

Szegedi Tudományegyetem

Informatikai Intézet

**Videólejátszó alkalmazás idegen nyelvet
tanulóknak**

Szakdolgozat

Készítette:

Krucnai Attila

programtervező informatikus
szakos hallgató

Témavezető:

Tóth Zoltán Gábor

egyetemi tanársegéd

Szeged

2019

Tartalomjegyzék

Feladatkiírás	3
Tartalmi összefoglaló	4
Bevezetés	5
1. Háttér	7
Háttér	7
1.1. Architektúrális áttekintés	10
2. Az alkalmazás funkcionálisai	11
2.1. Videólejátszó alapfunkciók	11
2.2. Teljes képernyős mód	13
2.3. Média- és feliratfájlok választása	15
2.4. Feliratok kezelése	17
2.4.1. Feliratfájlok beolvasása	17
2.4.2. Feliratok megjelenítése	19
2.5. Feliratok fordítása	21
2.5.1. Szerverrel történő kommunikáció	22
2.5.2. Fordítások megjelenítése	23
2.6. Nyelvek kiválasztása	25
2.7. Online feliratok letöltése	26
2.7.1. Feliratok keresése	27
2.7.2. Feliratok letöltése	29
2.8. Adatbázis műveletek	29
2.9. Szótár generálása ismeretlen szavakból	31
2.10. Tudásellenőrző kvíz generálása ismeretlen szavakból	33
2.11. Beállítások menü	35
2.12. Telepítőfájl létrehozása	37
Irodalomjegyzék	41
 Nyilatkozat	 43
Köszönetnyilvánítás	44

Feladatkiírás

Az idegennyelv tanulás folyamán sokszor nemcsak a tankönyvekből tanulunk. Az internet, könyvek, újságok, filmek mind szerves részét képezhetik egy idegennyelv elsajátításának. A szoftver azoknak az egyéneknek nyújthat segítséget, akik rendszeresen néznek idegennyelvű filmeket, idegennyelvű felirattal. A felhasználó egy ismeretlen szó esetén megállíthatja a lejátszást, és a mondatelemre kattintva megtudhatja az anyanyelvi jelentését. Így a szótározással eltöltött idő lényegesen kevesebb, kevésbé szakad meg a film folyamatossága. Az ismeretlen szavakat az alkalmazás egy adatbázisban tárolja, amelyből a film befejeztével egy szöszedet generálódik a filmből kiragadott példamondatokkal. A létrejött fájlt el lehet menteni, így a felhasználó a későbbiekben is átnézheti a nehezebben megjegyezhető szavakat. Egy másik lehetőség a film véget értevel, hogy az alkalmazás egy feleletválasztós kvízt generál, amely a korábban ismeretlen mondatelemeket tartalmazza. A tanuló így ellenőrizheti a saját tudását, valamint azt, hogy mennyire sikerült elsajátítania az idegen szavak jelentéseit.

Tartalmi összefoglaló

Jelen dolgozat célja egy a nyelvtanulást elősegítő alkalmazás munkamenetének dokumentálása. A feladat egy olyan asztali alkalmazás elkészítése volt, amely képes videófájlok lejátszására, feliratok dinamikus megjelenítésére, valamint a felirat szavaira kattintva azokról különböző fordításokat tud ismertetni a felhasználóval, olyan módon, hogy a lejátszott videó folytonossága nem szakad meg. Ezek után az alkalmazás lehetőséget biztosít egy tudásfelmérő teszt kitöltésére, illetve egy szószeret generálására, amely a felhasználó számára ismeretlen szavakból áll.

Az szoftver koncepciója egy általános tényre épít, miszerint mindenki szeret filmeket nézni. Mivel rengeteg műfaj áll ezekből a felhasználó rendelkezésére, így mindenki megtalálhatja a neki legmegfelelőbb filmkategóriát, amely filmjeit szívesen nézi. Emellett a filmek a nyelvtudás egyik forrása is lehet, mivel ezek rendszerint idegen nyelvűek, amelyeket később szinkronizálnak anyanyelvűre. Ezek alapján a kellemest összeköthetjük a hasznossal oly módon, hogy filmnézés közben sajátíthatjuk el egy-egy idegennyelv szavait, kifejezéseit, nyelvtani összetételeit, mindezt szórakozással egybekötve.

Az elkészült alkalmazás a fent említett feltételeket teljesíti, kényelmes felhasználói felületet biztosít a használója számára. Fő felhasználási területe tehát a nyelvtanulás, mivel idegennyelvű filmek nézése közben a felhasználó képes számára ismeretlen szavak jelentéseinek megismerésére. A film megnézése után pedig különböző eszközök állnak rendelkezésre, amellyel a néző a tudását ellenőrizheti, bővítheti.

Bevezetés

Szakedolgozatom célja, egy a nyelvtanulást elősegítő videolejátszó alkalmazás fejlesztése, melynek segítségével a felhasználók a szórakozást tanulással köthetik össze. Ennek eredményeképp úgy tudnak elsősorban idegennyelvű szavakat elsajátítani, hogy mellőzik a tanulással járó feszültséget, illetve kötöttséget. Tehát a nyelvek szókincsének elsajátítása, elmélyítése, tudatosítása, lehető legbiztonságosabb használata ezáltal könnyedebbé, szórakoztatóbbá és gördülékenyebbé válhat, mely egy hatékonyabb tanulási folyamatot tesz lehetővé. Mivel az általam elkészített alkalmazás elsősorban filmek lejátszására ajánlott, így a felhasználók egy sokkal inkább a hétköznapi kommunikációhoz alkalmazható szókincsre tehetnek szert. Ezen szoftver segítségével az idegennyelv-tanulástól ódzkodó személyek érdeklődése nagymértékben felkelthető, mivel a tanulás nem direkt módon, hanem filmnézés, szórakozás közben történik, a szavak az idegennyelven hallott és látott szövegben, kontextusban rögzülnek.

Napjainkban egyre inkább felmerül a kérdés, hogy a nyelvtanulást milyen módszerekkel lehet hatékonyabbá, illetve játékosabbá tenni. Kétségtelen azonban, hogy egy idegen nyelv magabiztos szintű elsajátításához a megfelelő méretű szókincs birtoklása elengedhetetlen. A szavak memorizálása sokszor monoton folyamat, valamint rengeteg ráfordított időt igényel. Ennek kapcsán saját tapasztalatból is biztonsággal állíthatom, hogy az idegen nyelvű szavak elsajátítása megítélésem szerint a jelenleg használatos oktatási módszerekkel igen monoton, mely a diákok motivátlanságát, elidegenülését eredményezheti. Mivel a motiváció hiánya gyakran tanulási nehézségeket eredményez, a nagy erőfeszítések árán megtanult szavak nem rögzülnek megfelelően, ha igen, akkor sokszor hibásan, így gyorsan el is felejtődnek. A jelenlegi, fejlett informatikai világban a fiatal generáció mindennapi életében természetes és elengedhetetlen az informatikai eszközök használata, ami lehetővé teszi az idegennyelv tanulás elősegítését is. Az informatikai eszközök sok-

oldalúsága, hordozhatósága, a nap minden órájában lehetővé teszi a használatot, amely a filmnézéssel és szórakozással egybekötött nyelvtanulást is elősegíti, így a fiatal generáció életvitelébe teljes mértékben beilleszthető. Egy alternatívaként merülhet fel a tanulásra az a videólejátszó alkalmazás, amelyet szakdolgozatom keretein belül készítettem el, azt a célt kitűzve magam elé, hogy közelebb hozzam egymáshoz a tanulást és a szórakozást.

Egy idegennyelv elsajátítása során sokszor nemcsak tankönyveket használunk. Az internet, könyvek, újságok, filmek mind szerves részét képezhetik a tanulásnak. A szoftver azoknak az egyéneknek nyújthat nagymértékben segítséget, akik rendszeresen néznek idegennyelvű filmeket, idegennyelvű felirattal. A felhasználó egy a számára ismeretlen szó esetén a film megállítása nélkül a mondatelemre kattintva megtudhatja az anyanyelvi jelentését. Így a szótározással eltöltött idő megszűnhet, mely eredményeképp a megtekintett film folyamatossága biztosított. Azon szavakat, melyek anyanyelvi megfelelőjét a felhasználó filmezés közben megtekinti, az alkalmazás egy adatbázisban tárolja, amelyből a film befejeztével egy szószedet generálódik. Az így összeállt lista különlegessége, hogy nem csak az egyes szavak idegen- illetve anyanyelvű alakjait tartalmazza, hanem ezeket a filmből kiragadott példamondatokkal, abban a kontextusban szemlélteti, amelyben az adott szó korábban előfordult. A létrejött fájl automatikusan mentésre kerül a felhasználó eszközére, így a későbbiekben is megtekinthetővé válik a lista. Egy további opcióként kínálkozik az a lehetőség, hogy a film végeztével az alkalmazás egy feleletválasztós kvízt generál azokból a mondatelemekből, melyeket a felhasználó korábban ismeretlenként jelölt meg. Ezáltal lehetősége nyílik a megszerzett tudás ellenőrzésére is.

Alkalmazásom célja tehát egy olyan alternatív lehetőség felkínálása az idegennyelvet tanulók részére, amellyel az ismeretlen szavak memorizálása szórakozással köthető össze, amely által az idegennyelv elsajátításával járó stressz csökkenthető, esetlegesen elkerülhetővé válik, így nagy eséllyel sikerélmény keletkezik, ezáltal tanulás hatékonysága is nő.

1. fejezet

Háttér

Az alkalmazás elkészítése előtt kutatást végeztem a témában lehetséges alternatívaként felmerülő szoftverekről. Tehát olyan megoldásokat kerestem, amelyek segítségével a felhasználó képes feliratok manipulálására, nyelvi fordítására, amely segítségével tudását bővítheti, elmélyítheti. Ezen kutató, illetve háttérmunkát rendkívül fontosnak tartom, hiszen ezáltal betekintést nyerhetünk a piaci igényekbe, következtetéseket vonhatunk le a már elérhető programok palettájából. A kutatás során elsődleges forrásként az internetet használtam, tehát a jelen dolgozatban bemutatott opciók nagy részben weboldalakról származnak. Itt rengeteg hasonló célokat szolgáló szoftver, honlap található, azonban ezek funkcionalitása eltér az általam fejleszteni kívánt alkalmazástól. Ezek mellett fontosnak tartom megemlíteni azt a tényt is, hogy az előzőekben említett, általam elkészített alkalmazás ingyenesen elérhető lesz, szemben a világhálón található hasonló alkalmazásokkal.

Kutatásom első részében olyan programokat kínáló megoldásokat kerestem, melyek képesek a feliratsímlők szövegét felismerni, valamint lefordítani. Erre számtalan online alkalmazás, illetve felület képes. Ezek a megoldások többnyire ingyenesek, mivel a weblapok rendszerint nagyméretű reklámfelületeket tartalmaznak. A működésük egyszerű: a felhasználó vagy drag&drop módszerrel, vagy tallózással kiválasztja az általa lefordítani kívánt megfelelő kiterjesztésű fájlt, amelyet feltölt a weboldalra. Ezután a képernyőn pillanatok alatt megjelenik az eredeti, valamint a lefordított szöveg. A felhasználó itt kiválaszthatja, módosíthatja a fordítás nyelvét, mind a fordítandó, mind pedig a fordított nyelvet. Legtöbbször nyelvfelismerő szolgáltatás is működik, amely nagy biztonsággal

képes meghatározni a forrásfájl nyelvét, ezzel is gyorsítva a fordítás folyamatát. Ha a felismerő hibázna lehetőség van manuálisan is megadni a nyelvet. Ezen megoldások nagy előnye, hogy gyorsak, könnyedén elérhetőek, illetve nem szükséges semmiféle külön alkalmazás telepítése a számítógépre. Működésükhöz csupán egy böngésző, valamint internetkapcsolat szükséges, így ezek akár mobileszközökön is használhatók. E megoldás nagy hátránya a pontatlanság, mivel a szöveget online fordítók segítségével alakítja át az alkalmazás. Az így keletkezett mondatok legtöbbször helytelenek, sok bennük a nyelvtani hiba, legtöbbször értelmetlenek. További hátrányuk, hogy működésükhöz elengedhetetlen az internetkapcsolat. Mivel fordításhoz feliratfájl feltöltése szükséges, így ebből adódóan gyenge internetelés esetén a sebességük drasztikusan lecsökkenhet, még akkor is, ha maguk a fájlok méretben nem túl nagyok.

