Susana Isabel Neto Ribeiro Rodrigues

Desenvolvimento de Software baseado em Modelos para a Construção de Aplicações Web Dinâmicas para Eventos Técnico-Científicos

Dissertação de Mestrado em
Informática de Gestão

Novembro de 2015

Para os meus avós maternos.

Sempre comigo.

Agradecimentos

Antes de mais gostaria de agradecer à Escola Superior de Tecnologia de Setúbal que me proporcionou as condições essenciais à elaboração deste projeto.

Ao meu orientador, o Professor Joaquim Filipe, que me orientou e apoiou durante o desenrolar deste projeto, demonstrando sempre interesse e disponibilidade ao longo deste trabalho.

Gostaria de agradecer à minha família, aos meus pais e ao meu irmão por todo o apoio e paciência ao longo de toda esta etapa.

Agradeço também aos colegas do meu grupo profissional Pedro Varela, Filipe Mariano e João Francisco, por todo o auxílio e troca de ideias que permitiram este projeto avançar.

Em especial gostaria de agradecer ao meu marido e colega de mestrado Rui Rodrigues, que acompanhou de perto todo o processo, por todo o apoio, motivação e carinho dado.

Resumo

Com o vasto volume de informação a circular diariamente na internet é crucial que a informação existente se mantenha atualizada, sob pena do utilizador perder o interesse em consultar a mesma. Como tal, torna-se importante que o *website* que pretendemos disponibilizar esteja de acordo com o modelo de negócio da organização e que a informação nele contida se mantenha atualizada sem a constante intervenção do gestor de dados do website. Assim, o trabalho de investigação aqui realizado tem por principal objetivo o desenvolvimento de uma metodologia que possibilite a modelação conceptual da estrutura e dinâmica de um *website* neste contexto. Este sistema assenta na modelação dos vários elementos que compõem o *website*, descrevendo-os sintática e semanticamente. A modelação deve ainda representar a dinâmica de cada página, proporcionando flexibilidade na definição e alteração do modelo do sistema de informação, facilitando também o desenvolvimento e manutenção do *software*. Pretende-se pois que o sistema de *software* implemente diretamente estes modelos, eliminando intervenções humanas decorrentes de ações temporais decorrentes de processos associados ao contexto em estudo. Estas alterações devem refletir-se de imediato no *website* colmatando assim as lacunas existentes nos sistemas de gestão de conteúdos atualmente disponíveis ao utilizador.

Palavras-chave: Metodologia, Aplicação Web, Dinamismo, Modelação, Website Automático.

Abstract

With the vast amount of information circulating daily on the internet is crucial that all the existing information is kept updated otherwise the user will lose interest in consulting it. As such, it is important that the website we want to provide is consistent with the organization's business model and that the information contained in it is kept updated without the constant intervention of the website manager. Thus, the investigation work carried out is primarily engaged in the development of a methodology that enables the conceptual modeling of both structure and dynamics of a website in this context. This system is based on the modeling of the various elements that comprise the website, describing them syntactically and semantically. The modeling should also represent the dynamics of each webpage, providing flexibility in the definition and updates of the information system model, also facilitating the development and maintenance of software. It is intended for the software system to directly implement these models, eliminating the human intervention that arises from actions occurring in the website life cycle, resulting from processes associated with the context in study. These changes should reflect immediately on the website thus filling the gaps in current content management systems available to the user.

Keywords: Methodology, Web Application, Dynamic, Modelling, Automated Website.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

API	*Application Programming Interface*
CASE	*Computer-Aided Software Engineering*
CMS	*Content Management System*
DNN	*Dot Net Nuke*
HTML	*HyperText Markup Language*
IIS	*Internet Information Systems*
MVC	*Model View Controller*
OO	*Orientado a Objetos (Object Oriented)*
PHP	*PHP: Hyper text Preprocessor (originalmente Personal Home Page)*
SI	*Sistema de Informação*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
XHTML	*eXtensible Hyper Text Markup Language*
XML	*eXtensible Markup Language*

Capítulo 1

1. Introdução

Este capítulo define o problema que este projeto de mestrado visa resolver, com base no qual se identificou um conjunto de objetivos que este trabalho pretende atingir. Seguidamente referem-se as principais contribuições deste trabalho, e também a metodologia desenvolvida para o mesmo. O capítulo termina com a identificação da estrutura do documento, apresentando de forma sucinta o conteúdo dos restantes capítulos.

1.1. Definição do Problema

No quotidiano a utilização da internet tem vindo a assumir um lugar de destaque crescente. O volume de informação circulante tem crescido a um ritmo vertiginoso, atingindo mais de 500 mil petabytes no ano transato. Pode-se encontrar todo o tipo de informação na internet, como notícias, anúncios, artigos científicos, vídeos, entre outros, acrescendo que a informação disponibilizada não é estática, estando em constante atualização. Torna-se assim necessário que as aplicações web acompanhem este dinamismo.

Com as alterações solicitadas, quer sejam através de novos requisitos, quer sejam devido a alterações a regras de negócio existentes, a necessidade da entrega rápida do produto final ao cliente pode conduzir a dificuldades no ritmo e qualidade do desenvolvimento do *software*, devido a uma eventual negligência da modelação existente. Esta eventualidade implica que o modelo do sistema de informação se possa encontrar bastante desfasado da realidade.

No caso presente, o problema centra-se na criação de *websites* de suporte à gestão de eventos de cariz científico, onde se sente a necessidade de uma constante atualização da informação. Isto é, existe a necessidade de inserir, remover ou modificar informação no *website* à medida que o tempo decorre, no que diz respeito a prazos (submissões, registos, etc.), como relativamente a listas de intervenientes tais como revisores ou oradores convidados, ou mesmo à inclusão de novos patrocinadores e apoios institucionais, por exemplo. Essencial é também a necessidade de o *website* se adaptar a alterações nas regras de atividade, que também ocorrem com alguma frequência. Este problema é ainda mais complexo e premente quando se pretende gerir várias conferências científicas simultaneamente e portanto existe a necessidade de reutilizar a estrutura e a metodologia de gestão dos *websites* de suporte às conferências de forma sistemática para assim minimizar o custo de manutenção. Neste contexto é crucial que o modelo seja o mais rigoroso possível e que possa ser facilmente alterado para assim poder estar coerente com a realidade subjacente e facilite a atualização do *software* que o implementa, minimizando os custos de manutenção do *website*.

1.1.1. Contextualização do Problema

Para tornar claro o contexto do problema a abordar nesta tese de mestrado, há que elucidar o que se entende sobre um evento de cariz científico.

Um evento de cariz científico consiste num ponto de encontro de membros de uma comunidade científica de uma determinada área de interesse, onde estes investigadores – tanto do mundo académico como de uma vertente industrial – podem apresentar e discutir os frutos do seu trabalho científico. O seu trabalho deve ser submetido como um artigo científico (devendo obedecer a um determinado formato), que deve ser sujeito de apreciação por um conjunto de profissionais altamente qualificados na área temática do evento (comité de revisores). Estes eventos são também uma excelente oportunidade para os participantes assistirem a palestras e painéis com a presença de investigadores de renome mundial na área científica do evento em questão. Geralmente estes eventos têm uma duração de 2-3 dias.

A designação mais comum destes eventos é conferência, embora existam outros tipos de evento, nomeadamente:

- *Workshops* – fornecem uma plataforma mais interativa e focada para a apresentação e discussão de novas e emergentes ideias;
- *Special sessions* – são eventos de pequena dimensão mas bastante especializados contendo um conjunto de apresentações altamente singularizadas em algum tema específico ou constituídas pelo trabalho associado a um projeto internacional em particular. O objetivo destes eventos é fornecer uma discussão focada em tópicos inovadores;
- *Doctoral Consortium* – simpósio dedicado a ajudar alunos de doutoramento no decurso do seu trabalho de investigação. Cada aluno deve apresentar um artigo descrevendo o seu trabalho numa fase intermédia, frente a um conselho consultivo (advisory board), composto por professores extremamente qualificados e experientes. O estudante deve preparar uma apresentação que ilustre inequivocamente os pontos principais do seu projeto, seguido de um debate com o advisory board, de modo a auxiliarem na progressão do trabalho;
- *Tutorials* – fornece uma plataforma para intercâmbio científico mais intenso entre investigadores interessados num tópico específico, servindo igualmente como ponto de encontro para a comunidade. Os tutoriais proporcionam aos participantes uma visão geral ampla de campos de pesquisa emergente;
- *Panel* – sessão de 90 a 120 minutos na qual entre 4 a 6 oradores, membros distintos da comunidade científica e/ou empresarial apresentam brevemente diferentes perspetivas ou opiniões em questões-chave com o intuito de estimular uma discussão provocativa e controversa entre os membros do painel e a audiência. Existem dois tipos de painel: painel de investigação (Research Panel), orientado para a comunidade académica e focado na discussão de tópicos de investigação; painel industrial

(Industrial Panel), orientado à promoção da troca de conhecimento entre os meios académico e industrial;

- *Demos (Demonstrations)* – fornecem aos investigadores e profissionais uma oportunidade interativa para a apresentação dos seus sistemas e/ou protótipos, através de uma sessão regular de apresentação oral ou através de uma exibição técnica. O formato comercial é evitável devendo a demonstração ser focada em aspetos técnicos. As demos são baseadas num ambiente informal estimulando os apresentadores e participantes a debater o trabalho apresentado. São também uma oportunidade para os participantes divulgarem os resultados práticos da sua investigação e criarem uma rede de contactos com outros investigadores e parceiros de negócio.

A organização deste tipo de evento engloba várias fases ao longo do seu ciclo de vida: preparação, submissão, revisão, seleção, registo e programa. Cada uma destas fases apresenta várias ações bem como um dinamismo próprio.

A fase de preparação consiste na angariação da informação base a colocar no *website* (tais como datas, local de realização, âmbito, áreas e tópicos, pessoas-chave e parceiros). À medida que a informação é adicionada ao sistema de informação, a mesma irá surgindo de forma automática no *website*. Esta informação está em constante atualização (por exemplo a adição de oradores convidados, parceiros ou membros do comité de revisão de artigos) sem necessidade de intervenção do *webmaster*.

A fase seguinte, submissão, apresenta mais modificações no *website*, como a inclusão do *link* para submissão em vários pontos do mesmo. É também possível visualizar as alterações referentes aos deadlines das datas de submissão, à medida que os mesmos vão sendo alcançados.

As fases de revisão e seleção não apresentam grandes alterações em termos de *website*, embora se verifique que o *link* de submissão deixe de estar disponível. Verificam-se também, tal como na fase anterior, alterações nos deadlines à medida que estes expiram.

A fase de registo apresenta mais modificações no *website* através da inclusão da página com os vários preços de registo, bem como o *link* associado ao mesmo.

Finalmente, na fase do programa, surge no *website* o programa associado aos vários dias do evento.

1.2. Objetivos

Com o propósito de responder à problemática enunciada na secção anterior foram definidos alguns objetivos para o presente trabalho.

O objetivo primário corresponde ao desenvolvimento de uma metodologia que permita a modelação conceptual quer da estrutura de um *website* quer da dinâmica do mesmo, no contexto já definido. A modelação das várias páginas será baseada em modelos classificativos e estruturais para definir declarativamente as zonas de cada página, descrevendo quer a sua sintaxe quer a sua semântica. Além disso, será necessário modelar a dinâmica de cada página, o que se fará através de máquinas de estados finitas, baseadas no modelo temporal subjacente ao próprio evento científico, para o qual existem *a priori* um conjunto de atividades e subactividades com as respetivas datas limite. Com a metodologia desenvolvida proporcionar-se-á uma elevada flexibilidade quer na definição quer na alteração do modelo do sistema de informação e julga-se que o elevado rigor da metodologia definida permite uma correspondente facilidade no desenvolvimento e manutenção do *software*.

Em segundo lugar, recorrendo a esta metodologia pretende-se criar um sistema de *software* que implemente diretamente os modelos acima referidos, permitindo assim evitar a intervenção do programador para aplicar as alterações que surgem quer devido a ações manuais efetuadas pelo gestor do website quer devido a ações automáticas, disparadas pela passagem do tempo (prazos limite) ou pela ocorrência de certas condições no sistema, no contexto do modelo dinâmico dos processos subjacentes à gestão de um evento científico. Estas alterações, representadas no modelo de dados do sistema de informação, têm de ser refletidas imediata e automaticamente na camada de apresentação da aplicação web.

Finalmente, o terceiro objetivo deste trabalho consiste em validar o sistema desenvolvido para avaliar se este sistema contribui para uma melhoria na qualidade do desenvolvimento destes *websites*, concomitante com uma redução dos custos, nomeadamente em termos do tempo gasto pelo webmaster e/ou do secretariado ou gestor de eventos, e avaliar também, comparativamente, a robustez do sistema desenvolvido relativamente a diversos sistemas de gestão de conteúdos, de natureza mais genérica, que serão estudados no capítulo 3.

1.3. Principais Contribuições

Com o desenvolvimento deste projeto pretende-se contribuir para a compreensão e melhoramento da qualidade e eficiência do desenvolvimento automático de *websites* para conferências científicas, incluindo os seguintes aspetos:

- Definição de um corpo conceptual para a modelação de um sistema de informação para gestão de conferências, como base para criação automática de aplicações web dinâmicas

- Representação da estrutura de cada página web usando modelos classificativos e estruturais para definir declarativamente as zonas de cada página,
- Representação da dinâmica de cada página web usando máquinas de estados e modelos temporais, garantindo a sua coerência, de acordo com os processos de organização de uma conferência, ao longo do tempo de vida da mesma,
- Criação de um mecanismo automático de desenvolvimento aplicacional adaptado às características de cada conferência, permitindo a geração e manutenção do *website* de suporte da mesma sem intervenção do programador e com intervenção limitada do secretariado/gestor do evento, com base nos processos definidos para a organização do evento.

1.4. Metodologia de Desenvolvimento

A produção deste trabalho abrangeu os seguintes pontos:

- Análise dos mecanismos de gestão de uma conferência científica para compreender os conceitos e processo subjacentes.
- Identificação dos problemas inerentes ao desenvolvimento dos *websites* de suporte a eventos deste tipo.
- Investigação de uma metodologia visando facilitar a criação automática de aplicações web dinâmicas, suportada por uma base teórica fundamentada.
- Definição de uma abordagem para a geração automática de *software* a partir da metodologia investigada, adaptável ao contexto do problema.
- Validação da abordagem definida e desenvolvida no presente trabalho.

1.5. Estrutura do Documento

Este trabalho encontra-se estruturado em seis capítulos.

No primeiro capítulo é apresentado o projeto, bem como o contexto no qual se insere, nomeadamente a área de aplicações web dinâmicas. Neste capítulo enquadra-se também a problemática inerente ao desenvolvimento deste trabalho, tal como os objetivos que se visam alcançar. É também apresentada a metodologia utilizada e a descrição do trabalho.

O segundo capítulo corresponde à parte dinâmica do enquadramento conceptual. É neste capítulo que se fará menção a máquinas de estado, ferramentas necessárias no desenvolvimento deste trabalho. É também referido o processo de modelação e a importância de mesmo no desenvolvimento de sistemas de *software*.

No terceiro capítulo elaborou-se um estudo comparativo de várias ferramentas de *software* existentes que se encontram geralmente sob a designação de “sistemas de gestão de conteúdos”

e são capazes de gerar aplicações web dinâmicas de forma automática, sem necessidade de programação.

No quarto capítulo apresenta-se o sistema desenvolvido. Nele se descrevem os requisitos, funcionalidades e todo o conjunto de características devidamente justificadas, que permitiram a conceção desta solução informática. A descrição do sistema encontrada neste capítulo possibilitará clarificar não só a sua implementação mas também as decisões tomadas referentes à sua realização.

No quinto capítulo o projeto desenvolvido é validado através do teste em casos concretos de organização de conferências científicas, através do acompanhamento da simulação da dinâmica do *website* de cada conferência de teste, ao longo de todo o ciclo de vida de cada uma delas.

O sexto e último capítulo apresenta uma análise crítica, identificando os pontos positivos e as limitações deste projeto, e referindo também o trabalho que se pode realizar futuramente para melhoria da metodologia e da ferramenta de *software* aqui apresentadas.

Capítulo 2

2. Modelação

Este capítulo incide sobre a modelação do projeto, em particular sobre a importância da mesma. Este capítulo começa por apresentar a noção de semiótica organizacional, apresentando a abordagem MEASUR e os vários métodos que a compõem.

Neste capítulo aborda-se também a teoria de máquinas de estados, tendo em conta a abordagem alternativa para o desenvolvimento do sistema associado a este trabalho.

Modelar reveste-se de uma importância significativa no desenvolvimento de *software*, uma vez que trata da conceção de modelos explicativos das características do mesmo. Assim, a modelação facilita o entendimento do projeto, ajudando na prevenção de erros de programação e funcionamento.

Neste projeto, como ferramentas de apoio à modelação, recorreu-se a máquinas de estados.

2.1. Semiótica Organizacional

A semiótica organizacional examina a natureza e características da informação, estudando qual o melhor modo de usar essa informação no contexto das atividades organizacionais e no domínio do negócio.

A semiótica organizacional trata a organização como um sistema de informação, no qual a informação é criada, processada, distribuída, armazenada e usada. Deste modo, a semiótica organizacional pode obter perspetivas frutíferas de estudo de várias culturas organizacionais, bem como da globalização (Gazendam, Jorna, & Liu, 2004).

De acordo com (Gazendam, Jorna, & Liu, 2004) existem três abordagens para o estudo do elemento elementar da comunicação em semiótica, nomeadamente:

- Abordagem baseada em texto;
- Abordagem baseada em sinais;
- Abordagem baseada em “memes” (elementos de um sistema cultural ou comportamental que possam ser considerados transmissíveis de um indivíduo para outro por meios não genéticos, especialmente através da imitação).

A escolha de uma destas abordagens terá consequências no tipo de trabalho empírico que é realizado. A semiótica baseada em texto tem o seu fundamento teórico através da leitura e análise textual, visto que os textos são encarados como relacionados com outros textos.

Na semiótica baseada em sinais, os sinais relacionam-se com o mundo (como objetos) e com a cognição humana (como interpretante). O trabalho empírico consiste na investigação da relação de sinais com a cognição humana, e com o mundo em que se referem.

Na semiótica baseada em memes, estes são encarados primeiramente do ponto de vista de transformação e seleção. Os memes relacionam-se com os seus portadores (seres vivos), que formam populações. O trabalho empírico irá focar-se no estudo da dinâmica da população de memes e portadores, bem com nos mecanismos de transferência e seleção. Tal acarreta bastante trabalho quantitativo e estatístico.

A semiótica organizacional tem encontrado o seu espaço com base nas suas aplicações práticas no campo de análise e *design* das organizações, nas transações económicas e sistemas de informação através de abordagens e métodos que têm sido desenvolvidos como alternativa aos métodos tradicionais de sistemas de informação. Alguns métodos bem conhecidos consistem na análise linguística da comunicação durante o trabalho, a análise de interações dos atores, análise de tarefas do ator, análise semântica, análise de conhecimento da tarefa, análise de normas e modelos de simulação de construção. Estas abordagens e métodos consistem apenas uma área abrangida pela semiótica organizacional.

De acordo com a semiótica organizacional uma organização pode ser caraterizada como uma comunidade de pessoas que partilham o conhecimento do comportamento desejável e participam na construção social desse conhecimento.

O objetivo da semiótica organizacional consiste na descoberta de novas maneiras de analisar, descrever e explicar a estrutura e o comportamento organizacional, cabendo ao analista ajudar os utilizadores a articular os seus problemas, descobrir os requisitos de informação e finalmente desenvolver uma solução sistemática.

Na ótica de (Martins E. , Index of /~eliane/Cursos/MO409/Curso2003/Apresentacoes, 2003) a estrutura dos sistemas de informação pode ser representada da seguinte forma:

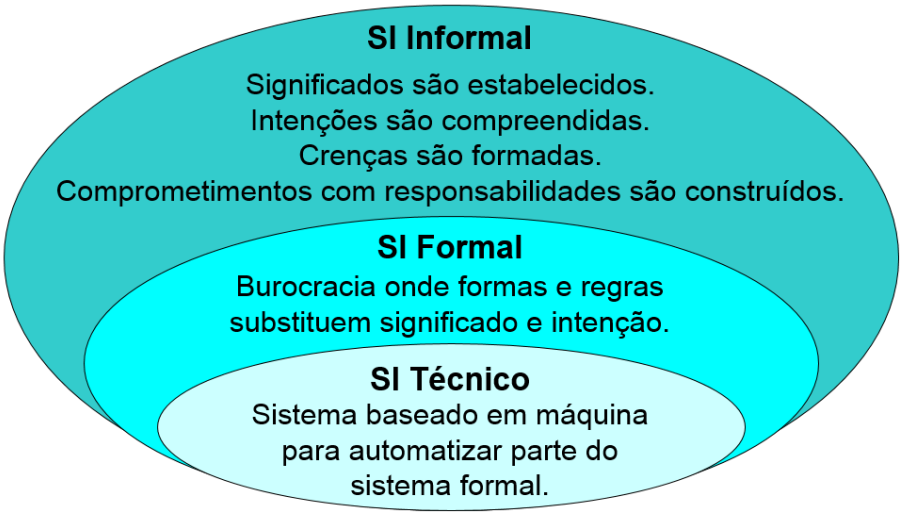


Figura 2-1- Estrutura dos Sistemas de Informação (adaptado de (Martins E., Index of /~eliane/Cursos/MO409/Curso2003/Apresentacoes, 2003))

Surge assim uma abordagem semiótica para sistemas de informação: MEASUR (*Methods*, *Means*, *Models*… *for Exploring*, *Eliciting*, *Evaluating*...*Articulating*, *Analysing*, *Assessing*... *and Structuring*, *Specifying*, *Stimulating*... *Users' Requirements*). Esta abordagem, surgida no final da década de 1970 através de Ronald Stamper, consiste num conjunto de métodos orientados por normas para modelação de sistemas de negócio e especificação de requisitos para o desenvolvimento de *software*. O conceito subjacente da abordagem consistia no facto de as próprias organizações serem sistemas de informação sendo a norma social a unidade apropriada de especificação. Numa situação convencional esta abordaria permitiria o auxílio na identificação e resolução de problemas, levando os gestores e utilizadores do sistema a um especificação precisa dos requisitos de informação, antes do desenvolvimento do sistema de *software* propriamente dito. A abordagem MEASUR apresenta vários métodos, nomeadamente:

- PAM – método de articulação de problemas

Conjunto de métodos possíveis de serem aplicados no estado inicial de um projeto, quando ainda estamos perante um problema vago e complexo. Permite o auxílio do utilizador na identificação de problemas que requeiram atenção, sendo o utilizador auxiliado também na definição de unidades de sistema validadas pelas partes interessadas, ou *stakeholders*. A ferramenta utilizada designa-se por PAT (*Problem Articulation Tool*).

- SAM – método de análise semântica

A unidade de sistema ou um problema focal é tido como ponto d partida para a análise. Este método permite auxiliar o utilizador a eliciar e representar os seus requisitos num modelo formal e preciso, designado por diagrama de ontologias. Através de um "facilitador", as funções requisitadas pelo sistema são especificadas no modelo ontológico que descreve uma visão dos agentes responsáveis no domínio do negócio em foco. O significado do sinal usado no modelo semântico, para representar o mundo do negócio é tratado como uma relação entre o sinal e as ações apropriadas.

- NAM – método de análise de normas

Este método foca-se nas normas sociais, culturais e organizacionais que governam as ações dos agentes no domínio do negócio. Uma norma pode assim definir a responsabilidade de um agente ocupando uma certa incumbência ou condição (pode, não pode, deve, etc.). Cada norma é associada a um padrão de ações no sistema.

- Método de análise de comunicação e controlo

Auxilia na análise das diversas comunicações existentes entre todos os possíveis agentes responsáveis e unidades de sistema dentro de um sistema focal. As mensagens são classificadas em informativas, coordenação e controlo, de acordo com o pretendido pelo emissor. As normas são assim adicionadas para a orientação de procedimentos, fluxos de mensagens, recompensas e punições.

- Método de análise de meta-sistemas

Este método permite o planeamento, sincronismo, análise de custo/benefício, gestão de projetos, etc. baseado nos resultados das fases anteriores. O próprio processo de mudança é tratado como um sistema social que requer análise, projeto e suporte.

Com base nos métodos acima mencionados segue-se uma tabela ilustrativa de algumas alternativas de desenvolvimento de um sistema de informação, aplicando estes métodos.

Tabela 2-1- Desenvolvimento de um SI: alternativas com base nos métodos MEASUR

Desenvolvimento de um SI	Alternativa 1	Alternativa 2	Alternativa 3
Análise de requisitos	SAM, NAM	SAM, NAM	SAM, NAM
Análise do sistema	SAM, NAM	SAM, NAM	Outros métodos (ex.: OO) ou análise estruturada
Desenho do sistema	SAM, NAM	Outros métodos (ex.: OO) ou *design* estruturado	Outros métodos (ex.: OO) ou *design* estruturado
Implementação do sistema	Outros métodos (ex.: OO) ou outras linguagens e ferramentas CASE	Outros métodos (ex.: OO) ou outras linguagens e ferramentas CASE	Outros métodos (ex.: OO) ou outras linguagens e ferramentas CASE

Para a análise de requisito aplicaram-se os métodos SAM e NAM.

A imagem seguinte ilustra as fases da análise semântica associadas ao método SAM.

