



Universidade do Minho
Licenciatura em Engenharia Informática

Reports Template Manager

Relatório de Laboratórios de Informática IV

Grupo N.º 10:

Filipe Adão Ferreira da Costa, N.º 38242

Filipe Correia Garcia, N.º 40680

Rui José Coelho da Silva, N.º 40650

Supervisão:

Prof. Orlando Belo

4 de Julho, 2008

A cerca deste documento

O presente documento descreve o relatório de **Laboratórios de Informática IV**, inserido no projecto proposto pelo grupo e intitulado **Reports Template Manager**.

Este documento foi escrito pelo grupo n.º 10, composto por Filipe Costa, Filipe Garcia e Rui Silva, e foi supervisionado por Orlando Belo.

Resumo

O mundo empresarial da actualidade segue a vanguarda da tecnologia e expande o seu mercado com a globalização sócio-económica, crescendo de forma exponencial. Inerente a este desenvolvimento advém uma competitividade muito activa, subindo a fasquia das empresas que pretendem vingar no mercado. A complexidade da estrutura de negócio e as necessidades de controlo de informação tornam-se exigentes ao ponto das soluções manuais de gestão serem insuficientes e até inutilizáveis. Surgem, assim, soluções informáticas denominadas por sistemas de Enterprise Resource Planning (ERP), que integram toda a informação e processos de uma empresa num sistema unificado.

No entanto, para uma melhor análise são necessários relatórios específicos da informação e as ferramentas usadas pelas soluções ERP são de difícil manuseamento ou extremamente caras. A complexidade que lhe é inerente causa problemas a nível de desenvolvimento de novos relatórios ou até a simples alteração visual dos mesmos. Nesta linha é apresentado o **Reports Template Manager**, um sistema para criação de relatórios.

O **Reports Template Manager** distingue-se pela sua fácil configuração e personalização consoante as necessidades da empresa.

Este relatório apresenta a concepção do **Reports Template Manager**, a qual pretende melhorar certos aspectos comparativamente com as outras ferramentas existentes, como a rapidez de desenvolvimento, flexibilidade e a personalização dos relatórios.

Conteúdo

Introdução	1
Contextualização do Projecto	1
Objectivos do Projecto	1
Organização do Documento	2
O Projecto	3
Enquadramento do Projecto	3
Estado da Arte	3
Objectivos e Contributos do Projecto	3
Desenvolvimento	5
Linguagens e Ferramentas Utilizadas	5
Análise	10
Desenho	10
Arquitectura do Sistema	10
Modelo de Dados	12
Esquema Conceptual da Base de Dados	13
Implementação	14
Conclusão	16
Síntese do Relatório	16
Ponto da Situação	16
Trabalho Futuro	16
Referências Bibliográficas	17
Referências Web	18

Introdução

Neste primeiro capítulo apresenta-se uma breve descrição do projecto realizado, bem como a sua importância na manipulação de relatórios numa empresa. São ainda mencionados os seus objectivos, bem como a organização deste documento.

Contextualização do Projecto

O sucesso de uma empresa está relacionado com a sua capacidade de organização, integração, optimização dos processos e análise dos dados. Uma interligação coordenada e eficaz entre os vários departamentos releva-se, contudo, bastante difícil, sendo tão ou mais complicada quanto maior for a empresa e os dados a serem analisados para um melhor rendimento da mesma.

A partilha de informação e conhecimento dos dados fornecidos pelos vários departamentos é analisado através de extensos relatórios analíticos que se alteram consoante as necessidades da empresa, o que se torna um problema quer na criação, quer na alteração de tais relatórios fornecidos pelo ERP que usam.

Em virtude da grande quantidade de relatórios que uma empresa necessita para satisfazer as suas análises, surge o **Reports Template Manager**, uma aplicação que permite tanto aos programadores como às empresas a criação e edição fácil e intuitiva de relatórios analíticos.

As funcionalidades oferecidas pelo **RTM** inserem-se na criação de relatórios através de uma aplicação “*drag and drop*” de componentes pré-definidos pelo programador. Podendo-se assim criar relatórios previamente inexistente ou a simples alteração dos existentes. É ainda oferecida uma API ao programador para a criação dos mesmos.

As principais vantagens do **RTM**, em relação a outras aplicações do mesmo tipo surge ao nível da simples criação de relatórios por parte da empresas, assim como uma API acessível que previne a escrita de centenas de linhas de código por parte do programador.

Objectivos do Projecto

O objectivo é a criação de um projecto assente em novas tecnologias e usando uma estrutura modular de modo a permitir que este seja usado como um *plugin*.

O *plugin* a desenvolver serve para colmatar algumas lacunas detectadas em outras aplicações existentes, assim como saciar as necessidades das empresas na criação de relatórios específicos, mas também a alteração dos fornecidos pelos ERPs para terem a sua uma imagem.

A criação deste *plugin* tornou-se também importante para nós, para o podermos integrar numa aplicação de Gestão Comercial desenvolvida por nós, onde a criação de tais relatórios tornou-se um processo dispendioso tanto a nível financeiro como de tempo.

