



UNIVERSIDADE DO MINHO
**Mestrado Integrado em Engenharia
Informática**
*Processamento e Representação de
Informação*



Arquivo Musical Digital

Grupo 14

Alexandre Teixeira — A73547

António Chaves — A75870

Miguel Guimarães — A66822

Braga, Janeiro de 2019

Resumo

O presente documento consolida todo o processo de desenvolvimento e utilização da aplicação desenvolvida no âmbito das UCS conjuntas Processamento e Representação de Informação e Gramáticas na Compreensão de Software.

Em primeira instância, será analisado o contexto da solução no enquadramento do problema proposto, explicitando claramente todas as decisões tomadas, seguido de um conjunto de comentários relativos à implementação final e possíveis melhoramentos a incluir numa versão posterior da aplicação.

De seguida, serão analisadas as soluções propostas para os enunciados de cada uma das UCS, com a explicação de todas as decisões tomadas, servindo também de manual de utilização.



Conteúdo

Contextualização do Problema	2
1 iBanda	3
1.1 Model	3
1.1.1 Article	3
1.1.2 Event	3
1.1.3 User	3
1.1.4 Works	4
1.1.5 Logs	4
1.2 Controller	4
1.3 Utilizadores	5
1.4 Rotas	5
1.4.1 User	6
1.4.2 Admin	6
1.5 API	7
1.6 Ingestão de SIP	8
1.7 View e jQuery	8
2 json2Mongo	12
2.1 Gramática	12
2.2 Java	14
2.2.1 Class antHandler.java	14
2.2.2 Main.java	14
2.3 Compilação e Execução	14
2.4 DSLs	14
2.4.1 Evento	14
2.4.2 Notícia	14
Conclusão	15
Anexos	16

Lista de Tabelas

Lista de Figuras

1	Home Page	8
2	Modal de Registo	9
3	Events from Admin	9
4	News from Admin	10
5	Logs from Admin	10
6	User Page	11



Contextualização do Problema

A proposição formalizada no enquadramento da UC de PRI sugeria a implementação de um aplicação WEB capaz de servir de plataforma de partilha de informação de interesse relativo a bandas filarmónicas, com base no Modelo de Referencia **OAIS**. A plataforma deve possuir um arquivo de obras musicais e partituras e garantir a persistência dos dados, através da ingestão de **SIP**. Além disso, é necessária atenção à criação de eventos e notícias, suportando pelos utilizadores do tipo produtor. Além disso é feita uma análise, muito simplista dos *Logs*, onde estes apenas atentão aos *Login* e *Logout*.

Para a UC de GCS, o grupo decidiu, para além da definição de uma DSL relativa aos eventos e notícias acima mencionados, criar uma aplicação Java/AntLR que, dado um ficheiro .json, preenchesse a respetiva coleção de uma base de dados Mongo. Esta poderá vir a ser útil no pré-povoamento da aplicação.



1 iBanda

1.1 Model

Os scripts derivados de Model incluem toda a informação dos Schemas representativos, dos elementos presentes nas coleções da base de dados.

1.1.1 Article

```
var articleSchema = new Schema({
  date : {type: String, required: true},
  title : {type: String, required: true},
  content : {type: String, required: true},
  author : {type: String, required: true}
})
```

1.1.2 Event

```
var EventSchema = new Schema({
  local: {type: String, required: true},
  theme: {type: String, required: true},
  description: {type: String},
  date: {type: String, required: true},
  hour: {type: String, required: true},
  duration: {type: Number, required: true},
  interested: {type: [String]}
})
```

1.1.3 User

```
var UserSchema = new Schema({
  email: {type: String, required: true, unique: true},
  password: {type: String, required: true},
  name: {type: String, required: true},
  utype: {type: Number, required: true},
  valid: {type: Boolean, required: true}
})
```



1.1.4 Works

```
var InstrSchema = new Schema({
  name:{type:String},
  filename:{type:String},
  path: {type: String},
  voz:{type:String,required:false},
  exists:Boolean
})
var WorkSchema = new Schema({
  title:{type:String,required:true,unique:true},
  type:{type:String,required:true},
  composer:{type:String,required:false},
  arrangement:{type:String,required:false},
  instruments:{type:[InstrSchema]}
})
```

1.1.5 Logs

```
var LogSchema = new Schema({
  user_id:{type:String,required:true},
  action:{type:String,required:true},
  date:{type:String,required:false}
})
```

1.2 Controller

Os scripts de classe Controller fazem a ligação entre as rotas do servidor e a base de dados. Neste são criados metodos para execução de queries à base de dados, onde estes possuem forma semelhante.

```
module.exports.FUNCTION_NAME = async () => {
  return MODEL
    .QUERY({PARAMETERS})
    .exec()
}
```



1.3 Utilizadores

Os utilizadores podem ser de três tipos:

- Administrador (utype = 0)
- Produtor (utype = 1)
- Consumidor (utype = 2)

O tipo de um utilizador limita o conjunto de ações a que este tem acesso, diferenciando o acesso, de cada um destes, utilizado pacotes como *passport* e *JSON Web Token*.

