UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Marko Đorđević

Digitalna forenzika

Tema: Forenzika fajl sistema

Profesori:

Prof. dr Bratislav Predić

Student:

Marko Đorđević, br. ind. 1168

Niš, Septembar 2021.

Sadržaj

1. Uvod

Nakon Drugog svetskog rata, informatički sistemi su polako ali neizbežno postali stub razvoja ljudskih aktivnosti. Ovaj trend se ubrzao razvojem računara, interneta, kao i konvergencijom računarstva, telekomunikacija, multimedije... Današnji svet umreženih digitalnih uređaja pruža mogućnosti i izazove za pervarante i istraživače, vlade, institucije, poslovanje, komunikaciju kao i za korisnike svesne važnosti očuvanja privatnosti.

Živimo u digitalnom svetu gde se većina informacija stvara, presreće, šalje, čuva i procesira u digitalnom obliku. Digitalne informacije se koriste u svakom aspektu života. Iako korišćenje podataka u digitalnom obliku donosi mnoge tehnološke i ekonomske prednosti, dovelo je do mnogih problema i izazova prilikom izvođenja forenzičke analize digitalnih dokaza. Ti izazovi proizilaze iz sledećih činjenica:

- Digitalni podatak je apstraktna reprezentacija informacije. Predstavlja samo sekvencu bitova i ne poseduje neka vidna svojstva koja bi ukazala na autentičnost ili poreklo informacija.
- Na tržištu postoje razne kolekcije uređaja koji se koriste za stvaranje i snimanje digitalnih informacija, uključujući kamere, audio snimače, digitalne asistente, mobilne telefone i računara. Aktivacija i uzimanje podataka iz tih uređaja može biti veoma zahtevan zadatak, a ukoliko su podaci skriveni, fragmentirani ili kriptovani, postaje još teže.
- Ako uzmemo u obzir relativno niske cene medija za čuvanje podataka, količina podataka s kojom se istraživač susreće je velika. Pregled velikih količina podataka je zahtevan posao koji oduzima previše vremena.
- Digitalni dokazi često putuju raznim kanalima, te su često „raštrkani“ na nekoliko uređaja, u nekoliko različitih formata, što njihovu analizu i slaganje u smislenu celinu čini veoma teškim.
- Digitalni podaci se lako brišu, modifikuju, prepravljaju, kriptuju, uz korišćenje mnogih lako dostupni antiforenzičkih alata.
- Digitalni podaci mogu biti ranjivi, te često zahtevaju brzo prikupljanje i obradu.

Kao odgovor na navede izazove razvijene su metode za rekonstunkciju i uzimanja dokaza iz digitalnih podataka, a potom i njihovu analizu. Te se metode mogu svrstati pod zajedničkim imenom digitalna forenzika.

Digitalna forenzika se može definisati kao skup poznatih metoda za čuvanje, sakupljanje, validaciju, identifikaciju, analizu, interpretaciju, dokumentaciju i prezentaciju digitalnih dokaza uzetih iz digitalnih izvora, s ciljem potvrđivanja ili pomoći pre rekonstunkciji događaja, često kriminalne prirode.

Kako se razvija tehnologija, tako se i digitalna forenzika ubrzano menja i evoluira, pokušavajući da zadrži korak s tehnologijama i razvojem novih softvera.

Kao jedna grana digitalne forenzike jeste i analiza fajl sistema o kojoj ćemo govoriti u ovom seminarskom radu.

U drugom delu govorićemo o samoj analizi fajl sistema. Govorićemo o samom procesu analizi fajl sistema odnosno na koji način se pristupa fajl sistemu.

U trećem delu, govorićemo o fajl sistemu detaljnije, odnosno o detaljima vezanim za analizu i obradićemo NTFS fajl sistem.

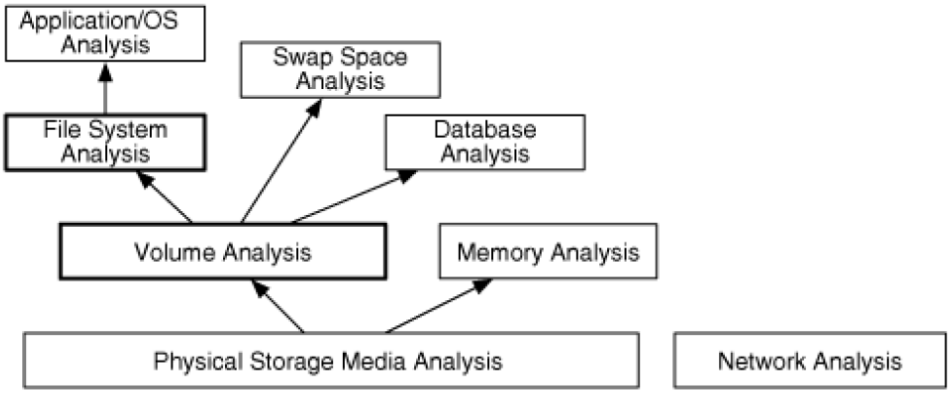
U četvrtom delu prikazaćemo rad DigiatlForensics aplikacije. Upoznaćemo se načinom rada aplikacije, prikazaćemo karakteristike aplikacije kroz primere.

2. Analiza fajl sistema

Analiza fajl sistema predstavlja ispitivanje podataka na particiji ili na disku. Postoji veliki broj rezultata ove analize od kojih su na primer, izlistavanja fajlova u folderima i pregled njihovih metapodataka i povraćaj obrisanih podataka.

Prilikom analize digitalnih podataka, podatak gledamo kao objekat koji su dizajnirali ljudi. Nadalje, sistemi za skladištenje većine digitalnih uređaja dizajnirani su tako da budu skalabilni i fleksibilni i imaju slojeviti dizajn. U skladu slojevitog dizajna definišu se i različiti tipovi analize fajl sistema. Ako počnemo od dna slojeva dizajna, postoje dve nezavisne oblasti analize. Jedan je zasnovan na uređajima za skladištenje a drugi na komunikacionim uređajima. U seminarskom radu obradićemo analizu uređaja za skladištenje, posebno postojecih (eng. Non-volatile) uređaja poput tvrdih diskova. Kada govorimo o postojećim memorijuma, uređajima, mislimo na uređaje koje nakon prestanka električne enerije mogu sadržati informacije.

Slika 1. prikazuje različite oblasti analize uređaja za skladištenje.



Slika 1: Oblasti analize uređaja za skladištenje

Sa slike 1 možemo videti da najniži sloj predstavlja sloj analize fizičkog medijuma za skladištenje podataka (eng. Physical Storage Media Analysis). Primer fizičkih medijuma za skladištenje uključuju čvrste diskove (eng. Hard disk), memorijske čipove i CD-ROM-ove. Analiza ovog područja može uključivati čitanje magnetnih podataka između dve trake podataka ili druge tehnike koje zahtevaju clean room. Clean room ili kako bi se bukvalno prevelo na srpski jezik „Čista soba“ predstavlja radno područje sa kontrolisanom temperaturom i vlažnošću radi zaštite osetljive opreme od kontaminacije, soba sa stalnim prilivom čistog vazduha bez prašine. Pretpostavićemo da imamo pouzdan način čitanja podataka sa fizičkog medijuma u tokovima sa 1-cama i 0-ma koji su bili zapisani na memorijskom uređaju.

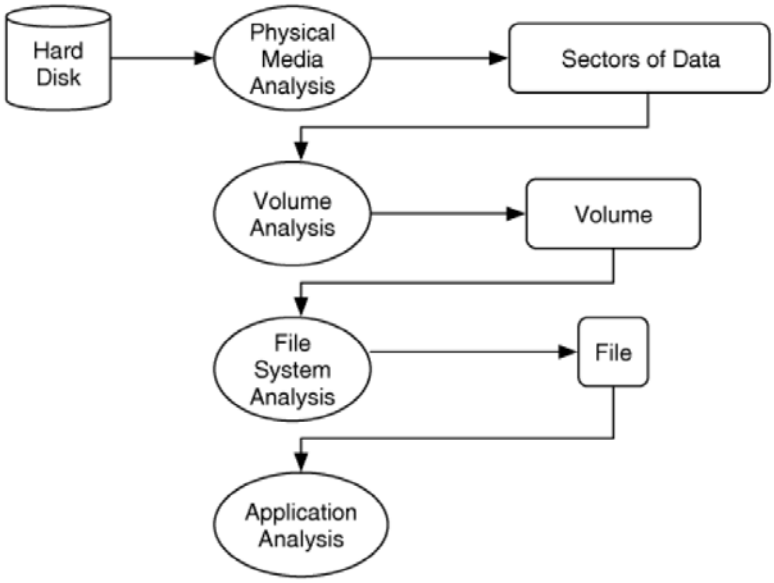
Postojeći uređaji za skladištenje informacija organizovani su u logičke jedinice diska (eng. Volumes). Logička jedinica diska je zbirka lokacija za skladištenje na koje korisnik ili aplikacija mogu pisati i čitati. Disk može sadržati više nosioca podataka u zavisnosti od konfiguracije. U ovom sloju postoje dva glavna koncepta. Jedan je particionisanje pri čemu jednu logičku jedinicu diska delimo na vise logičkih jedinica, dok je drugi sklapanje (eng.

Assembly) gde se kombinuju više logičkih jedinica u jednu veću logičku jedinicu koja se kasnije može podeliti. Neki mediji, kao što su na primer diskovi, ne poseduju podatke u logičkim jedinicama, već je čitav disk logička jedinica.

Unutar svake logičke jedinice može biti smeštena bilo koja vrsta podataka, ali najčešće sadržaji su fajl sistemi. Druge logičke jedinice mogu sadžati bazu podataka ili se koriste kao privremeni prostor za zamenu (eng. Swap space). Fajl sistem predstavlja zbirku struktura podataka koje aplikaciji omogućavaju kreiranje, čitanje i pisanje datoteka odnosno fajlova.

Da bismo razumeli šta se nalazi unutar fajla, potrebno je obraditi sloj aplikacije. Struktura svakog fajla zasniva se na aplikaciji ili operativnom sistemu koji je datoteku stvorio. Na primer, iz perspektive fajl sistema, Windows registarski fajl se ne razliku je od HTML stranice jer su obe datoteke, fajlovi. Interno , oni imaju veoma različite strukture i potrebno je razviti različite alate za svaku analizu. Kada posmatramo na apstraktnom nivou kao što smo i rekli fajl je fajl, ali sama struktura fajlova se razlikuje, odnosno ekstenzija fajl daje odgovor o samoj strukturi fajla. Same aplikacije definiše ekstenziju fajla prilikom pamćenja na fajl sistem, pa je zbog toga neizostavno obraditi i ovaj sloj prilikom analize fajl sistema.

Proces analize možemo videti na slici 2.



Slika 2: Proces analize podataka od fizičkog nivoa pa do nivoa aplikacija

Slika 2 prikazuje disk, memorijski medijum, koji se analizira kako bi se proizveo tok bajtova, koji se analiziraju na sloju logičkih jedinica, zatim se logičke jedinice analiziraju na sloju fajl sistema kako bi se proizvela datoteka, fajl. Fajl se zatim analizira na aplikacionom sloju.

3. Fajl sistem

Motivacija fajl sistema je prilično jednostavna: računarima je potrebna metoda za dugoročno skladištenje i preuzimanje podataka. Fajl sistem predstavlja način na koji se podaci smeštaju (organizuju) i čitaju sa nekog medijuma za skladištenje podataka (npr. Hard Disk). Bez fajl sistema podacima bi se veoma teško pristupalo jer bi bilo potrebno ustanoviti gde je početak a gde kraj podatka. Zato deljenjem podataka u odvojene celine i dodeljivanjem imena tim celinama, lako možemo da razdvojimo i identifikujemo podatke koji su nam potrebni. Zbog toga se uvodeni pojmovi fajl i folder odnosno datoteka i direktorijum. Korisnicima se nudi mogućnost hijararhijsku organizaciju podataka radi, kao što smo već i naveli brze pretrage.

Fajl sistem se sastoji od strukturnih i korisničkih podataka koji su organizovani tako da računar zna gde ih može pronaći. U većini slučajeva, fajl sistem je nezavisan od bilo kog računara.

Način rada fajl sistema baziran je na primeru rada jedne poslovne kancelarije, gde papiri sa nekim podacima na njima predstavljaju fajlove, a fascikle i fioke gde se ti papiri čuvaju predstavljaju direktorijum. Kada nam je potreban neki papir mi tačno znamo u kojoj fascikli ili fioci se on nalazi. Isto tako i fajl sistem zna koji fajl i u kom direktorijumu se nalazi i kako su oni smešteni na hard disku. Fajl sistem je hijararhijski organizovan, tako da se u jednom folderu mogu nalaziti fajlovi ali i drugi direktorijumi u kojima se dalje mogu nalaziti isto fajlovi i direktorijumi.

Fajl sistemi imaju posebne procedure i strukture koje se mogu koristiti za skladištenje jedne datoteke na disketi ili desetine hiljada datoteka u nizu za skladištenje. Svaka instanca sistema datoteka ima jedinstvenu veličinu, ali njena osnovna struktura omogućava obradu bilo kom računaru koji podržava tip sistema datoteka.