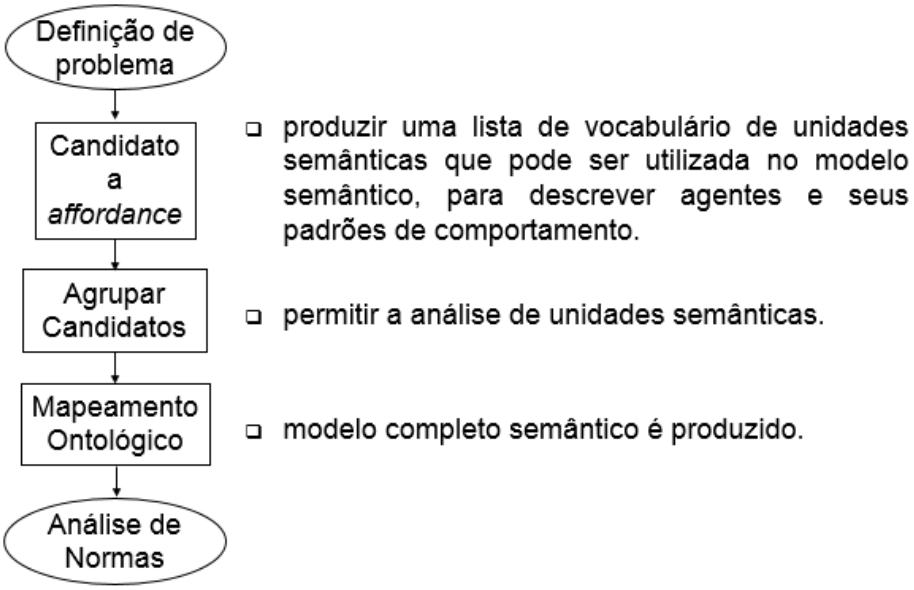


Figura 2-2- Fases da análise semântica (extraído de (Martins E., Index of /~eliane/Cursos/MO409/Curso2003/Apresentacoes, 2003))

Para a análise da norma (Martins E., Index of /~eliane/Cursos/MO409/Curso2003/Apresentacoes, 2003) sugere quatro passos, indicados na figura seguinte.

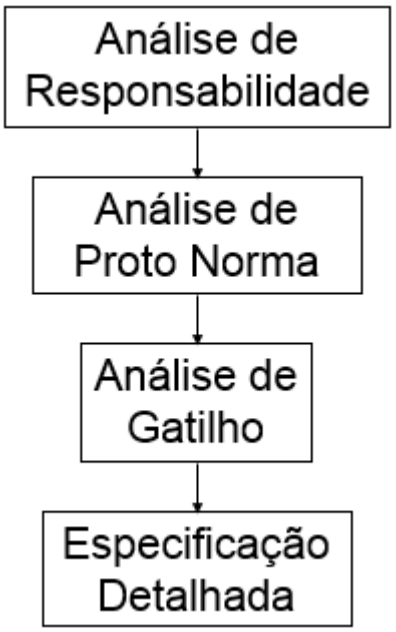


Figura 2-3 - Passos para a análise de normas (extraído de ((Martins E. , Index of /~eliane/Cursos/MO409/Curso2003/Apresentacoes, 2003))

Para o desenho do sistema, recorreu-se a máquinas de estados, para facilitarem a compreensão dos vários estados e ações que podem ser feitas sobre os mesmos.

No que diz respeito à implementação do sistema abordou-se por uma abordagem orientada a objetos.

2.2. Máquinas de Estados

Comecemos pela introdução ao conceito. Afinal, o que é uma máquina de estados?

Uma máquina de estados consiste:

- Um conjunto de eventos de entrada
- Um conjunto de eventos de saída
- Um conjunto de estados
- Uma função mapeadores de estados e entradas para as saídas
- Uma função que mapeia estados e entradas para estados (sendo estes designados por estados de transição)
- Uma descrição do estado inicial

As máquinas de estados são usadas em diversos campos, onde se incluem sistemas reativos, engenharia elétrica, linguística, ciências da computação, biologia, matemática e lógica.

Um caso particular de máquina de estados é a máquina de estados finitos. Designa-se por máquina de estados finitos aquela que tem um número limitado (ou finito) de estados possíveis. Estas máquinas de estados podem ser usadas tanto como ferramenta de desenvolvimento para abordar e solucionar problemas, bem como um formalismo para descrição da solução, para desenvolvimentos posteriores e manutenção do sistema.

Existem inúmeras formas de ilustrar máquinas de estados, desde simples tabelas a animações gráficas. Um exemplo de representação de uma máquina de estados encontra-se na figura abaixo.

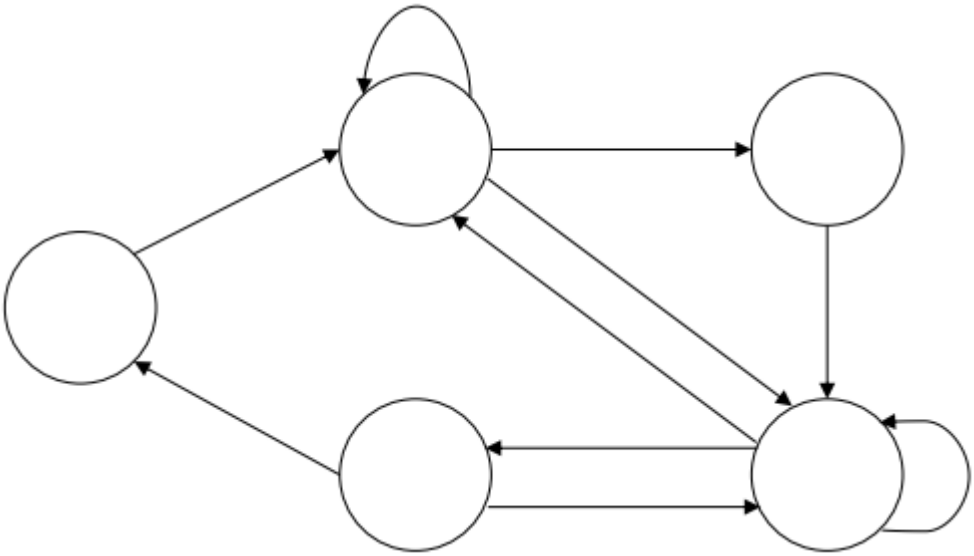


Figura 2-4- Exemplo de máquina de estados (extraído de (Lima, 2014)).

Este conceito corresponde à representação de uma máquina abstrata que se deve encontrar num dos seus estados finitos. Só é possível que a máquina esteja num único estado num dado momento, estado esse que se designa de estado atual. Um estado contém informação anterior, isto é, ilustra as mudanças ocorridas desde a entrada num estado no início do sistema até ao momento presente. Uma transição reflete uma mudança de estado, sendo descrita através de uma condição, que se deve verificar para a ocorrência da transição. Uma ação designa uma atividade a realizar num momento específico.

Uma definição de máquina de estados aplicada ao domínio da computação pode ser encontrada em (What is finite state machine? - Definition from WhatIs.com, 2014), onde se designa por máquina de estado todo e qualquer dispositivo que armazena o estado de algo num dado momento, podendo funcionar como entrada para alterar o estado e/ou causar uma ação ou saída (output) para ocorrer para uma dada alteração. Um exemplo de máquina de estados é um computador. Cada instrução máquina corresponde à entrada (ou *imput*) que provoca a mudança de um ou mais estados e pode despoletar a ocorrência de outras ações. Cada registo de dados contém um estado. O sistema operativo é por si mesmo um estado, sendo que cada aplicação que nele corre começa com um determinado estado inicial, podendo este mudar dependendo da manipulação dos dados na entrada. Contudo, na prática, as máquinas de estados são usadas na descrição e desenvolvimento de interações de programas ou dispositivos específicos.

Compreende-se assim o porquê de um computador poder ser visto como conjunto de estados bastante complexo, e cada programa nele contido corresponde a uma máquina de estados.

Resumidamente, pode-se descrever uma máquina de estados como:

- Um estado inicial ou registo de algo armazenado nalgum lugar
- Um conjunto de possíveis eventos de entrada
- Um conjunto de novos estados resultantes da entrada de informação
- Um conjunto de possíveis ações ou eventos de saída, resultantes de um novo estado

As máquinas de estados apresentam várias vantagens (Lima, 2014), entre elas, o facto de serem simples de implementar (existindo diversas formas de implementação). São também intuitivas, isto é, o seu entendimento é percetível a qualquer pessoa através da simples observação da sua representação visual. As máquinas de estados não se apresentam como estruturas rígidas, podendo ser facilmente ajustadas por alterações de requisitos ou funcionamento, pelo que apresentam flexibilidade.

Uma desvantagem das máquinas de estados surge associada a um aumento do grau de complexidade à situação que pretendem retratar, sendo que nesses casos, a representação visual torna-se muito complexa e de difícil entendimento. No entanto, esta desvantagem não é aplicada ao trabalho em questão, em que as máquinas de estados não apresentam elevado grau de complexidade.

A figura seguinte ilustra um exemplo de uma máquina de estados.

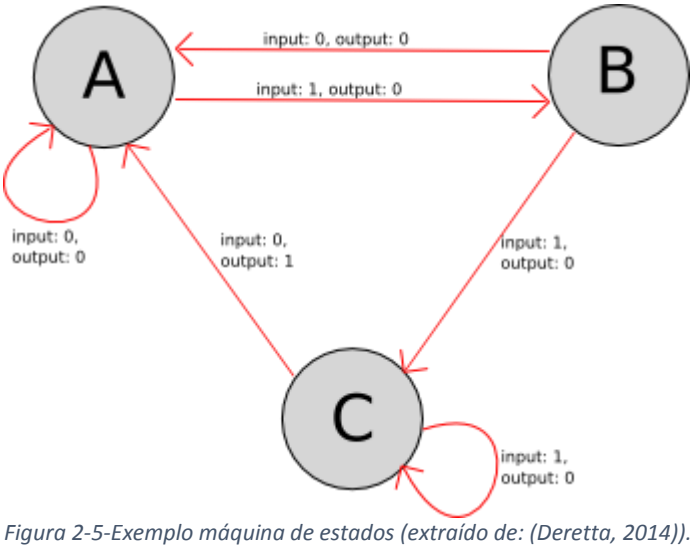


Figura 2-5-Exemplo máquina de estados (extraído de: (Deretta, 2014)).

Os círculos identificados por “A”, “B” e “C” representam estados, sendo “A” o estado inicial.
Os arcos a vermelho correspondem a transições.
O formalismo de funcionamento desta máquina de estados pode ser descrito do seguinte modo:

A (estado inicial):
Se *input* == 1 então transita para estado B e *output* ==0 senão permanece no estado A e *output* == 0
B:
Se *input* == 1 então transita para estado C e *output* ==0 senão transita para o estado A e *output* == 0
C:
Se *input* == 1 então permanece no estado C e *output* ==0 senão transita para o estado A e *output* ==1

2.2.1. Classificação das Máquinas de Estados

Na década de 1950 foram identificados dois tipos de máquinas de estados, as máquinas de Mealy e Moore. Estes dois modelos são os modelos básicos de máquinas de estados.

O primeiro tipo de máquina de estados foi identificado por G. H. Mealey, em 1955. Em 1956 surgem as máquinas de Moore (recebendo o nome pela sua identificação por E. F. Moore).

Nas máquinas de Moore as saídas dependem somente dos estados atuais. Um exemplo de aplicação de uma máquina de Moore consiste no desenvolvimento de um analisador léxico, ou um tradutor de linguagem (Brito, Martendal, & Oliveira, 2015). No fundo um analisador léxico determina os componentes básicos da linguagem, nomeadamente números, identificadores, separadores, etc. Um analisador léxico é então uma máquina de Moore na medida em que: um estado final é associado a cada unidade léxica sendo que cada estado final possui uma saída que descreve a unidade léxica identificada. Para os restantes estados não finais, a saída produzida corresponde a um conjunto vazio.

Nas máquinas de Mealy as saídas são obtidas em função dos estados atuais e das entradas, ou seja, são modificadas síncrona e assincronamente. Um exemplo de utilização frequente deste tipo de máquinas é um projeto de conversação entre um programa de computador e o seu utilizador (sistema pericial), sendo que neste caso o diálogo pode ser dirigido pelo programa ou pelo utilizador.

No caso de termos uma entrada vazia, o funcionamento das máquinas de Mealy e Moore não é equivalente. Ao passo que a máquina de Moore gera a palavra correspondente ao estado inicial, a máquina de Mealy não gera saída, visto que não executa nenhuma transição.

Nos restantes cenários, existe uma equivalência entre ambos os modelos. Significa isto que qualquer máquina de Moore pode ser simulada através de uma máquina de Mealy, para entradas não vazias, e qualquer máquina de Mealy pode ser simulada por uma máquina de Moore.

O modelo escolhido irá influenciar um projeto, embora não haja indicações acerca de que modelo é o melhor.

A escolha de um modelo depende da aplicação, dos meios de execução (por exemplo, sistemas de hardware são geralmente percebidos como modelos de Moore) e preferências pessoais do *designer* ou programador. Na prática, os modelos mistos são muitas vezes usados com vários tipos de ação (Wagner, 2005). A figura (EDASIM, 2015) permite-nos observar essa diferenciação.

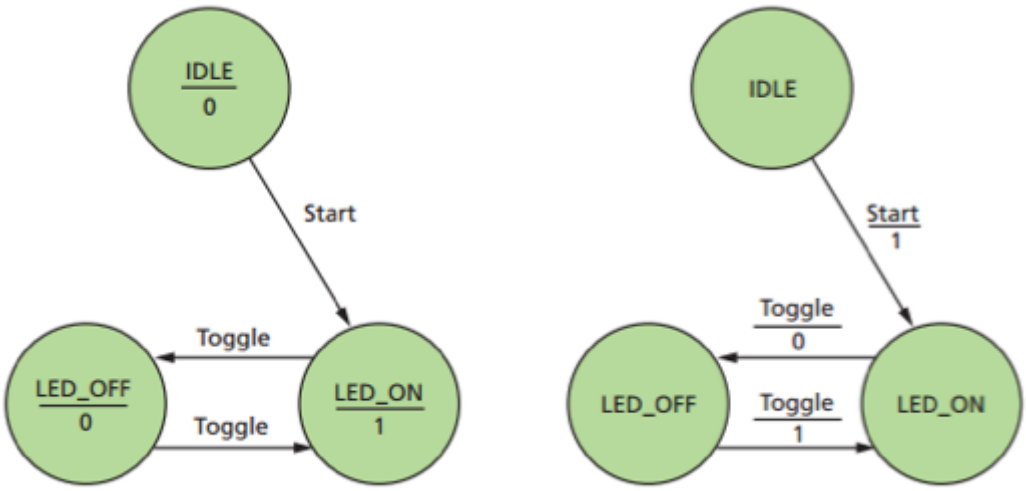


Figura 2-6- Máquina de estados de Moore (esquerda) e de Mealy (direita) (extraído de (EDASIM, 2015)).

2.3. A Importância da Modelação

Modelar consiste na elaboração de um modelo que explique as características de funcionamento bem como o comportamento de um *software*, sobre o qual ele será desenvolvido. A modelação permite o entendimento do *software* através de características que evitarão erros de programação e funcionamento futuros. Assim é fácil de perceber porque a modelação é uma parte importante no desenvolvimento de *software*.

Modelos são uma simplificação da realidade, sendo construídos para compreender melhor o *software* que pretendemos desenvolver; estes ajudam a visualizar o sistema como pretendemos que ele seja, permitindo especificar a estrutura ou comportamento do mesmo. São portanto ferramentas que permitem a demonstração de como serão construídas as estruturas de dados que irão fornecer o suporte aos processos de negócios, como os dados serão organizados e quais as relações que podemos estabelecer entre eles (Debastiani, 2015). Desse modo proporcionam um guia para construção do *software*, documentando todas as decisões tomadas.

Não obstante os esforços feitos, a falha dos projetos de *software* continua em ritmo crescente. De acordo com (Barjis, 2008) uma das causas principais para esta falha continua a ser uma fraca modelação dos projetos. Reconhece-se assim a importância da modelação no entendimento e desenho de sistemas de *software*. Torna-se importante que os modelos forneçam uma introspeção passível de verificação em processos de negócio subjacentes, de modo a projetar sistemas de *software* complexos.

O processo de modelação tornar-se-á mais crucial à medida que estes sistemas crescem em complexidade e dimensão. Hoje em dia muitos destes sistemas complexos são impulsionados por modelos, sendo estes a base central do processo de implementação destes sistemas.

No que concerne à abordagem da modelação existem três perspetivas distintas (American National Standards Institute, 1975):

- Modelação conceptual: consiste numa representação de alto nível, considerando somente o ponto de vista da criação de dados. Basicamente corresponde a um diagrama em blocos, onde se demonstram as relações entre entidades e os seus atributos.
- Modelação lógica: ilustra ligações entre tabelas da base de dados, chaves primárias, componentes de cada uma, etc.
- Modelação física: demonstra o armazenamento físico dos dados. Esta abordagem incorpora a análise das características e recursos necessários ao armazenamento e manipulação das estruturas de dados, correspondendo assim a uma sequência de comandos em SQL, para a criação da base de dados, através da criação de tabelas, sua estrutura e ligações.

Para proceder a uma correta modelação torna-se também necessária a identificação de objetos, nomeadamente:

- Coisas Tangíveis: elementos com existência concreta, que ocupam lugar no espaço.
- Funções: perceção dos objetos através da função por eles exercida (papel, atribuição, classificação, capacitação, etc.).
- Eventos ou Ocorrências: alguns objetos só conseguem ser individualizados ou percebidos enquanto uma certa ação se desenrola (identifica-se características que tornam determinado fato materializável).
- Interações: resultantes das associações entre objetos em função de um processo executado - cada objeto participante da interação preserva suas características não sendo impactados pela materialização da interação.
- Especificações: são elementos que definem características de outros objetos.

De acordo com (Ribeiro, 2008) em relação à metodologia existente para desenvolvimento de *software*, existem várias opções:

- Modelo em cascata
- Modelo evolutivo
- Modelo incremental
- Modelo formal
- Modelo em espiral

2.3.1. *Modelo Incremental*

No modelo incremental os requisitos são divididos em várias iterações denominadas *builds*. Existem múltiplos ciclos de desenvolvimento, sendo que estes ciclos permitem a ocorrência de módulos mais pequenos e fáceis de gerir. Cada módulo engloba as fases de definição de requisitos, desenho, implementação e testes. Uma versão funcional de *software* é produzida após o primeiro módulo, pelo que com este ciclo de vida do software se obtém software funcional bastante cedo. Cada *release* subsequente adicionará funcionalidades à *release* anterior. Este processo continua até que todo o sistema esteja completamente desenvolvido (ISTQB EXAM CERTIFICATION, 2015).

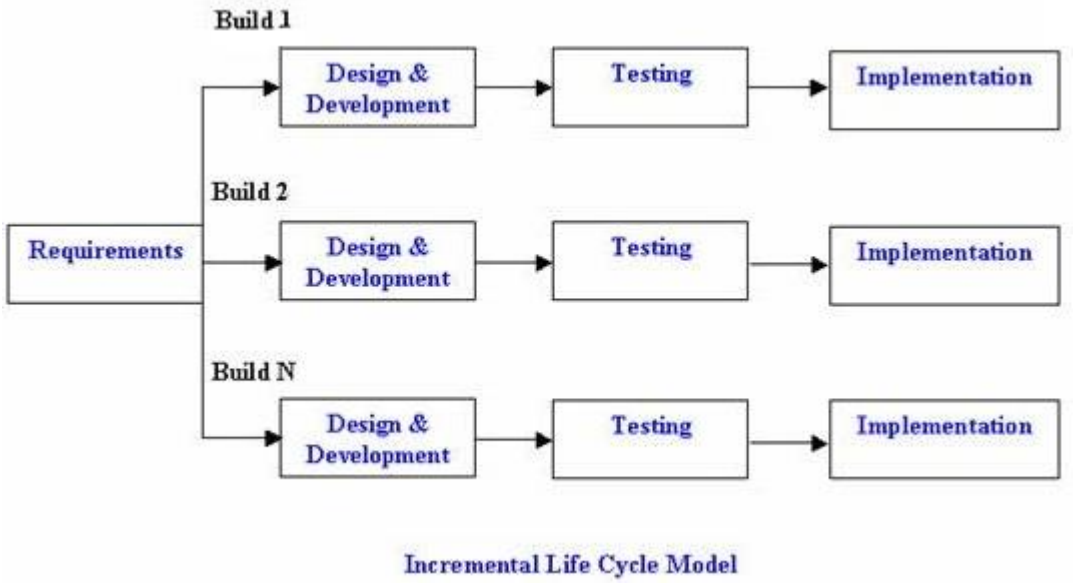


Figura 2-7- Diagrama do Modelo Incremental (extraído de (ISTQB EXAM CERTIFICATION, 2015).

De acordo com (ISTQB EXAM CERTIFICATION, 2015) algumas vantagens e desvantagens do modelo incremental são as seguintes:

Vantagens

- Gera *software* funcional rápida e facilmente durante o ciclo de vida do software
- Modelo mais flexível que outros, sendo menos dispendioso de modificar requisitos
- Mais fácil de testar e corrigir em pequenas iterações
- O cliente final fornece feedback no final de cada *built*
- Custo inicial mais baixo que nos demais modelos
- Cada *release* adiciona funcionalidades ao produto
- Risco de mudança de requisitos é reduzido
- Cada iteração é uma *milestone* facilmente gerível
- Carga de trabalho reduzida
- Fácil gestão de risco

- Núcleo do produto é desenvolvido primeiramente bem como as funcionalidades principais
- Após cada iteração são efetuados testes de regressão

Desvantagens

- Necessita de bom planeamento e *design*
- Cada fase de uma iteração é rígida e não se sobrepõe à seguinte
- À medida que se adicionam novas funcionalidades podem surgir problemas não evidentes em iterações anteriores, relacionados com a arquitetura do sistema

O modelo incremental deve ser usado perante as seguintes situações:

- Uma nova tecnologia é usada
- Quando os requisitos do sistema estão claramente definidos e entendidos
- Quando estamos perante aplicações web
- Existem funcionalidades e objetivos de alto risco
- Há necessidade de introduzir rapidamente o produto no mercado
- Quando os requisitos são claros e podem ser implementados por fases

2.4. Resumo e Conclusões

O capítulo apresentado descreve o conceito de semiótica organizacional, apresentando uma abordagem proposta por Ronald Stamper: MEASUR.

Tendo em conta os métodos apresentados nesta abordagem torna-se possível desenvolver um sistema de informação optando por diferentes métodos, sendo que neste capítulo se apresentaram três alternativas distintas de desenvolvimento de um sistema de informação. Refere-se também qual das alternativas foi usada no desenvolvimento deste sistema

Neste capítulo descreve-se também a modelação do *software* e a sua importância no ciclo de vida do mesmo. Uma vez que a modelação apresenta este relevo no desenvolvimento de *software* e tendo em conta as caraterísticas deste projeto optou-se pela modelação incremental.

Os modelos criados foram elaborados mediante máquinas de estado finitas, sendo a sua definição e estrutura explicitada neste capítulo.

Capítulo 3

3. Estado da Arte na Geração de Aplicações Web Dinâmicas

Neste capítulo pretende-se analisar algumas das várias aplicações existentes que permitem a criação automática de aplicações web, nomeadamente *websites*.

Conscientes das necessidades dos seus clientes, tais como programadores de *software* e web designers que necessitam de ferramentas que lhes permitam responder rapidamente às necessidades que lhes são solicitadas, várias são as empresas criadoras de software apostadas em criar mecanismos para uma conceção automática de aplicações para a web.

Surgem assim ferramentas CMS como resposta a esta solicitação. CMS, do inglês *Content Management System*, consiste numa aplicação informática que permite a criação, edição e gestão de conteúdos de uma forma fácil e organizada (iFourConsultancy, 2015). Este tipo de ferramenta é comummente utilizado para gerir o conteúdo de *websites*, tais como blogs, notícias, vendas, entre outros.

Basicamente, um CMS assemelha-se a uma *framework* pré-estruturada de um *website*, contendo recursos que disponibilizam facilmente ao utilizador a gestão e usabilidade desse mesmo site. Geralmente funcionam por meio de uma interface acedida através da internet.

Das várias opções existentes optou-se por realizar uma análise detalhada aos sistemas conhecidos como DotNetNuke e Telerik Sitefinity. Com esta análise pretende-se enunciar não só os pontos fortes e as necessidades que estas ferramentas pretendem solucionar, mas também as lacunas ainda não solucionadas.

3.1. Avaliação de Algumas Aplicações

A necessidade da criação de *websites* e respetiva gestão de conteúdos tem acompanhado o crescimento da utilização da internet. Para responder a esta necessidade, e tendo em conta que parte dos criadores de *websites* não possui conhecimentos técnicos (programação) foram surgindo ferramentas com o potencial de criar o website “por nós” de forma quase automática.

Estava lançado o caminho para as ferramentas de CMS. Existem várias ferramentas diferentes, sendo a sua utilização amplamente utilizada, como se pode verificar na figura seguinte.

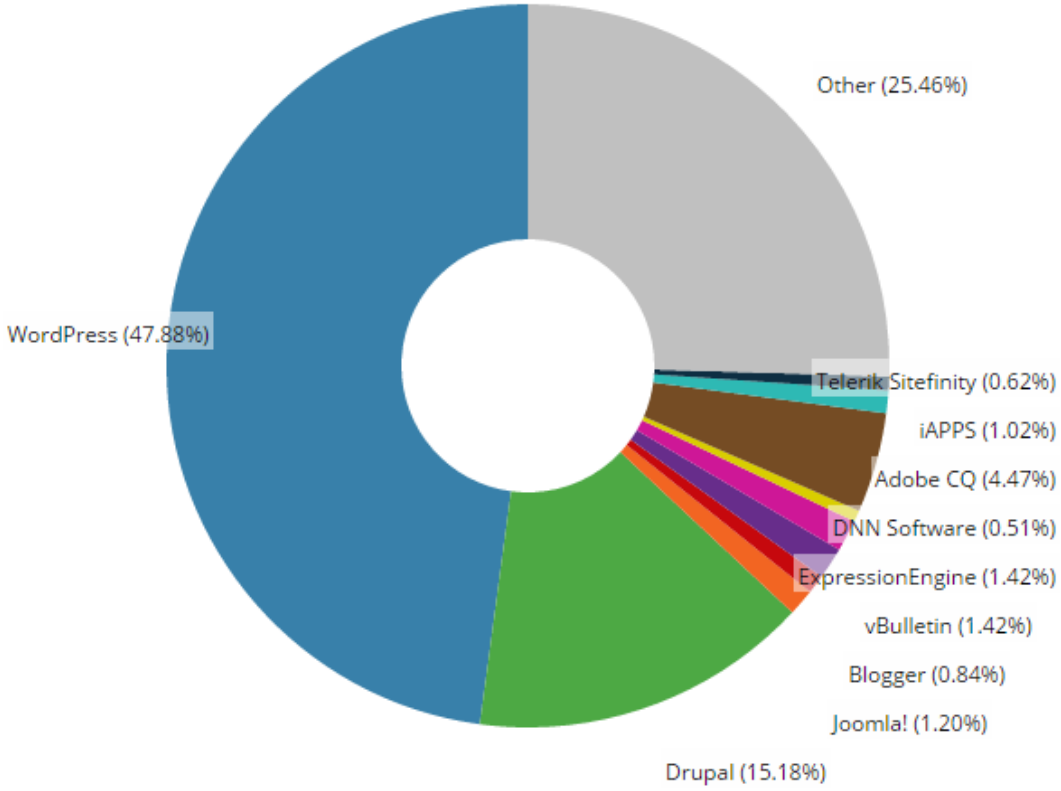


Figura 3-1- Ferramentas CMS- percentagem de utilização em websites (Obtido de (BuiltWith Pty Ltd, 2015)).

