**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ RAČUNARSTVA**

Petar Lesar

JAVAFX DESKTOP APLIKACIJA ZA ARHIVIRANJE DATOTEKA U OBLAKU

ZAVRŠNI RAD br.1469

Zagreb, rujan, 2022.

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ RAČUNARSTVA**

Petar Lesar

JMBAG: 0246087879

JAVAFX DESKTOP APLIKACIJA ZA ARHIVIRANJE DATOTEKA U OBLAKU

ZAVRŠNI RAD br. 1469

Zagreb, rujan, 2022.

# **Sažetak**

Cilj rada je izraditi grafičko sučelje za pohranu podataka na nekom servisu u oblaku. Za taj zadatak će se koristiti Java programski jezik i njegovo grafičko sučelje *JavaFX*. Za pisanje koda ću se koristiti *InteliJ*-om, a za izradu grafičkog sučelja ću koristiti *SceneBuilder*. Pomoću *DropBox* API-a će se uspostavljati veza s oblakom i vršit će se razne funkcije nad datotekama i direktorijama koji su spremljeni u oblaku. Poželjno je omogućiti što više raznih funkcionalnosti nad datotekama i direktorijima, te prikazati sve datoteke i direktoriju u tome prozoru.

Ključne riječi: DropBox, JavaFX, Java programski jezik, grafičko sučelje

Sadržaj

[**1.** **Uvod** 5](#_Toc112931318)

[**2.** **Dropbox** 6](#_Toc112931319)

[**2.1.** **Povijest Dropbox-a** 6](#_Toc112931320)

[**2.2.** **Dropbox danas** 6](#_Toc112931321)

[**3.** **Java** 8](#_Toc112931322)

[**3.1.** **Povijest Jave** 8](#_Toc112931323)

[**3.2.** **JavaFX** 8](#_Toc112931324)

[**3.2.1.** **JavaFX struktura** 8](#_Toc112931325)

[**3.2.2.** **SceneBuilder** 9](#_Toc112931326)

[**4.** **Pregled korištenih tehnologija** 10](#_Toc112931327)

[**4.1.** **Dropbox SDK** 10](#_Toc112931328)

[**4.1.1.** **Primjer osnova** 10](#_Toc112931329)

[**4.2.** **TreeView** 13](#_Toc112931330)

[**5.** **Programski kod** 14](#_Toc112931331)

[**5.1.** **MainApplication** 14](#_Toc112931332)

[**5.2.** **Main Controller i Token Window** 14](#_Toc112931333)

[**5.3.** **Files Window Controller** 16](#_Toc112931334)

[**5.3.1.** **UploadFile** 17](#_Toc112931335)

[**5.3.2.** **UploadFolder** 17](#_Toc112931336)

[**5.3.3.** **CreateFolder** 18](#_Toc112931337)

[**5.3.4.** **Refresh** 18](#_Toc112931338)

[**5.3.5.** **AutoFolder** 19](#_Toc112931339)

[**5.3.6.** **Izrada stabla** 19](#_Toc112931340)

[**5.3.7.** **CellFactory** 20](#_Toc112931341)

[**5.4.** **AutoFolder** 29](#_Toc112931342)

[**5.5.** **DropBoxService** 31](#_Toc112931343)

[**5.5.1.** **DeleteFile** 32](#_Toc112931344)

[**5.5.2.** **GetFolder** 33](#_Toc112931345)

[**5.5.3.** **GetFile** 33](#_Toc112931346)

[**5.5.4.** **RenameFile** 34](#_Toc112931347)

[**5.5.5.** **MoveFile** 34](#_Toc112931348)

[**5.5.6.** **CopyFile** 35](#_Toc112931349)

[**5.5.7.** **CreateFolder** 35](#_Toc112931350)

[**5.5.8.** **DownloadFolder** 36](#_Toc112931351)

[**5.5.9.** **UploadFile** 37](#_Toc112931352)

[**5.5.10.** **UploadFolder** 38](#_Toc112931353)

[**5.5.11.** **UploadFolderAuto** 40](#_Toc112931354)

[**5.5.12.** **UploadFileAuto** 40](#_Toc112931355)

[**5.5.13.** **DeleteFileAuto** 40](#_Toc112931356)

[**5.6.** **MoveWindowController** 40](#_Toc112931357)

[**5.7.** **CopyWindownController** 41](#_Toc112931358)

[**6.** **Zaključak** 42](#_Toc112931359)

[**Literatura** **Pogreška! Knjižna oznaka nije definirana.**](#_Toc112931360)

Popis oznaka i kratica

AWT Abstract Window Toolkit

API Application Programming Interface

FXML FX Markup Language

SDK Software development kit

# **Popis Tablica**

Tablica 1. Prikaz planove koje nudi DropBox 7

# **Popis Slika**

[Slika 1. Hijerarhija početne aplikacije 9](#_Toc113131530)

[Slika 2. Prikaz ScenBuilder programa 9](#_Toc113131531)

[Slika 3. Dependency za DropBox SDK 10](#_Toc113131532)

[Slika 4. Provjera tokena i spajanje u oblak 11](#_Toc113131533)

[Slika 5. Ispis datoteka u oblaku 12](#_Toc113131534)

[Slika 6. Učitavanje datoteke 12](#_Toc113131535)

[Slika 7. Preuzimanje datoteke 13](#_Toc113131536)

[Slika 8. Prikaz MainApplication koda 14](#_Toc113131537)

[Slika 9. MainController klasa 15](#_Toc113131538)

[Slika 10. Prozor za generiranje tokena 16](#_Toc113131539)

[Slika 11. Grafički prikaz prazne aplikacije 17](#_Toc113131540)

[Slika 12. UploadFile funkcija 17](#_Toc113131541)

[Slika 13. UploadFolder funkcija 18](#_Toc113131542)

[Slika 14. CreateFolder i Refresh funkcije 18](#_Toc113131543)

[Slika 15. AutoFolder funkcija 19](#_Toc113131544)

[Slika 16. CreateTree funkcija 19](#_Toc113131545)

[Slika 17. DisplayTreeView funkcija 20](#_Toc113131546)

[Slika 18. Brisanje i preuzimanje direktorija kod 22](#_Toc113131547)

[Slika 19. Brisanje i preuzimanje datoteke kod 23](#_Toc113131548)

[Slika 20. Premještaj datoteke kod 24](#_Toc113131549)

[Slika 21. Kopiranje datoteke kod 25](#_Toc113131550)

[Slika 22. Promjena naziva i datum kod 26](#_Toc113131551)

[Slika 23. Promjena direktorija kod 29](#_Toc113131552)

[Slika 24. AutoFolder konstruktor kod 30](#_Toc113131553)

[Slika 25. Run metoda AutoFolder 31](#_Toc113131554)

[Slika 26. Getter i setter klijenta kod 32](#_Toc113131555)

[Slika 27. Brisanje datoteke kod 33](#_Toc113131556)

[Slika 28. Odabir datoteke kod 33](#_Toc113131557)

[Slika 29. Dohvat datoteke kod 34](#_Toc113131558)

