőt tekintve nagy előrelépést jelenthet a korábban említett túlnyomó hozadékának köszönhetően. Rengeteg fejlesztő szerint előreláthatólag érdemes minél előbb elkezdeni a környezet tanulását. [28]

Ezen szakdolgozat célja a Flutter környezet határainak feszegetése, valamint a korábban említett WinDirStat szoftver modernizálása.

2.3 Irodalom kutatás összegzés

A korábban megvizsgált rendszerek közül kijelenthető, hogy a WinDirStat a legelterjedtebb. Ez részben betudható annak, hogy a Windows arányaiban a legelterjedtebb operációs rendszer platform a személyi számítógépek kategóriájában. Nyílt forráskód, valamint a hozzájáruló fejlesztők miatt a szoftver jól skálázott. Ennek köszönhető a rengeteg különböző, köztük magyar nyelv támogatás is. Azonban ez a szoftver vizualizációs eszközöket tekintve régóta nem részesült frissítésben. Kinézetét tekintve megállja a helyét, azonban összevetve a Mac OS X-es alternatívával, a DaisyDisk-kel, megfigyelhető, hogy a Daisy Disk egy modernebb UI trendet követ a WinDirStat-tal szemben. A WinDirStat-ban, valamint más említett szoftverekben megfigyelt fájl típus szerinti színezés az áttekinthetőséget növeli. Az ehhez tartozó jelölő tábla segíti a felhasználót ahhoz, hogy könnyebb összképet kapjon, milyen fájltípusok mekkora területet foglalnak egy adott mappaszerkezetben.

A hardverek fejlődése jelenlegi tárolókat tekintve, két fő kategóriába sorolható. Ez a két fő kategória a flash memória alapú tárolás, valamint a merevlemezek. Általánosságban a HDD olcsóbbak méretet tekintve, az SSD-k pedig ugyanabban az ár sávban átlagosan nagyobb írási és olvasási sebességet képesek biztosítani. (Azonban ilyenkor az SSD-k a mérete a HDD-khez képest kevesebb.)

Az SSD-re történő átállás miatt az analizáló szoftverek feldolgozása relatív gyorsabban tud végrehajtódni a HDD-kel összevetve. Ez a gyorsaság azonban nem minden felhasználó számára elérhető, azonban a jelenlegi tendenciát megvizsgálva kijelenthető, hogy a legtöbb mai, modern számítógépek esetében a tendencia jelenleg az SSD-k, flash alapú tárolók irányába hajlik a korábban említett előnyök miatt. Ugyanez elmondható külső tárolók esetében is, a korábban említett USB fejlődés nagy sávszélességgel rendelkező adatcsatornát képes biztosítani. Fontos, hogy adatfeldolgozást vizsgálva ez azt jelenti, hogy a fájl olvasása, mellett a fájlhoz tartozó metaadat olvasási sebessége is relatív gyorsabb.

Kijelenthető, hogy metaadatok közül a legfontosabb a fájl vagy a mappa mérete, mint tulajdonság. Azonban a többi metaadat lehetőséget ad a fájlok közötti szűrésre, lehetőséget ad keresőmotor implementációra.

A fájlok és mappák feldolgozása fájlrendszer szerinti kötöttséggel jár, mivel az adott fájlrendszer kötöttségéhez és konvencióihoz kell igazítani a szoftvert. Kijelenthető, hogy ez a legfőbb indok, amiért ilyen szoftverből cross-platform támogatással rendelkező alternatíva jelenleg nem létezik.

Megjelenítésnél megfigyelhető, hogy a hasonlórendszerekben vizsgált szoftverek túlnyomó része a Treemap módon építi fel a fa rendszert, a fájlméretek szerint. A Treemap lehetővé teszi egy fa struktúra megjelenítését egy adott méretű ablakban. Áttekintést tekintve könnyen, gyorsan átlátható. A Shneiderman által feltalált verziónak egy már tovább fejlesztett úgynevezett „kockásított" verziója a megszokott. (A „kockásítás" metodológiájával lehet elkerülni a hosszú vékony oszlopokat.) Ez rengeteg grafikusfejlesztői szoftver keretrendszerben külső telepíthető csomagon/csomagokon keresztül elérhető. Adott paraméterezések, keretek között ez egyszerűbb implementációt jelent.

3. Követelmény specifikáció

Az eddigiekben említett asztali alkalmazások alapján leszűrt adatokból kiindulva a főbb célok a következőek:

Operációs rendszert tekintve Windows 10 alapú rendszerhez kívánok elkészíteni egy lemez analizáló alkalmazást.

Funkciókat tekintve, a szoftver lehetőséget nyújt egy felhasználó által kiválasztott helyi lemezt, vagy mappa szerkezetet felmérni fájlméretek alapján. Az így összegyűjtött adatokat a felhasználó számára egy könnyen kezelhető és átlátható treemap formában kívánom megjeleníteni.

Amennyiben a treemap egy elemére kattint a felhasználó, az adott elemhez tartozó fájlon különböző lehetőségeket nyújt a szoftver:

- Megnyitni az adott fájlt vagy mappát Windows intézőben
- Törölni az adott fájlt vagy mappát lomtárba helyezéssel
- Törölni az adott fájlt vagy mappát véglegesen (Ilyenkor megerősítést kérve a felhasználótól)
- Fájl vagy mappa útvonalának másolása
- Fájl vagy mappa tulajdonságok lekérése

A szoftverben lehetőség van regisztrálni egy online API eléréshez. Az így regisztrált felhasználók számára bejelentkezést követően új menüpont jelenik meg. Ezen menüpontban a felhasználó lehetőséget kap analitikát naplózást megtekinteni. Az így létrehozott analitikát kizárólagosan csak a létrehozója éri el. A naplózott fájlok tárolása egy online adatbázisban történik.

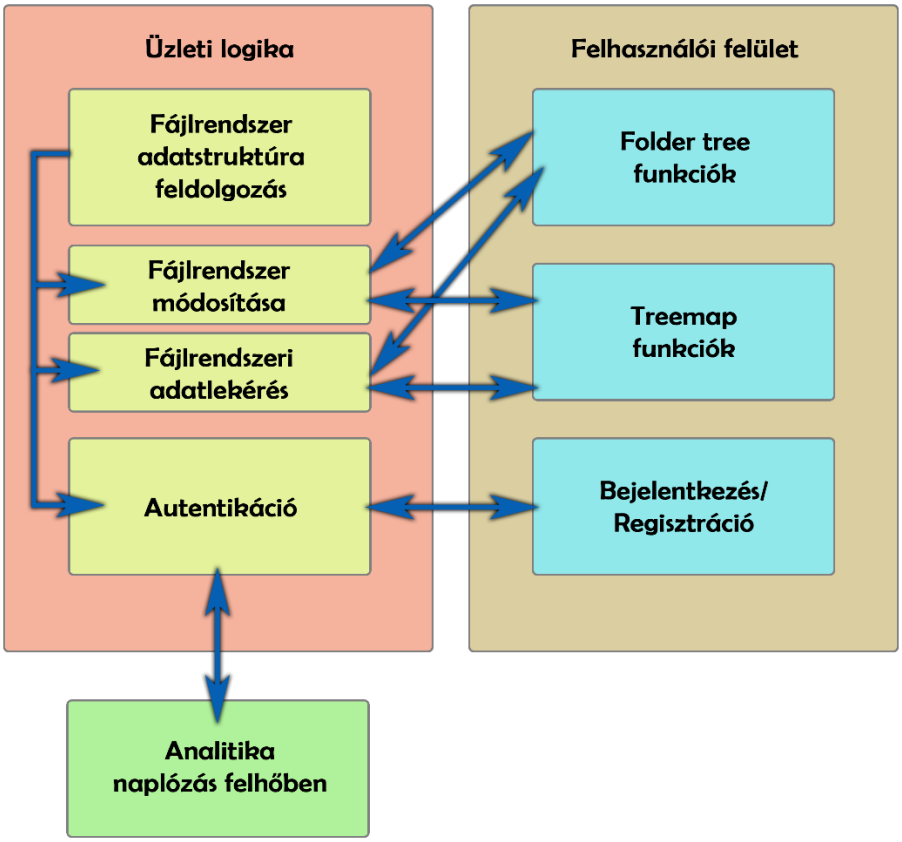
Megjelenítést tekintve, a szoftver célja a gyorsan, könnyen leolvasható információkat nyújtani. A treemap mellett egy hagyományos faszerkezetben úgynevezett folder tree-ben is megtekinthetőek és böngészhetőek a fájlok. Ebben a struktúrában kizárólag a szükséges információk szerepelnek fájlnév, méret, valamint a módosítás dátuma.

Dizájn tekintve, a mai szoftvertrendeket követve úgynevezett Flat UI design szerint kívánom készíteni. Ez egy minimalista megjelenítést foglal önmagában. Az okostelefonok alapértelmezett beállítás menüje is ezt a stílusirányzatot követi.

