Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова
Херсонський навчально-науковий інститут

Кафедра інформаційних технологій та фізико-математичних дисциплін

«Допущений до захисту»
Завідувач кафедри

д.т.н., доцент Гучек П.Й.
«____» червня 2024 р.

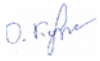
ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи на здобуття ступеня вищої освіти «бакалавр»

на тему: «Розробка програмного забезпечення медіаплеєра» ”

Виконала: студентка IV курсу, групи 4157
спеціальності
121 - Інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми
Інженерія програмного забезпечення
(назва освітньої програми)

Кудренко А.Є.
(прізвище та ініціали)

Керівник 
Гайдаєнко О.В.
(прізвище та ініціали)

Рецензент 
Ворона М.В
(прізвище та ініціали)

м. Миколаїв-Херсон - 2024 р.

Форма № Н-9.02

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова
ХЕРСОНСЬКИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

Кафедра інформаційних технологій та фізико-математичних дисциплін
Галузь знань 12 "Інформаційні технології"
(шифр і назва)
Спеціальність 121 "Інженерія програмного забезпечення"
(шифр і назва)
Освітня програма "Інженерія програмного забезпечення"
(назва)

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ Літвінова М.Б.
" 17 " березня 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студентки Кудренко Андрій Євгенович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Розробка програмного забезпечення медіаплеєра

Керівник роботи Гайдаєнко Оксана Володимирівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені розпорядженням по інституту від "04" березня 2024 року № 01

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1 Титульний аркуш, ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ (ДИПЛОМНУ) РОБОТУ, АНОТАЦІЯ (українською, англійською), ЗМІСТ, ПЕРЕЛІК УМОВНИХ ОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ (при необхідності).
4.2 ВСТУП
4.3 ОПИС ПРЕДМЕТНОЇ ГАЛУЗІ, ПОСТАНОВКА ЗАДАЧІ
4.4 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ПЗ)(зовнішні специфікації ПЗ, стани програми при виконанні, структура програми, структура даних, потоки даних та управління, проект ПЗ на рівні програмних модулів, випробування ПЗ)
4.5 РЕЗУЛЬТАТИ РОЗРОБКИ
4.6 РОЗДІЛ З ОХОРОНИ ПРАЦІ (ОХОРОНИ ОТОЧУЮЧОГО СЕРЕДОВИЩА)
4.7 ВИСНОВКИ

4.8 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

4.9 ДОДАТКИ (технічне завдання, опис програми, текст програми, програма та методика випробувань ПЗ, інструкція користувача)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

5.1 Діаграма варіантів використання програмного забезпечення медіаплеєра .

5.2 Діаграма компонентів і розгортання програмного забезпечення медіаплеєра .

5.3 Діаграма активності сеансу користувача програмного забезпечення медіаплеєра

5.4

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4.1-4.3	Дудченко О.М., декан		
4.4-4.5	Латанська Л.О., доцент		
4.6	Щедролосєв О.В., професор		

7. Дата видачі завдання 25 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Опис та аналіз предметної галузі	28.03.2024 р.	
2.	Постановка та формалізація задачі	06.04.2024 р.	
3.	Ескізний проєкт програмного забезпечення	13.04.2024 р.	
4.	Технічний проєкт програмного забезпечення	20.04.2024 р.	
5.	Робочий проєкт програмного забезпечення	27.04.2024 р.	
6.	Випробування програмного забезпечення	04.05.2024 р.	
7.	Розробка робочої документації	11.05.2024 р.	
8.	Виконання графічних документів	13.05.2024 р.	
9.	Вирішення питань охорони праці	18.05.2024 р.	
10.	Оформлення пояснювальної записки	25.05.2024 р.	
11.	Подання кваліфікаційної роботи на попередній захист	03.06.2024 р.	
12.	Подання кваліфікаційної роботи рецензенту	06.06.2024 р	
13.	Подання на кафедру ІТ та ФМД тексту остаточного варіанту роботи, підписаного її керівником, разом з заявою щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи	08.06.2024 р.	
14.	Подання на кафедру ІТ та ФМД електронних версії наступних документів у форматі pdf: кваліфікаційної (дипломної) роботи; файлу-опису кваліфікаційної роботи (згідно Застосунку до наказу ректора НУК від 19.05.2020 р. за №287-уч); презентації доповіді; письмового відгуку та рецензії на кваліфікаційну роботу	14.06.2024 р.	
15	Подання на кафедру ІТ ФМД письмового відгуку та рецензії на кваліфікаційну роботу	22.06.2024 р.	

Студент _____ Кудренко А.Є.
Підпис Прізвище та ініціали

Керівник роботи _____ Гайдаєнко О.В
Підпис Прізвище та ініціали

АНОТАЦІЯ

В даній роботі розробляється програмне забезпечення за темою: «Розробка програмного забезпечення медіаплеєра», призначене для відтворення медіа-файлів на мобільному пристрої з ОС Android.

Робота містить опис та аналіз предметної галузі, представлено проект програмного забезпечення, розділ з охорони праці, а також технічне завдання на розробку програмного забезпечення, текст програми, інструкцію користувача.

Android-додаток реалізовано в середовищі програмування Intellij IDEA.

Робота виконана на 57 сторінках машинописного тексту, містить 14 рисунків, 2 таблиці, 2 додатки та список використаних джерел з 5 найменувань.

Роботу викладено українською мовою.
Ключові слова: медіаплеєр, програмне забезпечення, відтворення аудіо, відтворення відео, плейлисти, бібліотека медіафайлів, субтитри, кросплатформовий, графічний інтерфейс, мультимедійні формати

ANNOTATION

This paper developed software on the topic: "Developing software media player", designed for media playback on mobile devices running Android.

The work contains a description and analysis of subject area, presented the project software section on health and technical requirements for software development, program text, user manual.

Android-application implemented in the programming environment Intellij IDEA.

Work on 57 pages of typewritten text, contains 14 figures, 2 tables, 2 applications and a list of sources of 5 items.

The work of the Ukrainian language.
Keywords: media player, software, audio playback, video playback, playlists, media library, subtitles, cross-platform, graphical user interface, multimedia formats.

ЗМІСТ

ВСТУП

Мобільні пристрої в житті людини є незамінним помічником, вони несуть користь тим, хто цінує свій час і не готовий витрачати його даремно. За рахунок компактності та функціональності мобільні пристрої стали невід’ємною частиною життя людини.

Android – вільна операційна система для мобільних телефонів, планшетних комп'ютерів, розумних годинників, телевізорів і смартбуків, що використовує ядро Linux, що розробляється Open Handset Alliance і належить Google. З моменту виходу першої версії у вересні 2008 року сталося 40 оновлень системи. Ці оновлення, як правило, стосуються виправлення виявлених помилок і додавання нових функцій в систему.

Музика та відео супроводжує людину протягом всього його життя. Пройшов час коли для прослуховування музики потрібно було носити з собою касетний плеєр чи CD-плеєр, а для перегляду відео потрібен був відеомагнітофон, касети для якого займали велику кількість місця.

Аналогів програмного забезпечення було знайдено велику кількість, але в кожному із них є як плюси так і мінуси. Основними недоліками є невелика кількість підтримуваних форматів медіа-файлів та спеціалізація на відео або аудіо-файлах. В даному додатку ці недоліки буде виправлено.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис та аналіз предметної галузі

Сучасний світ неможливо уявити без гаджетів, котрі супроводжують людину всюди у повсякденному житті. Однією з основних можливостей, яку надають гаджети є робота з мультимедіа. Практично кожен пристрій надає можливість для перегляду відео-файлів, прослуховування музичних композицій, перегляд фотографій та інше. І всі ці операції можливо проводити не лише над локальними даними, котрі зберігаються на внутрішній пам’яті пристрою чи на флеш-накопичувачі, а і над файлами з мережі інтернет (перегляд потокового відео).

В загальному вигляді мультимедіа – це комбінування різних форм представлення інформації на одному носієві, наприклад текстової, звукової і графічної.

Мультимедіа може бути грубо класифікована як лінійна й нелінійна.

Аналогом лінійного способу подання може бути кіно. Людина, що переглядає даний документ жодним чином не може вплинути на його зміст. Нелінійний спосіб подання інформації дозволяє людині брати участь у поданні інформації, взаємодіючи якимось чином із засобом відображення мультимедійних даних. Участь людини в даному процесі також називається «інтерактивністю».

Основними складовими мультимедіа є:

– Текст
– Аудіо
– Зображення
– Анімація
– Відео
– Інтерактивність.

В даній роботі увагу буде зосереджено на такій складовій, як аудіо.

Áудіо — загальний термін, що стосується звукових технологій.

Найчастіше під терміном аудіо розуміють звук, записаний на звуковому носії.

В свою чергу звук – це коливальний рух частинок пружного середовища, що поширюється у вигляді хвиль у газі, рідині чи твердому тілі.

Принцип запису і відтворення звуку в теорії досить простий: для запису необхідно відстежити кількість енергії, яка прикладена до певної точки молекулами, котрі формують звукові хвилі. Відтворення – це процес котрий змушує молекул повітря, котрі оточують динамік, рухатися так само, як вони робили це під час запису.

Звичайно, на практиці все складніше. Звук зазвичай записується одним з двох методів: аналоговим або цифровим. В обох випадках звукові хвилі записуються за допомогою мікрофона, головним елементом якого є мембрана, яка перетворює удари молекул в який-небудь вид сигналу. Спосіб обробки та зберігання сигналу і становить різницю між аналоговим і цифровим записом. Розглянемо більш детально процес формування цифрового запису.

При цифровому записі звуку стан мембрани мікрофону вимірюється за дискретний часовий такт і зберігається. Залежно від стану оточуючих молекул мембрана може вигинатися всередину або назовні, повертаючись потім в нейтральний стан. Процес вимірювання та збереження стану мікрофона називається семплюванням, оскільки ми зберігаємо семпли стану мембрани за дискретні такти часу. Кількість отриманих семплів за одиницю часу називається частотою дискретизації. Зазвичай часовий інтервал задається в секундах, а частота вимірюється в герцах (Гц). Чим більше семплів записується в секунду, тим вище якість запису.

Частота дискретизації - аж ніяк не єдиний параметр, що визначає якість звуку. Спосіб зберігання положень мембрани також має значення і теж є суб'єктом оцифровки. Стан мембрани – це розмір її відхилення від нейтрального положення. Оскільки напрям відхилення має значення, його значення зберігається зі знаком. Отже, положення мембрани під час певного тимчасового інтервалу – позитивне або негативне число. Це значення можна

зберігати різними способами: як ціле 8-, 16- або 32-бітове число зі знаком або як 32-бітове (і навіть 64-бітове) число з плаваючою крапкою. Кожен з цих типів даних має свою точність. Наприклад, 8-бітове ціле число може зберігати значення від -128 до 127. 32-бітний тип integer пропонує набагато більший діапазон. При зберіганні у вигляді float положення мембрани зазвичай нормалізується, щоб потрапляти в діапазон від -1 до 1. При цьому максимальне і мінімальне значення відповідають найбільшому відхилення мембрани від нейтрального положення в обидві сторони. Положення мембрани також називається амплітудою. Воно характеризує гучність звуку.

З одним мікрофоном можна записати тільки одноканальний звук (моно), в якому відсутнє відчуття простору. Маючи два мікрофони, можна вимірювати звукові хвилі в різних точках простору і отримувати таким образом так званий стереозвук. Досягти цього можна, наприклад, розміщуючи мікрофони зліва і праворуч від джерела звуку. Коли він потім відтворюється одночасно з двох динаміків, можна почути просторовий компонент аудіозапису. Однак це також означає, що необхідно зберігати під час запису вдвічі більше звукових семплів.

