**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

# Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ДОДАТКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ З ВИКОРИСТАННЯМ C++ ТА QT»**

Виконав: студент 4 курсу, групи ПД-41 спеціальності
121 Інженерії програмного забезпечення
(шифр і назва спеціальності)

Гончаренко Д.С.
(прізвище та ініціали)
Керівник Дібрівний О.А.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль_____
(прізвище та ініціали)

КИЇВ – 2021

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного
забезпечення
Негоденко О.В.
"____" _____2021 року

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА**
Гончаренко Денису Сергійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи: «РОЗРОБКА ДОДАТКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ З ВИКОРИСТАННЯМ C++ ТА QT»
Керівник роботи Дібрівний О.А., Старший викладач кафедри ІПЗ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом вищого навчального закладу від "12" березня 2021 року № 65.
2. Строк подання студентом роботи "1" червня 2021 року
3. Вхідні дані до роботи:
Qt; TagLib; QtCreator; Audio Bass Library; QSql; Mediator.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Аналіз додатків-аналогів.
4.2 Розбір і обґрунтування використання сторонніх бібліотек та інщого програмного забезпечення.
4.3 Демонстрація функціоналу додатку.
4.4 Роз'яснення щодо реалізації додатку.
5. Перелік демонстраціного матеріалу (назва основних слайдів)
1 Актуальність теми
2. Об'єкт та предмет дослідження
3. Мета дослідження

4. Метод дослідженя
5. Практичне значення одержаних результатів
6. Редагування метаданих
7. Програвання музики
8. Еквалайзер та візуалізація звукових частот
9. Плейлисти
10. Авторизування та зберігання даних

6. Дата видачі завдання_____19.04.2021_____

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапу бакалаврської роботи | Строк виконання | Примітка |
|---|---|---|---|
| 1 | Підбір науково-технічної літератури | 19.04.2021 | виконав |
| 2 | Аналіз технологій | 20.04.2021 | виконав |
| 3 | Проектування системи | 23.06.2021 | виконав |
| 4 | Розробка програмного забезпечення | 24.06 - 28.06 | виконав |
| 5 | Вступ, висновки, реферат | 27.06 - 28.06 | виконав |
| 6 | Попередній захист роботи | 11.05 | виконав |
| 7 | Здача роботи | 1.06 | виконав |

Студент _____ _Гончаренко Д.С._
( підпис ) (прізвище та ініціали)
Керівник роботи _____ _Дібрівний О.А._
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

*Текстова частина* бакалаврської роботи 40 с., рис. 14, 8 джерел.

*Об'єктом дослідження* є технології і способи реалізації прослуховування та обробки аудіо контенту

*Предметом дослідження* є методи та способи отримання якісного аудіо контенту.

*Метою роботи* є розробка програмного забезпечення для прослуховування аудіофайлів.

*Методи дослідження* – методи теорії інформації, порівняльний аналіз способів реалізацій різних провідних компаній у сфері розробки і обробки аудіо контенту.

У роботі проведено аналіз шляху розробки додатку для прослуховування аудіофайлів з використанням C++ та Qt. Досліджено актуальну для сьогодення технологію розробки аудіофайлів та визначено властивості, які дозволяють забезпечити вимоги для систем доступу та різноманітних модифікацій прослуховування за допомогою додатку .

На першому етапі проведено дослідження теоретичних основ програм для роботи з аудіоінформацією, проведено аналіз можливостей монетизації даного розробленого додатку. На другому етапі написано функціональний код для розроблення та прослуховування аудіофайлів з використанням C++ та Qt

Досліджено та запропоновано до реалізації універсальний алгоритм розроблення та прослуховування аудіофайлів з використанням C++ та Qt.

*Галузь використання* – сучасні системи аудіофайлів.

*Ключові слова:* **АУДІОФАЙЛИ, АУДІОПЛЕЄРИ, ОПТИМАЛЬНІ МЕТОДИ, АВТОКОРЕЛЯЦІЙНИЙ ПРИЙОМ СИГНАЛІВ, C++,Qt**

**ЗМІСТ**

# ВСТУП

**Актуальність теми.** На даному етапі розвитку сучасних технологій важливе місце займає технічна реалізація сервісів прослуховування та цифрової обробки аудіо контенту, зокрема дистанційно із використанням C++ та Qt. Такі програми пропонують можливість розробки додатків для прослуховування аудіофайлів і надають всі можливості для організації процесів збирання, змішування та обробки аудіоданих, а також для остаточного засвоєння. Їх наявність спрощує процес запису та обробки, виключаючи потребу в просторі для інструментів та придбання дорогого обладнання для управління звуком та змішування. C++ Builder -- середовище швидкої розробки (RAD), що випускається компанією Codegear, дочірньою фірмою компанії Embarcadero (раніше Borland). Призначена для написання програм на мові програмування C++. C++ Builder, об'єднує Бібліотеку візуальних компонентів і середовище програмування (IDE), написане на Delphi з компілятором C++. Цикл розробки аналогічний Delphi, але з істотними поліпшеннями, доданими в C++ Builder. Більшість компонентів, розроблених в Delphi, можна використовувати і в C++ Builder без модифікації, але, на жаль, зворотне твердження не вірне.

C++ Builder містить інструменти, які дозволяють здійснювати справжню візуальну розробку Windows-програм методом drag-and-drop, спрощуючи програмування завдяки WYSIWYG редакторові інтерфейсу, вбудованому в його середовище розробки.

Дослідження є доцільним оскільки дані програми широко застосовуються в наш час. Вони корисні як для звичайного користувача-початківця, що тільки починає свою роботу зі звуком, так і для користувачів, які мають певні навики роботи у цій царині.

**Метою роботи** є розробка програмного забезпечення для прослуховування аудіофайлів.

Для досягнення поставленої мети потрібно вирішити наступні завдання:

**Об’єктом дослідження** є технології і способи реалізації прослуховування та обробки аудіо контенту

**Предметом дослідження** є методи та способи отримання якісного аудіо контенту.

**Методом дослідження** є порівняльний аналіз способів реалізацій різних провідних компаній у сфері прослуховування і обробки аудіо контенту.

**Практичне значення одержаних результатів**. Отримані результати в роботі можна використати для вибору технологій оптимального прослуховування та обробки аудіо контенту.

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ПРОГРАМ ДЛЯ РОБОТИ З АУДІОІНФОРМАЦІЄЮ

## 1.1.Редактори цифрового аудіо

Тема програмного забезпечення дуже широка, тому ми лише коротко обговоримо тут основних представників програм обробки звуку.

Найважливіший клас програм - це цифрові аудіоредактори. Основними особливостями таких програм є принаймні можливість записувати (оцифровувати) аудіо та зберігати його на жорсткому диску. Розроблені представники цього типу програм дозволяють набагато більше: запис, багатоканальну компіляцію звуку на декількох віртуальних доріжках, обробку спецефектів (як вбудованих, так і зовнішніх підключень - про це пізніше), придушення шуму, вдосконалену навігацію та інструменти форма спектроскопа та інші віртуальні інструменти, управління / управління зовнішніми пристроями, перетворення звуку з формату у формат, генерація сигналів, записування компакт-дисків та багато іншого. Деякі з цих програм: Cool Edit Pro (Syntrillium), Sound Forge (Sonic Foundry), Nuendo (Steinberg), Samplitude Producer (Magix), Wavelab (Steinberg).

Основними функціями редактора Cool Edit Pro 2.0 є приклад робочого вікна програми в багатодорожному режимі: редагування та компіляція звуку на 128 доріжках, 45 вбудованих ефектів DSP, включаючи інструменти для освоєння, аналізу та відновлення звуку, 32-розрядні Обробка, підтримка аудіо з 24 параметрами біт / 192 кГц, потужні інструменти для роботи з шторами (петлями), підтримка DirectX, а також управління SMPTE / MTC, підтримка відео та MIDI тощо.

Основними особливостями редактора Sound Forge 6.0a є приклад робочого вікна програми: потужні неруйнівні функції редагування, багатозадачність фонової обробки завдань, підтримка файлів з параметрами до 32 біт / 192 кГц, попередньо встановлений менеджер, підтримка файли, розмір

яких перевищує 4 Великобританії, робота над відео, обробка великих наборів ефектів, відновлення після зависання, попередній перегляд застосованих ефектів, аналізатор спектра тощо.

## 1.2.Програми для написання музики

Не менш важлива у функціональному сенсі група програм - секвенсорів (програм для написання музики). Здебільшого такі програми використовують синтезатор MIDI (зовнішнє обладнання, вбудоване майже в кожну звукову карту, або програмне забезпечення, організоване спеціальним програмним забезпеченням). Такі програми надають користувачеві або звичайний стан нот (наприклад, програму Finale від CODA), або більш поширений метод маніпулювання звуком на комп'ютері, який називається фортепіанний рулон (це чіткіше представлення музики людям) Ноти незнайомі; в такому поданні є вісь вертикальна із зображенням клавіш піаніно, а час зміщується горизонтально, так що при створенні ділянок штрихів різної довжини досягається звучання певної ноти з певною тривалістю).

Є також програми, які дозволяють переглядати та редагувати аудіо в обох виставах. На додаток до обробки звуку, вдосконалені секвенсори можуть значною мірою дублювати функції програм цифрової обробки звуку - для запису на CD, поєднання MIDI-треків з цифровими сигналами та мастерингу. Яскраві представники цього програмного класу: FL Studio, Cubase, Ableton Live, Logic Audio, Cakewalk

FL Studio (раніше Fruity Loops) - це редактор секвенсорів для написання музики, створений у 1997 році програмістом Дідьє Дембреном (також відомим як «гол»), який розробляв програму протягом восьми років і виробляв її за допомогою Image-Line Software. Музика створюється шляхом запису та компіляції (запису) аудіо- або MIDI-матеріалів. Готову пісню можна записати у файл із розширенням WAV, MP3 або OGG. Програма написана мовою програмування Delphi.

Програма містить 4900 звукових ефектів, які можна використовувати для композиції та редагування музики. Сьогодні продуктом компанії користуються близько 80 000 користувачів у всьому світі.

Названий на честь героя фільму "Хакери" - Фрута Зашморга. В даний час розробляється дванадцята версія програми

інтерфейс

Основні елементи FL Studio:

- Step Sequencer - Дозволяє швидко створювати та редагувати лупи, додавати нові генератори (канали) та видаляти непотрібні генератори;
- Фортепіанний рулон - це двовимірна сітка, на вертикальній осі якої встановлений рівень висоти тону, по горизонталі час можливий більше, ніж кроковий секвенсор; Цифровий трекер аудіозаписувача;
- Список відтворення - дозволяє розмістити його в послідовності послідовності або циклі фортепіано, або розмістити аудіофайли;
- Змішувач - ось склад та обробка його плагінів та ефектів;
- Зразок браузера - легкий доступ до аудіофайлів, плагінів та налаштувань.

Системні вимоги

FL Studio 10.0 працює на комп'ютерах під управлінням Windows 2000 / XP / Vista / 7/8 (32- і 64-розрядні версії) та на Intel Macs з Boot Camp. Для роботи потрібен процесор з тактовою частотою 2 ГГц - AMD або Intel Pentium 3 з повною підтримкою SSE1. Потрібно 1 ГБ місця для зберігання та принаймні 1 ГБ оперативної пам'яті.

FL Studio обробляє звук за допомогою внутрішнього 32-розрядного алгоритму з плаваючою комою. Він підтримує частоту дискретизації до 192 кГц з драйверами WDM та ASIO.

FL Studio - це секвенсор шаблону доріжок, де музика створюється на панелях «Фортепіано-рол» та «Секвенсер», а потім компілюється у вікні списку відтворення та завершується у вікні «Міксер». Існує велика кількість

попередньо виготовлених інструментів та безліч ефектів, які можна використовувати в режимі реального часу.

Редактор фортепіанного рулону використовується для написання складних поліфонічних партій. Фортепіано - це, по суті, музична умова, але в більш доступній формі: ви можете писати партитуру лише на основі власного слуху. Редактор представлений у формі віртуальної клавіатури фортепіано і містить 128 клавіш. Кожна клавіша - це примітка. Цей тип візуального редактора позбавляє потреби в додаткових змінах і паузах символів, що значно спрощує створення партитури.

Плейлист цього редактора розділений на розділи. У цих розділах вказується період звучання ноти. Тобто клавіатура - вісь Y, а обмежувач часу - вісь X.

Основною складовою композиційного проекту є генератор (канал).

