Universidade do Minho
Escola de Engenharia

Hugo Filipe Rosas da Costa

Gestão dos Fluxos de Transporte na Urgência Hospitalar

Outubro de 2011

Gestão dos Fluxos de Transporte na Urgência Hospitalar
Hugo Filipe Rosas da Costa
UMinho | 2011

Universidade do Minho
Escola de Engenharia

Hugo Filipe Rosas da Costa

Gestão dos Fluxos de Transporte na Urgência Hospitalar

Dissertação de Mestrado
Mestrado Integrado em Engenharia Biomédica

Trabalho efetuado sob a orientação do
Professor Doutor António Carlos Abelha

Outubro de 2011

Agradecimentos

Ao longo de todo o percurso académico foram muitas as pessoas e as experiências que me marcaram como pessoa e que me prepararam para o mundo exigente do mercado de trabalho.

Em primeiro lugar, esta dissertação não seria possível sem a disponibilidade do Dr. António Abelha para orientar este ano de trabalho. Desde do momento em que o projeto foi proposto até ao momento em que o trabalho final foi entregue, tenho a agradecer a transmissão de conhecimentos fundamentais para a minha evolução ao longo deste ano. Sem os conselhos construtivos e inovadores que foram transmitidos não teriam sido obtidos resultados tão evoluídos e consistentes.

Aos meu colegas deste percurso universitário tenho a agradecer todas as experiências vividas, deixando uma palavra especial para aqueles que no ramo de informática médica promoveram a saudável partilha de informação, que tanto contribuiu para a evolução de todos nós.

À minha família agradeço a criação de todas as condições necessárias para que fosse possível uma evolução académica estável. Mesmo nos momentos mais difíceis sei que sempre fizeram os possíveis para me apoiar ao máximo.

Por último, um agradecimento especial para uma pessoa muito especial que tanto me motiva e inspira – Obrigado Isa. Sei que do teu lado todas as dificuldades vão ser sempre ultrapassadas.

Resumo

À medida que os sistemas de informação se vão tornando mais inovadores e consistentes, as vantagens da sua aplicação na área da saúde começam a ser consensuais. Administradores hospitalares, profissionais de saúde e pacientes começam a aperceber-se que a implementação de sistemas de informação mais abrangentes e inovadores pode conduzir a uma maior qualidade na prestação de cuidados de saúde.

No contexto de um serviço de urgência hospitalar, o tempo de espera é um fator essencial para determinar a satisfação dos pacientes. Os sistemas de informação atuais conduzem ao registo estruturado da informação clínica em formato eletrónico, abrindo novas perspetivas para a otimização de tarefas que utilizam essa informação. A otimização de algumas dessas tarefas pode levar à melhoria global do desempenho do serviço e a uma consequente redução do tempo de espera.

O objetivo desta dissertação passa por desenvolver uma aplicação Web que permita gerir o fluxo de transportes internos de um serviço de urgência, tirando partido das potencialidades dos novos sistemas de informação.

O desenvolvimento da aplicação foi realizado com recurso a recentes tecnologias Web, de forma a criar interfaces que promovam uma interação agradável com o utilizador e que garantam uma navegação fluente pelas várias opções disponíveis. A aplicação possui a capacidade de se adaptar automaticamente a qualquer resolução de ecrã e está otimizada para ser utilizada em dispositivos de visualização tácteis.

A aplicação desenvolvida efetua a sugestão dos transportes que devem ser escolhidos, de acordo com 3 critérios distintos: prioridade do paciente; localização do transporte; compatibilidades para iniciar transportes em grupo.

Em suma, é possível afirmar que a aplicação desenvolvida está preparada para responder aos desafios existentes no seu contexto de utilização, sendo de prever que a eficiência do transporte de pacientes possa aumentar com a implementação desta solução.

Abstract

As information systems are becoming more innovative and consistent, the benefits of their application in health care begin to be consensual. It means that hospital administrators, health professionals and patients begin to realize that the implementation of more comprehensive and innovative information systems may lead to higher quality in health care.

In emergency department context, waiting time is a key factor in determining patient satisfaction. Current information systems lead to electronic structured register of clinical information, opening new perspectives to optimize particular tasks that use this information. The optimization of some tasks may lead to overall improvement of service performance and consequent reduction in waiting time.

The purpose of this dissertation is to develop a web application that allow to manage the internal transport flow of an emergency department, leveraging the potential of new information systems.

Application development was carried out using the latest Web technologies to create interfaces that promote a pleasant interaction with the user and to ensure a fluent navigation through the several options available. The application has the ability to automatically adapt to any screen resolution and is optimized for use in tactile display devices.

The developed application makes suggestions to indicate which transport should be chosen according to three different criteria: patient priority, transport location; compatibilities to initiate a transport group.

In sum, is possible to state that the developed application is prepared to meet the challenges existing in its context of use. Thus, the implemented solution should increase the efficiency of regular patient transport tasks.

Conteúdo

Lista de Figuras

Lista de Códigos

Notação e Terminologia

Notação Geral

A notação ao longo do documento segue a seguinte convenção:

- Texto em itálico – para palavras em língua estrangeira (e.g., Inglês). Também utilizado para dar ênfase a um determinado termo ou expressão.
- Texto em negrito – utilizado para realçar um conceito ou palavra no meio de um parágrafo.

Acrónimos

ACSS – Administração Central do Sistema de Saúde

ADO.NET – ActiveX Data Object for .NET

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

ASP – Active Server Pages

CIL – Common Intermediate Language

CLR – Common Language runtime

CLS – Common Language Specification

CSS – Cascading Style Sheet

DOM – Document Object Model

HIS – Health Information System

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IGIF – Instituto de Gestão Informática e Financeira da Saúde

IIS – Internet Information Services

IP – Internet Protocol

MVC – Model View Controler

OCI – Oracle Call Interface

ODBC – Open Database Connectivity

ODP.NET – Oracle Data Provider for .NET

OLE DB – Object Linking and Embedding Database

RCE – Registo Clínico Eletrónico

RIA – Rich Internet Applications

SDLC – Software Development Life Cycle

SGBD – Sistemas de Gestão de Bases de Dados

SI – Sistemas de Informação

SONHO – Sistema Integrado de Informação Hospitalar

SQL – Structured Query Language

SU – Serviço de Urgência

TCP – Transmission Control Protocol

UCH – Unidades e Centros Hospitalares

VB – Visual Basic

W3C – World Wide Web Consortium

XML – Extensible Markup Language

* Esta dissertação foi redigida ao abrigo do novo acordo ortográfico.

CAPÍTULO 1

Introdução

Esta dissertação aborda o trabalho desenvolvido ao longo do quinto ano do Mestrado Integrado em Engenharia Biomédica no ramo de Informática Médica, tendo como objetivo desenvolver uma solução para a gestão do fluxo de transportes internos de pacientes dentro do serviço de urgência de uma unidade hospitalar.

1.1 Enquadramento

As últimas décadas têm sido ricas em avanços tecnológicos que têm moldado a vida das pessoas, tanto a nível pessoal como profissional. A constante evolução tecnológica nos sistemas informáticos, quer na parte de hardware como de software, tem vindo a revolucionar a forma como o ser humano realiza muitas das suas atividades. Desde a sistematização de tarefas repetitivas, passando pela forma como a informação é organizada e acedida, até à remodelada interação entre as próprias pessoas, os avanços dos sistemas informáticos têm sido fulcrais para melhorar a qualidade de vida das pessoas. Esta relação próxima entre a qualidade de vida e o nível de avanço tecnológico faz-se sentir tanto a nível pessoal como profissional. (Shortliffe e Cimino, 2006)

Atualmente, ao abordar o desenvolvimento de sistemas informáticos é indispensável

referir o notável desenvolvimento das redes de computadores e da própria Internet. As infraestruturas e hardware existentes conferem novas e melhores funcionalidades aos utilizadores dos sistemas informáticos, uma vez que a informação circula rapidamente entre os diversos terminais da rede. Isto resulta na possibilidade de aceder remotamente à informação quase à mesma velocidade a que é efetuado o acesso local.

No que diz respeito à utilização da Internet, para além de se verificar um aumento na velocidade de acesso aos recursos, também os servidores apresentam melhores características, desde a maior capacidade de processamento e armazenamento das suas máquinas, até à prevenção de falhas que permitem níveis de disponibilidade bastante elevados que se aproximam dos 100% de tempo de funcionamento normal. Estes aspetos têm levado a uma evolução na forma como informação digital é tratada (Shortliffe e Cimino, 2006; Colouris et al., 2011).

Com as potencialidades das redes de computadores e com o desenvolvimento de Sistemas de Informação (SI) cada vez mais sofisticados, verifica-se a tendência para alterar a forma de armazenamento e acesso à informação, passando do antigo paradigma de informação disponível localmente na máquina, onde o utilizador necessita de a usar, para um paradigma de informação disponível remotamente no terminal do utilizador, estando armazenada num servidor disponível na rede (Colouris et al., 2011).

No contexto hospitalar, e em geral no setor da saúde, quando a informática surgiu como solução para tratar a informação clínica subjacente à prestação de cuidados de saúde, cada serviço de uma instituição foi adquirindo os seus próprios meios informáticos. No entanto, a troca de informação entre os sistemas de informação dos diferentes serviços não é tida em conta e muitas vezes simplesmente não funciona. Este obstáculo surge porque, por exemplo ao nível do software, os Sistemas de Gestão de Bases de Dados (SGBD) onde é registada a informação clínica dos pacientes estão estruturados para um serviço específico, pelo que a sua integração com outros sistemas é dificultada (Walker et al., 2005; Chaudhry et al., 2006).

Nos anos mais recentes, as mudanças tecnológicas e conceptuais estão a permitir uma mudança de paradigma no que diz respeito ao registo, manipulação e consulta de informação. Para tal tem sido importante a consolidação e divulgação de dois conceitos

fundamentais:

- **HIS** (*Health Information System*), que se refere a um sistema de gestão de informação de uma unidade prestadora de cuidados de saúde em que a informação é organizada em função do paciente e não do serviço ou instituição. Este conceito de sistema de informação tem como objetivo atingir crescente escalabilidade e interoperabilidade nos sistemas de informação (Kuhn e Giuse, 2001).

- **RCE** (Registo Clínico Eletrónico), que é um repositório de informação relativa ao estado de saúde de um indivíduo objeto de cuidados. A informação é registada num formato suscetível de processamento por computador, armazenada e transmitida de forma segura, e acessível por múltiplos utilizadores autorizados. A sua finalidade primária é suportar cuidados de saúde integrados, contínuos, eficientes e com qualidade. Contém informação retrospetiva, corrente e prospetiva (Hayrinen et al., 2008).

Com a adoção destes conceitos, a informação passa a estar organizada a um nível generalizado, com perspetivas de escalabilidade e interoperabilidade, permitindo que a informação devidamente estruturada esteja acessível remotamente no local onde é necessária, através de interfaces rápidas e eficientes que disponibilizam a informação de acordo com as necessidades do utilizador.

Os avanços tecnológicos e conceptuais, até aqui referidos, levaram à evolução dos Sistemas de Informação (SI) e introduziram alterações substanciais na forma como a informação clínica é obtida, tratada, armazenada e utilizada. Estas mudanças permitem otimizar o fluxo de trabalho dos profissionais de saúde, melhorando a organização e execução de diversas tarefas que são realizadas continuamente no quotidiano destes trabalhadores (Shortliffe e Cimino, 2006).

A qualidade da prestação de cuidados de saúde é sempre um tópico de interesse público a que poucos cidadãos ficam indiferentes. Qualquer cidadão, quando necessita de cuidados de saúde, recorre à instituição de saúde que oferece serviços de maior qualidade. Assim, todas as instituições trabalham para oferecer aos seus pacientes os melhores cuidados de saúde possíveis, garantindo, não só que o caso clínico do paciente é resolvido, mas também que este é atendido de uma forma simples, rápida e eficiente.

Com o intuito de atingir um atendimento de maior qualidade, uma instituição de saúde tenta melhorar vários aspetos identificados como fulcrais para determinar a qualidade do seu desempenho. Um fator que assume um papel preponderante é o tempo de espera que o paciente enfrenta numa determinada etapa da sua passagem pela instituição de saúde (Sitzia e Wood, 1997; Kravitz, 1998). Qualquer que seja a etapa na qual existe uma espera excessiva, esta irá induzir uma noção de menor qualidade para os cuidados de saúde prestados, levando a uma repercussão negativa na satisfação do paciente (Eilers, 2002).

No caso específico do Serviço de Urgência (SU), a passagem de um paciente por este serviço pode estar sujeito a vários tempos de espera: tempo de espera para ser atendido na receção; para entrar na sala de triagem, para ser chamado pelo médico; ou para realizar um exame complementar de auxílio ao diagnóstico (Taylor e Benger, 2004; Hoot e Aronsky, 2008). Qualquer que seja o serviço ou unidade hospitalar no qual o paciente esteja a ser atendido, se este possuir dificuldades de locomoção e necessitar de auxílio para se deslocar no interior das instalações, o seu atendimento será prejudicado com mais um tempo de espera: O tempo de espera pelo transporte. Este tempo de espera por transporte tem uma repercussão ainda mais notória na satisfação do paciente quando surge no Serviço de Urgência, tal como abordado nesta dissertação.

1.2 Motivação e Objetivos

O enquadramento realizado na secção anterior ajuda a perceber as motivações que levam ao desenvolvimento do projeto associado a esta dissertação. Essencialmente, pretende-se melhorar o desempenho dos profissionais de saúde na execução de uma tarefa específica que se enquadra no contexto de um serviço de urgência hospitalar, não se pretendendo apenas melhorar o fluxo de trabalho dos profissionais de saúde mas também influenciar direta ou indiretamente a qualidade do atendimento prestado aos pacientes.

Trabalhar com a problemática da organização de tarefas em conjunção com a otimização dos tempos de espera perfilou-se como um desafio interessante, sendo que a um

nível mais específico, a gestão e organização do transporte de pacientes no interior no Serviço de Urgência de um hospital surge como um desafio ainda mais interessante pelos seguintes motivos:

- Visa melhorar o fluxo de trabalho dos profissionais encarregues de prestar auxílio aos pacientes com dificuldades de locomoção ou qualquer outra necessidade de acompanhamento.
- Ao melhorar a organização dos transportes e do fluxo de trabalho dos profissionais, o tempo de espera por transporte será menor para o mesmo número de profissionais a trabalhar, ou seja, com os mesmos custos é possível obter maior produtividade e melhor qualidade no atendimento.
- Por vezes, os atrasos nos transportes fazem com que o médico fique inativo enquanto espera pela chegada do paciente que chamou. Este atraso afeta diretamente o paciente que está a ser chamado e indiretamente os pacientes que, mesmo sem necessitar de transporte, vêm o seu atendimento adiado devido a atrasos nos transportes de pacientes prioritários.
- Por último, existe a motivação extra de promover a igualdade entre pacientes com e sem restrições de locomoção. Ao eliminar atrasos no transporte de pacientes com dificuldades de locomoção, estes não serão prejudicados com atrasos adicionais em relação pacientes sem restrições de locomoção.

Em conjunção com o apresentado anteriormente, outra motivação desta dissertação é explorar as potencialidades dos novos sistemas de informação disponíveis atualmente nos hospitais, uma vez que estes permitem que a informação esteja em constante atualização e disponível remotamente em qualquer terminal de acesso dentro ou fora da instituição. Desta forma, o objetivo central desta dissertação consiste em:

- Desenvolver uma aplicação Web que permita gerir o fluxo de transportes internos de um serviço de urgência hospitalar, tirando partido das potencialidades dos novos sistemas de informação implementados nas unidades hospitalares.

A aplicação a desenvolver deve satisfazer os seguintes requisitos técnicos:

- Independência da arquitetura da máquina onde a aplicação é executada.

- Compatibilidade com o SGBD existente (Oracle), de modo que a informação esteja em conformidade com as várias aplicações que podem alterar o seu estado.
- Disponibilização de uma interface prática e intuitiva ao utilizador, tanto a nível de menus como através do recurso a interfaces tácteis. Pretende-se minimizar o tempo despendido com a utilização da aplicação, de forma a que o tempo gasto na sua utilização seja inferior ao tempo ganho com a melhoria na organização dos transportes.
- Flexibilidade quanto ao tipo de dispositivo de visualização utilizado.

1.3 Estrutura

A presente dissertação tem como objetivo desenvolver uma solução informática para um problema específico. Deste modo, a dissertação assume um caráter essencialmente técnico, mais direcionado para as questões de desenvolvimento de software e não tanto para a pesquisa intensiva de bases científicas que sustentem uma teoria. Este caráter técnico influencia diretamente a estrutura da dissertação, deixando um espaço mais reduzido para a revisão bibliográfica.

No **Capítulo 2** são abordadas as linhas de orientação oficiais para o desenvolvimento de aplicações para visem informatizar o serviço de urgência. Estas orientações aplicam-se à informatização geral no serviço de urgência pelo que são apenas discutidas aquelas que podem ser aplicadas ao módulo desenvolvido nesta dissertação. É realizada um revisão bibliográfica que visa expor a problemática dos tempos de espera nos serviços de urgência, focando de seguida as soluções que vêm sendo propostas para amenizar este problema. O Sistema de Triagem de Manchester é também explicado, uma vez que este influencia as restantes tarefas do serviço de urgência.

O **Capítulo 3** apresenta o conjunto de tecnologias que foram investigadas para tornar possível o desenvolvimento do projeto desta dissertação. Para cada tecnologia analisada é realizada uma comparação com as tecnologias concorrentes que poderiam ser adotadas. Esta comparação utilizada o ponto de vista das necessidades da aplicação para determinar a solução que melhor se adequa aos objetivos propostos. Algumas das

tecnologias têm uma explicação mais extensa de forma a que o leitor compreenda os princípios que sustentam a aplicação desenvolvida.

O **Capítulo 4** explica o processo de desenvolvimento de software adotado. Nesse sentido, é realizada uma análise pormenorizada aos requisitos da aplicação assim como a todo o processo de planeamento dos componentes que irão dar origem à aplicação propriamente dita. Os resultados obtidos com a implementação das ideias concebidas são cuidadosamente descritos de seguida. Por último, são ainda apresentados os resultados dos testes efetuados e o método de publicação da aplicação.

No **Capítulo 5** são retiradas ilações acerca do trabalho realizada, havendo ainda espaço para a sugestão de trabalho a desenvolver como continuação do projeto.

CAPÍTULO 2

Trabalho Relacionado

2.1 Informatização do serviço de urgência

Em Portugal, a base informática que todos os hospitais têm em comum é a implementação do Sistema Integrado de Informação Hospitalar (SONHO). O SONHO assume um papel fundamental no registo de informação de caráter puramente administrativo, prestando um serviço importante para a gestão administrativa de utentes de um hospital. No entanto, este sistema não permite o registo de informação clínica adquirida na prestação de cuidados de saúde nos diversos serviços hospitalares (Direção-Geral da Saúde, 2004).

Os hospitais portugueses, motivados pela necessidade de melhorar a organização e desempenho dos seus serviços, iniciaram o processo de integração do SONHO com novos módulos informáticos que sejam capazes de gerir a informação clínica especifica de cada serviço.

O serviço de urgência, apresentando-se como um dos serviços fulcrais na estrutura de um hospital, deve-se dotar de aplicações informáticas que prestem auxílio às diversas atividades desempenhadas pelos diferentes profissionais de saúde. Estas ferramentas informáticas devem estar disponíveis no posto de trabalho de cada profissional, fazendo

com que o desempenho global de todo o serviço de urgência seja melhorado, tanto pelo registo facilitado de informação, como pela utilização mais prática e dinâmica dessa informação registada (Ministério da Saúde, 2006).

O Instituto de Gestão Informática e Financeira da Saúde (IGIF) foi, até final de 2006, responsável pela desenvolvimento e gestão dos recursos informáticos disponíveis nas instituições de saúde. No final desse ano, as funções do IGIF foram extintas, passando a gestão informática a ser parte das competências da Administração Central do Sistema de Saúde (ACSS). No entanto, antes da sua junção à ACSS, o IGIF desenvolveu um documento onde apresenta diversas orientações para a informatização de um serviço de urgência (IGIF, 2005).

No contexto desta dissertação, destaca-se que a solução geral para informatizar um serviço de urgência de um hospital deve possuir interfaces e funcionalidades específicas para cada especialidade e para cada tipo de profissional, permitindo a informatização de todos os registos efetuados no serviço de urgência, sempre em articulação e integração com o SONHO e outras aplicações informáticas existentes no hospital. Existem alguns requisitos que esta solução deve respeitar como um todo e também ao nível específico de cada módulo (IGIF, 2005):

- Desenvolvimento de aplicações com interfaces que facilitem a interação com o utilizador, nomeadamente, através da otimização para utilização de ecrãs tácteis.
- Garantir a identificação dos profissionais que, em cada momento, utilizam os terminais das aplicações. Os métodos utilizados devem ser eficientes e facilitar o rápido acesso à aplicação.
- No caso de ser realizada triagem de prioridades, devem ser realizados registos para acompanhar os tempos de espera a que cada utente está sujeito em cada fase do seu tratamento num serviço de urgência.
- Prever diversas situações de alerta relativamente ao processo de tratamento de um paciente, como por exemplo tempo de espera excessivo numa determinada etapa.
- Permitir a consulta de circuito do paciente no serviço de urgência, bem como

o seu estado atual. Assim, deve ser realizado um rastreamento completo das atividades do paciente desde a admissão até à alta, incluindo horas e locais de atendimento, especialidades e profissionais envolvidos, atos prescritos e realizados, entre outros. Esta informação permite aos profissionais de saúde consultar rapidamente a localização, estado clínico e tempo de espera de um paciente.

- Gerar diversas estatísticas relativas a aspetos clínicos e de gestão. Estas estatísticas podem ser importantes para avaliar o desempenho do serviço de urgência em geral e das suas especialidades em particular.

2.2 Problemática associada aos tempos de espera

Os tempos de espera a que os pacientes são sujeitos num serviço de urgência é um tema que, desde há bastante tempo, é alvo de estudo e discussão. Vários estudos indicam a existência de três fatores que influenciam a satisfação dos pacientes quando estes são chamados a avaliar a prestação de cuidados de saúde de que foram alvo: personalização dos cuidados prestados; informação relacionada com o atendimento; tempo de espera (Sitzia e Wood, 1997; Kravitz, 1998; Hoot e Aronsky, 2008).

O primeiro critério, aponta que a satisfação dos pacientes pode ser aumentada através da prestação de cuidados de saúde mais personalizados, que permitam uma comunicação mais próxima entre profissionais de saúde e pacientes (Cleary e McNeil, 2008).

O segundo critério está estreitamente relacionado com o primeiro, e explorada a necessidade de o paciente possuir o máximo de informação acerca do tratamento de que é alvo. Assim as explicações mais pormenorizadas por parte dos profissionais de saúde podem ser bastante importantes (Taylor e Benger, 2004).

O terceiro critério aponta os tempos de espera como um fator fulcral para a avaliação dos cuidados de saúde prestados num serviço de urgência. Os pacientes que são sujeitos a um tempo de espera mais elevado apresentam uma menor satisfação com os cuidados de saúde prestados (Figura 2.1). Este fator é potencializado pelo facto de a maioria do tempo que um paciente passa num serviço de urgência dever-se aos tempos de espera impostos em diversas fases do atendimento: espera pela triagem, pela primeira

observação médica, por exames, transporte entre salas, e até na alta administrativa (Press Ganey, 2010; Booth et al., 1992).

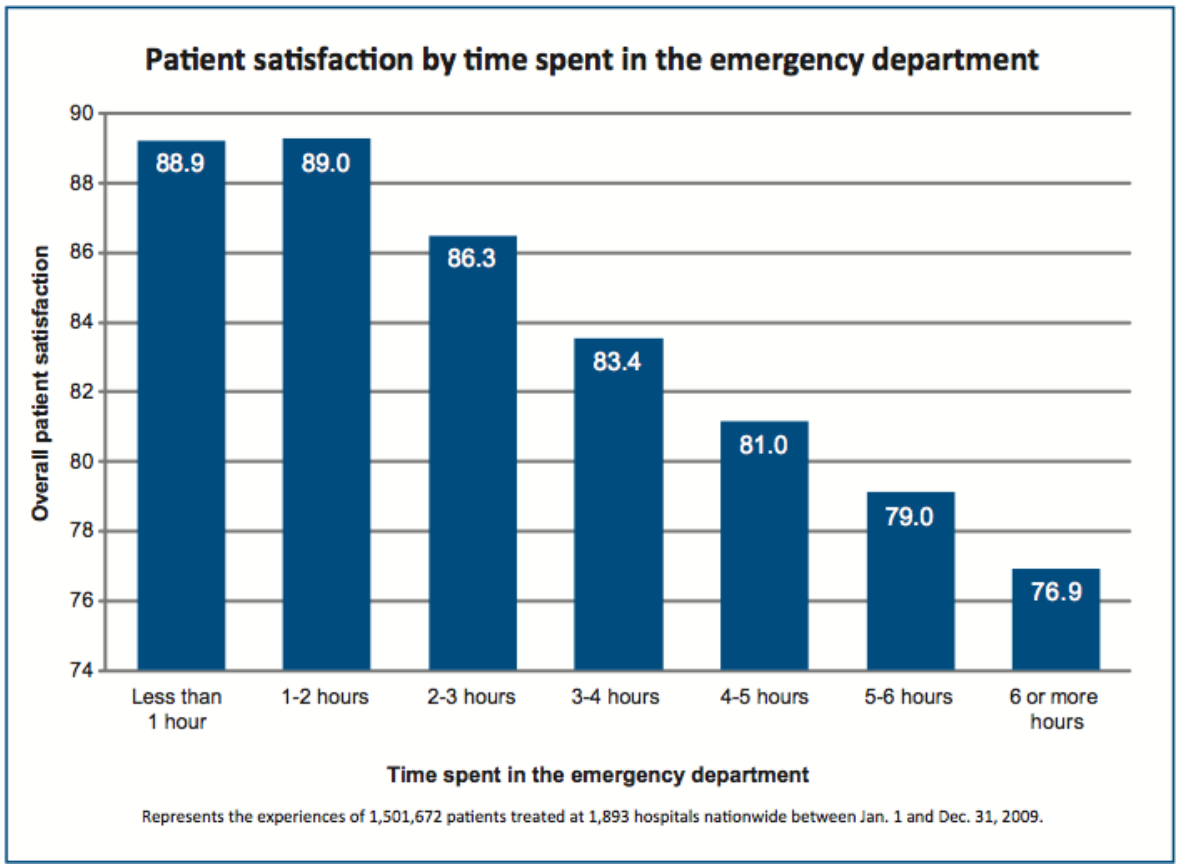


Figura 2.1: Nível de satisfação dos pacientes em função do tempo passado no serviço de urgência (adaptado de (Press Ganey, 2010))

Diversos estudos realizados em diferentes países apontam para o crescimento na procura da prestação de cuidados de saúde através do serviço de urgência, verificando-se uma tendência simultânea para a redução do número de serviços de urgência disponíveis. Estes aspetos têm resultado na sobrelotação deste serviço, aumento dos tempos de espera e consequentemente num aumento da taxa de pacientes a abandonar o serviço de urgência sem receber os cuidados de saúde pretendidos (Derlet e Richards, 2000; Bindman et al., 1991; Institute of Medicine, 2007).

Em Portugal o projeto "SIM-Cidadão"disponibiliza dados acerca das reclamações apresentadas pelos utentes do Serviço Nacional de Saúde. No caso das Unidades e Centros Hospitalares (UCH), é incontornável destacar que o serviço que mais reclamações reúne

é a Urgência Geral (Figura 2.2) e a causa mais frequentemente apontada para apresentar uma reclamação é o tempo de espera excessivo num serviço de urgência (Figura 2.3) (Direção-Geral da Saúde, 2011).

Serviços Hospitalares	2009		2010		Δ%
	n	%	n	%	
Urgência Geral	15.975	30,15	13.873	28,65	-4,98
Consulta Externa	1.942	3,67	1.775	3,67	0,01
Gestão de Doentes	724	1,37	728	1,50	10,02
Pediatria	568	1,07	567	1,17	9,22
Ortopedia	523	0,99	530	1,09	10,88
Portaria / Segurança	556	1,05	522	1,08	2,72
Cirurgia Geral	528	1,00	512	1,06	6,10
Medicina Interna	446	0,84	452	0,93	10,89
Imagiologia	461	0,87	435	0,90	3,24
Pediatria - SU	411	0,78	396	0,82	5,42

Figura 2.2: Os 10 serviços mais visados nas reclamações efetuadas nas Unidades e Centros Hospitalares (adaptado de (Direção-Geral da Saúde, 2011))

UCH					
2009			2010		
Causas	n	%	Causas	n	%
Tempo de espera no serviço de urgência	6.720	18,86	Tempo de espera no serviço de urgência	6240	18,85
Tempo de espera para atendimento	4.242	11,91	Tempo de espera para atendimento	3284	9,92
Falta de cortesia	1.937	5,44	Falta de cortesia	2157	6,51
Regras inadequadas/inaplicáveis	1.510	4,24	Má prática	1414	4,27
Má prática	1.467	4,12	Regras inadequadas/inaplicáveis	1287	3,89
Perfil desadequado	1.211	3,40	Tempo de espera para consultas de especialidade	1106	3,34
Tempo de espera para consultas de especialidade	1.205	3,38	Perfil desadequado	1086	3,28
Falta de Informação aos familiares	973	2,73	Falta de Informação aos familiares	878	2,65
Sistema de acompanhantes e visitas	955	2,68	Desrespeito no trato interpessoal	856	2,59
Desrespeito no trato interpessoal	925	2,60	Sistema de acompanhantes e visitas	810	2,45

Figura 2.3: As 10 causas mais mencionadas nas reclamações efetuadas nas Unidades e Centros Hospitalares (adaptado de (Direção-Geral da Saúde, 2011))

Pela análise realizada até este ponto, verifica-se que o acesso atempado ao atendimento num serviço de urgência é um dos fatores determinantes para a satisfação dos pacientes, apresentando-se desta forma como um dos principais critérios para avaliar a qualidade dos serviços prestados. Os hospitais, mesmo estando cientes da importância da redução dos tempos de espera, têm sérias dificuldades em contrariar o cenário de longas filas de espera, não conseguindo encontrar soluções ótimas para a número de profissionais de saúde disponíveis em cada momento, assim como para a organização desses profissionais e das suas tarefas (Hoot e Aronsky, 2008; Institute of Medicine, 2007).