4. Tervezés

4.1 Rendszerterv

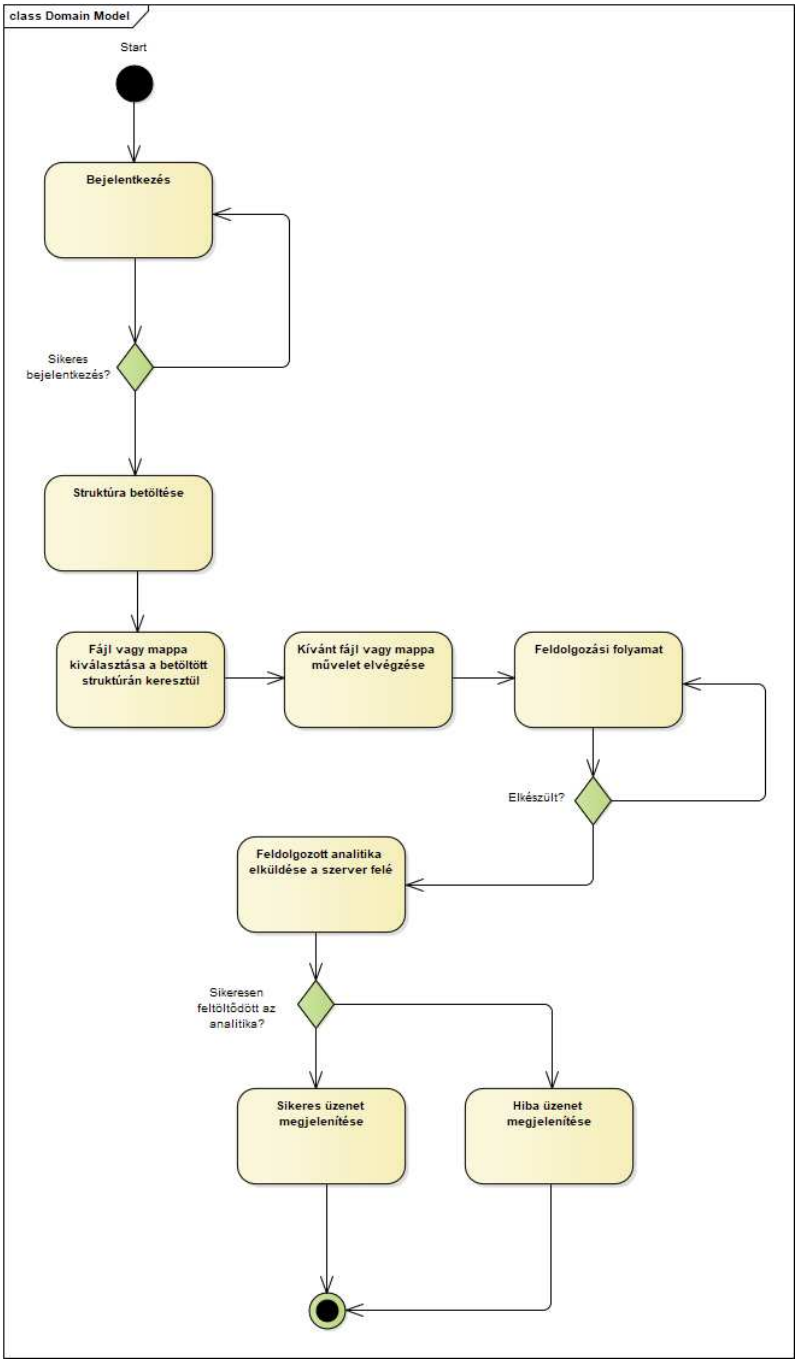
Készítendő rendszer megtervezéséhez a következő ábrát készítettem el:



4.1. ábra Az üzleti logika függőségi diagramja

A képről leolvasva látható, hogy a legnagyobb függőséget az fájlrendszeri adatstruktúra feldolgozása jelenti. Indítást követően a rendszer lehetőséget nyújt kiválasztani a vizsgálandó lemezt vagy mappát. Ezen feldolgozási folyamatnak az idejét a hardver erőforrás, valamint a mappa méret határozza meg. A felhasználó az így kapott adatokon keresztül képes fájlrendszeri módosításokat, valamint lekéréseket végezni. A treemap-ben megjelenített elemek főként vizuális, valamint böngészési lehetőséget nyújtanak. Mikor ezen kattint a felhasználó a folder tree felületen a kiválasztott fájlhoz vagy mappához ugrik a szoftver. Az így megjelenített elemen képes a szoftver fájlrendszeri módosításokat végezni.

Egy regisztrált felhasználó számára a szoftver lehetőséget nyújt naplózni, ilyenkor a végrehajtott fájlműveletről készül analitika, mely egy külső felhő alapú adatbázisban kerül eltárolásra. A kliens szoftver ezt, ezt egy online REST API híváson keresztül teszi. A felhasználói oldalról ez a folyamat a következőféleképpen ábrázolható:



4.2. ábra Analitika lépéseinek folyamatábrája

4.2 A rendszertől elvárt funkciók listája

- Meghajtó vagy mappa feldolgozása

  Az így kapott struktúrát megjeleníteni a felhasználó számára treemapben, valamint folder tree-ben

- Fájl vagy mappa adatok lekérése

  Tulajdonságok lekérése, a Windows alapértelmezett tulajdonságok felület megjelenítése.

- Útvonal másolása

  Lehetőséget adni a felhasználónak teljes útvonalat másolni.

- Regisztráció lehetősége a kliens alkalmazásban

  A kliens alkalmazáson belül lehetőség van új felhasználót létrehozni. (Internet kapcsolatot igényel a publikus API elérés miatt)

- Bejelentkezés lehetősége

- Analitika publikálás (Bejelentkezéshez kötött)

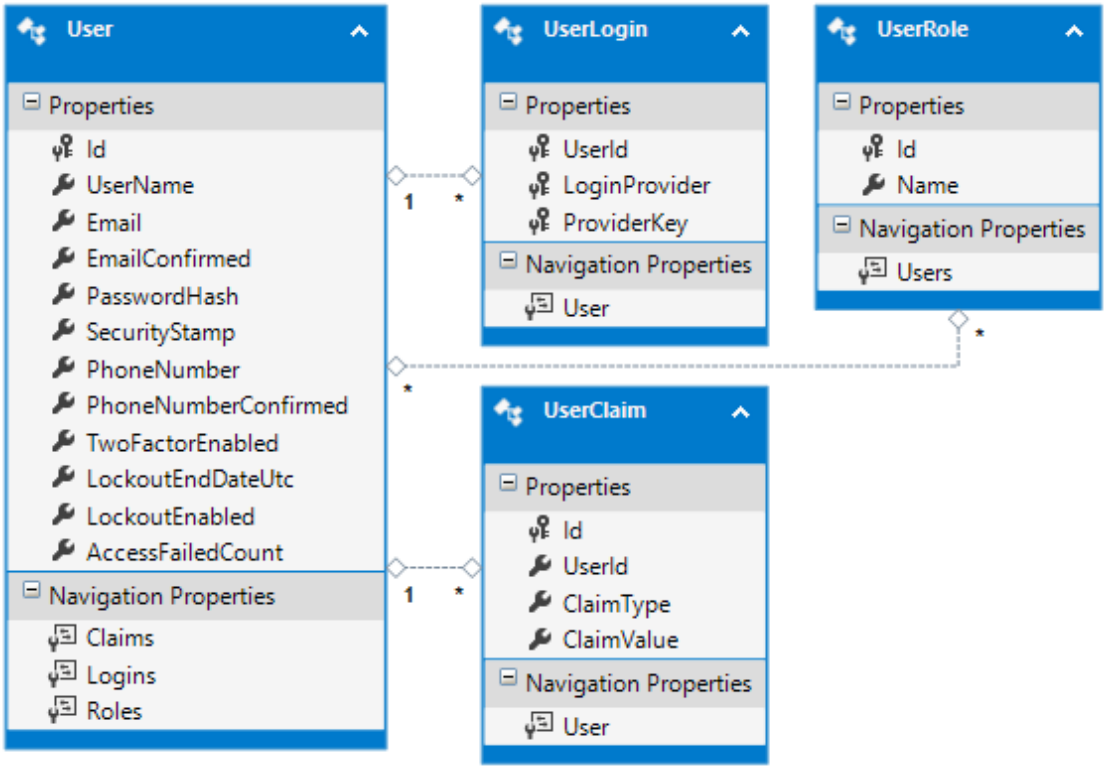
  Minden felhasználó kizárólagosan csak a saját publikált analitikáját/analitikáit képes elérni.

- Analitikát tároló védett API elérhetőség

Az API-kat ASP.NET alapon kívánom fejleszteni. Ehhez bejelentkezési és hitelesítési folyamathoz konvencionálisan az ASP.NET Identity felhasználó kezelő rendszer használata ajánlott. Rengeteg előnnyel jár, jól skálázható. Sok különböző akár külső bejelentkezési formát is képes támogatni, ami a mai modern web környezetben sokszor elvárt. Ilyen például a Facebook, Google alapú autentikáció.

Az Identity alapértelmezetten generált táblák alapján dolgozik, melyet a fejlesztő képes kiegészíteni és módosítani. Ez a rugalmasság a fejlesztés menetét könnyíti. Rettentően elterjedt, ezért a hivatalos dokumentációk mellett a harmadik féltől származó kódforrások is gyakoriak.

Az alapértelmezett táblák kapcsolatai az 4.3-as ábra alapján leolvashatóak. Jól mutatja, hogy a felhasználói csoport kezelését skálázhatóságát.