Os objectivos considerados mais importantes são então:

Fornecer um *plugin* constituído por uma **aplicação web** para a criação e edição dos relatórios por parte das empresas e uma **API** para os programadores desenvolverem relatórios a fornecer às empresas de forma rápida e simples.

Outras Tarefas

Para além destes objectivos propõe-se a criação de uma página *web*, para efeitos de demonstração, documentação para o uso da aplicação e *download* da mesma.

Organização do Documento

Além do presente capítulo introdutório, este relatório está organizado da seguinte forma:

No capítulo 2 é apresentado o projecto, referindo o seu enquadramento e o estado da arte, assim como os objectivos e a contribuição do mesmo para o mundo empresarial.

O capítulo 3 descreve as fases de desenvolvimento do projecto. As linguagens e ferramentas utilizadas, uma análise detalhada dos problemas observados e solução destes e por fim a sua implementação.

Por fim, são apresentadas as conclusões sobre o projecto são descritas no capítulo 4.

O Projecto

Este capítulo é dedicado ao projecto em questão, nomeadamente o seu enquadramento, motivação da sua realização, os objectivos que se pretendem atingir e a relação deste com os requisitos das empresas.

Enquadramento do Projecto

Para uma melhor compreensão do projecto **Reports Template Manager**, ficam descritos alguns pontos tais como, o estado da arte no âmbito do mesmo e a plataforma construída para suporte da construção e edição de relatórios.

Estado da Arte

Originalmente em empresas que usavam algum tipo de computação, o sistema era arranjado da forma cliente/servidor. Um grande servidor que servia as aplicações a terminais. Com a evolução dos PCs a nível de processador e armazenamento e a sua comercialização massiva que levou ao baixo custo dos mesmos, os programas passaram a ser criados de uma forma diferente.

O processamento passou a ser feito de uma forma mais distribuída e por algum tempo sem consequências. Este percurso levou a que as aplicações se comesçassem a tornar cada vez mais pesadas e não aumentava a experiência do utilizador. Também, por outro lado, a pouca modularidade dos programas não permitia grandes actualizações e uma difícil manutenção. As equipas de programadores tornavam-se cada vez maiores e era necessário muito esforço para a criação de algo bastante simples.

Uma vez que com numa aplicação web o processamento é todo feito no servidor, o utilizador final apenas requer uma máquina minimamente usável. Da mesma forma que consulta o seu *webmail* ou pesquisa na internet, poderá também criar os seus relatórios de forma simples. Isto torna-se possível devido ao grande melhoramento das linguagens de programação pensadas para funcionar em grande escala e sem grande consumo, como já falamos ao longo deste relatório. Actualmente a tecnologias disponíveis a nível da web são bastante poderosas, e deixou de ser possível apenas criar paginas estáticas que só mostram a informação num sentido.

Uma vez que o utilizador já está familiarizado com o conceito de *browser* e páginas de internet, sente-se bastante mais confortável. Torna-se assim mais simples introduzir um programa como este a nível empresarial. Com um conhecimento prévio do que é tratado pela aplicação - neste caso a criação de documentos - uma breve introdução à informática e uma pequena demonstração, o utilizador comum fica apto para realizar as tarefas necessárias.

A nível empresarial estes são conceitos bastante importantes, equivalem a um aumento de produção e uma poupança a nível de formação quer monetária quer temporal.

Objectivos e Contributos do Projecto

Inicialmente ao escolhermos a plataforma de desenvolvimento e sendo esta *open source*, sempre tivemos a ideia de devolver algo à comunidade. Assim decidimos que este também seria *open source*. Neste

sentido, em vez de criar uma aplicação independente, pensamos que se enquadraria melhor como um *plugin* adicional. Os *plugins* são bastante utilizados e na nova versão de *symfony* estão muito bem integrados, o que satisfaz os nossos objectivos.

De uma necessidade e a lacuna da mesma, ponderamos e crê-mos que também mais pessoas iriam beneficiar deste *plugin*. Então aliámos a necessidade de rapidamente obter um relatório de actividades, num formato leve e universal. Com isto fomos incentivados a optar pelo TCPDF como ferramenta de geração final de *pdf*, os seus benefícios vão ser analisados mais tarde. Os ficheiros *pdf* são uma solução bastante económica, pois um formato com uma entropia tão alta como do *pdf* é sempre uma boa escolha quando se pensa em quantidades empresariais , e sendo um documento bastante bem aceite a nível global, pensamos que este seria indiscutivelmente o formato a usar.

Possivelmente com a divulgação do **RTM** na comunidade esperamos receber bastante *feedback* e nada melhor que isso para conseguir compreender o quanto este projecto devolve à comunidade e realmente se se torna numa ferramenta útil não só para nós como para os demais.

Desenvolvimento

O **RTM** é um projecto que se encontra dividido em duas partes fundamentais:

Frontend

Apresentado com uma aplicação *web*, é responsável por fornecer uma interface para um simples utilizador criar e editar os seus relatórios de uma forma fácil e intuitiva.

Application Programming Interface (API)

É fornecida uma API para o programador, usando o **RTM** como um *plugin* na sua aplicação, poder criar e fornecer relatórios analíticos da sua aplicação.

Linguagens e Ferramentas Utilizadas

Esta secção apresenta alguns conceitos necessários para a compreensão deste relatório.