No que diz respeito à alocação de profissionais de saúde em função da afluência de pacientes, existem vários estudos que indicam quais os padrões de afluência aos serviços de urgência em função da hora e do dia da semana. Estes estudos também indicam abordagens para lidar com esta realidade (Green et al., 2006; Jensen e Crane, 2008). A Figura 2.4 mostra a variação de pacientes que recorrem aos serviços de urgência, em função da hora do dia e do dia da semana (Green et al., 2006). Cada serviço de urgência é caracterizado por uma realidade distinta, o que leva a que estes estudos possam não ser perfeitamente adequáveis a todas as situações. No entanto, servem como base de referência e podem ajudar na tomada de decisão dos hospitais.

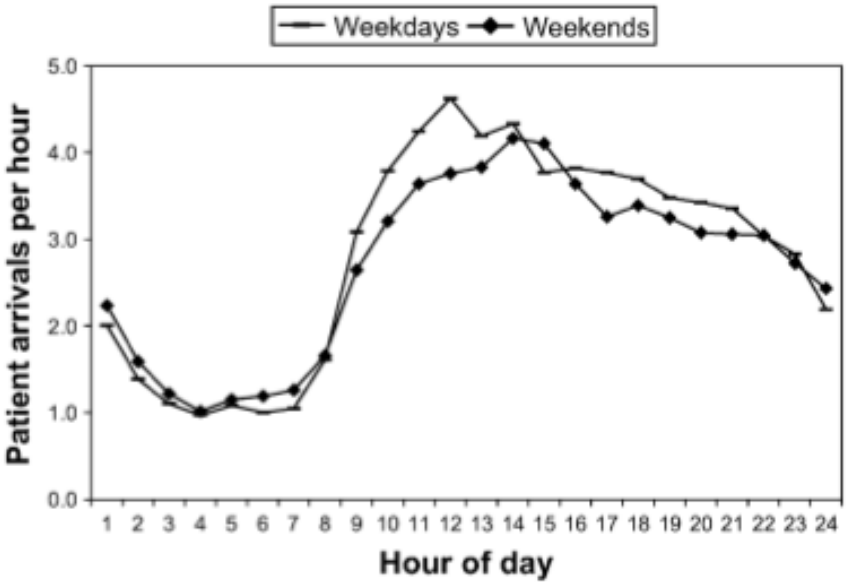


Figura 2.4: Média de pacientes que chegam em cada hora do dia. Os dias da semana são também comparados com os dias do fim de semana (adaptado de (Green et al., 2006))

A questão da organização do fluxo de trabalho dos profissionais de saúde, assim como a otimização da distribuição de tarefas são fatores organizativos essenciais para a correta gestão de recursos de um serviço de urgência hospitalar. Uma abordagem correta e profissional sobre estes tópicos pode ter um efeito imediato na redução do tempo de espera (Mason et al., 2006). A triagem, abordada em pormenor na Secção 2.3, é um excelente exemplo de uma tarefa que se encontra devidamente estruturada e organizada em quase todas as instituições que a adotam. O facto da tarefa de triagem

ser realizada de uma forma padrão, através de um processo claro e eficiente, permite reduzir o tempo despendido com a triagem em si, mas também permite que várias tarefas tirem proveito dos resultados triagem, uma vez que a utilização desse resultado pode ser útil para classificar o paciente nas mais variadas situações.

A otimização de pequenas tarefas é muitas vezes esquecida no contexto de um grande serviço de urgência, no entanto, a melhoria na eficiência destas tarefas leva a uma melhoria global do desempenho do serviço e consequentemente a uma redução dos tempos de espera (Campbell e Sinclair, 2004). Nesta dissertação, é explorada em pormenor a tarefa de transporte de pacientes no interior do serviço de urgência. Esta pequena tarefa pode ser bastante importante para o funcionamento eficiente do serviço de urgência, uma vez que influencia diretamente o atendimento atempado dos pacientes que necessitam de auxilio para se mover. Uma organização menos correta desta tarefa pode fazer com que o paciente não esteja presente atempadamente na sala onde é chamado, promovendo também atrasos gerais nas chamadas dos restantes pacientes. Na Secção 4.1 são descritos todos os pormenores associados a esta tarefa, bem como as possibilidades de melhorar o desempenho dos profissionais que a efetuam.

A redução do tempo de espera através da organização das tarefas do serviço de urgência é alvo de estudo de diversos autores. No entanto, estes estudos não focam a forma como deve ser realizada a organização de cada tarefa específica. A sua aposta passa por apresentar os princípios que devem ser seguidos para que seja obtida uma organização bem sucedida para as tarefas em geral (Mason et al., 2006):

- Melhorar os processos de tomada de decisão;
- Responsabilização de cada profissional pelo trabalho efetuado;
- Aumentar a motivação dos profissionais de saúde através da formação de pequenas equipas de trabalho;
- Treinar os profissionais para uma realização eficiente e qualificada das tarefas;
- Facilitar o acesso à informação relaciona com a tarefa e com o paciente.

2.3 Sistema de Triagem de Manchester

Os sistemas de triagem utilizados nos serviços de urgência hospitalar, normalmente, são orientados para identificar facilmente os casos clínicos mais urgentes, de forma a assegurar que estes recebem um tratamento prioritário. Em circunstâncias normais, um serviço de urgências tem recursos suficientes para tratar todos os paciente que afluem aos seus serviços. No entanto, os pacientes menos urgentes deverão estar sujeitos a um tempo de espera mais longo (Iserson e Moskop, 2007; Guisán et al., 2001).

O grupo de triagem de Manchester foi criado em 1994, tendo como objetivo estabelecer um consenso entre médicos e enfermeiros, no que diz respeito a normas para a realização da triagem (Manchester Triage Group, 2006). O Sistema de Triagem de Manchester, implementado durante no ano de 1997 em Manchester, é o resultado desse esforço. Rapidamente começou a ser amplamente divulgado pelo Reino Unido e mais tarde por outros países Europeus. O Grupo de Manchester permite a utilização do Sistema de Triagem de Manchester em Portugal, tendo apoiado a formação de médicos e enfermeiros para que estes estejam preparados para trabalhar segundo as normas e critérios seguidos por este sistema de triagem (Ministério da Saúde, 2006).

O Sistema de Triagem de Manchester, tal como a maioria dos sistemas de triagem, tem como principal objetivo identificar de uma forma objetiva e sistematizada os critérios de gravidade que indicam a prioridade clínica com que o paciente deve ser tratado, assim como o respetivo tempo de espera máximo a que esse paciente deve estar sujeito. Neste processo não está envolvido qualquer diagnóstico ao caso clínico do paciente.

O método de triagem proposto pelo sistema de Manchester consiste em identificar o sintoma inicial e seguir o respetivo fluxograma de decisão (total de 54 variantes). O fluxograma consiste em várias questões a serem colocadas ao paciente pela ordem especificada, constituindo os designados discriminadores. (Ministério da Saúde, 2006).

A utilização deste sistema classifica o estado do paciente em 5 categorias que podem ser identificadas por um número, nome, cor e tempo de espera (tempo limite aconselhado até ao início de observação médica). A Figura 2.5 ilustras as 5 categorias existentes.

Esta metodologia permite identificar precocemente os casos clínicos urgentes, de uma

Figura 2.5: Categorias de classificação da prioridade no Serviço de Urgência segundo o Sistema de Triagem de Manchester.

forma objetiva e contínua ao longo do tempo. Em caso de existir um agravamento da situação clínica do paciente, este deverá ser retriado de acordo com o elemento mais diferenciador em relação à triagem anterior (Manchester Triage Group, 2006).

Em Portugal, o Sistema de Triagem de Manchester, já pode ser considerado como uma norma nacional para a prática da triagem, isto em função do número elevado de implementações bem sucedidas em diferentes hospitais e também pelas orientações a nível dos organismos que gerem o destino da saúde em Portugal (Ministério da Saúde, 2006).

CAPÍTULO 3

Seleção da base tecnológica

Quando se pretende desenvolver um aplicação informática, a escolha das tecnologias a usar é uma das decisões mais importantes a realizar no planeamento do desenvolvimento da aplicação. No que diz respeito a tecnologias, é complicado afirmar que existe uma escolha ideal para resolver um problema, uma vez que existe uma variedade enorme de opções com características semelhantes que dificilmente se destacam em todos os aspetos importantes para a aplicação. No entanto, existe uma escolha que irá adequar-se melhor às necessidades da aplicação a desenvolver. Para tal, é importante ter em conta um conjunto de fatores que pode e deve influenciar a escolha dos programadores.

Em primeiro lugar é necessário realizar uma análise aprofundada aos objetivos e pré-requisitos da aplicação, de forma a perceber quais as características tecnológicas que são mais importantes no produto final. Em paralelo com esta análise é também importante ter em conta o contexto da aplicação, ou seja, quais as restrições existentes como necessidades específicas do utilizador alvo, especificações de acesso e armazenamento de dados, condições de integração com outras aplicações do sistema de informação, etc. Um último aspeto a ter em conta, que pode ser importante quando existem várias opções viáveis, é a experiência do programador em desenvolver aplicações recorrendo a determinadas tecnologias, uma vez que esta pode diminuir de forma considerável

o tempo de desenvolvimento. Todos os aspetos de decisão aqui apresentados serão utilizados para justificar as escolhas efetuadas.

3.1 Aplicações Web

Uma das principais questões que se colocam atualmente ao planear o desenvolvimento de uma aplicação é a escolha do paradigma de funcionamento, sendo que neste contexto surge a opção entre duas vertentes: as aplicações Web e as aplicações *Desktop*.

As aplicações tradicionais instaladas no sistema operativo do utilizador foram durante muitos anos a única opção para o desenvolvimento de aplicações. Estas aplicações *Desktop* continuam a ser utilizadas, principalmente, devido ao aproveitamento completo das potencialidades do hardware disponível na máquina do utilizador, tanto a nível de processamento como a nível gráfico. Outro aspeto que continua a ser favorável é o total controlo sobre o aspeto final da aplicação no ecrã do utilizador. (Dearle, 2007) Apesar das suas vantagens, uma aplicação *Desktop* pode não ser a solução mais indicada. Isto devido à necessidade de instalação, necessidade de esforço extra para realizar atualizações, dependência do sistema operativo para o qual é criada, ou devido à complicada integração com utilizadores remotos (Rossi et al., 2008).

A Web por seu turno, tem evoluído a um ritmo que seria impensável quando em 1991 foram publicados os primeiros documentos com a intenção de partilhar informação e investigações científicas. Desde então, em menos de 20 anos, tornou-se uma ferramenta indispensável para a maioria das pessoas e empresas. De uma Web que fornecia informação sobre produtos e serviços através de um conjunto de websites estáticos que apenas podiam ser consultados pelos visitantes, passou-se a uma Web centrada no utilizador, sendo este o principal responsável pelo conteúdo de cada página que visualiza, seja através das suas preferências como através da partilha da sua própria informação. Este novo modo de funcionamento, com receção de informação dinâmica, juntamente com uma forte componente interativa levou à designação de Web 2.0. Esta forma mais dinâmica e interativa de utilizar a Web foi em grande parte impulsionada pela habilidade criar páginas Web recorrendo a conteúdo armazenado em bases de dados (Rossi et al., 2008; Sven et al., 2009).

A melhoria funcional e estética das aplicações Web esteve inicialmente relacionada com desenvolvimentos de tecnologias que permitiram o processamento do lado do cliente, tais como AJAX, Javascript, ActiveX, Java ou Flash. As aplicações Web com este tipo de tecnologia são descarregadas por completo para o Browser do cliente, que as executa localmente. Para além de fazer com que a aplicação seja pesada para a máquina do cliente, o maior problema das tecnologias que permitem o processamento do lado do cliente é facto de não serem suportadas de igual modo por todos os Browsers e sistemas operativos (Sven et al., 2009). É principalmente por estas razões que surgem tecnologias Web, tais como ASP, PHP, Java EE ou ASP.NET, que levam o processamento para o lado do servidor. Em aplicações que usam este tipo de tecnologias, todo o código é executado do lado do servidor, e quando a sua execução termina, é enviada para o utilizador uma página HTML normal que pode ser visualizada em qualquer Browser (Rossi et al., 2008). A Figura 3.1 demonstra as diferenças entre os modelos de processamento no cliente e no servidor.

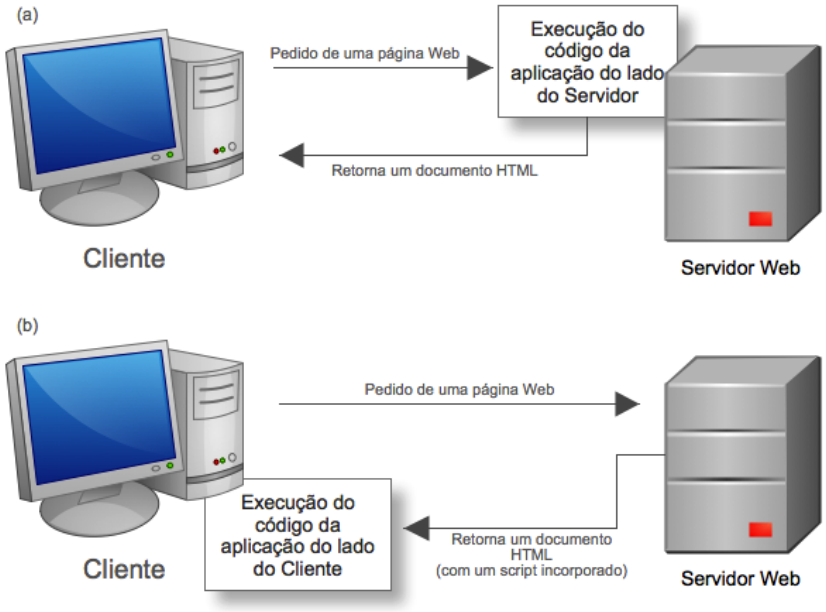


Figura 3.1: Diferenças entre a execução de código do lado do servidor (a) e do lado do cliente (b) (adaptado de (Macdonald, 2009))

Um dos maiores impulsos dados ao crescimento da utilização de aplicações Web foi o aparecimento do conceito de *Rich Internet Applications* (RIA), que se refere a apli-

cações Web independentes da plataforma, que são executadas no Web Browser do utilizador, não necessitando de instalação, mas que ainda assim apresentam características e funcionalidades semelhantes às aplicações tradicionais. O conceito de RIA representa a evolução do Browser de uma interação estática "pedido – resposta" para uma interação dinâmica e assíncrona. Este tipo de aplicações é orientado à usabilidade e interatividade que normalmente falhava nas aplicações tradicionais (Busch e Koch, 2009). Assim, apesar do domínio das aplicações Web que efetuam o processamento de código no servidor, a programação de determinadas funcionalidades no lado do cliente é uma realidade e permite criar interfaces mais ricas e robustas que respondem de forma mais afirmativa às necessidades do utilizador. Desta forma é normal surgirem associados os dois tipos de processamento, tirando partido das melhores características de cada um deles (Rossi et al., 2008). Um exemplo da junção destas tecnologias é a inclusão das potencialidades do AJAX como extensão à plataforma ASP.NET.

Com os avanços nas tecnologias que permitem uma interação natural e fluente entre o utilizador e a interface da aplicação Web, estas aplicações parecem aproximar-se das aplicações tradicionais em quase todos os aspetos. Assim, as aplicações Web perfilam-se como o futuro para a realização da maioria das nossas tarefas, através da exploração de vantagens como a facilidade de utilização, a independência da plataforma usada pelo utilizador, ou a menor quantidade de recursos exigidos à máquina do utilizador (Jazayeri, 2007). No entanto, há ainda questões a ter em conta para não optar sempre pelas aplicações Web: as diferenças de visualização entre browsers continuam a ser uma realidade e não permitem ter certeza do aspeto do produto final no dispositivo do utilizador. Outro aspeto prende-se com a instabilidade da Web, uma vez que algumas características das aplicações Web continuarão a depender de uma constante ligação ao servidor (Rossi et al., 2008).

Para determinar qual a melhor opção para desenvolver uma aplicação é necessário analisar os prós e contras de cada opção, tendo em conta os objetivos e pré-requisitos impostos no planeamento da aplicação. No contexto desta dissertação existem três aspetos que podem influenciar a escolha: a aplicação deve ser independente da arquitetura da máquina onde vai ser executada; deve existir a possibilidade de a aplicação ser facilmente adaptada a dispositivos móveis, de forma a que as mesmas funcionalidades

possam ser desempenhadas independentemente do dispositivo utilizado; a interface da aplicação deve ser prática e intuitiva de forma a otimizar a sua utilização.

Os dois primeiros pré-requisitos são cumpridos apenas pela solução Web. O último pré-requisito é mais fácil de cumprir recorrendo à solução *Desktop*, no entanto, com o estado atual da tecnologia Web é possível obter resultados satisfatórios também com uma solução Web (Torchiano et al., 2010). Desta forma, a solução mais adequada é optar por uma aplicação Web, uma vez que é a única que garante o cumprimento de todos os pré-requisitos.

3.2 ASP.NET

Tal como foi referido na Secção 3.1, as aplicações Web podem ser desenvolvidas recorrendo a diversas tecnologias que, essencialmente, se agrupam de acordo com o seu modo de processamento: tecnologias que efetuam o processamento no lado do cliente depois da aplicação ser descarregada e tecnologias que efetuam o processamento no servidor e apenas de seguida enviam para o cliente a página pedida.

Atualmente, a criação de aplicações Web com recurso a processamento no servidor é prática amplamente consensual no desenvolvimento Web. No contexto desta dissertação, as aplicações Web as que que melhor se adequam às necessidades da aplicação, uma vez que garante independência tanto do browser como do sistema operativo do cliente, garantindo também a segurança do código executado no servidor assim como menos recursos por parte da máquina do cliente. Apesar desta escolha, não é de excluir a inclusão de processamento do lado do cliente com o intuito de melhorar os mecanismos de interação entre o utilizador e aplicação, tornando a aplicação mais atrativa e fácil de usar, sem por em causa o correto funcionamento das funcionalidades base da aplicação devido a falhas na execução de código no cliente (Macdonald, 2009).

Para criar aplicações Web com processamento de dados no servidor é necessário escolher qual a linguagem de programação na qual vai ser escrito o código. Para esse efeito, existem várias opções que poderiam ser tidas em conta, como Python, Java, Ruby ou Perl. No entanto, principalmente devido à maior difusão no mercado e à enorme

quantidade de informação disponível, apenas duas opções foram analisadas de uma forma mais aprofundada: PHP e ASP.NET.

PHP é uma linguagem de código fonte aberto que permite desenvolver aplicações Web com bastante versatilidade, sendo por isso amplamente utilizado em todo o mundo. O código PHP que se encontra numa página pedida é executado pelo *runtime* PHP existente no servidor e só depois é enviada a página processada. A simplicidade do desenvolvimento de páginas PHP é positivo em diversos aspetos mas faz com que a sua estrutura de desenvolvimento misture as três componentes essenciais de uma aplicação Web: apresentação, código e acesso aos dados. Esta complicação é resolvida pelas plataformas de desenvolvimento, que numa tentativa de simplificar o desenvolvimento de software, implementam padrões de arquitetura de software para estruturar o seu desenvolvimento em diferentes camadas lógicas independentes. No entanto, cada plataforma de desenvolvimento pode seguir um modelo próprio para estruturar a aplicação, pelo que se torna importante perceber o seu funcionamento (Minetto, 2007; Vaswani, 2009).

ASP.NET, por sua vez, não pode ser considerada uma linguagem de programação no seu sentido convencional. No seu conceito fundamental, o ASP.NET pode lidar com várias linguagens de programação, podendo ser definido como uma ferramenta que funciona como *runtime* no servidor, de forma a permitir a execução de aplicações Web desenvolvidas com recurso à *.NET framework*. Para tal, o ASP.NET suporta quase todas as ferramentas disponíveis na biblioteca de classes *.NET framework* e tem uma ligação direta com esta cada vez que é necessário executar código no servidor (Guthrie e Robsman, 2007; Macdonald, 2009).

Por vezes, a distinção entre os conceitos de ASP.NET e *.NET framework* não é clara e o termo ASP.NET é usado para abordar toda as ferramentas da *.NET framework* que permitem o desenvolvimento de aplicações Web. Esta definição acaba por não ser a mais indicada para diferenciar os dois conceitos, uma vez que iria englobar muitas ferramentas da *.NET framework* que não são utilizadas exclusivamente para criar aplicações Web, como por exemplo as linguagens de programação ou ferramentas de acesso a bases de dados (Macdonald, 2009). No entanto, como no contexto desta dissertação apenas serão abordadas aplicações Web, o termo ASP.NET será usado para referir toda as ferramentas que permitem o desenvolvimento de aplicações ASP.NET, passando-se

também a usar o termo "aplicação ASP.NET" para referir aplicações Web criadas com recursos às ferramentas ASP.NET.

A tecnologia ASP.NET apresenta algumas caraterísticas próprias que fizeram com que a sua utilização fosse um sucesso em vários domínios do desenvolvimento Web. Este sucesso levou ao domínio deste mercado apesar de ser uma solução proprietária a enfrentar uma forte concorrência de soluções de licença aberta (MacDonald et al., 2010; Freeman, 2011).

- Um primeiro aspeto é a sua integração na *.NET framework*, uma vez que esta possui uma coleção de diversas partes funcionais, sendo que cada uma dessas partes contém dezenas de milhares de classes. Com tanta informação é de destacar que a sua organização em *namespaces* é bastante completa e intuitiva.

- O ponto anterior permite, desde logo, destacar também a possibilidade de programar em várias linguagens.

- A compilação do código para uma linguagem intermédia e a sua posterior execução num *runtime* comum é também importante, uma vez que permite tirar partido das vantagens associadas às linguagens de alto nível.

- Outro aspeto importante é o facto de ser orientada ao objeto. A coleção de objetos disponíveis não é impressionante, mas esse facto é ultrapassado pela possibilidade de explorar todas as potencialidades da programação orientada a objetos, sendo bastante interessante adicionar funcionalidades aos objetos existentes, como por exemplo no caso dos controlos em que é possível alterar apresentação ou resposta a eventos.

Pelas características anteriormente apresentadas é possível verificar que ambas tecnologias, PHP e ASP.NET, se adequam ao desenvolvimento de uma aplicação Web com os objetivos propostos nesta dissertação. Apesar das semelhanças, há fatores que podem determinar qual a melhor solução. Assim, destaca-se o facto de a solução ASP.NET implementar uma programação orientada a objetos que, para além das potencialidade que lhe são endereçadas, é também indicada devido à experiência adquirida em trabalhos anteriores. Para além deste aspeto, é possível indicar mais dois que, apesar

que não terem sido abordados até ao momento, são importantes no contexto da aplicação pretendida. Primeiro, o facto de existirem mais ferramentas de fácil utilização para enriquecer a interface de aplicações ASP.NET, destacando-se para esse efeito, a extensão ASP.NET AJAX. Por último, importa destacar que em ASP.NET é possível escolher entre dois modelos de desenvolvimento de aplicações Web (Web Forms ou MVC), enquanto que em PHP a escolha é mais confusa, uma vez que cada plataforma de desenvolvimento define o seu modelo de organização para o desenvolvimento de aplicações Web.

Segundo os aspetos apresentados até este ponto, decidiu-se optar por desenvolver uma aplicação ASP.NET. De seguida, serão analisados alguns conceitos relacionados com as aplicações ASP.NET, de forma a perceber quais as opções que podem ser tomadas para adequar a utilização desta vasta ferramenta às necessidades da aplicação desenvolvida no contexto desta dissertação.

3.2.1 Modelos de desenvolvimento Web em ASP.NET

De seguida são apresentadas os dois modelos disponibilizados pela *.NET framework* para o desenvolvimento de aplicações ASP.NET.

3.2.1.1 Web Forms

Esta tecnologia tenta criar uma camada de abstração para duas tecnologias subjacentes às aplicações Web: HTTP e HTML. Assim o ASP.NET Web Forms consiste um conjunto de controlos para criar a interface com o utilizador. Estes controlos estão incluídos no núcleo ASP.NET e que possuem funcionalidades de gestão de estado e ocorrência de eventos, permitindo simular a experiência de desenvolvimento de aplicações *Desktop* realizado com a tecnologia Windows Forms. A inclusão da funcionalidade *drag and drop* e a possibilidade de colocar o código num ficheiro separado, fazem com que o desenvolvimento de aplicações Web seja mais intuitivo e semelhante aos padrões desenvolvimento com que os programadores "*Desktop*" estão familiarizados. Este tipo de abordagem faz sentido, uma vez que esta tecnologia foi desenvolvida numa fase embrionária das aplicações Web em que era importante cativar muitos dos programadores

"*Desktop*" (Freeman, 2011).

O desenvolvimento de aplicações ASP.NET é estruturado de acordo com o modelo de código que for escolhido: *Single-Page* (ficheiro único com código e linguagem de elementos visuais) ou *Code-Behind* (ficheiros separados, um com os elementos visuais que formam a estrutura da página e outro com o código executável). Apesar de não existirem diferenças ao nível de execução do código, a possibilidade de optar por um modelo com ficheiros separados é bastante prática, uma vez que permite separar as tarefas de um designer das tarefas de um programador, não havendo risco de interferências indesejadas. Para além disso, o código pode ser reutilizado numa página com visual distinto (Freeman, 2011).

3.2.1.2 *MVC framework*

O conceito de MVC (*Modal-View-Controller*) já existe desde do final da década de 1970 mas está agora de volta devido à sua perfeita adequabilidade à natureza *stateless* dos pedidos HTTP. A ideia central passa pela divisão das aplicações em três partes lógicas distintas: O modelo, que inclui o código para a lógica específica do negócio da aplicação (por exemplo, a lógica de acesso a dados ou as regras de validação). A visualização, que cria uma representação adequada do modelo através da sua tradução para uma página HTML. E por último, o controlador, que coordena todas as operações, como lidar com as interações com utilizador, atualizar o modelo ou passar a informação para a visualização (MacDonald et al., 2010; Freeman, 2011). A Figura 3.2 demonstra, de uma forma simplificada, a interação entre as três componentes do modelo.

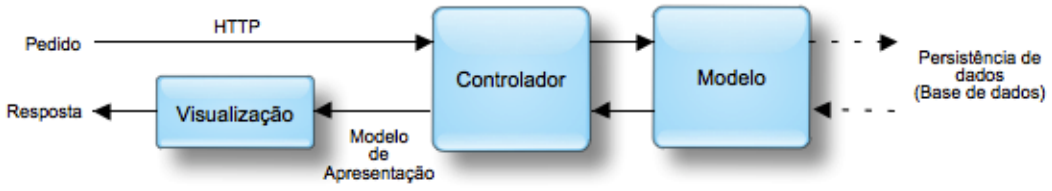


Figura 3.2: Interação entre os três componentes do modelo MVC (adaptado de (Freeman, 2011))

ASP.NET MVC corresponde a um novo conjunto de ferramentas que foram incluídas no núcleo ASP.NET e que sustentam uma nova abordagem para o desenvolvimento de aplicações Web, divergindo por completo do tipo de raciocínio que estava associado a tecnologia Web Forms, principalmente, pelo seu contacto próximo com o HTTP e HTML (Freeman, 2011).

3.2.1.3 Solução mais adequada

De acordo com as características apresentadas para cada modelo de desenvolvimento, ambos podem ser adotados para criar a aplicação pretendida. Apesar das enormes potencialidades do modelo MVC, este é difícil de implementar para utilizadores que não estejam ambientados à sua metodologia de desenvolvimento. Assim, devido à experiência passada em desenvolvimento com Web Forms e sabendo-se que este respeita os requisitos da aplicação a desenvolver, a utilização de Web Forms é a solução mais adequada para este projeto.

3.2.2 ASP.NET Ajax

O código ASP.NET tradicional é executado por completo no servidor. Assim, sempre que ocorre algum evento na página que o utilizador está a usar, o browser necessita de enviar alguma informação para o servidor, receber uma nova cópia da página como resposta e atualizar a página apresentada ao utilizador. Este processo, apesar de ser relativamente rápido, introduz sempre um momentâneo flash na página, para além de não ser adequado para eventos muito frequentes numa página, como movimentos do rato ou teclas pressionadas (Wenz, 2007).

