Projecto 2

Universidade de Aveiro

Rodrigo Rosmaninho, Eurico Dias,
João Trindade, Pedro Valério

Versão 1

Projecto 2

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

Rodrigo Rosmaninho, Eurico Dias,
João Trindade, Pedro Valério
(88802) r.rosmaninho@ua.pt, (72783) dias.eurico@ua.pt,
(89140) jatt@ua.pt, (88734) pedrovalerio@ua.pt

15 de junho de 2018

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O presente relatório foi elaborado com o propósito de descrever e analisar a elaboração e resultado do segundo projecto realizado no âmbito da unidade curricular de Laboratórios de Informática. Iniciando-se com uma apresentação do tema proposto, segue-se a estrutura do documento separada por capítulos representativos das componentes fundamentais do projecto desenvolvido.

O objectivo do projecto realizado foi a elaboração de um sistema que permite a composiçao de músicas com recurso a excertos de áudio, disponível através de um interface web. Este interface permite compôr músicas, ouvir composições de outras pessoas, carregar excertos para serem utilizados e descarregar músicas criadas. A informação de todas as músicas e excertos devem encontrar-se registados numa base de dados, que permita o acesso e modificação dos dados armazenados. A criação de músicas deve ser possível através de um programa gerador de excertos áudio, conforme a informação recebida sobre o tipo de música a criar.

Este documento encontra-se dividido em capítulos e secções descritivas das componentes referidas, sendo este o seu primeiro capítulo. No Capítulo 2, é descrito em detalhe o interface web criado, sendo as suas páginas explicadas separadamente, na Seção 2.1, Seção 2.2 e Seção 2.3. No Capítulo 3, é explicada a implementação da aplicação web e a forma como se interliga com os restantes componentes. No Capítulo 4, apresenta-se a organização do sistema de base de dados, bem como as suas funcionalidades. No Capítulo 5 é explicado o processo de criação de excertos musicais com base numa pauta num formato de dicionário, sendo o seu processo descrito em Seção 5.1, Seção 5.2, Seção 5.3 e Seção 5.4. Por último, no Capítulo 8, são apresentadas as conclusões do trabalho.

Capítulo 2

Interface Web

Em todas as páginas é possível alterar o avatar (foto) do utilizador clicando no seu nome no canto superior direito.
Nota: Em todas as páginas foi utilizado o Twitter Bootstrap (versão 4)

2.1 Lista de Músicas

A primeira página consiste numa lista de todas as músicas criadas até ao momento. As funcionalidades dinâmicas desta página são fornecidas pelo ficheiro **songlist.js**

A listagem é conseguida através de uma chamada ao endpoint **list/?type=songs**, que devolve um objeto JSON com um array de todas as músicas presentes no sistema. Por exemplo:


```
[
    {
      "name": "teste",
      "id": "8434efc3978369c2",
      "date": "2018-06-13 21:29:23Z23.227",
      "uses": 0,
      "path": "songs/8434efc3978369c2.wav",
      "votes": 1,
      "author": "r.rosmaninho"
    },
    {
      "name": "teste 2",
      "id": "4cc7daf0c5bc921a",
      "date": "2018-06-15 17:13:29Z29.465",
      "uses": 0,
      "path": "songs/4cc7daf0c5bc921a.wav",
      "votes": 0,
```

```
        "author": "r.rosmaninho"
    }
]
```

A informação de cada lista é depois adicionada à tabela presente em **songs.html**

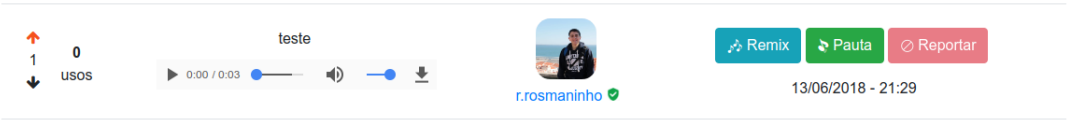

Figura 2.1: Aspeto de um item da tabela

É permitido ao utilizador votar na musica através de dois botões em forma de seta (um "para cima"e outro "para baixo"). Como se pode ver, as setas ficam coloridas para indicar que o utilizador já votou. Este processo acontece automáticamente a cada vez que o utilizador abre o website, necessitando, para isso, de receber informação sobre os votos efetuados pelo utilizador (proveniente da base de dados).

