Dokumentáció

Melyik zeneszolgáltatást válasszam?

Szoftverarchitektúrák tárgy házi feladat

Készítette:

Fekete Norbert Zoltán
CO0DA1
feno26@gmail.com

Unicsovics Milán György
M9GNTV
u.milan@gmail.com

Tartalomjegyzék

1. Követelményspecifikáció

1.1. Feladatkiírás

A felhasználó eszközein (asztali PC vagy mobileszköz) tárolt zenéi alapján a rendszer megadja, hogy melyik zeneszolgáltatás katalógusában (Spotify, Deezer, Google Music,iTunes) található meg a legtöbb a tárolt zenék közül. Ehhez a különböző zeneszolgáltatások API-ját használja fel.

1.2. A fejlesztői csapat

A csapat tagjai:

Csapattag neve	**Neptun-kód**	**E-mail cím**
Fekete Norbert Zoltán	CO0DA1	feno26@gmail.com
Unicsovics Milán György	M9GNTV	u.milan@gmail.com

1. táblázat. A csapat tagjai

A csapatban dedikált szerepek kiosztását a csapat kis mérete miatt nem tartottuk fontosnak.

1.3. Részletes feladatleírás

A projekt során célunk egy olyan alkalmazás készítése, amely segít a felhasználónak eligazodni a manapság egyre inkább elterjedő internetes zenei szolgáltatások világában. Ezt a felhasználó meglévő zenei gyűjteményének letapogatásával, majd annak a felhő alapú szolgáltatók készleteivel történő összevetésével éri el.

Az alapvető keresési egység a zenei album lesz. Az elemzési folyamat végeztével a felhasználó egy statisztikát kap, melyből kiolvashatja, mely zeneszolgáltatás gyűjteményével a legnagyobb az átfedés - vagyis mely szolgáltatónál találhat a legkönnyebben a saját stílusának megfelelő zenéket.

Emellett egyszerű, kulcsszavas keresésre is lesz lehetőség (meglévő zenefájlok nélküli kereséshez), amivel kényelmesen lehet majd egyszerre több szolgáltató készletét is lekérdezni.

A program első megközelítésben a *Deezer*, a *Spotify*, az *iTunes* és a *Last.Fm* szolgáltatásait fogja támogatni, de célunk egy általános architektúra kialakítása, mely később könnyedén bővíthető további szolgáltatásokkal (pl.: *Google Play Music*).

1.4. Technikai paraméterek

A definiált alkalmazást Python platformra készítjük el annak érdekében, hogy több operációs rendszeren (Windows, Linux, Mac OS) is lehessen használni. Szükség lesz ezen kívül néhány Python-os könyvtárhoz, ezeket a Python csomagkezelőjével (*pip*) egyszerűen lehet majd telepíteni.

A program felhasználói felületét Python GTK+ 3 segítségével fogjuk elkészíteni, a GUI-t magát deklaratív módon Glade segítségével állítjuk elő.

Külső függőségeink közé tartoznak a zeneszolgáltatások API-jai is:

- Spotify
- Deezer
- iTunes
- Last.fm
- (Google Music)

A fent felsorolt külső függőségek sebességének függvényében, lehet, hogy szükség lesz valamilyen szerver oldali komponens fejlesztésére is, amely egy fajta cache-ként szolgálva gyorsíthatja a program működését. A szerver elérhetősége viszont a program funkcionalitását nem befolyásolja.

1.5. Szótár

Zeneszolgáltatás Kereskedelmi streamelő alkalmazás, melyben a fehasználó zenéket hallgathat illetve vásárolhat.

Zeneállomány A zeneszolgáltatás vagy a felhasználó által birtokolt zenealbumok összessége.

Zenei katalógus A zeneszolgáltatás kereshető zeneállománya.

