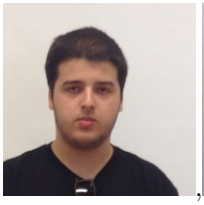
TP3-Gestão de Redes-MIEI

João Bernardo Freitas a74814
João Mendes a71862

3 Fevereiro 2020




1 Fase A- Definição de requisitos funcionais

Para a resolução deste trabalho vamos tentar implementar os seguintes requisitos genéricos.

- A capacidade de listar remotamente as músicas duma colecção musical armazenada num sistema de ficheiros. A listagem deve permitir algum tipo de organização hierárquica da colecção, nem que seja a implementada no próprio sistema de ficheiros (com directorias e subdirectoras).
- Um script/programa de instalação para cada componente já com uma configuração por defeito pré-definida.
- A possibilidade do utilizador escolher uma música para ser tocada no sistema remoto onde se encontra a colecção. Neste caso deve ser fornecido um conjunto de informações mínimas da música que está a ser tocada, como o nome e a duração total.
- Suporte para ficheiros musicais do tipo MP3.
- Suporte para vários tipos de formatos áudio para além do MP3, como exemplo, WAV e FLAC.
- Implementação de sistema de cache para a listagem das músicas na colecção.
- Implementação de logs de actividades de todo o sistema.
- Escolha dum grupo de músicas (álbum, obra, todas as músicas dum artista, etc.) para ser tocado no sistema local onde se encontra a colecção.

Segue-se um simples diagrama de sequência de modo a entender melhor alguns dos principais componentes deste trabalho, nomeadamente algumas das classes e métodos que possibilitam a implementação dos requisitos acima referidos.