Відтворення - в загальному випадку процес нескладний. Маючи звукові семпли в цифровому вигляді, збережені з конкретною частотою дискретизації у вигляді даних певного типу, можна вивести їх на пристрій відтворення звуку, який трансформує інформацію в сигнали для підключеного до нього динаміка. Динамік дешифрує сигнал і перетворює його в коливання своєї мембрани, яка, в свою чергу, змушує рухатися молекули повітря навколо неї і таким чином генерувати звукові хвилі. Все те ж саме, що і при записі, тільки в зворотному порядку.

Щоб користувач міг без всяких перешкод працювати з усіма цими функціями та можливостями, потрібна операційна система, яка буде посередником між апаратною частиною мобільного пристрою та користувачем. Операційна система надає користувачеві зручну графічну оболонку, за допомогою якої останній, навіть не знаючи принципу роботи

самого пристрою, може без утруднень з ним працювати та користуватися усіма його можливостями.

За даними компанії Gartner [4], яка надала статистику по ринку мобільних операційних систем за 2-й квартал 2015 року, складається наступна картина:

- Windows Phone 2.5% ринку;
- Android 82.2% ринку;
- iOS 14.6% ринку;
- BlackBerry 0.3% ринку;
- Інші ОС 0.4% ринку.

Виходячи з аналізу ринку мобільних операційних систем, найбільшу популярність має саме Android – вільна операційна система для мобільних телефонів, планшетних комп'ютерів, розумних годинників, телевізорів і смартбуків. Вона розробляється Open Handset Alliance і належить Google. З моменту виходу першої версії у вересні 2008 року сталося 40 оновлень системи. Ці оновлення, як правило, стосуються виправлення виявлених помилок і додавання нових функцій в систему.

Android - це не просто ще один дистрибутив Linux для мобільних пристроїв. При розробці для Android швидше за все, не доведеться мати справу з самим ядром Linux. С точки зору програміста, Android - платформа, котра абстрагує розробника від ядра і дозволяє створювати код на Java. Android володіє декількома корисними можливостями. По-перше, це фреймворк, що пропонує великий набір API для створення різних типів додатків і, крім того, що забезпечує можливості повторного використання і заміни компонентів, які пропонуються платформою і сторонніми додатками. По-друге, наявність віртуальної машини Dalvik, що відповідає за запуск додатків на Android. Крім того, до послуг розробника набір графічних бібліотек для 2D- і 3D-додатків, підтримка мультимедіа-форматів (Ogg Vorbis, MP3, MPEG-4, H.264, PNG), API для доступу до камери, GPS, компасу, акселерометру, сенсорного екрану, джойстика і клавіатури.

Архітектура Android формується з набору компонентів. кожен компонент побудований на основі елементів нижчого рівня. На рис. 1.1 представлений короткий огляд головних компонентів Android.

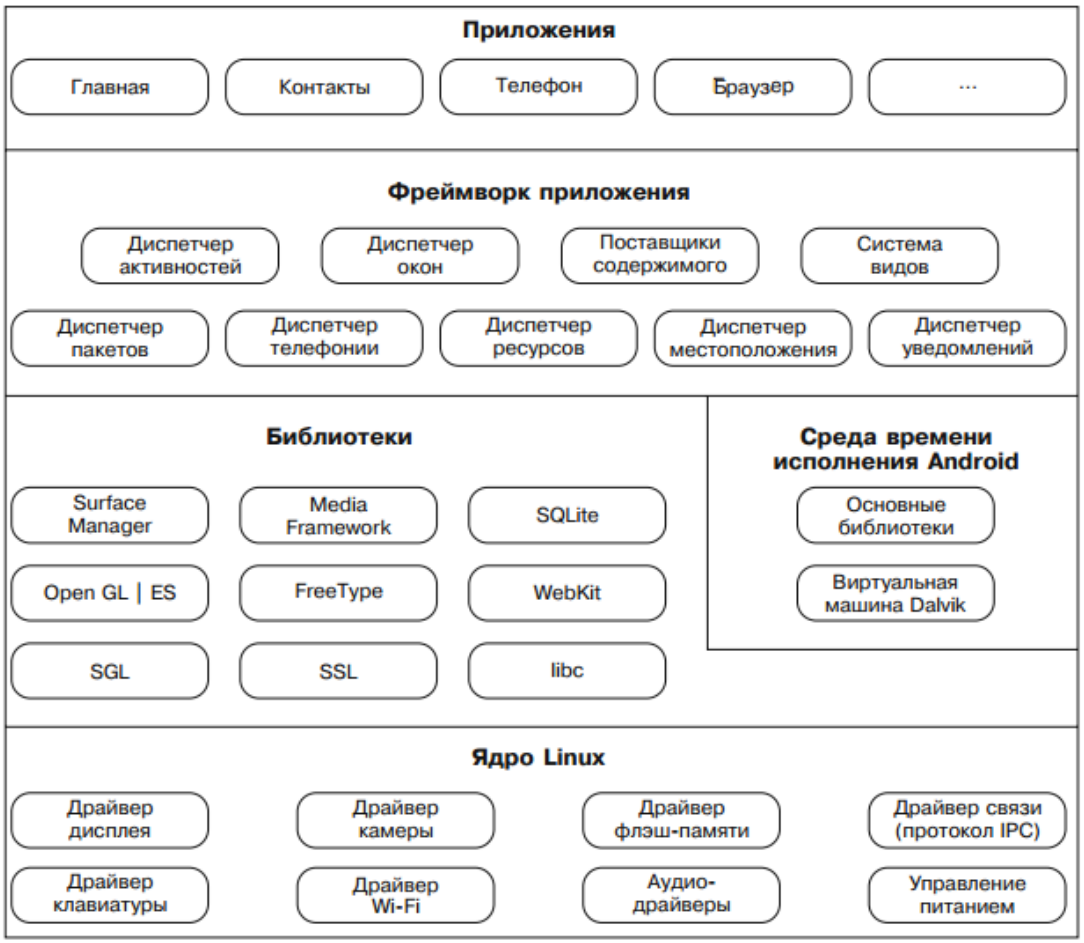


Рисунок 1.1 – Архітектура Android

У нижній частині рис. 1.1 можна побачити, що ядро Linux пропонує основні драйвери для апаратних компонентів системи. Крім того, ядро відповідає за пам'ять, управління процесами, підтримку мережі і т. д.

Середовище виконання Android, що є надбудовою над ядром, відповідає за породження і виконання додатків Android. Кожна програма працює у власному процесі зі своєю віртуальною машиною Dalvik. Dalvik запускає програми в байт-кодовому форматі DEX. Зазвичай звичайні Java-файли з розширенням CLASS перетворюються в формат DEX за допомогою спеціальної утиліти dx, наявної в SDK. Формат DEX займає набагато менше місця в пам'яті, ніж класичні файли типу CLASS, що досягається великим

стисненням, розбиттям на таблиці і злиттям декількох CLASS-файлів. Віртуальна машина Dalvik взаємодіє з бібліотеками ядра, що пропонують базовий функціонал для Java-програм. Ці бібліотеки мають у своєму розпорядженні не повний набір класів, доступних через Java SE, і використовують частину реалізації Apache Harmony. Крім іншого це означає, що в Java SE для Dalvik відсутні Swing і Abstract Window Toolkit, а також всі класи з Java ME. Однак є можливість використовувати (з обережністю) сторонні бібліотеки для Java SE на Dalvik.

До Android 2.2 (Froyo) весь код був інтерпретованим. У Froyo був представлений JIT-компілятор, здатний компілювати частини байт-коду в машинний код на льоту. Це значно збільшує продуктивність програмних додатків, що вимагають великих обчислень. JIT-компілятор може використовувати можливості процесора, спеціально призначені для складних обчислень, наприклад для операцій з плаваючою крапкою (Floating Point Unit, FPU). Крім того, в Dalvik включений власний збирач сміття (Garbage Collector, GC).

Крім бібліотек ядра, що пропонують деяку функціональність Java SE, існує також набір рідних бібліотек на C / C ++. Ці системні бібліотеки в більшості своїй відповідають за складні в обчислювальному сенсі завдання (відображення графіки, відтворення звуку, доступ до бази даних), що не дуже підходящі для віртуальної машини Dalvik.

Фреймворк додатків пов'язує разом системні бібліотеки та середовище виконання, створюючи таким чином призначену для користувача частину Android. Фреймворк управляє додатками і надає середовище, в якому вони працюють. Розробники створюють додатки для цього фреймворка за допомогою набору програмних інтерфейсів на Java, що охоплюють такі області, як розробка призначеного для користувача інтерфейсу, фонові служби, оповіщення, управління ресурсами, доступ до периферії і т. д. Всі ключові програми, що поставляються разом з ОС Android (наприклад, поштовий клієнт), написані за допомогою цих API.

Додатки, будь вони з інтерфейсом або з фоновими службами, можуть зв’язуватись з іншими додатками. Цей зв'язок дозволяє одним додаткам використовувати компоненти інших.

Аналогів програмного забезпечення було знайдено велику кількість, але в кожному із них є як плюси так і мінуси.

Програма «VLC Media Player» (див. рисунок 1.2) – користується великою популярністю, безкоштовний та з відкритим вихідним кодом, крос-платформний мультимедійний програвач, який відтворює більшість мультимедійних файлів та протоколи мережі потокової передачі.

Плюси і мінуси програмного продукту «VLC Media Player».

Плюси:

– без реклами;
– підтримка інтернет-трансляцій;
– вибір теми оформлення;
– додаткові функції для розробників;
– відтворення аудіо та відео файлів (mp3, ogg, wav, mp4, avi).

Мінуси:

– не зручний інтерфейс;
– відсутність вибору мови програмного забезпечення.

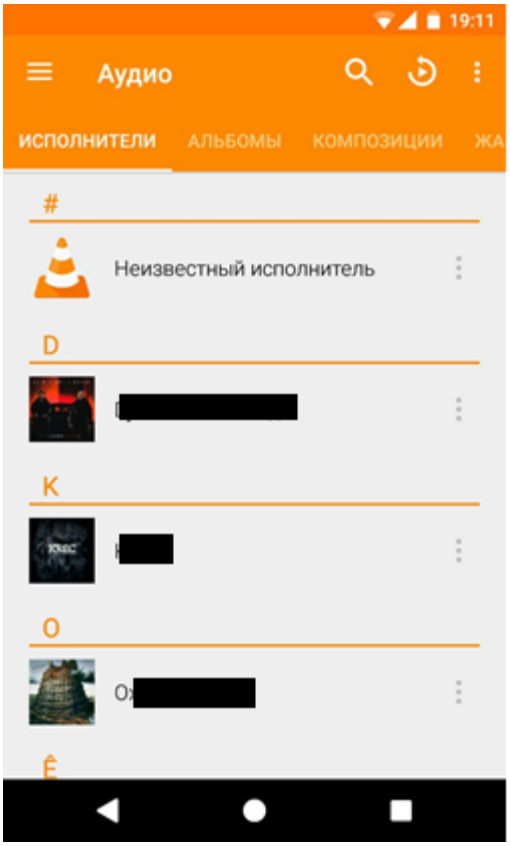


Рисунок 1.2 Програмне забезпечення «VLC Media Player»

Програмне забезпечення «Video Player for Android» (див. рисунок 1.3) є безкоштовний мультимедійний плеєр для операційної системи Android, програмне забезпечення дозволяє дивитися всі популярні формати відео на телефоні або планшеті з апаратним прискоренням для більш швидкого і більш плавного відтворення HD відео з неперевершеною легкістю і комфортом.