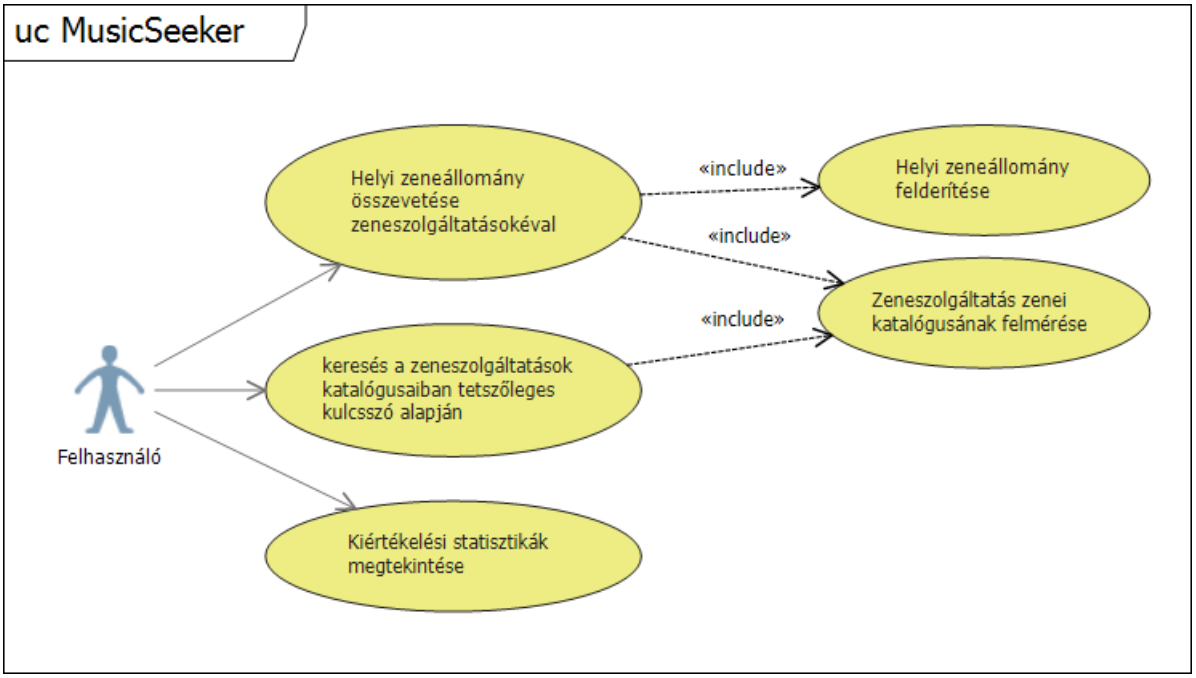
Kiértékelési statisztika A felhasználó által megtekinthető grafikus szemléltető eszköz, mely arra szolgál, hogy az egyes zeneszolgáltatásoknál, mely zenealbumok érhetőek el.

Zeneállomány felderítése A felhasználó által megadott helyen az elérhető zenealbumok és azok adatainak összegyűjtése.

Keresés zenékre Minden támogatott zeneszolgáltató átvizsgálása, hogy egy bizonyos zenealbum elérhető-e az adott platformon.

1.6. Essential use-case-ek

1.6.1. Use-case diagram



1. ábra. A MusicSeeker use case diagramja

2. Rendszerterv

2.1. A rendszer célja, funkciói és környezete

2.1.1. Feladatkiírás

A felhasználó eszközein (asztali PC vagy mobileszköz) tárolt zenéi alapján a rendszer megadja, hogy melyik zeneszolgáltatás katalógusában (Spotify, Deezer, Google Music, iTunes) található meg a legtöbb a tárolt zenék közül. Ehhez a különböző zeneszolgáltatások API-ját használja fel. A feladat részletes specifikációja a követelményspecifikáció dokumentumban olvasható.

2.1.2. A rendszer által biztosítandó tipikus funkciók

Vázlatosan az alábbi funkciók biztosítását várjuk el a rendszertől. (A funkciók részletes definíciója szintén a követelményspecifikáció dokumentumban olvasható.)