PHP 5

PHP (Hypertext Preprocessor) é uma linguagem de *script* orientada a objectos, livre e muito utilizada para gerar conteúdos dinâmicos. O código PHP nunca é visto pelo usuário uma vez que serve para gerar código HTML. Uma vez que é tão amplamente utilizado, existe imensa documentação e mesmo sendo esta uma linguagem de geração de código HTML existem bastantes outras plataformas/ferramentas que nos permitem a geração do próprio PHP. A utilizada por nós (Symfony) será revista mais a frente.

Algumas das muitas vantagens encontradas no PHP são:

- Licença gratuita e possibilidade de usar uma plataforma (SO) gratuita também;
- Velocidade de processamento, Eficiência e Segurança bastante otimizada;
- Código fonte livre;
- *Exceptions* (para controlo de fluxo)
- É a linguagem Web mais popular e que mais cresce no mercado;
- Possibilita a utilização dos maiores e mais utilizados sistemas de base de dados (PostgreSQL, FrontBase, SQLite, MySQL, IBM DB2, ODBC, Unix DBM, entre outros) sem necessitar de configuração externa.
- Uma vez que tem uma grande comunidade, esta sempre em actualização as suas falhas são corrigidas rapidamente.

TCPDF

É uma biblioteca Open Source do PHP. Esta permite-nos criar documentos PDF on-the-fly, isto é, de forma dinâmica.

A possibilidade de usar a biblioteca em qualquer sistema operativo sem necessitar de instalar algo externo torna-se bastante interessante, uma vez que outros sistemas seriam complicados de configurar. Também o facto de gerar o documento pdf de forma dinâmica, sem a necessidade de ter ficheiros intermédios que sejam necessários compilar para obter o documento final é bastante útil quando se pensa que assim estamos a poupar bastante espaço bem como capacidade de processamento.

Algumas razões pelas quais optamos pelo TCPDF:

- Não necessita de bibliotecas externas para as funções base.
- Suporta ISO, UTF-8 Unicode e línguas Right-To-Left.
- Possibilidade de encriptação de documentos
- Inclui gráficos e transformações morfológicas
- É possível incluir código HTML
- Fácil criação de bookmarks
- Suporta vários tipos de imagens
- Torna possível a criação de page-breaks, page numbering, page groups, line break e text justification de forma natural.
- Fácil criação de links
- Com a biblioteca Zlib suporta compressão

Doctrine

Doctrine é um ORM (Object Relational Mapper) para PHP. Este assenta no topo de um poderoso DBAL (Database Abstraction Layer) PHP. Uma das suas habilidades chave é a possibilidade de criar *queries* de base de dados em um ambiente OO (Object Oriented) com um dialecto SQL (DQL), tendo sido este inspirado em Hibernate. Isto proporciona aos programadores uma poderosa alternativa ao SQL mantendo uma máxima flexibilidade sem criar duplicação de código.

Algumas das funcionalidades são:

- Rápido, uma vez que executa menos *queries* e o PDO (PHP Data Objects) retorna os resultados mais rapidamente;
- Tem a possibilidade de suportar herança;
- Suporta de raiz relações n-para-n assim como complexos *joins* entre tabelas com apenas uma instrução;
- População de objectos para *queries* arbitrárias;

SQL

SQL (Structured Query Language) é reconhecido como uma linguagem *standard*, desenhada para organizar, gerir e retornar dados de uma base de dados. É essencialmente uma linguagem de programação para base de dados relacionais e nas últimas décadas os servidores de SQL tem sido cada vez mais fiáveis, escaláveis e de custo reduzido.

As principais vantagens de SQL são:

- Portabilidade (entre aplicações);
- *Standard* industrial;
- Definição de dados dinâmicos (suporta *queries* alfanuméricas de uma forma otimizada);
- Outras vantagens são:
 - *Reliability*: Com um servidor de SQL os clientes não "falam" directamente com a tabelas mas sim com um servidor inteligente de acesso aos dados. Em sua vez, este, lê e escreve os dados das tabelas. Se uma máquina de um cliente falha ou a rede tem problemas, isto não afecta em nenhuma forma a gestão da base de dados. Pelo contrário o gestor de base de dados garante que se uma transacção não pode ser terminada ou se os dados foram apenas parcialmente enviados a base de dados continua inalterada.
 - *Data Integrity*: A integridade da base de dados é melhorada com o uso de *triggers*. Podem ser usados sempre que um *record* é adicionado, actualizado ou removido.
 - *Better Performance*: Uma vez que a filtragem de dados é feita no servidor, não temos que transmitir dados inúteis que de nada vão servir ao utilizador final. Isto pode afectar em duas formas. Primeiramente o servidor é altamente optimizado e vai efectuar a filtragem de forma mais eficiente. Em segundo a rede tem um uma menor carga e pode trabalhar de forma mais eficiente
 - *Scalability*: Um sistema baseado num servidor de ficheiros é desenhado para pequenos *workgroups* e é escalado em média para 10 clientes concorrentes. Tanto o servidor como cliente SQL é uma arquitectura que suporta centenas e mesmo milhares de a acessos concorrentes sem significativos problemas de performance.

