Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

Mestrado em Desenvolvimento de Software e
Sistemas Interactivos

Sistema de Gestão Documental em ambiente Web para Escolas do Ensino Básico e Secundário – Extensão de CMS open source

Nuno Miguel Ascenção Ramalho

Fevereiro 2011

Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

Sistema de Gestão Documental em ambiente Web para Escolas do Ensino Básico e Secundário – Extensão de CMS open source

Nuno Miguel Ascenção Ramalho

Trabalho de Mestrado
Mestrado em Desenvolvimento de Software e Sistemas Interactivos

Trabalho efectuado sob a orientação do
Professor Doutor Osvaldo Santos

Fevereiro 2011

DECLARAÇÃO

Nome Nuno Miguel Ascenção Ramalho_____

E-mail: ramalho.nmr@gmail.com Telefone: +351 962 829 168_

Bilhete de Identidade: _11290871_

Título do trabalho

Sistema de Gestão Documental em ambiente Web para Escolas do Ensino Básico e Secundário – Extensão de CMS *open source*_____

Orientador(es):

Professor Doutor Osvaldo Santos_____ Ano de conclusão: __2011__

Designação do Mestrado:

Mestrado em Desenvolvimento de Software e Sistemas Interactivos

Instituto Politécnico de Castelo Branco, ___/___/_____

Assinatura: _____

Agradecimentos

Expresso o meu sincero agradecimento às pessoas que, directa ou indirectamente, contribuíram para a concretização deste trabalho com especial destaque para:

- a minha esposa Ana Rita, pelo apoio e compreensão demonstrados ao longo do percurso decorrido no desenvolvimento desta dissertação;
- o meu amigo Eng. Hélio Lameiras, que também se envolveu num processo de mestrado na mesma instituição e com o qual pude partilhar vários.
- o Professor Doutor Osvaldo Santos que, desde o início, demonstrou disponibilidade para me orientar, apoiando e incentivando-me ao longo desta jornada.

Lista de abreviaturas / siglas

- API - *Application Programming Interface*.
- CMS - *Content Management System*.
- CSS - *Cascading Style Sheets*.
- DDMS – *Digital Document Management Systems*.
- DBMS – *Database Management Systems*.
- EDMS - *Electronic Document Management Systems*.
- FCCN - Fundação para a Computação Científica Nacional.
- FTP – *File Transfer Protocol*.
- GED – Gestão Electrónica de Documentos.
- GEP - Gabinete de Estatística e Planeamento da Educação.
- HTML - *HyperText Markup Language*.
- IIS – *Internet Information Server*.
- LAN – *Local Area Network*.
- LDAP - *Lightweight Directory Access Protocol*.
- PDF – *Portable Document Format*.
- RPC – *Remote Procedure Call*.
- RSS - *Really Simple Syndication*.
- SGDEscolas – Sistema de Gestão Documental para Escolas do Ensino Básico e Secundário.
- SIS - *Single Instance Storage*.
- URI - *Uniform Resource Indicator*.
- URL - *Uniform Resource Locator*.
- WAN - *Wide Area Network*.
- WYSIWYG - *"What You See Is What You Get"*.
- XML - *Extensible Markup Language*.

Palavras chave

Extensão para CMS, Joomla, Sistema de Gestão Documental, Escolas Básicas e Secundárias.

Resumo

A presente dissertação estuda a criação de um sistema de gestão documental, direccionado a escolas do ensino básico e secundário, capaz de ser integrado num sistema gestor de conteúdos *open source*. É realizado o estudo da criação de uma extensão para um dos *Content Management System (CMS)* mais utilizados actualmente, o Joomla, tirando assim proveito das potencialidades de um sistema já fortemente testado e desenvolvido, usufruindo das vantagens da utilização deste tipo de plataformas. A extensão desenvolvida foca-se principalmente na actividade do professor, como criador e gestor de documentos relativos à sua actividade na escola.

Keywords

CMS extension, Joomla, Document Management System, Basic and secondary schools.

Abstract

This dissertation studies the creation of a document management system targeted at primary schools and secondary schools that can be integrated into a open source content management system. It is performed the study of the extension creation to the one of the Content Management System (CMS) most actually used, the Joomla, taking advantage of a system that is heavily developed and tested, benefiting of using this type of platforms. The developed extension focuses primarily on the teacher's activity as document creator and manager concerning his activities at school.

Índice geral

Índice de figuras

Índice de tabelas

1. Introdução

Neste capítulo pretende-se efectuar o enquadramento do tema da dissertação, identificando-se os objectivos propostos e a organização do documento.

1.1. Contexto da temática

O aparecimento de sistemas de gestão de documentos digitais (DDMS – *Digital Document Management Systems*) tornou-se numa grande vantagem para as organizações. Hoje, recorrendo a este tipo de sistemas, as organizações podem conciliar documentos em formato papel e digital no mesmo ambiente de procura, distribuição e armazenamento, com facilidade e flexibilidade. A nova geração deste tipo de sistemas é desenhada de modo que as organizações não necessitem de pessoal especializado para a sua implementação, reduzindo assim os custos de aquisição (Craine, 2008).

A gestão de documentos é geralmente definida como um mecanismo de captura, controlo e centralização de documentos no formato papel ou digital num sistema integral (Craine, 2008). Os Sistemas de Gestão Documental aplicam-se aos mais variados tipos de organização, trazendo como principais benefícios a redução de custos com papel, a necessidade de menor espaço de armazenamento de documentos, melhoria no acesso e segurança, a facilidade na recuperação em caso de desastre, a optimização de processos de negócio, o aumento da produtividade dos funcionários, a melhoria na prestação de serviços ao cliente, bem como a melhoria no cumprimento de normas regulamentares e um maior retorno no investimento nas Tecnologias de Informação (Craine, 2008; CMC, 2007; Zylab, n.d.).

Independentemente do tipo ou formato, os documentos são ferramentas de suporte às funções de qualquer sistema organizacional requerendo, qualquer actividade ou processo administrativo, um ou mais documentos para que sejam concluídos. Um dos tipos de organização onde existem inúmeros documentos relativos à sua actividade é a escola. Actualmente, as escolas de ensino básico e secundário deparam-se com um grande volume de documentos que circulam entre os diversos organismos internos. A gestão de todos esses documentos torna-se difícil, uma vez que, na maioria dos casos, a utilização das Tecnologias de Informação e Comunicação passa apenas pela criação de documentos em formato digital, para que depois sejam impressos e arquivados de forma arcaica em tradicionais dossiês, esquecendo o ficheiro criado na ferramenta de *Office*. "*Na gestão administrativa das escolas, observa-se que o leque de processos informatizados é reduzido e que apenas 5% das escolas utilizam sistemas de gestão documental electrónica*" (GEPE 2007, p.11). Os professores despendem demasiado tempo com aspectos burocráticos relacionados com a sua actividade nas escolas, criando inúmeros documentos e perdendo imenso tempo na sua organização. Em "*escolas estudadas pelo projecto O Risco Educativo no Ensino Básico. Focagens para uma Intervenção Integrada em torno do Sucesso Educativo, vários professores se referiam ao excesso de burocracia como um obstáculo ao desempenho da função lectiva*" (Costa 2008, p.63).

A falta de um sistema capaz de gerir eficazmente os diversos documentos que circulam nas escolas pode provovar vários problemas, nomeadamente, a existência de documentos repetidos,

não normalizados, dificuldades no seu acesso, destruição ou perda, autenticidade, prolongamento excessivo no seu ciclo de vida, etc. As escolas de hoje têm consciência da importância da adaptação às novas tecnologias. No entanto, a falta de recursos financeiros e as poucas alternativas dedicadas à gestão documental levam a que se continue a ignorar a necessidade de implementação de um sistema deste tipo.

Criar um sistema de raiz que seja fiável, seguro, funcional, que permita gerir os documentos que circulam numa escola básica ou secundária, pode tornar-se moroso e difícil de implementar. É facto que já existem boas aplicações de gestão documental, que até disponibilizam versões sem custos, como o caso do sistema Alfresco (Alfresco, 2010) e do knowledgeTree (knowledgeTree, 2010). No entanto, para que estes sistemas sejam implementados e adaptados correctamente a uma organização escolar, é necessária a intervenção de pessoal especializado, o que pode acarretar custos para a instituição. Dadas as inúmeras funcionalidades que estes sistemas fornecem, tornam-se complexos exigindo esforço por parte dos professores para a sua correcta utilização.

Segundo dados apresentados no estudo realizado pelo GEPE (2007, p.47) sobre a modernização tecnológica do ensino em Portugal, a proporção de escolas que utilizam sistemas de gestão electrónica de documentos é muito reduzida. Este facto demonstra que existem impedimentos que levam à não utilização destes sistemas. O mesmo estudo realizado pelo GEPE (2007, p. 51) revela que "*mesmo em escolas bem equipadas e cujos agentes têm a formação adequada, a utilização de tecnologia enfrenta normalmente alguma resistência por parte de alguns docentes, motivada quer pelo cepticismo em relação aos benefícios da utilização das TIC, quer pela alteração do status quo que implica, quer pelo acréscimo de tempo e de esforço de preparação que exige*.".

Numa realidade em que existe dificuldade na implementação de sistemas de gestão electrónica de documentos nas escolas e resistência à sua utilização, uma solução pode passar por capacitar uma ferramenta, com a qual os professores já estejam familiarizados, de modo a fornecer funcionalidades adequadas à gestão electrónica de documentos nestas instituições. Para facilitar a utilização em vários tipos de máquinas ou dispositivos e a partir de diversos locais, a utilização de aplicações Web é bastante vantajosa, uma vez que para serem executadas basta existir um navegador de Internet. Em termos de aplicações Web utilizadas pelas escolas, a plataforma de *e-learning* Moodle e o CMS Joomla são as mais utilizadas, até porque a Fundação para a Computação Científica Nacional (FCCN) fornece servidores Web gratuitos às instituições de ensino básico e secundário, no âmbito do Projecto Tecnológico. Estes servidores, acessíveis pelas escolas através do portal Serviços Internet às Escolas (FCCN, n.d.) já contêm as plataformas Moodle e Joomla pré-instaladas. Estas duas plataformas já são bastante conhecidas. Segundo o GEPE (2007), nas escolas que já utilizam plataformas de gestão de aprendizagem, cerca 60% utilizam o Moodle. Segundo um estudo efectuado por Bárcia (2008) revela que cerca de 20% das escolas inquiridas utilizam a plataforma Joomla para a gestão de conteúdos. A plataforma Moodle tem, em algumas escolas, vindo a ser utilizada para armazenar documentos, "*é utilizada como canal de interacção e comunicação entre agentes e como canal de distribuição de material de aula*" (GEPE 2007, p46). No entanto, não é uma ferramenta adequada à gestão

documental. Acrescentar funcionalidades à plataforma Moodle de modo a torná-la num sistema de gestão electrónica de documentos, sem ter de refazer o código na sua *framework,* teria limitações. Nesta plataforma, apenas são adicionadas novas funcionalidades através de módulos, estando a mesma especificamente orientada para actividades de sala de aula e gestão de aprendizagens (Moodle, 2010).

Relativamente à plataforma CMS Joomla, esta tem vindo a demonstrar grandes capacidades de adaptação às mais variadas tarefas. Graças às potencialidades da sua *framework* é possível a extensão do CMS Joomla ao nível de módulos, *plugins* e componentes, permitindo transformar esta plataforma num sistema capaz de fazer tudo o que é possível, fazer ao nível de aplicações para a Internet (Joomla, 2010). Basta navegar até ao site oficial de extensões para esta plataforma (Joomla! Extensions Directory, 2010) para constatar esse facto. Existem dezenas de categorias de extensões e mais de cerca de seis mil e seiscentas extensões oficialmente aprovadas pela comunidade. Assim, beneficiando de todas as potencialidades desta plataforma, da vantagem de não ter que se projectar todo um sistema de raiz, aproveitando todo o estudo, desenvolvimento e testes já efectuados que garantem a sua estabilidade, fiabilidade e segurança, desenvolver uma extensão para o Joomla, com o objectivo de a tornar num sistema capaz de fazer gestão electrónica de documentos criados pelos professores, no âmbito da sua actividade na escola, só poderá trazer vantagens e bons resultados para este tipo de instituições.

1.2. Objectivos

1.2.1. Objectivos gerais

Para melhorar os aspectos relacionados com a gestão de documentos nas escolas básicas e secundárias, a presente dissertação tem como objectivo realizar o estudo do desenvolvimento de um **Sistema de Gestão Documental *open source* em ambiente *Web*, utilizando a *Joomla Framework API (Application Programming Interface)***. O sistema deve consistir num componente que possa ser completamente integrado no CMS Joomla, a partir do seu gestor de extensões. O componente terá como nome SGDEscolas (com_sgdescolas) e focar-se-á na actividade do professor como criador e gestor de documentos relativos à sua actividade profissional. Tem como principais requisitos a rapidez e facilidade de implementação, ser utilizável por qualquer pessoa que detenha apenas os conhecimentos básicos da utilização de um computador e não possuir qualquer custo associado à implementação ou utilização.

1.2.2. Objectivos específicos

O presente trabalho visa atingir os seguintes objectivos específicos:

- **Estudar o desenvolvimento de componentes para o CMS Joomla** – esta plataforma tem características próprias que permitem, através da sua *framework*, o rápido desenvolvimento de novas aplicações. Estas aplicações integradas na plataforma Joomla são denominadas de extensões e a sua forma mais complexa caracteriza-se de componente.

- **Criar um componente para gestão de documentos** – aproveitando todas as potencialidades da plataforma Joomla e da sua *framework* será desenvolvido um componente que permitirá às escolas básicas e secundárias a gestão de documentos criados pelos professores no âmbito da sua actividade profissional. Esta aplicação deverá permitir as seguintes funcionalidades:
    - **Criação, edição e eliminação dos dados relativos aos utilizadores do componente** – como todas as aplicações de gestão, é importante recolher os dados relativos aos utilizadores, permitindo assim controlar acessos e acções com base nas suas características;
    - **Procura, listagem e apresentação dos dados dos utilizadores -** ter o rápido acesso à lista de utilizadores e aos seus dados facilita a gestão. Em cada ano lectivo existem mudanças no corpo docente e nas funções ou cargos associados a cada elemento. Os sistemas utilizados pelo corpo docente devem facilitar as tarefas associadas a estas mudanças;
    - **Criação, carregamento, edição, eliminação, procura e listagem de documentos -** o sistema deve permitir que estas tarefas não sejam um entrave à sua utilização. Dado que os principais utilizadores são professores e estes já despendem demasiado tempo com aspectos burocráticos, a gestão dos seus documentos deve ser simples, intuitiva e rápida. O não cumprimento destes requisitos pode levar ao fracasso do sistema;
    - **Publicação de documentos -** nas escolas existem documentos que devem ser publicados e disponibilizados a todo o corpo docente. A publicação dos documentos pode ainda estar associada a um período de tempo limitado;
    - **Registo e entrega oficial de documentos** – alguns documentos requerem a sua aceitação, registo e aprovação por parte dos órgãos de gestão da escola;
    - ***Workflow*** – existem determinados documentos que circulam de utilizador para utilizador até estarem concluídos e aprovados. Normalmente, correspondem a documentos que devem ser oficialmente entregues aos órgãos de gestão. Um documento deste tipo é criado, enviado para aprovação e devolvido para que seja revisto e corrigido, caso não seja aprovado;
    - **Partilha de documentos** – por vezes é pedido a um conjunto de docentes que elaborem determinado documento. Para facilitar o trabalho colaborativo, o sistema deve permitir a partilha de documentos, quer para consulta ou para edição;
    - **Controle das acções por parte dos utilizadores aos documentos** – em termos de gestão documental, é muito importante saber e controlar o que os utilizadores fazem aos documentos e quando executam essas tarefas;

- **Controlo no acesso a documentos** – os utilizadores só devem ter acesso aos documentos que realmente podem consultar ou trabalhar;
- **Criação, edição e eliminação de dossiers** – na organização escolar existem normalmente diversos dossiers que estão associados a determinados cargos ou funções que os professores desempenham, devendo os diversos documentos estar associados a estes dossiers;
- **Notificações** – o sistema deve permitir que sejam enviadas notificações por *e-mail* de determinadas tarefas executadas pelos seus utilizadores;
- ***Backup*** – é importante que, em qualquer altura, o administrador possa criar e obter do sistema todos ou parte dos documentos existentes, quer para salvaguarda, quer para posterior consulta.

1.3. Organização do documento

O presente documento, além da introdução está organizado em mais quatro capítulos:

- **Capítulo 2: Tecnologias de Gestão Documental e Conteúdos -** neste capítulo é realizada uma abordagem às características e funcionamento dos sistemas de gestão de documentos. Por forma a compreender como se pode integrar um sistema deste tipo num sistema de gestão de conteúdos, é feita uma análise à arquitectura de um *Content Management System* (CMS). Para realizar a integração de uma aplicação de gestão de documentos com o CMS Joomla foi feito o estudo do funcionamento desta plataforma e da sua *framework*. A utilização da *framework* do Joomla para a criação de extensões desta plataforma requer o conhecimento do seu padrão de programação para a criação de componentes (para esta plataforma), aspectos estes abordados também neste capítulo.
- **Capítulo 3: Desenvolvimento do componente SGDEscolas** – descrição do processo de desenvolvimento do componente responsável pela gestão dos documentos nas escolas básicas e secundárias, garantindo a total integração com a plataforma Joomla.
- **Capítulo 4: Testes e resultados** – demonstração das principais funcionalidades implementadas no componente desenvolvido.
- **Capítulo 5: Conclusões** - Neste capítulo apresentam-se as conclusões do trabalho efectuado, as principais contribuições e algumas propostas para trabalhos futuros.

2. Tecnologias de Gestão Documental e Conteúdos

Neste capítulo, pretende-se fazer uma clarificação dos conceitos que são objecto de estudo deste trabalho. O sistema criado no âmbito deste trabalho implica o cruzamento da tecnologia associada aos sistemas de gestão documental com a tecnologia associada aos sistemas de gestão de conteúdos. O estudo destes sistemas em geral e do CMS Joomla em particular permitirá a criação de um sistema que beneficie das suas potencialidades que seja de rápido desenvolvimento e de fácil implementação.

2.1. Sistemas de Gestão Documental

2.1.1. Gestão Electrónica de Documentos

A palavra documento é definida como sendo um qualquer objecto elaborado com o fim de reproduzir ou representar uma pessoa, um facto, um dito ou um acontecimento, um escrito que serve de prova, consistindo no âmbito da informática, num ficheiro que contém dados gerados por uma aplicação (In Infopédia 2010). Tanto em formato papel como em formato electrónico, os documentos integram os processos relativos ao funcionamento das organizações, podendo ser produzidos, editados, transferidos, partilhados, armazenados, avaliados e destruídos. Os sistemas capazes de organizar e gerir os documentos de uma organização através de um *software* específico denominam-se de *Document Management Systems* (DMS), *Electronic Document Management Systems* (EDMS) ou Gestão Electrónica de Documentos (GED). Um DMS actua como um polícia sinaleiro no centro de um cruzamento de informação: controla, organiza e encaminha o fluxo de informação (Heckman 2008). A implementação destes sistemas está normalmente associada à expressão "*paperless office*", que define um ambiente de trabalho, onde se foram eliminando os documentos em formato papel.

A adopção por parte das organizações de DMS, segundo Craine (2008), acarreta diversos benefícios, sendo os mais comuns:

- redução dos custos associada à diminuição da utilização de papel;
- optimização de processos de negócio;
- aumento da produtividade dos funcionários;
- aumento na qualidade dos serviços prestados aos clientes;
- aumento do retorno do investimento nas Tecnologias de Informação(TI);
- eliminação do espaço ocupado por arquivos;
- aumento da facilidade de acesso a documentos;
- aumento na segurança;
- melhoria na recuperação em caso de catástrofe;
- melhoria na conformidade e redução de riscos legais.

Em termos genéricos, um DMS contribui fortemente para a melhoria dos serviços prestados pelas organizações e, quando associados a soluções de digitalização, fax e e-mail, permitem gerir todos os documentos de uma organização.

Um DMS pode ser implementado em diversos sectores (financeiro, administração pública, indústria, serviços) e pode assumir maior ou menor complexidade consoante a organização onde for implementado. No entanto, qualquer DMS lida com dois aspectos cruciais: a informação que consta no documento e a informação acerca do documento em si.

Do ponto de vista da gestão de documentos, é importante distinguir entre a informação primária, a informação que consta no documento, e a informação secundária, informação acerca do documento. A informação secundária, normalmente referida como *meta-data* (meta-dados), é o que permite aos seres humanos e aos sistemas gestores de documentos a pesquisa, devolução e abertura dos documentos (Björk, 2003). Tanto a informação primária como a informação secundária encontram-se normalmente num repositório central.

Para a gestão de toda a informação relativa aos documentos, como por exemplo os meta-dados, os detalhes do ciclo de vida do documento, as permissões dos utilizadores, etc., os DMS são, em muitas situações, desenvolvidos de modo a comunicarem com *Database Management Systems* (DBMS), como por exemplo, SQL Server, Oracle e MySQL, aproveitando as suas potencialidades e facilitando o desenvolvimento do próprio sistema.

Actualmente, com a proliferação da Internet e com o avanço das tecnologias a ela associada, são cada vez mais os sistemas DMS a utilizar esta rede como meio de comunicação, os servidores *Web* como centros de armazenamento e os *browsers* como plataforma para as interfaces dos utilizadores.

2.1.2. Principais elementos de um *software* DMS

O *software* DMS é responsável por um conjunto de funções necessárias à gestão dos documentos de uma determinada organização, desde a sua introdução no sistema até à sua destruição. Existem vários aspectos a ter em conta:

- **infra-estrutura subjacente** – embora não faça parte da aplicação por si só, é condição prévia ter uma adequada infra-estrutura a servir de base ao DMS. A infra-estrutura consiste no conjunto computadores, postos de trabalho e servidores que se encontram interligados através de LANs e/ou WANs. Deve ter características como a independência do sistema operativo de rede, formato dos ficheiros, localização, tamanho do nome do ficheiro e monitorização das ligações (Cleveland, 1995);
- **ferramentas de suporte à criação de documentos** – estas ferramentas guiam os utilizadores durante o processo de criação dos documentos. Normalmente fazem parte da interface de criação de documentos no DMS e permitem registar os meta-dados, data e hora de criação ou revisão (Cleveland, 1995);
- ***workflow*** – automatização de parte ou totalidade de um processo de negócio, durante o qual os documentos, informações ou tarefas são passadas de um participante para outro, de acordo com um conjunto de regras procedimentais (WFMC, 1999). Suporta funções como a criação, revisão, encaminhamento, comentários, aprovação, pontos de decisão e estabelecimento de prazos e metas. O *workflow* é um aspecto central na gestão de documentos, uma vez que permite

às organizações o controlo e melhoria da eficácia do fluxo de documentos que suportam o seu negócio (Cleveland, 1995);

- **armazenamento** - o DMS deve proporcionar um sistema de armazenamento e pesquisa de documentos que seja seguro, eficaz, capaz de acompanhar o aumento do volume de documentos e os avanços tecnológicos. Os DMS recorrem a sistemas gestores de bases de dados para armazenar as informações referentes aos documentos, como os meta-dados e a sua localização no dispositivo de armazenamento, ou mesmo os próprios documentos (Cleveland, 1995);
- **mecanismos de controlo dos documentos** – controlam a informação relativa à entrada e saída do documento no sistema, informação do autor, editor, segurança e controlo de versões (Cleveland, 1995);
- **recuperação, visualização, impressão e distribuição dos documentos** - modo como os documentos são apresentados aos utilizadores. Os DMS devem estar preparados para distribuir os documentos em diferentes formatos, de acordo com o seu propósito. A informação relativa à distribuição, visualização e impressão dos documentos pode ser aproveitada para a definição de estratégias de negócio (Cleveland, 1995).

2.1.3. DMS e o ciclo de vida dos documentos

A entrada de um documento num *software* DMS dá início ao seu ciclo de vida. Em termos genéricos, segundo Curry (2008), o ciclo de vida de um documento passa pelas seguintes fases:

- **criação** – métodos para a criação, inicialização, colaboração no desenvolvimento de um novo documento;
- **localização** – deverá haver uma localização física, onde os documentos são armazenados e acedidos. Normalmente, a maior parte dos DMS necessita de uma única instância para armazenamento de um documento (SIS - S*ingle Instance Storage*), a fim de garantir que existe apenas uma versão válida.
- **autenticação /aprovação** - métodos utilizados para garantir que um documento é totalmente analisado e aprovado antes de ser considerado oficial, segundo as normas da organização.
- ***workflow*** – descreve os passos necessários para a passagem dos documentos de pessoa para pessoa , de acordo com os objectivos, como por exemplo, para a aprovação e publicação.
- ***filing*** - organizar os ficheiros num local físico e associar os respectivos meta-dados para que o documento seja facilmente encontrado.
- **distribuição** - descreve o método utilizado para que o documento chegue aos devidos destinatários.
- **recuperação** - método utilizado na procura de documentos, como por exemplo a procura de palavras-chave.
- **segurança** - métodos utilizados para garantir a integridade e segurança durante o ciclo de vida do documento.

- **retenção** - existem políticas e práticas utilizadas pelas organizações que estabelecem quanto tempo cada tipo de documento fica retido na organização.
- **arquivo** – consiste num subconjunto de políticas de retenção. O arquivo foca-se na preservação dos documentos que já terminaram a sua "vida activa", por um longo período de tempo e num formato que ainda permita a sua consulta.

Num DMS os documentos podem passar por todas as fases acima descritas. No entanto, ao observarmos de um modo mais generalista, podemos agrupá-las em 5 fases principais:

- criação;
- armazenamento;
- organização;
- recuperação/visualização;
- preservação ou eliminação.

Em qualquer DMS, seja ele simples ou de grande complexidade, todos os documentos passam por estas 5 fases.

2.2. CMS – *Content Management Systems*

2.2.1. O que é um CMS?

Um CMS consiste num *software* através do qual é possível criar, editar e gerir conteúdos. Estes conteúdos podem ser de diversos tipos ou formatos, como por exemplo, documentos de texto, ficheiros de áudio/vídeo, imagens, etc. Normalmente estes sistemas correm em ambiente *Web* e a sua utilização ou gestão não requer conhecimentos técnicos ao nível da criação de páginas para a Internet.

Os sistemas de gestão de conteúdos tornam a manutenção de um *website* mais prática e menos dispendiosa. Quando os *websites* eram construídos através da criação de um conjunto de páginas HTML estáticas, o processo de manutenção tornava-se muito dispendioso, uma vez que requeria a alteração de código em cada página para a introdução de novos textos e imagens. Este tipo de operação obrigava à contratação de programadores para realizar estas operações.

Como refere Shereves (2010, p.4), a utilização de um CMS traz vantagens como:

- aumento no controlo sobre o *website*;
- contribui para a melhoria no tempo de colocação no mercado de produtos, uma vez que a alteração dos conteúdos do *site* é mais rápida;
- menor custo por página;
- reduz os custos de manutenção do *website*.

A figura seguinte confronta o processo de actualização de um *website* com CMS e sem CMS.

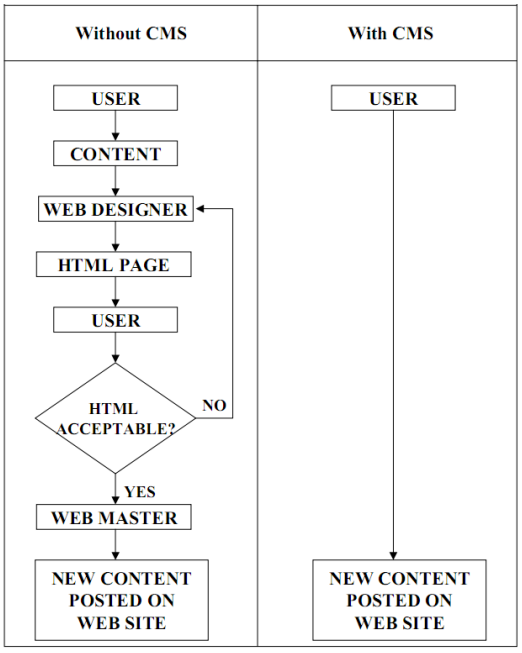


Figura 1 - Actualização de um *website* com e sem CMS (Zuckert, 2003)

Os CMS são sistemas utilizados por diversos grupos/tipos de utilizadores com diferentes responsabilidades e privilégios. Existem utilizadores responsáveis por criar, editar e gerir conteúdos e utilizadores visitantes com acesso limitado. Estes sistemas proporcionam a fácil gestão de um *website*, uma vez que tornam simples as tarefas de criação, gestão e consulta de conteúdos. Os Administradores destas plataformas têm a possibilidade de gerir todo o conteúdo do *site* e toda a informação relativa aos seus utilizadores, através de interfaces fáceis de utilizar e que se executam através de um qualquer *browser*. A plataforma recolhe os conteúdos de diversos utilizadores, faz a gestão do *workflow* desses conteúdos e operacionaliza a administração de todo o *site* (Harwani 2009, p.2-3).

A popularidade dos sistemas CMS reside no facto de haver a separação entre o conteúdo e a sua apresentação. Como consequência disso, o responsável pelo desenvolvimento do conteúdo pode concentrar-se nessa tarefa, enquanto os *Web designers* podem focar-se em dar uma aparência dinâmica a esse conteúdo, aplicando ou desenvolvendo diferentes *templates*, sem interferirem entre eles. Assim, o processo de desenvolvimento de conteúdos e do desenvolvimento do modo como são apresentados pode ser feito em simultâneo (Harwani 2009, p.3).

Tipicamente, um CMS apresenta as seguintes características (Shereves 2010, p.5):

- identificação dos principais utilizadores e das suas regras;
- capacidade de definir regras e responsabilidades;
- capacidade de definir o *workflow*;
- capacidade de agendar a publicação de conteúdos;
- capacidade de limitar o acesso a conteúdos e funcionalidades;
- capacidade de administração do sistema;
- capacidade de colocar o *site* em modo *offline* para tarefas de manutenção;
- capacidade de adicionar novos componentes.

2.2.2. Arquitectura de um CMS

Os sistemas dedicados à gestão de conteúdos são constituídos por diversos módulos. O seu desenvolvimento é moroso e requer pessoal especializado. Um típico CMS é constituído pelos seguintes módulos:

- sistema de autenticação;
- módulos para criação de contas;
- recuperação de palavras-chave;
- conteúdo popular;
- sistema de votações;
- *feeds* RSS;
- motor de pesquisa;
- gestor de temas;
- suporte a múltiplos idiomas;
- gestão de permissões (Harwani 2009, p.2).

A criação e gestão de um *website* através da utilização de um CMS torna-se numa tarefa simples, uma vez que os seus responsáveis restringem-se à configuração de módulos, gestão de conteúdos e gestão de utilizadores, tudo isto através de interfaces de fácil utilização e sem a necessidade de recorrer a linguagens de programação.

Uma aplicação CMS encontra-se alojada de forma segura num servidor *Web* e a sua arquitectura pode variar de produto para produto. Em determinados produtos podemos ter *software* localizado dentro da rede local da organização, para recolha e teste de conteúdos e *software* num servidor fora da rede local, para a publicação dos conteúdos na Internet (Boiko 2005, p.77).

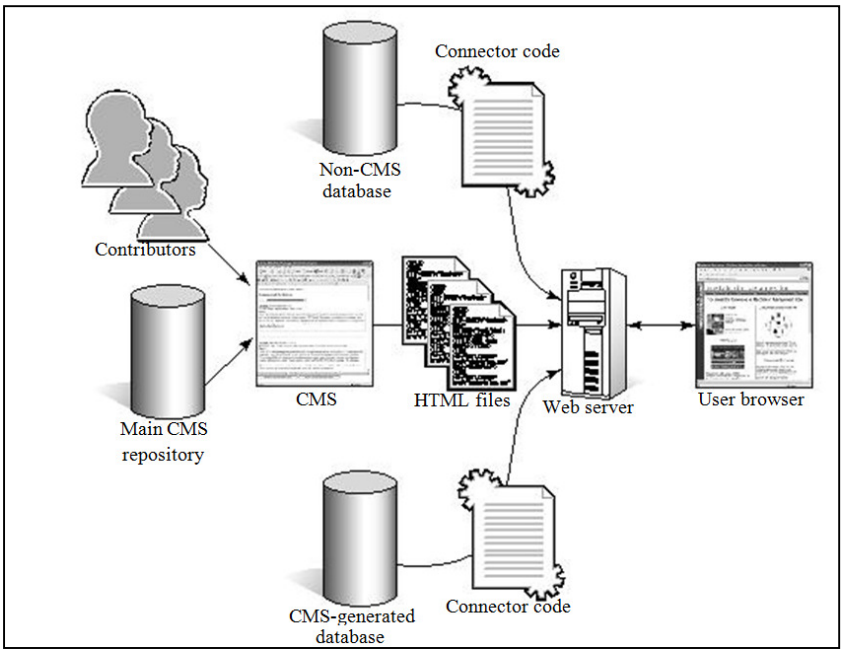


Figura 2 - Um CMS por detrás de todas as partes estáticas e dinâmicas do *site*. (Boiko 2005, p.78)

Por trás do servidor *Web* responsável por alojar o CMS, existe uma base de dados relacional ou um conjunto de ficheiros XML (*Extensible Markup Language*) que constituem o repositório de conteúdos. Neste repositório são também armazenados os dados relativos à administração do sistema e os recursos necessários à construção do *site*, como por exemplo, elementos gráficos e folhas de estilo. Num *software* CMS podem existir vários repositórios, mas o sistema tem a sua própria base de dados que é responsável pelo seu funcionamento e pela gestão de conteúdos dinâmicos. Os CMS podem conter também páginas HTML simples que correspondem às partes estáticas do sistema (Boiko 2005, p.78). Para tornar as interfaces amigáveis e adequadas ao tipo de CMS, estes sistemas utilizam *templates* (temas) que podem ser facilmente geridos pelos gestores do *site* e ajustáveis às existências da organização.

2.3. CMS Joomla

2.3.1. A plataforma Joomla

O CMS Joomla nasceu a partir da plataforma Mambo, criada pela empresa australiana Miro que disponibilizou o código fonte do sistema, segundo a licença GNU *General Public License* (GPL). A plataforma Mambo ganhou, num curto período de tempo, uma grande popularidade. Em 2005, problemas relacionados com a disputa dos direitos de autor, entre a equipa de desenvolvimento e a Mambo Steering Committee, contribuíram para que a maioria dos seus membros abandonassem o projecto. Esta separação resultou na criação de uma nova entidade, a Open Source Matters, que desenvolveu uma nova versão do Mambo denominada de Joomla. A primeira versão desta plataforma (V 1.0) foi lançada em Setembro de 2005 e, em Janeiro de 2008, foi lançada a sua maior revisão, a versão 1.5, que com a sua nova API permitiu a internacionalização da plataforma, uma vez que tem suporte a diferentes tipos de caracteres e idiomas (Severdia R, *et al*, 2010, p.1-2). Ao realizarmos uma pesquisa na Internet acerca dos "*top CMS*", facilmente verificamos que o Joomla é uma das plataformas mais utilizadas a nível mundial. Como plataforma CMS, o Joomla tira partido das vantagens destes sistemas, permitindo que os seus utilizadores se preocupem apenas com a criação de texto e introdução de alguns elementos gráficos. A plataforma é capaz de gerir todo o *site*, organizando os conteúdos, tornando-os pesquisáveis e apresentando-os correctamente.

2.3.2. Características do Joomla

O Joomla apresenta-se como um CMS *open source* poderoso, extensível e que contém as funcionalidades, apresentadas no *site* oficial (Joomla! Features Overview, 2010), que se passam a descrever.

Gestor de utilizadores - possui um sistema de registo que permite aos utilizadores a configuração de definições pessoais. Contém vários grupos de utilizadores com diferentes tipos de permissões relativamente ao acesso, edição, publicação e administração. Suporta múltiplos protocolos de autenticação, tais como LDAP[1] (*Lightweight Directory Access Protocol*) e OpenID[2].

[1] Directório de utilizadores de acesso para autenticação.

[2] Sistema de identificação de utilizadores na *Web* que permite, com apenas um *username* e *password*, a autenticação em vários *websites*.

Media manager **-** ferramenta que permite a gestão de ficheiros do tipo MIME (*Multipurpose Internet Mail Extensions*). Esta ferramenta está integrada com o editor de artigos, para que o utilizador, a qualquer momento, possa guardar imagens e outro tipo de ficheiros. Esta ferramenta permite ainda definir o tipo de extensões permitidas.

Gestor de *banners* - permite a gestão fácil de *banners*, através da criação de perfis de clientes. Tem funcionalidades como a definição do número de apresentações, *links* específicos, etc.

Gestor de contactos - facilita aos utilizadores o acesso a informações de contacto de outros utilizadores. Permite contactar com indivíduos específicos ou com grupos de indivíduos.

Sistema de inquéritos - inclui um sistema que permite a criação de inquéritos e apresentação de resultados obtidos.

Pesquisa - ajuda os utilizadores a navegar pelos itens mais populares e permite aos administradores o acesso a estatísticas de procura.

Gestor de *links* - permite a organização de *links*, através da sua classificação em categorias, e o registo do número de acessos (cliques).

Gestor de conteúdos - o Joomla facilita a gestão dos conteúdos de acordo com as preferências dos utilizadores, permite a classificação de artigos, o envio por *e-mail* e a gravação para PDF (*Portable Document Format*) com suporte às diferentes codificações de caracteres. Os administradores podem arquivar e ocultar conteúdos aos visitantes do *site*. Em *sites* públicos, a plataforma tem a capacidade de proteger os endereços de *e-mail* de *spambots*[3]. A utilização de editores WYSIWYG (*"What You See Is What You Get"*) possibilita aos utilizadores menos experientes criar facilmente artigos combinando texto e imagens de forma atractiva. Após a criação dos artigos, existem módulos pré-instalados que permitem mostrar os artigos mais populares, os últimos artigos inseridos, artigos relacionados, resumos de notícias, etc.

Gestor de *feeds* RSS (*Really Simple Syndication*) - o Joomla permite a subscrição de sinais RSS pelos seus utilizadores e agregação com diferentes fontes.

Gestor de Menus - através de um gestor de menus, é possível criar múltiplos menus com os diversos itens necessários à navegação no *site*. Estes menus podem ser posicionados em diferentes locais e podem assumir vários formatos. Interfaces de navegação são criadas automaticamente para ajudar os utilizadores na navegação através do *site*.

Gestor de Temas - para criar a aparência desejada do *site*, a plataforma contém um gestor que possibilita a integração e configuração de temas. Através deste gestor é possível a personalização do *site* ou apenas de partes específicas das páginas.

Sistema de ajuda integrado - o Joomla possui um sistema de ajuda por secções, um glossário com a explicação dos termos encontrados pela plataforma, um verificador de versões e painéis com informação que possibilitam ajudar os utilizadores a resolver determinados problemas.

Definições do sistema - os administradores podem aumentar o desempenho do *site* através de configurações de cache e compressão, fazer o *debug* ao *site* e a verificação de erros. Permite o acesso por FTP sem a necessidade de atribuir permissões de escrita a todos os

[3] Programa ou *script* que assiste o envio automático de mensagens *spam*.

ficheiros e pastas do *site*. Os administradores podem também comunicar por mensagens privadas com outros utilizadores ou através de *e-mails* massivos.

Serviços *Web* – permitem utilizar serviços RPC (*Remote Procedure Call*) e integração com a API e Blogger do Joomla.

Instalação de Componentes e extensões – permite a personalização, através da integração de novos componentes ou extensões de modo a satisfazer as necessidades.

2.3.3. Razões para a escolha da plataforma Joomla

O Joomla é actualmente um sistema gestor de conteúdos de eleição, não apenas por ser gratuito ou por já existir há algum tempo, uma vez que existem inúmeras aplicações CMS também nessas condições. A principal razão que leva à escolha do Joomla é a reputação e fidelidade que ganhou perante os utilizadores que foi conquistando. Existem pessoas dedicadas à plataforma Joomla que se encontram espalhadas por todo mundo e que conseguiram criar uma comunidade muito forte. Esta comunidade tem criado milhares de templates, componentes, módulos, e *plug-gins,* estendendo assim as capacidades desta plataforma. Esta comunidade disponibiliza ainda ajuda para os utilizadores menos experientes.