Плюси:

– відсутня реклама;
– підтримка інтернет-трансляцій;
– відтворення відео файлів (mp4, avi, 3gp).

Мінуси:

– незручний інтерфейс;
– відсутність відтворення аудіо файлів;
– відсутність вибору мови програмного забезпечення;
– відсутні налаштування.

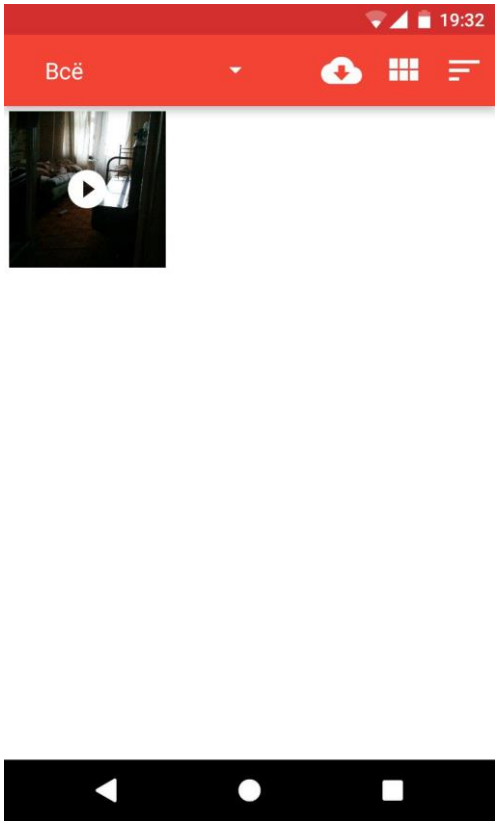


Рисунок 1.3 – Програмне забезпечення «Video Player for Android»

Програмне забезпечення «MX Player» (див. рисунок 1.4) мультимедійний плеєр який займає перше місце в рейтингу безплатних медіа програвачів Play Market.

Плюси:

– відтворення відео файлів (mp4, avi, 3gp);
– підтримка інтернет-трансляцій;
– вибору мови програмного забезпечення.

Мінуси:

– незручний інтерфейс;
– відсутність відтворення аудіо файлів;
– реклама;
– відсутні налаштування.

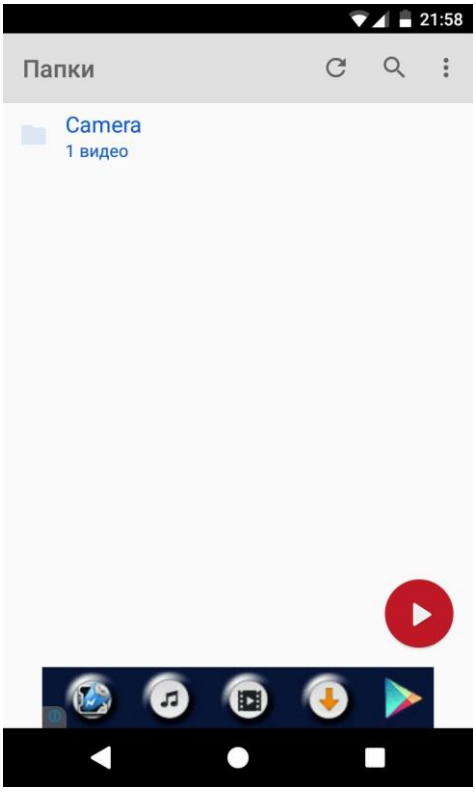


Рисунок 1.4 – Програмне забезпечення «MX Player»

За допомогою ж розроблюваного Android додатку, можна буде відтворювати відео та аудіо файли.

1.2 Постановка задачі

Виходячи з аналізу предметної галузі сформулюємо наступну постановку задачі: розробити програмне забезпечення для відтворення аудіо та відео файлів, з підтримкою інтернет-трансляцій.

Для реалізації поставленої мети, ПЗ повинно задовольняти наступні вимоги та виконувати наступні функції:

– відтворення аудіо файлів;

– пошук аудіо файлу за критеріями: виконавець, альбом, назва;

– створення фільтрів для аудіо файлів за наступними категоріями: виконавець, альбом;

– створення плей листів;

– відтворення відео файлів;

– відтворення відео через мережевий потік;

– вибір мови інтерфейсу.

Інтерфейс мобільного додатку повинен бути орієнтований на самого недосвідченого користувача, який буде дуже простий і зручний у повсякденному використанні.

Детальні вимоги до програмного забезпечення оформлено у вигляді документу «Технічне завдання» наведено у додатку А.

РОЗДІЛ 2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МЕДІАПЛЕЄРА

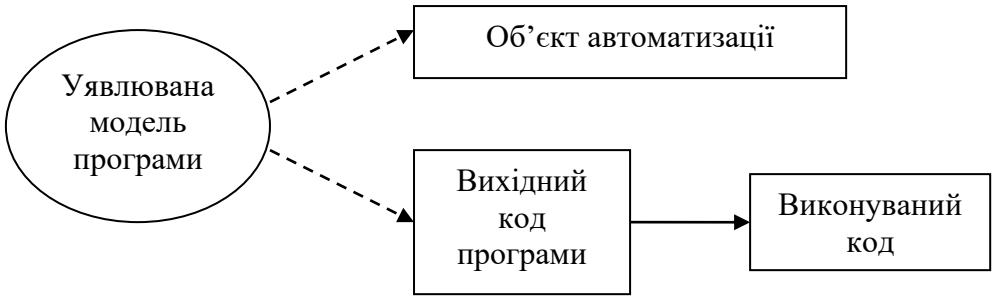
2.1 Ескізний проект

2.1.1 Вибір засобів моделювання

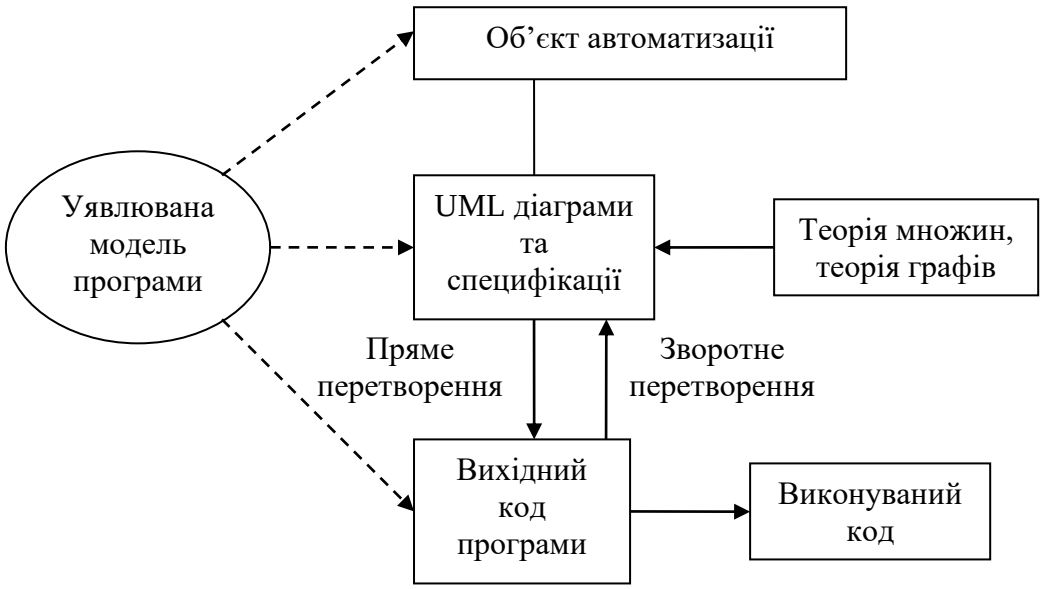
Для побудови складної інформаційної системи необхідним етапом є проектування. В останнє десятиліття в комп'ютерному світі намітилася тенденція проектування систем візуальними (наочними) моделями. Причому в нових методах проектування складних комп'ютерних систем, наприклад об’єктно-орієнтованому проектуванні (ООП) і об’єктно-орієнтованому аналізі (ООА), наочні моделі дуже часто зв'язуються з такими зоровими образами як «погляди», спрямовані на складну систему з різних точок зору. Набір з декількох наочних моделей (модельних поглядів) створює у свідомості фахівців інтегральний образ складної комп'ютерної системи, яку вони спільно проектують. Разом з тим, наочні моделі є ефективним засобом документування комп'ютерних систем і їх програмних забезпечень, а також мовою спілкування між програмістами, системними аналітиками й замовниками систем.

На поточний момент існує багато методологій моделювання систем, тому необхідно провести їх аналіз та визначити найзручніші засоби для моделювання розроблюваної системи.

Найбільш відомими візуальними моделями, використовуваними для проектування комп'ютерних систем і їхніх програмних забезпечень, є діаграми мови UML (Unified Modeling Language) і стандарту IDEF0, таблиці й діаграми стандарту IDEF1X. Ці візуальні моделі мають математичну основу у вигляді теорій графів, множин і матриць. На Рисунку 2.1 показано, чим відрізняється проектування з використанням UML від старих методів проектування.



а)



б)

Рисунок 2.1 – Схеми технологій програмування

Рисунок дозволяє зрозуміти причини революційних змін в області технологій програмування, викликаних появою мови UML. На ньому зображені дві схеми. Перша з них (Рисунок 2.1а) зображує ситуацію, що існувала в області технологій програмування до створення мови UML, друга (Рисунок 2.1б) – показує зміну ситуації після появи UML. На обох схемах ліворуч показані програмісти й уявлювані ними моделі комп'ютерних програм, а праворуч зображені коди програм і предметні області, у яких ці програми використаються. На другій схемі між предметними областями й програмними кодами з'явилися діаграми мови UML і їх математична основа – теорії множин і графів.

Як видно з Рисунку 2.1а об'єднання тексту програми (її вихідного коду) з характеристиками об'єкта автоматизації здійснюється тільки у свідомості програміста, а документальний зв'язок між ними відсутній.

Розглянемо тепер ситуацію, що виникла після появи мови UML (Рисунок 2.1б). Діаграми й специфікації мови UML зв'язали вихідний текст програми з характеристиками об'єкта автоматизації. При цьому UML діаграми опираються на теоретичний фундамент у вигляді теорії множин і теорії графів. Наявність теоретичної основи дозволяє спростити операції перетворення UML діаграм, зображених на екранах дисплеїв, і зменшити об'єм пам'яті, необхідної для зберігання діаграм.

Рисунок також показує, що UML діаграми можуть бути перетворені у вихідний код (пряме перетворення) і навпаки вихідний код може бути перетворений в діаграми (зворотне перетворення). У деяких випадках пряме перетворення може здійснюватися автоматично за допомогою програм конвертерів. У цей час група OMG активно працює над рішенням проблеми прямого перетворення діаграм UML. Зворотне перетворення може виконати тільки людина.

Документування вихідних кодів програм UML діаграмами й специфікаціями створює єдину мову спілкування між програмістами, а також між програмістами, системними аналітиками й замовниками автоматизованої системи. Але саме головне, що мова UML надала можливість широкої стандартизації мов програмування. Відомо, що в різних мовах програмування використаються однакові операції й методи, але вони мають різні назви й символьні позначення. Мова UML дозволяє стандартизувати як самі операції й методи мов програмування так і їхню термінологію.

Отже, для проектування розроблюваного ПЗ доцільно буде використати технологію SADT та мову UML. В якості середовища моделювання буде використаний найбільш відомий у світі CASE-засіб – пакет Rational Rose.

2.1.2 Діаграма варіантів використання

Перший крок у проектуванні програмного забезпечення – це створення контекстної моделі. Складемо контекстну діаграму у вигляді діаграми варіантів використання мови UML. Для цього виділимо зі словнику системи діючих осіб:

- Користувач.

