

**UNIVERSIDADE DO MINHO**

MESTRADO EM ENGENHARIA INFORMÁTICA

## PROCESSAMENTO DE LINGUAGENS E CONHECIMENTO

**Processamento e Representação de Informação**

**Gramáticas na Compreensão de Software**

TRABALHO PRÁTICO



---

ANDRÉ SALGUEIRO (A77617)  
FLÁVIA SILVA (A64303)  
JOANA PEREIRA (A78275)

1 de Fevereiro de 2019

## **Resumo**

Este documento diz respeito ao trabalho prático proposto nas unidades curriculares de Processamento e Representação de Informação e Gramáticas na Compreensão de Software da Universidade do Minho. O objetivo deste projeto consiste no desenvolvimento e implementação de uma aplicação *web* para um repositório digital de obras musicais e suas partituras. Com a aplicação desenvolvida é possível arquivar de forma segura e com controlo de acesso todas as informações relativas à banda. As principais funcionalidades passam pela consulta e gestão de obras, visualização de partituras, gestão de eventos, notícias, utilizadores, repertório, e uma pequena enciclopédia do material armazenado. Para controlar o acesso à aplicação foram também definidos três tipos de utilizadores com funcionalidades e permissões diferentes na aplicação. Para além disto, é também possível inserir informação na base de dados recorrendo a uma gramática de atributos. Para isso foram desenvolvidas duas gramáticas de atributos, uma para catalogar eventos e outra para introduzir notícias.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Análise e Especificação</b>	<b>3</b>
2.1	Descrição do Problema . . . . .	3
2.2	Descrição da Solução . . . . .	3
2.2.1	Organização da diretoria da aplicação . . . . .	3
<b>3</b>	<b>Conceção e Desenho da Solução</b>	<b>5</b>
3.1	Base de Dados . . . . .	5
3.1.1	Esquemas das Coleções . . . . .	5
3.2	Autenticação . . . . .	7
3.2.1	Especificação de Requisitos . . . . .	7
3.2.2	Implementação . . . . .	7
3.3	Administrador . . . . .	9
3.3.1	Especificação de Requisitos . . . . .	9
3.3.2	Funcionalidades . . . . .	11
3.4	Utilizador . . . . .	20
3.4.1	Especificação de Requisitos . . . . .	20
3.4.2	Funcionalidades . . . . .	20
3.5	Produtor . . . . .	25
3.5.1	Especificação de Requisitos . . . . .	25
3.5.2	Funcionalidades . . . . .	26
3.5.3	Exemplo . . . . .	26
3.6	Inserção de informação na BD recorrendo a GA's . . . . .	27
3.6.1	Inserção de Eventos . . . . .	27
3.6.2	Inserção de Notícias . . . . .	32
<b>4</b>	<b>Conclusões</b>	<b>37</b>

# Capítulo 1

## Introdução

Este trabalho prático enquadra-se nas unidades curriculares de Processamento e Representação de Informação e Gramáticas para a Compreensão de Software para o desenvolvimento de competências relativas a estas.

Actualmente uma banda filarmónica tem a necessidade de obter informação de uma obra ou mesmo visualizar uma partitura musical. Com a **iBanda** é possível ter acesso direto a qualquer partitura, desde que esta se encontre na posse da banda, em qualquer momento e lugar tendo em conta que terá de ter acesso à Internet.

De forma a fazer uma melhor gestão da aplicação, foram definidos três tipos de utilizadores: administrador, produtor e utilizador.

Além de poder aceder às obras e partituras, o(s) administrador(es) têm a possibilidade de gerir uma agenda de eventos, notícias, utilizadores, repertórios, assim como, construir uma pequena enciclopédia do material armazenado.

Para além da inserção de informação na base de dados (**Mongo**) através das interfaces do administrador e do produtor, é também possível catalogar eventos e introduzir notícias recorrendo a duas gramáticas de atributos desenvolvidas para o efeito. De uma forma geral, estas processam o ficheiro de *input*, criam os objetos e adicionam-nos à base de dados.

O produtor tem a capacidade de catalogar novas obras e adicionar novas partituras. Isto é conseguido através da submissão de um **ZIP** que contém um ficheiro **JSON** (*iBanda-SIP.json*) que descreve a obra a ser catalogada e um conjunto de partituras.

Já o utilizador normal pode consultar e editar o seu perfil, consultar obras e partituras, o repertório da banda, notícias, a enciclopédia e a agenda da banda. Pode também pesquisar por obras, eventos e repertórios.

## Estrutura do Relatório

Neste presente relatório é feita uma análise e especificação do problema, que passa maioritariamente pela sua descrição.

Posteriormente apresenta-se a conceção e o desenho da solução. Aqui é apresentada a forma de armazenamento de toda a informação na base de dados, nomeadamente os esquemas das coleções relativos aos utilizadores, obras, repertórios, eventos, e notícias. É também especificada a forma de autenticação na aplicação, apresentando os requisitos necessários e a forma de implementação. De seguida são apresentadas as permissões e funcionalidades de cada tipo de utilizador, assim como os requisitos definidos para cada um. Finalmente é também apresentada a forma de inserção da informação na base de dados recorrendo a gramáticas de atributos.

Por último, será apresentada uma conclusão sobre o projeto.

## Capítulo 2

# Análise e Especificação

### 2.1 Descrição do Problema

No dia a dia de uma banda filarmónica é importante ter acesso direto às informações da banda, em qualquer momento e lugar.

Este projeto consiste na criação de uma aplicação para um repositório digital de obras musicais e suas partituras. Desta forma, deve ser possível inserir novas obras e consultá-las, assim como, ter acesso às suas partituras. Adicionalmente, deve também ser possível fazer a gestão da agenda da banda, dos repertórios, das notícias e da enciclopédia do material armazenado.

Para além disto, é necessário controlar o acesso à informação da banda, ou seja, não deve ser permitido o acesso à aplicação por qualquer pessoa. É então obrigatório que exista uma autenticação inicial, sendo que é também importante que a palavra-passe de cada utilizador seja guardada de forma encriptada, aumentando a segurança dos dados da aplicação.

Tendo em conta que a informação deve ser controlada e restringida a determinados utilizadores, é importante também fazer uma verificação constante do tipo de utilizador que está autenticado, impedindo o acesso caso este não deva aceder a determinadas funcionalidades ou a determinado tipo de informação.

De forma a armazenar todos os dados da banda é também necessário recorrer a uma base de dados. No caso do **iBanda** vamos recorrer a uma base de dados **MongoDB**, onde foram definidas as coleções necessárias que vão suportar as operações de leitura e escrita de dados necessárias ao funcionamento da aplicação.

Definindo e implementando todos estes componentes, é possível obter como resultado final uma aplicação que permite suportar todas as necessidades de uma banda.

### 2.2 Descrição da Solução

Para o desenvolvimento da aplicação recorremos ao Express, uma *framework* do **Node.js** para aplicações *web*. Para além disso, vamos também recorrer ao padrão de arquitetura de *software* **MVC** (*Model-view-controller*) , que permite a separação da representação da informação da interação do utilizador com ela.

Para além disto, vamos também separar as rotas de cada tipo de utilizadores da API, permitindo também uma reutilização de pedidos e envios de informação.

#### 2.2.1 Organização da diretoria da aplicação

A aplicação vai estar organizada em:

**controllers** onde estão definidas as operações à base de dados para cada coleção

**models** onde estão definidos os esquemas das coleções

**public** que vai guardar as imagens, ficheiros *javascript* usados pelos utilizadores e pelo administrador na aplicação, partituras, *stylesheets*, os ZIPs submetidos pelo(s) produtor(es) e o ficheiro de exportação da agenda.

**routes** que define os pedidos HTML feitos pelos utilizadores à API

**sessions** onde estão guardadas as sessões dos utilizadores

**views** onde são definidas as vistas dos diferentes utilizadores

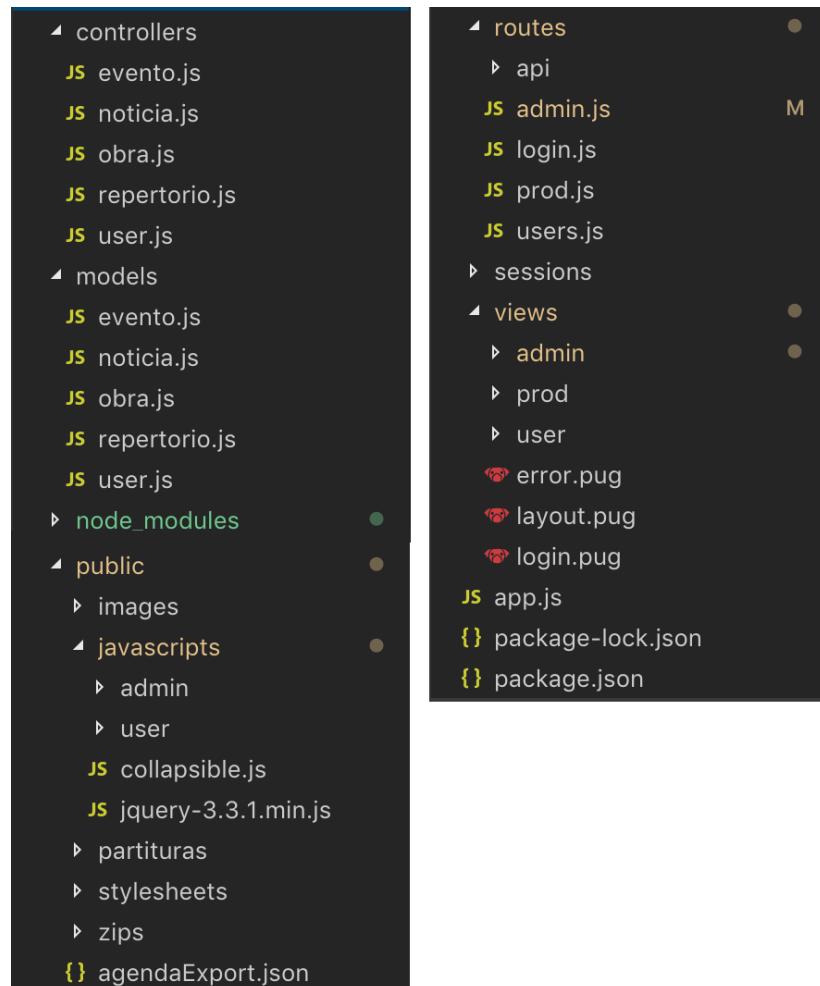


Figura 2.1: Organização da diretoria da aplicação.