Para além das operações de voto, é ainda possível:

- Visualizar o número de usos (vezes que a música foi utilizada para gerar outras músicas).
- Visualizar o nome da música (escolhido pelo utilizador que a criou).
- Reproduzir e descarregar a música (através da tag *audio* do HTML). A música encontra-se armazenada no sistema de ficheiros da aplicação Web e é servida pelo **CherryPy**
- Visualizar o email (sem @ua.pt) do autor da música e o seu avatar. Se o autor fizer parte da equipa de desenvolvimento do website aparece um símbolo indicativo. Para além disso, é possivel passar o rato pelo nome para visualizar o *karma* do autor (total de upvotes + downvotes)
- Visualizar a data e hora (em UTC) da criação da música.
- Fazer *Remix* da música em questão (Alterar uma música existente e guardá-la como uma nova música). O utilizador é redirecionado para a página de criação de músicas, que é pre-populada com a pauta da música em questão
- Descarregar uma imagem representativa da pauta da música em questão em formato *.png* através do botão *Pauta*

- Reportar uma música à equipa de desenvolvimento (por linguagem inapropriada no título da música por exemplo). Esta ação adiciona uma entrada na tabela *reports* da base de dados. Não é possível reportar músicas da autoria de um membro da equipa.

É possivel ordenar a lista por nome da música, data de criação, número de votos, e escolher entre ordem ascendente ou descendente. Ao escolher um novo tipo de ordenação a página é recarragada com os seguintes parametros (por exemplo):

```
/songs?order=votos&asc_desc=1
```

Pelo que o JavaScript faz o *request* da lista ao API de forma diferente

A lista das músicas é apresentada de forma paginada. Apenas 10 músicas são apresentadas por página. As restantes ficam com o estilo **display: none;**. Este comportamento é gerido pelo JavaScript em **songlist.js** e é importante notar que este recurso externo foi usado como inspiração.

2.2 Lista de Excertos

A segunda página consiste numa lista de todos os excertos (samples) presentes no sistema até ao momento. As funcionalidades dinâmicas desta página são fornecidas pelo ficheiro **sampleslist.js**

Tal como na página de músicas, a listagem é conseguida através de uma chamada a um endpoint do api. Neste caso **list/?type=samples**, que devolve um objeto JSON com um array de todos os excertos presentes no sistema. O objeto correspondente a cada elemento é extremamente semelhante ao que foi visto na Seção 2.1:

A informação de cada lista é depois adicionada à tabela presente em **samples.html**

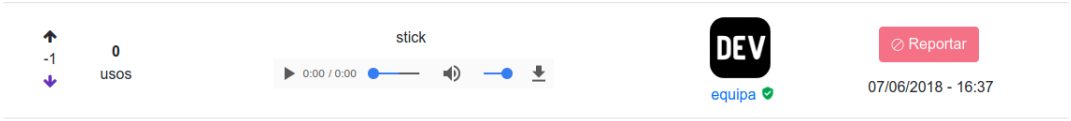

Figura 2.2: Aspeto de um item da tabela

Qualquer utilizador pode fazer upload de um novo excerto para a aplicação utilizando o *form* HTML disponível para o efeito. Apenas são aceites excertos em formato **.wav** e se o ficheiro for do tipo *stereo* é automaticamente convertido

para *mono* com a ajuda do comando bash *ffmpeg* No backend é feito o hash do conteúdo do ficheiro e a informação é guardada na base de dados.

Mais uma vez, é permitido ao utilizador votar no excerto através dos dois botões em forma de seta.

Para além destas operações, é ainda possível:

- Visualizar o número de usos (vezes que o excerto foi utilizado para gerar novas músicas).
- Visualizar o nome do excerto (escolhido pelo utilizador que fez o upload).
- Reproduzir e descarregar o excerto (através da tag *audio* do HTML). O excerto também se encontra armazenado no sistema de ficheiros da aplicação Web e é servido pelo **CherryPy**
- Visualizar o email (sem @ua.pt) do utilizador que fez upload e o seu avatar. Se o autor fizer parte da equipa de desenvolvimento do website aparece um símbolo indicativo. Para além disso, é possivel passar o rato pelo nome para visualizar o *karma* do autor (total de upvotes + downvotes)
- Visualizar a data e hora (em UTC) do upload do excerto.
- Reportar um excerto à equipa de desenvolvimento (por uso de sons ou linguagem inapropriada, por exemplo). Esta ação adiciona uma entrada na tabela *reports* da base de dados. Não é possível reportar excertos da autoria de um membro da equipa.