Svaki fajl i direktorijum ima tri važne kategorije: ime, sadržaj i metadata. Ime fajla ili direktorijuma se koristi za njihovo identifikovanje u fajl sistemu. Zato na primer nije moguće imati dva fajla ili direktorijuma sa istim imenom u okviru nekog direktorijuma. Sadržaj se više odnosi na fajlove jer su to ustvari podaci koji se čuvaju u ovkiru tog fajla. A metadata podaci su podaci koji sadrže dodatne informacije o fajlovima i direktorijumima tj. to su podaci koji opisuju fajlove i direktorijume. Neki od metapodataka su na primer putanja do fajla ili foldera, njihova veličina, vreme kada su kreirani, vreme kada im je poslednji put pristupano za čitanje ili upis, itd…

Operativni sistem mora da ima način na koji razume fajl sistem kako bi mogao da prikazuje, otvara i čuva podatke na njemu. Različiti operativni sistemi imaju različite načina za organizovanje i čuvanje podataka na hard diskovima ili nekim drugim skladištima podataka, zato se i njihovi fajl sistemi razlikuju. Windows, Mac i Linux svi koriste različite fajl sisteme, jer svako je razvijao fajl sistem onako kako je njima bilo potrebno. Tako da su neki fajl sistemi brži, neki stabilniji ili više skalabini.

Windows operativni sistem koristi fajl sisteme FAT (FAT32) i NTFS. „FAT“ je skraćenica od „File Allocation Table“ (Tabela alokacija fajlova), a „NTFS“ je skraćenica od „New Technology File System“ (Nova tehnologija fajl sistema). FAT32 fajl sistem se više koristi za starije verzije Windowsa operativnog sistema (Windows XP), dok se NTFS fajl sistem koristi za novije verzije Windowsa. Mac operativni sistem koristi HFS+ fajl sistem, dok Linux koristi

Ext2, Ext3 i Ext4 fajl sistem. Trenutno je u razvoju novi Btrfs fajl sistem za Linux koji bi trebao da zameni Ext4 jednog dana. U nastavku ovog rada biće opisan NTFS fajl sistem.

3.1. Kategorije podataka

Pre nego što krenemo sa ispitivanje fajl sistema, potrbeno je upoznati se sa referentnim modelom podataka fajl sistema, odnosno modelom podataka fajl sistema.

Kao referentni model koristićemo 5 kategorija podataka:

1. Fajl sistem
2. Sadržaj (eng. Content)
3. Metapodaci
4. Naziv fajlova
5. Aplikacija

Kategorija fajl sistema sadrži opšte informacije o fajl sistemu. Svi fajl sistemi imaju opštu strukturu, ali svaka instanca fajl sistema je jedinstvena jer ima jedinstvenu veličinu i može se podesiti za performanse. Podaci u ovoj kategoriji nam govore gde možemo pronaći određene strukture podataka i koliko je velika jedinica podataka.

Kategorija sadržaj sadrži podatke koji sadrže stvarni sadržaj fajla, što je i razlog zašto imamo fajl sisteme na prvom mestu. Većina podataka u fajl sistemu pripada ovoj kategoriji i obično je organizovano u zbirku kontejnera standardne veličine. Svaki fajl sistem dodeljuje drugačije ime kontejnerima, kao što su klasteri i blokovi.

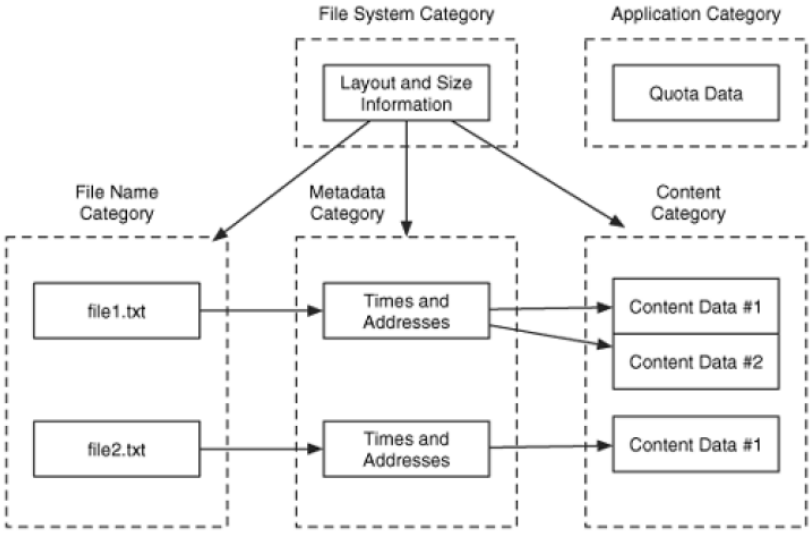
Kategorija metapodataka sadrži podatke koji opisuju fajl; to su podaci koji opisuju podatke. Ova kategorija sadrži informacije, na primer, gde se nalazi sadržaj datoteke, koliko je veliki fajl, datoteka, vreme i datumi kada je datoteka poslednji put pročitana ili upisana, kao i informacije o kontroli pristupa. Ova kategorija ne sadrži sadržaj same datoteke,fajla i možda ne sadrži naziv datoteke, fajla.

Kategorija naziva fajla sadrži podatke koji dodeljuju ime svakoj datoteci odnosno fajlu. U većini sistema ovi podaci se nalaze u sadržaju direktorijuma i predstavljaju listu imena fajlova sa odgovarajućom adresom metapodataka. Kategorija naziva fajla je slična imenu hosta na mreži. Mrežni uređaji međusobno komuniciraju pomoću IP adresa, koje je ljudima teško zapamtiti. Kada korisnik unese ime hosta udaljenog računara, lokalni računar mora prevesti ime u IP adresu da bi komunikacija mogla započeti.

Kategorija aplikacija sadrži podatke koji pružaju posebne funkcije. Ovi podaci nisu potrebni tokom procesa čitanja ili pisanja datoteke i, u mnogim slučajevima, ne moraju biti uključeni u specifikaciju fajl sistema. Ovi podaci su uključeni u specifikaciju jer bi moglo biti efikasnije implementirati ih u sistem datoteka umesto u fajl. Primeri podataka u ovoj kategoriji uključuju statistiku korisnika i dnevnike fajl sistema. Ovi podaci mogu biti korisni tokom istrage, ali

budući da nisu potrebni za pisanje i čitanje datoteke, mogli bi se lakše falsifikovati od ostalih podataka.

Relacija između navedenih 5 kategorija možemo videti sa slike 3



Slika 3: Relacija između 5 kategorija podataka

Kada smo uveli ovu kategorizaciju, sada ćemo da pokažemo na realnom primeru kada se vrši pretraživanje odgovarajućeg fajla u odnosu na kategoriju podatka. Ukoliko tražimo datoteku sa nastavkom „JPG“, fokusiraćemo se na tehnike analize kategorije imena fajlova. Ako na primer tražimo fajl sa određenom vrednošću, koristiemo tehnike analize kategorije metatpodataka jer ova kategorija sadrži adrese jedinica podataka.

U sledećim potpoglavljima opisivaćemo tehnike koje se koriste prilikom analize nabrojanih 5 kategorija.

3.1.1 Kategorija fajl sistem

Kategorija fajl sistem sadrži opšte podatke koji identifikuju kako je ovaj sistem datoteka jedinstven i gde se nalaze drugi važni podaci. U mnogim slučajevima, većina ovih podataka nalazi se u standardnoj strukturi podataka u prvim sektorima fajl sistema, slično kao da imamo mapu u predvorju zgrade. Pomoću ovih informacija mogu se pronaći lokacije drugih podataka, koje mogu varirati u zavisnosti od veličine datotečnog sistema.

Analiza podataka u kategoriji fajl sistema potrebna je za sve vrste analiza fajl sistema, jer u ovoj fazi ćemo pronaći lokacije struktura podataka u drugim kategorijama. Stoga, ako neki od ovih podataka postane oštećen ili se izgubi, dodatna analiza je teža jer moramo pronaći rezervne kopije ili pogoditi koje su bile vrednosti. Osim opštih informacija o izgledu, analiza ove kategorije može takođe pokazati verziju fajl sistema, aplikaciju koja je kreirala fajl sistem, datum kreiranja i oznaku fajl sistema. U ovoj kategoriji postoji malo podataka koje bi tipičan korisnik mogao postaviti ili pregledati bez pomoći hex uređivaća. U mnogim slučajevima, podaci o ne-rasporedu u ovoj kategoriji smatraju se nebitnim podacima i možda nisu tačni.

Podaci u ovoj kategoriji su obično pojedinačne i nezavisne vrednosti; stoga se s njima ne može mnogo učiniti osim prikazati ih ovlašćenom licu ili ih koristiti u nekom programskom alatu. Ako ručno oporavljamo (eng. Backup) podatke, informacije o fajl sistemu mogu biti korisne. Ako pokušavamo da utvrdite na kom računaru je sistem datoteka kreiran, ID ili verzija volumena mogu biti korisni. Strukture podataka u ovoj kategoriji često imaju neiskorištene vrednosti i lokacije za skladištenje koje se mogu koristiti za sakrivanje malih količina podataka. Provera doslednosti u ovoj kategoriji je upoređivanje veličine fajl sistema sa veličinom logičke jedinice u kome se on nalazi. Ako je logička jedinica veća, sektori nakon fajl sistema nazivaju se slabljenje logičke jedinice (eng. volume slack) i mogu se koristiti za skrivanje podataka.

3.1.2. Kategorija sadržaj

Kategorija sadržaj uključuje lokacije za skladištenje koje su dodeljene datotekama i direktorijumima tako da mogu da čuvaju podatke. Podaci u ovoj kategoriji su obično organizovani u grupe jednake veličine, koje nazivam jedinicama podataka, iako svaki sistem datoteka ima jedinstveno ime za njih, kao što je klaster ili blok. Jedinica podataka je ili u dodijeljenom ili neraspoređenom stanju. Obično postoji neka vrsta strukture podataka koja prati status alokacije svake jedinice podataka. Jedinica podatka je najmanja samostalna jedinica podataka memorije dostupna u datom fajl sistemu. Kod Unix sistema one su poznate kao blokovi. Istorijski veličina sektora svakog diska bila je 512 bajta – mnogi moderni fajl sistemi koriste 4096 bajta (4K) i više kao najmanje adresibilne jedinice podataka. Informacija dostupna kod jedinice podatka je jednostavna: kontekst jedinice. Ukoliko je jedinica dodeljena JPEG image-u, ona će sadržati deo JPEG podatka. Ukoliko joj je dodeljen tekst fajl sadržaće tekst.

Kada se kreira nova datoteka ili se postojeća datoteka poveća, operativni sistem traži neraspoređenu jedinicu podataka i dodeljuje je datoteci. Kada se datoteka izbriše, jedinice podataka koje su dodeljene datoteci postavljaju se u neraspoređeno stanje i mogu se dodeliti novim datotekama. Većina operativnih sistema ne briše sadržaj jedinice podataka kada nije raspoređena, iako neki alati za bezbedno brisanje i operativni sistemi pružaju tu mogućnost.

Analiza kategorije sadržaja vrši se radi oporavka izbrisanih podataka i pretraživanja na niskom nivou. U ovoj kategoriji ima mnogo podataka; stoga se obično ne analizira ručno.

3.1.3. Kategorija metapodataka

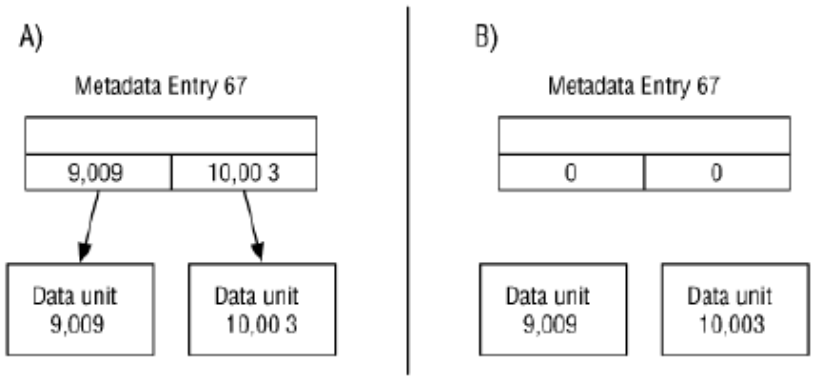
Kategorija metapodataka je mesto gde se nalaze opisni podaci. Ovde možemo pronaći, na primer, vreme poslednjeg pristupa i adrese jedinica podataka koje je datoteka dodelila. Nekoliko alata eksplicitno identifikuje analizu metapodataka; umesto toga, obično se spaja sa analizom kategorije imena datoteke.

Mnoge strukture metapodataka su uskladištene u tabeli fiksne ili dinamičke dužine, a svaki unos ima adresu. Kada se datoteka izbriše, unos metapodataka se postavlja u neraspoređeno stanje, a OS može izbrisati neke od vrednosti u unosu.

Analiza se sprovodi u kategoriji metapodataka radi utvrđivanja više detalja o određenoj datoteci ili radi pretraživanja datoteke koja ispunjava određene uslove. Ova kategorija ima tendenciju da ima više nebitnih podataka od drugih kategorija. Na primer, poslednje pristupano ili napisano vreme možda nije tačno ili OS možda nije primenio postavke kontrole pristupa datoteci; stoga, potrebni su dodatni dokazi da podrže nebitne podatke u ovoj kategoriji.

Ova kategorija metapodataka u analizi fajl sistema se koristi za povraćaj obrisanih podataka. Kod povraćaja obrisanih fajlova postoje dva glavna metoda za povraćaj brisanih podataka, a to su analize bazirane na metapodacima i metoda bazirana na aplikaciji.