Генератор синтезує або відтворює звук. Генераторів в дизайні композиції може бути необмежена кількість. Кожен генератор має свої власні налаштування, унікальний звук, що імітує кожен інструмент. Музичні ноти, записані у фортепіано-ролі, запрограмовані для генераторів. Бали в FL Studio мають кінцеву довжину. Партитури (візерунки) розташовуються один за одним у вікні списку відтворення (розташовані в тому порядку, як ви хочете в списку відтворення). Звук кожного генератора може оброблятися з багатьма ефектами.

Ви можете підключити будь-який плагін VST або DXi як генератор. Крім того, FL Studio можна використовувати в інших музичних редакторах як плагін VST або DXi.

У програмах користувача, що займається обробкою звуку, багато різних інструментів, як це було раніше, так і буде в майбутньому - універсальних комбайнів для роботи зі звуком не існує. Незважаючи на все різноманіття програмного забезпечення, програми часто використовують подібні механізми обробки звуку (наприклад, процесори ефектів та інші). У певний момент у розробці аудіо програмного забезпечення виробники зрозуміли, що

зручніше включати зовнішні інструменти до своїх програм, ніж створювати інструменти з нуля для кожної окремої програми. Так багато програм, що належать до тієї чи іншої групи програмного забезпечення, дозволяють підключати так звані «плагіни» - зовнішні модулі плагінів, щоб розширити можливості обробки звуку. Це стало можливим завдяки появі декількох стандартів для інтерфейсу між програмою та модулем підключення. Поки що існує два основних стандарти інтерфейсу: DX і VST. Наявність стандартів дозволяє асоціювати один і той же плагін із абсолютно різними програмами, не турбуючись про конфлікти та проблеми. Говорячи про самі плагіни, я повинен сказати, що це лише одна величезна родина програм. Як правило, один плагін - це механізм, який реалізує певний ефект, наприклад В. Реверберація або фільтр низьких частот.

Цікаві плагіни включають iZotope Vinyl (що дозволяє надати звуку ефект вінілової платівки - приклад робочого плагіна вікна в Cool Edit Pro). Antares AutoTune дозволяє напівавтоматично регулювати звук Orange Vocoder - чудовий вокодер, який відтворює звук різних інструментів, схожий на звук людського голосу.

Особливістю FL Studio є те, що за допомогою цього програмного продукту можна писати музичні композиції будь-якої складності, не маючи музичної підготовки та відповідних навичок одночасно. Через ці та інші фактори FL Studio є одним з найпопулярніших музичних редакторів, що використовується великою кількістю професійних музикантів та аматорів.

### 1.3.Програми-аналізатори аудіо

Обробка звуку та написання музики - це не просто творчі процеси. Іноді потрібен ретельний аналіз даних, а також здійснення пошуку недоліків у їх звучанні. Крім того, звук, з яким доводиться мати справу, не завжди може бути такої якості, яку ви бажаєте. У цьому контексті ми повинні згадати ряд програм аудіоаналізатора, спеціально розроблених для проведення аналізу

вимірювань аудіоданих. Такі програми допомагають подавати звукові дані зручніше, ніж традиційні редактори, і ретельно вивчати їх за допомогою різних інструментів, таких як: Б. ШПФ-аналізатори (конструктори для динамічних та статичних амплітудно-частотних властивостей), конструктори для сонограм та інші. Одна з найвідоміших і найрозвиненіших програм цього плану - програма SpectraLAB (Sound Technology Inc.), трохи простіша, але потужна - Analyzer2000 та Spectrogram.

SpectraLAB - це потужний та ефективний аналізатор звуку з можливістю вимірювати діапазон значень і має вбудований генератор сигналів.

Програма SpectraPLUS дозволяє здійснювати ШПФ-аналіз звукового сигналу в режимі реального часу з високою роздільною здатністю. Точність дискретизації аналізатора становить 24 біти, алгоритм ШПФ обробляє записи довжиною до 1048576 семплів, частота дискретизації досягає 200 кГц (в залежності від можливостей звукової карти це значення може бути навіть вище) і октави аналіз знаходиться в межах від 1/1 до 1/96.

Діаграма амплітудно-частотних властивостей

Для відображення інформації доступні такі режими: функція часу, спектрограма, фазовий спектр. Ефективним інструментом для аналізу складних спектральних візерунків є побудова різнокольорових тривимірних поверхонь. Крім того, доступні такі функції, як аналіз спотворень, цифрова фільтрація, застосування згладжування вікон, обробка, розподіл, перекриття, стоншення, утримання піків, вузькосмугове або октавне масштабування. Програмне середовище здатне записувати та відтворювати WAV-файли з подальшою обробкою.

Аналізатор SpectraPLUS містить двоканальний генератор сигналів. У кожному з каналів сигнали абсолютно незалежні. Можна створювати тони з одним або декількома тонами, рожевим або білим шумом, скануванням, пульсаціями, пилкоподібними імпульсами, квадратною та трикутною формою, DTMF та кроковими сигналами тощо.

2D спектрограмма звуку

У програмі є ряд додаткових утиліт:

- RT60 вимірює, аналізує і будує гістограму часу загасання реверберації в залежності від діапазону частот, а також спектральної характеристики акустичного середовища;
- LEQ проводить комплексну оцінку рівня шуму;
- Delay Finder здатна розрахувати час затримки між лівим і правим каналами в мілісекундах, футах або метрах;
- THD + N vs Frequency вимірює характеристики спотворень, привнесених звуковою картою в широкому частотному діапазоні;
- Stereo Phase Scope аналізує фази і полярності сигналів;

Крім цього ведеться запис логів, результати аналізу здатні зберігатися або роздруковуватися, а процес проведення вимірювань можна легко автоматизувати, налаштувавши запис спектральних даних інтервалами певної тривалості в потрібні проміжки часу. Існує можливість написання додаткових скриптів, що підтримують синтаксис DDE.

SpectraPLUS є нащадком відомих програм SpectraPRO і SpectraLAB. На сайті розробника представлені два варіанти програми: стандартна SpectraPLUS-SC (для роботи зі звуковими картами) і професійна SpectraPLUS-DT (для роботи з Data Translation DT-9800 Industrial A / D modules).

Програмне забезпечення SpectraPLUS було розроблено компанією Pioneer Hill Software. Організація була заснована в 1993 році, головний офіс компанії знаходиться в невеликому містечку Poulsbo (Вашингтон, США). Pioneer Hill Software спеціалізується на створенні програмного забезпечення в області цифрової обробки сигналів.

3D спектрограмма звуку

Інтерфейс данного середовища SpectraPLUS тільки англійський.

Програма відмінно працює на малопотужних комп'ютерах і не потребує будь-якого додаткового обладнання. Вимірюваний аудіосигнал подається на лінійний або мікрофонний вхід, після чого SpectraPLUS за допомогою звукової карти виконує аналого-цифрове перетворення. Таким чином, для

здійснення детального аналізу звуку досить лише 32- або 64-розрядної операційної системи Windows XP, Vista або 7, а також Windows-сумісної звукової карти.

### 1.4.Спеціалізовані реставратори та трекери аудіо

Спеціалізовані реставратори аудіо грають також важливу роль в обробці звуку. Такі програми дозволяють вам відновити втрачену якість звуку аудіоматеріалу, видалити небажані клацання, шуми, тріск і специфічні збої із записів з аудіокасет і внести інші корективи в звук. Такі програми: Dart, Clean (від Steinberg Inc.), Audio Cleaning Lab. (Від Magix Ent.), Коректор хвиль.

Основні особливості реставратора Clean 3.0 - робоче вікно програми: усунення всіх тріщин і шумів, режим автокорекції, набір ефектів для обробки індивідуального звуку, включаючи «об'ємний звук» з візуальною акустичною симуляцією ефекту, Запис на компакт-диск із підготовленими даними, "інтелектуальна" система сповіщень, підтримка зовнішніх плагінів VST та інших функцій.

Tracker - загальний термін для класу програмних музичних секвенсорів, які у своїй найпростішій формі дозволяють користувачеві розміщувати зразки звуку послідовно на декількох монофонічних каналах (треках) з часом. Інтерфейс трекера в основному цифровий. Нотатки вводяться за допомогою клавіатури, тоді як параметри, ефекти тощо вводяться у формі латинських літер та цифр (зазвичай шістнадцяткових). Готова музична композиція складається з декількох невеликих багатоканальних фрагментів - візерунків, порядок яких визначається основним списком - так званим списком замовлень.

Принципи роботи

Загальними елементами для всіх трекерів є зразки, нотатки, ефекти, канали (треки), шаблони та їх порядок.

Зразок - це невеликий фрагмент оцифрованого інструменту, голосу чи іншого звукового ефекту. Більшість трекерів дозволяють циклічно повторювати частину зразка та імітувати ноти тривалого звуку.

Примітка визначає частоту відтворення вибірки. Збільшення або зменшення швидкості відтворення оцифрованого зразка збільшує або зменшує висоту ноти (висоту тону), імітуючи інструментальні ноти (наприклад, C, C #, D тощо).

Ефект - спеціальна функція, що застосовується до конкретних нот. Загальними для всіх трекерів є такі ефекти, як зміна гучності, глісандо, портаменто, вібрато, ретригер, арпеджіо та панорамування.Патерн - група одночасно відтворених каналів, що представляє повноцінну частину музичної композиції.

Порядок - послідовність відтворення патернів, що визначає структуру музичної композиції. Усередині послідовності патерни можуть повторюватися, таким чином можливо відносно швидке створення загальної структури твору.

Існують також трекери, що використовують замість семплів синтез звуків в реальному часі. Багато з цих програм призначені для створення музики за допомогою мікросхем звукогенератор, таких як мікросхема OPL в звукових картах Adlib і SoundBlaster, або звукових мікросхем класичних побутових комп'ютерів. Більшість сучасних трекерів можуть використовувати в якості джерел звуків VST і інструменти.

Трекерна музика зазвичай зберігається в файлах, які називаються «модулями», де інформація про структуру композиції і семпли містяться всередині одного і того ж файлу. Відтворення більшості «класичних» форматів трекерна модулів підтримується такими популярними музичними плеєрами, як Winamp, XMMS, foobar2000, а також ModPlug Player, окремо пристосованим для програвання модулів основної частини коли-небудь існували форматів і їх варіацій. Найбільш поширеними форматами модулів є: MOD, S3M, XM і IT.

## 1.5.Мережні технології для створення аудіо контенту

Сервіси створення та цифрової обробки аудіо контенту із застосуванням мережних технологій мають забезпечувати всі можливості для організації процесів зведення, мікшування та обробки аудіо даних, а також фінального мастерінг. Вони спрощують процес запису і обробки, оскільки зникає потреба у великому просторі для інструментів та купівлі дорогої апаратури для регулювання звуку і його мікшування. В наступних розділах більш детально розглянуто основні хмарні технології.

З поняттям хмарових обчислень пов’язують такі сервісні технології, як:

- Інфраструктура як сервіс («Infrastructure-as-a-Service» або «IaaS»);
- Платформа як сервіс («Platform-as-a-Service» або «PaaS»);
- Програмне забезпечення як сервіс («Software-as-a-Service» або «SaaS»);
- Розглянемо кожну з цих технологій докладніше.

Інфраструктура як сервіс (IaaS) – це надання компютерної інфраструктури як послуги на основі концепції хмарових обчислень. «Інфраструктура як сервіс» складається з таких основних компонентів:

- апаратні засоби (сервери, системи зберігання даних, клієнтські системи, мережеве обладнання);
- операційні системи та системне програмне забезпечення (засоби віртуалізації, автоматизації, основні засоби управління ресурсами);
- сполучне-програмне забезпечення (наприклад, для управління системами).

"Інфраструктура як сервіс" базується на технології віртуалізації, яка дозволяє користувачеві розділити пристрої на частини, що відповідають сучасним вимогам компанії, збільшуючи тим самим ефективність використання існуючих обчислювальних потужностей. Користувач (бізнес або розробник програмного забезпечення) повинен платити лише за час сервера, простір пам’яті, пропускну здатність мережі та інші ресурси, які їм

дійсно потрібні для роботи. Крім того, IaaS пропонує замовнику всі адміністративні функції на одній інтегрованій платформі.


Рис. 1.5.1 – Компоненти хмарової інфраструктури

Інфраструктура як сервіс – позбавляє підприємства від необхідності підтримки складних інфраструктур центрів обробки даних, клієнтських і мережевих інфраструктур, а також дозволяє зменшити повязані з цим капітальні витрати та поточні витрати. Крім того, можна отримати додаткову економію, при наданні послуги в рамках інфраструктури спільного використання. Першопрохідцями в «IaaS» вважається компанія «Amazon», яка на сьогоднішній день пропонує два основних «IaaS-продукти»: EC2 (ElasticComputeCloud) і S3 (SimpleStorageService). «EC2» являє собою «Xenхостинг» зі статичними VPS (VirtualPrivateServer) – характеристиками, які не передбачають швидкого розширення. Сховище «S3» має інтерфейс WebDAV і підтримує роботу з багатьма відомими мовами програмування. Серед інших інфраструктурних сервісних компаній можна відзначити компанію «GoGrid», яка має дуже зручний інтерфейс для управління VPS, а

також «cloudstorage» з підтримкою протоколів SCP, FTP, SAMBA/CIFS, RSYNC, з можливістю динамічного масштабування.