Em termos de utilização, a criação de sites profissionais recorrendo à plataforma Joomla torna-se tão simples como o processamento de documentos para impressão. Basta o utilizador dominar as tarefas mais básicas para começar a utilizar normalmente a plataforma. Comparativamente com outros sistemas CMS, os conhecimentos técnicos exigidos são mínimos.

Para além das vantagens acima referidas, a plataforma Joomla apresenta interfaces e painéis de gestão intuitivos, edição WYSIWYG, capacidades avançadas de formatação, milhares de *templates* para *download*, pesquisa de palavras em todo conteúdo dos artigos (*full text search*), *plug-ins* para sites comerciais (*shopping carts*), motor de pesquisa avançado e agendamento de publicações (Holzner, 2009).

Actualmente, diversos tipos de organizações utilizam o Joomla e as escolas não são excepção. Para muitas instituições escolares, a plataforma Joomla já é utilizada na criação e gestão do seu *site* na Internet (Bárcia, 2008). Já existem neste tipo de instituições pessoas capazes de utilizar e tirar proveito das potencialidades desta plataforma. Assim, e por todas as vantagens da utilização do Joomla, o componente a desenvolver no âmbito desta tese estará preparado para ser integrado nesta plataforma. A aplicação criada não será mais uma plataforma com que os utilizadores tenham que se familiarizar, mas sim uma extensão do Joomla. Esta extensão terá como requisito fazer face às necessidades básicas de um EDMS, direccionado à gestão de documentos criados pelos professores de uma escola básica ou secundária e que são fruto do seu trabalho realizado ao longo do ano lectivo.

2.3.4. Estrutura da plataforma Joomla

A plataforma Joomla ou qualquer outro CMS em ambiente *Web* apresenta dois lados distintos, o *front end* e o *back end*. O *front end* corresponde ao lado do *site* visível aos visitantes ou utilizadores com sessão iniciada. O *back end* corresponde à camada de administração do *site*, onde é possível realizar tarefas de configuração, manutenção, limpeza, criação de estatísticas e

criação de conteúdos. Esta camada é apenas acessível pelos administradores do *site*. O acesso à camada de administração é realizado através de um endereço diferente do endereço principal do *site* (Graf 2008, p.13).

Os utilizadores de um CMS têm um identificador único *(username)* e são associados a grupos de utilizadores com determinados privilégios. Podem ser utilizadores apenas registados, autores e editores que acedem ao *site* do lado do *front end* ou administradores com controlo total da plataforma e acesso ao *back end*. O CMS tem a capacidade de apresentar os conteúdos e as opções às quais os utilizadores têm acesso com base nos seus privilégios (Graf 2008, p.13).

Relativamente ao conteúdo do site, este pode surgir de diferente formas, desde a apresentação de texto simples, a imagens, ligações para outras páginas, música, aplicações embebidas como o Google Maps ou a combinação de todos estes elementos. Os conteúdos são associados a categorias criadas e geridas pelos administradores. A utilização de *feeds* deste tipo de plataforma também é muito comum e pode ser associada a fontes exteriores ao *site* (Graf 2008, p.14).

O Joomla permite a integração de diversas extensões sob a forma de componentes, módulos, *templates* ou *plug-ins*. Os componentes são extensões que adicionam novas funcionalidades e que contêm uma área própria no lado de administração do Joomla, como por exemplo, loja *online*, galeria de fotografias, *newsletter* e fóruns. Possuem a lógica de negócio do *site* e utilizam a plataforma para apresentar o seu conteúdo.

Os *templates* são utilizados para definir o aspecto visual dos elementos que compõem as páginas do *site*, ou seja, define as cores utilizadas, tipos de letra, tamanhos de letra, imagens de fundo, espaçamentos, o modo de divisão das páginas, etc. Um *template* é constituído pelo menos por um ficheiro HTML, que define a estrutura das páginas e um ficheiro CSS para o design. No entanto, o *template* pode apresentar uma extensa estrutura de ficheiros com o intuito de preparar a plataforma Joomla para diferentes fins (Graf 2008, p.14).

As extensões do tipo *plug-in* são secções de código que é acrescentado em locais específicos da *framework*, de modo a alterar as suas funcionalidades. Por exemplo, pode ser utilizado no meio do texto de um artigo para carregar conteúdos de um determinado módulo (Graf 2008, p.15).

Para o suporte aos múltiplos idiomas que podem ser utilizados no CMS Joomla, existem ficheiros incorporados na sua estrutura que são responsáveis pela tradução da informação referente ao funcionamento e utilização do sistema, como por exemplo: tradução das mensagens de erro, da informação apresentada nos painéis de configuração, dos menus de administração, etc.

O CMS Joomla é desenvolvido sobre a *Joomla! Framework* API que contém todas a bibliotecas utilizadas no desenvolvimento da plataforma. Esta *framework* é também utilizada por programadores para o desenvolvimento de extensões para o Joomla ou para estabelecer a ligação ou integração com outras aplicações.

Para o desenvolvimento do componente SGDEscolas será utilizada a *Joomla! Framework*, de modo a garantir a sua integração com o CMS e usufruir das bibliotecas disponíveis e necessárias ao desenvolvimento da extensão.

2.3.5. Joomla! Framework

A *Joomla! Framework* consiste na estrutura de suporte ao desenvolvimento do CMS Joomla. Esta estrutura inclui bibliotecas de código utilizadas para o desenvolvimento de extensões para o Joomla e também pode ser utilizada no desenvolvimento de novas aplicações.

O sistema Joomla é composto por três camadas: camada extensão (*extension layer*), camada aplicação (*application layer*) e a camada *framework* (*framework layer*) (Joomla! *Framework*, 2010). A figura seguinte ilustra as três camadas.

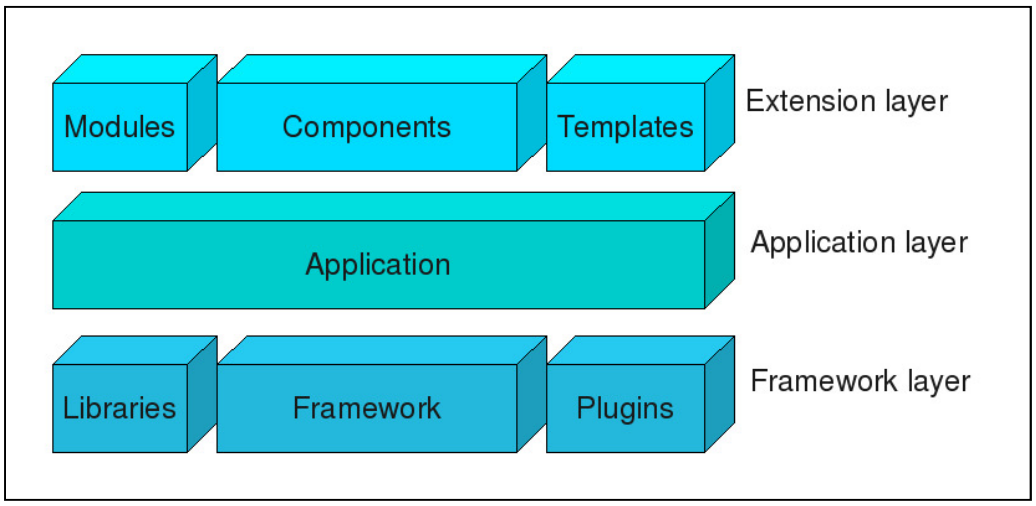


Figura 3 - As três camadas do sistema Joomla. (Joomla! *Framework*, 2010).

No topo deste sistema encontra-se a camada *extension layer* que aloja as extensões da plataforma Joomla, ou seja, módulos, componentes e *templates*.

A camada intermédia, *application layer*, contém aplicações, extensão da classe `JApplication` incluída *framework*. A versão 1.5 do CMS Joomla inclui as seguintes aplicações: JInstallation, responsável pela instalação e desinstalação do Joomla no servidor *Web*; JAdministrator, responsável pelo lado de administração do *site*; JSite, responsável pelo *front-end* do *website*; XML-RPC responsável pelo suporte à administração remota do *website* Joomla.

A camada inferior, *framework layer*, consiste na própria *framework* do Joomla com as suas respectivas classes, bibliotecas necessárias à *framework*, bibliotecas instaladas por outros programadores e *plugins* que adicionam funcionalidades à *framework* (Joomla! Official Documentation, 2010).

As classes da *framework* do Joomla encontram-se agrupadas em diversos pacotes. A seguinte figura mostra os principais pacotes que compõem a *framework*.

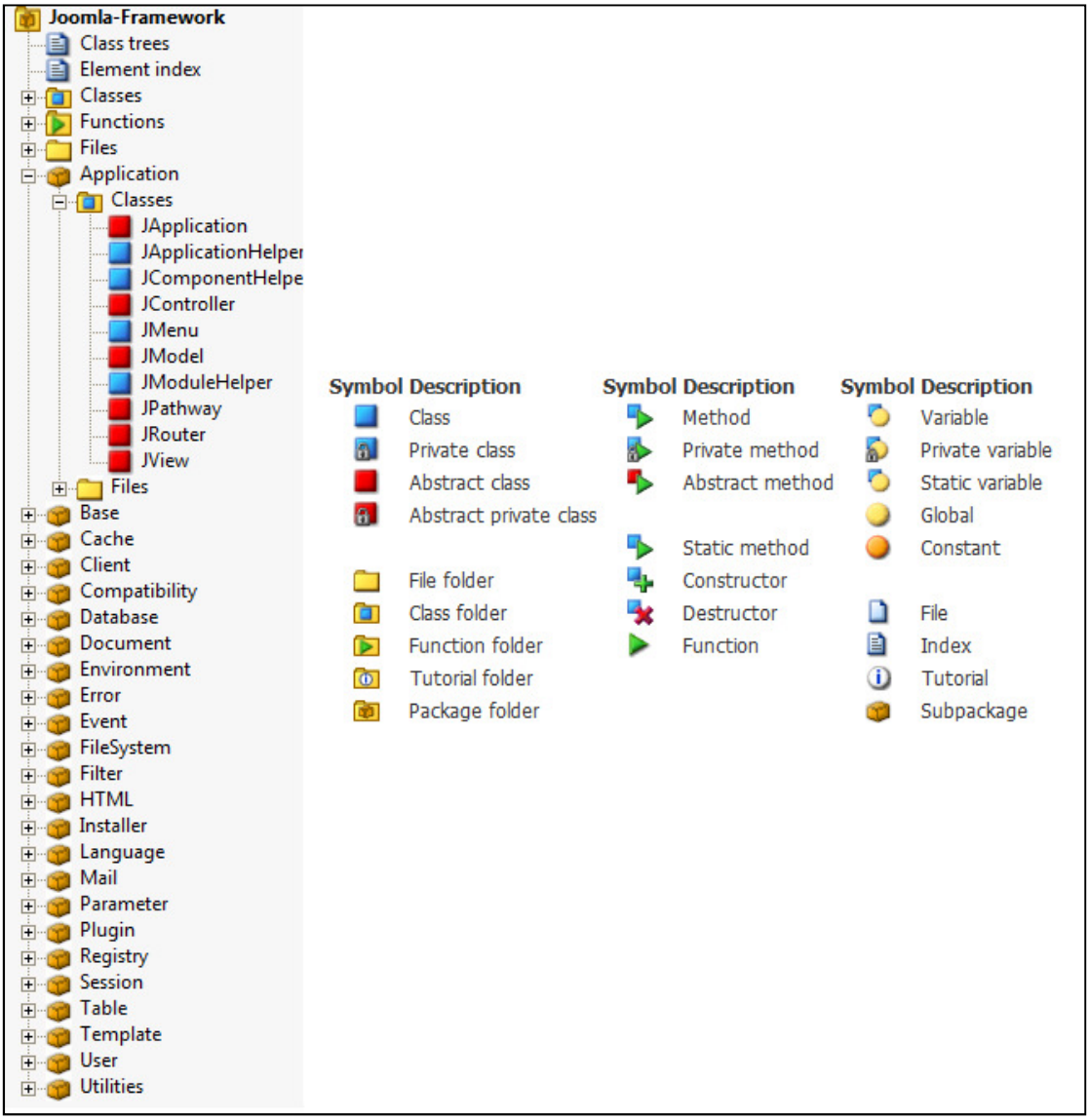


Figura 4 - Framework Joomla. (Joomla! 1.5 API Reference 2010).

Como podemos observar na figura, a *framework* apresenta uma grande variedade de subpacotes que contêm as classes utilizadas no desenvolvimento CMS Joomla.

2.3.6. Funcionamento do Joomla

O Joomla é uma plataforma desenvolvida em linguagem de *scripting* PHP orientada a objectos. Para o correcto desenvolvimento de extensões para este CMS torna-se necessário compreender o seu funcionamento. Assim, passam-se a descrever os aspectos fundamentais do funcionamento desta aplicação e do processo envolvido, desde que é efectuado um pedido no *browser* até à obtenção da sua resposta.

2.3.6.1. Processo pedido/resposta (*request to response*)

O processo pedido/resposta inicia-se através de um dos pontos de entrada da plataforma Joomla que, por razões de segurança, são apenas dois: um para o *front end,* através do ficheiro de `index.php` que se encontra na raiz da estrutura de directórios da plataforma; outro para o

back end, através do ficheiro `administrator/index.php`. No desenvolvimento de extensões deve ser assegurado que não serão adicionados novos pontos de entrada.

Na programação de *scripts* em PHP são normalmente utilizadas as variáveis pré-definidas `$_GET`, `$_POST`, `$_FILES`, `$_COOKIE` e `$_REQUEST` para a passagem de uma página para outra e para trabalhar com sessões. Na plataforma Joomla, estas variáveis não são acedidas directamente, mas sim através da classe estática `JRequest`. A utilização desta classe permite que seja possível processar entradas de dados no mesmo instante em que são devolvidos, reduzir a quantidade de código necessário e contribuir para a melhoria da segurança.

Os dois métodos mais utilizados da classe `JRequest` são: `JRequest::getVar()` para aceder aos dados contidos nas variáveis e `JRequest::setVar()` para definir o valor das variáveis. O código abaixo apresenta um exemplo de aplicação destes dois métodos.

```
JRequest::setVar('layout', 'gotodoss');
$cids = JRequest::getVar( 'cid', array(), 'post', 'array' );
```

Quando é feito um pedido (*request*) através do *frontend* do Joomla, é desencadeado o processamento desse pedido que passa pelas seguintes fases:

- carregamento dos ficheiros da *framework* (*Load Core*);
- criação da aplicação (*Build Application*);
- inicialização da aplicação (*Initialize Application*);
- encaminhamento da aplicação (*Route Application*);
- despacho da aplicação (*Dispatch Application*);
- transformação da aplicação (*Render Application*);
- envio da resposta (*Send Response*).

Este processo é apresentado nos diagramas seguintes. O processo desencadeado através do *back end* é bastante similar (Lanham 2010, p.37).

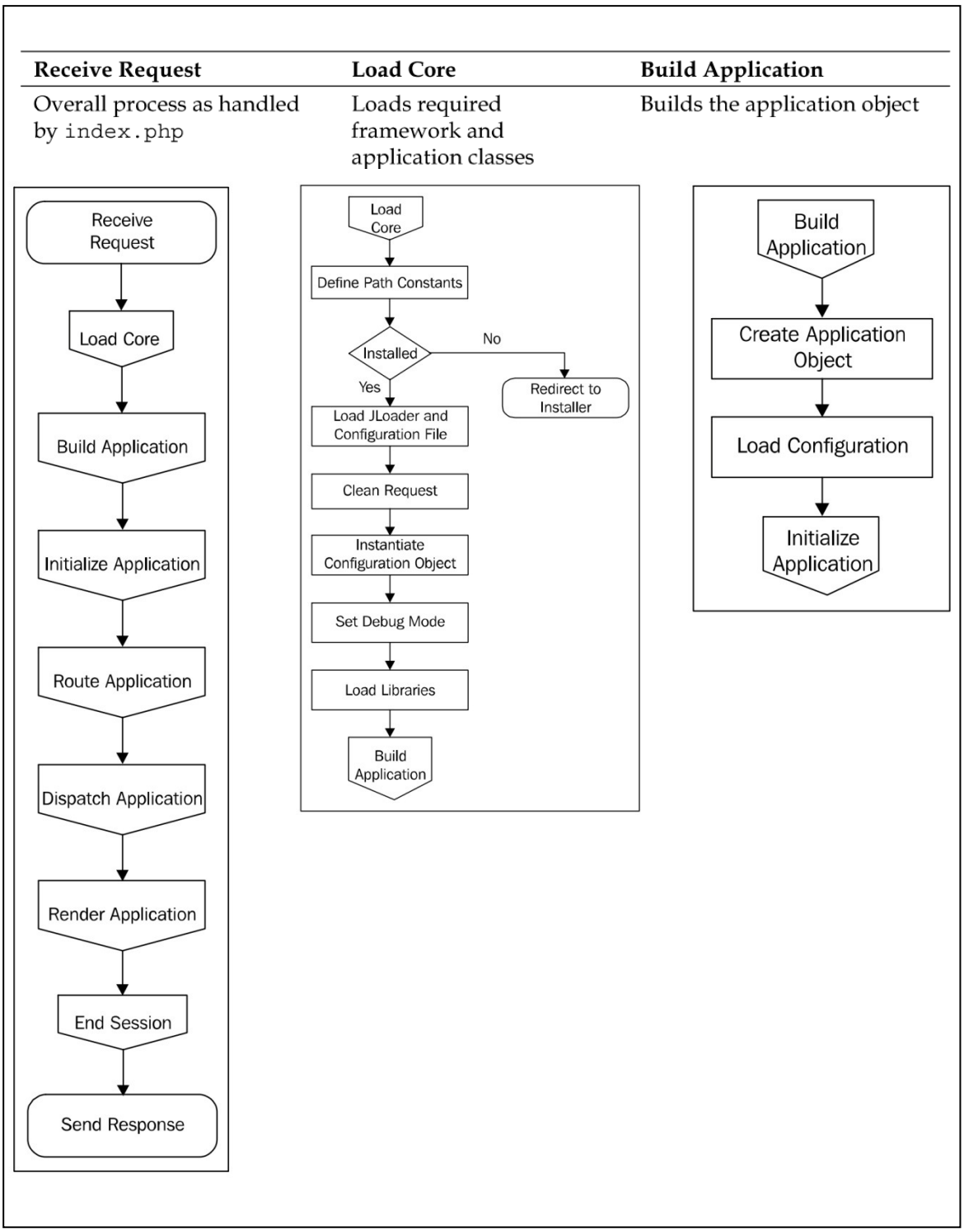


Figura 5 - Descrição geral do processo, carregamento da *framework* e criação da aplicação (Lanham 2010, p.34).

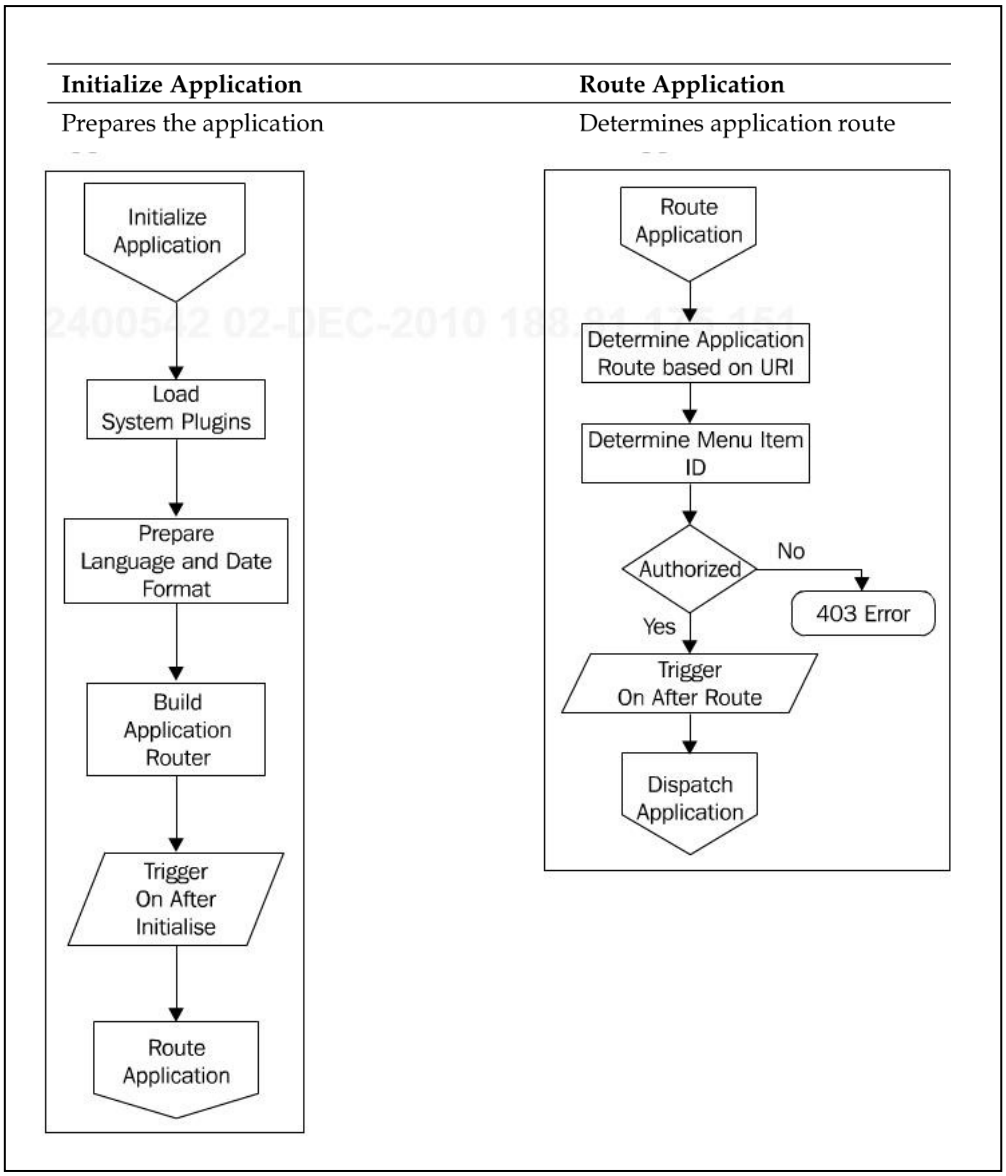


Figura 6 - Inicialização e encaminhamento da aplicação (Lanham 2010, p.35).

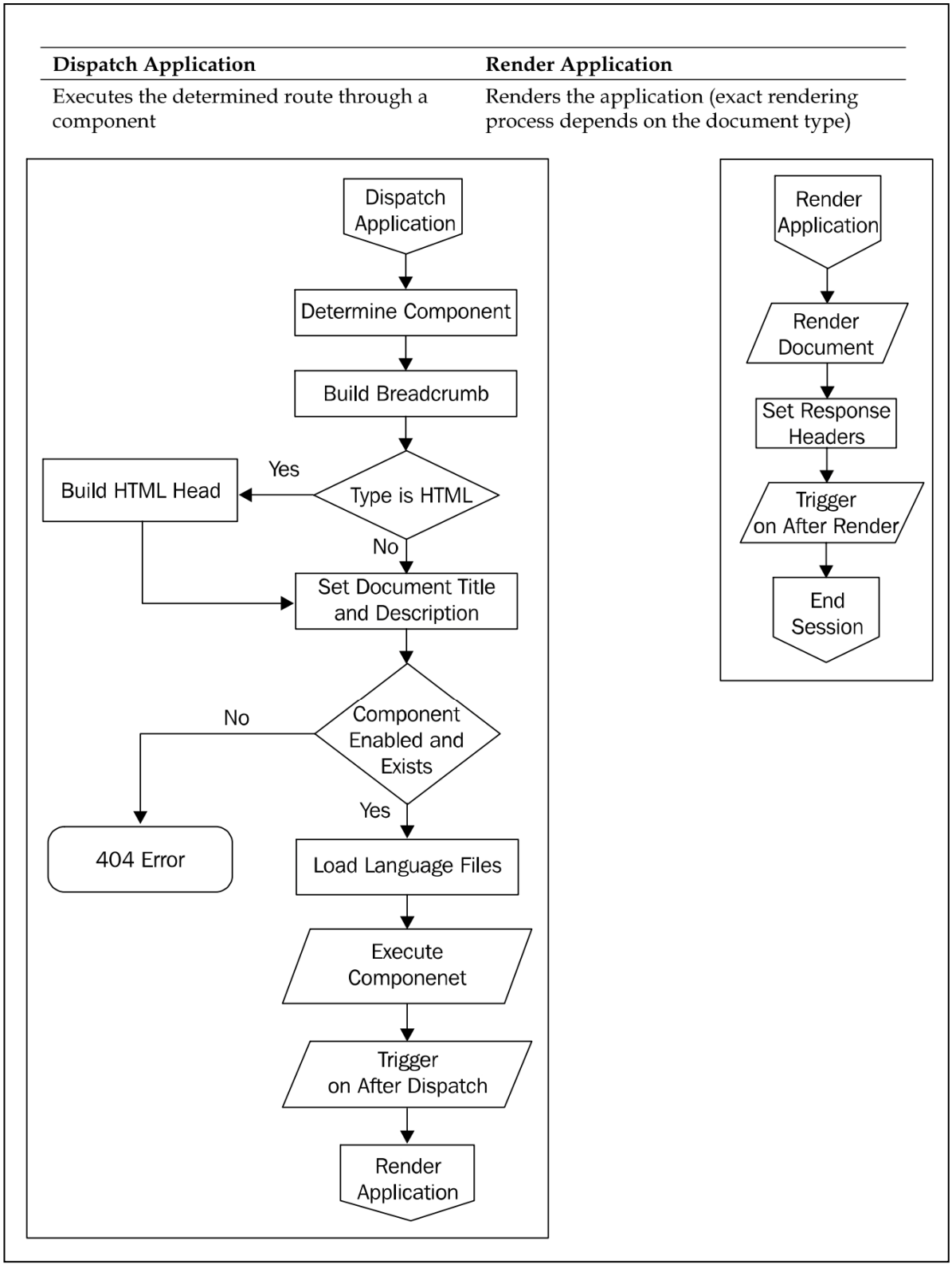


Figura 7- Despacho da aplicação (Lanham 2010, p.36).

O processamento do pedido começa com o carregamento dos ficheiros da *framework* Joomla (*Load Core*). Nesta fase são efectuados os seguintes passos:

1. Leitura do ficheiro `defines.php`, responsável por definir as constantes globais da *framework*.
2. Carregamento do ficheiro `framework.php`, caso ainda não tenha sido carregado;
3. Modificação das opções de configuração relativamente às *magic quotes* e compatibilidade com Zend.
4. Verificação da existência do ficheiro `configuration.php` ou da existência da aplicação responsável pela instalação da plataforma. Se a aplicação de instalação estiver presente, o processo é redireccionado para esta aplicação (`JInstallation`). Caso o ficheiro `configuration.php` e a aplicação de instalação não sejam encontrados significa que aplicação já existe.
5. Carregamento do ficheiro `import.php`, caso ainda não tenha sido carregado. Este ficheiro carrega a classe estática `JLoader` que importa as bibliotecas da *framework* que contêm as classes `JObject`, `JRequest`, `JResponse`, `JFactory`, `JVersion`, `JError`, `JExeption`, `JArrayHelper`, `JFilterInput`, `JFilterOutput`, `JText` e `JRoute`.
6. É feita a "limpeza" ao pedido, de modo a remover dados inesperados e a garantir que o tipo dos dados que contém é o previsto.
7. Carregamento do ficheiro `configuration.php`, caso ainda não tenha sido carregado.
8. É instanciado o objecto `JConfig`.
9. É definida a criação de relatórios de erro e opções `JDEBUG`.
10. São carregadas as funções e classes de compatibilidade do PHP.
11. Inicialização do *profiler* caso `JDEBUG` esteja definido.
12. São importadas as bibliotecas: `JMenu`, `JUser`, `JURI`,`JHTML`, `JUtility`, `JEvent`, `JDispatcher`, `JLanguage` e `JString` (Lanham 2010, p.37-38).

O segundo passo do processo consiste na criação da aplicação (*build application*) da *framework*. A aplicação consiste num objecto global utilizado para processar o pedido. As classes da aplicação são extensão da classe abstracta base `JApplication`. A classe responsável pelo *frontend* é a `JSite` e a classe responsável pelo *backend* é a `JAdministrator`. O objecto da aplicação é sempre guardado na variável global `$mainframe` e pode ser utilizado dentro de qualquer função ou método através da sua declaração (Lanham 2010, p.39). O seguinte código mostra um exemplo da sua utilização.

```
function _verificaUtilizadorSgdescolas()
{
        global $mainframe;
…
…
        $message = JText::_('ACCESS_DENY');
        $type = 'error';
        $mainframe->Redirect('index.php?', $message, $type);
…
…
```

O processo de criação da aplicação envolve os seguintes passos:

1. Definição da variável global `$mainframe` chamando `JFactory::getApplication('site')`, criando assim uma instância do objecto `JSite`.
2. Carregamento da configuração padrão e os dados da sessão.
3. Criação do objecto de configuração.
4. Criação de uma sessão, se requerida.
5. Definição da data e hora (*timestamp*) do pedido (Lanham 2010, p.39).

O terceiro passo do processo consiste na inicialização da aplicação (*initialize application*) e envolve os seguintes passos:

1. Definição do idioma a utilizar no *frontend*.
2. Chamada a `parent::initialise` (classe `JApplication`).
    - Chamada a `JFactory::getUser` para inicialização do objecto referente ao utilizador.
    - Chamada a `JFactory::getSession` para a criação da sessão.
    - Definição do editor especificado pelo utilizador ou editor padrão, caso não tenha sido especificado.
3. Importação do sistema de *plugins*.
4. Accionar o evento `onAfterInitialise` (Lanham 2010, p.40).

O quarto passo do processo corresponde ao encaminhamento da aplicação (*route application*). Aqui o URI (*Uniform Resource Indicator*) é analisado para determinar o componente a utilizar pelo pedido. Sempre que é enviado um pedido é gerado um URI que tem a estrutura representada na figura seguinte.

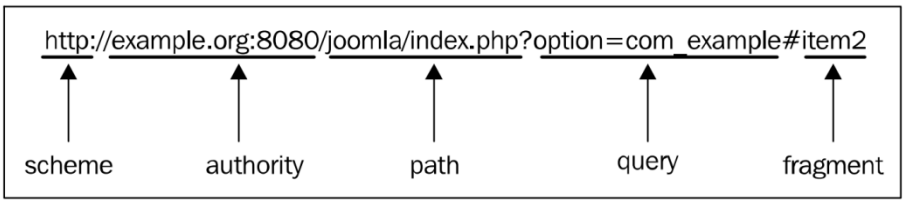


Figura 8 - Estrutura do URI (Lanham 2010, p.43).

O elemento "query" corresponde à parte do URI que devolve os dados. É composto por uma série de pares de valores chave separados por "&". A primeira chave "query" do URI que vemos no exemplo da figura anterior é a `option`. Este valor chave determina o componente requisitado, que neste caso corresponde ao componente `com_example`. Os nomes dos componentes no Joomla têm sempre o prefixo `com_`.

O encaminhamento da aplicação envolve os seguintes passos (Lanham 2010, p.45):

1. É devolvido do URI completo (`JURI::getInstance`).
2. Análise do URI e identificação da rota a seguir pela aplicação.
3. É determinado o identificador do menu (`JSite::getMenu`) e é feita a verificação do nível de acesso.

- Redirecciona o utilizador para fazer *login* se ainda não estiver autenticado.
- Gera erro e pára no caso de o utilizador já ter feito *login* mas não tem permissão de acesso.

4. É accionado o evento `OnAfterRoute`.

O quinto passo do processo pedido/resposta refere-se ao despacho da aplicação (*Dispatch Application*). É neste momento que se inicia a criação do documento que será apresentado ao utilizador. O documento consiste num objecto global que contém a resposta ao pedido inicial e pode ser do tipo HTML, PDF, RAW, *feed* e de erro. O processo de despacho envolve os seguintes passos:

1. É extraído o valor de `option` que se refere ao componente contido no pedido (`JRequest::getCmd('option')`).
2. É chamado o método `JSite::dispatch`.
   - É criado o objecto documento (`JDocument`) através do método `JFactory::getDocument`.
   - É devolvido o utilizador actual através do método `JFactory::getUser`.
   - É determinado o caminho do documento.
   - São devolvidos os parâmetros do documento.
   - No caso de documentos `html` são definidos os meta-dados.
   - É definido o URI base.
   - É definido o título e a descrição do documento.
   - É localizado e executado o componente, caso exista e esteja activo, caso contrário é devolvido um erro.
3. É accionado o evento `onAfterDispactch`.

A última fase do processamento do pedido corresponde ao processo de transformação do documento (*render application*), em que o formato final é determinado pelo tipo de documento a apresentar. Esta fase envolve os seguintes passos (Lanham 2010, p.47):

1. O documento é devolvido.
2. Se o documento for HTML é devolvido o *template* e são definidos os parâmetros.
3. São definidos os cabeçalhos do documento.
4. É enviado o corpo do documento.
5. É accionado o evento `onAfterRender`.

Após concluído o processo de processamento do pedido é enviada a resposta para o *browser*.

2.3.6.2. Estrutura de directórios

Para o sucesso no desenvolvimento de extensões para a plataforma Joomla é importante conhecer, não só o processo envolvido no processamento dos pedidos efectuados através do *browser*, mas também toda a estrutura de directórios da plataforma e respectiva *framework*. O pacote de instalação do CMS Joomla 1.5 contém os seguintes directórios:

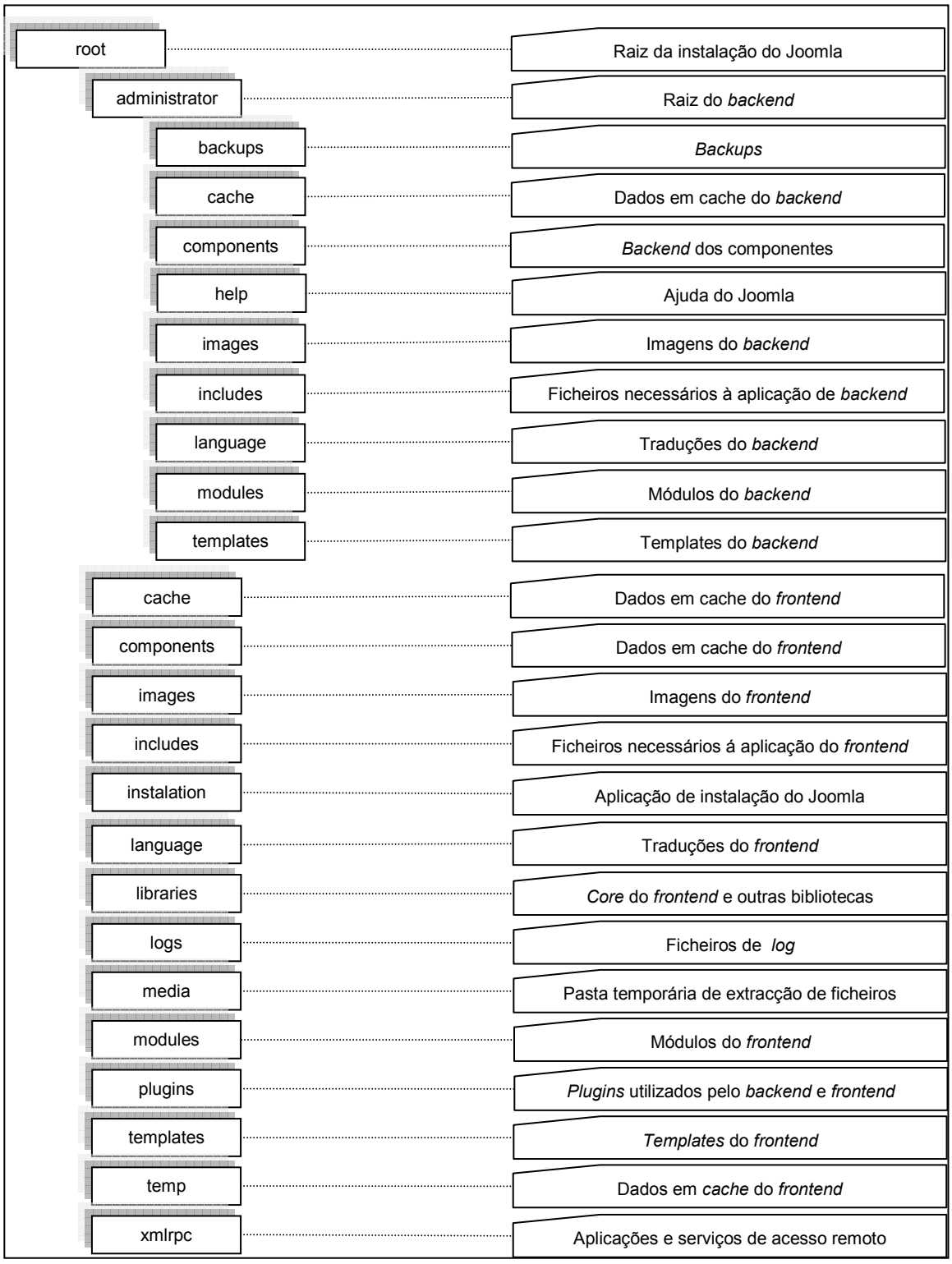


Figura 9- Estrutura de pasta da plataforma Joomla (Lanham 2010, p.48-49).

Ao analisar a estrutura de directórios apresentada na figura anterior, pode-se verificar que a pasta `administrator` contém as bibliotecas, componentes, módulos e *templates* necessários ao funcionamento do *back end* da plataforma Joomla. A maior parte dos directórios que se situam na raiz da instalação do Joomla são dedicados ao *front end*, no entanto, alguns *plugins* e bibliotecas situadas na raiz da instalação podem também ser utilizados pelo *back end*.

O componente criado no âmbito desta tese terá também um *back end* para a administração e um *frontend* para o utilizador comum. Assim, durante o processo de instalação do componente, serão criados os directórios `…/root/administrator/components/com_sgdescolas` para o *back end* e `…/root/components/com_sgdescolas` para o *frontend*.

2.3.7. Instalação do Joomla

O Joomla é uma plataforma pouco exigente ao nível de conhecimentos técnicos para a sua instalação e configuração. Pode ser instalado num servidor local ou num servidor remoto. A escolha de uma instalação local tem como principais vantagens: a liberdade na configuração, o conhecimento completo do sistema, o desempenho e a flexibilidade ao nível de *backups*. A instalação local é, no entanto, mais trabalhosa ao nível da configuração e manutenção. Em termos de desempenho, também não é garantido que se tenha melhores resultados na utilização de um servidor local do que na utilização de um servidor alugado a uma empresa prestadora de serviços *host*. Estas empresas conseguem obter óptimos resultados ao nível da segurança e do funcionamento dos serviços, em caso de ocorrência de falhas. Num servidor local torna-se difícil e dispendioso obter resultados semelhantes (Rahmel 2007, p.10).

Para o desenvolvimento e testes do Joomla é normalmente utilizada uma instalação local, podendo-se instalar e configurar os servidores individualmente ou através de pacotes já disponíveis, como por exemplo o XAMPP (XAMPP 2010) e o Wampserver (Wampserver 2010). O XAMPP está disponível para plataformas Linux, Windows, Mac OS e Solaris. O Wampserver está disponível apenas para plataformas Windows.

2.3.8. Requisitos e opções de configuração

A instalação e configuração do CMS Joomla requer a interacção entre um servidor Web, um motor de execução PHP, o servidor de bases de dados MySQL e a própria plataforma Joomla. O servidor Web mais indicado para a instalação do Joomla é o Apache, no entanto, pode também ser instalado num servidor IIS da Microsoft. É de referir que o servidor IIS apenas está disponível para a plataforma Windows, enquanto o servidor Apache está disponível para Windows, MacOS, e UNIX/Linux.