Os programadores sempre tentaram evitar este tipo de problemas utilizando Javascript, uma vez que esta é a linguagem de script executada do lado do cliente que é suportada de forma mais abrangente nos browsers atuais. De certa forma, a utilização de Javascript já tem sido prática comum para melhorar o funcionamento de controlos ASP.NET mais complexos. Por exemplo, um Menu ASP.NET responde imediatamente ao evento do deslocamento do rato por cima de um título recorrendo ao Javascript para alterar a visualização desse controlo, não havendo necessidade de efetuar *postback*. No

entanto, em determinadas situações é necessário atualizar a informação contida numa porção da página e a única solução existente é efetuar *postback* para o servidor e receber uma nova página HTML completa. Apesar deste mecanismo funcionar, continua a apresentar perdas de rendimento (Wenz, 2007; MacDonald et al., 2010).

Para melhorar os mecanismos de atualização de informação em páginas Web, foram desenvolvidos novos métodos que utilizam processamento do lado do cliente recorrendo ao Javascript. Uma das soluções que ganhou mais destaque foi o Ajax, apresentando-se como um atalho programático para técnicas aplicadas no lado do servidor que permitem fazer pedidos ao servidor e atualizar o conteúdo de uma determinada região sem atualizar a página completa.

Numa página Ajax, quando os evento são tratados por um controlo Ajax, é executado código do lado do cliente para efetuar um pedido assíncrono ao servidor. O servidor recebe este pedido, executa o código necessário e envia a informação que a página necessita, normalmente através de blocos XML. Do outro lado, o cliente, ao receber esta nova informação, utiliza-a para atualizar uma parte da página sem que haja perceção de atualização por parte do utilizador.

A crescente utilização do Ajax, em grande parte, está relacionada com a possibilidade de obter informação adicional do servidor sem atualizar a página por completo. Esta funcionalidade apenas é possível devido ao objeto *XMLHttpRequest*, uma vez que este permite ao cliente efetuar pedidos assíncronos ao servidor. Apesar deste conceito ser relativamente simples, permite às aplicações Web ter uma atualização limpa e mais agradável para o utilizador, tal como acontece nas aplicações *Desktop* (MacDonald et al., 2010; Roy e Gibbs, 2007; Wenz, 2007). Na Figura 3.3 são ilustrados os modos de atualização de uma página Web com e sem Ajax.

O ASP.NET AJAX disponibiliza um conjunto de controlos e outros componentes que podem ser usados para construir páginas ASP.NET. Estes controlos já incluem todo o Javascript necessário para garantir os seus efeitos. Assim, é possível desenvolver uma aplicação Web com efeitos Ajax programando de forma intuitiva tal como no modelo ASP.NET. O controlo sobre os efeitos Ajax não é elevado mas também não é exigido o mínimo conhecimento e esforço. De seguida são brevemente descritos alguns dos

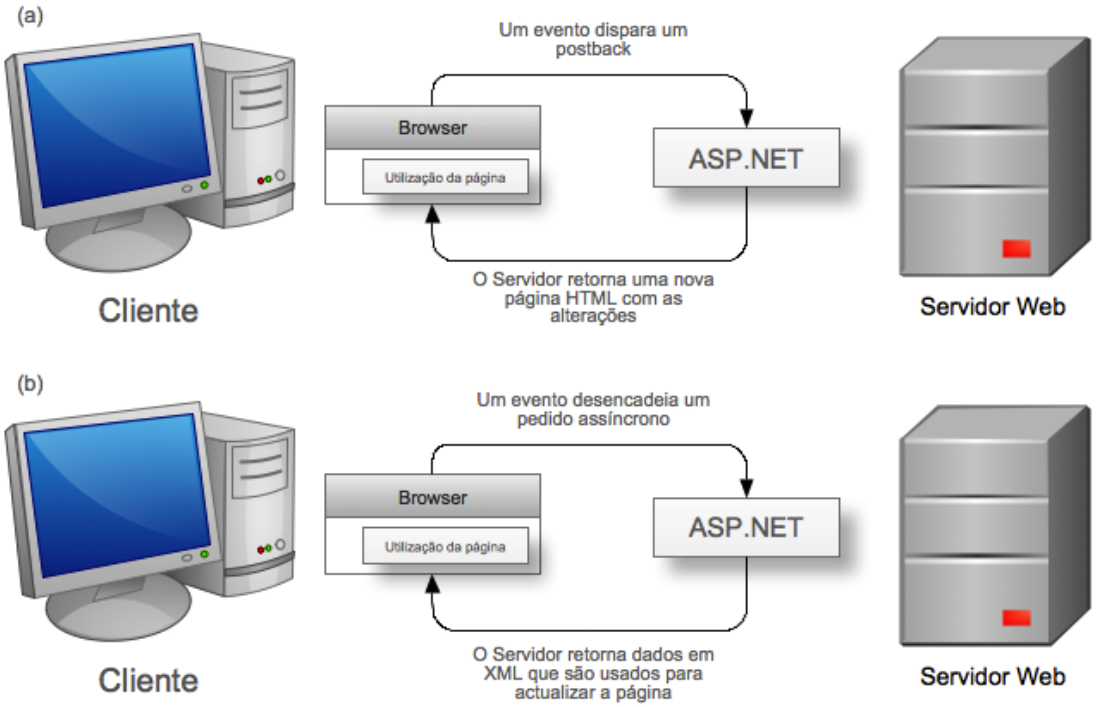


Figura 3.3: Comparação entre a atualização de uma página ASP.NET sem a tecnologia Ajax (a) e com a tecnologia Ajax (b) (adaptado de (MacDonald et al., 2010))

controlos mais importantes (MacDonald et al., 2010; Wenz, 2007):

- *ScriptManager* – é o componente que faz a ligação entre os controlos ASP.NET AJAX e as bibliotecas Javascript da plataforma ASP.NET AJAX. Assim, para que os controlos ASP.NET AJAX funcionem é necessário colocar este controlo na página.

- *UpdatePanel* – é o controlo que possibilita atualizações parciais nas páginas ASP.NET. A ideia consiste em dividir a página em blocos de atualização. Assim, quando um controlo está dentro de um *UpdatePanel* e dispara um evento que normalmente resultaria num *postback* completo da página, irá promover apenas a atualização do *UpdatePanel* em questão, uma vez que o evento é intercetado e resulta num pedido assíncrono. Este tipo de comportamento também pode ser obtido para controlos que não pertençam a um *UpdatePanel*. Para tal, são adicionados *Triggers* ao *UpdatePanel* de forma a que este intercete um evento específico, de um controlo específico, e seja desencadeado um pedido assíncrono

em vez do habitual *postback* (Roy e Gibbs, 2007).

- *Timer* – é um controlo que permite realizar atualizações à página sem que haja um evento desencadeado por uma ação do utilizador. A sua utilização consiste apenas na sua adição à página e na definição do intervalo de tempo, em milissegundos, que passa entre atualizações. O evento resultante tanto pode resultar num *postback* como num pedido assíncrono, dependendo se o evento *Tick* do controlo *Timer* foi ou não adicionado aos *Triggers* de um *UpdatePanel*.

Este conjunto ASP.NET AJAX apresenta funcionalidades interessantes mas é um pouco limitado no número de funcionalidades disponibilizadas, uma vez que apenas é constituído por três controlos. No entanto, o ASP.NET inclui uma sofisticada biblioteca de funções Javascript que podem ser usadas para criar uma diversidade enorme de efeitos mais avançados. Estas bibliotecas não são orientadas para os programadores de aplicações Web que se devem preocupar antes com a lógica de negócio, mas são ótimas para que os programadores de componentes ASP.NET possam desenvolver novos controlos fácies de utilizar nas aplicações Web. O conjunto mais conhecido de controlos desenvolvidos com recurso à tecnologia Ajax, é o *ASP.NET AJAX Control Toolkit*. Este *toolkit* consiste em dezenas de controlos que usam as bibliotecas ASP.NET AJAX para criar efeitos avançados. Para além dos novos controlos, são disponibilizadas ainda extensões para os controlos base ASP.NET, incrementando efeitos extra para melhorar a funcionalidade de cada controlo (MacDonald et al., 2010; Wenz, 2007).

Atualmente, devido à utilização massiva do Ajax, todos os browsers mais conhecidos suportam a sua utilização desde as seguintes versões:

- Firefox (qualquer)
- Google Chrome (qualquer)
- Internet Explorer 5
- Netscape 7
- Opera 7.6
- Safari 1.2

Apesar da vasta compatibilidade, há uma minoria de utilizadores a utilizar browsers que não suportam Ajax ou que usam os browsers com Javascript desativo. Nestes casos, a resposta de uma página Web com tecnologia ASP.NET Ajax, depende das funcionalidades utilizadas e da forma como são implementadas. Por exemplo, em aplicações que usem atualizações parciais nas suas páginas através do controlo *ASP.NET Update Panel*, a página iria continuar a funcionar em browsers que não suportem Ajax, efetuando apenas a substituição de atualizações parciais por *postbacks* da página completa. Outros controlos mais complexos podem não funcionar de forma correta ou simplesmente não funcionar (MacDonald et al., 2010).

3.3 Fundamentos da *.NET framework*

Tal como já foi dado a entender na Secção 3.2, a *.NET framework* não é uma tecnologia fácil de definir. A melhor forma de compreender o seu conceito é trata-la como um aglomerado de tecnologias que foram desenvolvidas para disponibilizar uma nova plataforma de desenvolvimento e execução de aplicações e *Web Services*. No seu desenvolvimento, a Microsoft teve por base alguns objetivos que permitiram atingir o sucesso atual. Desde logo, a disponibilização de um ambiente de programação orientado a objetos em que o código pode ser executado localmente ou remotamente. O ambiente de execução é um dos pontos chave, uma vez que visa promover a execução segura de código, a minimização de conflitos devido a diferenças de software de desenvolvimento e por último a eliminação dos problemas de performance dos ambientes de interpretação e script. Por último, houve também o esforço de tornar consistente a experiência de programação entre desenvolver aplicações Web e aplicações Windows (Macdonald, 2009).

Os principais componentes desta vasta plataforma serão explorados de seguida, evidenciando os aspetos mais relevantes no contexto desta dissertação.

3.3.1 Linguagens de programação .NET

Apesar da *.NET framework* permitir a execução de diversas linguagens de programação, esta incorpora duas linguagens essenciais que são normalmente usadas para desenvolver

aplicações Web: *Visual Basic* (VB) e *C#*. Estas linguagens são, em grande parte, equivalentes a nível de funcionalidades. De acordo com a Microsoft, a escolha de programar em VB ou *C#* é uma escolha que apenas depende do estilo do programador, e que não afeta a performance, interoperabilidade, ferramentas disponíveis, ou o tempo de desenvolvimento das aplicações. Assim, sendo a escolha da linguagem de programação uma decisão dependente das preferências do programador, decidiu-se optar pela utilização de VB devido à experiência passada em trabalhos com esta linguagem (Macdonald, 2009; MacDonald et al., 2010).

3.3.2 Linguagem Intermédia

Todas as linguagens .NET são compiladas noutra linguagem de baixo nível antes do código ser executado, dando origem a ficheiros com extensão EXE ou DLL. Esta linguagem de baixo nível é conhecida como CIL (*Common Intermediate Language*). O *runtime* da *.NET framework* apenas executa código escrito nesta linguagem intermédia.

Todas as linguagens .NET foram desenvolvidas com base no CIL, sendo esta a razão pela qual tanto o VB como o *C#* apresentam as mesmas funcionalidades e performance. Esta compatibilidade é formalizada com a CLS (*Common Language Specification*), tratando-se de uma especificação que garante que um componente escrito numa linguagem .NET pode ser utilizado em qualquer outra linguagem .NET (Macdonald, 2009; MacDonald et al., 2010).

3.3.3 Funcionamento do CLR

O *Common Language runtime* (CLR) é o motor de execução de código da *.NET framework*. Este *runtime* apenas executa código CIL, pelo que suporta todas as linguagens .NET. No entanto, o código CIL não é executado diretamente, havendo um passo intermédio de compilação que transforma o código CIL em código máquina nativo adequado à plataforma onde a aplicação está a ser executada (Figura 3.4). Para além desta função central, este fornece também diversas funcionalidades relacionadas com a gestão de objetos e com a verificação e otimização de código.

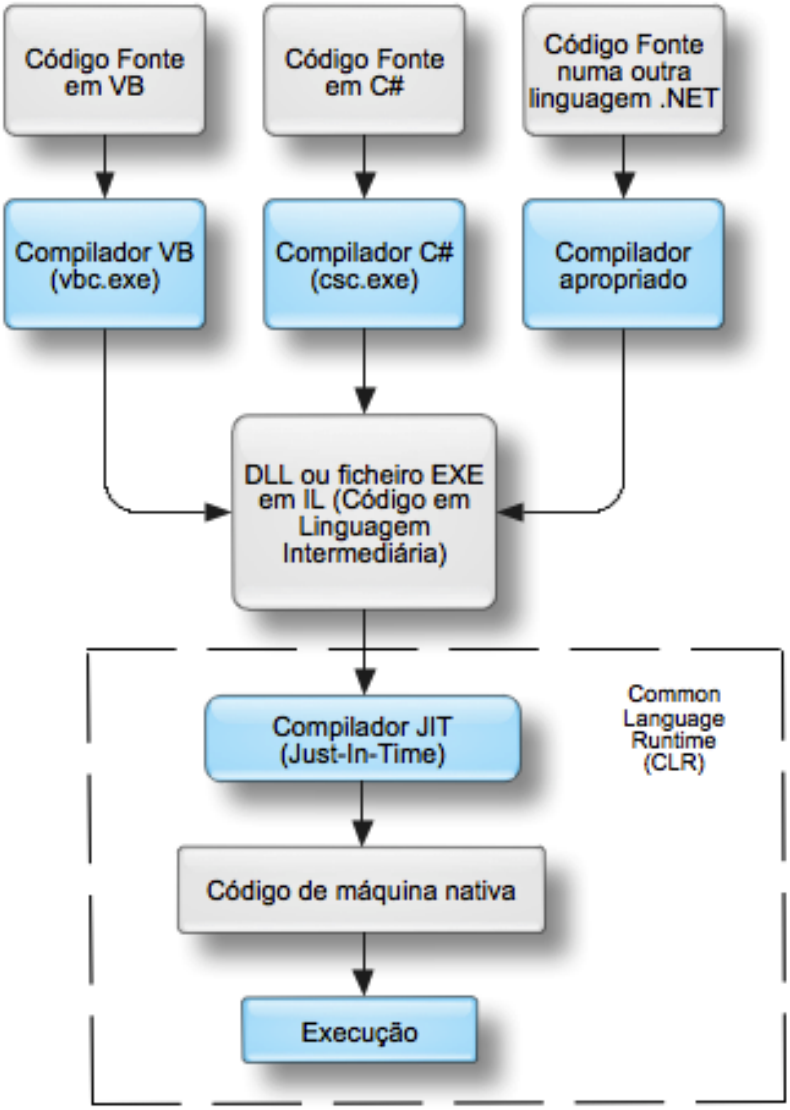


Figura 3.4: Execução de código na *.NET framework* (adaptado de (Macdonald, 2009))

No caso de aplicações ASP.NET, quando um cliente pede uma página, esta já está compilada em código CIL. Além disso, pode também existir uma versão compilada em código máquina nativo guardada em cache, aumentando a performance das operações realizadas no servidor. Depois do código máquina ser executado, a página HTML final é criada e enviada para o utilizador (Macdonald, 2009; MacDonald et al., 2010).

3.3.4 Biblioteca de classes .NET

A biblioteca de classes da *.NET framework* é um enorme repositório de classes que disponibilizam funcionalidades pré-concebidas para realizar uma enorme variedade de tarefas. Outras linguagens de alto nível como o Java também utilizam o conceito de biblioteca de classes. No entanto, a biblioteca de classes .NET é mais ambiciosa e abrangente do que em qualquer outra plataforma. Qualquer linguagem .NET pode usar as funcionalidades existentes na biblioteca de classes .NET, o que ajuda a aumentar a consistência entre as diferentes linguagens e elimina a necessidade de instalar numerosos componentes específicos. As classes existente nesta biblioteca podem ser específicas para o desenvolvimento Windows, especificas para o desenvolvimento Web ou podem ser comuns a ambos.

Esta biblioteca de classes pode ser vista como um conjunto, bem estruturado, de ferramentas de programação. A infraestrutura criada por este enorme conjunto de classes leva a que um programador necessite apenas de escrever o código relacionado com a lógica de negócio da sua aplicação. Por exemplo, numa aplicação Web, todos os assuntos relacionados com transações e concorrência em bases de dados são tratados por esta infraestrutura invisível de classes, permitindo que centenas de clientes acedam à mesma página ao mesmo tempo, sem que o programador necessite de regular esse acesso através do seu código (Macdonald, 2009).

3.3.5 ADO.NET

O ADO.NET é a tecnologia própria da *.NET framework* que permite efetuar o acesso a dados a partir de diferentes fontes de dados. Apesar do seu modelo de acesso a dados ser direcionado para bases de dados relacionais, o facto de ser extensível permite alterar o seu modelo, de forma a aceder a dados organizados em sistemas não relacionais. Uma das principais vantagens do ADO.NET é permitir escrever aplicações Web praticamente com o mesmo código de acesso a dados que é necessário numa aplicação *Desktop*.

Uma das diferenças principais entre o ADO.NET e outras tecnologias de acesso a dados é a forma como esta lida com o desafio de trabalhar com diversas fontes de dados. Na maioria das tecnologias, o programador usa um conjunto genérico de objetos indepen-

dentes da fonte de dados. No entanto, o ADO.NET usa o modelo baseado em *data providers*, sendo que um *data provider* consiste num conjunto de classes específicas para efetuar o acesso dados, executar comandos e obter dados. As classes essenciais que constituem um *data provider* são: *Connection* (efetuar conexão), *Command* (executar comandos), *DataReader* e *DataAdapter* (receber os dados para coloca-los noutro objeto).

O ADO.NET não inclui nenhum *data provider* genérico, optando por incluir *data providers* criados especificamente para cada fonte de dados. Cada um destes *data providers* tem uma implementação específica das classes referidas anteriormente de forma a otimizar o acesso a uma fonte de dados. A *.NET framework* inclui um conjunto de quatro fornecedores:

- ***SQL Server provider*** – otimizado para bases de dados *SQL Server*.
- ***OLE DB provider*** – fornece acesso para qualquer fonte de dados que tenha um *OLE DB driver*.
- ***Oracle provider*** – otimizado para bases de dados Oracle.
- ***ODBC provider*** – fornece acesso para qualquer fonte de dados que tenha um *ODBC driver*.

Apesar de apenas existirem estes fornecedores, o facto do modelo ADO.NET ser extensível permite aos programadores criarem facilmente os seus próprios *providers* para aceder a fontes de dados relacionais ou não relacionais.

Ao escolher o *provider* a ser utilizado, é aconselhável escolher um que seja específico para a fonte de dados em questão e apenas quando não existir nenhum disponível, recorrer a um *OLE DB driver* ou a um *ODBC driver* compatível com a fonte de dados (MacDonald et al., 2010).

No contexto desta dissertação, a fonte de dados a ser utilizada é o SGBD Oracle. Em termos de *data providers*, existem várias soluções disponíveis para efetuar a conexão a bases de dados Oracle. Uma vez que existem fornecedores especializados, é de excluir a utilização de *drivers OLE DB* ou *ODBC*. A opção fica restrita a duas soluções: o

Oracle provider disponibilizado pela Microsoft e o *Oracle provider* disponibilizado pela Oracle para efetuar conexões em aplicações .NET.

Nas versões mais recentes da *.NET framework*, o *Oracle provider* da Microsoft é considerado obsoleto, pelo que, apesar de continuar em funcionamento, a Microsoft aconselha a utilização do fornecedor de dados criado pela própria Oracle (ODP.NET), uma vez que esta disponibiliza um suporte especializado para tipos de dados próprios da Oracle, como *timestamp*, *large objects* ou dados XML. Este *data provider* disponibilizado para Oracle será abordado em pormenor na Secção correspondente ao SGBD Oracle (MacDonald et al., 2010).

3.3.6 *Data Binding*

O processo de obtenção de informação proveniente de uma fonte de dados é apenas um dos desafios do processo de disponibilizar informação ao utilizador. As aplicações modernas necessitam de disponibilizar a informação nas suas páginas através de uma visualização intuitiva, flexiva e atrativa.

As aplicações ASP.NET beneficiam de modelo bastante desenvolvido de ligação entre os componentes de acesso a dados e os componentes de visualização, ou seja, o modelo de *data binding*. Os *data source controls* permitem estabelecer uma ligação entre uma página Web e a fonte de dados, ligando diretamente à fonte de dados ou indiretamente através de componentes personalizados de acesso a dados. Uma vez configurados e associados a um controlo de apresentação, os *data source controls* tratam de todos os pormenores relacionados com o processo de *data binding*, ou seja, tratam de utilizar os componentes de acesso a dados, fazendo os pedidos necessários, e passam a informação obtida ao controlo de apresentação de forma a que este apenas tenha de enquadrar os dados recebidos nos padrões de visualização já definidos (MacDonald et al., 2010).

Os *data source controls* disponibilizados pela *.NET framework* são os seguinte:

- ***SqlDataSource*** – permite estabelecer conexão a qualquer fonte de dados que tenha um *data provider* ADO.NET.
- ***ObjectDataSource*** – permite estabelecer conexão a uma classe personalizada

de acesso a dados. Esta é a abordagem predileta para o desenvolvimento de aplicações profissionais.

- ***AccessDataSource*** – permite ler e escrever dados num ficheiro de uma *base de dados Access*.
- ***XmlDataSource*** – permite estabelecer ligação a um ficheiro XML.
- ***SiteMapDataSource*** – permite ligar ao ficheiro *Web.sitemap* que estabelece a estrutura de navegação do Website em questão.

De seguida são analisados os dois *data source controls* que se apresentam como hipótese para serem utilizados na aplicação ASP.NET a ser desenvolvida no contexto desta dissertação.

3.3.6.1 *SqlDataSource*

O *SqlDataSource* é controlo ASP.NET que se inclui na parte HTML da página Web, como qualquer outro controlo, representando uma conexão à base de dados através de um *provider* ADO.NET. O *SqlDataSource* utiliza atributos na sua etiqueta de declaração para indicar as *queries SQL* necessárias para chamar os comandos disponíveis. Quando um comando necessita de parâmetros para ser chamado, estes parâmetros são determinados numa etiqueta própria para o comando, como *<SelectParameters> ... </SelectParameters>* para o comando *select*.

Independentemente do *provider* ADO.NET utilizado, o *SqlDataSource* utiliza uma forma genérica de criar os objetos necessários (*Connection*, *Command*, ou *DataAdapter*). Apesar do programador não ter de se preocupar com o código necessário para efetuar a conexão, é importante perceber que única forma de permitir este tipo de abordagem é através da utilização do *data provider factory*. Desta forma, o código pode ser escrito utilizando objetos genéricos, havendo uma conversão para os objetos específicos do *data provider* a utilizar através da configuração do *data provider factory*.

A utilização do controlo *SqlDataSource* para efetuar *data binding*, apesar de ser bastante intuitiva e permitir o acesso direto aos dados, também leva o programador a colocar toda a lógica de acesso a dados na parte de design da página. Este procedimento

é um pouco controverso, uma vez que a separação destes componentes é aconselhável (MacDonald et al., 2010).

3.3.6.2 *ObjectDataSource*

O *ObjectDataSource* é um controlo que, assim como os restantes, se inclui nas declarações HTML de uma página Web, permitindo a ligação entre um controlo de representação de dados e um componente personalizado de acesso a dados. O *ObjectDataSource* é consideravelmente flexível, pelo que pode funcionar com uma variedade enorme de componentes. No entanto, a classe que contém a lógica de acesso aos dados tem que estar em conformidade com algumas regras:

- Toda a lógica deve estar presente na mesma classe, ou estando dividida em várias classes, estas devem ser envolvidas numa única classe de nível superior.
- Deve fornecer os resultados de uma *query* quando um método único é evocado.
- Quando o resultado de uma *query* são vários registos, estes devem ser representados numa coleção desses objetos .
- Podem ser utilizados métodos que usem instâncias ou métodos estáticos. No entanto, ao utilizar instâncias, os métodos da classe devem ter um construtor base sem argumentos.
- As instâncias devem ser *stateless*, uma vez que o *ObjectDataSource* cria uma instância quando necessita e destrói-a no final do pedido.

Os comandos disponíveis e o nome dos respetivos métodos da classe de acesso aos dados que vão ser executados são indicados na etiqueta de declaração do *ObjectDataSource*. A indicação do método que executa uma determinada operação é introduzida na parte de design da página, no entanto, toda a lógica de acesso aos dados e tratamento de dados está contida na classe de acesso a dados. Quando um comando necessita de parâmetros para ser chamado, estes parâmetros são apenas referenciados na parte de design, sendo o seu valor definido no tratamento do evento desencadeado pela chamada do comando em questão. Esta abordagem permite uma definição muito mais flexível

dos argumentos necessários em cada método evocado, assim como também permite efetuar tratamento dos valores introduzidos, prevenindo erros por valores incorretos ou inválidos (MacDonald et al., 2010).

3.4 Plataforma de desenvolvimento: *Visual Studio*

O *Visual Studio* é uma ferramenta de desenvolvimento, incluída na *.NET framework*, que disponibiliza um ambiente rico para desenvolver rapidamente aplicações bem estruturadas e com um nível de complexidade avançado. Apesar de ser possível criar aplicações Web através da escrita de código num simples editor de texto, esta tarefa seria muito mais difícil, trabalhosa e propícia a erros. Apesar de existirem outras ferramentas para o desenvolvimento de aplicações ASP.NET, o *Visual Studio* apresenta algumas características vantajosas (Macdonald, 2009):

- **Design das páginas Web** – facilita o desenvolvimento de páginas atrativas através da mecanismo *drag and drop*.
- **Deteção automática de erros** – informa o programador dos erros existentes à medida que este escreve código, indicando também situações potencialmente problemáticas.
- **Ferramentas de *Debugging*** – permite avaliar o código em funcionamento e acompanhar o conteúdo das variáveis. Também fornece um servidor Web virtual apenas para *Debug*.
- ***IntelliSense*** – disponibiliza uma poderosa ferramenta de apresentação de sugestões para completar instruções. Apresenta também informação importante como número e tipo dos parâmetros das funções ou o tipo de valor retornado de uma função.

3.5 Servidor Web: IIS

Quando uma aplicação ASP.NET está pronta a ser disponibilizada aos utilizadores, esta é publicada num servidor IIS (*Internet Information Services*). O IIS é a solução

da Microsoft para o mercado dos servidores Web, apresentando uma vasta gama de funcionalidades que rivaliza com as restantes soluções no mercado, como por exemplo servidores Apache. No entanto, desde o lançamento da *.NET framework*, os servidores IIS ganharam um novo argumento, uma vez que a gestão de aplicações ASP.NET passou a ser uma das suas funcionalidades fulcrais (Macdonald, 2009). O IIS está originalmente incorporado nos seguinte sistemas operativos:

- Windows Server 2003 (IIS 6.0)
- Windows Vista e Windows Server 2008 (IIS 7.0)
- Windows 7 e Windows Server 2008 R2 (IIS 7.5)

No entanto, os sistemas operativos para utilização comum (Windows Vista e Windows 7) não são aconselháveis para serem utilizados profissionalmente como servidor Web, uma vez que normalmente têm uma versão limitada do IIS, sendo apenas permitidas 10 ligações ao servidor simultâneas (Macdonald, 2009).

3.6 Outras Ferramentas

3.6.1 HTML

HTML, ou *HyperText Markup Language*, é uma linguagem especialmente concebida para criar documentos Web facilmente interpretáveis. Para tal, o HTML recorre a um conjunto de instruções especiais, designadas etiquetas (de *tags* ou *markup*), que são usadas para definir a estrutura e a disposição dos elementos numa página Web, assim como a sua visualização no browser (Shelly et al., 2008). Ao receber um novo documento HTML, o browser faz a sua interpretação para obter a correta apresentação da página, analisando a ordem pela qual os elementos aparecem, assim como o significado e comportamento de cada elemento. Para além disso, a maioria das etiquetas possui informação adicional na forma de atributos, sendo que essa informação tanto pode ser de caráter visual (definir características de estilo como altura, largura, cor, letra, etc) como de caráter funcional (por exemplo, definir qual o método executado em resposta a determinados eventos) (Schafer, 2010; Duckett, 2011).

No contexto das aplicações ASP.NET, o HTML continua a ser responsável pela estrutura da página, ou seja, apesar da extensão do ficheiro que contém a informação ser *.aspx*, toda a informação existente nesse ficheiro que visa definir a estrutura, disposição e visualização dos elementos da página é escrita com etiquetas HTML. As etiquetas utilizadas tanto podem pertencer ao conjunto de etiquetas base da versão HTML utilizada, como podem ser etiquetas que descrevam controlos ASP.NET (Macdonald, 2009).