MVC

MVC (Model-View-Controller) é uma forma de programação modular que leva este conceito ao extremo. Uma aplicação modular pode de forma dinâmica carregar e descarregar um modulo em tempo de execução, separando assim aplicações.

É comum dividir a aplicação em camadas separadas: apresentação (*interface*), domínio e acesso a dados. Em MVC a camada de apresentação também é separada da *view* e do *controller*.

- *Model*: A representação "domínio" específica da informação em que a aplicação opera. Por exemplo, aluno, professor e turma fazem parte do domínio de um sistema académico. É comum haver confusão pensando que *Model* é um outro nome para a camada de domínio. Lógica de domínio adiciona sentido à dados crus (por exemplo, calcular se hoje é aniversário do usuário, ou calcular o total de impostos e fretes sobre um determinado carrinho de compras). Muitas aplicações usam um mecanismo de armazenamento persistente (como banco de dados) para armazenar dados. MVC não cita

especificamente a camada para acesso aos dados, porque subentende-se que estes métodos estariam encapsulados pelo *Model*.

- *View*: "Renderiza" o *Model* em uma forma específica para a interação, geralmente uma interface de usuário.
- *Controller*: Processa e responde a eventos, geralmente acções do usuário, e pode invocar alterações no *Model*.

JavaScript

JavaScript é uma linguagem de *script* normalmente utilizada para o *client-side* em aplicações web. É uma linguagem dinâmica, fracamente tipada, baseada em *prototypes* com funções *first-class*. Foi influenciada em muitas linguagens como C e java mas no fim, o seu uso tornou-se bastante simples para os programadores.

Algumas razões para o seu uso são:

- *Syntax* similar a C. Uma vez que C é a linguagem de programação mais usada no mundo, usar uma *syntax* similar reduz a *learning curve*.
- OO *syntax* imita Java. Tem uma abordagem aos objectos muito parecida com a de java.
- Omnipresente nos *scripts* de HTML. Sendo uma linguagem que está suportada em todos os grandes *browsers*. Torna-se assim uma linguagem importante mesmo para a criação de pequenas funcionalidade como validação ou alertas que todos os *sites* necessitam mais cedo ou mais tarde.
- AJAX. No mundo da *Web 2.0* é essencial que um programador entenda Javascript de forma consolidada.
- Adobe Air, Google Gear, Mozilla Rhino são sistemas que incorporam javascript para criar algumas das maiores inovações actualmente na *web*.
- Aceitação. Mesmo sendo um filho da netscape e Javascript sendo uma marca registada da Sun, Javascript tem uma grande aceitação por parte quer dos usuários, quer dos programadores assim como dos criadores de *browsers*.

Para a criação deste projecto foi usada uma biblioteca chamada jQuery que passamos a descrever.

jQuery

jQuery é uma biblioteca rápida e concisa que simplifica a forma como percorremos os documentos *HTML*, gerimos eventos, criamos animações e adicionamos interacções de AJAX. Está criado para mudar a forma como programamos com Javascript.

Algumas das vantagens do jQuery são:

- *Chainability*: Porque os objectos de jQuery tem a funcionalidade de se "colarem" uns aos outros assim como de retornarem também objectos jQuery

- Uso de selectores CSS e operadores XPath. Uma vez que envia mensagens a objectos, tem implementado também a funcionalidade de selector no método \$.
- Fácil desenvolvimento de *plugins*. É bastante produtivo desenvolver *plugins* com o jQuery. Torna-se bastante fácil adicionar funcionalidade aos métodos já criados
- *Automatic looping*: É possível criar ciclos a partir de elementos DOM de um *array* e aplicar o método desejado. Em maior parte dos casos um *foreach* é o suficiente e bastante mais simples de usar que um iterador.
- Cria-se a si próprio. Enquanto o jQuery foi ficando maduro, tornou-se mais fácil criar *plugins* em cima da arquitectura já existente.

Symfony Framework

Symfony é uma *framework* web escrita em PHP que segue o paradigma MVC. Distribuído sobre a licença MIT, symfony é um projecto livre.

Algumas das razões pelas quais escolhemos symfony são:

- Ajax e Javascript;
- Interface de administração instantânea. Geração de CRUD (Create, Read, Update, Delete) e *admin interface*;
- *Magic Urls* usando o sistema de *routing*;
- *Form Handling*. Fácil preenchimento, validação e criação de *forms*;
- *Debugging* a partir do browser;
- Internacionalização e Localização (i18n & L10n);
- *Caching*;
- O ambiente de desenvolvimento;
- DRY (Don't Repeat Yourself). Uso de *partials*, *components*, *includes*, etc;
- Ligação a outras *frameworks*.

SVN

SVN ou Subversion é um sistema automatizado de controlo de versões evoluído do antigo CVS. É constituído por um servidor onde se localiza o repositório e vários clientes que vão actualizando e revendo a *working copy* (WC) actual.

Alguns pontos fortes de SVN são:

- *Backup* e *Restore* do projecto, caso algum ou todos programadores acidentalmente tenham problemas com a sua WC, terão sempre um sítio para o ir buscar. *Synchronization* permite que várias pessoas utilizem o sistema e estejam de forma sincronizada a trabalhar.