Програмний пакет Enomaly - це рішення для розгортання та управління віртуальними програмами в хмарі, а управління послугами здійснюється через браузер. Важливим доповненням є автоматичне масштабування віртуальних машин до поточного навантаження, а також автоматичне вирівнювання навантаження. До підтримуваних віртуальних архітектур належать гості Linux, Windows, Solaris та BSD. Для віртуалізації використовуються не тільки "Xen", а й "KVM" та "VMware". Eucalyptus - це програмний пакет з відкритим кодом для реалізації хмарних обчислень на кластерних системах. На даний момент інтерфейс сумісний з "Amazon EC2".

Платформа як сервіс - це надання інтегрованої платформи для розробки, тестування, розгортання та підтримки веб-додатків як послуг. Розробнику не потрібно купувати апаратне та програмне забезпечення для розгортання веб-програм. Підтримка не повинна бути організована. Доступ для клієнта може бути організований на основі лізингу. Основні переваги такого підходу:

- масштабованість;
- відмовостійкість;
- віртуалізація;
- безпека.

Масштабованість платформи як сервісу передбачає автоматичне виділення та звільнення необхідних ресурсів в залежності від кількості користувачів. Здатність створювати вихідний код і надавати його в загальний доступ розробникам значно підвищує продуктивність по створенню нових програм на основі «Platform-as-a-Service». Найвідомішим прикладом такої платформи є «AppEngine» від Google, яка пропонує хостинг для веб-програм з можливістю купувати додаткові обчислювальні ресурси (наприклад, для тестування високих навантажень).

Для запуску додатків «GoogleAppEngine» на віртуальних кластерних системах була розроблена платформа «AppScale». В системах веб-пошуку і

контекстної реклами компанії Yahoo використовується платформа «Hadoop», орієнтована на передачу великих обсягів даних між мережевими серверами. На базі «Hadoop» побудовані HBase (аналог бази даних GoogleBigTable), а також HDFS (HadoopDistributedFileSystem, аналог GoogleFileSystem). Ще одним яскравим представником «PaaS» є продукти компанії Mosso:

- CloudSites–веб-хостинг (Linux, Windows, Mail) для перевантажених веб-проектів з можливістю розширювати базові безкоштовні можливості за додаткову плату (трафік, сховище даних, обчислювальна потужність);
- CloudFiles – файловий хмаровий-хостинг із щомісячною погігабайтной оплатою. Управління здійснюється через браузер, або за допомогою API (PHP, Python, Java,.NET, Ruby);
- CloudServers – погодинна оренда серверів (RAM на годину), з можливістю вибору серверної операційної системи. Можна змінювати характеристики сервера, але не в режимі реального часу.

В центрі всієї хмарної інфраструктури Microsoft – операційна система Windows Azure. Windows Azure створює єдине середовище, що включає хмарові аналоги серверних продуктів Microsoft (реляційна база даних SQL Azure, що є аналогом SQL Server, а також Exchange Online, SharePoint Online і Microsoft Dynamics CRM Online) і інструменти розробки (.NET Framework і VisualStudio, оновлена в версії 2010 року набором Windows AzureTools). Так, наприклад, програміст, який створює сайт в VisualStudio 2010, може не виходячи з програми розмістити свій сайт в Windows Azure.

Програмне забезпечення як сервіс – це модель розгортання програми, яка передбачає надання програми кінцевому користувачеві у вигляді послуги. Доступ до такого додатку здійснюється за допомогою мережі, а частіше за все за допомогою Інтернет-браузера. В даному випадку, основна перевага моделі «Software-as-a-Service» для клієнта полягає у відсутності витрат, пов’язаних з встановленням, оновленням і підтримкою працездатності обладнання та програмного забезпечення. Цільова аудиторія – кінцеві споживачі. В моделі «SaaS»:

- програма пристосована для віддаленого використання;
- однією програмою можуть користуватися декілька клієнтів;
- оплата за послугу стягується, або як щомісячна абонентська плата, або на основі сумарного обсягу транзакцій.

З точки зору розробників програмного забезпечення, модель «SaaS» дозволить ефективно боротися з неліцензійним програмним забезпеченням, завдяки тому, що клієнт не може зберігати, копіювати та встановлювати програмне забезпечення. По-суті, програмне забезпечення в рамках «SaaS» можна розглядати як більш зручну і вигідну альтернативу внутрішнім інформаційним системам. Розвитком логіки «SaaS» є концепція «WaaS» (Workplaceas a Service – робоче місце як послуга). Тобто клієнт отримує в своє розпорядження повністю оснащене всім необхідним для роботи програмним забезпеченням віртуальне робоче місце.

За нещодавно опублікованими даними «SoftCloud», попитом користуються наступні «SaaS» програми (у порядку популярності): · Пошта; · Комунікації (VoIP); · Антиспам і антивірус; · Helpdesk; · Управління проектами; · Дистанційне навчання; · CRM; · Зберігання і резервування даних


Рис. 1.5.2 – Взаємозв'язок хмарових сервісів

Окрім різних варіантів надання послуг, існують різні варіанти надання хмарних систем. Приватна хмара - використовується для надання послуг у

компанії, яка є одночасно клієнтом та постачальником послуг. Це варіант «хмарної концепції», коли компанія створює її для себе в рамках організації. По-перше, впровадження приватної хмари усуває одну з важливих проблем, з якою клієнти неминуче стикаються, коли вони знайомляться з цією концепцією, - проблему захисту даних стосовно інформаційної безпеки. Оскільки хмара обмежена самою компанією, ця проблема вирішується за допомогою існуючих стандартних методів. Приватна хмара характеризується зменшенням витрат на пристрої за рахунок використання ресурсів, які не використовуються або використовуються неефективно. А також знизити вартість придбання обладнання за рахунок зменшення логістики. Насправді продуктивність зростає пропорційно, загальне навантаження зростає незалежно від кожного завдання, яке виникає. Публічна хмара - використовується хмарними провайдерами для надання послуг зовнішнім клієнтам. Змішана (гібридна) хмара - спільне використання приватних та державних моделей доставки. Взаємозв'язок між хмарами різних типів (рис. 1.3).


Рис. 1.5.3 – Взаємозв’язок між хмарами різних типів

Таким чином, хмарові технології при спільному використанні дозволяють користувачам користуватись обчислювальними потужностями і сховищами даних, які за допомогою певних технологій віртуалізації надаються їм як послуги.

# РОЗДІЛ 2. ОПИС ВИКОРИСТАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ФУНКЦІОНАЛУ ДОДАТКА

## 2.1.Вибір програмного забезпечення для розробки аудіоплеєра.

Для створення аудіо плеєру було використано мову програмування C++ з фреймворком Qt та бібліотеками BASS audio library і Taglib.

C++ - статично-типова обєктно-орієнтована мова програмування , гнучка та швидка в використанні через можливість не автоматичного виділення та очищення памяті, що несе в собі загрозу при неправильному використанні недосвідченими користувачами.

Qt - кроссплатформений інстументарій розробки програмного забезпечення (фреймворк), містить у собі велику кількість готових класів, як для графічного інтерфейсу, так і для роботи з базами даних та сітьовими протоколами, потоками.

BASS audio library - звукова бібліотека яка дозволяє редагувати потоки аудіо файлів в режимі реального часу.

Taglib - бібліотека для зчитування та редагування метаданних аудіофайлів.

## 2.2.Обґрунтування вибору Qt

Qt надає програмісту не тільки зручний набір бібліотек класів, а й певну модель розробки додатків, певний каркас їх структури. Дотримання принципів і правил «гарного стилю програмування на C ++ / Qt» істотно знижує частоту таких важко відловлювати помилок в додатках, як витоку пам'яті (memory leaks), необроблені виключення, незакриті файли або незвільнені дескриптори ресурсних об'єктів, чим нерідко страждають програми, написані «на чистому C ++ »без використання бібліотеки Qt.

Важливою перевагою Qt є добре продуманий, логічний і стрункий набір класів, що надає програмісту дуже високий рівень абстракції. Завдяки цьому програмістам, які використовують Qt, доводиться писати значно менше коду, ніж це має місце при використанні, наприклад, бібліотеки класів MFC. Сам же код виглядає стрункішою і простіше, логічніше і зрозуміліше, ніж аналогічний за функціональністю код MFC або код, написаний з використанням «рідного» для X11 тулкіта Xt. Його легше підтримувати і розвивати.

Qt не є мовою програмування сама по собі. Це фреймворк, написаний на C ++. Препроцесор, MOC (Meta-Object Compiler), використовується для розширення мови C ++ такими функціями, як сигнали та слоти. Перед етапом компіляції MOC аналізує вихідні файли, написані на розширеному Qt C ++, і генерує з них стандартні сумісні джерела C ++. Таким чином, сам фреймворк та програми / бібліотеки, що використовують його, можуть бути скомпільовані будь-яким стандартним компілятором C ++, таким як Clang, GCC, ICC, MinGW та MSVC.

За допомогою Qt графічні інтерфейси можна писати безпосередньо на C ++ за допомогою модуля Widgets. Qt також постачається з інтерактивним графічним інструментом під назвою Qt Designer, який функціонує як генератор коду для графічних інтерфейсів на основі віджетів. Qt Designer можна використовувати окремо, але він також інтегрований у Qt Creator.

## 2.3. QtCreator

Одним з найголовніших досягнень Qt Creator є те, що він дозволяє команді розробників працювати над проектом на різних платформах з використанням загальних інструментів для розробки і налагодження.

Але навіщо вам потрібні проекти? Щоб бути в змозі збирати і запускати додатки, Qt Creator потребує тієї ж інформації, яка буде потрібно компілятору. Ця інформація вказана в налаштуваннях збірки і запуску проекту.

Створення проекту дозволить вам:

- групувати файли разом;
- додати власні кроки збірки;
- включити форми і файли ресурсів;
- вказати налаштування для запускаються.

Ви можете або створити проект з нуля, або імпортувати існуючий проект. Qt Creator генерує всі необхідні файли в залежності від типу створюваного проекту. Наприклад, якщо ви оберете створення додатка з графічним інтерфейсом користувача (GUI), Qt Creator створить порожній .ui файл, який ви можете змінити в інтегрованому Qt Designer.

Qt Creator інтегрований з кроссплатформенними системами автоматизації збирання: qmake і CMake. Також ви можете імпортувати існуючі проекти, які не використовують qmake або CMake, і вказати Qt Creator просто проігнорувати вашу систему збирання.

Редактори

Qt Creator поставляється з редактором коду і Qt Designer для проектування і складання графічних інтерфейсів користувача (GUI) з віджетів Qt.

Так як він є IDE, Qt Creator відрізняється від текстового редактора тим, що знає як збирати і запускати додатки. Він розуміє мови C ++ і QML як код, а не як простий текст. Це дозволяє йому:

- дати вам можливість писати добре форматований код;
- вгадувати що ви хочете написати і доповнювати код;
- відображати повідомлення про помилки і попередження;
- дати вам можливість переміщатися між класами, функціями і символами;
- надавати вам контекстно-залежну довідку по класах, функцій і символам;
- осмислено перейменовувати символи так, що інші символи з таким же ім'ям але належать іншим областям дії не будуть перейменовані;

- показувати вам місце в коді де функція була описана або викликана.

Ви можете використовувати Qt Designer щоб мати у своєму розпорядженні і налаштовувати ваші віджети або діалоги і тестувати їх використовуючи різні стилі і дозволу екрану. Створені за допомогою Qt Designer віджети і форми легко інтегруються в програмний код з використанням механізму сигналів і слотів Qt, які дозволять вам легко визначити поведінку графічних елементів. Всі властивості, встановлені в Qt Designer, можуть бути динамічно змінені в коді. Більш того, такі особливості як просування віджетів і власні модулі дозволять вам використовувати власні віджети з Qt Designer.

Ви можете використовувати редактор для написання коду на Qt C ++ або на мові декларативного програмування QML.