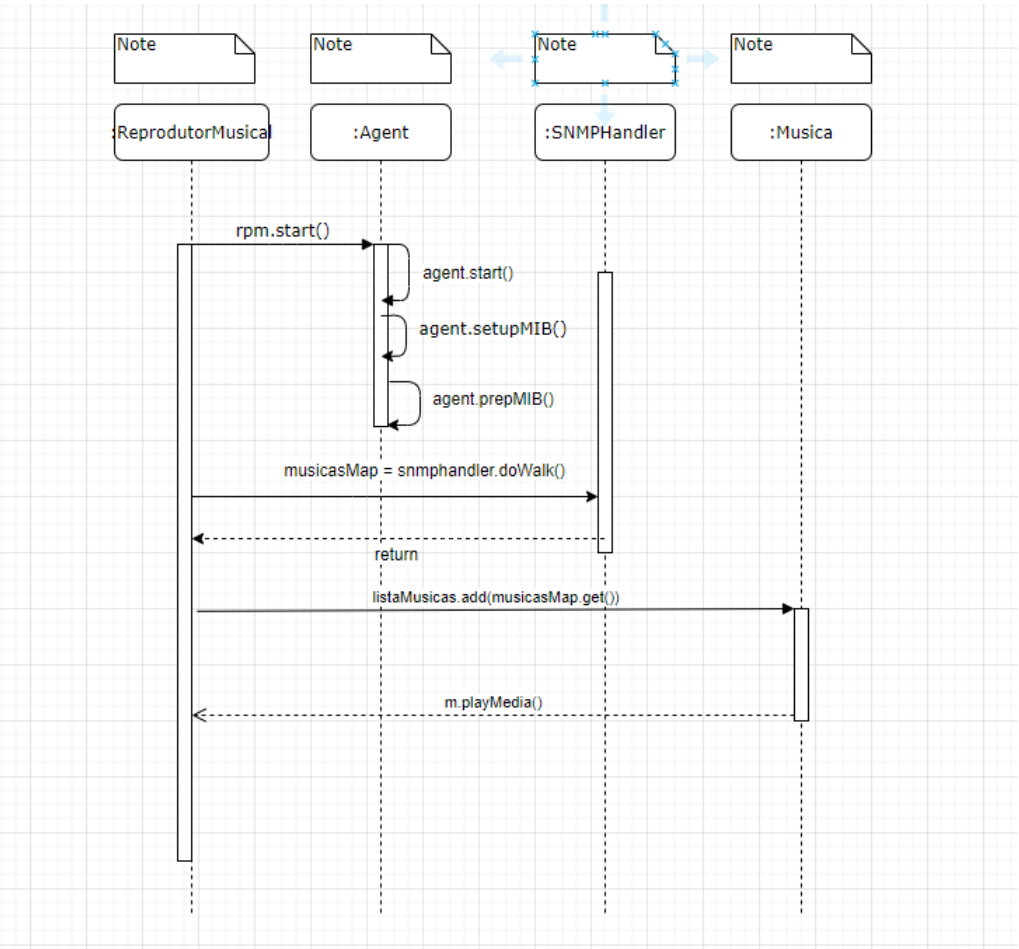
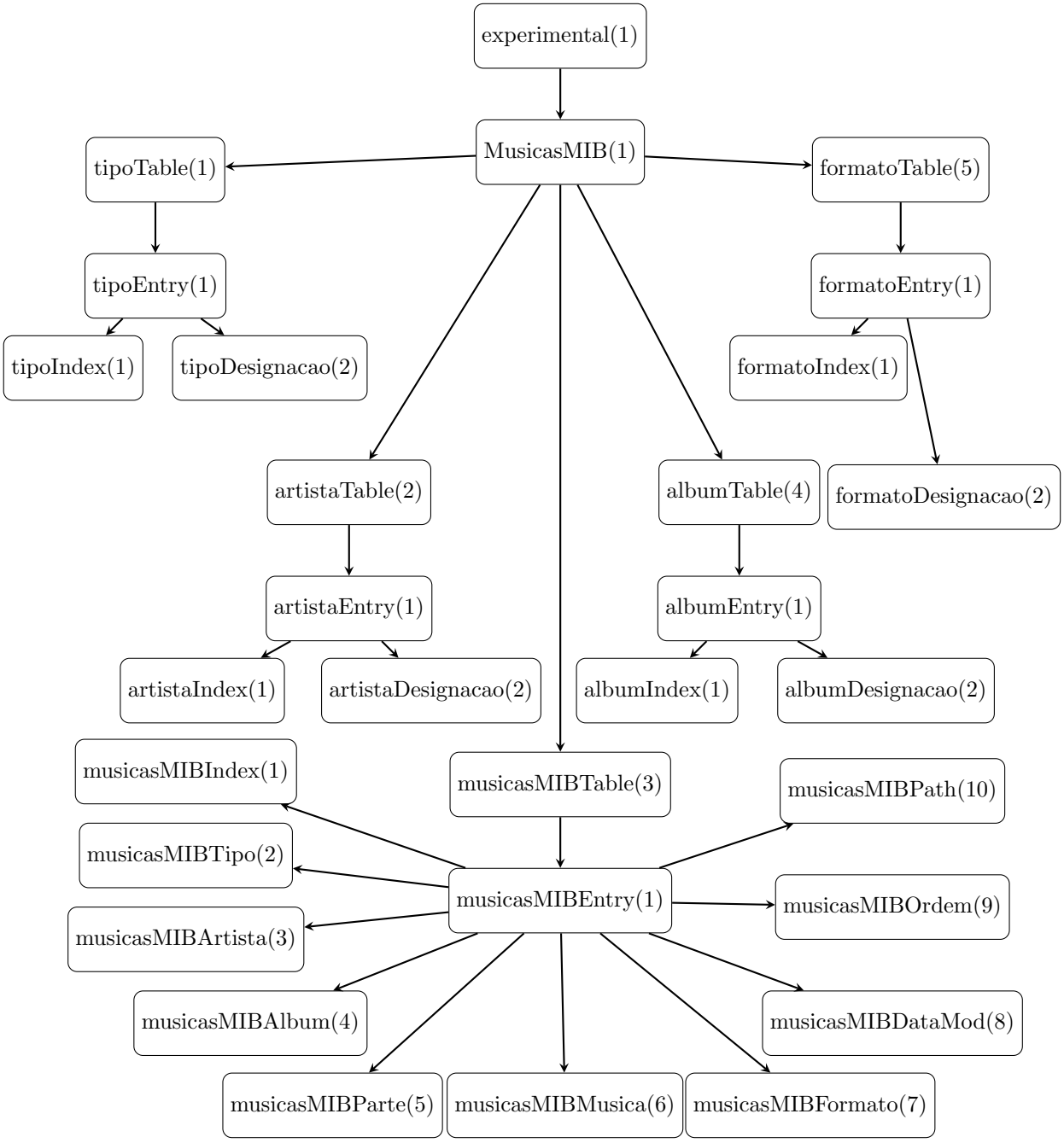