Побудуємо діаграму варіантів використання, яка зображена на рисунку 2.2.

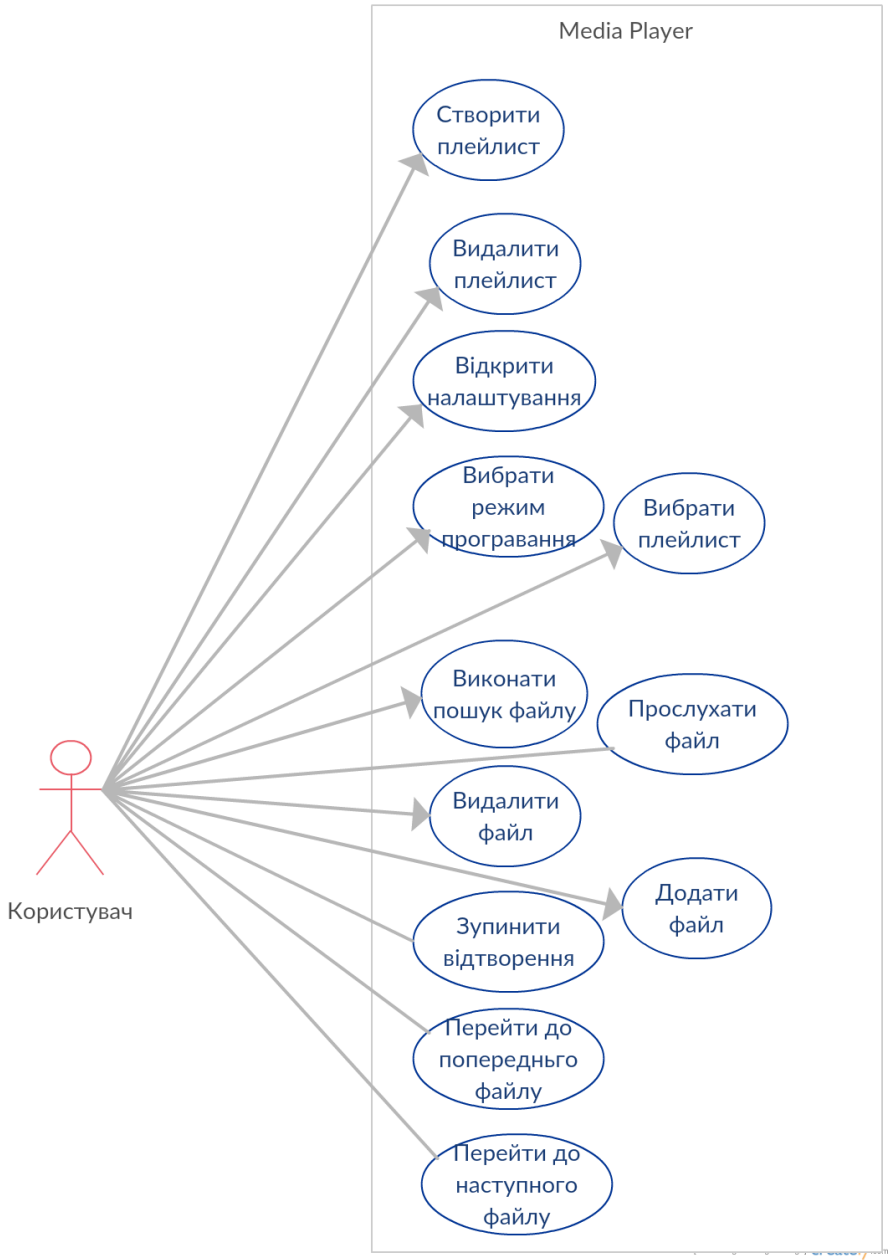


Рисунок 2.2 – Діаграма варіантів використання програмного забезпечення

Далі представимо опис кожного з варіантів використання у таблиці 2.1.

Таблиця 2.1 – Опис варіантів використання

Відкрити налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу змінити мову, оформлення інтерфейсу, розмір шрифту
Передумови	Відсутні
Основний потік подій	Відкривається вікно налаштувань. Користувач виконую необхідні налаштування.
Альтернативний потік подій	Відсутній
Постумови	Відсутній
Створити плейлист	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу створити плейлист
Передумови	Відсутні
Основний потік подій	Відкривається вікно створення плейлиста. Користувач вводить ім’я плейлиста та додає до нього медіа-файли.
Альтернативний потік подій	Відсутній
Постумови	Відсутній
Вибрати режим програвання	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє змінити режим програвання
Передумови	Відсутні
Основний потік подій	Відкривається діалогове вікно вибору режиму. Користувач вибирає необхідний пункт.
Альтернативний потік подій	Відсутній
Постумови	Відсутній
Виконати пошук файлу	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє виконати пошук файлу по найменуванню
Передумови	Відсутні
Основний потік подій	Відкривається діалогове вікно пошуку. Користувач вводить текст для пошуку та отримую перелік файлів, котрі задовольняють умові.
Альтернативний потік подій	Відсутній
Постумови	Відсутній

2.1.3 Розробка концептуальної моделі

З метою проектування програмного забезпечення та на основі контекстної діаграми розробимо концептуальну модель даних. Концептуальну модель зазвичай представляють у вигляді діаграми «сутність-зв'язок» (ER-діаграми). У мові UML їй відповідає діаграма класів. Діаграма класів системи представлена на рисунку 2.3.

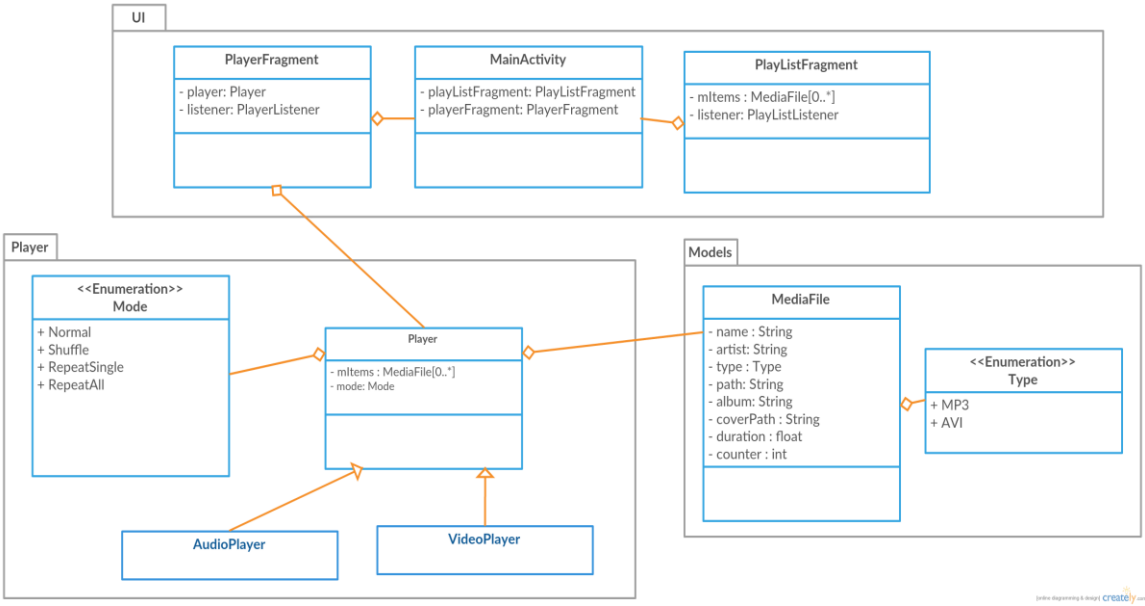


Рисунок 2.3 – Концептуальна модель програмного забезпечення

Виконаємо опис зв'язків між сутностями.

Класи «AudioPlayer» та «VideoPlayer» пов'язані з класом «Player» зв'язком наслідування, де клас «Player» є батьківським класом.

Клас «Player» пов'язаний з класами «Mode» та «MediaFile» зв'язком агрегації, де клас «Player» виступає в ролі цілого.

Клас «MediaFile» пов'язаний з класом «Type» зв'язком агрегації, де клас «MediaFile» виступає в ролі цілого.

Опишемо атрибути кожної сутності.

Атрибути сутності «Mode»:

- Normal – режим відтворення медіа-файлів у порядку слідування

- Shuffle – режим відтворення медіа-файлів у випадковому порядку

- RepeatSingle – режим для циклічного відтворення медіа-файлу

Атрибути сутності «MediaFile»:

- name - назва медіа-файлу

- artist - виконавець

- type – тип медіа-файлу

- path – шлях до медіа-файлу

-album - альбом

- duration - тривалість

- counter – лічильник відтворень

Атрибути сутності «Player»:

- items – список медіа-файлів

- mode – режим відтворення

Атрибути сутності «Type»:

- MP3 – тип медіа-файлу – mp3

- AVI – тип медіа-файлу – avi

2.1.4 Розробка діаграми станів

Діаграма станів – в UML, діаграма, яка описує всі можливі стани одного екземпляру певного класу і можливі послідовності його переходів з одного стану в інший, тобто моделює всі зміни станів об'єкта як його реакцію на зовнішні впливи.

Для користувача можливій такі переходи та стани:

Після запуску додатку, користувач потрапляє до головного вікна, де передбачена можливість завершити виконання додатку. З початкового стану користувач може перейти до вікна налаштувань, відкрити вікно програвача або запросити список медіа-файлів. Стан налаштувань дозволяє по завершенню повернутись до головного вікна. Стан перегляду медіа-файлів дозволяє перейти до програвача за допомогою вибору файлу з представленого списку. Зі стану програвача можна повернутись до головного вікна, перейти до

перегляду файлів, змінити режим програвання, зупинити або відновити програвання файлу, перейти до наступного або попереднього файлу.

Діаграма станів системи зображена на рисунку 2.4.

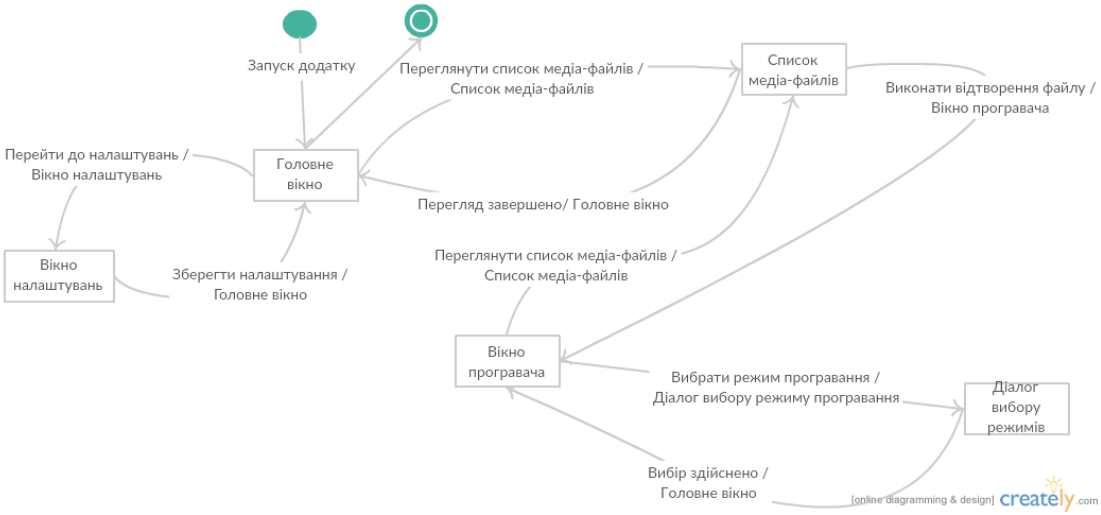


Рисунок 2.4 – Концептуальна модель програмного забезпечення

2.1.5 Проектування інтерфейсу користувача

Важливим етапом при проектуванні програмного забезпечення є розробка інтерфейсу. Додаток медіаплеєр розробляється на основі екранних форм, що містять поля для редагування, управляючі елементи – кнопки, компоненти перегляду графіки та ін.. Виходячи з опису діаграм використання можна виділити основні модулі, які потребують розробки інтерфейсу:

- перегляд плейлистів;

- пошуку файлів по типу;

-режим відтворення

На рисунку 2.5 показана форма режиму відтворення. На формі присутні елементи управління, котрі дозволяють зупинити або почати відтворення, перейти до наступного треку, перейти до попереднього треку.