Egy másik opcióként kínálkoznak a fentebb említett megoldáshoz nagyon hasonló szoftverek. Működésük majdnem analóg az online szövegfordítókkal, azonban lényeges különbség, hogy a fordítást videolejátszás közben is képes végrehajtani. Online ingyenesen elérhetőek, telepítést követően teljes értékű videolejátszó alkalmazásként képesek funkcionálni. Megfelelő médiafájl kiválasztása után lehetőség van feliratok fájlból történő megjelenítésére is. Az alkalmazás online fordítók segítségével alakítja át az eredeti szöveget a célnyelvre. Működésük gyors, hatékony, azonban felhasználói felületük sokszor hiányos, nem felel meg a mai kor elvárásainak. Továbbá, mivel a felirat sorait egyben kezelik, így a fordítások legtöbbször pontatlanok, hiányosak, valamint nyelvtanilag helytelenek. Fő előnyük tehát, hogy filmnézés közben is képes lefordított feliratokat megjeleníteni úgy, hogy nem szükséges a feliratokat előzőlegesen lefordítani. Egyes fajták arra is lehetőséget biztosítanak a felhasználónak, hogy a filmhez saját fordításokat készítsen. Mivel a használatukhoz az alkalmazást kötelező telepíteni a számítógépre, így működéséhez csak akkor szükséges internetkapcsolat, amikor a fordítás történik. Ekkor kevesebb az adatforgalom, mint az előzőleg említett megoldásnál, mivel nem szükséges fájlok feltöltése egy külső szerverre. Hátrányként megemlíthető az a tény, hogy a szoftverek a feliratokat csak egészben, legfeljebb soronként tudják lefordítani, így a keletkező szöveg rendkívül pontatlan.

Harmadik lehetőségként egy speciális alternatívát ismertetnék. Ezen belül olyan weboldalakat, szoftvereket kell megemlítsék, ahol a fordítás nem gépi módon, automatikusan,

esetleg mesterséges intelligencia segítségével történik, hanem valós emberek, személyek által. Egy weboldalt kiemelnék, és segítségével ismertetném a fent említett fordítási módszer működését. Az oldal neve *Amara*, amely weboldalán megtalálható a részletes leírás, ismertető, amely bemutatja a felhasználónak az oldal használatát. Alapvetően kétféle felhasználó jogosultsági szintet különböztethetünk meg. Az első az ingyenes, ahol a felhasználó csak az oldal alapfunkcióit éri el. Ez kimerül feliratok készítésében, videófájlokra illesztésében, illetve a létrejött feliratok manuális lefordításában. Ezek egy online felületen keresztül tehetők meg. Előnye, hogy gyors, kényelmes, felhasználóbarát, kiválthatóak vele a komplikált videoszerkesztő szoftverek bizonyos funkciói. Hátrányai közt említhető, hogy csak online érhető el, így folyamatos internetkapcsolatra van szükség, illetve maga a fordítási funkció teljes mértékben a felhasználó tudásán alapul, neki semmiféle nyelvi támogatást nem nyújt. A másik felhasználói szint díjköteles. Ezt igénybe véve a felhasználó azon kívül, hogy természetesen használhatja az ingyenes funkciókat, rengeteg új opció is nyílik előtte. Ezek közül a legfontosabb az, ahol videófájlok feltöltése után különböző anyanyelvű segítők készítenek feliratokat az általunk feltöltött videóhoz. Ez azonban idő- és erőforrás igényes, használata -az oldal szerint- kifejezetten vállalatoknak ajánlott. Ezen módszer hatalmas előnye, hogy a készített felirat nyelvtanilag és szemantikailag biztosan helyes, gépi fordítási hibáktól mentes. Gyenge pontja azonban az, hogy otthoni, felhasználói használatra kevésbé alkalmas, mint a fentebb említett opciók, mivel a felirat precíz elkészítése az emberi erőforrás bevonása miatt időigényes, valamint költséges. Továbbá kevésbé hatékony nyelvtanulásra, hiszen a felhasználónak a lefordított felirattal semmiféle interakciója nem lehet, az készen kerül elé. Emiatt, ha filmnézésre szeretnénk használni, ezt tehetjük egyszerűen anyanyelvű felirattal, ami kevésbé ösztönzi a felhasználót tanulásra, gondolkodásra, problémamegoldásra.

Összességében megállapítható, hogy olyan, kimondottan nyelvtanulást segítő alkalmazás, amelynek funkciói az általam elkészített programoméhoz hasonlóak, nem létezik, illetve kutatásom során nem találkoztam eggyel sem.

1.1. Architektúrális áttekintés

Kutatás után az alkalmazás megvalósításához szükséges technológiákat gyűjtöttem össze. Mivel ezt egy asztali, telepíthető szoftvernek szántam, így a *Java* nyelv, azon belül is a *Java 8* mellett döntöttem [2]. A választás azért előnyös, mivel a nyelv platformfüggetlen, így különböző operációs rendszerekre is telepíthető lesz az alkalmazás, valamint nagy mennyiségű API-k állnak rendelkezésre. A projekt, illetve a függőségek kezelésére *maven 3.6.0*-as verzióját használtam [1]. A külső könyvtárak kezelése ezzel gyorsabbá, egyszerűbbé válik. A szoftver videólejátszó funkcióit a *vlcj* keretrendszer segítségével valósítottam meg, amely a népszerű *VLC* videólejátszó alkalmazáshoz egy natív hívásokkal operáló *API*, amely segítségével lehetőség nyílik a *VLC* funkcionálisait *Java* nyelven használni [4]. A program grafikai felülete *Java Swing* segítségével íródott, s mivel a *vlcj* is támogatja az ebbe történő beágyazást, így egyszerű volt azt kiegészíteni, továbbfejleszteni. Az adatok tárolására szükségem volt egy adatbázis kezelő rendszerre. Mivel a szükséges adatbázis struktúra egyszerű, illetve az elvégzendő műveletek tárháza csekély, ezért végül a *H2* adatbázist választottam. Ennek nagy előnye, hogy képes *in-memory* adatbázis kezelésére, valamint a használatához nem szükséges további csomagok feltelepítése a számítógépre, egyszerűen a megfelelő *.jar* fájl importálásával elérhetővé válik az adatbázis. A program egyik legfontosabb funkciója, a szavak, mondatelemek fordítása a *Microsoft Translate API* beiktatásával valósult meg. Az API képes több, mint 60 nyelvfelismerésre, illetve fordításra. Használata egyszerű, működése gyors, pontos, illetve ami még fontos volt a projekt szempontjából, hogy képes egynél több találat megjelenítésére is.

A fejlesztés *Windows 10* operációs rendszer alatt történt. A programozáshoz, futtatáshoz, teszteléshez *IntelliJ IDEA* integrált fejlesztői környezetet használtam. A szoftver verziókövetése, illetve a fájlok tárolása *GitHub* repository-n keresztül valósult meg.

2. fejezet

Az alkalmazás funkcionalitásai

2.1. Videólejátszó alapfunkciók

A alábbi fejezetben az elkészült alkalmazás főbb funkcióit mutatom be, fejtem ki részletesen, külön kitérve a fontosabb, érdekesebb részekre.

A szoftver alapját, magját adó komponens a *VLC* java-n alapuló keretrendszer segítségével valósult meg. Első lépésként a *VLC* videólejátszó megfelelő operációs rendszerre való feltelepítése szükséges. Windows esetén lehet 32 illetve 64 bites verzió. E lépést kötelező megtenni, mivel az fejlesztett alkalmazás a *VLC* lejátszó funkcióit veszi igénybe, nélküle a szoftver nem működik. Ezután a *vlcj* nevű open source projektet vettem igénybe. Ez Java könyvtárakat ad a *VLC* média lejátszóhoz, így annak majdnem az összes natív funkciója elérhetővé válik a *LibVLC*-n keresztül. Ezen kívül egy keretrendszer használatát is elérhetővé teszi, amely egy egyszerű, magas szintű programozási modellt biztosít, ami magába foglalja a natív könyvtárakhoz való hozzáférést. E keretrendszer amennyire csak lehetséges megvédi a felhasználóját az eredeti könyvtár helytelen használatától, ami akár a fejlesztett alkalmazás összeomlását is okozhatja. Segítségével a fejlesztés magas szinten történhetett. Az alkalmazás elindítása után egy egyszerű videó lejátszó kezelőfelülete tárul elénk, amely a 2.1 ábrán látható.

A képernyő bal alsó sarkában találhatóak meg a videó lejátszását szabályozó gombok. Ezek rendre az előző fejezet, visszatekerés, megállítás, pillanatmegállítás, indítás, előre tekerés, valamint következő fejezet. Középen egy csúszka kapott helyet, amely a hangerőszabályzást teszi lehetővé. Ettől jobbra találhatóak a némító, képernyőfelvétel,



2.1. ábra. Az alkalmazás kezelőfelülete

médiaválasztó, teljes képernyő, felirat választó, fordítás nyelve, illetve az online feliratok gombok. Ezek működéséről későbbi fejezetekben írok részletesen. Feljebb helyezkedik el a keresősáv, és egy számláló balra, valamint egy jobbra. A bal számláló a lejátszott videó aktuális időpillanatát jelzi, másodperces pontossággal. A csúszka segítségével egyszerűen, intuitívan tekerhetünk a videóban. A jobboldali számláló az aktuális fejezetet mutatja. Az alkalmazás felső részén a menüsáv található, rajta öt különböző menüponttal. Az első segítségével médiafájlokat tallózhatunk, illetve kiléphetünk a szoftverből. A második lehetőség egy segítség opció, a harmadik az ellenőrző kvíz kitöltését teszi lehetővé. A *generate* menüt kibontva PDF fájlt generálhatunk az általunk ismeretlennek vélt szavakból. Felül az utolsó lehetőség a beállítások menüpont, ahol a felirat méretét, illetve késleltetését állíthatjuk be milliszekundumos pontossággal. Ezek áttekintése után már tudunk média fájlokat hozzáadni a szoftverhez. Ehhez kattintsunk az *Eject* gombra az alsó panelről, vagy válasszuk a *Media, Play File...* menüpontot. Ekkor megnyílik a felhasználó előtt egy tallózó ablak, ahol böngészhetünk a saját fájlrendszerünkben. Alap beállítás szerint csak a média fájlok jelennek meg az ablakban, azonban ez bármikor felülírható. Itt lehetőség van keresésre, részletes, vagy listás megjelenítésre, feljebb navigálásra, főoldalra való visszatérésre. A szoftver képes rengeteg különböző kódolású videó-, és hangfájl lejátszására. Kilépésre a *Cancel* gomb megnyomásával van lehetőség. Ha megtaláltuk a nekünk megfelelő fájlt, a *Play* gombra való kattintással kezdetjük el a lejátszást. Az ablakot bármikor átméretezhetjük, amivel a benne lejátszott tartalom is méretet vált, fekete sávokkal kitöltve a képarányának nem megfelelő részeket. Lehetőségünk van a lejátszó

némítására, a *Toggle Mute* gombra való kattintással. Ezzel egy időben a gomb ikonja is a megfelelő állapotba vált, jelezve ezzel, hogy a lejátszás le van-e némítva, vagy sem. Az ezt megvalósító kódrészlet a 2.1-es ábrán található.