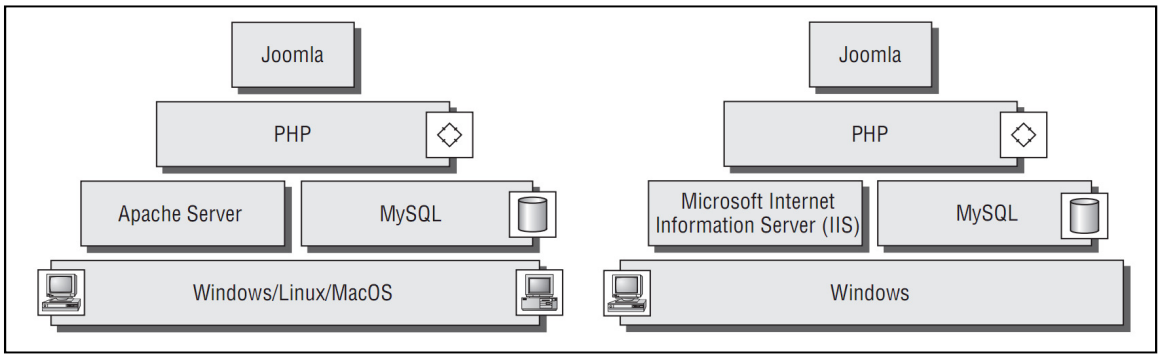


Figura 10- As duas opções a nível de servidores para instalação do Joomla (Rahmel 2007, p.10).

Na utilização de um servidor Apache configurado PHP e MySQL, quando é efectuado um pedido Web são realizados os seguintes passos:

1. O cliente envia um pedido ao servidor Apache em termos mensagens http, GET ou POST.
2. O Servidor Apache analisa o pedido, localiza o script PHP e executa-o.
3. Dependendo do pedido do utilizador, o *script* PHP obtém ou actualiza a informação da base de dados no servidor MySQL.
4. A base de dados MySQL devolve a informação e o estado da base de dados para o *script* PHP.
5. O *script* PHP combina a informação da base de dados com um *template* HTML e envia-os para o servidor Apache.
6. O Apache envia uma resposta HTTP em forma de documento HTML para o *browser* do cliente (Harwani 2009, p.13).

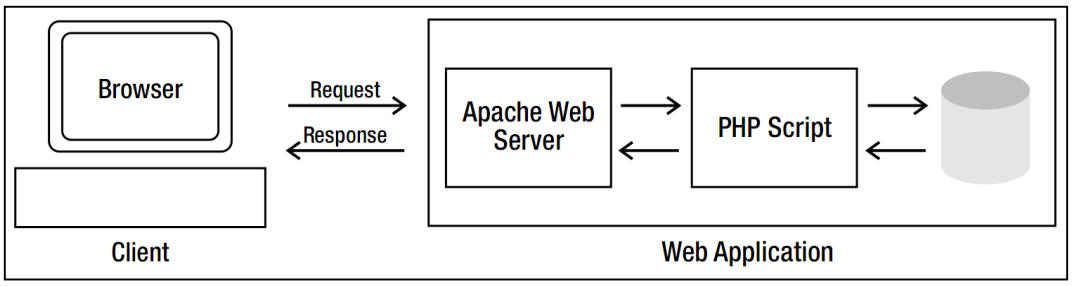


Figura 11- Ciclo de vida de um pedido Web (Harwani 2009, p.13).

No sistema Joomla, o motor PHP executa a aplicação que utiliza um *plugin* para endereçar o servidor MySQL, onde são armazenados os dados do sistema. As definições de configuração do Joomla são armazenadas no ficheiro `configuration.php`, enquanto os restantes dados (conteúdos, secções, categorias, informação relativa às extensões instaladas) são armazenados no servidor MySQL (Rahmel 2007, p.10).

Relativamente aos requisitos para a instalação do Joomla 1.5.x deve-se ter atenção as versões de *software* apresentadas na tabela seguinte.

Tabela 1 - Requisitos para a instalação do Joomla (Joomla! *Technical Requirements* 2010).

Software	Recomendado	Mínimo
PHP	5.2 ou superior	4.3.10 Não utilizar PHP 4.3.9, 4.4.2 ou 5.0.4 devido a *bugs*
MySQL	4.1.x ou superior Não utilizar 6.x, o Joomla 1.5.x não é compatível	3.23
Apache (com mod_mysql, mod_xml e mod_zlib)	2.x	1.3
Microsoft IIS	7	6

Para efectuar a instalação da plataforma Joomla é então necessário realizar os seguintes passos:

1. Preparação de um servidor local ou remoto. Para a instalação do Joomla deverá ser preparado um servidor que obedeça aos requisitos mínimos mencionados anteriormente.
2. *Download* do pacote Joomla. O pacote do Joomla está disponível para download no site oficial (Joomla 2010) ou na comunidade portuguesa do Joomla (JoomlaPT 2010). Nestes *sites* estão também disponíveis guias de instalação e iniciação.
3. Criar uma directoria para o Joomla, se não for o *site* principal do servidor.
4. Mover o pacote de instalação para o servidor.
5. Extrair os ficheiros para o directório, onde será colocado o Joomla.
6. Executar o *browser* com o endereço de acesso à directoria, onde o Joomla foi instalado.

2.3.9. Aplicação de instalação

O Joomla inclui uma aplicação de instalação que se executa no *browser*, a partir da directoria onde se encontra o *site*. Esta aplicação consiste num assistente passo-a-passo que vai solicitando as opções de configuração do sistema e que apresenta as interfaces que se passam a descrever:

1. **Selecção do Idioma** - O primeiro ecrã apresentado ao utilizador permite escolher o idioma da interface de administração do Joomla. Como o pacote utilizado no desenvolvimento desta tese foi retirado da comunidade portuguesa do Joomla, estão apenas disponíveis os idiomas Inglês (en-GB) e Português (pt-PT).

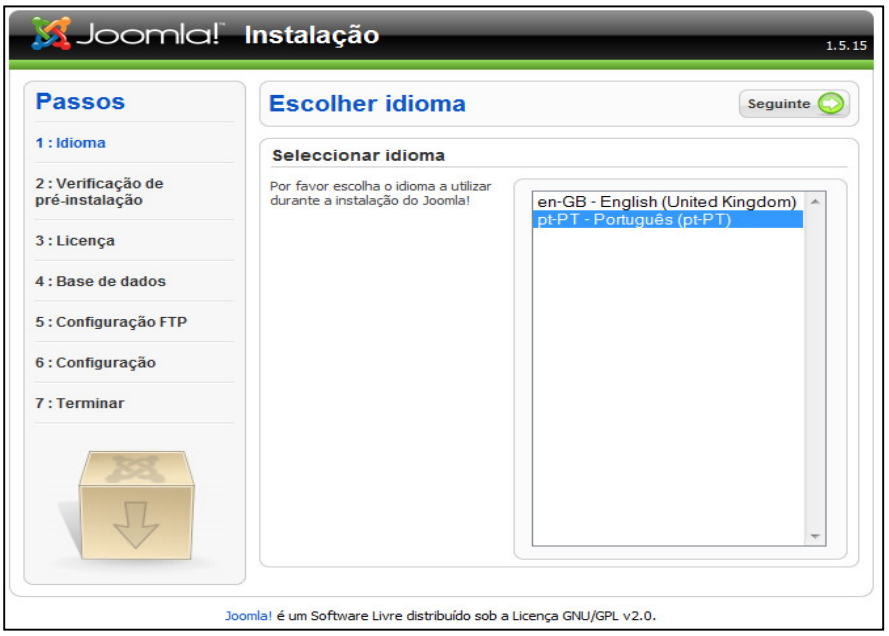


Figura 12- Instalação do Joomla – selecção do idioma.

2. **Verificação de pré-instalação** – Neste momento é realizada a verificação do sistema para determinar se o servidor está devidamente configurado para assegurar o correcto funcionamento do Joomla.

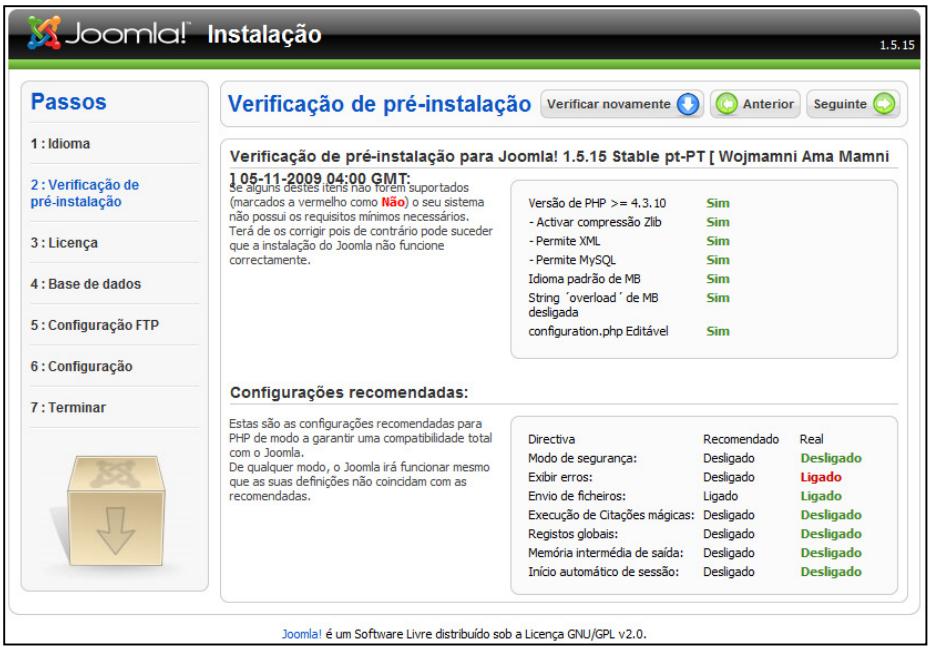


Figura 13- Instalação do Joomla – verificação de pré-instalação.

3. **Licença** – o *software* Joomla é licenciado pela versão 2.0 da licença GNU/GPL.

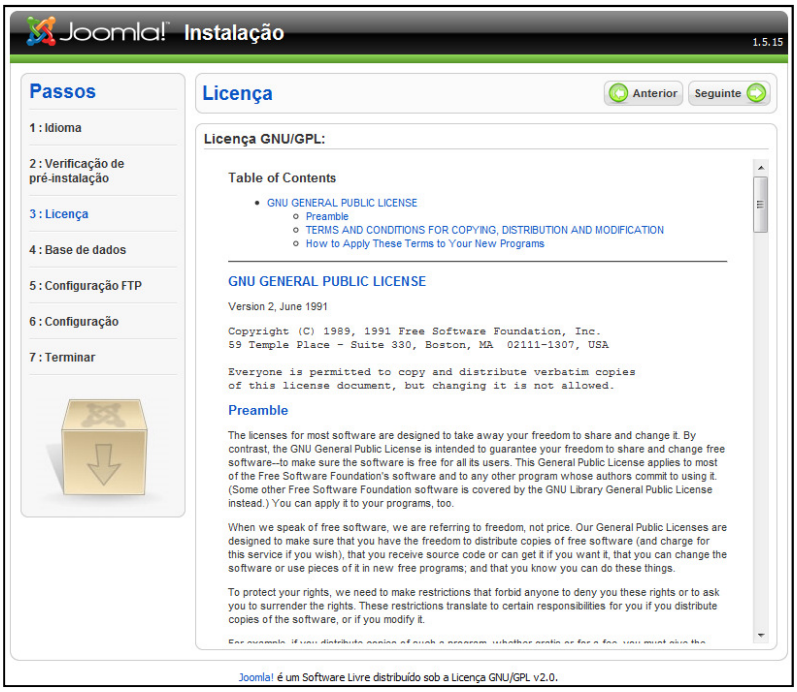


Figura 14- Instalação do Joomla – licença.

4. **Configuração da base de dados -** A aplicação de instalação pede a introdução dos parâmetros de acesso à base de dados que será utilizada pelo Joomla. É pedido o nome do servidor, utilizador da base de dados e o nome da base de dados. A base de dados e um utilizador com todas as permissões a ela relativas já devem ter sido previamente criadas durante o processo de preparação do servidor.

Figura 15- Instalação do Joomla – configuração da base de dados.

5. **Configuração FTP** – A configuração FTP foi incluída no Joomla para evitar problemas com servidores que imponham determinadas restrições ao nível das operações no sistema ou da utilização do modo seguro do PHP.

Figura 16 - Instalação do Joomla – configuração da base de dados.

6. **Configuração do *site*** – Neste passo é configurado o nome do *site*, o *e-mail* do administrador e a senha de acesso do utilizador `admin`. Aqui pode também ser feita a instalação de dados de exemplo para facilitar a compreensão da plataforma ou a importação de um *script* `.sql` de migração.

Figura 17 - Instalação do Joomla – configuração principal.

7. **Finalização da instalação** - Este é o passo final da instalação. Se não existirem erros, é apresentada uma mensagem de felicitação e um aviso para que o administrador remova o directório `instalation` do servidor. Enquanto este directório não for removido o *site* não se executa.

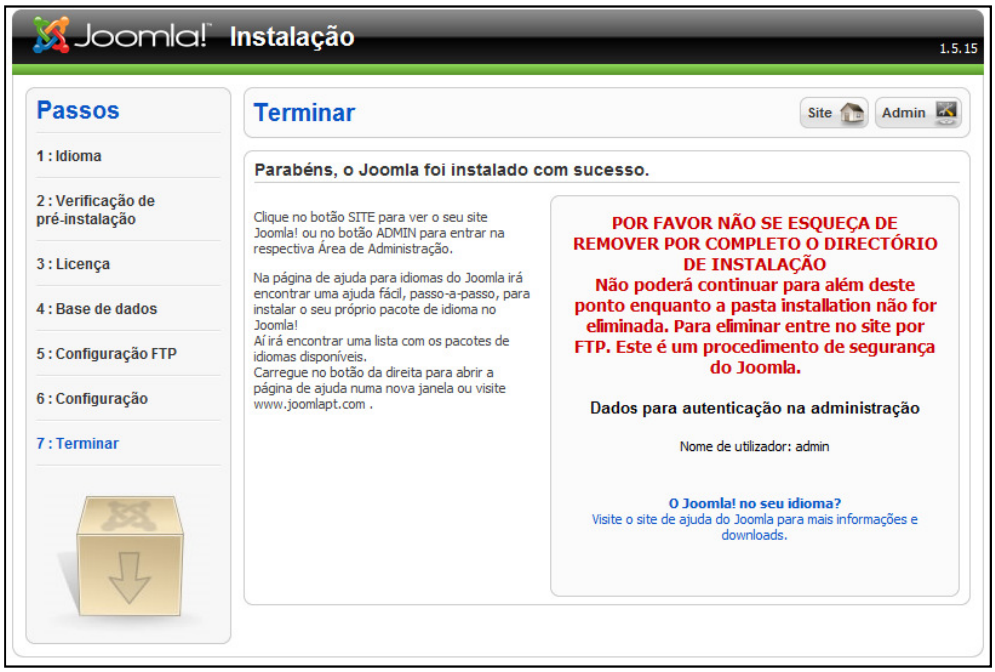


Figura 18 - Instalação do Joomla – finalização.

Após a execução com sucesso dos passos aqui descritos, o *site* pode ser executado a partir do seu endereço, como por exemplo, http://localhost/testingjoomla/, no caso de uma instalação no directório `testingjoomla` de um servidor Web local.

Figura 19 - Execução do Joomla sem conteúdos.

Com o *site* a funcionar, o administrador já pode aceder à interface de administração através do directório `administrator` do Joomla, no caso do exemplo acima demonstrado, `http://localhost/testingjoomla/administrator`.

2.3.10. Site Joomla – *front end*

O *front end* do Joomla é a interface através da qual os visitantes podem visualizar os conteúdos do *site*. O administrador pode definir os conteúdos que no *front end* são visíveis a todos os visitantes e os conteúdos que são visíveis apenas para os utilizadores registados e autenticados.

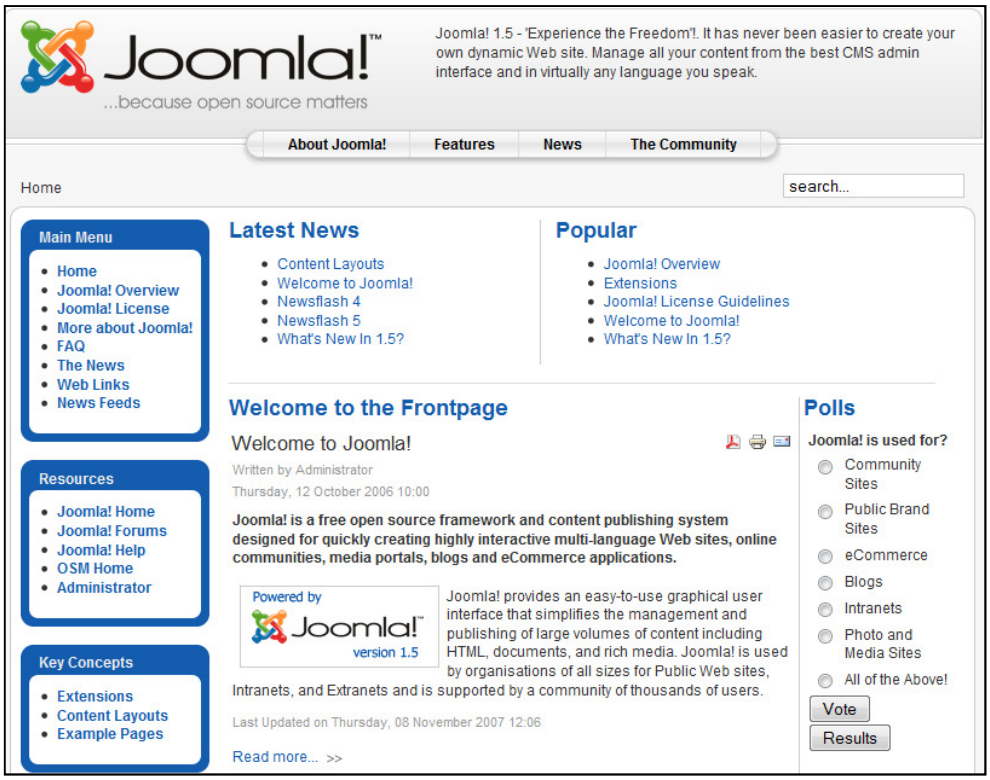


Figura 20 - *Front end* do Joomla.

O *layout* do *front end* é constituído por vários módulos. Estes módulos podem conter menus, conteúdos, publicidade, elementos decorativos e funções adicionais. Podem ser posicionados consoante as definições utilizadas pelo administrador. O aspecto geral do *front end* é apresentado de acordo com um *template* que também pode ser configurado pelo administrador.

O Joomla 1.5.x organiza automaticamente o seu conteúdo de acordo com secções, categorias e artigos. Cada secção pode conter várias categorias. Os artigos, que correspondem efectivamente aos conteúdos do *site*, são normalmente associados a uma dessas categorias. As secções e categorias são criadas apenas pelos administradores, enquanto os artigos podem ser criados pelo administrador ou por um utilizador do tipo Autor, Editor ou Director.

2.3.11. Interface de administração - *back end*

A interface de administração (*back end*) do CMS Joomla permite gerir todo o *site*. O administrador tem acesso através do URL `[nome do domínio]/administrator` e deverá efectuar a autenticação através do utilizador `admin` e da palavra-chave escolhida durante o processo de instalação do sistema.

Figura 21- Painel de autenticação do *back end* Joomla.

Após realizada a autenticação, é apresentado ao administrador o painel de controlo do sistema. Este painel está desenhado de modo que os administradores possam aceder rapidamente às ferramentas mais utilizadas.

Figura 22- Painel de controlo do *back end* Joomla.

No topo da interface encontra-se a barra de navegação principal do *back end* que permite aceder aos menus: `Site`, `Menus`, `Componentes`, `Extensões`, `Ferramentas` e `Ajuda`.

O menu `Site` permite aceder ao painel de controlo, gestão de utilizadores, gestão de ficheiros multimédia, configurações globais e sair do *back end*. Através do menu `Menus` é possível aceder ao gestor de menus, aos menus já criados e menus eliminados. O menu `Artigos`

apresenta o gestor de artigos, a lista dos artigos eliminados, o gestor de secções, o gestor de categorias e o gestor da página principal do *site*. No menu `Componentes` estão disponíveis os componentes instalados no sistema, tais como, agregador de sinais, contactos, inquéritos e outros. O menu `Extensões` permite o acesso à instalação ou desinstalação de extensões, ao gestor de módulos, ao gestor de *plugins*, ao gestor de temas e ao gestor de idiomas. Através do menu `Ferramentas`, o administrador pode enviar e ler mensagens, fazer a validação geral do *site* e limpar a cache do sistema. No menu `Ajuda` estão disponíveis diversos tópicos de ajuda do Joomla e informações acerca do sistema.

2.4. Desenvolvimento de componentes Joomla 1.5.x

Das extensões disponíveis para o Joomla, os componentes são as mais importantes, uma vez que sempre que se executa uma página, o Joomla carrega e executa um determinado componente. Consequentemente, as principais funcionalidades do Joomla, que correspondem à gestão de conteúdos, são também desempenhadas por um componente.

Como já foi dito anteriormente, o Joomla divide-se em interfaces de *back end* e de *front end*, e como as suas funcionalidades dependem de diversos componentes, normalmente cada componente apresenta o seu próprio *back end* e *front end*. O *back end* apresentará as funcionalidades dedicadas aos administradores e o *front end* apresentará as funcionalidades dedicadas ao utilizador comum.

Os componentes são também o tipo de extensões mais complexas do Joomla, por isso, a sua criação requer bastante cuidado, quer na fase de planeamento, quer na fase de desenvolvimento.

Antes de iniciar a criação de um componente, existem três questões essenciais:

- Qual o objectivo do componente?
- Quais as características específicas que possuem os conteúdos do *site* que tornem necessária a inclusão de um componente próprio?
- Existe algum componente disponível que satisfaça, se não todos, a maioria dos requisitos necessários (Lanham 2010, p.134)?

Estas questões são importantes, uma vez que se deve ter noção das reais necessidades de utilizar o Joomla para apresentar conteúdos de uma forma diferente ou de criar um novo mecanismo que facilite a realização de determinadas tarefas.

A criação de componentes para o Joomla requer o conhecimento e aplicação do padrão de programação MVC (*Model-View-Controller*) e da estrutura a ele associada para a construção deste tipo de extensões.

2.4.1. Estrutura de um componente

Os componentes Joomla apresentam uma estrutura de directórios típica. É importante ter a noção desta estrutura para facilitar o correcto desenvolvimento deste tipo de extensões.

No Joomla, cada componente é identificado por um nome que é único, que não tem espaços e que tem o prefixo `com_`. Esse nome, para além de ser utilizado no URI para identificar

um pedido realizado ao componente, identifica também o directório que contém os ficheiros de código responsáveis pelo seu funcionamento. O código é dividido em dois directórios principais, um do lado do *back end*, localizado na pasta `administrator/components` e outro do lado do *front end*, localizado na pasta `components` da raiz do Joomla.

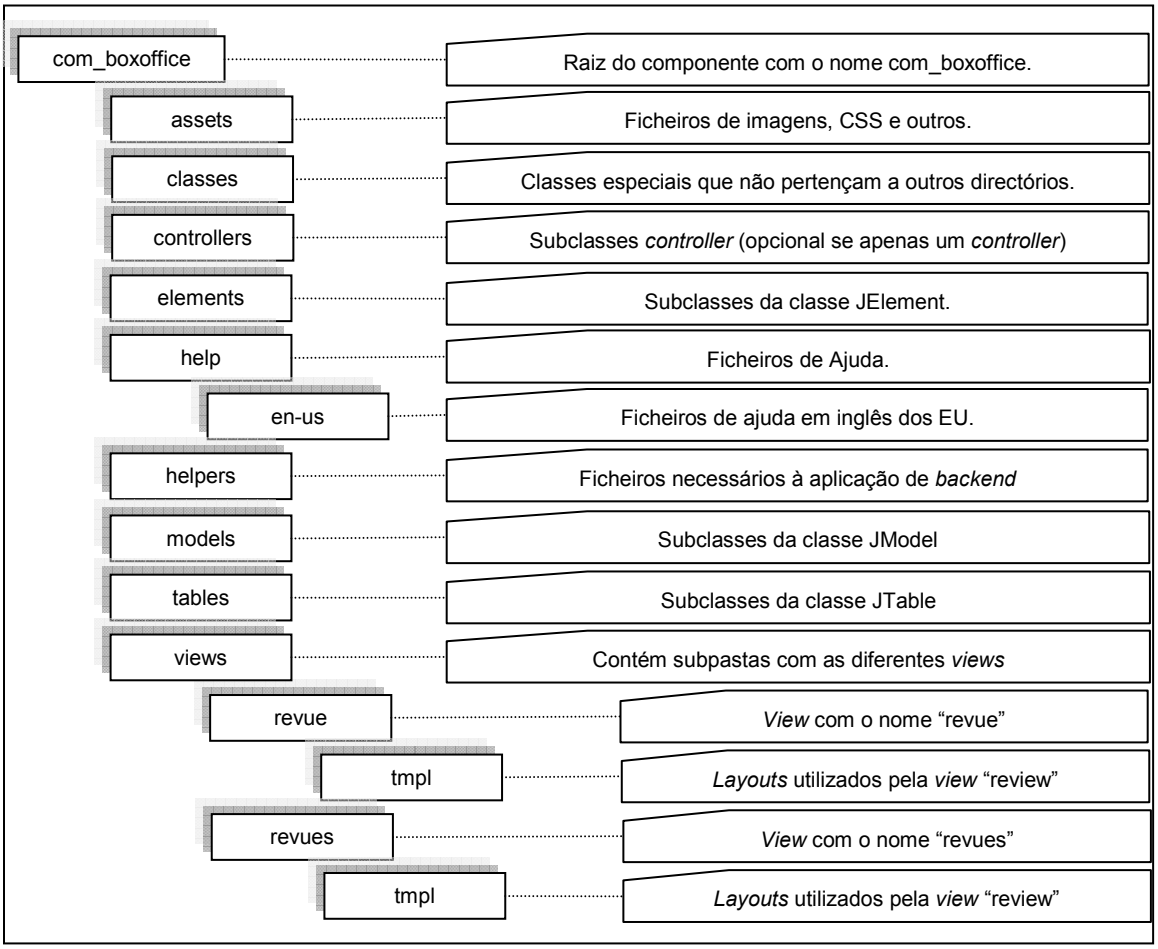


Figura 23 - Estrutura típica de um componente MVC (Lanham 2010, p.109).

A figura acima ilustra a estrutura típica do directório de *back end* de um componente MVC para Joomla. A estrutura referente ao *front end* é idêntica, mas não contém os directórios `elements`, `help` e `tables`.

A figura que se segue mostra parte da estrutura de directórios de uma instalação do Joomla 1.5.x e a localização de um dos seus componentes base, o `com_banners`.

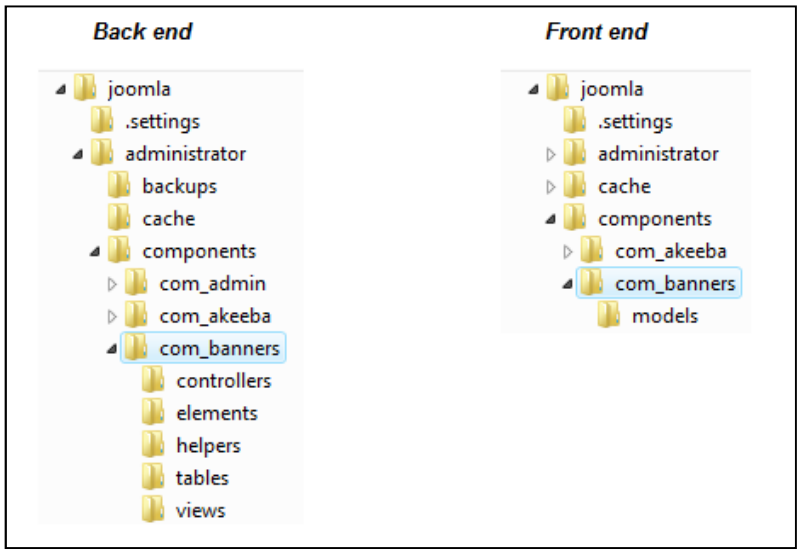


Figura 24 - Localização e estrutura de directórios do componente com_banners

O componente `com_banner` já vem pré-instalado no Joomla. Pode-se verificar que apresenta uma estrutura no lado do *back end* com maior número de directórios do que do lado do *front end*. Isto acontece uma vez que as interfaces e as funcionalidades são diferentes nestas duas partes, sendo a mais complexa do lado do *back end*, onde se encontram as interfaces de administração.

2.4.2. Padrão MVC

O padrão de programação *Model-View-Controller* surgiu com o intuito de proporcionar uma boa forma de organizar o código das aplicações. Este padrão separa o desenho de *software* em três partes funcionalmente diferentes: acesso aos dados, apresentação dos dados e a lógica de funcionamento. Assim, pretende-se que seja possível alterar aspectos relativos ao modo como os dados são apresentados sem ter que reprogramar toda a aplicação.

As três partes do padrão MVC são: *model*, *view* e *controller*. A *controller* e *view* podem ser consideradas como pertencendo à camada de apresentação, enquanto a *model* pode ser vista como uma fusão entre a lógica de funcionamento da aplicação e a camada de acesso aos dados. Cada elemento do padrão MVC é representado no Joomla pelas classes abstractas `JModel`, `JView` e `JController`. Estas classes estão localizadas na biblioteca `joomla.application.component` (Lanham 2010, p.136).

As classes *model* são responsáveis pelo encapsulamento dos dados da aplicação. Estas classes contêm os métodos necessários para devolver, adicionar, remover ou actualizar os dados armazenados num determinado local que, no Joomla, é na maioria das vezes uma base de dados do servidor MySQL. Assim, se uma aplicação trocar de um sistema que utiliza ficheiros para armazenar as suas informações para um sistema que utiliza uma base de dados, as classes *model* são as únicas que necessitam de ser alteradas, mantendo as *controller* e *view* (Joomla! *Developing MVC Component*).

A secção *view* do padrão MVC é responsável pela apresentação dos dados. A *view* devolve os dados provenientes das classes *model* (passados através da secção *controller*) e envia-os para *layouts* que serão apresentados aos utilizadores. Ao utilizar a *view* para apresentar dados no formato HTML, os *layouts* fornecem uma camada de controlo extra que permite a apresentação em diferentes formatos. Assim, páginas HTML podem ser facilmente formatadas para outros formatos, como por exemplo PDF ou *feeds*.

Os dados apresentados numa *view* podem ser oriundos de uma ou mais classes *model*. Estas *model* são automaticamente associadas à respectiva *view*, através das classes *controller* (Lanham 2010, p.137).

A parte *controller* é responsável por responder às acções dos utilizadores. No caso das aplicações Web, a acção do utilizador é geralmente um pedido de uma página (*page request*). As classes *controller* determinam o pedido efectuado pelo utilizador e respondem de forma adequada desencadeando a *model* para fazer a manipulação de dados e enviá-los para a *view*. A parte *controller* não apresenta os dados na *model*, apenas acciona os métodos da *model* responsáveis pela manipulação dos dados e em seguida passa a *model* para a *view* responsável por apresentar esses dados (Joomla! *Developing MVC Component*, 2010).

Existem alguns aspectos importantes a ter em conta na construção da parte *controller* (Lanham 2010, p.137):

- se apenas existir uma entidade principal, deve-se considerar em construir apenas uma classe *controller*;
- se existirem várias entidades, deve-se considerar em construir uma classe *controller* para cada entidade;
- se for necessário manipular várias classes *controller*, deve-se considerar a criação de outra *controller* para instanciar as restantes classes;
- se existirem entidades idênticas, deve-se considerar a criação de uma *controller* abstracta que implemente tarefas semelhantes.

Na construção de componentes MVC para o Joomla são criadas classes *model*, *view*, e *controller* que derivam das classes `JModel`, `JController` e `JView` da API do Joomla. Estas classes relacionam-se como é demonstrado na figura seguinte.

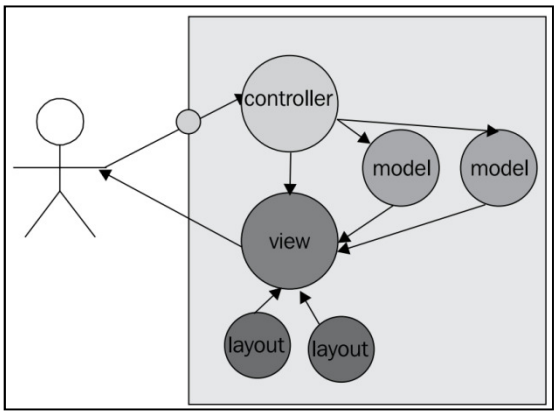


Figura 25 - Relação entre os elementos de um componente MVC (Lanham 2010, p.138)

O Utilizador envia um pedido ao componente para realizar determinada tarefa, o pedido é recebido através do ponto de entrada que identifica o *controller* que deverá ser utilizado. O *controller* utiliza as *models* e *views* necessárias com base na tarefa desencadeada no ponto de entrada. Cada *view* pode ainda ter vários *layouts*. Na figura estão apenas representadas uma *controller* e uma *view*, no entanto cada componente pode conter várias.

2.4.3. Criação de um componente MVC

Para compreender com maior detalhe o processo de criação de um componente MVC para o Joomla, será seguidamente exemplificada a criação de um componente simples, de nome `com_hello` e que terá como objectivo apresentar mensagens de boas vindas. O exemplo que aqui será demonstrado, bem como a sua explicação, foi baseado na série de artigos *Developing a Model-View-Controller Component*, utilizando a versão 1.5 do Joomla e que está disponível no *site* Joomla! Official Documentation.

Para a criação do componente `com_hello` vão ser necessários os seguintes ficheiros:

- `site/hello.php` - consiste no ponto de entrada do componente do lado do *front end*;
- `site/controller.php` - *controller* base;
- `site/views/hello/view.html.php` - devolve os dados necessários e coloca-os no *template*;
- `site/views/hello/tmpl/default.php` - *template* para o *output*;
- `site/models/hello.php` - *model* para a leitura de dados da base de dados;
- `admin/hello.php` - ponto de entrada do componente do lado do *back end*;
- `admin/controller.php` - *controller* base;
- `admin/controllers/hello.php` *controller* adicional;
- `admin/models/hello.php` *model* principal;
- `admin/models/hellos.php` *model* adicional;
- `admin/tables/hello.php` classes extensão de `JTable` para definir as tabelas utilizadas pelo componente;
- `admin/views/hellos/view.html.php` - devolve os dados necessários e coloca-os no *template*;

- `admin/views/hellos/tmpl/default.php` - *template* padrão para o *output*;
- `admin/views/hello/view.html.php` - devolve os dados necessários e coloca-os no *template* adicional;
- `admin/views/hello/tmpl/form.php` - *template* para o *output* adicional;
- `admin/sql/install.mysql.utf8.sql` - *script* para a criação das tabelas necessárias na base de dados do Joomla durante o processo de instalação do componente;
- `admin/sql/uninstall.mysql.utf8.sql` - *script* para eliminar as tabelas da base dados do Joomla durante o processo de desinstalação do componente.
- `install.xml` - ficheiro XML responsável pela instalação do componente;

Em todos os directórios do componente, para evitar que o seu conteúdo seja visível através do *browser,* deve ser incluído um ficheiro `index.html` que apresente uma página em branco, por exemplo, um ficheiro com o código: `<html><body bgcolor="#FFFFFF"></body></html>`.

Relativamente à estrutura dos componentes, os ficheiros responsáveis pelo *front end* da aplicação são colocados no directório `site`, enquanto que os ficheiros de administração são colocados no directório `admin`. Os componentes são instalados na plataforma Joomla através do gestor de extensões. Após a sua instalação, os ficheiros de *front end* (interface do utilizador) são colocados no directório `[raizdositejoomla]/components/com_[nomedocomponente]` e os ficheiros relativos ao *back end* (administração do componente) são colocados no directório `[raizdositejoomla]/administrator/components/com_[nomedocomponente]`. O ficheiro principal (*main file*) do componente tem o próprio nome do componente e localiza-se tanto na raiz do directório `site` como na raiz do directório `admin`, denominando-se de `[nomedocomponente].php`. Na raiz do directório `site` e `admin` é também colocado o *controller* principal do componente. No caso de serem necessários vários *controllers* adicionais, estes devem ser colocados no directório `[raizdositejoomla]/components/com_[nomedocomponente]/controllers` e têm o nome `[nomedocontroller].php`.

Os ficheiros *view* devem estar localizados no directório `[raizdositejoomla]/components/com_[nomedocomponente]/views` e `[raizdositejoomla]/administrator/components/com_[nomedocomponente]/views`. Se for apenas necessário um ficheiro *view* este deverá ter o mesmo nome do componente. No caso de existirem vários ficheiros *view*, o ficheiro *view* principal é o que deverá ter o mesmo nome do componente. No caso de haver necessidade de ter *views* adicionais, os seus ficheiros devem estar localizados em diferentes subdirectórios do directório `views` do componente. Para cada *view* o *template* (*layout*) padrão deve estar localizado no directório `[raizdositejoomla]/components/com_[nomedocomponente]/views/[nomedaview]/tmpl` e `[raizdositejoomla]/administrator/components/com_[nomedocomponente]/views/[nomedaview]/tmpl`.

Os ficheiros *model* estão localizados no directório `[raizdositejoomla]/components/com_[nomedocomponente]/models` e `[raizdositejoomla]/administrator/components/com_[nomedocomponente]/models`. O nome dos ficheiros *model* deve coincidir com o nome da *view* que apresentará os dados devolvidos pela *model*.

A localização de todos os ficheiros é definida no ficheiro `xml` responsável pela instalação do componente, normalmente denominado de `install.xml` ou `[nomedocomponente].xml`. Os nomes dos ficheiros e directórios do componente devem estar escritos em minúsculas para evitar problemas em sistemas Unix/Linux.

Outro dos aspectos a ter em conta na criação de componentes MVC para o Joomla, diz respeito às normas utilizadas para atribuir o nome às classes do componente. A classe *controller* base do *front end* deverá ter o nome `[NomedoComponente]Controller`. No *back end*, ao nome da classe *controller* base deve ser adicionado um "s" à frente de `[NomedoComponente]`, ficando com o nome `[NomedoComponente]sController`. O nome das classes *controller* adicionais e que se encontram no directório `controllers` devem ter o nome `[NomedoComponente]Controller[NomedoController]` para o *front end* (directório site) e `[NomedoComponente]sController[NomedoController]` para o *back end* (directório admin). Para as classes *model* e *view* aplicam-se as mesmas normas.

2.4.3.1. *Front end* - criação do ponto de entrada

O acesso ao sistema Joomla é feito por um único ponto de entrada, o ficheiro `index.php` do *front end* da aplicação ou o ficheiro `administrator/index.php` para o *back end*. Seguidamente, a aplicação carrega o componente proveniente do pedido (*request*) realizado através do *browser* e que consta no valor de `option` do URL ou dos dados POST. No caso do componente `com_hello` o URL será: `index.php?option=com_hello&view=hello`. Através deste pedido é carregado o ficheiro principal do componente (`components/com_hello/hello.php`.), que corresponde ao seu único ponto de entrada. O código escrito neste ficheiro é bastante comum entre os componentes MVC para Joomla.

```
<?php
/**
 * Hello World entry point file for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://dev.joomla.org/component/option,com_jd-wiki/Itemid,31/ id,tutorials:
components/
 * @license           GNU/GPL
 */

// no direct access
defined('_JEXEC') or die('Restricted access');

// Require the base controller
require_once (JPATH_COMPONENT.DS.'controller.php');

// Require specific controller if requested
if($controller = JRequest::getVar('controller')) {
       require_once (JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php');
}

// Create the controller
$classname    = 'HelloController'.$controller;
```

```
$controller = new $classname();

// Perform the Request task
$controller->execute( JRequest::getVar('task'));

// Redirect if set by the controller
$controller->redirect();

?>
```

Por questões de segurança, deve-se começar por verificar se existem permissões para execução do ficheiro através da constante `_JEXEC` que está definida no ficheiro `index.php` da raiz do *site*. No caso de se tentar aceder directamente ao ficheiro `hello.php` é devolvida a mensagem "Restricted Access", uma vez que `_JEXEC` ainda não foi definida.