- helyi zeneállomány összevetése zeneszolgáltatásokéval, melynek lépései:
  - helyi zeneállomány felderítése
  - zeneszolgáltatások katalógusainak felmérése
  - kiértékelési statisztika megjelenítése, szolgáltatóajánlás
- keresés a zeneszolgáltatások katalógusaiban tetszőleges kulcsszó alapján

2.1.3. A program környezete

A szoftvert vastagkliens alkalmazásként készítettük el. Annak érdekében, hogy több operációs rendszeren (Windows, Linux, Mac OS X stb.) is futtatható legyen, platformfüggetlen megoldásokat választottunk. Az alkalmazást Python platformra készítettük el. Szükséges ezen kívül néhány Python könyvtár telepítése is, ami a Python csomagkezelőjével (`pip`) egyszerűen megtehető. A felhasznált csomagok a következőek:

- Stagger
- Requests

A program felhasználói felületét Python GTK+ 3 segítségével készítettük el, így a GTK grafikus könyvtár telepítése is szükséges.

Külső függőségeink közé tartoznak a zeneszolgáltatások is:

- Spotify
- Deezer
- iTunes
- Last.fm

Ezen szolgáltatások eléréséhez természetesen internetkapcsolat szükséges.

2.2. Tervezés és implementáció

Az alkalmazást a skálázhatóság és továbbfejleszthetőség miatt egy többrétegű alkalmazásként készítettük el. Az egyes rétegek jól definiáltan különválnak egymástól.

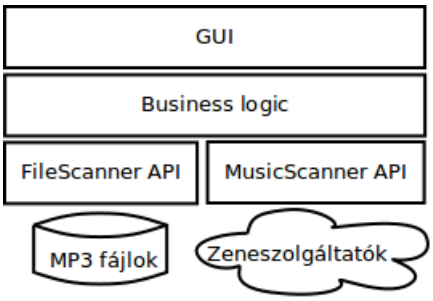
Az általunk elkészített programot MusicSeeker-nek neveztük el, mely névvel az alkalmazás legfőbb funkciójára akartunk utalni, a zenék keresésére.

A fejezetben áttekintést adunk a program architektúrájáról, bemutatjuk az egyes komponensek feladatait és felelősségeit, illetve ismertetjük a grafikus felhasználói felület felépítését.

2.2.1. Architektúra

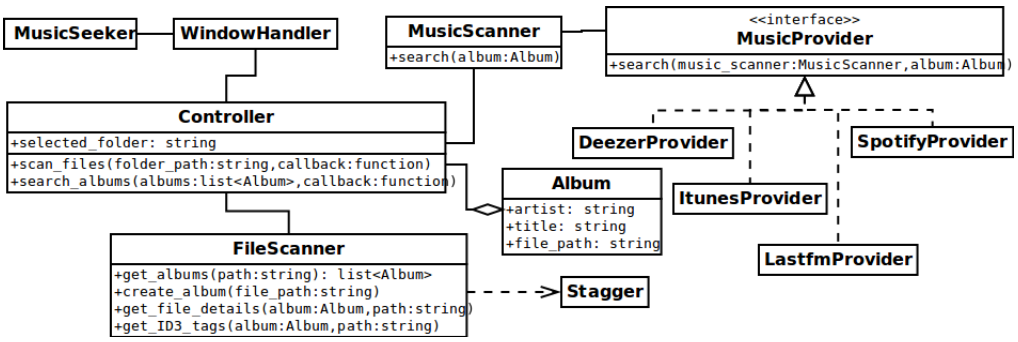
A MusicSeeker architektúrája 4 rétegre bontható fel:

- adatréteg
- adatelérési réteg
- üzleti logikai réteg
- felhasználói felület



2. ábra. A MusicSeeker architektúrája

Az egyes rétegek célját és tervezési szempontjait a következő alfejezetekben fogjuk ismertetni. A különböző rétegek tervezésekor használt mintákat, pedig az alkalmazás alábbi, magasszintű osztálydiagrammján szemléltetjük.