A atribuição de diferentes tipos de web tokens aos utilizadores permite singularizá-los, negando o acesso às funcionalidades que não lhe são permitidas. Visto que o registo na aplicação tem de ser aceite por um admin, o uso de tokens permite proteger eficientemente cada rota da camada api presente na aplicação.

Com isto, foram implementadas duas Vistas, associadas à aplicação:

- Admin
- User

Cada uma será carregada de acordo com o tipo de utilizador em sessão e terá diferentes elementos. Onde no caso dos utilizadores, a quantidade de informação disponível, varia de acordo com o seu tipo.

Podemos então antenar às funcionalidades a que cada utilizador tem acesso.

	Consumidor	Produtor	Admin
Adicionar	Não	Sim	Sim
Remover	Não	Itens Próprios	Sim
Editar	Não	Itens Próprios	Sim
Ver	Sim	Sim	Sim

1.4 Rotas

As rotas recebem os pedidos GET e POST e tratam-nos devidamente, onde o encaminhamento é feito para uma de três direções Admin, api e Front. Com o suporte do módulo *Passport*. Para garantir a segurança, todas as rotas estão protegidas, com a exceção da pagina inicial. A título de exemplo, um Consumidor não poderá aceder a rotas de administração.

De maneira a garantir a passagem do **token** é necessário verificar se este esta presente no request ou no Header dos pedidos.

```
if(req && req.session) token = req.session.token
if (typeof token === 'undefined' &&
    typeof req.headers.authorization !== 'undefined')
    token = req.headers.authorization.replace('Bearer ', '')
```

Esta solução foi necessária devido a pedidos executados pelo **Axios**, onde tem de ser injetado no header o valor do token.

Visto isto é necessário ter atenção as diferentes rotas



1.4.1 User

- > GET **/**: Render da pagina inicial dos Utilizadores;
- > GET **/logout**: Destroi sessão, executando o logout;
- > GET **/articles**: Render da pagina de noticias a apresentar ao utilizador;
- > POST **/articles**: Inserção de Noticias pelo Utilizador;
- > GET **/articles/remove/:uid**: Remove noticia através de id;
- > POST **/articles/update**: Atualiza Noticia pelo Utilizador;
- > GET **/events**: Render da pagina de eventos a apresentar ao utilizador;
- > POST **/events**: Inserção de eventos pelo Utilizador;
- > GET **/events/remove/:uid**: Remove evento através de id;
- > POST **/events/update**: Atualiza evento pelo Utilizador;
- > GET **/works**: Render da pagina inicial dos Obras Musicais;
- > POST **/works**: Insere Obra Musical;

1.4.2 Admin

- > GET **/**: Render da pagina inicial do Administrador;
- > GET **/users**: Render da pagina de utilizadores a apresentar ao administrador;
- > POST **/users**: Inserção de utilizadores pelo administrador;
- > GET **/users/remove/:uid**: Remove utilizadores através de id;
- > POST **/users/update**: Atualiza utilizadores pelo administrador;
- > GET **/users/ativar/:uid**: Ativa Utilizador;
- > GET **/users/deativar/:uid**: Desativa Utilizador;
- > GET **/articles**: Render da pagina de noticias a apresentar ao administrador;
- > POST **/articles**: Inserção de Noticias pelo administrador;
- > GET **/articles/remove/:uid**: Remove noticia através de id;
- > POST **/articles/update**: Atualiza Noticia pelo administrador;
- > GET **/events**: Render da pagina de eventos a apresentar ao administrador;
- > POST **/events**: Inserção de eventos pelo administrador;
- > GET **/events/remove/:uid**: Remove evento através de id;
- > POST **/events/update**: Atualiza evento pelo administrador;



- > GET **/works**: Render da pagina inicial dos Obras Musicais;
- > GET **/works/remove/:uid**: Remove do Obra Musical;
- > POST **/works**: Insere Obra Musical;
- > GET **/logs**: Render da pagina inicial dos Logs;