Tendo em conta que estas ferramentas apresentam características que as tornam favoráveis em determinado contexto em detrimento de outras, torna-se difícil afirmar que a ferramenta A é melhor que a ferramenta B. A realidade é que não existe O melhor sistema CMS (Johnston, 2015), mas sim um sistema CMS que apresenta melhores resultados para aquele contexto específico. Quer isto dizer que todos os sistemas possuem pontos fortes, mas no entanto uns sistemas são mais adequados para a criação de blogs, outros são mais indicados para projetos complexos, que requeiram mais do que uma *framework* e outros são melhores para a criação de websites multilinguísticos.

É importante salientar que mesmo entre estes sistemas existem diferentes graus de complexidade. Quer isto dizer que certos sistemas permitem criar *websites* sem necessidade de conhecimentos de programação, embora sejam também algo limitados ao estilo e formatação que oferecem, ao passo que existem outros mais robustos, onde se existir necessidade de adaptação ou até desenvolvimento de conteúdos, é possível fazê-lo embora implicando conhecimentos técnicos por porte do utilizador.

Começaremos por fazer uma breve análise das ferramentas mais utilizadas, nomeadamente WordPress, Drupal e Joomla.

A ferramenta WordPress é um sistema essencialmente usado na criação de blogs, embora também permita a criação de *websites*. Caracteriza-se por ser uma ferramenta gratuita e de fácil utilização, apresentando uma vasta comunidade de utilizadores e programadores, o que resulta na existência de vários *plugins* disponíveis, que permitem uma personalização do

website (Cawley, 2015). Uma parte destes *plugins* são gratuitos e podemos fazer *download* dos mesmos. (jGaunt, 2015) No entanto, torna-se demasiado dependente destes mesmos *plugins* quando se pretende um *website* mais personalizado e quanto mais *plugins* instalarmos mais a performance do nosso website será afetada. Outro problema associado a estes *plugins* é que a maior parte deles não passa por testes de qualidade (Johnston, 2015). Existiram também problemas de segurança associados ao WordPress. Uma vez que o código *open source* está disponível para todos é fácil encontrar lacunas a nível de segurança (jGaunt, 2015).

A ideia principal do Drupal consiste em tirar partido das melhores partes de outros sistemas CMS, fóruns, blogs, *wikis* e plataformas de e-commerce para criar um sistema capaz de criar qualquer tipo de aplicação web, sendo portanto um sistema CMS bastante poderoso (jGaunt, 2015). O Drupal encoraja os programadores a desenvolverem uma abordagem comum e estruturada. Esta abordagem facilita a utilização dos módulos por parte de outros utilizadores (jGaunt, 2015). Não obstante, o Drupal tem reputação de ser um sistema de difícil implementação e modificação.

No meio destes dois sistemas encontra-se o Joomla: não sendo tão fácil de usar como o WordPress também não é tão poderoso como o Drupal. O ponto positivo deste sistema é o facto de ter uma larga biblioteca de *plugins* gratuitos, os quais se podem fazer *download*. Contudo, o sistema não é muito *user-friendly*, o que significa que existe a necessidade de se instalarem *plugins* (jGaunt, 2015).

No entanto, os dois sistemas escolhidos para fazer uma análise detalhada foram os sistemas DotNetNuke e Telerik Sitefinity. Isto porque são sistemas mais robustos.

3.1.1. DotNetNuke

A plataforma DNN é um *software open source* com a intenção de possibilitar a gestão de *websites* sem a necessidade de conhecimentos técnicos elevados, pretendendo também ser extensível a um grande número de aplicativos de terceiros para fornecer funcionalidades não incluídas nos módulos centrais do DNN. Esta plataforma possibilita ainda o uso de *skins*, para mudar o visual de um *website* utilizador desta ferramenta. A plataforma DotNetNuke pode ser usada não só como CMS mas também como *framework* de desenvolvimento aplicacional.

Ao contrário da grande maioria de soluções CMS que assenta na linguagem de programação PHP, o DotNetNuke tem o *software* desenvolvido em ASP.NET. Devido à linguagem de programação utilizada, é mais adequado para servidores Windows. Como a maioria das empresas aloja as suas intranets em servidores *Windows*, torna-se uma mais valia utilizar este sistema. Outro aspeto positivo prende-se com o facto de os programadores poderem personalizar aplicações web, graças à sua API ser *open*, enquanto os utilizadores finais beneficiam de um sistema de fácil utilização (Cawley, 2015).

Algumas das características desta ferramenta são (iFourConsultancy, 2015):

- Rico editor de texto
- Gestão de arquivos
- Fácil criação de módulos
- Núcleo desenvolvido em C#
- Instalação indivual, vários portais
- API Social (versões 6.2 e superiores)
- Segurança robusta
- Administração de recursos como funções de segurança, protecção de conteúdos e log do site
- Permite envio de emails em massa
- Pronto para a *cloud* compatível com MS Azure

3.1.1.1. Arquitetura

No que diz respeito à arquitetura, o DotNetNuke usa um modelo de arquitetura em 3 camadas, com uma estrutura central de apoio à sua estrutura modular extensível (ver figura ("DNNstack" by Audiohifi at English Wikipedia. , 2015)).

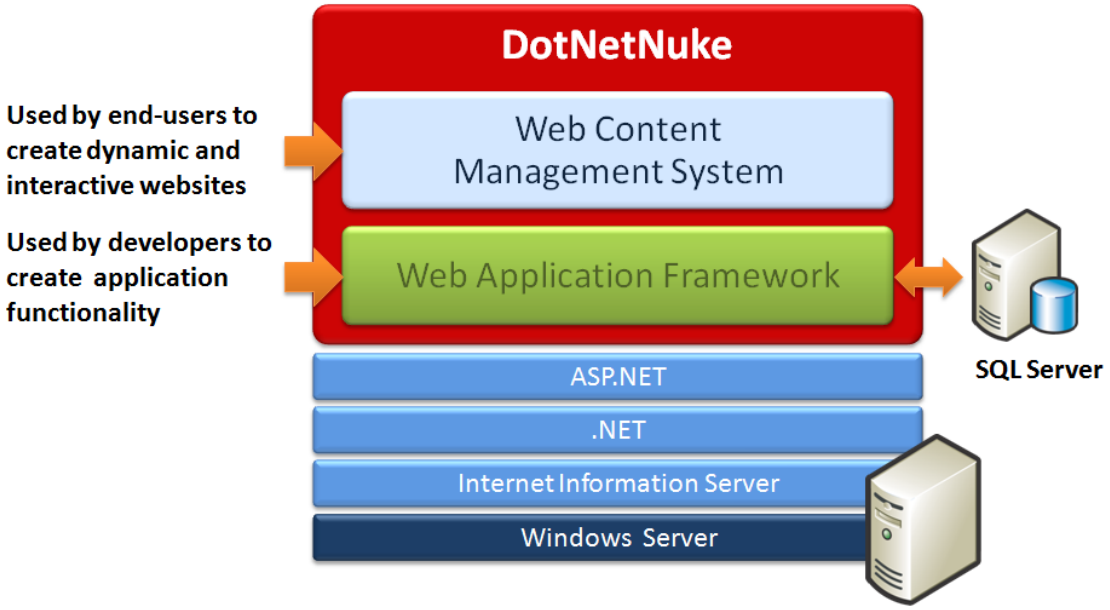


Figura 3-2- Modelo de arquitetura em 3 camadas do DotNetNuke (extraído de ("DNNstack" by Audiohifi at English Wikipedia., 2015)).

O DotNetNuke pode ser estendido usando módulos de terceiros para adição de funcionalidades. Tal permite que os vários *websites* criados com esta ferramenta possam ser personalizados de acordo com o seu criador. As versões do DNN 7.x.x requerem *Windows Server* 2008 ou *Windows Server* 2012, a respetiva versão SQL e IIS 7+.

3.1.1.2. Módulos

As funcionalidades básicas do DNN podem ser expandidas através da adição de módulos externos, provenientes tanto de uma biblioteca de módulos existente como de desenvolvimento personalizado de determinada funcionalidade. A plataforma DNN fornece funcionalidades básicas como segurança, administração de utilizadores e gestão de conteúdos, enquanto os módulos são usados para ajustar o *website* às necessidades específicas de implementação. Existe um conjunto de módulos disponíveis em conjunto com o sistema. Estes módulos providenciam funcionalidades necessárias para criar um sistema de *e-commerce*, uma intranet, um *website* público ou uma aplicação web customizada. A sua manutenção está a cargo de uma comunidade de voluntários na comunidade DotNetCommunityForge (DNN, 2015).

Um módulo pode ser carregado e instalado automaticamente numa instalação deste sistema através das páginas de administração do DNN. Após a instalação do módulo por parte do administrador, este pode ser colocado em qualquer página do *website* e as suas permissões de aceso podem ser configuradas. Em termos de linguagem de programação, um módulo pode ser desenvolvido tanto em VB.NET como em C#.

3.1.1.3. Skins

Uma *skin* descreve o *layout* visual ou aparência as páginas do website. Podem ser diferentes em diferentes conteúdos das páginas.

Uma arquitetura *skinning* fornece uma separação entre design e conteúdo, permitindo que um *web designer* crie *skins* sem a necessidade de possuir conhecimentos técnicos de programação em ASP.NET, sendo necessário somente o conhecimento de HTML e uma compreensão de como preparar e montar as próprias *skins*. Basicamente, *skins* consistem em ficheiros HTML contemplando contentores (*tokens*) para conteúdo, menus e outras funcionalidades, bem como ficheiros de suporte tais como imagens, folhas de estilo e JavaScript, compactados num arquivo ZIP. Algumas *skins* estão incluídas na instalação do DNN ao passo que outras estão disponíveis a partir de outras fontes. Todas as páginas novas apresentam o mesmo aspeto, que corresponde à *skin* Default do *website*. (Nash, 2015)

Tal como os módulos, as *skins* podem ser carregadas e instaladas automaticamente através das páginas de administração.

A figura seguinte ("DNNmodules" by Audiohifi at English Wikipedia. , 2015) ilustra como se ligam os módulos e as *skins*.

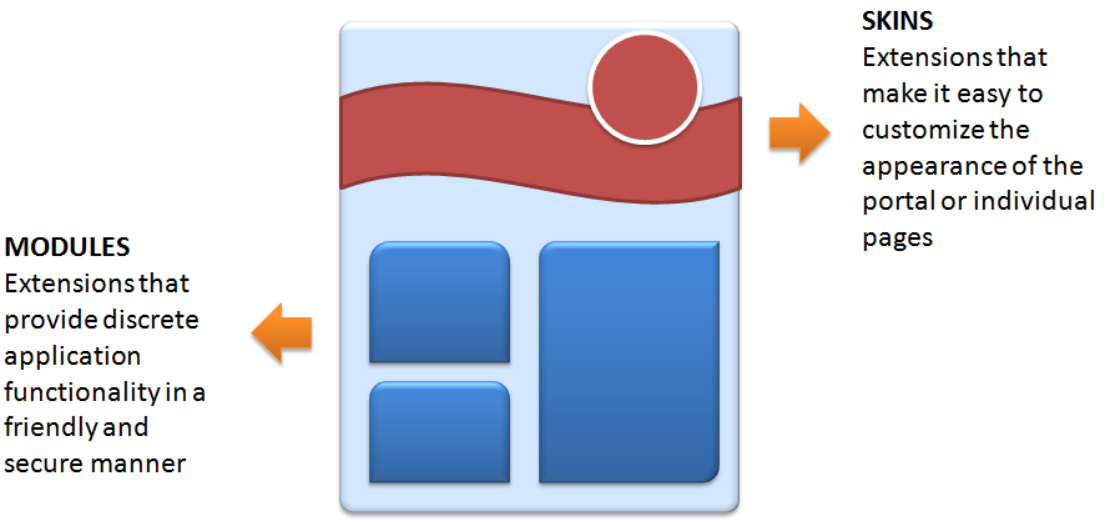


Figura 3-3- Módulos do DotNetNuke (extraído de ("DNNmodules" by Audiohifi at English Wikipedia.,2015).

Segundo (Nash, 2015), um *website* desenvolvido em DNN é composto por páginas, sendo que estas exibem informação aos visitantes do *website*. Para navegar pelas várias páginas existentes existe um item clicável denominado menu. Essas páginas são acedidas através de *links* existentes em cada um dos itens de menu existentes.

A informação existente em cada página é adicionada através da colocação de módulos nas páginas. Estes módulos são responsáveis por adicionar funcionalidades ao *website*. Diferentes tipos de módulos são disponibilizados para mostrar diferentes tipos de conteúdo. Podem ser agrupados de diferentes modos, para criar o *design* pretendido a um website específico. Os módulos podem ser simples, contendo só texto, ou podem ser módulos HTML, o que permite a inserção de código HTML ou texto consoante o pretendido pelo utilizador ou podem ser módulos mais complexos, como por exemplo uma ferramenta de *chat* online.

No que diz respeito a números, seguidamente apresentam-se alguns gráficos explicativos. No gráfico seguinte ilustra-se demograficamente a proveniência dos utilizadores do sistema.

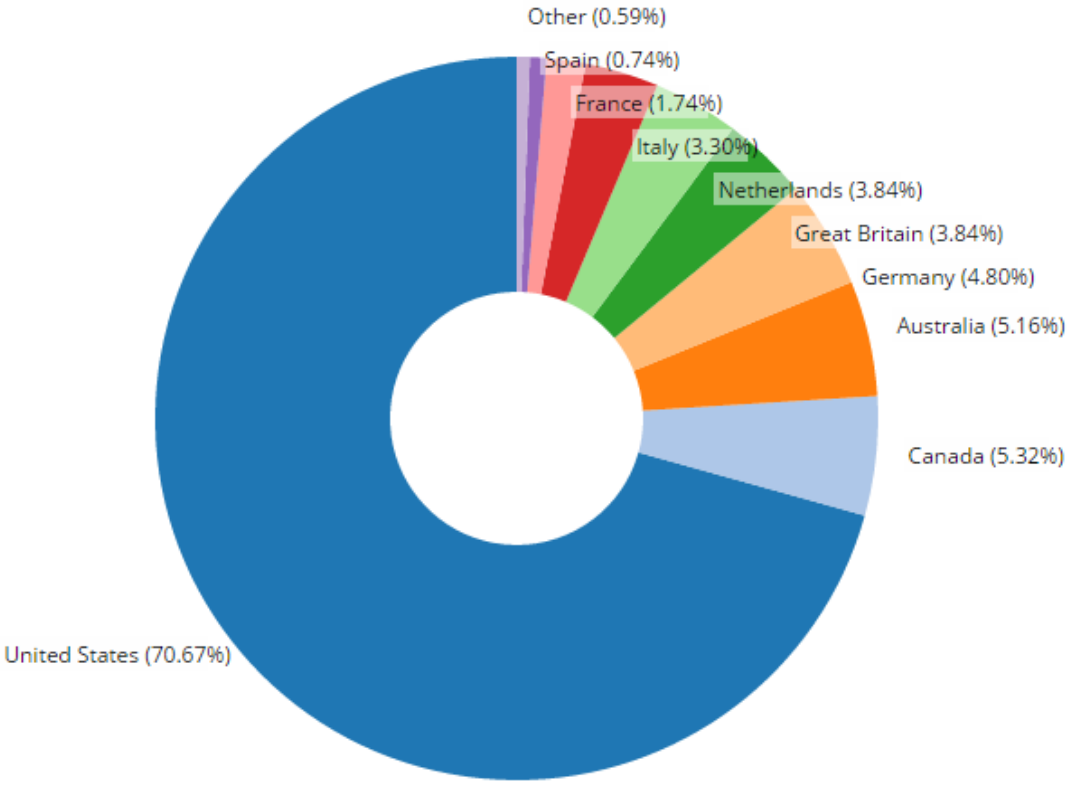


Figura 3-4- Origem demográfica dos utilizadores de DNN (Obtido de (BuiltWith Pty Ltd, 2015)).

A figura abaixo ilustra a variação de utilizadores do DNN em detrimento de outros sistemas simulares. O gráfico a verde ilustra os utilizadores que migraram de outros sistemas para o DNN e o gráfico a vermelho ilustra os utilizadores que deixaram de utilizar este sistema em prol de outros similares.

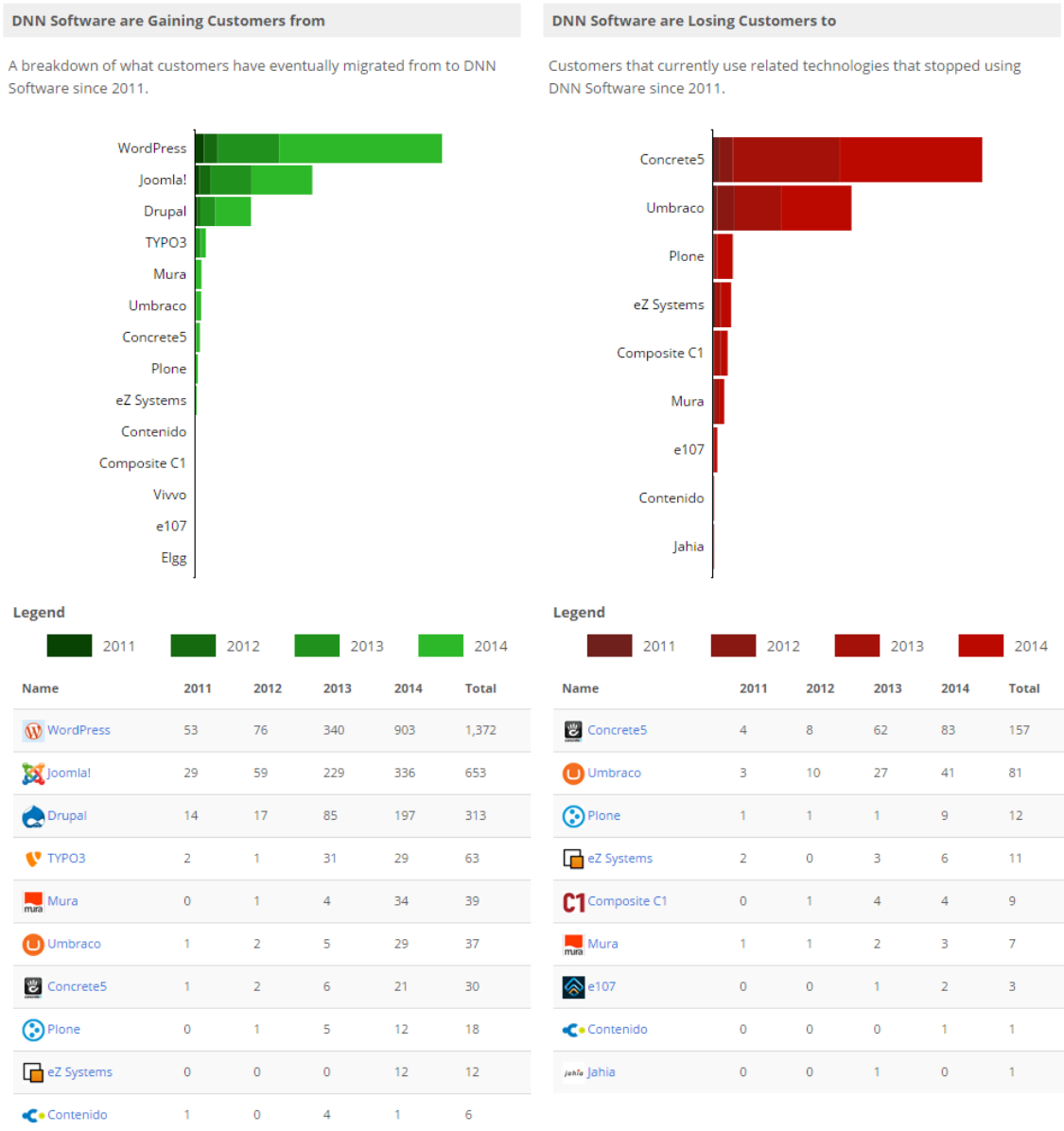


Figura 3-5-Variação de utilizadores de DNN (Obtido de (BuiltWith Pty Ltd, 2015)).

3.1.1.4. Prós da Utilização do DNN

- Ampla difusão na internet
- *Open source*
- Interface de administração bastante simples de usar
- Possibilidade de editar páginas diretamente, através de um simples clique no conteúdo pretendido (é necessário ter o login feito para executar esta operação)
- Grande variedade de extensões e módulos disponíveis. Muitos deles são livres de custos
- Dinamismo em termos de funcionalidades, características e segurança
- Prioridade em segurança: ênfase na validação, criptografia, controlo de "*bugs*" e ameaças potenciais

- Pronto a usar
- Comunidade de apoio online
- Personalizável
- Fácil de integrar
- Recursos avançados

3.1.1.5. Contras do DNN

- Depende de uma base de dados SQL, o que implica custos adicionais
- Dispende muitos recursos da máquina
- Tempo de resposta dos *websites* é elevado
- Parte do código disponibilizado está obsoleto, sendo necessário pesquisar o código para retificar a situação
- Difícil aplicação *skins*
- Difícil de aplicar a um caso específico, sendo necessário desenvolver pacotes para o DNN com código específico, ou criar *user controls* em .NET
- Dependência do URL. Se for necessário mudar o nome do domínio a base de dados usada torna-se inútil
- Não suporta múltiplas linguagens. Embora possam ser instaladas, não é possível escrever conteúdo separado para cada linguagem
- Nem todos os módulos disponíveis são gratuitos

3.1.2. Telerik Sitefinity

No que diz respeito ao Telerik Sitefinity, irmos começar pela descrição da arquitetura.

3.1.2.1. Arquitetura

A ferramenta Telerik Sitefinity consiste em cinco partes principais com uma infraestrutura comum, tal como se pode observar na figura abaixo.

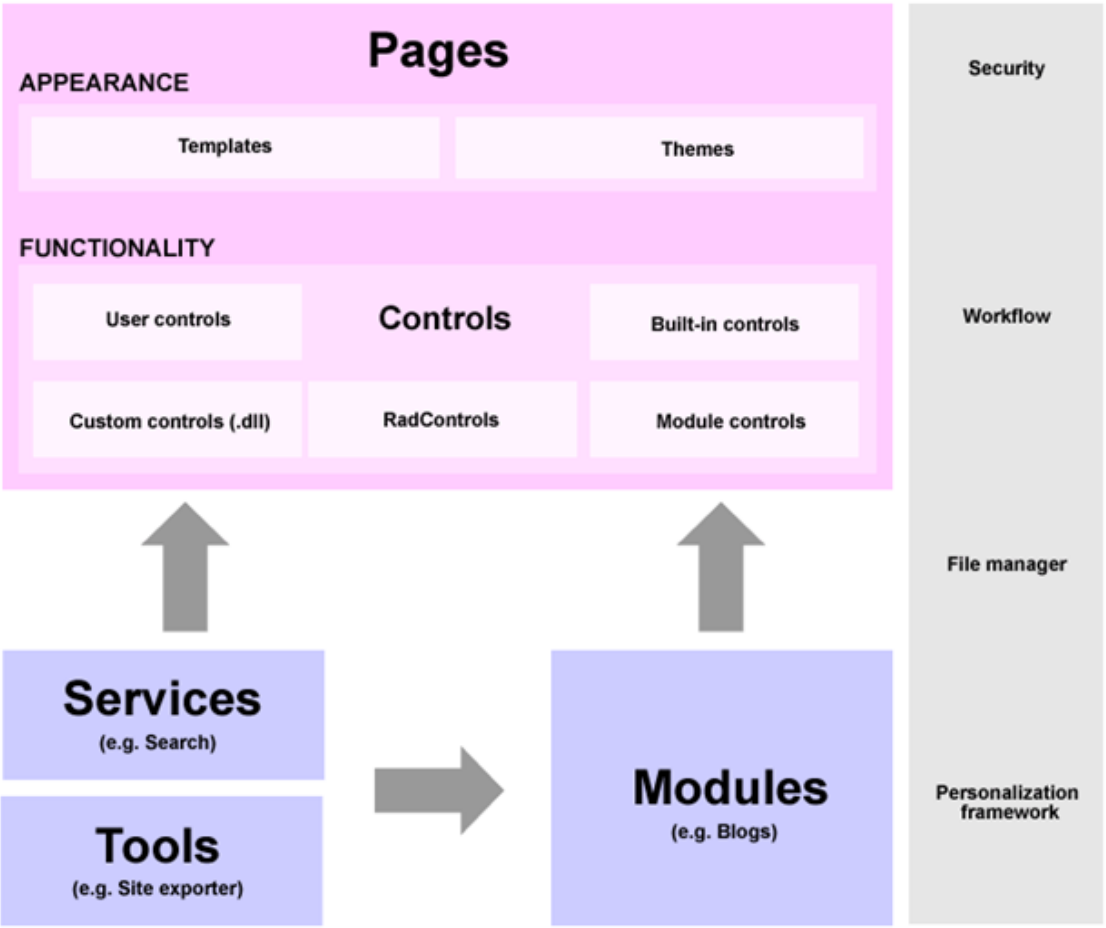


Figura 3-6- Arquitetura Telerik Sitefinity (Obtido de (Telerik, 2015)).

As principais partes do Sitefinity são: páginas, controlos, módulos, serviços e ferramentas. A infraestrutura comum partilhada pelas várias partes constituintes do Sitefinity incluem: segurança, *workflow*, gestão de ficheiros e *framework* personalizável (Telerik, 2015).