# Capítulo 3

## Conceção e Desenho da Solução

### 3.1 Base de Dados

De forma a guardar toda a informação relativa à aplicação numa base de dados **MongoDB**, foram definidos os esquemas de cada uma das coleções necessárias, neste caso, para as coleções dos utilizadores, das obras, do repertório, dos eventos e das notícias. Para aceder à base de dados '**amd**' e a todas as coleções foi utilizada a ferramenta **Mongoose**.

```
//Ligaçao à BD
mongoose.connect('mongodb://127.0.0.1:27017/ amd', {useNewUrlParser: true})
  .then(()=> console.log('Mongo Status: ' + mongoose.connection.readyState))
  .catch(()=> console.log('Erro na conexão ao Mongo.'))
```

Listing 1: Ligação à base de dados feita em app.js.

#### 3.1.1 Esquemas das Coleções

Os utilizadores fazem *login* na aplicação através do email e da palavra-passe. Desta forma, é necessário que o email seja único para cada utilizador. Para além disso, podem assumir diferentes papéis na aplicação, o de administrador, utilizador ou produtor (guardado no campo tipo). Esta restrição de valores no tipo dos utilizadores é importante, porque permite fazer a verificação do utilizador que está autenticado e permitir ou não o acesso a determinadas funcionalidades da aplicação. Para além disso, um utilizador pode também ter associados um nome, que é obrigatório, e, opcionalmente, o instrumento que tocam, a data de nascimento, o *link* para uma imagem e uma descrição das suas habilitações.

```
var UserSchema = new Schema({
  nome: {type: String, required: true},
  email: {type: String, required: true, unique: true},
  passwd: {type: String, required: true},
  instr: {type: String},
  dataNasc: {type: String},
  tipo: {type: String, required: true},
  img_path: {type: String},
  habilitacoes: {type: String}
})
```

Listing 2: Esquema dos utilizadores.

Também para as obras foi necessário definir o esquema da coleção para guardar toda a informação necessária destas. Cada obra contém, obrigatoriamente, um título, que é único para cada uma, um tipo e um compositor. Opcionalmente, podem ter o nome de quem fez o arranjo e uma lista de instrumentos. Cada instrumento desta lista tem um nome e uma partitura, que é constituída pelo nome do ficheiro da partitura, pela voz, pela afinação e pela clave.

```
var PartituraSchema = new Schema({
  path: {type: String},
  voz: {type: String},
  afinacao: {type: String},
  clave: {type: String}
})

var InstrumentoSchema = new Schema({
  nome: {type: String},
  partitura: {type: PartituraSchema}
})

var ObraSchema = new Schema({
  titulo: {type: String, required: true, unique: true},
  tipo: {type: String, required: true},
  compositor: {type: String, required: true},
  arranjo: {type: String},
  instrumentos: {type: [InstrumentoSchema]}
})
```

Listing 3: Esquema das obras.

Relativamente ao repertório, é apenas necessário guardar a designação do evento e a lista dos títulos das obras associadas a esse evento. Como o campo `designacao` dos eventos, e o campo `titulo` das obras é único para cada um, é possível obter toda a informação sobre o evento e sobre as obras apenas com estes valores.

```
var RepertorioSchema = new Schema({
  evento: {type: String, required: true},
  obras: {type: [String]}
})
```

Listing 4: Esquema do repertório.

Quanto aos eventos, estes têm como campos obrigatórios a designação, única para cada um, a data, o tipo e o local. Opcionalmente é possível também guardar o horário em que ocorreu ou vai ocorrer esse evento (hora de inicio e hora de fim) e informações sobre o mesmo.

```
var HorarioSchema = new Schema({
  hinicio: {type: String},
  hfim: {type: String}
})

var EventoSchema = new Schema({
  data: {type: String, required: true},
  tipo: {type: String, required: true},
  local: {type: String, required: true},
  horario: {type: HorarioSchema},
  designacao: {type: String, required: true, unique: true},
```

```

        informacoes: {type: String}
    })

```

Listing 5: Esquema dos eventos.

Finalmente, no que diz respeito às notícias, estas têm como campos obrigatórios a data, o título e o corpo da notícia. Facultativamente pode ser associada a uma notícia um subtítulo, um conjunto de palavras-reservadas, o *link* para uma imagem e o estado de visibilidade destas, ou seja, se os utilizadores da aplicação podem ou não ver a notícia.

```

var NoticiaSchema = new Schema({
    data: {type: String, required: true},
    titulo: {type: String, required: true},
    subtítulo: {type: String},
    hash: {type: String},
    img_path: {type: String},
    corpo: {type: String, required: true},
    visivel: {type: Boolean}
})

```

Listing 6: Esquema das notícias.

## 3.2 Autenticação

A aplicação deverá possibilitar que os utilizadores se autentiquem e tenham acesso às funcionalidades disponibilizadas para estes. Vamos utilizar como credenciais de autenticação o *email* e a *password* de cada utilizador. Vamos também recorrer aos módulos **passport**, **passport-local**, que vamos utilizar para definir a estratégia local de autenticação, **uuid/v4**, **express-session**, **session-file-store**, que vai guardar os dados das sessões do lado do servidor na pasta `/sessions` e **bcrypt** para a encriptação das palavras-passe dos utilizadores.

### 3.2.1 Especificação de Requisitos

A aplicação deve ser capaz de permitir que:

- Qualquer utilizador faça *login* com as suas credenciais
- Qualquer utilizador termine sessão da sua conta, caso estiver autenticado
- As palavras-passe dos utilizadores sejam encriptadas antes de serem guardadas
- Seja restringido o acesso a determinadas funcionalidades, dependendo do tipo de utilizador

### 3.2.2 Implementação

```

passport.use('login', new LocalStrategy({
    usernameField: 'email',
    passwordField: 'passwd'
}), async (email, passwd, done) => {
    try {
        var user = await User.findOne({email})
        if (!user)

```

```

        return done(null, false, {message: 'O utilizador não existe.'})
    var valid = await user.isValidPassword(passwd)
    if (!valid)
        return done(null, false, {message: 'Password inválida.'})
    return done(null, user, {message: 'Login feito com sucesso.'})
}
catch(error) {
    return done(error)
}
})
})
})

```

Listing 7: Estratégia local de autenticação.

Dependendo do tipo de utilizador que se autentica, este é redirecionado para uma página inicial diferente, isto porque cada um tem acesso a diferentes funcionalidades da aplicação. Para isso, é feita a verificação do tipo do utilizador que é guardado em `req.user` quando é feito o *login*.

```

router.post('/login', passport.authenticate('login', { failureRedirect: '/' }), function(req, res) {
    if(req.user.tipo=='Administrador')
        res.redirect('http://localhost:3000/admin')
    if(req.user.tipo=='Utilizador')
        res.redirect('http://localhost:3000/user')
    if(req.user.tipo=='Produtor')
        res.redirect('http://localhost:3000/prod')
})

```

Listing 8: Redirecionamento dependendo do tipo de utilizador, após autenticação.

```

router.get('/logout', verificaAutenticacao, (req, res) => {
    req.logOut();
    res.redirect('http://localhost:3000/')
})

```

Listing 9: Terminar sessão de um utilizador.

É também necessário verificar se um utilizador está autenticado quando é feito um pedido à aplicação. Isto porque o acesso pode ser impedido dependendo do tipo do utilizador. Em Listing 10 podemos ver essa verificação feita para o administrador. Esta verificação é feita de forma semelhante para os outros tipos de utilizador.

```

//verifica se é o administrador que está autenticado (impede o acesso a utilizadores e produtores)
function verificaAutenticacaoAdmin(req, res, next) {
    if (req.isAuthenticated()) {
        if (req.user.tipo != 'Administrador') {
            console.log('Não é o administrador.')
            res.redirect('http://localhost:3000/')
        }
        else {
            console.log('Está autenticado!')
            next()
        }
    }
}

```

```

    else {
      console.log('Não está autenticado!')
      res.redirect('http://localhost:3000/')
    }
}

```

Listing 10: Verificação da autenticação de um utilizador.

Quando um utilizador é registado na aplicação, é necessário encriptar a palavra-passe para ser guardada na base de dados. Para isso recorremos à função hash da biblioteca **bcrypt**.

```

UserSchema.pre('save', async function (next) {
  var hash = await bcrypt.hash(this.passwd, 10)
  this.passwd = hash
  next()
})

```

Listing 11: Encriptação da palavra-passe do utilizador.

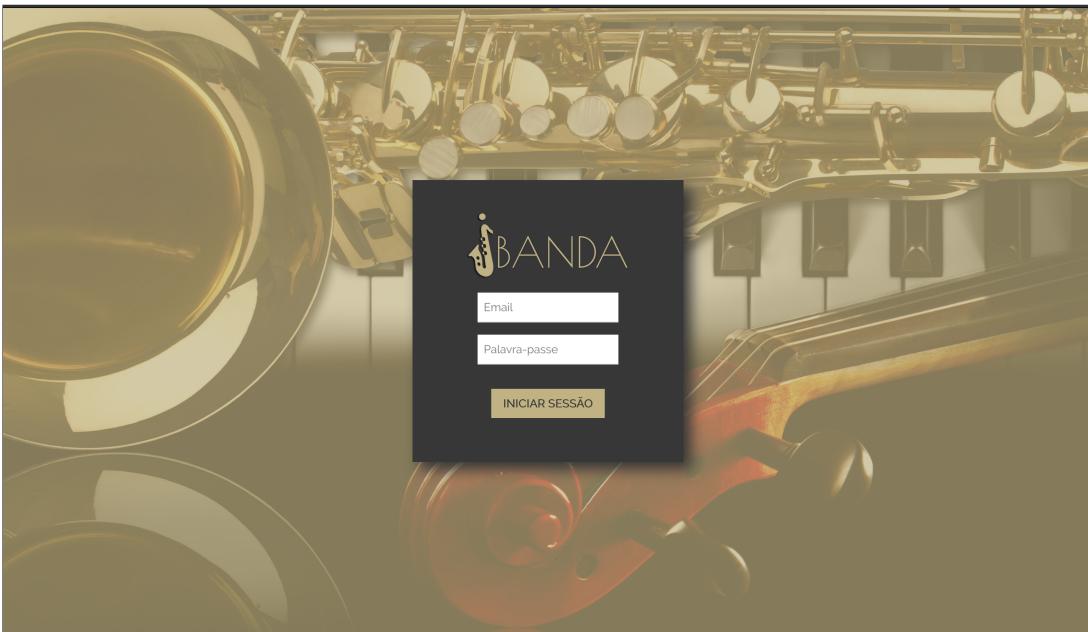


Figura 3.1: Página inicial de *login* na aplicação.