Kódrészlet 2.1. Némítást implementáló kódrészlet

```
toggleMuteButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        mediaPlayer.mute();
        if(mediaPlayer.isMute()){
            toggleMuteButton.setIcon(new ImageIcon(getClass()
                .getClassLoader()
                .getResource("icons/sound.png")));
        } else {
            toggleMuteButton.setIcon(new ImageIcon(getClass()
                .getClassLoader()
                .getResource("icons/sound_mute.png")));
        }
    }
});
```

2.2. Teljes képernyős mód

Az eddig elkészült alkalmazás már képes videofájlok lejátszására, illetve alapvető vezérlési funkciókat is ismer, azonban a szoftver egy fontos része hiányos, tekintve, hogy elsősorban film lejátszására ajánlott. Ez pedig a teljes képernyős mód. Az alkalmazás, amint a felhasználó megnyomja a teljes képernyős mód gombot, kitölti a képernyőfelület egészét, eltüntetve ezzel a menüsávot és a kezelőpanelt.

Az új funkció alkalmazása azonban egy új problémát vetett fel: ha a képernyő egésze foglalt, és nem jelenik meg a kezelősáv, akkor a felhasználó nem tudja a teljes képernyős módot bezárni. Ennek megoldására egy általánosan használt módszert alkalmaztam. Ha a felhasználó a kurzort a képernyő alsó felére húzza, a vezérlőpanel megjelenik, és megjelenítve is marad mindaddig, amíg a felhasználó el nem távolítja az alsó területről a kurzort. Itt egyszerűen a teljes képernyős mód gombjára kattintva kiléphetünk abból. En-

nek a megvalósításáért, illetve a panel megjelenítésének kezeléséért egy *MouseListener* felel. Ahogyan a kódrészlet 2.2-n látható, amennyiben az alkalmazás teljes képernyős módban van, a figyelő ellenőrzi a kurzor aktuális Y koordináta szerinti pozícióját, és ha ez egy bizonyos szám alatti, vagy feletti, akkor a szerint jeleníti meg, valamint rejt el a vezérlőpanelt, illetve ennek megfelelően igazítja a vásznat és a feliratokat.

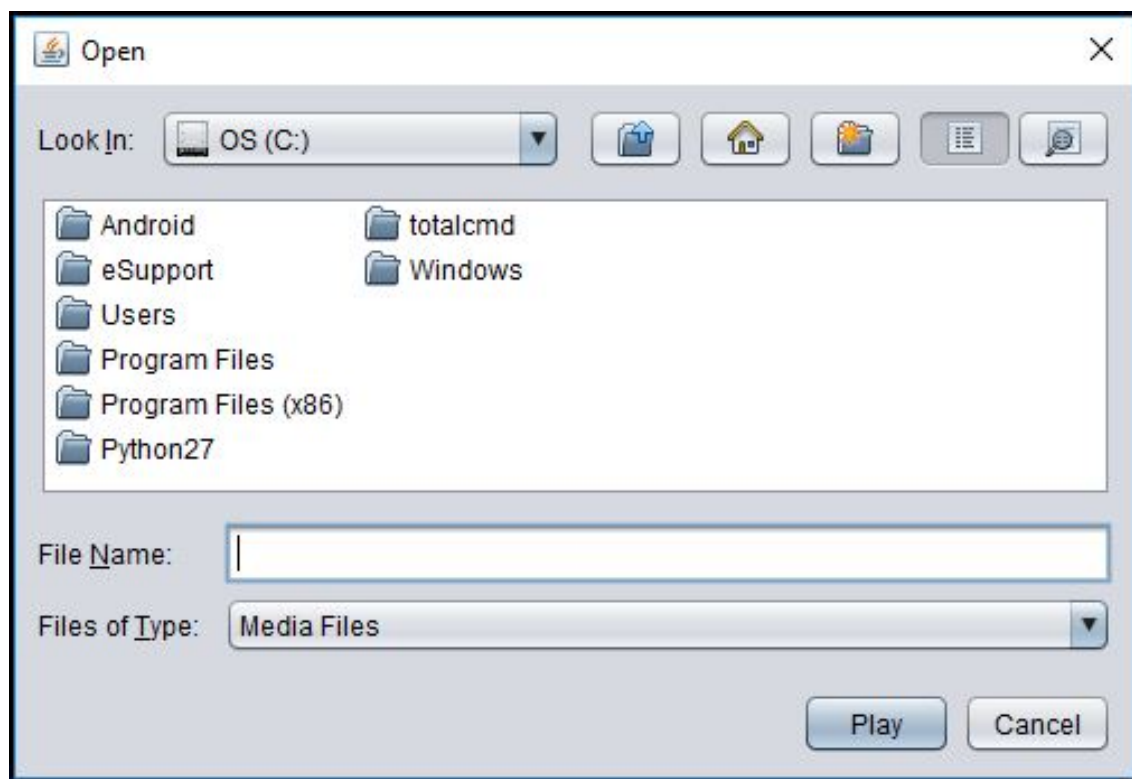
Kódrészlet 2.2. A *MouseListener* megvalósítása

@Override

```
public void mouseMoved(MouseEvent e) {  
    if (mediaPlayer.isFullScreen()) {  
        if (e.getYOnScreen() < Toolkit  
            .getDefaultToolkit()  
            .getScreenSize().height - 5  
            && controlsPanel.isVisible()) {  
            controlsPanel.setVisible(false);  
            ((SubtitleOverlay)(mediaPlayer.getOverlay()))  
                .decreaseYOffset(controlsPanel.getHeight());  
        } else if (e.getYOnScreen() >= Toolkit  
            .getDefaultToolkit()  
            .getScreenSize().height - 5  
            && !controlsPanel.isVisible()) {  
            controlsPanel.setVisible(true);  
            ((SubtitleOverlay)(mediaPlayer.getOverlay()))  
                .increaseYOffset(controlsPanel.getHeight());  
        }  
    }  
}
```

Ezen felül további kényelmi funkciókat is implementáltam a hatékonyabb teljes képernyős mód kezeléséért. A mód aktiválható a vászonra történő dupla kattintással, és meg is szüntethető ugyanezzel a módszerrel. Ehhez szintén egy *MouseListener*-t hívtam segítségül, amely figyeli az egymás utáni kattintások számát, és ha ez az érték kettővel egyenlő, be- valamint kilép a teljes képernyős módból. Egy másik megvalósított segédfunkció az *Esc* gombbal történő vezérlés. Amennyiben az alkalmazás teljes képernyőn fut, a mód megszakítható az *Esc* gomb egyszeri megnyomásával. Mindkét lehetőség kényelmesebb, gyorsabb használatot tesz lehetővé, ezzel is növelve a felhasználói komfortérzetet.

2.3. Média- és felirutfájlok választása



2.2. ábra. Médiafájl választó

A médiafájlok lejátszása a videólejátszó alkalmazás elengedhetetlen részét képezi, hiszen enélkül a felhasználó képtelen lenne, vagy csak komplikált módszerek igénybevételével tudná megtekinteni videó fájljait. A feliratok megjelenítése a szoftver egy másik sarkallatos pontja. E funkció teszi lehetővé a felhasználó számára, hogy filmjeit idegen nyelven feliratozva tekinthesse meg. Mindkettő megvalósítása egy tállózó segítségével történik, amely a megszokott módon, gyorsan, kényelmesen enged utat a fájlválasztásnak. Alapesetben a választó a felhasználó *home* könyvtárában nyílik, onnan navigálva kereshetők a tartalmak. A fájlok kiválasztása után azok azonnal megjelennek a képernyőn, valamint videó fájlok esetén el is indulnak. A tállózók megvalósítása *Java Swing* komponensek, azon belül *JButton*, *JFileChooser* segítségével történtek. A médiafájl választás folyamata a *Load/Eject media* gomb megnyomásával indítható el, amely a vezérlőpanel jobb alsó részén elérhető, és a 2.2-es ábrán látható ablakot jeleníti meg. A gombhoz használt ikont a *resources/icons* könyvtárból importáltam. A fájlválasztáshoz a könnyebb használat érdekében szűrőket definiáltam, amelyek a fájlok kiterjesztése alapján szűrik

meg az éppen aktuális könyvtár tartalmát, és jelenítik meg csak azokat az elemeket, amelyek a feltételnek megfelelnek. Ezek külön-külön a videó, audió, lejátszási lista, illetve a médiafájlokra való szűrést teszik lehetővé. Ezeken felül elérhető egy *All files* opció is, amely filterezés nélkül az összes elérhető fájlt kilistázza a felhasználó számára. Ezen szűrőket a fájlválasztó *Files of Type* címke mellett található legördülő listából választhatjuk ki. Amennyiben ennél specifikusabb módon szeretnénk keresni, lehetőségünk van a keresőmező használatára. Ezt a *File Name* felirat mellett található mezőbe írt kulcsszavak segítségével tehetjük meg. Ha gyorsabban szeretnénk navigálni a könyvtárak között, ezt megtehetjük a választó felső részén fellelhető legördülő listából kiválasztható elemekkel. Itt megtalálhatók a számítógépen elérhető fő könyvtárak, mint például a különböző meghajtók, asztal és egyéb nagyobb méretű mappák. A lista helyzetétől jobbra felfedezhető néhány további gomb. Ezek rendre: egy szinttel feljebb lépés a könyvtárhierarchiában, vissza a *home* könyvtárba, új mappa létrehozása, lista- illetve részletes nézet közötti váltás. Ha megtaláltuk az általunk lejátszani kívánt fájlt, kijelölés után a *Play* gomb megnyomásával, illetve a fájlon történő dupla kattintással indíthatjuk el azt. Ugyanezen komponens használatával valósítottam meg a felirattípus kiválasztásának kezelését is. Az ablakot a kezelőpanelen fellelhető *Select subtitle* gombra történő kattintással jeleníthetjük meg. A különbség csupán annyi a két ablak között, hogy a felirattípus-választó ablakban más szűrők definiáltak. Ez értelemszerűen a felirattípusok kizárólagos megjelenítésére ad lehetőséget. Mindkét ablak esetében kényelmi funkcióként implementálásra került az elérési útvonal megjegyzése. Amikor a felhasználó fájlt választ, legyen a média, vagy felirat, az alkalmazás elmenti a fájl elérési útvonalát, és ez alapján állítja a tallózó az aktuális kiinduló könyvtárát.

Kódrészlet 2.3. Aktuális könyvtár beállítása

```
if ( actualFile != null )
    { fileChooser.setCurrentDirectory ( actualFile );
    }
```

A 2.3-as kódrészlet néhány sorában mindösszesen annyi történik, hogy amennyiben az aktuális fájl, azaz az utolsóként kiválasztott fájl nem üres, akkor a tallózó aktuális könyvtára a fájl tartalmazó mappa lesz. Ugyanez a kódrészlet megtalálható a felirattípus választó kódjában is. Mindezekon felül a fájlválasztó másik módon is megjeleníthető. A

felső menüsávban a *Media* menüpont alatt található *Play Media...* menü kiválasztásával a tallózó megjelenik a képernyőn. Itt akár gyorsbillentyűvel, úgynevezett mnemonikkal is hatékonyabbá tehetjük a fájlválasztást elindítását. A menüsávban található *Media* az "m", a *Play File...* menü pedig az "f" billentyűk lenyomásával hívható elő. Az implementált két tallózó a fent említett részletek alapján alkalmas fájlok gyors és hatékony szelekciójára, olyan módon, hogy azt egy átlagfelhasználó is könnyedén, kevés kutatással megteheti.

2.4. Feliratok kezelése

A feliratok megjelenítése az alkalmazás magját alkotó funkciók közé sorolható, hiszen enélkül maga a nyelvtanulási folyamat nehezen lenne megvalósítható. Tehát e lehetőség használhatósága kulcsfontosságú a projekt tekintetében. Ennek a működését, implementálását bemutató fejezet logikailag két részre osztható: az *.srt* kiterjesztésű fájlok beolvasása, kezelése, illetve a feliratok képernyőre történő kirajzolása.