Todos os *websites* são compostos por páginas, o mesmo aplicando-se às páginas em Sitefinity. Todavia, estas apresentam algumas particularidades. A aparência de uma página é definida por *templates* e temas, ao passo que as suas funcionalidades são definidas através de controlos.

Para um melhor entendimento do conceito, criou-se a seguinte tabela comparativa:

Tabela 3-1- Comparação página HTML e página Sitefinity.

	Página HTML clássica	Página Sitefinity	Abordagem Sitefinity
Ficheiro subjacente	Sim	Não	Páginas são construídas no instante
Markup HTML definido na página	Sim	Não	*Markup* definido nos *templates*
Conteúdo definido na página	Sim	Não	Conteúdo é representado através de controlos
Folha de estilo diretamente incluída na página ou com *link* para a página	Sim	Não	Temas são aplicados à página. Podem ter múltiplas *skins* e folhas de estilo

Todas as páginas tem um determinado número de propriedades como título e URL. Para além disso as páginas servem como contentor para os controlos, os quais disponibilizam o seu conteúdo. O aspeto da página é definido pelo seu *template*, enquanto o seu comportamento (*look and feel*) são definidos pelo tema (Telerik, 2015).

Os controlos representam o conteúdo ou a funcionalidade em si. Não pode existir fora da página. O que significa que para se poder usar um controlo é necessário ter uma página na qual o controlo é colocado. Os controlos podem ser muito simples, possuindo conteúdo não estruturado como texto e imagens. No entanto, os controlos também pode ser bastante complexos, por exemplo um controlo para a criação de um fórum (Telerik, 2015).

Os módulos são pequenas aplicações existentes dentro do Sitefinity. Esta ferramenta já possui alguns módulos pré-construídos, nomeadamente Notícias, Blogs e Imagens e Documentos. Para além destes, o Sitefinity é facilmente extensível através de novos módulos. Ao criar um módulo o programador tem a opção de criar um módulo intra-site ou como *plugin*. No geral, os módulos possuem uma interface de administração e controlos públicos. A interface de administração é usada para todo o tipo de trabalho que requeira ser executado por utilizadores autorizados, ao passo que os controlos públicos podem ser colocados nas páginas como qualquer outro controlo (Telerik, 2015).

Enquanto as páginas e os controlos são tangíveis e partes visíveis do Sitefinity, outras funcionalidades comuns não são visíveis pelo utilizador final, sendo definidas como serviço. Um bom exemplo de um serviço é a Pesquisa. A pesquisa é uma funcionalidade que geralmente ocorre sem ser vista pelo utilizador. Alguns controlos mais simples disponibilizam uma caixa de texto para inserção da informação ou disponibilizam uma lista de resultados, mas é na realidade o serviço o responsável por todo o trabalho. Os serviços têm o propósito de fornecer uma funcionalidade a outras partes do Sitefinity, não tendo dificuldades em prestar o serviço às páginas, controlos e módulos.

Ferramentas são pequenas aplicações dentro do Sitefinity que geralmente não têm impacto para os visitantes do *website*. O seu objetivo é providenciar funcionalidades para o administrador do *website* ou do próprio Sitefinity. Um bom exemplo de uma ferramenta consiste num gerador de *reports*. As ferramentas não tem interface pública e localizam-se exclusivamente no lado do administrador do Sitefinity.

O gráfico seguinte ilustra a origem demográfica dos utilizadores deste sistema.

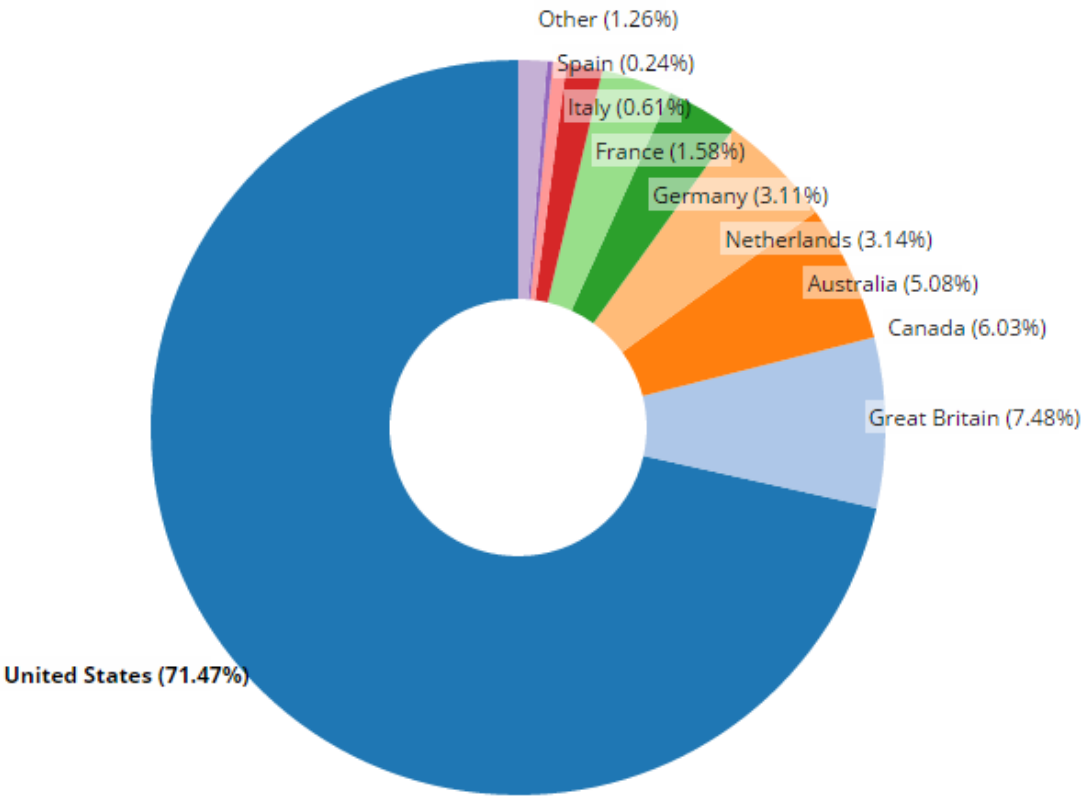


Figura 3-7- Origem demográfica dos utilizadores de Telerik Sitefinity (Obtido de (BuiltWith Pty Ltd, 2015)).

A figura abaixo (BuiltWith Pty Ltd, 2015) ilustra a variação de utilizadores do Sitefinity em detrimento de outros sistemas simulares. O gráfico a verde ilustra os utilizadores que migraram de outros sistemas para o Sitefinity e o gráfico a vermelho ilustra os utilizadores que deixaram de utilizar este sistema em prol de outros similares.

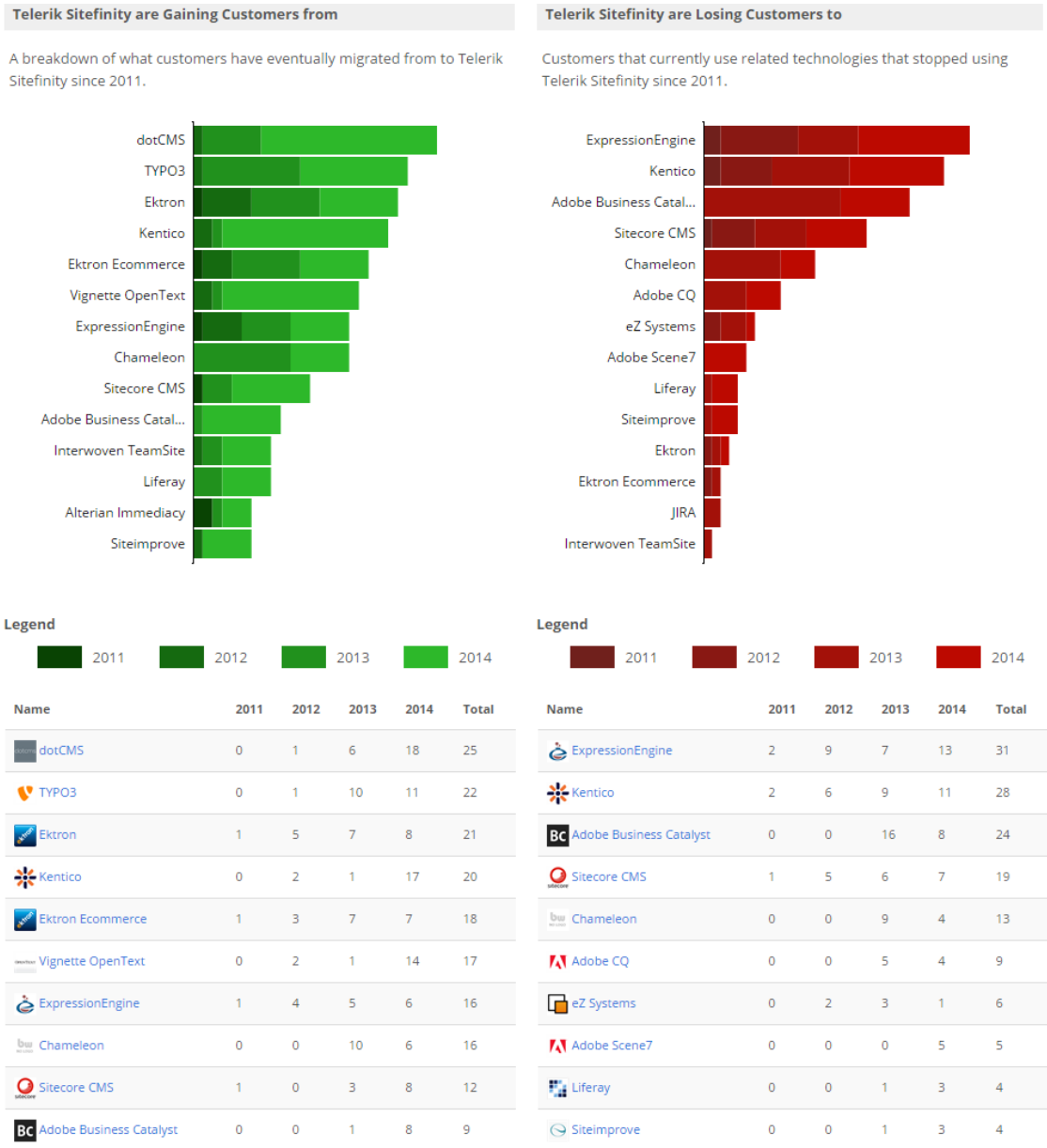


Figura 3-8- Variação de utilizadores de Telerik Sitefinity (Obtido de (BuiltWith Pty Ltd, 2015)).

3.1.2.2. Prós da Utilização do Sitefinity

- Forma simples de otimizar um *website* a partir da gestão da estrutura URL e gestão da meta-informação (Shaw, 2015)
- Suporta múltiplas linguagens
- Suporta uma taxonomia rica
- *Layout* flexível
- Templates reativos (Bakshi, 2015)

3.1.2.3. Contras do Sitefinity

- Funcionalidades limitadas
- Se pretendermos aumentar a complexidade do *website*, para além de um título e texto, torna-se necessária a existência de um programador (Bakshi, 2015).

3.2. Resumo e Conclusões

Neste capítulo foram apresentadas algumas das ferramentas existentes no mercado e utilizadas amiúde para a criação de *websites*, com especial destaque para o DotNetNuke e o Telerik Sitefinity.

Após uma análise das suas características foi necessário efetuar uma comparação de acordo com as funcionalidades consideradas importantes para uma organização que vise operar eficazmente uma ferramenta deste género.

Foi possível concluir que apesar de ambas as ferramentas terem os seus atributos, a sua adaptação a um contexto de criação de *websites* de eventos científicos, ou outro tipo de websites cuja informação existente seja fluida mediante uma variação temporal, não é a mais favorável.

Existem dificuldades de integração com a base de dados onde se encontra a informação a implementar nos *websites*, bem como uma lacuna na existência de módulos personalizados, de acordo com as necessidades da organização. Estes fatores são determinantes para um desenvolvimento de uma ferramenta viável no desenvolvimento de aplicações web a que as ferramentas estudadas não conseguem corresponder.

Capítulo 4

4. Projeto

Neste capítulo dar-se-á a conhecer a informação pertinente relativa à implementação da ferramenta descrita nos capítulos anteriores.

Começar-se-á por elaborar uma descrição deste projeto, referindo a ligação à metodologia usada, bem como os objetivos que se pretendem alcançar.

Para tal, serão referidos os requisitos necessários à implementação deste *software*. Os requisitos foram definidos tendo por base não só a análise comparativa elaborada no capítulo anterior, como por necessidades existentes na organização em que se efetuou a validação do sistema.

Neste capítulo será também apresentada a arquitetura do sistema, por meio da representação esquemática da base de dados.

Nesta secção é também mencionada a funcionalidade deste sistema. Para tal, será usado um *website* dinâmico em que a informação que nele se representa é alterada mediante um conjunto de condições ocorrentes num intervalo temporal.

Todas as ações tomadas aquando da implementação deste sistema encontram-se devidamente justificadas neste capítulo.

4.1. Descrição Conceptual

4.1.1. Programação Orientada a Objetos - Paralelismo

Os conceitos aqui apresentados podem ser facilmente percebidos, se fizermos uma analogia com alguns conceitos existentes na programação orientada a objetos.

De acordo com (Mendes & Marcelino, 2002), um objeto consiste numa combinação de dados ou atributos, também conhecidos como variáveis, e ações (ou métodos) íntima e logicamente relacionados. Um objeto caracteriza-se por três componentes, nomeadamente identidade, atributos e comportamento. A identidade basicamente permite ao sistema identificar o objeto, sendo que os atributos possibilitam a caracterização do objeto e o comportamento estabelece o conjunto de funcionalidades que o objeto pode realizar, quer sejam por opção própria quer sejam por solicitação de outros objetos.

Fazendo uma ponte entre estes conceitos existentes na programação orientada a objetos e ao conceito de máquinas de estados, o qual foi utilizado neste trabalho e que serão explicadas no capítulo seguinte, o comportamento corresponde a ações, sendo que os atributos correspondem a estados, e estes estados possuem propriedades a si associadas.

Os objetos são então definidos como instâncias de uma determinada entidade, sendo que uma entidade é algo, concreto ou abstrato, abstraído do mundo real e modelado em forma de tabela onde se guarda informação.

Segundo (Martins F. M., 2002) por classe entendem-se todos os objetos que guardarão a estrutura e o comportamento comuns a todos os objetos criados a partir dessa mesma classe, objetos esses designados por instâncias, sendo que todo o objeto criado a partir de uma dada classe exibirá todas as propriedades estruturais bem como as funcionalidades definidas pela sua classe. Podemos então afirmar que os objetos são na realidade instâncias de uma classe.

Supondo que existem duas classes em que as suas diferenças são inferiores às suas semelhanças em estrutura e comportamento, enfatizando a noção de que uma consiste apenas num caso particular ou especial da outra, faz todo o sentido a criação de um mecanismo de reutilização, de modo a que a definição desta nova classe possa ser facilmente conseguida a partir da classe semelhante já existente (Martins F. M., 2002). Tal mecanismo facilitador da definição de classes deve ser baseado nas noções de similaridade e especialização, tornando-se assim um mecanismo adicional de relacionamento entre classes para estes casos específicos. Na programação orientada a objetos a solução para esta questão passa pela introdução da hierarquia de classes. A figura seguinte ilustra um exemplo de hierarquia de classes (Palmeira, 2015).

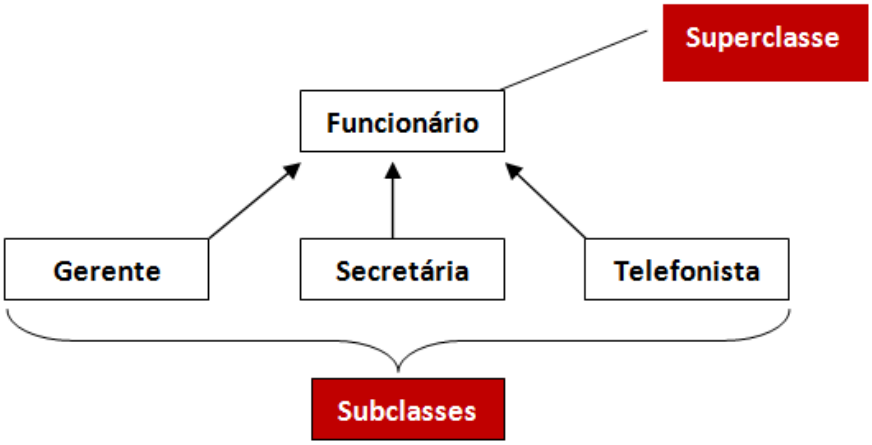


Figura 4-1- Hierarquia de classes (extraído de (Palmeira, 2015)).

Como se pode observar na figura, a classe "Funcionário" corresponde à superclasse, ou seja, uma generalização comum das restantes, sendo "Gerente", "Secretária" e "Telefonista" subclasses de "Funcionário", isto é, especializações. Quer isto dizer que "Telefonista", "Gerente" e "Secretária" são casos particulares de "Funcionário". Pode-se afirmar que estamos perante uma hierarquia de especialização, em que um a subclasse é uma extensão ou refinamento da superclasse, sendo mais detalhada que esta, por possuir mais estrutura de dados ou comportamento *(Martins F. M., 2002)*. Esta especialização pode ser simultaneamente estrutural e comportamental, uma vez que a subclasse pode necessitar de mais estrutura de dados que a superclasse para a sua representação, podendo também necessitar de um maior conjunto de métodos representativos do seu comportamento. Perante esta situação temos que nos perguntar: De que forma é que a subclasse pode reutilizar a estrutura e código da sua superclasse, que serão úteis para a sua definição, uma vez que sabemos que por definição de especialização ambas terão muito em comum? A resposta surge com o conceito de herança.

Herança corresponde a poder utilizar o que se definiu na superclasse sem ter que definir de novo, ou seja, reutilizando. O mecanismo de herança é fundamental à reutilização e partilha de código, o que facilita a conceção de *software*. Este mecanismo estabelece as seguintes propriedades entre uma subclasse B e a sua superclasse A:

- B herda de A todas as variáveis e métodos de instância que não sejam privados (ou seja próprios de A);
- B pode definir novas variáveis e novos métodos próprios;
- B pode redefinir variáveis e métodos herdados;

4.1.2. M.V.C.

MVC, acrónimo de *Model-View-Controller*, é um padrão de arquitetura de *software* usado na implementação de interfaces de utilizador e aplicações web. A premissa geral do MVC consiste na separação de conceitos bem como na reutilização de código.

O padrão MVC tem-se revelado bastante popular uma vez que isola a lógica da aplicação da camada de interface de utilizador, suportando a separação de interesses. Com o aumento da complexidade das aplicações desenvolvidas, sempre visando a programação orientada a objetos, torna-se relevante a separação entre os dados e a apresentação das aplicações. Assim alterações feitas no layout não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

Esse padrão resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o utilizador, introduzindo um componente entre os dois: o controlador.

Este padrão é composto por três camadas interligadas de forma a garantir a separação de representações internas da informação, do modo como esses mesmos dados são apresentados ao utilizador.

- Modelo (*Model*): representa a camada de negócio. Esta camada é responsável pela representação da informação, através da definição e manutenção dos dados da aplicação, regras de negócio, lógica e funções. Permite a modelação de objetos (e dados armazenados) a partir de uma base de dados. O Modelo responde a pedidos provenientes do Visualizador, respondendo também a instruções do Controlador para se atualizar.
- Visualizador (*View*): representa a camada de visualização. Trata da disponibilização de dados ao utilizador. Regra geral, os visualizadores são criados a partir de modelos. Podem existir vários visualizadores do mesmo objeto. Cabe ao Controlador decidir qual a apresentação de informação a utilizar, transmitindo essa decisão ao Visualizador.
- Controlador (*Controller*): controla a interação entre as duas camadas anteriores. É responsável por responder a *inputs* do utilizador, executando interações nos modelos dos vários objetos. O Controlo recebe e valida estes *inputs* executando de seguida as operações responsáveis pela modificação do estado do modelo de dados.

A interligação entre estas camadas pode ser explicitada mediante a figura abaixo (tutorialspoint.com, 2015).

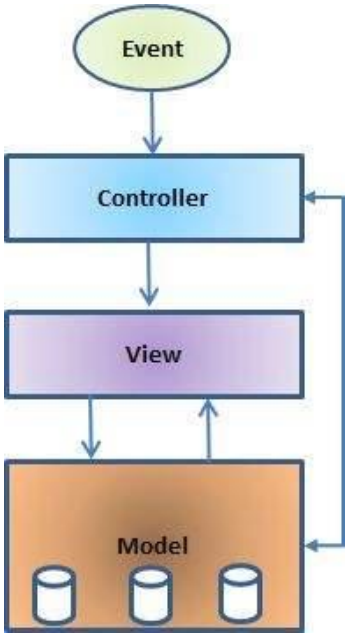


Figura 4-2- MVC: representação (extraído de (tutorialspoint.com, 2015)).

O controlador recebe todos os pedidos feitos à aplicação e seguidamente, trabalha com o Modelo no intuito de preparar toda a informação necessária para o Visualizador, o qual irá usar essa informação preparada pelo Controlador, para gerar uma resposta final para o utilizador.

Um controlador pode enviar comandos para o visualizador associado para alterar a apresentação da visualização do modelo (por exemplo, percorrer um documento). Ele também pode enviar comandos para o modelo para atualizar o estado do mesmo (por exemplo, editar um documento).

Um modelo notifica os visualizadores e controladores associados a si quando ocorre uma alteração ao seu estado. Esta notificação permite que os visualizadores produzam saídas atualizadas e que os controladores alterem o conjunto de comandos disponíveis.

O visualizador requisita ao modelo, através do controlador, a informação necessária para gerar uma representação.

4.2. Descrição Operacional

Na descrição operacional deste projeto serão mencionadas as operações necessárias à realização da atividade principal do contexto deste trabalho, ou seja, a criação automática de um *website* de um evento científico.

A criação de um *website* envolve os seguintes passos:

a) Definição das regras de negócio: Essas regras dizem respeito à definição das páginas que existirão no *website*, bem como ao modo como a informação nelas contida é visualizada pelo utilizador final. Esta informação relaciona-se com a camada C (*Control*) do padrão MVC descrito na secção anterior.
b) Definição das datas: há que determinar a data de início e término do evento. Essa ação envolve uma sequência de passos não determinante para o *website*, pelo que não serão mencionadas. Devem ser ainda determinadas as datas importantes para os utilizadores, e que surgirão no *website*.
c) Definir local de realização: esta informação depois de decidida deverá ser apresentada no *website* (detalhes do local onde se realiza o evento (local físico, cidade ou região onde ocorre, evento social, como chegar ao local, pontos de interesse, etc.).
d) Determinar pessoas chave no evento: devem ser convidadas pessoas de renome na área da conferência para desempenharem alguns papéis fundamentais no processo, tais como serem *chairs*, revisores e *keynotes*.
e) Determinar parceiros: há que incluir associações, empresas, organizações e universidades para serem parceiras do evento.
f) Definir registo de participação: há que definir os tipos de inscrição existentes, bem como os respetivos preços. Há ainda que determinar se há situações de exceção, nomeadamente descontos a membros de parceiros do evento.
g) Criação de elementos gráficos: definir o esquema de cores a utilizar pelo *website*, criar o *banner* identificador do evento, bem como o respetivo *favicon*.
h) Criação de materiais publicitários: para ajudar na divulgação do evento são criados elementos publicitários, nomeadamente um *flyer* e um *poster*, nos quais constam o resumo da informação principal do evento.
i) Criação da informação acima referida no sistema de apoio à gestão de eventos: esta informação acima referida está associada ao modelo do sistema (camada Model no padrão MVC). A informação a criar, para além da supracitada, inclui ainda detalhes do evento (tais como, nome, âmbito, data e local de realização, tópico associados, entre outros). Esta informação será disponibilizada ao utilizador final mediante as várias páginas existentes no *website*.

A determinação do modo correto de disponibilizar a informação terá em conta os vários estados e ações que podem ser realizados sobre as diversas entidades existentes. Uma ação irá conduzir a um novo estado, que possuirá propriedades (ou atributos) relativamente ao modo adequado de transmitir a informação pretendida aos vários utilizadores do *website*.

4.3. Descrição do Sistema

O sistema de *software* desenvolvido no âmbito deste trabalho assenta no desenvolvimento de uma metodologia que possibilita a modelação conceptual da estrutura e dinâmica do *website*, no contexto previamente definido, isto é, *websites* de eventos de cariz científico.

Esta modelação conceptual é baseada em modelos classificativos e estruturais, permitindo a definição de zonas nas várias páginas do *website*, descrevendo-se a sintaxe e semântica do mesmo. A representação da dinâmica de cada página web é conseguida através da utilização de máquinas de estados e modelos temporais. Serão apresentados exemplos das mesmas mais à frente neste capítulo.

Seguidamente, será descrita a estrutura do Layout deste sistema. Basicamente o ficheiro *_Layout.cshtml* é um recurso que permite a definição da estrutura comum bem como dos elementos de marcação do interface (interface *markup* elements) do *website*. Tal inclui cabeçalhos, rodapés, definições de estilo e barras de navegação. O layout fornece um modo conveniente de partilha de estrutura e conteúdo por qualquer página do *website*, denominada Content Page (ou página de conteúdos), removendo assim a necessidade de duplicação de código para elementos partilhados dentro do website. Funciona portanto de modo similar a uma *Master Page*.

Para definir secções do Layout cujo conteúdo deve ser substituído pelo conteúdo das páginas respetivas, recorre-se a espaços reservados para o conteúdo (@RenderBody()). O método RenderBody corresponde ao ContentPlaceHolder, sendo que este método indica onde se devem colocar as várias views, o correspondente às páginas de conteúdo que definem o conteúdo a inserir nos *placeholders* do Layout.

Quando se utiliza uma Master Page e respetivas páginas de conteúdos relacionados, é necessário adicionar as *tags* XHTML necessárias (tais como html, head e body) somente na Master Page, não sendo utilizadas para criar as outras páginas.

Tal como qualquer página HTML, o Layout enconta-se dividido em duas *tags* principas: <head> e <body>, o cabeçalho e corpo da página respetivamente.