Também é, novamente, possivel ordenar a lista por nome do excerto, data de criação, número de votos, e escolher entre ordem ascendente ou descendente.

À semelhança da lista de músicas, a lista dos excertos é apresentada de forma paginada. Apenas 10 excertos são apresentados por página. Os restantes ficam com o estilo **display: none;**. Este comportamento é gerido pelo JavaScript em **samplelist.js**.

2.3 Criação de Música

Na página de criação de música é apresentada ao utilizador uma matriz de células, em que, a cada linha, corresponde, ainda, um botão para seleccionar um excerto e um botão para seleccionar efeitos.

O processo de criação de música, baseia-se em seleccionar excertos para utilizar e, por cada excerto, as vezes e frequência com que é reproduzida na música, seleccionando as células que correspondem a um batimento, finalmente, os efeitos desejados.

Quando a página é carregada, são recebidas e armazenadas as listas de efeitos, excertos e músicas da aplicação web, para serem utilizadas. São, também,

adicionadas 12 linhas vazias (por defeito), escrevendo o respectivo código html com recurso a funções *javascript*.

A selecção do excerto inicia-se carregando no botão de excerto, que abre um *popup* que contém a lista de excertos disponíveis numa primeira página e a lista de músicas criadas numa segunda, estando ambas disponíveis para a criação de música. Apesar de existir um botão para cada linha, todos abrem o mesmo *popup*, pelo que no momento em que este se abre, é registado qual botão o efectuou, para saber qual das linhas deve alterar. Este *popup*, permite reproduzir os excertos e músicas antes de seleccionar e, também, procurar um elemento específico, escrevendo parte do nome. Todos excertos escolhidos são armazenados numa lista, em que cada índice corresponde a uma linha

Para especificar os batimentos associados a cada excerto, cada linha dispõe de 16 células, que se podem activar, indicando que, nesse batimento, o excerto/-música escolhido deve começar a ser reproduzido. Só se podem activar células numa dada linha, caso haja um excerto associado. Os dados de todas as células são guardados numa matriz.

Na selecção de efeitos, existe, para cada linha um botão que abre uma lista *drop-down* com todos efeitos disponíveis. A cada efeito está associada uma caixa de selecção, para ser possível escolher vários efeitos para cada excerto. Esta informação é guardada numa lista de listas, na qual , a cada índice, corresponde a lista de efeitos a utilizar.

Para além das funcionalidades descritas, existe, ainda um botão que adiciona uma nova linha vazia, uma marca *input* para especificar os batimentos por minuto e outra para dar um nome à música. No momento de submissão, é gerado um objecto json, com recurso às três listas. Este é enviado para a aplicação web para ser gerada a música.

É, também, possível a alteração de uma música existente, situação na qual, em vez de ser criada uma matriz vazia, é preenchida com a informação do dicionário recebido da aplicação.

Capítulo 3

Aplicação Web

A aplicação web consiste num programa python (**api.py**) que serve conteúdos estáticos (html, css, js, imagens, etc), apresentando também métodos que permitem o fluxo de informação entre os diversos componentes do sistema. Permite o envio de informação proveniente da base de dados para o website e o registo de nova informação vinda da página.