2.4.1. Feliratfájlok beolvasása

Ahhoz, hogy a szoftver képes legyen feliratokat megjeleníteni, tudnia kell azokat a felhasználó számítógépéről felolvasni, kezelni. Az alkalmazás jelenleg csak a *SubRip*, azaz *.srt* kiterjesztésű fájlok kezelését támogatja, mivel az említett formátum a legelterjedtebb típusok közé sorolható. Előállításuk, interneten való fellelésük rendkívül könnyű, gyors, emiatt nagy népszerűségnek örvend mind a felirat elkészítői, mind a felirat használói közt. Strukturális felépítése egyszerű, a 2.4-es kódrészleten látható.

Kódrészlet 2.4. A *.srt* kiterjesztésű fájlok felépítése

```
1
00:00:01,100 --> 00:00:01,500
First line
Second line

2
00:00:25,0004 --> 00:00:30,200
Additional text ...
```

A fájl blokkokra osztható, amelyek egy-egy megjelenítendő feliratrészletnek felelnek meg. Minden ilyen rész első sorában egy sorszám található. Ezek a részletek azonosítására szolgálnak. A következő sorban a kezdő-, illetve a végidőpontok helyezkednek el, közöttük egy szigorúan „->” alakú nyílal, előtte és utána szóközökkel határolva. Az időpontokat konvenció szerint óra:perc:másodperc,ezredmásodperc formátumban kötelező megadni. Ezután a vászonra kirajzolandó sorok találhatóak. Itt fontos megemlíteni, hogy a fájl érzékeny a sortörésekre, hiszen ezek alapján tesz különbséget a más-más sorokba kerülő szövegek között. A szöveg sora(i) után legalább egy üres sornak kell szerepelnie. Amennyiben ez hiányzik, a következő blokk az előző szövegrészébe kerül. A feliratfájlok kezeléséért az *srt* package-ben található *Java* osztályok felelősek. A fő funkcionalitásokat, azaz a feliratok beolvasását, feldolgozását az *SRTReader.java* nevű osztály végzi az itt található *read*, illetve *parse* metódusok segítségével. Az előbbi egy *File* objektumot vár, amit a soronként dolgoz fel, és minden soron alkalmazza a *parse* metódust, ami egy *TreeSet*-hez adja hozzá a feldolgozott blokkoknak megfelelő *Java* objektumokat. Részletes működése a 2.5-ös kódrészleten látható.

Kódrészlet 2.5. Srt fájlok feldolgozása

```
public static SRTInfo read(File srtFile)
throws InvalidSRTException, SRTReaderException {
    if (!srtFile.exists()) {
        throw new SRTReaderException(srtFile.getAbsolutePath()
            + " does not exist");
    }
    if (!srtFile.isFile()) {
        throw new SRTReaderException(srtFile.getAbsolutePath()
            + " is not a regular file");
    }

    SRTInfo srtInfo = new SRTInfo();
    try (BufferedReader br = new BufferedReader
        (new InputStreamReader(
            new FileInputStream(srtFile),
            StandardCharsets.UTF_8))) {
        BufferedLineReader reader = new BufferedLineReader(br);
        while (true) {
            srtInfo.add(parse(reader));
        }
    }
```

```
    }  
    } catch (EOFException e) {  
    } catch (IOException e) {  
        throw new SRTReaderException(e);  
    }  
    return srtInfo;  
}
```

A függvény képes kivételek kezelésére is. Amennyiben az átadott fájl nem létezik, például végrehajtás közben törölték, illetve, ha a fájl nem reguláris fájl, azaz nem fájl-rendszerben tárolt bájtok sorozata, az alkalmazás erről megfelelő hibaüzenetet továbbít.

2.4.2. Feliratok megjelenítése

Mivel a feliratokat tartalmazó fájlokat az alkalmazás már be tudja olvasni, valamint képes azok kezelésére, a soron következő megvalósítandó funkció ezen feliratok képernyőn történő megjelenítése, olyan módon, ahogyan az egy általános videólejátszó szoftvertől elvárható. Fontos szempont tehát az időzítés, illetve a feliratok megfelelő formátumú kirajzolása. Ezen felül követelmény az is, hogy a megjelenő feliratokra kattintva láthatóak legyen az elérhető fordítások is. Ugyan a *vlcj* függvénykönyvtárában található metódus, amely a feliratsímlék beállítását végzi, de mivel e funkció implementációja teljesen rejtett a *Java*-s környezet előtt, így nem volt arra lehetőség, hogy az egyes kirajzolt mondatrészekhez olyan figyelőt rendeljek, amely az egér kattintásait elemzi. Emiatt saját felirat megjelenítő komponenst kellett létrehozni, ami segítségével kattintásra elkérhető a szóból alkotott *String* objektum, így majd a fordítás is megvalósulhat. A funkcióért a *SubtitleOverlay* osztály felelős, amely elkészíti a megjelenítendő szöveget, és ki is rajzolja azt. Az osztály őse egy *AWT* komponens, egy *Window*, amely egy láthatatlan réteggént kerül a szoftver felhasználói felületére. Egyetlen feladata, a feliratok megjelenítése, pozicionálása, interakció megvalósítása. A felirat megjelenítésének megvalósítását a *paint* függvény hajtja végre, amely a 2.6 kódrészleten látható.

Kódrészlet 2.6. A felirat megjelenítése

```
@Override  
public void paint(Graphics g) {  
    super.paint(g);
```

```
g.clearRect(0, 0, this.getWidth(), this.getHeight());

Graphics2D g2 = (Graphics2D) g;
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);

g2.setFont(new Font("Serif", Font.PLAIN, fontSize));
g2.setColor(new Color(255, 255, 255));

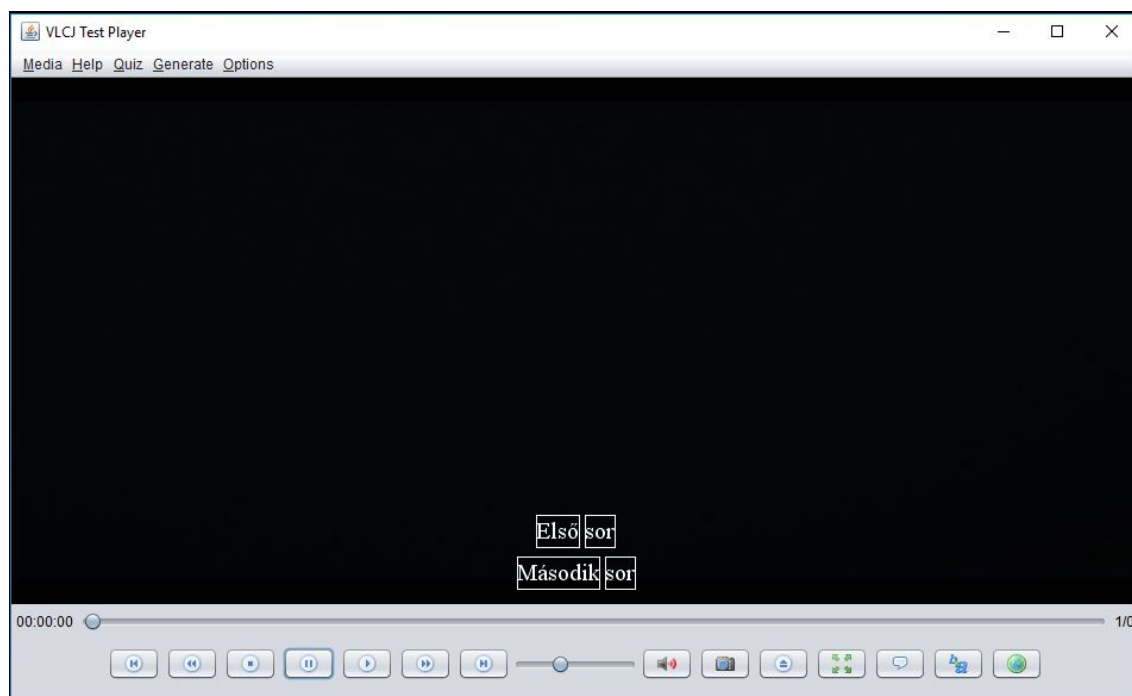
if(actSubtitle == null) {
    return;
}

calculateSubtitleBoundingBox(g2, actSubtitle);

for (Entry<String, Rectangle2D> box : boundingBoxes) {
    g2.drawString(box.getKey(),
        (int) box.getValue().getX(),
        (int)(box.getValue().getY()+box.getValue().getHeight()));
}
}
```

A kattintásra történő szöveg visszaadásának megvalósítása az egyes szavakat befoglaló kattintható, láthatatlan téglalapok segítségével történik. Az antialiasing, betűtípus, valamint a betűszín beállítása után a meghívott *calculateSubtitleBoundingBox* végzi el a mondatelemek köré szánt négyszögek kiszámítását. A szavak, valamint az őket körülvevő téglalapok egy kulcs-érték párokat tároló Map-be kerülnek. A map elemeit alkalmazás a kattintás pozíciójának vizsgálatakor elemzi, és amennyiben a kattintás valamely map-ben fellelhető négyszög területén belülre esik, a szoftver visszaadja a hozzá tartozó kulcsot, jelen esetben a *String* objektumot, amely a kattintott mondatelemet tartalmazza. Így tehát lehetőség nyílik a felhasználó egerrel történő kattintásait nyomon követni, és ennek megfelelően elvégezni a szükséges fordítást. A 2.3-as ábrán látható módon, a kód minimális módosításával megjeleníthetők a befoglaló téglalapok, így jól látható, hogy a szoftver mely területekre figyel a felhasználó kattintásai során.

A bemutatott két funkció alkalmazásba történő implementálásával lehetőség nyílt arra, hogy a felhasználó interakcióit a szoftver dinamikusan, a film lejátszása közben kezelje,



2.3. ábra. A befoglaló négyzetek megjelenítve a szavak körül

illetve ennek alapján megfelelő fordításokat végezzen beépített fordító API segítségével.

2.5. Feliratok fordítása

Miután már lehetőségünk van arra, hogy a felhasználó szavakra történő kattintásait nyomon kövessük, a következő lépés ezek lefordítása a kívánt nyelvre. Ehhez szükségünk van egy fordító *API*-ra, azaz egy alkalmazásprogramozási felületre, amely segítségével megvalósíthatóvá válik a szavak anyanyelvre való átültetése. Ehhez a *Microsoft Translate API* interfészre esett a választásom, mivel az összes alternatíva közül ez az egyetlen ingyenes, bizonyos karakterszám eléréséig. Az *API* működésének lényege, hogy a videólejátszó alkalmazás interneten keresztül elküldi a megadott címre a fordítani kívánt szót, majd innen egy *JSON* formátumú válasz érkezik, ami tartalmazza az összes információt, -beleértve a fordításokat is- amelyet a szerver képes volt előállítani az elküldött információk alapján. A szolgáltatás többféle fordítási módot is támogat:

- fordítás, azaz több mondatos szövegek fordítása
- transliteráció, azaz nem latin betűkészletet használó nyelvek átbetűzése
- szófordítás, azaz egyes szavak szótározása

Az alkalmazás szempontjából az utóbbi lesz megfelelő, mivel ez a funkció lehetővé teszi egy szóhoz tartozó több fordítás megjelenítését is, amely a felhasználó szókincsének bővítésére rendkívül hasznos.

2.5.1. Szerverrel történő kommunikáció

A szolgáltatás használatához regisztrálást követően igényelnem kellett egy egyedi kulcsot, amely a fiókom azonosítását szolgálja. A *TranslateService.java* osztály valósítja meg az információáramlást szerver és kliens között. Mivel az *API* egy másik URL-en képes nyelvfelismerésre is, ezért az osztályba két publikus metódus került. Egy, amely a kiválasztott nyelv esetén hívódik, egy másik pedig amikor a nyelvfelismerés aktív. Működése a 2.7-es kódrészleten látható.