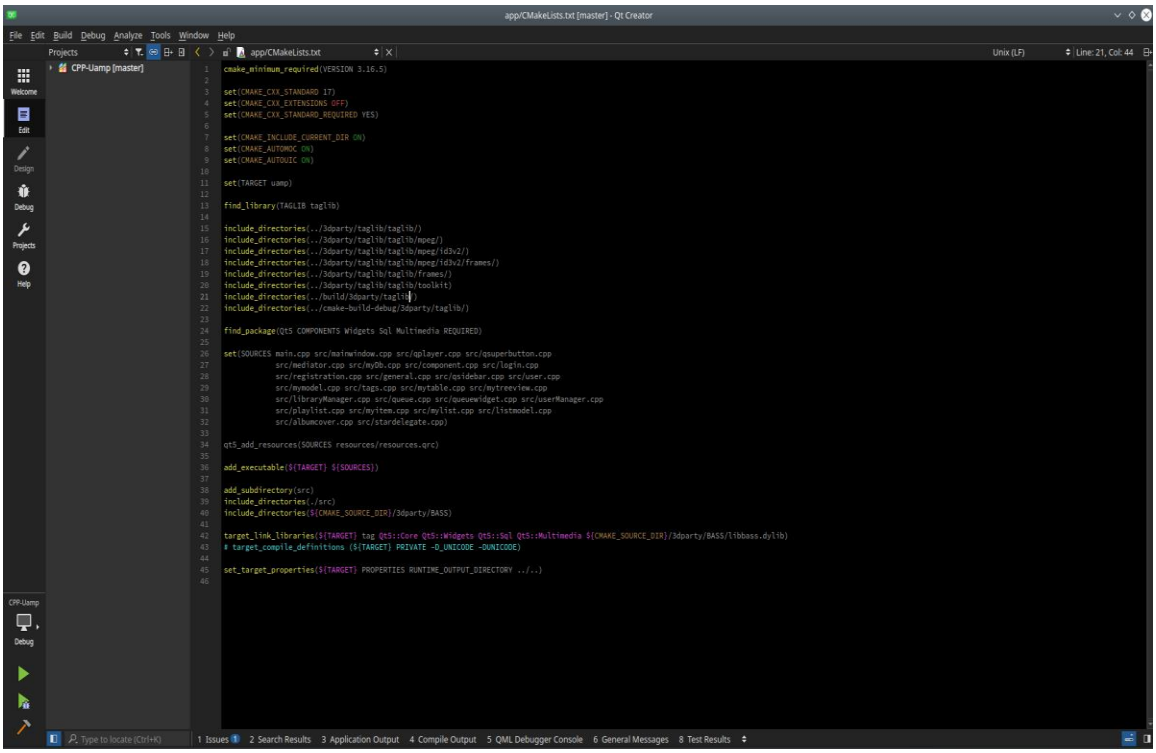



Рис. 2.3.1 – Інтерфейс QtCreator

ви можете використовувати QML для створення дуже гнучкого інтерфейсу користувача з великого набору елементів QML. QML допомагає розробникам і дизайнерам працювати разом над створенням гнучких призначених для

користувача інтерфейсів, які поширяться на портативних пристроях, таких як мобільні телефони, медіаплеєри, неттопи і нетбуки.

QML це розширення JavaScript, яке надає механізм декларативною збірки дерева об'єктів з елементів QML. QML покращує інтеграцію між JavaScript і існуючою системою Qt, заснованої на QObject, додає підтримку автоматичного зв'язування властивостей і забезпечує мережеву прозорість на рівні мови.

## 2.4.Інтерфейс додатку та його опис

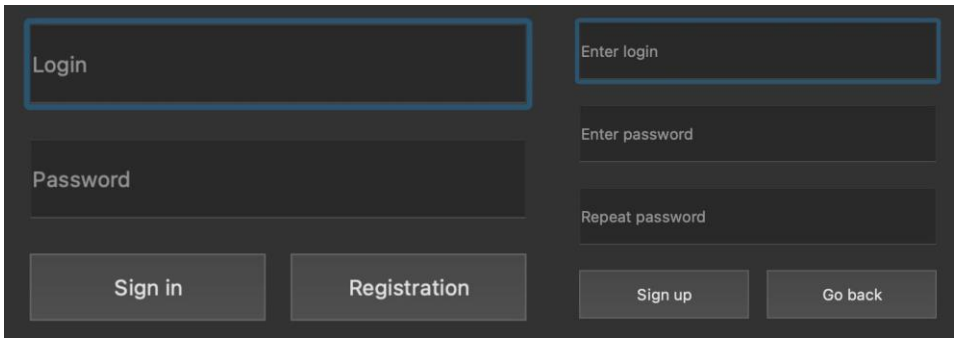



Рис. 2.4.1 – Экрани реєстрації та входу в аккаунт

Перед початком використання аудіоплеєру Uamp користувач має зареєструвати аккаунт який буде занесено в базу даних та в якому будуть зберігатися його плейлісти.

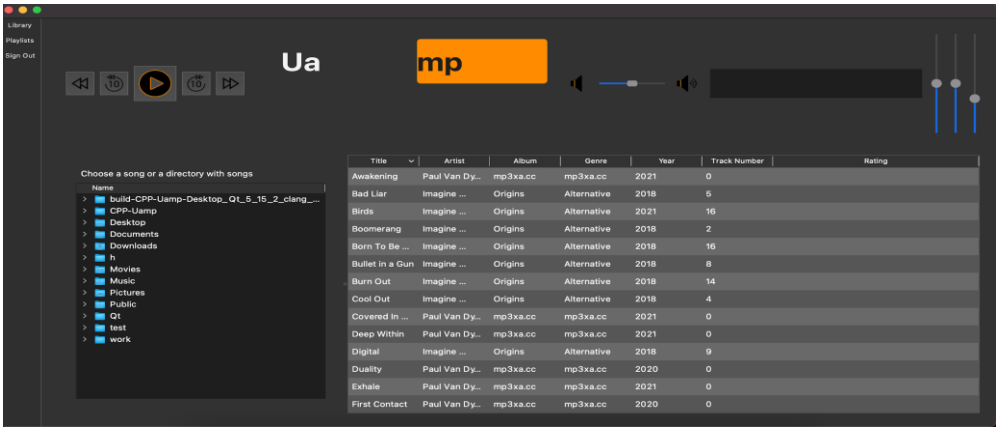



Рис. 2.4.2 – Основне меню

Для того щоб розпочати користування плеєром користувач має додати музику до бібліотеки.

Він може це зробити за допомогою натистенням по аудіофайлу всередині дерева вибору(Treeview) або ПКМ на папку та вибрати підходящий варіант.

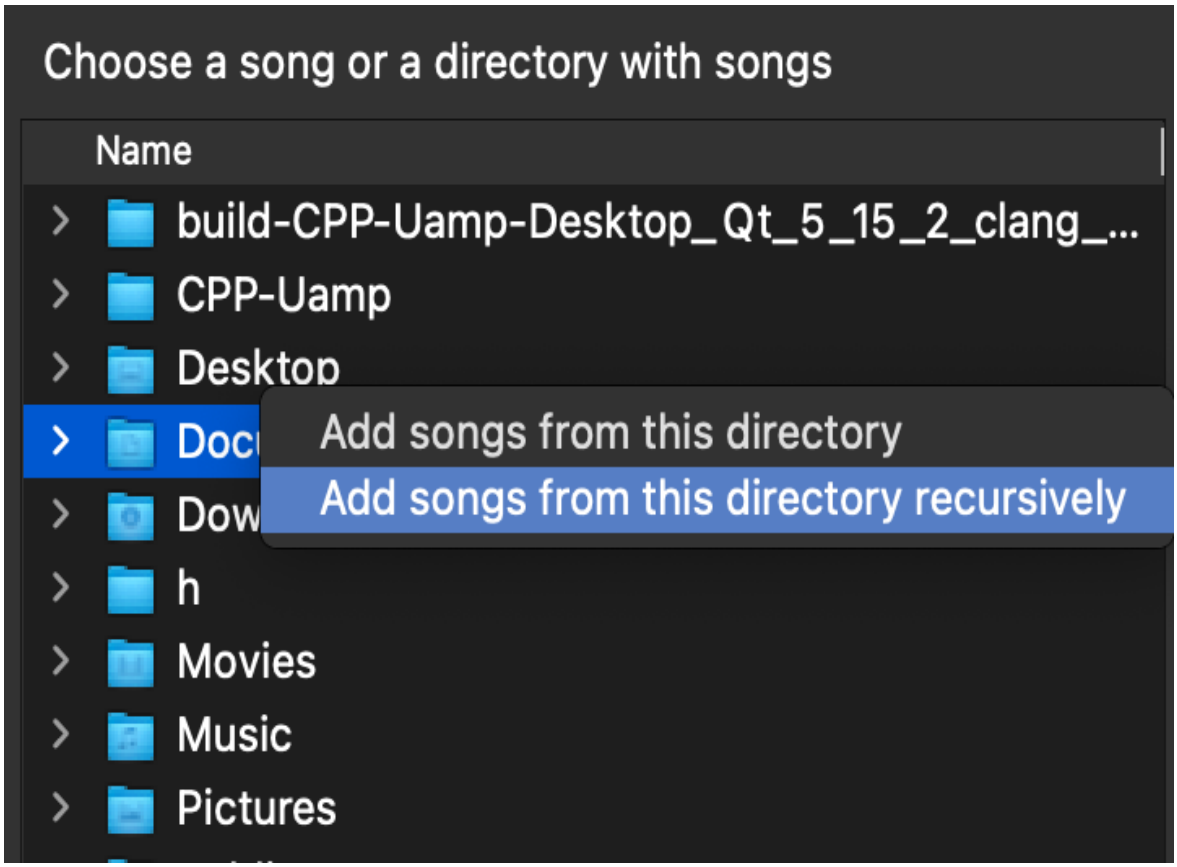



Рис. 2.4.3 – Один із способів додати аудіофайли до бібліотеки

## 2.5.Редагування Метаданих аудіофайлу

З допомогою бібліотеки Taglib користувач може редагувати метадані аудіофайлів( невеликий шматочок файлу в якому зберігається інформація про назву трека, альбом, ім'я виконавця).

Для користувача це виглядає так:

ПКМ по тегу-> “edit tag”-> та ввести данні.

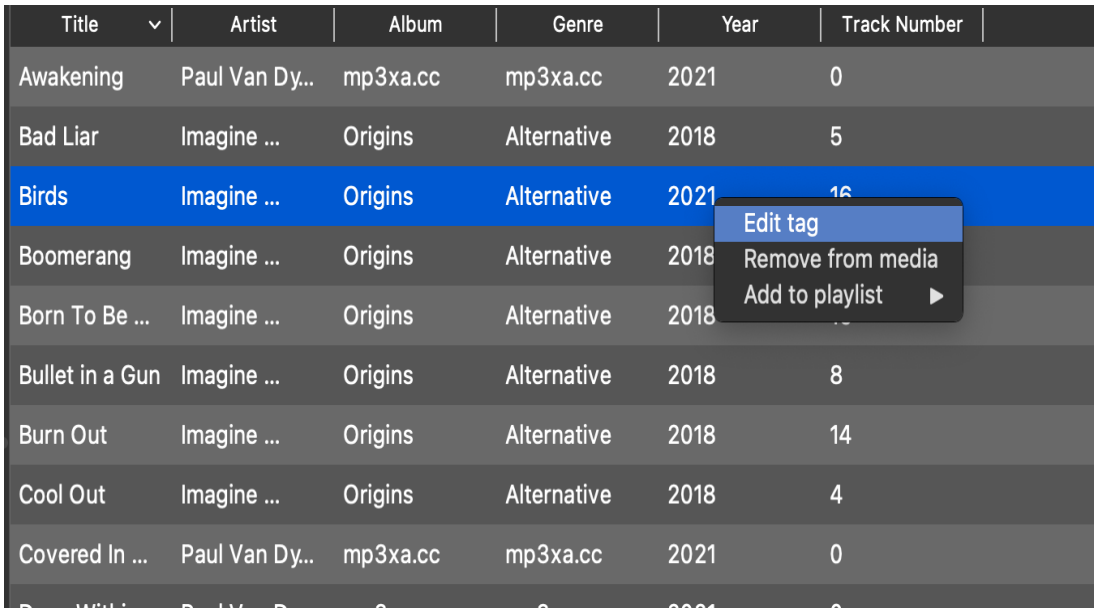



Рис. 2.5.1 – Змінення тегу

BASS audio library також дозволяє редагувати метаданні аудіофайлів але значно програє бібліотеці Taglib по деяким параметрам:

- TagLib працює швидко - тести показали, що вона приблизно в 6 разів швидше, ніж id3lib, і в 3 рази швидше, ніж BASS audio library, при читанні тегів ;
- TagLib добре документована - кожен клас, простір імен, функція і перерахування в TagLib задокументовані;
- TagLib підтримує Unicode - стандарти ID3v2 і Ogg Vorbis призначені для підтримки Unicode, як і TagLib;
- TagLib простий - TagLib пропонує рівень абстракції, який дозволяє легко ігнорувати відмінності між різними форматами файлів і їх реалізаціями.