Em particular, esta aplicação expõe as seguintes funções:

- /**list?type=songs** Quando invocado, devolve um array em JSON com a lista de todas as músicas presentes no sistema. Um exemplo do JSON pode ser visto na Seção 2.1. Permite os seguintes argumentos opcionais:
  - ⋆ **order** Especifica a ordem pela qual as músicas devem ser listadas
  - ⋆ **asc_desc** Especifica se é usada ordem ascendente ou descendente
- /**list?type=samples** Quando invocado, devolve um array em JSON com a lista de todos os excertos presentes no sistema. Permite os mesmos argumentos opcionais que a função anterior.
- /**list?type=effects** Quando invocado, devolve um array em JSON com a lista de todos os efeitos disponíveis para a criação de músicas.
- /**list?type=votes** Quando invocado, devolve um array em JSON com a lista de todos os votos efetuados pelo utilizador corrente.
- /**get?id=identificador** Permite obter um excerto ou uma música com base num identificador fornecido.
- /**getImage?id=identificador** Permite obter uma imagem da pauta com base num identificador fornecido.
- /**getKarma?user=email** Permite obter o karma de um utilizador.

- /**avatar?id=identificador** Permite obter um excerto ou uma música com base num identificador fornecido.
- /**Admins** Devolve a lista de utilizadores admins.
- /**report?id=identificador** Permite reportar uma música ou excerto com base no seu identificadores.
- /**put** Permite enviar a pauta de uma nova música para que esta seja criada (Método POST). A pauta é depois enviada para **Sound.py** para iniciar o processo de criação da música.
- /**newSample** Permite fazer upload de um novo excerto. (Método POST). É feito o hash e conversão de stereo para mono se aplicável. De seguida guarda-se o ficheiro no diretório e a sua informação na base de dados.
- /**newAvatar** Permite fazer upload de um novo avatar para o utilizador corrente (Método POST).
- /**vote?id=identificador&user=uid&points=1** Permite a um utilizador a emissão de um voto numa música. O campo **id** identifica a música, o campo **user** identifica o utilizador e o campo **points** especifica o número de pontos a atribuir (+1 ou -1).
- /**songgen?id=identificador** Permite obter a pauta de uma determinada música através do seu identificador.
- /**user** Permite obter o utilizador universal (UU) da pessoa que está a utilizar o website (sem @ua.pt).

Capítulo 4

Persistência

O armazenamento dos dados provenientes da aplicação, isto é, a persistência, é garantida por uma base de dados relacional criada em **SQLite3**. As tabelas existentes relacionam-se entre si atráves de relações de um para muitos (*one-to-many relationship*)/muitos para um (*many-to-one relationship*), sendo que esta depende da entidade que estamos a considerar para a descrição. As tabelas são:

- ***users***: aqui são armazenadas informações referentes às propriedades que um utilizador contém no contexto da aplicação, que inclui o identificador universal do utilizador (excetuando o sufixo de email "@ua.pt"), o nome do utilizador (se este quiser ser identificado por outro nome, como por exemplo o seu nome próprio), o número de votos associado (que funciona como "karma", que tem a função de diferenciar a qualidade das músicas dos utilizadores), e a foto de utilizador, que aparece em cada música criada ou excerto adicionado, por forma a identificar a pessoa em questão com maior facilidade.

- ***samples***: nesta tabela, são persistidas informações referentes aos excertos existentes no *backend* da aplicação, incluindo o identificador interno nas tabelas da base de dados do autor (chave estrangeira que referencia a tabela ***users***), os primeiros 16 carateres da síntese Message-Digest algorithm 5 (MD5) do conteúdo do excerto, que funcionam como identificador do excerto na aplicação, o nome do excerto, o número de vezes que um excerto foi utilizado, a data em que foi adicionado, os votos que cada excerto conseguiu angariar, e o caminho relativo à raíz do presente projeto, no sistema de ficheiros do servidor.

- ***songgen***: esta tabela armazena a pauta musical serializada a partir do JavaScript Object Notation (JSON) que é enviado pelo servidor da aplicação, e numa fase posterior, permite que qualquer música adicionada à aplicação seja editável (*remix* da música). Esta tabela inclui as chaves do JSON, que é devolvido após a criação da música, tal como foi recomendado pelo enunciado deste projeto. Esta entidade contém colunas que

armazenam a síntese do conteúdo das músicas (processo idêntico à síntese efetuada nos excertos), o número de batimentos por minuto (Beats per Minute (BPM)), e os arrays de excertos, efeitos e posicionamento destes ao longo da música. No que toca aos arrays, é usado o tipo de dados *blob* para a persistência, para facilidade na manipulação dos dados por parte dos outros segmentos de código deste projeto. Para a conversão em bytes destas estruturas de dados para a inserção na base de dados destes objetos ser válida, é usado o pacote **marshal**, que facilita a conversão do objeto para um **"objeto Marshal"**, representado em bytes, e vice-versa.