3. ábra. A MusicSeeker magasszintű osztálydiagrammja

Adatréteg

Célja: Adatok szolgáltatása a felsőbb rétegek számára, zenékről és zeneszolgáltatások könyvtárairól.

A hagyományos adatbázis réteget itt egy főleg külső komponensekből álló réteg alkotja, hiszen a program működéséhez szükséges adatok, minden esetben felhasználói beavatkozásra, on-the-fly lesznek összegyűjtve.

A gépen található zenealbumok adatai, a lokális gépről tallózott mappából, az MP3 fájlokat kigyűjtve és elemezve lesznek meghatározva.

A felhőben található zeneszolgáltatások könyvtárának adatai, azok API-jainak felhasználásával lesznek összegyűjtve.

Adathozzáférési réteg (Data Access Layer)

Célja: Az adathozzáférési réteg két részből áll, egyrészt a lokális gépen található zenék metaadataihoz és a felhőben található zeneszolgáltatások könyvtárainak adataihoz nyújtanak hozzáférést.

FileScanner API Ezen a komponensen keresztül lehet a lokális gépről elérhető zenék metaadatait lekérdezni, azok ID3 címkéinek kiolvasásával, majd a megszerzett adatokból az alkalmazás központi szerepét játszó entitás példányait létrehozni.

Megvalósítás: A FileScanner `get_albums` függvényét meghívva megkeresi egy adott mappában az összes MP3 fájlt, majd azok ID3 címkét

kiolvassa és Album példányokból álló listát ad vissza. Működését tekintve a FacadeWrapper mintát valósítja meg, ugyanis egy külső Stagger nevű könyvtár kezelését fedi el az ID3 címkék kiolvasásakor. Az Album példányok elkészítését Builder mintával készítettük el, hogy a továbbfejlesztéskor ezek az entitások könnyen kiegészíthetőek legyenek új attribútumokkal, amelyeket változatos módokon lehet beszerezni.

MusicScanner API A MusicScanner arra szolgál, hogy a felhőben található zeneszolgáltatók könyvtáraiból adatokat tudjunk szerezni. Ezt elsősorban arra fogjuk használni, hogy megvizsgáljuk, hogy a lokális gépen elérhető zenealbumok megtalálhatóak-e az adott zeneszolgáltatásban.

Megvalósítás: A MusicScanner működése a Visitor minta alapján lett megvalósítva. A minta terminológiáját tekintve a különböző zeneszolgáltatók adatszolgáltató osztályai lesznek a visitorok, amelyek adatokat tudnak szerezni különböző albumokról. Így a rendszer tetszőlegesen kibővíthető új zeneszolgáltatásokkal, de a MusicProvideren keresztül új funkciók is fejleszthetők a rendszerbe.

A különböző zeneszolgáltatások közül végül 4-et választottunk ki, és építettünk be az alkalmazásba:

- Deezer
- iTunes
- Last.fm
- Spotify

Ezekre azért esett a választás, mert nagyon egyszerű REST API-hoz hasonló, webes elérési felületük van, melyet nagyon egyszerűen tudtunk implementálni. A webszolgáltatások meghívásához a requests könyvtárat használtuk fel. A Last.fm kivételével minden szolgáltatás regisztráció nélkül is használható, a használatához kötődő kvótákat megvizsgálva úgy döntöttünk, hogy nem szükséges a követelményspecifikációban említett cache szolgáltatás implementálása.

Üzleti logikai réteg (Business Logic Layer)

Célja: Az üzleti logikai réteg felel a különböző fő használati esetek végrehajtásáért, ezek pedig a helyi állomány vizsgálata, és információk gyűjtése felhőszolgáltatóktól, és statisztika készítése az eredmények alapján. Ezen tevékenységeket az alatta levő rétegeket használva éri el.

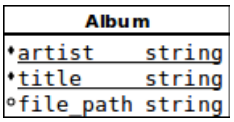
Megvalósítás: A zenei albumokkal kapcsolatos adatok gyűjtését a Controller osztály valósítja meg, mind a helyi zeneállomány, mind a zeneszolgáltatásokban található információk alapján. Statisztikák készítéséért a Statistics osztály felelős.