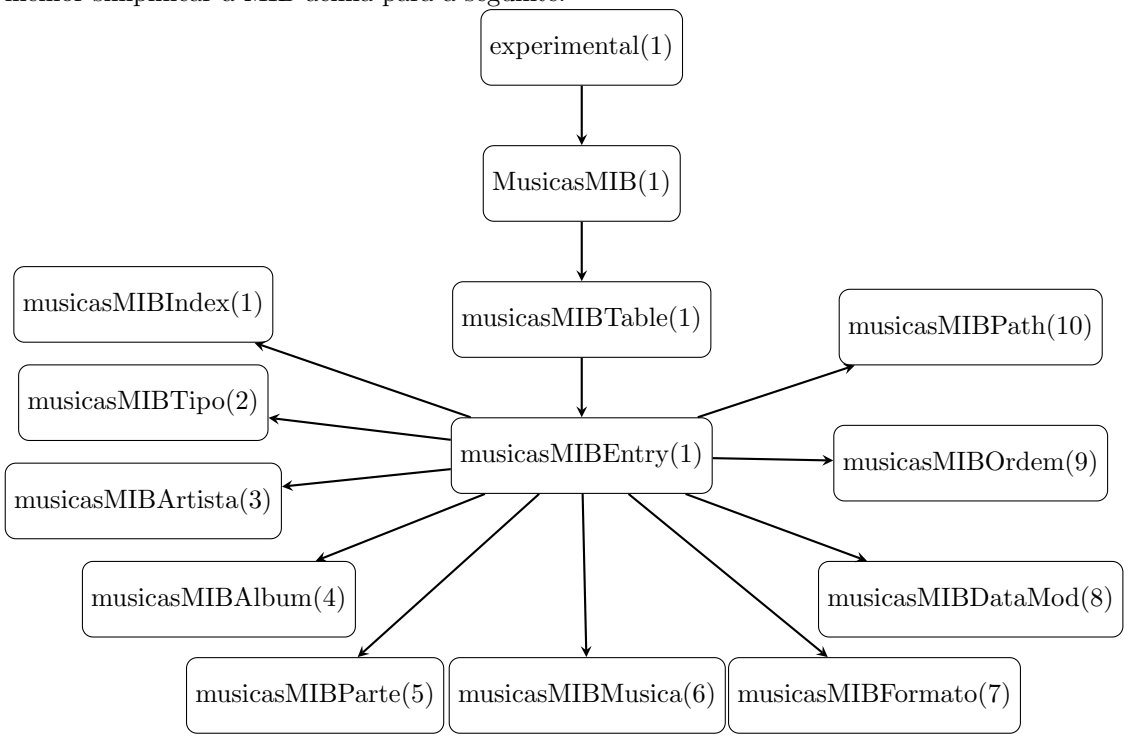
Figure 1: Diagrama de sequência

2 Fase B- Definição da MIB para servidor musical

Para definir a MIB do servidor musical temos de ter em conta os requisitos escolhidos visto que os objectos que vamos criar irão depender destes.
Como tal decidimos que a nossa MIB iria ser da seguinte forma:


experimental(1)
MusicasMIB(1)
tipoTable(1)
formatoTable(5)
tipoEntry(1)
formatoEntry(1)
tipoIndex(1)
tipoDesignacao(2)
formatoIndex(1)
formatoDesignacao(2)
artistaTable(2)
albumTable(4)
artistaEntry(1)
albumEntry(1)
artistaIndex(1)
artistaDesignacao(2)
albumIndex(1)
albumDesignacao(2)
musicasMIBIndex(1)
musicasMIBTable(3)
musicasMIBPath(10)
musicasMIBTipo(2)
musicasMIBEntry(1)
musicasMIBOrdem(9)
musicasMIBArtista(3)
musicasMIBAlbum(4)
musicasMIBDataMod(8)
musicasMIBParte(5)
musicasMIBMusica(6)
musicasMIBFormato(7)

Porém á medida que resolvemos o trabalho chegamos á conclusão que seria melhor simplificar a MIB acima para a seguinte:



3 Fase C- Implementação e teste do protótipo do agente SNMP

Nesta fase tivemos que criar as classes **Agent.java** e **MusicasMIB.java** que em conjunto criam a tabela de acordo com os ficheiros lidos, sendo que para esta fase foi utilizada a API **snmp4j-agent** e **snmp4j-log4j**.
Para além dessas duas classes também definimos **Musicas.java**, que é a classe que define o objecto Musica, e **ReprodutorMusical.java** que é a classe principal do nosso projecto.
Começamos então por definir as paths relevantes

```
Path currentRelativePath = Paths.get("");
String path = currentRelativePath.toAbsolutePath().toString();
String musicpath=path+File.separator+"Music"+File.separator;
```

Figure 2: Definir as paths

NOTA: As músicas têm de estar na pasta Music.
De seguida arrancamos o agente na porta **6666** e populamos a **MIB** com os metadados referentes ás músicas incluídas na pasta **Music**.

