Проект за музикален mp3 плейър

АВТОРИ: РАФАЕЛ ВЕНЦИСЛАВОВ ПЕТРОВ

Съдържание

Встъпление

1. Цели на проекта
   1.1. Прослушване на дадена песен.
   1.2. Манипулации с песни – пауза, стоп, следваща, предишна.
   1.3. Манипулации с плейлисти- създаване и слушане на плейлисти
   1.4. Максимално опростен и удобен интерфейс.
   1.5. Поддръжка на mp3 - файлове.

2. Потребителски изисквания
   - може потребителят да добавя песен за прослушване.
   - може потребителят да добавя много песни за прослушване.
   - може потребителят да премахва песен за прослушване.
   - може потребителят да премахва всички песни за прослушване.
   - може потребителят да създава плейлист.
   - може потребителят да вижда всички песни в плейлист.
   - може потребителят да зарежда плейлист.
   - може потребителят да вижда кой е заредения плейлист
   - може потребителят да вижда добавените песни за прослушване.
   - може потребителят да паузира и пуска песен.
   - може потребителят да пуска следваща песен.
   - може потребителят да пуска предишна песен.
   - може потребителят да вижда слайдер, който репрезентира песента, когато се прослушва.
   - може потребителят контролира силата на звука за приложението, чрез слайдер.
   - може потребителят да види цялото заглавие на песента

3. Аудитория
   Приложението е насочено към хората, които търсят опростен дизайн за мп3 плейър, който да им предоставя всичко нужно – прослушване, създаване на плейлисти по негов вкус и избор

Изложение

4. Разработка
   4.1. Архитектура на приложението:

   <u>Подход на двустепенната архитектура -</u> Двуслойната архитектура, често наричана клиент-сървър архитектура, е системна архитектура, в която презентационният слой и данните са физически разделени. Това води до два слоя:
   -Клиент (презентационен слой): Това е мястото, където потребителят взаимодейства със системата. Може да е десктоп приложение, уеб браузър или друг потребителски интерфейс.
   Този слой изпраща заявки до сървъра и след това показва получените от сървъра резултати на потребителя.

   -Сървър (слой данни): Това може да бъде сървър на база данни, сървър на приложения или и двете. Той обработва заявките, получени от клиента, взаимодейства със съхранението на данни (като база данни), ако е необходимо, и изпраща резултатите обратно на клиента.

4.2. Разделение на проекта

- Play.py – Файл съдържащ основните класове, които визуализират приложението и го стартират.
- Music_play_db_setup.py – Скриптовия файл, който се използва за генериране на базата данни, ако тя не съществува. Файлът съдържащ основните функции, които отговарят за комуникацията с базата данни и предаването на данни
- Папка audio– директория на приложението, от където взима музиката.
- Папка buttons – Директория с основните изображения, с които работи приложението.

4.3. *Технологии:*

Code First подход – подход в програмирането, основан на написването първо на подходящи модели и по техен образ се създава базата данни, въпреки че не е стриктно придържане към начина, по който Code First обикновено се практикува с помощта на модерни Обектно-релационно картографиране(Object relational mapping) рамки като SQLAlchemy за Python.

Repository Pattern: Този модел изолира логиката за достъп до данни в отделен "repository" клас, който капсулира всички заявки към базата данни. Въпреки че има функции, които капсулират достъпа до базата данни, кодът не използва стриктен "Repository Pattern", където всеки ресурс (напр. песен или плейлист) има свой отделен "repository" клас.

Dependency Injection: В кода няма ясни признаци за използване на "Dependency Injection". "Dependency Injection" е методология, при която зависимостите на даден обект (напр. база данни или друг сървиз) се подават към този обект, вместо обектът сам да си създава или намира тези зависимости.

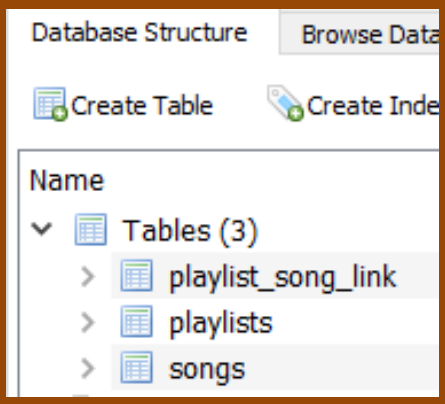
4.4. Библиотеки и модули:

- Tkinter - обвързване на Python към Tk GUI инструментариума. Това е стандартният интерфейс на Python към Tk GUI инструментариума и е де факто стандартният GUI на Python.
- Time - време: Този модул предоставя различни функции, свързани с времето. В музикален плейър това може да се използва за проследяване на дължини на песни, закъснения или работа с други задачи, базирани на времето.
- Threading- Модулът за нишки на Python се използва за създаване на нишки. В контекста на музикален плейър нишката може да позволи на приложението да извършва множество операции едновременно, като възпроизвеждане на музика, като същевременно потребителя може да работи с интерфейса
- Pygame – библиотека към програмния език Python. С нея потребителите могат лесно и същевременно бързо да създават игри.
- Time – модул в Python, който добавя различни функции свързани с време и дата
- Os – Модул в Python, който позволява на приложението да взаимодейства с операционната система.

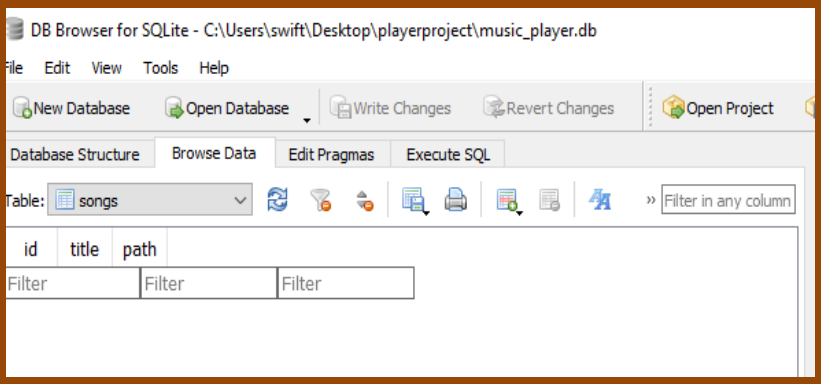
- Mutagen – Модул в Python, който се използва за извличане на метаданни от аудио файлове.
- SQLite – релационна база данни с отворен код поддържаща стандарта SQL. Реализирана е като библиотека към приложенията, а не като самостоятелно работеща програма.

4.5. *Data Base Management System* – DB Browser for SQLite;