[Slika 30. Preimenovanje datoteke kod 34](#_Toc113131559)

[Slika 31. Premještaj datoteke kod 35](#_Toc113131560)

[Slika 32. Kopiranje datoteke kod 35](#_Toc113131561)

[Slika 33. Izrada direkotrija 36](#_Toc113131562)

[Slika 34. Preuzimanje direktorija 37](#_Toc113131563)

[Slika 35. Učitavanje datoteke kod 38](#_Toc113131564)

[Slika 36. Učitavanje direktorija kod 40](#_Toc113131565)

[Slika 37. MovieWindow prozor 41](#_Toc113131566)

# **Uvod**

Pohrana podataka je kroz godine drastično napredovala, od veće memorije za pohranu pa sve do pohranjivanja podataka na internetu. Kako svi žele imati pristup svojim podacima cijelo vrijeme, spremaju ih na neke servise u oblaku. To omogućuje da se može pristupiti nekom dokumentu s više različitih računala ili uređaja bez potrebe premještanja s jednog uređaja na drugi. S vremenom je takvih servisa i usluga postalo sve više i nude razne opcija za spremanje podataka s različitim cijenama za te usluge. Od besplatnih usluga koje nude manje količine memorije pa sve do cijelih tvrtka kojima se podaci nalaze na nekom servisu, svi danas koriste neki oblik spremanja podataka u oblaku. Svaki taj servis ima neke dobre strane, ali i neke lošije u kojima ih neki drugi servis nadmašuje. Stoga ovisno o potrebama korisnika i količini sredstava koja ima na raspolaganju bira koji će odabrati. Neki servisi mogu ponuditi više memorijskog prostora, dok neki možda imaju određene funkcionalnosti koje su potrebne korisniku. Najbolje bi bilo pronaći servis koji nudi dovoljno veliki memorijski prostor i ostale funkcionalnosti nadoknaditi s izradom programa koji ih omogućava. Za izradu programa potrebno je i pronaći servis koji nudi izradu programa pomoću njihovog servisa. Istražujući sve poznatije servisa za pohranu podataka nađen je *DropBox*. On nudi dovoljno veliki memorijski prostor, a ima razvijene API metode za korištenje u nekom izrađenom programu. Program neće imati nikakve potrebe za više korisnika, nego će biti samo za osobnu upotrebu. *DropBox* ima ponudu za izradu programa u više programskih jezika, ali je odlučeno za *Javu*. Program će biti *desktop* aplikacija, a ne da se koristi u nekom pregledniku ili slično. Java se u tome čini kao najbolji odabir jer nudi izradu programa s grafičkim sučeljem i vrlo moćnim programskim jezikom. Cilj je izraditi program u kojem se lako snalazi i nudi osnovne operacije nad datotekama koje bi mogle zatrebati.

# **Dropbox**

*Dropbox* je online sustav za pohranu podataka u oblaku. Služi za lakši dohvat i spremanje podataka. Nalazi se u oblaku što znači da je uvijek povezani s internetom i ako postoji pristup internetu postoji i pristup podacima na *Dropbox*-u. Onje potpuno besplatan za korištenje, ali s ograničenim prostorom za pohranu.

## **Povijest Dropbox-a**

Drew Houston je 2007. godine napravio program s ciljem da jednostavno i lagano korisnici mogu pohranjivati, dijeliti i sinkronizirati datoteke kroz više uređaja. Taj program je nazvao *DropBox*. Prvo izdanje *Dropbox*-a je bilo 2008. kada je iz beta verzije omogućio korištenje za javnost.

## **Dropbox danas**

*Dropbox* je još danas jedan od vodećih sustava za pohranu podataka uz *Google*-ov *Google Disk* i *Microsoft*-ov *OneDrive*. Nakon svih petnaest godina se i dalje drži poslovne politike *freemium*. Znači da se može koristi potpuno besplatno, ali ima ograničenu memoriju koju korisnik može koristit. Naravno, postoje i planovi koji se plaćaju mjesečno ovisno o potrebama koje korisnik ili neka tvrtka treba.

Tablica . Prikaz planove koje nudi DropBox

|  |  |  |  |
| --- | --- | --- | --- |
| **Naziv** | **Memorijski prostor** | **Cijena** | **Značajke** |
| Basic | 2 GB | Besplatno | Bez enkripcije memorije |
| Plus | 2000 GB | 12€ mjesečno | S enkripcijom memorije, 30 dana oporavka podataka |
| Family | 2000 GB | 20€ mjesečno | Nudi sve isto sve što i Plus, ali za šest korisnika i zajednički direktorij |
| Professional | 3000 GB | 17€ mjesečno | Nudi sve što i Plus, nudi veću sigurnost podataka i mogućnost učitavanja većih datoteka |
| DropBox Business | 5000 GB | 15-22€ mjesečno po korisniku tima | Centralizirano i organizirano za timove neke tvrtke, do šest mjeseci oporavka podataka |

# **Java**

Java je objektno orijentirani, klasni programski jezik. Izmislio ju je James Gosling početkom 90-ih godina. Najvažnija razlika koja dijeli Javu od ostalih programskih jezika je način na koji se pokreče programski kod. Java prilikom pokretanja svoj kod kompilira prvo u takozvani *bytecode* koji se onda pokreće koristeći java virtualnu mašinu. Sintaksa koju koristi je slična ostalim poznatim programskim jezicima kao što su C ili C++.

## **Povijest Jave**

Java je započela svoj razvoj 1991. godine s idejom da se napravi jednostavniji programski jezika koji je sličan C-u. Razvoj Jave se odvijao u tvrtki *Sun Microsystems* i 1995. godine je izašla prva verzije Jave zvana „Java 1.0“. Java je jezik koji napreduje pomoću njegove ogromne zajednice koja radi na novim stvarima kako bi sebi olakšali rad u Javi. Te promjene onda tvrtka *Sun* implementira u novu verziju Jave. Godine 2022. je izašla najnovija verzija Java 18, ali planirana je i Java 19.

## **JavaFX**

JavaFX je grafičko korisničko sučelje za javu. Pomoću njega je omogućena jednostavna izrada grafičkih aplikacija u javi. JavaFX se koristi često zbog svoje jednostavnosti za izradu aplikacija. Postoji mnogo alternativnih sučelja kao što je *Swing* ili AWT, ali zbog svoje moći koje pruža JavaFX programeri ga vrlo rado koriste.