3.6.2 CSS

Com a crescente utilização da Internet, as páginas Web passaram de simples documentos de texto a complexos documentos com apresentação cuidada e apelativa, recorrendo a diversos instrumentos para melhorar o seu aspeto. Uma dessas ferramentas é o *Cascading Style Sheet* (CSS) e o seu funcionamento consiste em permitir associar regras de estilo, escritas num ficheiro próprio, aos elementos que aparecem no documento HTML. Estas regras ditam a forma como o conteúdo desses elementos deve ser visualizado. As regras existentes num ficheiro CSS podem ser de três níveis (Powell, 2010):

- **Nível 1** – Regras que definem a visualização de um tipo de elemento. Por exemplo, ao declarar uma destas regras para o elemento *<p>*, todas as ocorrências deste elemento no documento irão respeitar esta regra.
- **Nível 2** – Regras que definem a visualização de uma classe de elementos, ou seja, a regra é aplicada a todos os elementos que tiverem o atributo class definido com o nome atribuído à regra.
- **Nível 3** – Regras que definem a visualização de um elemento com um determinado *id*, ou seja, estas regras são apenas aplicadas ao elementos que tenham o atributo *id* com o mesmo nome atribuído à regra.

O ficheiro CSS pode ser associado a um determinado documento HTML através da utilização da etiqueta *<link />*, sendo que este elemento é introduzido no interior do elemento *head*, presente em todos os documentos HTML. De seguida, é dado um exemplo (Código 3.1) que faz referência a um ficheiro CSS utilizando o elemento *link* com os atributos necessários (Duckett, 2011).

Código 3.1: Referência a um ficheiro CSS utilizando o elemento link.

```
1 <link rel="stylesheet" type="text/css" href="../exemplo.css" />
```

Para além da definição em ficheiro próprio, estas regras também podem ser definidas no documento HTML diretamente na linha de declaração do próprio elemento, com recurso ao atributo *style*. Por vezes, pode existir sobreposição de regras, o que pode levar a que mais do que uma regra seja utilizada para definir diferentes características. No entanto, a mesma característica pode ser definida em várias regras diferentes. Nesses casos, a definição mais específica é a predominante, logo, as regras são implementadas por esta ordem: definição com o atributo *style*, regras de tipo 3, regras de tipo 2 e regras de tipo 1. Por fim, no caso de existir mais do que uma regra, do mesmo nível para a mesma característica, é aplicada a mais recente, ou seja, a última no ficheiro CSS (Powell, 2010).

3.6.3 Javascript

O Javascript é uma linguagem de programação orientada a objetos que é interpretada e não compilada. A quando da sua criação, o Javascript foi pensado como uma linguagem que poderia ser incorporada em qualquer aplicação para a fornecer ferramentas de criação de scripts. No entanto, desde o seu lançamento, o Javascript começou a ter o seu interpretador incorporado nos Browsers Web mais conceituados, pelo que o sua utilização mais conhecida passou a ser a criação de scripts interpretáveis do lado do cliente (Crockford, 2008).

A utilização de Javascript na programação de aplicações Web do lado do cliente combina a facilidade de criação de scripts, fornecida pelos interpretadores incorporados, com as potencialidades do Modelo de Objeto do Documento (DOM). O DOM é uma API disponibilizada pelo browser Web num objeto designado *Document* e representa de forma programática os vários elementos de um documento (documento HTML, neste caso), permitindo aceder e alterar o conteúdo desses elementos. Ao tirar partido destas funcionalidades, os scripts do lado do cliente permitem abandonar o conceito de Web constituída por páginas HTML estáticas e passar para uma Web de páginas com conteúdo dinâmico, maior interatividade com o utilizador e com efeitos visuais inovadores

(Flanagan, 2006).

Para ter acesso a mais propriedades e métodos importantes na interação com o utilizador existe o objeto *window*. Este objeto permite efetuar diferentes tarefas como recolher informação específica acerca do dispositivo de visualização utilizado, abrir novas janelas ou confirmar ações através de avisos em janelas próprias. O objeto *window* é criado especificamente para cada janela que aparece no ecrã e difere do objeto *Document*, na medida em que este está incluído no objeto *window* e representa apenas uma parte das suas funcionalidades (Pollock, 2010).

Uma vez que nesta dissertação o Javascript é abordado como um script do lado do cliente utilizado em páginas Web, é importante perceber como é que este pode ser incluído/associado ao documento HTML de forma a que o Browser saiba quando executar os scripts existentes.

A forma mais comum de introduzir um script no documento HTML é usar as etiquetas *<script> ... </script>* colocando o código Javascript entre estas. No entanto, a primeira etiqueta necessita de incluir um atributo que especifique a linguagem na qual o script é escrito, ou seja, é necessário alterar para *<script type="text/javascript">*. Estas etiquetas também podem ser usadas para evocar um ficheiro Javascript externo (Código 3.2). Este ficheiro consiste num ficheiro de texto que apenas contém código Javascript, sendo utilizada a extensão *.js*. Com esta opção, é possível economizar bastante tempo através da reutilização de código por introdução da chamada do ficheiro externo em detrimento da cópia do código para cada página HTML (Pollock, 2010).

Código 3.2: Chamada de um ficheiro Javascript externo.

```
1 <script type="text/javascript" src="yourfile.js"></script>
```

A utilização de funções em Javascript pode trazer várias vantagens à organização do código, seja pela sua separação em blocos funcionais, como pela reutilização do seu código através da simples chamada da função.

A utilização do código Javascript é muitas vezes associada a eventos que ocorrem na página que o utilizador está a visualizar. Desta forma, as funções também ganham uma importância extra, uma vez que para executar uma função em resposta ao desencadear

de um evento apenas é necessário indicar a nome da função. Assim, para executar a função *soma()* ao clicar num determinado botão apenas é necessário incluir o seguinte atributo na etiqueta do botão (Código 3.3).

Código 3.3: Indicação da função Javascript que responde ao clique do botão.

```
1 <input type="button" value="Somar" onclick="soma();" />
```

Os eventos que podem ser usados para este efeito podem variar entre um simples clique, alteração do índice selecionado numa lista, ou o evento *load* do elemento *body* (Pollock, 2010).

3.7 Bases de dados relacionais: Oracle

Uma base de dados com a informação bem estruturada é ponto de partida essencial para armazenar informação de forma persistente, de forma a que posteriormente possa ser acedida e atualizada por qualquer aplicação.

O conceito de base de dados refere-se à forma como os dados são organizados, tanto a nível físico como lógico, com o intuito de se tornarem persistentes. O armazenamento de dados numa base de dados pressupõe certas características como correção, disponibilidade, usabilidade ou versatilidade, que são bastante vantajosas em relação a qualquer outro método de organização de dados persistentes. Desta forma, qualquer base de dados é construída com recurso a um sistema de gestão de bases de dados (SGBD). Este software de gestão disponibiliza ferramentas imprescindíveis para qualquer gestor de bases de dados, isto porque a estrutura de uma base de dados, normalmente, é muito complexa para ser gerida sem auxílio de um software adequado (Greenwald et al, 2008).

No contexto desta dissertação, relativamente à escolha do SGBD, apesar de existir um vasto leque de opções, não é necessário analisar qual é aquele que melhor se adequa ao problema apresentado, uma vez que os pré-requisitos já indicam que o SGBD a utilizar é Oracle. A base de dados Oracle será utilizada por outras aplicações que trabalhem com informação do mesmo domínio. Esta integração de informação proveniente de

diversas aplicações e de sistemas de informação distintos permite criar aplicações com informação constantemente atualizada.

De seguida são abordados alguns tópicos relacionados com o funcionamento dos serviços inerentes a uma bases de dados Oracle, principalmente numa perspetiva de acesso remoto por parte de um cliente.

3.7.1 *Oracle Net Services*

Um SGBD moderno, como é o caso do Oracle, necessita de estar preparado para os desafios do paradigma de negócio atual, ou seja, uma base de dados não pode limitar-se a disponibilizar os melhores serviços de gestão de dados a um nível de acesso local, tem de disponibilizar serviços que possibilitem o acesso remoto confiável durante 24 horas dos 7 dias de semana. Apenas desta forma uma base de dados pode satisfazer os requisitos de instituições que tenham aplicações Web a trabalhar constantemente com informação persistente (Oracle, 2008).

A solução Oracle para as questões relacionados com o funcionamento em rede das suas bases de dados é o *Oracle Net Services*. Estes serviços fornecem diversas soluções de conectividade para bases de dados em sistemas distribuídos e heterogéneos, facilitando as configurações de rede e a sua manutenção, assim como o diagnóstico ao funcionamento da rede. O componente básico deste conjunto de serviços é o *Oracle Net*, que permite estabelecer conexão entre uma aplicação Web, que se apresenta como cliente, e o servidor da base de dados.

Este componente de software (*Oracle Net*) está disponível no cliente e no servidor. Do lado do cliente a aplicação Web comunica com o *Oracle Net* e este comunica com o protocolo TCP/IP de forma a possibilitar a conectividade com o servidor da base de dados. O servidor da base de dados recebe o pedido através do protocolo TCP/IP e este encaminha os pedidos para o *Oracle Net* que, por sua vez, comunica com o SGBD para este processar o pedido do cliente. A Figura 3.5 exemplifica o funcionamento do *Oracle Net* numa arquitetura multicamada de três camadas.

Apesar do funcionamento do *Oracle Net* ser idêntico no cliente e no servidor da base

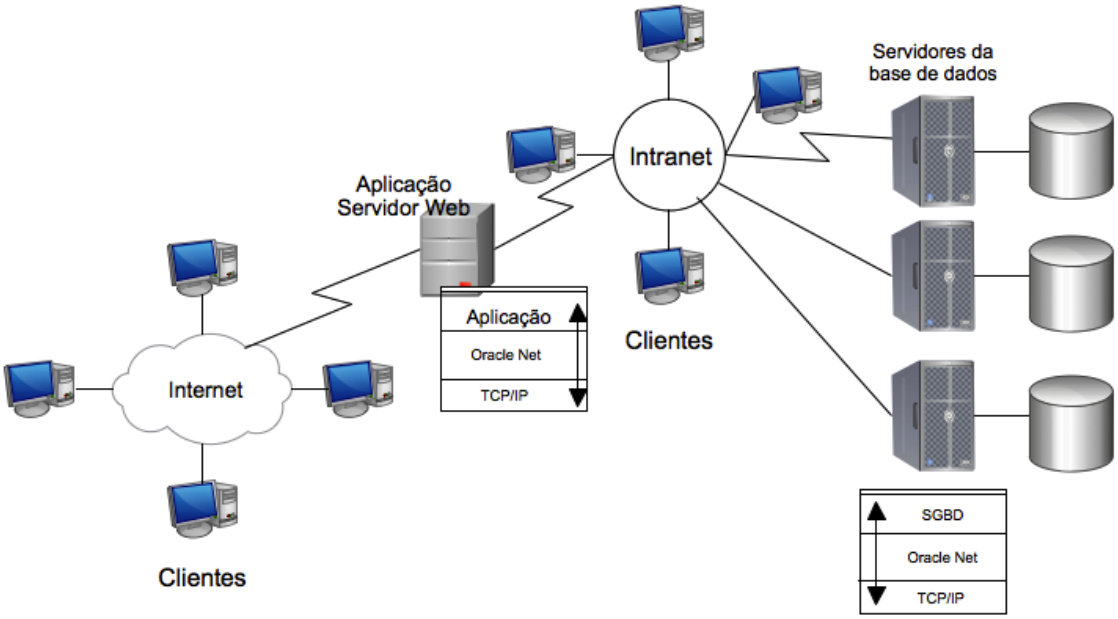


Figura 3.5: Arquitetura multicamada com o servidor Web a funcionar simultaneamente como servidor e cliente. *Oracle Net* estabelece a conexão entre servidor Web e servidor da base de dados (adaptado de (Oracle, 2008))

de dados, existe um operação que apenas é efetuada no servidor da base de dados. A operação em questão é responsável por receber a conexão inicial através de um *Oracle Net Listener* (normalmente designado apenas por *listener*). Este *listener* está configurado para um protocolo de rede e lida com os pedidos efetuados ao servidor por clientes configurados com o mesmo protocolo. Depois da conexão ser estabelecida, o cliente e o servidor comunicam sem intervenção do *listener*. A Figura 3.6 ilustra o funcionamento do *Oracle Net* tendo em conta a ação do *listener*.

Um servidor de base de dados pode ter várias bases de dados definidas como diferentes serviços que o cliente pode utilizar de forma distinta. Quando um cliente pretende conectar-se a um serviço específico, a identificação desse serviço é colocada no parâmetro *SERVICE_NAME* dos dados de conexão enviados pelo cliente. Os diferentes serviços estão registados no *listener*, pelo que a ação deste último encaminha o pedido para o serviço em questão (Oracle, 2008).

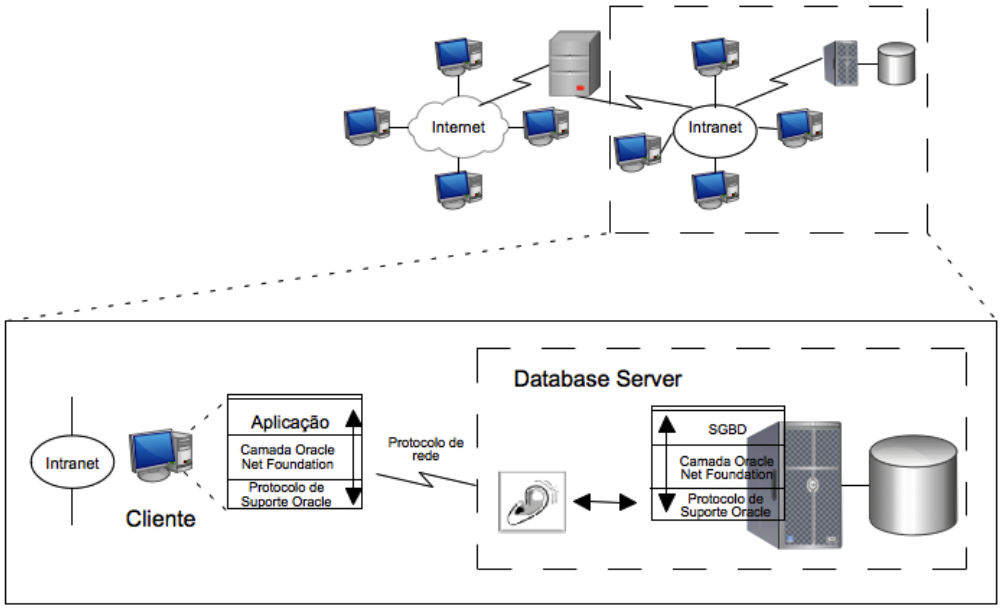


Figura 3.6: Funcionamento do *listener* na mediação das comunicações entre servidor Web e servidor de base de dados (adaptado de (Oracle, 2008))

3.7.2 Cliente Oracle

A utilização de uma base de dados Oracle através de uma ligação de rede permite ao cliente tirar partido de aplicações independentes da base de dados, que consultam e alteram a informação do seu domínio. Para que as aplicações Web, alojadas num servidor Web, possam comunicar com um servidor de base de dados Oracle é necessário que o servidor Web possua um *Oracle client* instalado. Este cliente Oracle inclui o componente *Oracle Net*, necessário para comunicações com a base de dados e recorre à *Oracle Call Interface* (OCI) para fornecer a base sobre a qual são criadas as diversas interfaces de acesso a bases de dados Oracle, específicas para cada linguagem de programação utilizada para desenvolver aplicações Web (Oracle, 2011a,b).

3.7.3 ODP.NET

A sigla ODP.NET surge da designação inglesa *Oracle Data Provider for .NET* e representa a solução proposta pela Oracle para efetuar o acesso a bases de dados Oracle no contexto ADO.NET. Com o ODP.NET, a Oracle fornece o seu próprio *data provider* para aplicações .NET, disponibilizando as mesmas funcionalidades de outros *data pro-*

viders mas otimizando o seu funcionamento de acordo com as características próprias de uma base de dados Oracle.

Entre as funcionalidades únicas que o ODP.NET possui, destaca-se a disponibilização de formatos de dados próprios da Oracle, como *timestamp*, *large objects* ou XML. Também se destaca a possibilidade efetuar operações avançadas nas transações e *queuing*. Em suma, a utilização do ODP.NET faz com que as aplicações ASP.NET tenham ligações à base de dados mais rápidas, estáveis, seguras e flexíveis (Oracle, 2011c).

3.7.4 SQL Oracle

O termo *Structured Query Language* (SQL) refere-se a uma linguagem especialmente concebida para lidar com diversas tarefas de gestão de dados em bases de dados relacionais. A sua utilização estende-se por vários tipos de operações, desde inserir, selecionar, atualizar e eliminar dados, passando pela criação e alteração de tabelas e relações entre estas, chegando também ao nível da gestão de acesso aos dados.

Tal como na maioria dos SGBD atuais, o SGBD Oracle utiliza o SQL como linguagem para gerir as suas bases de dados. No entanto, o dialeto SQL utilizado numa base de dados Oracle é uma implementação das normas definidas pelo *standard* SQL. Desta forma, existem pequenas diferenças na forma como a implementação Oracle do SQL lida com pequenos pormenores da linguagem mas, de um modo geral, as linguagens podem ser consideradas equivalentes em termos de funcionalidades e semelhantes em termos de escrita à exceção de alguns pormenores.

No que diz respeito a aplicações Web, estas enviam comandos SQL para serem executados na base de dados, de forma a ser possível efetuar as operações pretendidas (Mishra e Beaulieu, 2004).

CAPÍTULO 4

Aplicação desenvolvida

O desenvolvimento de software é um processo complexo que se pode tornar ainda mais complicado se não for devidamente organizado e planeado (Cusumano e Yoffie, 1999). A maioria das empresas de desenvolvimento de software dispensa recursos para procurar soluções que melhorem o seu processo de desenvolvimento de software através do seu planeamento e sistematização exaustivos. Este investimento visa aumentar a produtividade e a qualidade dos seus produtos. O conceito mais utilizado para abordar o planeamento deste processo é o ciclo de vida do desenvolvimento de software (SDLC – *Software Development Life Cycle*).

Existem vários modelos para estruturar o ciclo de vida de desenvolvimento de software e várias metodologias para os implementar. A escolha do modelo para o SDLC, a aplicar em cada caso, tem que ser ponderada de acordo com as características do projeto, ou seja, um projeto será atrasado se lhe for atribuído um modelo de SDLC muito complexo ou extenso para as suas características, ou será propício a erros e derrapagens orçamentais se lhe for atribuído um modelo demasiado simples ou curto para as suas características (Ruparelia, 2010; Madhavji, 1991).

No contexto da aplicação Web apresentada nesta dissertação, o objetivo passa pelo desenvolvimento de um software, desde a fase de análise de pré-requisitos até a uma fase

em que a aplicação, devidamente testada, possa ser utilizada num ambiente profissional. No entanto, o suporte técnico a possíveis utilizadores, assim como a elaboração de novas versões, não se encaixa nos objetivos propostos para o trabalho relacionado com esta dissertação, isto devido ao limite temporal do projeto e ao seu caráter de investigação. Assim, para este projeto, os modelos com uma fase de manutenção e revisão inexistente ou muito reduzida são mais indicados para estruturar o ciclo de vida do desenvolvimento de software.

Um vez que o projeto desta dissertação consiste apenas no desenvolvimento de uma aplicação e não na sua posterior melhoria ou *upgrade*, é apropriado falar apenas de um modelo para o processo de desenvolvimento de software e não do seu ciclo de vida. A diferença aqui pretendida reside no facto de não existir ligação entre o final do desenvolvimento de uma primeira versão do software e a definição de novos requisitos para uma nova versão, tal como ilustrado na Figura 4.1.

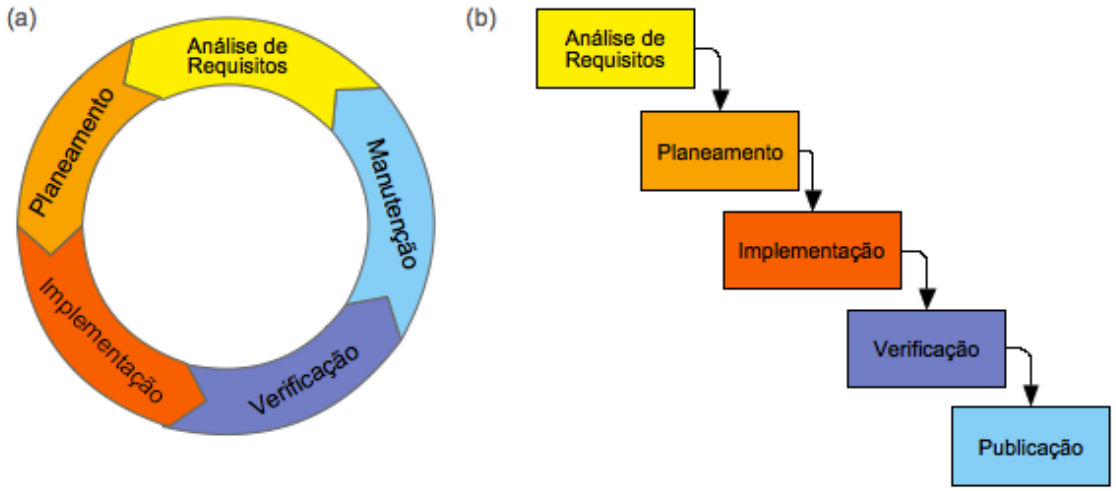


Figura 4.1: Dois esquemas de 5 fases que ilustram a diferença entre o ciclo de vida (esquerda) e o processo (direita) de desenvolvimento de software.

O modelo *waterfall* é um dos mais antigos modelos utilizados para estruturar o processo de desenvolvimento de software e é ainda hoje um dos mais conhecidos. Este modelo divide o processo de desenvolvimento em fases bem separadas, adotando um esquema em cascata para relacionar as tarefas entre si, tal como ilustrado na Figura 4.2a. Esta organização em cascata indica que uma tarefa seguinte só é iniciada quando

os propósitos da tarefa em execução são atingidos (Ruparelia, 2010; Parekh, 2005). As fases do modelo *waterfall* são:

- Definição e análise dos requisitos;
- Planeamento da aplicação e do sistema;
- Implementação e testes de cada parte da aplicação;
- Integração e testes ao sistema;
- Publicação e manutenção.

Este modelo tem como principal vantagem a distinção clara entre as diferentes tarefas que, associada à atribuição de objetivos específicos para cada tarefa, permite obter uma maior disciplina e qualidade na realização de cada tarefa. Outro aspeto positivo é a definição inicial de objetivos, ainda antes do inicio do desenvolvimento propriamente dito, uma vez que permite analisar o projeto como um todo e perspetivar as necessidades e dificuldades que deverão surgir (Satalkar, 2010a).

Apesar das vantagens, são também apontados alguns pontos negativos a este modelo rígido de desenvolvimento de software. Desta forma, dependendo do contexto da aplicação a desenvolver, poderão surgir novos objetivos à medida que o projeto vai avançando, pelo que pode ser necessária a reestruturação de alguns aspetos (Nurmuliani et al., 2004). Outro aspeto negativo é a rigidez da execução sequencial de tarefas. Por vezes é necessário voltar a uma tarefa anterior para melhorar certos aspetos, como por exemplo, voltar à fase de planeamento para evitar diversos problemas na implementação. Pela mesma razão, em diversos casos pode ser interessante realizar tarefas em paralelo e não de forma sequencial, tirando partido do conhecimento que vai sendo gerado (Parekh, 2005).

De forma a contornar as desvantagens da aplicação do modelo *waterfall* ao desenvolvimento de software, foram surgindo novas abordagens a este modelo, alterando pequenos pormenores que vão de encontro à construção mais dinâmica de software. De uma forma geral, pode-se destacar o aparecimento de um modelo *waterfall* modificado que mantém a divisão do processo desenvolvimento de software em cinco tarefas, tal

como sugerido no modelo original, no entanto, a rigidez da realização sequencial dessas tarefas é alterada para um modelo que permite a realização simultânea de várias tarefas, assim como voltar à tarefa anterior para corrigir pormenores que não estejam em conformidade com o desejado. Apesar desta maior flexibilidade do modelo, o concretizar de cada fase continua sujeito a uma validação dos resultados obtidos fazendo, para efeito, uma verificação da conformidade dos resultados em relação aos objetivos propostos para essa fase (Satalkar, 2010b). Na Figura 4.2 são ilustradas as diferenças entre o modelo *waterfall* original (Figura 4.2a) e modelo modificado (Figura 4.2b).

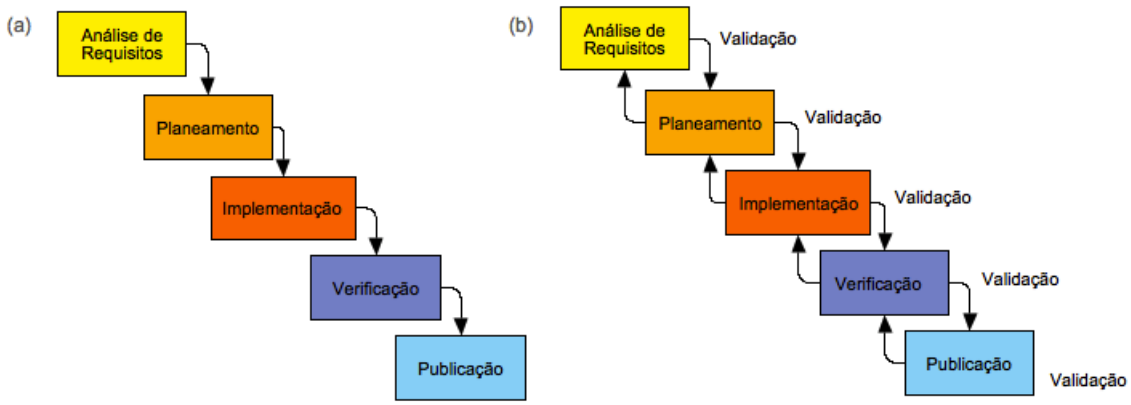


Figura 4.2: Diferenças entre a versão original do modelo *waterfall* (a) e a versão modificada (b) (adaptado de (Parekh, 2005; Satalkar, 2010b)

Nas secções seguintes será apresentada a abordagem efetuada ao problema que o projeto desta dissertação visa resolver, assim como a solução final proposta. Para tal, vai ser explicado o trabalho efetuado em cada fase do processo de desenvolvimento da aplicação, apresentando-se progressivamente a solução proposta para o problema.

4.1 Análise de requisitos

A aplicação desenvolvida no contexto desta dissertação tem vários requisitos que importa analisar e contextualizar. Desta forma é possível compreender a motivação de cada requisito e, consequentemente, encontrar as soluções para cada pormenor da aplicação que levam a obter resultados finais em concordância com os requisitos.

O objetivo central é criar uma aplicação baseada em tecnologia Web que permita gerir o fluxo de transportes internos de um serviço de urgência hospitalar. Para além deste objetivo central, outros requisitos são abordados na subsecções seguintes. De forma a permitir uma fácil alusão à aplicação Web desenvolvida, esta será referida utilizando a designação ***patientmove***.

4.1.1 Contextualização da aplicação

O recurso a soluções informáticas para melhorar a performance dos serviços hospitalares e das suas respetivas tarefas já começa a ser prática comum em muitas instituições de prestação de cuidados de saúde. Com os rápidos avanços tecnológicos a que se assiste atualmente, os sistemas de informação disponíveis nos hospitais vão adquirindo novas funcionalidades que influenciam diretamente a qualidade dos cuidados de saúde prestados aos pacientes. Este aumento de qualidade é bastante importante para as instituições e para os serviços governamentais, pelo que tem motivado investimentos no desenvolvimento de soluções informáticas que tratem de tarefas cada vez mais específicas.

No caso concreto do *patientmove*, este visa resolver um problema de otimização de uma tarefa realizada no serviço de urgência hospitalar. A tarefa em questão é o transporte de pacientes no interior de um Serviço de Urgência. Com esta aplicação é pretendido que a organização dos transportes seja substancialmente melhorada, disponibilizando um terminal de fácil utilização, constantemente atualizado e com as tarefas ordenadas por prioridade.

Uma vez que estamos enquadrados no contexto de uma urgência hospitalar, o tempo é um fator fulcral para todos os intervenientes e em especial para os utentes. Assim, através da utilização do *patientmove* é essencial que o tempo gasto a escolher o transporte a efetuar seja inferior àquele que era habitual com o sistema anterior.

A poupança de tempo pode ser conseguida através da disponibilização de interfaces rápidas e de fácil utilização (por exemplo através de menus simplificados), assim como através da sugestão otimizada do próximo transporte a efetuar (por exemplo, tendo em conta a distância mínima entre a localização do profissional de saúde e o local de início

do transporte). Outro aspeto importante é o facto de que tecnologia a usar deverá ser o mais leve possível em termos gráficos e de processamento, de forma a tornar a aplicação o mais rápida possível. Para tal, devem ser evitados efeitos visuais que prejudiquem o desempenho da aplicação ou que provoquem atrasos na visualização da informação.

4.1.2 Especificidades do "transporte de pacientes"

De uma forma muito simplificada, o transporte de um paciente no interior de uma urgência hospitalar está divido em três acontecimentos distintos: pedido de transporte, início e final.