Разяснения: *Базата данни се състои от три таблици:*



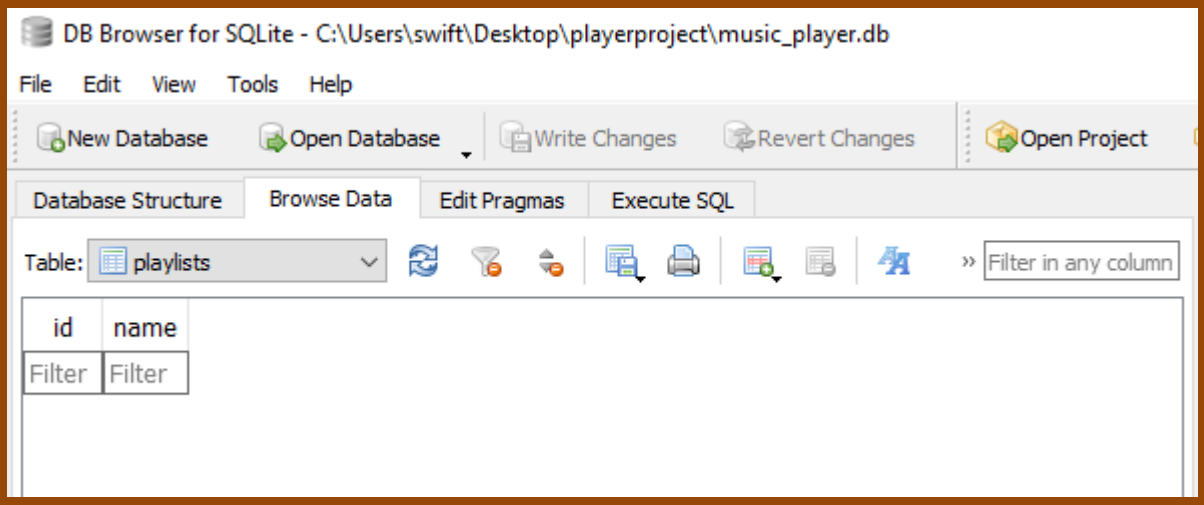
songs



songs

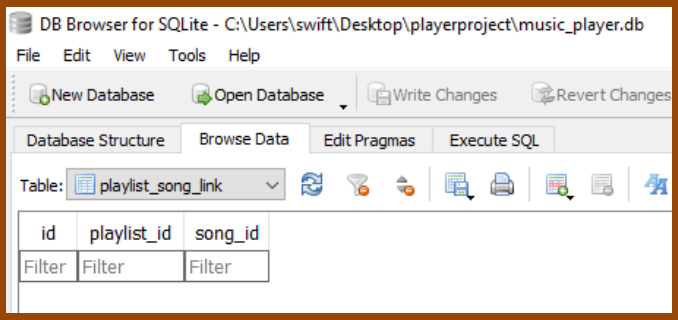
- **id: уникален идентификатор (ключ) за всяка песен. Също е основен ключ.**
- **title: текстово поле, съхраняващо заглавието на песента. По изискване, то не може да бъде празно.**
- **path: текстово поле, съхраняващо пътя до файла на песента или някакъв друг референтен адрес. По изискване, то не може да бъде празно.**

Playlists



- **id: уникален идентификатор (ключ) за всяка плейлиста. Този идентификатор е основен ключ (PRIMARY KEY), което означава, че всеки запис в тази таблица трябва да има уникално стойност за id.**
- **name: текстово поле, съхраняващо името на плейлистата. По изискване (NOT NULL), то не може да бъде празно.**

playlist_song_link:



- **Тази таблица е така наречената "linking table" или таблица за свързване. Тя свързва плейлистите с песните, позволявайки на една плейлиста да съдържа много песни, а една песен да бъде част от много плейлисти (отношение "много към много" или "many-to-many").**
- **id: уникален идентификатор (ключ) за всеки запис в таблицата.**
- **playlist_id: идентификатора на плейлистата. Това поле се свързва (FOREIGN KEY) с id полето на таблицата playlists.**
- **song_id: идентификатора на песента. Това поле се свързва (FOREIGN KEY) с id полето на таблицата songs**

Така когато искате да добавите песен към плейлиста, просто добавяте нов запис в playlist_song_link с идентификаторите на песента и плейлистата, които искате да свържете.

4.6. Модули, Класове, функции и методи:

- *music_player_db_setup.py:*
- -Този модул предоставя начин за управление на плейлисти и песни за музикален плейър. Той използва SQLite за съхранение на данни.
- База данни:
- DATABASE_NAME = 'music_player.db': Указва името на базата данни, където ще се съхраняват плейлисти и песни.
- Връзка:
- create_connection():
- Цел: Създава връзка с база данни към базата данни SQLite.
- Връща: Обект за връзка или Няма.
- Изпълнение:
- execute_query(conn, query, values=None):
- Цел: Изпълнява една заявка.
- Параметри:
- conn: Обект за връзка с база данни.
- заявка: низ на SQL заявка.
- стойности: Кортеж от стойности за вмъкване/актуализиране (по подразбиране е Няма).
- Връща: SQLite курсорен обект.
- Създаване на таблица:
- create_tables():
- Цел: Инициализира всички таблици в базата данни, включително плейлисти, песни и playlist_song_link.
- Добавяне на нови данни:
- add_playlist(плейлист_име):
- Цел: Добавя нов плейлист към базата данни.
- Параметри:
- playlist_name: Име на новия плейлист.
- add_song(song_title, song_path):
- Цел: Добавя нова песен към базата данни.
- Параметри:
- song_title: Заглавие на песента.
- song_path: Пътят на песента.
- Връща: ID на добавената песен.
- add_song_to_playlist(плейлист_име, песен_път):
- Цел: Добавя песен към определен плейлист. Това също обработва добавянето на песен към таблицата с песни, ако тя не съществува.
- Параметри:
- playlist_name: Име на плейлиста.
- song_path: Пътят на песента.
- link_song_to_playlist(playlist_id, song_id):
- Цел: Създава връзка между песен и плейлист в таблицата playlist_song_link.
- Параметри:
- playlist_id: ID на плейлиста.
- song_id: ID на песента

.