## 2.6.Плейліст

Для зручності користувача в плеєрі є можливість оперувати плейлістами.

Зліва зверху розташоване меню для навігації по програмі завдяки якої можна перейти до плейлістів, які можна додати, видалити або зберегти його в форматі .m3u та в подальшому імпортувати його як плейліст до іншого користувача в цьому плеєрі чи в інший.

Також користувач може сортувати чергу відтворення музики якщо натисне на один із тегів.

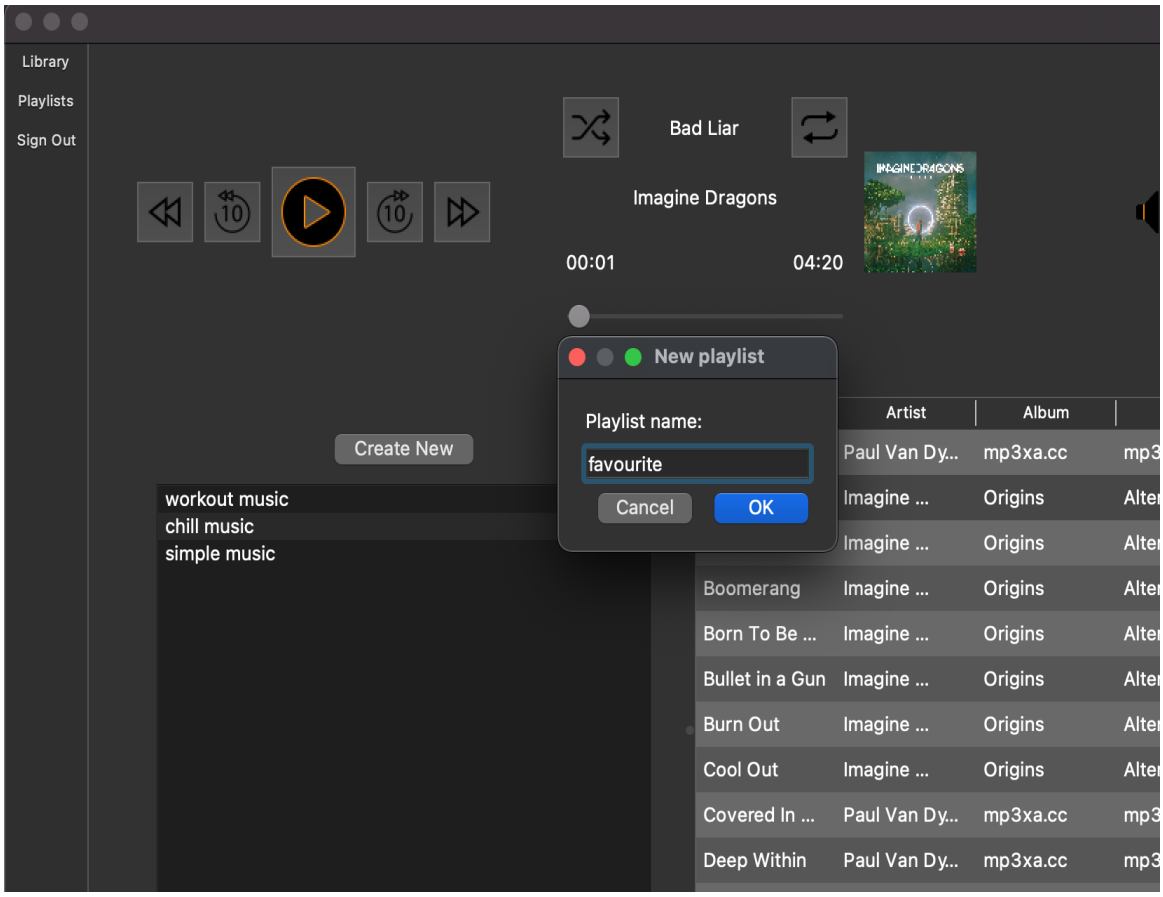



Рис. 2.6.1 – Створення нового плейлиста

Структура плейлісту має формат M3U та кодування UTF-8 і містить шляхи до одного або кількох мультимедійних файлів або стрімінгових потоках, а також додаткові відомості, такі як: найменування для відображення, сортування, графічні елементи та інше.

У першому рядку M3U файлу обов'язкова вказівка директиви #EXTM3U - це підзаголовок, який робить формат списку відтворення зрозумілим для

системи. За заголовком слід рядок #EXTINF, що містить відомості про медіа-файлі.

Крім тривалості і заголовка для кожного запису, в M3U плейлистах є ряд додаткових параметрів:

- «Tvg-name» - вказівка відображуваного найменування;
- «Tvg-logo» - логотип каналу;
- «Audio-track» - аудіодоріжка каналу (мови);
- «Group-title2»: найменування групи (спортивний, освітній і тому подібні).

```
find_library(TAGLIB taglib)

include_directories(../3dparty/taglib/taglib/)
include_directories(../3dparty/taglib/taglib/mpeg/)
include_directories(../3dparty/taglib/taglib/mpeg/id3v2/)
include_directories(../3dparty/taglib/taglib/mpeg/id3v2/frames/)
include_directories(../3dparty/taglib/taglib/frames/)
include_directories(../3dparty/taglib/taglib/toolkit)
include_directories(../build/3dparty/taglib/)
include_directories(../cmake-build-debug/3dparty/taglib/)
```

Рис. 2.6.2 – підключення бібліотеки Taglib

## 2.7.Програвання музики та еквалайзер


Рис. 2.7.1 – Інтерфейс програвача

На рис 2.7.1 показано складові інтерфейсу програвача

- переключення поточної пісні на попередню;
- відмотати 10 секунд назад;
- включити пісню / пауза;
- відмотати 10 секунд вперед;
- переключення поточної пісні на наступну;
- відтворення музики згідно з порядком плейліста / відтворення упереміш;
- назва пісні;
- відтворення плейлісту до кінця / повторення однієї пісні / повторення плейлісту( після закінчення останньої пісні у черзі переключення на першу пісню);
- назва групи;
- слайдер з хронометражем пісні, якщо його перетягнути буде змінений поточний хронометраж;
- малюнок пісні, якщо він є;
- слайдер гучності пісні;
- візуалізація частот поточної пісні;
- еквалайзер який складається з 3 вертикальних слайдерів і дає можливість корекції спектральних властивостей акустичного сигналу.

Музика відтворюється за допомогою бібліотека BASS - це аудіо-бібліотека, яка надає безліч функцій:

- декодування без відтворення. Музичні потоки і MOD можна виводити будь-яким способом: кодувати, записувати на диск, передавати по мережі і т. Д;
- потоки файлів. MP3 / MP2 / MP1 / OGG / WAV / AIFF потокова передача файлів;

- підтримка 64-бітної архітектури Підтримуються як 32-бітна, так і 64-бітна архітектури;
- багатоканальна потокова передача. Підтримка більш ніж простого стерео, включаючи багатоканальні файли OGG / WAV / AIFF;
- 3D-звук;
- відтворення семплів / потоків / музики в будь-якому 3D-положенні.

## 2.8.Система сигналів і слотів

Уся взаємодія з користувачем була реалізована за допомогою системою сигналів і слотів, яку підтримує Qt.

Сигнал виконується тоді, коли відбувається певна подія. Віджети Qt мають безліч зумовлених сигналів. Слотом є функція, яка викликається у відповідь на певний сигнал. Віджети Qt також мають безліч зумовлених слотів, але успадкування від віджетів і додавання власних слотів є звичайною практикою,

Сигнали випускаються об'єктом, коли його внутрішній стан змінилося в певному напрямку, яке може бути цікаво іншим об'єктам. Сигнали є публічно доступними функціями і можуть бути викликані будь-де, але рекомендується їх викликати тільки в класі, де вони були визначені, а також в його підкласах.

Коли сигнал викликаний, слот підключений до нього зазвичай виконується негайно, просто як нормальна функція. Це можливо тому, що механізм сигналів та слотів є незалежним від будь-яких циклів в GUI. Виконання коду слід викликати директивою emit, яка викличе всі слоти. У тих ситуаціях, коли використовуються черзі підключень, код буде запускати сигнал, а слоти будуть виконані дещо пізніше.

Слот викликається тоді, коли сигнал підключений до нього був викликаний. Слоти є нормальною C ++ функцією і може бути викликана; вони особливі тільки тим, що до них підключаються сигнали.

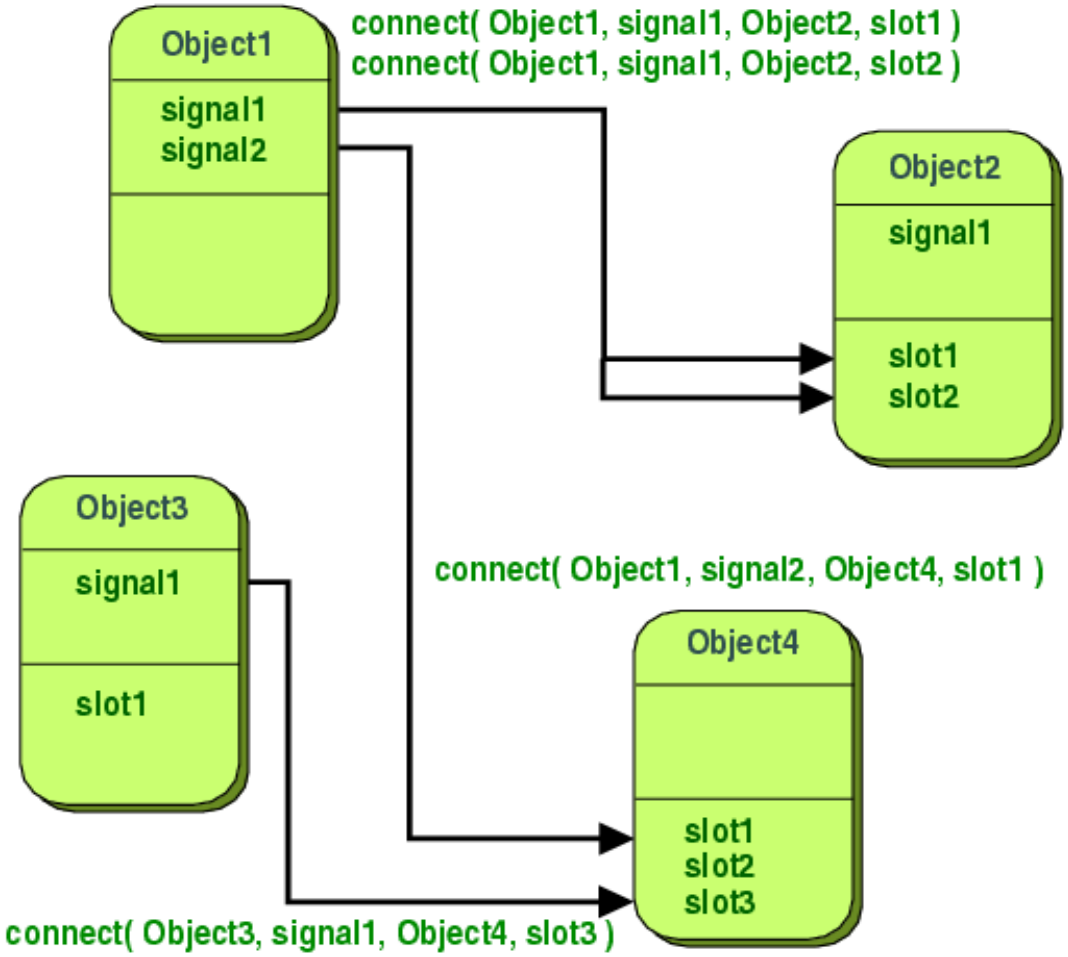



Рис. 2.8.1. – Схематичний приклад Сигналів с слотів

Також слоти можуть виконуватися як звичайні функції, вони підкоряються звичайними правилами C ++, коли викликаються безпосередньо. Однак, як слоти, вони можуть бути викликані іншими компонентами, незважаючи на їх рівень доступу, через сигнал-слотове підключення. Це означає, що сигнал випускається з одного з класів і може бути переданий в приватний слот, який буде викликаний з цього незв'язаного класу.

```
connect(actionLibrary, SIGNAL(triggered()), mediator, SLOT(backToLibrary()));
connect(actionPlaylists, SIGNAL(triggered()), mediator, SLOT(backToPlaylists()));
QObject::connect(actionSignOut, SIGNAL(triggered()), mediator, SLOT(backToSignIn()));
```

Рис. 2.8.2 – Приклад використання сигналів і слотів

## 2.9.Система збирання cmake

CMake - кроссплатформенна автоматизована система збирання проектів. Безпосередньо складанням вона не займається, а тільки генерує Makefile, який потім буде виконаний утилітою make.