- "*Undo*" a curto e a longo prazo. Enquanto se trabalha num ficheiro várias vezes seguimos uma linha de pensamento que no final não serve para produção. Com o SVN é possível ir buscar a ultima WC ou andar para trás no tempo e ir buscar uma WC mais antiga.
- Função de *tracking*. O SVN também nos permite deixar pequenas mensagens a explicar o porquê de uma alteração. Estas mensagens são tão importantes como os comentários ao largo do programa.
- *Branching e merging*. Estando a criar os ficheiros numa *sandbox*, podemos sempre criar vários ramos, seguir em distintas formas de resolver o mesmo problema, modificar areas de forma isolada e controlar o seu desenvolvimento. No final temos a possibilidade de escolher a mais correcta.

Para este projecto estamos a usar SVN integrado uma plataforma chamada Trac que será revista de seguida.

Trac

O Trac (Integrated SCM & Project Management) é uma aplicação web que permite gerir um projecto de desenvolvimento de software.

Através de uma interface minimalista integra um sistema de *ticket reporting* bastante personalizável, associando tarefas aos programadores. A sua integração com o Subversion permite uma fácil e rápida navegação no código fonte do projecto, sendo ainda possível visualizar os ficheiros em revisões passadas.

Outra das potencialidades fornecidas pelo Trac e muito utilizada no projecto referido neste relatório, é a integração ao conteúdo Wiki, permitindo uma integração de toda a documentação do projecto com a lista de tarefas e código fonte.

Análise

Os requisitos funcionais do **RTM** são resolver os problemas encontrados em outras aplicações semelhantes.

Existe uma grande dificuldade no mundo empresarial em ter uma imagem própria nos seus documentos assim como a criação específica de relatórios analíticos. Como resolução é então necessário fornecer uma forma simples e intuitiva para a criação e edição desses relatórios.

Outro requisito importante é a criação dos mesmos por parte dos programadores, pois as APIs fornecidas pelas aplicações existentes são complexas, morosas e de difícil implementação.

Desenho

Esta secção apresenta a arquitectura do sistema, o seu modelo de dados, bem como o esquema conceptual da base de dados.

Arquitectura do Sistema

O **RTM** é organizado em duas partes distintas. Numa a visão de um simples utilizador e noutra a visão por parte do programador.

O diagrama UML de *use-cases* da figura 3.1 ilustra as funcionalidades às quais um utilizador do **RTM** terá acesso.

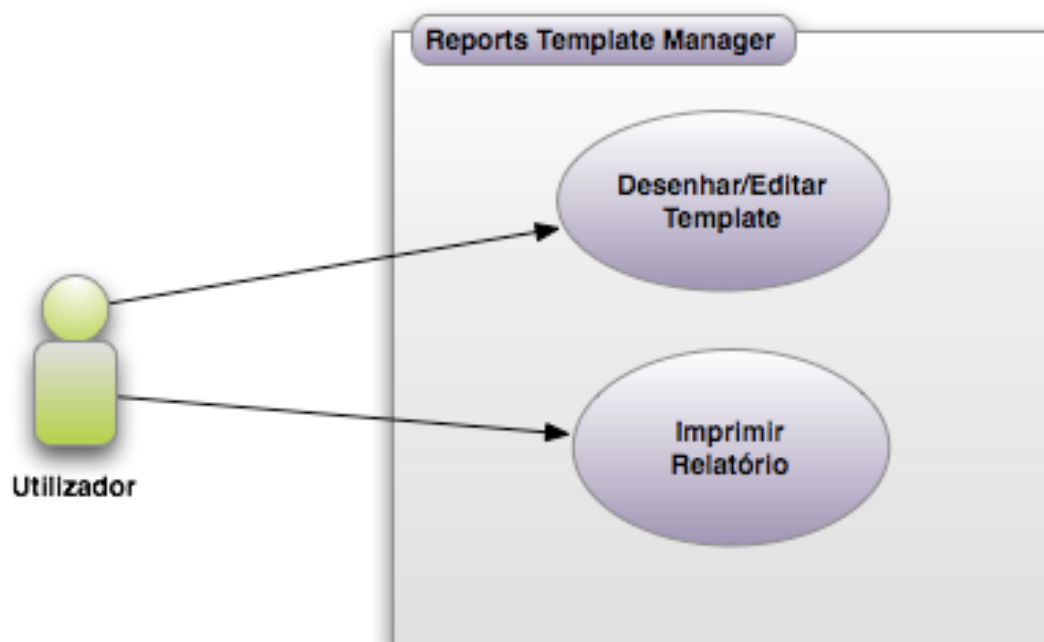


Figura 3.1: Diagrama de Use-Cases para o utilizador do RTM.

Segue-se uma breve descrição de cada um:

Desenhar/Editar Template

O utilizador pode criar um *template* à sua medida, com a informação que deseja ver analisada ou então editar para incrementar ou decrementar a informação visível ou apenas mudar a imagem do mesmo.

Imprimir Relatório

O **RTM** fornece ao utilizador uma listagem de todos os relatórios existente, podendo o utilizador navegar por entre eles e imprimir o desejado.

Na figura 3.2 é apresentado o diagrama de *use-cases* para o programador de uma outra aplicação usando o *plugin RTM*.