O pedido de transporte é uma ação que pode ser realizada por diversos profissionais da área da saúde, podendo cada profissional pedir transportes em mais do que uma situação:

- O enfermeiro da triagem pode pedir transporte da sala de triagem para a sala de espera;
- O médico pode pedir transporte da sala de espera para o consultório;
- O médico pode pedir transporte do consultório para a sala de imagem;
- O técnico de imagem pode pedir transporte da sala de imagem para a sala de espera;

Estas são algumas das situações em que são efetuados pedidos de transporte de um paciente. Porém, podem existir mais situações que não são abordadas nesta lista. O ponto essencial é que estes pedidos vão ser desencadeados através da utilização das aplicações com que cada um desses profissionais trabalham diariamente. Assim, o *patientmove* não deverá ser responsável por efetuar pedidos de transporte, mas sim gerir os pedidos existentes para que seja mais rápido e fácil escolher o próximo transporte a ser efetuado.

A operação de iniciar um transporte é realizada por um auxiliar de saúde e consiste na visualização da lista de pedidos existentes e na consequente escolha de um dos transportes. Com esta escolha, a realização do transporte fica reservada, de forma a garantir

que outros profissionais não tenham a possibilidade de escolher o mesmo transporte. Esta é a operação mais importante do *patientmove*, uma vez que é neste passo que os pedidos de transporte são analisados e organizados de acordo com os critérios de prioridade definidos. Assim, a lista de pedidos que é apresentada ao utilizador estará devidamente ordenada, ajudando o profissional a escolher o transporte mais importante e a perder o mínimo de tempo possível.

A operação de finalizar um transporte também é realizada por um auxiliar de saúde e consiste na confirmação que o transporte foi efetuado com sucesso. Esta operação deve ser registada no *patientmove* de forma a que este seja encerrado e passe para o arquivo de transportes. O facto de haver uma lista de transportes reservados e de cada transporte ser removido dessa lista quando é finalizado, permite detetar facilmente transportes que sejam esquecidos depois de terem sido reservados.

Para além destas funcionalidades, uma ferramenta que lide com a problemática do transporte de pacientes tem que estar preparada para eventuais acontecimentos que não sejam exatamente os esperados. Assim, o registo dos seguintes acontecimentos deve estar disponível no *patientmove*:

- O profissional que efetua o transporte pode ter que cancelar a sua reservar de um transporte devido a diversos motivos. Assim o transporte voltará à lista de pedidos.
- O paciente pode abandonar as instalações do hospital, ser transferido para outra unidade, ou até falecer. Assim, poderá ser necessário abortar pedidos de transportes ou transportes já reservados.

4.1.3 Considerações de utilização

Existem algumas particularidades relativas às condições de utilização do *patientmove* que, de algum modo, podem moldar o seu desenvolvimento. Pretende-se analisar aspetos relacionados com os dispositivos onde a aplicação será executada, com a forma como o utilizador irá interagir com a aplicação ou com a área de visualização que a aplicação deverá ter durante a utilização.

Em primeiro lugar, é necessário perceber em que tipo de dispositivos o *patientmove* irá ser utilizado. A utilização básica do *patientmove* será realizada em plataformas Windows, Mac OS X ou Linux e a interface com o utilizador deverá estar disponível em locais estratégicos do serviço de urgência. A utilização transversal do *patientmove* nas diversas plataformas é assegurada pela sua natureza Web. Para ser executado no Browser é importante respeitar os *standards* do W3C (*World Wide Web Consortium*) e não utilizar ferramentas específicas de algum Browser ou plataforma.

A utilização do *patientmove* em dispositivos móveis também é interessante, uma vez que a possibilidade de consultar informação em qualquer local permitiria poupar tempo, pois não haveria necessidade de deslocações até aos terminais onde o *patientmove* está disponível. Os Browsers dos dispositivos móveis mais recentes começam a usufruir de quase todas as potencialidades dos Browsers normais, pelo que com alguns cuidados será expectável que seja possível desenvolver uma aplicação compatível com todos os dispositivos. No Anexo A são disponibilizadas tabelas de compatibilidade dos Browsers de dispositivos móveis com diversas tecnologias essências para as aplicações Web.

Os dispositivos nos quais o utilizador poderá interagir com o *patientmove* deverão ser utilizados apenas para efetuar essa tarefa, de forma a estarem constantemente disponíveis para uma utilização imediata em qualquer momento. O facto das operações realizados no *patientmove* não obrigarem a qualquer tipo de introdução de dados por parte do utilizador, faz com que a interação do utilizador com a aplicação seja baseada na escolha de opções que são apresentadas em conjunto com informação pertinente. Tendo em consideração os dois tópicos referidos neste parágrafo é sensato realizar todos os esforços para que o *patientmove* seja uma aplicação otimizada para ecrãs tácteis e para um funcionamento em ecrã completo.

Para responder à necessidade de funcionamento em dispositivos com resoluções de ecrã bastante diferentes e para que seja possível o funcionamento em ecrã completo para qualquer resolução, o *patientmove* terá de possuir a capacidade de se adaptar à resolução de ecrã de cada dispositivo onde é executado.

4.1.4 Utilizador alvo

O utilizador comum do *patientmove* são os profissionais responsáveis por realizar os transportes de pacientes no interior do serviço de urgência. Dependendo da organização do serviço de urgência, esta função tanto pode ser desempenhada por maqueiros ou por auxiliares de saúde.

Cada operação de iniciar transporte requer a escolha de um utilizador que ficará associado à execução dessa tarefa. Para definir os utilizadores disponíveis, a solução mais adequada será recorrer a informação disponível na base de dados, de forma a determinar os trabalhadores que realizam esta tarefa e que se encontram de serviço. Em relação à operação de finalizar transporte, a escolha do utilizador poderá ajudar a filtrar os transportes iniciados, de forma a listar apenas os transportes reservados por esse utilizador.

Uma vez que a informação gerida pelo *patientmove* não é confidencial nem sensível em qualquer sentido, a autenticação de utilizadores pode ser excluída, permitindo a mudança de utilizador num simples clique para seleção de um utilizador diferente. Esta abordagem permite ir ao encontro das necessidades da aplicação, aumentando a sua eficiência e facilidade de utilização.

4.1.5 Informação relevante para descrever o paciente e o transporte

Para efetuar o transporte de um paciente, o profissional em questão tem que tomar conhecimento de alguns dados acerca desse paciente. Para determinar qual a informação relativa ao paciente, que é relevante para o transporte, o ideal é contactar diretamente com os profissionais que exercem essas funções no seu dia a dia. Apesar de não ter sido efetuado um contacto direto, toda a informação necessária foi transmitida pelo orientador desta dissertação que, para além da vasta experiencia na área da saúde, tem contato direto com os profissionais em questão.

Depois de uma análise cuidada das necessidades existentes, a seguinte informação foi considerada pertinente para descrever a situação de um paciente: nome; idade; sexo;

resultado da triagem (traduzido por uma cor ou por um número de 1 a 5); tipo de locomoção – Cadeira de rodas, maca ou pedestre; e existência de acompanhante.

Para além da informação do paciente, existe informação fornecida pelo próprio pedido de transporte que é essencial para descrever as características de um transporte de um paciente: local de origem; local de destino; hora do pedido (ou o tempo que passou desde esse momento).

Com os dados apresentados, qualquer profissional responsável por realizar o transporte de pacientes tem à sua disposição toda a informação necessária para realizar um transporte eficaz e sensível às especificidades do paciente em questão. A informação disponível permite definir o percurso através dos dados do pedido (origem e destino), identificar o paciente pelos seus dados pessoais (nome, idade, sexo, locomoção) e avaliar a urgência do transporte (prioridade de triagem em conjunção com o tempo de espera).

4.1.6 Gestão de prioridades

Um dos principais objetivos do *patientmove* é organizar os transportes em espera, avaliando a conjugação das características que determinam a prioridade que deve ser atribuída a cada um transporte. Com a construção de um modelo de avaliação da urgência de cada transporte, é possível apresentar os transportes ao utilizador sob a forma de uma lista ordenada por urgência.

As características relevantes para determinar a prioridade de um transporte são apenas duas: a primeira e mais importante é a cor resultante da triagem do paciente, isto porque o nível de prioridade resultante da triagem é considerado em todos os passos de atendimento ao paciente e não apenas para triar a entrada do paciente para uma primeira observação; a segunda característica é o tempo de espera que o pedido de transporte já acumulou, sendo importante para distinguir transportes com a mesma prioridade de triagem e também para acautelar casos de espera excessiva dos transportes com prioridade de triagem mais baixa.

A prioridade associada a um transporte não deve ser considerada um critério inflexível, uma vez que impossibilitaria a otimização do percurso do profissional que efetua os

transportes. A otimização do percurso é importante porque leva a uma poupança de tempo nas deslocações do profissional até ao local onde se encontra o próximo paciente. Assim, de uma forma geral, as tarefas prioritárias serão realizadas mais cedo em conjugação com critérios proximidade.

4.1.7 Especificações técnicas

No Capítulo 3 é realizada a análise de todas as tecnologias relevantes para o desenvolvimento do *patientmove*, assim como a comparação com tecnologias concorrentes que se poderiam perspetivar como boas alternativas. De acordo com as ilações retiradas nesse capítulo, de seguida são apresentadas as características técnicas determinadas para o desenvolvimento do *patientmove*.

O *patientmove* será uma aplicação Web desenvolvida com recurso às potencialidades da *.NET framework* e mais especificamente ao conjunto de ferramentas ASP.NET. Assim, será obtida uma aplicação ASP.NET que deverá ser publicada num servidor IIS. Para acrescentar uma maior dinâmica na interação do utilizador com a aplicação, utilizam-se funcionalidades disponibilizadas pela plataforma ASP.NET AJAX. Em termos visuais, todos os pormenores são tratados através da manipulação da etiquetas HTML e das suas propriedades de estilo (CSS).

Em termos de fonte de dados, é utilizada uma base de dados relacional Oracle, uma vez que um dos objetivos passa por garantir que a aplicação Web tenha em consideração a otimização do código de acesso a dados para o SGBD Oracle. Esta necessidade de utilizar produtos Oracle deve-se ao facto de já existir uma base de dados Oracle em funcionamento, que contém informação utilizada pelo *patientmove*. Esta informação é atualizada por diversas aplicações integradas no sistema de informação da instituição de saúde em causa, pelo que o *patientmove* passa a ser mais uma aplicação a contribuir para a melhoria do sistema global. Assim sendo, já existe um modelo relacional a respeitar de forma a garantir a conformidade com a informação registada pelas restantes aplicações. No entanto, poderão ser promovidas algumas alterações no modelo relacional da base de dados, de forma a que este englobe toda a informação necessária para o correto funcionamento do *patientmove*. Na Secção 4.2.2, o modelo relacional da

base de dados é descrito de uma forma pormenorizada.

4.2 Planeamento

Nesta Secção serão apresentados os aspetos relacionados com o design do *patientmove*, assim como o sistema em que este funcionará. Toda a informação presenta nesta Secção é importante para compreender o trabalho de planeamento que foi necessário para desenvolver o *patientmove* e alcançar os resultados evidenciados nas secções seguintes.

Nesta fase é necessário refletir acerca das funcionalidades que melhor representam os objetivos da aplicação e, consequentemente, optar por uma interface adequada para essas funcionalidades. Outro aspeto relevante é projetar a forma como a aplicação comunica com os restantes elementos do sistema e também as caraterísticas desses elementos que podem ser importantes para o funcionamento do *patientmove*.

4.2.1 Arquitetura

O *patientmove* é uma aplicação Web desenvolvida para funcionar num sistema distribuído com uma arquitetura multicamada (*multi-tier*) constituída por três camadas: a camada de apresentação (Browser do lado do cliente); a camada aplicacional (servidor IIS que aloja a aplicação Web); e a camada de armazenamento de dados (servidor da base de dados).

A arquitetura representada na Figura 4.3 indica os componentes presentes em cada nodo do sistema e também a método de comunicação entre cada os nodos:

- O nodo da camada de apresentação é aquele que funciona sempre como cliente, efetuando os seus pedidos ao servidor da camada aplicacional. Normalmente, o Browser é a aplicação que funciona do lado cliente e que permite a interação do utilizador com o sistema, efetuando os pedidos necessários e apresentando ao utilizador a resposta recebida. A comunicação é efetuada através de pedidos HTTP normais ou através de *XMLHttpRequest* para atualizar parcialmente a página.

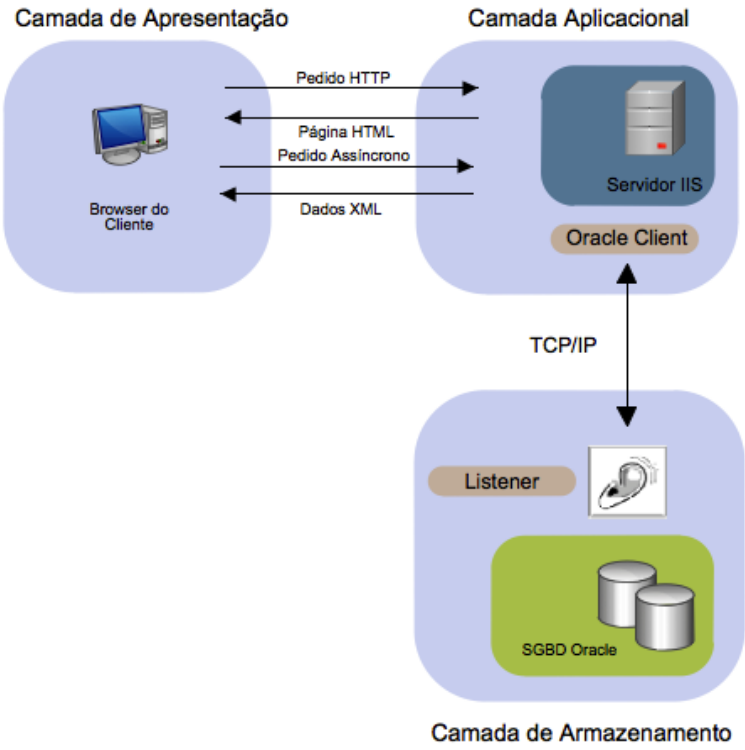


Figura 4.3: Arquitetura multicamada proposta para o *patientmove*.

- Numa arquitetura de três camadas, o nodo da camada aplicacional funciona simultaneamente como servidor e cliente. Para funcionar como servidor, este nodo tem um servidor IIS que recebe os pedidos do cliente e que trata de todas as funcionalidade relacionadas com a execução de aplicações ASP.NET. O funcionamento como cliente surge da necessidade de comunicar com o servidor da base de dados, de forma a obter ou armazenar informação importante para as aplicações ASP.NET. A comunicação entre camada aplicacional e camada de armazenamento de dados é realizado em cima do protocolo TCP e utiliza as especificações do SGBD Oracle.

- Na camada de armazenamento de dados encontra-se o servidor da base de dados. Existem vários componentes do SGBD Oracle que funcionam continuamente neste servidor para facilitar a interação com os serviços disponíveis (cada serviço pode representar uma base de dados independente). O componente responsável por receber os pedidos e encaminhá-los para o serviço correto é o *listener*. De-

pois do *listener* examinar devidamente o conteúdo do pedido, a comunicação é estabelecida de forma direta entre o cliente (camada aplicacional) e o serviço da base de dados requisitado.

4.2.2 Modelo Relacional

O modelo relacional do *patientmove* é, em grande parte, limitado pelo modelo relacional que já se encontrava aplicado na base de dados para satisfazer as necessidades de armazenamento de dados das restantes aplicações. Com o desenvolvimento do *patientmove*, passou a existir uma nova tarefa a ser desempenhada por uma aplicação informática (transporte de pacientes no serviço de urgência). Assim, surgiu a necessidade de preparar o modelo relacional para o registo de nova informação que descreva as características de um transporte.

A Figura 4.4 ilustra a parte do modelo relacional que é relevante para o funcionamento do *patientmove*, isto depois de efetuadas as alterações necessárias ao modelo original. Optou-se por não apresentar o modelo relacional completo da base de dados que gere a informação do serviço de urgência, uma vez que é bastante complexo e apenas acrescentaria informação que é utilizada para outras tarefas e não para o transporte de pacientes.

A tabela onde é gerida toda a informação relacionada com o transporte de pacientes (`ED_TRANSPORTES_IN`) inclui informação específica desta tarefa (colunas normais), assim como referências a outras tabelas que completam a descrição do transporte com informação já existente (colunas que funcionam como chave estrangeira).

Quando é realizado um pedido de transporte por um profissional de saúde, a aplicação utilizada para esse efeito gera uma nova entrada, na tabela de gestão dos transportes (`ED_TRANSPORTES_IN`), com os dados necessários para descrever o pedido. A referência do transporte (`ref_transporte`), que é a chave primária da tabela, é gerada automaticamente no momento de inserção de uma nova entrada. Na operação de inserção de um novo transporte são fornecidos, como chaves estrangeiras, três dados essenciais para descrever o transporte:

- O primeiro é o código da sala de origem (`cod_sala_ori`), ou seja, a localização

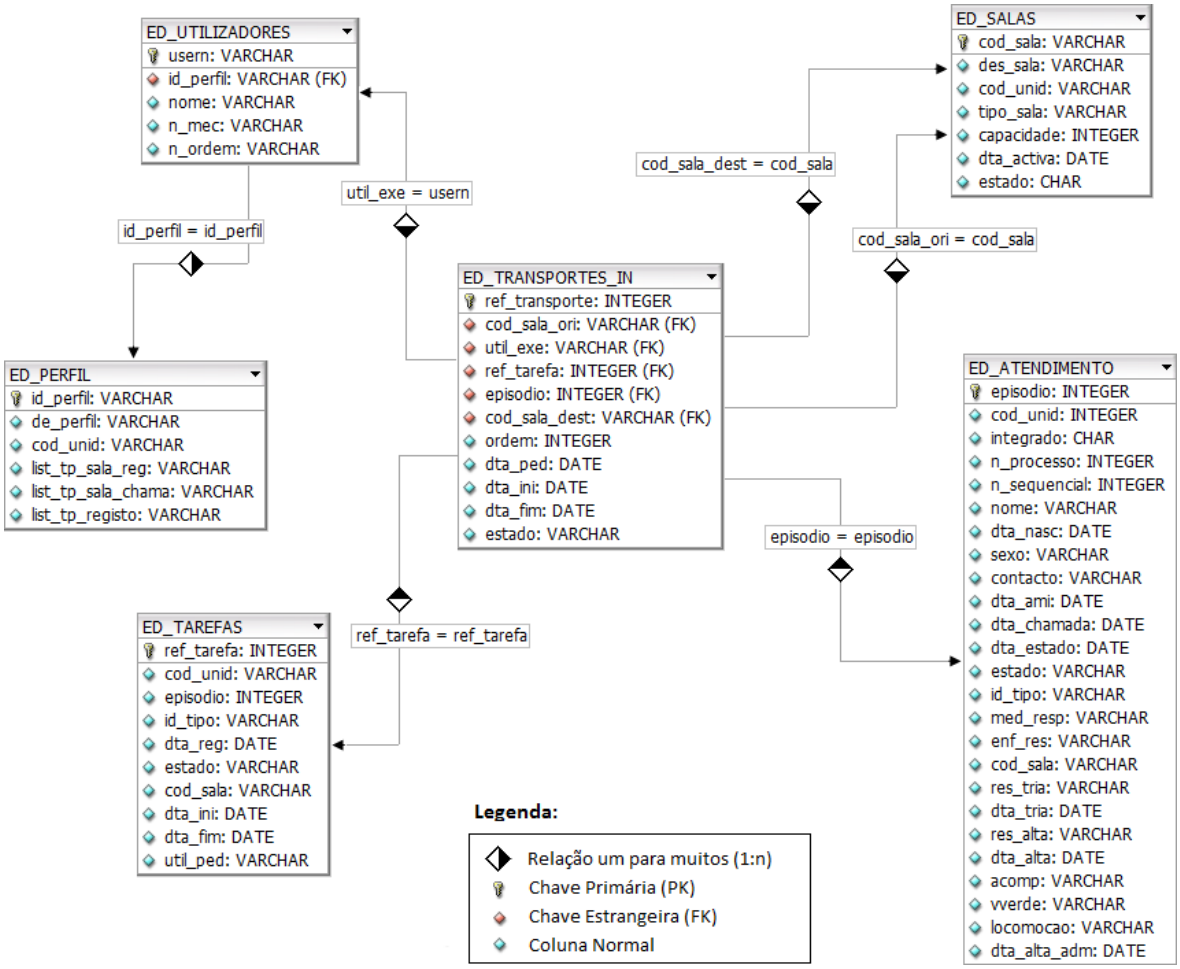


Figura 4.4: Modelo relacional do *patientmove*.

do paciente antes do transporte. Esta chave estrangeira estabelece uma relação de um para muitos (1:n) com a tabela que contém a informação das salas do serviço de urgência (`COD_SALAS`).

- O segundo é o código da sala de destino (`cod_sala_dest`), ou seja, a localização desejada para o paciente após o transporte. Esta chave estrangeira também estabelece uma relação de um para muitos (1:n) com a tabela que contém a informação das salas (`COD_SALAS`).

- O terceiro é o código que identifica o episódio de atendimento ao paciente no serviço de urgência (`episodio`). Esta indicação do episódio funciona como chave estrangeira, estabelecendo uma relação de uma para muitos (1:n) com a tabela que contém toda a informação registada acerca do paciente neste episódio de atendimento (`ED_ATENDIMENTO`)

Nos casos em que o transporte faz parte de uma tarefa mais abrangente, existe uma coluna da tabela de transportes (`ED_TRANSPORTES_IN`) onde é possível indicar a referência da tarefa (`ref_tarefa`). Esta coluna funciona como chave estrangeira, estabelecendo uma relação de um para muitos (1:n) com a tabela que contém a informação das tarefas gerais (`ED_TAREFAS`).

A data e hora em que o pedido é realizado são registadas numa única coluna (`dta_ped`), sendo importante para determinar o tempo de espera de cada transporte. O estado em que o transporte se encontra é também um dado muito importante, uma vez que indica ao *patientmove* a situação em que cada transporte se encontra (`estado`): em espera (0); iniciado (1); terminado (2); ou abortado (9). No momento do pedido, o transporte encontra-se em estado de espera (0).

Existe apenas mais um detalhe que pode ser introduzido ao realizar o pedido. Tem especial relevância quando é realizado um pedido simultâneo de vários transportes para o mesmo episódio de urgência. Nesta situação, é criada uma sequência de deslocações que o paciente deve realizar dentro da urgência, numa ordem específica. Assim é atribuída uma ordenação aos transportes referentes ao mesmo episódio, ficando essa informação guardada numa coluna específica para o efeito (`ordem`). Caso não existam transportes em sequência, o preenchimento deste campo é irrelevante.

As restantes colunas dizem respeito a informação que vai sendo adquirida com a utilização do *patientmove*, nunca através da solicitação de dados ao utilizador, mas sim pela recolha de informação subjacente à utilização da aplicação.

Na operação de iniciar transporte é introduzida a data e hora na coluna que regista a data de início de cada transporte (`dta_ini`). O outro valor introduzido indica o nome do utilizador que iniciou o transporte (`util_exe`). Esta coluna é uma chave estrangeira que estabelece uma relação de um para muitos (1:n) com a tabela que contém a informação dos utilizadores.

A operação de finalização de um transporte leva ao preenchimento da coluna relativo à data e hora (`dta_fim`) em que o transporte é dado como terminado.

As tabelas, referenciadas por alguma chave estrangeira da tabela de transportes (`ED_`

TRANSPORTES_IN), contêm informação importante para caracterizar os transportes. No entanto, essas mesmas tabelas são utilizadas para diversas finalidades, logo contêm informação que pode não ser relevante para o contexto do transporte de pacientes. De seguida são indicadas as colunas, de cada tabela, que são importantes para o transporte de pacientes:

- Na tabela que contém a informação do episódio de atendimento (ED_ATENDIMENTO), para além da chave primária (episodio), são importantes as informações que caracterizam o estado do paciente: nome (nome), data de nascimento (dta_nasc), género (sexo), resultado da triagem com valores numérico de 1 a 5 (res_tria), existência de acompanhante (acomp) e tipo de locomoção (locomocao).

- Da tabela que contém a informação das salas (ED_SALAS), apenas é utilizado para apresentação o campo que contém o nome da sala por extenso (des_sala). A relação é estabelecida através do código da sala (cod_sala), tratando-se da chave primária da tabela. Para verificar se cada sala pode ser utilizada, também é verificado se o código da unidade (cod_unid) corresponde à unidade de urgência em que o *patientmove* é utilizado, enquanto que o estado da sala (estado) deve ser 1 para garantir que esta está ativa.

- Da tabela que contém a informação que descreve os utilizadores do sistema (ED_UTILIZADORES), é utilizado o campo que contém o nome do utilizador (nome). A relação é estabelecida através do nome de registo (usern), tratando-se da chave primária da tabela. De forma a verificar se o utilizador desempenha a tarefa de transportar pacientes, a sua identificação de perfil deve ser específica para a tarefa de transporte (id_perfil = TRANS). Por último, através da relação da tabela de utilizadores (ED_UTILIZADORES) com a tabela de perfis (ED_PERFIL) é verificado se o utilizador trabalha na mesma unidade de urgência em que o *patientmove* está a operar.

- Quanto à tabela que contém a informação de tarefas gerais (ED_TAREFAS), a relação é realizada através da sua chave primária (ref_tarefa). Nenhum campo desta tabela é utilizado para caracterizar os transportes.

4.2.3 Funcionalidades a implementar

As funcionalidade básicas que o *patientmove* deve implementar já foram referidas na Secção 4.1.2. No entanto, é necessário especificar as funcionalidades que podem ser implementadas para permitir a realização de cada operação referida nessa Secção, ou seja, é necessário pensar que cada operação pode ser otimizada de diversas formas, sendo expectável que o *patientmove* ofereça as soluções necessárias para otimizar a escolha de transportes em situações diversas.

A operação de iniciar um transporte é a mais importante do *patientmove*, pelo que é aquela que apresenta mais opções para a sua realização. Isto significa que o utilizador poderá iniciar transporte recorrendo a diferentes opções, sendo que a distinção entre cada opção reside nos critérios de seleção e apresentação de transportes que são considerados mais importantes. Para cobrir todas as necessidades do utilizador, são disponibilizadas três opções distintas para iniciar transporte:

- **Listagem dos transportes ordenados por prioridade** – é disponibilizada uma lista de todos os transportes em espera, onde o primeiro transporte é o mais urgente e o último é o menos urgente.

- **Mapa com a indicação do número transportes em espera em cada sala** – o utilizador tem um mapa onde pode ter uma perceção imediata dos pacientes que se encontram mais próximos. Cada sala, que tenha transportes em espera, tem a cor de fundo igual à cor de triagem do paciente mais urgente. Assim, apenas com um simples olhar é possível perceber a localização dos transportes a realizar, tendo também em conta onde se encontram os mais urgentes. Ao escolher uma sala, são apresentados os transportes em espera para essa sala, estando ordenados por prioridade.

- **Lista de transportes que ficam no percurso de um transporte escolhido** – recorrendo ao transporte escolhido (seja pelo utilizador, ou pelo sistema), é criada uma lista de transportes que têm o seu inicio e destino próximos do percurso do transporte original. Esta lista especial permite que o utilizador escolha os transportes que pretende associar à escolha inicial, iniciando todos os transportes escolhidos com apenas um clique. Esta opção permite poupar o tempo de

uma nova consulta dos dados e escolha de transporte, permitindo também que o profissional ajude dois pacientes em simultâneo caso estes apenas necessitem de orientação.

Em qualquer uma das opções anteriores, sempre que é necessário ordenar transportes por prioridade, o critério mais importante de é o resultado da triagem, sendo aplicado como critério secundário o tempo de espera pelo transporte.

A operação de terminar transporte, devido à sua simplicidade, não exige várias opções para a sua realização. Quando um utilizador pretende terminar um transporte, apenas necessita de indicar o seu nome de utilizador e é apresentada a lista de transportes iniciados por si, ordenada por tempo passado desde o início do transporte. Adicionalmente, o utilizador pode consultar a lista de transportes iniciados por todos os utilizadores.

Seja a iniciar transporte ou a terminar transporte, existe sempre a opção de abortar o transporte, uma vez que pode surgir um contratempo em qualquer momento que leve à anulação do transporte, esteja este em estado de espera ou iniciado.

A última funcionalidade a implementar é a disponibilização da opção de cancelar o início de um transporte, ou seja, um transporte que tenha sido iniciado/reservado pode voltar ao estado de espera se assim for necessário.

4.2.4 Interface

O *patientmove* é uma aplicação em que se pretende que todos os processos sejam simplificados ao máximo, tanto a nível de menus, como a nível de painéis de apresentação de dados. No seguimento desta filosofia, a interface pensada foi, desde o inicio, bastante simples e intuitiva para qualquer utilizador. Assim, é importante manter a mesma estrutura base da página em qualquer situação, alterando apenas painel de dados a visualizar de acordo com a operação a realizar.