Kódrészlet 2.7. Függvény, amely kiválasztott nyelv esetén hívódik

```
public TranslateResponse [] PostWithGivenLanguages (
    String from ,
    String to ,
    String string ) throws IOException {
    givenLanguageUrl += from + "&to=" ;
    givenLanguageUrl += to ;

    Gson gson = new GsonBuilder ()
        . setPrettyPrinting ()
        . create () ;
    return gson . fromJson ( createResponse ( string ,
        givenLanguageUrl ) .
        body () . string () , TranslateResponse [] . class ) ;
}
```

Ebben az esetben a metódus három paraméterrel rendelkezik:

- String from, a forrásnyelv
- String to, a cél nyelv
- String string, a fordítandó szöveg

Működése közben előállít egy URL-t, amely tartalmazza a fent említett forrás- és cél nyelvet, valamint a felhasználáshoz szükséges kulcsot, beégetett módon. A *createResponse*

metódus felelős a szerverrel történő kommunikációért, beleértve a kérés elküldését, illetve a válasz átadását is. A kapott *JSON* formátumú szöveget *Gson* segítségével deszerializálja, így jön létre egy *TranslateResponse* tömb, amely a szervertől kapott választ hivatott reprezentálni. Ez tartalmazza az elérhető fordításokból alkotott listát, mellyel tovább dolgozhatunk.

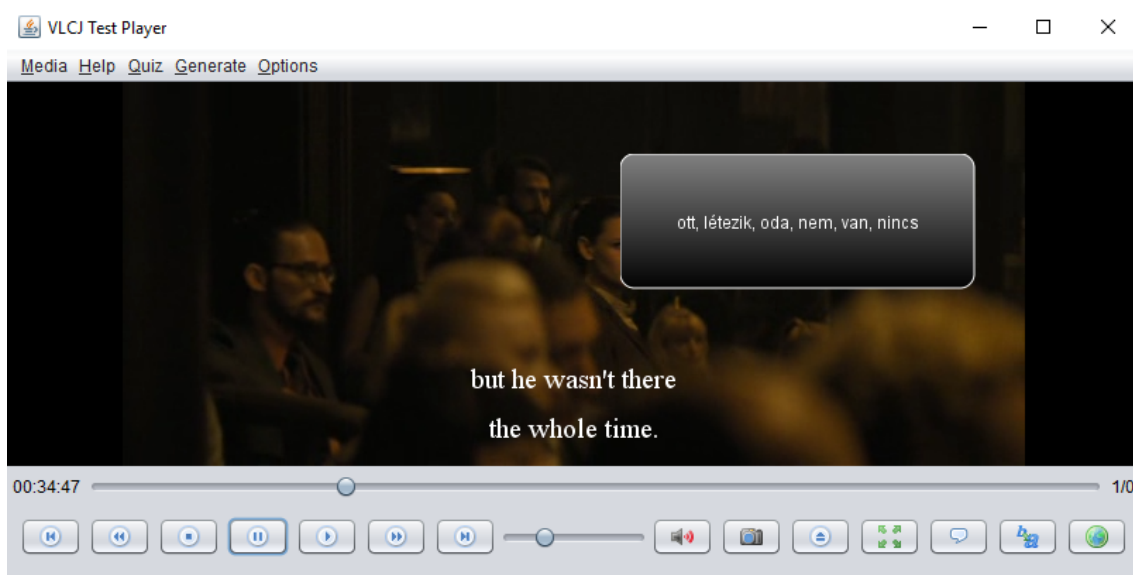
Kódrészlet 2.8. Függvény, amely detektált nyelv esetén hívódik

```
public DetectedLanguageResponse [] PostWithLanguageDetection (
    String string ) throws IOException {
    Gson gson = new GsonBuilder ()
        . setPrettyPrinting ()
        . create ();
    return gson . fromJson ( createResponse
        ( string ,
        " https :// api . cognitive . microsofttranslator . com
        / detect ? api - version = 3.0 " )
        . body () . string () ,
        DetectedLanguageResponse [] . class );
}
```

A 2.8-as ábrán látható a detektált nyelv esetén meghívott függvény. Az előző metódus-hoz képest annyi különbség fedezhető fel, hogy a paraméterlistában csak a lefordítandó szöveg szerepel, és a POST kérés másik címre fut be. Ezen kívül a feldolgozott *JSON* egy *DetectedTranslateResponse*-okat tartalmazó tömbbe kerül.

2.5.2. Fordítások megjelenítése

A fordítások és annak egyéb adatai deszerializáció után kinyerhetők egy-egy objektum-tömbből. A következő feladat ezek felhasználóbarát megjelenítése volt. További kritérium, hogy komponens nem szakíthatja meg a videó folytonosságát, illetve nem zavarhatja a nézőt a jelenléte. Mindezek miatt egy felugró ablakot készítettem, amely mondatelemre történő kattintás esetén megjelenik, majd néhány másodperc elteltével eltűnik a képernyőről. Az ablak tartalmazza az aktuális szó összes elérhető fordítását, ha pedig ilyen nem található a *No translations available* felirat jelenik meg. Ennek a megjelenítéséért a *PopupMenuBuilder.java* osztály felelős. Maga az ablak egy *JPanel*-ből származik, melyet a *paintComponent* metódus segítségével rajzol az alkalmazás a képernyőre. A build-



2.4. ábra. A megjelenő felugró ablak

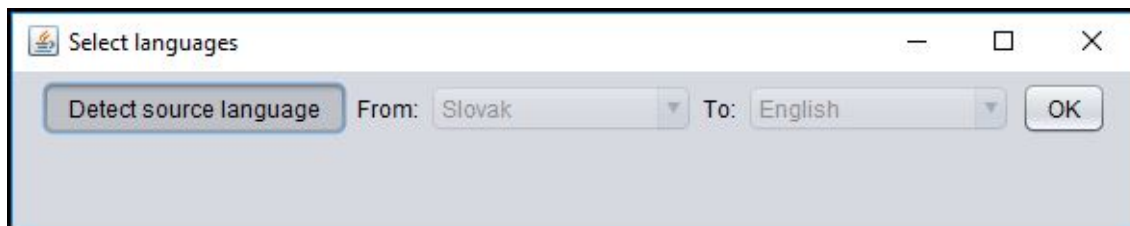
der osztályt olyan módon készítettem el, hogy segítségével könnyedén, és kényelmesen lehessen felugró ablakokat megjeleníteni a felhasználói a felületen. A 2.9-es kódrészleten látható módon, pozíció, időtartam, üzenet megadásával jeleníthető meg a komponens. Jelen esetben a kattintás helye felett, három másodperces időtartammal jelenik meg az üzenet, amely a fordítást tartalmazza.

Kódrészlet 2.9. A *PopupMessageBuilder* használata

```
new PopupMessageBuilder ()  
    .at(new Point((int) e.getPoint().getX(), e.getY()  
    -(int)( Toolkit.getDefaultToolkit().getScreenSize().height*0.1)))  
    .withDelay(3000)  
    .withMessage(word.getMeaning())  
    .show();  
  
new PopupMessageBuilder ()  
    .at(new Point((int) e.getPoint().getX(), e.getY()  
    -(int) ( Toolkit.getDefaultToolkit().getScreenSize().height*0.1)))  
    .withDelay(3000)  
    .withMessage("No translations available")  
    .show();
```


A felugró ablak megjelenését filmnézés közben a 2.4-es számú ábrán láthatjuk.

2.6. Nyelvek kiválasztása



2.5. ábra. A nyelvválasztó kezelőfelülete

Miután az alkalmazás képes kezelni a fordításokat, a következő lépés a fordítás nyelvének megadása volt. Mivel a szoftver képes nyelvfelismerésre is, ezért a nyelvválasztó funkciót úgy valósítottam meg, hogy a felhasználó képes legyen e funkció be- és kikapcsolására, illetve egyéni nyelvpárok kiválasztására is. Ez csak azután tehető meg, hogy sikeresen hozzáadunk egy felirutfájlt a lejátszóhoz. Ha ezt megtettük a kezelőfelület elérhető a *Select translator languages* gombra történő kattintással. Ekkor egy panel nyílik a felhasználó képernyőjére, amely a 2.5-ös ábrán látható módon, egy *JToggleButton*-on kívül kettő *JComboBox*-ot tartalmaz egy *JButton* mellett. Alapesetben a nyelvfelismerés mindig aktív, így a két legördülő listából nem lehet választani. Ez kikapcsolható a *Detect languages* gombra való kattintással. Ekkor elérhetővé válik a két lista is, ahol tetszőlegesen választhatunk a fordító által támogatott nyelvek közül. Választás után az *OK* gombra kattintva az alkalmazás elmenti a kiválasztott elemeket, tehát, ha bezárjuk az ablakot, majd ismét megnyitjuk azt, az eredmény a bezárás előtti állapottal megegyező. Így a megjelenített opciók mindig az aktuális, mentett konfigurációt tükrözik. A fordítás nyelvének megváltoztatása akár filmnézés közben is elvégezhető, nem szükséges hozzá a tartalom újratöltése, azonban minden új felirutfájl lejátszóhoz történő hozzáadása alaphelyzetbe állítja a kiválasztott nyelveket. Az *API* által támogatott nyelvek listája rendkívül széles, a legtöbb ismert nyelvet támogatja.

Mivel az *API* nyelvkódokat használ a nyelvek azonosítására, például *"hu"*, viszont a felhasználó számára megjelenített nyelvek teljes értékű szavak, például *"Hungarian"*, szükség volt ezek összekapcsolására. A nyelv-nyelvkód párokat egy *HashMap*-ben tárol-

juk kulcs-érték páronként, amelyeket minden egyes nyelvállításkor megvizsgálunk, és a kulcs alapján az alkalmazás beállítja az annak megfelelő kódot. Ennek megvalósítása a 2.10-es kódrészletben látható.

Kódrészlet 2.10. Aktuális nyelvek beállítása

```
while ( it.hasNext() ) {  
    Map.Entry pair = (Map.Entry) it.next();  
    if ( pair.getKey().equals(fromComboBox.getSelectedItem()) ) {  
        currentFromLanguage = (String) pair.getValue();  
    }  
    if ( pair.getKey().equals(toComboBox.getSelectedItem()) ) {  
        currentToLanguage = (String) pair.getValue();  
    }  
    it.remove();  
}
```

Mint említettem, alapesetben a nyelvdetektálás mindig aktív. Ez azt jelenti, hogy a felhasználónak nem kell a nyelvválasztással foglalkoznia, elvégzi ezt helyette az alkalmazás. Emellett ilyen esetben a célnyelv, azaz az a nyelv amire a fordítás történik, mindig a felhasználó számítógépének rendszernyelve, amelyet az alkalmazás a *user.language* változóból olvas ki. Így a a felhasználó számára legkényelmesebben, legkevesebb konfigurációval, mindemellett nagy pontossággal valószínűsíthető, hogy a fordítás mely két nyelv között fog végbemenni, hiszen az *API* nyelvfelismerő funkciója rendkívül megbízható, illetve legtöbbször az operációs rendszer nyelve megegyezik a felhasználója anyanyelvével.