CMake може перевіряти наявність необхідних бібліотек і підключати їх, збирати проекти під різними компіляторами і операційними системами.

Мови C і C ++ влаштовані таким чином, що щоб скористатися будь-якої сторонньої бібліотекою, вам необхідно вказати компілятору шлях до

- директорії, що містить заголовки бібліотеки;
- директорії, що містить саму бібліотеку.

Заголовки вказуються прапором -I в Unix-подібних ОС і прапором / I в компіляторі Microsoft:

Аналогічно, директорії для бібліотек вказуються за допомогою -L / / LIBPATH.

Зараз багато розробників-початківців відмовляються від Cmake і вважають його застарілим та користуються вбудованими в IDE інструментами, бо вони дозволяють просто створити декілька файлі, написати програму та

Чи є Cmake застарілим? Відповідь - звичайно ні. Підтвердження цьому можна знайти не тільки в тому, що системи збирання постійно і активно розвиваються, але і в тому що навіть в Visual Studio, тієї самої IDE, якій не була потрібна ніяка система збирання, була додана підтримка CMake.

Пара слів про Visual Studio. Ця IDE мала свій формат проектів, і, по суті, свою власну систему збирання. Проекти студії в функціональному сенсі були еквівалентні Makefile або CMakeLists.txt . Незручність користування ними, на мій погляд, полягає в наступному:

- відсутня етап конфігурації. На сьогоднішній день доволі важко вказати студії , що йому потрібна стороння бібліотека, та так щоб при спробі

зібрати проект, студія б запитала мене про місцезнаходження цієї бібліотеки, замість того щоб видавати невалідні та нечитаємі помилки;

```
cmake_minimum_required(VERSION 3.16.5)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_EXTENSIONS OFF)
set(CMAKE_CXX_STANDARD_REQUIRED YES)

set(CMAKE_INCLUDE_CURRENT_DIR ON)
set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTOUIC ON)

set(TARGET vamp)

find_library(TAGLIB taglib)

include_directories(../3dparty/taglib/taglib/)
include_directories(../3dparty/taglib/taglib/mpeg/)
include_directories(../3dparty/taglib/taglib/mpeg/id3v2/)
include_directories(../3dparty/taglib/taglib/mpeg/id3v2/frames/)
include_directories(../3dparty/taglib/taglib/frames/)
include_directories(../3dparty/taglib/taglib/toolkit)
include_directories(../build/3dparty/taglib/)
include_directories(../cmake-build-debug/3dparty/taglib/)

find_package(Qt5 COMPONENTS Widgets Sql Multimedia REQUIRED)

set(SOURCES main.cpp src/mainwindow.cpp src/qplayer.cpp src/qsuperbutton.cpp
            src/mediator.cpp src/myDb.cpp src/component.cpp src/login.cpp
            src/registration.cpp src/general.cpp src/qsidebar.cpp src/user.cpp
            src/mymodel.cpp src/tags.cpp src/mytable.cpp src/mytreeview.cpp
            src/libraryManager.cpp src/queue.cpp src/queuewidget.cpp src/userManager.cpp
            src/playlist.cpp src/myitem.cpp src/mylist.cpp src/listmodel.cpp
            src/albumcover.cpp src/stardelegate.cpp)

qt5_add_resources(SOURCES resources/resources.qrc)

add_executable(${TARGET} ${SOURCES})

add_subdirectory(src)
include_directories(./src)
include_directories(${CMAKE_SOURCE_DIR}/3dparty/BASS)
```

Рис. 2.9.1 – cmake файл для підключення сторонніх бібліотек та файлів проекту

- складно використовувати сторонні утиліти в процесі складання. Наприклад, у мене є проект, який спочатку збирає програму на Хаскелл, запускає її, а вона виробляє код на C ++, який вже перетворюється в кінцеву програму. На CMake це доволі тривіальна задача, а використовуючи стандартні інструменти студії це скоріш за все просто неможливо.

# РОЗДІЛ 3. РЕАЛІЗАЦІЯ ДОДАТКУ,МЕТОДІВ ТА КЛАСІВ

## 3.1.Основні дані про розробку додатків ,методів і класів

Як кожна программа Uamp починається з main функції

```
int main(int argc, char *argv[])
{
    BASS_Init ( device: -1,  freq: 44100, BASS_DEVICE_3D ,  win: 0,  dsguid: NULL);

    QApplication app( &: argc, argv);
    QFile File( name: ":/qmain.qss");
    QString StyleSheet;
    Mediator *mediator = new Mediator;
    int res = 0;

    File.open(QFile::ReadOnly);
    StyleSheet = QLatin1String( s: File.readAll());
    app.setStyleSheet(StyleSheet);
    res = app.exec();
    BASS_Free();

    return res;
}
```

Рис. 3.1.1 – Main.cpp

BASS_Init ініціалізує звуковий потік з яким в майбутньому буде працювати программа також ми зчитуємо таблицю стилів для віджетів Qt та ініціалізуємо її.

## 3.2.Розміщення віджетів на екрані

Qt дозволяє користуватися вбудованим інструментрієм для створення та розміщення віджетів Qt Designer або робити це вручну.

Qt Designer - це інструмент Qt для проектування і створення графічних користувацьких інтерфейсів (GUI) з віджетами Qt. Ви можете складати і налаштовувати свої вікна або діалогові вікна в режимі «що бачиш, те й отримуєш» (WYSIWYG) і тестувати їх, використовуючи різні стилі і дозволу.

Віджети та форми, створені за допомогою Qt Designer, легко інтегруються з програмним кодом, використовуючи механізм сигналів та слотів Qt, так що ви можете легко призначати поведінку графічним елементам. Всі властивості, встановлені в Qt Designer, можна динамічно змінювати в коді. Крім того, такі функції, як просування віджетів і настроюються плагіни, дозволяють вам використовувати ваші власні компоненти з Qt Designer.

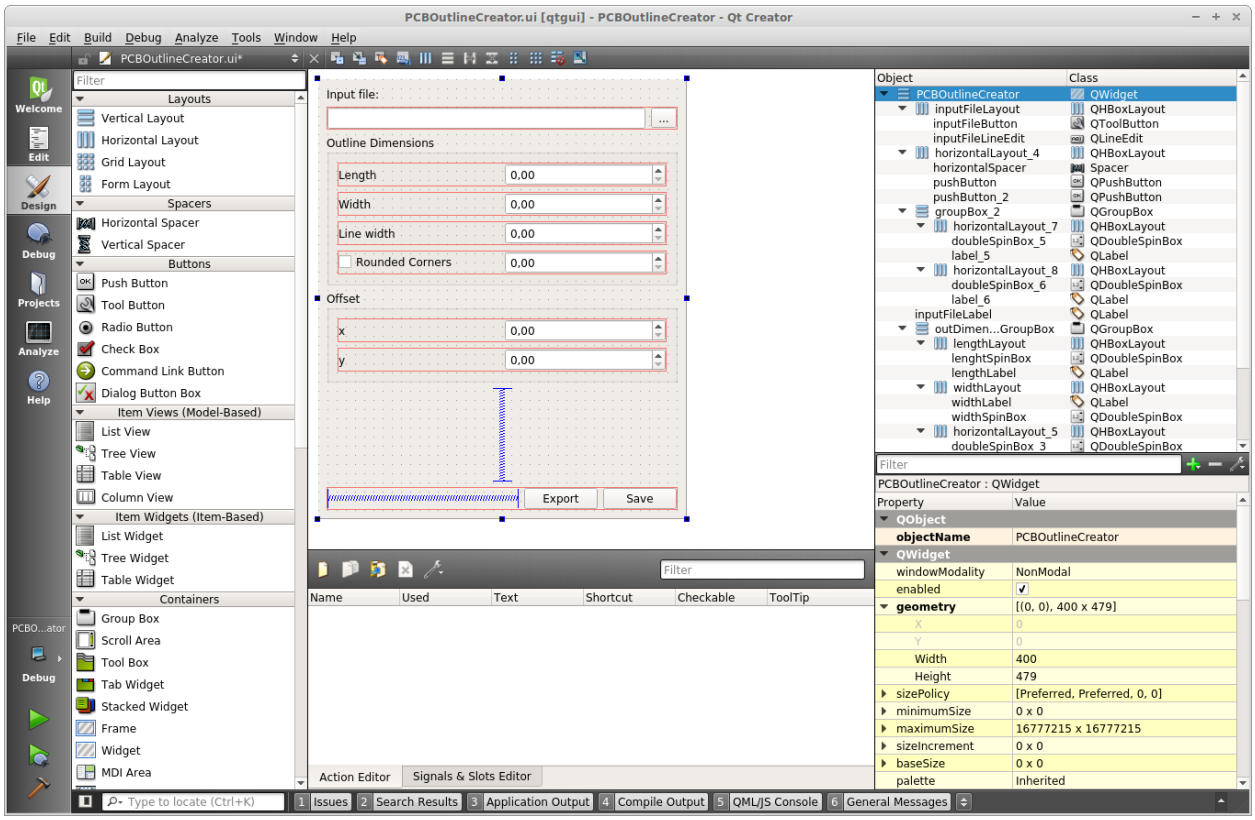



Рис. 3.2.1 – Інтерфейс Qt Designer

Як я вже зазначив Qt дозволяє розробнику розміщувати елементи вручну, це зменшує швидкість розробки але збільшує гнучкість і варіативність, оскільки це дозволяє власноруч вводити всі параметри для віджетів та створювати власні шаблонні віджети.

Також є можливість використовувати Qt Style Sheet термінологія та синтаксичні правила QSS майже ідентичні правилам HTML CSS, що дозволяє використовувати її навіть починаючим розробникам.

```
LoginScreen::LoginScreen(const Mediator *mediator_, QWidget *parent) :
                    QWidget(parent), Component(mediator_)
{
    layoutOuter = new QGridLayout( parent: this);

    loginField = new QLineEdit( parent: this);
    loginField->setPlaceholderText("Login");
    loginField->setMinimumWidth( minw: 300);
    passwordField = new QLineEdit( parent: this);
    passwordField->setPlaceholderText("Password");
    passwordField->setEchoMode(QLineEdit::Password);
    signInButton = new QPushButton( text: "Sign in",  parent: this);
    registrationButton = new QPushButton( text: "Registration",  parent: this);


    layoutOuter->addWidget(loginField,  row: 1,  column: 1,  rowSpan: 1,  columnSpan: 2);
    layoutOuter->addWidget(passwordField,  row: 2,  column: 1,  rowSpan: 1,  columnSpan: 2);
    layoutOuter->addWidget(signInButton,  row: 3,  column: 1);
    layoutOuter->addWidget(registrationButton,  row: 3,  column: 2);

    layoutOuter->setColumnStretch( column: 0,  stretch: 2);
    layoutOuter->setColumnStretch( column: layoutOuter->columnCount(),  stretch: 2);
    layoutOuter->setRowStretch( row: 0,  stretch: 2);
    layoutOuter->setRowStretch( row: layoutOuter->rowCount(),  stretch: 2);

    layoutOuter->setVerticalSpacing( spacing: 20);

    loginField->setMinimumHeight( minh: 50);
    passwordField->setMinimumHeight( minh: 50);
    signInButton->setMinimumHeight( minh: 50);
    registrationButton->setMinimumHeight( minh: 50);

    connect(signInButton, SIGNAL( arg: clicked()),  receiver: reinterpret_cast<const QObject *> (mediator), SLOT( arg: signInTry()));
    connect(registrationButton, SIGNAL( arg: clicked()),  receiver: reinterpret_cast<const QObject *> (mediator), SLOT( arg: registrationOpen()));
}
```

Рис. 3.2.2 – Створення та розміщення віджетів у вікні логіну

Найлегший спосіб завдання правильного розташування віджетів полягає у використанні вбудованих менеджерів компонування: QHBoxLayout, QVBoxLayout, QGridLayout і QFormLayout. Ці класи успадковані від QLayout, який, в свою чергу, походить від QObject (а не від QWidget). Вони беруть на себе турботи по управлінню геометрією безлічі віджетів. Для створення більш складних компоновок ви можете поміщати менеджери компоновок один в одного.

QHBoxLayout розташовує  віджети в горизонтальну лінію з напрямком розміщення зліва направо.QVBoxLayout розташовує  віджети в вертикальну лінію з напрямком зверху вниз. QGridLayout розташовує  віджети в двовимірній сітці. Віджети можуть займати кілька комірок.