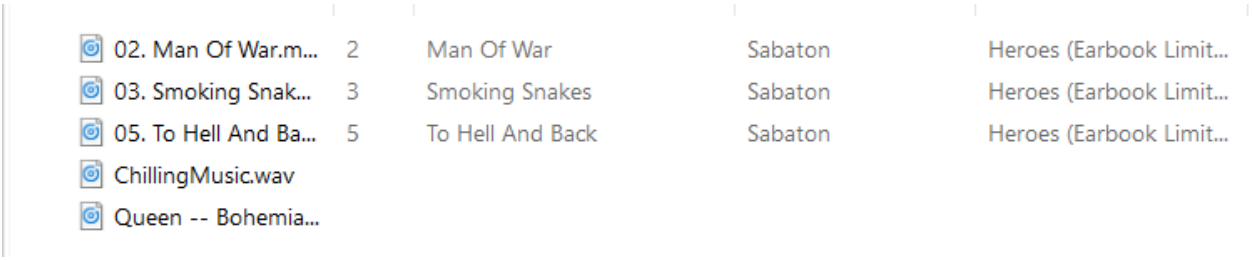


Figure 3: Conteúdo da pasta Music

```java
public void start(String path) throws IOException,InterruptedException{
    File folder=new File(path);
    HashMap<String,File> ficheiros=new HashMap<String,File>();
    ficheiros=listFilesForFolder(folder,ficheiros);
    int i=1;
    Agent agent=new Agent("0.0.0.0/6666");
    agent.start();
    agent.setupMIB();
    for(File fich:ficheiros.values()){
        Musica m=new Musica(fich,i,this);
        agent.prepMIB(m);
        i++;
    }
}
```

Figure 4: Arranque do agente

```java
public void setupMIB(){
    MOFactory moFactory=DefaultMOFactory.getInstance();
    this.mib=new MusicasMIB(moFactory);
    try{
        mib.registerMOs(getServer(),new OctetString("public"));
    }catch(DuplicateRegistrationException e){
        e.printStackTrace();
    }
}
public void prepMIB(Musica m){
    fillImg(this.mib.getmusicasMIBEntry(),m.getIndex(),m.getTipo(),m.getArtista(),m.getAlbum(),m.getParte(),m.getMusica(),
    m.getFormato(),m.getDataMod(),m.getOrdem(),m.getPath());
}
public void fillImg(MOTable img, int index, String tipo, String artista, String album ,String parte, String musica, String formato,
                    String data, int ordem, String path){
    img.addRow(img.createRow(new OID(Integer.toString(index)), new Variable[]{
        new Integer32(index),new OctetString(tipo),new OctetString(artista),new OctetString(album),new OctetString(parte),
        new OctetString(musica),new OctetString(formato),new OctetString(data),new Integer32(ordem),new OctetString(path)
    }));
}
```

Figure 5: Popular a MIB

Após populado podemos fazer um **snmpwalk** para verificar que foi tudo adicionado correctamente.

```
joao@DESKTOP-7UHPH4R:/mnt/c/Users/joaob/Desktop$ snmpwalk -m +MusicasMIB -v2c -c public localhost:6666 1.3.6.1.3.2020.1
MusicasMIB::musicasMIBIndex.1 = INTEGER: 1
MusicasMIB::musicasMIBIndex.2 = INTEGER: 2
MusicasMIB::musicasMIBIndex.3 = INTEGER: 3
MusicasMIB::musicasMIBIndex.4 = INTEGER: 4
MusicasMIB::musicasMIBIndex.5 = INTEGER: 5
MusicasMIB::musicasMIBTipo.1 = STRING: "Heavy Metal / Power Metal"
MusicasMIB::musicasMIBTipo.2 = STRING: "Heavy Metal / Power Metal"
MusicasMIB::musicasMIBTipo.3 = STRING: "N/A"
MusicasMIB::musicasMIBTipo.4 = STRING: "Heavy Metal / Power Metal"
MusicasMIB::musicasMIBTipo.5 = STRING: "N/A"
MusicasMIB::musicasMIBArtista.1 = STRING: "Sabaton"
MusicasMIB::musicasMIBArtista.2 = STRING: "Sabaton"
MusicasMIB::musicasMIBArtista.3 = STRING: "N/A"
MusicasMIB::musicasMIBArtista.4 = STRING: "Sabaton"
MusicasMIB::musicasMIBArtista.5 = STRING: "N/A"
MusicasMIB::musicasMIBAlbum.1 = STRING: "Heroes (Earbook Limited Edition) CD 1"
MusicasMIB::musicasMIBAlbum.2 = STRING: "Heroes (Earbook Limited Edition) CD 1"
MusicasMIB::musicasMIBAlbum.3 = STRING: "N/A"
MusicasMIB::musicasMIBAlbum.4 = STRING: "Heroes (Earbook Limited Edition) CD 2"
MusicasMIB::musicasMIBAlbum.5 = STRING: "N/A"
MusicasMIB::musicasMIBParte.1 = STRING: "CD 1"
MusicasMIB::musicasMIBParte.2 = STRING: "CD 1"
MusicasMIB::musicasMIBParte.3 = STRING: "CD0"
MusicasMIB::musicasMIBParte.4 = STRING: "CD 2"
MusicasMIB::musicasMIBParte.5 = STRING: "CD0"
MusicasMIB::musicasMIBMusica.1 = STRING: " Smoking Snakes"
MusicasMIB::musicasMIBMusica.2 = STRING: " To Hell And Back"
MusicasMIB::musicasMIBMusica.3 = STRING: "ChillingMusic"
MusicasMIB::musicasMIBMusica.4 = STRING: " Man Of War"
MusicasMIB::musicasMIBMusica.5 = STRING: "Queen -- Bohemian Rhapsody"
```