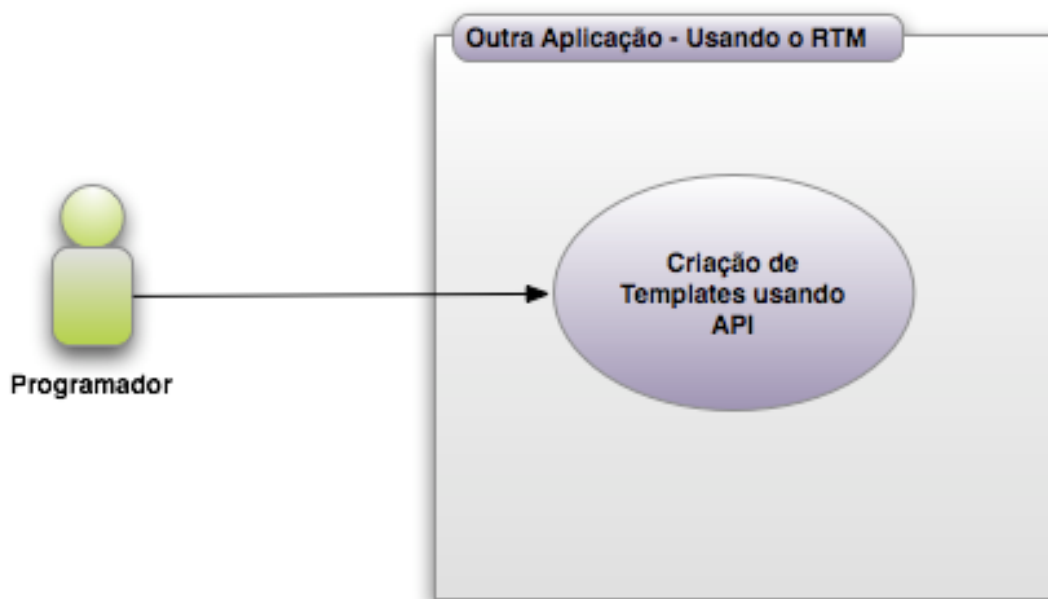


Figura 3.2: Diagrama de Use-Cases para o programador de outra aplicação.

Criação de Templates Usando a API

Um programador, tendo o plugin do **RTM**, no seu projecto poderá usar a **API** fornecida para a criação de *templates* para a sua aplicação.

Modelo de Dados

Na figura 3.3 pode-se visualizar o modelo de dados utilizador pelo **RTM**.

Um *utilizador* representa uma empresa, que usa o *frontend* do **Reports Template Manager** para a criação, edição, visualização ou remoção de *templates* constituídos por *header*, *components* e *company*.

Por outro lado, o *programador*, após a instalação do *plugin* poderá usar a *API* fornecida para a criação de *templates*.

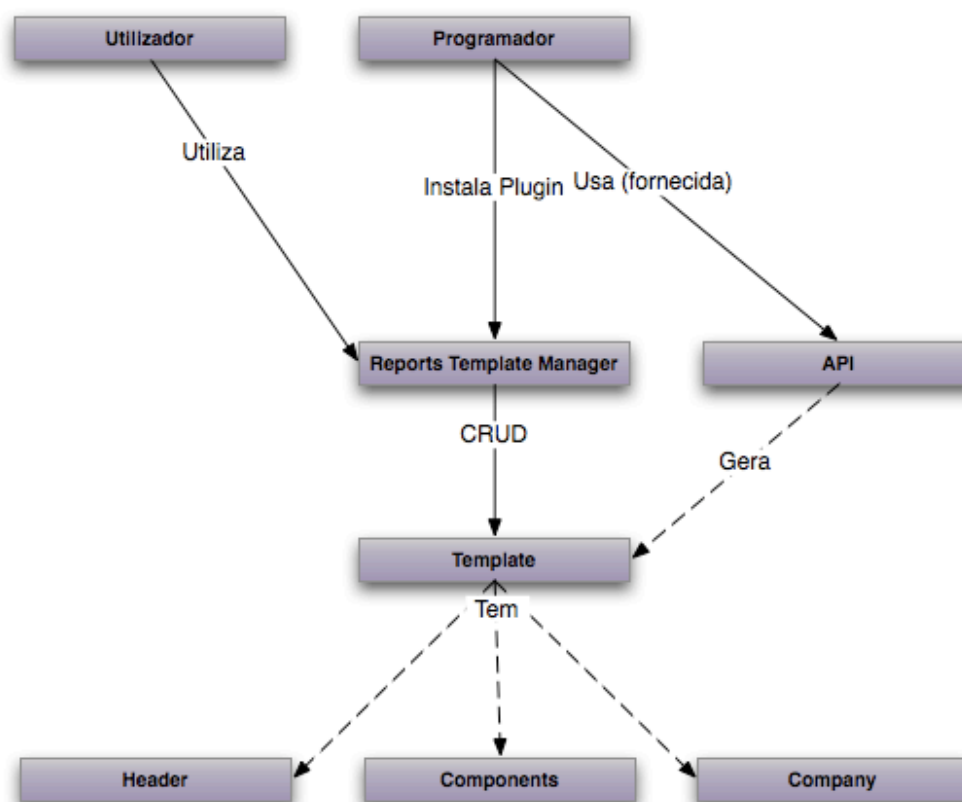


Figura 3.3: Modelo de dados.