- Извличане на данни:
- get_all_songs():
- Цел: Извлича всички песни от базата данни.
- Връща: Списък със заглавия на песни и кортежи на пътя.
- get_all_playlists():
- Цел: Извлича всички плейлисти от базата данни.
- Връща: Списък с идентификатори на плейлисти и кортежи с имена.
- get_songs_from_playlist(playlist_id):
- Цел: Извлича всички песни от определен плейлист.
- Параметри:
- playlist_id: ID на плейлиста.
- Връща: Списък със заглавия на песни и кортежи на пътя.
- Валидиране:
- is_song_in_db(song_title, filepath):
- Цел: Проверява дали песен с посоченото заглавие и път към файла съществува в таблицата с песните.
- Параметри:
- song_title: Заглавие на песента.
- filepath: Пътят на песента.
- Връща: True, ако съществува, False в противен случай.
- is_song_in_playlist(song_title, playlist_id):
- Цел: Проверява дали дадена песен вече е в определен плейлист.
- Параметри:
- song_title: Заглавие на песента.
- playlist_id: ID на плейлиста.
- Връща: True, ако съществува, False в противен случай.
- Помощни функции:
- add_song_to_playlist_if_not_exists(song_title, song_path, playlist_name):
- Цел: Добавя песен към таблицата с песни (ако не съществува) и след това я добавя към даден плейлист.
- Параметри:
- song_title: Заглавие на песента.
- song_path: Пътят на песента.
- playlist_name: Име на плейлиста.
- get_song_path (име на плейлист, заглавие на песен):
- Цел: Извлича пътя на песен въз основа на заглавието и името на плейлиста.
- Параметри:
- playlist_name: Име на плейлиста.
- song_title: Заглавие на песента.
- Връща се: Пътят на песента.
- get_songs_in_playlist(име на плейлист):
- Цел: Извлича списък с песни в определен плейлист.
- Параметри:
- playlist_name: Име на плейлиста.
- Връща: списък със заглавия на песни.
- get_playlist_name_by_id(playlist_id):

- Цел: Извлича името на плейлист, използвайки неговия идентификатор.

- Параметри:
- playlist_id: ID на плейлиста.
- Връща: Име на плейлиста.
- Инициализация:
- create_tables(): Този ред инициализира таблиците, когато модулът се импортира.

Приложението е структурирано с помощта на GUI рамката на tkinter и е организирано с помощта на множество рамки за групиране на свързани компоненти заедно.
Кодът осигурява логиката за управление, възпроизвеждане и манипулиране на плейлисти и песни. Той включва функции като създаване на плейлисти, добавяне на песни, контролиране на възпроизвеждането (възпроизвеждане, пауза, спиране) и други свързани функции.

- ***play.py:***
- Глобални:
- current_song_index: Тази глобална променлива държи позицията на текущо възпроизвежданата или избраната песен в плейлиста.
- playlist_songs: Списък, съдържащ песните в избрания плейлист.
- Основни функции:
- search_playlists():
- Цел: Тази функция позволява на потребителите да търсят в плейлисти по ключова дума.
- Логика:
- Грабва въведеното от потребителя от изпълнимия модул Entry.
- Търси плейлисти за съвпадения по начин, който не прави разлика между главни и малки букви.
- Актуализира all_playlists_listbox, за да покаже съответстващите плейлисти.
- play_next_song():
- Предназначение: Автоматично преминава към и възпроизвежда следващата песен в списъка за изпълнение, след като текущата песен завърши.
- Логика:
- Проверява дали current_song_index е в обхвата на песните в плейлиста.
- Използва модула pygame.mixer за зареждане и възпроизвеждане на следващата песен.
- Определя дължината на песента и настройва времето за възпроизвеждане чрез функцията play_time().
- Използва функцията root.after() за периодична проверка за завършване на песента.
- check_for_song_end():
- Цел: Проверява на интервали дали възпроизвеждането на текущата песен е приключило.