4.3. ábra Identity alapértelmezetten generált táblái

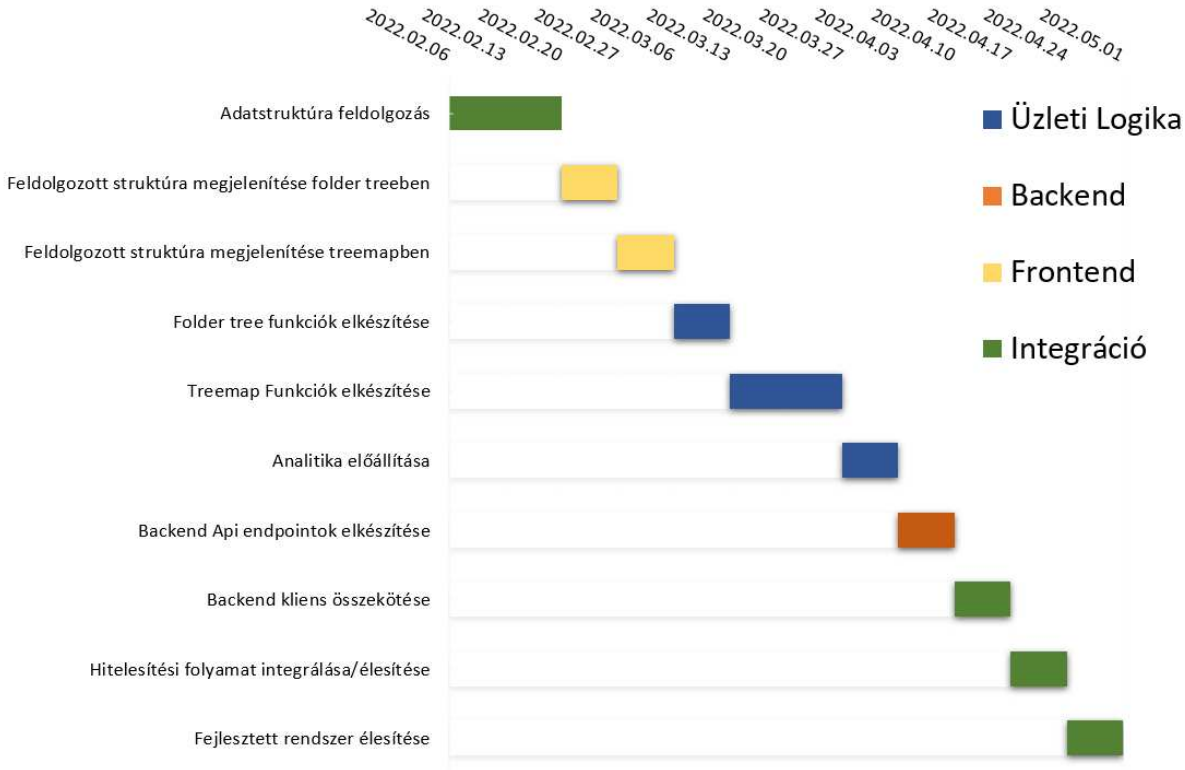
Ehhez viszonyítva a backend része az alkalmazásnak ASP.NET alapokon készül, C# nyelven.

4.3 Az így elkészített alkalmazás rész az API-k listája

- `api/Auth/Login`
  `HTTP POST` metódus, szerkezeti tartalma:
  A bejelentkezéshez szükséges email vagy felhasználó név, valamint jelszó.
- `api/Auth/Register`
  `HTTP POST` metódus, szerkezeti tartalma:
  A felhasználó személyes adatai, valamint a kívánt jelszava.
  (felhasználó név, vezetéknév, keresztnév, email, jelszó, megerősítő jelszó)
- `api/Operation/AddOperationAnalitic`
  `HTTP POST` metódus, szerkezeti tartalma:
  Egy kliens által végrehajtott művelet melyet feltölt a szerverre. Adatbázis oldalt ezt köti a feltöltést végző felhasználói fiókhoz.
- `api/Operation/GetOperationAnalitic`
  `HTTP GET` metódus, szerkezeti tartalma:
  Egy lista felhasználó által végrehajtott analitikákra. Egy lista elem szerkezete: Id (egyéni azonosító). Emellett még szerepel az elvégzési dátum és idő, valamint a felhasználó által végrehajtott művelet és fájl vagy mappa elérési útja.
  Visszaadja egy felhasználó által végrehajtott műveletek analitikát rendezve a feltöltési dátum és idő szerint növekvő sorrendben (időrendi sorrendben).
  A kérés szűrhető szöveges egyezés, valamint dátum szerint, a query stringen keresztül.
- `api/Operation/GetOperationAnaliticFile`
  `HTTP GET` metódus, szerkezeti tartalma:
  Azonos az előzőleg említett API leírásával, ezen keresztül azonban tabulátorral tagolt fájlként képes elérni a felhasználó az adott napra szűrt naplózási analitikát.

4.5 Fejlesztés tervezett menete

A fejlesztés tervezésének időbeosztáshoz elkészítettem az alábbi Gantt Diagramot.

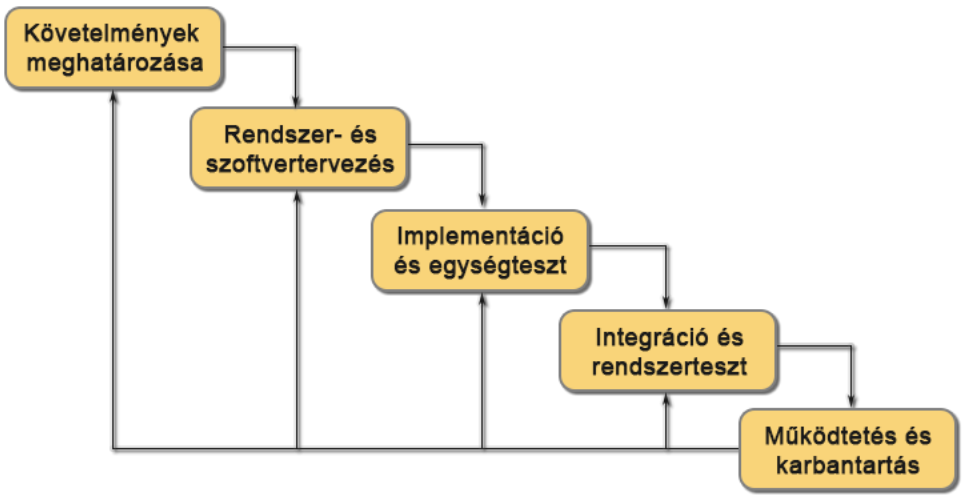


4.4. ábra Gantt diagram a tervezett haladásról

A képen jól látható az adott fejlesztési, valamint integrációs részekre tervezett idő. Ez az intervallum a fejlesztés során változhat. Az ilyen változásokról későbbiekben a fejlesztési naplóban írok. Nagyobb kihívást jelent az adatstruktúra feldolgozása és az ehhez tartozó logika. A fejlesztés során szintén ilyen nagyobb kihívás a Treemap-hez tartozó funkcionalitások elkészítése, valamint ezek szinkronizálása a folder tree-vel.

Haladási elvet tekintve vízesés modell szerint kívánok haladni.

Egyetemi tanulmányaim során szakdolgozat íráshoz kifejezetten optimális fejlesztési mintának találtam. Az alábbi képeken a folyamat, valamint ezen fejlesztési módszertan előnyei láthatóak. Ez az ábra Szabó Zsolt volt oktatóm diái által készült, valamint az összegzése is. [29]



4.5. ábra Vízesés modell elemei

Előnyök	Hátrányok
• Szekvenciális, könnyen megérthető • Követi a klasszikus mérnöki feladatsort • Jól elválasztott, logikus fázisok • Könnyen kezelhető • Ideális, ha mi vagyunk a megrendelő, a tervező, a fejlesztő, a tesztelő (pl. Szakdolgozat esetében)	• Túl merev, a követelmények változásra nem tud reagálni • Változás esetében szerződés módosítás, bontás állhat fenn • Megrendelő nincs bevonva a fejlesztés menetébe • Nincs korai prototípus lehetőség

Következtetésképpen a problémák túlnyomó része egy szakdolgozat esetében nem áll fenn, a követelmény változás kezelése azonban szükséges.

5. Fejlesztési napló

5.1 Fejlesztési napló bevezetése

Az alkalmazás készítése során nagy figyelmet fordítottam arra, hogy teljes mértékben cross-platform támogatást biztosítson az alkalmazás. A funkciók elsősorban Windows specifikusan készültek, azonban a tovább fejleszthetőség érdekében sok metodológia és csomag választás esetében nagymértékben befolyásolta a döntéshozatalt. A dokumentum írásakor teljes mértékben cross-platform támogatással rendelkező hasonló lemez analizációs alkalmazás nem létezik. Így kitűztem azt a célt, hogy az általam írt alkalmazás a flutter fejlesztési környezet által nyújtott lehetőségekkel képes legyen elérni egy teljesen cross-platform működést.

A kutatás során összegzett funkciók, alapján megvizsgáltam a rendelkezésre álló cross-platform lehetőséget nyújtó csomagokat. Proof-of-concept elven megvizsgáltam akár többet is egy adott funkció kívánt működésének eléréshez. A flutter fejlesztési közösség által elkészített csomagok kereshető listája megtalálható a *https://pub.dev/* weboldalon.

5.2 Adatstruktúra feldolgozása és megjelenítése

Első lépésként az alkalmazásban szükséges lehetőséget nyújtani a felhasználó számára kiválasztani mappát vagy meghajtót, melyet a kiválasztást követően az alkalmazás feldolgoz. Erre a *file_picker* csomag tökéletes megoldást kínál, a flutter által támogatott platformokat tekintve teljes mértékben cross-platform megoldás.