Para efectuar o carregamento dos ficheiros necessários, utiliza-se a constante `JPATH_COMPONENT` que contém o caminho absoluto do componente em execução e que, no caso do *front end* deste componente, corresponde a `components/com_hello`. No ponto de entrada correspondente ao *front end* do componente são carregados os ficheiros relativos ao *controller* base, e no caso de serem necessários ficheiros *controller* adicionais, estes são carregados a partir do directório *controllers*. A constante `DS` contém o separador de directórios utilizado pelo sistema no qual o Joomla foi instalado, que corresponderá a "/" no caso de sistemas UNIX/Linux e "\" no caso de sistemas Windows. Esta situação é automaticamente definida pela *framework* do Joomla.

O método `JRequest::getVar('controller')` procura o valor da variável `controller` no URL ou nos dados POST do pedido. No caso do URL ser `index.php?option=com_hello&controller=controller_name`, o método irá devolver o valor `controller_name`.

Após efectuar o carregamento dos ficheiros *controller* necessários ao componente, é efectivamente criado o *controller* e executada a tarefa definida através do URL ou dos dados POST. O URL `index.php?option=com_hello&task=sometask` identifica que a tarefa a ser executada é `sometask`. No caso de não ser definida uma tarefa, é executada a tarefa padrão `display` e a variável `view` irá definir o que será apresentado na interface. Algumas das tarefas mais comuns são: `save`, `edit`, `new` e `remove`. Por fim, o *controller* pode decidir redireccionar a página após a conclusão da tarefa.

Resumindo, o ponto de entrada do componente (`hello.php`), passa o controlo da aplicação para o *controller* e trata da execução da tarefa especificada no pedido.

2.4.3.2. *Front end* - criação da classe *controller* base

A classe *controller* é responsável por dar resposta às acções (ou tarefas) dos utilizadores. O componente `com_hello` apresenta apenas uma tarefa que corresponde à apresentação de uma mensagem de boas vindas e por isso, o seu *controller* é bastante simples. Esta *controller* não necessita de efectuar qualquer tratamento aos dados, necessita apenas de efectuar o carregamento da *view* adequada, bastando para isso criar o método `display()` na classe

controller. Como a maioria das funcionalidades já estão implementadas na classe `JController` da *framework* é apenas necessário invocar o método `JController::display()`.

A classe *controller* base é implementada no ficheiro `site/controller.php` e tem o seguinte código:

```
<?php
/**
 * Hello World default controller
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://dev.joomla.org/component/option,com_jd-wiki/Itemid,31/id,
tutorials:components/
 * @license           GNU/GPL
 */

jimport('joomla.application.component.controller');

/**
 * Hello World Component Controller
 *
 * @package           HelloWorld
 */
class HelloController extends JController
{
        /**
         * Method to display the view
         *
         * @access    public
         */
        function display()
        {
                parent::display();
        }

}
?>
```

O método `JController::display()` instancia a *view*, atribui a *model* à *view*, e inicializa a *view*. Este método verifica as variáveis `view` e `layout` do pedido. A variável `view` determina a *view* que será instanciada e a variável `layout` determina o *layout* a ser utilizado. A *view* refere-se normalmente ao conjunto de dados que se pretendem visualizar, enquanto que o *layout* corresponde ao modo como esses dados são organizados.

No *front end* do componente `com_hello` tem-se apenas uma *view* de nome `hello` e um *layout* de nome `default`.

2.4.3.3. *Front end* - criação da classe *model*

As classes *model* são responsáveis pela manipulação dos dados. Um componente MVC pode ter várias classes *model*, sendo normalmente utilizada uma classe *model* por cada entidade. Como o componente `com_hello` tem apenas uma entidade (tem apenas uma tabela na base de dados) terá apenas uma classe *model*. Tendo em conta as normas relativas à *framework* do Joomla para a atribuição dos nomes às classes, a classe *model* do *front end* do componente terá o nome `HelloModelHello`. Todas as classes *model* derivam da classe `JModel` da *framework*.

A classe `HelloModelHello` responsável por devolver da base de dados a mensagem de boas vindas, foi implementada no ficheiro `site/model/hello.php` e contém o seguinte código:

```php
<?php
/**
 * Hello Model for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://dev.joomla.org/component/option,com_jd-wiki/Itemid,31/id,tutorials:
components/
 * @license           GNU/GPL
 */

// Check to ensure this file is included in Joomla!
defined('_JEXEC') or die();

jimport( 'joomla.application.component.model' );

/**
 * Hello Model
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HelloModelHello extends JModel
{
        /**
         * Gets the greeting
         * @return string The greeting to be displayed to the user
         */
        function getGreeting()
        {
                $db =& JFactory::getDBO();

                $query = 'SELECT greeting FROM #__hello';
                $db->setQuery( $query );
                $greeting = $db->loadResult();

                return $greeting;
        }
}
```

Como se pode observar no código acima, foi criado o método `getGreeting()` no qual é estabelecida a ligação à base de dados e devolvida a mensagem da respectiva tabela. Para o acesso aos dados começa-se por obter a referência a um objecto da base de dados. Como o CMS Joomla já utiliza a base de dados, já existe uma ligação e por isso é apenas necessário obter a referência dessa ligação através do método `JFactory::getDBO()`. Através da referência ao objecto da base de dados é possível criar a consulta e obter o resultado. A consulta é definida através do método `setQuery($query)` e o resultado pode ser obtido através do método `loadResult()`.

2.4.3.4. *Front end* - criação da classe *view e* do *layout*

A *view* tem como função devolver os dados a serem apresentados e colocá-los no *template*. Podem existir várias *views*, mas cada uma tem o seu próprio directório, que por sua vez se encontra dentro do directório `views`. Em cada directório da *view* é definido um ficheiro para cada tipo de documento que a *view* suporta (HTML, PDF, *feed* e RAW). No componente `com_hello` será utilizado um documento HTML e será criado o directório `tmpl` que irá conter os templates (*layouts*) utilizados pela *view*.

A classe *view* do *front end* do componente terá o nome `HelloViewHello` e, como todas as classes *view*, deriva da classe `JView` da *framework*. A implementação desta classe é realizada no ficheiro `site/views/hello/view.html.php` e apresenta o seguinte código:

```
<?php
/**
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_2
 * @license    GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

jimport( 'joomla.application.component.view');

/**
 * HTML View class for the HelloWorld Component
 *
 * @package    HelloWorld
 */
class HelloViewHello extends JView
{
        function display($tpl = null)
        {
                $model =& $this->getModel();
                $greeting = $model->getGreeting();
                $this->assignRef( 'greeting', $greeting );

                parent::display($tpl);
        }
}
```

A *framework* do Joomla está configurada de modo a que cada *controller* carregue automaticamente a *model* com o mesmo nome da *view*. Uma vez que a *view* do componente tem o nome "`hello`", a classe *model* com o nome "`hello`" é automaticamente carregada e colocada na respectiva *view*. Assim, a referência à classe *model* é facilmente obtida através do método `JView::getModel()`. No caso de haver necessidade de aceder a uma *model* com um nome diferente da *view*, pode-se passar o nome da *model* através do método, por exemplo, `getModel('nomedamodel')`. Após obtida a referência à *model* do exemplo aqui seguido, é possível aceder ao seu método `getGreeting()` que é responsável por devolver os dados. Os dados são colocados no *template* através do método `JView::assignRef`.

A apresentação dos dados é realizada através do *template* que recebe os dados passados para a *view*. No componente `com_hello` foi criado o *template* `site/views/hello/tmpl/default.php` com o seguinte código:

```
<?php defined('_JEXEC') or die('Restricted access'); ?>
      <h1><?php echo $this->greeting; ?></h1>
```

Este *template* apresenta os dados que chegaram à *view*, através do acesso às variáveis passadas pelo método `JView::assignRef` na *view* e que podem ser acedidas a partir do *template* utilizando `$this->nomedavariavel`, no exemplo, através de `$this->greeting`.

2.4.3.5. *Back end* - criação do ponto de entrada

A criação do *back end* do componente é bastante similar à criação do seu *front end*. A estrutura de directórios do *back end* é idêntica, mas agora dentro do directório raiz `admin`. O ponto de entrada do *back end* é o ficheiro `hello.php`. Este ficheiro é idêntico ao ficheiro `hello.php` do *front end*, com excepção do nome da classe *controller* que carrega, uma vez que se passa a chamar `HellosController` (com "s" a seguir a "Hello"). O *controller* base é também denominado de `controller.php` e apenas o nome da classe passa a ser `HellosController`, em vez de `HelloController` como no *front end*. Esta diferença permitirá que a classe `JController` carregue por defeito a *view* `hellos` que apresentará a lista de mensagens de boas vindas do componente. O seguinte código corresponde então ao ficheiro `admin/hello.php`, ponto de entrada do *back end*.

```
<?php
/**
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license    GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

// Require the base controller

require_once( JPATH_COMPONENT.DS.'controller.php' );

// Require specific controller if requested
if($controller = JRequest::getWord('controller')) {
        $path = JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php';
        if (file_exists($path)) {
                require_once $path;
        } else {
                $controller = '';
        }
}

// Create the controller
$classname    = 'HellosController'.$controller;
$controller   = new $classname( );

// Perform the Request task
$controller->execute( JRequest::getVar( 'task' ) );

// Redirect if set by the controller
$controller->redirect();
```

Como se pode observar, o código do ponto de entrada do *back end* é idêntico ao código do ponto de entrada do *front end* já apresentado e descrito anteriormente.

2.4.3.6. *Back end* - criação de classes JTable

Para se poder manipular a tabela utilizada pelo componente é necessário criar uma classe derivada da classe `JTable` da *framework*. Esta classe é criada no ficheiro `hello.php`, ficheiro este que tem o mesmo nome da tabela e onde é implementada a classe `TableHello` derivada da classe `JTable`. Esta classe é então implementada no ficheiro `admin/tables/hello.php` e apresenta o seguinte código:

```php
<?php
/**
 * Hello World table class
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license           GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

/**
 * Hello Table class
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class TableHello extends JTable
{
        /**
         * Primary Key
         *
         * @var int
         */
        var $id = null;

        /**
         * @var string
         */
        var $greeting = null;

        /**
         * Constructor
         *
         * @param object Database connector object
         */
        function TableHello(& $db) {
                parent::__construct('#__hello', 'id', $db);
        }
      }
```

São criados os atributos `id` e `greeting` que correspondem aos campos da tabela `hello`. É também criado o construtor que especifica o nome da tabela e o campo chave.

2.4.3.7. *Back end* - criação de classes *model*

No do *back end* do componente `com_hello` vão ser utilizadas duas interfaces, uma para visualizar e operar a lista de mensagens e outra para visualizar e operar uma mensagem específica. Como existiram duas interfaces diferentes terão que ser criadas as suas respectivas *views*. No desenvolvimento de componentes MVC para o Joomla é boa prática criar uma classe *model* para cada *view*, portanto serão também criadas duas classes *model*. Às classes *model* foram dados os nomes `HellosModelHellos` e `HellosModelHello`, sendo implementadas nos respectivos ficheiros `admin/models/hellos.php` e `admin/models/hello.php`.

A classe `HellosModelHellos` é muito simples, uma vez que tem apenas como objectivo devolver as mensagens de boas vidas que constam numa tabela da base de dados. Esta classe é implementada no ficheiro `admin/models/hellos.php` e apresenta o seguinte código:

```php
<?php
/**
 * Hellos Model for Hello World Component
 *
```

```
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license           GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

jimport( 'joomla.application.component.model' );

/**
 * Hello Model
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosModelHellos extends JModel
{
        /**
         * Hellos data array
         *
         * @var array
         */
        var $_data;


        /**
         * Returns the query
         * @return string The query to be used to retrieve the rows from the database
         */
        function _buildQuery()
        {
                $query = ' SELECT * '
                        . ' FROM #__hello '
                ;

                return $query;
        }

        /**
         * Retrieves the hello data
         * @return array Array of objects containing the data from the database
         */
        function getData()
        {
                // Lets load the data if it doesn't already exist
                if (empty( $this->_data ))
                {
                        $query = $this->_buildQuery();
                        $this->_data = $this->_getList( $query );
                }

                return $this->_data;
        }
}
```

Como se pode observar no código acima, a classe `HellosModelHellos` apresenta apenas dois métodos, `_buildQuery()` e `getData()`. O método `_buildQuery()` (método privado, começa por "_") é responsável pela criação da consulta à base de dados. Tendo em conta que todas as tabelas da base de dados do Joomla têm um prefixo, que normalmente é "`jos_`", podendo ser outro, quando são efectuadas consultas em SQL deve-se utilizar um prefixo simbólico "`#__`" (cardinal e dois caracteres *underscore*) para que o componente possa funcionar correctamente, independentemente do prefixo utilizado na instalação do Joomla. O método `getData()`irá obter a consulta e devolver os registos da base de dados. Como pode haver a necessidade de se devolver os dados mais do que uma vez quando a página é carregada, foi criado o atributo `_data` para guardar esses dados, evitando a necessidade de realizar novamente a consulta à base de dados. Pode-se verificar que aqui existe um conflito na convenção de nomes

dos atributos da classe. As classes ou atributos protegidos e privados devem ser precedidos de um "_" mas como neste caso, o atributo é devolvido para a *view*, onde será directamente acedido o atributo `_data`, é considerado público. A classe `JModel` da *framework* já contém o método `_getList()` que devolve uma lista de registos (que corresponde a uma lista de objectos), sendo apenas necessário passar a consulta como parâmetro (`_getList( $query )`).

A classe *model* `HellosModelHello` que é responsável por devolver, guardar e eliminar registos na base de dados, está implementada no ficheiro `admin/models/hello.php` e apresenta o seguinte código:

```
<?php
/**
 * Hello Model for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license           GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

jimport('joomla.application.component.model');

/**
 * Hello Hello Model
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosModelHello extends JModel
{

        /**
         * Hellos data array
         *
         * @var array
         */
        var $_data;

        /**
         * Hellos id
         *
         * @var int
         */
        var $_id;

        /**
         * Constructor that retrieves the ID from the request
         *
         * @access     public
         * @return     void
         */
        function __construct()
        {
                parent::__construct();

                $array = JRequest::getVar('cid',  0, '', 'array');
                $this->setId((int)$array[0]);
        }

        /**
         * Method to set the hello identifier
         *
         * @access     public
         * @param      int Hello identifier
         * @return     void
         */
        function setId($id)
```

```
{
        // Set id and wipe data
        $this->_id            = $id;
        $this->_data    = null;
}

/**
 * Method to get a hello
 * @return object with data
 */
function &getData()
{
        // Load the data
        if (empty( $this->_data )) {
                $query = ' SELECT * FROM #__hello '.
                                '  WHERE id = '.$this->_id;
                $this->_db->setQuery( $query );
                $this->_data = $this->_db->loadObject();
        }
        if (!$this->_data) {
                $this->_data = new stdClass();
                $this->_data->id = 0;
                $this->_data->greeting = null;
        }
        return $this->_data;
}

/**
 * Method to store a record
 *
 * @access     public
 * @return     boolean True on success
 */
function store()
{
        $row =& $this->getTable();

        $data = JRequest::get( 'post' );

        // Bind the form fields to the hello table
        if (!$row->bind($data)) {
                $this->setError($this->_db->getErrorMsg());
                return false;
        }

        // Make sure the hello record is valid
        if (!$row->check()) {
                $this->setError($this->_db->getErrorMsg());
                return false;
        }

        // Store the web link table to the database
        if (!$row->store()) {
                $this->setError( $row->getErrorMsg() );
                return false;
        }

        return true;
}

/**
 * Method to delete record(s)
 *
 * @access     public
 * @return     boolean True on success
 */
function delete()
{
        $cids = JRequest::getVar( 'cid', array(0), 'post', 'array' );

        $row =& $this->getTable();

        if (count( $cids )) {
                foreach($cids as $cid) {
                        if (!$row->delete( $cid )) {
                                $this->setError( $row->getErrorMsg() );
                                return false;
                        }
                }
        }
```

```
            return true;
        }

}
```

Esta classe *model* tem como atributos `_id` e `_data`. O atributo `_id` contém o identificador da mensagem e o atributo `_data` contém a mensagem.

A função construtora `__construct()` vai utilizar o método `JRequest::getVar()` para aceder aos dados do pedido. O primeiro parâmetro corresponde ao nome da variável do formulário e o segundo é o valor padrão a atribuir, caso não exista um valor atribuído à variável. O terceiro parâmetro identifica o método utilizado no pedido (POST, GET, FILES, COOKIE, METHOD) e o quarto corresponde ao tipo de dados ao qual deverá pertencer o valor da variável contida no pedido. O construtor carrega o primeiro valor do *array* `cid` e atribui esse valor ao `id`. O método `setID()` é utilizado para definir o `_id` da mensagem. Havendo alteração do `_id` implica a alteração dos dados correspondentes, então é realizada a limpeza do atributo _data.

O método `getData()` verifica se o atributo `_data` já foi definido. Caso se verifique é devolvido, caso contrário os dados são carregados a partir da base de dados.

Para guardar as mensagens na base de dados foi criado o método `store()`. Neste método é obtida a referência à tabela da base de dados através do método `getTable()`. Note-se que não foi necessário identificar o nome da tabela, isto porque foi criado o ficheiro `hello.php` no directório `tables`, no qual foi implementada a classe da tabela com o nome `TableHello`. Ao seguir esta convenção, a classe `JModel` consegue criar o objecto automaticamente sem ser necessário indicar o nome da tabela como parâmetro.

Após obtida a referência à tabela `hello`, os dados do formulário são atribuídos a um objecto da tabela, verificados e é guardado o registo na base de dados. Estas três últimas tarefas são efectuadas métodos `bind()`, `check()` e `store()`que pertencem à classe JTable da *framework*. Em alternativa poderia ter sito utilizado o método `save()` que já combina estes três métodos.

Nesta classe *model* foi também criado o método `delete()` que permite eliminar um registo da base de dados. Aqui, é invocado o método `JRequest::getVar()` que devolve o *array* com os identificadores a eliminar. Obtida a referência à tabela (`$row =& $this->getTable();`), os registos são eliminados através do método `delete()` da classe `JTable`.

2.4.3.8. *Back end* - criação das classes *view* e dos *layout's*

No *back end* do componente `com_hello` foram criadas duas classes *view*, a classe `HellosViewHellos` e a classe `HellosViewHello`. A classe `HellosViewHellos` é responsável pela interface principal do componente e apresenta, através do *layout* `default.php`, a lista de mensagens. Esta classe apresenta o seguinte código:

```
<?php
/**
 * Hellos View for Hello World Component
 *
 * @package    Joomla.Tutorials
```

```
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license           GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport( 'joomla.application.component.view' );
/**
 * Hellos View
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosViewHellos extends JView
{
        /**
         * Hellos view display method
         * @return void
         **/
        function display($tpl = null)
        {
                JToolBarHelper::title(   JText::_( 'Hello Manager' ), 'generic.png' );
                JToolBarHelper::deleteList();
                JToolBarHelper::editListX();
                JToolBarHelper::addNewX();

                // Get data from the model
                $items          = & $this->get( 'Data');

                $this->assignRef('items',            $items);

                parent::display($tpl);
        }
}
```

No método `display()` desta *view* é criada uma barra de ferramentas que permite accionar as tarefas de manipulação das mensagens (adicionar, editar e eliminar); é obtida a referência ao método da classe *model* que devolve os dados da base de dados e os envia para o *layout* através do método `assignRef()` da classe `JView`.

A barra de ferramentas é criada recorrendo a métodos da classe `JToolBarHelper` da *framework*. O método `title( JText::_( 'Hello Manager' ), 'generic.png' )` adiciona o título à barra e define a imagem a utilizar no botão. Note-se que no primeiro parâmetro está a ser utilizado o método `JText::_` para facilitar a tradução do idioma do componente. Este método vai procurar a *string* no ficheiro de idioma do componente e devolve a sua tradução, caso não encontre é devolvida a mesma mensagem que foi passada como parâmetro. A figura seguinte mostra a barra de ferramentas do componente.

Figura 26 - Barra de ferramentas do componente `com_hello`

O método `deleteList()` pode ter até três parâmetros opcionais. O primeiro corresponde a uma *string* com a mensagem de confirmação de eliminação dos registos. O segundo parâmetro corresponde à tarefa enviada no pedido e que accionará o respectivo método no *controller* (por defeito é "remove"). O terceiro parâmetro é o texto que aparece por baixo do botão.

Os métodos `editListX()` e `addNewX()` podem conter dois parâmetros opcionais. O primeiro é a tarefa (por defeito é "edit" e "add", respectivamente) e o segundo é o texto que

aparece por baixo do botão. Para efeitos de tradução, pode também ser utilizado o método `JText::_` no segundo parâmetro destes dois métodos.

O *layout* que apresenta os dados enviados para a *view* é implementado no ficheiro `admin/views/hellos/tmpl/default.php` e apresenta o seguinte código:

```
<?php defined('_JEXEC') or die('Restricted access'); ?>
<form action="index.php" method="post" name="adminForm">
<div id="editcell">
	<table class="adminlist">
	<thead>
		<tr>
			<th width="5">
				<?php echo JText::_( 'ID' ); ?>
			</th>
			<th width="20">
				<input type="checkbox" name="toggle" value=""
onclick="checkAll(<?php echo count( $this->items ); ?>);" />
			</th>
			<th>
				<?php echo JText::_( 'Greeting' ); ?>
			</th>
		</tr>
	</thead>

	<?php
	$k = 0;

	for ($i=0, $n=count( $this->items ); $i < $n; $i++) {
		$row = &$this->items[$i];
		$checked	= JHTML::_('grid.id',   $i, $row->id );
		$link		= JRoute::_(
'index.php?option=com_hello&controller=hello&task=edit&cid[]='. $row->id );
		?>
		<tr class="<?php echo "row$k"; ?>">
			<td>
				<?php echo $row->id; ?>
			</td>
			<td>
				<?php echo $checked; ?>
			</td>
			<td>
				<a href="<?php echo $link; ?>"><?php echo $row->greeting;
?></a>
			</td>
		</tr>
		<?php
		$k = 1 - $k;
	}
	?>

	</table>

</div>

<input type="hidden" name="option" value="com_hello" />
<input type="hidden" name="task" value="" />
<input type="hidden" name="boxchecked" value="0" />
<input type="hidden" name="controller" value="hello" />

</form>
```

Para facilitar a construção dos *layouts* do componente devem ser utilizadas as CSS e os *scripts* Javascript que o pacote do Joomla já fornece. Assim, o nome do formulário deve ser `adminForm` e deve ser utilizado o método `POST` ao submeter. Para a selecção das mensagens são utilizadas caixas de selecção (*checkbox*) com o nome `toggle`. A função Javascript `checkAll()` já se encontra implementada no ficheiro `/includes/js/joomla.javascript.js` da instalação do Joomla. Utilizando esta função, é apenas necessário passar como parâmetro o número de mensagens da lista e ao clicar na *checkbox* é possível

seleccionar ou desseleccionar todas as caixas de selecção. Através do método `JHTML::_()` é possível gerar as caixas de selecção para cada linha da tabela e que correspondem a um registo individual. Os *links* associados às mensagens são criados através do método `JRoute::_()` e contêm a variável `option` com o nome do componente, a variável `controller` com o nome do *controller*, a variável `task` com o nome da tarefa a executar e o *array* `cid` com o identificador das mensagens (`JRoute::_( 'index.php?option=com_hello&controller=hello&task=edit&cid[]='. $row->id );`).

Os últimos quatro campos do formulário estão ocultos. O primeiro campo, com o nome de `option`, é necessário para que o Joomla não deixe de executar o componente. O segundo campo, o campo `task`, irá guardar a tarefa escolhida através do botão da barra de ferramentas do componente. O campo com o nome `boxchecked` servirá para guardar o número de caixas de selecção seleccionadas. O último campo, com o nome `controller`, é utilizado para se especificar o nome do *controller* responsável por tratar a tarefa enviada pelo formulário.

Com a implementação da *view* `hellos` e do seu *layout* `default`, a interface apresentada no *back end* será idêntica à interface da figura seguinte.

Figura 27 - *Layout* `default` da *view* `hellos` no componente `com_hello`.

A classe `HellosViewHello` responsável por apresentar o *layout* `form` com formulário de edição ou adição de mensagens, é implementada no ficheiro `admin/views/hello/view.html.php` e apresenta o seguinte código:

```
<?php
/**
 * Hello View for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license            GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

jimport( 'joomla.application.component.view' );

/**
 * Hello View
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosViewHello extends JView
{
```

```
        /**
         * display method of Hello view
         * @return void
         **/
        function display($tpl = null)
        {
                //get the hello
                $hello          =& $this->get('Data');
                $isNew          = ($hello->id < 1);

                $text = $isNew ? JText::_( 'New' ) : JText::_( 'Edit' );
                JToolBarHelper::title(   JText::_( 'Hello' ).': <small><small>[ ' . $text.'
]</small></small>' );
                JToolBarHelper::save();
                if ($isNew)  {
                        JToolBarHelper::cancel();
                } else {
                        // for existing items the button is renamed `close`
                        JToolBarHelper::cancel( 'cancel', 'Close' );
                }

                $this->assignRef('hello',            $hello);

                parent::display($tpl);
        }
}
```

À semelhança da *view* `hellos`, o método `display()` da *view* `hello` começa com a criação da barra de ferramentas e com a referência ao método da *model* que contém os dados. Os dados são então enviados através do método `assignRef()` para o *layout* `form` da *view*. Para a construção do *layout* `form` da *view* é criado o ficheiro `admin/views/hellos/tmpl/form.php` com o seguinte código:

```
<?php defined('_JEXEC') or die('Restricted access'); ?>

<form action="index.php" method="post" name="adminForm" id="adminForm">

<div class="col100">
        <fieldset class="adminform">
                <legend><?php echo JText::_( 'Details' ); ?></legend>

                <table class="admintable">
                <tr>
                        <td width="100" align="right" class="key">
                                <label for="greeting">
                                        <?php echo JText::_( 'Greeting' ); ?>:
                                </label>
                        </td>
                        <td>
                                <input class="text_area" type="text" name="greeting"
id="greeting" size="32" maxlength="250" value="<?php echo $this->hello->greeting;?>" />
                        </td>
                </tr>
        </table>
        </fieldset>
</div>

<div class="clr"></div>

<input type="hidden" name="option" value="com_hello" />
<input type="hidden" name="id" value="<?php echo $this->hello->id; ?>" />
<input type="hidden" name="task" value="" />
<input type="hidden" name="controller" value="hello" />
</form>
```

Importa aqui destacar que foi adicionado um campo oculto de nome `id` que vai guardar o identificador da mensagem, uma vez que o utilizador não necessita (nem deve) alterar o valor

deste campo. Para facilitar a criação desta interface são novamente utilizadas as classes disponíveis nas CSS do Joomla, tais como, `adminForm` e `adminTable`.

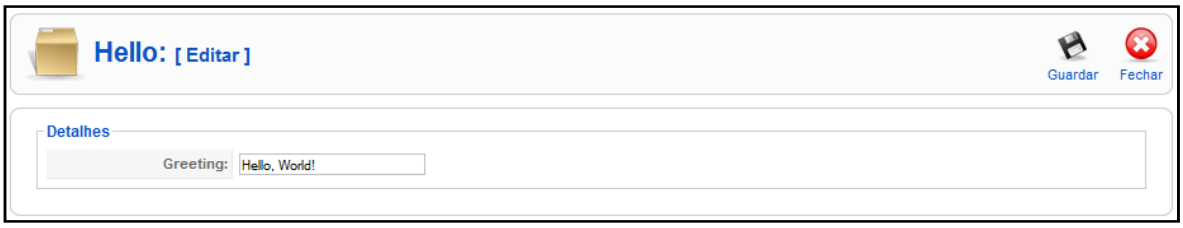


Figura 28 - *Layout* `form` correspondente à *view* `hello` do componente `com_hello`

Através das *views* `hellos`, `hello` e dos seus respectivos *layouts*, os utilizadores do lado *back end* do Joomla podem fazer toda a gestão das mensagens referentes ao componente `com_hello`.

2.4.3.9. *Back end* - criação das classes *controller*

O componente `com_hello` apresenta duas classes *controller*: `HellosController`, derivada da classe `JController` e implementada no ficheiro `admin/controller.php`; `HellosControllerHello`, derivada da classe `HellosController` e implementada no ficheiro `admin/controllers/hello.php`. A classe `HellosController` é a classe *controller* base do componente, responsável apenas por efectuar o carregamento da *view* adequada através do método `display()`. Esta classe apresenta o seguinte código:

```
<?php
/**
 * Hello World default controller
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license            GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

jimport('joomla.application.component.controller');

/**
 * Hello World Component Controller
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosController extends JController
{
        /**
         * Method to display the view
         *
         * @access     public
         */
        function display()
        {
                parent::display();
        }
      }
```

A classe `HellosControllerHello` é a classe *controller* adicional que irá tratar das tarefas executadas através das classes *view*. Estas tarefas correspondem às acções dos utilizadores nas interfaces (nos *layouts* do componente) e que são identificadas nos dados POST dos pedidos efectuados pelo cliente através da variável `task`. Por exemplo, se um pedido apresentar o URL `index.php?option=com_hello&controller=hello&task=edit&cid[]=1` significa que será executada a tarefa `edit` (`task=edit`) que corresponde à execução do método com o mesmo nome (`edit()`) no *controller* `hello` (`controller=hello`) do componente `com_hello` (`option=com_hello`).

O código implementado nesta classe é o seguinte:

```
<?php
/**
 * Hello Controller for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_4
 * @license           GNU/GPL
 */

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

/**
 * Hello Hello Controller
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosControllerHello extends HellosController
{
	/**
	 * constructor (registers additional tasks to methods)
	 * @return void
	 */
	function __construct()
	{
		parent::__construct();

		// Register Extra tasks
		$this->registerTask( 'add'  ,	'edit' );
	}

	/**
	 * display the edit form
	 * @return void
	 */
	function edit()
	{
		JRequest::setVar( 'view', 'hello' );
		JRequest::setVar( 'layout', 'form'  );
		JRequest::setVar('hidemainmenu', 1);

		parent::display();
	}

	/**
	 * save a record (and redirect to main page)
	 * @return void
	 */
	function save()
	{
		$model = $this->getModel('hello');

		if ($model->store()) {
			$msg = JText::_( 'Greeting Saved!' );
		} else {
			$msg = JText::_( 'Error Saving Greeting' );
		}
```

```
                // Check the table in so it can be edited.... we are done with it anyway
                $link = 'index.php?option=com_hello';
                $this->setRedirect($link, $msg);
        }

        /**
         * remove record(s)
         * @return void
         */
        function remove()
        {
                $model = $this->getModel('hello');
                if(!$model->delete()) {
                        $msg = JText::_( 'Error: One or More Greetings Could not be Deleted'
);
                } else {
                        $msg = JText::_( 'Greeting(s) Deleted' );
                }

                $this->setRedirect( 'index.php?option=com_hello', $msg );
        }

        /**
         * cancel editing a record
         * @return void
         */
        function cancel()
        {
                $msg = JText::_( 'Operation Cancelled' );
                $this->setRedirect( 'index.php?option=com_hello', $msg );
        }
}
```

No componente `com_hello` são permitidas as tarefas adicionar, editar e remover. Adicionar e editar são consideradas a mesma tarefa, uma vez que o formulário apresentado ao utilizador é o mesmo. A única diferença é que, no caso de se estar a adicionar um registo, os campos do formulário aparecem vazios. Assim, no método construtor da classe `HellosControllerHello`, a tarefa `add` é mapeada no apontador `edit` através do método `$this->registerTask( 'add', 'edit' )`. O primeiro parâmetro deste método corresponde à tarefa a mapear e o segundo ao método no qual a tarefa é mapeada.

O método `edit()` é bastante simples, apenas especifica a *view* e o *layout* a executar. A *view* será a `hello` e o *layout* o `form`. Neste método é desactivado o menu principal da interface de administração do Joomla, para prevenir que o utilizador navegue para outras interfaces sem guardar as alterações ou cancelar correctamente a operação de editar ou adicionar do componente. Para atribuir os valores às variáveis do pedido, é utilizado o método `JRequest::setVar()`, em que o primeiro parâmetro é o nome da variável e o segundo parâmetro o valor atribuído.

O método `save()` do *controller* é executado sempre que a tarefa contida no pedido e que corresponde ao valor da variável `task` seja `save`. Neste método é obtido o objecto *model* e invocado o seu método `store()`, responsável por guardar um registo na tabela `#__hello` da base de dados. O método `store()` já foi implementado no ficheiro `admin/model/hello.php`, durante a criação da classe `HellosModelHello` responsável pela manipulação da base de dados. Após a execução é feito o redireccionamento para a página principal do componente.

Para eliminar registos da tabela utilizada pelo componente, foi criado na classe *controller* o método `remove()` que é executado quando no pedido o valor da variável `task` é `remove`.

Através do objecto da classe *model* é invocado o método `delete()` que elimina os registos da tabela #__hello, cujo identificador é passado através do pedido no *array* `cid`.

Nesta classe *controller* foi também criado um método `cancel()` que redirecciona a execução do componente para a sua página principal. A página principal do componente corresponde ao *layout* `default` da *view* `hellos`.

2.4.3.10. Criação do *script* de instalação SQL

Nos componentes MVC para o CMS Joomla é bastante comum a utilização da base de dados desta plataforma para albergar novas tabelas que permitam a manipulação dos dados próprios do componente. Assim, o processo de instalação de novas extensões suporta a execução de consultas realizadas em código SQL e que são escritas e guardadas num ficheiro de texto. Este ficheiro é normalmente colocado no directório `admin/sql` do componente e tem como nome `install.mysql.utf8.sql`. Este nome significa que é um ficheiro de instalação, tem código compatível com o servidor de base de dados MySQL e codificação UTF-8.

O ficheiro SQL de instalação do componente `com_hello` apresenta o seguinte código:

```
DROP TABLE IF EXISTS `#__hello`;

CREATE TABLE `#__hello` (
  `id` int(11) NOT NULL auto_increment,
  `greeting` varchar(25) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=0 DEFAULT CHARSET=utf8;

INSERT INTO `#__hello` (`greeting`) VALUES ('Hello, World!'), ('Bonjour, Monde!'), ('Ciao,
Mondo!');
```

Como se pode verificar o ficheiro `install.mysql.utf8.sql` contém o código para a criação da tabela `#__hello` e para a inserção de alguns registos. Os nomes das tabelas utilizados nas consultas SQL devem começar com o prefixo "#__" (cardinal e dois *underscores*), uma vez que a plataforma Joomla substituirá este prefixo durante a instalação do componente, pelo prefixo utilizado nas tabelas da sua base de dados. O prefixo padrão utilizado nas tabelas do Joomla é "jos_", neste caso a tabela do componente será criada com o nome `jos_hello`. O facto de poder escolher o prefixo na instalação do Joomla permite a existência de várias instalações da plataforma na mesma base de dados sem que ocorram conflitos com outras aplicações. Justamente para evitar conflitos com os nomes das tabelas, na criação de extensões para o Joomla deve-se utilizar a convenção `#__nomedaextensão_nomedatabela`.

Para a tabela utilizada no componente `com_hello` foram criados dois campos: o campo `id` que será a chave primária e que corresponde ao identificador único do registo; o campo `greeting` responsável por guardar o corpo da mensagem. No *script* foi também introduzido o código para a introdução de algumas mensagens na tabela.

No processo de desinstalação do componente devem ser removidas as suas tabelas. Para isso, é criado o ficheiro `uninstall.mysql.utf8.sql` que contém o *script* necessário à remoção das tabelas da base de dados. No caso do componente `com_hello`, como só existe uma, o ficheiro contém apenas o código: `DROP TABLE IF EXISTS `#__hello`;`.

2.4.3.11. Criação do ficheiro de instalação XML

O ficheiro XML de instalação vai conter todos os detalhes da instalação do componente. Este ficheiro é conhecido por *XML manifest file*, qualquer erro que contenha, pode resultar na falha da instalação ou na instalação parcial do componente. O ficheiro deve ser guardado com a codificação UTF-8. Para a instalação do componente `com_hello` foi criado o ficheiro `install.xml` com o seguinte conteúdo:

```
<?xml version="1.0" encoding="utf-8"?>
<install type="component" version="1.5.0">
	<name>Hello</name>
	<!-- The following elements are optional and free of formatting conttraints -->
	<creationDate>2007-02-22</creationDate>
	<author>John Doe</author>
	<authorEmail>john.doe@example.org</authorEmail>
	<authorUrl>http://www.example.org</authorUrl>
	<copyright>Copyright Info</copyright>
	<license>License Info</license>
	<!--  The version string is recorded in the components table -->
	<version>4.01</version>
	<!-- The description is optional and defaults to the name -->
	<description>Description of the component ...</description>

	<!-- Site Main File Copy Section -->
	<files folder="site">
		<filename>controller.php</filename>
		<filename>hello.php</filename>
		<filename>index.html</filename>
		<filename>models/hello.php</filename>
		<filename>views/hello/index.html</filename>
		<filename>views/hello/view.html.php</filename>
		<filename>views/hello/tmpl/index.html</filename>
		<filename>views/hello/tmpl/default.php</filename>
		<filename>views/index.html</filename>
	</files>

	<install>
		<sql folder="admin/sql">
			<file driver="mysql" charset="utf8">install.mysql.utf8.sql</file>
		</sql>
	</install>
	<uninstall>
		<sql  folder="admin/sql">
			<file driver="mysql" charset="utf8">uninstall.mysql.utf8.sql</file>
		</sql>
	</uninstall>

	<administration>
		<!-- Administration Menu Section -->
		<menu>Hello World!</menu>

		<!-- Administration Main File Copy Section -->
		<!-- Note the folder attribute: This attribute describes the folder
			to copy FROM in the package to install therefore files copied
			in this section are copied from /admin/ in the package -->
		<files folder="admin">
			<!-- Site Main File Copy Section -->
			<filename>hello.php</filename>
			<filename>index.html</filename>
			<filename>sql/install.mysql.utf8.sql</filename>
			<filename>sql/uninstall.mysql.utf8.sql</filename>
			<filename>controller.php</filename>
			<filename>controllers/hello.php</filename>
			<filename>controllers/index.html</filename>
			<filename>models/hellos.php</filename>
			<filename>models/hello.php</filename>
			<filename>models/index.html</filename>
			<filename>tables/hello.php</filename>
			<filename>tables/index.html</filename>
			<filename>views/hello/view.html.php</filename>
			<filename>views/hello/tmpl/form.php</filename>
			<filename>views/hello/index.html</filename>
			<filename>views/hello/tmpl/index.html</filename>
```

```
                <filename>views/hellos/view.html.php</filename>
                <filename>views/hellos/index.html</filename>
                <filename>views/hellos/tmpl/default.php</filename>
                <filename>views/hellos/tmpl/index.html</filename>
            </files>
    </administration>
</install>
```

No início do ficheiro devem ser colocadas as informações relativas ao componente, tais como: nome do componente, versão, data de criação, dados do autor, licença, direitos de cópia e uma pequena descrição. Estas informações são colocadas dentro das *tags* `name`, `creationDate`, `author`, `authorEmail`, `authorUrl`, `copyright`, `license`, `version`, `description`.

Para a criação e remoção das tabelas na base de dados devem ser identificados os *scripts* necessários através da *tag* `install` com o parâmetro `sql`.

Todos os ficheiros necessários à instalação e funcionamento do componente devem estar identificados dentro da *tag* `filename`. No caso dos ficheiros do *back end*, estas *tags* devem estar localizadas dentro da *tag* `files` com o parâmetro `folder="admin"`. Todos os detalhes relativos ao funcionamento do *back end* devem estar dentro da *tag* `administration`.

Para os ficheiros de *front end* devem ser colocadas dentro da *tag* file com o parâmetro `folder="site"`.

2.5. Resumo

Actualmente a facilidade de acesso às Tecnologias da Informação e Comunicação tem potenciado o aumento do volume de documentos em formato digital existentes nas organizações. Hoje em dia, este tipo de documentos desempenha um papel muito importante para o sucesso da sua actividade, sendo portanto, uma boa prática a implementação de sistemas capazes de realizar uma gestão eficaz desses documentos. Estes sistemas são normalmente conhecidos por *Document Management Systems* (DMS), *Electronic Document Management Systems* (EDMS) ou simplesmente Gestão Electrónica de Documentos (GED) e representam um papel fundamental na melhoria dos serviços prestados pelas organizações.