Povraćaj obrisanih fajlova zasnovan na metapodacima funkcioniše kada metapodaci iz izbrisane datoteke još uvek postoje. Ako su metapodaci obrisani ili je struktura metapodataka raspodeljena u novoj datoteci, tada se koristi metod baziran na aplikaciji. Oporavak se radi tako što se pronađe struktura metapodataka za datoteku i zatim se samo pročitaju vrednosti. Ukoliko se prostor za datoteku označi samo kao da je sloodan, sam sadržaj metapodatka za datoteku sadrži i dalje adresu jedinica podataka sa kojih se može pročitati. Ukoliko pak sa druge strane operativni sistem obriše adresu kada je datoteka izbrisana primenjuje se posebne tehnike oporavka. Ilustracija ovo primera data je na slici 4.



Slika 4: Primena metapodataka prilikom povraćaja izbrisanih fajlova

Prilikom oporavka fajlova zasnovanog na metapodacima moramo biti oprezni prilikom čitanja strukture metapodataka jer podaci mogu postati nesinhronizovani jer su jedinice podataka dodeljeni novim datotekama.

3.1.4. Kategorija naziva fajla

Kategorija naziva fajla uključuje nazive fajla i omogućava korisniku da se poziva na datoteku po imenu umesto po adresi metapodataka. U osnovi, ova kategorija podataka uključuje samo naziv fajla i njenu adresu metapodataka. Neki fajl sistemi takođe mogu da sadrže informacije o vrsti fajla ili privremene informacije, ali to nije standard.

Važan deo analize imena fajla je da se utvrdi gde se nalazi osnovni direktorijum jer nam je potreban da pronađemo datoteku kada se navede njena puna putanja. Korenski direktorijum je osnovni direktorijum. Na primer, u operativnom sistemu Windows C: \ je osnovni direktorijum pogona C: \. Svaki sistem datoteka ima svoj način definisanja lokacije osnovnog direktorijuma.

3.1.5. Kategorija aplikacija

Neki fajl sistemi sadrže podatke koji pripadaju kategoriji aplikacija. Ovi podaci nisu bitni za fajl sistem i obično postoje kao posebni podaci o fajl sistemu umesto unutar normalne datoteke jer su efikasniji.

Tehnički, svaki fajl koju OS ili aplikacija kreira može biti dizajnirana kao funkcija u fajl sistemu. Na primer, Acme Software bi mogao da odluči da bi njegov OS bio brži da je deo sistema datoteka rezervisan za adresar. Umesto čuvanja imena i adresa u fajlu, oni bi bili sačuvani u posebnom odeljku logičke jedinice. To može dovesti do poboljšanja performansi, ali nije neophodno za fajl sistem.

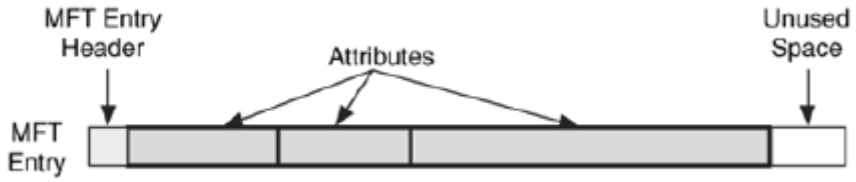
3.2. Analiza NTFS fajl sistema

NTFS (New Techonologies File System) je fajl sistem razvijen od strane Microsoft-a koji se koristi u svim Windows operativnim sistemima od Windows XP verzije. Pre NTFS-a korišćen je FAT32 fajl sistem koji se danas koristi kod eksternih skladišta za čuvanje podataka kao što je na primer USB fleš.

NTFS fajl sistem je dizajniran da bude pouzdan, siguran i da ima podršku za uređaje za skladištenje velike količine podataka. Skalabilnost se obezbeđuje korišćenjem generičkih struktura podataka koji se wrap-uju oko struktura podataka sa specifičnim sadržajem. Ovo predstavlja skalabilni dizajn jer se unutrašnja struktura podataka može vremenom promeniti kako se novi zahtevi postavljaju na fajl sistem, a opšti wrap-er može biti konstantan.

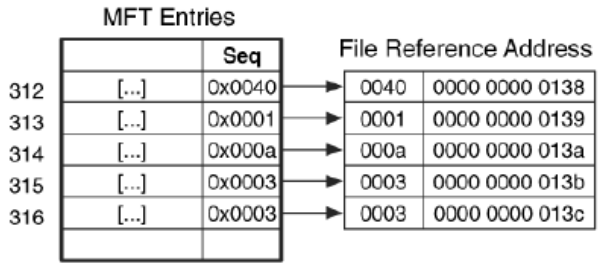
Najvažniji koncept koji se mora razumeti u dizajnu NTFS fajl sistema jeste to da se svi podaci smeštaju i čuvaju kao fajlovi. Svi podaci koji su bitni za operativni sistem se takođe smeštaju u fajlove koji su najčešće sakriveni od korisnika, ali je i te fajlove moguće pronaći u ovkiru particije kao i svaki drugi običan fajl. Za razliku od drugih fajl sistema NTFS fajl sistem nema specifičan layout, već se ceo fajl sistem smatra kao oblast podataka i da bilo koji sektor može se dodeliti fajlu.

Glavni deo NTFS-a je MFT (Master File Table) tabela koja sadrži informacije o svim fajlovima i folderima. Na početku NTFS-a se prvo nalazi Boot Sector a zatim MFT tabela. Svaki fajl i folder su bar jednom uneti u ovu tabelu. Svaki vrsta u ovoj tabeli odnosi se na jedan fajl ili folder. Informacije koje se čuvaju u vrsti zauzimaju 1 KB prostora, a samo prvih 42B se imaju definisanu svrhu, dok ostali bajtovi sadrže attribute koji su ustvari manje strukture podataka (npr. jedan atribut se koristi za čuvanje imena fajla). Kao i sve drugo u NTFS-u i MFT tabela je fajl. Čak i prva vrsta u MFT tabeli se odnosi na nju i ima ime $MFT i u ovoj vrsti se nalazi lokacija MFT tabele na disku i to je jedino mesto gde može da se pronađe lokacija MFT tabele. Veličina MFT tabele nije fiksa već se vremenom dinamički menja. Jednom dodata vrsta u MFT tabeli se više nikad ne briše, iako folder ili fajl je obrisan. Slika 5 prikazuje osnovni raspored unosa u MFT tabeli gde postoje indomacije zaglavlja i tri atributa.



Slika 5: Prikaz osnovnog rasporeda unosa u MFT tabeli

Svakom unosu se daje adresa na osnovu lokacije u tabeli počevši od 0 i imaju veličinu 1024 bajta (tačna veličina je definisana u boot sektoru). Ukoliko se desi da atributi jednog fajla ne mogu da stanu u jednu vrstu MFT tabele onda se kreiraju nove vrste sve dok se atributi ne sačuvaju, a tada prva vrsta koja se odnosi na taj fajl se naziva base file record, i svaka sledeća vrsta sadrži jedno polje u kome se nalazi adresa ove prve vrste. Na slici 6. dat je primer MFT tabele.



Slika 6: Primer MFT tabele

Svaki MFT unos takođe ima 16-bitni redni broj koji se uvećava kada se unos dodeli. Na primer, MFT unos 313 sa rednim(eng. Sequence number) brojem 1. Ukoliko se fajl sa unosom 313 obriše, novi ulaz označiće novi fajl. Kada se unos ponovo dodeli, on ima novi redni broj 2. MFT unos i redni broj se kombinuju, sa rednim brojem u gornjim 16-bitovima, kako bi se formirala 64-bitna referentna adresa fajla, kao što je prikazano na slici 6.

3.2.1. Metapodaci fajlova

Pošto je svaki bajt u logičkoj jedinici dodeljen fajlu, moraju postojati fajlovi koji čuvaju administrativne podatke fajl sistema. Microsoft naziva ove fajlove naziva fajlove metapodataka, ali to može izazvati zabunu jer se takođe pozivamo na metapodatke fajlova. Nazivaću ove posebne fajlove metapodaci fajl sistema.

Prvih 16 vrsta u MFT tabeli su rezervisana za sistemske fajlove. Ovi rezervisani unosi koji se ne koriste imaju samo osvnovne i opšte informacije. Svaki fajl metapodataka fajl sistema je navedena u korenskom direktorijumu, iako su obično sakrivene od većine korisnika. Naziv svakog fajla metapodataka počinje sa „$“, a prvo slovo je napisano velikim slovima.

Na slici 7. prikazani su sistemski fajlovi koji se nalaze u MFT tabeli.

Entry	File Name	Description
0	$MFT	The entry for the MFT itself.
1	$MFTMirr	Contains a backup of the first entries in the MFT. See the "File System Category" section in Chapter 12.
2	$LogFile	Contains the journal that records the metadata transactions. See the "Application Category" section in Chapter 12.
3	$Volume	Contains the volume information such as the label, identifier, and version. See the "File System Category" section in Chapter 12.
4	$AttrDef	Contains the attribute information, such as the identifier values, name, and sizes. See the "File System Category" section in Chapter 12.
5	.	Contains the root directory of the file system. See the "File Name Category" section in Chapter 12.
6	$Bitmap	Contains the allocation status of each cluster in the file system. See the "Content Category" section in Chapter 12.
7	$Boot	Contains the boot sector and boot code for the file system. See the "File System Category" section in Chapter 12.
8	$BadClus	Contains the clusters that have bad sectors. See the "Content Category" section in Chapter 12.
9	$Secure	Contains information about the security and access control for the files (Windows 2000 and XP version only). See the "Metadata Category" section in Chapter 12.
10	$Upcase	Contains the uppercase version of every Unicode character.
11	$Extend	A directory that contains files for optional extensions. Microsoft does not typically place the files in this directory into the reserved MFT entries.

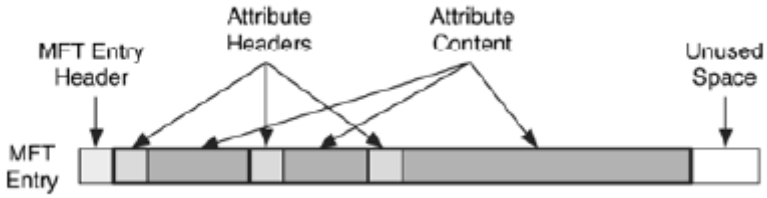
Slika 7: Sistemski fajlovi koji se nalaze u prvih 16 vrsta u MFT tabeli

3.2.2 Koncepti atributa unosa u MFT

Unos MFT ima malu unutrašnju strukturu i najvišu ako se koristi za skladištenje atributa, a to su strukture podataka koje skladište određenu vrstu podataka. Postoji mnogo vrsta atributa, a svaki ima svoju unutrašnju strukturu. Na primer, postoje atributi za naziv fajla, datum i vreme, pa čak i njen sadržaj. Ovo je jedan od načina na koji se NTFS veoma razlikuje od drugih fajl sistema. Većina sistema datoteka postoji za čitanje i pisanje sadržaja fajlova, ali NTFS postoji za čitanje i pisanje atributa, od kojih jedan slučajno sadrži sadržaj fajla.

Razmotrimo analogiju koja opisuje MFT unos kao veliku kutiju koja je u početku prazna. Atributi su slični manjim kutijama unutar veće kutije gde manje kutije mogu biti bilo kog oblika koji najefikasnije čuva objekat. Na primer, šešir se može čuvati u kutiji sa kratkim okruglim slojem, a plakat u dugoj kutiji.

Iako svaki tip atributa skladišti različitu vrstu podataka, svi atributi imaju dva dela: zaglavlje i sadržaj. Slika 8 prikazuje MFT unos sa četiri para zaglavlja i sadržaja. Zaglavlje je opšte i standardno za sve atribute. Sadržaj je specifičan za vrstu atributa i može biti bilo koje veličine. Ako pomislimo na analogiju naših kutija, uvek se nalaze iste osnovne informacije sa spoljne strane svake male kutije, ali oblik svake kutije može biti drugačiji.



Slika 8: MFT unos četiri para zaglavlja i sadržaja

Atributi zaglavlja

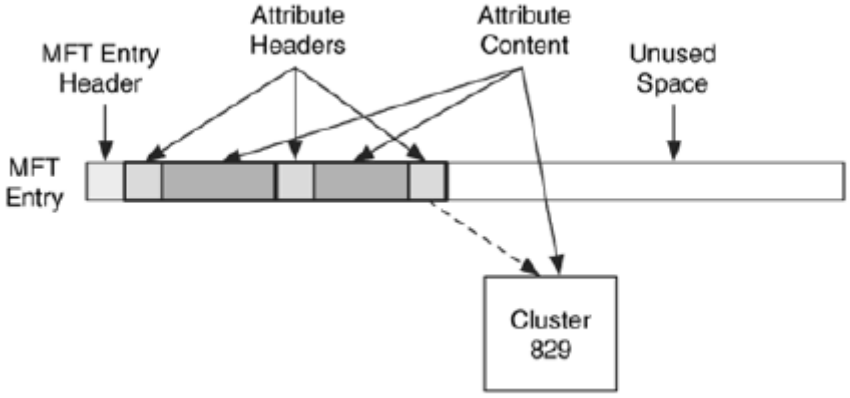
Atributi zaglavlja identifikuje tip atributa, njegovu veličinu i naziv. Takođe poseduje fleg za identifikaciju da li je vrednost kompresovana ili šifrovana. Tip atributa je numerički identifikator zasnovan na tipu podataka. Unos MFT može imati više atributa istog tipa.