1.5 API

- > POST **/login**: Login de Utilizador;
- > POST **/registo**: Registo de Utilizador;
- > GET **/users**: Retorna utilizadores;
- > POST **/users**: Inserção de utilizador;
- > GET **/users/remove/:uid**: Remove utilizadores através de id;
- > POST **/users/update**: Atualiza utilizador;
- > GET **/users/ativar/:uid**: Ativa Utilizador;
- > GET **/users/deativar/:uid**: Desativa Utilizador;
- > GET **/articles**: Retorna noticias;
- > POST **/articles**: Inserção de Noticias;
- > GET **/articles/remove/:uid**: Remove noticia através de id;
- > POST **/articles/update**: Atualiza Noticia;
- > GET **/events**: Retorna eventos;
- > POST **/events**: Inserção de evento;
- > GET **/events/remove/:uid**: Remove evento através de id;
- > POST **/events/update**: Atualiza evento;
- > GET **/works**: Retorna Obras Musicais;
- > GET **/works/remove/:uid**: Remove Obra Musical;
- > POST **/works**: Insere Obra Musical;
- > GET **/logs**: Retorna Logs;



1.6 Ingestão de SIP

Aquando da reinicialização do servidor, após estabelecida a conexão com o Mongo, é apagada a informação contida na coleção referente às obras musicais, para dar lugar à leitura do ficheiro **iBanda-SIP.json**, incluído no ficheiro **sheets.zip**. Após a operação, é feita a leitura do conjunto de ficheiros presentes no SIP e, para cada entrada, é feita uma pesquisa de cada partitura nas subpastas de `/sheets/iBanda-PDFs`. Todas as entradas são incluídas nos elementos de entrada na BD, sendo sinalizados pelo valor do booleano **exists**, verdadeiro caso a partitura exista. A adição deste elemento permite, aquando da listagem das obras e partituras existentes, definir se há, ou não, ligação entre estes.

A leitura dos ficheiros e pastas do ficheiro **sheets.zip** foi feita com recurso ao módulo `node-stream-zip` que permite, não só o reconhecimento da estrutura interna de um ficheiro, como dá a possibilidade de leitura de cada um dos ficheiros comprimidos.

1.7 View e jQuery

Classe de ficheiros que engloba os geradores de HTML, definindo o aspecto de cada rota da aplicação.

Os scripts de jQuery relativos a cada página são carregados no ficheiro `.pug`, de modo a reduzir o payload, sendo apenas utilizados os scripts necessários.