### 3.3.Шаблон проектування

Для розробки додатку був використаний шаблон проектування посередник(Mediator).

Основні параметри шаблону посередник:

- патерн Mediator визначає об'єкт, що інкапсулює взаємодію безлічі об'єктів. Mediator робить систему слабо пов'язаної, позбавляючи об'єкти від необхідності посилатися один на одного, що дозволяє змінювати взаємодію між ними незалежно;
- патерн Mediator вводить посередника для розв'язання безлічі взаємодіючих об'єктів;
- замінює взаємодія "все з усіма" взаємодією "один з усіма".

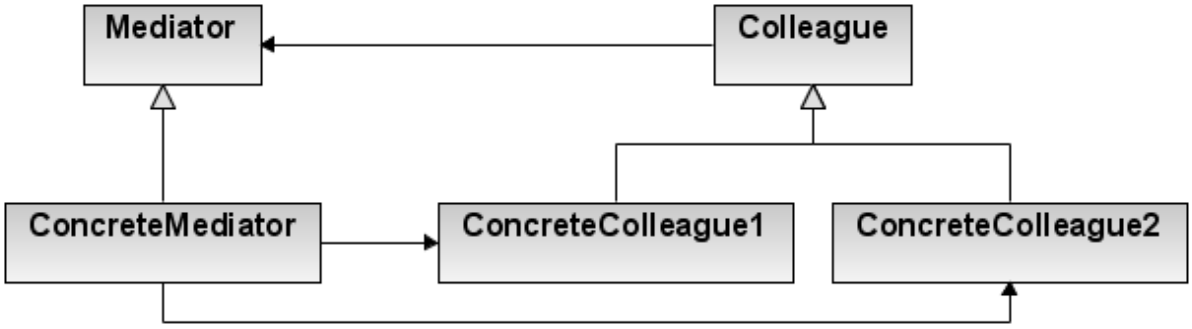



Рис. 3.3.1 – Структура шаблону проектування Посередник

Mediator – посередник:

- визначає інтерфейс для обміну інформацією з об'єктами *Colleague*;
    - ConcreteMediator – конкретний посередник;
- реалізує кооперативну поведінку, координуючи дії об'єктів *Colleague*;
- володіє інформацією про колег, та підраховує їх;
    - Класи Colleague – колеги:
- кожному класу *Colleague* відомо про свій об'єкт *Mediator*;

- усі колеги обмінюються інформацією виключно через посередника, інакше за його відсутності їм довелося б спілкуватися між собою напряму.

Особливості паттерну посередник:

- Патерн Mediator показує, як можна розділити відправників і одержувачів запитів з урахуванням їх особливостей. У Mediator відправник і одержувач посилаються один на одного побічно, через об'єкт-посередник;
- Mediator використовує об'єкт-посередник для інкапсуляції взаємодії між іншими об'єктами;
- Mediator відволікає функціональність існуючих класів. Mediator абстрагує / централізує взаємодія між об'єктами-колегами, додає нову функціональність і відомий всім об'єктам-колегам (тобто визначає двонаправлений протокол взаємодії).

### 3.4.Клас Mediator

Об’єкт класу Mediator це один з головних класів який виконує 2 основні ролі які ми зараз розберемо.

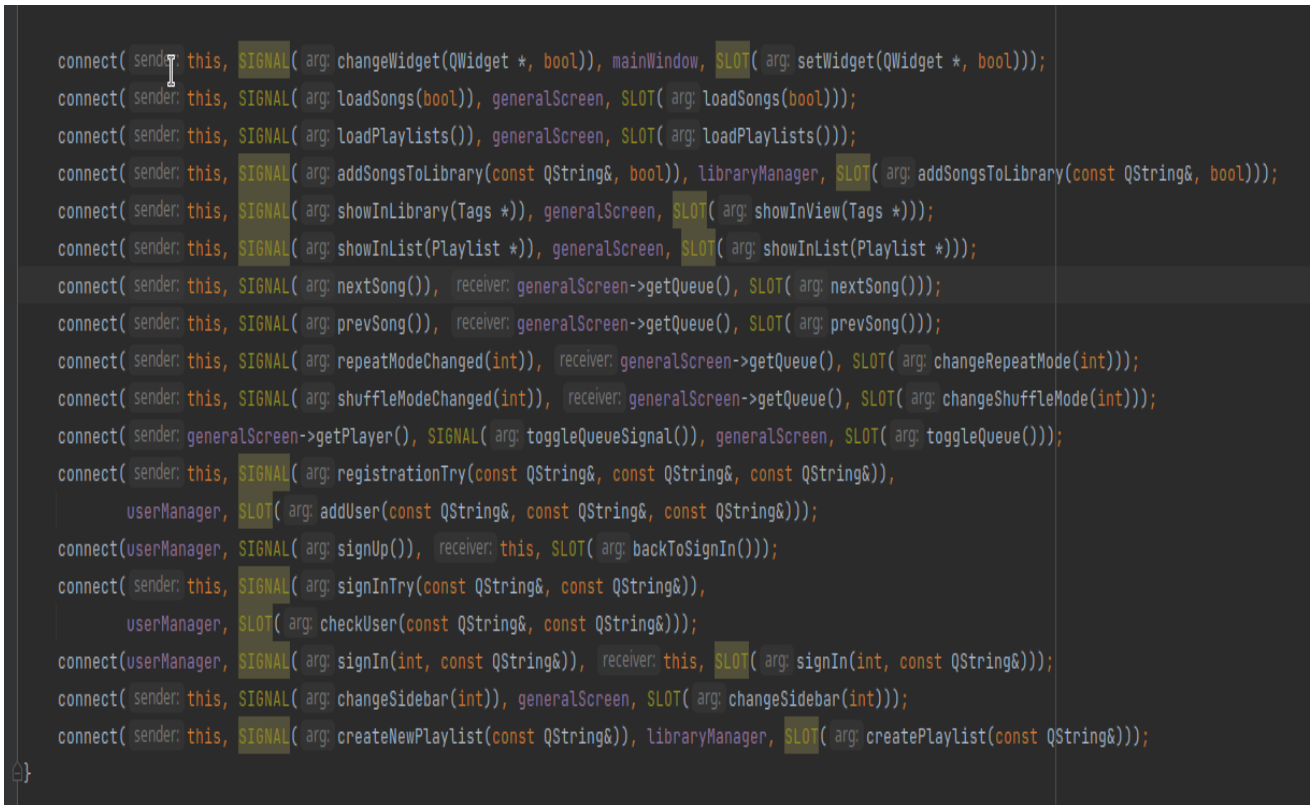
```
Mediator::Mediator() : QObject() {
    db = new MyDb( mediator_: this);
    libraryManager = new LibraryManager( mediator: this);
    userManager = new UserManager( mediator: this);

    mainWindow = new MainWindow( mediator: this);
    loginScreen = new LoginScreen( mediator_: this);
    mainWindow->layoutOuter->addWidget(loginScreen);
    registrationScreen = new RegistrationScreen( mediator_: this);
    mainWindow->layoutOuter->addWidget(registrationScreen);
    generalScreen = new GeneralScreen( mediator: this);
    mainWindow->layoutOuter->addWidget(generalScreen);

    mainWindow->layoutOuter->setCurrentWidget(loginScreen);
    mainWindow->show();
```

Рис. 3.4.1 – Конструктор класа Mediator 1

Перша роль яку виконує клас Mediator це створення об’єктів усіх основних класів з якими може взаємодіяти користувач.



```
    connect( sender: this, SIGNAL( arg: changeWidget(QWidget *, bool)), mainWindow, SLOT( arg: setWidget(QWidget *, bool)));
    connect( sender: this, SIGNAL( arg: loadSongs(bool)), generalScreen, SLOT( arg: loadSongs(bool)));
    connect( sender: this, SIGNAL( arg: loadPlaylists()), generalScreen, SLOT( arg: loadPlaylists()));
    connect( sender: this, SIGNAL( arg: addSongsToLibrary(const QString&, bool)), libraryManager, SLOT( arg: addSongsToLibrary(const QString&, bool)));
    connect( sender: this, SIGNAL( arg: showInLibrary(Tags *)), generalScreen, SLOT( arg: showInView(Tags *)));
    connect( sender: this, SIGNAL( arg: showInList(Playlist *)), generalScreen, SLOT( arg: showInList(Playlist *)));
    connect( sender: this, SIGNAL( arg: nextSong()),  receiver: generalScreen->getQueue(), SLOT( arg: nextSong()));
    connect( sender: this, SIGNAL( arg: prevSong()),  receiver: generalScreen->getQueue(), SLOT( arg: prevSong()));
    connect( sender: this, SIGNAL( arg: repeatModeChanged(int)),  receiver: generalScreen->getQueue(), SLOT( arg: changeRepeatMode(int)));
    connect( sender: this, SIGNAL( arg: shuffleModeChanged(int)),  receiver: generalScreen->getQueue(), SLOT( arg: changeShuffleMode(int)));
    connect( sender: generalScreen->getPlayer(), SIGNAL( arg: toggleQueueSignal()), generalScreen, SLOT( arg: toggleQueue()));
    connect( sender: this, SIGNAL( arg: registrationTry(const QString&, const QString&, const QString&)),
            userManager, SLOT( arg: addUser(const QString&, const QString&, const QString&)));
    connect(userManager, SIGNAL( arg: signUp()),  receiver: this, SLOT( arg: backToSignIn()));
    connect( sender: this, SIGNAL( arg: signInTry(const QString&, const QString&)),
            userManager, SLOT( arg: checkUser(const QString&, const QString&)));
    connect(userManager, SIGNAL( arg: signIn(int, const QString&)),  receiver: this, SLOT( arg: signIn(int, const QString&)));
    connect( sender: this, SIGNAL( arg: changeSidebar(int)), generalScreen, SLOT( arg: changeSidebar(int)));
    connect( sender: this, SIGNAL( arg: createNewPlaylist(const QString&)), libraryManager, SLOT( arg: createPlaylist(const QString&)));
}
```

Рис. 3.4.2 – конструктор класа Mediator 2

Друга роль яку виконує клас  Mediator це створення сигналів для класів об’єкти яких він створив та підтримання взаємодії між користувачем і програмою та передачею даних між цими класами.

### 3.5.Клас  MyTreeView

Як ми можемо побачити з рисунку клас MyTreeView є нащадком класу QTreeView давайте розглянемо його більш детально.

QTreeView  реалізує деревоподібне уявлення елементів моделі. Цей клас використовується для надання стандартних ієрархічних списків, що забезпечується архітектурою модель / подання Qt.

Архітектура модель / подання гарантує, що вміст уявлення у вигляді дерева оновлюється по мірі зміни моделі.

QTreeView реалізує інтерфейси, визначені класом QAbstractItemView, щоб дозволити йому відображати дані, надані моделями, похідними від класу QAbstractItemModel.

Елементи, у яких є дочірні елементи, можуть бути в розгорнутому (дочірні елементи видно) або згорнутому (дочірні елементи приховані) стані. Коли цей стан змінюється, випромінюється сигнал collapsed () або extended () з модельним індексом відповідного елемента.

```
class MyTreeView : public QTreeView, public Component {
    Q_OBJECT

    QFileSystemModel *model_filesystem;
    QMenu *context_menu;
    QAction *action_import;
    QAction *action_import_recursive;

public:
    MyTreeView(const Mediator *mediator);
    ~MyTreeView();

signals:
    void songImported(const QString& path, bool recursive);

public slots:
    void mouseDoubleClickEvent(QMouseEvent *event);
    void mousePressEvent(QMouseEvent *event);
    void importSong();
    void importSongRecursive();
};
```

Рис. 3.5.1 – Оголошення класу MyTreeView у файлі-заголовку

Тепер коли ми розібралися як він працює можемо перейти до нашого класу MyTreeView він використвується для відображення дерева каталогів та можливості користувача додати музику до бібліотеки, як по одному файлу так і папками або рекурсивно.

### 3.6.Клас MyModel

Як ми можемо побачити з рисунку клас MyModel є нащадком класу QAbstractTableModel давайте розглянемо його більш детально.

QAbstractTableModel надає стандартний інтерфейс для моделей, які представляють свої дані як двовимірний масив елементів. Він не використовується безпосередньо, але повинен бути розділений на підкласи. Оскільки модель надає більш спеціалізований інтерфейс, ніж QAbstractItemModel, вона не підходить для використання з деревоподібними уявленнями, хоча її можна використовувати для надання даних QListView.