Para iniciar a construção da interface do *patientmove* foi necessário dividir a sua página em blocos básicos que representem as diferentes zonas de atividade da página, cada uma com o seu propósito. A Figura 4.5 ilustra o esquema de blocos básicos, pensado para o *patientmove*:

- No topo da página (1) é colocado um cabeçalho que tem como objetivo incluir informação genérica acerca da aplicação, como nome do hospital, logótipo do hospital, nome da aplicação, etc.
- No canto superior direito (2) existe um pequeno bloco que permite ao utilizador identificar-se de forma rápida.
- Do lado direito da página (4) fica uma barra lateral com um menu que inclui todas as funcionalidades que podem ser utilizadas no *patientmove*.
- O restante espaço da página (3) fica reservado para o conteúdo de cada opção que o utilizador escolhe no menu da direita. A informação disponível neste painel de conteúdo é o ponto mais importante da aplicação, uma vez que será disponibilizada toda a informação dos transportes, tanto na opção de iniciar transporte como na opção de terminar transporte.

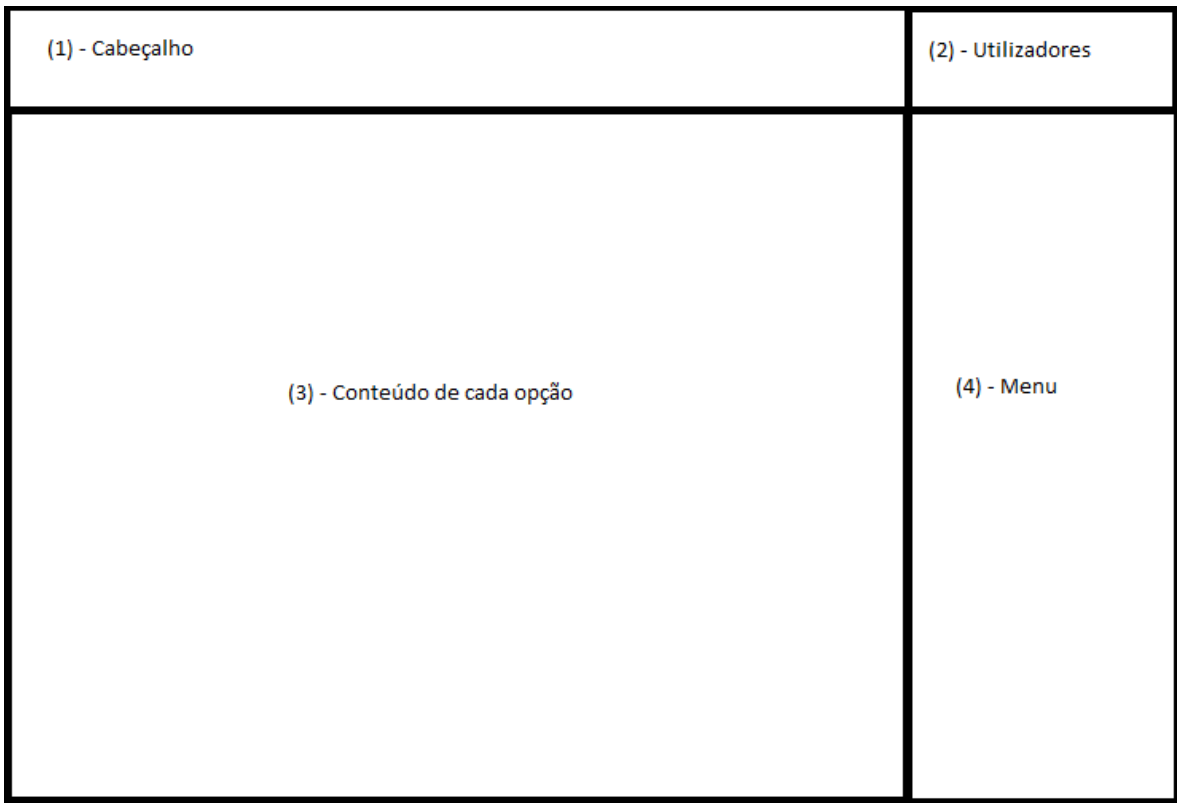


Figura 4.5: Estrutura de blocos básicos do *patientmove*.

Esta organização dos blocos provou ser bastante produtiva, pelo que foi mantida em

toda a fase de design.

Em relação aos blocos 1, 2 e 4 não existem pormenores específicos de design que necessitem de um estudo mais aprofundado, uma vez que a informação presente em cada um é simples e fácil de representar. No entanto, no bloco 3, é expectável que seja apresentada bastante informação diferente, uma vez que todos os transportes serão apresentados neste painel, podendo a sua representação assumir diversas formas em função da opção de visualização. De uma forma simplificada, existem duas formas de visualizar a informação relativa aos transportes: através de uma lista e através de uma mapa.

A visualização em forma de lista é a mais utilizada no *patientmove*. Esta pode ser utilizada para: visualizar a lista completa de transportes a iniciar, visualizar a lista completa de transportes a terminar; visualizar a lista de transportes que podem ser iniciados em grupo.

Sabendo-se que a lista de transportes é utilizada em três situações, é importante que a interface seja otimizada no sentido de facilitar a interação do utilizador com o *patientmove*, ou seja, a interface deve permitir ao utilizador assimilar toda a informação acerca de um transporte no mínimo de tempo possível, facilitando a comparação entre transportes e a decisão final de escolher o transporte a iniciar ou terminar.

A primeira interface pensada para a listagem de transportes no painel de conteúdo do *patientmove* foi a utilização de uma tabela, em que cada linha inclui toda a informação de um transporte. A seleção do transporte escolhido seria realizada através de um botão presente à esquerda de cada linha, sendo que as linhas tinham que estar suficientemente afastadas para permitir a correta utilização de interfaces tácteis. A Figura 4.6 representa de uma forma simplifica a estrutura idealizada. Esta abordagem foi excluída por ser considerada muito rígida na sua estrutura. Isto deve-se ao facto de não permitir a livre formatação de cada elemento, uma vez que obriga à colocação de toda a informação de um transporte na mesma linha. Outro aspeto negativo era a necessidade de incluir um botão extra para selecionar o transporte. O ideal seria clicar na própria linha para o respetivo item ser selecionado.

Tendo em conta as fragilidades que foram apontadas à interface em forma de tabela,

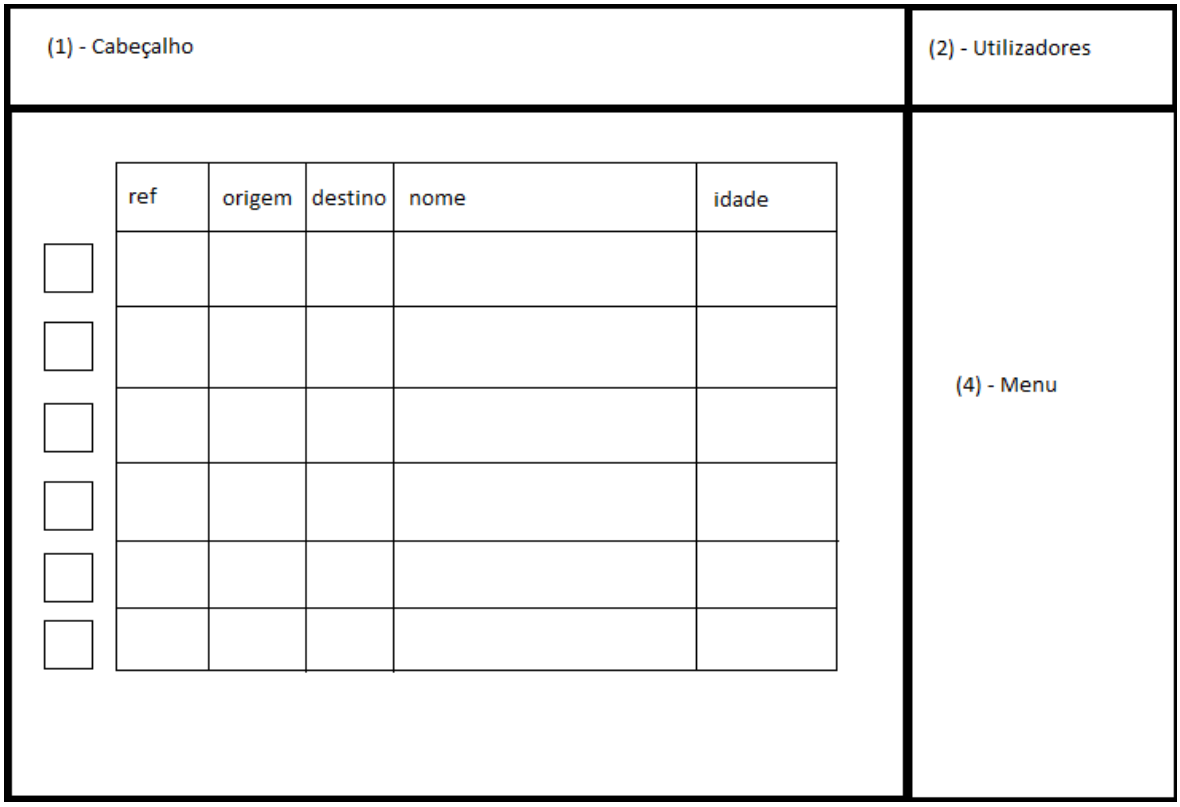


Figura 4.6: Interface com listagem de transportes em tabela.

avançou-se para a pesquisa de uma nova abordagem à listagem de elementos, que permitisse uma maior liberdade na disposição da informação de cada transporte e que continuasse a permitir a disposição de vários transportes por página de uma forma organizada.

A solução encontrada para satisfazer as necessidades referidas anteriormente foi a utilização de uma lista, em que cada elemento tem o seu próprio painel com a informação organizada da forma mais indicada para descrever o transporte. Os elementos da lista ficam separados por algum espaço de forma a serem facilmente distinguidos pelo utilizador. A Figura 4.7 representa esta interface de uma forma geral. Para selecionar um transporte, o utilizador terá apenas de clicar dentro do painel que engloba a informação do transporte pretendido. Assim, esta abordagem permite uma utilização muito mais intuitiva e rápida, indo de encontro aos objetivos propostos para o *patientmove*.

Uma questão importante, relacionada com esta interface em lista, é a organização da informação no interior do painel individual de cada transporte, uma vez que não seria indicado representar toda a informação em simples linhas de texto que requerem uma

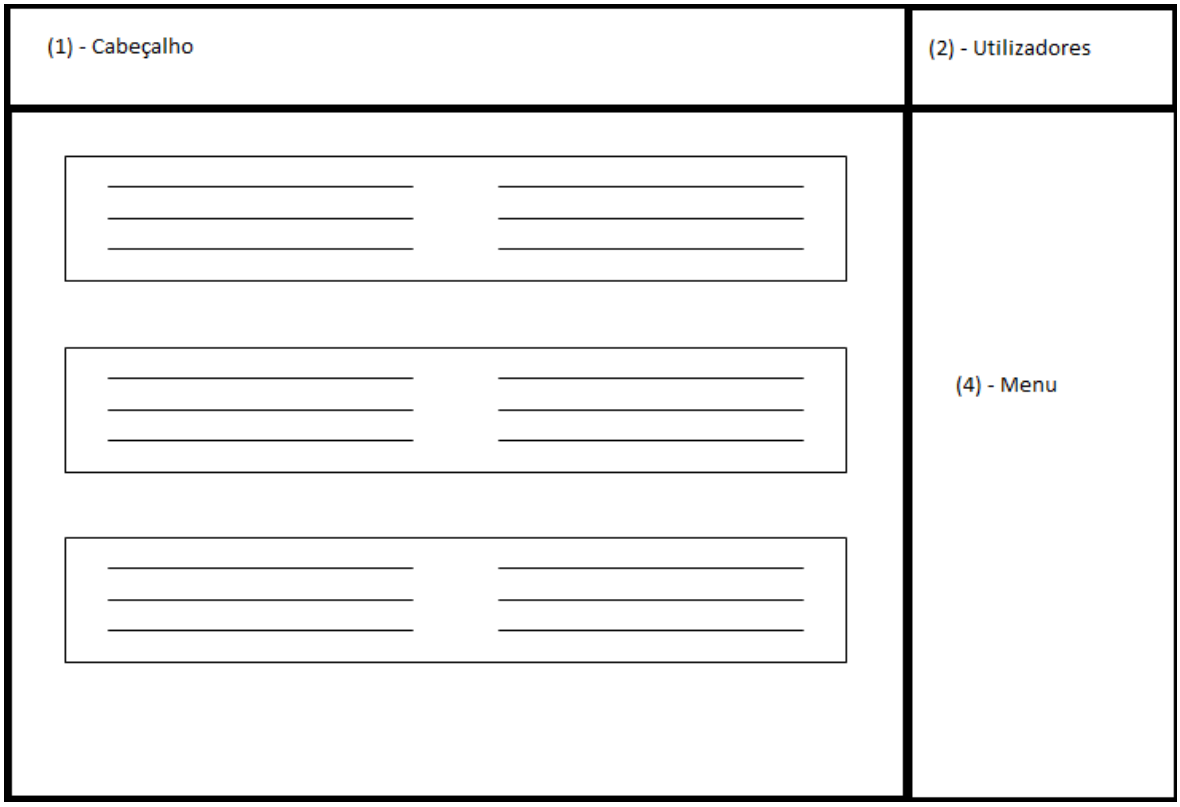


Figura 4.7: Interface com lista de transportes em que cada elemento singular representa um transporte.

leitura atenta por parte do utilizador. Assim, o raciocínio seguido para organizar a informação dentro do painel assenta nos pressupostos de que existe informação mais importante do que outra e que cada dado tem uma forma ideal de ser representado em função do seu significado. Desta forma, foram tidos em conta os seguintes aspetos para a informação do transporte a introduzir no painel:

- **Resultado de triagem** – a prioridade de triagem associada a cada utilizador pode ser expressa por uma cor. Esta é, sem dúvida, a forma mais rápida de identificar o resultado de triagem. Para ser obtida uma noção imediata da prioridade de cada paciente, a cor de triagem é utilizada para definir a cor de fundo do painel do transporte.

- **Origem e destino do transporte** – Estes dados representam os nomes das salas de origem e destino do transporte. A relação entre as duas é óbvia e a perceção do utilizador é imediata se o nome sala de origem for colocado numa caixa de texto e existir uma seta a apontar para a outra caixa de texto com o

nome da sala de destino.

- **Locomoção e Acompanhante** – Estes dados têm uma gama limitada de valores, sendo que no caso da existência de acompanhante há dois valores (sim/nao) e no tipo de locomoção há três (cadeira de rodas/maca/a pé). Esta situação é ótima para a substituição do texto por imagens que representem cada situação de uma forma inequívoca.

- **Nome, idade e tempo de espera** – Estes dados são representados pelo texto normal. O tempo de espera poderia ser representado num relógio analógico mas a sua leitura seria mais complicada.

- **Sexo** – Este dado também apresenta apenas duas opções (Masculino ou Feminino) pelo que a sua substituição por uma imagem é o ideal. No caso de não existir sexo definido, não surge nenhuma imagem.

- **Ocupação e Capacidade da sala de destino** – Estes dados acerca da sala de destino podem ser preciosos para evitar transportes para salas que estejam sobrelotadas. A forma mais indica para visualizar esta informação é através da utilização do formato ocupação/capacidade. Quando a ocupação for maior ou igual que a capacidade, a caixa de texto que contém esta informação passa a ter o fundo com uma cor encarnada.

A Figura 4.8 ilustra o painel de um transporte tal como projetado pela análise anterior. Trata-se de um exemplo da interface final ilustrando um transporte com origem na "Sala de Gessos" e destino na sala "Sala 2", envolvendo a paciente "Florentina Joaquina" de 50 anos de idade e sexo feminino. Esta paciente tem acompanhante e move-se com auxilio de uma cadeira de rodas. O seu resultado de triagem é o nível amarelo e encontra-se à espera do transporte há 21 horas e 40 segundos. A sala de destino tem capacidade para 1 paciente, não se encontrando, de momento, nenhum paciente no seu interior. Como se pode comprovar por esta análise toda esta informação pode ser interpretada de uma forma rápida sem a leitura exaustiva de texto.

Como já foi referido, para além da lista, existe outra forma de abordar a escolha de um transporte: a visualização de um mapa. Como não podia deixar de ser, a interface

Figura 4.8: Painel que engloba a informação relativa a um transporte, fazendo parte da visualização em lista.

desta opção consiste em colocar informação sobreposta num mapa que representa a localização das salas no serviço de urgência. A Figura 4.9 mostra uma perspetiva da interface, já com a imagem que contém as salas do serviço de urgência devidamente organizadas. A imagem das salas foi produzida no *Google SketchUp* e é uma reprodução exata da planta fornecida em papel.

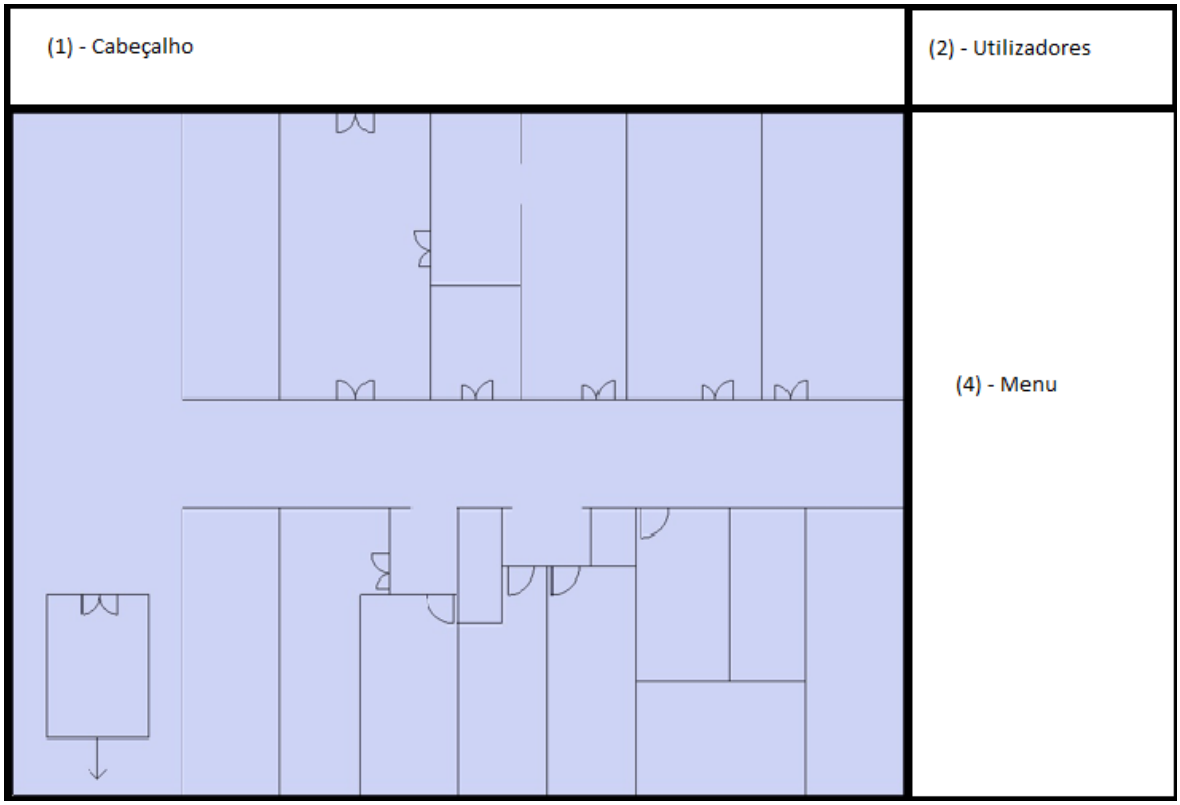


Figura 4.9: Interface com mapa das salas existentes no serviço de urgência.

A imagem com a planta do serviço de urgência funciona como um fundo no painel de conteúdo do *patientmove*. Assim, a interface pode ser construída em cima desta imagem, criando painéis que ocupem a mesma área das salas em utilização. Cada um dos painéis irá representar uma sala e, deste modo, pode incluir alguma informação que

ajude o utilizador a decidir a sala que pretende escolher (visualizando posteriormente uma lista de transportes a iniciar nessa sala). Existem dois dados bastante importantes:

- O número de transportes que se encontram em espera para serem iniciados e que têm inicio na sala em questão. Esta informação é representada na forma de texto.
- O resultado de triagem dos paciente associados aos transportes dessa sala. De entre os transporte dessa sala, apenas é utilizada a cor de triagem mais urgente para definir a cor de fundo painel da sala em questão.

Estes painéis, que representam as salas, possuem a funcionalidade de responderem ao clique no seu interior, pelo que a interação do utilizador com o mapa fica bastante simplificada.

4.3 Implementação e Descrição

Nesta Secção é descrita a fase de implementação, na qual todas as ideias retiradas da fase de análise de requisitos e da fase de planeamento e design da aplicação são colocadas em prática. Isto equivale a dizer que todos os pensamentos são colocados em código, de forma a obter uma aplicação que esteja de acordo com o planeado. A explicação das soluções encontradas para implementar cada uma das funcionalidades, assim como a descrição do resultado final obtido, são apresentadas nesta Secção.

4.3.1 Funcionalidades gerais

Existem algumas funcionalidades do *patientmove* que são utilizadas durante a realização das mais variadas ações. Assim, para evitar a referência repetitiva à sua implementação, é realizada uma abordagem geral às funcionalidades que têm uma utilização constante no *patientmove*.

4.3.1.1 Acesso a dados

O acesso a dados consiste na utilização de classes que forneçam uma interface de acesso a uma base de dados remota. O *data provider* ODP.NET foi utilizado para

disponibilizar todos os objetos que permitem à aplicação ASP.NET interagir com a base de dados Oracle.

No *patientmove*, toda a lógica de acesso a dados está incluída numa classe que foi criada apenas para lidar com essa função. Na Figura 4.10 são apresentados os métodos desta classe, que foi designada "MeuObjectoTrans". Cada método público desta classe realiza uma operação distinta na base de dados, podendo ser evocado diretamente no código da aplicação ou através do processo de *data binding* de um controlo. Podem ser encontradas mais informações acerca de *data binding* na Secção 3.3.6.

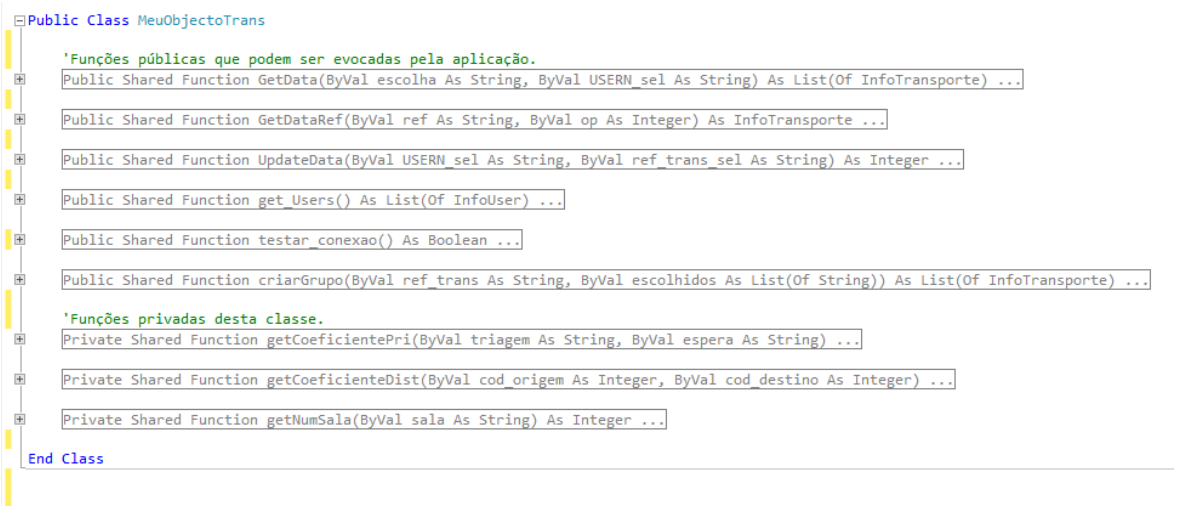


Figura 4.10: Métodos da classe "MeuObjectoTrans", utilizada como *ObjectDataSource* em diversos controlos no *patientmove*.

4.3.1.2 Painel de utilizadores

O painel de utilizadores encontra-se no canto superior direito do *patientmove*, tal como ilustrado na Figura 4.11. O painel consiste numa *combobox* que contém os nomes dos utilizadores disponíveis para efetuar transportes. Estes utilizadores são carregados através da informação existente na base de dados. Existe um utilizador "TODOS" que não pode iniciar transportes porque não se trata de um utilizador real registado no sistema. Trata-se de um utilizador *default* que vem automaticamente escolhido quando a página é carregada mas que apenas pode ser utilizado para visualizar os transportes iniciados por todos os utilizadores.

No painel de utilizadores apenas existe mais um controlo. Trata-se de uma imagem representativa de um utilizador, que funciona como botão para atualizar a lista de utilizadores.

Figura 4.11: Aspeto geral do *patientmove*, no momento de escolher utilizador.

4.3.1.3 Lista de transportes

As listas de transportes são utilizadas sempre que é necessário mostrar vários transportes que podem ser escolhidos. Mesmo na escolha de transportes através do mapa, depois de escolhida a sala, é disponibilizada um lista com os transportes dessa sala. Devido a este papel chave no funcionamento do *patientmove*, a implementação da lista de transportes foi cuidadosamente estudada até chegar à solução apresentada, podendo ser observada por exemplo na Figura 4.11.

A solução encontrada para criar uma lista dinâmica com a informação estruturada foi a utilização do novo e poderoso controlo ASP.NET para a visualização de dados: o *ListView*. O acesso a dados é realizado através do processo de *data binding*, tal como explicado na Secção 4.3.1.1. Depois de receber os dados, a *ListView* utiliza os *templates* pré-definidos para repetir a criação de um novo elemento por cada registo recebido.

A característica que distingue a *ListView* dos outros controlos é a sua extrema flexibilidade na apresentação dos dados. A definição de *templates* pré-definidos é importante, mas os resultados mais surpreendentes surgem da adaptação destes *templates* aos dados de cada elemento. Esta personalização de *templates* é possível através do evento *ItemDataBound* da *ListView*, uma vez que este permite aplicar o *template* ao novo elemento de uma forma personalizada, alterando determinadas características em função da informação do elemento. É através deste evento que, em função da informação recebida da base de dados, cada elemento apresenta imagens diferentes na sua apresentação em lista (Figura 4.11).

Dependendo dos dados retornados da base de dados, os elementos criados pela *ListView* podem, ou não, ser em número reduzido. Assim, de forma a encaixar os elementos na área de visualização disponível foi associado a todas as *ListViews* um controlo de paginação, designado *DataPager*. Este controlo define um número máximo de elementos por página e os elementos restantes são colocados em novas páginas, estando automaticamente disponível uma barra de paginação para permitir a navegação entre páginas da *ListView*.

4.3.1.4 Atualizações parciais e *Timer*

A tecnologia ASP.NET AJAX marca uma presença muito forte no funcionamento do *patientmove*. Todos os blocos básicos da interface, apresentados na Secção 4.2.4, podem ser atualizados sem necessidade de um *postback* completo da página. Isto acontece porque todos os blocos possuem o seu conteúdo no interior de um *UpdatePanel*. Assim, todas as ações do utilizador são tratadas como pedidos assíncronos e nunca se verificam atualizações completas da página. No entanto, isto não significa que os eventos desencadeados num bloco não possam provocar a atualização simultânea de um *UpdatePanel* de outro bloco. O caso mais lógico, em que esta situação ocorre, é na escolha de uma opção do menu, uma vez que tanto será atualizado o painel do menu com o painel do conteúdo.

Para além das atualizações promovidas pela interação do utilizador, existe um controlo designado *Timer* que efetua atualizações periódicas ao *UpdatePanel* presente no bloco do conteúdo. O *Timer* possui um atributo designado *Interval*, no qual é definido o

intervalo de tempo que separa as atualização promovidas por esta controlo. No caso do *patientmove* o intervalo de tempo foi definido como sendo 10 segundos (10000 milissegundos). Este intervalo permite que as atualizações não sejam demasiado frequentes, de forma a não incomodar o utilizador, e permite também que não sejam muito espaçadas, de forma a não induzir o utilizador em erro, uma vez que atualizações muito espaçadas podem levar a pensar que não existem transportes para efetuar, quando na realidade é a aplicação que está desatualizada.

4.3.1.5 Mensagens de confirmação

Sempre que é necessário efetuar uma operação de iniciar, terminar ou abortar um transporte, surge uma mensagem de confirmação com os dados do transporte que estamos prestes a alterar. Apesar de retirar alguma fluidez à utilização do *patientmove*, esta mensagem pode prevenir diversos erros associados a cliques indesejados e interpretações erradas que, sem esta confirmação, não poderiam ser revertidos. Uma vez mais, a interface disponibilizada é extremamente simples, estando direcionada para a rápida confirmação da informação, tal como é ilustrado na Figura 4.12 para a situação de um transporte a ser iniciado.