Figura 1: Home Page

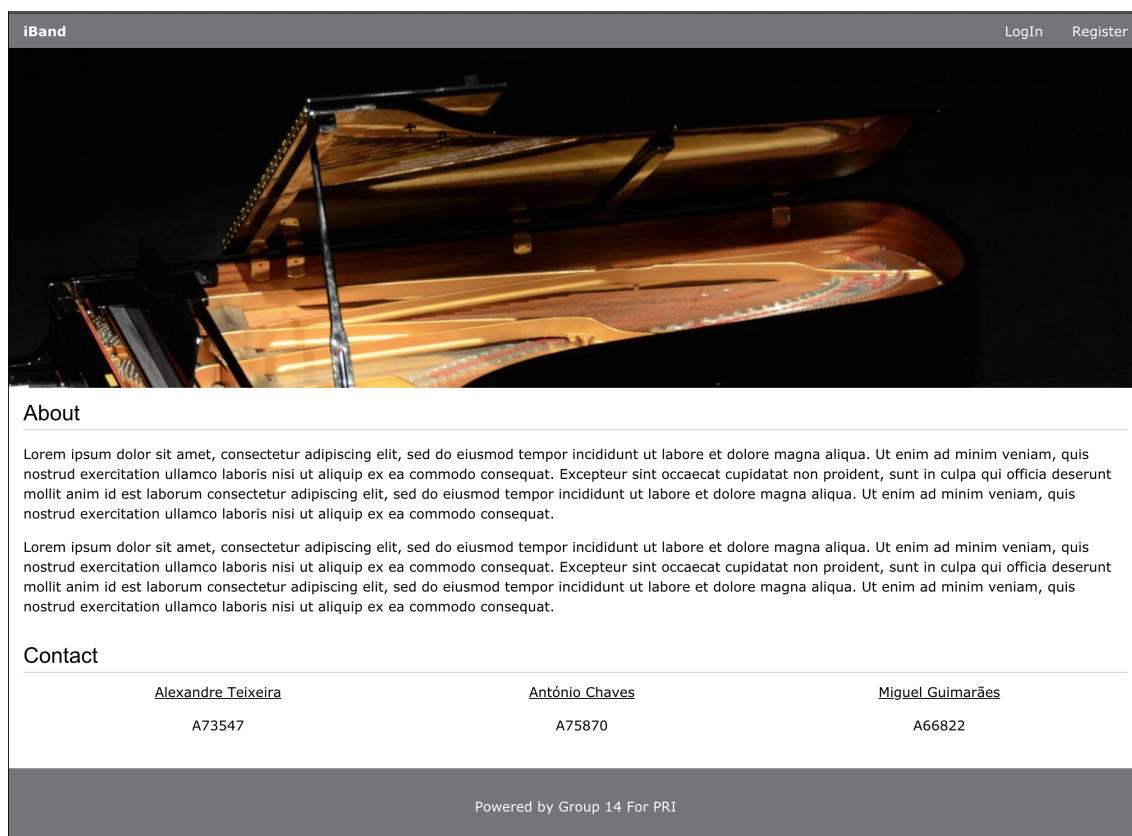




Figura 2: Modal de Registo

Email

Password

Name











User Type

☐ Producer

☒ Client

[Register](#)

Figura 3: Events from Admin

iBand					
User View Logs Users Events Musical Works News Logout					
Events List					
Export Events Create Event					
Theme	Author	Local	Description	Date	Actions
Musical	admin@admin.admin	Castello Lopes Cinemas Guimarães	Musical - The Great Showman	2019-03-07 22:10 Duration: 90	 
Classic Concert	producer@producer.producer	Centro Cultural do Porto	Espetaculo Interativo	2019-03-07 22:10 Duration: 90	 
Opera	admin@admin.admin	Braga Centro Cultural	Espetaculo de Opera	2019-03-07 22:10 Duration: 90	 
Jazz Festival	producer@producer.producer	Centro Cultural Vila Flor	Espetaculo Musical para toda a familia	2019-02-14 20:00 Duration: 90	 
Orquestra Filarmonica de Los Angels	producer@producer.producer	Altice Arena	Orquestra Filarmonica de Los Angels	2019-02-14 20:30 Duration: 130	 

Powered by Group 14 For PRI



Figura 4: News from Admin

iBand

User ViewLogsUsersEventsMusical WorksNewsLogout

Articles List

Create Article

Title	Author	Content	Date	Actions
Castello Lopes Cinemas Guimarães	admin@admin.admin	Espaço muito pequeno e a qualidade deixa a desejar	2019-1-1	<div><div></div><div></div></div>
Musical - The Great Showman	admin@admin.admin	É um filme bastante engraçado, para todos os gostos	2019-01-31	<div><div></div><div></div></div>
Centro Cultural Vila Flor	producer@producer.producer	Um espaço muito agradável, repleto de espaços verdes	2019-0-31	<div><div></div><div></div></div>

Powered by Group 14 For PRI

Figura 5: Logs from Admin

iBand			User View	Logs	Users	Events	Musical Works	News	Logout
Logs									
User	Data	Action							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 05:10:54 GMT+0000 (Western European Standard Time)	login							
5c509d176df5d9135291a717	Fri Feb 01 2019 05:09:23 GMT+0000 (Western European Standard Time)	logout							
5c509d176df5d9135291a717	Fri Feb 01 2019 05:02:52 GMT+0000 (Western European Standard Time)	login							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 05:02:43 GMT+0000 (Western European Standard Time)	logout							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 04:51:26 GMT+0000 (Western European Standard Time)	login							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 04:49:13 GMT+0000 (Western European Standard Time)	login							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 04:47:24 GMT+0000 (Western European Standard Time)	login							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 03:57:30 GMT+0000 (Western European Standard Time)	login							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 03:55:47 GMT+0000 (Western European Standard Time)	logout							
5c53505d6981343d1c34ccb2	Fri Feb 01 2019 03:53:29 GMT+0000 (Western European Standard Time)	login							



Figura 6: User Page

IBanda

EventsMusical WorksNewsLogout

producer@producer.producer

Welcome to the user front page, you can select multiple options as Producer.

News

admin@admin.admin
Castello Lopes Cinemas Guimarães
2019-1-1

Espaço muito pequeno e a qualidade deixa a desejar

admin@admin.admin
Musical - The Great Showman
2019-01-31

É um filme bastante engraçado, para todos os gostos

producer@producer.producer
Centro Cultural Vila Flor
2019-0-31

Um espaço muito agradável, repleto de espaços verdes

Events

Musical
admin@admin.admin
Castello Lopes Cinemas Guimarães
Musical - The Great Showman

Date: 2019-03-07 22:10
Duration: 90Min



2 json2Mongo

A solução proposta parte de uma adaptação de um parser de json, disponível em <https://json.org> e foi adaptada às necessidades dos ficheiros .json fornecidos.