### 3.3 Administrador

Todas as rotas referentes ao administrador estão definidas no ficheiro `/routes/admin.js`. Estas estão restritas a este e em cada uma delas é feita a verificação do tipo do utilizador, se é 'Administrador', que está autenticado.

#### 3.3.1 Especificação de Requisitos

O administrador, que tem o papel de gerir a informação da aplicação, deve ser capaz de:

- Consultar todos os utilizadores
- Registar novos utilizadores
- Editar o perfil de um utilizador
- Remover utilizadores
- Pesquisar por utilizadores
- Consultar obras
- Editar obras
- Remover obras
- Pesquisar obras
- Submeter partituras
- Descarregar partituras
- Consultar o repertório da banda
- Adicionar informação ao repertório da banda
- Remover informação do repertório
- Editar o repertório
- Pesquisar no repertório por eventos e obras
- Consultar os eventos da banda
- Registar novos eventos
- Editar eventos
- Remover eventos
- Pesquisar eventos
- Exportar a agenda
- Consultar notícias
- Inserir notícias
- Editar notícias
- Tornar uma notícia visível/invisível
- Consultar a enciclopédia, que contém todas as partituras armazenadas
- Remover partituras da enciclopédia

### 3.3.2 Funcionalidades

#### Utilizadores

O administrador têm a capacidade de consultar todos os utilizadores que registou na aplicação. Para o registo destes é obrigatório que cada email seja único. Para além da consulta e do registo, também a edição e a remoção de utilizadores é possível por parte do administrador. Para facilitar este trabalho, é também disponibilizado ao administrador uma pesquisa por utilizadores, quer por email, nome ou tipo.

The screenshot displays the 'Utilizadores' (Users) section of the iBANDA application. At the top, there's a navigation bar with links for UTILIZADORES, AGENDA, ENCICLOPÉDIA, OBRAS, NOTÍCIAS, REPERTÓRIO, and a power icon. The central area has a header 'NOVO UTILIZADOR' with a user icon. Below it is a search bar labeled 'PESQUISAR' with a magnifying glass icon. Three user profiles are listed in cards:

- António Manuel** (Utilizador): Placeholder image of a man, email: antonio@email.com, birthdate: 1985-09-23, instrument: Pandeireta, hobby: Também toca flauta. Buttons: REMOVER, EDITAR.
- Rute** (Produtor): Placeholder image of a woman, email: root@email.com, birthdate: 1998-08-15. Buttons: REMOVER, EDITAR.
- sysadmin** (Administrador): Placeholder image of a person with a mask, email: admin@mail.pt, birthdate: \*\*\*\*\*. Buttons: REMOVER, EDITAR.

Figura 3.2: Página do administrador de consulta dos utilizadores.

The screenshot shows the 'NOVO UTILIZADOR' (New User) registration form. The top navigation bar is identical to Figure 3.2. The form itself has a header 'NOVO UTILIZADOR' with a user icon. It contains the following fields:

- Nome (Name): Input field with a person icon.
- Email: Input field with an envelope icon.
- Palavra-Passe (Password): Input field with a lock icon.
- Instrumento (Instrument): Input field with a drum icon.
- Data de Nascimento (Birthdate): Input field with a calendar icon.
- Role selection:
  - Administrador
  - Utilizador
  - Produtor
- Link da Imagem (Image Link): Input field with a camera icon.
- Habilidades (Abilities): Text area input field with a speech bubble icon.

A note at the bottom left says: \* Campos de preenchimento obrigatório (Required fields). A large 'ADICIONAR' (Add) button is at the bottom right.

Figura 3.3: Formulário do administrador para registo de utilizadores.

The screenshot shows the 'NOVO UTILIZADOR' (New User) search page. At the top, there are three search input fields: 'Pesquisar pelo tipo:' (Search by type:), 'Introduza o tipo do utilizador.'; 'Pesquisar pelo nome:' (Search by name:), 'Introduza o nome do utilizador.'; and 'Pesquisar pelo email:' (Search by email:), 'Introduza o email.'. Below these fields is a grid of three user profiles:

- António Manuel** (Utilizador): Profile picture of a man, email: antonio@email.com, password: \*\*\*\*\*, birthdate: 1985-09-23, hobby: Pandeireta.
- Rute** (Produtor): Profile picture of a woman, email: root@email.com, password: \*\*\*\*\*, birthdate: 1998-08-15.
- sysadmin** (Administrador): Profile picture of a mask, email: admin@mail.pt, password: \*\*\*\*\*.

At the bottom right of the grid, there are two buttons: 'REMOVER' (Remove) and 'EDITAR' (Edit).

Figura 3.4: Campos de pesquisa por utilizadores.

The screenshot shows the 'EDITAR UTILIZADOR' (Edit User) form. The user profile is displayed on the left, and the edit form on the right contains the following fields:

	António Manuel
	antonio@email.com
	*****
	Pandeireta
	1985-09-23
	<a href="https://d1yn1kh78jj1rr.cloudfront.net/image/thumbnail/rnp9KOKv">https://d1yn1kh78jj1rr.cloudfront.net/image/thumbnail/rnp9KOKv</a>
	Também toca flauta.

At the bottom of the form are two buttons: 'EDITAR' (Edit) and 'EDITAR UTILIZADOR' (Edit User).

Figura 3.5: Formulário de edição de utilizadores.

## Obras

O administrador tem a capacidade de consultar todas as obras guardadas na aplicação. Para além disso, pode também remover obras e submeter partituras, que através do nome vão ser associadas a uma obra para serem posteriormente descarregadas, quer pelo próprio administrador, quer pelos utilizadores. O administrador pode também fazer pesquisas pelas obras, por tipo, compositor ou título. O título das obras é também um valor único para cada uma, ou seja, não pode ser inserida uma obra com um título que já exista.

The screenshot shows the 'OBRAS' (Works) section of the iBANDA administrator interface. At the top, there are navigation links: UTILIZADORES, AGENDA, ENCICLOPÉDIA, OBRAS (highlighted in yellow), NOTÍCIAS, REPERTÓRIO, and a power icon. Below the header, there are two search bars: 'SUBMETER PARTITURA' with a musical note icon and 'PESQUISAR' with a magnifying glass icon. The main content area displays six musical works in a grid:

- Xutos Medley** (Ligeiro/Fantasia) by Xutos e Pontapés and Luis Cardoso. Includes 'Instrumentos' (Instruments) and 'Edit' buttons.
- Xabia** (Marcha de Desfile) by Salvador Salvá. Includes 'Instrumentos' (Instruments) and 'Edit' buttons.
- Vila Franca** (Marcha de Concerto/Pasodoble) by Jorge Salgueiro. Includes 'Instrumentos' (Instruments) and 'Edit' buttons.
- Vamos à Romaria** (Rapsódia) by Valdemar Sequeira.
- Transfiguração** (Marcha de Procissão) by António Almeida da Silva.
- Tarântula** (Marcha de Desfile) by Amílcar Moraes.

Figura 3.6: Página do administrador de consulta das obras.

The screenshot shows the 'OBRAS' (Works) section of the iBANDA administrator interface. At the top, there are navigation links: UTILIZADORES, AGENDA, ENCICLOPÉDIA, OBRAS (highlighted in yellow), NOTÍCIAS, REPERTÓRIO, and a power icon. Below the header, there is a 'SUBMETER PARTITURA' button with a musical note icon. A 'Ficheiro da Partitura:' field contains 'Explorar...' and 'Nenhum ficheiro selecionado.' buttons. A 'SUBMETER' button is below this field. The main content area includes a 'PESQUISAR' search bar with a magnifying glass icon. There are three search input fields: 'Pesquisar pelo tipo:' (Search by type: Introduza o tipo da obra.), 'Pesquisar pelo compositor:' (Search by composer: Introduza o compositor.), and 'Pesquisar pelo título:' (Search by title: Introduza o título da obra.). Below these fields, the same six musical works from Figure 3.6 are listed in a grid.

Figura 3.7: Campos de pesquisa por obras e submissão de partituras.

## Repertório

O administrador tem a capacidade de consultar o repertório da banda, ou seja, que obras estão associadas a cada evento. Aqui, é possível fazer a pesquisa quer pela designação do evento, quer pelo título da obra. É também permitida a visualização da informação de cada evento e de cada obra. Isto é permitido através da unicidade do título da obra e da designação do evento. É também permitido que o administrador elimine ou edite informação do repertório.

Figura 3.8: Página do administrador de consulta do repertório.

Figura 3.9: Formulário de criação de repertório.

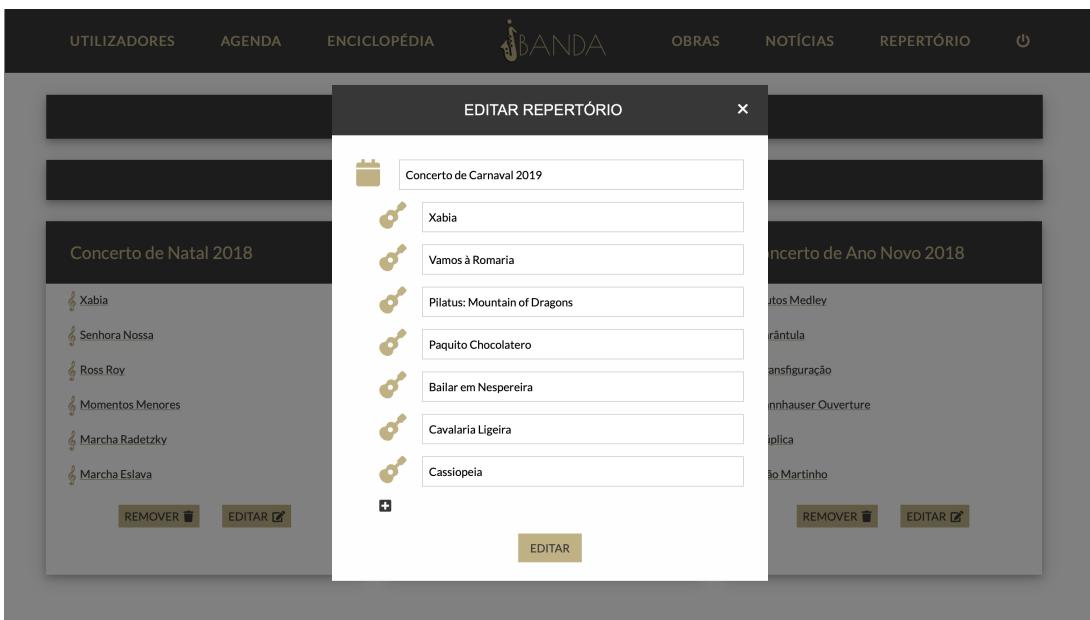


Figura 3.10: Formulário de edição do repertório.