- Логика:
- Ако възпроизвеждането на песента завърши, то задейства функцията play_next_song().
- Ако не, той проверява отново след 500 милисекунди.
- create_new_playlist():
- Цел: Улеснява създаването на нов плейлист.
- Логика:
- Подканва потребителя за име на нов списък за изпълнение.
- Добавя плейлиста към базата данни.
- Обновява дисплея на плейлиста (all_playlists_listbox).
- song_addition_handler():
- Цел: Управлява добавянето на песни към избран плейлист.
- Логика:
- Потвърждава дали е избран плейлист.
- Подканва потребителя да избере файл с песен.
- Добавя избраната песен към избрания плейлист в базата данни.
- Обновява списъка с песни и подробностите за плейлиста.
- update_song_listbox():
- Цел: Отразява песните в списъка с песни въз основа на избрания списък за изпълнение.
- Логика: Извлича и показва песни, свързани с избрания плейлист.
- play_playlist():
- Предназначение: Инициира възпроизвеждане на всички песни от избран плейлист.
- Логика:
- Потвърждава избора на плейлист.
- Събира песни, свързани с плейлиста от базата данни.
- Стартира възпроизвеждането на списъка с песни в отделна нишка.
- play_time():
- Предназначение: Следи напредъка на възпроизвеждането на песни.
- Логика:
- Актуализира показаното текущо време и продължителност на песента.
- Променя позицията на плъзгача на песента въз основа на напредъка на възпроизвеждането.
- add_song():
- Цел: Добавя една песен към плейлиста.
- Логика: Подканва потребителя да избере песен и показва избраната песен в списъка с песни.
- играй():
- Цел: Инициира възпроизвеждането на избрана песен.
- Логика:
- Използва pygame.mixer за зареждане и възпроизвеждане на песента.
- Регулира съответно плъзгача на песните и нивата на звука.
- стоп():
- Предназначение: Спира възпроизвеждането на текущата песен.
- Логика:
- Нулира плъзгача на песента и състоянието на възпроизвеждане.

- Спира песента и изчиства всички избрани песни.
- пауза (e_пауза):
- Предназначение: Поставя на пауза и възобновява възпроизвеждането на песен.
- Логика: В зависимост от параметъра is_paused, функцията или ще постави на пауза, или ще възобнови песента с помощта на pygame.mixer.
- 13.slide(x):
- Цел: Регулира позицията на възпроизвеждане на песента с помощта на плъзгач.
- Логика: Променя позицията на възпроизвеждане въз основа на текущата стойност на плъзгача.
- Помощни функции:
- update_all_playlists_with_songs():
- Цел: Опреснява дисплея на плейлиста с песни, свързани с тях.
- Логика: Извлича всички плейлисти и свързаните с тях песни от базата данни, след което актуализира списъка.
- add_many_songs():
- Цел: Позволява добавянето на множество песни към плейлиста.
- Логика: Позволява на потребителя да избира няколко песни и попълва списъка с песни с избраните песни.
- next_song() & previous_song():
- Цел: Навигирайте в списъка с песни.
- Логика: Настройва current_song_index и използва pygame.mixer за зареждане и възпроизвеждане на съответната песен.
- delete_song() & delete_all_songs():
- Цел: Премахва песни от плейлиста.
- Логика: Спира песента, ако се възпроизвежда и изтрива избраната песен или всички песни от списъка.

• Графичната част от кодът:

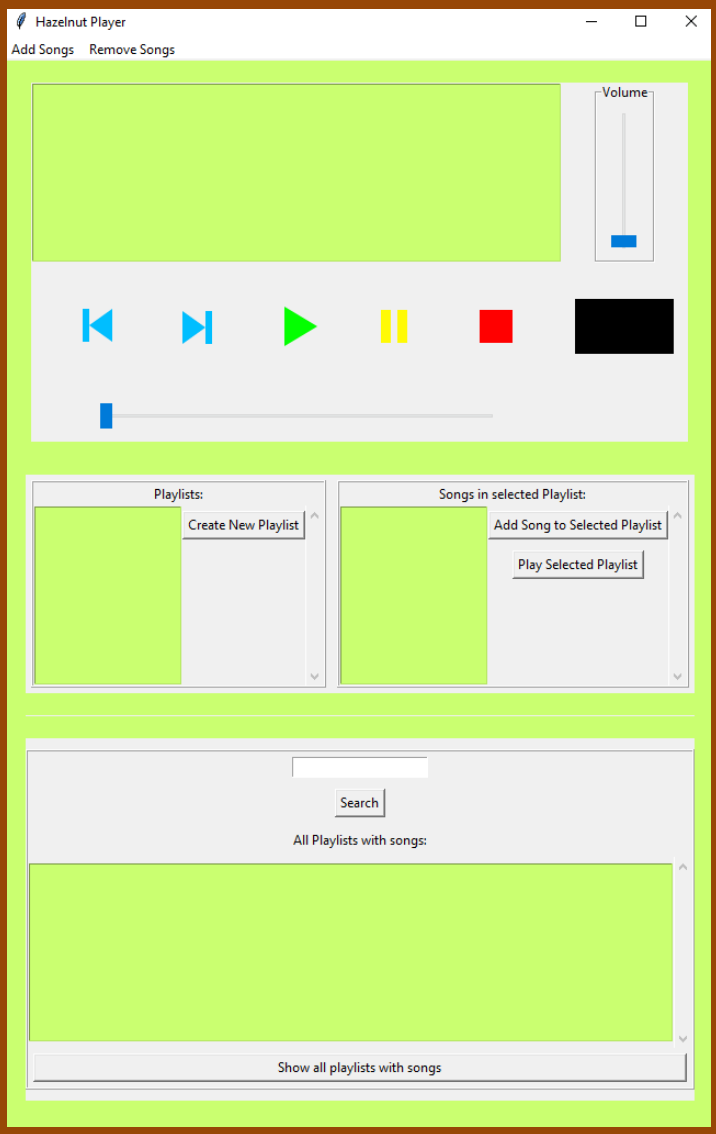
- Главна рамка:
- Създава се първичен кадър с име master_frame и се пакетира с подложка от 20 пиксела по оста y. Тази рамка действа като основен контейнер за много от основните компоненти на играча.
- Компоненти в основната рамка:
- Кутия за плейлист (song_box):
- Списъчно поле, което показва списъка с песни, добавени към плейъра.
- Има зелен фон и показва текущо избраната песен с бял фон и черен текст.
- Бутони за управление на плейъра:
- Показва се с помощта на изображения, заредени чрез класа PhotoImage.
- Пет налични бутона: Назад, Напред, Възпроизвеждане, Пауза и Стоп.
- Всеки бутон е поставен в специална рамка, наречена controls_frame.
- Контрол на звука:
- Пет изображения, представящи различни нива на звука.
- Измервателят на звука (volume_meter) показва текущото ниво на звука.
- Плъзгач за сила на звука (volume_slider) позволява на потребителите да регулират силата на звука. Намира се вътре в volume_frame.

- Плъзгач за музикална позиция (my_slider):
- Хоризонтален плъзгач, който вероятно представлява напредъка на възпроизвеждането на песента.
- Лента на състоянието (status_bar):
- Етикет, поставен в долната част на прозореца, може би използван за показване на състоянието на плейъра или други съобщения.
- Горно меню (my_menu):
- Падащо меню в горната лента на приложението.
- Предлага опции за добавяне на една/множество песни и изтриване на песни от плейлиста.
- Рамки за организация:
- Три основни рамки (top_frame, middle_frame, bottom_frame) се създават и опаковат за управление и организиране на оформлението:
- Компоненти в горната рамка:
- Рамка на плейлист (playlist_frame):
- Съдържа списъчно поле (playlist_listbox), което вероятно показва запазени плейлисти.
- Предоставя опция за създаване на нов плейлист чрез btn_create_playlist.
- Рамка на песни (songs_frame):
- Показва песни от избран плейлист.
- Предлага опции за добавяне на песни към избран плейлист или възпроизвеждане на плейлиста чрез бутоните btn_add_song и btn_play_playlist.
- Компоненти в долната рамка:
- Рамка на всички плейлисти (all_playlists_frame):
- Показва всички плейлисти със свързаните с тях песни.

- Има функция за търсене за намиране на конкретни плейлисти.
- Предоставя опция за актуализиране на списъка чрез btn_update_all_playlists.
- Окончателни конфигурации:
- Цвят на фона:
- Цветът на фона на главния прозорец е зададен на тъмнозелено1.
- Основен контур:
- root.mainloop() поддържа приложението работещо и изчаква потребителски взаимодействия.

4.7. Употреба на приложението:

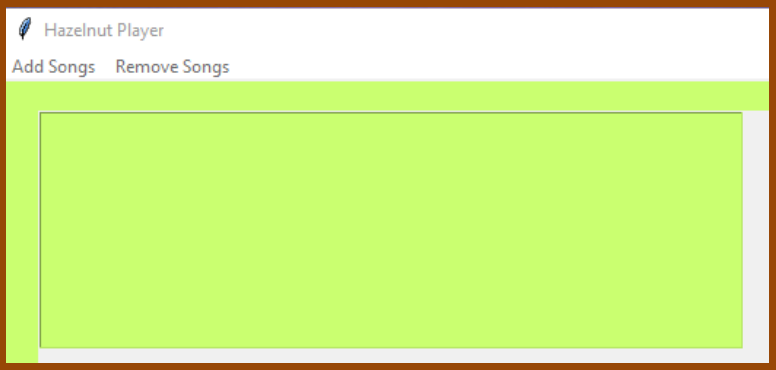
Първи стъпкиПри първоначално инициализиране потребителят вижда този екран:



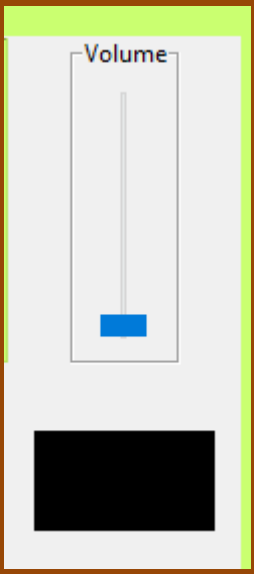
Основни контроли

При зареждане потребителят вижда набор от контроли за песни, контрол на звука и други.

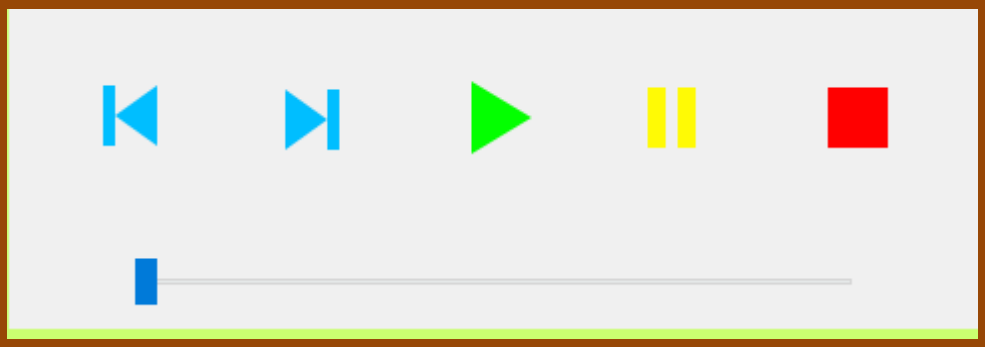
➢ Лист с песни за изпълнение – в него се визуализират песните, готови да бъдат прослушани.



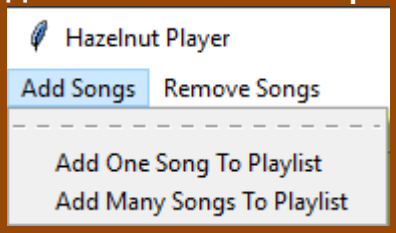
➢ Контрол на звука – използва се за регулиране на звука на приложението чрез слайдер и визуализирането на всяка степен на сила на звука със картинки



-Контроли за песните – използвате се за стартиране, пауза, изваждане на песен, следваща песен, предишна песен, слайдер. Ако песен не е маркирана, при кликане на следваща песен/старт се зарежда първата в листът за прослушване. При кликане на предишна песен, ако не е избрана такава се зарежда последнта в листа за прослушване. Чрез слайдера потребителят вижда колко време е преминало от песента и може да превърта напред или назад като манипулира позицията на курсора.

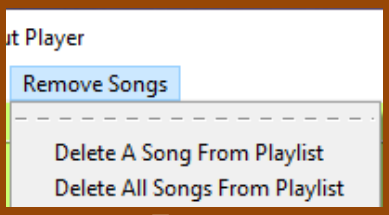


- **Добавяне на песен за прослушване**



Добавянето на песен става чрез основното меню на приложението (най - отгоре), използвайки опцията Add Songs. Кликайки на нея, потребителят вижда две опции – да добави една песен или много в плейлистът за изпълнение. При избор се отваря диалогов прозорец, който го подканва да избере песен от дадена директория или от някоя по негов избор.При така добавена песен тя се запазва само във временната памет на приложението, не в някой плейлист!