Figure 6: Parte 1 do snmpwalk

```
MusicasMIB::musicasMIBFormato.1 = STRING: "mp3"
MusicasMIB::musicasMIBFormato.2 = STRING: "mp3"
MusicasMIB::musicasMIBFormato.3 = STRING: "wav"
MusicasMIB::musicasMIBFormato.4 = STRING: "mp3"
MusicasMIB::musicasMIBFormato.5 = STRING: "mp3"
MusicasMIB::musicasMIBDataMod.1 = STRING: "2020-02-03 at 11:49:01 WET"
MusicasMIB::musicasMIBDataMod.2 = STRING: "2020-02-03 at 11:49:02 WET"
MusicasMIB::musicasMIBDataMod.3 = STRING: "2020-02-03 at 11:49:02 WET"
MusicasMIB::musicasMIBDataMod.4 = STRING: "2020-02-03 at 11:49:02 WET"
MusicasMIB::musicasMIBDataMod.5 = STRING: "2020-02-03 at 11:49:02 WET"
MusicasMIB::musicasMIBOrdem.1 = INTEGER: 3
MusicasMIB::musicasMIBOrdem.2 = INTEGER: 5
MusicasMIB::musicasMIBOrdem.3 = INTEGER: 0
MusicasMIB::musicasMIBOrdem.4 = INTEGER: 2
MusicasMIB::musicasMIBOrdem.5 = INTEGER: 0
MusicasMIB::musicasMIBPath.1 = STRING: "C:\\Users\\joaob\\Desktop\\GR TP3\\Music\\03. Smoking Snakes.mp3"
MusicasMIB::musicasMIBPath.2 = STRING: "C:\\Users\\joaob\\Desktop\\GR TP3\\Music\\05. To Hell And Back.mp3"
MusicasMIB::musicasMIBPath.3 = STRING: "C:\\Users\\joaob\\Desktop\\GR TP3\\Music\\ChillingMusic.wav"
MusicasMIB::musicasMIBPath.4 = STRING: "C:\\Users\\joaob\\Desktop\\GR TP3\\Music\\02. Man Of War.mp3"
MusicasMIB::musicasMIBPath.5 = STRING: "C:\\Users\\joaob\\Desktop\\GR TP3\\Music\\Queen -- Bohemian Rhapsody.mp3"
```

Figure 7: Parte 2 do snmpwalk

Com a **MIB** populada decidimos que, de forma a minimizar a sobrecarga da rede, era melhor efetuar um **snmpwalk** para recuperar todas as informações relativas ás músicas.
Isso é feito através da classe **SNMPHandler.java**.

```java
public class SNMPHandler {
    private String ip="localhost";
    private int port=6666;
    private String community = "public";
    private int snmpVersion = SnmpConstants.version2c;
    private CommunityTarget comtarget;
    private TransportMapping transport;

    public SNMPHandler() throws IOException {
        transport = new DefaultUdpTransportMapping();
        transport.listen();
        // Create Target Address object
        comtarget = new CommunityTarget();
        comtarget.setCommunity(new OctetString(community));
        comtarget.setVersion(snmpVersion);
        comtarget.setAddress(new UdpAddress(ip + "/" + port));
        comtarget.setRetries(2);
        comtarget.setTimeout(1000);

    }
```

Figure 8: Definição do SNMPHandler

```java
public Map<String, String> snmpWalk(String tableOid, CommunityTarget target) throws IOException {
    Map<String, String> result = new TreeMap<>();
    TransportMapping transport = new DefaultUdpTransportMapping();
    Snmp snmp = new Snmp(transport);
    transport.listen();
    TreeUtils treeUtils = new TreeUtils(snmp, new DefaultPDUFactory());
    List<TreeEvent> events = treeUtils.getSubtree(target, new OID(tableOid));
    if (events == null || events.size() == 0) {
        System.out.println("Error: Unable to read table...");
        return result;
    }
    for (TreeEvent event : events) {
        if (event == null) {
            continue;
        }
        if (event.isError()) {
            System.out.println("Error: table OID [" + tableOid + "] " + event.getErrorMessage());
            continue;
        }
        VariableBinding[] varBindings = event.getVariableBindings();
        if (varBindings == null || varBindings.length == 0) {
            System.out.println("Vazio no "+ tableOid);
            continue;
        }
        for (VariableBinding varBinding : varBindings) {
            if (varBinding == null) {
                continue;
            }
            String v=varBinding.getOid().toString();
            String[] sub=v.split("\\.");
            String s=sub[sub.length-1];
            result.put("." + s, varBinding.getVariable().toString());
        }
    }
    snmp.close();
    return result;
}
```