2.1 Gramática

```
grammar ant;
@parser::header{
    import com.mongodb.*;
}
@parser::members{
    antHandler handler = new antHandler();
}

json
    : obj (VIRG obj)*
    ;

obj
    : FK p1=pair (VIRG p2=pair)* BK
    | FK BK
    ;

pair
    : property DPTS member
    ;

property
    : EVENT_PROPERTY
    | NEWS_PROPERTY
    ;

member
    : array
    | NUMBER
    | STRING
    | date
    | coords
    ;

array
    : FP v1=arrayUnit(VIRG v2=arrayUnit)* BP
    | FP BP
    ;

arrayUnit
    : STRING
    | NUMBER
    ;

coords
    : FK coord
    ;
```



```
coord
  : COORD DPTS STRING
  ;
date
  : d=NUMBER FSL m=NUMBER FSL a=NUMBER VIRG h=NUMBER DPTS mm=NUMBER
  ;
FK : '{';
BK : '}' ;
FP : '[' ;
BP : ']' ;
VIRG: ',' ;
DPTS: ':' ;
FSL: '/' ;
EVENT_PROPERTY
  : 'local'
  | 'tema'
  | 'descricao'
  | 'horario'
  | 'interesse'
  | 'coordenadas'
  ;
NEWS_PROPERTY
  : 'data'
  | 'titulo'
  | 'texto'
  | 'autor'
  | 'pchave'
  ;
COORD
  : 'latitude'
  | 'longitude'
  ;
STRING
  : '"' (ESC | SAFECODEPOINT)* '"'
  ;
fragment ESC
  : '\\' (["\\/\bfnrt] | UNICODE)
  ;
fragment UNICODE
  : 'u' HEX HEX HEX HEX
  ;
fragment HEX
  : [0-9a-fA-F]
  ;
fragment SAFECODEPOINT
  : ~ ["\\ \u0000-\u001F]
  ;
NUMBER
```



```
      : '-'? INT ('.' [0-9]+)?  
      ;  
fragment INT  
      : '0' | [1-9][0-9]*  
      ;  
WS  
      : [ \t\r\n]+ -> skip  
      ;
```

2.2 Java

2.2.1 Class `antHandler.java`

Classe de controlo, conta com a persistência da conexão à BD, trata do setup de coleção e de BD. Tem algumas prompts textuais que servem para ajudar o utilizador a escolher as opções corretas, tratando de todos os inputs.

2.2.2 `Main.java`

Classe inicial da aplicação, trata da geração dos objetos **Lexer** e **Parser**, assim como do fluxo de dados proveniente do ficheiro definido.

Os ficheiros a ser lidos devem ser guardados na directoria `/GrammarMongo/json/`.

2.3 Compilação e Execução

Sugere-se a compilação e execução da aplicação com o IDE IntelliJ e as devidas dependências instaladas. Com o projeto bem configurado, é possível testar o sistema facilmente, sendo apenas necessário fornecer o ficheiro de entrada e os nomes pedidos.

2.4 DSLs

2.4.1 Evento

```
Evento -> {local, descricao, tema, horario, data,  
           hora, duracao, ListaInteresse}  
ListaInteresse -> [nome]
```

2.4.2 Notícia

```
Notícia -> {data, título, conteúdo, autor}
```




Conclusão

Chegando ao final da implementação, é possível entender o estado das aplicações e aferir a sua qualidade.

Em relação à submissão de GCS, a entrega não só cumpre com o pedido, como também oferece uma solução sólida para leitura de ficheiros .json (para que qualquer ficheiro .json seja lido a partir da gramática especificada apenas é necessário efetuar um pequeno conjunto de alterações). Ainda assim, o resultado submetido é mais eficaz no preenchimento da base de dados, pois conta com as palavras reservadas que vão ser encontradas nos ficheiros.

Quando à aplicação WEB, foi procurado seguir o modelo de implementação MVC e com proteção de todas as rotas, de modo a oferecer uma aplicação que não só é de fácil manutenção, como também segura. A generalidade dos requisitos propostos no enunciado foi coberta no desenvolvimento e foi tido em conta o aspeto da aplicação ao longo do seu desenvolvimento.



Anexos