- ***songs***: esta entidade guarda todas as informações referentes às músicas já criadas (que não sejam inerentes à pauta musical, mas sim às suas propriedades no contexto da aplicação). Contém o id do autor (chave estrangeira referente ao id do utilizador na tabela *users*), a síntese do conteúdo da música (o identificador da música no contexto da aplicação, que também é uma chave estrangeira), o nome da música, a data em que foi adicionada, os votos que lhe foram atribuídos, os usos da música (uma música pode ser utilizada como excerto noutra música), e o caminho relativo à raíz do projeto.

- ***votes***: esta tabela guarda a informação de cada voto efetuado. Isto inclui o id do utilizador que fez o voto (chave estrangeira que refere *users*), o identificador *hash* da música (também esta uma chave estrangeira, referenciando *songgen*), o tipo de voto - 1 ou -1 - que depende se o voto é, respetivamente, positivo ou negativo, e a data em que o voto foi efetuado.

- ***reports***: aqui são armazenadas as informações acerca das denúncias feitas a uma música ou excerto. A função presente desta tabela apenas passa por uma denúncia básica que faz com que qualquer utilizador possa reportar qualquer ilícito ou obscenidade no conteúdo ou identificação dos elementos musicais, que posteriormente seria candidato a revisão pelos administradores da aplicação. Contém o identificador estrangeiro do utilizador que reportou o elemento, que referencia a tabela *users*, o identificador do elemento na aplicação (a síntese) e a data na qual a denúncia foi efetuada.

- ***effects***: nesta entidade, são armazenados apenas o nome do efeito, para se efetuar uma persistência inicial destes, para referência da sua existência na aplicação.

  As funções de acesso a esta base de dados foram incluídas no ficheiro **db.py**, na pasta **server**/. Estas funções têm prefixos *get* e *set*, para identificar se são funções que correspondem a *queries* de seleção ou de inserção/atualização de registos. Estas funções são maioritariamente protegidas a ataques de ***Structured Query Language (SQL) Injection***, recorrendo a *queries* parametrizáveis e, no caso de concatenação de *strings* para a construção de frases de execução SQL com colunas dinâmicas, existe validação ou verificação do nome das colunas recebidas.

Capítulo 5

Gerador de Músicas

O processo de criação de música é efectuado através de um programa escrito em linguagem Python, que utiliza as funcionalidades da biblioteca **wav** e **struct**. Este programa permite a criação de ficheiros WAVEform audio file format (WAV), utilizando outros ficheiros do mesmo tipo, cada um representante de uma faixa de música do ficheiro final. Caso a pauta assim o exija, também é possível a aplicação de efeitos a estas faixas, para conseguir uma maior variedade de resultados. Este processo divide-se em diferentes subprocessos executados sequencialmente.

5.1 Interpretação da pauta

No momento de criação de música, a aplicação web chama a função *init* deste programa que deverá receber um dicionário que contém informação sobre o tipo de música a criar, dividida pelas suas chaves, apresentadas na Seção 5.1. Antes da interpretação da pauta fornecida, é definida a frequência da música a 44100Hz.

- **bpm**-indica os batimentos por minuto da música a compôr.
- **samples**-lista contendo os identificadores de cada excerto que deve ser utilizado.
- **effects**-lista de listas de efeitos a aplicar aos excertos. Cada uma das listas desta chave indica um ou mais efeitos a aplicar ao excerto do ìndice correpondente na chave anterior.
- **music**-lista de listas representativa da pauta da música desejada. Cada índice desta chave corresponde a um batimento e, a cada batimento, corresponde a lista de excertos que devem começar a ser reproduzidos no mesmo, identificados por números que correspondem aos índices da chave **samples**.

A lista de excertos a serem utilizados contèm identificadores, logo, no início da função, o programa acede à base de dados, fornecendo o identificador, recebendo o caminho do ficheiro desejado. De seguida são, a partir da pauta, definidos os valores fundamentais do ficheiro, nomeadamente o número de faixas, a duração em segundos e duração em *frames* (encontrados a partir dos batimentos por minuto e da frequência).