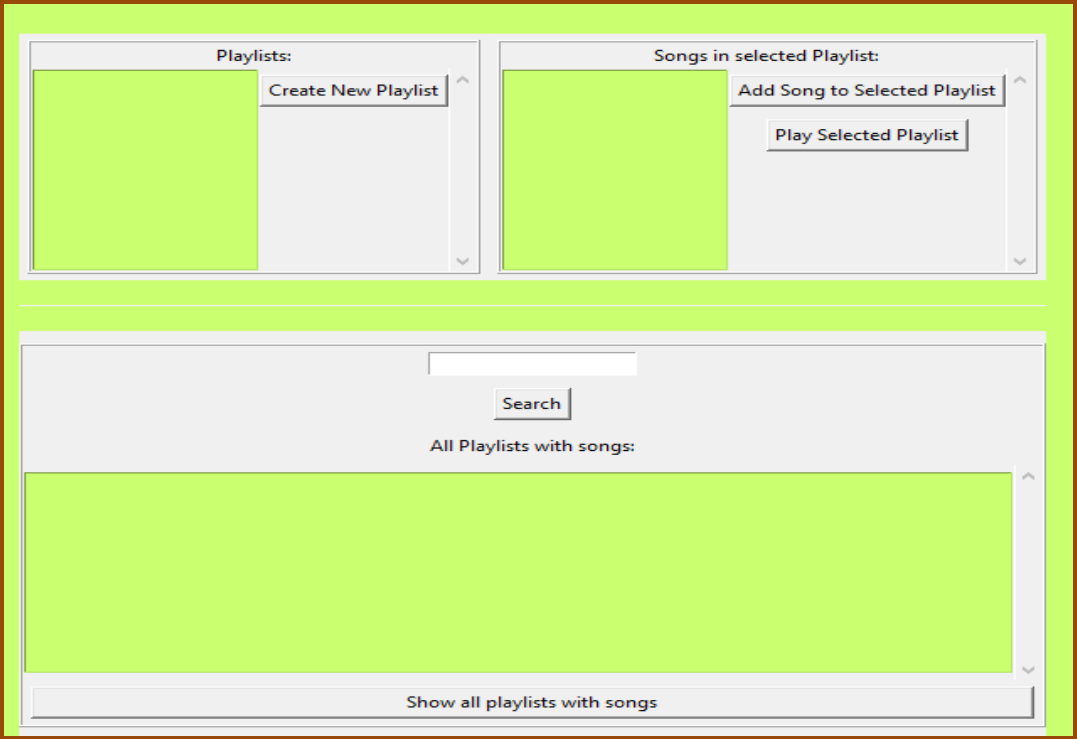
Премахване на песен за прослушване



Премахването на песен става чрез основното меню на приложението(най - отгоре), използвайки опцията Remove Songs. Кликайки на нея, потребителят отново вижда две опции – да премахне една селектирана песен или да премахне всички песни от плейлистът за

изпълнение.При така премахната песен, тя отпада само от временната памет на приложението.

- **Манипулация на плейлисти**



Състой се от три списъчни кутии и общо 5 бутона с които да извършваме манипулациите.
При натискане на бутона „Create New Playlist“ потребителя може да създаде колкото си поиска нови плейлисти с които се показват веднага в „All Playlists with songs:“ кутията. Когато този плейлист бива създаден потребителя може да добави песен към този плейлист който е маркиран в „Playlists:“ кутията с бутона „Add Song to Selected Playlist“.
А щом вече потребителя е конфигурирал по свой вкус плейлистът или колкото на брой той желае плейлисти с песните може да се наслади на любимите си песни избрани в конкретния плейлист като натисне бутона Play Selected Playlist независимо дали е маркиран в кутията Playlists или All Playlists with songs. И от двете кутии е възможно възпроизвеждането на плейлист

Заключение

5. Други подобни приложения

   Приложението е разработено по образец на всеки нормален mp3-плейър. Други подобни приложения са Groove Music, iTunes, Spotify, WMP, mp3 – плейърите на телефоните и други.

6. Бъдещо развитие на системата и недостатъци

   Развитие

   - Да може потребителя сам да решава коя песен да си пуска от избран от него плейлист
   - Да може потребителя да вижда колко е дълга като времетраене
   - Добавяне на бутон за разбъркано прослушване.

- Липсата за опция за изтриване на вече създадени плейлисти
- Добавяне на бутон за спиране и стартиране за завъртането на плейлиста
- Промяна на интерфейсът за да изглежда по - модерен и иновативен.

- Добавяне на бутон за прослушване на избраната песен отново
- Добавяне на жанрове музика
- Добавяне на любими

Вече известни проблеми

- Лиспата на правилно функциониращ слайдър който да визуализира времетраенето на песента от който да превърта на конкретна част от песента и песента да продължи от тази конкретна част.

7. Version control
   За контрол на версиите е използван GitHub.
   Линк: https://github.com/RaffyPetrov/playerproject