MyModel клас відображає інформацію про трек (його назву, виконавця, альбом, жанр, рік і тд.) який знаходиться в бібліотеці або плейлісті але усі ці об’єкти важливо десь зберігати щоб валідно їх відображати бо їх не статична кількість, бо користувач може як додавати нові треки до бібліотеки / плейлісту так і видаляти старі щоб у цьому розібратися пропоную подивитися на наступний клас.

```
    QVariant headerData(int nSection,
                        Qt::Orientation orientation,
                        int nRole
                        ) const;

    Qt::ItemFlags flags(const QModelIndex& index) const;

    void setNewData(std::deque<Tags *>&& data);

    void addData(Tags *tags);

    void sort(int column, Qt::SortOrder order = Qt::AscendingOrder);

    std::deque<Tags *> getData(void) const;

    void remove(const QModelIndex& index);

signals:
    void insertToQueue(Tags *song);
    void sortQueue(std::deque<Tags *> data, Qt::SortOrder order, int tag);
    void removeFromQueue(Tags *song);
};
```

Рис 3.6.2 – Оголошення класу MyModel у файлі-заголовку (2)

### 3.7.Клас MyTable

```
class MyTable : public QTableView, public Component
{
    Q_OBJECT
    MyModel *model = new MyModel( data: std::deque<Tags *>());
    QMenu *mainMenu;
    QMenu *playlistMenu;
    QAction *editAction;
    QAction *removeAction;
    std::vector<QAction *> playlistActions;

public:
    explicit MyTable(Mediator *mediator, QWidget *parent = nullptr);
    ~MyTable();

    MyModel *getModel() {
        return model;
    }

signals:
    void sendSongToPlayer(Tags *tags);
    void updateQueue(Tags *tags);
    void removeSong(int id);

public slots:
    void sendNextSong(const QModelIndex &index);
    void showContextMenuRequested(const QPoint &pos);
    void removeSong();
};
```

Рис. 3.7.1 – Оголошення класу MyTable у файлі заголовку

Клас MyTable зберігає у собі деку об’єктів класу MyModel та слоти для маніпулювання над цими об’єктами ( видалення, додавання в плейліст чи редагування метаданих).

# ВИСНОВКИ

У даній роботі було створено додаток для прослуховування аудіо файлів. В першому розділі було досліджено функіонал додатків-аналогів та принцип їх роботи. Розібрали мережні технології для створення та обробки аудіо контенту.

У другому розділі обрано та застосовано інструментарій для написання програмного забезпечення, перелічено основні бібліотеки, та обґрунтуван їх вибір, перелічено їх сильні та слабкі сторони. Показан інтерфейс додатку та можливості які він надає користувачу.

У третьому розділі застосовано необхідний інструментарій та знання на практиці для побудови моделі. Розібрані основні класи програми та принцип їх роботи. Описан вибраний для додатку шаблон проектування.

Редагування метаданих доволі важлива складова аудіо плеєру, яку розробники часто не включають в свої додатки, але в наш час метадані можуть бути заспамлені рекламою і не відображають дійсну інформацію яку хоче побачити користувач, тому важливо дати йому можливість редагувати їх.

Використаний шаблон проектування Медіатор ідеально підходить для цього проекту оскільки він з легкістю дозволяє обмінюватися інформацією між собою багатьом класам одразу.

Додаток підтримує більшу частину сучасного функціоналу аудіо плеєра такі як:

- Підтримка плейлистів
- Різні режими програвання музики
- Різні режими перемотки музики
- Редагування частот аудіо файлів та їх візуалізація
- Редагування метаданих

Але додаток потребує значних візуальних змін, та підтримку інших систем таких як Windows та Android.

Підсумовуючи все вище написане, в ході роботи я створив додаток для прослуховування аудіо файлів та розібрав інструментарій який використав, і на основі цих даних інший розробник може створити свою версію додатка з тим функціоналом який він буде вважати потрібним.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Некрашевич І.Р. Аналіз програмного забезпечення для обробки та мастерінгу аудіо. XI міжнародна наукова-технічна конференція молодих вчених «Електроніка-2018», 3-5 квітня 2018 року.
2. Некрашевич І.Р. Загальні концепції та використання WEB AUDIO API. Науково-технічна конференція студентів, аспірантів та науковців кафедри ЗТРІ, 25 травня 2018 року
3. Толубко В. Б., Беркман Л. Н. Методи оптимізації: підручник для вищих навчальнихзакладів за напрямом «Телекомунікації» / - Київ: ДУТ, 2016. – 442 с.
4. Стеклов В. К., Беркман Л. Н. Проектування телекомунікаційних мереж: підручник для ВНЗ / - Київ: Техніка, 2002. – 792 с.
5. Варфоломеєва О. Г. Методика розрахунку показників ефективності системи управління мережами телекомунікацій із застосуванням методу експертних оцінок // Зв'язок. - 2005. - №7(59). - С. 22-25.
6. Бондарчук А. П., Твердохліб М. Г. Покращення оптимального проектування мережі FGN для трьох показників якості // Цифрові технології. - 2010. - Випуск 8. - С. 125-127.
7. Женченко М. Загальна і спеціальна бібліографія: навч. посіб. /Марина Женченко. — Київ: Жнець, 2011. — 255 с.
8. Комп'ютерний моніторинг і інформаційні технології [Електронний ресурс] : матеріали студент. наук.-практ. конф., 25 квіт. 2005, Донецьк / Донец. нац. техн. ун-т, Каф. комп'ютер. систем моніторингу. – Текст. і граф. дані (250 МБ). – Донецьк, 2005. – 1 електрон. опт. диск (CD-ROM). – Назва з етикетки диска. слово «електронні» в інформації про вид ресурсу «Електрон. текст. дані» дозволяється опускати, якщо в описі є загальне позначення матеріалу – [Електронний ресурс]

**ДОДАТОК**

# ДИПЛОМНА РОБОТА

# «РОЗРОБКА ДОДАТКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ З ВИКОРИСТАННЯМ C++ ТА QT»

Виконавець
студент 4 курсу
групи ПД–41
Гончаренко Д.С.
Керівник
Дібрівний О.А.

Київ 2021

## Актуальність теми

На даному етапі розвитку сучасних технологій важливе місце займає технічна ре алізація сервісів прослуховування та цифрової обробки аудіо контенту, зокрема дистанційно із використанням C++ та Qt. Такі програми пропонують можливість розробки додатків для прослуховування аудіофайлів і надають всі можливості д ля організації процесів збирання, змішування та обробки аудіоданих,
а також для остаточного засвоєння.
Їх наявність спрощує процес запису та обробки, виключаючи потребу в просторі для інструментів та придбання дорогого обладнання для управління звуком та з мішування.

# ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- ***Об'єкт дослідження*** – технології і способи реалізації прослуховування та обробки аудіо контенту
- ***Предмет дослідження*** –
методи та способи отримання якісного аудіо контенту.

# МЕТА ДОСЛІДЖЕННЯ

- ***Мета роботи*** – розробка програмного забезпечення для прослуховування аудіофайлів.

## Метод дослідження

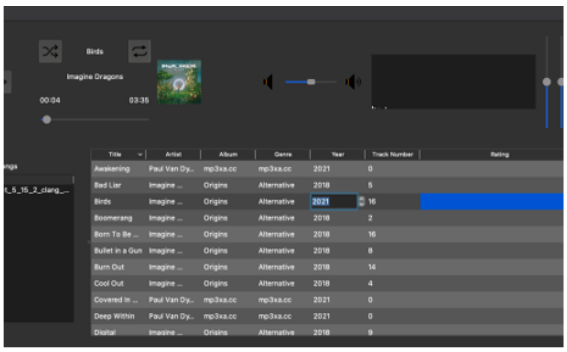
- Основний використанний метод дослідження це порівняльний аналіз способів реалізацій різних провідних компаній у сфе рі прослуховування і обробки аудіо контенту.

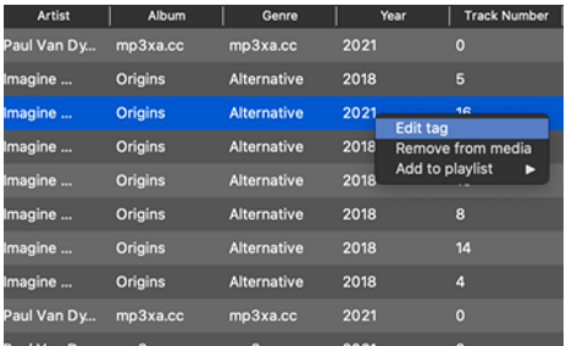
## Практичне значення одержаних результатів

Отримані результати в роботі можна використати для вибору технологій оптимального прослуховування та обробки аудіо контенту.

## Редагування метаданих

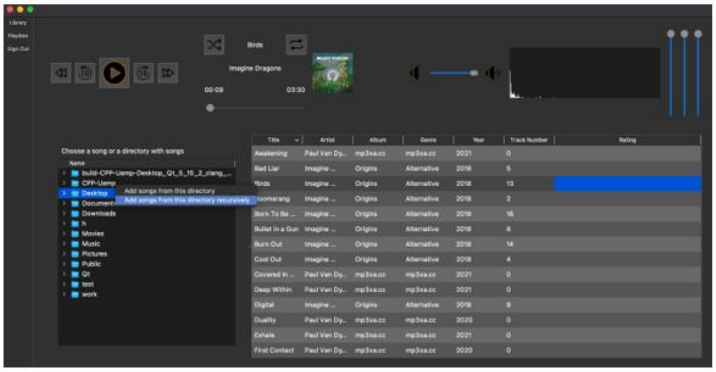
- Додаток який я розробив дозволяє редагувати метадані завдяки бібліотеки Taglib.






## Програвання музики

- Плеєр дозволяє прослуховування музики в декількох режимах.

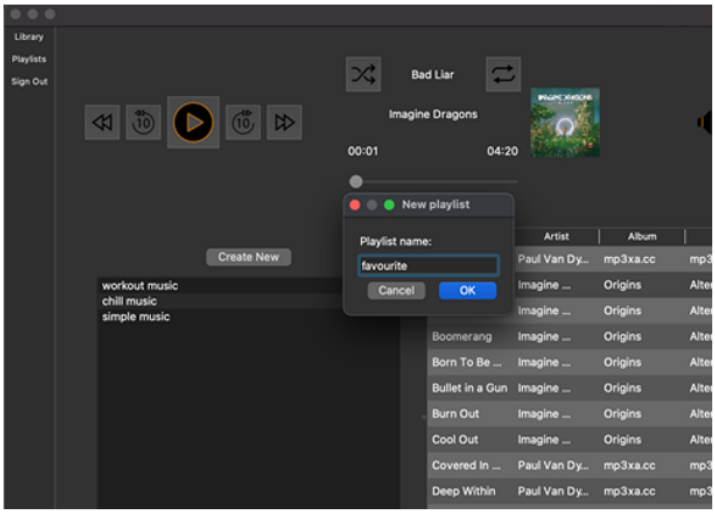


## Еквалайзер та візуалізація звукових частот

- Додаток дозволяє редагувати звукові частоти треки в режимі реального часу та дивитися на поточну частоту хвиль різного діапазону.

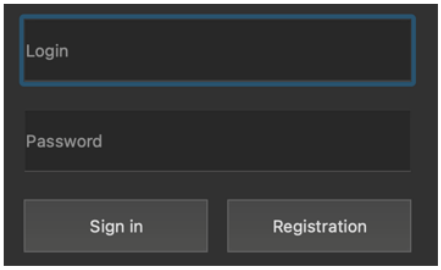
## Плейлисти

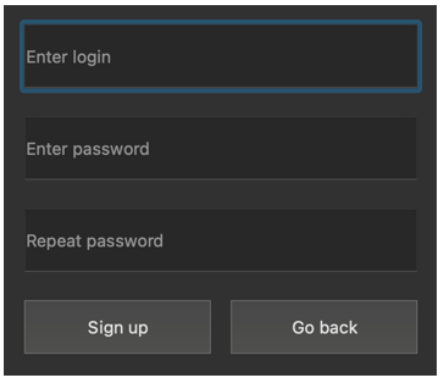
- Додаток дозволяє створювати,імпортувати,експортувати та видаляти плейлісти.



## Авторизування та зберігання данних

- Для входу в додаток користувачу потрібно створити аккаунт.
- Наступний раз коли користувач знову ввійде в додаток всі плейлісти які він створив та треки які він додав в бібліотеку будуть завантажені з бази даних.




# Участь у конференціях

- **ІІ ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ «СУЧАСНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ ІОТ»** З ТЕМОЮ АКТУАЛЬНІСТЬ СТВОРЕННЯ ДОДАТКУ ДЛЯ ПРОСЛУХОВУВАННЯ АУДІОФАЙЛІВ

- **ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ "ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ"** З ТЕМОЮ ОСОБЛИВОСТІ ВІДЕО ДЛЯ ІНТЕРНЕТУ