Figura 3.11: Campos de pesquisa do repertório.

## Agenda

O administrador pode fazer a gestão completa da agenda da banda. Isto inclui inserir ou remover eventos, editar eventos existentes, consultar todos os eventos e pesquisar eventos por tipo, designação, data (exata ou os eventos após uma data) e local. A designação de cada evento é única para cada um. Através da utilização do **moment** do JavaScript foi possível determinar se os eventos já ocorreram (com data anterior à atual) e apresentá-los de forma diferente aos utilizadores, neste caso, em escala de cinzentos. Em app.js foi então definido app.locals.moment = require('moment') e em /views/admin/eventos.pug, página

que renderiza os eventos do lado do administrador, é feita a verificação da data para cada evento com `if moment().format('YYYY-MM-DD').toString()<e.data`, em que `e.data` corresponde à data do evento. Para além disto, os administradores podem também exportar a agenda completa em formato JSON.

The screenshot shows a web application interface for managing a band's agenda. At the top, there are navigation links: UTILIZADORES, AGENDA, ENCICLOPÉDIA, OBRAS, NOTÍCIAS, REPÓRTERIO, and a power icon. The central area has a search bar labeled 'NOVO EVENTO' and 'PESQUISAR'. Below this, a grid displays several events:

- Concerto**: Concerto de Carnaval 2019. Date: 2019-03-02, Location: Braga, Start: 17:00. Note: O concerto vai realizar-se no Teatro Circo. Actions: REMOVER, EDITAR.
- Ensaio**: Ensaio para o concerto de Carnaval 2019. Date: 2019-02-27, Location: Vila Verde, Start: 19:00. Actions: REMOVER, EDITAR.
- Concerto**: Concerto de Ano Novo 2018. Date: 2018-12-29, Location: Amares, Start: 19:00, End: 21:00. Note: O concerto vai realizar-se no Mosteiro de Santo André de Rendufe. Actions: REMOVER, EDITAR.
- Concerto**: Concerto de Natal 2018. (Partial view).
- Ensaio**: Ensaio para o concerto. (Partial view).

A large button labeled 'Exportar Agenda' with a download icon is located on the left side of the event grid.

Figura 3.12: Página do administrador de consulta da agenda da banda.

The screenshot shows a form for creating a new event. The top navigation bar is identical to Figure 3.12. The form itself has a header 'NOVO EVENTO' with a camera icon. It contains the following fields:

- Designação**: Input field with a music note icon.
- Data: AAAA-MM-DD**: Input field with a calendar icon.
- Local**: Input field with a location pin icon.
- Concerto** / **Ensaio**: Radio buttons for event type.
- Hora de Início** and **Hora de Fim**: Input fields with a clock icon.
- Informações**: Input field with an info icon.

A note at the bottom says: \* Campos de preenchimento obrigatório. A 'ADICIONAR' button is at the bottom right. Below the form is a search bar labeled 'PESQUISAR'.

Figura 3.13: Formulário do administrador para criação de eventos.

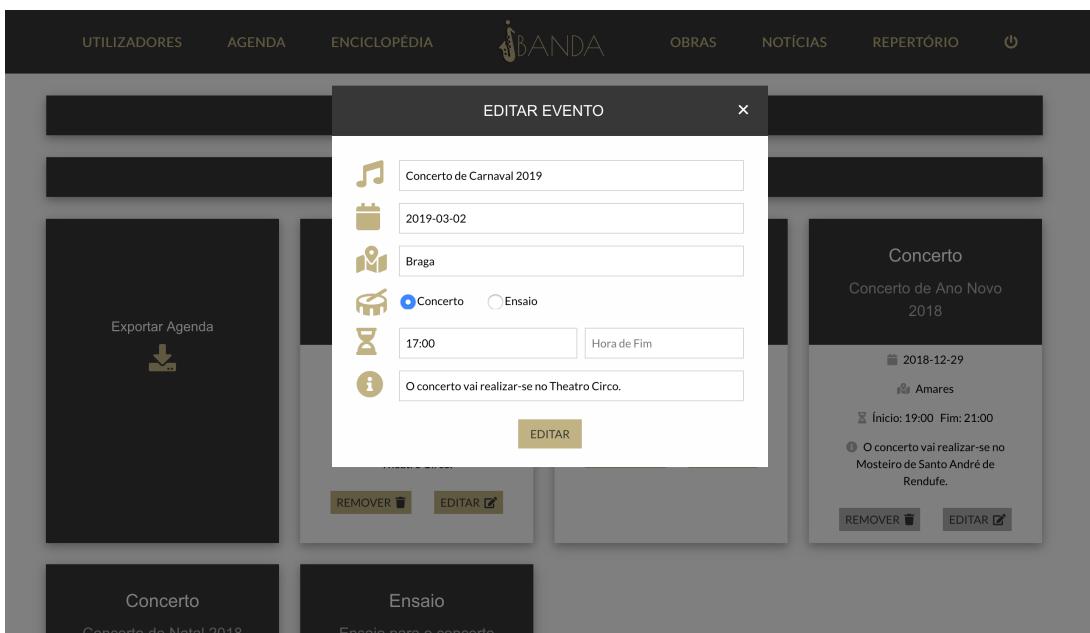


Figura 3.14: Formulário do administrador para edição de eventos.

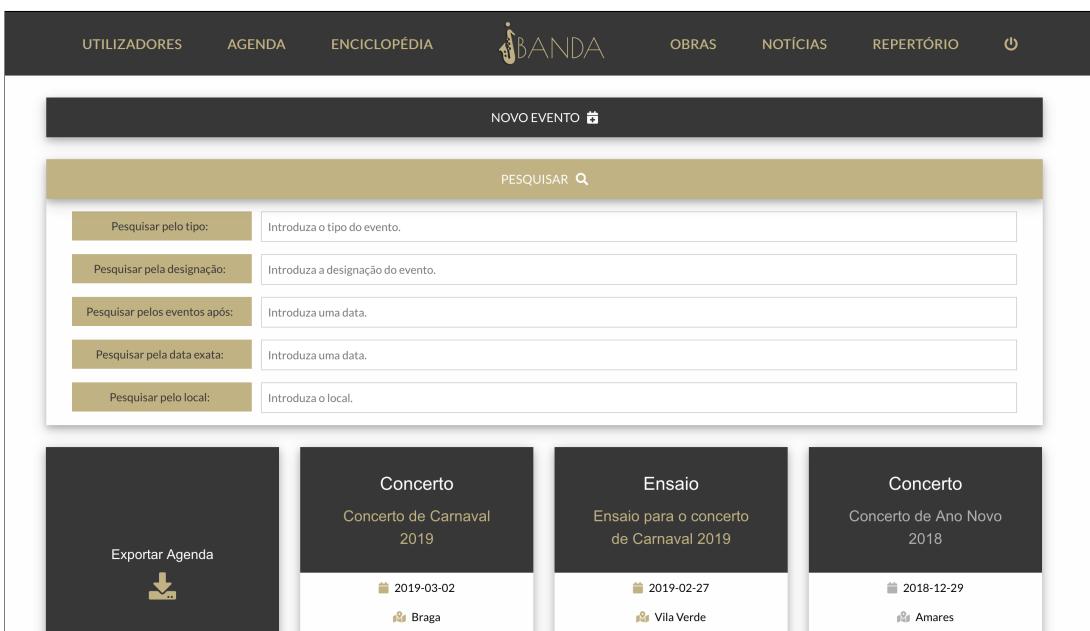


Figura 3.15: Campos de pesquisa por eventos.

## Notícias

O administrador pode criar ou editar notícias e consultar todas as notícias existentes. Pode também torná-las visíveis ou não para os utilizadores.

The screenshot shows the iBANDA website interface. At the top, there is a navigation bar with links: UTILIZADORES, AGENDA, ENCICLOPÉDIA, iBANDA (logo), OBRAS, NOTÍCIAS, REPERTÓRIO, and a power icon. Below the navigation bar, there is a dark header bar with the text "NOVA NOTÍCIA" and a camera icon. The main content area displays two news items.

**News Item 1:**



Banda Filarmónica deu Concerto de Beneficência  
Concerto gera receitas para ajudar o canil de Braga

O concerto do passado dia 15 de Janeiro, permitiu angariar dinheiro para ajudar todos os animais albergados no canil de Braga. Este encontra-se com grandes dificuldades monetárias devido ao número elevado de animais que alberga. Foi então possível

**News Item 2:**



Banda Filarmónica de São Brás dá concerto de Ano Novo  
Grupo Fole Percussion vai ser o convidado da Banda Filarmónica

A Banda Filarmónica de São Brás de Alportel dá um concerto de Ano Novo no próximo dia 13 de Janeiro, às 16h30, na Igreja Matriz daquela vila. Este concerto tem como objetivo a divulgação do trabalho desta banda, no âmbito da formação de músicos e de público, além de desejar um bom ano a todos através da música. Dedicado aos instrumentos solistas, o concerto deste ano conta com a presença e performance do grupo Fole Percussion, grupo constituído por três elementos: José Silva (guitarrista e vocalista), Jorge Alves (acordeão)

Figura 3.16: Página do administrador de consulta de notícias e visualização de uma notícia invisível.

The screenshot shows the "NOVA NOTÍCIA" (New News) creation form. The form consists of several input fields with icons on the left:

- Título (Title): Represented by double quotes (‘’).
- Subtítulo (Subtitle): Represented by double quotes with a dot (‘.’).
- Data (Date): Represented by a calendar icon.
- Palavras-Chave (Keywords): Represented by a hash tag (#).
- Link da imagem (Image Link): Represented by a photo icon.
- Corpo (Body): Represented by a speech bubble icon.