O cabeçalho engloba a metadata do *website*, isto é, dados sobre os dados. A metadata de um *website* consiste nos dados do documento HTML. Esta informação não é visualizada no ecrã. A informação inserida na metadata contempla *links* externos, scripts, folhas de estilo bem como o

título do documento HTML. A *tag* body define o corpo do documento HTML. Nesta *tag* podemos encontrar todo o conteúdo do HTML, tal como texto, imagens, listas, tabelas, links, etc.

Neste Layout o cabeçalho não é exceção e nele encontram-se referências a folhas de estilo para construção do *website*, scripts (Javascript) e metadata variada, tal como se pode ver no excerto abaixo.

```
<head id="Head1" runat="server">
   <meta charset="utf-8" />
      <title>@ViewBag.Title</title>
      <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
      <meta name="viewport" content="width=device-width" />
      <meta http-equiv="content-language" content="en" />
      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
      <meta http-equiv="imagetoolbar" content="no" />
      <meta name="author" content="" />
      <meta id="PageTitle" runat="server" name="title" content="" />
      <meta id="PageLocation" runat="server" name="location" content="" />
      <meta id="PageDescription" runat="server" name="description" content="" />
      <meta name="keywords" content="" />
      <meta name="language" content="en" />
      <link rel="icon" href="App_Themes/favicon.ico" type="image/ico" />
   <link rel="shortcut icon" href="App_Themes/favicon.ico" type="image/x-icon" />
     <link rel="stylesheet" href="App_Themes/photomorpher.css" type="text/css" />
     <script type="text/javascript" src="JavaScript/photomorpher.js"></script>
</head>
```

O body é composto por vários *divs*, cujo posicionamento é ilustrado na figura abaixo.

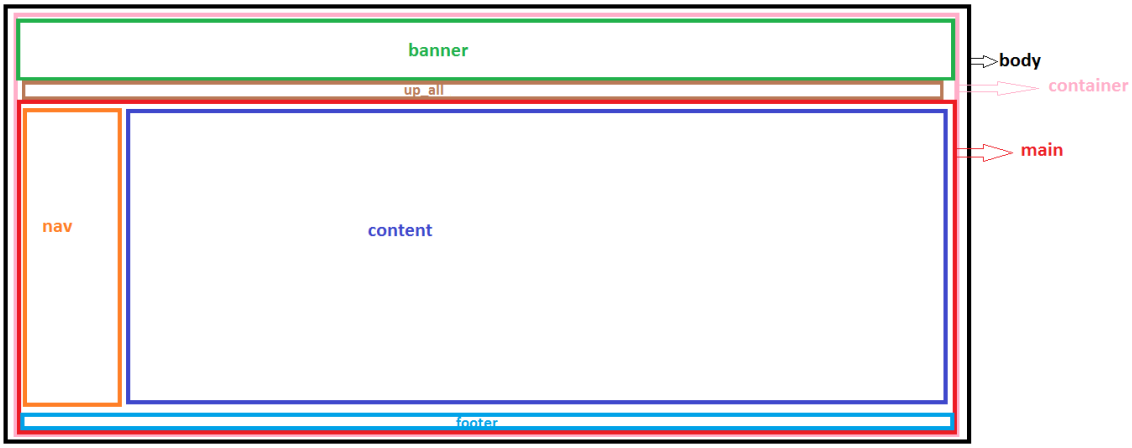


Figura 4-3- Elementos da MasterPage contidos na tag <body> em HTML.

O *div* que podemos ver a rosa na imagem, *container*, engloba todos os diversos *divs* existentes.

No *div* denominado *banner*, a verde na imagem, encontra-se uma representação gráfica identificadora do evento (nome, acrónimo, data e local de realização). Esta imagem é clicável, sendo que redireciona o utilizador do site para a *homepage* do mesmo. Este *div* é constantemente visível no *website*, independentemente da página visualizada.

O *div up_all* apresenta vários *links* informativos, tais como os contactos do secretariado responsável pelo evento, FAQs, entre outros. Este *div* é constantemente visível no *website*, independentemente da página visualizada.

O *div footer* apresenta a data do último update efectuado ao *website*, bem como a organização detentora do *copyright* do mesmo. Este *div* é constantemente visível no *website*, independentemente da página visualizada.

O *div main* é responsável por englobar o conteúdo principal do *website*, que se encontra inserido nos dois *divs* descritos de seguida.

O *div nav* é responsável por visualizar o menu com os *links* para as várias páginas disponíveis no *website*, bem como os logotipos de alguns apoios e patrocínios. Este *div* é constantemente visível no *website*, independentemente da página visualizada.

No *div content* encontramos a informação visualizada de acordo com a página em que o utilizador navega num determinado instante. Assim sendo, este conteúdo visualizado está em constante mutação.

Para proceder ao desenvolvimento deste projeto, utilizou-se programação declarativa.

A programação declarativa consiste em transmitir ao computador "o que" deve ser feito, sendo que compete ao computador decidir qual é a melhor solução para a solicitação pedida. As principais linguagens declarativas são o XML e o SQL. A programação declarativa apresenta algumas vantagens, nomeadamente a facilidade de acesso a bases de dados (SQL) e a conversão de objetos complexos por *Binding* para circularem pela rede (XML). A desvantagem deste tipo de programação prende-se com a ilegibilidade do código, quando usada de forma funcional.

Uma listagem de todas as páginas existentes e respetivas *views* que as compõem pode ser encontrada no Anexo II – Componentes de Software.

Se definirmos a data de realização do evento como T, os vários períodos temporais podem ser definidos como T – N, sendo N medido em dias. Assim, o início de vida do evento pode ser definido por T – 360. Nesta altura o evento será constituído por um número reduzido de páginas, isto pois a esta altura ainda existe pouca informação (teremos datas e informação geral no que diz respeito ao local de realização, parceiros, submissão e *templates* de artigos) a disponibilizar ao utilizador.

Para consultar as várias páginas existentes no *website* existe um menu situado lateralmente no lado esquerdo do website. No menu também a sua representação obedece a uma estrutura hierárquica, sendo que cada menu é composto por itens (denominados menuItems), que por sua vez também podem conter itens. Nos *websites* das conferências existem dois menuItems num primeiro nível: *Actions* e *Information*. Em *Actions* encontramos as diversas ações que o utilizador pode executar ao consultar o *website*, tais como submeter um artigo científico ou aceder ao seu espaço enquanto autor ou revisor de artigos, ao passo que em *Information* encontramos toda a informação existente relativa ao evento. O item *Information* encontra-se segmentado em vários itens, de acordo com a categoria de informação das páginas que o constituem. Assim, os sub-itens do menu *Information* são: Conference Details, Invited

Speakers, Satellite Events, Authors Kit, Registration, Travel and Accommodation, Local Information, Awards, Partners e Previous Conferences.

4.4. Requisitos

4.4.1. Requisitos Funcionais

Os requisitos enumerados de seguida representam as funcionalidades que são necessárias implementar como resposta à problemática em estudo. Recorrendo à técnica de MoSCoW (Internationa Institute of Business Analysis, 2015), os principais requisitos deste sistema serão priorizados mediante a sua importância no projeto. A técnica de MoSCoW é caraterizada pelas categorias descritas de seguida:

- M (*Must*) – indica um requisito que deve ser verificado no produto final para que este possa ser considerado bem-sucedido;
- S (*Should*) – representa um requisito importante que deve ser implementado, se possível. É considerado um requisito crítico, embora sem a sua implementação o *software* desenvolvido funcione corretamente;
- C (*Could*) – corresponde a um requisito não necessita de ser verificado na versão inicial do produto, podendo surgir numa versão futura se existirem recursos para tal;
- W (*Won't*) – consiste num requisito que o cliente não tenciona ver implementado numa fase inicial do *software*.

Na tabela seguinte apresenta-se a lista de requisitos funcionais, sendo que as colunas correspondem respetivamente ao identificador do requisito, sua descrição e prioridade (classificação da categoria do MoSCoW).

Tabela 4-1- Requisitos funcionais do sistema

Id	Descrição	Prioridade
RF1	O sistema deve obter informação proveniente de uma base de dados	Must
RF2	O sistema deve possuir uma camada de abstração que permita apresentar ao utilizador entidades ao invés de modelos de dados	Must
RF3	O sistema deve ser construído de forma modular e dinâmica de acordo com o paradigma orientada para objetos	Must
RF4	O sistema deve construir o website de acordo com modelos	Must
RF5	O sistema deve permitir escolher a view mais adequada para o contexto em que se insere	Must
RF6	O sistema deve permitir a reordenação da informação a ser visualizada	Must
RF7	O sistema deve permitir a atualização dos modelos	Must
RF8	O sistema deve permitir ao utilizador alterar a configuração do website	Must
RF9	O sistema deve gerar a informação de acordo com configurações do browser do utilizador final	Should

4.4.2. Requisitos Não Funcionais

Os requisitos não funcionais descrevem qualidades ou restrições do sistema e não as suas funcionalidades.

Tabela 4-2- Requisitos não funcionais do sistema

Id	Descrição	Âmbito
RNF1	A configuração do sistema deve ser acedida apenas por utilizadores autenticados	Segurança
RNF2	O sistema deve funcionar em vários *browsers*	Operacional
RNF3	O sistema deve possuir um tempo de resposta inferior a 3 segundos	Desempenho
RNF4	O sistema deve ser de fácil utilização	Usabilidade

4.5. Justificação de Opções Tomadas

Neste capítulo dar-se-á a conhecer a informação pertinente relativa à implementação da ferramenta descrita nos capítulos anteriores.

4.5.1. Base de Dados

A base de dados desenvolvida para este projeto apresenta-se segmentada em duas secções principais, cujas tabelas são mencionadas na secção 4.6. estas secções são fáceis de relacionar com o padrão MVC, sendo que correspondem às camadas *Model* e *Controller*.

4.5.2. Software

No que diz respeito ao desenvolvimento do *software* optou-se pelo desenvolvimento de uma aplicação “ASP.NET MVC 4 Web Application”, criando assim uma aplicação orientada para a *web* através da programação em ASP.NET MVC usando a WEB API da Microsoft.

Tendo em conta que a ideia do padrão MVC não só se mostrava bastante adequada ao que se pretende desenvolver neste trabalho mas também é recomendada para a criação de *websites*, foi a opção tomada para desenvolvimento do *software*.

4.6. Arquitetura

Começaremos por apresentar os diagramas de base de dados, desenvolvidos para esta aplicação, bem como a explicação das tabelas criadas.

Relativamente à arquitetura da base de dados verifica-se que a informação existente sobre cada página encontra-se aqui representada, bem como a dinâmica da página. Por exemplo, podemos indicar o nome da página e o evento em que a mesma se encontra. Associada a cada página *web* existem várias *views* (formas de visualização da informação por parte do utilizador), sendo que podemos definir várias propriedades para a *view* aplicada naquela página. Quanto mais propriedades definirmos nas *views* mais configurável será o *website* em questão.

A separação de conceitos a nível conceptual e implementacional é perceptível pela separação dos diagramas de base de dados: de um lado o modelo de negócio contendo os conceitos necessários ao evento e por outro lado os conceitos relativos à criação do *website* (páginas, *views* e propriedades).

Esta arquitetura apresenta assim a possibilidade de uma abstração dos conceitos teóricos (nível conceptual) em relação à implementação (nível implementacional).

Por exemplo, em termos conceptuais, o conceito de pessoa aplicado a este contexto, e aos vários papeis que pode ter no evento, pode ser apresentado mediante o esquema seguinte.

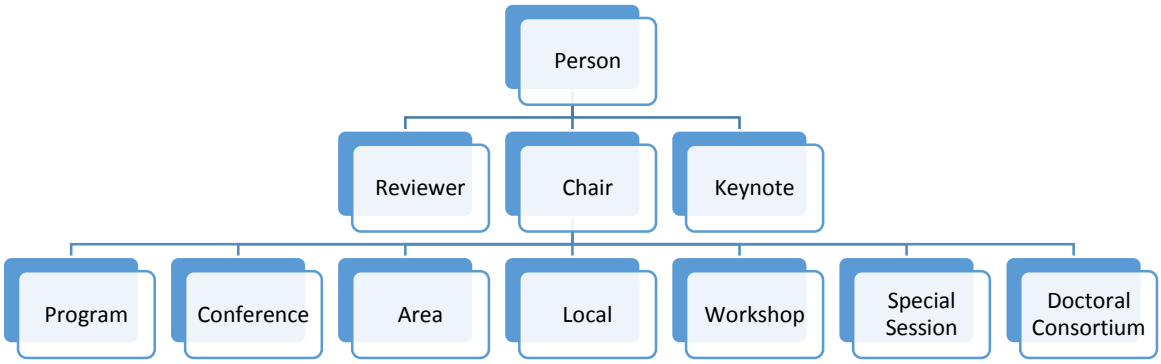


Figura 4-4- Representação conceptual da entidade Pessoa

Ou seja, uma pessoa pode assumir alguns papéis principais, como Chair, Keynote ou Reviewer, sendo que existem vários tipos de Chair, que se encontram identificados no nível abaixo. No entanto, em termos de implementação este conceito encontra-se representado mediante 4 tabelas: *Person*, *Role*, *PersonEventRole* e *Keynote*. Isto pois em termos de implementação faz sentido agregar esta informação mediante os atributos associados à classe (representação do conceito). Por exemplo, todos os chairs apresentam uma role diferente, no entanto as propriedades são comuns a todos eles (nome, afiliação, país, foto e biografia), não existindo necessidade da criação de uma tabela para cada um deles. Esta lógica aplica-se a

todos os aspetos associados a este contexto, inclusive em termos de regras de negócio, como se pode ver no esquema abaixo.

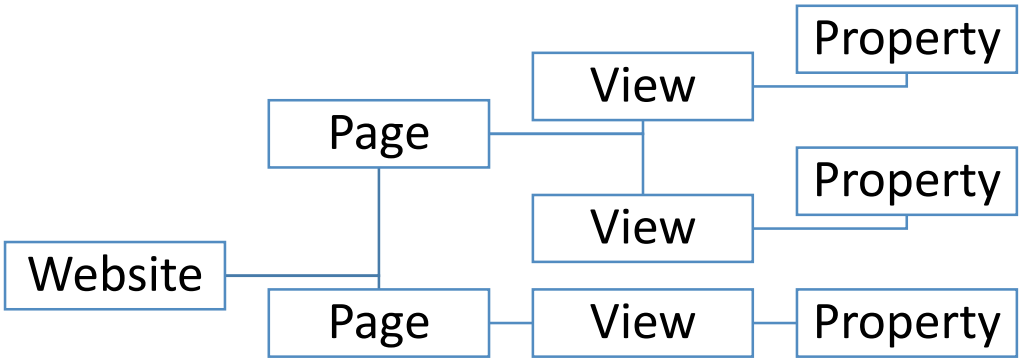


Figura 4-5- Representação estrutura website

Ou seja, um *website* é representado por um conjunto de páginas, que serão compostas por um número variável de *views* (dependendo da informação a visualizar), que por sua vez terão propriedades associadas, permitindo a sua configuração dinamicamente.

Deste modo a arquitetura permite a definição de toda a estrutura do website em base de dados, para posterior manipulação da visualização.

4.6.1. Arquitetura Aplicada ao Contexto

No exemplo concreto de desenvolvimento, a informação existente em algumas das tabelas acima mencionadas encontra-se no Anexo I.

Podemos observar a existência de dois diagramas distintos. Aplicando a ideologia do MVC, podemos constatar que o primeiro diagrama diz respeito à camada do Modelo. Nele constam as tabelas que dizem respeito à modelação do sistema. As tabelas contidas neste diagrama podem ser agrupadas em seis grupos principais: tabelas relacionadas com o programa, tabelas relacionadas com registo, tabelas relacionadas com a *venue*, tabelas relacionadas com o evento, tabelas relacionadas com o artigo e tabelas relacionadas com a pessoa.

Relativamente a registo, as tabelas existentes são:

- RegistrationType – indica o tipo de registo que se pode efetuar (neste caso membro e regular);
- RegistrationFeeType – corresponde aos tipos de tarifa de registo possíveis (*early*, *late* e *onsite*);
- EventItem – esta tabela contém o valor do registo para um determinado tipo e tarifa correspondente, num determinado evento.

O programa contempla as seguintes tabelas:

- Program – esta tabela contém informação de um programa num determinado evento;
- ProgramSession – relaciona o programa e as várias sessões existentes;
- Room – contém informação referente às salas existentes numa determinada *venue*, onde decorrem as sessões;
- Session – contempla informação relativa às sessões. Cada sessão terá uma data e sala associada;
- SessionType – tabela indicadora dos tipos de sessões existentes (podem ser do tipo *Keynote*, *Oral Presentation*, *Poster Presentation*, *Welcome Cocktail*, etc.).
- PaperSession – tabela que relaciona os artigos com a sessão. Cada sessão será composta por vários artigos (*papers*), caso se trate de uma sessão dos tipos *Oral* ou *Poster Presentation*.

As tabelas associadas à *venue* são as seguintes:

- LocalInformation – tabela com informação relativa ao local de realização do evento
- SocialEvent – tabela com informação relativa ao evento social
- Venue – tabela que relaciona informação do local e hotel, juntamente com a *venue* de realização do evento
- Hotel – tabela com informações relativas ao hotel onde o evento irá ocorrer
- PhotoLocalInformation – tabela de relação entre o local de realização do evento e as fotos usadas para a criação da página
- PhotoVenue – tabela de relação entre a *venue* do evento e as fotos usadas para a criação da página
- PhotoSocialEvent – tabela de relação entre o evento social e as fotos usadas para a criação da página
- PhotoHotel – tabela de relação entre o hotel de realização do evento e as fotos usadas para a criação da página
- EventSocialEvent – tabela que relaciona a informação do evento social ao evento
- EventVenue – tabela de relação entre a *venue* com o evento

Relacionada com a pessoa temos as tabelas:

- Person – tabela com informação da pessoa
- PaperPerson – tabela de relação entre o artigo e a pessoa
- Role – tabela contendo informação relativa aos roles existentes
- PersonEventRole – tabela relacional entre a pessoa e o role desempenhado num determinado evento
- Keynote – contem informação relativa a *keynotes*
- Country – tabela com informação dos vários países
- Photo – tabela contendo informação relativa a fotos

No que diz respeito ao artigo temos:

- Paper – contempla informação relativamente ao artigo
- PaperType – indica o tipo de artigo (*regular* ou *position*)
- EventPaperType – relaciona o tipo de artigo com o evento
- Award – contém informação relacionada com o tipo de prémio (*best paper*, *best student paper*, *best poster* e *best phd project*)
- PaperAward – tabela de relação entre o artigo e o prémio ganho
- Area – tabela contendo as várias áreas científicas existentes
- Topic – tabela com informação dos tópicos existentes
- EventTopic – tabela de relação entre o evento e os tópicos a ele associados

Entre as tabelas com informação relacionada com o evento existem:

- Event – tabela contendo toda a informação relativa à criação do evento
- EventType – tabela indicadora dos tipos de evento existentes
- PreviousEvent – relaciona a edição corrente do evento com anos anteriores
- EventMaterials – tabela indicadora de materiais
- EventMaterialsType – tabela indicadora dos tipos de materiais que podem existir no evento (flyer, poster, etc.)
- EventMaterialsEvent – relaciona os materiais com o evento
- EventDate – tabela de relação entre o tipo de data e o evento. Representa a data associada ao evento.
- DateType – informação relativa a tipos de data existentes
- EventPartner – tabela de relação entre o evento e o parceiro
- Partner – tabela com informação relativa à criação de parceiros
- PartnerType – tabela indicadora dos tipos de parceiro existentes (académico, media, industrial, etc.)
- EventPartnerType – relaciona o evento e o tipo de parceiro

O segundo diagrama corresponde à camada de controlo. Nele encontram-se as tabelas que permitem a definição das regras do negócio.

As tabelas Page, PageView e View dizem respeito mais concretamente à visualização da informação. Seguidamente apresentam-se as descrições das tabelas:

- Page – contém informação relativa às páginas existentes
- View – apresenta informação relacionada com as *views* criadas para as diferentes formas de apresentação da informação aos utilizadores (representa a camada View do padrão M.V.C.)
- PageView – tabela de relação entre páginas e *views*. Basicamente permite determinar que *views* são usadas nas várias páginas

- Property – contém dados relativos às propriedades existentes no sistema. Estas propriedades são criadas mediante as regras de negócio existentes
- ViewProperty – associa propriedades às *views*. As propriedades são importantes para determinar qual a *view* correta a aplicar a uma determinada página
- Action – tabela onde se armazenam as ações criadas no sistema. São estas ações que irão despoletar a alteração automática da informação, bem como da sua visualização no *website*
- PageViewProperty – tabela de relação entre as propriedades de uma determinada *view* aplicada numa página
- ActionViewProperty – tabela de relação entre a propriedade de uma *view* e a ação a ela associada
- Entity – armazena as várias entidades existentes no contexto de um *website*
- ViewEntity – relaciona as entidades e as *views* associadas a essas mesmas entidades

4.7. Componentes de Software

Para o desenvolvimento da aplicação foram elaborados vários componentes, que permitem a parametrização da informação para a criação dos automatismos existentes no *website*. A sua listagem pode ser consultada no Anexo II.

Para determinar que componentes deveriam ser desenvolvidos fez-se uma listagem de todas as páginas do *website* e a respetiva estrutura das mesmas (disponível no anexo III). Sabendo a estrutura de cada página torna-se mais fácil passar à implementação.

A cada página corresponde um conceito ou entidade associada ao evento. Em termos de *software* cada entidade é representada no modelo de dados por uma tabela. No que diz respeito a MVC esta representação é obtida por intermédio da criação de uma classe. Para obtenção da informação da base de dados, bem como codificação das regras de negócio, há que criar um *Controller*. O *Controller* tem a função de enviar os dados recebidos e tratados para a *View*, que fará a disponibilização dos dados para o utilizador.

4.8. Funcionamento

Para ilustrar o funcionamento do sistema, selecionaram-se duas classes e as respetivas entidades. Existem dois tipos de entidades possíveis, nomeadamente grupo e elemento. O grupo representa essa entidade como um todo, isto é, como um conjunto de elementos apresentando características enquanto coletivo. Um elemento corresponde a um único objeto daquela classe, possuindo determinadas características enquanto indivíduo.

Para cada um dos exemplos que ilustrar-se-ão a seguir, indica-se a classe, entidade e tipo de entidade correspondente. Em cada um destes tipos de entidade encontra-se associado um

conjunto de informação. Esta informação diz respeito a ações, que podem ser executadas automática ou manualmente, desencadeadas sobre a entidade, bem como os respetivos estados pelas quais ela passa. Cada ação tem um role associado, isto é, um agente do sistema responsável pela execução da mesma. Existem também informação relativa às páginas do *website* que são afetadas pelas alterações de estados, bem como os controlos que constituem essas páginas. A informação relativa às propriedades diz respeito aos atributos das entidades que sofrem alteração. Para possibilitar um melhor entendimento desta dinâmica, existe uma máquina de estados associada a estas transições de estado, bem como um diagrama temporal no qual podemos verificar a evolução dos mesmos. Para facilitar a interpretação da máquina de estados, cada ação corresponde a uma letra do alfabeto e cada estado é representado pelas iniciais das palavras que o compõem.

As classes escolhidas a título exemplificativo foram: Keynotes (entidade Keynote Lectures) e Dates (Upcoming Deadlines).

Classe: Keynotes

Entidade: Keynote Lectures

Tipo de Entidade: Grupo

Ações:
a – colocar informação no sistema
b – remover informação no sistema

A tabela abaixo ilustra o significado de um determinado estado. Por significado entende-se a sua representação no *website* mediante as propriedades deste tipo de entidade. É de salientar que as propriedades são comuns a todos os estados.

Tabela 4-3- Relação entre estados, significado e propriedades da entidade Keynote Lectures – Grupo.

ESTADOS	SIGNIFICADO	Propriedades
SE – Sem Elementos	Não existe informação a visualizar no website	• Título • Cardinalidade
UE – Um Elemento	• Título do Grupo: Keynote Speaker • Cardinalidade: 1	
VE – Vários Elementos	• Título do Grupo: Keynote Speakers • Cardinalidade: >1	

A tabela abaixo ilustra a relação entre os possíveis estados da entidade, bem como das páginas do *website* afetadas pelo mesmo. Para cada página é referido o controlo responsável por essa alteração.

Tabela 4-4- Relação entre estados e páginas afetadas pela sua alteração para a entidade Keynote Lectures – Grupo.

ESTADOS	PÁGINAS AFECTADAS	VIEW ASSOCIADA
UE, VE	Homepage	KeynoteShort
	Call for Papers	KeynoteShort
	Keynote Speakers	KeynotesPage
	Previous Invited Speakers (anos anteriores)	PreviousInvitedSpeakers

A máquina de estados parte de um estado inicial (SE, Sem Elementos), sendo que a ação indicada por a (colocar informação no sistema) faz com que o estado transite para UE (Um Elemento). Ao estarmos neste estado, se ocorrer a ação representada por b – remover informação do sistema – a máquina volta ao estado SE. Se ao invés disso a ação a acontecer, a máquina passa para o estado VE (Vários Elementos). Se neste estado ocorrer novamente a ação a máquina irá manter-se neste estado. Se ocorrer a ação b a máquina mantém-se neste estado até que o executar da ação b a retorne ao estado UE, quando só apresenta um elemento.

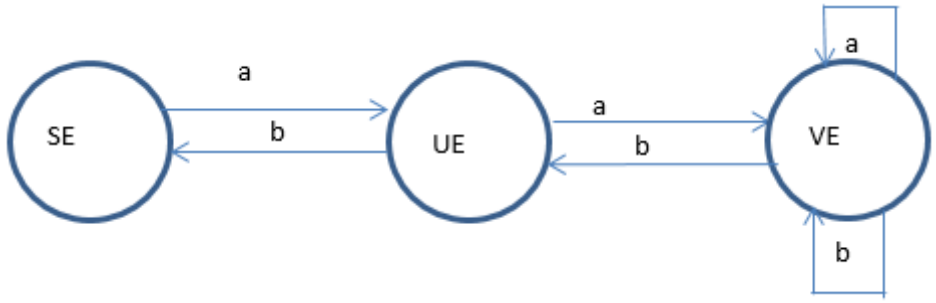


Figura 4-6-Máquina de estados para a entidade Keynote Lectures – Grupo.