### **JavaFX struktura**

*JavaFX* struktura je veoma slična binarnom stablu. Glavni korijen aplikacije je *Stage* koji predstavlja prozor koji korisnik vidi. Tome S*tage*-u pripada u većini slučajeva jedna scena koja je pripadni dio S*tage*-a. U tome *Stage*-u se nalazi neki *Node* koji predstavlja *Container* cijele te scene. Unutar toga se nalaze komponente aplikacije koja se koristi. Za izgled scene u kojoj se nalaze *Node*-ovi postoji FXML datoteka. U njoj je opisan hijerarhijski izgled neke scene. Naravno, ta FXML datoteka sama po sebi nema nikakvu funkciju osim što drži elemente na sceni. Kako bi se dodala ta funkcionalnost elementima ili sceni potrebno je svakoj FXML datoteci odrediti C*ontroller* klasu. *Controller* klasa je dio java koda koji služi za opisivanje funkcionalnosti za svaki element u toj sceni. Može se odrediti što će se dogoditi kad se pritisne na neki gumb ili nešto složenije kao detekcija promjene teksta u nekom elementu. Stoga se početna JavaFX aplikacija koja se automatski generira sastoji od jedne FXML datoteke, Java *Controller*-a i main java klase u kojoj se definira početni *main* *stage* i scena prilikom pokretanja aplikacije.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika . Hijerarhija početne aplikacije

### **SceneBuilder**

SceneBuilder je besplatan i *open* *source* program koji korisniku pomaže pri izradi JavaFX aplikacija da ne bi korisnik morao sam pisati izgled scene u FXML datoteku i kompilirati program kako bi vidio izgled aplikacije. SceneBuilder korisniku omogućava da otvori zadanu FXML datoteku i u njoj uređuje izgled scene. Uz lakše pisanje FXML datoteke još ima puno korisnih značajki. Nudi veliki izbor komponenti koje može koristiti, pokazuje za svaku komponentu postavke i izgled koji se može urediti, te prikazuje hijerarhiju scene koju je jednostavno promijeniti obzirom na preferencije korisnika.

Slika na kojoj se prikazuje tekst, snimka zaslona, računalo, na zatvorenom

Opis je automatski generiran

Slika . Prikaz ScenBuilder programa

# **Pregled korištenih tehnologija**

Za ovaj projekt je bilo potrebno koristiti puno objekti koje se ne nalaze sami po sebi u običnoj javi. Kao prvo, jedan od toga je i SceneBuilder koji treba korisnik sam preuzeti i instalirati. Osim toga potrebno je i korištenje puno vanjskih biblioteka koje se importiraju u programski kod.

## **Dropbox SDK**

Dropbox je vidio da njegovi korisnici žele pokušati i sami izraditi svoje aplikacije koje im omogućavaju upravljanje s datotekama u oblaku te su iz tog razloga osmislili vlastitu SDK platformu. SDK platforma je skupina alata i API-a za programere. Dropbox je napravio par službenih SDK-ova za jezike kao što su Python, JavaScript ili C. Naravno, postoji i SDK za Javu. Takozvani Dropbox *Core* SDK je trenutno na njegovoj drugoj verziji i ima podršku za Javu 11 pa nadalje. Kako bi se uopće moglo upravljati datotekama potrebno je prijaviti se i dobiti dozvolu za rad aplikacije. Nakon izrade aplikacije se dobije neki *access token* koji omogućava upravljanje datotekama pomoću njihovog API-a.

### **Primjer osnova**

U dokumentaciji u kojoj je opisano kako API radi u javi je dano puno primjera kako spojiti aplikaciju ako je rađena za *Android* uređaj ili za neku drugu verziju Java aplikacije. Nalazi se i neki osnovni primjer u običnom Java jeziku kako se spojiti na aplikaciju i neke osnove funkcije kao na primjer kako preuzeti ili učitati datoteku. Aplikacija zahtjeva pristup službenim datotekama od Dropbox-a pa tako aplikacija koja je korištena mora biti Maven Java aplikacija.



Slika . Dependency za DropBox SDK

Nakon što se doda dependency u aplikaciju, nastaje pristup bibliotekama od Dropbox-a. Kako bi se pristupilo baš određenom direktoriju koji je *Dropbox* dodijelio za izradu aplikacije potreban je *access token* koji je *Dropbox* dodijelio. Taj *token* se koristi prilikom definiranja klijenta kojim se upravlja s većinom funkcija koje pruža API. Taj klijent mora biti po zadanome DbxClientV2. To je objektna klasa koja se dobije prilikom inicijalizirana klijenta za uporabu. To je glavni objekt koji je potreban za upravljanje datotekama pomoću aplikacije.



Slika . Provjera tokena i spajanje u oblak

Jedna od funkcionalnosti koja se može napraviti je ispis svih datoteka koje se nalaze u *root* direktoriju. Koristio se klijent koji se prije inicijalizirao i njegova metoda files. Ta funkcija je najvažnija funkcija koja se koristi. Potrebna je za bilo kakve operacije nad datotekama koje se nalaze u oblaku. Kada su se pretraživale datoteke u oblaku odabrala se lokacija iz koje su dohvaćene željene datoteke. Ti podaci su se spremili u poseban objekt ListFolderResult. To je lista podatka koja se sastoji od objekta zvan Metadata. Metadata sadrži sve bitne informacije o nekoj datoteci, od imena ili putanje, pa sve do kada je zadnji put uređena ili napravljena datoteka.



Slika . Ispis datoteka u oblaku

Jedna od važnih funkcija koja postoji je i ta koja služi za učitavanje datoteka u oblak. Kod toga postoje i neki limiti koje ta funkcija ima. Funkcija ima mogućnost učitavanja samo jedne datoteke u oblak. Ta datoteka ne može biti nikakav direktorij nego mora biti obična datoteka, kao na primjer tekstualna datoteka ili slika. Za prijenos je korišten uploadBuilder, pomoću njega se odabralo mjesto spremanja datoteke u oblaku i koja datoteka se prenijela. Datoteka se odabrala pomoću InputStream-a. UploadBuilder kreira FileMetaData objekt koji je proširio MetaData objekt. To je poseban objekt koji sadrži samo podatke vezane za datoteke. 

Slika . Učitavanje datoteke

Zadnja važnija funkcija je ona koja je korištena za preuzimanje datoteka s oblaka. Način na koji je preuzeta datoteka sličan je učitavanju datoteka. Ovoga puta je bilo potrebno inicirati DbxDownloader objekt koji je primio putanju od datoteke u oblaku. Nakon toga se odredio objekt FileOutputStream u kojem se odabralo mjesto na koje se sprema određena datoteka, a zatim se uzeo downloader objekt i pomoću njegove funkcije se preuzela datoteka na lokaciju određenu pomoću FileOutputStream-a. Preuzimanje također ima ograničenja nad funkcijama kao i učitavanje. Nije moguće preuzeti direktorij nego samo datoteke.



Slika . Preuzimanje datoteke

## **TreeView**

TreeView služi za prikaz hijerarhije nekog objekta. Glavni element ili korijen je obično onaj u kojem se nalaze ostale grane ili listovi, po tome je dobilo naziv stablo. Svaki element u TreeView-u se zove TreeItem. Može se postaviti da taj TreeItem bude list koji je sam za sebe ili da je grana koja se onda proširuje i popunjava s listovima. Uobičajeni izgled TreeItem-a se sastoji od nekog teksta i neke takozvane grafike. To znači da se može svaki TreeItem popuniti s tekstom i staviti slika. Postoji i opcija da se nadjača ta metoda i napravi vlastiti izgled za takozvani TreeCell. Postoji par vrsta TreeCell-ova koje je moguće nadjačati koji postoje u Javi. TextFieldTreeCell na primjer sadrži tekstualni element u sebi ili CheckBoxTreeCell koji sadrži potvrdni okvir. Može se i napraviti vlastiti TreeCell koji prima neke druge objekte koji se ne nalaze u klasičnoj Javi. Taj TreeCell će primati neke druge objekte umjesto na primjer String objekta. To je vrlo korisno ako se radi s prilagođenim objektima.