Below the form, there is a note: \* Campos de preenchimento obrigatório (Optional fields). A "ADICIONAR" (Add) button is located at the bottom right of the form area. At the very bottom, there is a horizontal image of a band performing.

Figura 3.17: Formulário do administrador para criação de notícias.

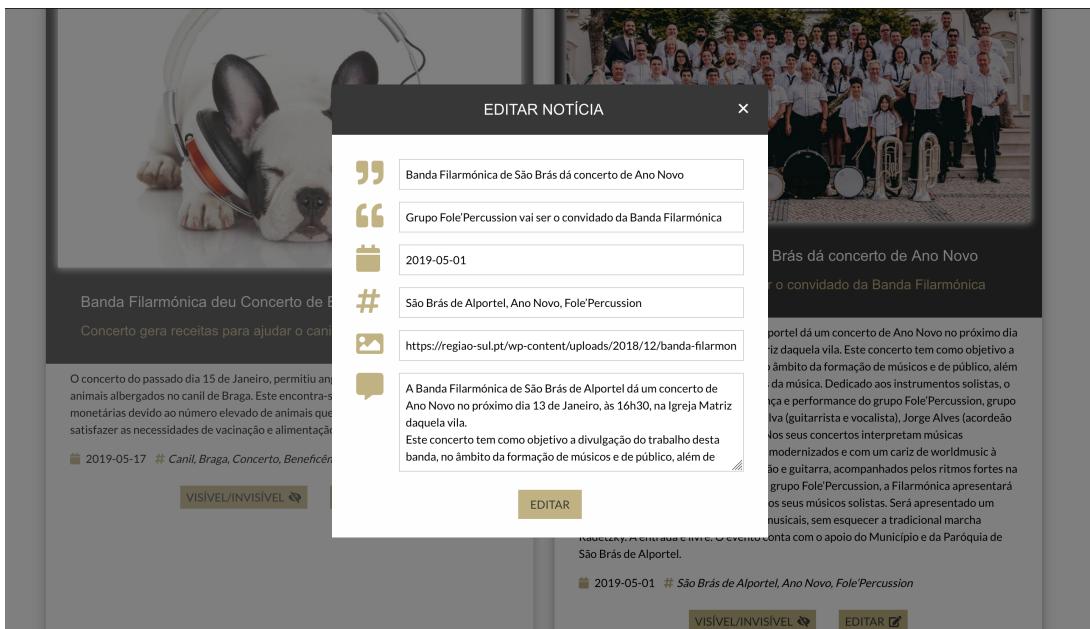


Figura 3.18: Formulário do administrador para edição de notícias.

## Enciclopédia

O administrador tem a capacidade de visualizar todas as partituras que foram submetidas na aplicação. Aqui pode também descarregar qualquer partitura disponível. Isto é conseguido listando todos os documentos que estão armazenados na pasta /public/partituras/, local onde são armazenados todos os ficheiros das partituras submetidos.

Partitura	Ação
12Abril-clarinete1.pdf	X
12Abril-clarinete2.pdf	X
12Abril-contrabaixo.pdf	X
12Abril-eufonio-fa-sib.pdf	X
12Abril-eufonio-sol-do.pdf	X
12Abril-eufonio-sol-sib.pdf	X
12Abril-flauta.pdf	X
12Abril-requinta.pdf	X
12Abril-saxA.pdf	X
12Abril-saxB.jpg	X

Figura 3.19: Página de visualização da enciclopédia e remoção de partituras.

## 3.4 Utilizador

Todas as rotas referentes ao utilizador estão definidas no ficheiro `/routes/users.js`. Estas estão restritas a este e em cada uma delas é feita a verificação do tipo do utilizador, se é 'Utilizador', que está autenticado.

### 3.4.1 Especificação de Requisitos

Os utilizadores da aplicação, que correspondem aos elementos da banda filarmónica, deverão ser capazes de:

- Consultar o seu perfil
- Editar o seu perfil
- Consultar obras
- Pesquisar obras
- Descarregar partituras
- Consultar o repertório da banda
- Pesquisar no repertório por eventos e obras
- Consultar os eventos da banda
- Pesquisar eventos
- Exportar a agenda
- Consultar notícias
- Consultar a enciclopédia, que contém todas as partituras armazenadas

### 3.4.2 Funcionalidades

#### Perfil

O utilizador tem a capacidade de visualizar e editar o seu perfil.

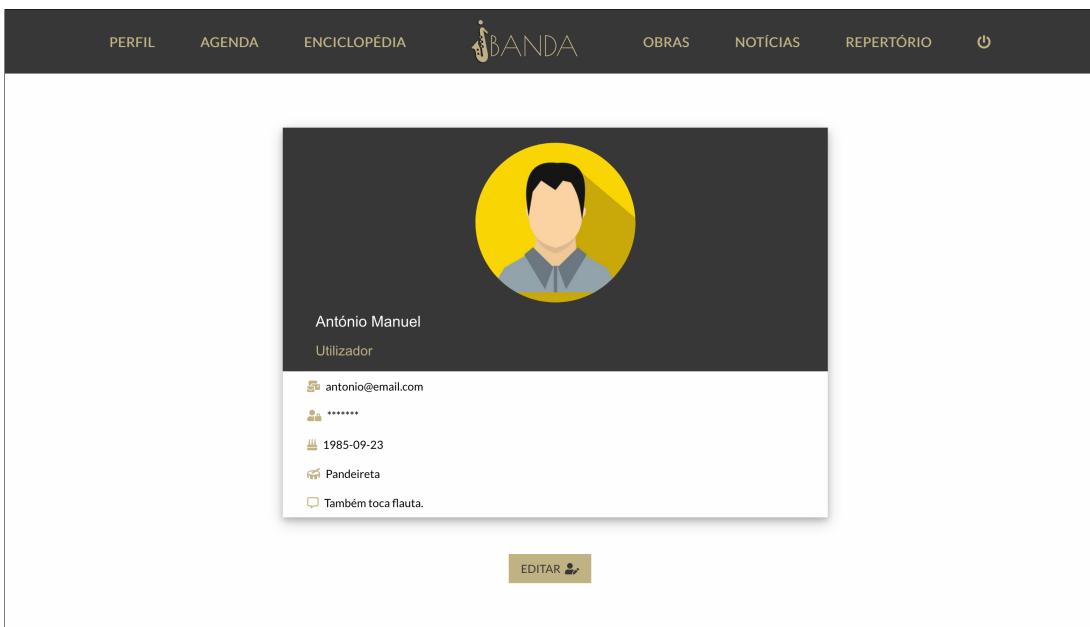


Figura 3.20: Visualização do perfil do utilizador.

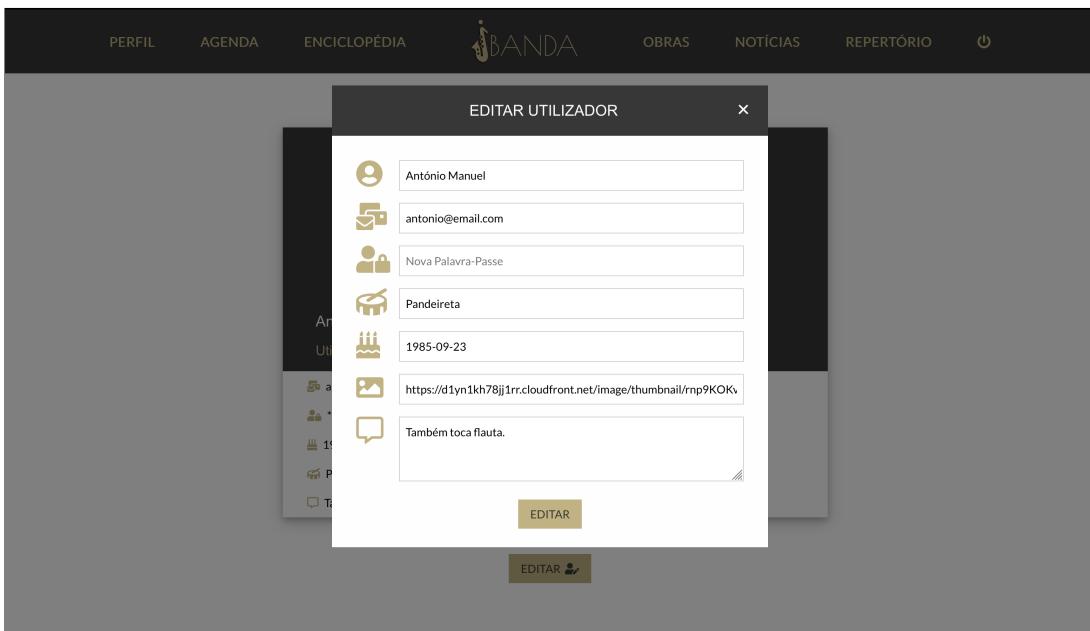


Figura 3.21: Formulário de edição do perfil do utilizador.

## Obras

O utilizador tem a capacidade de consultar todas as obras guardadas na aplicação. Para além disso, pode também fazer pesquisas pelas obras, por tipo, compositor ou título.

Figura 3.22: Página do utilizador de consulta e pesquisa das obras.

## Repertório

O utilizador pode consultar o repertório da banda e fazer a pesquisa quer pela designação do evento, quer pelo título da obra. É também permitida a visualização da informação de cada evento e de cada obra.

Figura 3.23: Página do utilizador de consulta e pesquisa do repertório.

Xabia

Marcha de Desfile

Salvador Salvá

Instrumentos

- Sax Baritono [Xabia-saxB.pdf](#)
- Bombo [Xabia-bombo-pratos.pdf](#)
- Pratos [Xabia-bombo-pratos.pdf](#)
- Clarinete Baixo [Xabia-clarinete-baixo.pdf](#)
- Clarinete [Xabia-clarinete1.pdf](#)
- Clarinete [Xabia-clarinete2.pdf](#)

Figura 3.24: Página de visualização das características de uma obra (semelhante aos eventos) do repertório.

## Agenda

O utilizador pode consultar a agenda da banda e pesquisar eventos por tipo, designação, data (exata ou os eventos após uma data) e local. Através da utilização do **moment** do JavaScript, assim como para o administrador, foi possível apresentar aos utilizadores os eventos já realizados em escala de cíntezos. Para além disto, podem também exportar a agenda completa em formato JSON.