Para conseguir manipular os ficheiros da forma pretendida, estes são lidos e guardados numa lista de *bytes* e, posteriormente numa lista de listas inteiros através da função *unpack* da biblioteca **struct**, em que cada lista contém a informação sonora de um dos excertos em formato de lista de inteiros, na qual cada índice corresponde a um *frame* de áudio. Deste modo, estes dados podem ser alterados. Por último, é adicionado ao número de *frames* o número de *frames* do maior excerto fornecido, para que não haja cortes de som no fim da música.

5.2 Aplicação de Efeitos

Após interpretada a pauta fornecida, segue-se a aplicação de efeitos a cada uma das listas da lista de excertos obtida do processo anterior. Isto é efectuado iterando sobre a lista de listas e aplicando o efeito correspondente (especificado na pauta) a cada uma., com recurso a funções específicas para cada efeito. Os efeitos disponíveis encontram-se na Seção 5.2.

- **eco**-cria um efeito de eco no excerto, atingido concatenando o mesmo excerto, começando em momentos diferentes e progressivamente mais baixo. Este efeito aumenta, também o tamanho do excerto em 50%.
- **fade**-esta função é utilizada para criar vários tipos de *fade*, dependendo do seu segundo argumento, que especifica qual o tipo desejado. Todos estes tipos atingem-se multiplicando todos os *frames* do excerto por um valor variável entre 0 e 1, calculado com uma de duas funções.
  - ⋆ **in**-Utiliza a função *linear* linear com declive positivo para aumentar gradualmente o volume.
  - ⋆ **out**-Utiliza a função *linear* linear com declive negativo para diminuir gradualmente o volume.
  - ⋆ **inout**-Utiliza a função **branch** módulo que aplica *fade-in* até metade do excerto e *fade-out* depois.
  - ⋆ **inout**-Utiliza a função **branch** módulo que aplica *fade-out* até metade do excerto e *fade-in* depois.
- **reverse**-inverte o excerto.

5.3 Composição da música

Após o programa obter todos os excertos separados e com os seus respectivos efeitos, procede à composição da música. Isto é conseguido, percorrendo todos

os *frames* da música final (calculados previamente) e perceber o que deve ser escrito. Inicialmente, são inicializadas duas listas vazias com o comprimento igual ao número de excertos. Uma destas listas (*samp_i*) será utilizada para guardar o índice do último *frame* escrito de cada excerto, a outra (*track_frame*) irá conter o valor inteiro do *frame* de cada excerto nos índices determinadaos pela anterior. Com a inicialização concluída, inicia-se o ciclo que percorrerá os *frames*. A cada iteração em que o valor do *frame* actual corresponde a uma nova batida, são percorridos, também, todos os excertos e, por cada um, é analisada a pauta. Se o índice (*tr*) do excerto analisado estiver contido na batida, o valor de *samp_i[tr]*) é posto a zero, de modo a reproduzir do início o excerto. Quando definidos todos os valores de *samp_i*, ainda em cada excerto, os valores de *track_frame* são definidos com base nos anteriores. Se *samp_i[tr]* for menor que o comprimento total do excerto, *track_frame[tr]* terá o valor do *frame* do excerto nesse índice, caso contrário, significará que o excerto chegou ao fim, pelo que o valor deverá ser zero. No final de cada iteração de batida, é feita a média dos valores de *track_frame* diferentes de zero e adicionados a uma variável em formato de *bytes* com recurso à função *pack* da biblioteca **struct**.

5.4 Escrita do ficheiro

Após a obtenção dos dados da música, estes são gravados num ficheiro, com um nome obtido através de uma síntese md5 feita aos dados obtidos, da qual são aproveitados os prmeiros 16 bytes. Este nome será o identificador da música que será devolvido à aplicação web para armazenar na base de dados.