Рисунок 2.5 – Режим відтворення звукового-файлу

На рисунку 2.6 показана форма пошуку по типу файлів. На формі представлені елементи списку, після вибору елемента користувач потрапить в плейлист з файлами вказаного типу.

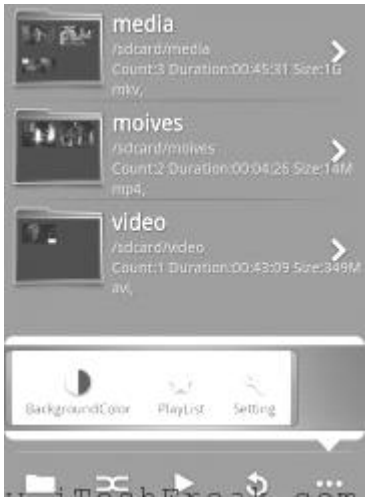


Рисунок 2.6 – Форма пошуку по типу файла

На рисунку 2.7 представлено форму вибору плейлиста. Користувач може переглянути всі плиейлисти котрі він створив.

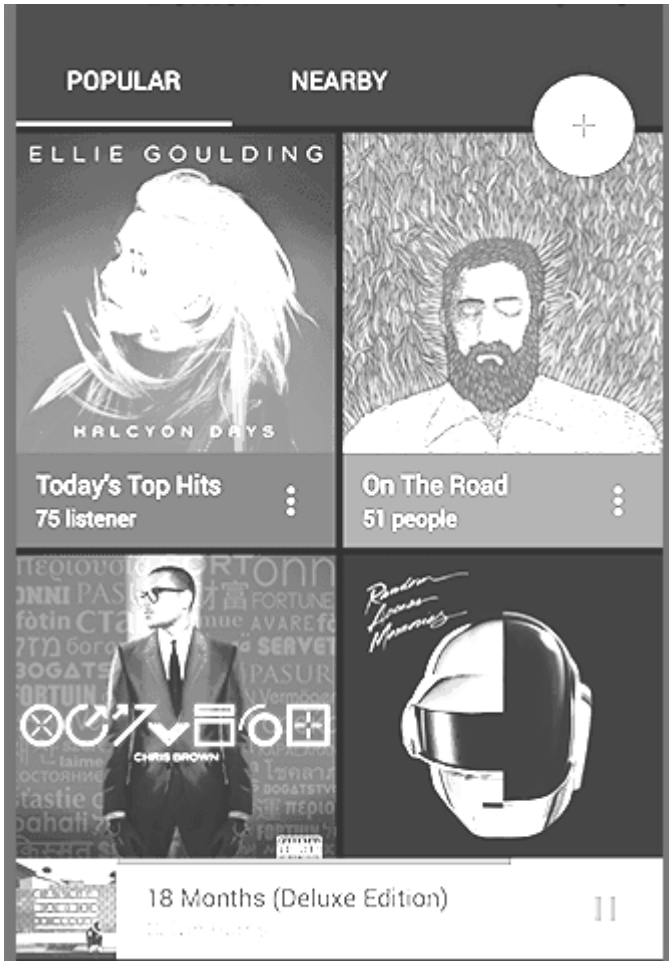


Рисунок 2.7 – Форма вибору плейлиста

1.2 Технічний Проект

1.2.1 Розробка діаграми класів програмного забезпечення

Діаграма класів – показує статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

На рисунку 2.8 представлено діаграму класів. Розглянемо більш детально обов’язки класів та функціонал, який вони будуть надавати.

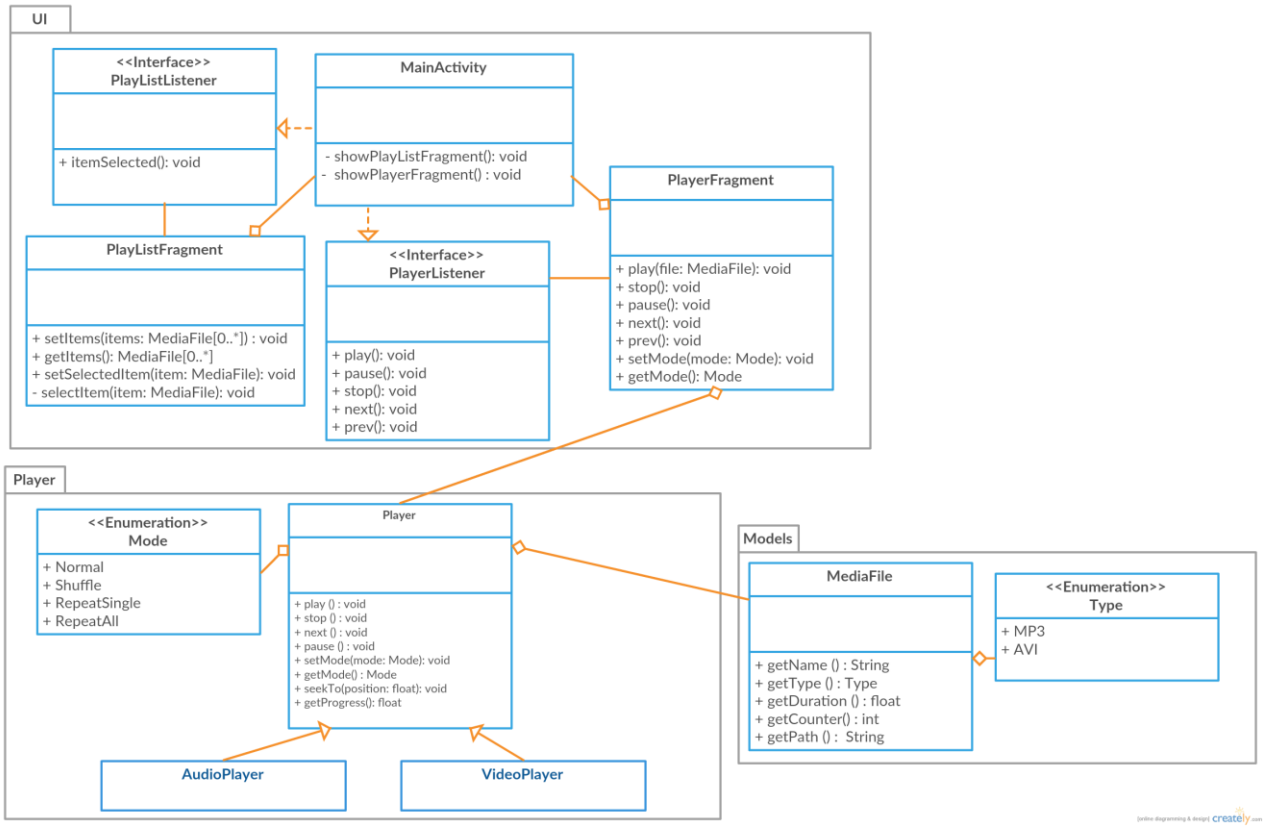


Рисунок 2.8 – Діаграма класів програмного забезпечення

Обов’язки класів:

- Клас «Player» містить базовий функціонал програвача: почати відтворення, зупинити відтворення, наступний файл, попередній файл, перейти до позиції, змінити режим програвання;
- Клас «AudioPlayer» та «VideoPlayer» містять реалізації базового функціоналу для аудіо- та відео-програвача відповідно;
- Клас «MediaFile» - це клас-модель, який містить інформацію про медіа-файл та методи доступу.
- Клас «PlayerFragment» відповідає за візуалізацію елементів управління, надає користувачеві можливість взаємодіяти з програвачем
- Клас «PlayListFragment» надає можливість перегляду списку медіа-файлів
- Клас «MainActivity» - являється точкою входу. Є посередником для класів «PlayerFragment» та «PlayListFragment»

Опишемо методи кожного класу.

Методи клас «Player»:

- play() – почати відтворення файлу
- stop() – зупинити відтворення файлу
- next() – перейти до наступного файлу
- pause() – зупинити відтворення
- setMode() – змінити режим відтворення
- getMode() – повернути поточний режим відтворення
- seekToPosition() – перейти до позиції
- getProgress() – повернути поточну позицію
- Методи клас «MediaFile»:
- getName() – повернути ім’я файлу
- getType() – повернути тип файлу
- getDuration() – повернути тривалість файлу
- getCounter() – повернути кількість відтворень
- getPath() – повернути шлях до файлу
- Методи клас «PlayerFragment»:
- play() – почати відтворення файлу
- stop() – зупинити відтворення файлу
- next() – перейти до наступного файлу
- pause() – зупинити відтворення
- setMode() – змінити режим відтворення
- getMode() – повернути поточний режим відтворення

Методи клас «PlayListFragment»:

- setItems() – додати файли до плей-листа
- getItems() – повернути файли
- setSlectedItem() – повернути виділений файл
- Методи клас «MainActivity»:
- showPlaylistFragment() – відобразити PlaylistFragment
- showPlayerFragment () - відобразити PlayerFragment

Методи інтерфейсу «PlayListListener»:

- itemSelected() – сповістити про вибір файлу
- Методи інтерфейсу «PlayerListener»:
- play() – сповістити про початок відтворення
- pause() – сповістити про зупинку відтворення
- next() – сповістити про перехід до наступного файлу
- prev() - сповістити про перехід до попереднього файлу

1.2.2 Розробка діаграм послідовностей

Діаграма послідовності – в UML, діаграма, що відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

Кожна з діаграм послідовностей відповідає варіанту використання системи. Побудуємо діаграми послідовностей для варіантів використання програмного забезпечення, представлених на рисунку X.

Розглянемо діаграму послідовності для вибору плейлиста для відтворення.

Перші два кроки – це вибір плейлиста, відправка всіх його файлів до списку програвання, та відображення списку програвання.

Третій та четвертий кроки дозволяють користувачу вибрати файл для програвання в поточному списку програвання.

Всі послідуючі кроки – це відтворення медіа-файлу.

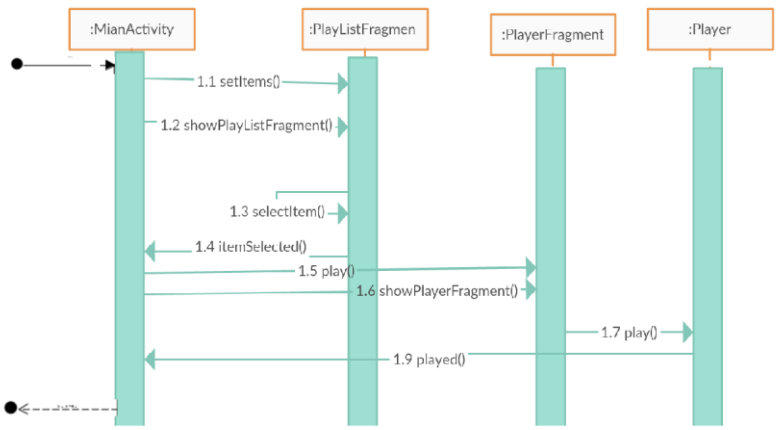


Рисунок 2.9 – Діаграма послідовності

1.3 Робочий проект

1.3.1 Вибір засобів реалізації

На етапі розробки програмного забезпечення визначаємо вимоги до мови програмування:

операційна система Android;

графічний інтерфейс користувача;

об’єктна орієнтованість

Приймаючи до уваги наведені вимоги для вирішення поставленого завдання була обрана мова програмування Java . Java (джава) – об’єктно-орієнтована мова програмування високого рівня, з автоматичним керуванням пам’яттю та високорівневими структурами даних, таких як словники (хеш-таблиці), списки, кортежі. Підтримує класи, модулі (які можуть бути об’єднані в пакети), обробку виключень, а також багато потокові обчислення. Java має простий і виразний синтаксис. Java – активно розвиваюча мова, нові версії виходять приблизно раз в два з половиною роки.