Nekim atributima se može dodeliti ime i oni se čuvaju u UTF-16 Unicode u zaglavlju atributa. Atributu takođe ima dodeljenu vrednost identifikatora koja je jedinstvena za taj MFT unos. Ako unos ima više atributa istog tipa, ovaj identifikator se može koristiti za njihovo razlikovanje.

Atributi sadržaja

Sadržaj atributa može imati bilo koji format i bilo koju veličinu. Na primer, jedan od atributa se koristi za skladištenje sadržaja fajla, tako da može biti veličine nekoliko MB ili GB. Nije praktično skladištiti ovu količinu podataka u unosu MFT, koja je samo 1.024 bajta.

Da bi se rešio ovaj problem, NTFS nudi dve lokacije na kojima se može skladištiti sadržaj atributa. Rezidentni atribut skladišti svoj sadržaj u unosu MFT sa zaglavljem atributa. Ovo radi samo za male atribute. Nerezidentni atribut skladišti svoj sadržaj u spoljnom klasteru u sistemu datoteka. Zaglavlje atributa identifikuje da li je atribut rezident ili nerezident. Ako je atribut rezidentan, sadržaj će odmah pratiti zaglavlje. Ako je atribut nerezidentni, zaglavlje će dati adrese klastera. Na slici 9 vidimo primer unosa MFT koji smo videli ranije, ali sada je njegov treći atribut prevelik da bi stao u MFT i dodelio je klaster 829.



Slika 9: Izgled MFT unosa rezidentnog atributa

3.2.3. Standardni tipovi atributa

Kao što je prethodno rečeno svaki atribut ima svoj tip. Tip atributa se definiše preko njegovog numeričkog broja. Na slici 10 su prikazani svi standardni tipovi atributa u MFT tabeli. Pored numeričkog broja tip atributa ima ime koje je počinje sa znakom $ i sva su slova velika i opis koji tip šta predstavlja. Ovi atributi nisu dostupni za sve fajlove. Na primer folderi nemaju atribut tipa $DATA, a fajlovi nemaju atribut tipa $INDEX_ROOT.

Type Identifier	Name	Description
16	$STANDARD_INFORMATION	General information, such as flags; the last accessed, written, and created times; and the owner and security ID.
32	$ATTRIBUTE_LIST	List where other attributes for file can be found.
48	$FILE_NAME	File name, in Unicode, and the last accessed, written, and created times.
64	$VOLUME_VERSION	Volume information. Exists only in version 1.2 (Windows NT).
64	$OBJECT_ID	A 16-byte unique identifier for the file or directory. Exists only in versions 3.0+ and after (Windows 2000+).
80	$SECURITY_ DESCRIPTOR	The access control and security properties of the file.
96	$VOLUME_NAME	Volume name.
112	$VOLUME_ INFORMATION	File system version and other flags.
128	$DATA	File contents.
144	$INDEX_ROOT	Root node of an index tree.
160	$INDEX_ALLOCATION	Nodes of an index tree rooted in $INDEX_ROOT attribute.
176	$BITMAP	A bitmap for the $MFT file and for indexes.
192	$SYMBOLIC_LINK	Soft link information. Exists only in version 1.2 (Windows NT).
192	$REPARSE_POINT	Contains data about a reparse point, which is used as a soft link in version 3.0+ (Windows 2000+).
208	$EA_INFORMATION	Used for backward compatibility with OS/2 applications (HPFS).
224	$EA	Used for backward compatibility with OS/2 applications (HPFS).
256	$LOGGED_UTILITY_STREAM	Contains keys and information about encrypted attributes in version 3.0+ (Windows 2000+).

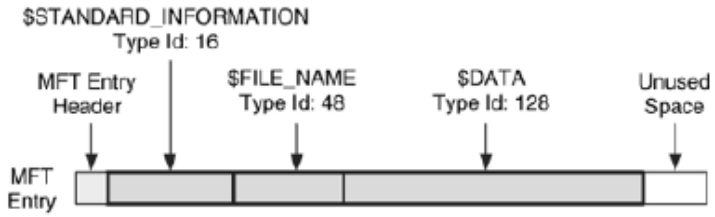
Slika 10: Standardni tipovi atributa u MFT tabeli

Skoro svaki dodeljeni MFT unos ima atribut tipa $FILE_NAME i $ STANDARD_INFORMATION. Jedini izuzetak su ne-bazični MFT unosi. Atribut $ FILE_NAME sadrži naziv fajla, veličinu i vremenske informacije. Atribut $ STANDARD_INFORMATION sadrži privremene, vlasničke i bezbednosne informacije. Kasniji atribut postoji za svaku fajl i direktorijum jer sadrži podatke potrebne za jačanje sigurnosti podataka. U apstraktnom smislu, u ovom atributu nema bitnih podataka, ali karakteristike fajl sistema na nivou aplikacije zahtevaju da on postoji. Oba ova atributa su uvek stalna.

Svaka datoteka ima atribut $ DATA koji sadrži sadržaj datoteke. Ako je sadržaj veličine otprilike 700 bajta, on postaje nerezidentni i pamti se u klastere. Kada fajl ima više atributa $ DATA, dodatni atributi se ponekad nazivaju alternativni tokovi podataka. Podrazumevani atribut $ DATA koji se stvara prilikom kreiranja datoteke nema pridruženo ime, ali dodatni atributi $ DATA moraju da imaju. Treba imati na umu da se ime atributa razliku od naziva tipa. Na primer $ DATA je naziv tipa atributa, a naziv atributa bi mogao da bude „fred“.

Svaki direktorijum ima atribut $ INDEKS_ROOT koji sadrži informacije o fajlovima i poddirektorijumima koji se u njemu nalaze. Ako je direktorijum veliki, atributi $ INDEKS_ALLOCATION i $ BITMAP se takođe koriste za skladištenje informacija. Da stvari budu zbunjujuće, moguće je da direktorijum ima atribut $ DATA pored atributa $ INDEKS_ROOT. Drugim rečima, direktorijum može da skladišti i sadržaj datoteke i listu svojih datoteka i poddirektorijuma. Atribut $ DATA može da skladišti bilo koji sadržaj koji aplikacija ili korisnik žele da uskladište tamo. Atributi $ INDEKS_ROOT i $

INDEKS_ALLOCATION za direktorijum obično imaju naziv "$ I30.". Na slici 11 je prikazan primer MFT unosa kada su atributima dodeljna imena i tipovi.

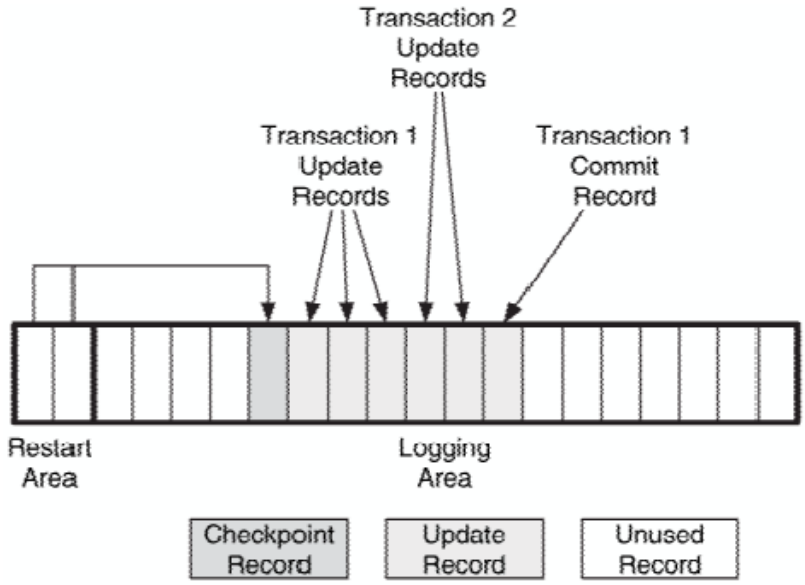


Slika 11: Primer MFT unosa kada su atributima dodeljena imena i tipovi

3.2.4. Journaling u NTFS fajl sistemu

Kako bi se povećala pouzdanost fajl sistema, Microsoft je dodao journaling u NTFS fajl sistem. Ovaj mehanizam predstavlja mehanizam u kome fajl sistem pre nego što izvrši bilo koju promenu na fajl sistemu, pamti podatke u okviru jednog log fajla. Ovaj mehanizam je napravljen sa idejom da ukoliko dođe do pada OS-a prilikom ažuriranja nekog fajla na fajl sistemu, taj fajl može postati korumpirani, ali ukoliko je pre promene upisan u log fajl, fajl sistem može opet da izvrši promene nad fajlom nakon pokretanja operativnog sistema. Lokacija log fajla se nalazi u MFT tabeli na 2 mestu sa imenom $LogFile. Sve promene se čuvaju u okviru $DATA atributa.

Log fajl sadrži dva glavna dela, jedan je restard deo a drugi je log deo. Slika 12 ilustruje $DATA atribute u okviru log fajla u MFT tabeli.



Slika 12: Ilustracija $Data atributa u okviru log fajla u MFT tabeli

Kao što se i vidi sa slike, restart deo sadrži informacije koje pomažu sistemu da se oporavi ako dođe do greške i sadrži pokazivač na poslednju uspešnu izvršenu transakciju koja se nalazi u log delu. Log deo sadži različite tiove zapisa i svaki zapis ima svoj LSN (eng. Logial Sequence Number) koji je jedinstvena 64 bitna vrednost. LSN se inkrementira za svaki novi fajl. Pošto je log deo fiksne dužine ukoliko nema prostora za nove zapise, novi zapisi se pamte na početak. Zbog toga zapisi na početku fajla mogu imati veću vrednost od onih na kraju. Postoji više tipova zapisa od kojih je jedan update zapis koji se najviše koristi za opis transakcije pre nego što se izvrši upis. Veliki broj transakcija zahteva vise od jedno zapisa jer su podeljeni na više manjih operacija i svaka od tih operacija ima svoj zapis. Svaki update zapis pored LSN-a sadrži i informaciju o tome šta će operacija uraditi i drugi deo koji se odnosi šta sistem treba da uradi kako bi poništila ovu operaciju. Ovi zapisi se kreiraju pre nego što se transakcija izvrši i ažuriraju se kada se transakcija izvrši i naziva se commit zapis. Drugi tip zapisa jeste Checkpoint zapis koji govori sistemu odakle treba da krene kako bi verifikovao fajl sistem. Checkpoin zapis se kreira na svakih 5 sekundi i on se pamti u restart delu log fajla. Da bi verifikovao fajl sistem, operativni sistem pronalazi poslednji checkpoint zapis i identifikuje poslednju transakciju koja je započela. Ako je transakcija završena i commit zapis postoji, operativni sistem proverava da li je sadržaj stvarno ažuriran na disku ili je izgubljen u kešu. Ako transakcija nije izvršena i ne postoji commit zapis, operativni sistem vraća sve u stanje pre nego što je transakcija započela.

3.2.5. Analiza NTFS Fajl sistema pomoću pet kategorija

Analizom strukture fajl sistema otkrivamo informacije kako je taj fajl sistem konfigurisan kao i njegov izgled. Kod analize NTFS fajl sistema prvi korak je analiza boot sektora koji je, u stvari, prvi sektor fajl sistema i deo je $Boot fajla. Analizom ovog sektora nalazimo početnu lokaciju MFT tabele kao i veličinu svake vrste MFT tabele. Posedujući ove informacije možemo procesirati prvu vrstu MFT tabele koja se odnosi na $MFT fajl i koja sadrži lokacije ostatka MFT tabele. Ukoliko je bilo koji podataka korumpiran, a znamo veličinu particije, možemo proračunati lokaciju središnjeg sektora fajl sistema gde se nalaze backup kopije prvih vrsta MFT tabele koji se nalaze u $MFTMirr fajlu. Kada smo to pronašli onda možemo da procesiramo $Volume and $AttrDef sistemske fajlove iz kojih dobijamo informaciju o verziji fajl sistema, particiji i specijalnim atributima.

Pod analizom sadržaja podrazumevamo lociranje određenog klastera, određivanje njegovog statusa kao i procesiranje njegovog sadržaja. Pronalaženje nekog klastera je jednostavan posao zato što je prvi klaste na početku fajl sistema i u njemu se nalazi definisana veličina svakog drugog klastera. Status klastera se određuje lociranje $Bitmap fajla i procesiranjem njegovog $DATA atributa. Uobičajena praksa je da se izbaci nealociran proctor iz fajl sistema a to se postiže tako što se ispituje $Bitmap fajl i sadržaj svakog klastera koji je 0 se izbacuje.

Analiza meta podataka se koristi kako bi se dobile više informacija o nekom fajlu ili folderu. Ona obuhvata process lociranja MFT vrste za taj određeni fajl ili folder kao i njeno procesiranje. Da bi pronašli tu vrstu potrebno je da prvo lociramo MFT koristeći početnu adresu u boot sektoru, zatim kada lociramo tabelu onda je na redu da se analiziraju vrste. Neki fajl ili folder može da ima više vrsta zbog svoje veličine tako da je neophodno procesirati listu atributa $ATTRIBUTE_LIST za dodatne adrese koje trebaju da se procesiraju. Prilikom analize vrste

prvo se locira prvi atribut. Analizira se njegovo zaglavlje, određuje se njegov tip i procesira se njegov sadržaj. Sledeci atribut se određuje dužinom prvog atributa i tako se redom dok se ne isprocesiraju svi atributi. Ako želimo neke osnovne informacije možemo odmah da tražimo $STANDARD_INFORMATION atribut i $DATA atribut koji sadrži sadržaj fajla.