A kiválasztott útvonal feldolgozásához fa adatstruktúrát használtam. A beépített *dart.io* csomag segítségével a fájlhoz vagy mappához tartozó metaadat osztályszintű kiszervezése így valósult meg.

5.2.1 FolderTree struktúra kialakítása

A FolderTree struktúra kialakításához a *list_treeview* nevű csomag került felhasználásra, ez biztosítja a kialakított fa struktúra megjelenítését olyan módon, melyben a mappák nyithatóak és csukhatóak. A megjelenített mappák és fájlok neve mellett az alkalmazás megjeleníti az állomány által foglalt területet. Az alkalmazás a fájl vagy mappa méret byte értékét tárolja. Ebből az értékből a megjelenített szöveg Windows-hoz hasonlóan két tizedesjegy kerekítéssel a megfelelő információegységet tartalmazza. Ez a kerekítés byte egységtől yottabyte egységig kezelt. A jövőbiztosság érdekében a byte érték tárolására az alkalmazás 64 bites integer-t használ, ennek köszönhetően a legnagyobb beolvasható fájl/mappa mérete 8192 petabyte lehet.

A FolderTree struktúra kialakítása során több fejlesztési nehézség jelentkezett. Az állomány által foglalt területek meghatározása fájlokra lekérhető azonban magának a mappáknak nem minden fájlrendszerben van mérete. Az alkalmazás a mappa méretet így a mappában szereplő fájlok és mappák mérete alapján határozza meg.

Másik probléma, ami jelentkezett még ezzel a csomaggal az az, hogy nem pozícionált listázót használ. Az alkalmazásnak kritériuma, hogy adott pozíció kereshetőséget biztosítsa a TreeMap struktúra böngészhetőségével. Ezen probléma javításához a csomag felülbírálása (override-olása) lett a megoldás.

Ehhez a csomag lokális implementációja szükséges, ezzel elkerülve azt, hogy egy frissítés szabotálja a kívánt működését. A csomagban szereplő alapértelmezett listázó ki lett cserélve egy másik csomag által kínált pozícionált listává. Ez a *scrollable_postioned_list* csomag.

5.2.2 TreeMap struktúra kialakítása

A TeeMap struktúra vizualizálja a beolvasott fájlok méretét, valamint biztosítja a fájlok egy kattintásra történő böngészhetőségét. Az implementálás során két csomag került megvizsgálásra. Az egyik a *syncfusion_flutter_treemap* volt. Ez a csomag képes komplex treemap struktúra kialakítására és testreszabására. Probléma azonban, hogy a hivatalos oldaluk megtekintésekor a használata csak trial verzióban érhető el, amely egy hónap után lejár. A másik megvizsgált, majd implementált csomag *treemap* volt. Ez a csomag nem kínál akkora testreszabhatóságot azonban, a projekt céljainak megfelel.

Böngészi lehetőség érdekében elkészült az a funkció, melynek során adott TreeMap mező kiválasztását követően a FolderTree struktúrában az ennek megfelelő fájl kijelölésre kerül. Ehhez az alkalmazás a fájl teljes útvonalát használja kulcsként. Fontos megemlíteni azonban azt, hogy a két struktúra bemeneti adathalmaza egyezik. Felépítést tekintve a TreeMap komponens leszármazottja a FolderTree-nek. Változás esetében a két komponens kommunikációja így biztosított. (Például fájl vagy mappa törlés esetében a FolderTree struktúra módosítja a bemeneti fa struktúrát. Ezt követően erről a változásról értesíti a TreeMap struktúrát.)

5.2.3 Fájltípusok meghatározása

A típus lekéréshez több metodológia megfontolásra került.

Első gondolatként MIME típus alapján történő típus azonosítás volt tervezett, ez azonban több komplex problémát okozott. A módszer vizsgálásához a *mime* nevű csomag került felhasználásra. A MIME típus elsősorban internetes üzenetek küldéséhez használt típus meghatározáshoz készült. Kijelenthető, hogy egy átlagos mindennapi használatban lévő Windows specifikus számítógépen nem csak MIME típus által ismert fájlok találhatóak. Szintén szükségesnek és hasznosnak találtam azt, hogy az alkalmazást használó felhasználó a fájltípusokról kapjon egy rövid leírást. Ez a leírás összegzi a fájltípus szerepét a számítógépén. Hosszas kutatást követően az alábbi megoldás lett implementálva.

A fájltípusok a kiterjesztésből lettek meghatározva. A fájltípus kiterjesztések gyűjteményéről és leírásáról találtam egy json filet a github gists platformon. [30] Ez rengeteg fájltípust tartalmaz. Szintén tartalmaz egy leírás listát, melyben egy adott fájltípus különböző használati lehetőségeit felsorolja. Ez lehet több is, mivel egy a kiterjesztés jelölését több különböző szoftver és/vagy operációs rendszer másra használhatja. Ehhez készült egy konzolos alkalmazás, mely ezt a json struktúrát átalakítja, és illeszti az alkalmazás által megszabott kritériumokhoz. Ez a konzolos alkalmazás bevezette a színezést. mely egy újabb felépítési és fejlesztési döntés elé hozott.

Az utóbbi fájltípus meghatározás jelen állapotban összesen 3170 különböző fájltípust képes felismerni. Az alkalmazás továbbfejlesztése során, valamint a jövőbiztosság érdekében ez könnyedén kiegészíthető.

5.2.4 Fájltípusok színezésének módszere

Jelen megoldás szerint a fájltípusok jelöléséhez a TreeMap struktúra egy adott fájl típushoz egy előre megadott színt társít.

A fájltípusokhoz társított színek során, fontos, hogy megkülönböztethető színek legyenek hozzárendelve a fájltípusokhoz. Ehhez felhasználtam egy olyan algoritmust. mely képes N darab különböző színt generálni. Az algoritmus ezt nem véletlenszerű érték alapján teszi, hanem az N tartományhoz viszonyított index alapján a lehető leginkább eltérő szín értéket rendeli.

Ezen a ponton a fejlesztésben két oldalról közelíthettem, meg a kialakított színkiosztást.

Az első lehetőség az, hogy az alkalmazás a feldolgozáskor határozza meg azt, hogy éppen hányadik különböző fájltípus színt generálja. Ez előny mivel, a lehető leginkább eltérő színek jelennek meg minden feldolgozást követően. Viszont személyes véleményem

szerint ez zavaró, és összetévesztő lehet a felhasználóval szemben, így végül másik módszert választottam.

Az alkalmazásban jelenleg a színkiosztás előre generált. Az előbb említett algoritmus szerint ez olyan módon történik, hogy ezt már a fájltípus json fájlba legenerálja. Véleményem szerint ez olyan szempontból jobb megoldás, hogy a felhasználó képes memorizálni azt, hogy egy adott szín melyik fájl típushoz tartozik, így a gyakran használt fájlokhoz nem szükséges használnia a színjelölési listát, mely ezt a konzisztens információt prezentálja a számára.

5.2.5 Fájltípus súgó

Az előző pontban említett színezés jelöléséhez készült egy fájltípus súgó lista.

Ez a lista négy fő részből áll:

- Fájltípus színjelölése
- Kiterjesztés elnevezése
- Összesített fájlméret az adott fájltípusra vonatkozóan
- Rövid leírás az adott fájltípus használatikörével kapcsolatban

A lista alapértelmezetten nem jelenik meg, ezzel maximalizálva a megjeleníthető adatmennyiséget a TreeMap valamint a FolderTree struktúrákban. A lista egy felső dedikált menü ponton keresztül átfedéses alapon ideiglenesen megjeleníthető. Megjelenítési sorrendet tekintve fájltípushoz tartozó összesített fájlméret szerint csökkenő sorrendben áttekinthető. Adott fájltípus itt kiválasztható. Kiválasztását követően, az alkalmazás tájékoztatja a felhasználót a típus használatikörivel kapcsolatban.

Előfordulhat azonban az, hogy az alkalmazás nem ismer fel egy adott fájltípust. Ilyenkor ezt ismeretlen (Unknown) jelöléssel látja el.

5.3 API kommunikáció

Az alkalmazás menü nézetén keresztül lehetőségünk van felhasználót létrehozni regisztrációval, vagy bejelentkezni már létező felhasználóval. A regisztrációt követően az alkalmazás automatikusan belépteti a felhasználót. A kialakított felhasználói rendszerben nem szükséges megerősítő email fogadása és aktiválása, a felhasználó regisztrációt követően rendelkezik a szükséges jogosultságokkal.

Az alkalmazásban egy bejelentkezett felhasználó számára a következő műveletekről napló készül:

- Útvonal megnyitása
- Tulajdonságok megnyitása
- Végleges törlés
- Lomtárba helyezés
- Útvonal másolása

A naplózási analitika a felhasználóhoz van kötve, minden felhasználó csak a saját maga által elvégzett műveleteket képes áttekinteni. Az áttekintéshez egy külön felhasználói felület készült, melyben az alkalmazás mindig az aktuális nap naplózását jeleníti meg alapértelmezetten. Amennyiben korábbi nap naplózását kívánja áttekinteni a felhasználó, erre is van lehetősége.