Java легко портується і працює майже на всіх відомих платформах. Існують порти під Microsoft Windows, усі варіанти UNIX (включаючи GNU/Linux), Plan 9, Mac OS і Mac OS X, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390 та Symbian.

Основним недоліком мови Java являється порівняно невисока швидкість виконання програм. Однак вважається, що цей недолік компенсується за рахунок зменшення часу розробки програм і Java-програмісти вважають, що на Java можна вирішити задачу у середньому в 3-5 разів швидше ніж на C++ чи іншій мові.

2.3.2 Опис середовища розробки програмного забезпечення, що працює на платформі Android

Вибір середовища програмування для розроблюваного ПЗ варто здійснювати за наступними критеріями: доступність, функціональність, гнучкість. Оскільки для програмування поставленої задачі було обрано мову

програмування Java та цільову платформу – Google Android, то середовище розробки потрібно обирати за цими критеріями.

Існують наступні середовища розробки, які є найбільш гнучкими та популярними:

Eclipse;

Intellij IDEA.

При розробці програмного забезпечення буде використано середовище розробки Intellij IDEA, яке вже придбане розробником. Це рішення обумовлене широкою загальною популярністю даного середовища програмування та можливостями. Воно містить вільні Android SDK (інструменти для розробки) та багато корисних надбудов та плагінів.

2.3.3 Кодування програмного забезпечення

Виконавши аналіз середовищ розробки програмного забезпечення (пункт дипломної роботи 2.3.2) на мові програмування Java, було обрано саме Intellij IDEA. Для програмування Android-додатку, функціональні вимоги якого представлені в постановці задачі (більше детально – в Додатку А) дане середовище програмування більш ніж підходить. Проектування ПЗ показує, що першим і головним завданням є кодування саме MediaPlayerActivity – головного класу в додатку.

Представимо лістинг коду даного класу:

Лістинг 1 – Клас MediaPlayerActivity

```
import android.app.PendingIntent;
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.net.Uri;
import android.os.Bundle;
import android.os.IBinder;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by Dante on 12/29/2015.
 */
public class MediaPlayerActivity extends AppCompatActivity
                        implements PlayListFragment.PlayListListener,
        PlayerFragment.PlayerControllerListener {
```

```
    public static final int RESULT_CODE = 5;
    public static final String INTENT_EXTRA_KEY = "pendingIntent";

    private PlayerFragment playerFragment;
    private PlayListFragment playListFragment;

    private MusicService musicService;
    private Intent playIntent;
    private boolean serviceBound = false;
    private String directory = "";

    private final String PLAYLIST_FRAGMENT_TAG = "playlist_fragment_tags";
    private final String PLAYER_FRAGMENT_TAG = "player_fragment_tag";


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_main);

        FragmentManager fragmentManager = getSupportFragmentManager();
        playerFragment = (PlayerFragment)
fragmentManager.findFragmentByTag(PLAYER_FRAGMENT_TAG);
        if(playerFragment == null){
            playerFragment = PlayerFragment.newInstance(this);
            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.replace(R.id.mpaLinearLayoutControlPanel, playerFragment,
PLAYER_FRAGMENT_TAG).commit();
        }
        playerFragment.setPlayerControllerListener(this);
        playListFragment = (PlayListFragment)
fragmentManager.findFragmentByTag(PLAYLIST_FRAGMENT_TAG);
        if(playListFragment == null){
            playListFragment = PlayListFragment.newInstance(this);
            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.replace(R.id.mpaLinearLayoutPlayList, playListFragment,
PLAYLIST_FRAGMENT_TAG).commit();
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        PendingIntent pendingIntent = createPendingResult(RESULT_CODE, new Intent(), 0);
        if(playIntent == null){
            playIntent = new Intent(this, MusicService.class);
            playIntent.putExtra(INTENT_EXTRA_KEY, pendingIntent);

            startService(playIntent);
            bindService(playIntent, serviceConnection, 0);
        }else {
            bindService(playIntent, serviceConnection, 0);
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_CODE){
            selectItemInPlayList();
            playerFragment.setCurrentItem(musicService.getSelectedItem());
        }
    }

    private ServiceConnection serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            musicService = ((MusicService.MusicBinder) service).getService();
```

```
            Intent intent = getIntent();
            Uri uri = intent.getData();
            if (uri != null){
                musicService.clearPlayList();
                playListFragment.loadPlayList(uri);
            }

            if(!musicService.isEmptyPlayList()){
                playListFragment.loadPlayListByPath(musicService.getPath());
                selectItemInPlayList();
            }else {
                playerFragment.setCurrentItem(musicService.getSelectedItem());
                playListFragment.setSelectedItem(musicService.getSelectedItem(),
musicService.getCurrentSong());
            }
            serviceBound = true;
            playerFragment.updateControl();

        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            serviceBound = false;
        }
    };

    @Override
    protected void onStop() {
        super.onStop();
        if(serviceBound) {
            unbindService(serviceConnection);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    public void nextSong() {
        if(!serviceBound)
            return;
        musicService.playNext();
        selectItemInPlayList();
        playerFragment.setCurrentItem(musicService.getSelectedItem());
    }

    @Override
    public void prevSong() {
        if(!serviceBound)
            return;
        musicService.playPrev();
        selectItemInPlayList();
        playerFragment.setCurrentItem(musicService.getSelectedItem());
    }

    @Override
    public void play() {
```

```
    if(!serviceBound)
        return;
    musicService.play();
    selectItemInPlayList();
    playerFragment.setCurrentItem(musicService.getSelectedItem());
}

@Override
public void stop() {
    if(!serviceBound)
        return;
    musicService.stop();
}

@Override
public void pause() {
    if(!serviceBound)
        return;
    musicService.pause();
}

@Override
public void shuffle() {
    if(!serviceBound)
        return;
    musicService.shuffle();
}

@Override
public boolean isShuffle() {
    if(!serviceBound)
        return false;
    return musicService.isShuffle();
}

@Override
public int getDuration() {
    if(!serviceBound)
        return 0;
    return musicService.getDuration();
}

@Override
public int getCurrentPosition() {
    if(!serviceBound)
        return 0;
    return musicService.getCurrentPosition();
}

@Override
public void seekTo(int pos) {
    if(!serviceBound)
        return;
    musicService.seekTo(pos);
}

@Override
public boolean isPaused() {
    if(!serviceBound)
        return false;
    return musicService.isPaused();
}

@Override
public boolean isPlayed() {
    if(!serviceBound)
        return false;
    return musicService.isPlayed();
}
```

```
    @Override
    public boolean isStopped() {
        if(!serviceBound)
            return true;
        return musicService.isStopped();
    }

    @Override
    public void playListChanged(List<Item> playList, String path) {
        if (!serviceBound)
            return;
        musicService.setPlayList(playList);
        playerFragment.setCurrentItem(musicService.getSelectedItem());
        musicService.setPath(path);
    }

    @Override
    public void itemSelected(int item, long id) {
        if (!serviceBound)
            return;
        if (musicService.getCurrentSongId() != id)
            musicService.play(item);

        playerFragment.updateControl();
        selectItemInPlayList();
        playerFragment.setCurrentItem(musicService.getSelectedItem());

    }

    private void selectItemInPlayList(){
        Item it = musicService.getSelectedItem();
        int pos = musicService.getCurrentSong();
        if (it == null)
            return;
        playListFragment.setSelectedItem(it, pos);
    }
}
```

Інші класи, такі як клас списку композицій, представлення опцій представлені в Додатку В.

РОЗДІЛ 3. РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис компонент програми

Для реалізації поставленої задачі було розроблено 11 класів: FileDialog.java, Item.java, MediaPlayerActivity.java, MediaProvider.java MusicService.java, MyCursorAdapter.java, PlayListFragment.java PlayerFragment.java, SortDialog.java, SortableListView.java, Utilities.java.

По завершенню процесу розробки програмного забезпечення було отримано один файл MP.apk.

Для того, щоб програмний продукт міг повноцінно функціонувати необхідно мати пристрій з версією Android 5.0 та вище.

3.2 Діаграма розгортання програмного забезпечення

Діаграма розгортання – діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти та об’єкти, що виконуються на цих вузлах.

Діаграма розгортання на рисунку 3.1 говорить про те, що вузлом на якому буде працювати програмне забезпечення є мобільний пристрій з операційною системою Android на котрому встановлено файл додатку MP.apk та можлива наявність зовнішнього флеш-накопичувача.

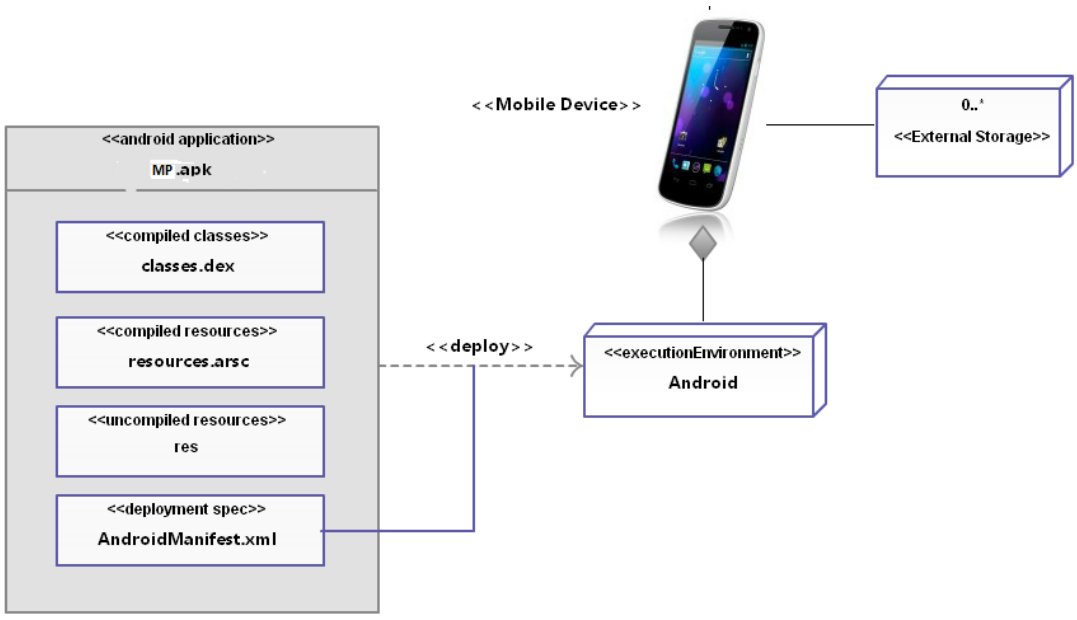


Рисунок 2.10 – Діаграма розгортання

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці – система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі роботи.

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності працівників розумової праці. Їх робота стала інтенсивніше, напруженою, такий, що вимагає значних витрат розумової, емоційної і фізичної енергії. Це зажадало ком-комплексного рішення проблем ергономіки, гігієни та організації роботи, регламентації режимів праці та відпочинку.