Nesta era da informação digital existe um tipo de sistemas que proporciona a criação, organização e partilha rápida de informação, através da utilização das tecnologias associadas à Internet. São sistemas capazes de fazer a gestão de conteúdos, denominados de *Content Management Systems* (CMS) e que actualmente são cada vez mais utilizados pelas organizações. Estes sistemas têm a grande vantagem de proporcionar a separação do processo de desenvolvimento dos conteúdos do processo de desenvolvimento de mecanismos que permitem a sua apresentação. Um dos sistemas CMS mais utilizados actualmente é a plataforma Joomla. Este sistema tem a capacidade de tornar a criação de um *site* tão simples como o processamento de documentos para impressão.

Os Sistemas DMS e CMS apresentam semelhanças na sua arquitectura de funcionamento, pelo que o cruzamento destes dois sistemas pode constituir numa vantagem para as organizações.

Uma das grandes potencialidades do CMS Joomla é o facto de possuir uma *framework* capaz de proporcionar o desenvolvimento rápido de extensões. Através do padrão de programação MVC, é possível a criação de extensões complexas, denominadas de componentes e que podem constituir-se numa aplicação completa e repleta de funcionalidades.

A utilização do padrão MVC permite a separação do desenho do *software* em três partes funcionalmente diferentes: acesso aos dados, apresentação dos dados e a lógica de funcionamento, permitindo assim que seja possível alterar aspectos relativos ao modo como os dados são apresentados sem ter que reprogramar toda a aplicação.

3. Desenvolvimento do componente SGDEscolas

O sistema gestor de conteúdos Joomla é hoje em dia uma plataforma que beneficia de inúmeras extensões gratuitas e comerciais. Já existem extensões que se podem utilizar para os mais variadíssimos fins, no entanto, são poucas as extensões especificamente direccionadas para a gestão escolar ou para as diferentes actividades que garantem o funcionamento das escolas. Uma das mais importantes para o correcto funcionamento destas organizações diz respeito à gestão de documentos. As escolas estão repletas de documentos, quer em formato papel, quer em formato digital. Todos os dias os professores e outros agentes criam novos documentos e não são utilizados sistemas capazes de facilitar a sua gestão. Visto isto, aproveitando as potencialidades do CMS Joomla, pretende-se, neste capítulo e com o estudo já efectuado sobre os sistemas de gestão de documental, sistemas gestores de conteúdos, CMS Joomla e desenvolvimento de componentes para o Joomla, estudar e demonstrar a criação de um componente que possa ser uma ferramenta essencial para as escolas na gestão dos seus documentos. A criação deste componente terá como prioridade satisfazer as necessidades relativas à criação e organização de documentos produzidos pelos professores, no âmbito da sua actividade profissional. Futuramente, o componente aqui desenvolvido terá as condições necessárias para poder evoluir e tornar-se numa ferramenta poderosa para a gestão escolar.

3.1. Análise e requisitos

A extensão a desenvolver tem características de uma aplicação independente, no entanto, como será totalmente integrada no sistema Joomla, deve assumir a forma mais complexa de extensões Joomla, ou seja, um componente. De forma a garantir a total integração com o CMS, no desenvolvimento deste componente devem ser seguidas as normas e convenções requeridas no desenvolvimento deste tipo de extensões. O desenvolvimento do componente seguirá então o padrão *Model-View-Controller* e utilizará as bibliotecas da API do Joomla e que constituem a sua *framework*.

3.1.1. Preparação da estação de trabalho

No desenvolvimento dos componentes devem ser garantidas as condições necessárias ao nível da estação de trabalho. Assim, foi preparada uma máquina com as seguintes condições:

- Servidor local com o pacote XAMPP 1.7.3.
  - PHP 5.3.1.
  - MySQL 5.1.41.
  - Apache 2.2.14.
- *Framework* Joomla contida no pacote do Joomla 1.5.15 pt-pt.
- Eclipse e Notepad++ para a criação e edição de ficheiros código .php, .html, .xml e .sql.

Todo o *software* utilizado encontra-se de acordo com a GNU *General Public License*, não trazendo quaisquer custos para o desenvolvimento do componente.

Para facilitar o desenvolvimento, no PHP foi activada a opção que permite mostrar os erros e a extensão xdebug para o *debug*.

3.1.2. Requisitos do componente

O componente aqui desenvolvido tem como objectivo primordial satisfazer as principais necessidades ao nível da gestão de documentos, criados por professores de uma organização escolar do ensino básico e secundário. Para facilitar a sua aceitação por parte dos utilizadores, o sistema deve possuir interfaces simples e intuitivas que não obriguem o preenchimento de demasiados parâmetros.

Os documentos podem ser introduzidos no sistema por diversos utilizadores. Na sua introdução, o utilizador deve obrigatoriamente atribuir um nome, uma categoria, o ano lectivo. Podem também ser associados ao documento, um ficheiro que represente o próprio documento, uma descrição e palavras-chave que poderão ser utilizadas para facilitar a procura. Aos utilizadores deve ser permitido publicar os seus documentos, de modo que todos os utilizadores os possam consultar. O sistema deve ainda registar as datas de criação, edição e publicação de cada documento.

Nas escolas existem obrigatoriamente diversos tipos de dossiers, nos quais devem ser inseridos determinados documentos. Assim, o sistema deverá permitir a criação de dossiers e a associação de documentos a eles.

Os utilizadores do componente serão os professores da escola onde será implementado. Assim, para além das informações relativas aos dados pessoais, devem ser colhidas informações que identifiquem a sua situação profissional, tais como: número de funcionário, grupo de recrutamento, categoria profissional e cargos ou outras situações. A plataforma Joomla já permite o registo e criação de utilizadores, no entanto, é necessário fazer a distinção entre um utilizador do CMS Joomla e um utilizador do componente criado para a gestão de documentos da escola. Esta distinção é importante, uma vez que um utilizador do CMS Joomla não deve ser obrigatoriamente um utilizador no sistema de gestão de documentos aqui desenvolvido. Assim, na extensão desenvolvida deve ser possível criar os seus próprios utilizadores com base em utilizadores já registados no CMS Joomla.

Ao longo do ano lectivo, existem momentos em que os professores devem entregar à direcção da escola determinados documentos. O sistema deverá então possibilitar a criação de um processo de registo e entrega oficial, em que seja possível a identificação do estado de aprovação ou da necessidade de revisão dos documentos entregues.

Com o intuito de possibilitar o trabalho colaborativo, os documentos podem ser partilhados por diversos utilizadores, permitindo a edição ou apenas a sua visualização. Para facilitar este trabalho deverá existir um processo que notifique os utilizadores das alterações efectuadas aos seus documentos.

Como qualquer aplicação de gestão de informação é crucial que no componente aqui desenvolvido seja possível a criação de cópias de segurança. No âmbito da gestão de documentos em organizações escolares, deve existir um processo de obter todos os documentos do sistema ou documentos que se enquadrem em determinados critérios. Por exemplo, deve ser possível

retirar, de uma só vez, todas as actas relativas ao ano lectivo em curso, todos os relatórios de coordenação, todos os documentos de determinado professor, etc.

Existem inúmeras funcionalidades que podem ser implementadas num sistema gestor de documentos. No entanto, no âmbito desta dissertação pretende-se essencialmente criar um componente de fácil utilização e implementação. Este componente será um caso de estudo da criação de extensões do tipo componente para a plataforma Joomla e constituirá uma ferramenta prática e útil na gestão de documentos nas escolas.

Como é comum na maioria dos componentes, o sistema desenvolvido terá um *back end* para administração e um *front end* voltado para o utilizador comum.

3.2. Desenho do componente

A integração do componente com o CMS Joomla é feita através da camada superior do sistema e o seu desenvolvimento deverá seguir o padrão de programação MVC. Atendendo à convenção de nomes utilizada para identificar os componentes desenvolvidos para o Joomla, o componente será denominado de `com_sgdescolas`.

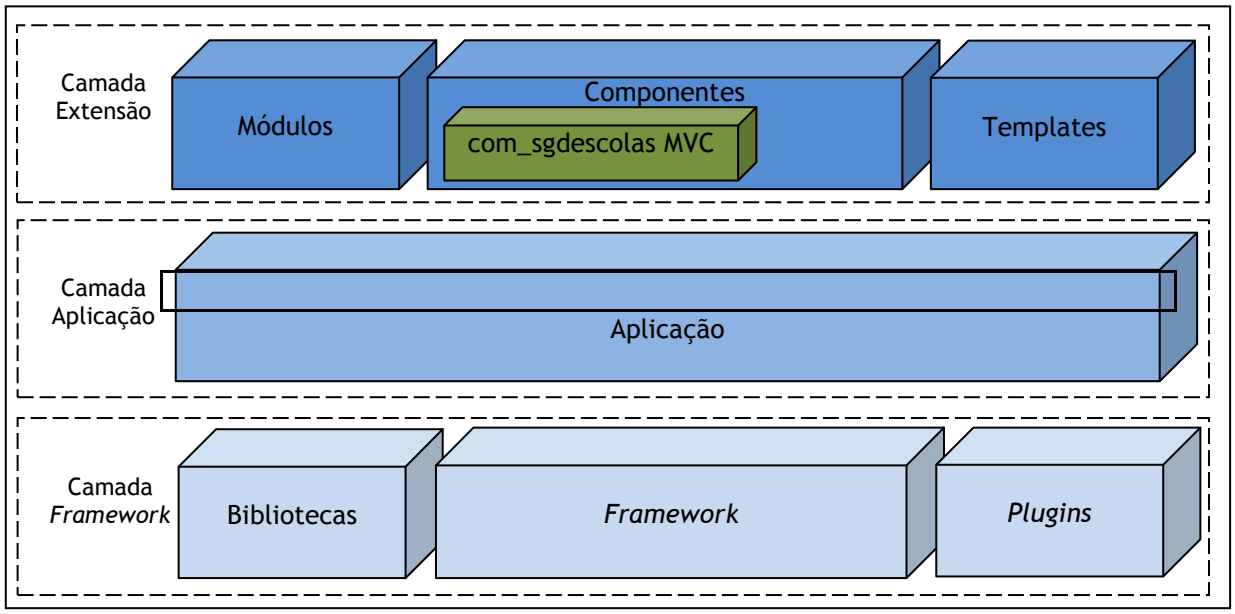


Figura 29 - Componente com_sgdescolas no sistema Joomla

Na figura pode-se observar a localização do componente `com_sgdescolas` no sistema Joomla. É executado sobre a aplicação do Joomla, pode usufruir da sua *framework* e pode inclusive utilizar módulos, templates ou *plugins* já existentes.

Para realizar a gestão da informação referente aos documentos e utilizadores geridos pelo componente, é utilizada a base de dados do Joomla, onde serão alojadas as tabelas necessárias. Através da criação de classes *model*, o acesso aos dados é realizado de forma simples e segura. As interfaces do componente são criadas com base nas classes *view* e nos seus *layouts*. Todas as acções realizadas pelos utilizadores sobre estas interfaces serão controladas pelas classes *controller*. As classes *model*, *view* e *controller* necessárias ao funcionamento do componente

derivam das classes `JModel`, `JView` e `JController` já disponíveis através da *framework* do sistema Joomla.

3.2.1. Diagramas *use case*

O diagrama apresentado na figura seguinte, demonstra em termos genéricos as funções realizadas pelos actores administrador e utilizador no sistema.

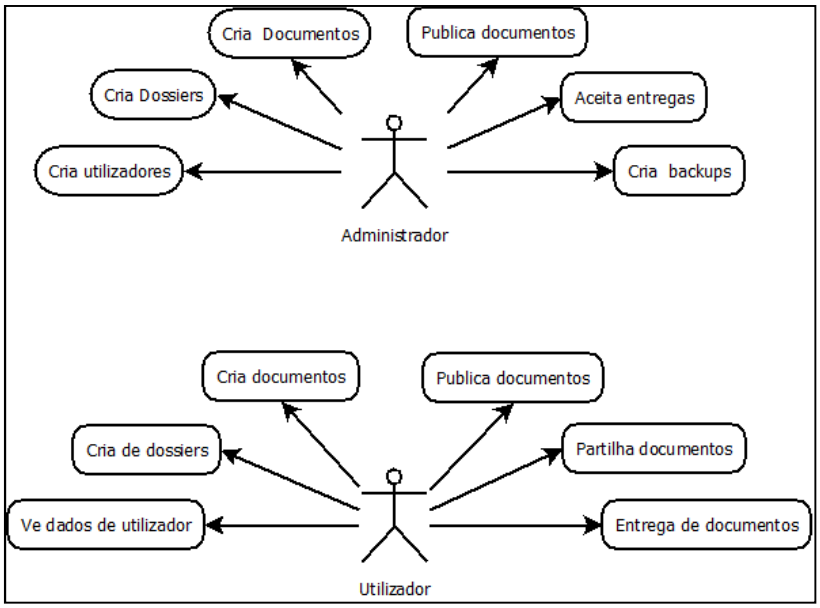


Figura 30 - Diagramas de *Use case*.

Como se pode observar na figura, o administrador pode efectuar a criação de utilizadores, criação de dossiers, criação de documentos, publicação de documentos, aceitação de entregas e criação de *backups*. O utilizador pode ver os seus dados, criar dossiers, criar , publicar , partilhar e entregar documentos à Direcção. Na aplicação, para além das actividades apresentadas nos diagramas, serão também criadas as funcionalidades que normalmente se implementam em conjunto com essas actividades, tais como editar, eliminar documentos, etc.

3.2.2. Modelo Entidade-Relação

O componente `com_sgdescolas` necessita, para guardar as informações necessárias ao seu funcionamento, de utilizar tabelas próprias na base de dados do Joomla. Para a implementação das funcionalidades de sistema de gestão de documentos do componente, foram identificadas as seguintes entidades:

- **Utilizadores** – professores que utilizam o sistema.
- **Grupo de recrutamento** – áreas em que um professor pode leccionar.
- **Categoria profissional** – situações profissionais do professor.
- **Cargo** – cargos do professor na escola.
- **Documentos** – documentos introduzidos pelos professores.
- **Categoria documentos** - vários tipos de documentos;

- **Estado do documento** - estados em que os documentos se podem encontrar no sistema.
- **Dossier** - dossiers existentes na escola.
- **Tipo dossier** - vários tipos de dossiers.
- **Partilha** – repositório de documentos que estão partilhados.
- **Entrega Documentos** – representa as entregas oficiais de documentos.
- **Estados de entrega** – representa os diferentes estados relativos às entregas oficiais de documentos.
- **Tipo de estado de entrega** – representa os diversos tipos de estados relativos às entregas.

As entidades que se acabaram de identificar devem relacionar-se entre si, de modo a permitir a correcta organização da informação e o correcto funcionamento do sistema. O modelo Entidade-Relação apresentado na figura seguinte ilustra essas relações.

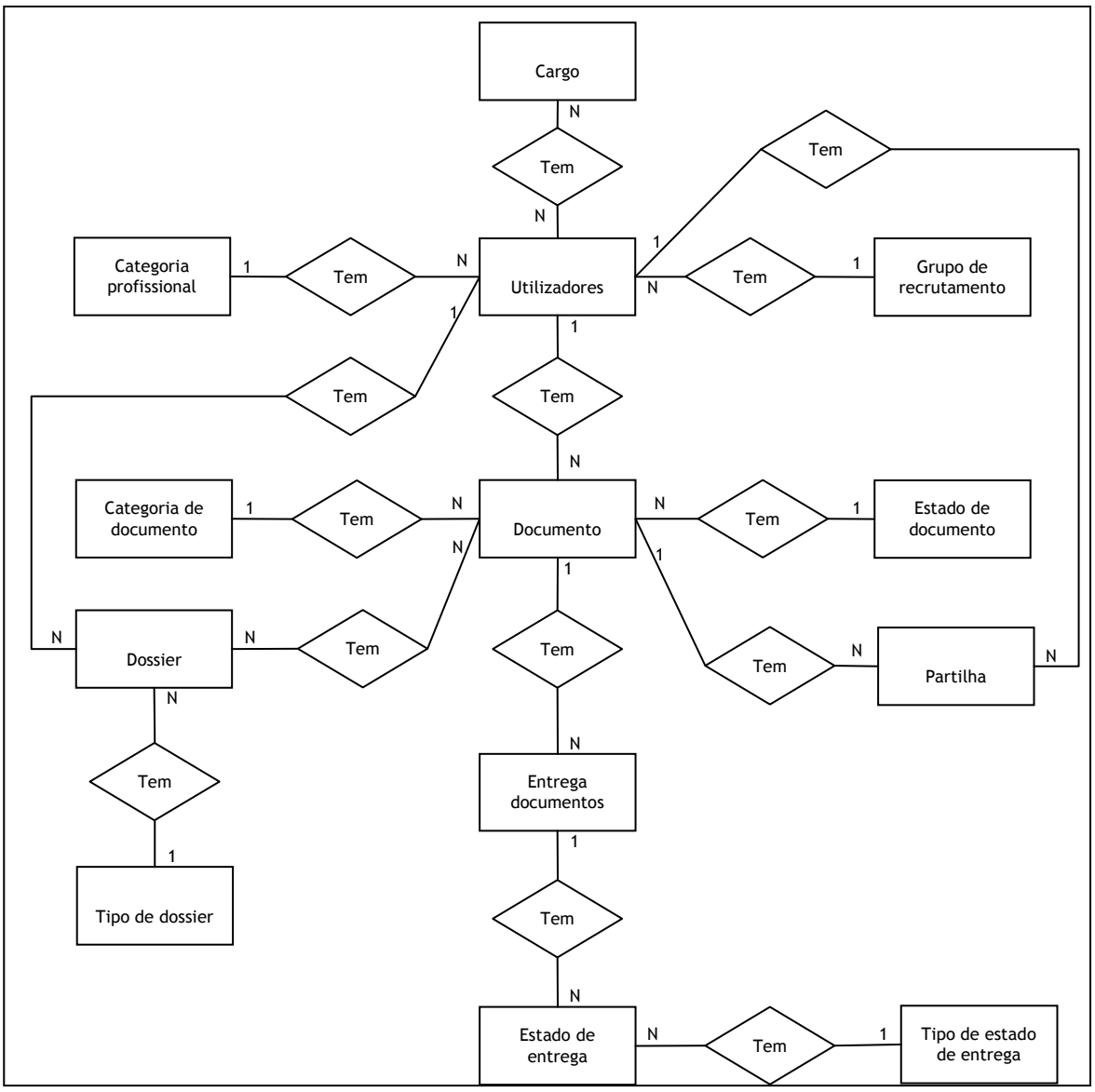


Figura 31 - Modelo Entidade-Relação do componente com_sgdescolas

Através da análise do modelo Entidade-Relação pode-se verificar que os utilizadores podem ter vários cargos e um mesmo cargo pode ser atribuído a vários utilizadores. Cada utilizador tem apenas uma categoria profissional, podendo haver vários utilizadores nessa mesma categoria. Os diversos utilizadores têm também um grupo de recrutamento. Cada utilizador do sistema pode ter vários documentos e vários dossiers. Os dossiers têm um tipo de dossier.

Relativamente aos documentos pode-se verificar que estes podem ter uma categoria e um estado. Os documentos podem ainda estar em diversos dossiers e em diversas partilhas. Cada partilha pode ter vários utilizadores. Um documento pode ainda ter várias entregas, cada entrega pode ter vários estados de entrega e cada estado de entrega tem um tipo de estado.

No anexo 1 pode ser consultada a representação do modelo físico, criado com base no modelo Entidade-Relação aqui apresentado.

3.3. Desenvolvimento do *back end*

O *back end* do componente `com_sgdescolas` corresponde à parte da aplicação que se executa no lado da administração do CMS Joomla. Como é comum em extensões do tipo componente, após a instalação é possível aceder ao *back end* através do menu `Components` que se encontra na interface de administração do Joomla.

Tendo em conta os requisitos da aplicação, através das interfaces de administração do componente o utilizador poderá realizar as tarefas de gestão de utilizadores, de documentos, de dossiers, de entregas e *backup* de documentos. Para facilitar a utilização, a interface principal do componente (ponto de entrada da aplicação) consiste num painel de controlo que permite o acesso a todas as funcionalidades. No processo de instalação é também garantida a criação de submenus para o acesso às diferentes interfaces de gestão.

No desenvolvimento do *back end* foi necessário criar toda a estrutura MVC do componente, através da criação de várias classes *model*, *view*, *controller*. Foram ainda criadas as classes responsáveis pela instalação, criação de menus e botões do componente. Assim, tendo em conta o padrão MVC e a estrutura de directórios base de um componente Joomla, no desenvolvimento do *back end* que se situa no directório `com_sgdescolas/admin` do componente, foi criada a seguinte estrutura de directórios representada na figura seguinte.

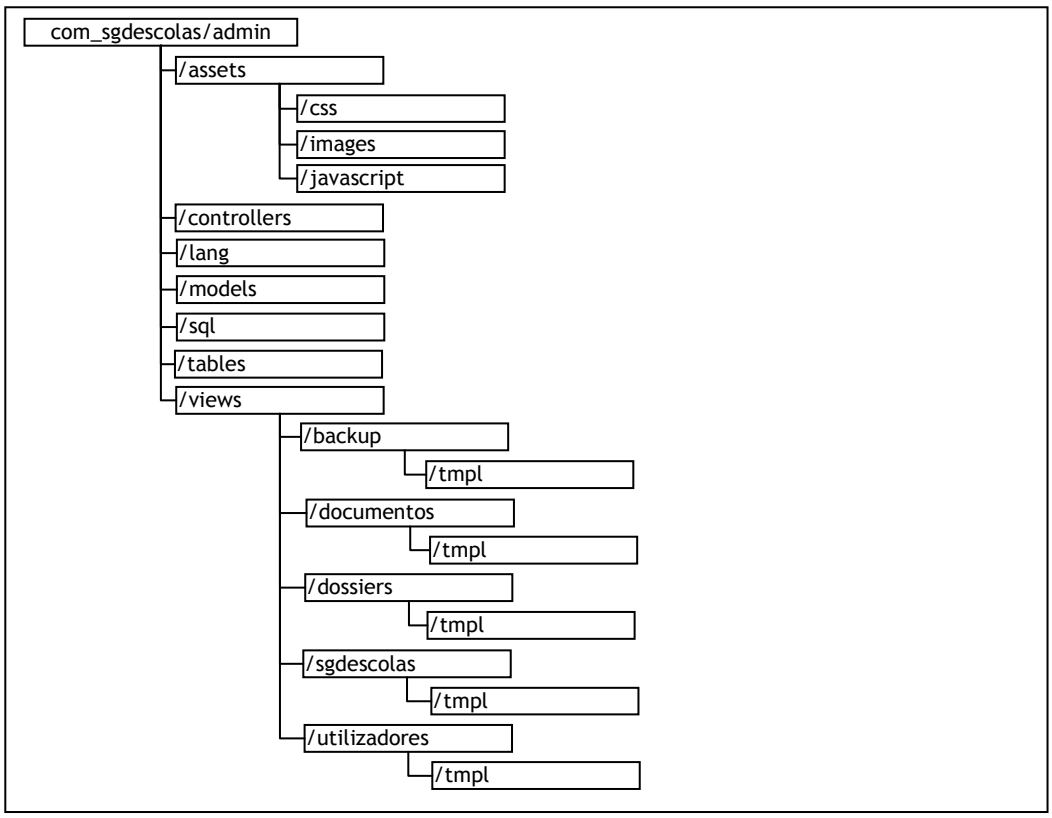


Figura 32 – Estrutura de directórios do *back end* do componente `com_sgdescolas`.

Como se pode observar na figura, o directório `admin` do componente contém os directórios `assets`, `controllers`, `lang`, `models`, `sql`, `tables` e `views`. O directório `assets` contém os directórios `css`, `images` e `javascript`, onde se encontram os ficheiros de folhas de estilo, imagens e *javascript*, respectivamente. Estes ficheiros servem para apoiar a construção das interfaces. O directório `controller` contém os ficheiros onde estão implementadas as classes derivadas da classe `JController` da *framework* do Joomla, responsáveis por controlar as acções dos utilizadores. O Directório lang contém o ficheiro de tradução ou personalização de mensagens. O directório `models` contém os ficheiros onde estão implementadas as classes derivadas da classe `JModel` da *framework,* responsáveis pelo acesso às tabelas do componente, que foram criadas na base de dados do Joomla durante o processo de instalação. O directório `sql` contém um *script* em linguagem SQL para a criação das tabelas necessárias ao funcionamento do componente e um *script* para eliminar essas mesmas tabelas, se o componente for removido do Joomla a partir do gestor de extensões. O directório `tables` contém os ficheiros onde foram implementadas as classes derivadas da classe `JTable` da *framework,* responsáveis pela criação dos objectos referentes às principais entidades do sistema. No directório `views` está criada toda uma estrutura onde estão inseridas as classes derivadas da classe `JView` da *framework,* responsáveis pela apresentação dos dados e os ficheiros onde foram criados os diferentes *layouts* que definem o modo como os dados da respectiva *view* são apresentados. Na raiz do directório `admin` encontram-se as implementações das classes responsáveis pela instalação do componente, pelas opções de configuração e pela construção das barras de ferramentas do componente.

No anexo 2 pode ser consultada toda a estrutura de directórios do componente.

3.3.1. Classes *Model*

As classes *model* são implementadas em ficheiros localizados no directório `models` do componente. Tendo em conta os requisitos do presente sistema de gestão de documentos e as entidades sobre as quais os utilizadores interagem directamente, foram implementados os ficheiros representados na seguinte figura.

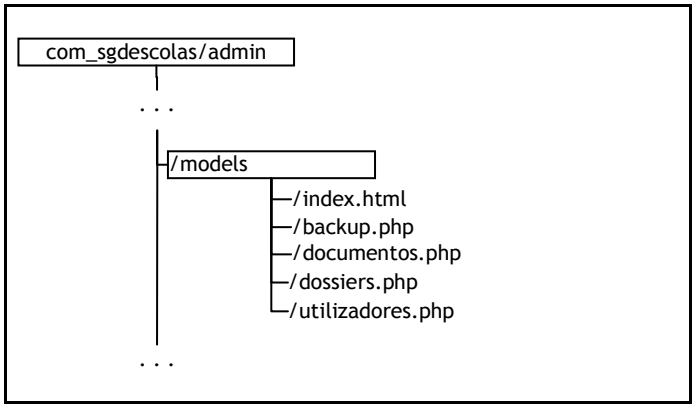


Figura 33 - Directório `models` do *back end*

O ficheiro `utilizadores.php` contém a implementação da classe `SgdescolasModelUtilizadores`. Esta classe contém os métodos necessários à gestão dos dados relativos aos utilizadores do sistema.

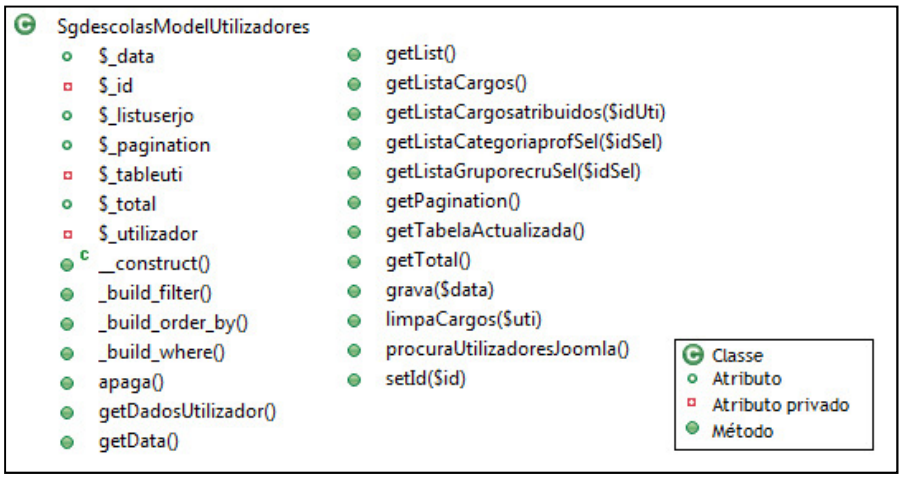


Figura 34 – Classe SgdescolasModelUtilizadores

Na figura está representada a classe `SgdescolasModelUtilizadores`. Os métodos implementados nesta classe permitem a comunicação com a base de dados para: listagem de todos os utilizadores; aplicação de filtros e paginação à lista de utilizadores; acesso aos dados dos utilizadores; gravar, editar e eliminar os dados dos utilizadores; acesso aos utilizadores registados na plataforma Joomla.

Tal como em todas as classes *model*, nesta classe é efectuada a comunicação com a base de dados do CMS Joomla. Esta comunicação entre as classes *model* do componente e a base de dados do Joomla pode-se observar no método responsável por devolver os dados de um utilizador:

```
function &getDadosUtilizador()
{
        $db = & JFactory::getDBO();
```

```
        if ($this->_utilizador) {
                return $this->_utilizador;
        }

        $query = ' SELECT a.id AS id_joomla, a.name, a.username, a.email, u.*'
                .' FROM #__users a, #__sgdescolas_utilizadores u'
                .' WHERE a.id=u.id_utilizador_joomla AND u.id=\''.$this->_id.'\'';

                 $db->setQuery( $query );
                 $this->_utilizador  = $db->loadObject();

        return $this->_utilizador;
}
```

Uma vez que a plataforma Joomla já se encontra em execução, existe uma ligação estabelecida através de um objecto global da base de dados. Para comunicar com a base de dados não é necessário estabelecer uma nova ligação, basta obter a referência a esse objecto recorrendo ao método `&JFactory::getDBO()`. Este método devolve um objecto do tipo `JDatabase`. Para realizar uma consulta à base de dados é utilizado o método `setQuery()` do objecto. Este método recebe como parâmetro uma *string* com a respectiva consulta. No método apresentado acima pode-se verificar este aspecto na utilização do código `$db->setQuery( $query )`. O método `loadObject()` carrega para o objecto os dados do primeiro registo que a consulta devolve, correspondendo, neste caso, aos dados do utilizador.

Sempre que for necessário obter uma lista de registos através da consulta à base de dados pode utilizar-se o método `loadObjectList()`, como demonstra o seguinte código:

```
…
$db->setQuery( $query );
$rows = $db->loadObjectList();
…
```

Quando numa consulta é necessário gerar mensagens em erro, recorre-se ao método `getError()` do objecto da base de dados que devolve um *string* com o erro proveniente da base de dados. Este método associado ao método `setError()` implementado na classe `JModel` da *framework*, permite a geração de mensagens de erro por parte da classe *model*.

```
…
if (!$db->query()) {
        $this->setError($db->getError());
        return false;
}
…
```

Como se pode verificar no código acima, através do método `setError()` é definida uma mensagem de erro para os objectos da classe *model* com base na mensagem de erro proveniente do objecto da base de dados.

Passando agora para o ficheiro `documentos.php`, que também se encontra no directório *models*, este consiste na implementação da classe `SgdescolasModelDocumentos`. Esta classe é responsável pela gestão dos dados, na base de dados, relativos aos documentos. Apresenta os atributos e métodos representados na figura seguinte.

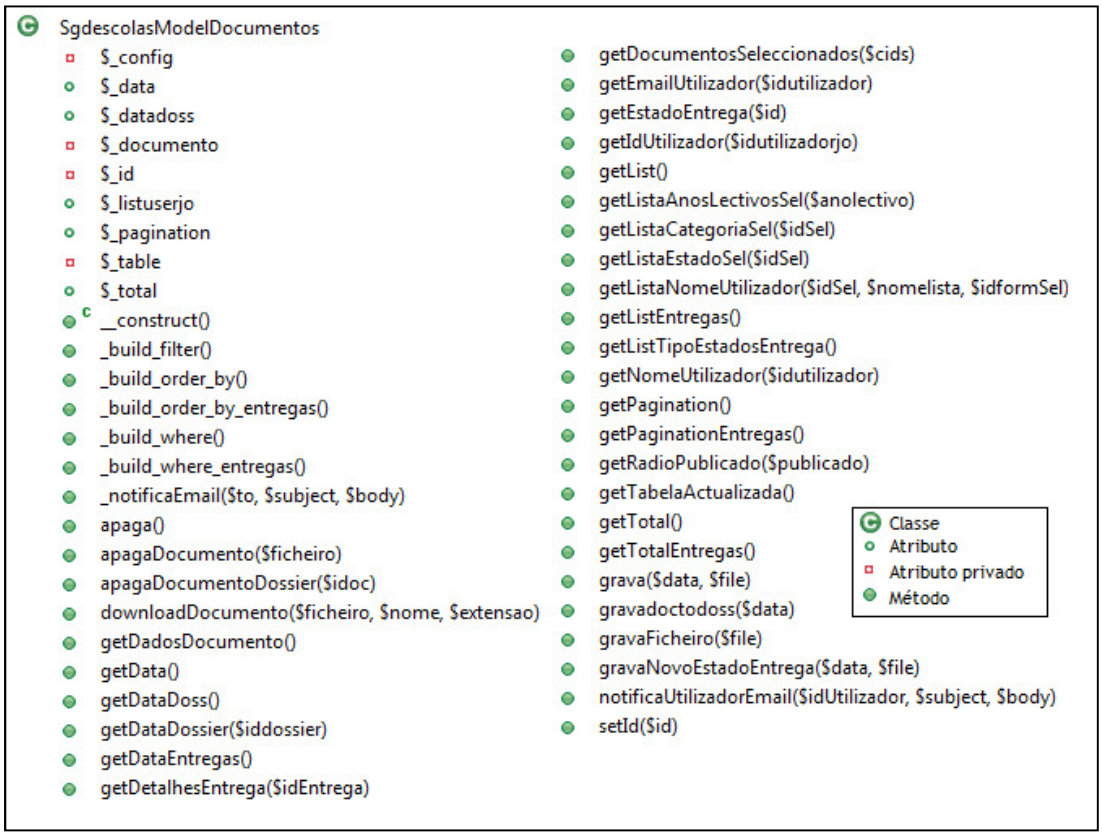


Figura 35 - Classe SgdescolasModelDocumentos

Como se pode observar na figura anterior, na classe `SgdescolasModelDocumentos` estão implementados os métodos que permitem: a listagem de todos os documentos; aplicação de filtros e paginação à lista de documentos; acesso aos dados dos documentos; gravar, editar e eliminar os dados dos documentos na base de dados e no directório específico no servidor; relação com os dossiers; *download* de documentos; gestão de entregas; notificações com base no *email* dos utilizadores e do administrador do CMS Joomla.

A comunicação com a base de dados do Joomla baseia-se nos mesmos métodos utilizados pela classe `SgdescolasModelUtilizadores`.

O ficheiro `dossiers.php` que se encontra no directório `models`, contém a implementação da classe `SgdescolasModelDossiers`. Os seus atributos e métodos encontram-se representados na figura seguinte.

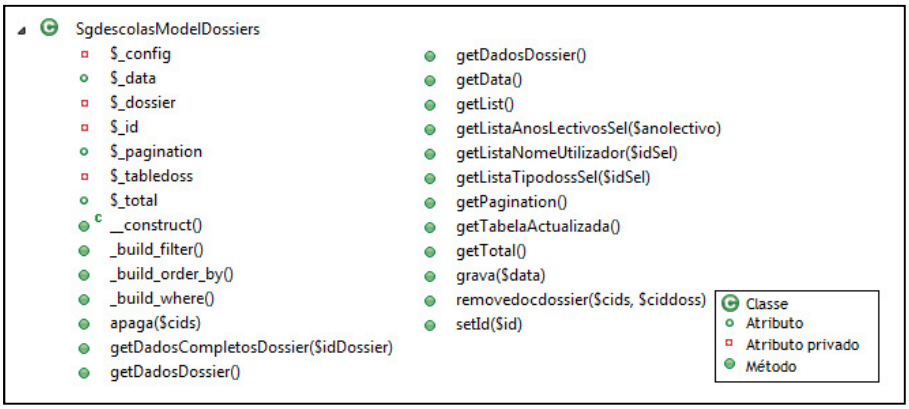


Figura 36- Classe SgdescolasModelDossiers

Através da implementação da classe `SgdescolasModelDossiers` é possível realizar a comunicação com a base de dados permitindo assim: obter a listagem de todos os dossiers; aplicação de filtros e paginação à lista de dossiers; acesso aos dados dos dossiers; gravar, editar e eliminar os dados dos dossiers; acesso aos utilizadores registados na plataforma Joomla.

Para permitir a criação de *backups* dos documentos existentes no sistema, foi criada a classe SgdescolasModelBackup que realiza os acessos à base de dados necessários à escolha dos documentos a incluir no *backup*.

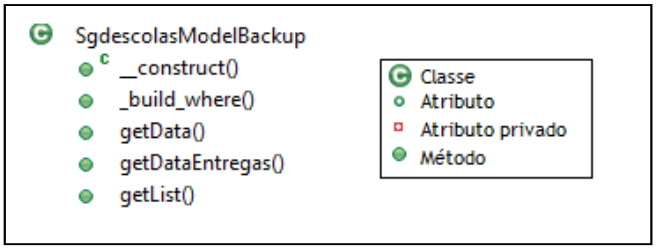


Figura 37 - Classe SgdescolasModelBackup

Através dos métodos implementados nesta classe é possível obter a lista de todos os documentos e aplicar filtros a essas listas.

3.3.2. Classes *View* e *layouts*

As classes *view* são implementadas em ficheiros localizados no directório `views` do componente. Estas classes são responsáveis pela apresentação dos dados nas interfaces da aplicação. Na figura seguinte está representada toda a estrutura criada neste directório.

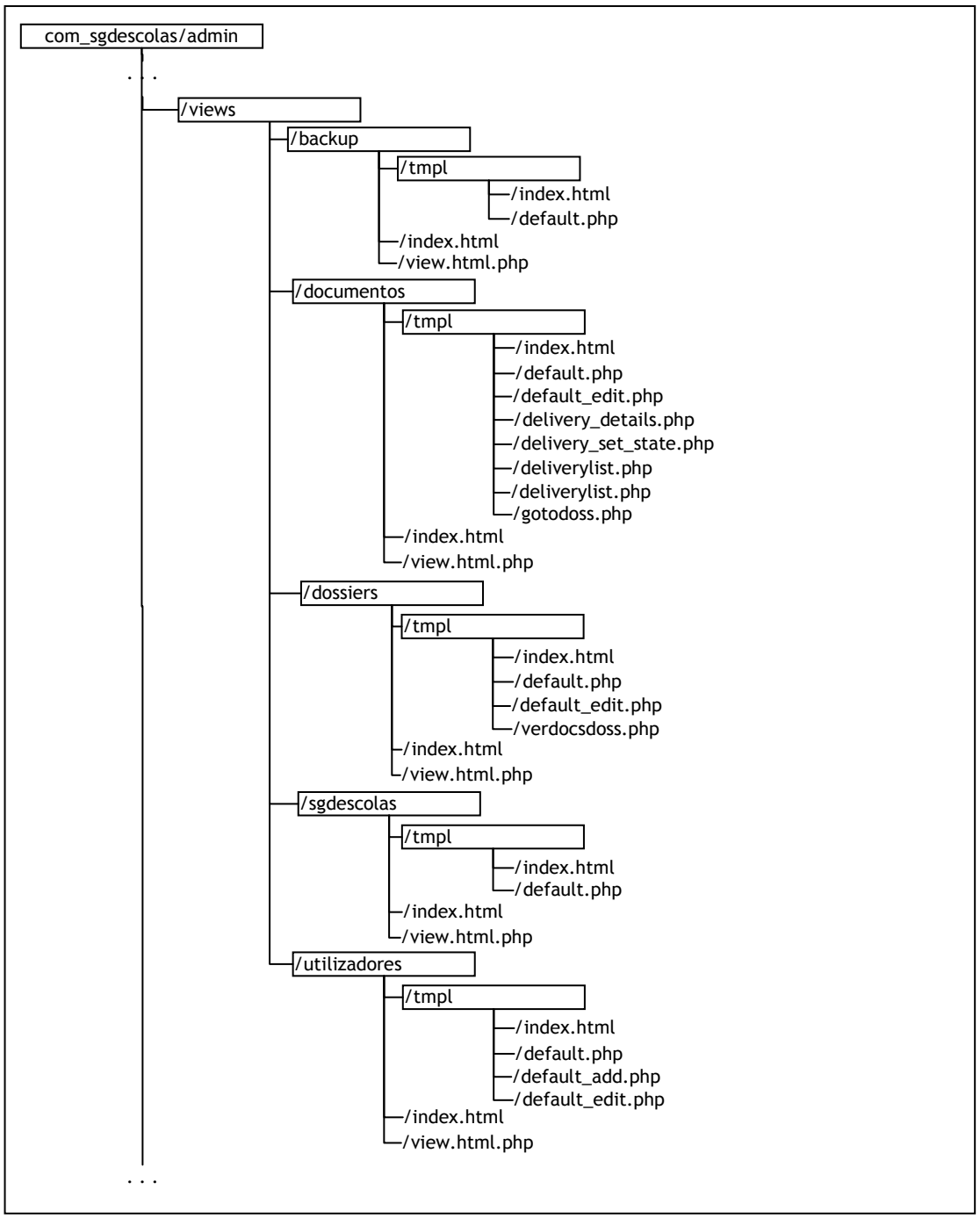


Figura 38 – Directório models do *back end*.