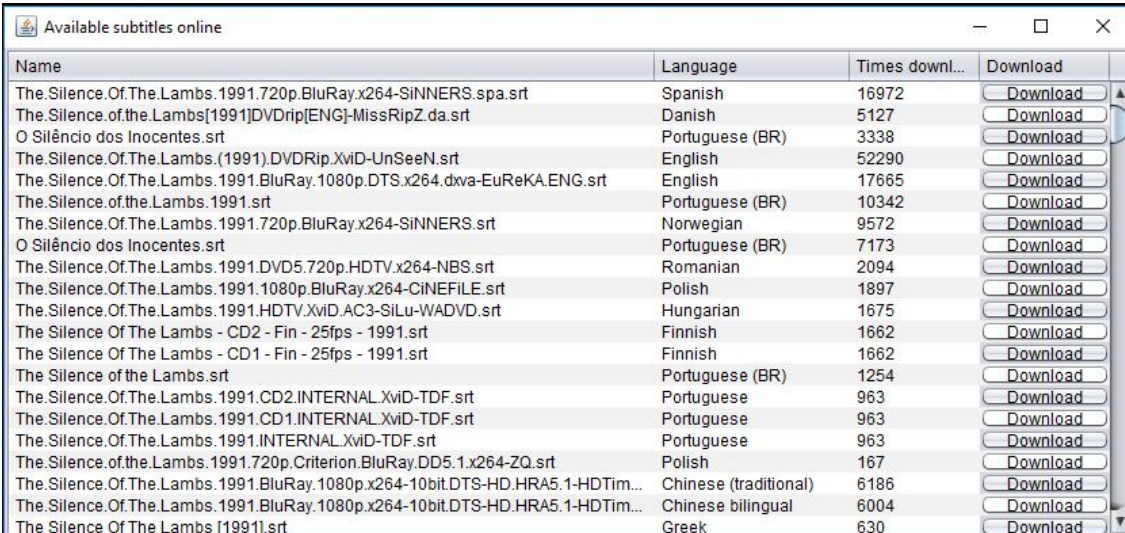
2.7. Online feliratok letöltése

Az alkalmazást a kényelmesebb és felhasználóbarátabb használatért kiegészítettem egy, az alapspecifikáción kívül eső funkcióval, mely az online feliratok keresése és letöltése. Habár korlátlan mennyiségű felirat áll rendelkezésre online, a felhasználók egy része mégsem ismeri ezek beszerzésének módjait. Legfőképpen nekik nyújthat nagy segítséget az implementált funkció, azonban mindenki számára egy kényelmes megoldást kínál. A megvalósításhoz az egyik legnagyobb felirat-adatbázissal rendelkező weboldal saját *API*-ját, az *opensub4j* nevű interfészt hívtam segítségül. A weboldalon történő ingyenes

regisztráció után már használhattam is a felületet.

Első lépésként felvettem egy új függőséget az alkalmazás *pom* fájljába. Ettől kezdve elérhetővé váltak számomra a csomagban definiált osztályok. A funkció implementálását a *SRTSearchFrame.java* osztály tartalmazza. Működése két részből áll: a feliratok keresése, illetve ezek letöltése, kitömörítése.

2.7.1. Feliratok keresése



Name	Language	Times downl...	Download
The.Silence.Of.The.Lambs.1991.720p.BluRay.x264-SiNNERS.spa.srt	Spanish	16972	Download
The.Silence.of.the.Lambs[1991]DVDRip[ENG]-MissRipZ.da.srt	Danish	5127	Download
O Silêncio dos Inocentes.srt	Portuguese (BR)	3338	Download
The.Silence.Of.The.Lambs.(1991).DVDRip.XviD-UnSeeN.srt	English	52290	Download
The.Silence.Of.The.Lambs.1991.BluRay.1080p.DTS.x264.dwa-EuReKA.ENG.srt	English	17665	Download
The.Silence.of.the.Lambs.1991.srt	Portuguese (BR)	10342	Download
The.Silence.Of.The.Lambs.1991.720p.BluRay.x264-SiNNERS.srt	Norwegian	9572	Download
O Silêncio dos Inocentes.srt	Portuguese (BR)	7173	Download
The.Silence.Of.The.Lambs.1991.DVD5.720p.HDTV.x264-NBS.srt	Romanian	2094	Download
The.Silence.Of.The.Lambs.1991.1080p.BluRay.x264-CiNEFiLE.srt	Polish	1897	Download
The.Silence.Of.The.Lambs.1991.HDTV.XviD.AC3-SiLU-WADVD.srt	Hungarian	1675	Download
The.Silence.Of.The.Lambs - CD2 - Fin - 25fps - 1991.srt	Finnish	1662	Download
The.Silence.Of.The.Lambs - CD1 - Fin - 25fps - 1991.srt	Finnish	1662	Download
The.Silence.of.the.Lambs.srt	Portuguese (BR)	1254	Download
The.Silence.Of.The.Lambs.1991.CD2.INTERNAL.XviD-TDF.srt	Portuguese	963	Download
The.Silence.Of.The.Lambs.1991.CD1.INTERNAL.XviD-TDF.srt	Portuguese	963	Download
The.Silence.Of.The.Lambs.1991.INTERNAL.XviD-TDF.srt	Portuguese	963	Download
The.Silence.of.the.Lambs.1991.720p.Criterion.BluRay.DD5.1.x264-ZQ.srt	Polish	167	Download
The.Silence.Of.The.Lambs.1991.BluRay.1080p.x264-10bit.DTS-HD.HRA5.1-HDTim...	Chinese (traditional)	6186	Download
The.Silence.Of.The.Lambs.1991.BluRay.1080p.x264-10bit.DTS-HD.HRA5.1-HDTim...	Chinese bilingual	6004	Download
The.Silence.Of.The.Lambs.[1991].srt	Greek	630	Download

2.6. ábra. Keresési találatokat tartalmazó táblázat

A keresőpanel megjeleníthető az alkalmazás alsó kezelőfelületén a *Search for subtitles online* gombra történő kattintással. Ekkor egy egyszerű felület tárul a felhasználó elé. Itt választhat, hogy filmhez, vagy sorozathoz készült feliratot keres. A kettőt azért volt szükséges szétválasztani, mert a két kérés címe mindkét esetben más. A keresési módok között a bal oldalon lévő *JToggleButton*-ra történő kattintással van lehetőség váltásra. Filmek esetében csak a címet, sorozatok esetében pedig a címet, illetve lehetőség szerint az évadot, epizódot kell megadnunk. A *Search* gombra történő kattintással elindíthatjuk a keresést, melyet egy függvény végez, ami a 2.11-es kódrészleten látható. A keresés nyelvét egy rendszerváltozóból kapja, a név-évad-epizód hármast pedig a megfelelő *JTextField*-ekből. Ezután a *.srt* kiterjesztésre való szűrés megy végbe.

Kódrészlet 2.11. Feliratok keresése sorozatokhoz valamint filmekhez

```
// if it is a serial
```



2.7. ábra. A letöltés helyét megjelenítő információs panel

```

if (isSerial.isSelected()) {
    subtitles = osClient.searchSubtitles(
        System.getProperty("user.language"),
        nameTextField.getText(),
        seasonTextField.getText(),
        episodeTextField.getText())
        .stream()
        .filter(sub -> sub.getFormat()
            .equals("srt"))
        .collect(Collectors.toList());
    // if it is a movie
} else {
    // ...
}

```

A művelethez természetesen internetkapcsolat szükséges, ennek hiányában az alkalmazás hibaiüzenetet jelenít meg a felhasználónak, az *An unexpected error has occurred. Please check your internet connection.* szöveggel. Mivel az *API* képes kezelni a hiányos, esetleg hibás keresési feltéteket, így legtöbbször elég a film vagy sorozat címének egy részét is megadni. Amennyiben a keresés sikeres egy táblázat tárul elénk, amely az összes megtalált felirat közül -az alkalmazás kompatibilitása miatt-, csak a *.srt* kiterjesztésűeket tartalmazza. Ez megtekinthető a 2.6-os ábrán. A táblázatban megjelenik a feliratfájl neve, nyelve, letöltéseinek száma, valamint egy *Download* gomb, mely a felirat letöltését indítja el.

2.7.2. Feliratok letöltése

Miután megtaláltuk a számunkra megfelelő feliratot, kattintsunk a letöltés gombra. Sikeres letöltés esetén egy *JOptionPane* jelenik meg a képernyőn, ami a 2.7-es ábrán látható módon tájékoztatja a felhasználót a letöltött fájl tartalmazó könyvtárról. Ez alapértelmezetten a *user.dir* rendszerváltozóból kerül kiolvasásra, azonban amennyiben bármilyen médiafájl már importálásra került a lejátszóba, a letöltés annak a fájlnek a tartalmazó könyvtárába kerül. A letöltést a *Wget.java* osztály végzi, amely egyetlen metódust tartalmaz. Ez paraméterként a mentendő fájl URL-jét, valamint egy útvonalat vár, ahova a kiírás történik. Törzsében egy *URLConnection*-ön keresztül egy *GET* kérést indít, amely *InputStream*-ként adja át az adatokat. Mivel az állomány tömörített, ezért *ArchiveInputStream* használatára is szükség van, amely a *stream*-et megfelelően kezeli. A metódus létrehoz egy *OutputStream*-et, amelybe átkonvertálja az *InputStream* tartalmát. Végül mindkét *stream*-et lezárja. E folyamat segítségével az alkalmazás képes a letöltött állományt kitömörítve elmenteni a felhasználó merevlemezére, amely így már azonnal importálható a lejátszóba.

2.8. Adatbázis műveletek

WORDS		
PK	words_id	VARCHAR(36)
	foreign_word	VARCHAR(50)
	meaning	VARCHAR(100)
	example	VARCHAR(300)
	filename	VARCHAR(100)

2.1. táblázat. Adatbázis struktúra

Ahhoz, hogy az alkalmazás további funkcióit, beleértve a pdf fájl, illetve tudásellenőrző teszt generálását implementálni tudjuk szükségünk van egy adatbázisra, amely az eddig elvégzett fordításokat tartalmazza kiegészítő adatokkal. Ehhez egy igazán könnyűsúlyú, egyszerűen telepíthető és használható adatbázist, a *H2* adatbázist vettem segítségül. Használatához csupán a megfelelő függőséget kellett az alkalmazás *.pom* állományába importálnom. Az adatbázist *liquibase* segítségével hoztam létre [3]. Az ehhez szükséges *changelog.xml* mellett létrehoztam egy *create-table.xml* fájlt, amely alapján a *liquibase*

megalkotja az adatbázistáblát, mely struktúrája a 2.1-es számú táblázatban látható. Ez a leírófájl úgynevezett *changeSet*-eket tartalmaz, melyek adatbázis műveleteket írnak le, és a *liquibase* sorban, egymás után hajt végre ezeket az adatbázison. Az első létrehozza a táblát, ha még az nem volt előtte létrehozva, a második pedig hozzáadja az elsődleges kulcsot, amely jelen esetben a *words_id*. Ez egy egyedi azonosító, melynek ismeretében pontosan meghatározható bármely rekord az adatbázisban. Formátuma *UUID*, ami egy 36 karakter hosszú egyedi karakterlánc. A tábla további elemei:

- **foreign_word**, a mondatelem, amelyre a felhasználó kattintott, legfeljebb 50 karakter hosszú
- **meaning**, a mondatelem jelentése(i), legfeljebb 100 karakter hosszú
- **example**, példamondat, amely a mondatelem kontextusából kerül ki, legfeljebb 300 karakter hosszú
- **filename**, feliratfájl neve, amelyben az aktuális mondatelem szerepelt, legfeljebb 100 karakter hosszú

Miután a tábla elkészült, lehetőség nyílik azon a szükséges adatbázis műveletek végrehajtására. Az objektumok a nyelvfordítás elvégeztével kerülnek be az adatbázisba. Amennyiben már megtalálhatóak ott, nem történik újbóli mentés. Mivel az alkalmazás működtetéséhez csupán néhány lekérdezésre, és beszúrára volt szükségem, egyszerű *jdbc* műveletek használata mellett döntöttem. Ezek a *WordService* osztályban kerültek implementálásra. Megvalósított műveletek:

- **create**, a szavak adatbázisba történő mentésére szolgál
- **getAllByFilename**, visszatér az összes szóval, melyek fájlnevei megegyeznek a paraméterben kapott fájlnevvvel
- **isAlreadySaved**, visszatérési értéke igaz vagy hamis, attól függően, hogy a paraméterben kapott szó megtalálható-e már az adatbázisban
- **deleteAll**, törli az összes rekordot az adatbázisból
- **getAll**, visszatér az összes adatbázisban található rekorddal

Egy metódus felépítése a 2.12-es kódrészleten látható, és a következőképpen működik: először létrehoz egy *Connection* kapcsolatot, amely az adatbázisfájl, és a hozzá tartozó azonosítók (felhasználónév, jelszó) ismeretében lehetséges. Majd ezen végrehajt egy műveletet, melyet *SQL* nyelven kell megadnunk a *jdbc*-nek. Amennyiben a művelet végrehajtása közben kivétel dobódott, az alkalmazás egy *JOptionPane* hibapanelet jelenít meg a képernyőn, a felhasználó tudtára adva, hogy váratlan hiba keletkezett a folyamat közben.