PESQUISAR	
<a href="#">Exportar Agenda</a>	Concerto Concerto de Carnaval 2019 <span style="color: #ccc;">2019-03-02</span> <span style="color: #ccc;">Braga</span> <span style="color: #ccc;">Início: 17:00</span> <small>● O concerto vai realizar-se no Teatro Circo.</small>
	Ensaio Ensaio para o concerto de Carnaval 2019 <span style="color: #ccc;">2019-02-27</span> <span style="color: #ccc;">Vila Verde</span> <span style="color: #ccc;">Início: 19:00</span>
	Concerto Concerto de Ano Novo 2018 <span style="color: #ccc;">2018-12-29</span> <span style="color: #ccc;">Amares</span> <span style="color: #ccc;">Início: 19:00 Fim: 21:00</span> <small>● O concerto vai realizar-se no Mosteiro de Santo André de Rendufe.</small>
	Concerto Concerto de Natal 2018 <span style="color: #ccc;">2018-12-23</span> <span style="color: #ccc;">Braga</span> <span style="color: #ccc;">Início: 19:00 Fim: 21:00</span>
	Ensaio Ensaio para o concerto de Natal 2018 <span style="color: #ccc;">2018-12-22</span> <span style="color: #ccc;">Vila Verde</span>

Figura 3.25: Página do utilizador de consulta da agenda da banda.

The screenshot shows the iBANDA application's search interface. At the top, there are navigation links: 'PERFIL', 'AGENDA', 'ENCICLOPÉDIA', the 'iBANDA' logo, 'OBRAS', 'NOTÍCIAS', 'REPERTÓRIO', and a power icon. Below the navigation is a search bar with the placeholder 'PESQUISAR' and a magnifying glass icon. Underneath the search bar are five input fields for searching events by type, name, date, exact date, and location. To the left, there is a dark panel with a download icon and the text 'Exportar Agenda'. To the right, there are four event cards: 'Concerto Concerto de Carnaval 2019' (date: 2019-03-02, location: Braga, start: 17:00, note: 'O concerto vai realizar-se no Teatro Circo.'), 'Ensaio Ensaio para o concerto de Carnaval 2019' (date: 2019-02-27, location: Vila Verde, start: 19:00), 'Concerto Concerto de Ano Novo 2018' (date: 2018-12-29, location: Amares, start: 19:00, end: 21:00, note: 'O concerto vai realizar-se no Mosteiro de Santo André de Amares'), and a partially visible card for 'Concerto Concerto de Carnaval 2019'.

Figura 3.26: Campos de pesquisa por eventos.

## Notícias

O utilizador visualizar todas as notícias existentes, marcadas como visíveis pelo administrador.

The screenshot shows the iBANDA application's news interface. At the top, there are navigation links: 'PERFIL', 'AGENDA', 'ENCICLOPÉDIA', the 'iBANDA' logo, 'OBRAS', 'NOTÍCIAS', 'REPERTÓRIO', and a power icon. The main area displays two news items. The first item features a large image of a band, the text 'Banda Filarmónica de São Brás dá concerto de Ano Novo' and 'Grupo Fole'Percussion vai ser o convidado da Banda Filarmónica', and a detailed description about their New Year's concert. The second item features a cartoon illustration of band members and the text 'Banda Filarmónica de Santa Maria de Bouro' and a detailed description about their participation in the Concerto de Reis.

Figura 3.27: Página do utilizador de visualização de notícias.

## Enciclopédia

O utilizador tem a capacidade de visualizar todas as partituras que foram submetidas na aplicação. Aqui pode também descarregar qualquer partitura disponível.

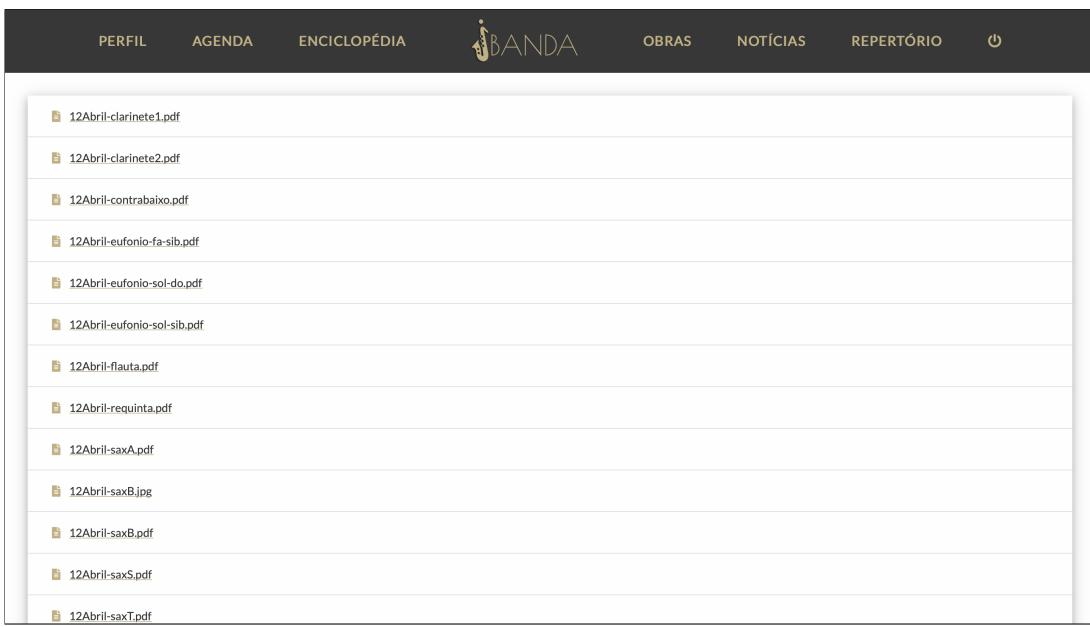


Figura 3.28: Página de visualização da enciclopédia.

## 3.5 Produtor

Todas as rotas referentes ao produtor estão definidas no ficheiro `/routes/prod.js`. Estas estão restritas a este e em cada uma delas é feita a verificação do tipo do utilizador, se é 'Produtor', que está autenticado.

### 3.5.1 Especificação de Requisitos

O produtor deverá ser capaz de:

- Catalogar novas obras
- Submeter partituras

### 3.5.2 Funcionalidades

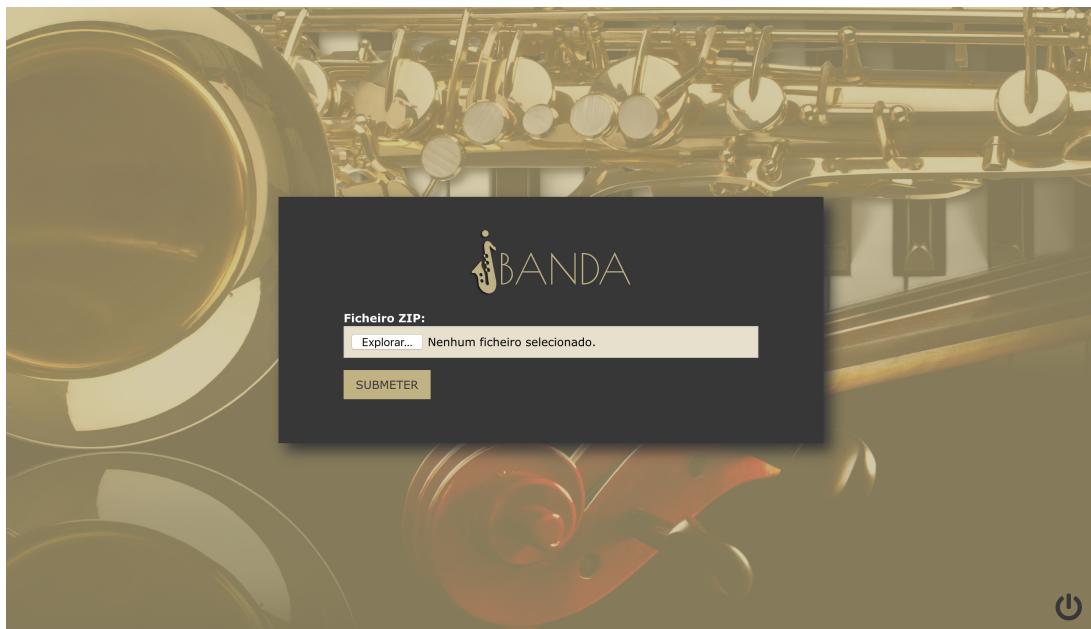


Figura 3.29: Página inicial do produtor.

O produtor tem a capacidade de submeter um ficheiro **ZIP** na aplicação. Este ficheiro contém um ficheiro **JSON**, que tem sempre o nome `iBanda-SIP.json`, que descreve a obra a ser catalogada, e, opcionalmente, um conjunto de partituras. Quando este submete o **ZIP**, este é movido para a pasta `/public/zips/` e depois descomprimido para `/public/partituras/`. Depois disto, o ficheiro **JSON** é processado, e a obra é guardada na base de dados. Finalmente, o ficheiro `iBanda-SIP.json` é eliminado da pasta `/public/partituras/`. Aqui, foi utilizado o módulo `'fs'` do `Node.js`, que permitiu ler e posteriormente eliminar o ficheiro **JSON**, e o módulo `'extract-zip'`, que permitiu fazer a extração dos ficheiros do **ZIP**.

### 3.5.3 Exemplo

Tomando como exemplo um ZIP que contém os ficheiros `CMistico-maestro.pdf`, `CMistico-saxB.pdf` e `iBanda-SIP.json`, em que o conteúdo deste último pode ser consultado em Listing 12, foi possível adicionar uma nova obra e submeter as partituras na aplicação.

```
{  
    "titulo": "Corpo Místico",  
    "tipo": "Marcha de Procissão",  
    "compositor": "Amílcar Morais",  
    "instrumentos": [  
        {  
            "nome": "Sax Barítono",  
            "partitura": {  
                "path": "CMistico-saxB.pdf"  
            }  
        },  
        {  
            "nome": "Maestro",  
            "partitura": {  
                "path": "CMistico-maestro.pdf"  
            }  
        }  
    ]  
}
```

```

        "path" : "CMistico-maestro.pdf"
    }
}
]
}

```

Listing 12: Conteúdo do ficheiro iBanda-SIP.json que descreve a obra 'Corpo Místico'.

Figura 3.30: Resultado obtido (vista do administrador) da pesquisa da obra com o título 'corpo místico' após o envio do ZIP.