O modo como os dados são organizados e representados nessas interfaces depende dos seus *layouts*. Deste modo, cada *view* deve estar organizada dentro de um directório com o nome da *view* e os diferentes *layouts* devem estar contidos num subdirectório da própria *view*. Como os *layouts* criados neste componente são responsáveis por organizar a informação em documentos HTML, devem ser inseridos num subdirectório com o nome `tmpl`. Ao *layout* que a *view* deve carregar por defeito, deve ser dado o nome `default`. Assim a *view* funcionará correctamente, mesmo que no pedido não seja identificado o *layout* a apresentar. A implementação da classe *view* deve ser feita num ficheiro de nome `view.html.php`. É importante seguir esta estrutura, uma vez que corresponde ao padrão de desenvolvimento de componentes MVC para o Joomla, garantindo assim a correcta utilização da *framework*.

No componente foram desenvolvidas as *views* sgdescolas, utilizadores, documentos, dossiers e *backups*. As classes responsáveis por estas *views* são derivadas da classe `JView` da *framework* e denominam-se de `SgdescolasViewSgdescolas`, `SgdescolasViewUtilizadores`, `SgdescolasViewDocumentos`, `SgdescolasViewDossiers` e `SgdescolasViewBackup`.

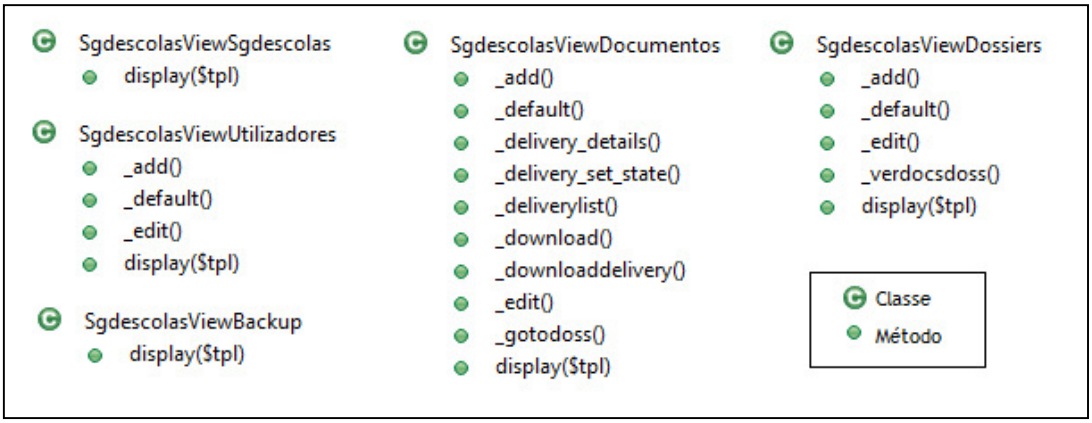


Figura 39 - Classes *view* do *back end*.

A figura acima apresenta as classes *view* e os respectivos métodos.

A *view* sgdescolas corresponde à *view* apresentada no ponto de entrada do sistema e apresenta o painel de controlo do componente. Esta *view* não necessita de obter dados da base de dados e por isso não existe uma *model* com o mesmo nome.

As restantes *views* são responsáveis por enviar os dados para as interfaces de gestão de utilizadores, documentos, dossiers e *backup*. Através das classes *model* com o mesmo das *views*, estas classes recebem os dados da base de dados e prepara-los para serem utilizados e organizados nos respectivos *layouts*. O seguinte código demonstra como uma *view* recebe os dados da respectiva *model* e os prepara para o respectivo *layout*.

```
function _default()
{
        global $option;

        // Referência para  Model Documentos
        $model =& $this->getModel('documentos');

        // Carrega dados
        $lists  = $model->getList();
        $rows   = $model->getData();
        $pagination = $model->getPagination();
        // atribui os dados às respectivas variáveis no layout
        $this->assignRef('lists',$lists);
        $this->assignRef('items',$rows);
        $this->assignRef('pagination',$pagination);
        $this->assignRef('option',$option);

}
```

No código apresentado é feita a referência à classe *model* a ser utilizada através do método `getModel()` implementado na classe `JView` da *framework*. Neste método foi passado o parâmetro documentos que significa que será obtida a referência à classe `SgdescolasModelDocumentos`. Como as *views* têm o mesmo nome que as *models* não era necessário passar o nome da *model* através do método. No entanto para se perceber melhor a

origem dos dados, optou-se por identificar o nome da *model* utilizada. Obtida a referência à classe *model* já é possível aceder aos seus métodos (`$rows = $model->getData();`). Os dados provenientes dos métodos da classe *model* referenciada são então passados para o *layout* da *view* através do método `assignRef()`. O primeiro parâmetro deste método corresponde ao nome da variável do *layout* que receberá os dados.

3.3.3. Classes *Controller*

As classes *controller* são responsáveis por controlar as acções do utilizador. É através destas classes que é possível desencadear, por exemplo, os processos de gravar, editar e eliminar registos. Estas classes recebem os dados provenientes das interfaces através dos dados POST do pedido, verificam esses dados e desencadeiam ou não uma ou várias acções. Podem, por exemplo, chamar os métodos de uma classe *model* para enviar os dados do pedido para a base de dados.

Em qualquer componente deve existir uma classe controller principal. Quando existem várias *views* deve existir também uma classe *controller* para cada uma dessas *views*. Estas classes encontram-se no directório `controllers` do componente.

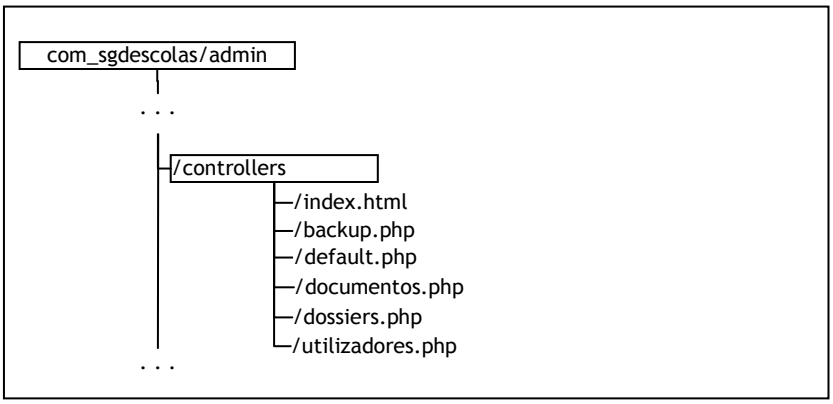
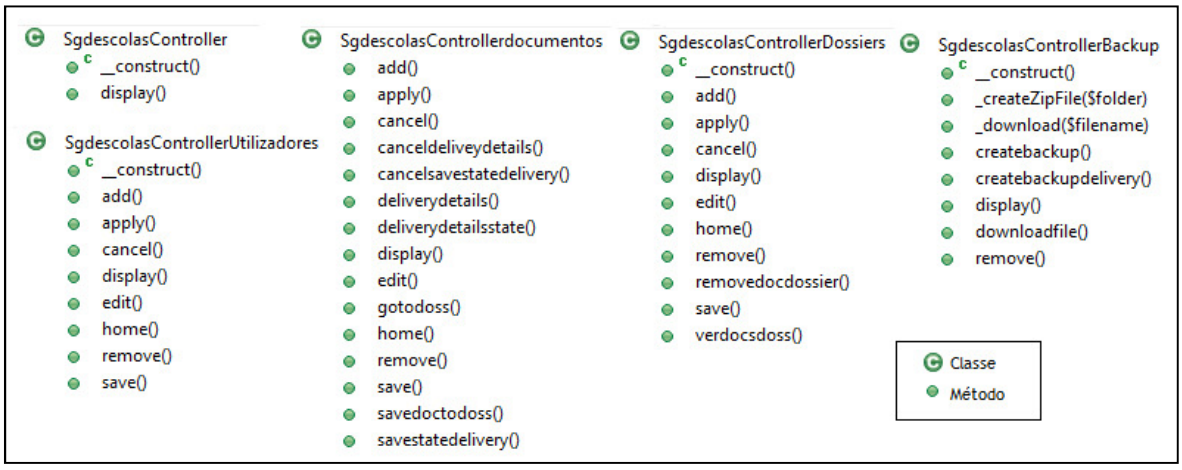


Figura 40 - Directório *controller* do componente

Os ficheiros utilizadores.php, documentos.php, dossiers.php, backup.php e default.php apresentados na figura anterior, contêm a implementação das classes controller `SgdescolasControllerUtilizadores`, `SgdescolasControllerDocumentos`, `SgdescolasControllerDossiers`, `SgdescolasControllerBackup`, `SgdescolasController`, respectivamente. Estas classes derivam da classe `JController` da *framework*.



A figura acima apresenta as classes *controller* e os seus métodos. As classes `SgdescolasControllerUtilizadores`, `SgdescolasControllerDocumentos`, e `SgdescolasControllerDossiers` utilizam os métodos das classes *model* com o mesmo nome para permitir adicionar, editar, eliminar registos da base de dados. Como exemplo, o seguinte código demonstra o método `remove()` da classe `SgdescolasControllerUtilizadores` que controla a eliminação de registos dos utilizadores da base de dados.

```
public function remove()
{
  // Referência à classe model
  $modelo = $this->getModel('utilizadores');

  // Obtém os ids provenientes do pedido
  $cids = JRequest::getVar( 'cid', array(0), 'post', 'array' );

  // desencadeia o método para remover os registos e verifica o resultado
  if(!$modelo->apaga())
  {
     $message = JText::_( 'USER_DELETE_ERROR' );
     $message .= ' ['.$model->getError().']';
     $type = 'error';
  }
  else
  {
     $message = JText::_( 'USER_DELETE_OK' );
     $type = 'message';
  }

  // Redirecciona para a view
  $this->setRedirect('index.php?option='.JRequest::getCmd('option').'&view='.
  JRequest::getCmd('view'), $message, $type);
}
```

Caso a tarefa proveniente do pedido seja `remove` é desencadeado o método com o mesmo nome no *controller*. Para além da tarefa a realizar, o pedido contém também os identificadores dos registos a remover que o utilizador da aplicação seleccionou através do *layout* da *view* utilizadores. O método `JRequest::getVar( 'cid', array(0), 'post', 'array' )` permite aceder à variável `cid` do pedido que contém um *array* com os registos seleccionados. Através do método `setRedirect()` a aplicação é redireccionada para a *view* do componente que deverá ser apresentada ao utilizador após a acção. O primeiro parâmetro corresponde ao caminho, o segundo à mensagem de confirmação da acção e o terceiro ao tipo da mensagem (de erro ou sucesso).

3.3.4. Barras de ferramentas

As barras de ferramentas do componente contêm os botões através dos quais o utilizador desencadeia as suas acções. Existem diferentes barras de ferramentas que se adequam às interfaces e acções que os utilizadores podem realizar. Para a criação das barras de ferramentas, de modo a serem utilizadas no componente e devidamente integradas no *back end* do Joomla, foram criados os ficheiros `toolbar.sgdescolas.php`, `toolbar.sgdescolas.class.php` e `toolbar.sgdescolas.html.php`.

O ficheiro `toolbar.sgdescolas.php` é responsável por identificar as acções dos utilizadores através do pedido efectuado no *browser* e decidir que barra de ferramentas deve ser visualizada. Quando no *back end* o Joomla faz o primeiro carregamento do componente, procura o ficheiro com o nome `toolbar.nomedocomponente.php`. No caso do componente `sgdescolas` o Joomla vai carregar o ficheiro `toolbar.sgdescolas.php`. O seguinte código demonstra como é feita a decisão da barra que deverá ser apresentada.

```
...
...
...

//incluir necessários para mostrar a toolbar.
require_once( JApplicationHelper::getPath( 'toolbar_html' ) );

//identificação da tarefa, view e layout
$task = JRequest::getCmd('task','display');
$layout = JRequest::getCmd('layout','default');
$view = JRequest::getCmd('view','default');

//escola da barra de ferramentas
switch ($view) {
        case "documentos" :{
                switch ($task) {
                        case "add":
                                TOOLBAR_sgdescolas::MENU_NOVO_DOCUMENTO();
                                break;

                        case "edit":
                                TOOLBAR_sgdescolas::MENU_EDITAR_DOCUMENTO();
                                break;
                                        ...
                                        ...
                                        ...
                default :
                                switch ($layout) {
                                        case "deliverylist":
                                        TOOLBAR_sgdescolas::MENU_LISTA_ENTREGAS();
                                        break;

                                default :
                                TOOLBAR_sgdescolas::MENU_DOCUMENTOS ();
                                }
                                        ...
                                        ...
                                        ...
```

Através do método `JRequest::getCmd()` são identificadas as tarefas, as *views* e os *layouts* executados pelo utilizador. O primeiro parâmetro corresponde ao nome da variável a verificar (`task`, `layout` e `view`) e o segundo o valor assumido, caso a variável não contenha qualquer valor. Seguidamente são executados os métodos relativos à barra de ferramentas responsáveis por apresentar os botões de acordo com a acção, *view* e *layout* do pedido.

O ficheiro `toolbar.sgdescolas.class.php` contém a implementação da classe `SgdescolasToolBar` responsável por apresentar os botões das barras de ferramentas. A figura seguinte apresenta os métodos desta classe.

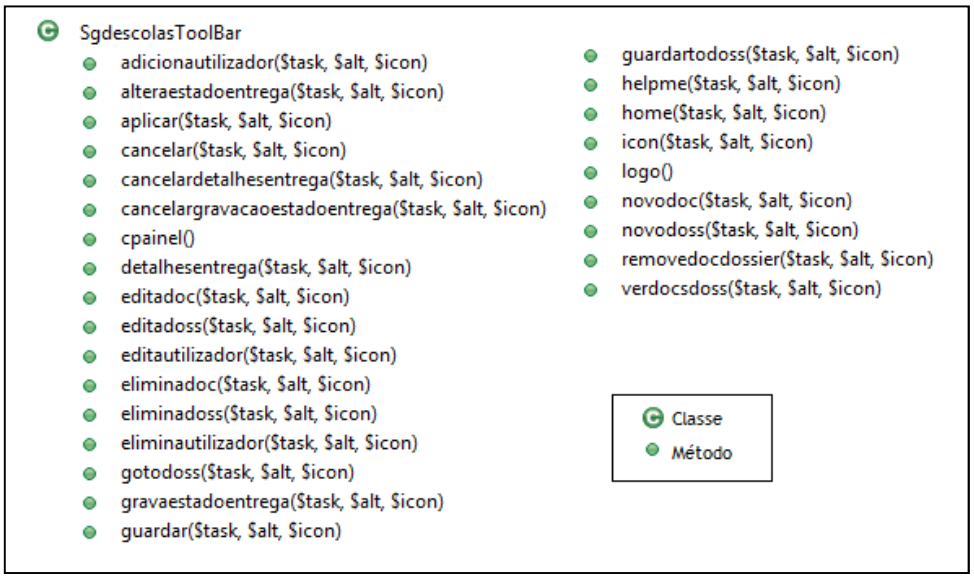


Figura 41 - Classe `SgdescolasToolBar`

Na generalidade dos métodos que se podem observar, são recebidos três parâmetros. O primeiro (`task`) corresponde à tarefa a executar, o segundo (`alt`) ao texto que identifica o botão, e o terceiro (`icon`) ao ícone gráfico que representa o botão. O código seguinte exemplifica um dos métodos.

```
function novodoc($task='add', $alt='Novo', $icon='novodoc')
{
    $alt=JText::_('TOOLBAR_ICON_NEW_DOC');
    SgdescolasToolBar::icon($task, $alt, $icon);
}
```

Apesar do método ser bastante simples, é de realçar que o valor atribuído ao primeiro parâmetro (`task`) deve coincidir com o nome da função na classe *controller* que é responsável por controlar a acção do utilizador.

Para a integração com um ficheiro de tradução, na função pode-se atribuir ao segundo parâmetro o valor devolvido pelo método `JText::_()` que procura no ficheiro o valor correspondente à expressão passada como parâmetro.

Relativamente ao valor do terceiro parâmetro, este permitirá carregar o ícone com base no nome que consta no ficheiro CSS e que representa o caminho e o nome do ficheiro de imagem do botão. Para que o Joomla carregue devidamente as imagens referentes aos botões da barra de ferramentas, através classe `JToolBarHelper` da *framework*, no ficheiro CSS utilizado pelo componente (`…/assets/css/geral.css`) a definição dos botões deve seguir obrigatoriamente a seguinte estrutura:

```
.icon-32-novodoc {
        background-image: url(../images/icon-32-helpme.png);
}
```

A classe `JToolBarHelper` irá procurar o nome do ícone no final de `.icone-32-` e o nome do ficheiro deve ter como prefixo `icon-32-` .

O ficheiro `toolbar.sgdescolas.html.php` contém a classe responsável pelas barras de ferramentas. Nesta classe é implementada a organização dos botões e atribuído o título às diferentes barras. Na figura seguinte podemos verificar os métodos desta classe.

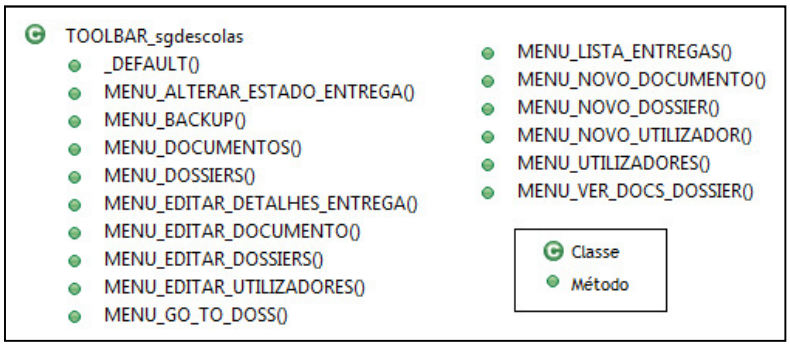


Figura 42 - Classe `TOOLBAR_sgdescolas`

Como se pode observar na figura, a classe `TOOLBAR_sgdescolas` contém os métodos responsáveis pela construção das diversas barras de ferramentas, que serão visualizadas de acordo com a interface em execução. A título de exemplo, observe-se o seguinte código:

```
function MENU_DOCUMENTOS() {

        JToolBarHelper::title( JText::_( 'DOCUMENT_MANAGEMENT' ), 'documentos.png' );
        SgdescolasToolBar::gotodoss();
        SgdescolasToolBar::eliminadoc();
        SgdescolasToolBar::editadoc();
        SgdescolasToolBar::novodoc();
        JToolBarHelper::spacer();
        SgdescolasToolBar::home();
}
```

Nesta função é utilizado o método `JToolBarHelper::title()` que é responsável por atribuir o nome à barra de ferramentas e o seu ícone representativo. Para o carregamento da imagem relativa ao ícone, recorre-se ao mesmo método utilizado nos botões da barra de ferramentas, no entanto, o nome na CSS que se refere à imagem deve ter como prefixo `.icone-48-` . A figura seguinte demonstra o resultado da construção da barra de ferramentas construída com base no método `TOOLBAR_sgdescolas::MENU_DOCUMENTOS()`, visualizada no *layout* que apresenta lista de documentos.

Figura 43 - Barra de Ferramentas - Gestão de documentos.

3.3.5. Paginação, ordenação, filtragem e procura de registos

A maioria dos componentes Joomla controla uma grande quantidade de objectos que representam os registos nas tabelas da base de dados. Para facilitar os aspectos relacionados com a paginação, ordenação, filtragem e procura destes registos, a *framework* do Joomla possui os métodos necessários para realizar estas tarefas.

No que diz respeito à paginação, esta é utilizada de modo a dividir os dados provenientes da base de dados em múltiplas páginas. Estas tarefas são realizadas através dos métodos da classe `JPagination` da *framework*. A esta classe estão associados os atributos `limit`, `limitstart`, `total` e `_viewall`. O atributo `limit` corresponde ao número máximo de registos a apresentar por página, o atributo `limitstart` corresponde ao primeiro registo de uma página, o atributo total contém o número total de registos e o atributo `_viewall` é utilizado para ignorar a paginação e apresentar todos os registos. A figura seguinte representa a barra de rodapé responsável pela paginação.

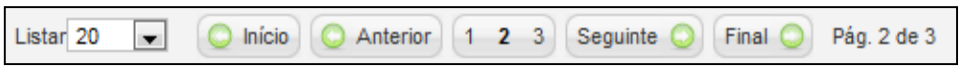


Figura 44 - Paginação

A classe `JPagination` já fornece os métodos necessários à criação das barras que permitem a navegação pelas múltiplas páginas no lado do *back end* dos componentes. Para tratar os valores das variáveis referentes à paginação é normalmente adicionado um método `getPagination()` à classe *model* responsável por tratar os registos nas tabelas da base de dados. O seguinte código, implementado na classe `SgdescolasModelDocumentos`, demonstra este método:

```
function getPagination(){

  global $mainframe, $option;

  if ($this->_pagination) {
   return $this->_pagination;
  }

  jimport('joomla.html.pagination');

  $limit = $mainframe->getUserStateFromRequest( 'global.list.limit', 'limit',
                                                $mainframe->getCfg('list_limit'), 'int' );

  $limitstart = $mainframe->getUserStateFromRequest( $option.'.limitstart', 'limitstart',
                                                     0, 'int' );

 if (!$this->_total) {
       $this->getTotal();
 }

 $pagination = new JPagination( $this->_total, $limitstart, $limit );

 $this->_pagination = $pagination;

return $this->_pagination;

}
```

Na implementação deste método deve ser utilizado um atributo `_pagination` para fazer de cache ao objecto de paginação. Para o mesmo efeito, pode-se também utilizar um atributo `_total` para guardar o número total de registos. Para se obter o valor das variáveis `limit` e `limitstart` é utilizado o método `getUserStateFromRequest()`. No primeiro parâmetro deste método é passada a variável `global.list.limit`, que é utilizada pelo Joomla para limitar o número de registos apresentados por página. A variável `limitstart` utilizada neste método, devolvida depois da variável `$option`, permite ao Joomla identificar o componente onde é utilizada.

O método `getTotal()` tem também de ser criado na classe *model* para devolver o número total de registos envolvidos na paginação.

```
function getTotal(){

        $db =& JFactory::getDBO();

        if ($this->_total) {
                return $this->_total;
         }

        $where = $this->_build_where();

        $query  = 'SELECT COUNT(d.id)'
                . ' FROM #__sgdescolas_documentos AS d'
                . $filter
                . $where;

        $db->setQuery( $query );
        $this->_total = $db->loadResult();

        return $this->_total;

}
```

O método apresentado utiliza um método privado `_build_where()` para que a consulta devolva apenas o total dos registos pretendidos, baseando-se em opções seleccionadas pelo utilizador. Este método será demonstrado mais à frente.

Após a construção da função `getPagination()` e da função `getTotal()`, a função criada para devolver um conjunto de registos da base de dados deve utilizar os valores das variáveis de paginação na execução da consulta. Assim, o método `setQuery()` do objecto da base de dados, que é responsável por definir a consulta a executar, deve ser executado como demonstra o seguinte código:

```
                                        ...
                                        ...
        $pagination = $this->getPagination();

        $db->setQuery( $query, $pagination->limitstart, $pagination->limit );
        $rows = $db->loadObjectList();
                                        ...
                                        ...
```

Como se pode observar, o primeiro parâmetro do método `setQuery()` corresponde a uma *string* que terá a consulta em sql, o segundo ao valor da variável `limitstart` que indica o número de registo onde começa a paginação e o terceiro contém a variável `limit` que corresponde ao número de registos a apresentar por página.

Para facilitar a navegação entre os registos provenientes de determinada consulta, foi implementado um processo que permite a ordenação desses registos. Este processo assume que o componente está estruturado de acordo com o padrão MVC e utiliza a classe JHTML da *framework* que é responsável pelos elementos HTML e o método `JHTML::_()` .

No formulário do *layout* devem existir dois campos ocultos: `filter_order`, que define o campo a ordenar; `filter_order_Dir`, que define se a ordem é ascendente ou descendente.

```
<input type="hidden" name="filter_order" value="<?php echo $this->lists['order']; ?>" />
```

```
<input type="hidden" name="filter_order_Dir" value="<?php echo $this->lists['order_Dir'];
?>/>
```

O atributo `lists` está associado à classe *view* e contém o campo ordenado e a ordem.

No cabeçalho da tabela, por cada coluna que seja ordenável, deve ser incluído o código semelhante a:

```
<?php echo JHTML::_('grid.sort', 'DOC_NAME','d.nome',@$this->lists['order_Dir'],
@$this->lists['order']); ?>
```

Como se pode observar, no primeiro parâmetro deve ser passada a *string* `grid.sort`, o segundo parâmetro é o nome da coluna e que aparece no *layout*, o terceiro é a direcção actual da ordenação e o quarto corresponde à coluna actualmente ordenada.

Na classe *model* utilizada para aceder à base de dados, deve ser criado um método privado (por exemplo `_build_order_by()`) que permita a construção da cláusula ORDER BY com base nas variáveis de ordenação constantes no pedido. Como exemplo temos o seguinte método:

```
function _build_order_by(){

  global $mainframe, $option;

  $filter_order = $mainframe->getUserStateFromRequest( "$option.filter_order_docs",
       'filter_order', 'd.nome', 'cmd' );

  $filter_order_Dir = $mainframe->getUserStateFromRequest( "$option.filter_order_Dir_docs",
       'filter_order_Dir',    '', 'word' );

  $orderby = ' ORDER BY '. $filter_order .' '. $filter_order_Dir;

  return $orderby;
}
```

A variável global `$mainframe` é um objecto da classe `JApplication`. Como neste caso se está a trabalhar no *back end*, representa uma instância da classe a `JAdministrator`. A variável global `option` representa o componente. Através do método `getUserStateFromRequest()` utilizado na função, é obtida a coluna e a ordem para a criação da consulta. Assim, no método da classe *model* responsável pela criação da consulta que devolve os registos a apresentar no *layout*, basta recorrer ao método acima criado, da seguinte forma:

```
$query  = 'SELECT d.* FROM #__sgdescolas_documentos '. this->_build_order_by();
```

Outra das funcionalidades muito importantes quando se lida com uma grande quantidade de registos é a procura e aplicação de filtros. Este é implementado no componente de forma bastante similar ao processo de ordenação. A diferença é que neste caso será construída uma cláusula WHERE. Para facilitar a utilização da aplicação, foram acrescentados ao formulário dos *layouts*, onde é apresentada a lista dos registos da base de dados, um campo de procura e vários campos para aplicação de filtros.

Figura 45 - Barra de filtragem e procura de registos.

Na classe *model* deve então ser construído um método privado que crie a cláusula WHERE a ser utilizada no método que devolve os registos da tabela. O código seguinte demonstra a criação do método `_build_where()` aplicado numa das classes *model* do componente.

```
function _build_where(){

  global $mainframe, $option;

  $db =& JFactory::getDBO();

//verificação dos valores das caixas de selecção e texto de procura

  $filter_categoria = $mainframe->getUserStateFromRequest( "$option.filter_categoria_docs",
                                                            'filter_categoria',0,'int' );

  $filter_criador = $mainframe->getUserStateFromRequest( "$option.filter_criador_docs",
                                                        'filter_criador', 0,'int' );

  $filter_anolectivo = $mainframe->getUserStateFromRequest(
                      "$option.filter_anolectivo_docs",'filter_anolectivo','','string' );

  $search = $mainframe->getUserStateFromRequest( "$option.search_document", 'search',
                                                '', 'string' );
  $search      = JString::strtolower( $search );


  $where = array();

//Verificação do texto de procura
  if (isset( $search ) && $search!= ''){
    $searchEscaped = $db->Quote( '%'.$db->getEscaped( $search, true ).'%', false );
    $where[] = 'd.nome LIKE '.$searchEscaped.' OR d.palavras_chave LIKE '.$searchEscaped;
  }

//Verificação dos valores dos filtros
  if ( $filter_categoria )
    $where[] = 'd.id_categoria = '.$db->Quote($filter_categoria);

  if ( $filter_criador )
      $where[] = 'd.id_utilizador_criador = '.$db->Quote($filter_criador);

  if ( $filter_anolectivo )
      $where[] = 'd.ano_lectivo LIKE '.$db->Quote($filter_anolectivo);

//Construção da cláusula WHERE
  $where = ( count( $where ) ? ' WHERE (' . implode( ') AND (', $where ) . ')' : '' );

return $where;
}
```

No código apresentado começa-se então por aceder aos valores dos campos do formulários e que são responsáveis pela procura e filtragem utilizando o método `getUserStateFromRequest()`. Em seguida é construída a claúsula WHERE.

3.3.6. Instalação e configuração

O processo de instalação de um componente no Joomla deve ser executado a partir do seu gestor de extensões, disponível no submenu `instalar/desinstalar` do menu `extensões` do Joomla. É neste processo que são copiados os ficheiros necessários ao funcionamento do *back end* e do *front end*. São também executados procedimentos de verificação e criação de condições necessárias ao correcto funcionamento do componente. Para a instalação do componente deve ser criado um pacote "`nomedocomponente.zip`", contendo toda a sua

estrutura de directórios e respectivos ficheiros. Para o componente aqui desenvolvido foi então criado um pacote de nome `com_sgdescolas.zip`.

O componente `com_sgdescolas` para funcionar correctamente necessita que no processo de instalação sejam executadas as seguintes tarefas:

- **Cópia da estrutura de directórios e dos seus respectivos ficheiros para as directorias do Joomla**. Este processo é realizado pela própria plataforma Joomla com base no ficheiro XML *manifest* `install.xml` colocado na raiz da estrutura de directórios do componente.
- **Criação das tabelas do componente**. A plataforma Joomla responsabiliza-se por executar o *script* SQL com o código que permite a criação das tabelas do componente na sua base de dados. No caso de desinstalação do componente, a plataforma executará o *script* de remoção das tabelas.
- **Criação de directórios**. O componente necessita que no processo de instalação sejam criados dois directórios, um para armazenar os ficheiros relativos aos componentes e outro que irá conter os ficheiros de *backup*. Para estes directórios são também definidas, no processo de instalação, as políticas de segurança que evitam a visualização e o *download* directo destes ficheiros.

Para que as tarefas acima referidas sejam executadas foram criados os seguintes ficheiros:

- `install.xml`;
- `sql/install.mysql.utf8.sql`;
- `install.sgdescolas.php`;
- `install.sgdescolas.class.php`;
- `configuration.php`;
- `defines.php`;
- `.htaccess`.

O ficheiro `install.xml` é o ficheiro XML *manifest* responsável por todos os detalhes de instalação. É também neste ficheiro que se faz a configuração do menu e submenus do componente no *back end* do Joomla. Este ficheiro encontra-se estruturado como se demonstra no seguinte código.

```
<?xml version="1.0" encoding="utf-8"?>
<install type="component" version="1.5.0">
        <name>SGDEscolas</name>
        <creationDate>September 2010</creationDate>
        <author>Nuno Ramalho</author>
        <authorEmail>n-m-ramalho@sapo.pt</authorEmail>
        <authorUrl>http://www.sgdescolas.alfacedastic.com/</authorUrl>
        <copyright>Copyright © 2010 - All rights reserved.</copyright>
        <license>GNU/GPL</license>
        <version>1.0.0</version>
        <description>Sistema de Gestão Documental em Ambiente Web para Escolas do Ensino
                Básico e Secundário
        </description>
        <install>
                <sql folder="admin/sql">
                        <file driver="mysql" charset="utf8">install.mysql.utf8.sql</file>
                </sql>
```

```
    </install>
                                        ...

    <files folder="site">
    <!-- instalar controllers   -->
           <filename>controllers/index.html</filename>
           <filename>controllers/default.php</filename>
           <filename>controllers/documentos.php</filename>

                                        ...

    <installfile>install.sgdescolas.php</installfile>

    <administration>
    <!-- Menu de administração -->
           <menu img="components/com_sgdescolas/assets/images/icon-16-
                      sgdescolas.png">SGDEscolas</menu>
           <submenu>
                  <menu img="components/com_sgdescolas/assets/images/icon-16-
                             utilizadores.png"
                        link="option=com_sgdescolas&amp;view=utilizadores">
                   Utilizadores</menu>

                                        ...

    <!-- Ficheiros de administração -->
           <files folder="admin">
                  <filename>admin.sgdescolas.php</filename>
                  <filename>toolbar.sgdescolas.php</filename>

                                        ...

    <!--    instalar scripts base de dados   -->
           <files folder="admin/sql">
                  <filename>install.mysql.utf8.sql</filename>
                  <filename>uninstall.mysql.utf8.sql</filename>
                  <filename>index.html</filename>
           </files>

           <languages folder="admin/lang">
                  <language tag="pt-PT">pt-PT.com_sgdescolas.ini</language>
           </languages>

     </administration>
    </install>
```

O código acima demonstra algumas partes do ficheiro `install.xml`, onde é visível a informação geral do componente (nome, autor, versão, etc.), o modo como são identificados os ficheiros sql (*tags* `<install>`,`<sql>`) e de idioma (*tag* `<languages>`), os ficheiros a copiar (*tag* `<files>`) e a construção do menu do componente no back end do Joomla (tags `<administration>`,`<menu>`,`<submenu>`).

No ficheiro `sql/install.mysql.utf8.sql` encontra-se o *script* SQL que permite a criação da estrutura de tabelas do componente.

O ficheiro `install.sgdescolas.php` é responsável por executar os métodos necessários à apresentação das informações relativas ao componente e criação dos directórios para armazenar os documentos e os *backups*, no processo de instalação. O ficheiro `install.sgdescolas.class.php` contém a implementação destes métodos.

É no ficheiro `defines.php` que estão definidas as constantes com as informações gerais do componente (Nome do autor, versão, etc).

O ficheiro `configuration.php` contém a implementação da classe que define os parâmetros de configuração do componente, tais como, tamanho máximo do ficheiro para *upload*, tipos de ficheiro permitidos no *upload*, nome do directório onde são armazenados os

ficheiros relativos aos documentos, nome do directório para os *backups*, ano lectivo inicial e ano lectivo final.

O ficheiro `.htaccess` é copiado para os directórios onde serão guardados os ficheiros relativos aos documentos e *backups*. Este ficheiro contém o código que nega o acesso aos ficheiros do directório onde se encontra, prevenindo assim que se consiga realizar o *download* sem o controlo do componente.

3.3.7. Idioma

Uma das características da plataforma Joomla é o suporte a múltiplos idiomas. Cada idioma utiliza ficheiros com extensão `ini` e que se localizam no directório `language` da raiz do Joomla e do directório `administrator`. Para cada idioma deve existir dentro do directório `language` um subdirectório cujo nome deve ser igual ao da *tag*, definida de acordo com o RFC 3066 (Alvestrand, 2001). Assim os directórios que contêm os ficheiros de tradução em Português (de Portugal) devem denominar-se de pt-PT. As expressões de tradução que constam no ficheiro devem estar divididas em duas partes: o nome da *string* de tradução e o texto de tradução. O nome da *string* de tradução é utilizado pelos métodos da classe `JText` da *framework*. Caso os nomes das *strings* de tradução não estejam em maiúsculas não é possível devolver a sua tradução. O nome dos ficheiros de tradução deve estar no formato `tag.nomedaextensão.ini` (Kennard 2007, p.266), assim o nome do ficheiro de tradução do componente é `pt-PT.com_sgdescolas.ini`. É essencial que o ficheiro de tradução se encontre na codificação UTF-8. O código seguinte apresenta um excerto do ficheiro de tradução utilizado pelo componente.

```
…
DOCUMENT_LIST=Lista de documentos
DOCUMENT_MANAGEMENT=Gestão de documentos
DOCUMENT_SHARE_SETTINGS=Definições de partilha
DOSSIER_MANAGEMENT=Gestão de dossiers
EDIT_DOCUMENT=Editar documento
…
```

Para que sejam enviadas as mensagem traduzidas durante a execução da aplicação, pode-se recorrer ao método `JText::_()` como demonstra o seguinte exemplo.

```
                                        ...
                                        ...
if($model->grava($data, $file))
        {
                // Mostra mensagem de sucesso
                $message = JText::_('DOCUMENT_SAVE_OK');
                $type = 'message';

                                        ...
                                        ...
```

Como se pode observar no código acima, a variável `$message` ficará com o valor devolvido pelo método `JText::_('DOCUMENT_SAVE_OK')`. O método procura no ficheiro de tradução a expressão `'DOCUMENT_SAVE_OK'`, que corresponderá a uma mensagem do tipo "`Documento guardado correctamente`".

3.4. Desenvolvimento do *front end*

O *front end* do componente `com_sgdescolas` corresponde à parte da aplicação que é executada pelo utilizador comum do CMS Joomla. Tendo em conta os requisitos da aplicação, através das interfaces existentes no *front end* do componente, os seus utilizadores podem visualizar os seus dados, realizar a gestão dos seus documentos, dossiers, entregas, documentos públicos e partilhas. Para facilitar a utilização do sistema foi criada uma interface principal que consiste num painel de controlo que permite o acesso a todas as funcionalidades executadas pelos utilizadores.

No desenvolvimento do *front end* foi necessário criar uma nova estrutura MVC do componente, criando as várias classes *model*, *view, controller*, da mesma forma que foi criada a estrutura do *back end*.

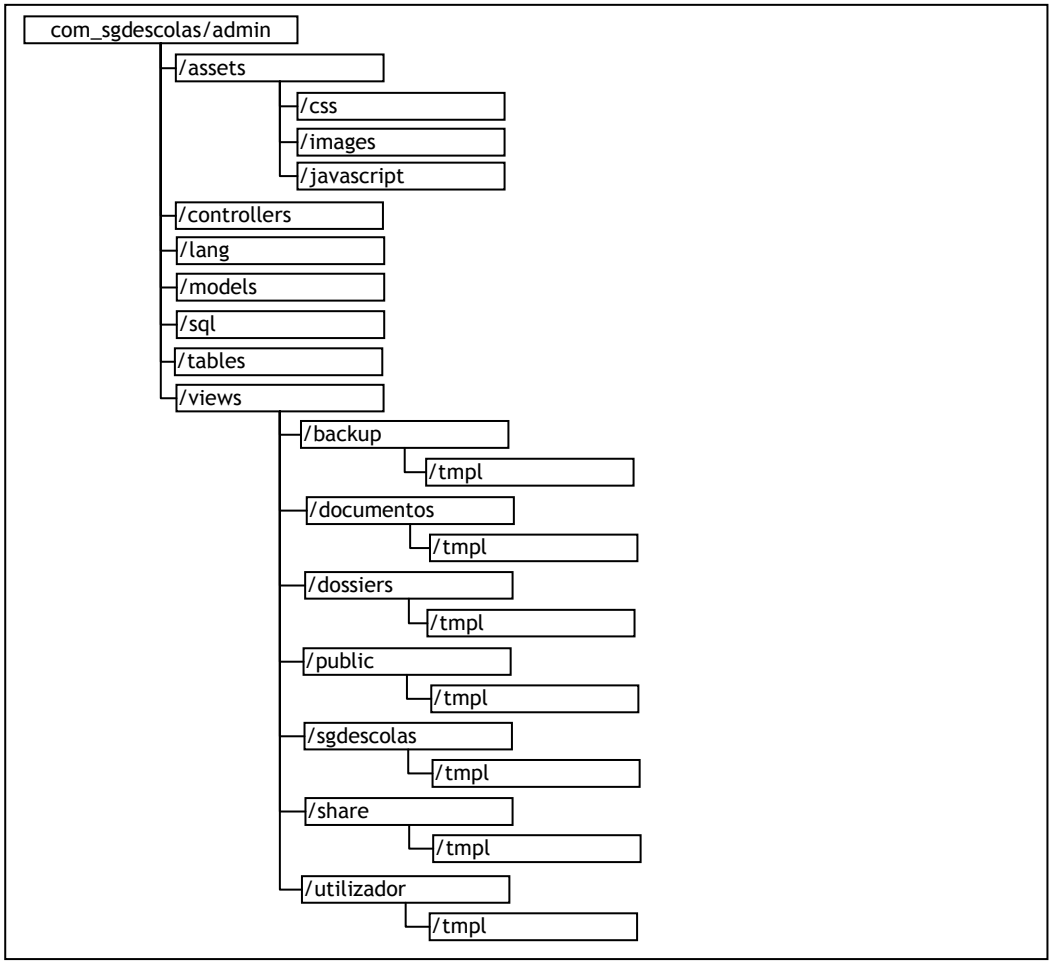


Figura 46 - Estrutura de directórios do *front end*

Através da estrutura apresentada na figura acima, pode-se observar a que é similar à estrutura do *back end*, apresentando ainda mais duas *views*, a `public` e `share`. A *view public* é responsável por apresentar os documentos públicos e a *view share* apresenta os documentos partilhados com o utilizador. Para estas *views* foram também criadas as respectivas classes *controller*, a fim de controlar as acções dos utilizadores para as funcionalidades de acesso aos documentos públicos e ficheiros partilhados. Não foi necessária a criação de novas classes *model*

para estas *views*, uma vez que utilizam também os métodos implementados na classe *model* documentos.