Grafikus felhasználói felület

Célja: A grafikus felület felelős a felhasználóval való interakcióért. A felhasználó végrehajthat műveleteket, és azok eredményéről tájékoztatást kap.

Megvalósítás: Maga a grafikus felület gyakorlatilag teljes egészében deklaratív módon Glade segítségével lett megvalósítva. A felülethez kapcsolódó eseménykezelőket a WindowHandler osztály valósítja meg, valamint az üzleti logikai réteg által létrehozott eredményeket jeleníti meg a felhasználó számára. A hosszan futó feladatokat a Future minta segítségével háttérszálon végeztük el, és a felhasználót folyamatosan tájékoztattuk a végrehajtani kívánt folyamat állapotáról.

2.2.2. Adatterv

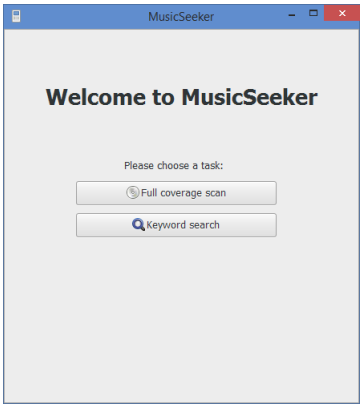


4. ábra. Album entitás

Album entitás Az alkalmazás működésében központi szerepet tölt be, és gyakorlatilag egyetlen entitásként létezik az Album, mely egy zenei albumot jelöl. A zenei albumot, annak előadója és címe azonosítja, egyéb attribútuma még az útvonal is, melyen keresztül elérhető a fájlrendszerben.

2.2.3. GUI-terv

Amikor a felhasználó elindítja az alkalmazást, egy üdvözlőképernyő fogadja (5. ábra). A specifikációban kijelölt két fő usecase-nek megfelelően két fő menüpont fogadja: egy, amellyel egyszerű, kulcsszó alapú kereséseket végezhet a támogatott felhőszolgáltatók adatbázisán, és egy másik, amellyel komplex letapogatást és összehasonlítást végezhet, a saját, meglévő zenei gyűjteménye alapján.

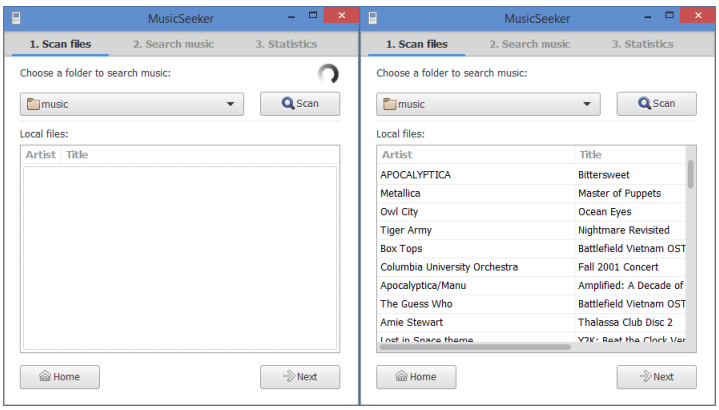


5. ábra. A MusicSeeker kezdőképernyő

A felhasználó a két gomb segítségével juthat a konkrét funkciókhoz tartozó képernyőkre, ahonnan az ottani Home gomb segítségével bármikor visszatalálhat.

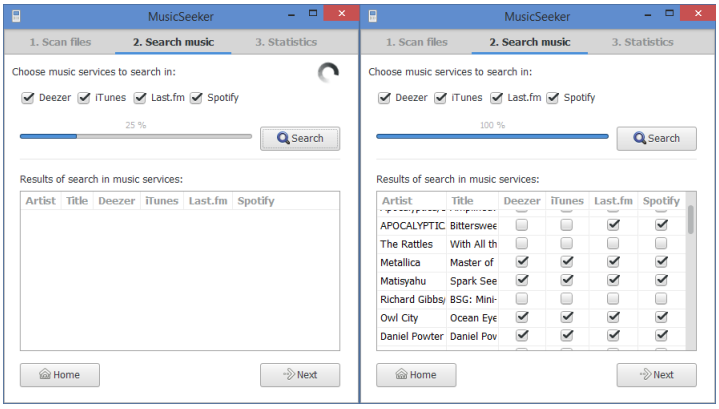
A Full coverage scan egy összetettebb funkció, ezért egy varázsló jellegű felületet kapott (6. ábra).