Kódrészlet 2.12. Implementált deleteAll metódus

```
public void deleteAll() {  
    try (Connection connection = DriverManager  
        .getConnection("jdbc:h2:file:~/basicv1c1j", "sa", "")) {  
        Statement stat = connection.createStatement();  
        stat.executeUpdate("DELETE FROM WORDS");  
    } catch (SQLException e) {  
        JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),  
            "An unexpected error has occurred",  
            "Error",  
            JOptionPane.ERROR_MESSAGE);  
    }  
}
```

Az imént említett függvények megvalósítása után egyszerűen lekérdezhetővé váltak a többi funkció által igényelt adatok. Így ezen komponensek implementálásával folytattam a fejlesztést.

2.9. Szótár generálása ismeretlen szavakból

Az alkalmazás képes fordítások megjelenítésére, valamint adatbázisba való elmentésükre, így a következő megvalósított funkció a szótárgenerálás volt. A szoftver az adatbázisból lekérdezés segítségével kinyeri a szükséges adatokat, majd ezekből egy *.pdf* kiterjesztésű szöszedetet állít elő. A 2.8-es ábrán látható módon a szótár tartalmazza az ismeretlen szót, azaz a mondatelemet, amire a felhasználó kattintott, a jelentését vagy jelentéseit, valamint a kontextust, a mondatot, amelyben a kattintott elem szerepelt.

Először szükség volt egy függőség importálására az alkalmazás *pom* fájljába. A választás az *itextpdf* API-jára esett, mely segítségével az említett funkcionalitás megvaló-

Word	Meaning	Example
conversation	beszélgetés, csevegést, társalgás, párbeszéd, témakör	And, uh, I cut the conversation short.
confused	zavaros, megzavarodott, összekeverni, összekeverte, téveszteni	And I... And I said, "Are you confused?"
calm	nyugodt, nyugalom, megnyugtatni, szélcsend, lecsillapítani, csendes	I was just trying to calm him down

2.8. ábra. Ismeretlen szavakból generált szótár

sítása maradéktalanul kivitelezhető. Ezután a *PDFGeneratorService* osztályt alkottam meg, amely a fájlgenerálásért felelős. Ebben található a *createDictionary* metódus, amely az adatbázis adataiból a konkrét szószeretetet kreálja. Egy új *Document* objektum létrehozása után a *PdfWriter* a dokumentum és egy *FileOutputStream* segítségével hozza létre a merevlemezen a tényleges fájlt. A fájl neve alapesetben az aktuális feliratfájl nevével egyezik meg. Ezután egy táblázat kerül a dokumentumba, mely 3 oszlopból áll. A táblázat fejlécét az *addTableHeader* függvény készíti el, így az tartalmazza a oszlopok feliratait, valamint egy világosszürke hátteret állít be neki. Az adatbázisból elkérjük az aktuális fájlnev alapján a rekordokat, majd ezen végig iterálunk, olyan módon, hogy minden rekordból egy *Word* objektumot készítünk, amit pedig új sorként adunk a táblázathoz. A bejárás végeztével hozzáadjuk a táblázatot a dokumentumhoz, majd bezárjuk azt, ezzel lezárva a fájlt, és elérhetővé téve azt a felhasználónak. A folyamat forráskódját a 2.13-as kódrészlet tartalmazza.

Kódrészlet 2.13. PDF fájl megalkotása

```
document.open();
PdfPTable table = new PdfPTable(3);
addTableHeader(table);

for (Word word : new WordService()
    .getAllByFilename(
        PlayerControlsPanel.actualSubtitleFile.getName())) {
    addRow(table,
        word.getForeignWord(),
        word.getMeaning(),
```

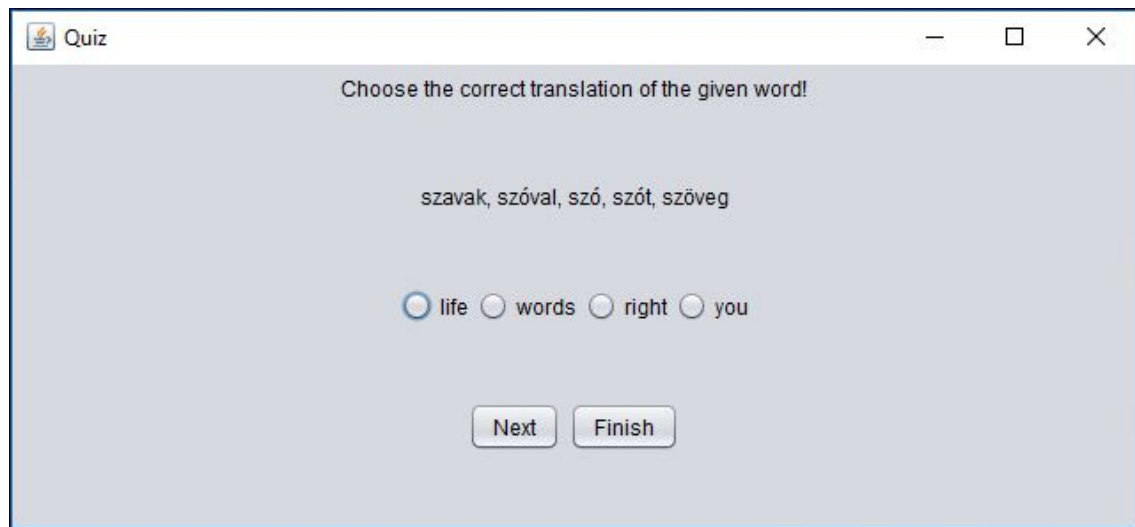


```
word . getExample ( ) ;  
}  
document . add ( table ) ;  
document . close ( ) ;
```

E funkció megvalósításával lehetőség nyílt a felhasználó számára, hogy a neki ismeretlen szavakat és ezek jelentését kényelmesen hordozható formában elmentse. Ezáltal egy későbbi alkalommal, esetleg tanulás közben e szavak memorizálására fokozott figyelmet fordíthat, anélkül, hogy a kigyűjtésükhöz bármiféle plusz eszköz szükségeltetett volna, hiszen mindez filmnézés, szórakozás közben történt.

2.10. Tudásellenőrző kvíz generálása ismeretlen szavakból

A funkció célja, hogy egy film végignézése után a felhasználó játékos módon ellenőrizni tudja, hogy az általa ismeretlen szavak mennyire rögzültek filmnézés alatt. Erre egy kvíz tökéletes megoldás, hiszen így gyorsan, könnyedén, alternatívák közül választva érhet el a felhasználó egy adott pontszámot, amely a filmnézés közben szerzett nyelvtudását tükrözi. Egy ilyen tudásellenőrző még jobban ösztönözheti kitöltőjét az ismeretlen szavak elsajátítására, hiszen a kvízben elért magas pontszám sikerélménnyel párosulhat.



2.9. ábra. Egy a kvízben előforduló kérdés

Implementálás során egy *Quiz.java* osztályt készítettem, amely a *JFrame* osztály lezármazottja és képes a kvíz előállítására az adatbázisból kapott adatokból, valamint en-

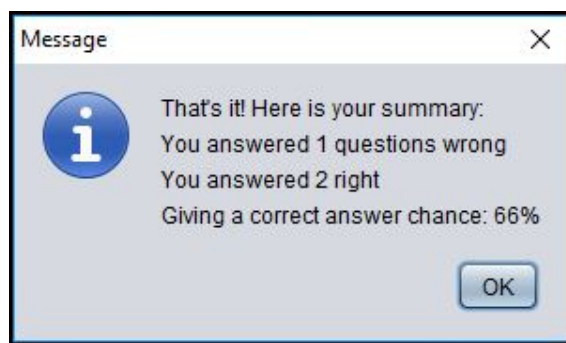
nek megjelenítésére, kezelőfelület biztosítására. Ezen belül található egy beágyazott osztály, a *RadioQuestion*, amely egy feleletválasztós kérdést reprezentál, őse a *JPanel* osztály. A működési elv egyszerű: a kvíz ablaka mindig egy kérdést jelenít meg, amit helyes válasz esetén cserél, rossz válasz esetén pedig felugró ablakkal jelzi, hogy a megadott válasz helytelen. Egy kérdés mindig tartalmazza az utasítást, alatta a fordítást vagy fordításokat, lentebb pedig a választási lehetőségeket. A panel alján a *Next*, valamint a *Finish* gombok találhatók meg. Az ablak megtekinthető a 2.9-es ábrán. A megoldás bejelölése után a *Next*-re kattintva új kérdés következik, a *Finish*-re kattintva pedig befejezhetjük a kvízt, amelyet egy pontösszesítő ablak jelez. A tudásfelmérő elindítható az alkalmazás felső menüsorából, a *Quiz/Take quiz...* menüpontot választva. Ahhoz, hogy a szoftver helyesen tudja megalkotni a kérdéseket alternatívákkal, legalább 6 megfelelő rekordot kell tartalmaznia az adatbázisnak. Amennyiben ez nem teljesül a felhasználót erről egy ablak tájékoztatja, egyébként megjelenik a kvíz ablaka, benne az első kérdéssel.

Kódrészlet 2.14. Kérdések előállítása

```
words = wordService.getAllByFilename(
    PlayerControlsPanel
    .actualSubtitleFile.getName());

questions = new RadioQuestion[words.size()];
for (int i = 0; i < questions.length; i++) {
    questions[i] = new RadioQuestion(
        generateRandomAnswers(i),
        words.get(i).getMeaning(),
        words.get(i).getForeignWord(),
        this);
}
```

A kérdések a *Quiz.java* osztály konstruktorában kerülnek előállításra, és egy tömbben tárolódnak. Először az adatbázisban tárolt szavakat fájlnev szerint lekérdezi, amiket egy listába ment el. Majd ugyanilyen mérettel létrejön egy tömb a kérdés objektumoknak. A függvény végig iterál a tömbön, és minden helyre egy kérdés objektumot hoz létre az alábbi paraméterekkel: Az első paraméter a *generateRandomAnswers* függvény, ami egy *String* tömbbel tér vissza, mely három véletlenszerű alternatívát és egy megoldást tartalmaz. A következő paraméterek az eredeti szó jelentése(i), az eredeti, idegen szó,



2.10. ábra. Pontösszesítő ablak

valamint maga az *Quiz* objektum. A kérdések előállításáért felelős forráskód sorok a 2.14-es kódrészletben találhatók. Az alkalmazás ettől kezdve váltogatja a kérdéseket, amikor pedig mindre válaszoltunk egy összesítő ablakot jelenít meg, amelyben láthatjuk hányszor válaszolunk helyesen, és hányszor helytelenül. Az ablak megtekinthető a 2.10-es ábrán.

A bemutatottak alapján megállapítható, hogy a kvíz generálás funkció segíthet rögzíteni, vagy akár elmélyíteni a filmezés közben szerzett információkat, mindezt szórakozással egybekötve.

2.11. Beállítások menü



2.11. ábra. Beállítások ablak

Miután a fő funkciók megvalósításával elkészültem, néhány apró kényelmi kiegészítő funkciót is implementáltam. Név szerint a felirat betűméretének állítását, illetve a felirat késleltetést. Ezek egy *Options* ablakban kaptak helyet, amely 2.11-es ábrán látható.

A lejátszó felső menüsávjában megtalálható *Options* menüpontra kattintva jeleníthe-

tő meg a beállításokat tartalmazó ablak. Felül, a *Font size* felirat mellett lévő beviteli mezőben a felirat méretét, az alatta lévő *Subtitle delay in milliseconds* mellett található mezőben pedig a felirat késleltetését állíthatjuk milliszekundumos pontossággal. A változtatások elvetése a *Cancel*, mentése az *OK* gomb megnyomásával történik. Mindkét esetben az alkalmazás egy-egy statikus változó értékét írja felül. A *SubtitleOverlay* osztály a feliratok megjelenítése során ezen változók értékeit használja.