Tipo de Entidade: Elemento

Ações:

a – colocar informação no sistema
b – colocar atributo is on web com valor 1
c - colocar como cancelado no sistema
d – verificar se atributo ReducedSite está a true
    || verificar se passaram 21 dias da data de cancelamento
e – “des-cancelar”
f – colocar atributo is on web com valor 0
g – atualizar informação no sistema

A tabela abaixo ilustra o significado de um determinado estado. As propriedades são as mesmas em qualquer dos estados descritos.

Tabela 4-5- Relação entre estados, significado e propriedades da entidade Keynote Lectures – Elemento.

ESTADOS	SIGNIFICADO	PROPRIEDADES
NE – Não Existente	Não existe informação a visualizar no website	• Nome • Afiliação • País • Título • Resumo • Foto • Biografia
NV – Não Visível	Se o estado anterior era Cancelado, corresponde a remover toda a informação sobre o keynote	
V – Visível	• Nome, afiliação e país na homepage • Nome, afiliação e país em secção na página de Call For Papers • Título, Nome, afiliação, país, Abstract, foto e bio na página dos Invited Speakers	
C – Cancelado	• Na homepage coloca afiliação e país a cinzento; keynote passa para o final da lista de keynotes. Após o país, coloca "(cancelled)" a itálico. • Na página de call for papers coloca afiliação e país a cinzento; keynote passa para o final da lista de keynotes. Após o país, coloca "(cancelled)" a itálico. • Na página de Invited Speakers • Na listagem remove o título; coloca a afiliação e país a cinzento; após o país coloca "cancelled" a itálico; keynote passa para o final da lista de keynotes • Na informação detalhada, coloca o keynote no final da lista; substitui o título por "* CANCELLED *"; coloca todo o texto (nome, afiliação, país, bio e abstract) a cinzento	

Tabela 4-6- Relação entre estados e páginas afetadas pela sua alteração para a entidade Keynote Lectures – Elemento.

ESTADOS	PÁGINAS AFECTADAS	VIEW ASSOCIADA
V	Homepage	KeynoteShort
	Call for Papers	KeynoteShort
	Keynote Speakers	KeynotesPage
	Previous Invited Speakers (anos anteriores)	PreviousInvitedSpeakers
C, NV	Homepage	KeynoteShort
	Call for Papers	KeynoteShort
	Keynote Speakers	KeynotesPage

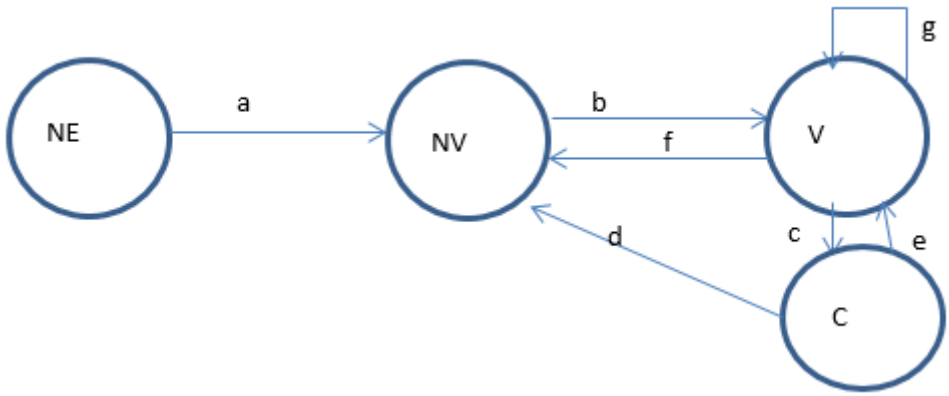


Figura 4-7- Máquina de estados para a entidade Keynote Lectures – Elemento.

Para a construção do diagrama de existência, criou-se a tabela abaixo. A sua leitura deve ser feita do seguinte modo: When (o sistema encontra-se num determinado estado), If (uma determinada ação ocorre), Then (um agente executa uma ação e o sistema transita para o caso final).

Deste modo, a primeira linha pode ser lida como: quando o sistema está no estado NE (Não tem Elementos) e uma pessoa não existe no sistema, então o secretariado cria a pessoa e atribui-lhe o role passando o sistema ao estado NV (Não visível). O mesmo se verifica para as linhas seguintes.

Tabela 4-7- Regras de construção do diagrama temporal para a Entidade Keynote Lectures – Elemento.

		Then		
When (Estado)	If	Agente	Ação	Estado Final
NE	Pessoa não existe	Secretariado	Criar Pessoa no sistema e atribuir role de Keynote Speaker	NV
NV	Is on web = false	Secretariado	Colocar is on web = true	V
V	Keynote a cancelar	Secretariado	Colocar como cancelado no sistema	C
C	Keynote novamente disponível	Secretariado	Remover atributo cancelado do sistema	V
C	Data cancelamento superior a 21 dias	Automático	Remover informação do website	NV
C	ReducedSite	Automático	Remover informação do website	NV
V	Is on web = true	Secretariado	Is on web = false	NV

Como se pode verificar no diagrama temporal abaixo, existem vários estados ao longo do tempo. A representação a tracejado indica uma ocorrência possível embora não obrigatória. Ou seja, o sistema contempla a possibilidade de um *keynote lecture* ser cancelado e deixar de estar visível no *website*, embora tal evento possa nunca ocorrer.

Figura 4-8- Diagrama Temporal para a Entidade Keynote Lectures – Elemento.

O aspeto do *website* será o indicado nas várias figuras abaixo. No exemplo aqui ilustrado, existem vários *keynote lectures* sendo que um deles se encontra cancelado. As várias imagens dizem respeito às alterações visíveis na *Homepage* do website da conferência, bem como na respetiva página de *keynote speakers*.

KEYNOTE LECTURES

Glenn Wintrich, Dell, United States
Rob Koper, Open University of the Netherlands, Netherlands
Neil Morris, University of Leeds, United Kingdom
Edmundo Tovar, Madrid Polytechnic University, Spain
Eric Mazur, Harvard University, United States *(Cancelled)*

Figura 4-9- Visualização da informação sobre Keynotes na Homepage do website.

Keynote Lectures

Glenn Wintrich, Dell, United States
Title: *When the sharing economy, IT and education collide*

Rob Koper, Open University of the Netherlands, Netherlands
Title: *How to Teach Humans (with Machines)*

Neil Morris, University of Leeds, United Kingdom
Title: *Digital technology and Higher Education: delivering benefits for student education*

Edmundo Tovar, Madrid Polytechnic University, Spain
Title: *Available Soon*

Eric Mazur, Harvard University, United States *(Cancelled)*

Figura 4-10- Visualização da informação sobre Keynotes na página KeynoteSpeakers do website: Pormenor da listagem de keynotes.

Keynote Lecture

Edmundo Tovar
Madrid Polytechnic University
Spain

Brief Bio
Edmundo Tovar (M'94–SM'06) received the computer engineering degree and Ph.D. degree in informatics from the Madrid Technical University (Universidad Polite´cnica de Madrid, UPM), Madrid, Spain, in 1986 and 1994, respectively. He is currently with the UPM as a Professor of information technology in enterprise with the Languages and Information Systems Department, and a leader of an Innovation Group in Educa- tion in the UPM focused in the application of Web technologies to OpenCourseware and OER. He has been participating in different European projects about OER. He has served as an elected member of the Board of Directors of the OpenCourseWare Consortium (2009–2013), Executive Director of the OCW Office of the UPM (2008–2012), and currently Executive Director of the Open Education Office at UPM. Prof. Tovar is a member of the IEEE Education Society Board of Governors (2005–2012) and Vice President of Educational Activities and Awards (2013–2014). He is a member on behalf of the Education Society of the University Resources Committee of the IEEE Educational Activities Board. He is a Certified Software Development Professional (CSDP) of the IEEE Computer Society (2005 to date). He is on the Steering Committee of the IEEE EDUCON and the IEEE Frontiers in Education (FIE) Conference (International and Europe Chair, 2008 to date) and is a program and planning committee member, reviewer, and chairman of several others. He is Co-Chair of FIE 2014, organized in Madrid, Spain, by the IEEE and the ASEE. He was awarded the IEEE EDUCON 2011 Meritorious Service Award (jointly with Manuel Castro) of the EDUCON 2011 conference; the 2008 Distinguished Chapter Leadership Award of the IEEE Education Society, and the 2011 Best Chapter Award (by the IEEE Region 8) and the 2007 Chapter Achievement Award (by the IEEE Education Society) for collective work in the Spanish Chapter of the IEEE Education Society. He received the award for Educational Innovation of the UPM in 2007.

Abstract
Available Soon

Assessment: The silent killer of learning

*** CANCELLED ***

Eric Mazur
Harvard University
United States

Brief Bio
Eric Mazur is the Balkanski Professor of Physics and Applied Physics at Harvard University and Dean of Applied Physics. He is a prominent physicist known for his contributions in nanophotonics, an internationally recognized educational innovator, and a successful entrepreneur. In education he is widely known for his work on Peer Instruction, an interactive teaching method aimed at engaging students in the classroom and beyond. In 2014 Mazur became the inaugural recipient of the Minerva Prize for Advancements in Higher Education. He has received many awards for his work in physics and in education and has founded several successful companies. Mazur is Chief Academic Advisor for Turning Technologies, a company developing interactive response systems for the education market. Dr. Mazur has widely published in peer-reviewed journals and holds numerous patents. He has also written extensively on education and is the author of Peer Instruction: A User's Manual (Prentice Hall, 1997), a book that explains how to teach large lecture classes interactively, and of the Principles and Practice of Physics (Pearson, 2014), a book that presents a groundbreaking new approach to teaching introductory calculus-based physics. Mazur is a sought-after speaker on optics and on education.

Abstract
Why is it that stellar students sometimes fail in the workplace while dropouts succeed? One reason is that most, if not all, of our current assessment practices are inauthentic. Just as the lecture focuses on the delivery of information to students, so does assessment often focus on having students regurgitate that same information back to the instructor. Consequently, assessment fails to focus on the skills that are relevant in life in the 21st century. Assessment has been called the "hidden curriculum" as it is an important driver of students' study habits. Unless we rethink our approach to assessment, it will be very difficult to produce a meaningful change in education.

Figura 4-11- Visualização da informação sobre Keynotes na página KeynoteSpeakers do website: Detalhe da informação.

Classe: Dates

Entidade: Upcoming Deadlines

Ações:
a – criar datas no sistema
b – verificar se data atual < dataBD
&& tipoPaper !=position
&& dataBD∈Conjunto das 5 datas mais próximas
c – verificar se data atual > dataBD
&& data atual < dataBDExtensão
d – validar se data > dataBD
&& não existe extensão
e – validar se data > dataBD
f – verificar se data atual < dataBD
&& tipoPaper == position
&& dataBD∈Conjunto das 5 datas mais próximas
&& (data atual > dataBDExt && tipoPaper != Position && tipoData = submission)
g – verificar se data atual > dataBD
&& tipoPaper = position
&& data Atual < dataBDExtensão && (data atual > dataBDExt && tipoPaper != Position & tipoData = submission)

A tabela abaixo indica o significado de um determinado estado, sendo que o significado corresponde à sua representação no *website*, tendo em conta as propriedades da entidade. As propriedades são as mesmas em qualquer dos estados descritos.

Tabela 4-8- Relação entre estados, significado e propriedades da entidade Upcoming Deadlines.

ESTADOS	SIGNIFICADO	PROPRIEDADES
NE – Não Existente	Não existe informação a visualizar no website	Font-Bold
NV – Não Visível	Remove datas	
V – Visível	Datas (RP e PP do evento) surgem a bold	
VE – Visível Extensão	Datas (RP e PP do evento) surgem a bold	

Na tabela seguinte temos a relação entre os possíveis estados da entidade e as páginas do *website* que são afetadas pelo mesmo. Para cada página são referidos os controlos responsáveis por essa alteração.

Tabela 4-9- Relação entre estados e páginas afetadas pela sua alteração para a entidade Upcoming Deadlines.

ESTADOS	PÁGINAS AFECTADAS	VIEW ASSOCIADA
V, VE, NV	Homepage	UpcomingDeadlines

A máquina de estados parte de um estado inicial (NE, Não Existente), sendo que a ação indicada por a (criar datas no sistema) faz com que o estado transite para NV (Não Visível). Ao estarmos neste estado, se ocorrer a ação representada por b – validar se data atual é inferior à data existente na base de dados, o tipo de artigo é regular e a data do sistema pertence ao conjunto das 5 datas mais próximas – ou por f (validar se data atual é inferior à data existente na base de

dados, o tipo de artigo é *position*, a data do sistema pertence ao conjunto das 5 datas mais próximas, e valida também se data atual é superior à data de extensão quando se trata de um regular *paper* e o tipo de data diz respeito a submissão) a máquina passa para o estado V, Visível. Se no estado V ocorrer a ação d (validar se data atual é superior à data existente no sistema e não existe data de extensão) a máquina transita para o estado NV. Ainda no estado V, o despoletar das ações c (validar se data atual é superior à data da base de dados e validar se data atual é inferior à data de extensão existente no sistema) e g (validar se a data atual é superior à data existente no sistema, para *papers* do tipo *position* em que a data atual é inferior à data de extensão, validando também se para *papers* do tipo regular a data de submissão é superior à data de extensão existente) fará a máquina transitar para o estado VE (Visível Data de Extensão). Ao se encontrar neste estado a máquina irá passar para o estado NV após ocorrência da ação representada por e (validar se data atual é superior à data da base de dados).

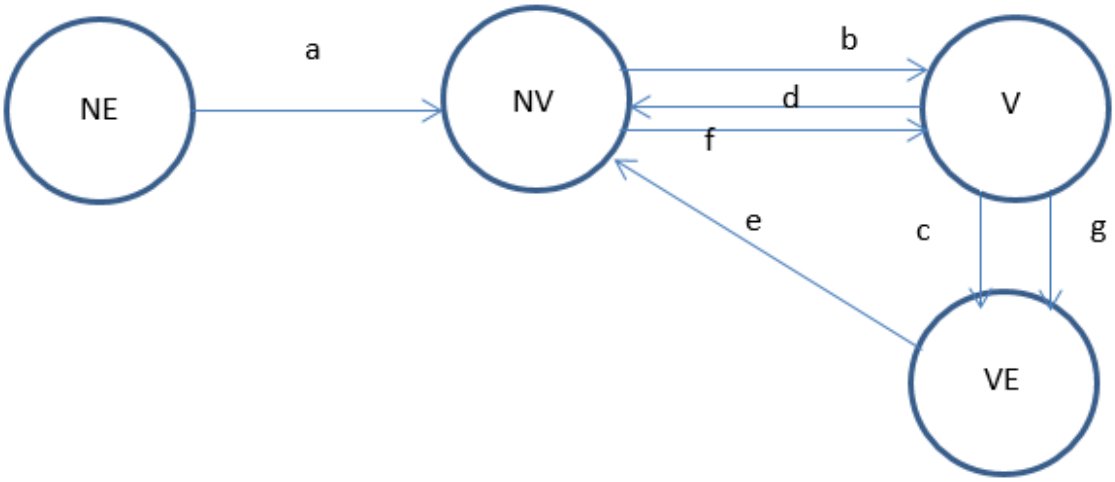


Figura 4-12- Máquina de estados para a entidade Upcoming Deadlines.

Para facilitar a construção do diagrama temporal criou-se a tabela abaixo. A coluna When corresponde ao estado inicial do sistema, sendo que a coluna If indica a ocorrência de uma determinada ação e Then representa um agente executante de uma ação que faz o sistema passar para um novo estado.

Tabela 4-10- Regras de construção do diagrama temporal para a Entidade Upcoming Deadlines.

		Then		
When (Estado)	If	Agente	Ação	Estado Final
NE	Não existem datas no sistema	Secretariado	Inserir datas	NV
NV	Data atual inferior à data da BD e o tipo de paper é regular e data da BD faz parte do conjunto das 5 datas mais próximas	Automático	Visualizar data	V
V	Data atual superior à data BD e data atual é inferior à data de extensão da BD	Automático	Visualizar data	VE
V	Data atual superior à data BD e não existe extensão de data	Automático	Remover data	NV
VE	Data atual superior à data BD	Automático	Remover data	NV
NV	Data atual inferior à data da BD e tipo de paper é position e data da BD faz parte do conjunto das 5 datas mais próximas e data atual é superior à data de extensão da submissão de regular papers	Automático	Visualizar data	V
V	Data atual é superior à data da BD e tipo de paper é position e a data atual é inferior à data de extensão da BD e a data atual é superior à data de extensão de submissão de regular papers	Automático	Visualizar data	VE

Como se pode verificar no diagrama temporal seguinte existem vários estados ao longo do tempo. A representação a tracejado indica uma ocorrência que embora não seja obrigatória é passível de ocorrer. Por exemplo, se não existir uma data de extensão de prazo, o estado do sistema transita de Visível (V) para Não Visível (NV) após o término da data em questão.

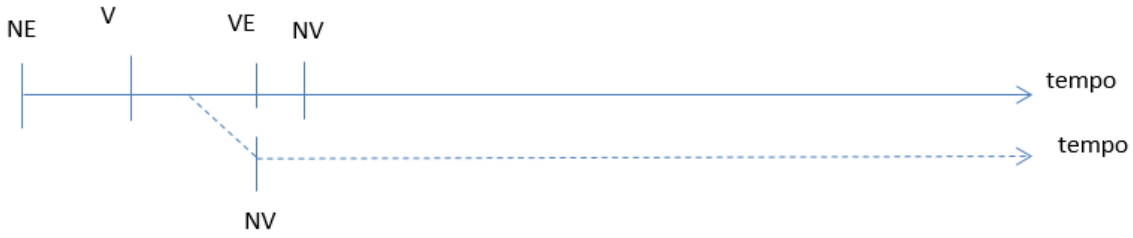


Figura 4-13- Diagrama Temporal para a Entidade Upcoming Deadlines.

O aspeto do *website* será o indicado nas várias figuras abaixo. No exemplo aqui podemos visualizar as datas visíveis inicialmente, isto é, datas que surgem quando ainda não foi atingido nenhum dos deadlines definidos no sistema.

UPCOMING DEADLINES
Regular Paper Submission: **September 8, 2015**
Regular Paper Authors Notification: **November 26, 2015**
Regular Paper Camera Ready and Registration: **December 14, 2015**

Figura 4-14- Datas inicialmente visíveis na Homepage.

Na figura seguinte podemos ver a alteração ocorrida após o primeiro deadline ser alcançado.

Figura 4-15- Datas após primeira mudança de estado.

Seguidamente podemos observar que após o término do prazo de submissão surgem novas datas.

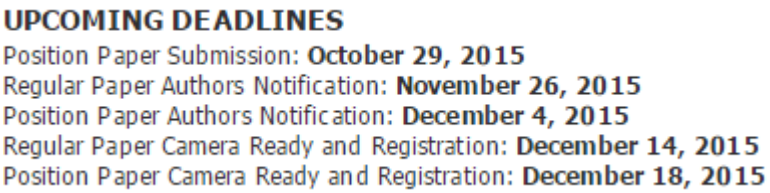


Figura 4-16- Datas visíveis após deadline de extensão expirar.

4.9. Resumo e Conclusões

Neste capítulo para além da descrição da base teórica de contextualização do projeto, apresenta também as caraterísticas técnicas do sistema desenvolvido.

Foram revistas algumas noções de programação orientadas a objetos que apresentam um paralelismo com este trabalho, nomeadamente a noção de entidades, classes e objetos.

É também referida a importância dos conceitos de hierarquia e herança de classes, que serão utilizados na conceção do sistema de *software* a desenvolver no âmbito deste trabalho.

Para além das noções de programação orientada a objetos acima referidas, refere-se também o padrão M.V.C., devido à sua ampla utilização em aplicações web e os benefícios que dele derivam. A sua premissa de separação de conceitos e reutilização de código fazem dele um padrão extremamente popular na atualidade e essa ideologia foi implementada neste trabalho, devido à clara necessidade de separação de conceitos.

Em termos de descrição operacional são referidos, com uma pequena descrição, os passos principais à criação de um *website*.

Após uma breve descrição do sistema são mencionados os seus requisitos, a partir dos quais foram definidos os respetivos módulos. Na definição dos módulos e respetivas caraterísticas tornou-se necessário explicitar e justificar algumas das opções tomadas, de modo a garantir que o sistema desenvolvido fosse ao encontro da abordagem escolhida. É também mencionada a arquitetura do sistema, bem como uma listagem dos componentes que o constituem. Finalmente são demonstradas algumas funcionalidades do sistema, nomeadamente a gestão de um *keynote speaker* e as diferentes datas limites.

Capítulo 5

5. Validação do Projeto

Neste capítulo será apresentada a validação efetuada ao sistema desenvolvido, estando este inserido num contexto organizacional.

A validação de um sistema de *software* é uma fase importante no seu ciclo de vida. A validação deve responder à questão “Estamos a construir o produto certo?”, ou seja, há que garantir que o *software* faz aquilo que o utilizador pretende que ele faça. Os objetivos da validação são descobrir erros no sistema e averiguar se o sistema é efetivamente útil e usável nas situações a que se destina. Deste modo, a validação permite-nos possuir confiança no *software*, isto é, confiar que o software desenvolvido é adequado ao seu propósito de utilização. A confiança no *software* pode derivar de três fatores, nomeadamente a função do software – quão crítico este é para a organização, das expectativas do utilizador e do próprio ambiente de marketing envolvente.

As atividades de validação podem possuir duas vertentes:

- Estática
    - Inspeção de *software*: análise da representação estática do sistema para deteção de problemas. Esta inspeção pode ser complementada pela análise da documentação existente, bem como pela análise de código.
- Dinâmica
    - Testes de *software*: consiste em exercitar o sistema observando o seu comportamento operacional. Para tal, o sistema é executado com dados de teste e o seu comportamento é observado e analisado.

Um método para se proceder à correta validação do sistema consiste em realizar uma comparação com os outros sistemas estudados no capítulo 3, de modo a clarificar os pontos fortes e fracos do sistema desenvolvido no âmbito deste trabalho.

Para proceder à verificação do sistema torna-se necessário garantir que o sistema é capaz de produzir um conjunto de elementos que vão de encontro às regras previamente definidas.

5.1. Método de Validação

Para validar o *software* desenvolvido no âmbito desta tese há que responder a duas questões fulcrais:

- O sistema funciona como previsto?
- Quais são as mais-valias apresentadas por este sistema em relação aos demais sistemas existentes, em particular aos sistemas já referidos no capítulo 3?

Para podermos responder à primeira questão é importante definir qual é o funcionamento previsto do sistema. Para tal, é também importante que exista uma clara definição das fronteiras do contexto de aplicação do sistema.

Para dar resposta à segunda questão referida há que executar um teste comparativo entre este sistema e os sistemas referidos no capítulo 3, nomeadamente DotNetNuke e Telerik Sitefinity. Com esta comparação direta pretende-se averiguar se o sistema desenvolvido nesta tese é mais útil aos utilizadores do que os outros sistemas já existentes dentro do contexto já mencionado.

Com base nestas questões é fácil de perceber a importância da qualidade do *software*. Na figura abaixo (Martins E. , 2015) encontra-se uma representação do modelo de qualidade proposto por McCall em 1977.

Como se pode observar, existem três vertentes de fatores de qualidade, nomeadamente características operacionais (relacionadas com a utilização do produto), habilidade para ser alterado (diz respeito às alterações do produto) e adaptabilidade a novos ambientes.

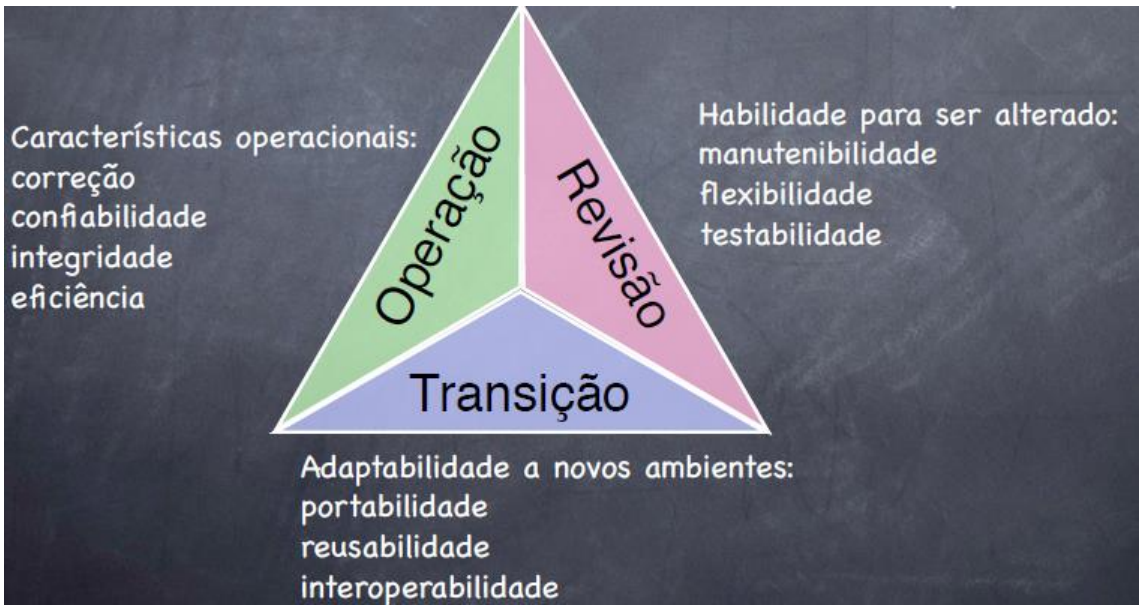


Figura 5-1- Modelo de Qualidade de McCall et al, 1977 (extraído de: (Martins E. , 2015)).

Embora todos estes aspetos sejam importantes, iremos focar apenas alguns deles, nomeadamente:

- Confiabilidade – o quanto um programa executa a função pretendida com a precisão exigida

- Manutenibilidade – o esforço necessário para localizar e eliminar erros num programa
- Flexibilidade – esforço necessário para modificar o programa

Tendo em conta estes aspetos, as características de qualidade abordadas serão a funcionalidade e manutenibilidade.

De modo a determinarmos a funcionalidade do sistema, isto é, se o *software* satisfaz as necessidades do utilizador, é necessário determinar as fronteiras de utilização do sistema.