# **Programski kod**

Prilikom kreiranja aplikacije bilo je potrebno napraviti JavaFX Maven aplikaciju. Novo generirana aplikacija ima dvije klase. Controller koji upravlja s FXML datotekom i main klasu koja služi za pokretanje cijele aplikacije.

## **MainApplication**

Glavna main klasa nije previše komplicirana. Sadrži mainStage koji je glavni prozor u aplikaciji i metode za mijenjanje scene u tome Stage-u. MainStage je statični Stage objekt kojem se je moglo iz vanjskih klasa pristupiti i promijeniti scenu koju on sadrži. To je koristilo kako se ne bi uvijek moralo prilikom promjene prozora zatvarati stari i otvarati novi prozor.



Slika . Prikaz MainApplication koda

## **Main Controller i Token Window**

Prilikom korištenja aplikacije je potreban klijenta s nekim accessTokenom, pa je mainController klasa sadržavala jednostavnu funkciju koja je provjeravala ako je *token* koji je upisan u properties datoteci ispravan. U slučaj da nije, uhvatila se iznimka za krivi *token* i otvorio novi prozor u kojem je bilo potrebno generirati novi accessToken za tu aplikaciju. Nakon toga je bilo potrebno opet provjeriti ispravnost *tokena*. Nakon uspješne provjere tek tada se je dobio pristup ostatku aplikacije.



Slika . MainController klasa

Generiranje accessToken-a vršilo se u novom prozoru aplikacije. Najprije je korisnik morao provjeriti ispravne podatke vezane za aplikaciju, te nakon toga generirati link za dohvat token-a. Taj token nije bio isti kao i accessToken, pomoću njega, appKey-a i appSecret-a koje je korisnik provjerio na početku se generira pravi accessToken koji se spremio u properties datoteku. Taj accessToken je jedinstveni za svakog korisnika koji koristi aplikaciju, tako dva korisnika ne mogu vidjeti tuđe datoteke.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika . Prozor za generiranje tokena

## **Files Window Controller**

FilesWindowController je glavna klasa aplikacije, u ovoj klasi se nalazi većinski dio programskoga koda. Ova klasa s njezinom FXML datotekom sadrži glavni prikaz aplikacije i sve glavne funkcije koje se dešavaju u njoj.

Običan prozor se sastoji od nekog zaglavlja s gumbovima koji sadrže neke funkcije i većinski dio je TreeView element. Svaki taj gumb je imao neku posebnu funkciju koja je određena pomoću @FXML anotacije. Svrha te anotacije je kako bi gumbi imali pristup toj funkciji i pritiskom na gumb bi je pokrenuli.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika . Grafički prikaz prazne aplikacije

### **UploadFile**

Gumb učitaj datoteku služio se za pokretanje uploadFile funkcije. U toj funkciji prvo se pozvala DropBoxService klasa i njezina metoda za odabir datoteke na računalu. Ta metoda vraća apsolutnu String adresu od odabrane datoteke. Taj String se predao opet DropBoxService klasi i njezinoj metodi uploadFile koja je tu datoteku učitala u oblak. Na kraju se pozvala funkcija refreshPage koja je ponovo izgradila stablo temeljeno na novim promjenama u oblaku.



Slika . UploadFile funkcija

### **UploadFolder**

Učitaj direktorij gumb služi za pokretanje uploadFolder funkcije. U toj funkciji se prvo pozvala DropBoxService klasa i njezina metoda za odabir direktorija na računalu. Ta metoda vratila je apsolutnu String adresu od odabranog direktorija. String se predao progressBarController klasi i njezinoj metodi uploadFolder koja je taj odabrani direktorij učitala u oblak i usput pratila napredak učitavanja. Na kraju se pozvala funkcija displayTreeView koja je ponovno izgradila stablo temeljeno na novim promjenama u oblaku. 

Slika . UploadFolder funkcija

### **CreateFolder**

Napravi direktorij gumb služi za stvaranje direktorija u trenutnom direktoriju u kojem se korisnik nalazio.

### **RefreshPage**

Osvježi gumb ponovno je izgradio cijelo stablo ovisno dali se radilo o pretraživanju ili običnom prikazu i prikazao ga na ekranu.



Slika . CreateFolder i Refresh funkcije

### **AutoFolder**

Dodaj sinkronizaciju gumb poziva funkciju u kojoj se najprije inicijalizirao novi AutoFolder objekt s putanjom na neki direktorij koji je korisnik odabrao. Nakon toga se je inicijalizirao i novi Timer objekt koji se koristio kako bi se zakazali AutoFolder objekt koji se je pokrenu nakon jedne sekunde. Na taj način se osigura da ta funkcija bez problema radi.



Slika . AutoFolder funkcija

### **Izrada stabla**

Na početku je TreeView okvir potpuno prazan, ali ga se naknadno popunilo kada su se dobili svi dokumenti s oblaka. Za to su se koristile dvije funkcije.

Prva funkcija koja se koristila je createTree. Ona služi za izradu stabla koje se prikaže u glavnom dijelu aplikacije. Funkcija se koristila metodom listFolder za dohvat svih datoteka na određenoj putanji.



Slika . CreateTree funkcija

Druga funkcija je najveća i najbitnija funkcija koja se nalazi u ovoj klasi. Ona se sastoji od određivanja šablone izgleda jednoga treeItem-a, ali i cijelog treeView-a. Pomoću nje se prikazalo stablo koje se kreiralo iz datoteka u oblaku. U funkciji je postojao poziv treeView-a na metodu setCellFactory. U toj metodi se inicirao novi izgled ćelija u tome treeView-u. Nakon toga su se napravile liste sa svim datotekama i direktorijama na željenoj lokaciji i pomoću nje se je pozvala funkcija createTree u kojoj se izradilo to stablo. Ovisno o tome je li korisnik htio sortirati prikaz ili ne, prikaže se stablo.