A legelső lépés, hogy a felhasználó szkennelje a lokális zenei gyűjteményét. Ehhez csupán a megfelelő mappát kell kiválasztania, majd a Scan gombra kattintania. Ekkor az alkalmazás rekurzívan összegyűjti a mappa alatt található valamennyi mp3 fájl albuminformációját, amit a művelet végeztével egy egyszerű táblázatban meg is jelenít.



6. ábra. Lokális gyűjtemény letapogatása

A táblázat megtekintése után a Next gomb továbbvezeti a felhasználót a következő lépéshez, amely már a felhőszolgáltatók tényleges lekérdezése (7. ábra).



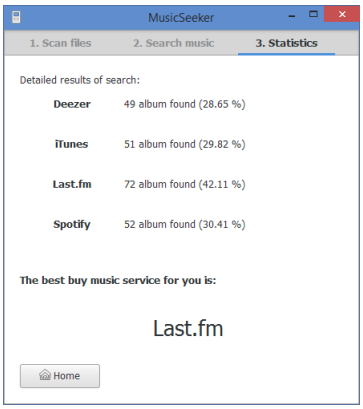
7. ábra. Zeneszolgáltatók letapogatása

Ez a lépés már több időt vesz igénybe, mivel a szolgáltatók általában korlátozzák a másodpercenként küldhető kérések számát. Ezért ezen a felületen elhelyeztünk egy progressbart, mely a lekérdezések állapotát jelzi.

A keresés előtt a felhasználó egyszerű checkboxokkal választhatja ki, a támogatott szolgáltatók közül melyek kínálata érdekli.

A letapogatás befejeztével táblázatos formában jelenítjük meg az eredményeket: az egyes lokálisan felderített albumok mellett egy-egy checkboxszal jelöljük, mely szolgáltatóknál érhetők el.

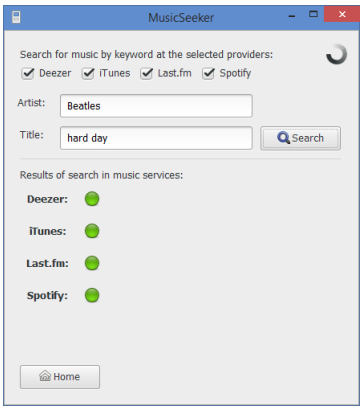
A Next gomb itt már a végső nézethez, a statisztikákhoz vezet (8. ábra). Ezen az oldalon a keresés aggregált eredménye látszik, vagyis hogy melyik szolgáltatónál mennyi található meg az albumainkból. Az adatok alapján tippet adunk a felhasználónak, melyik zeneszolgáltatót érdemes választania, ha szeretne előfizetni - azaz melyik szolgáltatóval legnagyobb az átfedés.



8. ábra. Statisztika nézet a letapogatás után

Az alkalmazás másik fő funkciója, a kulcsszavas keresés jóval egyszerűbb,

mindössze egy oldalas (9. ábra). A szolgáltatókat itt is checkboxokkal választhatjuk ki. A kívánt szerző illetve album nevét beírva, majd a Search gombra kattintva lefut a lekérdezés. A szolgáltatók nevei mellett zöld illetve piros kör jelzi, hogy a kulcsszavarka érkezett-e találat az adott szolgáltató rendszeréből.



9. ábra. Kulcsszavas keresés a zeneszolgáltatóknál

2.3. Telepítési leírás

A Python program interpretált futtatásához a Python futtatókörnyezet 3.x verziója szükséges. A Python könyvtárak telepítéséhez a `pip` nevű csomagkezelő program telepítése ajánlott.