No que diz respeito à utilidade deste sistema esta encontra-se otimizada para *websites* com evolução temporal da informação. Neste caso em particular, o tipo de *websites* criados com esta ferramenta são sites de organização de eventos científicos, embora entre este tipo de sites se possa incluir a organização de eventos de cariz lúdico, leilões e sites de compras online com descontos. Em todos estes tipos de *websites* a ideia principal é a modificação de alteração em algum ponto do tempo, sendo que essa modificação possa ser despoletada automaticamente (ao atingir uma determinada data por exemplo), ou manualmente (por exemplo o cancelamento de alguma atividade).

Uma vez estabelecidas as regras de negócio e a informação seja inserida na base de dados, o sistema tem capacidade de adaptar a apresentação da informação consoante o *website* a que diz respeito.

Posto isto, podemos afirmar que existe flexibilidade por parte deste sistema, desde que devidamente usado no contexto em questão.

Relativamente à manutenibilidade do sistema, comecemos por definir o que é a manutenção. De acordo com (Amorfab, 2015) por manutenção entende-se qualquer trabalho no *software* feito depois que ele se torne operacional ou passa para a produção. Este trabalho pode ser correcção de erros, revisão de requisitos originais e aumento de funcionalidades e performance.

Para determinarmos se o nosso sistema funciona como pretendido (primeira questão a responder) há que identificar as várias fases que constituem o ciclo de vida do evento, averiguando se conseguimos em cada fase realizar as várias tarefas associadas à mesma o mais automaticamente possível. A listagem das várias fases que constituem um evento, bem como a sua ocorrência cronológica, é disponibilizada no anexo IV.

Sabendo as várias fases bem como as suas ações principais há que introduzir esta informação corretamente na base de dados criada para este efeito. Ao tentar fazer a inserção na base de dados, constatou-se ser possível inserir com sucesso as várias ações decorrentes ao longo do ciclo de vida do evento. Com base na informação criada e listada no anexo III foi também possível inserir com sucesso a informação das páginas e a sua estrutura, representada por intermédio das views e suas propriedades.

Um exemplo concreto desta validação é ilustrado seguidamente com a página Program Committee.

Esta página pode ser visualizada mediante a escolha de uma das várias *views* existentes para a mesma. Por exemplo, podemos observar os nomes a *bold* seguidos da afiliação e país numa das *views*, ao passo que outra view mostrará apenas o nome ao utilizador.

Começamos por escolher a páginas, das várias existentes, tal como se pode obervar na figura abaixo.

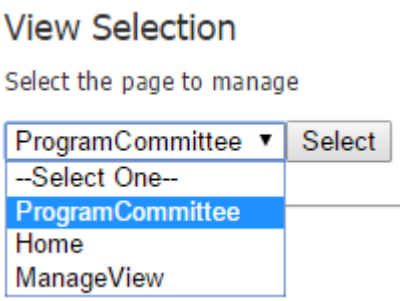


Figura 5-2- Escolha de página a gerir

Após escolher a página são disponibilizadas as várias *views* existentes, bem como as propriedades a escolher.

View Name	Property Visible	
ProgramCommitteeMembersComplete	true	
ProgramCommitteeMembersShort	false	turn on

Figura 5-3- Visualização de views e suas propriedades

Ao selecionarmos a *view ProgramCommitteeMembersComplete*, este será o aspeto da página:

Figura 5-4- Aspeto da página Program Committee após escolha de uma view

Para ativar a outra *view* disponível, mudando desse modo o aspeto da página, basta clicar na opção “turn on”, não sendo necessária nenhuma intervenção por parte do *webmaster* para modificar o aspeto da página no website.

View Name	Property Visible	
ProgramCommitteeMembersComplete	false	turn on
ProgramCommitteeMembersShort	true	

Figura 5-5- Mudança de atributo

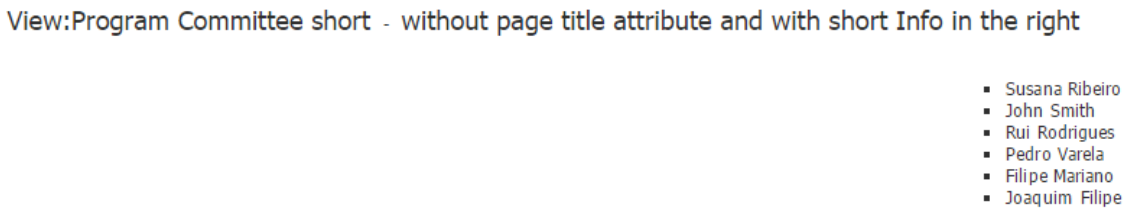


Figura 5-6- Aspeto da página após escolha da outra view

Deste modo verifica-se que o sistema faz o pretendido, permitindo a alteração do *website* mediante a modelação do mesmo na base de dados, não necessitando desse modo da intervenção do *webmaster* ou programador.

5.2. Resumo e Conclusões

A validação é uma fase fundamental para a verificação da utilidade e qualidade do sistema.

Para efetuar a verificação do sistema efetou-se parte do ciclo de vida do *website*, de modo a determinar se era possível a alteração do *website* através da modelação do mesmo em base de dados, sem necessidade de intervenção humana ou nova programação.

Após essa verificação foi possível concluir que o sistema cumpriu com sucesso os objetivos que desencadearam o seu desenvolvimento.

Capítulo 6

6. Conclusão

Neste capítulo encontrar-se-ão as conclusões relativamente ao projeto desenvolvido, bem como o trabalho futuro que pode ser desenvolvido por forma a melhorar a metodologia criada bem como o próprio sistema de criação de *websites*.

6.1. Conclusões

A circulação de informação na internet atinge valores cada vez mais elevados, bastando um clique para que qualquer internauta aceda a uma panóplia de dados sobre o tema desejado. Para garantirmos que a nossa informação é acedida e consultada pelo maior número de pessoas possível há que garantir que a informação se encontra atualizada dispendendo do menos tempo e recursos possível. Para a criação de *websites* de suporte à gestão de eventos de cariz científico, o contexto definido para este trabalho de mestrado, a atualização constante de informação é uma necessidade premente, tal como a adaptação do mesmo às regras da atividade, sendo necessário gerir vários eventos em simultâneo. Desse modo surgiu a necessidade de desenvolvimento de uma metodologia que permita a adaptação do modelo do *software* à realidade deste contexto, reutilizando assim a estrutura existente para os vários *websites*.

A metodologia desenvolvida assenta na definição declarativa das várias páginas do *website*, tal como as suas respetivas secções. Para auxiliar ao entendimento da dinâmica de cada página *web* foram utilizadas máquinas de estado. Desse modo é possível entender quais os vários estados pelos quais a informação pode passar, bem como quais são as ações que os originam. Mediante a página *web* a visualizar cabe ao sistema determinar qual a *view* mais apropriada para visualização dos conteúdos solicitados pelo utilizador.

Com esta metodologia criou-se um sistema de *software* capaz de implementar diretamente os modelos das várias páginas, evitando assim a intervenção do *webmaster* para aplicar as várias alterações ocorridas ao longo do ciclo de vida do evento. Estas ações podem ser despoletadas pelo gestor da conferência ou de forma automática, fruto de certas condições do sistema (prazos temporais por exemplo), no contexto do modelo dinâmico dos processos associados à gestão deste tipo de eventos. A ocorrência destas alterações para além de representadas no modelo de dados, são repercutidas automaticamente e com efeito imediato na visualização dos dados do website.

As vantagens desta metodologia são claras, uma vez que é possível fazer a gestão do website através da modelação da base de dados ao invés de depender de um programador para criação/atualização de conteúdo, tal como acontece frequentemente com os sistemas estudados

no capítulo 3. Desta forma reduzem-se custos de tempo e mão-de-obra na organização que implemente esta metodologia.

A modelação revela-se de crucial importância, não só para esta tese de mestrado, mas para qualquer projeto de *software*, visto que os modelos correspondem a uma simplificação da realidade ajudando assim não só a visualizar o sistema mas também determinando o seu comportamento esperado, permitindo deste modo um maior entendimento sobre o *software* a desenvolver.

Assim, modelar o sistema antes do desenvolvimento propriamente dito contribui para uma menor probabilidade de falha no projeto, sendo que atualmente uma das principais falhas no desenvolvimento dos projetos de *software* se prende com uma modelação defeituosa e insuficiente (Barjis, 2008).

O modelo deve fornecer uma introspeção passível de verificação pelo que a sua manutenção não deve ser descurada, de forma a garantir a manutenção do mesmo.

Neste trabalho, a modelação foi elaborada mediante a implementação do padrão MVC, permitindo assim uma separação de conceitos entre o modo como a informação é disponibilizada ao utilizador final e o modo como a mesma é obtida de acordo com as regras do negócio.

O padrão MVC mostrou-se extremamente ajustável ao trabalho aqui apresentado na medida em que a separação de conceitos possibilita que alterações feitas no *layout* do *website* não afetem a manipulação de dados, tal como uma reorganização de dados não implica uma alteração no *layout*.

No *software* aqui desenvolvido, a camada *View* apresenta os vários modos de visualização da informação no *website*, sendo que a escolha do modo de visualização a utilizar é definida mediante a informação contida na camada de controlo, onde se determinam as regras a aplicar a cada página do *website*. Estas regras definem quais as propriedades (atributos) que devem ser utilizadas de acordo com a página em questão. Tal permite que a mesma informação possa ser visualizada de formas diferentes.

A arquitetura desenvolvida com este sistema revela-se bastante útil na medida em que permite a separação de conceitos da camada conceptual e implementacional, permitindo um maior entendimento do sistema como um todo, não descurando o entendimento das partes. A manipulação de visualização de conteúdos ao utilizador final mediante o modelo inserido na base de dados é outra mais-valia desta arquitetura, possibilitando assim a atualização da informação sem necessidade recorrente de um *webmaster* ou programador.

Contudo, esta metodologia apresenta algumas limitações, tanto a nível do modelo estático como do modelo dinâmico.

As limitações do modelo estático prendem-se com o facto da necessidade da edição de conteúdo, ou código, para implementar alguma alteração de textos ou processos. Uma vez que

uma organização é um ser vivo, em constante mudança, estas alterações são passíveis de ocorrer.

Em termos do modelo dinâmico, estamos limitados perante as views existentes, pois são elas que disponibilizam a informação ao utilizador. A alteração de textos ou processos, implica a criação de uma nova view, de modo a disponibilizar esse pedido.

Ao existir um modelo dinâmico cada *view* criada deverá ter um conjunto de propriedades de modo a tornar toda a sua visualização configurável, passando a responsabilidade da alteração da mesma para um não programador, eliminando assim a necessidade de intervenção de um programador ou *webmaster*. Uma maior versatilidade nas views exigirá menos manutenção, além de que a existência de mais *views* possibilita um maior leque de opções ao utilizador.

6.2. Trabalho Futuro

Uma vez que a metodologia desenvolvida nesta tese já permite automatizar uma parte dos *websites*, embora não na totalidade não se pode afirmar que este trabalho se encontra concluído. Como tal, existem ainda alguns aspetos a desenvolver com o intuito de completar e melhorar esta metodologia.

Como trabalho futuro seria interessante ter um *template* do *website*. Este *template* seria uma estrutura em base de dados (nova tabela) que conteria as páginas e as *views* associadas a essas páginas, tais como as suas propriedades, que por defeito (*default*) um *website* deveria conter. Assim, quando se pretendesse criar o *website* de um novo evento bastaria apenas associar o evento ao template numa tabela em base de dados criada para esse efeito, ao invés de estar a associar página a página ao evento, bem como *views* a cada página e configurar propriedades dessas mesmas *views*. Em termos do modelo, tal template representa uma pequena alteração, nomeadamente a criação da estrutura em base de dados e o respetivo mecanismo de ligação destes conceitos.

Tendo em conta as limitações referidas na secção anterior, uma forma de remover uma dessas limitações seria a colocação do texto como propriedade da view, algo que será possível quando esta metodologia estiver implementada na íntegra. Assim, a propriedade *default* teria o texto padrão a colocar nos *websites*, sendo que podemos alterar esta propriedade especificamente para uma dada página, eliminando dessa forma a programação na totalidade.

No que diz respeito aos processos, se as propriedades forem pensadas numa perspetiva de apoio à mudança, podemos eliminar a programação de novas *views*. Por exemplo, se tivermos uma lista de pessoas podemos logo pensar no que pode ser alterado em termos de visualização (como campos a apresentar, ordenação, etc.), se conseguirmos analisar estas possíveis alterações pode-se efetuar a programação da *view* logo de início antevendo estas mudanças e evitando assim futuras alterações.

Bibliografia

(2015). Obtido de "DNNstack" by Audiohifi at English Wikipedia. : http://commons.wikimedia.org/wiki/File:DNNstack.png#/media/File:DNNstack.png

(2015). Obtido de "DNNmodules" by Audiohifi at English Wikipedia. : https://en.wikipedia.org/wiki/DotNetNuke#/media/File:DNNmodules.png

American National Standards Institute. (1975). *ANSI/X3/SPARC Study Group on Data Base Management Systems.* Interim Report. FDT (Bulletin of ACM SIGMOD) 7:2.

Amorfab. (2015). Obtido de Manutenção de Software: http://pt.slideshare.net/amorfab/manuencao-de-software

Bakshi, S. (2015). *The Battle of the Blogging Platforms: Sitefinity vs. WordPress*. Obtido de http://www.matrixgroup.net/snackoclock/2014/02/the-battle-of-the-blogging-platforms-sitefinity-vs-wordpress/

Barjis, J. (2008). The importance of business process modeling in software systems design. *Science of Computer Programming 71*, 73-87.

Brito, R. C., Martendal, D. M., & Oliveira, H. E. (May de 2015). *Máquinas de estados finitos de Mealy e Moore.* Obtido de http://www.inf.ufsc.br/~barreto/trabaluno/TC_roberta_diogo_henrique.pdf

BuiltWith Pty Ltd. (19 de Março de 2015). *CMS technologies Web Usage Statistics*. Obtido de http://trends.builtwith.com/cms#

Cawley, C. (18 de Março de 2015). Obtido de 10 Most Popular Content Management Systems Online: http://www.makeuseof.com/tag/10-popular-content-management-systems-online/

Debastiani, C. A. (2015). *Definindo Escopo em Projetos de Software.* Obtido de https://novatec.com.br/livros/defescopo/capitulo9788575224298.pdf

Deretta, G. P. (2014). *Finite state machines*. Obtido de http://www.crystalclearsoftware.com/soc/coroutine/coroutine/finite_state_machines.html

DNN. (Março de 2015). *DNN Forge*. Obtido de DNN Software: http://www.dotnetnuke.com/Development/Forge.aspx

EDASIM. (2015). Obtido de EDASIM: http://www.edasim.com/br/un-8/artigo-47

Gazendam, H. W., Jorna, R. J., & Liu, K. (2004). Round Table Workshop 'An organizational semiotic view on interculturality and globalization' at the IASS 2004 Conference . Lyon.

iFourConsultancy. (2015). *Comparative analysis of .NET based content management systems - Kentico, DotNetNuke and Umbraco*. Obtido de http://pt.slideshare.net/ifourconsultancy/comparative-analysis-of-net-based-content-management-systems-kentico-dot-netnuke-and-umbraco

Internationa Institute of Business Analysis. (2015). *A Guide to the Business Analysis Body of Knowledge.* Obtido de http://www.p2080.co.il/go/p2080h/files/4071469962.0.pdf

ISTQB EXAM CERTIFICATION. (2015). *What is Incremental model- advantages, disadvantages and when to use it?* Obtido de http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/

jGaunt. (2015). *Top 10 Content Management Systems 2015*. Obtido de http://jgaunt.hubpages.com/hub/Top-10-Content-Management-Systems

Johnston, M. (18 de Março de 2015). *WordPress, Joomla and Drupal are NOT the Best CMS*. Obtido de CMS Critic: http://www.cmscritic.com/wordpress-joomla-and-drupal-are-not-the-best-cms/

Lima, E. S. (15 de Setembro de 2014). *INF 1771 - Inteligência Artificial (2014.1)*. Obtido de Edirlei Soares de Lima: http://edirlei.3dgb.com.br/aulas/ia_2013_1/IA_Aula_22_Maquinas_de_Estados_Finitos_2013.pdf

Martins, E. (2003). *Index of /~eliane/Cursos/MO409/Curso2003/Apresentacoes.* Obtido de Eliane Martins: http://www.ic.unicamp.br/~eliane/Cursos/MO409/Curso2003/Apresentacoes/Ontologia-SO.ppt

Martins, E. (2015). *Qualidade de Software*. Obtido de http://www.ic.unicamp.br/~ranido/mc626/Qualidade.pdf

Martins, F. M. (2002). *Programação Orientada aos Objectos em JAVA 2.* FCA.

Mendes, A. J., & Marcelino, M. J. (2002). *Fundamentos de Programação em JAVA 2.* FCA.

Nash, C. (2015). *Benefits of DotNetNuke*. Obtido de http://www.datasprings.com/resources/dnn-tutorials/artmid/535/articleid/6/benefits-of-dotnetnuke?AspxAutoDetectCookieSupport=1

Palmeira, T. (2015). *Abstração, Encapsulamento e Herança: Pilares da POO em Java*. Obtido de http://www.devmedia.com.br/abstracao-encapsulamento-e-heranca-pilares-da-poo-em-java/26366

Ribeiro, A. M. (2008). *Um Processo de Modelação de Sistemas.* Obtido de http://repositorium.sdum.uminho.pt/bitstream/1822/8579/1/tese%20final.pdf

Shaw, S. (2015). *A .NET CMS comparison and review*. Obtido de http://www.branded3.com/blogs/a-net-cms-comparison-and-review/

Telerik. (2015). *Sitefinity CMS*. Obtido de http://www.telerik.com/help/sitefinity/developer-manual/sitefinity-architecture.html

tutorialspoint.com. (2015). *Basic MVC Architecture*. Obtido de tutorialspoint.com: http://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm

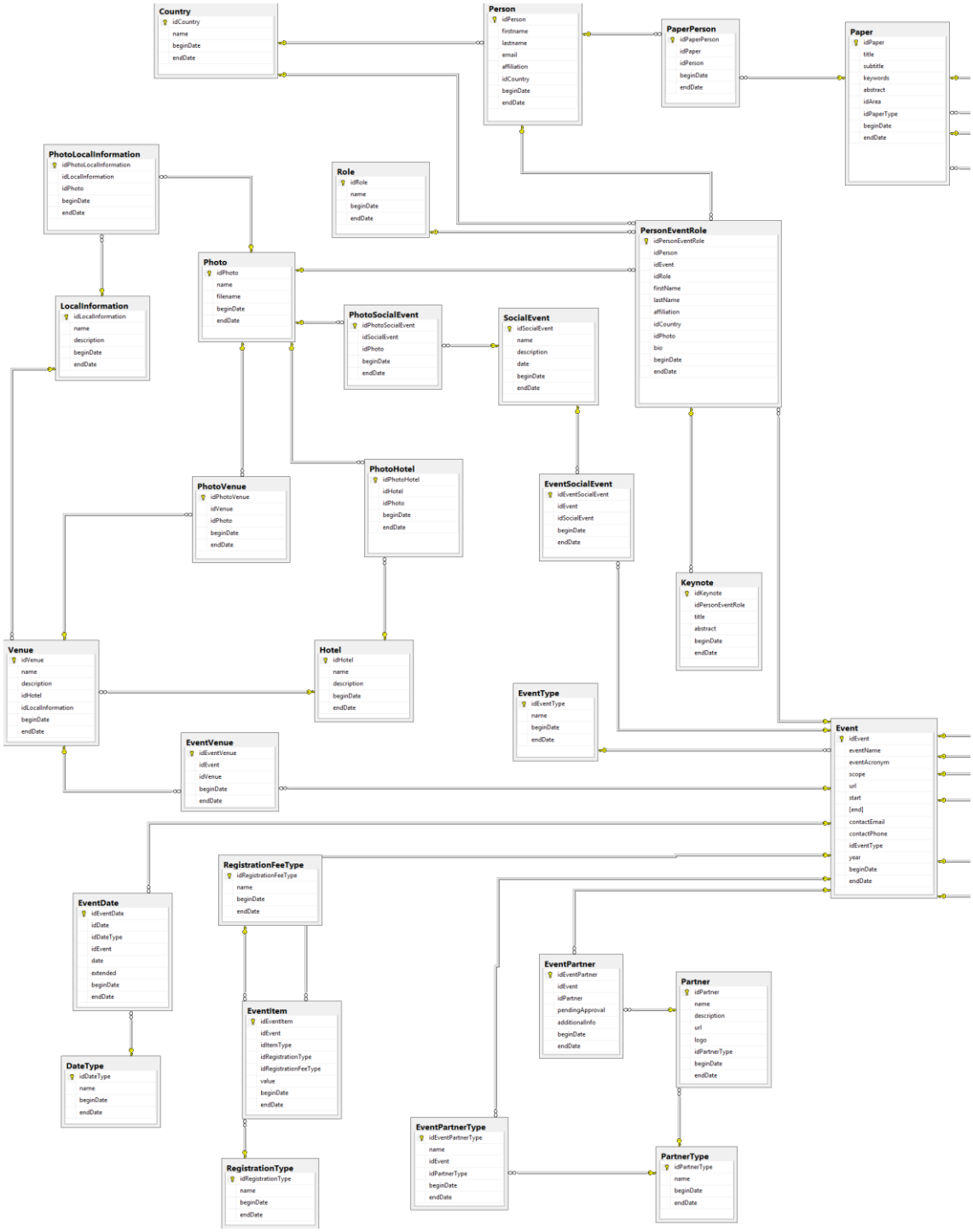
Wagner, F. (2005). *Moore or Mealy model?* Obtido de http://www.stateworks.com/active/download/TN10-Moore-Or-Mealy-Model.pdf

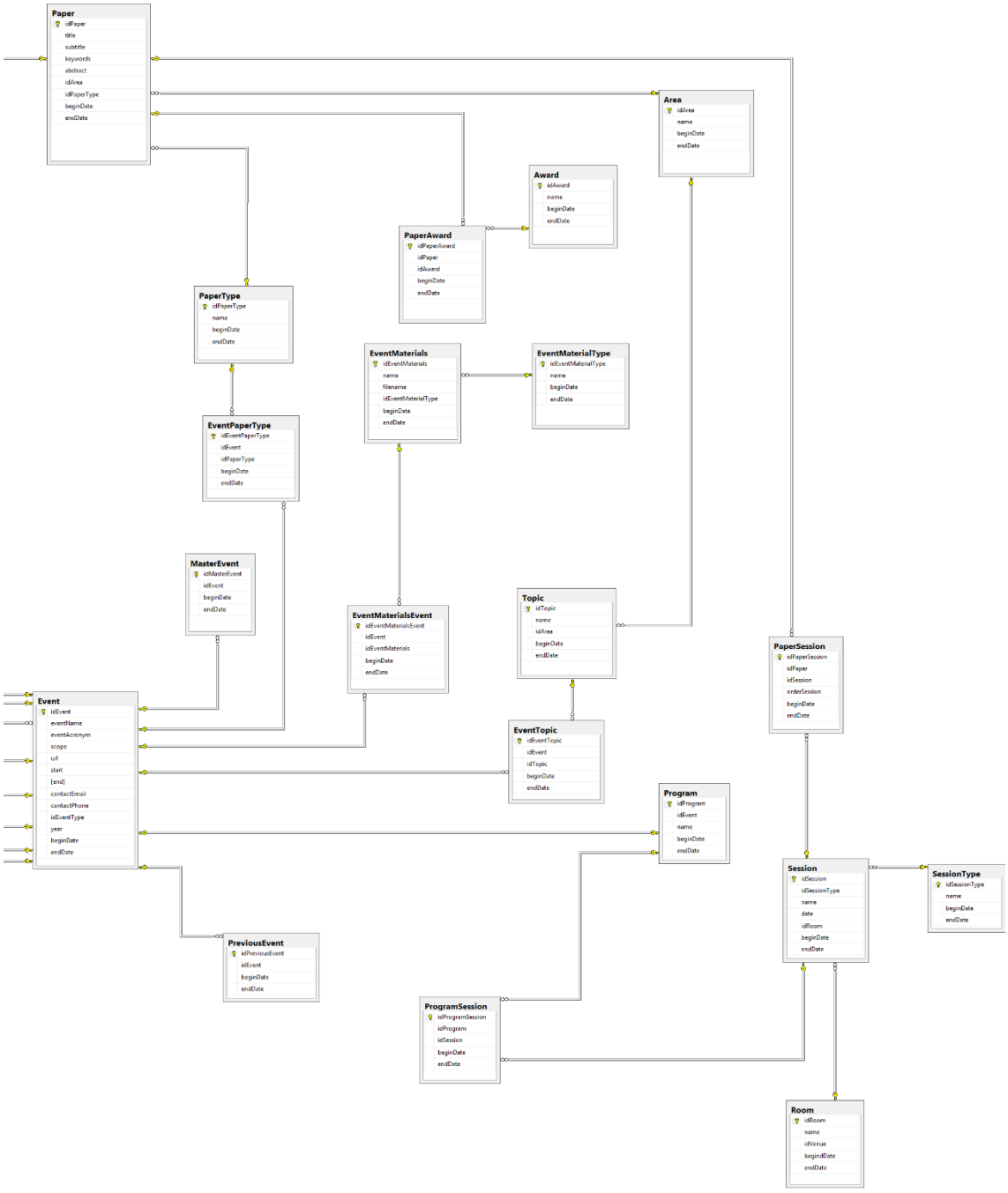
What is finite state machine? - Definition from WhatIs.com. (15 de Setembro de 2014). Obtido de WhatIs.com: http://whatis.techtarget.com/definition/finite-state-machine

Anexo I

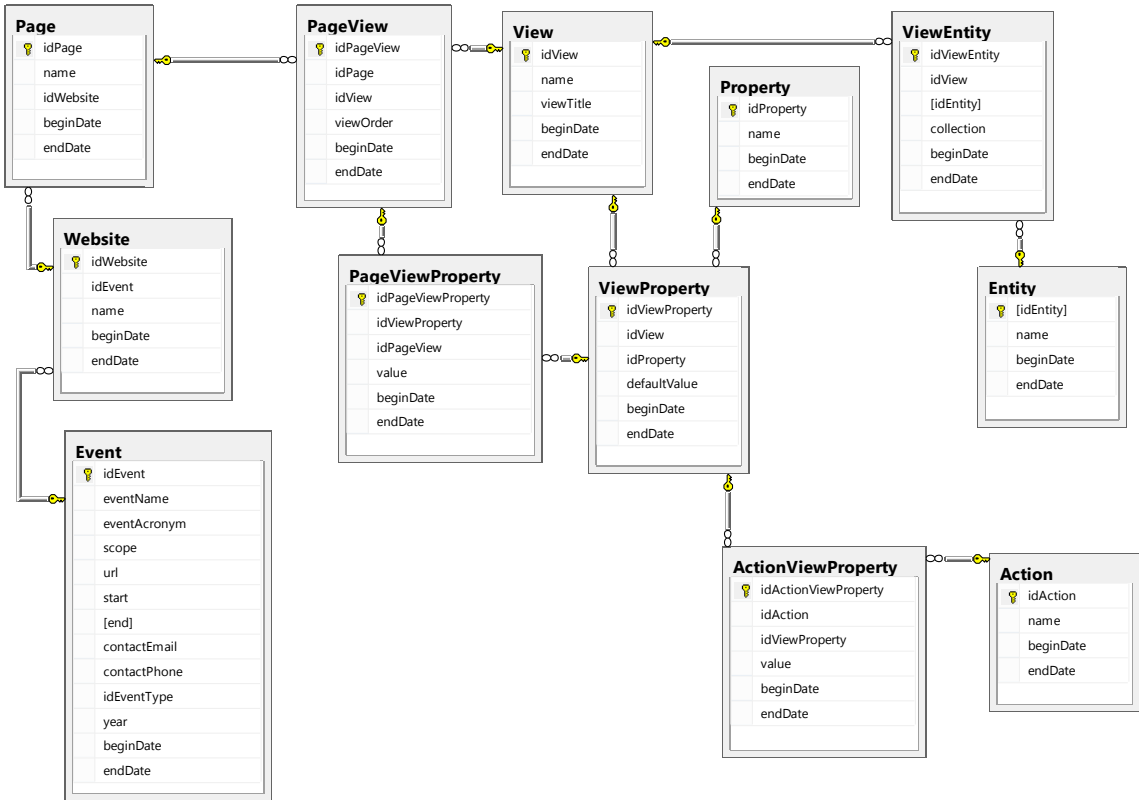
Tabelas de Base de Dados

A descrição das tabelas aqui anexada corresponde à informação inserida na base de dados, permitindo um maior entendimento da aplicação.