Capítulo 6

Geração da imagem da pauta

A geração da matriz representativa da pauta musical foi criada em Python, com recurso à package ***pillow***. A função recebe o identificador de música associado (a hash MD5) e recebe os dados num objeto JSON, que contém a pauta. Considerando cada setor da matriz da música como uma seção de píxeis quadrada, de lado fixo, e com a informação da existência de excertos num dado setor da matriz, é possível colorar corretamente cada uma destas seções. Associando as linhas de cada uma destas seções ao array de excertos da pauta e as colunas ao array da música, basta passar por cada um dos pares de variáveis, (x, y), que representam as coordenadas do píxel na imagem, para obter a imagem de uma matriz representativa básica, gerando-a píxel a píxel. A imagem é retornada como um objeto **Image**.
A Figura 6.1 mostra um exemplo de uma imagem previamente gerada desta matriz.

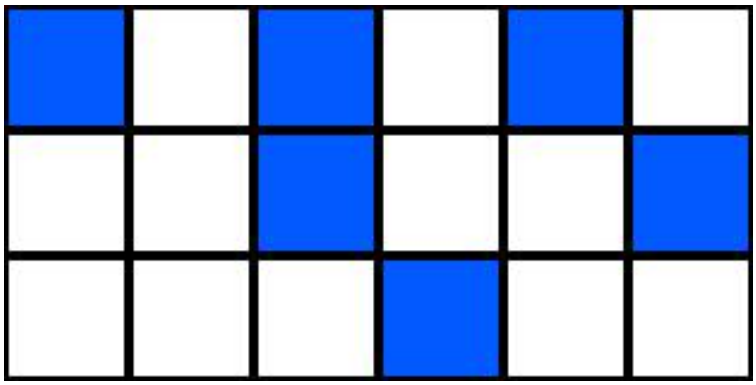

Figura 6.1: Exemplo de uma imagem da matriz representativa.

Capítulo 7

Testes

Para garantir o bom funcionamento do programa, foram efectuados diversos testes unitários aos diferentes programas.

Na criação, foram testados os efeitos que podem ser adicionados aos excertos. Como as funções de efeitos recebem listas de *frames*, para as testar, forneceram-se listas de poucos valores estratégicos de modo a garantir aspectos como o comportamento das funções com listas vazias ou com valores iguais. Testaram-se, ainda, as funções linear e módulo, testando com abcissas nos extremos do intervalo e valores médios.

No que toca a testes unitários ao acesso à base de dados, foram efetuados testes à integridade das *queries* de *get*. Todas as funções deste tipo foram testadas, sendo as mais gerais (como **getSamples()**, **getSongs()**, que são as funções sem argumentos e **getEffects()**) apenas para o caso de existir um erro interno na base de dados, sendo que o *output* irá ser sempre muito variável, dependendo dos dados que forem introduzidos. Para o resto das funções, averigou-se algumas variações entre parâmetros, e parâmetros que não irão ser geralmente enviados, ou que retornem erros previsíveis (como **ELEMENT_NOT_EXISTENT** ou **PARAM_ERROR**). Da inserção de parâmetros válidos nestes testes, esperam-se *outputs* previsíveis.

Capítulo 8

Conclusões

Todas as funcionalidades pedidas funcionam e os testes unitários passam sem erros.

Divisão do Trabalho

- **Rodrigo Rosmaninho - 25%**
  - ⋆ Aplicação Web (CherryPy - api.py)
  - ⋆ JavaScript das páginas de lista de músicas e excertos
  - ⋆ Ajuda na base de dados
  - ⋆ Ajuda nas páginas HTML e create.js
- **Eurico Dias - 25%**
  - ⋆ Script de criação da base de dados (proj2db.sql)
  - ⋆ Funções python de leitura e escrita na base de dados (db.py)
  - ⋆ Gerador de imagens (genImage.py)
  - ⋆ Ajuda na aplicação web (CherryPy - api.py)
- **João Trindade - 25%**
  - ⋆ Páginas HTML
  - ⋆ CSS (main.css)
  - ⋆ Ajuda no Javascript
- **Pedro Valério - 25%**
  - ⋆ Gerador de Músicas (Sound.py)
  - ⋆ Javascript da página de geração de músicas
  - ⋆ Função de tratar e guardar informação da música na DB
  - ⋆ Ajuda na página HTML create.html

Acrónimos

WAV WAVEform audio file format

MD5 Message-Digest algorithm 5

JSON JavaScript Object Notation

SQL Structured Query Language

BPM Beats per Minute