Figure 9: Definição do snmpwalk

```java
public HashMap<Integer,Musica> doWalk(String index,String tipo, String artista, String album, String parte, String titulo,
String formato, String data, String ordem, String path) throws IOException{
    Map<String,String> indexes=snmpWalk(index,this.comtarget);
    Map<String,String> tipos=snmpWalk(tipo,this.comtarget);
    Map<String,String> artistas=snmpWalk(artista,this.comtarget);
    Map<String,String> albuns=snmpWalk(album,this.comtarget);
    Map<String,String> partes=snmpWalk(parte,this.comtarget);
    Map<String,String> titulos=snmpWalk(titulo,this.comtarget);
    Map<String,String> formatos=snmpWalk(formato,this.comtarget);
    Map<String,String> datas=snmpWalk(data,this.comtarget);
    Map<String,String> ordens=snmpWalk(ordem,this.comtarget);
    Map<String,String> paths=snmpWalk(path,this.comtarget);
    HashMap<Integer,Musica> musicas=new HashMap<>();

    for(String s: indexes.keySet()){
        int i=Integer.parseInt(indexes.get(s));
        String type=tipos.get(s).toString();
        String artist=artistas.get(s).toString();
        String al=albuns.get(s).toString();
        String part=partes.get(s).toString();
        String title=titulos.get(s).toString();
        String format=formatos.get(s).toString();
        String d=datas.get(s).toString();
        //String order=ordens.get(s).toString();
        int order=Integer.parseInt(ordens.get(s));
        String p=paths.get(s).toString();
        Musica m=new Musica(i,type,artist,al,part,title,format,d,order,p);
        musicas.put(i,m);
    }
    return musicas;
}
```

Figure 10: Leitura de toda a MIB

Após termos toda as informações relativas ás músicas no sistema podemos procurar músicas pelo título, autor, álbum, criar Playlists e reproduzir uma música/playlist. Para tal definimos a função **playMedia** que através do **path** da música e da API **javafx** consegue reproduzir uma música.

```java
public void playMedia(String filepath,ReprodutorMusical rpm,String title){
    try{
        File musicpath=new File(filepath);
        if(musicpath.exists()){
            Media media = new Media(musicpath.toURI().toString());
            MediaPlayer mediaPlayer = new MediaPlayer(media);
            BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in));
            String op = "";
            mediaPlayer.play();
            while(op!="-1"){
                rpm.printMusica();
                op = teclado.readLine();
                switch(op){
                    case "1":
                        if(mediaPlayer.getStatus()==MediaPlayer.Status.PLAYING) {
                            System.out.println("Music paused "+title);
                            mediaPlayer.pause();
                        }else{
                            System.out.println("Error-Music not playing");
                        }
                        break;
                    case "2":
                        if(mediaPlayer.getStatus()==MediaPlayer.Status.PLAYING){
                            System.out.println("Error-Music already Playing");
                        }else{
                            System.out.println("Starting music "+title);
                            mediaPlayer.play();
                        }
                        break;
```

Figure 11: Definição da função playMedia parte 1

```
                    case "3":
                        if(mediaPlayer.getStatus()==MediaPlayer.Status.PLAYING){
                            System.out.println("Stopping music "+title);
                            mediaPlayer.stop();
                        }else{
                            System.out.println("Error-Music not playing");
                        }
                        break;
                    case "0":
                        System.out.println("Skipping music "+title);
                        if(mediaPlayer.getStatus()==MediaPlayer.Status.PLAYING) mediaPlayer.stop();
                        op="-1";
                        break;
                    default:
                        System.out.println("     Comando desconhecido!");
                }
            }
        }else{
            System.out.println("Can't find File: "+filepath);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
  }
}
```

Figure 12: Definição da função playMedia parte 2

Para cumprir os requisitos propostos de garantir logs do sistema, e caching das músicas lidas pelo foram criadas três funções que se seguem. A função **writeLogs** irá escrever no ficheiro logs um timestamp de ações mais relevantes efetuadas pelo utilizador no sistema. As funções **writeCache** e **readCache** escrevem e lêem respetivamente num ficheiro cache.ser todas as musicas lidas previamente pelo **doWalk**.

```
public void writeLogs(String path, String logs,FileHandler fh){
    Logger logger = Logger.getLogger("MyLogs");
    logger.setUseParentHandlers(false);
    try {
        // This block configure the logger with handler and formatter
        logger.addHandler(fh);
        SimpleFormatter formatter = new SimpleFormatter();
        fh.setFormatter(formatter);

        // the following statement is used to log any messages
            logger.info(logs);

    } catch (SecurityException e) {
      e.printStackTrace();
    }
}
```

Figure 13: Definição da função writeLogs

```java
public void writeCache(String path, HashMap<Integer, Musica> musicasMap) {
    try {
        // write object to file
        FileOutputStream fos = new FileOutputStream(path +File.separator+"cache.ser"+File.separator);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(musicasMap);
        oos.close();

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public HashMap<Integer, Musica> readCache(String path) {
    HashMap<Integer, Musica> musicasMap = new HashMap<>();
    try {
        // read object from file
        FileInputStream fis = new FileInputStream(path + File.separator+"cache.ser"+File.separator);
        ObjectInputStream ois = new ObjectInputStream(fis);
        musicasMap = (HashMap<Integer, Musica>) ois.readObject();
        ois.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
    return musicasMap;
}
```

Figure 14: Definição das funções writeCache e readCache

Por fim também construímos uma interface de texto para ajudar o utilizador a navegar a aplicação.