Na estrutura do *front end* não existem os directórios `sql` e `tables`. Estes directórios contêm os *scripts* sql e as classes para a criação das entidades. São necessários apenas no *back end* do componente. Toda a estrutura de directórios do *front end* pode ser consultada no anexo 3.

Como todo o desenvolvimento do *front end* é semelhante ao desenvolvimento do *back end*, agora serão apenas focados alguns dos aspectos característicos do *front end*.

3.4.1. Classes *view*, *model* e *controller*

No desenvolvimento do *front end* foram desenvolvidas mais duas classes *view* que no *back end*. A classe `SgdescolasViewPublic` e a classe `SgdescolasViewShare`. Os métodos destas classes encontram-se representados na figura seguinte.

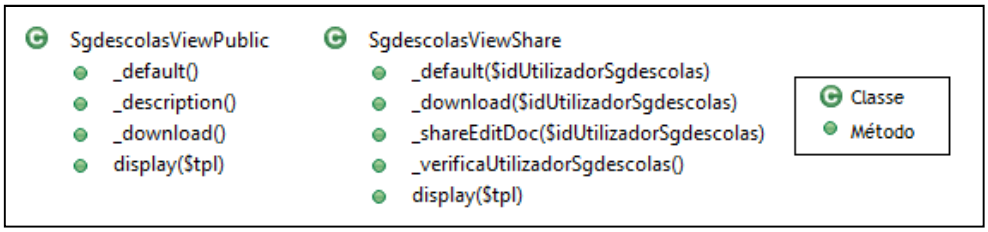


Figura 47- Classes `SgdescolasViewPublic` e `SgdescolasViewShare` do *front end*.

Estas classes *view* recorrem à classe *model* documentos do *front end* para aceder aos dados da base de dados. Os métodos implementados na classe `SgdescolasViewPublic` permitem visualizar a lista de documentos públicos, as suas características e realizar o download desses documentos. Os métodos implementados na classe `SgdescolasViewShare` permitem aceder aos documentos partilhados. Para cada uma destas *views* foram criados os respectivos *layouts*.

Com o objectivo de controlar as acções dos utilizadores nestas *views* foi necessária a implementação das respectivas classes controller.

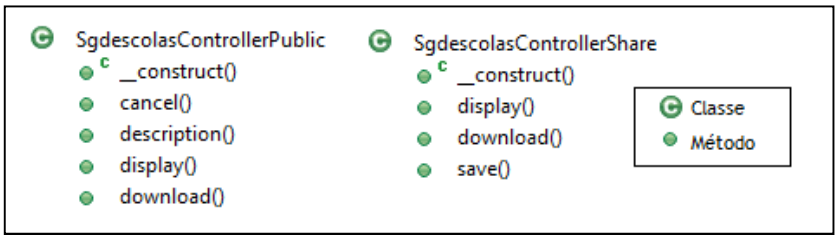


Figura 48 - Classes `SgdescolasControllerPublic` e `SgdescolasControllerShare` do *front end*.

Através dos métodos destas classes *controller*, o componente controla a visualização dos documentos públicos e partilhados e acções permitidas sobre estes documentos.

3.4.2. Integração com o gestor de itens de menu

Para a utilização do *front end* o administrador deve construir um menu que permita o acesso às interfaces do componente, para isso, a partir do gestor de menus do Joomla, poderá ser criado um novo menu para albergar os itens correspondentes aos *links* para as diferentes interfaces do componente. Estes *links* são adicionados através do gestor de itens do menu criado para o componente. Na criação do novo item para o menu, o Joomla irá reconhecer o componente, as suas *views* e os seus *layouts*. O administrador terá apenas de escolher o *layout* referente ao item do menu que está a criar.

A figura seguinte demonstra o reconhecimento do componente `SGDEscolas` pelo gestor de itens de menu do Joomla.

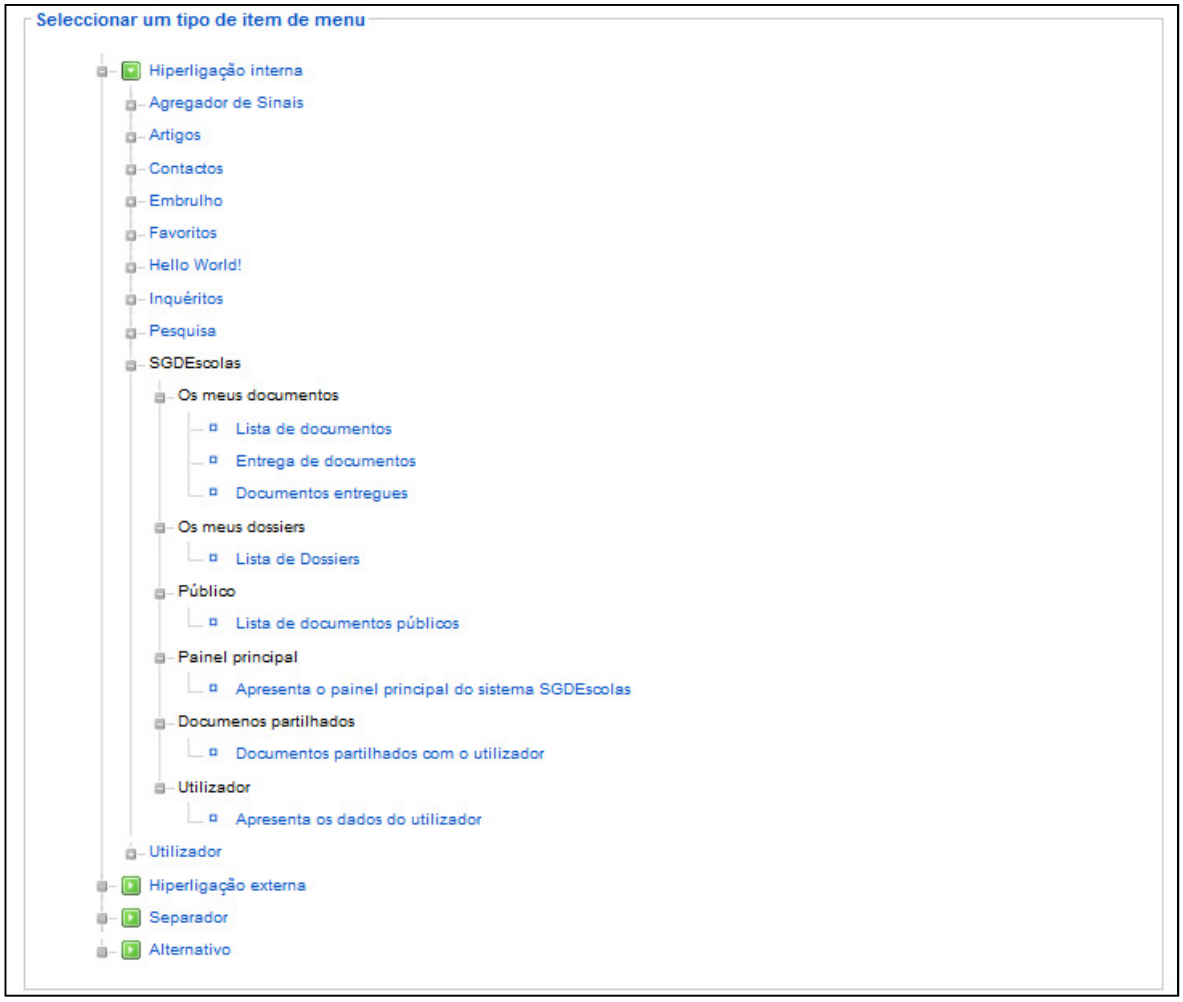


Figura 49 - Criação de itens para o menu do componente

Na figura pode-se visualizar o componente SGDEscolas, as diferentes *views* (Os meus documentos, Os meus dossiers, Público, Painel principal, Documentos partilhados e Utilizador) e respectivos *layouts* (Lista de documentos, Entrega de documentos, Documentos entregues, etc.)

Os nomes das *views* que constam na selecção de itens para um menu, aparecem por defeito com o nome do directório da *view* e os seus *layouts* com o nome dos respectivos ficheiros. Por exemplo, no componente temos uma *view* documentos, relativa ao directório `site/views/documentos` e um *layout default*, uma vez que o ficheiro responsável pelo *layout*

tem o nome de `default.php` (`site/views/documentos/tmpl/default.php`). Para que se possa dar um nome e uma descrição personalizada é necessário criar ficheiros xml que serão utilizados pelo Joomla durante a criação da árvore que permite a escolha dos itens de menu. Assim, dentro do directório da *view* documentos foi criado um ficheiro `metadata.xml` que tem o nome e a descrição da *view*. Este ficheiro tem o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<metadata>
	<view title="DOCUMENTS">
		<message><![CDATA[DOCUMENTVIEWSELECT]]></message>
	</view>
</metadata>
```

A *tag view* que tem o parâmetro `title` identifica o nome da *view* enquanto, que *tag* `message` representa a descrição. O Joomla procurará no ficheiro de tradução as *string* passadas nestas duas *tags* (no exemplo, `DOCUMENTS` e `DOCUMENTVIEWSELECT`).

Para o nome e descrição dos *layouts* é utilizado o mesmo método, mas agora adicionando um ficheiro xml com o mesmo nome do ficheiro responsável pela criação do *layout*. Como exemplo, para o *layout default* foi criado um ficheiro `default.xml` com o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<metadata>
	<layout title="DOCUMENTLIST">
		<message>
			<![CDATA[DOCUMENTLISTDESCRIPTION]]>
		</message>
	</layout>
	<state>
		<name>DOCUMENTLIST</name>
		<description>DOCUMENTLISTDESCRIPTION</description>
		<url>
		</url>
		<params>
		</params>

		<advanced>
		</advanced>
	</state>
</metadata>
```

A *tag layout* identifica o nome da *view* através do parâmetro `title`, enquanto que a *tag* `message` representa a descrição. Para que na interface onde são preenchidos os parâmetros dos itens de menu apareça também este nome e descrição, dentro da *tag* state é utilizada a *tag* `name` e `description`. Neste ficheiro podem também ser atribuídos outros parâmetros, tais como o endereço da ligação do item. A figura seguinte mostra o resultado da utilização destes ficheiros xml na árvore de selecção de itens de menu.

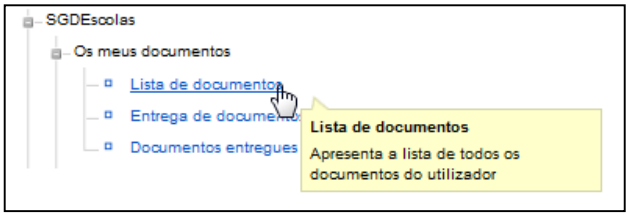


Figura 50 - Árvore de selecção de itens de menu personalizada para o componente.

Na próxima figura é apresentada a interface de configuração do item de menu seleccionado, onde é visível a secção Tipo de item de menu com a informação constante no ficheiro xml.

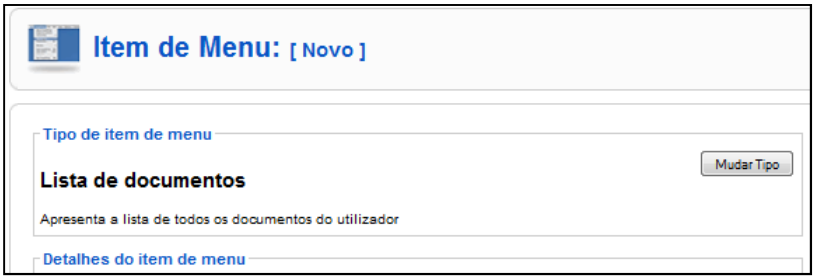


Figura 51 – Nome e descrição do item de menu

3.5. Resumo

A criação do componente SGDEscolas tira partido da utilização das bibliotecas da *framework do* Joomla. Para garantir a integração com o CMS Joomla todo o componente é desenvolvido segundo o padrão MVC. Este padrão prevê a criação de uma estrutura de directórios que separa o lado de *back end* do lado de *front end* do componente e que se rege pela criação de directórios para as classes *model*, *view* e *controllers*. Toda a estrutura do *back end* é idêntica à estrutura do *front end*. O desenvolvimento do componente passa por criar as classes *model* que fazem a comunicação com a base de dados, as classes *view* e os seus *layout* para a apresentação dos dados e as classes *controller* que controlam as acções do utilizador.

No lado do *back end*, para além do padrão MVC, há que ter em atenção os aspectos relativos à instalação do componente e que garantem que são reunidas as condições para o seu correcto funcionamento. Todos os ficheiros devem ser copiados para os respectivos directórios, sendo para isso necessária a identificação no ficheiro xml de instalação. Devem ser criadas as tabelas utilizadas pelo componente na base de dados do Joomla, através da criação de *scripts* sql. Para alojar os ficheiros relativos aos documentos e *backups* são também criados no processo de instalação os respectivos directórios. Por questões de segurança, são definidas políticas que negam o acesso directo ao conteúdo destes directórios. Estas tarefas são todas executadas automaticamente durante o processo de instalação do componente, a partir do gestor de extensões do Joomla.

No lado do *front end*, a criação de ficheiros xml associados às *views* e *layouts* facilita a criação dos itens de menu que permitem o acesso às interfaces de utilizador.

O componente utiliza um ficheiro de idioma permitindo, para além a tradução, a facilidade de personalização de mensagens devolvidas pela aplicação.

4. Testes e Resultados

Este capítulo reflecte a experimentação de algumas das funcionalidades implementadas no componente SGDEscolas (com_sgdescolas) para Joomla 1.5.x, desenvolvido na presente dissertação. O componente foi instalado na versão 1.5.15 da plataforma Joomla, com o pacote de tradução pt-PT.

4.1. *Back end* SGDEscolas

Através do gestor de extensões da plataforma Joomla foi realizada a instalação do pacote `com_sgdescolas.zip`. A instalação bem sucedida do componente apresenta as mensagens ilustradas na figura seguinte.

Figura 52- Instalação do componente `com_sgdescolas`.

Durante este processo de instalação foi criada toda a estrutura de directórios do componente na plataforma Joomla, copiados todos os ficheiros necessários, criadas as tabelas do componente na base de dados do Joomla e os directórios para alojar os documentos. Se por algum motivo de configuração do servidor não forem criados os directórios para alojar os documentos, é devolvida uma mensagem que explica como criar esses directórios manualmente.

Após a instalação do componente verificou-se que este é reconhecido pelo CMS Joomla, através da lista de componentes, no seu gestor de extensões. Na figura seguinte é possível observar na lista de componentes instalados, o nome do componente SGDEscolas, se está activo, a versão, data da versão, autor e compatibilidade com o Joomla.

Nº	Componente	Activado	Versão	Data	Autor	Compatibilidade
1	Newsfeeds	✓	1.5.0	April 2006	Joomla! Project	✓
2	Banners	✓	1.5.0	April 2006	Joomla! Project	✓
3	Weblinks	✓	1.5.0	April 2006	Joomla! Project	✓
4	Polls	✓	1.5.0	July 2004	Joomla! Project	✓
5	SGDEscolas	✓	1.0.0	September 2010	Nuno Ramalho	✓

Figura 53 - Reconhecimento do componente SGDEscolas pelo gestor de extensões.

O reconhecimento e integração do componente é também visível na criação de itens de menu, que permitem aos utilizadores do *front end* aceder às diversas interfaces da aplicação. Na figura seguinte pode-se observar as principais interfaces (principais *views* do componente) no processo de criação de itens de menu do Joomla.

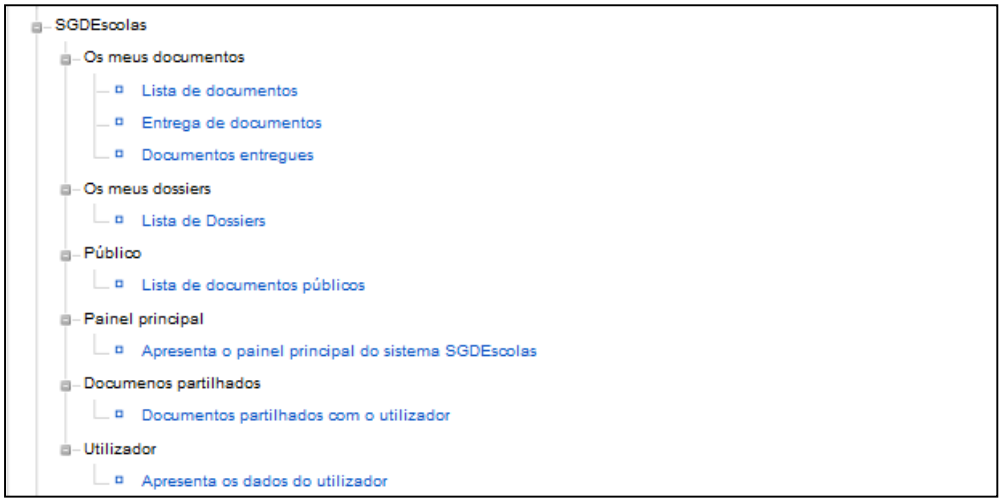


Figura 54 - Selecção da interface associada a um item de menu.

Após a instalação do componente é possível aceder às suas funcionalidades no lado do *back end*, a partir do menu componentes.

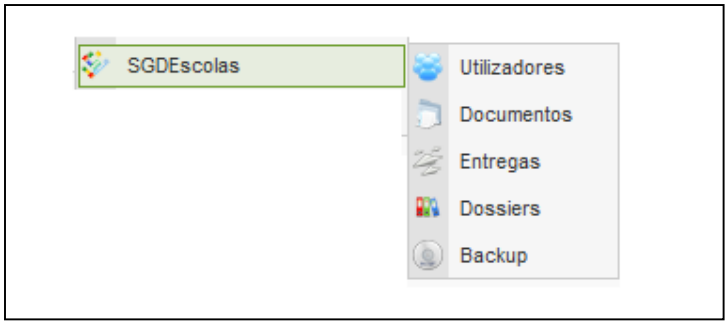


Figura 55 – Menu do componente SGDEscolas no *back end* do Joomla.

O administrador pode aceder às tarefas de gestão de utilizadores, documentos, entregas, dossiers e *backup*, a partir dos submenus do componente ou através do seu painel principal.

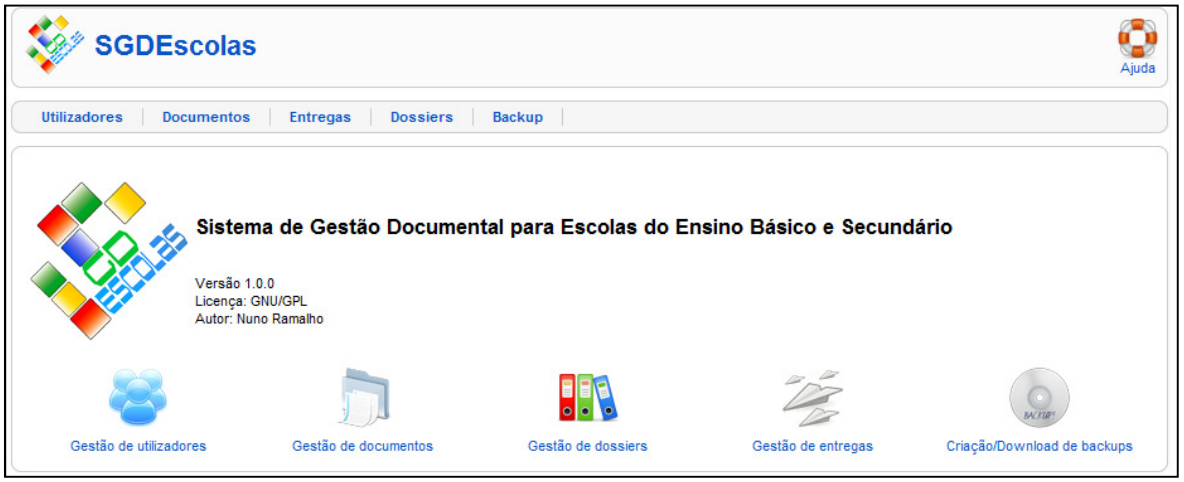


Figura 56 - Painel principal do componente no lado *back end* do Joomla.

A gestão de utilizadores apresenta como interface principal a listagem dos utilizadores associados ao componente e uma barra de ferramentas que permite a sua gestão.

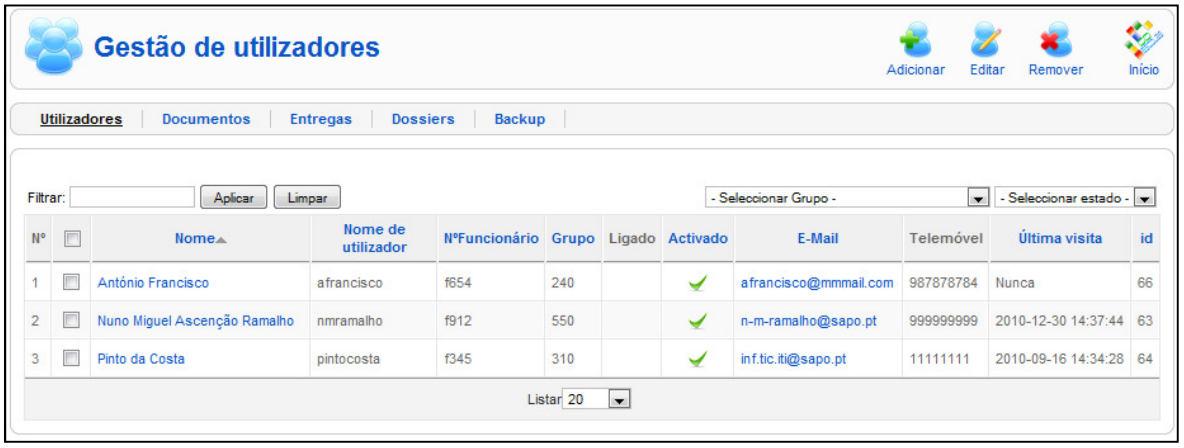


Figura 57 - Interface de gestão de utilizadores.

A partir da interface de gestão de utilizadores é possível adicionar, editar e remover utilizadores. O administrador pode ainda ordenar a lista de utilizadores pelos diferentes campos, procurar e aplicar filtros à lista.

Através do botão adicionar, o administrador pode associar um utilizador presente no sistema Joomla ao componente SGDEscolas. A figura seguinte demonstra a respectiva interface.

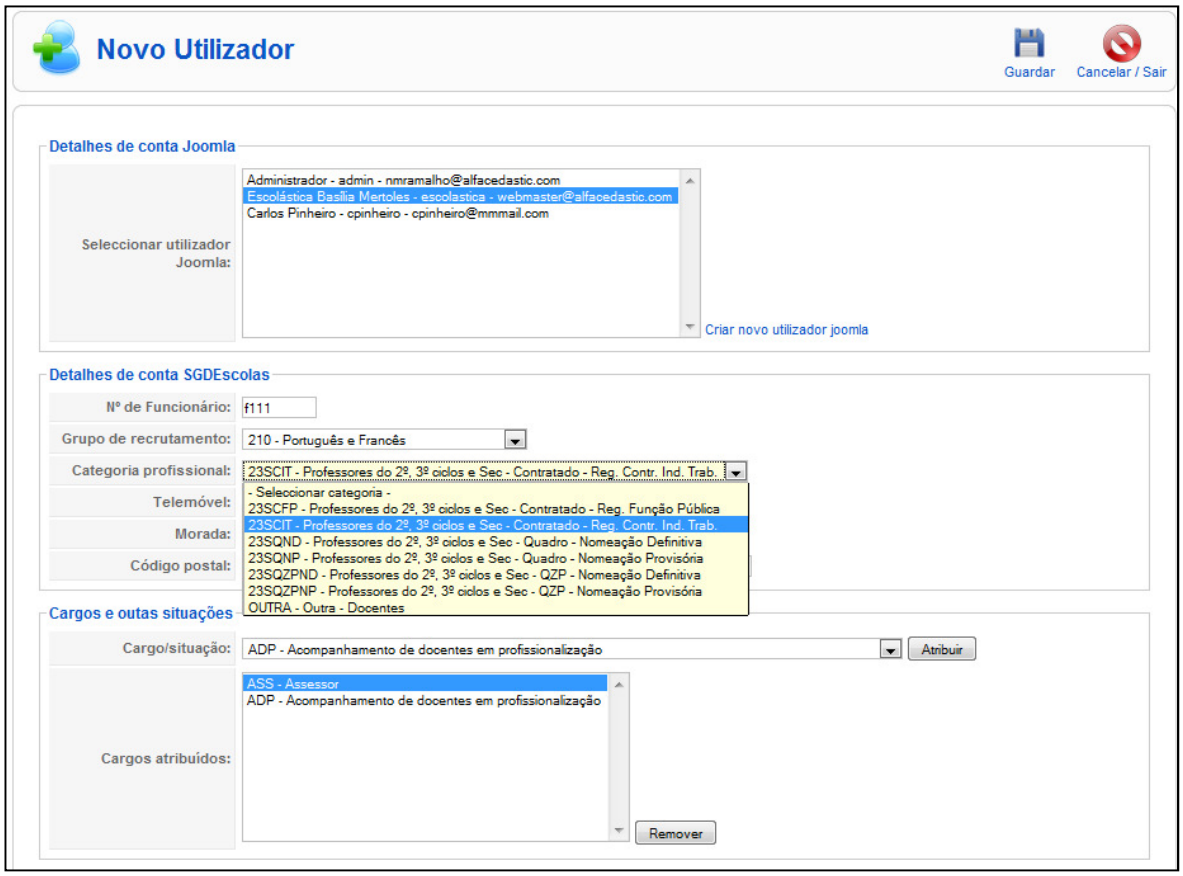


Figura 58 - Interface para adicionar um utilizador ao componente SGDEscolas.

Através da interface acima demonstrada, o administrador pode seleccionar um dos utilizadores do Joomla que ainda não pertence ao componente SGDEscolas, definir os atributos relativos aos seus dados pessoais, situação profissional e cargos ou funções que despenha na escola. Caso o utilizador ainda não exista na plataforma Joomla, é possível aceder directamente à interface do Joomla que permite essa criação.

Para gerir os documentos do sistema, o administrador pode aceder à interface de gestão a partir dos menus ou do painel de administração. Nesta interface, o administrador pode criar editar, eliminar e associar documentos a dossiers. É também possível filtrar os documentos por categoria, criador e ano lectivo, procurar e ordenar pelos vários parâmetros. O Administrador pode ainda realizar o *download* directo de documentos a partir desta lista. A figura seguinte apresenta o aspecto desta interface.

Figura 59 - Interface de gestão de documentos no *back end*.

O administrador pode adicionar documentos ao sistema através de uma interface intuitiva como demonstra a seguinte figura.

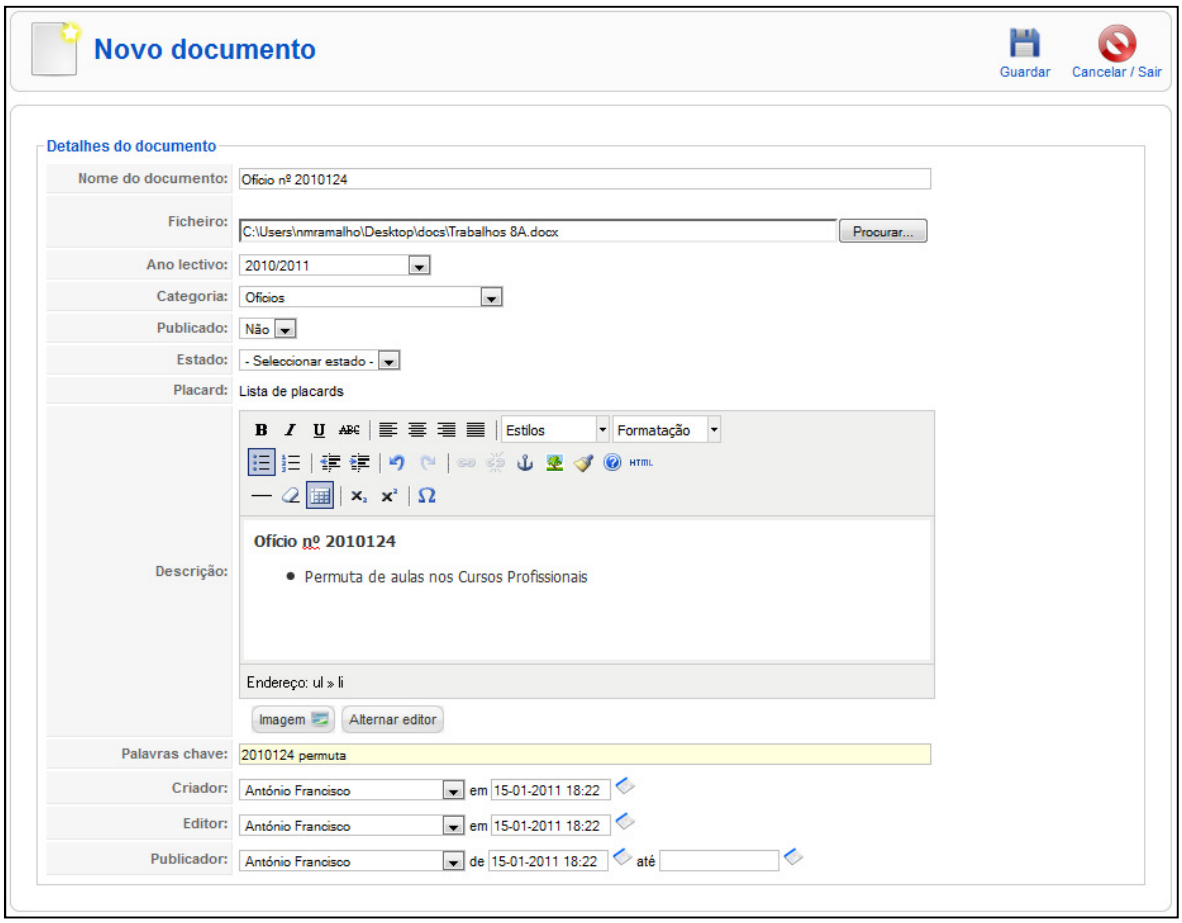


Figura 60 - Criação de um documento no *back end*.

Através do *back end* do componente, o administrador pode verificar os documentos oficialmente entregues pelos utilizadores à Direcção. Através da lista de documentos entregues é possível verificar os detalhes da entrega. Aqui o administrador deve verificar o documento e alterar o seu estado. Se o documento estiver em conformidade, pode alterar o estado do documento enviado para "aprovado". Caso detecte problemas com o documento, o administrador pode enviálo para revisão, indicando as suas observações ou enviando um novo documento (ou o recebido) com as sugestões de correcção.

Figura 61 - Interface de alteração de estado do documento entregue.

Após a alteração do estado do documento, o utilizador recebe uma notificação no seu *e-mail*. O sistema regista todo o processo de envio, revisão, reenvio e aprovação dos documentos. Para evitar conflitos, o administrador não consegue realizar alterações ao histórico dos documentos entregues, impossibilitando a deturpação do processo. O administrador pode apenas alterar o último estado da entrega e realizar o *download* dos vários documentos constantes no histórico. No entanto, o utilizador recebe uma notificação de alteração de estado. Através da figura seguinte pode-se observar o processo de entrega de um documento registado no *back end* do componente.

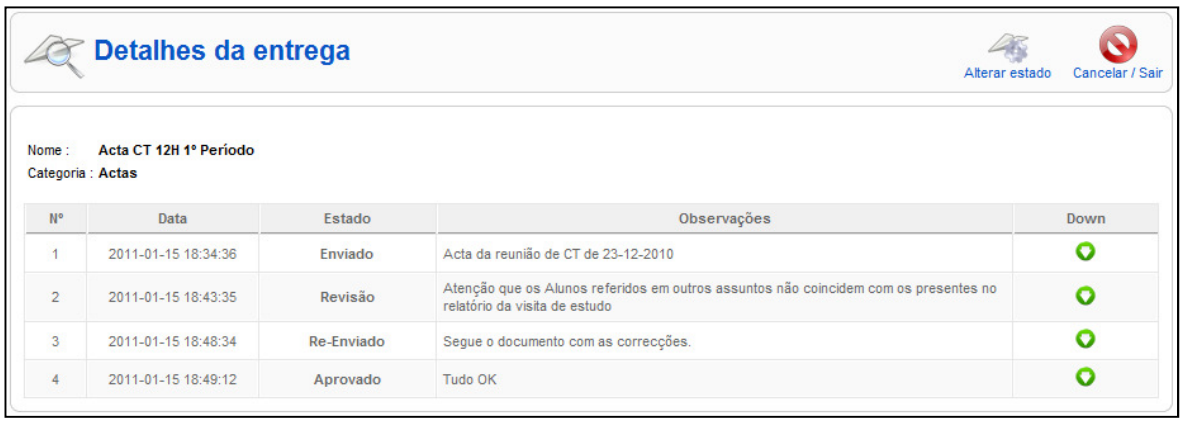


Figura 62 - Exemplo de um processo de entrega oficial de documentos.

Através do *back end* é possível realizar o *backup* e download dos documentos existentes no sistema. Os *backups* podem ser configurados por ano lectivo, tipo de documento e criador. Também podem ser feitos *backups* dos documentos oficialmente entregues.

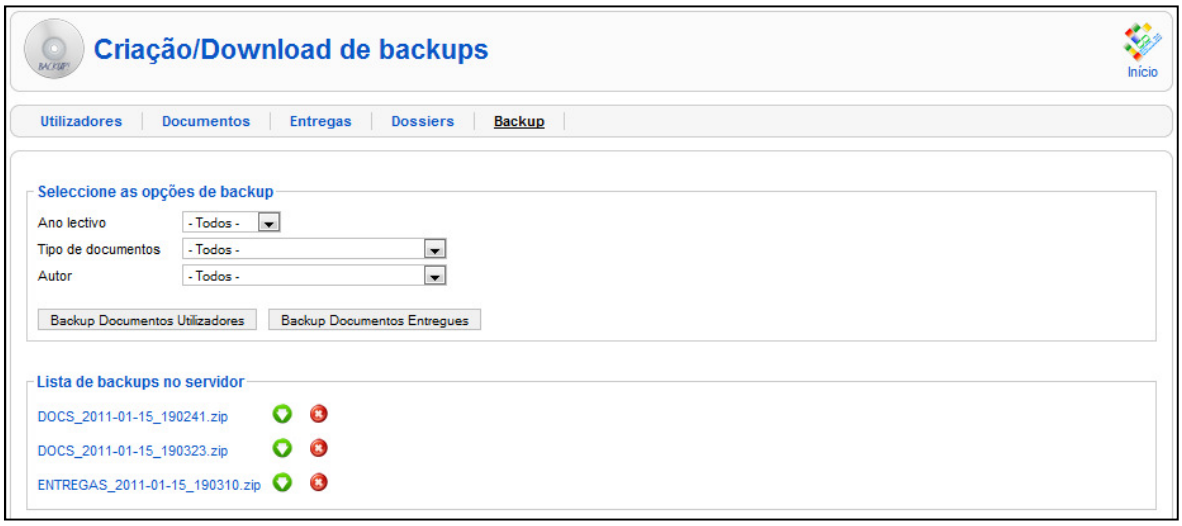


Figura 63 - Interface de *backup* de documentos.

4.2. *Front end* SGDEscolas

Relativamente ao *front end*, para que os utilizadores tenham acesso às funcionalidades do componente, é necessário criar um menu com os itens de acesso às suas interfaces. Este menu é criado no Joomla, a partir do gestor de menus e do gestor de itens do menu.

A partir do menu criado para o efeito, os utilizadores do sistema podem aceder ao painel principal no *front end* que permite o acesso a todas as funcionalidades.

Figura 64 - Painel principal do *front end*.

Como se pode observar na figura anterior, através do painel principal do *front end* o utilizador pode aceder aos seus dados, documentos, dossier, documentos partilhados com o utilizador, documentos públicos, realizar a entrega oficial de documentos e visualizar os documentos entregues. Na lateral também é possível observar o menu criado pelo administrador no gestor de menus do Joomla.

Através da interface "O meus Documentos" o utilizador pode, de forma simples e intuitiva, realizar toda a gestão de documentos. A figura seguinte apresenta esta interface e as suas funcionalidades.

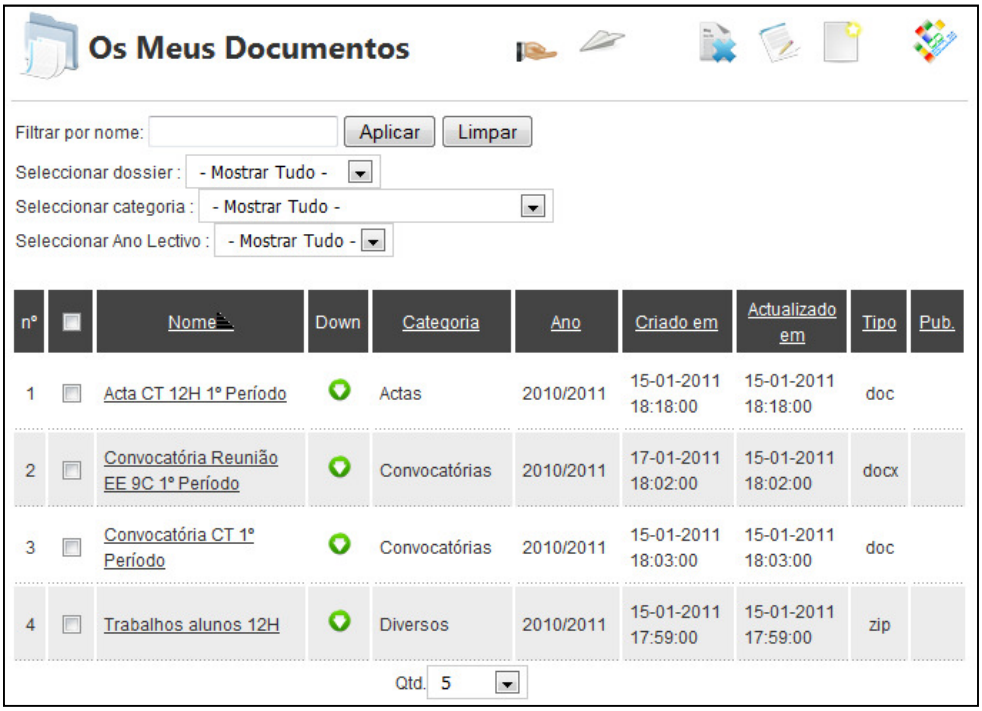


Figura 65 - Interface de administração de documentos no *front end*.

Como se pode observar na figura, através da interface apresentada, o utilizador pode criar, editar, eliminar, partilhar e realizar a entrega oficial de documentos. É também possível a aplicação de filtros por nome de documento, dossier, categoria e ano lectivo. Através da listagem é também possível a ordenação pelos diversos parâmetros e o download directo de documentos.

Para facilitar o trabalho colaborativo, o utilizador pode partilhar e aceder a documentos partilhados com o próprio. A partilha é realizada de forma simples, como demonstra a figura seguinte, e pode ser efectuada com permissões de visualização ou edição. Os documentos partilhados através desta interface são visualizados na interface.