Дана система включає:

- аналіз шкідливих факторів і небезпек, які діють на людину;
- розробка заходів по усуненню несприятливих дій на чоловіка і створенню нормальних умов роботи.

4.2 Аналіз небезпечних і шкідливих факторів на робочому місці розробника медіаплеєра

При роботі на персональному комп'ютері співробітники відділу стикаються з дією таких небезпечних фізичних і психологічних чинників як - підвищена температура зовнішнього середовища, недостатня освітленість робочої зони, відсутність або нестача природного світла, електричний струм, статична електрика, розумове перенапруження, перенапруження очей, монотонність праці, емоційні перевантаження. Розглянемо чинники докладніше.

Велику небезпеку становлять дисплеї з електронно-променевими трубками (ЕЛТ) - джерела шкідливих випромінювань, небезпечно впливає на здоров'я операторів.

При роботі монітора виникають два типи випромінювань: електростатичне і електромагнітне. Навколо електростатичного зарядженого

монітора підвищується концентрація пилу. Електромагнітне випромінювання створюється магнітними ка-тушками, які знаходяться біля цокольній частині ЕЛТ. Результати медич-ських досліджень показують, що пил, який електризується, може викликати опік шкіри, а електромагнітне випромінювання може призводити до зниження загальної продуктивності оператора.

Крім цього важливим моментом є показники частоти вертикальної і горизонтальної розгортки ЕЛТ. Прийняті стандарти на монітори не дозволяють використовувати ЕЛТ і відеоадаптери з частотою вертикальної розгортки менше 75 Гц.

Мікрокліматичні параметри впливають на самопочуття і здоров'я людини, а також на надійність роботи засобів обчислювальної техніки. Для комфортної роботи приймається температура 230 С і 180 С в теплу і холодну пору року відповідно, при відносній вологості 55%. [ГОСТ 12.1.005-88].

Також існують інші небезпечні шкідливі фактори, такі як:

Небезпека ураження електричним струмом.

Все обладнання ЕОМ, як будь-яка електроустановка, представляють для людини велику потенційну небезпеку, оскільки в процесі експлуатації або проведення профілактичних робіт людина може торкнутися струмовідні частини.

Експериментальні дослідження показали, що людина відчуває подразнюючу дію змінного струму промислової частоти силою 0,6-1,5 мА і постійного струму 5-7 мА. Ці струми не становлять серйозної небезпеки для діяльності організму людини, оскільки при такій силі струму можливо самостійне звільнення осіб від контакту з струмоведучими частинами. Для змінного струму промислової частоти сила не відпускає струму знаходиться в межах 6-20 мА. Постійний струм не викликає не відпускає ефекту, але призводить до сильних больових відчуттів, сила такого струму 15-80 мА і більше. (ГОСТ 12.1.038-82)

Недостатня освітленість робочого місця. Правильно спроектоване і виконане висвітлення в приміщенні, де працюють оператори, забезпечує високу працездатність, надає позитивний психологічний вплив на тих, хто працює, сприяє підвищенню продуктивності роботи.

Рекомендована освітленість для роботи для роботи з екраном дисплея становить 400-750 лк. Рекомендовані яскравості в полі зору операторів повинні лежати в межах 1:5-1:10.

Можливість виникнення пожежі.

Приміщення, в якому розміщені ПЕОМ за категоріями пожежної небезпеки відноситься до категорії "В". Зазвичай в ньому знаходиться велика кількість можливих джерел спалаху: кабельні лінії, що використовуються для живлення ПЕОМ від мережі змінного струму напругою 220 В, електронно-променева трубка монітора, яка є вибухонебезпечною без додаткового захисту, устаткування, меблі з горючих матеріалів, папір.

4.3 Розробка заходів по зменшенню дії небезпечних і шкідливих факторів

При розробці заходів, які забезпечують охорону праці та БЖ, враховують всі шкідливі і небезпечні фактори, зменшуючи кожен з них до допустимих значень. Тим не менше, тільки дотримання правил і норм технічної безпеки та охорони праці забезпечить гарну робочу обстановку і запобіжить нещасні випадки на робочому місці.

При придбанні моніторів, слід вибирати монітори узгоджених з рекомендаціями щодо зменшення електричних і магнітних полів, а також що підтримують вертикальну частоту розгортки 75 Гц і більше.

Вимоги до мікроклімату в робочому приміщенні.

Для підтримки необхідних мікрокліматичних параметрів необхідно встановлювати кондиціоноване обладнання. Для зменшення кількості пилу встановлюють періодичність вологого прибирання в приміщенні.

Вимоги до електробезпеки.

Велике значення для запобігання електротравматизму має правильна організація обслуговування діючих електроустановок, проведення ремонтних та профілактичних робіт, що здійснюється за допомогою наступних заходів: допуск до роботи, нагляд під час роботи, виробництво відключень під час ремонту, вивішування попереджувальних плакатів і знаків безпеки, перевірка відсутності напруги, накладення заземлення.

Для забезпечення електробезпеки обслуговуючого персоналу передбачених заземлювальні пристрої, до яких підключені всі металеві частини робочого обладнання. У зв'язку з необхідністю розміщення проводів електроживлення обладнання без зайвих витрат на переобладнання приміщення, доцільно влаштовувати підлога з підпіллям.

Для зниження величин виникаючих статичних зарядів в обчислювальних центрах застосовують покриття технологічне з одношарового антистатичного лінолеуму марки ASN. Можна застосовувати загальне і місцеве зволоження повітря. Одним з нових методів зменшення статичної напруги в приміщенні є нейтралізація електрики іонізованим газом.

Вимоги до освітленості.

В дисплейних залах, звичайно, застосовують одностороннє природне бічне освітлення. З метою уникнення прямого сонячного світла використовують приміщення з вікнами з північної, північно-східній або північно-західною орієнтацією. Монітори розташовують подалі від вікон і так, щоб вікна були збоку. Якщо екран монітора розташований до вікна, необхідні спеціальні екранують пристрої.

Для штучного освітлення приміщень слід використовувати люмінесцентні лампи, оскільки в них висока світлова віддача (до 75 лм / Вт і більше), тривалий термін служби (до 10000 годин), мала яскравість поверхні, яка світиться, близький до природного спектральний склад випромінюваного світла , який забезпечує хороше перенесення кольорів. Найбільш прийнятними

для дисплейних приміщень є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (білого-тепло-білого світла) потужністю 40, 80 Вт

Для виключення засвічена екранів дисплеїв прямими світловими потоками світильники загального освітлення розташовують збоку від робочого місця, паралельно лінії зору оператора і стіні з вікнами. Таке розміщення світильників дозволяє проводити їх послідовне включення залежно від величини природної освітленості і виключає роздратування очей смугами світла і тіні, що чергуються, виникають при поперечному розташуванні світильників.

Вимоги до пожежної безпеки.

Для запобігання виникнення пожежі необхідно передбачити заходи пожежної профілактики: дотримання протипожежних вимог під час проектування і експлуатації систем вентиляції згідно зі СНиП 1.01.02-84; дотримання умов пожежної безпеки електроустановок згідно вимог; наявність засобів сповіщення:

- Пожежні сповіщувачі (ЛІП-1, ІП-105 2/1 і т.д.);
- Установки пожежогасінні (АУП);
- Інструкції по заходам протипожежної безпеки, план евакуації людей і технічних засобів.

Для поліпшення умов пожежної безпеки в приміщенні має бути встановлений підлога з негорючих матеріалів, технологічно знімний; папір зберігається в металевій шафі; в наявності два вогнегасники типу OU-5, а також два димових датчика; в машинному залі систематично проводиться прибирання та вентилювання приміщення.

Загальні вимоги з техніки безпеки

Кожен користувач ЕОМ зобов'язаний:

- Дотримуватися правил внутрішнього трудового розпорядку;
- Не вносити на територію підприємства і не розпивати спиртні напої, палити лише у відведеному для цієї мети місці.

На території підприємства необхідно виконувати наступні вимоги:

Рухатися тільки по тротуарах, у разі пересування по проїжджій частині дороги потрібно йти по лівій стороні назустріч рухомого транспорту, виконуючи заходи обережності.

При вимушеній зупинці на середині дороги не кидатися в сторони, а стояти на одному місці, щоб дати можливість водієві об'їхати Вас з тієї чи іншої сторони.

Обходити на безпечній відстані місця, де ведуться висотні роботи.

До основних небезпек, які можуть призвести до травм при роботі оператора ЕОМ, відносять небезпеку ураження електричним струмом та іонізуючим випромінюванням.

Необхідно стежити за тим, щоб підлога в машинному залі була рівна і неслизький, все люки, кабелю, проводу повинні бути закриті або захищені.

Суворо дотримуватися правил пожежної безпеки на робочому місці. У разі загоряння відключити електроживлення і зателефонувати в пожежну частину, пояснивши, що і де горить.

Вимоги до безпеки перед початком роботи

Прийнявши ЕОМ від змінника, необхідно перевірити, чи добре прибране робоче місце, ознайомитися з неполадками, які були в попередній зміні, в роботі ЕОМ і з прийнятими заходами по їх усуненню. Також перед початком роботи необхідно:

- Отримати завдання у керівника роботи, а також інструктаж з техніки безпеки.
- Підготувати необхідні інструменти, прилади. Перевірити їх ис-правность, якщо необхідно, отримати захисні пристосування, запчастини, радіодеталі, матеріали.
- Перевірити надійність кабелів в місцях їх підключення до джерел живлення.
- Перевірити відсутність замикання між земляними ланцюгами та ланцюгами живлення напруги.

- Перевірити наявність, справність і відповідність по струму запобіжників в блоках ремонтованої ЕОМ або пристроїв.

Вимоги до безпеки під час роботи:

- Не дозволяється залишати ЕОМ, що знаходиться під напругою, без спостереження.
- Не дозволяється замінювати знімні елементи і проводити переміщення і перемонтування на включеній ЕОМ.
- Не дозволяється користуватися несправною апаратурою та інструментами.
- Не дозволяється поєднувати і роз'єднувати розетки і вилки роз'ємів, які знаходяться під напругою.
- Не дозволяється знімати кришки та щити, які закривають доступ до струмоведучих частин, при включенні обладнання.
- Не дозволяється змінювати запобіжники під напругою.
- При заміні запобіжників в блоках і пристроях, суворо керівництво-тися маркування по струму.
- Не дозволяється користуватися паяльником з незаземленим корпусом.
- При ремонті знімних блоків системи електроживлення їх корпус необхідно заземлити.
- Всі операції, пов'язані з установкою переносних приладів і вимірами, повинні виключати дотик струмоведучих частин.
- При ремонті системи електроживлення необхідно вивішувати плакати, НЕ ВКЛЮЧАТИ! ПРАЦЮЮТЬ ЛЮДИ або НЕ ВКЛЮЧАТИ! РОБОТА НА ЛІНІЇ
- При проведенні робіт необхідно вимагати присутності другої людини, допущеного до роботи з електричними установкам і з напругою до 1000 вольт.

Вимоги до безпеки по закінченню робіт:

- Вимкніть електроживлення від ЕОМ і обладнання, яке забезпечує роботу ЕОМ.
- Приведіть у порядок робоче місце.
- Повідомте керівника робіт (начальника зміни, начальника машини) про закінчення роботи та її результати.
- Зробіть запис в "Журналі зауважень про роботу ЕОМ".

ВИСНОВОК

В даному дипломному проекті було пройдено шлях розробки програмного забезпечення від викладення вимог до написання коду, відладки та тестування. В якості базової мови для проектування обрано мову програмування Java, що дало змогу в короткий термін розробити повноцінний програмний продукт. Таким чином були закріплені знання мови програмування Java, принципи об’єктно – орієнтованого програмування.