Slika . DisplayTreeView funkcija

### **CellFactory**

U klasi FilesWindowController u funkciji displayTreeView se nalazi setCellFactory metoda. Cell Factory je metoda koja služi kako bi se u TreeView-u odredila neka druga vrsta ćelija. Potrebno je bilo napraviti vlastitu ćeliju koja prima MetaData objekt. Umjesto da primam neki objekt kao String primam MetaData jer sadrži puno više informacija koje se kasnije koriste za prikaz određene ćelije. Unutar setCellFactory metode se pozvao CallBack objekt u kojem se nadjačala funkcija call. Unutar nje se izradio novi treeCell koji je bio šablona za sve ostale ćelije i na kraju se taj treeCell objekt vratio i svaka ćelija je izgledala kao ta šablona. Najprije se je nadjačala updateItem metoda koja je zaslužna za izgled. Unutar prvog if uvjeta je pozvana isEmpty metoda kojoj ovisno o tome ako je ćelija prazna ne popunjava ništa. Boolean varijabla isFolder služila je za identifikaciju ako je item direktorij, ovisno o tome je drugačije generiran izgled ćelije. Izgledi ćelije za datoteke i direktorije se razlikuju po izgledu, ali i funkcijama koje se mogu vršiti nad njima. Sljedeće je varijabla path koja pokazuje putanju za datoteku u oblaku i varijabla name koja je naziv te datoteke u oblaku. Zbog tih varijabli je bilo najvažnije napraviti novu vrstu TreeCell-a jer nije bilo moguće te sve informacije drugačije predati za izradu ćelije. ContexMenu objekt služio je kako bi se ovisno dali se radilo o datoteci ili direktoriju dodale neke od osnovnih funkcija za upravljanje nad njima u oblaku.

Prvi if uvjet provjeravao je radi li se o direktoriju ili ne. Ovaj dio nije isti datotekama i direktorijima jer se razlikuju po funkcijama. Inicijalizirao se MenuItem za brisanje direktorija i dodao u ContextMenu koji se inicijalizirao na početku. Nakon toga je dodan novi EventHandler koji se aktivirao kad se pritisnuo taj MenuItem. U EventHandler-u handle metodu se pokrenuo proces za brisanje direktorija. Pozvana je klasa DropBoxService i njezina funkcija DeleteFolder kojoj je predana putanja od odabranog direktorija. Nakon toga se uzeo odabrani TreeItem i obrisalo ga se iz TreeView-a. Sljedeći MenuItem je inicijaliziran kako bi se koristio za preuzimanje direktorija. Nakon inicijalizacije dodan je u već postojeći ContextMenu. Tome MenuItem-u je dodan vlastiti EventHandler koji se aktivirao pritiskom na njega. Handle metoda je prvo pozvala klasu DropBoxService i njezinu metodu getFolder u kojoj se dohvatio neki direktorij i njezina putanja u koju je korisnik spremio taj direktorij. Nakon toga se pozvala metoda downloadFolder kojoj se predala putanju od direktorija u oblaku i putanju kamo će se spremiti taj direktorij na računalu.



Slika . Brisanje i preuzimanje direktorija kod

Sljedeća if naredba provjeravala je ako item nije direktorij. Ako se radilo o bilo kakvoj vrsti datoteke koja nije direktorij ušlo se unutar if naredbe. Prvo se inicijalizira MenuItem za brisanje datoteke i dodao se u contexMenu. Inicijalizirao se novi EventHandler koji se aktivirao klikom na taj MenuItem. Unutar handle metode pozvala se klasa DropBoxService i metoda deleteFile kojoj se predala putanja od odabranog item u oblaku. Nakon toga se uzeo taj objekt i obrisalo ga se iz treeView-a. Drugi MenuItem je bio zaslužan za preuzimanje datoteke na računalo. Inicijalizira se novi MenuItem i dodalo ga se u contexMenu. Inicijalizirao se novi EventHandler koji se aktivirao klikom na njega. Pozvala se klasa DropBoxService i metoda getFolder koja je vratila putanju direktorija u koji se spremila datoteka. Zatim se pozvala metoda downloadFile koja je primila tu putanju i putanju datoteke koja se preuzima.



Slika . Brisanje i preuzimanje datoteke kod

Nakon if naredbi slijedio je dio koji je zajednički datotekama i direktorijima. Najprije je naredba za premještaj datoteke. Move naredba nema nikakve razlike za datoteke i direktorije pa po tome ima i zajedničku funkciju. Premještaj datoteke je napravljen isto kao i brisanje i preuzimanje pa je također napravljen novi MenuItem. Taj MenuItem se dodao u contexMenu i pritiskom na njega se aktivirao novi EventHandler. Unutar njegove handle metode prvo se inicirao FXMLLoader koji je služio za učitavanje scena u JavaFx-u. FXMLLoader-u se zatim dodala lokacija na MoveWindow.fxml datoteku koja služi kao pomoćni okvir za premještaj datoteke na novu lokaciju. Zatim se inicirao objekt Group koji je služio za grupiranje svih elemenata u MoveWindow okviru. Kreirana su dva Label-a i jednom je postavljen neki tekst, a drugi je ostao prazan. Zatim se drugome postavio tekst na putanju od datoteke koja se planirala premještati. To je razdvojeno u dva Label-a, kako bi se lakše pristupilo putanji datoteke u drugom prozoru. Zatim se napravio HBox u koji su dodana ta dva Label-a. Taj HBox je dodan u grupu inicijaliziranu prije. Zatim se toj grupi također dodao FXMLLoader od prije. Grupa funkcionira tako da se na vrhu prvo pokaže prvo dijete koje je dodano unutra, a zatim drugo. Ta grupa je zatim dodana u novo inicijaliziranu scenu pomoću koje se onda pokreće novi Stage koji će biti zaslužan za premještaj te datoteke na novu lokaciju.



Slika . Premještaj datoteke kod

Sljedeća je metoda za kopiranje datoteke na neku lokaciju u oblaku. Ona je zajednička i datotekama i direktorijima. Prvo se inicijalizirao MenuItem za kopiranje datoteke i dodalo ga se u contexMenu. Tome MenuItem-u je dodan EventHandler koji se aktivirao klikom na njega. Unutar njegove handle metode se kreirao FXMLLoader objekt i stavljena mu je lokacija na CopyWindow.fxml datoteku. Kreirala se nova grupa i dva Label-a, prvome je stavljen nekakav tekst, a drugoga se popunilo s putanjom datoteke koja se planira kopirati. Oba Label-a su stavljena u HBox i taj HBox je dodan u grupu od prije. Toj istoj grupu se dodao FXMLLoader sa FXML datotekom. Tu grupu se predalo u novo inicijaliziranoj sceni. Ta scena je predana Stage-u i pokreće se novi okvir u kojem se kopira datoteka na novu lokaciju.



Slika . Kopiranje datoteke kod

Time su pokrivene sve osnovne funkcije nad datotekama. Također se može dodati funkcija za preimenovanje datoteke. Međutim preimenovanje je lakše napraviti pomoću TextField-a nego da se koristio ContextMenu. TextField je element u JavaFX-u u koji korisnik može čitati i pisati neki tekst. Prvo se inicijalizirao HBox i jedan TextField. Prilikom inicijaliziranja TextField-a odmah se postavio tekst na naziv datoteke. Tome TextField-u se dodao Listener koji se pokrenuo kad se fokusiralo na taj TextField, to znači da kad se pritisnuo taj TextField se aktivirao Listener. Nakon toga umjesto da se čekalo da se izgubi fokus nad objektom, čekalo se kad se pritisne i otpusti tipka Enter na tipkovnici. Kad se otpustila tipka Enter pozvala se DropBoxService klasa i njezina metoda renameFile kojoj se predala putanja od datoteke koja će se preimenovati i novi naziv koji je zapisano u TextField-u. Nakon toga se pozvala refreshPage za ponovno učitavanje datoteka s novim promjenama. Taj TextField se dodao u HBox koji je bio inicijaliziran na početku. Prije TextField-a se dodao ImageView element koji ovisno o datoteci ili direktoriju stavi sliku o čemu se radi. Također je poželjno bilo dodati datum kada je nešto mijenjano na datoteci. To se postiglo tako da se pozvala if funkcija i provjerila je li datoteka ili je direktorij. Ako se radilo o datoteci uzeo se iz FileMetaData objekta Date objekt sa zadnjom promjenom i pretvori se u Label koji je dodan u HBox.