Esquema Conceptual da Base de Dados

Logo a seguir à definição das funcionalidades da aplicação, o esquema conceptual da base de dados foi uma operação importante na concepção do sistema para que esta se torna-se numa base de dados simples. As tabelas que constituem o sistema são as seguintes:

Template

Esta tabela é usada para manter a informação de todos os *templates* criados. É constituída por um *registo da tabela header*, um da *tabela company* e um *name*.

Header

Para armazenar as posições relativa aos *headers*.

Company

Para armazenar as posições relativa as *companies*.

Template Component

Esta tabela contém os *components* que estão associados a um *template*, um registo da *tabela component*, e as suas coordenadas.

Component

Esta tabela contém os *components*, tendo um *name*.

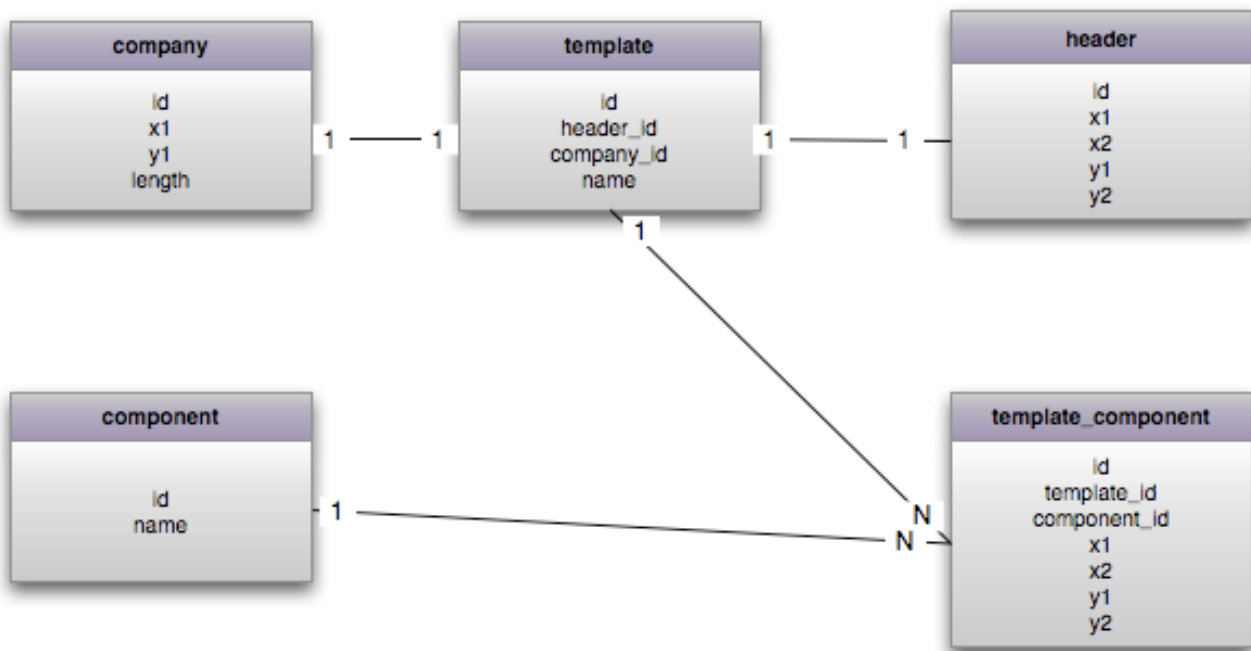


Figura 3.4: Esquema Conceptual da Base de Dados do **RTM**.

Implementação

Esta secção apresenta o trabalho realizado, acompanhado por algumas imagens que exemplificam.

Id	SfReportTemplateHeader	SfReportTemplateCompany	Name	Created at	Updated at	Actions
1	0-79-0-37	0-52-79		3 June 2008 12:12	3 June 2008 12:12	
2	0-79-0-37	5-46-79		3 June 2008 12:12	3 June 2008 12:12	
3	0-79-0-37	0-52-79		3 June 2008 12:23	3 June 2008 12:23	

Figura 3.5: Listagem de Templates.

Na figura 3.5, visível acima temos um lista de *templates* já criados e ao seu lado direito um filtro pelo nome. Em baixo da listagem temos um botão para se criar um *template* novo e um botão de *preview* para vermos como ficou o *template*.

Na figura 3.6 temos o *frontend* para criar um *template*, do lado direito podemos ver os componentes que podemos adicionar ao *template* e depois podemos com “*drag and drop*” move-los, assim como alterar as suas dimensões na folha.