Analiza imena fajla se koristi kako bi se fajlovi i folderi pronašli koristeći njihova imena. Ona obuhvata process lociranja foldera, procesiranja njegovog sadržaja i lokacija fajla. Analiza imena fajlova i indeksa kod NTFS-a je komplikovano. Prvo je neophodno locirati root folder koji se nalazi na 5-om mestu u MFT tabeli. Da bi procesirali folder prvo se analizira sadržaj $INDEX_ROOT i $INDEX_ALLOCATION atributa. Ovi atributi sadrže listu indeksa koji se odnose na čvorove u stablu. Svaki indeks predstavlja jedan fajl u okviru tog foldera. Takođe svaki indeks može imati jedan ili više indeksa. Svi ovi indeksi se pamte kao B stablo i moraju biti ponovo sortirani svaki put kada se dodaju novi fajlovi ili obrišu postojeći.

4. DigitalForensics

U ovom poglavlju biće detaljno opisana implementacija aplikacije DigitalForensics za analizu fajl sistema.

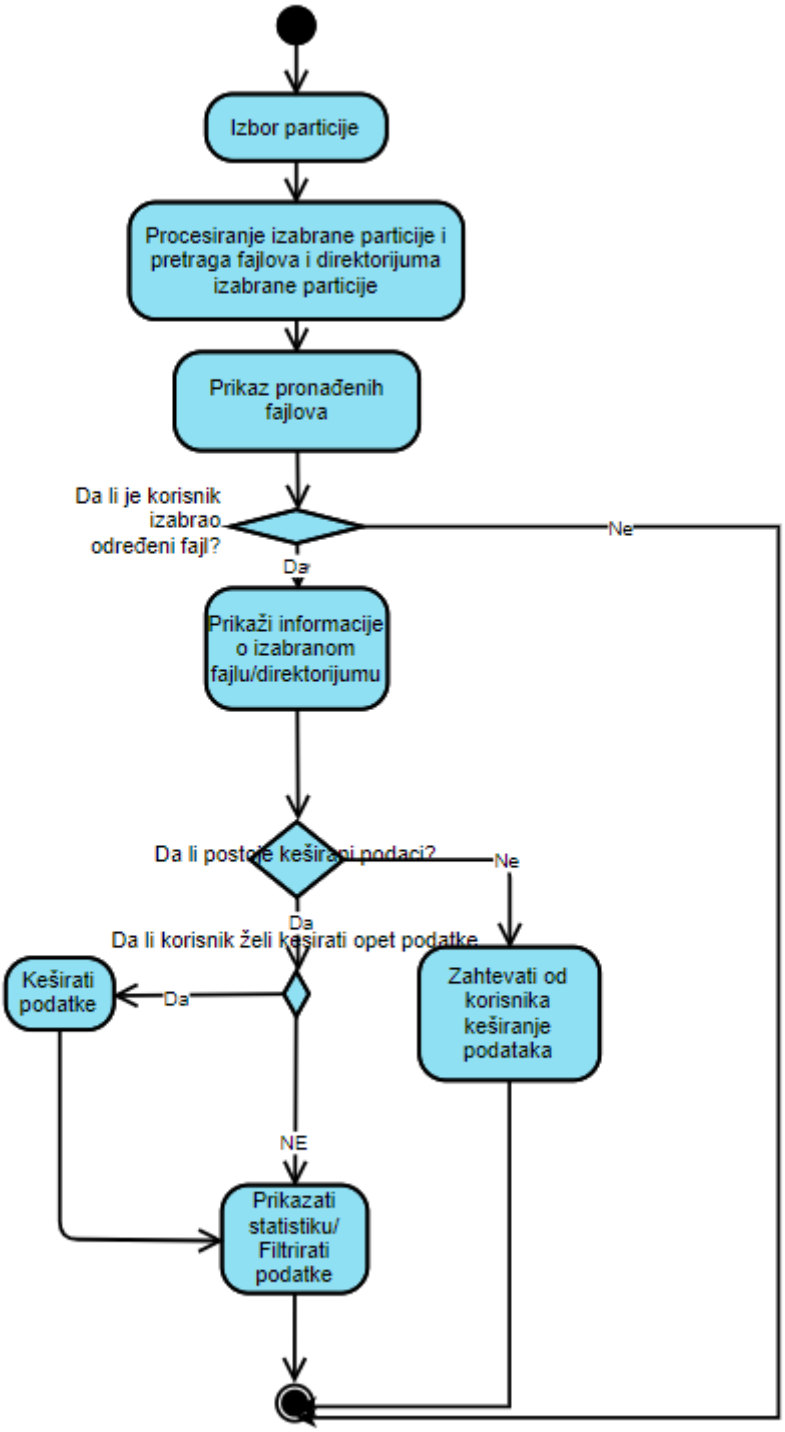
Ova aplikacija predstavlja alat za analizu fajl sistema. Koristi se za prikaz svih fajlova i direktorijuma koji se nalazi na fajl sistemu. Pored prikaza svih fajlova na fajl sistemu ova aplikacija nudi mogućnost filtriranja podataka po različitim kriterijuma, kao što su filtitranje podataka u odnosu na vrednost metapodataka i filtriranje podataka na osnovu vremenskog perioda. Pored filtriranje podataka u odnosu na zadate kriterijume dodata je i mogućnost kreiranje keširanih podataka odnosno čuvanje metapodataka fajlova i direktorijuma radi efikasnijeg pretraživanja. Koriteći keširane podatke moguće je pratiti aktivnosti koji izvršeni nad fajlovima i direktorijumima kroz neki period kao i efikasno pretraživati fajl sistem. Detaljnije o načinu funkcionisanju keširanja podataka kao i o filtriranju aplikacija biće prikazane u sledećim odeljcima.

4.1. Struktura aplikacije

Aplikacija je kreirana u C# programskom jeziku, Windows form aplikacija, a kao alat za keširanja metapodataka o fajlovima i folderima korišćen je ElasticSearch.

Da bismo pokrenuli aplikaciju neophodno je imati ElasticSearch na svom računaru.

Na slici 13, prikazan je UML dijagram aktivnosti aplikacije.



Slika 13: UML dijagram aktivnosti aplikacije

Nakon što se pokrene aplikacija, korisniku se nudi mogućnost odabira particije za koju želi da izvriši analizu. Odabirom željene particije u pozadini se pretražuje izabrana particija i prikazuje se korisniku svi pronađeni fajlovi kao i direktorijumi u vidu liste. Ukoliko korisnik izabre pronađeni fajl ili direktorijum iz liste fajlova i direktorijuma, korisniku će se prikazati metapodaci izabranog fajla ili pak direktorijuma. Da bi se prikazala statistika o određenom

fajlu ili direktorijumu neophodno je imati keširane podatke o izabranoj particiji. Razlog ovakvog uslova jeste pre svega brzina. Elasticsearch se korisi za brzo pretraživanje informacija, i samim tim ukoliko korisnik želi da vidi statistiku o izabranom fajlu ili direktorijumu neophodno je posedovati keširanu vrednost za taj fajl, odnosno keširanu vrednost odabrane particije radi bržeg pretraživanja informacija o izabranom fajlu ili direktorijumu. Ukoliko ne poseduje keširanu vrednost korisniku se neće prikazati statistika, prikazaće se obaveštenje o neuspeloj statistici. Ukoliko ne postoji keširana vrednost podataka o izabranoj particiji, korisnik će morati sam da pokrene keširanje same particije nakon koga je moguć prikaz statistike fajlova. Nakon ispitivanja statusa keširanih podataka i keširanje samih podataka moguće je pretraživati i filtrirati podatke o određenoj particiji.

4.2. Proces keširanja

Prilikom izrade aplikacije osnovni problem koji je trebao biti rešen jeste efikasno pretraživanje i sam prikaz informacija korisniku u što kraćem roku, kao i smeštanje podataka potražujući što manji memorijski prostor. Rešenje ovog problema leži u keširanju podataka fajl sistema.

U poglavlju 4.1. kada smo opisivali dijagram aktivnosti aplikacije pomenuli smo keširanje podataka. Ideja keširanja podataka jeste pamćenje svih bitnih metapodataka fajlova i direktorijuma kako bi se, pre svega, podaci lakše pretraživali a zatim lakše analizirati promene nad fajlom ili direktorijom za neki dati period. Takođe rekli smo da smo korsitili Elasticsearch za pamćenje informacija o fajlovima i direktorijuma fajl sistema.

Elasticsearch je pretraživač zasnovan na biblioteci Lucene. Pruža distribuiranu, multiklijentski, full-text pretraživački engin sa HTTP web interfejsom i JSON dokumentima bez šeme. Elasticsearch je razvijen u Javi. Elasticsearch je dostupan u Java, .NET, PHP, Python i drugim programskim jezicima. U odnosu na engin-e baza podataka Elasticsearch je najpopularniji i najbrži pretraživač podataka.

U osnovi Elasticsearch je deo Elastic Stack-a. Elastic Stack predstavlja zbirku 4 open-source projekta:

1. Elasticsearch

2. Logstash – deo Elastic Stack-a koji ima ulogu u prikupljanju podataka iz različitih izvora, izvršavanje različitih transformacija nad prikupljenim podacima.

3. Kibana – predstavlja sloj vizuelizacije kod Elastic Stack-a, pružajući korisnicima mogućnost analize i vizuelizacije podataka

4. Beats. – predstavljaju agente čija je uloga prikupljanje različitih vrsta podataka koji se kasnije prosleđuju u Elastic Stack.

Ova 4 projekta zajedno se najčešće koriste za nadgledanje, reštavanje problema i zaštitu IT okruženja kao i kod poslovne inteligencije i Web analitike. Beats i Logstash brinu o prikupljanju i obradi podataka, Elasticsearch indeksira i skladišti podatke, a Kibana pruža korisnicima korisnički interfejst za postavljanje upita nad podacima i njihovu vizuelizaciju.

Slika 14: Prikaz Elastic Stack-a

Za potrebe projekta koristili smo samo jedan deo Elastic Stack-a a to je Elasticsearch deo koji nam je potreban za skladištenje podataka. Samu pripremu podataka i samu vizuelizaciju podataka radimo na nivou aplikacije pa je jedini deo Stack-a potreban jeste deo koji će služiti za pamćenje podataka.

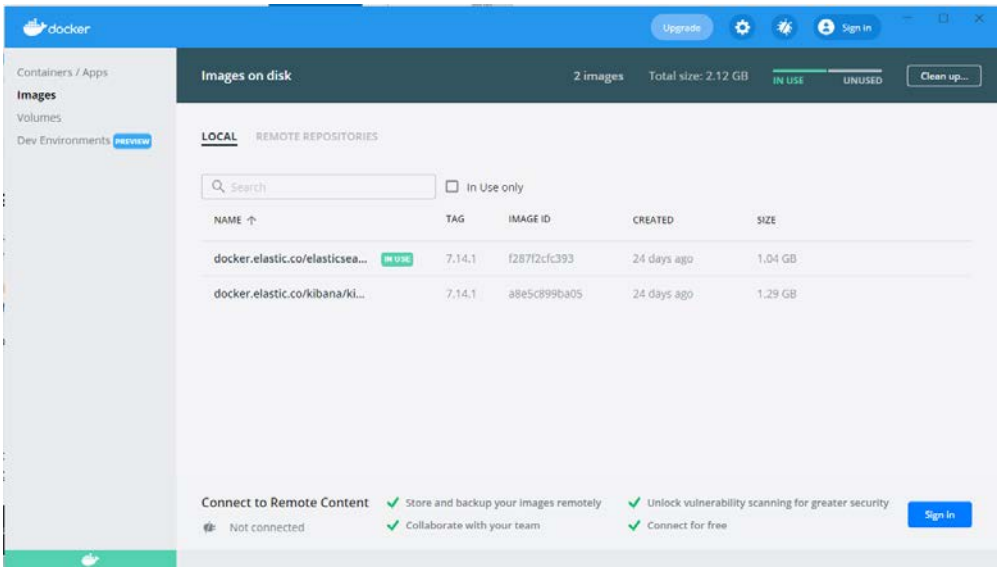
Podaci u Elasticsearch-u se pamte na klasterima koji sadrži više čvorovima u kojima se pamte podaci. Svaki čvor pamti podatke u indeksima koji predstavlja zbirku dokumenata koje poseduju slične karakteristike. Dokument predstavlja osnovnu jedinicu informacija koja može biti indeksirana. Dokument je nista drugo nego JSON objekat.

Razlog korišćenje Elasticsearcha jeste brzina izvršavanja upita za zadati podatak, pronalazi informacije o podacima u milisekundama u struktuiranim i nestruktuiranim dataset-ovima kao i generisanje najviše relevatnih resultat upita nad podacima.

Da bi korisnici mogli da keširaju podatke neophodno je preuzeti i konfigurisati Elasticsearch. Elasticsearch takođe postoji kao i Elasticsearch servis na AWS, GCO I Azure Cloud-u, ali pamćenje informacija o fajl sistemu na nekoj drugoj platformi i ako je bezbednost Cloud-a i dalje na visokom nivou i nije rešenje. Zbog privatnosti fajl sistema, ideja jeste preuzeti Elasticsearch i skladištiti informacije lokalno. Elasticsearch možete preuzeti sa sledećeg sajta (https://www.elastic.co/guide/en/elasticsearch/reference/7.14/install-elasticsearch.html) ili možete preuzeti kao Docker „sliku“ (eng. Images). Tok izrade aplikacije Elasticsearch smo preuzeli i konfigaricali preko Docker image.