## 3.6 Inserção de informação na BD recorrendo a GA's

Para além da informação submetida pelo administrador, é também possível introduzir informação na base de dados **Mongo** através de uma Gramática de Atributos desenvolvida para o efeito. No contexto do projeto **iBanda**, foram desenvolvidas duas Gramáticas de Atributos, uma com a capacidade de introduzir eventos na BD, e outra com a capacidade de introduzir notícias.

### 3.6.1 Inserção de Eventos

#### Gramática Independente de Contexto

```

grammar gic_eventos;

sistema : (evento)+;
;
evento : infoBasica infoOpcional;
;
infoBasica : 'DESIGNACAO:' designacao 'DATA:' data 'TIPO:' tipo 'LOCAL:' local;
;
infoOpcional : ('HORARIO:' horario)? ('INFORMAÇÕES:' informacoes)?
;
```

```

;
data : DATA
;
tipo : 'Concerto'
| 'Ensaio'
;
local : TEXT
;
horario : hinicio '-' hfim
| hinicio
;
hinicio : HORA
;
hfim : HORA
;
designacao : TEXT
;
informacoes : TEXT
;

/*Analizador Léxico */
TEXT: [a-zA-Z0-9áéíóúàèìòùãõç\ \.,\;]+;
HORA: [0-9] ':' [0-9] [0-9];
DATA: [0-9] [0-9] [0-9] '-' [0-9] [0-9] '-' [0-9] [0-9];
SEPARADOR: ('\r'? '\n' | ' ' | '\t')+ -> skip;

```

Listing 13: Gramática Independente de Contexto para descrição de eventos.

Usando a GIC especificada em Listing 13, definimos uma frase exemplo (Listing 14) de forma a fazer a sua validação.

```

DATA:2019-03-29
TIPO:Concerto
LOCAL:Amares
HORARIO:19:00-20:00
DESIGNACAO:Concerto de Páscoa 2019
INFORMAÇÕES:O concerto vai decorrer no Mosteiro de Tibães.

```

Listing 14: Exemplo de uma frase válida para eventos.

Esta frase exemplo apresenta apenas uma entrada de um evento na base de dados. No entanto, é possível definir vários eventos na mesma execução. Utilizando a opção 'Run in TestRig' disponível no menu 'Run' do *NetBeans* foi possível obter a árvore sintática mostrada na Figura 3.31.

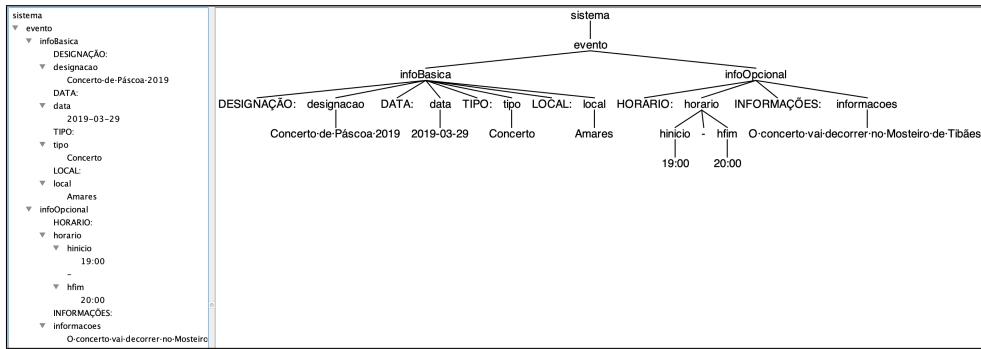


Figura 3.31: Diagrama da árvore de Sintaxe para a frase exemplo referente à inserção de eventos.

## Gramática de Atributos

```

grammar eventos;
@header {
    import com.mongodb.*;
    import com.mongodb.MongoClient;
    import com.mongodb.client.MongoCollection;
    import com.mongodb.client.MongoDatabase;
    import com.mongodb.client.model.Filters;
    import static com.mongodb.client.model.Filters.*;
    import static com.mongodb.client.model.Updates.*;
    import com.mongodb.client.model.UpdateOptions;
    import com.mongodb.client.result.*;
    import org.bson.Document;
    import org.bson.types.ObjectId;
}
sistema
@init {
    MongoClient mongoClient=new MongoClient("localhost", 27017);
    MongoDatabase database = mongoClient.getDatabase("amd");
    MongoCollection<Document> coll = database.getCollection("eventos");
}

@after {
    mongoClient.close();
}
: (evento {
    Document entry = new Document("data",(""+$evento.valordata)
        .append("designacao",(""+$evento.valordesignacao)
        .append("tipo",(""+$evento.valortipo)
        .append("local",(""+$evento.valorlocal)
        .append("horario", new Document("inicio",(""+$evento.valorhorarioI).append("hfim",
        "+$evento.valorhorarioF))
        .append("informacoes",(""+$evento.valorinformacoes);
    coll.insertOne(entry);
})+
;
evento returns [String valordesignacao, String valordata, String valortipo, String valorlocal, String
valorhorarioI, String valorhorarioF, String valorinformacoes]
: infoBasica infoOpcional {
    $evento.valordesignacao=$infoBasica.valordesignacao;
    $evento.valordata = $infoBasica.valordata;
    $evento.valortipo=$infoBasica.valortipo;
}

```

```

        $evento.valorlocal=$infoBasica.valorlocal;
        $evento.valorhorarioI=$infoOpcional.valorhorarioI;
        $evento.valorhorarioF=$infoOpcional.valorhorarioF;
        $evento.valorinformacoes=$infoOpcional.valorinformacoes;
    }
;

infoBasica returns [String valordata, String valortipo , String valorlocal, String valordesignacao]
: 'DESIGNAÇÃO:' designacao 'DATA:' data 'TIPO:' tipo 'LOCAL:' local {
    $infoBasica.valordesignacao=$designacao.valor;
    $infoBasica.valordata=$data.valor;
    $infoBasica.valortipo=$tipo.valor;
    $infoBasica.valorlocal=$local.valor;
}
;

infoOpcional returns [String valorhorarioI, String valorhorarioF, String valorinformacoes]
: ('HORARIO:' horario)? ('INFORMAÇÕES:' informacoes)? {
    try {
        $infoOpcional.valorhorarioI=$horario.valorI;
    } catch(Exception e) {
        System.out.println("Exceção no Horário de início.");
        $infoOpcional.valorhorarioI="";
    }
    try {
        $infoOpcional.valorhorarioF=$horario.valorF;
    } catch(Exception e) {
        System.out.println("Exceção no Horário de fim.");
        $infoOpcional.valorhorarioF="";
    }
    try {
        $infoOpcional.valorinformacoes=$informacoes.valor;
    } catch(Exception e) {
        System.out.println("Exceção nas Informações.");
        $infoOpcional.valorinformacoes="";
    }
}
;

data returns [String valor]
: DATA {$data.valor=$DATA.text;}
;

tipo returns [String valor]
: 'Concerto' {$tipo.valor="Concerto";}
| 'Ensaio' {$tipo.valor="Ensaio";}
;
local returns [String valor]
: TEXT {$local.valor=$TEXT.text.trim();}
;
horario returns [String valorI, String valorF]
: inicio {
    $horario.valorI=$inicio.valor;
} '-' hfim {
    $horario.valorF=$hfim.valor;
}
| inicio {
    $horario.valorI=$inicio.valor;
    $horario.valorF="";
}
;

```

```

hinicio returns [String valor]
: HORA {$hinicio.valor=$HORA.text;}
;

hfim returns [String valor]
: HORA {$hfim.valor=$HORA.text;}
;

designacao returns [String valor]
: TEXT {$designacao.valor=$TEXT.text.trim();}
;

informacoes returns [String valor]
: TEXT {$informacoes.valor=$TEXT.text.trim();}
;

/*Analizador Léxico */
TEXT: [a-zA-Z0-9áéíóúâèìòùãõç\ \.\,\;]+;
HORA: [0-9] [0-9] ':' [0-9] [0-9];
DATA: [0-9] [0-9] [0-9] [0-9] '-' [0-9] [0-9] '-' [0-9] [0-9];
SEPARADOR: ('\r'? '\n' | ' ' | '\t')+ -> skip;

```

Listing 15: Gramática de Atributos que permite a inserção de eventos.

Nesta Gramática de Atributos foram adicionados à GIC anterior atributos e regras de cálculo para permitir inserir eventos na base de dados utilizada pela aplicação **iBanda**.

Inicialmente foi necessário criar a ligação com o Mongo. Mais concretamente à base de dados 'amd', à coleção 'eventos'. Depois, durante o processamento do ficheiro de *input*, são guardados todos os valores de cada campo da coleção. Como existem campos que são opcionais, foram utilizados dois símbolos não-terminais para representar estes dois tipos de informação: *infoBasica* e *infoOpcional*. Como a gramática permite a inserção de vários eventos, de cada vez que é processado um evento este é introduzido na BD. No final é fechada a conexão ao Mongo.

Utilizando a frase exemplo descrita em Listing 14 e a Gramática de Atributos descrita em Listing 15, foi possível introduzir um novo evento na agenda da banda. Podemos comprovar isto na imagem 3.32 que mostra a pesquisa do utilizador pela designação '*páscoa*' do evento.

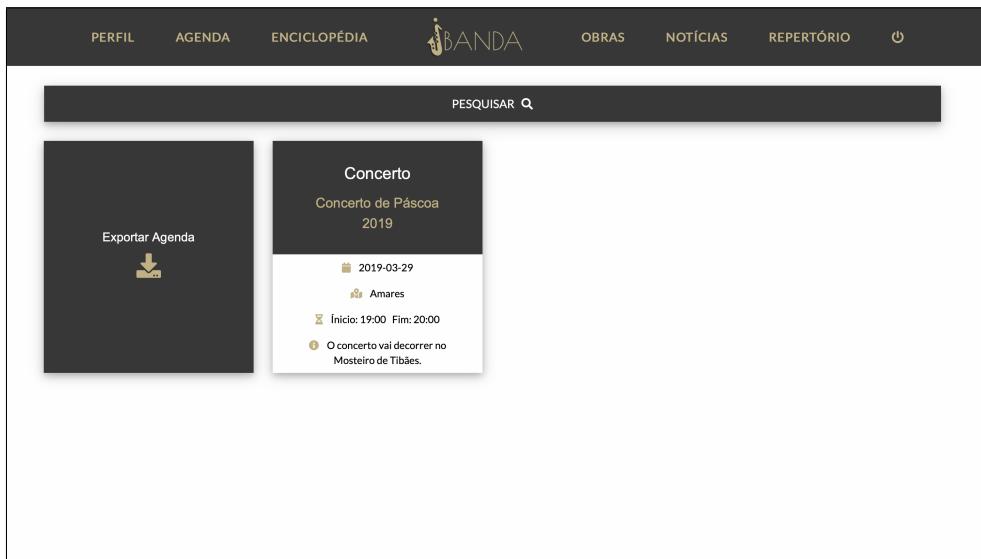


Figura 3.32: Visualização, do utilizador, do evento inserido através da GA.