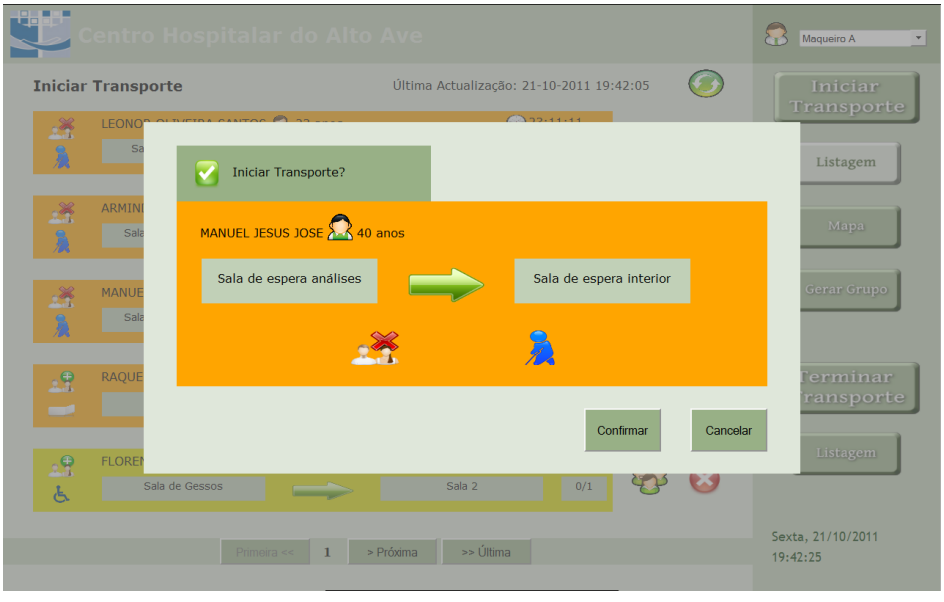


Figura 4.12: Mensagem de confirmação utilizada para as operações de iniciar, terminar e abortar transporte – Situação específica de um transporte a ser iniciado.

4.3.1.6 Mensagens de erro

As mensagens de erro podem ser bastante importantes para informar o utilizador que está a efetuar uma ação incorreta ou que existe alguma incorreção no sistema sobre a qual não tem controlo. A Figura 4.13 exemplifica uma mensagem de erro comum. Os erros normais que podem surgir durante a utilização do *patientmove*, passam por:

- Iniciar um transporte com o utilizador "TODOS" selecionado;
- Tentar alterar um transporte (iniciar, terminar, abortar ou devolver), quando este já foi alterado previamente noutro terminal;
- Conexão à base de dados sem resposta (Figura 4.13);
- Não existirem utilizadores disponíveis na base de dados;

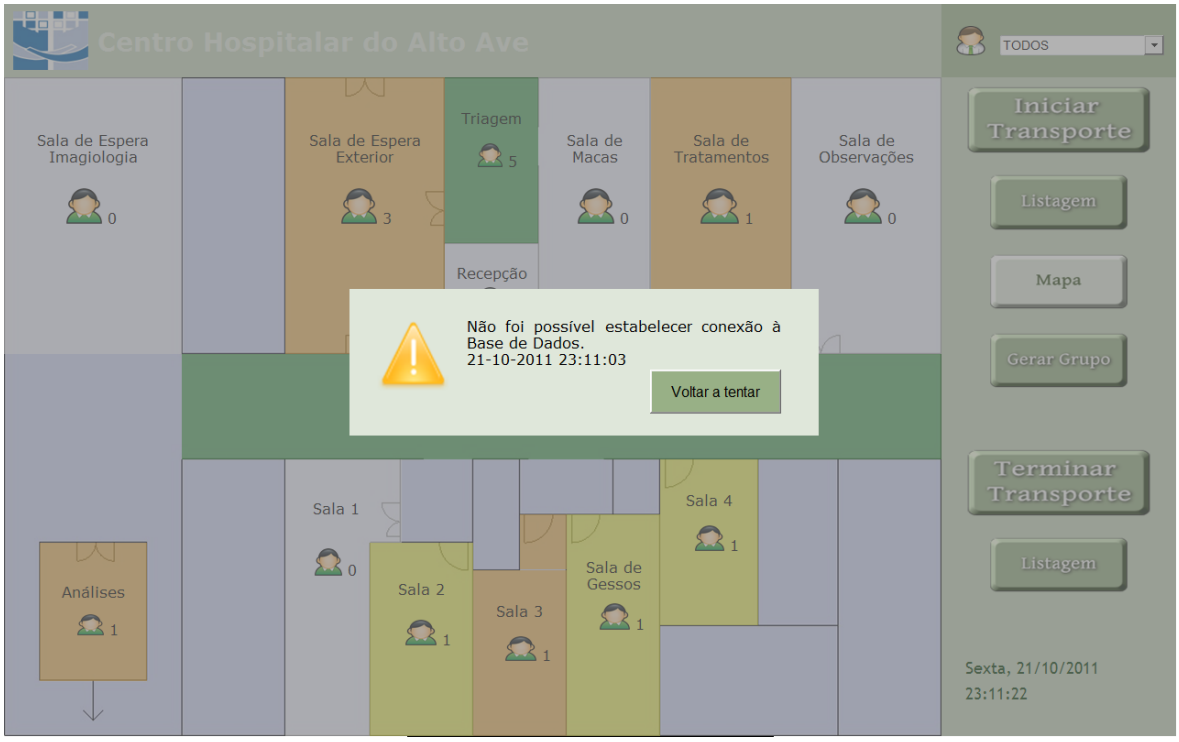


Figura 4.13: Mensagem de erro a informar o utilizador que existem problemas com a conexão à base de dados.

4.3.1.7 Ajuste dos elementos da página à resolução de ecrã

Na Secção 4.1.3, foram analisadas as condições de utilização do *patientmove*. A principal orientação retirada dessa análise foi a chamada de atenção para a necessidade de preparar a aplicação para funcionar em modo de ecrã inteiro em qualquer resolução. Deste modo, o *patientmove* foi desenvolvido com a forte convicção de criar uma aplicação dinâmica que se ajuste à resolução de ecrã do dispositivo de visualização utilizado.

De forma a garantir a variação de tamanho dos elementos presentes na página, todos estes elementos foram dimensionados com recurso a dimensões relativas (percentagem). Para além das dimensões, também o posicionamento foi determinado com recurso a deslocações medidas em percentagem. Os valores de percentagem utilizados em cada elemento têm como referência o tamanho do seu elemento ancestral. Como o elemento ancestral também está dimensionado e posicionado com recurso a percentagens, cria-se uma cadeia de referencias que leva até ao elemento *body*.

O elemento *body* é aquele que define diversos atributos gerais da página, entre os quais, a largura a e altura de exibição. Para definir os valores dos atributos largura e altura do elemento *body*, são invocadas funções Javascript que retornam um valor em pixéis da largura e da altura da resolução de ecrã utilizada no sistema anfitrião (Anexo B). Com a definição destes dois atributos, todos os elementos da página podem definir o seu tamanho através da conjugação da percentagem com o tamanho do elemento ancestral.

A resolução nativa do *patientmove* é 1280x800, ou seja, esta é resolução para a qual a sua interface está otimizada. No entanto, na versão final do *patientmove*, não é definida nenhuma resolução, sendo sempre utilizadas as funções Javascript. A adaptação à resolução que estiver a ser utilizada é realizada apenas quando a página é carregada pela primeira vez, não existindo qualquer diferença na restante utilização da aplicação.

Em termos de diferenças de visualização entre resoluções distintas, não pode ser apontado qualquer problema, com exceção para a diferença de proporção entre diferentes resoluções. Isto significa que é natural uma resolução com proporção do tipo 16:9 verificar um aspeto diferente de uma resolução 4:3. Na Figura 4.14 são ilustradas as

diferenças de visualização entre duas resoluções com proporções distintas: 1280x800 -> 16:10 (a) e 800:600 -> 4:3 (b).

Figura 4.14: Diferenças de visualização entre duas resoluções com proporções distintas: a) 1280x800 - 16:10; b) 800:600 - 4:3.

4.3.1.8 Ajuste do tamanho de letra à resolução de ecrã

A definição do tamanho da letra é realizada de uma forma idêntica à definição do tamanho dos elementos da página, ou seja, cada texto que surge na página tem o seu tamanho de letra definido em percentagem. A percentagem que define o tamanho da letra é relativa ao tamanho de letra utilizado no elemento ancestral. Esta cadeia de relações, leva novamente ao elemento *body* da página.

O elemento *body* permite definir o tamanho de letra a utilizar, estando por defeito definido a 100%, o que equivale a 16 pixéis. Uma vez mais, no momento em que a página carrega, é executada uma função Javascript para fazer variar o tamanho da letra em função da resolução do ecrã (Anexo B). O tamanho original de 100% é atribuído à resolução 1280x800. Para outras resoluções, é aplicada uma variação ao tamanho de letra equivalente à variação da resolução. Como existem duas dimensões para a resolução de ecrã (altura e largura), o tamanho de letra varia de acordo com a variação da dimensão limitativa, ou seja, de acordo com aquela que diminuir mais ou crescer menos. Esta escolha moderada permite afirmar que um texto que encaixa numa *label* à resolução de 1280x800, também encaixa nessa mesma *label* em qualquer outra resolução, seja a variação mais acentuada em termos de altura ou largura.

4.3.2 Iniciar transporte

A operação de iniciar transporte corresponde à escolha de um transporte que se encontra em espera, ficando o utilizador em questão responsável por efetuar esse transporte.

Nesta operação, de entre todos os transportes, são apresentados ao utilizador apenas os transportes que se encontrem em espera. Assim, o comando SQL para selecionar todos os transportes em espera apenas tem que respeitar esse critério (estado = 0).

Ao ser escolhido, o transporte passa a estar no estado iniciado (estado = 1), pelo que passará a fazer parte da lista de transportes a terminar. Ao mesmo tempo, o utilizador que fez a escolha fica associado à realização do transporte através do seu nome de utilizador (util_exe = usern). Assim, independentemente da visualização utilizada, a escolha de um transporte é sempre traduzida na execução do mesmo comando SQL

que realiza as alterações necessárias na base de dados.

4.3.2.1 Listagem

Nesta opção, todos os transportes em espera são apresentados numa lista, tal como ilustrado na Figura 4.15. A sua ordem respeita os critérios estabelecidos para a ordenação por prioridade, ou seja, surgem primeiro os transportes em que o paciente tem prioridade de triagem mais elevada, sendo esse critério representado pela cor do painel de cada transporte. No caso de prioridades de triagem iguais, surgem em primeiro lugar os transportes com um tempo de espera mais elevado.

Figura 4.15: Aspeto geral do *patientmove* na opção de listagem de transportes a iniciar.

A opção de iniciar transporte através de uma lista de transportes ordenada, permite ao utilizador realizar três ações distintas:

- **Reservar a realização de um transporte** – Esta ação corresponde à operação de iniciar transporte. Para escolher o transporte a iniciar, o utilizador apenas tem que clicar na área colorida que engloba a informação do transporte. Se o utilizador "TODOS" estiver selecionado, surge uma mensagem de erro, uma vez

que o transporte tem que ficar associado a um utilizador registado no sistema. Se estiver tudo bem com o pedido, surge uma mensagem de confirmação.

- **Iniciar criação de um grupo de transportes** – Esta ação leva o utilizador a um novo painel de conteúdo, onde pode formar um grupo de transportes para iniciar em conjunto com o transporte escolhido. Esta opção é selecionada ao clicar na imagem de grupo, que se encontra à direita do painel do transporte.

- **Eliminar um transporte** – Esta ação tem como finalidade abortar a realização de um transporte, ou seja, se por algum motivo já não é necessário realizar um transporte que tinha sido pedido, esta opção permite que o registo desse transporte passe do estado "em espera" (0) para o estado "abandonado" (9). Os transportes abandonados não voltam a ser geridos no *patientmove*, ficando o seu registo na base de dados apenas para efeitos de gestão. Esta opção é selecionada ao clicar na imagem de cancelar, que se encontra à direita do painel do transporte.

4.3.2.2 Mapa

Ao utilizar o mapa para iniciar um transporte, o utilizador tem uma perceção imediata da localização dos pacientes que necessitam de transporte, uma vez que cada sala tem a indicação do número de transportes em espera. Os transportes mais urgentes também podem ser facilmente detetados através da cor de fundo de cada sala. A Figura 4.16 ilustra o referido anteriormente, através de uma situação em que existem diversos transportes em espera, distribuídos por diferentes salas. Os espaços cinzentos representam salas que não são relevantes para a realização de transportes, enquanto que os espaços brancos representam salas que não têm nenhum transporte em espera.

Um utilizador, quando decide a sala onde pretende iniciar transporte, apenas necessita de clicar no respetivo painel para que surja um *modalpopup* com uma lista de elementos, que contêm a informação pormenorizada de cada transporte com inicio nessa sala (Figura 4.17). Esta lista tem a mesma estrutura da listagem da Secção 4.3.2.1, pelo que para iniciar um dos transportes da sala basta clicar no respetivo painel.

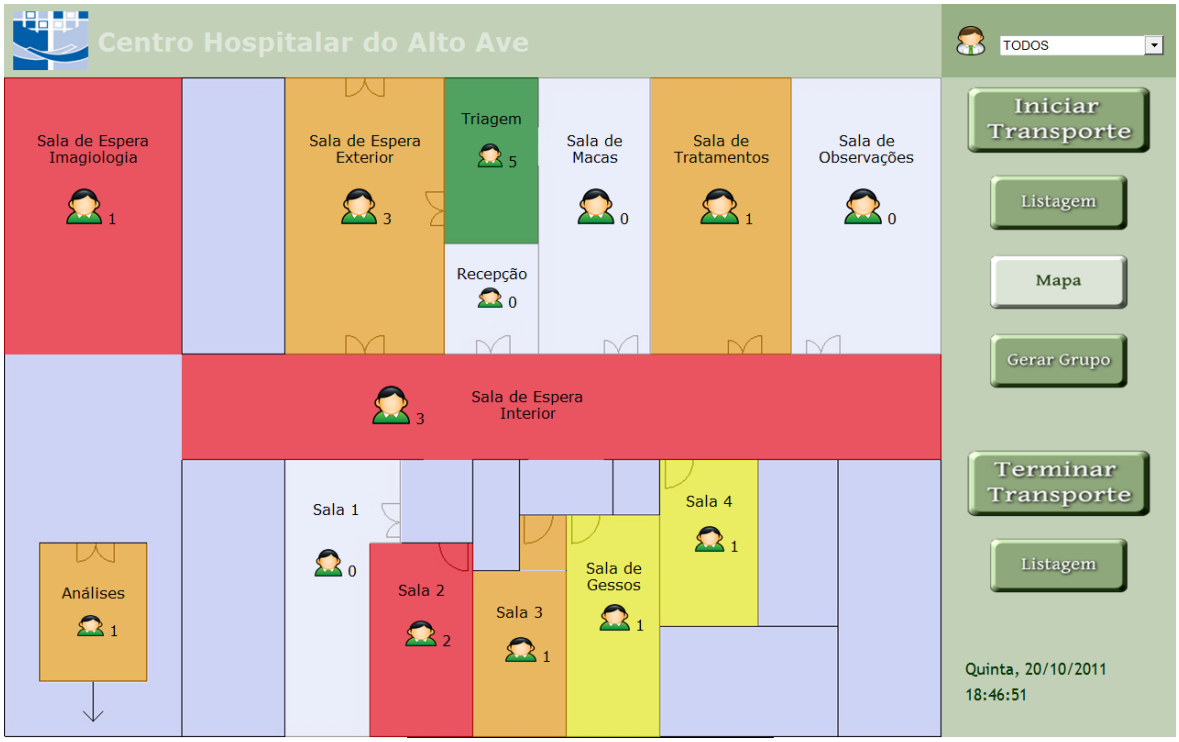


Figura 4.16: Aspeto geral do *patientmove* com a utilização do mapa para iniciar transportes.

4.3.2.3 Grupo de transportes

A opção de gerar/criar um grupo de transportes é um pouco diferente das restantes. Esta funcionalidade requer a escolha prévia de um transporte central a efetuar, podendo esta escolha ser realizada de duas formas distintas:

- Pelo utilizador, através do ícone de grupo associado a cada transporte (existente no painel com a listagem de transportes a iniciar). O transporte correspondente ao ícone clicado é utilizado como ponto de partida para a sugestão dos restantes transportes que podem integrar o grupo.
- Pelo *patientmove*, no caso do utilizador usar a opção "Gerar Grupo" do menu. Nesta situação, o *patientmove* opta por sugerir transportes para a criação de um grupo, em que o transporte central é o mais urgente de todos os transportes em espera.

Depois de escolhido o transporte que funciona como ponto de partida, a aplicação sugere ao utilizador os transportes passíveis de serem iniciados em conjunto com o transporte

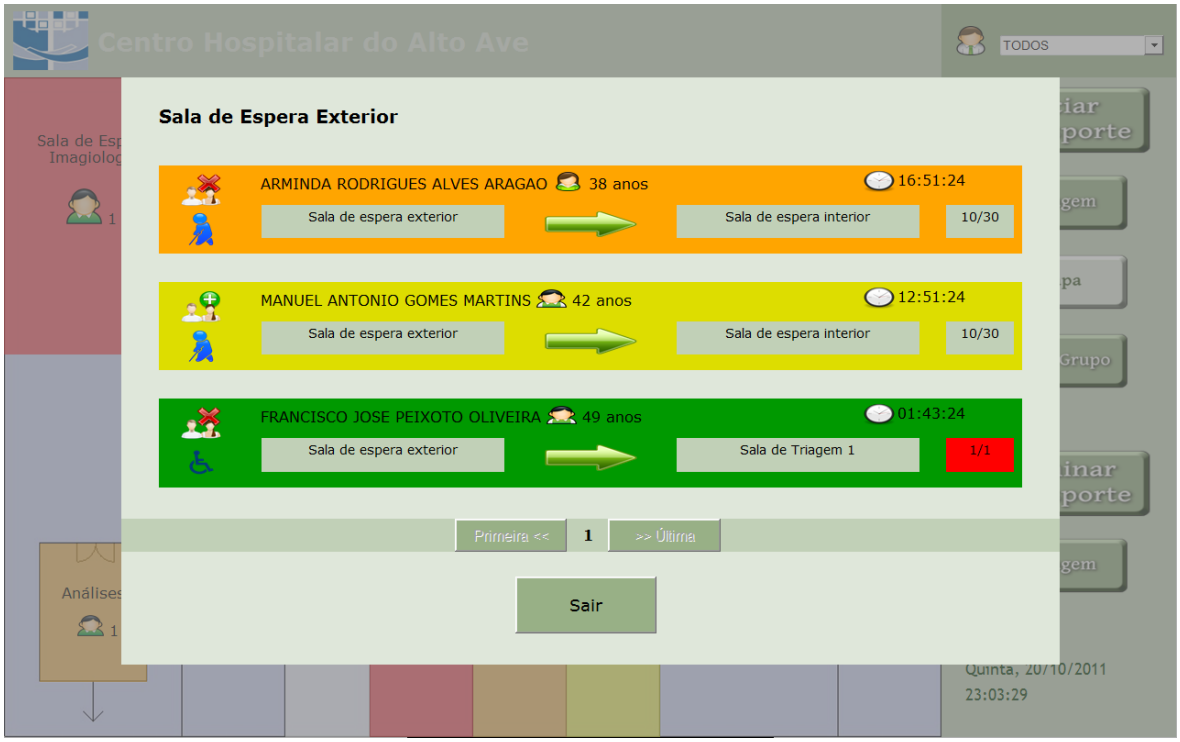


Figura 4.17: Visualização da lista de transportes com inicio na "Sala de Espera Exterior", depois desta ter sido escolhida do mapa.

escolhido. Para serem compatíveis com a integração num grupo, os transportes têm que satisfazer os seguintes critérios em relação ao transporte inicial:

- A origem do transporte sugerido deve ficar situada entre a origem e o destino transporte inicial.
- O destino do transporte sugerido deve fazer com que o transporte não seja realizado em sentido inverso ao transporte inicial. Isto garante que o profissional não cria um grupo em que seja obrigado a andar em várias direções para efetuar todos os transportes.
- Os transportes sugeridos devem permitir que o profissional tenha a capacidade de realizar vários transportes em simultâneo. Assim, não é permitido agrupar dois transportes em que o meio de locomoção ocupe por completo o profissional (cadeira de rodas ou maca). Apenas um transporte desse tipo pode ser adicionado ao grupo, em conjunto com outros em que o paciente apenas apresente dificuldades de locomoção. O facto do paciente ter um acompanhante pode servir de atenuante para esta restrição.

Para que o *patientmove* consiga avaliar a distância e a direção entre a origem e o destino de cada transporte, foi criada uma matriz de distâncias (Anexo C) que classifica a distância com um valor numérico de -6 a 6, em que o zero representa a menor distância, o -6 representa a maior distância com o destino à esquerda da origem e o 6 representa a maior distância com o destino à direita da origem.

Com o intuito de explicar os critérios de localização que condicionam a escolha dos transportes sugeridos para a criação de um grupo, podemos seguir o exemplo da Figura 4.18, em que um transporte "A" funciona como ponto de partida para o criação de um grupo, tendo distância -4 entre a sua origem e destino. As distâncias representadas nas salas, são determinadas em relação à origem do transporte "A". Os transportes sugeridos para a criação do grupo devem respeitar os seguintes critérios:

- A origem deve estar localizada entre a origem e o destino do transporte "A". Logo, no exemplo apresentado, apenas podem ser sugeridos transportes que tenham o seu inicio perto da origem do transporte "A", ou seja, distância de 0 a -4 unidades (as salas permitidas para origem dos transportes estão marcadas a verde, sendo de referir que a distância 1 é considerada sempre permitida, uma vez que representa salas com ligação direta).

- O seu destino deve ficar à esquerda da sua origem, de forma a garantir a mesma direção do transporte "A".

Assim, no exemplo da Figura 4.18, o transporte "B" garante os requisitos anteriores. No entanto, o transporte "C" não cumpre os requisitos porque tem um uma direção oposta ao transporte "A", enquanto o transporte "D" não cumpre os requisitos porque a sua origem está muito longe do percurso do transporte "A".

Na Figura 4.19 é apresentado um exemplo da interface que permite a criação de um grupo de transportes. Esta interface é simples e intuitiva, consistindo apenas numa lista de transportes que são compatíveis com o transporte inicial. O transporte inicial também é incluído na lista mas está previamente selecionado e não é possível excluí-lo do grupo. Para adicionar novos transportes ao grupo, basta clicar no painel do transporte desejado, ficando automaticamente com o ícone verde de seleção à esquerda

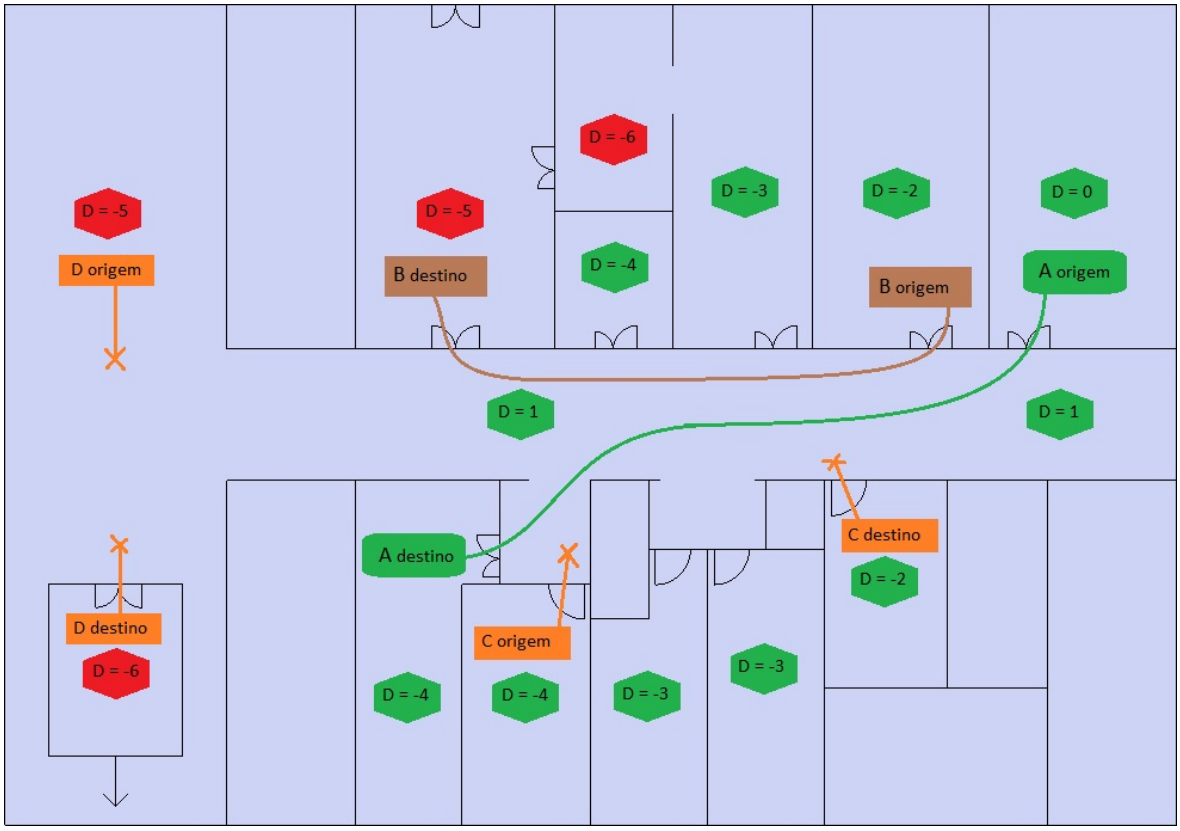


Figura 4.18: Exemplo com as possibilidades de criação de grupo para um transporte genérico "A".

do painel. Para remover um transporte do grupo, o procedimento é o mesmo.

À medida que são adicionados novos transportes ao grupo, podem surgir novas restrições à adição de outros transportes. Esta situação ocorre, principalmente, nos transporte em que o meio de locomoção é cadeira de rodas ou maca. Quando um transporte com estas características é adicionado, os restantes ficam com a sua adição ao grupo bloqueada. Apenas a remoção desse transporte desbloqueia novamente a adição de diferentes transportes em que o meio de locomoção seja cadeira de rodas ou maca.

A criação de grupos apenas está disponível quando não existem transportes com nível máximo de prioridade, ou seja, transportes em que o respetivo paciente tem como resultado de triagem a cor vermelha. Neste caso, a lista de transportes fica reduzida apenas a um transporte com cor de fundo vermelha.

Em qualquer situação, a lista de transporte é iniciada através do botão "Iniciar", disponível no funda da página de criação de grupos. Este botão verifica se todos os transpor-

Figura 4.19: Aspeto geral do *patientmove* na opção "Gerar Grupo", que permite criar um grupo de transportes compatíveis.

tes selecionados estão prontos a ser iniciados. Em caso positivo, inicia os transportes selecionados através do procedimento normal de iniciar transporte. No caso de ser detetado algum erro, a operação de iniciar é cancelada para todos os transportes, sendo exibida uma mensagem erro ao utilizador.

4.3.3 Terminar transporte

A operação de terminar transporte corresponde à ação de comunicar ao *patientmove*, que um transporte ativo (estado = 1) já foi realizado pelo utilizador responsável e que portanto, já pode passar ao estado de transporte terminado (estado = 2).

Nesta operação, de entre todos os transportes, são apresentados ao utilizador apenas os transportes que se encontrem ativos. A única variável que pode surgir na seleção dos transportes ativos é a definição do utilizador responsável pelo transporte. Esta opção de visualizar apenas os transportes associados a um determinado utilizador, permite filtrar rapidamente a lista completa de transportes ativos de forma a encontrar rapidamente o transporte correto. Na Figura 4.20 é apresentada a interface disponibilizada ao

utilizador para terminar transporte, sendo nesse caso apresentados os transportes ativos para todos os utilizadores.

Figura 4.20: Aspeto geral do *patientmove* na opção de terminar transporte.

A visualização em lista é a única disponível, sendo suficiente para cobrir todas as necessidades da operação a realizar, dada a sua simplicidade. A ordem pela qual os transportes são organizados, responde a critérios bastantes diferentes da listagem de transportes em espera. No caso dos transportes a terminar, os primeiros transportes a surgir são aqueles que estão iniciados há mais tempo (diferença entra a hora atual e a hora de inicio), não havendo qualquer influência da prioridade do transporte. Esta opção segue a lógia de que um utilizador tende a terminar primeiro os transportes que foram iniciados em primeiro lugar. Este painel de terminar transporte permite efetuar as seguintes ações:

- **Comunicar a realização de um transporte** – O utilizador escolhe o transporte a terminar através de um clique na área colorida que engloba a informação desse transporte. Caso não haja qualquer erro surge uma mensagem de confirmação.

- **Devolver o transporte ao estado "em espera"** – Esta ação leva o transporte selecionado a ser tratado como um novo pedido, voltando à lista de transportes em espera (estado = 0). A realização desta operação é provocada pelo clique no ícone verde de retroceder, que se encontra à direita do painel do transporte.

- **Eliminar um transporte** – Esta ação tem como finalidade abortar a realização de um transporte, ou seja, permite que o registo desse transporte passe de um estado "iniciado" (1) para o estado "abandonado" (9). Esta ação é desencadeada ao clicar no ícone de cancelar, que se encontra à direita do painel do transporte.

4.4 Testes e resultados

A fase de testes à utilização do *patientmove* levou a diversas melhorias em pormenores que, inicialmente, não estavam a corresponder aos resultados pretendidos. As principais alterações foram efetuadas ao nível da apresentação, havendo também necessidade de alterar o modo de operação de algumas funcionalidades.