4 Fase F- Construção do manual de utilização

Ao correr a aplicação irá aparecer o seguinte menu.

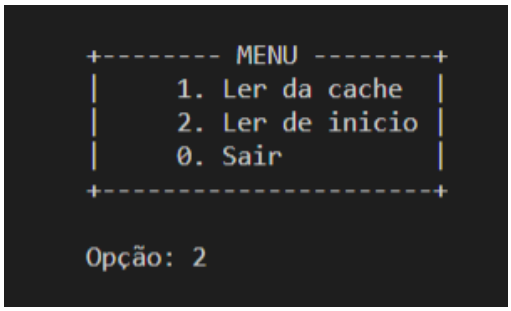


Figure 15: Menu inicial

Se escolher a opção 1, o programa irá ler a partir de o ficheiro **cache.ser** que foi criado na última vez que se leu de início previamente. É apresentado o

mesmo menu em qualquer um dos dois casos de leitura.

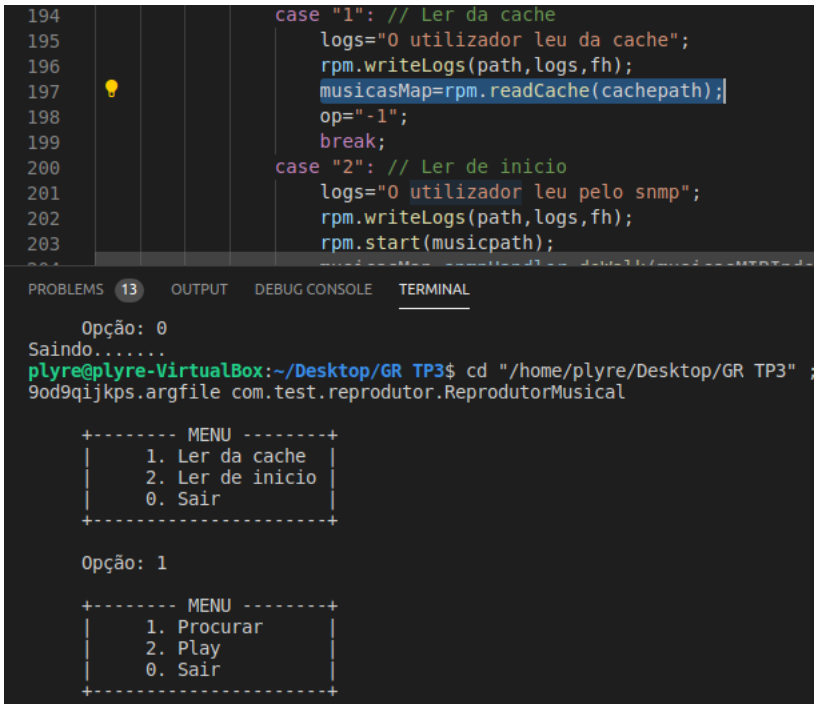


Figure 16: Menu principal após leitura do ficheiro cache.ser

Se escolher a opção 2, o programa irá ler de inicio todos os ficheiros da pasta **Music** e irá popular a MIB.
De seguida teremos um menu onde podemos procurar por músicas ou reproduzir músicas.