Реалізацію частини програмного продукту для відтворення аудіо-файлів було виконана повністю.

В звіті були сформульовані вимоги до програмного продукту та оформлені у вигляді документу «Технічне завдання».

Розроблено розділ з охорони праці де розглянути питання аналізу небезпечних і шкідливих факторів, характерних при роботі з персональним комп’ютером та розроблені заходи щодо забезпечення сприятливих умов праці програміста.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Optical character recognition [Електронний ресурс]// https://en.wikipedia.org – 2019 - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ Optical_character_recognition

2. Dr.B.Rama, Santosh Kumar Henge OCR-The2 layered approach for classification and identification of telugu hand written mixed consonants and conjunct consonants by using advanced fuzzy logic controller // Department of Computer Science, Kakatiya University, Warangal,India – 2016 – c.77-78

3. Лавренюк М.С., Новіков О.М. Огляд методів машинного навчання для класифікації великих обсягів супутникових даних - System Research &Information Technologies, 2018, № 1. - С. 54-57

4. Огляд підходів розпізнавання машинописних текстів [Електронний ресурс]// http://asu.kpi.ua – 2019 - Режим доступу до ресурсу: http://asu.kpi.ua/wp-content/uploads/2019/05/ISTU-2019-16-17.05.pdf

5. Расстояние Дамерау-Левенштейна [Електронний ресурс]// https://ru.wikipedia.org – 2019 - Режим доступу до ресурсу:https://ru.wikipedia.org/wiki/Расстояние_Дамерау_Левенштейна– № 1 (36). – с. 61-63

6. Методичні рекомендації щодо підготовки звіту з переддипломної практики та виконання кваліфікаційної роботи бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» / С. Б. Приходько, Л. М. Макарова, Л. О. Латанська, С. В. Суслов, Т. А. Фаріонова. – Миколаїв: НУК, 2023. – 53 с.

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МЕДІАПЛЕЄРА

А.1 Загальні відомості

Назва розробки: Програмне забезпечення медіаплеєра для відтворення аудіо- та відеофайлів.

Підстава для розробки: Необхідність створення зручного, багатофункціонального та кросплатформового медіаплеєра для користувачів ПК і мобільних пристроїв.

Призначення системи: Забезпечення відтворення мультимедійного контенту у різних форматах, організація бібліотеки медіафайлів, підтримка плейлистів та базових функцій управління відтворенням.

А.2 Мета та завдання розробки

Мета: Створення надійного та зручного програмного медіаплеєра для відтворення аудіо- та відеофайлів різних форматів.

Завдання розробки:

Аналіз існуючих медіаплеєрів та визначення основних функцій;

Проєктування інтерфейсу користувача;

Реалізація відтворення аудіо та відео;

Підтримка популярних форматів файлів (MP3, WAV, MP4, AVI та ін.);

Реалізація управління плейлистами;

Забезпечення можливості відображення субтитрів та керування гучністю;

Оптимізація продуктивності та ресурсоємності;

Тестування та впровадження програмного продукту.

А.3 Об’єкт автоматизації

Об’єктом автоматизації є процес відтворення мультимедійних файлів на персональному комп’ютері та мобільних пристроях користувачів.

А.4 Функціональні вимоги

Медіаплеєр повинен забезпечувати:

Відтворення аудіо та відео різних форматів;

Створення та редагування плейлистів;

Управління відтворенням: пауза, стоп, наступний/попередній трек;

Регулювання гучності та налаштування звуку;

Відображення інформації про файл (назва, тривалість, формат);

Відображення субтитрів для відео;

Пошук файлів у бібліотеці;

Збереження налаштувань користувача.

А.5 Нефункціональні вимоги

Кросплатформеність (Windows, macOS, Linux, мобільні ОС);

Інтуїтивно зрозумілий та сучасний інтерфейс;

Висока продуктивність та швидке завантаження файлів;

Стабільність роботи та відсутність критичних помилок;

Мінімальне споживання ресурсів системи;

Безпека даних користувача.

А.6 Вимоги до програмного та технічного забезпечення

Операційні системи: Windows 10/11, macOS, Linux, Android, iOS;

Мови програмування: C++, Python, JavaScript, або інші сучасні мови для кросплатформових додатків;

Фреймворки: Qt, Electron, React Native або аналогічні для GUI;

Бібліотеки для обробки мультимедіа: FFmpeg, VLC SDK, MediaPlayer API;

Доступ до файлової системи користувача для роботи з медіафайлами.

А.7 Етапи розробки

Аналіз предметної області та вимог.

Проєктування архітектури програми та інтерфейсу користувача.

Програмна реалізація модулів: відтворення аудіо, відтворення відео, плейлисти, субтитри.

Тестування та налагодження функціоналу.

Впровадження та підготовка користувацької документації.

А.8 Порядок контролю та приймання

Контроль виконання здійснюється через тестування всіх функціональних модулів;

Перевірка стабільності та продуктивності;

Демонстрація роботи плеєра та відповідність технічним вимогам;

Підписання акту приймання-передачі розробки.

ДОДАТОК Б. ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МЕДІАПЛЕЄРА

Б.1 Загальні відомості

Медіаплеєр призначений для відтворення аудіо- та відеофайлів різних форматів (MP3, WAV, MP4, AVI та ін.). Програма дозволяє організовувати бібліотеку медіафайлів, створювати плейлисти, керувати відтворенням та налаштовувати відображення субтитрів.

Система працює на персональних комп’ютерах та мобільних пристроях із сучасними операційними системами. Доступ до програми здійснюється через графічний інтерфейс.

Б.2 Вимоги до користувача

Для роботи з медіаплеєром користувач повинен мати:

ПК, ноутбук або мобільний пристрій;

Операційну систему: Windows 10/11, macOS, Linux, Android або iOS;

Доступ до локальної файлової системи для роботи з медіафайлами;

Базові навички роботи з комп’ютером та графічним інтерфейсом.

Б.3 Запуск програми

Встановіть програму медіаплеєра, слідуючи інструкціям інсталятора.

Запустіть програму подвійним кліком на значок плеєра.

На головному екрані відкривається бібліотека медіафайлів та панель керування відтворенням.

Б.4 Основні функції

Відтворення файлів

Натисніть кнопку «Відкрити файл» або «Додати до бібліотеки».

Оберіть аудіо- або відеофайл у локальній папці.

Натисніть кнопку «Відтворити» для запуску відтворення.

Використовуйте кнопки «Пауза», «Стоп», «Наступний», «Попередній» для управління відтворенням.

Управління гучністю та звуком

Використовуйте повзунок гучності для регулювання рівня звуку.

Можна включити або вимкнути звук кнопкою «Mute».

Плейлисти

Створіть новий плейлист через меню «Плейлисти» → «Створити плейлист».

Додайте медіафайли у плейлист.

Відтворюйте весь плейлист або окремі файли.

Субтитри

Для відеофайлів можна додати субтитри через кнопку «Додати субтитри».

Формати субтитрів підтримуються стандартні: SRT, ASS, VTT.

Інформація про файл

При відтворенні файлу відображаються: назва, формат, тривалість та розмір.

Можна переглянути додаткову інформацію через меню «Властивості файлу».

Б.5 Пошук файлів у бібліотеці

Введіть назву файлу або виконавця у поле пошуку.

Система відобразить всі відповідні медіафайли.

Клацніть на файл для відтворення.

Б.6 Налаштування програми

Вибір теми інтерфейсу: світла або темна.

Автоматичне відновлення останнього сеансу відтворення.

Налаштування форматів, які відображаються у бібліотеці.

Збереження параметрів плейлистів та історії відтворення.

Б.7 Завершення роботи з програмою

Натисніть кнопку «Вихід» у меню.

Закрийте програму через стандартну кнопку закриття вікна.

Рекомендація: перед виходом переконайтеся, що плейлисти та зміни налаштувань збережені.

Б.8 Усунення можливих проблем

Файл не відтворюється: перевірте підтримуваний формат файлу.

Відео без звуку: перевірте гучність та налаштування аудіо.

Програма зависає: перезапустіть медіаплеєр або систему.

Субтитри не відображаються: перевірте формат файлу субтитрів.

ДОДАТОК В. ОРГАНІЗАЦІЯ БІБЛІОТЕКИ МЕДІАФАЙЛІВ У ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ МЕДІАПЛЕЄРА

В.1 Загальні відомості

Бібліотека медіафайлів — це структурований модуль медіаплеєра, який забезпечує зберігання, пошук, сортування та управління аудіо- та відеофайлами користувача.
Вона реалізована у вигляді бази даних або файлової структури з метаданими для ефективної роботи плеєра.

В.2 Структура бібліотеки медіафайлів

Основні компоненти бібліотеки:

Категорії файлів:

Аудіо;

Відео;

Плейлисти.

Метадані файлів:

Назва файлу;

Тип/формат файлу (MP3, WAV, MP4, AVI, тощо);

Виконавець/автор;

Жанр;

Дата додавання;

Тривалість;

Шлях до файлу на локальному диску;

Наявність субтитрів (для відео).

Плейлисти:

Користувацькі (створені вручну);

Автоматичні (наприклад, останні додані, улюблені).

В.3 Таблиці/структури даних у бібліотеці

Таблиця MediaFiles (Медіафайли)

Поле	Тип даних	Опис
file_id	INT (PK)	Унікальний ідентифікатор файлу
file_name	VARCHAR	Назва файлу
file_type	VARCHAR	Тип файлу (audio/video)
format	VARCHAR	Формат файлу (MP3, WAV, MP4, AVI)
author	VARCHAR	Виконавець або автор
genre	VARCHAR	Жанр
duration	TIME	Тривалість файлу
file_path	VARCHAR	Шлях до файлу у файловій системі
added_date	DATETIME	Дата додавання файлу до бібліотеки
subtitle_path	VARCHAR	Шлях до субтитрів (якщо є)

Таблиця Playlists (Плейлисти)

Поле	Тип даних	Опис
playlist_id	INT (PK)	Унікальний ідентифікатор плейлисту
name	VARCHAR	Назва плейлисту
creation_date	DATETIME	Дата створення
type	VARCHAR	Тип плейлисту (користувацький / авто)

Таблиця PlaylistItems (Елементи плейлисту)

Поле	Тип даних	Опис
item_id	INT (PK)	Унікальний ідентифікатор елемента
playlist_id	INT (FK)	Ідентифікатор плейлисту
file_id	INT (FK)	Ідентифікатор медіафайлу
order_index	INT	Порядок відтворення у плейлисті

В.4 Алгоритми роботи з бібліотекою

Додавання файлу:

Перевірка формату файлу;

Збереження файлу у відповідну папку;

Додавання запису у таблицю MediaFiles.

Створення плейлисту:

Користувач вводить назву;

Призначається playlist_id та дата створення;

Плейлист зберігається у таблиці Playlists.

Додавання файлів у плейлист:

Обираються файли з MediaFiles;

Створюються записи у PlaylistItems з порядком відтворення.

Пошук файлів у бібліотеці:

За назвою, автором, жанром або форматом;

Відображення результатів у графічному інтерфейсі.

Відтворення плейлисту:

Завантаження всіх файлів з PlaylistItems у порядку order_index;

Відтворення через аудіо/відео модуль плеєра.

Висновки

Структурована бібліотека медіафайлів дозволяє:

швидко знаходити і відтворювати потрібні файли;

організовувати персональні та автоматичні плейлисти;

зберігати додаткову інформацію про файли та субтитри;

забезпечити масштабованість та гнучкість роботи плеєра.