Kódrészlet 2.15. Feliratok késleltetése

```
public void update(boolean force) {  
    // ...  
    // subtitle delay  
    long time = mediaPlayer.getTime() + subtitleDelay;  
    // ...  
}
```

A 2.15-ös kódrészletben látható módon késleltetés módosítása esetén a függvény törzsében, amely a feliratok képernyőn történő frissítését végzi, az aktuális időhöz hozzáadjuk a beállított késleltetést. Így a felirat időzítése a kívánt értékkel elcsúszik. Az eltolás mindkét irányban lehetséges, attól függően, hogy pozitív, vagy negatív értékkel rendelkezik a *subtitleDelay*. Csak egész számok adhatóak meg, alapértelmezett értéke nulla.

Kódrészlet 2.16. Feliratok mérete

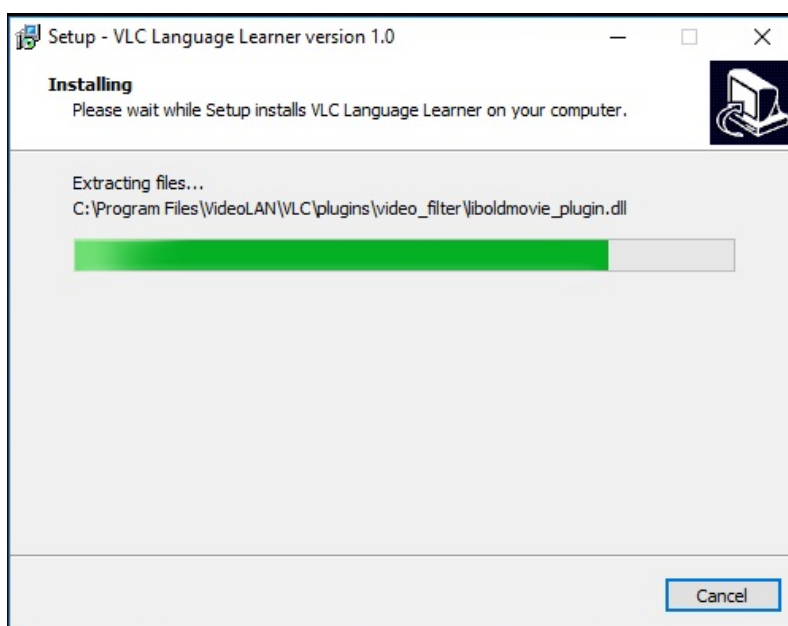
```
public void paint(Graphics g) {  
    // ...  
    g2.setFont(new Font("Serif", Font.PLAIN, fontSize));  
    // ...  
}
```

Betűméret módosítása esetén a feliratokat kirajzoló metódus törzsében történik a módosítás, hiszen a betűméretet értékét egy statikus változóból olvassa ki az alkalmazás a 2.16-os kódrészletben látható módon. A megjelenített szöveg így a megadott méretűre változik. Fontos megemlíteni, hogy csak egész, nullánál nagyobb betűméret adható meg, egyéb esetben az alkalmazás hibaüzenetet jelenít meg. Alapértelmezett értéke 20, amely ablakos megjelenítésnél elegendő lehet, azonban teljes képernyős módban legtöbbször kevés, így ajánlott ennek módosítása. A feliratok méretének állítására gyorsbillentyűk

is használhatók. Nagyításra a *Ctrl* +, kicsinyítésre pedig a *Ctrl* - billentyűkombinációk alkalmasak.

A két segédfunkcióval a feliratozás pontossága minden esetben biztosítható, hiszen, ha a feliratfájl esetlegesen téves időpillanatokat, vagy a médiafájl csúszást tartalmaz, akkor ezek orvosolhatóak a bemutatott funkcióval. Emellett pedig a felirat láthatósága is mindig biztosítható a betűméret módosításával.

2.12. Telepítőfájl létrehozása



2.12. ábra. Az alkalmazás telepítése

Az alkalmazás elkészültével egy telepítőfájlt készítettem el, amely segítségével egyszerűen, gyorsan telepíthetőek a szükséges állományok bármely számítógépre. Jelenleg az alkalmazás csak 64 bites Windows operációs rendszerű számítógépeket támogat, melyhez egy telepítőfájl készült.

Először a *.jar* fájl állítottam elő *maven* segítségével. Ehhez a *mvn package* parancsot kellett kiadnom a projekt könyvtárában. Ennek hatására a *maven* a projekt fájljaiból előállított egy hordozható *.jar* fájlt. A szoftver ezen formája azért nem elegendő számunkra, mert ahhoz, hogy az alkalmazás megfelelően működjön szükség van a *VLC* könyvtárakra is, amelyeket a *jar* nyilvánvalóan nem tartalmazza. Evégett hoztam létre egy telepítőt,

amely minden szükséges könyvtárat feltelepít a felhasználó számítógépére.

Ehhez egy speciális alkalmazást használtam, melynek neve *Inno Setup*. A szoftver képes a megadott fájlokból telepítővarázslót előállítani. Így bármely felhasználó néhány egyszerű kattintással telepítheti magának az alkalmazást. Ehhez először egy úgynevezett *Inno Setup Script*-et kellett elkészítenem, amely alapján a szoftver a telepítőfájl összeállítását végzi. Itt szinte bármit megadhatunk, amely a telepítés folyamatát befolyásolja. A fájl nevét, verzióját, alapértelmezett telepítési könyvtárát. A megfelelő telepítő létrehozásához elengedhetetlen információt tartalmaz a *Files* blokk. Itt kerülnek definiálásra azok a fájlok, amelyeket a telepítőszoftver összecsomagol. Ezen kívül ikon beállítására is van lehetőség, én a VLC eredeti ikonjának kissé módosított változatát használtam. A szkript egy része megtekinthető a 2.17-es kódrészleten.

Kódrészlet 2.17. Telepítőhöz szükséges fájlok

[Files]

Source: "D:\workspace\VideoLAN*";

DestDir: "{ app }";

Flags: ignoreversion recursesubdirs createallsubdirs

Source: "D:\workspace\VideoLAN\basic-vlcj-1.0.jar";

DestDir: "{ app }";

Flags: ignoreversion

Source: "D:\workspace\VideoLAN\icon.ico";

DestDir: "{ app }";

Flags: ignoreversion;

A szkript összeállítása után a *Run* gombra kattintva az *Inno Setup* elkészíti a telepítőfájlt. Miután végzett megjelenik a telepítővarázsló, amely már az alkalmazásunk elkészült telepítője. Kiválaszthatjuk a telepítés helyét, amely alapértelmezetten a *C:/Program Files/VideoLAN* könyvtár. Ezután a *Next* gombra kattintva egy *checkbox* segítségével kiválaszthatjuk, hogy szeretnénk-e az asztalon parancsikont létrehozni. Ismét kattintsunk a *Next* gombra. Egy összesítő ablak jelenik meg, amelyen láthatjuk a telepítőben megadott útvonalat, illetve azt, hogy kiválasztottuk-e a parancsikont generálást. Amennyiben valamelyikkel nem értünk egyet, vagy esetleg rosszul adtuk meg az útvonalat, a *Back* gombbal visszatérhetünk egészen a varázsló elejéhez. Ha mindennel egyetértünk, kattintsunk az *Install* gombra. A telepítő elkezd a szükséges fájlok megjelölt helyre történő másolását. A folyamatról készült képernyőkép a 2.12-es ábrán látható. Miután végzett az alkalmazás

indítható, az asztalon található parancsikon segítségével, amennyiben ezt kértük, vagy a telepítés helyén a *VLC Language Learner.jar* fájlra történő dupla kattintással. Az alkalmazás bármikor törölhető a Windows *Programok telepítése és törlése* menüpontban, illetve bármikor újratelepíthető az elkészült telepítőfájl segítségével. A telepítő működése jelenleg csak Windows rendszeren, azon belül is 64 bites verzió biztosított.

Befejezés

Szakdolgozatom célja egy a nyelvtanulást elősegítő alkalmazás fejlesztése volt, amely segítségével a felhasználók szórakozással egybekötve, filmnézés közben tehetnek szert értékes nyelvtudásra. Ez hatványozottan érvényes szavak memorizálása esetén, hiszen a filmekhez készült idegennyelvű feliratok kitűnő forrást biztosítanak bármely nyelv elsajátításához. A tanulás folyamata így könnyedebbé, szórakoztatóbbá válik, illetve a feliratok szóhasználatából eredendően, leginkább a hétköznapi szókincs fejlesztésére alkalmasabb. Mindezek mellett a szavak memorizálására fordított idő lecsökkenhet, valamint ennek folyamata hatékonyabbá válhat.

A megvalósított funkciók mind a fentebb említett nyelvtanulási módszerek színesítését szolgálják. A videolejátszó alapfunkciók kényelmes és felhasználóbarát kezelést tesznek lehetővé, egészen a lejátszás irányításától, a teljes képernyős funkción át, a média- és felirattípusok kiválasztásának lehetőségéig. Emellett a szoftver képes feliratok dinamikus megjelenítésére, valamint a felhasználó szavakra történő interakcióit, egérgattintásait is képes kezelni. A néző több mint 60 nyelv közül választhat, amelyeken a szoftver képes fordításokat végezni. Így közel 3600 féle nyelvpár közül válogathat a felhasználó, amely a legtöbb esetben kielégít bármilyen igényt. Ha nem áll rendelkezésünkre felirattípus, az online feliratkeresővel több ezer, interneten megtalálható fájl között böngészhetünk, mind ezt kényelmesen, az alkalmazás elhagyása nélkül. A beállított felirat és a film, valamint felhasználói képernyőméret és felirat betűméret összhangjának megteremtése érdekében valósítottam meg a beállítások panelt, amellyel mindez egyszerűen létrehozható. Tudásunk ellenőrzésére két alternatíva közül is választhatunk. Első lehetőségként a generált szószedet kínálkozik, mely segítségével a tanuló átismételheti az általa ismeretlennek vélt szavakat. Ez tartalmazza az idegen szót, jelentéseit, illetve a kontextust, melyben az adott mondatem szerepelt. Emellett lehetőségünk van egy tudásellenőrző kvíz kitöltésére is,

amely az idegen szavak memorizálásához nyújt segítséget játékos módon.

Jövőbeli tervek között szerepel, további telepítőfájlok elkészítése különböző operációs rendszerekhez. Hasznos lenne egy új funkció is, amely a kattintott szót felolvassa, kiejti, így még több információhoz juttatva a felhasználót. Valamint a szóhoz egy teljes értékű szócikket is megjelenítené, hasonlót mint az egynyelvű szótárakban szereplő. Ebből a tanuló megtudhatná a szó szófaját, definícióját, fonetikai átírását, szinonímáit, morfológiai tulajdonságai, valamint példamondatokat is láthatna egyes jelentéseire. Mindezek mellett a szoftver jövőbeli teljes körű támogatását tervezem.

Irodalomjegyzék

- [1] Apache maven. <https://maven.apache.org/guides/introduction/introduction-to-the-pom.html>, 2019. [Online; megtekintve 2019. 05. 10.].
- [2] Java™ platform, standard edition 8 api specification (2019). oracle documents. <https://docs.oracle.com/javase/8/docs/api/>, 2019. [Online; megtekintve 2019. 05. 10.].
- [3] Liquibase. <http://www.liquibase.org/documentation/databasechangelog.html>, 2019. [Online; megtekintve 2019. 05. 10.].
- [4] Vlcj. <https://github.com/caprica/vlcj>, 2019. [Online; megtekintve 2019. 05. 10.].

Nyilatkozat

Alulírott szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Tanszékén készítettem, diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

Szeged, 2019. május 14.

.....

aláírás

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani Tóth Zoltán Gábornak, a témavezetőmnek, az elmúlt két félév során a szakdolgozatom elkészítésében nyújtott segítségéért.