A felületen egy konstans megjelenő sáv tartalmazza az alábbi funkcionalitásokat:

- Szöveges szűrési sáv
- Dátum kiválasztó
- Szűrés elvégzése (szöveg beképelését követően)
- Naplózási analitika letöltése (A kitöltött szűrőfeltételek szerint)

Lehetőség van beállított szűrőfeltételek szerint szöveges állományként letölteni egy adott napra vonatkozó naplózást. Ez alapértelmezetten tabulátorral van tagolva, így könnyedén felhasználható más alkalmazásokban is, mint például MS Excel-ben.

5.4 Fájlműveletek biztosítása, cross-platform módszerrel

5.4.1 Flutter által biztosított Method Channel fejlesztésimód

Az alkalmazás fejlesztése során nagy erőfeszítésbe került a cross-platform irány felé nyitottnak maradni, mind fejlesztésben, mind a továbbfejlesztési lehetőségek biztosításában. A választott szoftvercsomag, a Flutter nagy mértkében támogatja az ilyen módon történő fejlesztés lehetőségét.

Kijelenthető, hogy a különböző fájlműveletek elvégzésnek menete a futtatott platformtól, az Operációs rendszertől függ. Például egy fájl lomtárba helyezésének módját/lehetőségét ez az adott platform határozza meg. Ilyen művelet kezeléséhez szükséges platform specifikus kód implementáció.

A projekt készítése során az úgynevezett Method Channel módszer került alkalmazásra. A szoftvercsomag egy ilyen módon megírt csatornán keresztül biztosítja azt, hogy egy adott művelet egy adott platform specifikus kódrészletet hívjon meg. Ez az alkalmazás készítésekor elsősorban Windows specifikusan készült. Windows esetében az alapértelmezett platform specifikus nyelv a C++.

Egy ilyen műveletre példa az alkalmazásban szereplő „properties” megnyitása Windows platformon. Ennek során az alkalmazás UI oldalon kivált egy Method Channel műveletet, és átadja az adott fájl vagy mappa útvonalát. A Flutter ekkor a host, jelen példában Windows szerinti C++ aszinkron kódrészletet futtatja. Ez a kódrészlet a Windows API-n keresztül megnyitja a kapott útvonalhoz tartozó *Tulajdonságok* (*Properties*) ablakot. Ezt követően értesíti a UI-t a művelet sikerességéről.

Amennyiben egy adott platformon nem implementált az adott Method Channel művelet, az alkalmazás futás idejű kivételt vált ki.

5.4.2 Windows specifikus Method Channel

A Windows specifikus Method Channel implementációjához a Proof of concept fejlesztési elv lett használva. Ennek során elkészült az *operands.cpp* C++ fájl, melyben a Windows Api hívások manuális tesztelésre kerültek. Nagy problémát jelentett az ékezetes karakterek kezelése. Ehhez saját konverter metódus került bevezetésre, melyben UTF-8-as kiterjesztett karakterláncot készített a bemeneti útvonalból. Az így átalakított kiterjesztett karakterlánc lett felhasználva a *windows.h* C++ csomag hivatalos dokumentációja alapján. Ennek megfelelő metódus szignatúrához illesztettem a kódot. Így valósult meg az, hogy a különböző megengedett speciális karakterek, mint például az ékezetes karakterek is feldolgozásra kerüljenek.

A Proof of concept implementálást és a dokumentációt olvasva sok hasznos információt fedeztem fel. Ebből néhány példa:

- A hibaüzeneteket sok esetben automatikusan kezeli, például amennyiben egy művelet során az adott fájl nem létezik, erről a felhasználó minden esetben értesül.
- Végleges törlés, valamint lomtárba helyezéskor automatikusan megjeleníti a beépített folyamatjelző sávot.
- Amennyiben a lomtár megengedett méretét meghaladja egy törlendő fájl vagy mappa, erről az értesítés automatikusan megtörténik.

A *windows.h* csomag leírása során rengeteget ír a különböző eljárások kompatibilitásával kapcsolatban. Összességében elmondható az, hogy Windows XP-s operációs rendszerektől felfele teljes mértékben kompatibilis a műveletek felhasználása. Egy kivételről azonban rengeteget ír, ez a Windows Vista rendszeren történő felhasználás. Ez a legtöbb metódusban nem támogatott, ilyen operációs rendszerre való fejlesztés esetében ehhez az operációs rendszerhez készült dedikált hívások szükségesek.

6. Tesztelés

A fejlesztés közben a Windows specifikus műveletek Proof of Concept alapú tesztelésen estek keresztül, az így készített kód a forrásállomány között szerepel. Ez az asztali alkalmazás üzleti logikájának fontos része. A naplózási analitika Unit tesztelésre került, a különböző Unit- és manuális tesztelések és eredményeik a következő táblázatban kerülnek összefoglalásra:

#	Követelmény	Tesztleírás	Mérés és cél	Eredmény
1.	Sikeres regisztráció a megfelelő adatok kitöltésével.	Új felhasználó regisztrálása a rendszerbe, megfelelő adatok megadásával.	Megfelelő még nem létező felhasználói adatok kitöltését követően adunk vissza hitelesítési tokent.	
2.	Sikeres bejelentkezési azonosítás email cím alapján.	Létező felhasználó képes belépni a rendszerbe e-mail cím segítségével.	Mock-olt már létező felhasználók belépési adatainak megadását követően adunk vissza hitelesítési tokent e-mail címek alapján.	
3.	Sikeres bejelentkezési azonosítás felhasználónév alapján.	Létező felhasználó képes belépni a rendszerbe felhasználó név segítségével.	Mock-olt már létező felhasználók belépési adatainak megadását követően adunk vissza hitelesítési tokent felhasználó nevek alapján.	

4.	Sikertelen regisztráció, létező e-mail cím felhasználása a regisztráció során.	Egy már regisztrált e-mail címmel történő regisztráció nem lehetséges.	Mock-olt már létező felhasználói e-mail címek megadását követően a regisztráció meghiúsul, nem jön létre új felhasználó.	
5.	Sikertelen regisztráció, létező felhasználó név felhasználása a regisztráció során.	Egy már regisztrált felhasználó névvel történő regisztráció nem lehetséges.	Mock-olt már létező felhasználók felhasználó nevének megadását követően a regisztráció meghiúsul, nem jön létre új felhasználó.	
6.	Sikertelen regisztráció, a beadott jelszavak nem egyeznek meg.	A regisztrációs felület kitöltése során a jelszó és megerősítési jelszó nem egyezik meg, a regisztráció nem lehetséges.	Amennyiben a beírt jelszó és a megerősítési jelszó nem egyezik, a regisztráció meghiúsul, nem jön létre új felhasználó.	
7.	Sikertelen belépés, a megadott felhasználó név nem létezik.	A belépés során megadott felhasználó névhez nem tartozik, regisztrált felhasználó.	Amennyiben nem létező felhasználó név kerül megadásra a belépés során, a belépés meghiúsul, nem adunk vissza hitelesítési tokent.	
8.	Sikertelen belépés, a megadott e-mail cím nem létezik.	A belépés során megadott e-mail címhez nem tartozik, regisztrált felhasználó.	Amennyiben nem létező e-mail cím kerül megadásra a belépés során, a belépés meghiúsul, nem adunk vissza hitelesítési tokent.	

9.	Sikeres naplózási analitika hozzáadása és hozzáfűzése felhasználóhoz.	Új naplózási analitika művelet mentése és felhasználóhoz kötése a rendszerbe, megfelelő adatok megadásával.	Egy Mock-olt felhasználóhoz hozzáfűzzünk egy új naplózási analitikát, ellenőrizzük, hogy a hozzáadás meghívódik.	
10.	Sikertelen naplózási analitika hozzáadása, nem létező felhasználóra.	Új naplózási analitika művelet mentése és felhasználóhoz kötése, azonban a felhasználó nem létezik.	Nem létezik olyan Mock-olt felhasználó, akihez az analitkát lehetne kötni, így a művelet meghiúsul, nem jön létre új analitika.	
11.	Sikeres naplózási analitika lekérése, szöveges szűrés nélkül.	Naplózási analitika lekérése dátum és felhasználó specifikusan.	Létező Mock-olt felhasználóhoz kötött naplózási analitikák lekérése, és szűrése dátum szerint.	
12.	Sikeres naplózási analitika lekérése, szöveges szűrés nélkül.	Naplózási analitika lekérése dátum, szöveges szűrés és felhasználó specifikusan.	Létező Mock-olt felhasználóhoz kötött naplózási analitikák lekérése, és szűrése dátum, valamint szöveges egyezés szerint.	
13.	Útvonal megnyitása Windows Specifikus fájl böngészővel	Útvonal szerint az alkalmazás képes megnyitni a fájl böngészőt az adott helyen, ékezetes karakterekkel is.	Jogosultságnak megfelelően minden létező fájl vagy mappa útvonalra sikeresen megnyílik a fájl böngésző.	

19.	Fájl vagy mappa tulajdonságainak megjelenítése Windows beépített funkciójával.	Az alkalmazás képes megnyitni mappára vagy fájlra a „Tulajdonságok” fület.	Jogosultságnak megfelelően minden létező fájl vagy mappa útvonalra sikeresen megnyílik a fül.	
20.	Fájl vagy mappa lomtárba helyezése Windows beépített funkciójával.	Az alkalmazás képes lomtárba helyezni egy adott fájlt vagy mappát.	Jogosultságnak és lomtár méretnek megfelelően minden létező fájl vagy mappa sikeresen áthelyezhető a lomtárba.	
21.	Fájl vagy mappa útvonalának másolása.	Az alkalmazás képes egy adott fájl vagy mappa útvonalának másolására.	A másolás minden esetben megfelelően működik.	
22.	Fájl vagy mappa végleges törlése Windows beépített funkciójával.	Az alkalmazás képes véglegesen egy adott fájlt vagy mappát.	Jogosultságnak megfelelően minden létező fájl vagy mappa sikeresen törölhető véglegesen.	