Na kraju updateItem funkcije se taj HBox u kojem se nalazilo ime datoteke i datum dodao pomoću setGraphic metode. Također se i dodao contextMenu pomoću metode setContextMenu.



Slika . Promjena naziva i datum kod

Cilj je da bi aplikacija bila što sličniji normalnom pregledavanju datoteka kao na računalu. Stoga se sa lijeve strane nalazio TreeView koji prikazuje sve direktorije i omogućava ulazak u njih. Na te TreeCell-ove se dodalo da se duplim klikom otvori ta mapa u aplikaciji. Najprije se nad TreeCell-om dodao događaj koji se aktivirao klikom miša na njega. Unutar toga EventHandler-a se nadjačala metoda handle. Provjerilo se ako TreeCell koji je kliknut nije prazan i inicijalizirao Metadata objekt koji se dobije od TreeCell-ovoga TreeItem-a. Bolje je da se aktivira duplim klikom miša, a ne samo jednim. Ima if naredba u kojoj se provjeravalo ako je ClickCount nad događajem koji se pokrenuo klikom miša jednak 2 i ako je taj item koji se kliknuo direktorij. Unutar te if naredbe prvo se inicijalizirao HyperLink objekt koji je vraćao korisnika na neki prijašnji direktorij. Napravio se HyperLink za svaki direktorij koji se nalazio između početnoga i trenutnoga direktorija. Zatim se inicijalizirao novi EventHandler koji je služio za dodavanje događaja koji vraća korisnika u prijašnji direktorij. Unutar njega se prvo pozvala funkcija displayTreeView s parametrom koji je pokazivao na prošli direktorij. Nakon toga se promijenio String koji je pokazivao na trenutni direktorij na direktorij u kojem je korisnik trenutno bio. Zatim slijedi for petlja koja je uspoređivala svu djecu u VBox u kojem su se nalazili HyperLink-ovi za natrag i ovisno o tome koliko se željelo ići natrag toliko ih se izbrisalo. Tako se postiglo brisanje HyperLink-ova koji pokazuju u neki direktorij koji se nalazio unutar trenutnoga, ali je ostavljao direktorije za natrag ako se nije otišlo natrag do početnog direktorija. Taj EventHandler se dodao na svaki HyperLink za nazad.

Time je postignuto sve bitno za ćeliju koja se generira. Uredilo se kako će izgledati nove ćelije i ta šablona ćelije se predala natrag. Na temelju toga se popunio TreeView.



Slika . Promjena direktorija kod

Iznad TreeView-a za prikaz direktorija se nalazilo polje za pretraživanje datoteka u aplikaciji. Upisao se traženi pojam i aplikacija je ispisala sve datoteke i direktorije koji imaju naziv sličan upisanome tekstu. Pretraživanje radi na isti način kao i obično pregledavanje rezultata, samo umjesto putanje traži String naziva koji se želi pronaći. Ispod gumba aplikacije se nalazi ime direktorija u kojem se korisnik trenutno nalazi.

## **AutoFolder**

Budući da je naporno voditi brigu oko toga koje datoteke su stavljene u oblak, a koje su još na računalu i nisu spremljene, dobro je koristiti automatsko učitavanje u oblak. Imati određenu mapu koja je klonirana u oblaku i sve promjene koje se dogode u toj mapi se naprave i u oblaku. To je jedna od težih funkcija koje se mogu izvoditi. Prvo je potrebno pratiti sve promjene koje su se dogodile u nekoj mapi, a zatim je potrebno to isto reproducirati u korisnikovom direktoriju u oblaku.

AutoFolder objekt ima tri elementa u sebi, ima nekakav WatchService, to je zaslužno za praćenje promjena u nekoj mapi. Ima nekakav path objekt s kojim je određena putanja foldera kojega se prati i ima WatchKey objekt koji ima određene vrste promjena koje će se pratiti nad direktorijem. Konstruktor za AutoFolder prima samo jedan parametar i to je za putanju do direktorija. U konstruktoru se dalje inicijalizirao određeni WatchService nad tom putanjom i odredio mu se koje key promjene mora pratiti. Na kraju toga se učitao taj cijeli direktorij u oblak kako bi sve datoteke bile odmah na njemu.

WatchService radi na principu da mora biti u nekoj beskonačnoj petlji i vrtjeti se konstantno čekajući da se neka promjena dogodi. Nakon što se uhvatila promjena on nju obradi ovisno o vrsti promjene i resetira WatchKey na početne postavke, očekujući sljedeću promjenu. Za korisnikov direktorij su se pratile tri vrste promjena, ako je nastalo nešto novo u direktoriju, ako je neka već postojeća datoteka promijenjena i ako je nešto izbrisano iz toga direktorija.



Slika . AutoFolder konstruktor kod

JavaFX je program koji se vrti na samo jednoj dretvi, ali ako se pokrene AutoFolder onda će ta dretva cijelo vrijeme tražiti promjene i neće se moći ništa drugo raditi u aplikaciji. Stoga je bilo potrebno pokretnu paralelno dvije dretve, jednu koja vrti AutoFolder, a drugu koja vrti ostatak aplikacije. Međutim, s JavaFX-om to nije tako jednostavno, budući da ono dopušta sam rad s jednom dretvom i nije moguće dodati još jednu dretvu. Način na koji se je to zaobišlo je bio da se napravio takozvani TimerTask koji se nakon što se odredio direktorij nad kojim će se pratiti promjene. Pokrenuo se TimerTask nakon jedne sekunde i on se konstantno vrtio na posebnoj dretvi, a omogućavao je normalno raditi u aplikaciji. Stoga je AutoFolder stavljen da proširuje taj TimerTask objekt i dodana mu je metoda run koja se pokrene nakon te jedne sekunde. Tako se taj program konstanto vrtio na toj drugoj dretvi, a na prvoj normalno radi aplikacija.



Slika . Run metoda AutoFolder

## **DropBoxService**

DropBoxService je klasa u kojoj se odvija većina operacija između aplikacije i oblaka. U njoj su opisane funkcije koje vrše operacije nad datotekama. Najvažniji objekt u klasi je klijent koji je potreban za sve funkcije u klasi. Kao i getter i setter metode za klijenta. Getter prvo provjerava ako klijent nije null i ako je dobro postavljen accessToken nad njim. Ako klijent ne postoji onda se poziva metoda setter-a. Setter radi klijent pomoću accesToken kojeg dobije iz getToken metode. Metoda getToken iz properties datoteke uzima accessToken i vraća ga natrag.