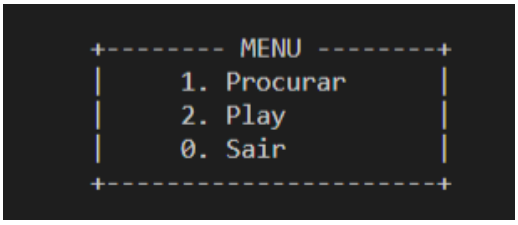


Figure 17: Menu principal

Se escolher a opção 1, irá aparecer um menu onde é possível escolher o tipo de procura que se pode fazer, por exemplo listar todas as músicas no sistema e listar todas as músicas de um certo artista.

```
    Opção: 1

    +--------- Procura ---------+
    |     1. Todas as músicas   |
    |     2. Procura por titulo |
    |     3. Procura por artista|
    |     4. Procura por album  |
    |     5. Procura por tipo   |
    |     0. Voltar             |
    +---------------------------+

    Opção: 1
Index: 1-> Smoking Snakes->Sabaton->Heroes (Earbook Limited Edition) CD 1->Heavy Metal / Power Metal
------------------------------------------------------------------------
Index: 2-> To Hell And Back->Sabaton->Heroes (Earbook Limited Edition) CD 1->Heavy Metal / Power Metal
------------------------------------------------------------------------
Index: 3->ChillingMusic->N/A->N/A->N/A
------------------------------------------------------------------------
Index: 4-> Man Of War->Sabaton->Heroes (Earbook Limited Edition) CD 2->Heavy Metal / Power Metal
------------------------------------------------------------------------
Index: 5->Queen -- Bohemian Rhapsody->N/A->N/A->N/A
------------------------------------------------------------------------
```

Figure 18: Listagem de todas as músicas

```
    +--------- Procura ---------+
    |     1. Todas as músicas   |
    |     2. Procura por titulo |
    |     3. Procura por artista|
    |     4. Procura por album  |
    |     5. Procura por tipo   |
    |     0. Voltar             |
    +---------------------------+

    Opção: 3
Indique o artista
sabat
Index: 1-> Smoking Snakes->Sabaton
------------------------------------------------------------------------
Index: 2-> To Hell And Back->Sabaton
------------------------------------------------------------------------
Index: 4-> Man Of War->Sabaton
------------------------------------------------------------------------
```

Figure 19: Todas as músicas de um artista

Ao escolher a opção 2 é possível reproduzir uma só música, dado o seu índice, e criar/reproduzir uma playlist.

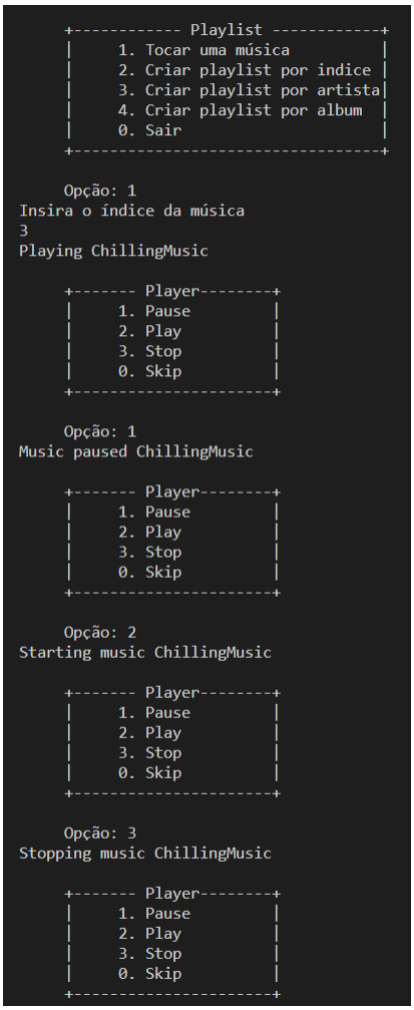


Figure 20: Reprodução de um ficheiro .wav

Ao reproduzir uma música é possível pausar a reprodução, recomeçar a reprodução de inicio ou onde pausou e é possível fazer skip.

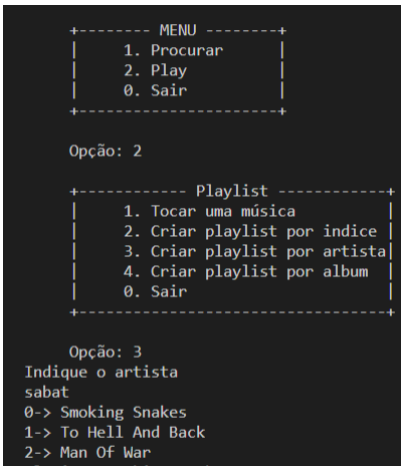


Figure 21: Criação de uma playlist através do artista

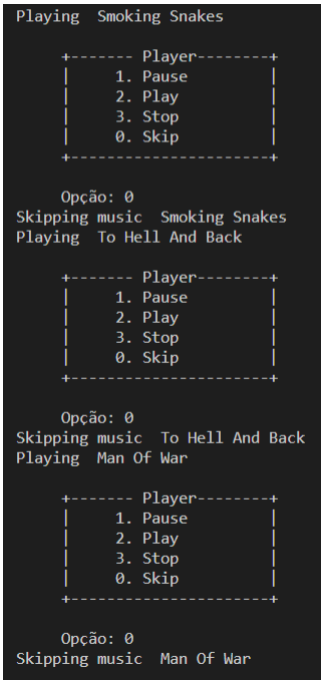


Figure 22: Reprodução da playlist

5 Outras considerações

A razão para não guardarmos o ficheiro música em **SNMP** deveu-se ao facto de não ser possível enviar o ficheiro por causa do tamanho limite de cada mensagem.