6.1. táblázat Tesztelési napló

7. Továbbfejlesztési lehetőségek

Az alkalmazás fejlesztése során felhasznált csomagok és módszereknek köszönhetően, a szoftver továbbfejleszthető a Flutter szoftvercsomag által támogatott különböző platformokra. Kijelenthető, hogy a különböző operációs rendszerek, különböző fájlkezelési műveletek meghívását igénylik. Ebből adódóan minden platformhoz szükséges elkészíteni a korábban említett platform specifikus Method Channel-t. Ezek lefejlesztése és működésének tesztelése különböző virtualizált vagy valós eszköz hozzáférését igénylik.

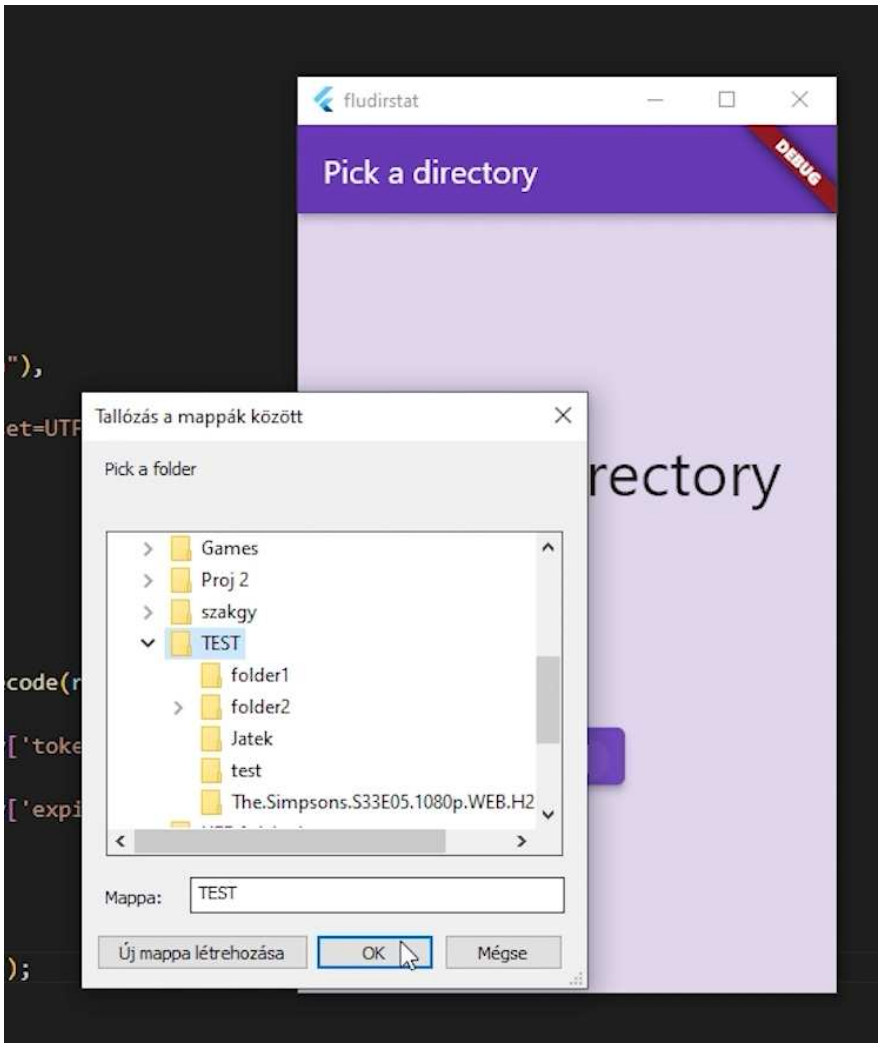
A szoftver grafikai interfész szempontjából, amennyiben az alkalmazás továbbfejlesztése során Android, valamint IOS rendszerek támogatás bevezetése igényelt, fontos, hogy ennek megfelelően a kijelző mérethez és az emberi ujj mérethez viszonyított reszponzivitásra fokozott figyelem kerüljön. Az alkalmazás jelen állapotban, a legtöbb felületen erre fokozottan ügyel, azonban véleményem szerint a Treemap, valamint a folder tree struktúrát érdemes átalakítani olyan módon, melyben a felhasználó a lehető legnagyobb elérhető képernyőmérethez viszonyított megjelenítést kínálja, a jelenlegi kettéosztott elhelyezési struktúrát leváltva.

Szintén továbbfejlesztési lehetőségként a Windows 10-be beépített Defender általi fájl vizsgálathoz hasonló metodológia cross-platform bevezetésének ötlete merült fel. Az alkalmazás backend oldalán elkészült egy olyan endpoint, amely a VirusTotal API segítségével képes különböző vírusírtók szerint (köztük a Windows Defender által használt adatbázis szerint is) képes áttekinteni, hogy az adott fájl veszélyt jelent-e a felhasználó számítógépére. Ezt a funkciót a regisztrált felhasználók számára elérhető funkcióként implementálható. Mivel ez API hívás melynek során az adott fájlt szükséges elküldeni a backend felé, így nem platform specifikus művelet. Az így kapott válasz alapján a felhasználó számára megjeleníthető, hogy a felsorolt vírusírtó adatbázisok szerint melyek ítélik meg vírusosnak az adott fájlt, milyen indokkal. Szintén megjeleníthető az, hogy összességében darabszám szerint mennyi szerint veszélyforrás az adott fájl. Ennek a funkciónak a teljes mértékű lefejlesztésére sajnos nem került idő.

8. Képek az elkészült rendszerről

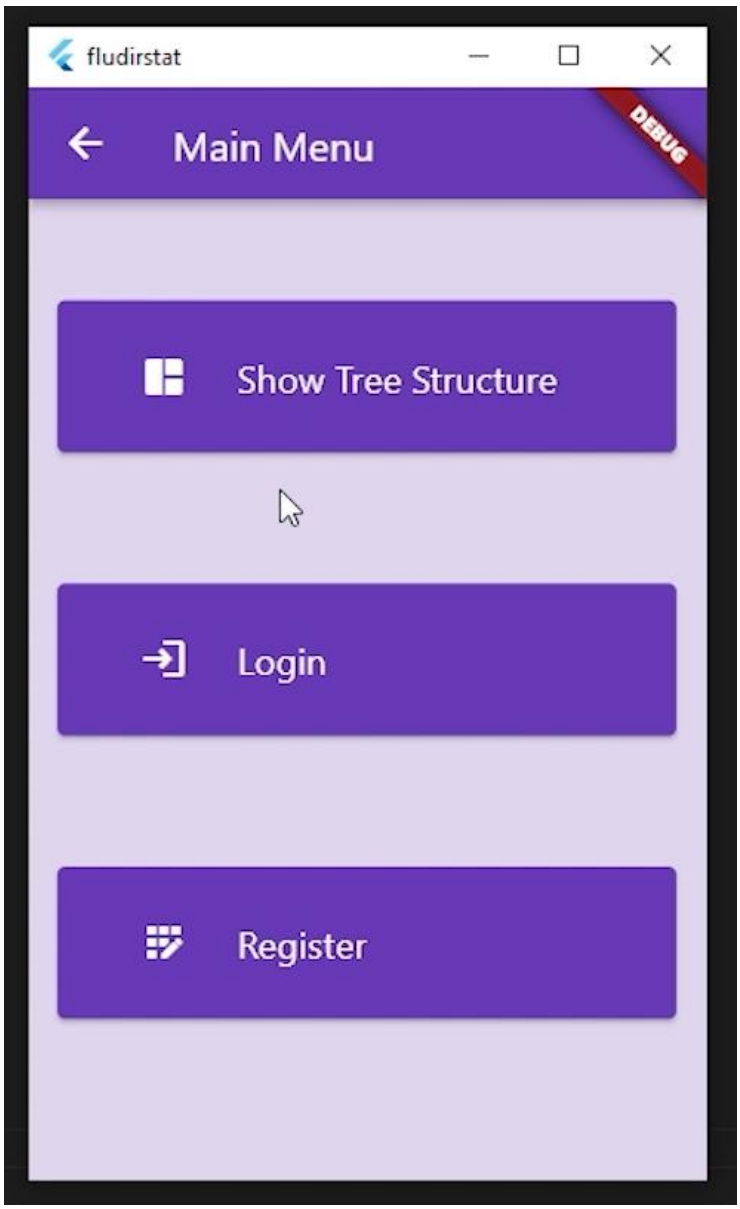
A futtatott alkalmazásból a következő képeket készítettem. (A jobb felül látható „Debug” jelölés azért látható mivel az alkalmazás fejlesztői környezetben fut)

Az alábbi ábrán a fő oldalon keresztül történő útvonal választás látható.