Slika . Getter i setter klijenta kod

### **DeleteFile**

Funkcija deleteFile prima putanju od datoteke u oblaku i poziva nad klijentom files metodu pomoću koje je moguće upravljanje nad datotekama. Pomoću te metode se zove deleteV2 metoda koja prima putanju i tako briše datoteka s oblaka.



Slika . Brisanje datoteke kod

### **GetFolder**

GetFolder funkcija koristi se kako bi se odabrao neki direktorij na računalu i uzela njegova putanja. Najprije se inicijalizira Stage i DirectoryChooser. DirectoryChooser se koristio kako bi korisnik odabrao neki direktorij na računalu. Zatim se inicijalizirao File koji se dobio tako da se pozvao DirectoryChooser u novom Stage-u. Nakon odabira direktorija provjerio se je li odabran direktorij, te se vratila String vrijednost putanje odabranog direktorija.



Slika . Odabir datoteke kod

### **GetFile**

Funkcija GetFile je zaslužna za dohvat neke datoteka s računala. Nakon inicijalizacije Stage-a i FileChooser-a. FileChooser je zaslužan samo za odabir datoteka na računalu, ne daje mogućnost odabira direktorija. Inicijalizira se File koji je napravljen pomoću FileChoosera koji je pokrenut u novom Stage-u. Provjerilo se ako nije ništa odabrano, ako je sve u redu vraća se String putanje.



Slika . Dohvat datoteke kod

### **RenameFile**

RenamFile prima dva parametra, jedan naziv datoteke se preimenuje, a drugi je novi naziv za datoteku. Prije početka preimenovanja potrebno je urediti malo putanju od datoteke. Prvo se inicijalizira String objekt kojem se uzeo samo zadnji dio putanje uključujući i kosu crtu, a zatim se pomoću toga izrezala apsolutna putanja datoteke da nema taj zadnji dio u putanji. Budući da ne postoji nikakva funkcija za preimenovanje datoteka u *Dropbox*-ovom SDK-u, trebalo se nekako drugačije snaći. Najbolje rješenje je to da se koristila moveV2 metoda u kojoj se jednostavno datoteka premjesti na isto mjesto, ali s drugim nazivom. Pozivao se klijent s files metodom i pomoću nje moveV2. Toj metodi se predala stara putanja kao prvi parametar, a drugi parametar je apsolutna putanja bez starog naziva datoteke i spojilo ga se novim nazivom.



Slika . Preimenovanje datoteke kod

### **MoveFile**

Pomicanje datoteke se vrši jednostavnije nego preimenovanje jer nije potrebno za tolikom manipulacijom nad String-ovima. Potrebno je uzeti iz putanja datoteke koja se je premješta naziv datoteke. Zatim se prilikom poziva moveV2 metode od klijenta predaje početna adresa koja se premješta i spojenu adresu na koju se premješta s nazivom datoteke.



Slika . Premještaj datoteke kod

### **CopyFile**

Kopiranje datoteke je potpuno identično premještanju datoteke. Najprije se dobije naziv datoteke koju korisnik kopira. Zatim klijent poziva metodu copyV2 kojoj se predaje početna putanja datoteke i spojena putanja od lokacije gdje će se kopirati datoteka i naziv datoteke.



Slika . Kopiranje datoteke kod

### **CreateFolder**

Izrada mape je jednostavna funkcija koja primi dva parametra, prvi je putanja od direktorija gdje se radi novi direktorij, a drugi parametar je naziv za direktorij. Klijent je pozvao metodu createFolderV2 kojoj se predao trenutni direktorij spojen s nazivom direktorija i kosom crtom.



Slika . Izrada direkotrija

### **DownloadFolder**

Preuzimanje direktorija je kompliciranije za izvesti jer ne postoji nikakva metoda pomoću koje je moguće preuzeti cijeli direktorij. Potrebno je postepeno prolaziti kroz cijeli direktorij i preuzimati datoteku po datoteku. DownloadFolder funkcija prvo prima parametar koji pokazuje putanju od direktorija koji se preuzima, a drugi parametar je putanja gdje se preuzima taj direktorij. Najprije se na lokaciju gdje se preuzima direktorij radi novi direktorij koji se popunjava. Zatim se napravi lista sa svim datotekama i direktorijima koji se nalaze u tome direktoriju. Ulazi se u beskonačnu petlju koja se vrti tako dugo dok nema više datoteka za preuzeti. Za svaki element u toj listi provjerava se je li direktorij ili nije. Ako je taj element direktorij rekurzivno se poziva funkcija DownloadFolder s putanjom toga direktorija. Ako se ipak radi o datoteci onda klijent objekt poziva metodu download kojoj predaje putanju od trenutnog elementa. Pomoću FileOutputStream-a se preuzima datoteku na računala. Nakon što se preuzeo svaki element u direktoriju izlazi se iz beskonačne petlje.



Slika . Preuzimanje direktorija

### **UploadFile**

Učitavanje datoteke je jednostavna funkcija za učitavanje datoteke u oblak. Predaje joj se putanju od datoteke koja se učitava. FileChooser prilikom odabira datoteke vraća apsolutnu putanju gdje je kosa crta okrenuta prema lijevo, a *Dropbox* prihvaća samo desno okrenuto okomitu crtu. Zato je potrebno prvo obraditi putanju tako da se makne sav apsolutni dio iz putanje i ostane samo naziv datoteke, te da se okrenu sve okomite crte na drugu stranu. Nakon toga pomoću InputStream-a se učitava datoteka i koristeći se metodom uploadBuilder kojoj se predaje nova putanja od datoteke, učitava se u oblak.



Slika . Učitavanje datoteke kod

### **UploadFolder**

Učitavanje direktorija radi na drugačiji princip od svih ostalih funkcija. Umjesto da se pokrene normalno funkcija napravi se prvo Task objekt. Unutar njega se nalazi cijela funkcija za učitavanje direktorija. Na kraju se samo taj Task pokrene u drugoj dretvi. To se radi kako bi se omogućilo praćenje napretka učitavanja direktorija u oblak. Svaki put kada se neka datoteka učita promijeni se vrijednost u ProgressBar-u. Učitavanje direktorija radi na principu preuzimanja direktorija. Ne postoji metoda koja omogućava direktno učitavanje cijelog direktorija u oblak. Funkciji se zato preda putanja od direktorija koji se učitava u oblak. Napravi se lista s putanjama koju se popune pomoću walk metode. Ona prolazi kroz cijeli direktorij i vrati apsolutne putanje svih datoteka i direktorija koje se nalaze unutra. Zatim se ta lista pretvara u listu String-ova. Inicijalizira se String koji govori kako glasi apsolutna putanja za taj direktorij. Budući da se ne može koristiti taj dio u oblaku, koristi se kako bi se izrezao taj dio iz svih putanja koje se nalaze listi. Koristeći petlju prolazi se kroz svaki element u listi i provjera se je li trenutni element direktorij. Ako je direktorij izreže se apsolutni dio iz njega i zamjene se lijevo okomite crte s desno okrenutima koje koristi aplikacija. Budući da se radi o direktoriju kreira se novi direktorij s nazivom elementa. Ako element nije direktorij već se radi o datoteci tada se također makne apsolutni dio iz putanje i zamjenom okomite crte na one koje se koriste u aplikaciji. Koristeći InputStream-a ta datoteka se učita na njezino mjesto gdje je i bila na računalu. Nakon što se prođe sve elemente uspješno je učitan cijeli direktorij u oblak.