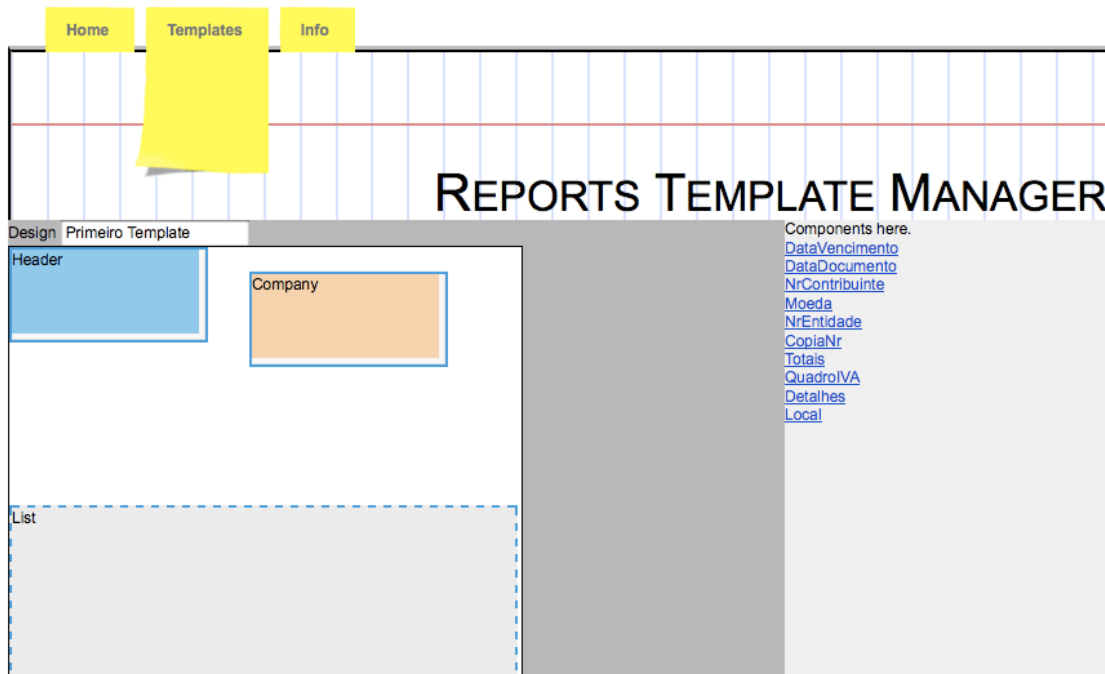


Figura 3.6: Frontend para a criação de templates.

Conclusão

Neste capítulo sintetiza-se o que foi descrito neste documento e apresentam-se as conclusões finais acerca do projecto, comparando-se os objectivos iniciais e resultados alcançados. Finalmente, descrevem-se as perspectivas de continuação do projecto.

Síntese do Relatório

O documento que se apresenta veicula a importância de um sistema que permita o fácil manuseamento de relatórios que na sua criação, que na edição dos mesmos. Foi apresentado a primeira versão do **RTM**, iniciado pela necessidade de tal ferramenta e a oportunidade de poder implementá-lo no âmbito do projecto de Laboratórios de Informática IV. Foram introduzidos os objectivos que pautaram o presente projecto e o desenvolvimento destes objectivos. No capítulo do desenvolvimento é exposta a análise, o desenho e implementação do **Reports Template Manager**, tarefas estas que permitiram atingir os objectivos propostos.

Ponto da Situação

O que foi desenvolvido até ao momento preenche todos os requisitos que necessitávamos para a criação de complexos relatórios de uma forma mais simples e intuitiva, que era um dos objectivos.

A ferramenta desenvolvida não se encontra ainda disponível para o uso de outros programadores, nem para a criação ou alteração dos *templates* por parte das empresas, mas a sua implementação será para breve (que está a ser implementado neste momento na aplicação de Gestão Comercial previamente mencionada).

É um projecto bastante extensível, e por isso estará em constante desenvolvimento.

Trabalho Futuro

Como já tínhamos referido no início, o **RTM** foi criado para ser incorporado num programa de Gestão previamente criado. No futuro vamos resolver qualquer problema de incompatibilidade e integrar de forma transparente no programa.

Queremos também disponibilizar o *plugin para download* da comunidade de symfony, Para tal temos que criar alguma documentação extra e exemplos de implementação simples.

Vamos também criar uma versão usando Propel (nesta versão foi usado Doctrine) visto a *framework symfony* usar também este ORM.

Também, para ficar mais *user friendly*, pensamos que será melhor mudar o *design* do *template*, de modo a tornar mais simples para futuras mudanças a nível do programador e mais personalizáveis.

Esperamos que uma vez que esteja online, a comunidade actue de forma activa e nos ajude com a evolução do projecto, quer seja a referir novas *features*, quer seja a implementar as mesmas.

Referências Bibliográficas

[Zaninotto and Potencier, 2007] Zaninotto, F. and Potencier, F. (2007). The Definitive Guide to Symphony.

Referências Web

[Site do Symfony Book](#)

[Site do Doctrine](#)

[Site do TCPDF](#)

[Informação de SQL](#)

[SQL v.s. Procedures](#)

[Wiki de MVC](#)

[Why use symfony](#)

[Wiki de Symfony](#)

[Propel Vs Doctrine](#)

[Informação de svn](#)

[Why svn](#)

[PHP vs ASP](#)

[Wiki de PHP](#)

[10 reasons to use Javascript](#)

[Why use Javascript](#)

[Wiki de Javascript](#)

[Sitio do JQuery](#)

[Wiki do JQuery](#)

[jQuery philosophy](#)

[Informação do TCPDF](#)

[Why use TCPDF](#)