Naredba za preuzimanja Elasticsearch image-a data je u nastavku.

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:7.14.1
```



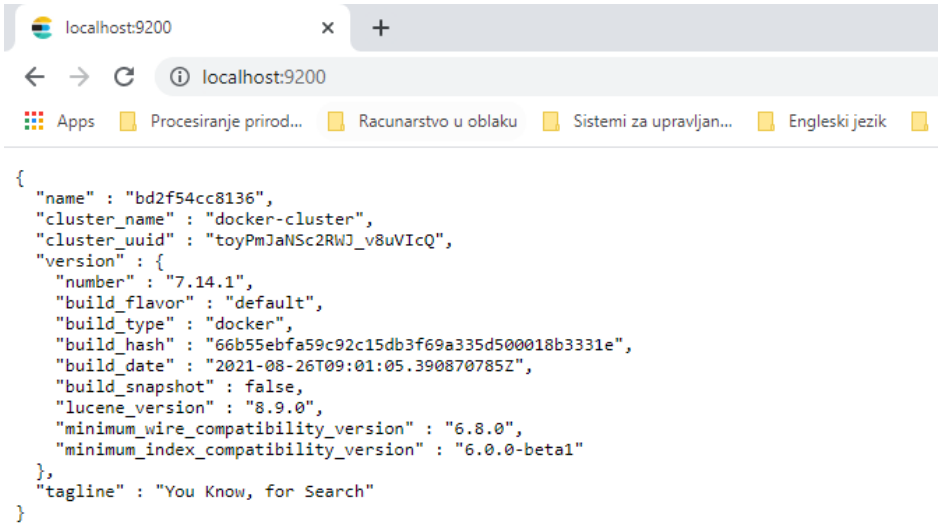
Slika 15:Prikaz Elasticsearch image u doker-u

Prilikom korišćenja i izrade aplikacije koristili smo single-node Elasticsearch klaster za pamćenje i pretraživanje podataka.

Da bi smo pokrenuli Elasticsearch klaste potrebno je izvršiti sledeću komandu u command prompt-u:

```
docker run -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node"
docker.elastic.co/elasticsearch/elasticsearch:7.14.1
```

Da bismo proverili da li je pokrenut Elasticsearch servis, potrebno je u pretraživači preitupiti adresi localhost:9200 koji je u stvari Elasticsearch servis, slika 16.

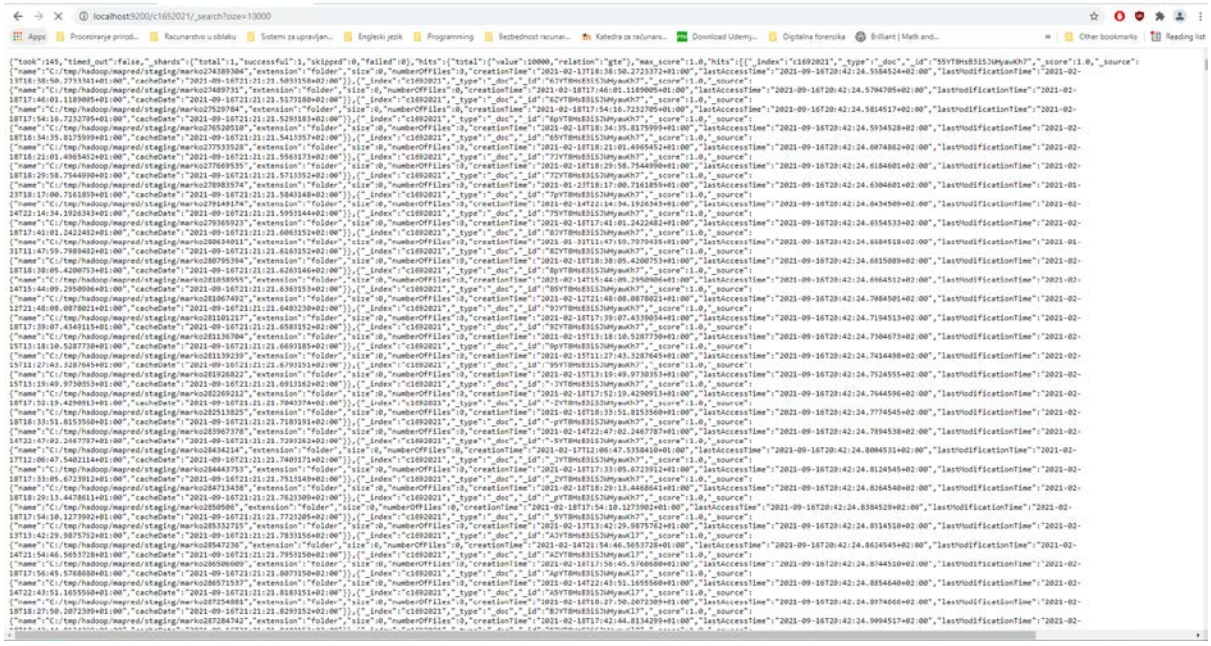


Slika 16: Provera Elasticsearch servis-a

Keširanje se vrši na korisnički zahtev. Zbog samog slobodnog prostora na korisnikovom računaru, odnosno na Elasticsearch servisu, ovaj način keširanja pruža korisnicima samostavlno obavljanje keširanja. Ukoliko korisnik ne obavi keširanje podataka aplikacija će samo služiti za prikaz fajlova na fajl sistemu.

Kao što smo i rekli, ideja keširanja jeste da se pretraži ceo fajl sistem i za svaki fajl, odnosno direktoriju, sačuvaju podaci o putanje fajla ili datoteke, zatim ekstenziji, veličini, broj fajlova, ukoliko se radi o direktorijumu, vremenu kreiranja fajla ili datoteke, vremenu poslednje modifikacije i vreme poslednjeg pristupa. Ovi podaci o fajlovima i direktorijumima fajl sistema se pamte u okviru liste koja prilikom keširanja iniciraju kreiranju jedinstvenog indeksa na Elasticseach kalsteru u kome će biti smešteni dokumenti odnosno podaci iz liste.

Izgled keširanog podatka je dat na slici 16.

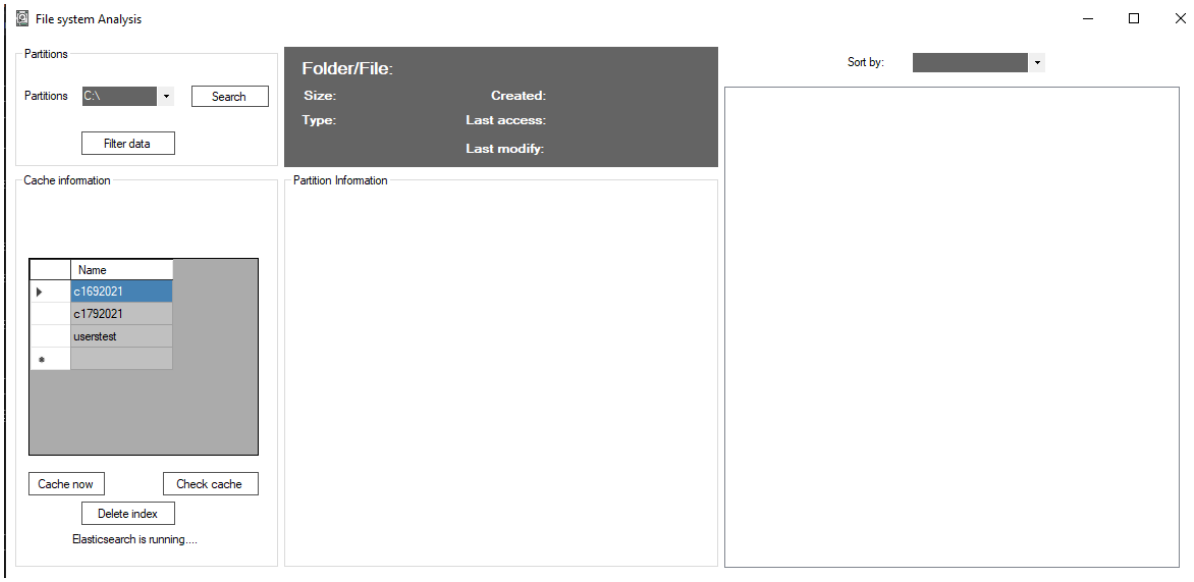


Slika 16: Prikaz keširanih informacija

Pored kreiranja korisniku je i data mogućnost brisanja kreiranog keša.

4.3. Funkcionalnost aplikacije

Na slici 17 je prikazan korisnički interfejs aplikacije DigitalForensics odnosno glavna forma same aplikacije.



Slika 17: Prikaz korisničkog interfejsa nakon pokretanja aplikacije

Aplikacija se sastoji iz sledecih delova:

1. Partitions deo – sastoji se iz izbora particije, kao i dva dugmeta koji služe za pretraživanje i filtriranje izabrane particije

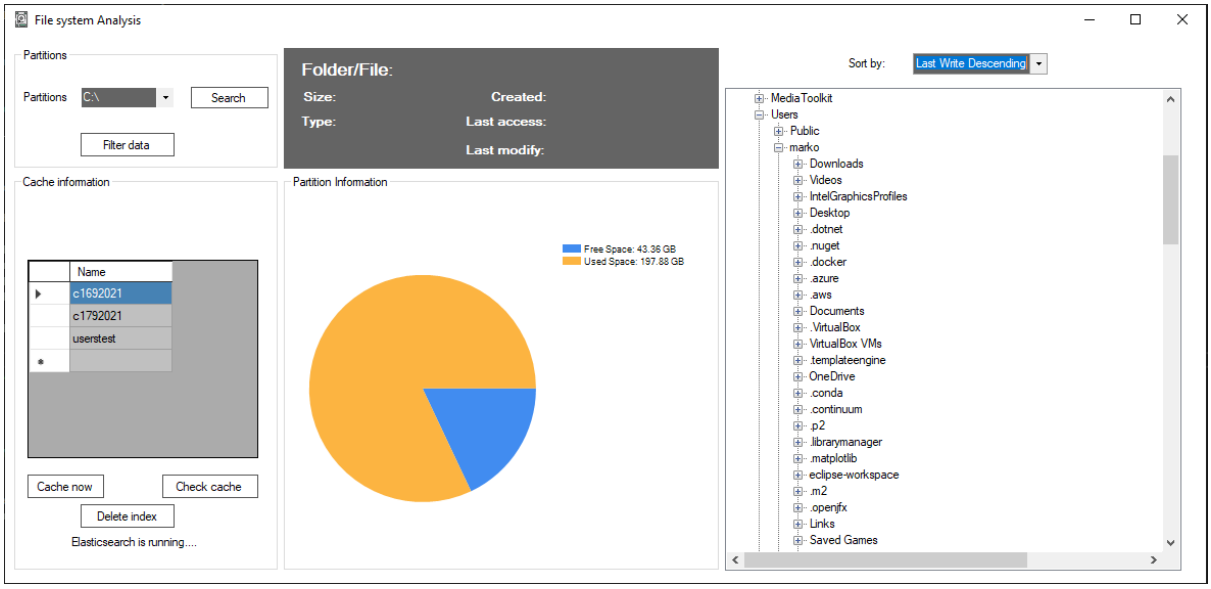
2. Cache information deo – sastoji se iz DataGridView-a koji prikazuje sve indekse koji postaje u Elasticsearch-u, kao i tri dugmeta koji služe za kreiranje keš podataka (eng. Cache now), zatim provera indeksa u Elasticsearch-u (Ceck cache) i brisane selektovanog indeksa (Delete cache)

3. Partition information deo – služi za prikaz informacije o izabranoj particiji, preciznije prikazuje zauzetost memorije selektovane particije

4. Panel deo – informacija o selektovanom fajlu ili direktorijumu

5. Three view deo – prikaz fajlova i direktorijuma izabrane particije. Pored prikaza korisnicima se nudi i mogućnost sortiranja rezultata pretrage. Rezultate je moguće sortirati prema imenu, po velicini, po broju fajlova ukoliko se radi o direktorijumu, po datumu kreiranja, po vremenu poslednje modigikacije i po vremenu poslednjeg pristupa.

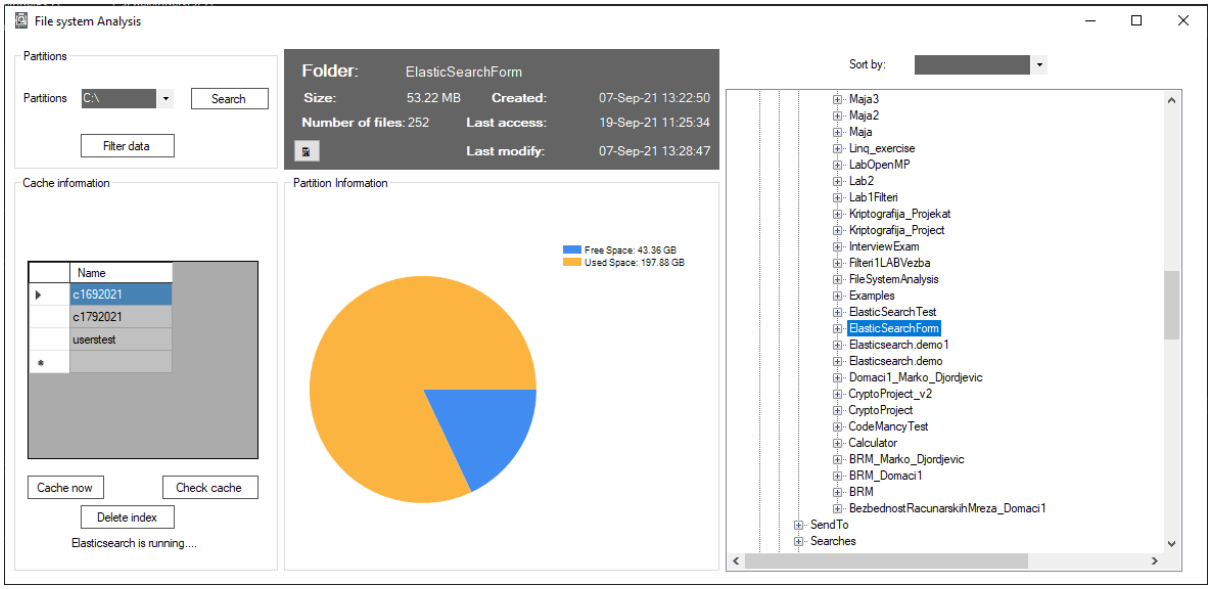
Nakon izbora particije i klikom na dugme Search, prikazuje se korisnička forma koja izgleda kao na slici 18.



Slika 18: Prikaz korisničke forme nakon iniciranja poziva za pretraživanja odgovarajuće particije

Pretraživanje fajlova i direktorijuma se vrsi korišćenjem klasa DirectoryInfo i FileInfo koji nudi .Net framework.

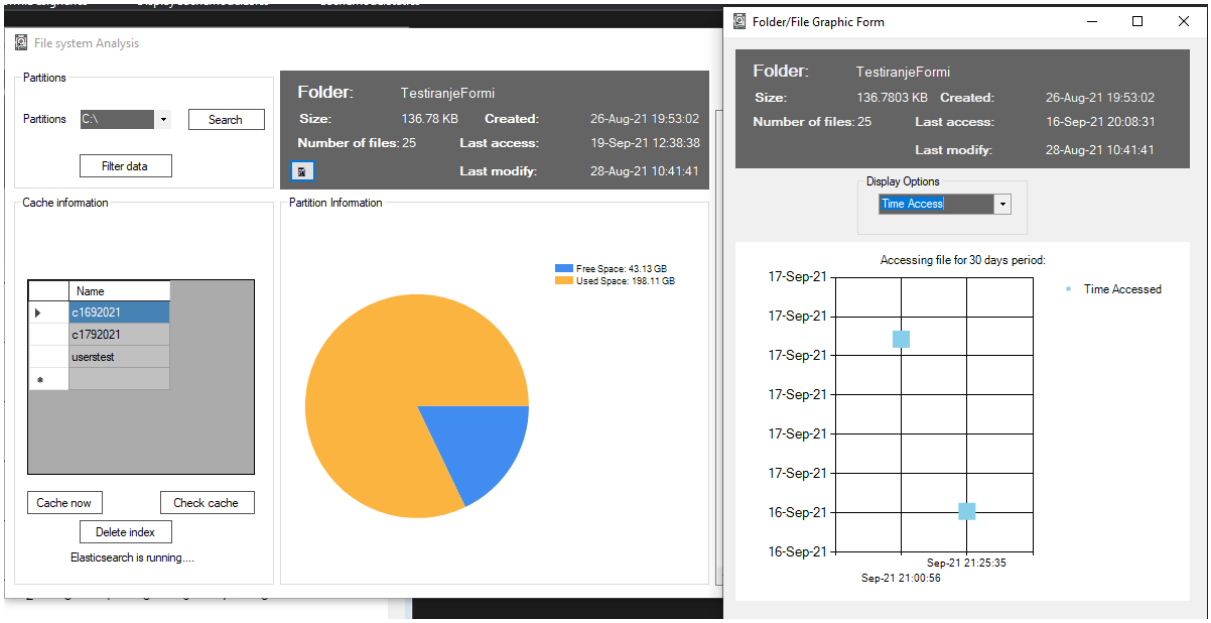
Kao što sa slike 18 možemo videti u Partition information delu prikazuju se informacija o izaranoj particiji, preciznije o iskorišćenom memorijskom prostoru particije. U Three view delu prikazuje se informacija o direktorijumima i fajlovima izabrane particije. Klikom na bilo koji fajl ili direktorijum sa korisničke forme prikazaće se informacija o izabranom fajlu ili direktorijumu u Panel delu, slika 19 .



Slika 19: Korisnički interfejs nakon odabira direktorijuma ili fajla iz Three view dela

Kao što i sa slike 19 možemo videti korisniku se nudi mogućnost prikaza statistke o izabranom direktorijumu ili fajlu. Informaciju možemo videti klikom na dugme koje se nalazi u Panel delu.

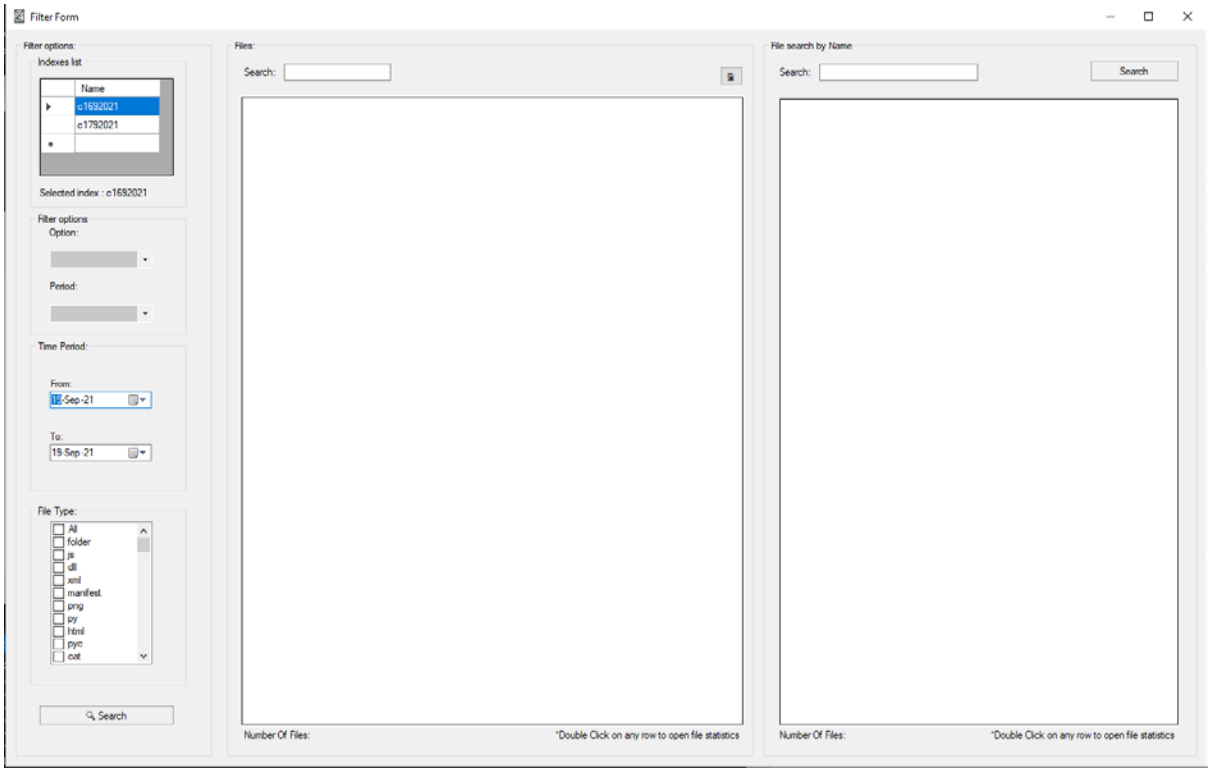
Treba voditi računa prilikom prikaza statistike. Potrebno je sačekati sve informacije o izabranom fajlu ili diretkorijumu da bi se rezultat prikazao u korisničkoj formi Folder/File Graphic Form. Razlog čekanja svih potrebnih informacija o direktorijumu ili fajlu jeste zbog pretraživanja same statistike. Kao što smo i rekli, informacije koje se pamte na Elasticsearch-u su podaci o putanje fajla ili datoteke, zatim ekstenziji, veličini, broj fajlova, ukoliko se radi o direktorijumu, vremenu kreiranja fajla ili direktorijuma, vremenu poslednje modifikacije i vreme poslednjeg pristupa, a pošto Elasticsearch pamti dokumente kao običan tekst i Elasticsearch je full-text engin pa samo pretraživanje može dovesti do greške odnosno da se prikažu rezultati za drugi direktorijum ili fajl jer prilikom pretraživanja uzmimamo u obzir sve parametre određenog fajla ili direktorijuma.



Slika 20: Prikaz statistike za izabrani fajl ili direktorijum

Statistiku prikazuje promenu metapodataka izabranog fajla ili direktorijuma u periodu od 30 dana. Moguće je prikazati statistiku izabranog fajla u pogledu veličine, broja fajlova, poslednjeg pristupa fajla ili poslednje modifikacije fajla u roku od 30 dana.

Glavna i najvažnija forma ove aplikacije jeste Filter forma i prikazana je na slici 21 .



Slika 21: Prikaz filter forme

Sa slike možemo videti da se Filter forma sastoji iz sledećih delova:

1. Filter options deo