### 3.6.2 Inserção de Notícias

#### Gramática Independente de Contexto

```

grammar noticias_gic;

sistema : (noticia)+
;
noticia : infoBasica infoOpcional
;
infoBasica : 'DATA:' data 'TÍTULO:' titulo 'CORPO:' corpo
;
infoOpcional : ('SUBTÍTULO:' subtítulo)? ('HASH:' hash)? ('IMG:' imgpath)? ('VISÍVEL:' visivel)?
;
data : DATA
;
titulo : TEXT
;
corpo : TEXT
;
subtítulo : TEXT
;
imgpath : TEXT
;
hash : TEXT
;
visivel : 'true'
| 'false'
;

/*Analizador Léxico */
TEXT: ((\'\')|\'\"') ~((\'\')|\'\"')* ((\'\')|\'\"'));
DATA: [0-9][0-9][0-9] '-' [0-9][0-9] '-' [0-9][0-9];
SEPARADOR: [ \t\n\r]+ -> skip;

```

Listing 16: Gramática Independente de Contexto para descrição de notícias.

Apresentada a gramática independente de contexto em Listing 16, vamos apresentar uma frase exemplo da mesma.

```

DATA: 2019-01-12
TÍTULO: 'Mosteiro de Rendufe acolhe Concerto de Ano Novo da Banda Filarmónica de Amares'
CORPO: 'A Banda Filarmónica de Amares vai organizar um Concerto de Ano Novo amanhã, 13 de Janeiro, pelas 17h, no Mosteiro de Santo André de Rendufe. A iniciativa conta com o apoio do Município de Amares e da Paróquia de Rendufe.'
HASH: 'Amares, Ano Novo, Mosteiro de Santo André de Rendufe'
IMG: 'https://cdn.cmjornal.pt/images/2017-07/img_818x455$2017_07_13_01_56_22_647840.jpg'
VISÍVEL: true

```

Listing 17: Exemplo de uma frase válida para notícias.

Esta frase exemplo apresenta apenas uma entrada de uma notícia na base de dados. No entanto, é possível definir várias notícias na mesma execução. Utilizando a opção 'Run in TestRig' disponível no menu 'Run' do *NetBeans* foi possível obter a árvore sintática mostrada na Figura 3.33.

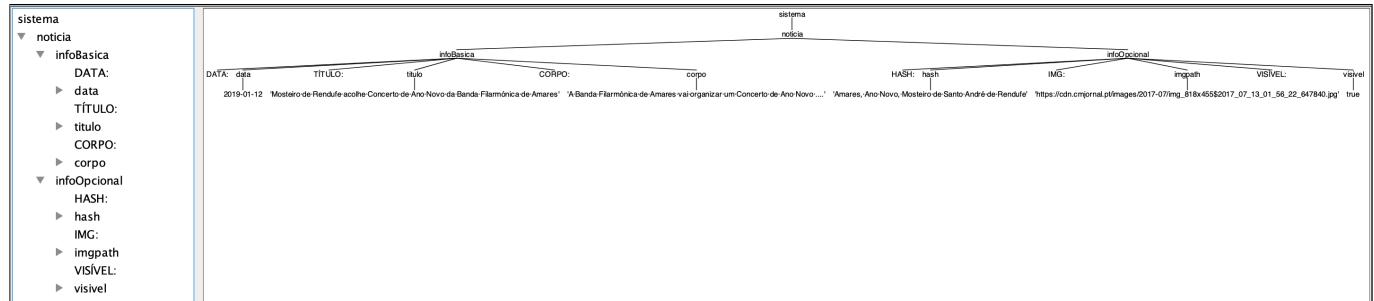


Figura 3.33: Diagrama da árvore de Sintaxe para a frase exemplo referente à inserção de notícias.

## Gramática de Atributos

```

grammar noticias;

@header {
    import java.io.*;
    import com.mongodb.*;
    import com.mongodb.MongoClient;
    import com.mongodb.client.MongoCollection;
    import com.mongodb.client.MongoDatabase;
    import com.mongodb.client.model.Filters;
    import static com.mongodb.client.model.Filters.*;
    import static com.mongodb.client.model.Updates.*;
    import com.mongodb.client.model.UpdateOptions;
    import com.mongodb.client.result.*;
    import org.bson.Document;
    import org.bson.types.ObjectId;
}

sistema
@init {

```

```

MongoClient mongoClient = new MongoClient("localhost", 27017);
MongoDatabase database = mongoClient.getDatabase("amd");
MongoCollection<Document> coll = database.getCollection("noticias");

}

@after {
    mongoClient.close();
}

: (noticia {
    Document entry = new Document("titulo", ""+$noticia.valortitulo)
        .append("subtitulo", ""+$noticia.valorsubtitulo)
        .append("data", ""+$noticia.valordata)
        .append("hash", ""+$noticia.valorhash)
        .append("img_path", ""+$noticia.valorimgpath)
        .append("corpo", ""+$noticia.valorcorpo)
        .append("visivel", ""+$noticia.valorvisivel);
    coll.insertOne(entry);
})+
;

noticia returns [String valordata, String valortitulo , String valorcorpo, String valorsubtitulo,
String valorhash, String valorimgpath, String valorvisivel]
: infoBasica infoOpcional {
    $noticia.valordata=$infoBasica.valordata;
    $noticia.valortitulo=$infoBasica.valortitulo;
    $noticia.valorcorpo=$infoBasica.valorcorpo;
    $noticia.valorsubtitulo=$infoOpcional.valorsubtitulo;
    $noticia.valorhash=$infoOpcional.valorhash;
    $noticia.valorimgpath=$infoOpcional.valorimgpath;
    $noticia.valorvisivel=$infoOpcional.valorvisivel;
}
;
infoBasica returns [String valordata, String valortitulo , String valorcorpo]
: 'DATA:' data 'TÍTULO:' titulo 'CORPO:' corpo {
    $infoBasica.valordata=$data.valor;
    $infoBasica.valortitulo=$titulo.valor;
    $infoBasica.valorcorpo=$corpo.valor;
}
;
infoOpcional returns [String valorsubtitulo, String valorhash, String valorimgpath, String
valorvisivel]
@init {
}

: ('SUBTÍTULO:' subtitulo)? ('HASH:' hash)? ('IMG:' imgpath)? ('VISÍVEL:' visivel)? {
    try {
        $infoOpcional.valorsubtitulo=$subtitulo.valor;
    } catch(Exception e) {
        System.out.println("Exceção no subtítulo.");
        $infoOpcional.valorsubtitulo="";
    }
    try {
        $infoOpcional.valorhash=$hash.valor;
    } catch(Exception e) {
        System.out.println("Exceção no Hash.");
        $infoOpcional.valorhash="";
    }
}

```

```

        try {
            $infoOpcional.valorimgpath=$imgpath.valor;
        } catch(Exception e) {
            System.out.println("Exceção no Imagem.");
            $infoOpcional.valorimgpath="";
        }
        try {
            $infoOpcional.valorvisivel=$visivel.valor;
        } catch(Exception e) {
            System.out.println("Exceção no Visível.");
            $infoOpcional.valorvisivel="";
        }
    }
}

data returns [String valor]
: DATA {$data.valor=$DATA.text;}
;

titulo returns [String valor]
: TEXT {$titulo.valor=$TEXT.text.replaceAll("\\n", "");}
;

corpo returns [String valor]
: TEXT {$corpo.valor=$TEXT.text.replaceAll("\\n", "");}
;

subtitulo returns [String valor]
: TEXT {$subtitulo.valor=$TEXT.text.replaceAll("\\n", "");}
;

imgpath returns [String valor]
: TEXT {$imgpath.valor=$TEXT.text.replaceAll("\\n", "");}
;

hash returns [String valor]
: TEXT {$hash.valor=$TEXT.text.replaceAll("\\n", "");}
;

visivel returns [String valor]
: 'true' {$visivel.valor="true";}
| 'false' {$visivel.valor="false";}
;

/*Analizador Léxico */
TEXT: ((\\'|\\\"') ~((\\'|\\\"')*) ((\\'|\\\"'));
DATA: [0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9];
SEPARADOR: [ \t\n\r]+ -> skip;

```

Listing 18: Gramática de Atributos que permite a inserção de notícias.

A Gramática de atributos em cima definida corresponde aos atributos e regras de cálculos adicionados à GIC com o objetivo de inserir novas notícias na base de dados da aplicação **iBanda**.

De início, como feito anteriormente, é necessária fazer a ligação ao Mongo. A base de dados utilizada é a "amd" mas desta vez à coleção das notícias. Como feito para os eventos, durante o processamento do ficheiro de *input* são guardados todos os valores de cada campo da coleção sendo que este está separado por dois tipos de informação: *infoBasica* e *infoOpcional*. Visto que decidimos que seria necessário o processamento de mais do que uma notícia decidimos que ao fim de cada notícia esta era inserida na BD. Finalizando todo este processamento é fechada a conexão ao Mongo.

Concluindo, para demonstrar os resultados obtidos com esta Gramática de Atributos, processando a nossa frase descrita em Listing 17, é mostrado o resultado através da visualização da notícia na página do utilizador.



Figura 3.34: Visualização, do utilizador, da notícia inserida através da GA.

## **Capítulo 4**

# **Conclusões**

Ao longo deste relatório está exposto todo o trabalho realizado para este projeto, assim como as decisões tomadas na implementação do código.

Consideramos que o *frontend* e o *backend* da aplicação estão bastante positivos. Apesar de termos estruturado bastante bem como o nosso projeto ia ser criado, não conseguimos desenvolver na totalidade os requisitos propostos no enunciado.

As gramáticas de atributos para a inserção de informação foram criadas com sucesso, assim como a ligação destas à base de dados e encontram-se presentes na ultima parte deste relatório.

Como trabalho futuro gostaríamos de completar os poucos requisitos do enunciado que nos faltam que dizem respeito à edição e criação de obras pelo administrador.