A szükséges könyvtárak a programhoz csatolt `requirements.txt` nevű fájlban találhatóak meg, amelyeket a `pip` nevű program fel tud dolgozni, és telepíti a megfelelő könyvtárakat.

2.3.1. Linux

1. Listing. Telepítés Linux környezetben

```
sudo apt-get install python3-all python3-all-dev
   python3-gi python-glade2
sudo pip -r requirements.txt
```

2.3.2. Windows

CPython környezet megfelelőverziójának telepítése: `http://python.org/download`

Ha véletlen probléma adódik a Stagger nevű könyvtárral, azt manuálisan kell telepíteni az alábbi címről: `https://pypi.python.org/pypi/stagger/0.4.2`

A grafikus felület működéséhez a Python mellett szükséges a GTK grafikus könyvtár is. Windows esetén mi a következő telepítőkészletet használtuk: `http://sourceforge.net/projects/pygobjectwin32/` Ez nem csak magát a GTK-t tartalmazza, hanem különböző - többek között Pythonhoz szükséges - fejlesztőeszközöket is, ezek telepítése a főprogram futtatásához opcionális.

2.4. A program készítése során felhasznált eszközök

Glade Segítség GTK-s felületek egyszerűbb, XML alapú előállításához. Grafikus designert is tartalmaz. Windows-on a fent említett összevont telepítőkészletnek (PyGObjectWin32) is része.

GitHub Verziókezelés.

Skype Kommunikáció, kapcsolattartás.

2.4.1. Linux

Sublime Text3 Könnyűsúlyú kiterjeszthető szövegkezelő, mely IDE-ként is megállja a helyét.

EditorConfig Egységes IDE beállítások kezelésére alkalmas konfigurációs formátum.

2.4.2. Windows

Python Tools for Visual Studio Integrált fejlesztőkörnyezet Pythonhoz

2.5. Továbbfejlesztési lehetőségek

A program feladatának egyszerűségéből adódóan leginkább felületi fejlesztéseket lehetne még elvégezni:

- Felületi elemek dinamikus átmérezetése.
- Ergonómikusabb táblázatok, illetve checkbox-ok, nagyobb mennyiségű támogatott zeneszolgáltatás esetén.
- Statisztikai nézet továbbfejlesztése, esetleg más jellegű aggregált adatok kijelzése.

- Minél több zeneszolgáltatás integrációja.

2.6. Összefoglalás

Munkánk során megterveztük, implementáltuk és dokumentáltuk a MusicSeeker nevű alkalmazást, mely segít a felhasználónak saját ízlésének megfelelő zeneszolgáltatót választani, meglévő zenei gyűjteménye alapján.

Az alkalmazás 3 rétegű architektúrát használ: adatelérési réteg, logikai réteg és grafikus felhasználói felület. A helyi adatokat a felhasználó meglévő mp3 fájljaiból olvassa ki, a stagger könyvtár segítségével. A zeneszolgáltatók katalógusait azok REST interfészén keresztül éri el. A grafikus felület a GTK könyvtárat használja.

Bízunk benne, hogy munkánk eredményeként egy hasznos kis programot kaptunk, mely segítséget nyújthat a zeneszolgáltatást kereső embereknek, hogy azok a számukra lehető legmegfelelőbb tartalomhoz juthassanak hozzá a pénzükért.

2.7. Hivatkozások

- CPython: `http://python.org/`
- PyGObjectWin32: `http://sourceforge.net/projects/pygobjectwin32/`
- Glade: `https://glade.gnome.org/`
- Stagger: `https://pypi.python.org/pypi/stagger/0.4.2`
- Requests: `http://docs.python-requests.org/`
- Sublime Text3: `http://www.sublimetext.com/3`
- EditorConfig: `http://editorconfig.org/`
- PTVS: `http://pytools.codeplex.com/`
- GitHub: `https://github.com/`
- Skype: `http://www.skype.com/`