Slika . Učitavanje direktorija kod

### **UploadFolderAuto**

UploadFoldeAuto je funkcija koja radi na isti princip kao i njezina osnovna funkcija, ali se razlikuje po tome što prima dodatni parametar kojim je moguće odrediti lokaciju na kojoj se učita direktorij. To uglavnom ima svrhu jer prilikom automatskog praćenja promjena na odabranom direktoriju ne želi se učitati novi direktorij na početnu stranicu nego baš na određeno mjesto i u određeni direktorij. Tada se prilikom izrade direktorija ili učitavanja datoteke preda metodama i početna putanja koja se spaja s putanjom datoteke ili direktorija.

### **UploadFileAuto**

UploadFileAuto se također koristi samo prilikom automatskog praćenja direktorija. Isto prima dodatni parametar koji određuje lokaciju na koju će se učitati neka datoteka. Taj parametar se zatim spaja s putanjom datoteke i učitava u oblak kao i normalna UploadFile funkcija.

### **DeleteFileAuto**

Brisanje datoteke se radi na principu kao i normalna funkcija, ali ovoga puta se koristi dodatni parametar kojim se određuje početni dio putanje koji se briše s oblaka.

## **MoveWindowController**

MoveWindow služi kao dodatni prozor koji se otvori kad korisnik želi premještati neku datoteku ili direktorij na drugo mjesto. Scena se sastoji od dva gumba koji potvrđuju hoće li se napraviti premještaj ili ne. TreeView u kojem su dodani svi direktoriji i klikom na neki TreeItem se bira u koji se premješta datoteka. Na vrhu se nalaziti informativni tekst koji govori koja datoteka se premješta na novu lokaciju. Tu je također zapisana i putanja datoteke koja se premješta.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika . MovieWindow prozor

Gumb Potvrdi prvo uzme iz Label-a putanju od datoteke koja se premješta i zajedno s putanjom od odabranog direktorija poziva DropBoxService i njezinu metodu moveFile i premješta datoteku na novu lokaciju. Nakon toga zatvara taj prozor. Drugi gumb služi za prekid premještaja datoteke i samo zatvara taj prozor bez ikakve promjene.

Također postoje i funkcije koje popunjavaju TreeView na isti način kao što je to i napravljeno u FilesWindowController-u. Jedina promjena jest ta da ispisuje samo direktorije bez ikakvih datoteka i nemam nikakve funkcije za rad s njima. Ispisuje samo naziv direktorija i stavljen im je jedan novi EventHandler. Taj EventHandler služi kako bi se prilikom pritiska miša na neki direktorij odabrao on za premještaj datoteke. To radi tako da se mijenja naziv String objekt koji je dostupan cijeloj klasi.

## **CopyWindownController**

CopyWindow je izgledom identičan MoveWindow-u. Razlikuje se samo po funkciju za koju služi. Sastoji se od dva gumba, TreeView-a i Label-a na vrhu. Gumbi služe za prekid kopiranja ili za potvrdu da će se ta datoteka kopirati na određeno mjesto. Gumb za potvrdu opet uzima putanju datoteke koja se kopira i koristeći se odabranom putanjom direktorija u koji se kopira se poziva DropBoxService i njezina metodu za kopiranje datoteke.

Koristeći se sličnim funkcijama kao u FilesWindowController-u popuni se TreeView s direktorijima u koje je moguće kopirati datoteku. Identično kao i kod prozora za premještaj datoteke napravi se samo ispis naziva direktorija bez ikakvih funkcija. Dodavanjem EventHandler-a koji klikom na direktorij odabire u koji se kopira datoteka, mijenjajući u njemu String objekt koji prikazuje na direktorij u koji se kopira datoteka.

# **Zaključak**

U izradi ove aplikacije koristio sam se programom koji je meni bio vrlo dobro poznat. Zbog toga nisam imao nikakvih velikih poteškoća u izradi grafičkog sučelja. Ipak, bilo je nekih poteškoća oko odabira načina na koji će se prikazivati sve pohranjene datoteke jer nisam očekivao da će biti toliko problema prilikom dodavanja tih datoteka u neki prikaz.

Programski dio aplikacije je teži za napraviti. Najprije je bilo potrebno proučiti svu dokumentaciju potrebnu za spajanje s našim podacima u oblaku. To je sve bila hrpa novih informacija od kojih je većina bila već zastarjela, ali i dalje ipak postoji potpora za njih, pa sve do nekih funkcija i metoda koje nisam koristio. Prenijeti to svo znanje u programski kod koji će se služiti tim funkcijama bila je poteškoća sama po sebi. Bilo je problema prilikom implementacije toga u aplikaciji, želeći to postići na najefikasniji mogući način. Neke funkcije su bile puno lakše za dodati i napraviti od ostalih, dok je za nekoliko njih trebalo uložiti više truda i bolje ih napisati i napraviti. Na kraju je još bilo za dodati neke estetske elemente koji služe za lakše snalaženje u grafičkom sučelju.

Sve u svemu jako sam zadovoljan kako je ispala aplikacija i koje funkcije pruža. Za prvu verziju aplikacije je ispala vrlo dobro. Ipak, ima još stvari koje bi se mogle dodati za bolje upravljanje datotekama. Naučio sam neke nove funkcionalnosti koje pruža *Java* i mogućnosti koje se mogu napraviti pomoću *Dropbox*-ovoga API-a.

# **Literatura**

1. Eisenmann, Thomas R., Michael Pao, Lauren Barley. Dropbox: „It Just Works.“. Harvard Business School 2011;811-065.
2. Get more space and powerful features, s Interneta, <https://www.dropbox.com/space-upgrade?oqa=rnro&reason=upgrade>, [Pristupano 16.8.2022]
3. DBX Platform developer guides, s Interneta, <https://www.dropbox.com/developers/reference>, [Pristupano 18.8.2022]
4. Official Dropbox Java SDK unspecified API, s Interneta, <https://dropbox.github.io/dropbox-sdk-java/api-docs/v5.3.0/index.html>, [Pristupano 31.5.2022]
5. History of Java Programming Language, s Interneta, <https://www.freejavaguide.com/history.html>, [Pristupano, 18.5.2022]
6. Using JavaFX UI Controls, s Interneta, <https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm>, [Pristupano 2.6.2022]
7. SceneBuilder, s Interneta, <https://github.com/gluonhq/scenebuilder>, [Pristupano 28.5.2022]