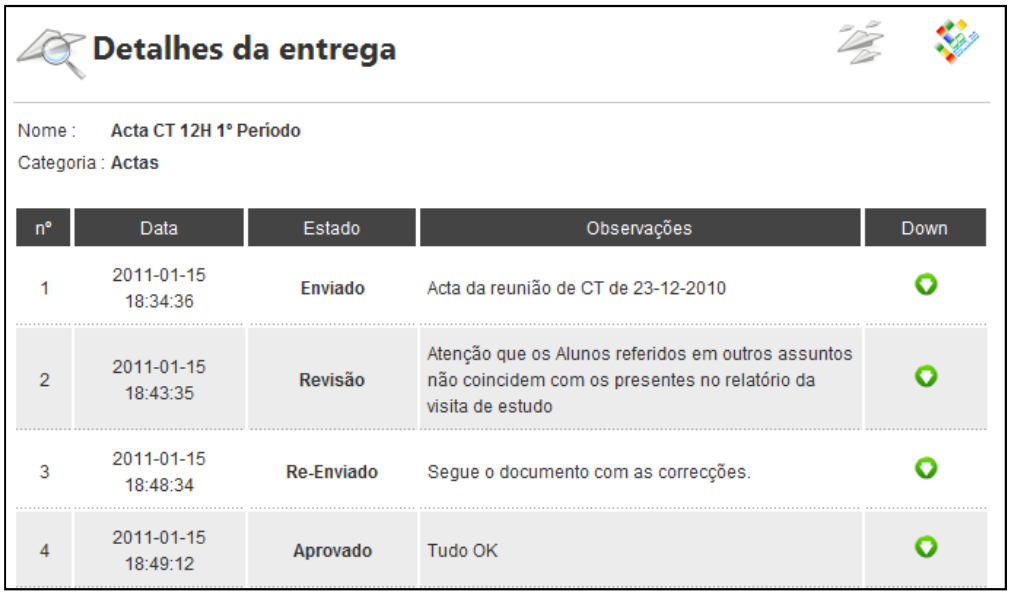
Figura 66 - Interface de partilha de documentos no *front end*.

Como se pode observar na figura, o utilizador está a partilhar o documento "Trabalhos alunos 12H" em modo de visualização com três outros utilizadores. Caso um utilizador altere um documento que foi partilhado consigo, o criador recebe a notificação dessa alteração.

Através da interface "Documentos entregues", o utilizador pode verificar os documentos oficialmente entregues à Direcção e o seu estado.

Figura 67 - Interface de listagem de documentos entregues.

Para que o utilizador possa verificar o processo de entrega de determinado documento, pode aceder a uma interface que mostra todo o detalhe.



Detalhes da entrega

Nome : **Acta CT 12H 1º Período**

Categoria : **Actas**

nº	Data	Estado	Observações	Down
1	2011-01-15 18:34:36	**Enviado**	Acta da reunião de CT de 23-12-2010	
2	2011-01-15 18:43:35	**Revisão**	Atenção que os Alunos referidos em outros assuntos não coincidem com os presentes no relatório da visita de estudo	
3	2011-01-15 18:48:34	**Re-Enviado**	Segue o documento com as correcções.	
4	2011-01-15 18:49:12	**Aprovado**	Tudo OK	

Figura 68 - Detalhe do processo de uma entrega oficial de documentos.

A figura acima apresenta o exemplo do processo de entrega de uma acta, onde são visíveis as datas de alteração de estado e os comentários referentes ao documento. É também possível realizar o *download* do documento nas diferentes etapas do processo.

4.3. Análise dos resultados

Para além das funcionalidades até ao momento demonstradas, houve um número considerável de outras funcionalidades que também foram testadas durante o desenvolvimento do componente.

A lista que se segue apresenta os principais testes realizados ao componente:

- instalação e configuração do componente;
- configuração de extensões permitidas, tamanho máximo dos ficheiros e nome dos directórios;
- funcionalidade dos menus, submenus, barras de ferramentas e outras ligações;
- criação, edição, eliminação de utilizadores;
- atribuição de cargos aos utilizadores no processo de criação e edição;
- criação, edição, *download*, partilha, publicação, procura, ordenação, listagem e entrega oficial de documentos;
- controlo no acesso aos documentos partilhados e públicos;
- criação, edição, procura, ordenação e listagem de dossiers;
- atribuição de documentos a dossiers;
- controlo do *Workflow* no processo de entrega de documentos;
- controlo no acesso aos documentos por parte de utilizadores não autorizados;
- *upload* de ficheiros para o servidor;
- controlo no acesso aos documentos nos directórios do servidor;
- *Backup* e *download* de backups;

- protecção de entrada de dados nas interfaces;
- funcionalidade das interfaces;
- notificações por *e-mail*.

Após testadas as funcionalidades aqui mencionadas, pode-se verificar o bom comportamento do componente. Verificou-se que a plataforma Joomla e sua *framework* foram capazes de suportar um sistema deste tipo. Desde a instalação até à sua utilização o componente demonstrou a total integração com o CMS Joomla.

As diversas interfaces foram construídas com o objectivo de simplificar a execução das tarefas por parte dos utilizadores, o que pode contibuir fortemente para a implementação e utilização deste sistema.

5. Conclusões

Este trabalho envolveu o estudo da criação de um DMS direccionado para as escolas básicas e secundários que tira proveito da plataforma CMS Joomla e da sua *framework*.

No capítulo 2 – Tecnologias de Gestão Documental e Conteúdos, foram abordados os aspectos relativos aos sistemas de gestão documental, sistemas gestores de conteúdos, CMS Joomla e desenvolvimento de extensões para esta plataforma. No desenvolvimento de um DMS é requisito ter uma adequada infra-estrutura e as ferramentas necessárias à criação, armazenamento, controlo e recuperação dos documentos. Aspectos como a identificação dos utilizadores e *workflow* demonstram-se fundamentais no processo de negócio da organização.

Desenvolver um DMS integrado na plataforma Joomla demonstrou-se vantajoso, uma vez que permitiu o desenvolvimento rápido do sistema através do aproveitamento das funcionalidades já implementadas nas bibliotecas do pacote Joomla! Framework API. Permitiu ainda associar todas as funcionalidades do sistema gestor de conteúdos e a possibilidade de inclusão de novas extensões.

O rigor na aplicação do padrão de programação MVC demonstrou-se fundamental para o sucesso no desenvolvimento do componente, estudado no capítulo 3 - Desenvolvimento do componente SGDEscolas. Este padrão permite a separação da parte responsável pelo acesso aos dados, apresentação e lógica de funcionamento da aplicação.

Com o presente trabalho, foi possível a criação de uma extensão que permite a gestão de documentos, na qual os utilizadores executam as diversas tarefas através de interfaces desenvolvidas com o intuito de facilitar a utilização do sistema. Durante o desenvolvimento da extensão foram também implementadas medidas para garantir a segurança no acesso aos documentos. Estas medidas baseiam-se no controlo dos utilizadores que podem aceder ao componente ou aos documentos por ele geridos e na definição de permissões relativas aos locais onde estes documentos são guardados.

No capítulo 4 – Testes e resultados - verificou-se toda a integração do componente com a plataforma Joomla através do sucesso na instalação, no reconhecimento e compatibilidade no gestor de estensões, bem como no reconhecimento das diversas *views* durante a criação de itens de menu para o *front end*.

Como sistema de gestão electrónica de documentos, pode-se verificar que o componente apresenta os principais elementos característicos dos sistemas DMS:

- o componente possui uma infra-estrutura subjacente que consiste na plataforma Joomla, na sua *framework* e nas tecnologias a ela associadas;
- possui ferramentas de suporte à criação de documentos que consistem nas interfaces criadas para o efeito;
- consegue suportar workflow, como demonstra o processo de entrega oficial de documentos;
- possibilita o armazenamento seguro e eficaz dos documentos no servidor, proporcionado pela criação de directórios específicos e definição das políticas de acesso;

- fornece mecanismos de controlo dos documentos, tais como, a informação relativa à entrada e saída do documento no sistema, informação do autor, editor, etc;
- possibilita a recuperação, visualização, impressão e distribuição dos documentos, do modo como os documentos são apresentados aos utilizadores.

5.1. Principais contribuições

Com o desenvolvimento da presente dissertação, podem-se verificar dois aspectos globais que identificam a importância do estudo aqui realizado. Um dos aspectos é o apoio às organizações escolares, no que diz respeito à gestão dos documentos produzidos pelos seus principais actores, os professores. Outro aspecto, consiste no estudo da criação de extensões do tipo componente que tirem partido de uma das plataformas de gestão de conteúdos para a Web mais utilizadas actualmente. Decompondo estes dois aspectos globais, podem-se verificar as seguintes contribuições:

- resumo dos principais conceitos relativos aos sistemas de gestão electrónica de documentos;
- resumo dos aspectos relativos aos sistemas de gestão de conteúdos e sua arquitectura;
- esclarecimento e demonstração do funcionamento do CMS Joomla e da sua *framework*;
- demonstração das potencialidades do CMS Joomla e da sua *framework*;
- demonstração da criação de componentes para o CMS Joomla;
- disponibilização de uma solução gratuita, de fácil implementação e utilização que possibilita a gestão electrónica dos documentos nas organizações escolares;
- constituição de uma base de trabalho para a evolução de um componente Joomla DMS capaz de satisfazer os requisitos relativos aos diversos processos de negócio existentes nas organizações escolares.

5.2. Trabalho futuro

Partindo da base criada com o desenvolvimento da presente dissertação, o componente SGDEscolas, verifica-se a existência de algumas funcionalidades adicionais que poderiam ser desenvolvidas com o intuito de melhorar este sistema, que se passam a enumerar:

- Interfaces para a gestão de tipos de documentos, cargos e funções, situação profissional, etc. – neste momento a aplicação depende exclusivamente do *script* sql para a introdução dos registos referentes a estes itens. Para facilitar a introdução de novos registos devem ser desenvolvidas as interfaces para o efeito.
- Interface de configuração do sistema – neste momento os aspectos relativos à configuração do sistema têm que ser definidos editando directamente o ficheiro de configuração no servidor. É relevante existir uma interface que permita através do *back end* definir esses parâmetros.

- Criar no *front end* um mecanismo que permita executar acções de administração próprias do componente por parte dos elementos da direcção da organização, evitando assim o contacto com todo o *back end* do Joomla.
- Criar um mecanismo de controlo de versões que permita a incrementação automática da versão consoante as actualizações ao documento e que permita manter um histórico dessas versões.
- Criar um mecanismo de assinaturas digitais que assegure a autenticidade e o valor legal da assinatura. Este mecanismo deverá permitir assinaturas por diversos utilizadores.

Um dos aspectos fundamentais a ter em conta na implementação de novas funcionalidades ao componente consiste em manter a facilidade na utilização. A aplicação deve ser vista como um meio facilitador e não como um entrave à actividade docente nas escolas. A introdução, acesso e organização dos documentos no sistema devem ser feitas com o menor dispêndio de tempo possível, não esquecendo, contudo, os aspectos relativos à segurança e fiabilidade.

6. Referências

Alfresco, 2010. *Alfresco Open Source Content Management - Alfresco Community*. [Em linha] Disponível em: <http://www.alfresco.com/products/networks/community/> [Acedido em Outubro de 2010].

Alvestrand, H., 2001. *Tags for the Identification of Languages*, Request for Comments: 3066. IETF. [Em linha] Disponível em: <http://www.ietf.org/rfc/rfc3066.txt> [Acedido e Dezembro de 2010].

Björk, B., 2003. *Electronic document management in construction - research issues and results*, ITcon Vol. 8, pg. 105-117, [Em linha] Disponível em: <http://www.itcon.org/2003/9> [Acedido em Agosto de 2010].

Boiko, B., 2005. *Content Management Bible*, *2nd Edition*. Wiley Publishing, Inc.

Bárcia, L., 2008. *A Utilização da Plataforma Joomla! Na Escola*. Universidade Católica Portuguesa. [Em linha] Disponível em: <http://www.esnips.com/doc/54c34ee9-3e58-4da1-b490-c02f4f05624d/Disserta%C3%A7%C3%A3o-de-mestrado-de-Luis-B%C3%A1rcia> [Acedido em Dezembro de 2010].

Cleveland, G., 1995. *Overview of Document Management Technology*, National Library of Canada. White paper. [Em linha] Disponível em: < http://archive.ifla.org/ VI/5/op/udtop2/udtop2.htm > [Acedido em Agosto de 2010].

CMC, 2007. *Document Management Overview – A guide to the benefits, technology and implemention essentials of digital document management solutions*.Compulink Management Center, Inc. [Em linha] Disponível em: <http://www.laserfiche.com/ pdf/brochures/DocumentMgntOver.pdf> [Acedido em Agosto de 2010].

Costa, M., 2008. *Política de Escola e Representações sobre o Insucesso Escolar. Um Estudo de Caso Comparativo*. [Em linha] Disponível em: <http://repositorio-iul.iscte.pt/bitstream/10071/1292/8/Disserta%C3%A7%C3%A3o%20de%20Mestrado_PE%20e%20Representa%C3%A7%C3%B5es%20sobre%20insucesso%20escolar.pdf> [Acedido em Dezembro de 2010].

Curry, B. English, B., 2008. *Microsoft® Office SharePoint® Server 2007 Best Practices*. Microsoft Press. [Em linha] Disponível em: <http://technet.microsoft.com/en-us/library/dd163514.aspx> [Acedido em Setembro de 2010].

Craine, K., 2008. *Why Document Management?*. Craine Communications Group. White paper. [Em linha] Disponível em: <http://www.document-trategy.com/Computhink_Why_ Document_Management_White_Paper.pdf> [Acedido em Agosto de 2010].

FCCN, n.d.. *Edu.PT - Programa Internet na Escola FCCN*. Fundação para a Computação Científica Nacional. [Em linha] Disponível em: <http://www.fccn.pt/index.php ?module=pagemaster&PAGE_user_op=view_page&PAGE_id=30&MMN_position=2:2> [Acedido em Setembro de 2010].

GEPE, 2007, *Estudo de Diagnóstico: a modernização tecnológica do sistema de ensino em Portugal - Principais resultados*, Gabinete de Estatística e Planeamento da Educação, Ministério da Educação, Portugal.

Graf, H., 2008, *Building Websites with Joomla! 1.5*, Packt Publishing.

Holzner, S. Conner, N., 2009, *Joomla! for Dummies*, Wiley Publishing, Inc.

Harwani, B., 2009, *Foundation Joomla!*, Friendsoft.

Heckman, J., 2008. *Why Document Management?*. White paper. Heckman Consulting. [Em linha] Disponível em: http://www.heckmanco.com/docs/DMWhitePaper.pdf> [Acedido em Agosto de 2010].

In Infopédia, n.d. *Documento*. Porto Editora. 2003-2010. [Em linha] Disponível em: <http://www.infopedia.pt/lingua-portuguesa/Documento> [Acedido em Agosto de 2010].

Joomla! 1.5 API Reference, 2010. *Joomla-Framework*. [Em linha] Disponível em: <http://api.joomla.org/li_Joomla-Framework.html> [Acedido em Agosto de 2010].

Joomla!, 2010. [Em linha] Disponível em: <http://www.joomla.org/> [Acedido em Maio de 2010].

Joomla! Extensions Directory, 2010. [Em linha] Disponível em: http://extensions. joomla.org/> [acedido em Maio de 2010].

Joomla! Features Overview, 2010. [Em linha] Disponível em: <http://www.joomla.org/ core-features.html> [Acedido em Outubro de 2010].

Joomla! *Technical Requirements*, 2010. [Em linha] Disponível em: <http://www. joomla.org/technical-requirements.html> [Acedido em Outubro de 2010].

Joomla! *Framework*, 2010. [Em linha] Disponível em : <http://docs.joomla.org/ Framework [Acedido em Agosto de 2010].

Joomla! *Developing MVC Component*, 2010. *Developing a Model-View-Controller Component - Part 1*. [Em linha] Disponível em : <http://docs.joomla.org/Developing_a_Model-View-Controller_ Component_-_Part_1#Introduction> [Acedido em Agosto de 2010].

JoomlaPT, 2010, Comunidade Portuguesa Joomla. [Em linha] Disponível em: <http://www.knowledgetree.org/> [Acedido em Maio de 2010].

knowledgeTree, 2010. *KnowledgeTree Document Management Made Simple*. [Em linha] Disponível em: <http://www.knowledgetree.org/> [Acedido em Outubro de 2010].

Lanham, C. Kennard, J., 2010. *Mastering Joomla! 1.5 Extension and Framework Development*, Packt Publishing.

Kennard, J., 2007. *Mastering Joomla! 1.5 Extension and Framework Development*. Packt Publishing.

Moodle, 2010. *Moodle.org: open-source community-based tools for learning*. [Em linha] Disponível em: <http://www.moodle.org/> [Acedido em Outubro de 2010].

Rahmel, D., 2007, Professional Joomla. Wiley Publishing, Inc.

Severdia, R. Crowder, K., 2010. *Using Joomla*. O'Reilly Media, Inc.

Shereves, R., 2010. *Joomla! Bible*. Wiley Publishing, Inc.

WFMC, 1999. *Workflow Management Coalition - Terminology & Glossary*. Documento número WFMC-TC-1011 Ver 3. linha] Disponível em: <http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html> [Acedido em Setembro de 2010].

Wampserver, 2010. [Em linha] Disponível em: < http://www.wampserver.com/> [acedido em Maio de 2010].

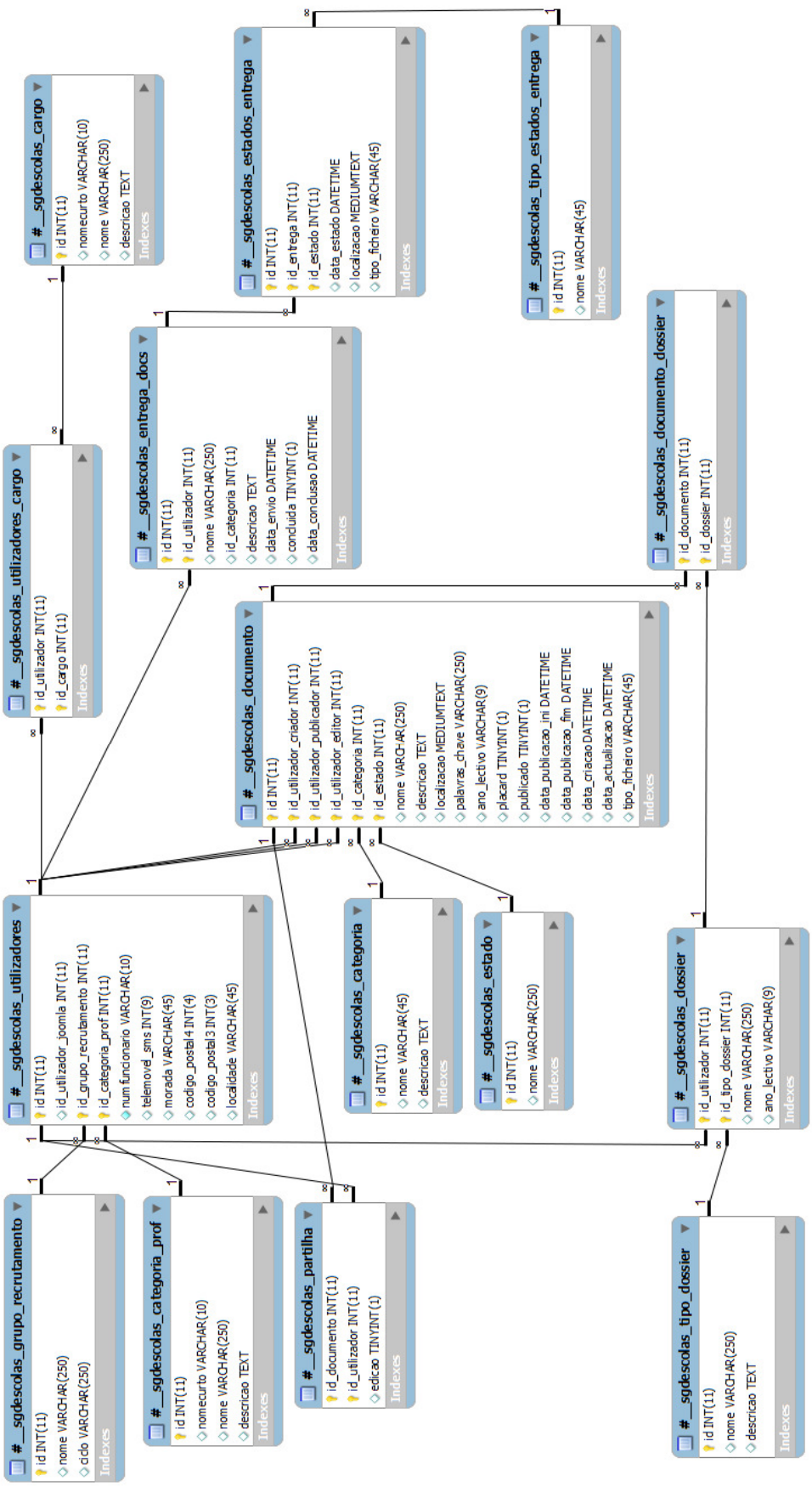
XAMPP, 2010. [Em linha] Disponível em: < http://www.apachefriends.org> [Acedido em Maio de 2010].

Zucker, B., 2003. *The Right Content Management System Will Improve Your Web ROI*. White paper. Bridgeline Software Inc. [Em linha] Disponível em: <http://www.bly.com/newsite/Pages/ CMS%20White%20Paper%202-03.pdf> [Acedido em Setembro de 2010].

ZyLAB, 2001. *Know the cost of filing your paper documents*. White paper. ZyLAB Technologies B.V. [Em linha] Disponível em: <https://www.esigtek.com/media/documents/zylab_white_paper_cost_of_filing_your_paper_documents.pdf> [Acedido em Agosto de 2010].

ANEXOS

Anexo 1. Modelo físico do componente SGDEscolas



Anexo 2. Estrutura de directórios do *back end*

```
com_sgdescolas\admin\
com_sgdescolas\admin\admin.sgdescolas.php
com_sgdescolas\admin\configuration.php
com_sgdescolas\admin\defines.php
com_sgdescolas\admin\htaccess.txt
com_sgdescolas\admin\index.html
com_sgdescolas\admin\install.sgdescolas.class.php
com_sgdescolas\admin\install.sgdescolas.php
com_sgdescolas\admin\toolbar.sgdescolas.class.php
com_sgdescolas\admin\toolbar.sgdescolas.html.php
com_sgdescolas\admin\toolbar.sgdescolas.php
com_sgdescolas\admin\assets\
com_sgdescolas\admin\assets\index.html
com_sgdescolas\admin\assets\css\
com_sgdescolas\admin\assets\css\geral.css
com_sgdescolas\admin\assets\css\index.html
com_sgdescolas\admin\assets\images\
com_sgdescolas\admin\assets\images\hdot2.gif
com_sgdescolas\admin\assets\images\icon-128-logo.png
com_sgdescolas\admin\assets\images\icon-16-backup.png
com_sgdescolas\admin\assets\images\icon-16-calendar.png
com_sgdescolas\admin\assets\images\icon-16-delete.png
com_sgdescolas\admin\assets\images\icon-16-documentos.png
com_sgdescolas\admin\assets\images\icon-16-dossiers.png
com_sgdescolas\admin\assets\images\icon-16-download.png
com_sgdescolas\admin\assets\images\icon-16-editadoc.png
com_sgdescolas\admin\assets\images\icon-16-perigo.png
com_sgdescolas\admin\assets\images\icon-16-review.png
com_sgdescolas\admin\assets\images\icon-16-sendlist.png
com_sgdescolas\admin\assets\images\icon-16-sgdescolas.png
com_sgdescolas\admin\assets\images\icon-16-share.png
com_sgdescolas\admin\assets\images\icon-16-tick.png
com_sgdescolas\admin\assets\images\icon-16-utilizadores.png
com_sgdescolas\admin\assets\images\icon-32-adicionauti.png
com_sgdescolas\admin\assets\images\icon-32-aplicar.png
com_sgdescolas\admin\assets\images\icon-32-cancelar.png
com_sgdescolas\admin\assets\images\icon-32-editadoc.png
com_sgdescolas\admin\assets\images\icon-32-editadoss.png
com_sgdescolas\admin\assets\images\icon-32-editauti.png
com_sgdescolas\admin\assets\images\icon-32-eliminadoc.png
com_sgdescolas\admin\assets\images\icon-32-eliminadoss.png
com_sgdescolas\admin\assets\images\icon-32-eliminauti.png
com_sgdescolas\admin\assets\images\icon-32-gotodoss.png
com_sgdescolas\admin\assets\images\icon-32-guardar.png
com_sgdescolas\admin\assets\images\icon-32-helpme.png
com_sgdescolas\admin\assets\images\icon-32-logo.png
com_sgdescolas\admin\assets\images\icon-32-novodoc.png
com_sgdescolas\admin\assets\images\icon-32-novodoss.png
com_sgdescolas\admin\assets\images\icon-32-removedocsdoss.png
com_sgdescolas\admin\assets\images\icon-32-review.png
com_sgdescolas\admin\assets\images\icon-32-send.png
com_sgdescolas\admin\assets\images\icon-32-senddetails.png
com_sgdescolas\admin\assets\images\icon-32-sendlist.png
com_sgdescolas\admin\assets\images\icon-32-sendstate.png
com_sgdescolas\admin\assets\images\icon-32-share.png
com_sgdescolas\admin\assets\images\icon-32-utilizadores.png
com_sgdescolas\admin\assets\images\icon-32-verdocsdoss.png
com_sgdescolas\admin\assets\images\icon-48-adicionauti.png
com_sgdescolas\admin\assets\images\icon-48-backup.png
com_sgdescolas\admin\assets\images\icon-48-desc.png
com_sgdescolas\admin\assets\images\icon-48-documentos.png
com_sgdescolas\admin\assets\images\icon-48-dossiers.png
com_sgdescolas\admin\assets\images\icon-48-editadoc.png
com_sgdescolas\admin\assets\images\icon-48-editadoss.png
com_sgdescolas\admin\assets\images\icon-48-editauti.png
com_sgdescolas\admin\assets\images\icon-48-gotodoss.png
com_sgdescolas\admin\assets\images\icon-48-logo.png
com_sgdescolas\admin\assets\images\icon-48-novodoc.png
com_sgdescolas\admin\assets\images\icon-48-novodoss.png
com_sgdescolas\admin\assets\images\icon-48-public.png
com_sgdescolas\admin\assets\images\icon-48-review.png
com_sgdescolas\admin\assets\images\icon-48-send.png
com_sgdescolas\admin\assets\images\icon-48-senddetails.png
com_sgdescolas\admin\assets\images\icon-48-sendlist.png
com_sgdescolas\admin\assets\images\icon-48-sendstate.png
com_sgdescolas\admin\assets\images\icon-48-share.png
```

```
com_sgdescolas\admin\assets\images\icon-48-user.png
com_sgdescolas\admin\assets\images\icon-48-utilizadores.png
com_sgdescolas\admin\assets\images\icon-48-verdocsdoss.png
com_sgdescolas\admin\assets\images\index.html
com_sgdescolas\admin\assets\javascript\
com_sgdescolas\admin\assets\javascript\index.html
com_sgdescolas\admin\controllers\
com_sgdescolas\admin\controllers\backup.php
com_sgdescolas\admin\controllers\default.php
com_sgdescolas\admin\controllers\documentos.php
com_sgdescolas\admin\controllers\dossiers.php
com_sgdescolas\admin\controllers\index.html
com_sgdescolas\admin\controllers\utilizadores.php
com_sgdescolas\admin\lang\
com_sgdescolas\admin\lang\pt-PT.com_sgdescolas.ini
com_sgdescolas\admin\models\
com_sgdescolas\admin\models\backup.php
com_sgdescolas\admin\models\documentos.php
com_sgdescolas\admin\models\dossiers.php
com_sgdescolas\admin\models\index.html
com_sgdescolas\admin\models\utilizadores.php
com_sgdescolas\admin\sql\
com_sgdescolas\admin\sql\index.html
com_sgdescolas\admin\sql\install.mysql.utf8.sql
com_sgdescolas\admin\sql\uninstall.mysql.utf8.sql
com_sgdescolas\admin\tables\
com_sgdescolas\admin\tables\documentos.php
com_sgdescolas\admin\tables\dossiers.php
com_sgdescolas\admin\tables\index.html
com_sgdescolas\admin\tables\utilizadores.php
com_sgdescolas\admin\views\backup\
com_sgdescolas\admin\views\backup\index.html
com_sgdescolas\admin\views\backup\view.html.php
com_sgdescolas\admin\views\backup\tmpl\
com_sgdescolas\admin\views\backup\tmpl\default.php
com_sgdescolas\admin\views\backup\tmpl\index.html
com_sgdescolas\admin\views\documentos\
com_sgdescolas\admin\views\documentos\index.html
com_sgdescolas\admin\views\documentos\view.html.php
com_sgdescolas\admin\views\documentos\tmpl\
com_sgdescolas\admin\views\documentos\tmpl\default.php
com_sgdescolas\admin\views\documentos\tmpl\default_edit.php
com_sgdescolas\admin\views\documentos\tmpl\delivery_details.php
com_sgdescolas\admin\views\documentos\tmpl\delivery_set_state.php
com_sgdescolas\admin\views\documentos\tmpl\deliverylist.php
com_sgdescolas\admin\views\documentos\tmpl\gotodoss.php
com_sgdescolas\admin\views\documentos\tmpl\index.html
com_sgdescolas\admin\views\dossiers\
com_sgdescolas\admin\views\dossiers\index.html
com_sgdescolas\admin\views\dossiers\view.html.php
com_sgdescolas\admin\views\dossiers\tmpl\
com_sgdescolas\admin\views\dossiers\tmpl\default.php
com_sgdescolas\admin\views\dossiers\tmpl\default_edit.php
com_sgdescolas\admin\views\dossiers\tmpl\index.html
com_sgdescolas\admin\views\dossiers\tmpl\verdocsdoss.php
com_sgdescolas\admin\views\sgdescolas\
com_sgdescolas\admin\views\sgdescolas\index.html
com_sgdescolas\admin\views\sgdescolas\view.html.php
com_sgdescolas\admin\views\sgdescolas\tmpl\
com_sgdescolas\admin\views\sgdescolas\tmpl\default.php
com_sgdescolas\admin\views\sgdescolas\tmpl\index.html
com_sgdescolas\admin\views\utilizadores\
com_sgdescolas\admin\views\utilizadores\index.html
com_sgdescolas\admin\views\utilizadores\view.html.php
com_sgdescolas\admin\views\utilizadores\tmpl\
com_sgdescolas\admin\views\utilizadores\tmpl\default.php
com_sgdescolas\admin\views\utilizadores\tmpl\default_add.php
com_sgdescolas\admin\views\utilizadores\tmpl\default_edit.php
com_sgdescolas\admin\views\utilizadores\tmpl\index.html
```

Anexo 3. Estrutura de directórios do *front end*

```
com_sgdescolas\site\
com_sgdescolas\site\configuration.php
com_sgdescolas\site\index.html
com_sgdescolas\site\sgdescolas.php
com_sgdescolas\site\assets\
com_sgdescolas\site\assets\index.html
com_sgdescolas\site\assets\css\
com_sgdescolas\site\assets\css\geral.css
com_sgdescolas\site\assets\css\index.html
com_sgdescolas\site\assets\images\
com_sgdescolas\site\assets\images\hdot2.gif
com_sgdescolas\site\assets\images\icon-128-logo.png
com_sgdescolas\site\assets\images\icon-16-backup.png
com_sgdescolas\site\assets\images\icon-16-calendar.png
com_sgdescolas\site\assets\images\icon-16-delete.png
com_sgdescolas\site\assets\images\icon-16-documentos.png
com_sgdescolas\site\assets\images\icon-16-dossiers.png
com_sgdescolas\site\assets\images\icon-16-download.png
com_sgdescolas\site\assets\images\icon-16-editadoc.png
com_sgdescolas\site\assets\images\icon-16-perigo.png
com_sgdescolas\site\assets\images\icon-16-review.png
com_sgdescolas\site\assets\images\icon-16-sendlist.png
com_sgdescolas\site\assets\images\icon-16-sgdescolas.png
com_sgdescolas\site\assets\images\icon-16-share.png
com_sgdescolas\site\assets\images\icon-16-tick.png
com_sgdescolas\site\assets\images\icon-16-utilizadores.png
com_sgdescolas\site\assets\images\icon-32-adicionauti.png
com_sgdescolas\site\assets\images\icon-32-aplicar.png
com_sgdescolas\site\assets\images\icon-32-cancelar.png
com_sgdescolas\site\assets\images\icon-32-editadoc.png
com_sgdescolas\site\assets\images\icon-32-editadoss.png
com_sgdescolas\site\assets\images\icon-32-editauti.png
com_sgdescolas\site\assets\images\icon-32-eliminadoc.png
com_sgdescolas\site\assets\images\icon-32-eliminadoss.png
com_sgdescolas\site\assets\images\icon-32-eliminauti.png
com_sgdescolas\site\assets\images\icon-32-gotodoss.png
com_sgdescolas\site\assets\images\icon-32-guardar.png
com_sgdescolas\site\assets\images\icon-32-helpme.png
com_sgdescolas\site\assets\images\icon-32-logo.png
com_sgdescolas\site\assets\images\icon-32-novodoc.png
com_sgdescolas\site\assets\images\icon-32-novodoss.png
com_sgdescolas\site\assets\images\icon-32-removedocsdoss.png
com_sgdescolas\site\assets\images\icon-32-review.png
com_sgdescolas\site\assets\images\icon-32-send.png
com_sgdescolas\site\assets\images\icon-32-senddetails.png
com_sgdescolas\site\assets\images\icon-32-sendlist.png
com_sgdescolas\site\assets\images\icon-32-sendstate.png
com_sgdescolas\site\assets\images\icon-32-share.png
com_sgdescolas\site\assets\images\icon-32-utilizadores.png
com_sgdescolas\site\assets\images\icon-32-verdocsdoss.png
com_sgdescolas\site\assets\images\icon-48-adicionauti.png
com_sgdescolas\site\assets\images\icon-48-backup.png
com_sgdescolas\site\assets\images\icon-48-desc.png
com_sgdescolas\site\assets\images\icon-48-documentos.png
com_sgdescolas\site\assets\images\icon-48-dossiers.png
com_sgdescolas\site\assets\images\icon-48-editadoc.png
com_sgdescolas\site\assets\images\icon-48-editadoss.png
com_sgdescolas\site\assets\images\icon-48-editauti.png
com_sgdescolas\site\assets\images\icon-48-gotodoss.png
com_sgdescolas\site\assets\images\icon-48-logo.png
com_sgdescolas\site\assets\images\icon-48-novodoc.png
com_sgdescolas\site\assets\images\icon-48-novodoss.png
com_sgdescolas\site\assets\images\icon-48-public.png
com_sgdescolas\site\assets\images\icon-48-review.png
com_sgdescolas\site\assets\images\icon-48-send.png
com_sgdescolas\site\assets\images\icon-48-senddetails.png
com_sgdescolas\site\assets\images\icon-48-sendlist.png
com_sgdescolas\site\assets\images\icon-48-sendstate.png
com_sgdescolas\site\assets\images\icon-48-share.png
com_sgdescolas\site\assets\images\icon-48-user.png
com_sgdescolas\site\assets\images\icon-48-utilizadores.png
com_sgdescolas\site\assets\images\icon-48-verdocsdoss.png
com_sgdescolas\site\assets\images\index.html
com_sgdescolas\site\assets\images\mail_find-256.png
com_sgdescolas\site\assets\javascript\
com_sgdescolas\site\assets\javascript\index.html
```

```
com_sgdescolas\site\assets\javascript\wz_tooltip.js
com_sgdescolas\site\controllers\
com_sgdescolas\site\controllers\default.php
com_sgdescolas\site\controllers\documentos.php
com_sgdescolas\site\controllers\dossiers.php
com_sgdescolas\site\controllers\index.html
com_sgdescolas\site\controllers\public.php
com_sgdescolas\site\controllers\share.php
com_sgdescolas\site\controllers\utilizador.php
com_sgdescolas\site\lang\
com_sgdescolas\site\lang\pt-PT.com_sgdescolas.ini
com_sgdescolas\site\models\
com_sgdescolas\site\models\documentos.php
com_sgdescolas\site\models\dossiers.php
com_sgdescolas\site\models\index.html
com_sgdescolas\site\models\utilizador.php
com_sgdescolas\site\views\documentos\
com_sgdescolas\site\views\documentos\index.html
com_sgdescolas\site\views\documentos\metadata.xml
com_sgdescolas\site\views\documentos\view.html.php
com_sgdescolas\site\views\documentos\tmpl\
com_sgdescolas\site\views\documentos\tmpl\default.php
com_sgdescolas\site\views\documentos\tmpl\default.xml
com_sgdescolas\site\views\documentos\tmpl\default_add.php
com_sgdescolas\site\views\documentos\tmpl\default_edit.php
com_sgdescolas\site\views\documentos\tmpl\default_share.php
com_sgdescolas\site\views\documentos\tmpl\delivery.php
com_sgdescolas\site\views\documentos\tmpl\delivery.xml
com_sgdescolas\site\views\documentos\tmpl\delivery_details.php
com_sgdescolas\site\views\documentos\tmpl\delivery_mydoc.php
com_sgdescolas\site\views\documentos\tmpl\delivery_review.php
com_sgdescolas\site\views\documentos\tmpl\deliverylist.php
com_sgdescolas\site\views\documentos\tmpl\deliverylist.xml
com_sgdescolas\site\views\documentos\tmpl\gotodoss.php
com_sgdescolas\site\views\documentos\tmpl\gotodoss.xml
com_sgdescolas\site\views\documentos\tmpl\index.html
com_sgdescolas\site\views\dossiers\
com_sgdescolas\site\views\dossiers\index.html
com_sgdescolas\site\views\dossiers\metadata.xml
com_sgdescolas\site\views\dossiers\view.html.php
com_sgdescolas\site\views\dossiers\tmpl\
com_sgdescolas\site\views\dossiers\tmpl\default.php
com_sgdescolas\site\views\dossiers\tmpl\default.xml
com_sgdescolas\site\views\dossiers\tmpl\default_edit.php
com_sgdescolas\site\views\dossiers\tmpl\index.html
com_sgdescolas\site\views\public\
com_sgdescolas\site\views\public\index.html
com_sgdescolas\site\views\public\metadata.xml
com_sgdescolas\site\views\public\view.html.php
com_sgdescolas\site\views\public\tmpl\
com_sgdescolas\site\views\public\tmpl\default.php
com_sgdescolas\site\views\public\tmpl\default.xml
com_sgdescolas\site\views\public\tmpl\default_description.php
com_sgdescolas\site\views\public\tmpl\index.html
com_sgdescolas\site\views\sgdescolas\
com_sgdescolas\site\views\sgdescolas\index.html
com_sgdescolas\site\views\sgdescolas\metadata.xml
com_sgdescolas\site\views\sgdescolas\view.html.php
com_sgdescolas\site\views\sgdescolas\tmpl\
com_sgdescolas\site\views\sgdescolas\tmpl\default.php
com_sgdescolas\site\views\sgdescolas\tmpl\default.xml
com_sgdescolas\site\views\sgdescolas\tmpl\index.html
com_sgdescolas\site\views\share\
com_sgdescolas\site\views\share\index.html
com_sgdescolas\site\views\share\metadata.xml
com_sgdescolas\site\views\share\view.html.php
com_sgdescolas\site\views\share\tmpl\
com_sgdescolas\site\views\share\tmpl\default.php
com_sgdescolas\site\views\share\tmpl\default.xml
com_sgdescolas\site\views\share\tmpl\default_edit_share.php
com_sgdescolas\site\views\share\tmpl\index.html
com_sgdescolas\site\views\utilizador\
com_sgdescolas\site\views\utilizador\index.html
com_sgdescolas\site\views\utilizador\metadata.xml
com_sgdescolas\site\views\utilizador\view.html.php
com_sgdescolas\site\views\utilizador\tmpl\
com_sgdescolas\site\views\utilizador\tmpl\default.php
com_sgdescolas\site\views\utilizador\tmpl\default.xml
com_sgdescolas\site\views\utilizador\tmpl\index.html
```