Country
idCountry
name
beginDate
endDate
Person
idPerson
firstname
lastname
email
affiliation
idCountry
beginDate
endDate
PaperPerson
idPaperPerson
idPaper
idPerson
beginDate
endDate
Paper
idPaper
title
subtitle
keywords
abstract
idArea
idPaperType
beginDate
endDate
PhotoLocalInformation
idPhotoLocalInformation
idLocalInformation
idPhoto
beginDate
endDate
Role
idRole
name
beginDate
endDate
PersonEventRole
idPersonEventRole
idPerson
idEvent
idRole
firstName
lastName
affiliation
idCountry
idPhoto
bio
beginDate
endDate
Photo
idPhoto
name
filename
beginDate
endDate
LocalInformation
idLocalInformation
name
description
beginDate
endDate
PhotoSocialEvent
idPhotoSocialEvent
idSocialEvent
idPhoto
beginDate
endDate
SocialEvent
idSocialEvent
name
description
date
beginDate
endDate
PhotoHotel
idPhotoHotel
idHotel
idPhoto
beginDate
endDate
PhotoVenue
idPhotoVenue
idVenue
idPhoto
beginDate
endDate
EventSocialEvent
idEventSocialEvent
idEvent
idSocialEvent
beginDate
endDate
Keynote
idKeynote
idPersonEventRole
title
abstract
beginDate
endDate
Venue
idVenue
name
description
idHotel
idLocalInformation
beginDate
endDate
Hotel
idHotel
name
description
beginDate
endDate
EventType
idEventType
name
beginDate
endDate
Event
idEvent
eventName
eventAcronym
scope
url
start
[end]
contactEmail
contactPhone
idEventType
year
beginDate
endDate
EventVenue
idEventVenue
idEvent
idVenue
beginDate
endDate
RegistrationFeeType
idRegistrationFeeType
name
beginDate
endDate
EventDate
idEventDate
idDate
idDateType
idEvent
date
extended
beginDate
endDate
EventPartner
idEventPartner
idEvent
idPartner
pendingApproval
additionalInfo
beginDate
endDate
Partner
idPartner
name
description
url
logo
idPartnerType
beginDate
endDate
EventItem
idEventItem
idEvent
idItemType
idRegistrationType
idRegistrationFeeType
value
beginDate
endDate
DateType
idDateType
name
beginDate
endDate
EventPartnerType
idEventPartnerType
name
idEvent
idPartnerType
beginDate
endDate
PartnerType
idPartnerType
name
beginDate
endDate
RegistrationType
idRegistrationType
name
beginDate
endDate


Paper
idPaper
title
subtitle
keywords
abstract
idArea
idPaperType
beginDate
endDate
Area
idArea
name
beginDate
endDate
Award
idAward
name
beginDate
endDate
PaperAward
idPaperAward
idPaper
idAward
beginDate
endDate
PaperType
idPaperType
name
beginDate
endDate
EventMaterials
idEventMaterials
name
filename
idEventMaterialType
beginDate
endDate
EventMaterialType
idEventMaterialType
name
beginDate
endDate
EventPaperType
idEventPaperType
idEvent
idPaperType
beginDate
endDate
MasterEvent
idMasterEvent
idEvent
beginDate
endDate
EventMaterialsEvent
idEventMaterialsEvent
idEvent
idEventMaterials
beginDate
endDate
Topic
idTopic
name
idArea
beginDate
endDate
PaperSession
idPaperSession
idPaper
idSession
orderSession
beginDate
endDate
Event
idEvent
eventName
eventAcronym
scope
url
start
[end]
contactEmail
contactPhone
idEventType
year
beginDate
endDate
EventTopic
idEventTopic
idEvent
idTopic
beginDate
endDate
Program
idProgram
idEvent
name
beginDate
endDate
Session
idSession
idSessionType
name
date
idRoom
beginDate
endDate
SessionType
idSessionType
name
beginDate
endDate
PreviousEvent
idPreviousEvent
idEvent
beginDate
endDate
ProgramSession
idProgramSession
idProgram
idSession
beginDate
endDate
Room
idRoom
name
idVenue
beginDate
endDate

A.2


Page
idPage
name
idWebsite
beginDate
endDate
PageView
idPageView
idPage
idView
viewOrder
beginDate
endDate
View
idView
name
viewTitle
beginDate
endDate
Property
idProperty
name
beginDate
endDate
ViewEntity
idViewEntity
idView
[idEntity]
collection
beginDate
endDate
Website
idWebsite
idEvent
name
beginDate
endDate
PageViewProperty
idPageViewProperty
idViewProperty
idPageView
value
beginDate
endDate
ViewProperty
idViewProperty
idView
idProperty
defaultValue
beginDate
endDate
Entity
[idEntity]
name
beginDate
endDate
Event
idEvent
eventName
eventAcronym
scope
url
start
[end]
contactEmail
contactPhone
idEventType
year
beginDate
endDate
ActionViewProperty
idActionViewProperty
idAction
idViewProperty
value
beginDate
endDate
Action
idAction
name
beginDate
endDate

Anexo II

Componentes de Software

Os componentes de *software* desenvolvidos nesta aplicação encontram-se aqui enunciados.

Nome página	URL	View
Home	Home	• Menus/EventsAssociated • EventDetails/DownloadPosterFlyer • Image/ImageControl • Dates/UpcomingDeadlines • EventDetails/MiniScope • AreasTopics/ConferenceAreasShort • Committees/ConferenceChairsShort • Committees/ProgramChairShort • Committees/AreaChairShortHome • Committees/LocalChairShort • Committees/KeynoteShortHome • SatelliteEvents/WorkshopsShortLink • SatelliteEvents/SpecialSessionsShortLink • SatelliteEvents/TutorialShort • SatelliteEvents/DemoShort • SatelliteEvents/PanelShort • SatelliteEvents/DCShortJoint • Statistics/NewStatisticsControl • Publications/PublicationControl • Organizations/SponsorCenter • Partners/PartnersControl
Log In	http://www.insticc.org/Primoris/	
Contacts	Contacts	EventDetails/ContactsHome
FAQs	FAQ	EventDetails/ControlFAQs
INSTICC Portal	http://www.insticc.org/	
On-line Registration	Link gerado para o evento	
Submit a Paper	Link gerado para o evento	
Authors Area	Link gerado para o evento	
Reviewers Area	Link gerado para o evento	
Important Dates	ImportantDates	Dates/EventImportantDates
Call for Papers	CallForPapers	• Menus/MenuCFP • EventDetails/Scope • AreasTopics/ConferenceAreasCFP • Committees/KeynoteShort • Publications/SubmissionGuidelinesCFP • Publications/CFPPublications • Dates/ImportantDates • EventDetails/SecretariatContacts • Venue/VenueSummary • Committees/ConferenceChairs • Committees/ProgramChair • Committees/AreaChair

		• Committees/LocalChair • Committees/ProgramCommittee
Program Committee	ProgramCommittee	• Committees/ProgramChair • Committees/ProgramCommittee
Technical Program	TechnicalProgram	TechnicalProgram/TechnicalProgramControl
Event Chairs	EventChairs	Committees/ChairsControl
Conference Venue	ConferenceVenue	Venue/VenuePage.ascx
Reaching the Venue	ReachingTheVenue	Travel/HowToGetThere
Keynote Speakers	KeynoteSpeakers	Committees/KeynotesPage
Previous Invited Speakers	PreviousInvitedSpeakers	Committees/PreviousInvitedSpeakersControl
Workshops	Workshops	SatelliteEvents/WorkshopsPage
Workshop Proposal	EventProposalWS	EventProposals/EventProposalControl
Special Sessions	SpecialSessions	SatelliteEvents/SpecialSessions
Special Session Proposal	EventProposalSS	EventProposals/EventProposalControl
Tutorials	Tutorials	• SatelliteEvents/TutorialTop • SatelliteEvents/TutorialPage
Tutorial Proposal	EventProposalTutorial	EventProposals/EventProposalControl
Demos	Demos	• SatelliteEvents/DemoTop • SatelliteEvents/DemoPage
Demo Proposal	EventProposalDemo	EventProposals/EventProposalControl
Panels	Panel	• SatelliteEvents/PanelTop • SatelliteEvents/PanelPage
Panel Proposal	EventProposalPanel	EventProposals/EventProposalControl
Doctoral Consortium	DoctoralConsortium	• SatelliteEvents/ScopeWorkshops • SatelliteEvents/DoctoralConsortiumChairDetail • SatelliteEvents/DCRules • Dates/DoctoralConsortiumImportantDates
Guidelines and Templates	GuidelinesTemplates	• Menus/MenuGuidelines • Publications/SubmissionGuidelines • Publications/PaperFormats • Publications/PaperTemplates
INSTICC Ethical Norms regarding Plagiarism and Self-Plagiarism	NormsPlagiarism	Publications/EthicalNorms
Presentation Details	PresentationDetails	Publications/PresentationGuidelines
Registration Fees	RegistrationFees	Registration/RegistrationFees

Important Information	ImportantInformation	Registration/ImportantInformation
Social Event	SocialEvent	Registration/SocialEvent
Reaching the City	ReachingCity	Travel/ReachCity
Visa Information	VisaInformation	Travel/VisaInformation
Hotel Reservation	HotelReservation	Hotel/HotelReservation
Hotel Information	HotelInformation	Hotel/HotelInformation
About the Region	AboutCity	Travel/LocalInformation
What to See and Do	SeeAndDo	Travel/WhatSeeDo
Useful Information	UsefulInfo	Travel/UsefullInfo
Best Paper Award	BestPaperAward	BestPapers/BestPaperAwards
Previous Awards	PreviousAwards	BestPapers/PreviousAwards
Books Published	BooksPublished	Publications/BooksPublished
Academic Partners	RDCommunity	• Partners/PartnerInfo • Partners/PartnerDetailsControl
Industrial Partners	IndustrialPartners	• Partners/PartnerInfo • Menus/MenuIndustrialPartner • Partners/PartnerDetailsControl
Tutorials and Instrumental Courses	TutorialsIC	• EventDetails/PartnershipControl • EventProposals/PartnershipProposalControl
Demo Sessions	Demosession	• EventDetails/PartnershipControl • EventProposals/PartnershipProposalControl
Exhibition	Exhibition	• EventDetails/PartnershipControl • EventProposals/PartnershipProposalControl
Exhibitor	Exhibitors	Partners/ExhibitorDetailsControl
Publicity Opportunities	Publicity	• EventDetails/PartnershipControl • EventProposals/PartnershipProposalControl
Delegates and Keynotes Sponsoring	Sponsoring	EventDetails/PartnershipControl
Institutional Partners	Institutions	• Partners/PartnerInfo.ascx • Partners/PartnerDetailsControl.ascx
Media Partners	MediaPartner	• Partners/PartnerInfo.ascx • Partners/PartnerDetailsControl.ascx
Partner Events	PartnerEvent	• Partners/PartnerInfo.ascx • Partners/PartnerDetailsControl
Publication Partners	PublicationPartner	• Partners/PartnerInfo • Partners/PartnerDetailsControl
Websites	Websites	PreviousConferences/Websites
Abstracts	Abstracts	Abstracts/AbstractsControl
Thanks Page	ThanksPage	• NextConferences/NextEdition • EventDetails/AllConferences • EventDetails/ScitePressLink

Anexo III

Páginas Web e sua Estrutura

As páginas web e a sua respetiva estrutura encontram-se aqui listadas.

- Homepage
  - Eventos relacionados
    - Texto informativo
  - Próximas datas
    - Tipo
    - data
  - Scope
    - Texto informativo
  - Áreas
    - Lista áreas
  - Fotos
  - Conference chair
    - Nome
    - Afiliação
    - País
  - Program chair
    - Nome
    - Afiliação
    - País
  - Keynote speakers
    - Nome
    - Afiliação
    - País
  - Satellite events
    - Tipo
    - Nome
    - Acrónimo
    - Nome chairs
    - Próxima data associada
  - Material promocional
    - Flyer
    - Poster
  - Sponsors
    - Tipo
    - Logo
    - Url
- Contacts
  - Título
  - Informação secretariado
    - Morada
    - Telefone
    - Fax
    - Email
- FAQs
  - Título

    - Secção A
        - Perguntas
    - Secção B
        - Respostas
- Important Dates
    - Título
    - Tipo
    - Secção datas
        - Tipo
        - Datas chave
- Program Committee
    - Título
    - Chairs
        - Título
        - Nome
        - Afiliação
        - País
    - Revisores
        - Título
        - Nome
        - Afiliação
        - País
- Event Chairs
    - Título
    - Tipo
    - Informação
        - Foto
        - Nome
        - Afiliação
        - País
        - Bibliografia
- Keynotes
    - Título
    - Índice
        - Nome
        - Afiliação
        - País
        - Título da keynotes
    - Lista keynotes
        - Título da keynotes

            - Foto
            - Nome
            - Afiliação
            - País
            - Abstract
            - Vídeo
- Previous Invited Speakers
    - Título
    - Texto agradecimento
    - Index anos
    - Lista indexada por ano
        - Nome
        - Afiliação
        - País
        - Título da keynotes
    - Ano
    - Listagem keynotes
        - Título keynotes
        - Abstract
        - Foto
        - Nome
        - Afiliação
        - País
        - Vídeo
- Previous Conferences
    - Título
    - Mini banner
    - Link url
- Previous Abstracts
    - Título
    - Ano
    - Satellite events por ano
- Previous Awards
    - Título
    - Texto explicativo
    - Ano
    - Tipo de award
    - Área
    - Informação award
        - Título

            - Nome autores
- Partners (Academic, Publication, Event e Institutional)
    - Título
    - Texto descritivo
    - Lista partners
        - Título
        - Logo
        - Descrição
        - URL
- Partners (Media)
    - Título
    - Texto descritivo
    - Formulário proposta
    - Lista partners
        - Título
        - Logo
        - Descrição
        - URL
- Partners (Industrial)
    - Título
    - Texto descritivo
    - Links opções
    - Lista partners
        - Título
        - Logo
        - Descrição
        - URL
- Best Paper Awards
    - Título
    - Parágrafo explicativo dos tipos de award
    - Texto indicativo dos prémios recebidos
        - listagem
- Importante information
    - Título
    - Texto informativo associado ao registo
        - Política cancelamento
        - Informação descontos
        - Benefícios estudantes
            - Listagem de possibilidades de desconto
- Registration fees

    - Título
    - Texto explicativo
    - Datas limite registo early
        - Tipo de data
        - Data limite
    - Datas limite late
        - Tipo data
        - Data limite
    - Preços base
        - Valor
        - Tipo registo
    - Texto explicativo
    - Tabela principal preços
    - Conteúdo registo
    - Tabela preços DC
    - Tabela preços itens adicionais
- Visa information
    - Título
    - Texto informativo
- Useful information
    - Título
    - Foto
    - Texto informativo
    - Hora local
    - Meteorologia local
- Conference Venue
    - Título
    - Texto informativo
    - Fotos
- Reaching the Venue
    - Título
    - Texto informaivo
    - Mapa e legenda
- Reaching City
    - Título
    - Texto informativo
    - Mapa e legenda
- Technical Program
    - Título
    - Overview programa (imagem)

- Workshops
    - Título
    - Texto explicativo
    - Data limite proposta
    - Link formulário proposta
    - Listagem workshops
        - Acrónimo
        - Ano
        - Nome evento
        - Nome chairs
    - Informação detalhada
        - Datas principais
        - Scope
        - Link para página
        - Botão submissão artigo
        - Chairs
            - Nome
            - Afiliação
            - País
            - Foto
- Special Sessions
    - Título
    - Texto explicativo
    - Data limite proposta
    - Link formulário proposta
    - Listagem special sessions
        - Acrónimo
        - Ano
        - Nome evento
        - Nome chairs
    - Informação detalhada
        - Datas principais
        - Scope
        - Link para página
        - Botão submissão artigo
        - Chairs
            - Nome
            - Afiliação
            - País
            - Foto

- Tutorial
    - Título
    - Texto explicativo
    - Data limite proposta
    - Link formulário proposta
    - Listagem tutoriais
        - Nome evento
        - Nome lecturers
    - Informação detalhada
        - Título
        - Informação lecturer
            - Nome
            - Afiliação
            - País
            - Foto
            - Biografia
        - Informação relativa ao tutorial
        - Contacto secretariado
- Demo
    - Título
    - Texto explicativo
    - Data limite proposta
    - Link formulário proposta
    - Listagem demos
        - Nome evento
        - Nome chairs
    - Informação detalhada
        - Título
        - Informação chair
            - Nome
            - Afiliação
            - País
            - Foto
            - Biografia
        - Informação relativa à demo
        - Contacto secretariado
- Panel
    - Título
    - Texto explicativo
    - Data limite proposta

    - Link formulário proposta
    - Listagem painel
        - Nome evento
        - Nome chairs
    - Informação detalhada
        - Título
        - Informação chair
            - Nome
            - Afiliação
            - País
            - Foto
            - Biografia
        - Informação relativa ao painel
        - Contacto secretariado
- Doctoral Consortium
    - Título
    - Texto informativo
    - Advisory board
        - Nome
        - Afiliação
        - País
        - Indicador chair
    - Regras participação
    - Instruções submissão trabalho
        - Texto explicativo
        - Botão submissão
    - Informação preparação apresentação
    - Informação awards
    - Datas chave
- Open Communications
    - Título
    - Texto informativo
    - Data limite submissão
    - Botão submissão
- European Project Space
    - Título
    - Texto informativo
    - Formas de participação
    - Data limite proposta
    - Link formulário proposta

    - Chairs
        - Nome
        - Afiliação
        - País
    - Painelistas
        - Nome
        - Afiliação
        - País
    - Link programa detalhado
    - Vídeo
    - Lista participantes
        - Logo
        - Url
- EPS Program
    - Título
    - Informação programa
        - Hora início
        - Hora fim
        - Tipo sessão
        - Nome
        - Sala
- Industrial Track
    - Título
    - Texto informativo
    - Data limite propostas
    - Link formulário proposta
    - Chair
        - Nome
        - Afiliação
        - País
    - Titulo painel
    - Participantes painel
        - Nome
        - Afiliação
        - País
    - Participantes
        - Logo
        - Url
- Call for Papers
    - Título

- Menu
- Scope
  - Texto informativo
- Áreas
  - Nome
  - Lista tópicos associados
- Keynote speakers
  - Nome
  - Afiliação
  - país
- Submissão
  - Texto informativo
- Publicação
  - Texto informativo
- Datas
  - Tipo de evento
  - Secção datas
    - Tipo
    - Datas chave
- Informação secretariado
  - Morada
  - Telefone
  - Fax
  - Email
  - Web url
- Venue
  - Texto informativo
- Conference chair
  - Nome
  - Afiliação
  - País
- Program chair
  - Nome
  - Afiliação
  - País
- Program committee
  - Nome
  - Afiliação
  - País

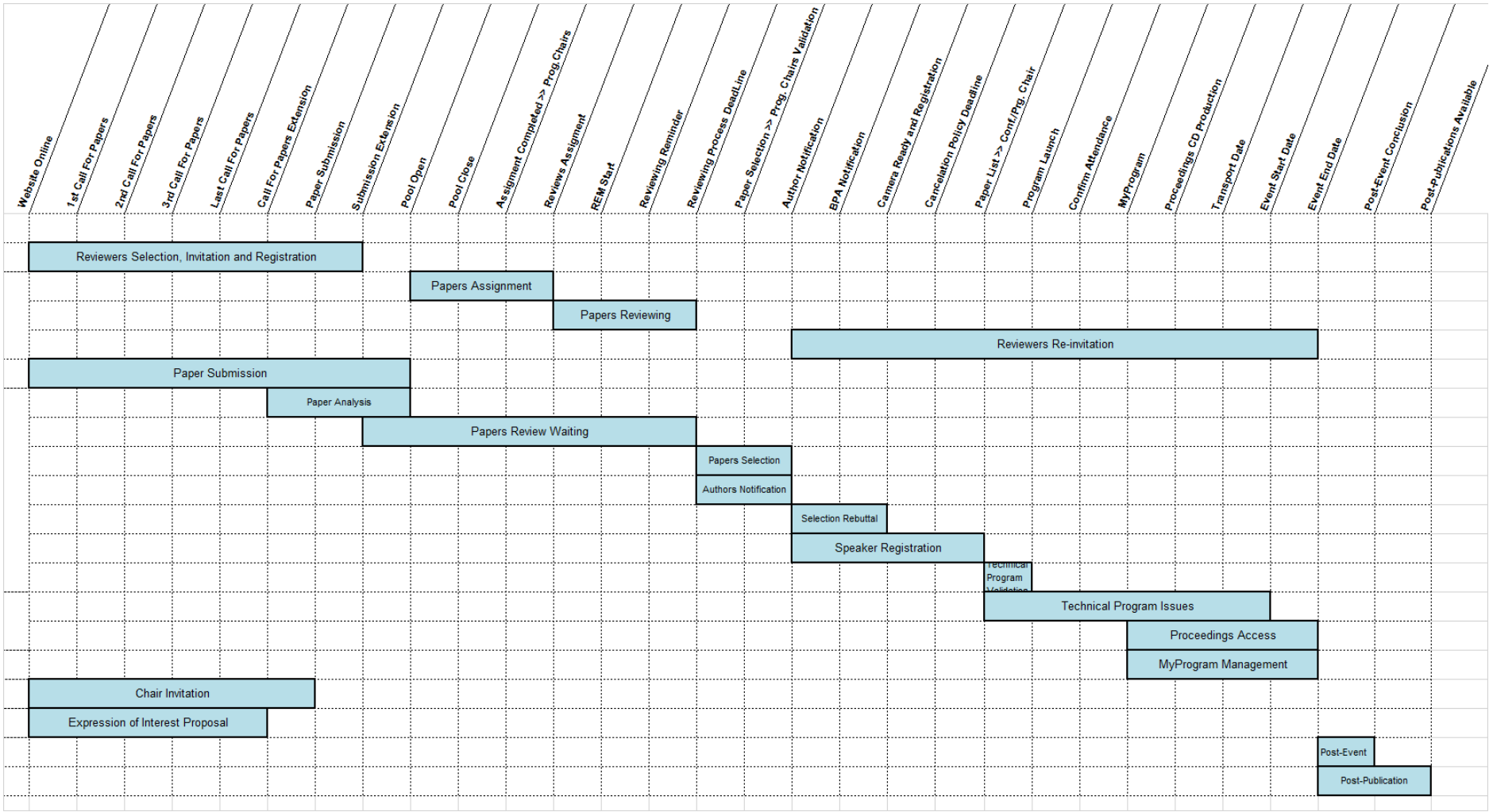
- Guidelines and Templates

- Título
- Menu
- Secção
    - Título
    - Texto informativo

Anexo IV

Fases do Ciclo de Vida do Evento

Este anexo contempla as fases do ciclo de vida do evento bem como as suas tarefas principais.


Website Online
1st Call For Papers
2nd Call For Papers
3rd Call For Papers
Last Call For Papers
Call For Papers Extension
Paper Submission
Submission Extension
Pool Open
Pool Close
Assignment Completed >> Prog.Chairs
Reviews Assignment
REM Start
Reviewing Reminder
Reviewing Process DeadLine
Paper Selection >> Prog. Chairs Validation
Author Notification
BPA Notification
Camera Ready and Registration
Cancelation Policy Deadline
Paper List >> Conf./Prg. Chair
Program Launch
Confirm Attendance
MyProgram
Proceedings CD Production
Transport Date
Event Start Date
Event End Date
Post-Event Conclusion
Post-Publications Available
Reviewers Selection, Invitation and Registration
Papers Assignment
Papers Reviewing
Reviewers Re-invitation
Paper Submission
Paper Analysis
Papers Review Waiting
Papers Selection
Authors Notification
Selection Rebuttal
Speaker Registration
Technical Program Issues
Proceedings Access
MyProgram Management
Chair Invitation
Expression of Interest Proposal
Post-Event
Post-Publication