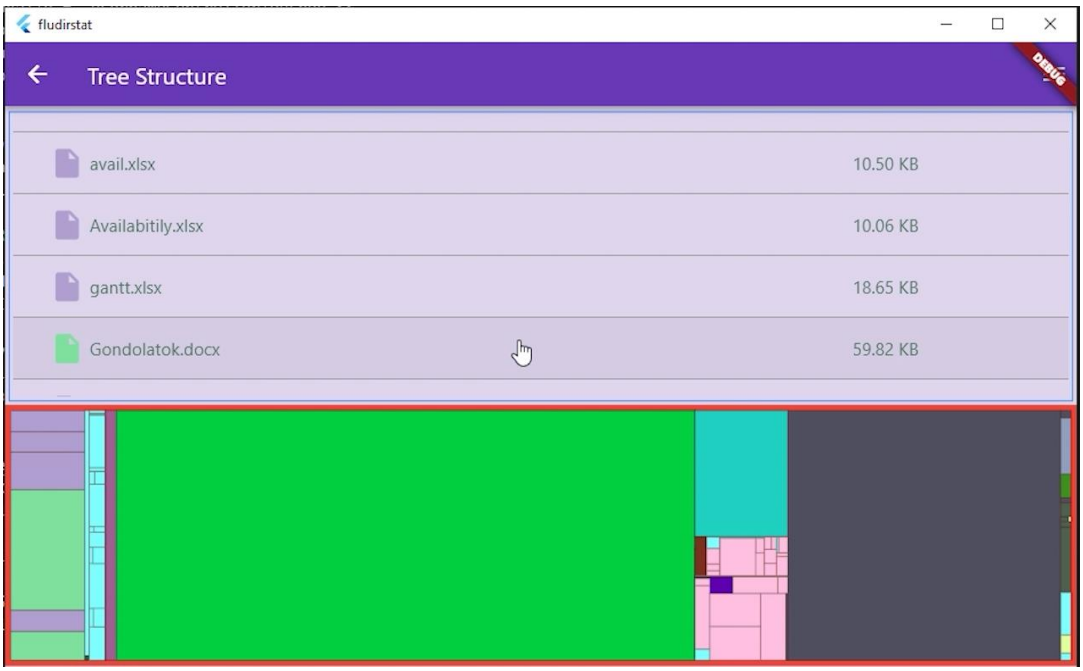
8.1. ábra Betöltendő mappa kiválasztása

Az alábbi ábrán az alkalmazás főmenüje, navigálási menüje látható. Innen lehet tovább lépni a betöltött struktúra megtekintéséhez, a belépési felülethez, valamint a regisztrációhoz.

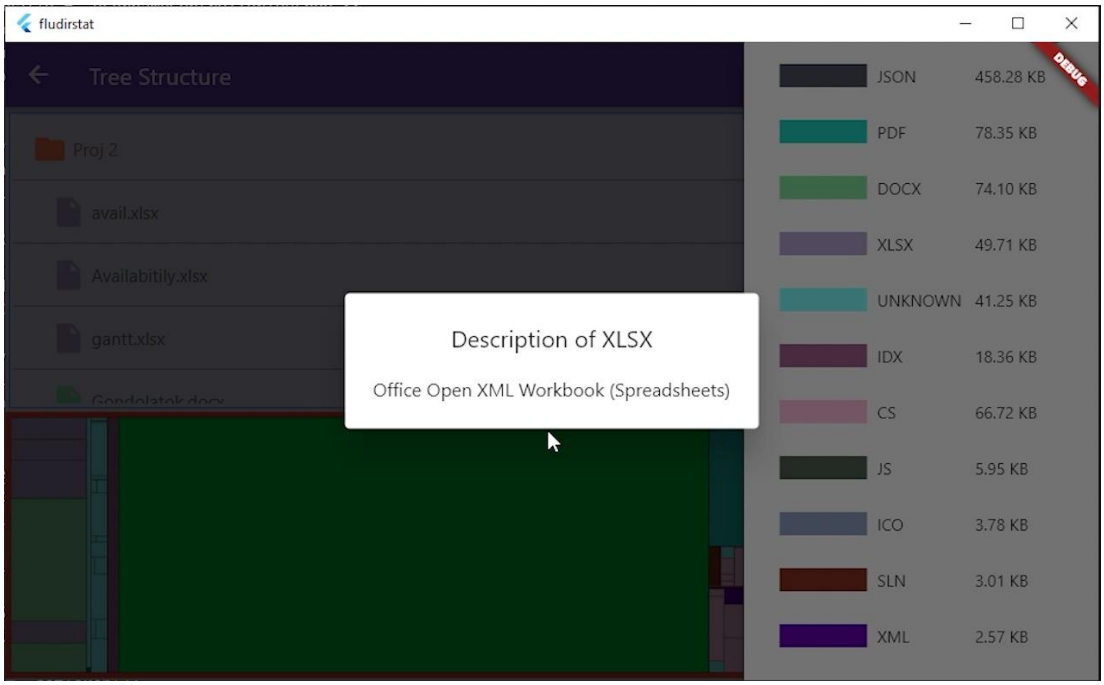


8.2. ábra Az alkalmazás főmenüje

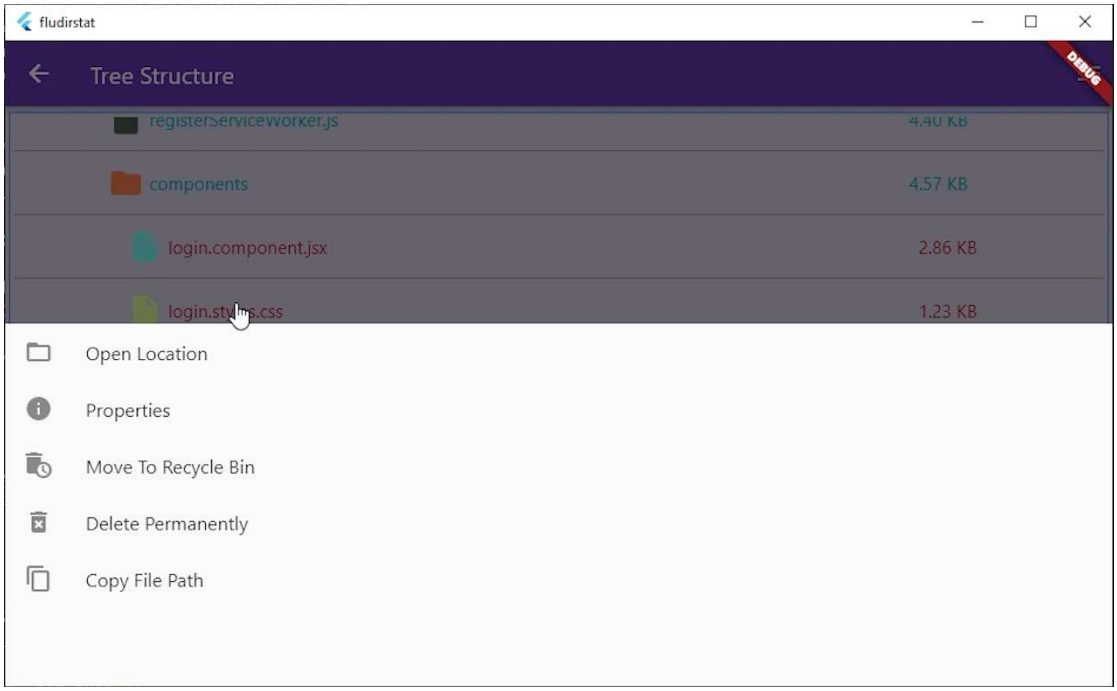
Az alábbi képen a betöltött, feldolgozott FolderTree valamint TreeMap struktúra megjelenítése látható.



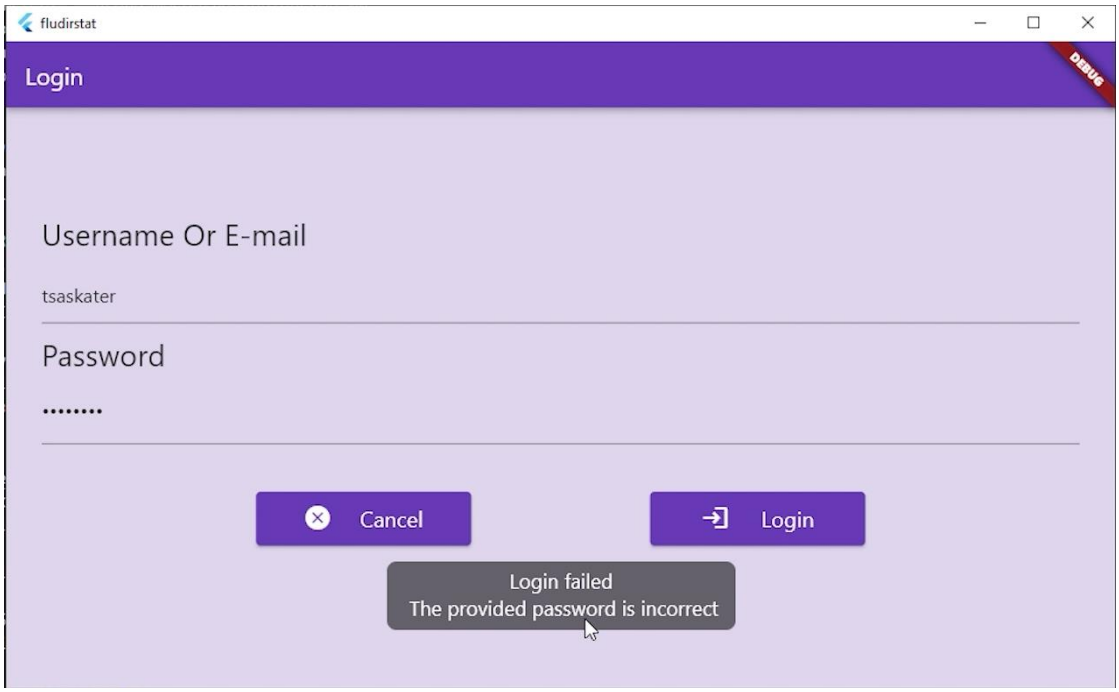
8.3. ábra A betöltött megjelenítési struktúrák