2. Files deo – služi za prikaz pronađenih fajlova i direktorijuma u keš podacima

3. File search By Name deo – služi za pretragu fajlova/direktorijuma po imenu u keš podacima

Filter options deo se koristi za odabir parametara za pretraživanje podataka. Sastoji se iz sledećih delova:

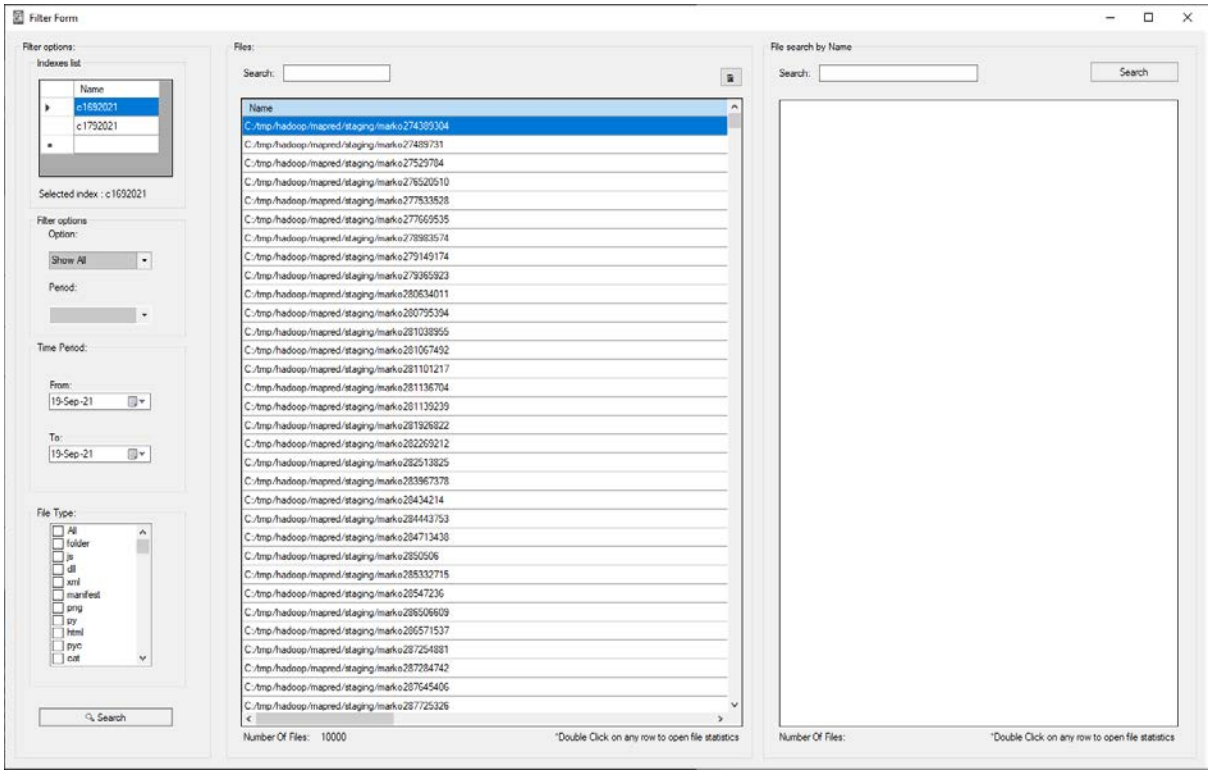
1. Indexes list – prikazuje listu indeksa keširanih podataka i selekcijom odgovarajućeg indeksa iniciramo pretraživanje tog keš indeksa.

2. Filter options – služi za odabir parametara za pretraživanje podataka. U padajućem meniju Options biramo da li želimo da prikažemo sve fajlove, da li želimo da prikažemo fajlove po vremenu kreiranja, po vremenu modifikacije ili vremenu pristupa. Zatim, u padajućem meniju Period biramo po kom vremenskom kriterijumu ćemo prikazati podatke, da li ćemo konkretno za dan, za perion ili za period od 30 dana.

3. Time period – ova sekcija služi za odabir vremenskog kriterijuma. Sastoji se iz dva data time picker-a, From i To, koji služe za podešavanja vremenskog perioda .

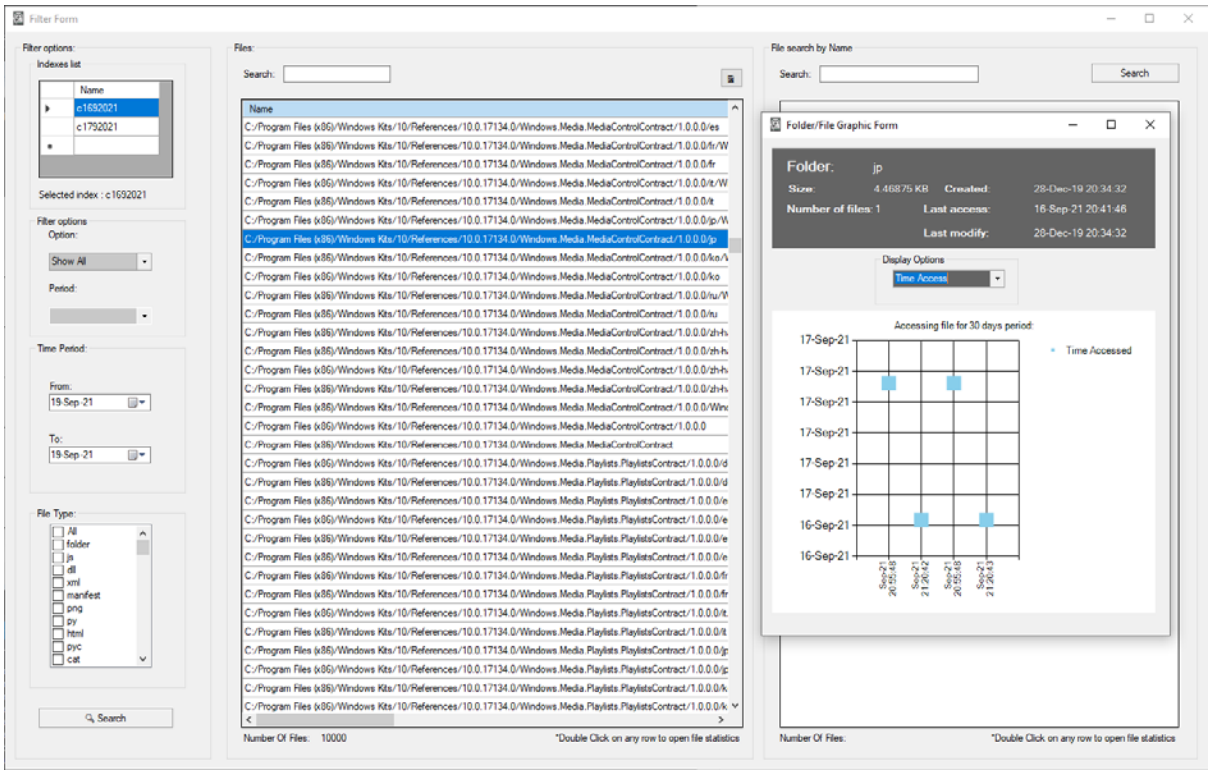
4. File Type – sekcija koja nam služi za odabir ekstenzije fajlova koje pretražujemo.

Klikom na dugme Search inicira se upit koji će vratiti iz Elasticsearch-a informacije o fajlovima ili direktorijumima za zadati kriterijum. Primer je prikazan na slici 22.



Slika 22: Primer pretrage keš podataka

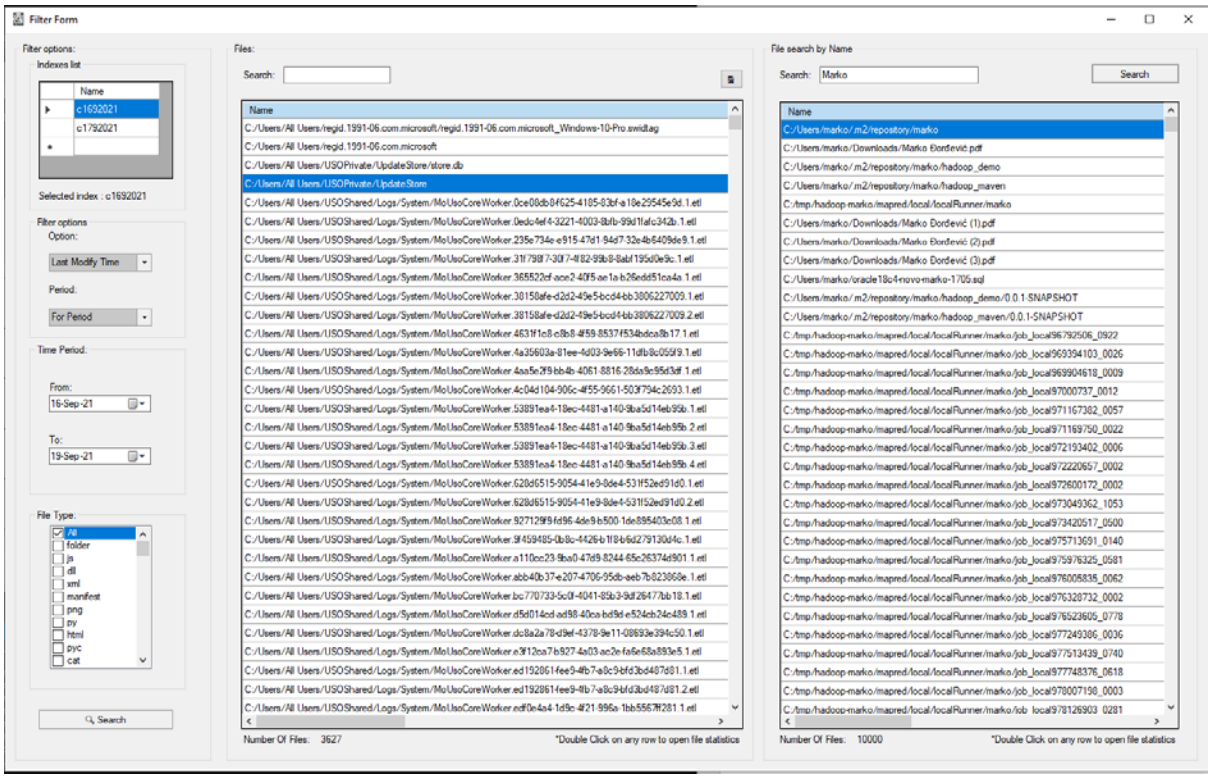
Sa slike 22 možemo videti da je rezultat prikazan u sekciji Files i dvostrukim klikom na bilo koji izabrani fajl prikazaće se statistika o izabranom fajlu odnosno direktorijumu.



Slika 23: Prikaz statistike izabranog fajla ili direktorijuma

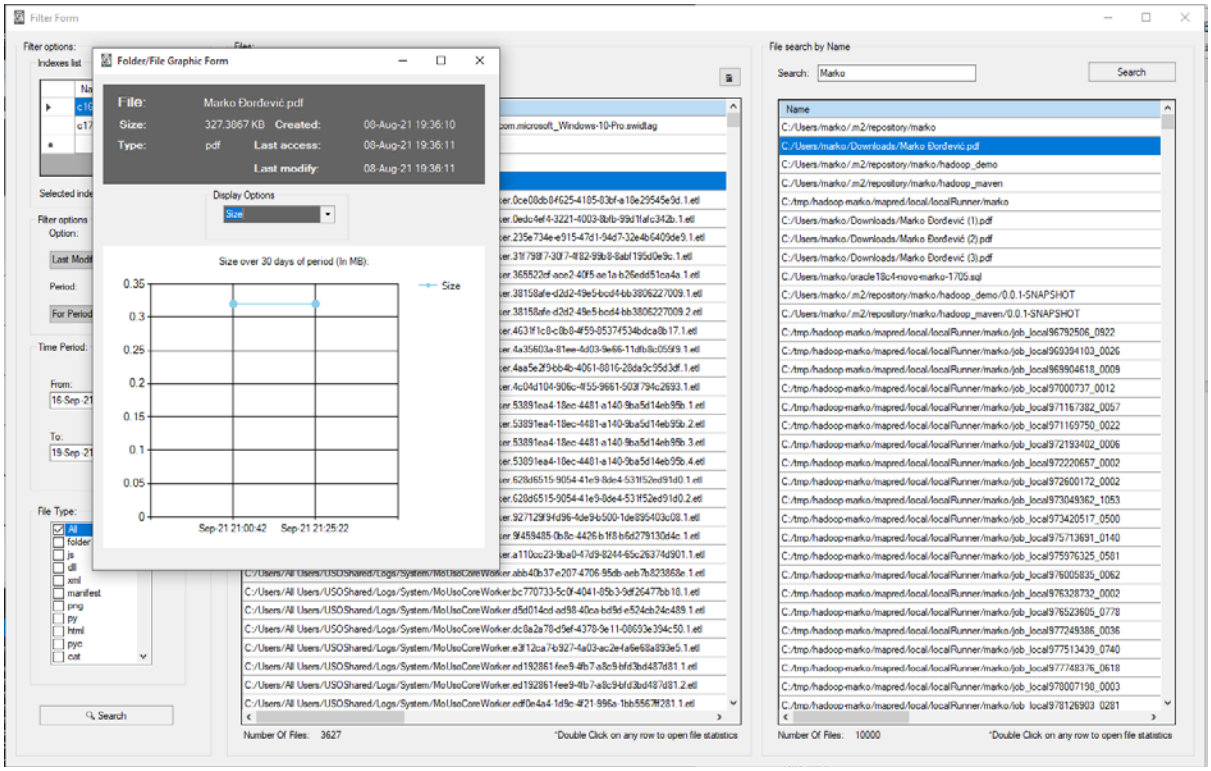
Ipak prilikom pretraživanja postoji ograničenje. Elasticsearch prilikom pretraživanja korisnicima dostavlja prvih 10000 pronađenih podataka.

Ukoliko želimo da pretražujemo samo po imenu fajlova ili direktorijuma, omogućeno je pretraživanje u sekciji File search By Name. U ovoj sekciji potrebno je uneti samo teks fajla ili foldera i prikazaće korisnicima rezultat pretrage. Ukoliko želimo da vidimo statistiku pronađenog fajla moguće je dvoklikom na fajl videti statistiku pronađenog fajla. Primer pretraživanja po imenu fajla ili direktorijama dat je na slici 24.



Slika 24: Prikaz pretraživanja fajlova ili direktorijuma po imenu.

Kao što smo i rekli, dvoklikom na bilo koji rezultat pretrage prikazuje se statistika određenog fajla, slika 25.



Slika 25: Prikaz rezultata statistike izabranog fajla

5. Zaključak

U ovom seminarskom govorili smo o analizi fajl sistema, sa posebnim akcentom na NTFS fajl sistem. Takođe prikazali smo aplikaciju DigitalForensics koja služi za analizu fajl sistema.

Prilikom rada na seminarskom radu nailazili smo na bezbroj alata koji se koristi za analizu fajl sistema. Neki od alata su besplatni dok neki se pak plaćaju ali svi alati imaju isti cilj a to je prikazivanje svi informacija koje se nalaze na fajl sistemu. Razlika između alata jeste samo u funkcionalnostima koje nude.

Kao prvi predstavnik najpoznatijeg alata za analizu fajl sistema jeste WinDirStat. On je najstariji i najrasprostanjeniji alat za analizu sistema. Veoma je jednostavan. Ovaj softver za analizu fajl sistema pokazuje pored potrošnje podataka, se koristi kao alat za čišćenje (eng. Cleanup) različite verzije Windows operativnog sistema. Pored prikaza fajl sistema u obliku stabla, nudi i grafički prikaz strukture fajl sistema. Svaki fajl i folder su predstavljeni kao pravougaonik i u zavisnosti od ekstenzije taj pravougaonik ima neku boju. Što je procenat koji folder odnosno fajl zauzima to je i pravougaonik veći i biće više vidljiviji u grafičkom prikazu. Kao kritika ovog alata jeste vreme obrade fajl sistema.

Kao drugi predstavnik je SpaceSniffer alat koji u poređenju sa WinDirStat alatom ne prikazuje stablo fajl sistema već grafički prikazuje stanja fajl sistema, poput WinDirStat-a. Prilikom obrade željenih particija korisniku se svo vreme na ekranu iscrtava fajl sistem. Kao i kod WinDirStat-a svaki pravougaonig predstavlja neki direktorijum ili fajl. U poređenju sa WinDirStat alatom dosta je brži i ima bolji grafički prikaz, ali u poređenju sa informacijama WinDirStat je i dalje u prednosti.

TreeSiza alat je najmlađi alat među svim alatima za analizu fajl sistema. Kao i svi do sad i TreeSize na početku nudi opciju biranja particije, ali je moguće i izbor samo konkretnog foldera. Skeniranje kod TreeSize-a je veoma brzo, jer kako navodi kompanija koja je vlasnik ovog alata oni za skeniranje koriste MFT tabelu, a ako nema pristup MFT tabeli onda se skeniranje vrši u dva paralelna treda za free verziju i do tridesetdva treda kod plaćene verzije TreeSiza-a. U odnosu na WinDirStat alata pruža jako više informacija, ali je sam prikaz informacija loš u odnosu na WinDirStat.

Ipak pored navedenih aplikacija, aplikacija DigitalForensics predstavlja poboljšanje WinDirStat-a u pogledu pretraživanja samog sistema. Mogućnost keširanja podataka značajno ubrzavamo vreme pretrage fajl sistema kao i pregled statistike metapodataka fajlova odnosno direktorijuma. Ovo je inicijalna verzija aplikacije, planira se usavršavanje aplikaciju u pogledu funkcionalnosti (planira se prikaz obrisanih fajlova na fajl sistemu) i usavršavanje postojećih funkcionalnost.