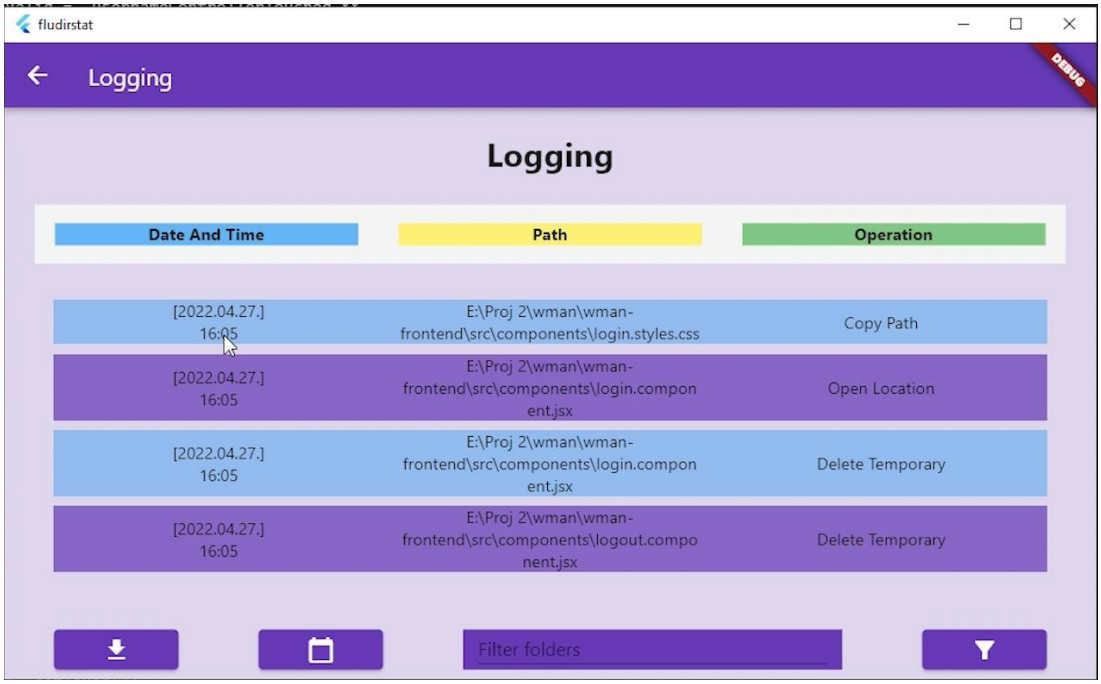
8.4. ábra Fájltípus leírás megtekintése XLSX formátumra



8.5. ábra Fájlokra vagy mappákra kiszabható műveletek megtekintése



8.6. ábra Bejelentkezési felület, rossz jelszavas belépéssel



8.7. ábra Naplózási analitika áttekintése a felhasználói felületen keresztül

9. Összegzés

A szakdolgozatom során megismerkedtem a Flutter nyújtotta cross platform lehetőségekkel, és fejlesztési metodológiákkal. Mivel a Flutter egy relatív új nyelv, így rengeteg esetben a dokumentációhoz kellett fordulnom, valamint teljesen önálló megvalósítási képet kellett alkotnom a különböző problémákra. A fejlesztés során megismerkedtem a Method Channel fogalmával, ezen keresztül pedig a Windows specifikus fájlműveleteket végző API-k kezelésével C++ alapon. Véleményem szerint ez az az implementáció mely a szakdolgozatomban a legtöbb energiát és időbefektetést igényelte.

Az elkészült rendszer jelen állapotban képes megjeleníteni az általam kívánt vizualizációs struktúrákat reszponzív módon. Képes 3170 különböző fájltípus szerint színezni, valamint ezekről a fájltípusokról egy általános leírást nyújtani. Windows alapú rendszerekben pedig képes elvégezni a kívánt fájlokra és mappákra az eredetileg kiszabott műveleteket. Ezen műveletek naplózását pedig könnyedén képesek áttekinteni a felhasználók akár szövegszerkesztőben letöltött állományként, akár magában a szoftver grafikus interfészén keresztül.

Külön köszönetet szeretnék nyilvánítani a konzulensemnek Sipos Miklósnak, aki nagy mértékben segítette motiválta, valamint ellenőrizte a szakdolgozatomban a megfelelő irányban történő haladását.

Irodalomjegyzék

[1] Du Command linux examples, (https://www.geeksforgeeks.org/du-command-linux-examples/), utoljára megtekintve: 2021.12.01.

[2] Gdu A pretty fast disk usage analyzer for Linux, (https://www.tecmint.com/gdu-disk-usage-analyzer-for-linux/), utoljára megtekintve: 2021.12.01.

[3] KDirStat hivatalos főoldala, (http://kdirstat.sourceforge.net/), utoljára megtekintve: 2021.12.01.

[4] QDirStat hivatalos GitHub oldala, (https://github.com/shundhammer/qdirstat), utoljára megtekintve: 2021.12.01.

[5] WinDirStat background and history hivatalos főoldal, (https://windirstat.net/background.html), utoljára megtekintve: 2021.12.01.

[6] WinDirStat hivatalos Github oldala, (https://github.com/windirstat/windirstat), utoljára megtekintve: 2021.12.01.

[7] SpaceSniffer hivatalos FossHub oldala, (https://www.fosshub.com/SpaceSniffer.html), utoljára megtekintve: 2021.12.01.

[8] Disk Inventory X hivatalos oldala, (http://www.derlien.com/), utoljára megtekintve: 2021.12.01.

[9] GrandPerspective SourcForge oldala, (https://sourceforge.net/projects/grandperspectiv/files/stats/timeline), utoljára megtekintve: 2021.12.01.

[10] GrandPerspective hivatalos főoldala, (http://grandperspectiv.sourceforge.net/), utoljára megtekintve: 2021.12.01.

[11] DaisyDisk hivatalos főoldala, (https://daisydiskapp.com/index.html), utoljára megtekintve: 2021.12.01.

[12] Scanner szoftver Wikipedia oldala, (https://daisydiskapp.com/index.html), utoljára megtekintve: 2021.12.04.

[13] Different types of ROM ,(https://studyelectrical.com/2017/06/different-types-of-rom-prom-eprom.html), utoljára megtekintve: 2021.12.04.

[14] IBM 305 RAMAC, (https://www.historyofinformation.com/detail.php?id=740) utoljára megtekintve: 2021.12.04.

[15] Sony Vaio UX UMPC, (https://nbnews.info/en/news/397), utoljára megtekintve: 2021.12.04.

[16] SSD vs. HDD, (https://web.archive.org/web/20110820095531/http://elitepcbuilding.com/ssd-vs-hdd) utoljára megtekintve: 2021.12.04.

[17] The SSD endurance experiment, (https://techreport.com/review/27909/the-ssd-endurance-experiment-theyre-all-dead/), utoljára megtekintve: 2021.12.04.

[18] Commission proposes a common charger for electronic devices, (https://ec.europa.eu/commission/presscorner/detail/en/ip_21_4613), utoljára megtekintve: 2021.12.04.
[19] ISO 9660, (https://www.ibm.com/docs/hu/i/7.3?topic=formats-iso-9660) , utoljára megtekintve: 2021.12.04.
[20] What are A and B drives used for?, (https://www.howtogeek.com/122891/what-are-the-windows-a-and-b-drives-used-for/), utoljára megtekintve: 2021.12.04.
[21] File and directory restrictions on Windows, (https://docs.axway.com/bundle/sync_win_errors/page/file_and_directory_name_restrictions_on_windows.html), utoljára megtekintve: 2021.12.04.
[22] Understanding Metadata, (https://groups.niso.org/apps/group_public/download.php/17446/Understanding%20Metadata.pdf), utoljára megtekintve: 2021.12.04.
[23] 0 size Files and How Metadata is Stored, (https://www.youtube.com/watch?v=tXCbIu0JqZs), utoljára megtekintve: 2021.12.04.
[24] History of Treemap Research at the University of Maryland, (http://www.cs.umd.edu/hcil/treemap-history/index.shtml), utoljára megtekintve: 2021.12.04.
[25] One Million Items Treemap, (http://www.cs.umd.edu/hcil/VisuMillion/), utoljára megtekintve: 2021.12.04.
[26] The Human-Computer Interaction Pioneers Project, (https://hcipioneers.wordpress.com/), utoljára megtekintve: 2021.12.04.
[27] Electron Js apps, (https://www.electronjs.org/apps), utoljára megtekintve: 2021.12.04.
[28] Flutter vs Electron, (https://blog.codemagic.io/flutter-vs-electron/), utoljára megtekintve: 2021.12.04.
[29] Szoftvertechnológia és grafikus felhasználói felület tervezése, (https://users.nik.uni-obuda.hu/prog4/), utoljára megtekintve: 2021.12.04.
[30] List of all file extensions with description in JSON format , (https://gist.github.com/Eseperio/6e40a66b25f5404bab40d87a55012e67), utoljára megtekintve: 2022.05.07.

Ábrajegyzék

Táblajegyzék