4.4.1 Testes de desempenho

No momento em que as melhorias terminaram e foi alcançada a solução final, iniciaram-se os testes à performance do *patientmove*. Todas as atualizações efetuadas durante a utilização do *patientmove* são realizadas através de pedidos assíncronos. Assim, apenas é realizada uma atualizada completa da página quando esta é carregada pela primeira vez. Deste modo, os testes focaram-se na determinação do tempo de resposta aos pedidos assíncronos efetuados pela tecnologia Ajax, uma vez que são estes que vão afetar a real utilização da aplicação. Para determinar os tempo de resposta foi utilizado o browser Mozilla Firefox com o *add-on Firebug*.

A primeira gama de testes foi realizada com todos os componentes da arquitetura multicamada do *patientmove* a funcionar na mesma máquina, ou seja, browser do cliente, servidor Web e serviço de base de dados foram colocados em funcionamento num computador pessoal. Nestas condições, os pedidos assíncronos tiveram um tempo médio de resposta de 62 milissegundos.

A segunda gama de testes foi realizada com os componentes da arquitetura multicamada do patientemove distribuídos pela Internet. O serviço de base de dados ficou alojado num servidor do Departamento de Informática da Universidade do Minho, enquanto que a aplicação Web ficou alojada num servidor IIS a operar online num computador pessoal. O browser do cliente efetua os pedidos através de outra máquina pelo que os componentes ficaram completamente distribuídos pela Internet. Nestas condições, os pedidos assíncronos tiveram um tempo médio de resposta bastante variável devido às condicionantes do estado da rede. Em média os tempos obtidos rondaram os 355 milissegundos.

De uma forma geral, mesmo com os componentes destruídos pela Internet, não é percetível ao utilizador qualquer atraso devido a um tempo excessivo de carregamento das atualização, fazendo com que a utilização da aplicação seja bastante fluida.

Numa futura implementação real, os componentes poderão comunicar através de uma intranet ou através da Internet, sendo de esperar que no caso da Internet os tempos de atualização estejam próximos dos 355 milissegundos e no caso da intranet sejam inferiores.

4.4.2 Diferenças de visualização entre browsers

O *patientmove* foi testado nos browsers mais utilizados atualmente no mercado global. Foram realizados testes completos em cinco browsers que podem ser utilizados em diversas plataformas: Mozilla Firefox, Internet Explorer, Opera, Safari e Google Chrome.

Os testes realizados passaram por utilizar todas as funcionalidades do *patientmove* em diversas situações diferentes. Em relação ao tamanho da janela foi testada a visualização em modo normal e em modo de ecrã completo. No que diz respeito à resolução de ecrã foram testadas diferentes resoluções com proporções distintas. Foram também exploradas possíveis diferenças entre browsers a funcionar em diferentes sistemas operativos.

Os resultados obtidos desta análise indicam que o *patientmove* tem um comportamento homogéneo nos diferentes browsers testados. Apenas se podem indicar pequenas dife-

renças na interpretação de pormenores de visualização, mas nada que altere a visualização geral da página ou que mude drasticamente a visualização de um pormenor.

4.4.3 Funcionamento em dispositivos móveis

Ao longo do desenvolvimento do *patientmove* não foi dada especial atenção à otimização do seu funcionamento em dispositivos móveis. No entanto, com a expansão das potencialidades dos novos dispositivos móveis, a sua crescente utilização nas mais variadas tarefas começa a ser uma realidade. No caso da tarefa de efetuar o transporte de pacientes, também não se pode ignorar as vantagens da utilização destes dispositivos. Seria bastante interessante que, quando o profissional termina um transporte, pudesse escolher nesse mesmo local o próximo transporte a realizar, utilizando para esse efeito o seu próprio dispositivo móvel.

Os testes efetuados à utilização em dispositivos móveis foram realizados através do *Opera Mobile Emulator*. Trata-se de um software que permite emular em qualquer *Desktop* a versão mais recente do browser *Opera Mini*, disponibilizando uma gama de mais de 20 dispositivos móveis que podem ser escolhidos para a simulação. Os testes realizados contribuíram para a deteção de algumas opções do *patientmove* que não eram amigáveis para a utilização em dispositivos móveis. Assim, os aspetos que não exigiam uma total remodelação da aplicação foram alterados de forma a obter-se uma aplicação compatível com os características dos dispositivos móveis.

Depois das melhorias efetuadas, o *patientmove*, quando utilizado em dispositivos móveis, já apresenta uma visualização semelhante à obtida em browsers *Desktop*. De uma forma geral, a utilização em *tablets* é efetuada sem dificuldades assinaláveis (Figura 4.21).

No caso de telemóveis e *smartphones*, continuam a existir alguns problemas com o ajustamento da interface à resolução de ecrã (Figura 4.22). De acordo com os resultados visualizados, a deteção da resolução de ecrã do dispositivo está a funcionar normalmente, uma vez que a interface adquire as proporções corretas. No entanto, ocorre uma diminuição *standard* do tamanho das páginas para que estas sejam adaptadas ao tamanho do ecrã. Assim, aparentemente, o browser do dispositivo móvel não se

Figura 4.21: Visualização do *patientmove* no browser *Opera Mini* a ser executado num *Samsung Galaxy Tab*. Imagem obtida através do *Opera Mobile Emulator*.

apercebe que a resolução já está ajustada ao tamanho do ecrã e aplica a redução de tamanho da mesma forma.

O problema apresentado em telemóveis e *smartphones* pode ser resolvido através da criação de uma versão *patientmove mobile* que tenha a sua resolução definida em função da resolução do dispositivo alvo, eliminando a funcionalidade de adaptação automática à resolução do dispositivo. Em alternativa, poderia ser criada uma versão completamente nova para dispositivos móveis, que utilizasse os mesmos ideais mas que tivesse a sua interface otimizada para as características destes dispositivos.

4.5 Publicação

A publicação online do *patientmove* tem de ser realizada num servidor IIS de forma a utilizar as ferramentas da *.NET framework* essenciais para a execução da aplicação. As configurações do novo diretório criado no IIS não têm que ser alteradas, havendo apenas a necessidade de verificar se todas as ferramentas ASP.NET da versão 3.5 ou superior estão instaladas em conjunto com o servidor IIS.

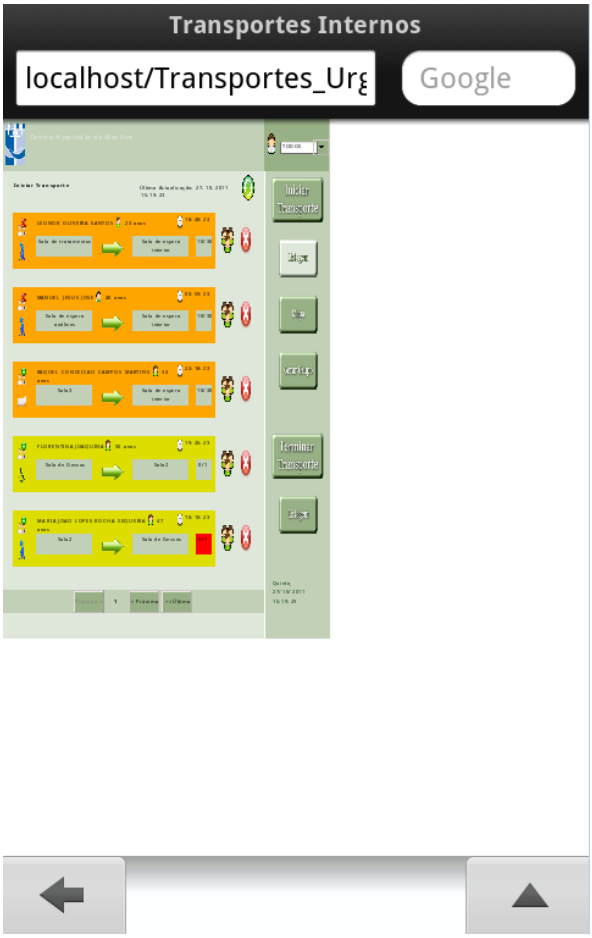


Figura 4.22: Visualização do *patientmove* no browser *Opera Mini* a ser executado no smartphone *Samsung Galaxy S*. Imagem obtida através do *Opera Mobile Emulator*.

No que diz respeito às configurações de comunicação com a base de dados, é importante evitar os conflitos de compatibilidade entre o *Oracle client* instalado no servidor Web e a versão ODP.NET utilizada neste projeto. Assim, foram colocados na pasta do projeto todos os ficheiros *.dll* necessários para assegurar uma comunicação independente com a base de dados. Assim, a aplicação pode ser publicada em qualquer servidor ISS sem ser necessário ter em conta a versão do Oracle cliente instalado ou inclusive se existe ou não algum instalado.

Os dados de conexão à base de dados são configurados no ficheiro *web.config* na Secção

ConectionStrings. Esta opção permite que os dados de conexão sejam configurados em qualquer momento, o que leva a uma maior flexibilidade na escolha da base de dados em funcionamento. O facto de toda a informação se localizar neste ficheiro leva a que o nome de utilizador e password utilizados estejam desprotegidos a intrusos. Para proteger esta informação, deve ser utilizado um mecanismo de encriptação, sendo que a opção mais fácil passa pela utilização do executável *aspnet_regiis.exe*, que permite encriptar blocos do ficheiro *web.config* (Figura 4.23).

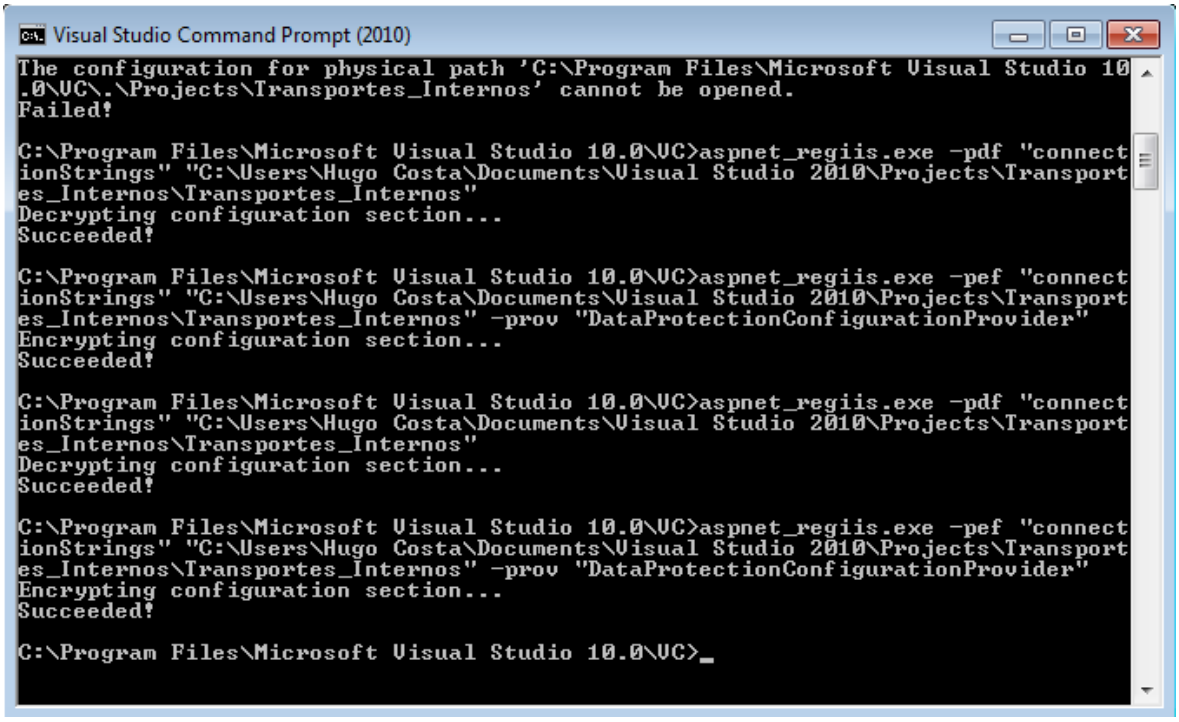


Figura 4.23: Encriptação dos dados de conexão à base de dados com recurso à ferramenta *aspnet_regiis.exe*.

CAPÍTULO 5

Conclusões e Trabalho Futuro

Os sistemas de informação implementados nas Unidades Hospitalares são cada vez mais eficientes e abrangentes. O registo de informação clínica em formato eletrónico foi um dos primeiros objetivos dos sistemas de informação na área da saúde. No entanto, este foi apenas o primeiro passo. Com a informação completamente estruturada e armazenada em bases de dados relacionais, surge a possibilidade de otimizar o desempenho de diversas tarefas para as quais não existia informação estruturada.

O serviço de urgência é o serviço hospitalar que mais necessita de informação a circular de forma eficiente e rápida. Nesta dissertação, a informação disponível acerca dos episódios de urgência, foi utilizada para otimizar a tarefa de transporte de pacientes no interior do serviço de urgência. A aplicação desenvolvida deverá ser integrada no sistema de informação existente no serviço de urgência, tirando partido das funcionalidades já implementadas, de forma a fornecer ao utilizador uma abordagem inovadora para a seleção dos transportes a realizar, permitindo que este perca o mínimo de tempo possível na gestão desta tarefa.

Para orientar o desenvolvimento da aplicação, foram estudadas em pormenor todas as atividades que um auxiliar de saúde realiza no ato de transportar um paciente. Este estudo permitiu implementar funcionalidades que respondem diretamente às necessi-

dades de quem realiza o transporte. Assim, o utilizador continua com um fluxo de trabalho idêntico ao que acontecia anteriormente, tendo a vantagem de desfrutar de uma aplicação informática, constantemente atualizada, que sugere os melhores procedimentos a realizar em operação.

A aplicação desenvolvida baseia-se em tecnologias Web recentes que tornam a interação do utilizador com a aplicação bastante agradável e fluida. A adaptação automática a qualquer resolução de ecrã, a independência do sistema operativo do cliente e a otimização da interfaces para ecrãs tácteis, são características que permitem afirmar que a aplicação está preparada para ser utilizada em diferentes ambientes, sem que existam perdas na sua performance.

Toda a organização a nível de menus e conteúdo está orientada para uma utilização simples e intuitiva, o que permite a realização das operações necessárias no tempo mínimo necessário, tempo esse que fica disponível para o contacto mais próximo com os pacientes ou para realizar mais transportes em espera.

Um dos principais objetivos para a aplicação era implementar a sugestão de transportes prioritários, de acordo com diferentes critérios. Este objetivo levou á disponibilização de três possibilidades para a escolha de transportes a realizar, estando cada uma delas otimizada para um critério diferente. De uma forma geral, o utilizador pode efetuar a sua escolha em função da sua localização no serviço de urgência, em função da prioridade associada a cada paciente ou em função do compatibilidade dos transportes para serem iniciados em grupo. Assim, o utilizador pode escolher, em cada momento, a sugestão que melhor se adapta às suas necessidades.

Em suma, é possível afirmar que os objetivos propostos foram atingidos e que a aplicação desenvolvida está preparada para responder aos desafios existentes no seu contexto de utilização. As especificidades das urgências hospitalares foram tidas em conta e ajudaram a moldar o desenvolvimento da aplicação, prevendo-se que a realização do transporte de pacientes possa ser beneficiar da utilização desta aplicação para a gestão das suas tarefas.

Como trabalho futuro a curto prazo seria interessante efetuar a instalação da aplicação num servidor Web pertencente ao serviço de informação de um hospital, mais

especificamente de um Serviço de Urgência. Assim, a aplicação poderia ser testada no ambiente real para o qual foi desenvolvida, sendo obtidos resultados realistas para a performance da aplicação num contexto de utilização do dia a dia. Além disso, seria desde logo possível obter o feedback dos profissionais que vão lidar diariamente com a aplicação.

Outro aspeto que pode ser interessante melhorar a média prazo, passará por responder às constantes evoluções tecnológicas dos dispositivos móveis. Com o aumento das vantagens associadas à utilização deste tipo de dispositivos, este podem perfilar-se, num futuro próximo, como a melhor solução para executar a aplicação desenvolvida. Seria interessante que o utilizador, em vez de se deslocar a um terminal fixo para consultar a informação, pudesse utilizar um dispositivo móvel para consultar a informação no local onde terminou o último transporte. A aplicação desenvolvida não está completamente otimizada para este tipo de dispositivos. Assim, a conversão de algumas funcionalidades para tecnologias móveis ou a criação de uma interface completamente otimizada para dispositivos móveis serão soluções a ter em consideração.

Bibliografia

(Bindman et al., 1991) Bindman, A.B. et al., 1991. *Consequences of queuing for care at a public hospital emergency department.* Jama The Journal Of The American Medical Association, 266(8), pp. 1091-1096.

(Booth et al., 1992) Booth, A.J. et al., 1992. *Waiting times and patient satisfaction in the accident and emergency department.* Archives of Emergency Medicine, 9(2), pp. 162-168.

(Busch e Koch, 2009) Busch, M. e Koch, N., 2009. *Rich Internet Applications.* IEEE Internet Computing, 14(3), p.9-12.

(Campbell e Sinclair, 2004) Campbell, S.G. e Sinclair, D.E., 2004. *Strategies for managing a busy emergency department.* CJEM Canadian journal of emergency medical care JCMU journal canadien de soins medicaux durgence, 6(4), pp. 271-276.

(Chaudhry et al., 2006) Chaudhry, B. et al., 2006. *Systematic Review: Impact of Health Information Technology on Quality, Efficiency, and Costs of Medical Care.* Annals of Internal Medicine, 144(10), pp. 742-752.

(Cleary e McNeil, 2008) Cleary, P.D. e McNeil, B.J., 1988. *Patient Satisfaction as an Indicator of Quality Care.* Inquiry, 25(1), pp. 25-36.

(Colouris et al., 2011) Colouris, G., Dollimore, J., Kindberg, T. e Blair, G., 2011. *Distributed Systems Concepts and Design.* 5ª ed. Boston: Addison-Wesley, p. 2-26.

(Crockford, 2008) Crockford, D., 2008. *JavaScript: The Good Parts*. USA: O'Reilly Media, p. 1-4.

(Cusumano e Yoffie, 1999) Cusumano, M.A. e Yoffie, D.B., 1999. *Software Development on Internet Time*. Computer, 32(10), pp. 60-69.

(Dearle, 2007) Dearle, A., 2007. *Software Deployment, Past, Present and Future.*. Future of Software Engineering FOSE 07, pp. 269-284.

(Derlet e Richards, 2000) Derlet, R.W. e Richards, J.R., 2000. *Overcrowding in the nation?s emergency departments: complex causes and disturbing effects*. Annals of Emergency Medicine, 35(1), pp. 63-68.

(Direção-Geral da Saúde, 2004) Direção-Geral da Saúde - Ministério da Saúde (Portugal), 2004. *Gestão da informação e do conhecimento*. Plano Nacional de Saúde 2004/2010. Lisboa. (Online) Disponível em: "`http://www.dgsaude.min-saude.pt/pns/vol2_322.html`"(Acedido em 26 maio 2011).

(Direção-Geral da Saúde, 2011) Direção-Geral da Saúde - Ministério da Saúde (Portugal), 2011. *Sistema SIM-Cidadão Relatório 2010*. Lisboa. (Online) Disponível em: "`http://www.dgs.pt/ms/8/default.aspx?pl=&id=5521&acess=0&codigono=001100150043AAAAAAAAAAAA`"(Acedido em 26 maio 2011).

(Duckett, 2011) Duckett, J., 2011. *Beginning Web Programming with HTML, XHTML, and CSS*. Indianapolis: John Wiley e Sons, p. 1-7, 217-220.

(Eilers, 2002) Eilers, G.M., 2002. *Improving patient satisfaction with waiting time*. Journal of American college health J of ACH, 53(1), pp. 41-43.

(Flanagan, 2010) Flanagan, D., 2006 *JavaScript: the definitive guide*. USA: O'Reilly Media, p. 1-10.

(Freeman, 2011) Freeman, A., 2010. *Applied ASP.NET 4 in Context*. New York: Apress, p. 13-18, 553-560, 1239-1293.

(Green et al., 2006) Green, L.V. et al., 2006. *Using queueing theory to increase the effectiveness of emergency department provider staffing*. Academic emergency me-

dicine official journal of the Society for Academic Emergency Medicine, 13(1), pp. 61-68., 266(8), p.1091-1096.

(Greenwald et al, 2008) Greenwald, R., Stackowiak, R., Stern, J., 2008. *Oracle Essentials: Oracle Database 11g*. USA: O'Reilly Media, p. 1-27.

(Guisán et al., 2001) Guisán, C. A. et al., 2001. *Triage criteria in a emergency department*. Anales Espanoles De Pediatria, 54(3), p.233-237.

(Guthrie e Robsman 2007) Guthrie, S. D. e Robsman, D., Microsoft Corporation., 2007. *PHP: ASP.NET HTTP Runtime*. U.S. Pat. 7,162,723 B2.

(Hayrinen et al., 2008) Hayrinen, K., Saranto, K. e Nykanen, P., 2008. *Definition, structure, content, use and impacts of electronic health records: a review of the research literature*. International Journal of Medical Informatics, 77(5), p.291-304.

(Hoot e Aronsky, 2008) Hoot, N.R. e Aronsky, D., 2008. *Systematic review of emergency department crowding: causes, effects, and solutions*. Annals of Emergency Medicine, 52(2), pp. 126-136.

(IGIF, 2005) Instituto de Gestão Informática e Financeira da Saúde (IGIF) - Ministério da Saúde de Portugal, 2005. *Informatização Clínica dos Serviços de Urgência - Linhas Básicas de Orientação*. (Circular Normativa nº1 de 2/Dez/2005) Lisboa. (Online) Disponível em: "http://www.acss.min-saude.pt/INForma%C3%A7%C3%A3o/Circulares/tabid/100/language/pt-PT/Default.aspx?PageContentMode=1"(Acedido em 26 maio 2011).

(Institute of Medicine, 2007) Institute of Medicine - Committee on the Future of Emergency Care in the United States Health System, 2007. *Hospital-based emergency care: at the breaking point*. Washington: The National Academies Press.

(Iserson e Moskop, 2004) Iserson, K.V. e Moskop, J.C., 2007. *Triage in medicine, part I: Concept, history, and types*. Annals of Emergency Medicine, 49(3), p.275-281.

(Jazayeri, 2007) Jazayeri, M., 2007. *Some Trends in Web Application Development*. Future of Software Engineering FOSE 07, pp. 199-213.

(Jensen e Crane, 2008) Jensen, K. e Crane, J., 2008. *Improving patient flow in the emergency department*. Healthcare financial management journal of the Healthcare Financial Management Association, 62(11), pp. 104-106, 108.

(Koch, 2010) Koch, P. P., 2010. *Mobile compatibility tables*. (Online) Disponível em: "`http://www.quirksmode.org/m/table.html`"(Acedido em 10 junho 2011).

(Kravitz, 1998) Kravitz, R., 1998. *Patient Satisfaction with Health Care*. Journal of General Internal Medicine, 13(4), pp. 280-282.

(Kuhn e Giuse, 2001) Kuhn, K.A. e Giuse, D.A., 2001. *From hospital information systems to health information systems. Problems, challenges, perspectives*. Methods of Information in Medicine, 40(4), pp. 275-87.

(Macdonald, 2010) Macdonald, M., 2010. *Beginning ASP.NET 4 in VB 2010*. USA: Apress, p. 6-15, 863-865.

(MacDonald et al., 2010) MacDonald, M., Freeman, A. e Szpuszta, M., 2010. *Pro ASP.NET 4 in C# 2010*. 4ª ed. New York: Apress, p. 2-19, 277-282, 353-423, 1239-1293.

(Madhavji, 1991) Madhavji, N.H., 1991. *Software Engineering: The process cycle*. Software Engineering Journal, 6(5), pp. 234-242.

(Manchester Triage Group, 2006) Manchester Triage Group, 2006. *Emergency Triage*. 2ª ed. Oxford: BlackWell Publishing.

(Mason et al., 2006) Mason, S. et al., 2006. *What are the organisational factors that influence waiting times in Emergency Departments?*. (Online) Disponível em: "`http://www.sdo.nihr.ac.uk/files/project/49-final-report.pdf`"(Acedido em 15 junho 2011).

(Minetto, 2007) Minetto, E. L., 2007. *Frameworks para Desenvolvimento em PHP*. Brasil: Novatec, p. 14-22.

(Ministério da Saúde, 2006) Ministério da Saúde - Hospitais SA, 2006. *O Serviço de Urgência - Recomendações para a organização dos cuidados urgentes e emergentes*. Lisboa.

(Mishra e Beaulieu, 2004) Mishra, S. e Beaulieu, A., 2004. *Mastering Oracle SQL*. USA: O'Reilly Media, p. 1-17.

(Nurmuliani et al., 2004) Nurmuliani, N., Zowghi, D. e Fowell, S., 2004. *Analysis of requirements volatility during software development life cycle*. In 2004 Australian Software Engineering Conference Proceedings. Ieee, pp. 28-37.

(Oracle, 2008) Oracle, 2008. *Part I - Getting Started with Oracle Net Services*. Oracle Database Net Services Administrator's Guide 11g Release 1. (Online) Disponível em: "http://download.oracle.com/docs/cd/B28359_01/network.111/b28316/toc.htm"(Acedido em 13 março 2011).

(Oracle, 2011a) Oracle, 2011a. *Oracle Call Interface*. (Online) Disponível em: "http://www.oracle.com/technetwork/database/features/oci/index.html"(Acedido em 13 março 2011).

(Oracle, 2011b) Oracle, 2011b. *Oracle Database Instant Client*. (Online) Disponível em: "http://www.oracle.com/technetwork/database/features/instant-client/index-100365.html"(Acedido em 13 março 2011).

(Oracle, 2011c) Oracle, 2011c. *Oracle Data Provider for .NET*. (Online) Disponível em: "http://www.oracle.com/technetwork/topics/dotnet/index-085163.html"(Acedido em 13 março 2011).

(Parekh, 2005) Parekh, N., 2005. *The Waterfall Model Explained*. (Online) Disponível em: "http://www.buzzle.com/editorials/1-5-2005-63768.asp"(Acedido em 02 dezembro 2010).

(Pollock, 2010) Pollock, J., 2010 *Javascript: A Beginner's Guide*. USA: The McGraw-Hill Companies, p. 1-28, 148-152, 241-248, 298-305.

(Powell, 2010) Powell, T. A., 2010. *HTML & CSS: the complete reference*. 5ª ed. USA: The McGray-Hill Companies, p. 429-439.

(Press Ganey, 2010) Press Ganey, 2010. *Pulse Report for Emergency Department*. Columbia: Press Ganey Associates. (Online) Disponível em: "http:

//www.pressganey.com/researchresources/hospitals/emergencyDepartment/emergencyPulsereport.aspx"(Acedido em 10 junho 2011).

(Rossi et al., 2008) Rossi, G., Pastor, O., SchWabe, D. e Olsina, L., 2008. *Web Engineering: Modelling and Implementing Web Applications*. London: Springer, p. 7-29.

(Roy e Gibbs, 2007) Roy, B. e Gibbs, M., 2007. *ASP.NET AJAX UpdatePanel Control*. USA: O'Reilly Media.

(Ruparelia, 2010) Ruparelia, N.B., 2010. *Software development lifecycle models*. ACM SIGSOFT Software Engineering Notes, 35(3), pp. 8.

(Satalkar, 2010a) Satalkar, B., 2010a. *Waterfall Model Advantages*. (Online) Disponível em: "http://www.buzzle.com/articles/waterfall-model-advantages.html"(Acedido em 02 dezembro 2010).

(Satalkar, 2010b) Satalkar, B., 2010b. *Modified Waterfall Model*. (Online) Disponível em: "http://www.buzzle.com/articles/modified-waterfall-model.html"(Acedido em 02 dezembro 2010).

(Schafer, 2010) Schafer, S. M., 2010. *HTML, XHTML, and CSS Bible*. 5ª ed. Indianapolis: Wiley Publishing, p. 4-13.

(Shelly et al., 2008) Shelly, G. B., Woods, D. M., Dorin, W. J., 2008. *HTML: Comprehensive Concepts and Techniques*. 4ª ed. Boston: Cengage Learning, p. 8-11.

(Shortliffe e Cimino, 2006) Shortliffe, E. D. e Cimino, J. J. eds., 2006. *Biomedical Informatics: Computer Applications in Health Care and Biomedicine*. 3ª ed. New York: Springer, p. 3-13.

(Sitzia e Wood, 1997) Sitzia, J. e Wood, N., 1997. *Patient satisfaction: a review of issues and concepts*. Social science medicine, 45(12), pp. 1829-1843.

(Sven et al., 2009) Sven, C., Florian, D., Dolog, P. e Maristella, M., 2009. *Engineering Web Applications*. Berlin: Springer, p. 1-4.

(Taylor e Benger, 2004) Taylor, C. e Benger, J., 2004. *Patient satisfaction in emergency medicine*. Emergency medicine journal EMJ, 21(5), p.528-532.

(Torchiano et al., 2010) Torchiano, M., Ricca, F. e Marchetto, A., 2010. *Are web applications more defect-prone than desktop applications?*. International Journal on Software Tools for Technology Transfer, 13(2), p.151-166.

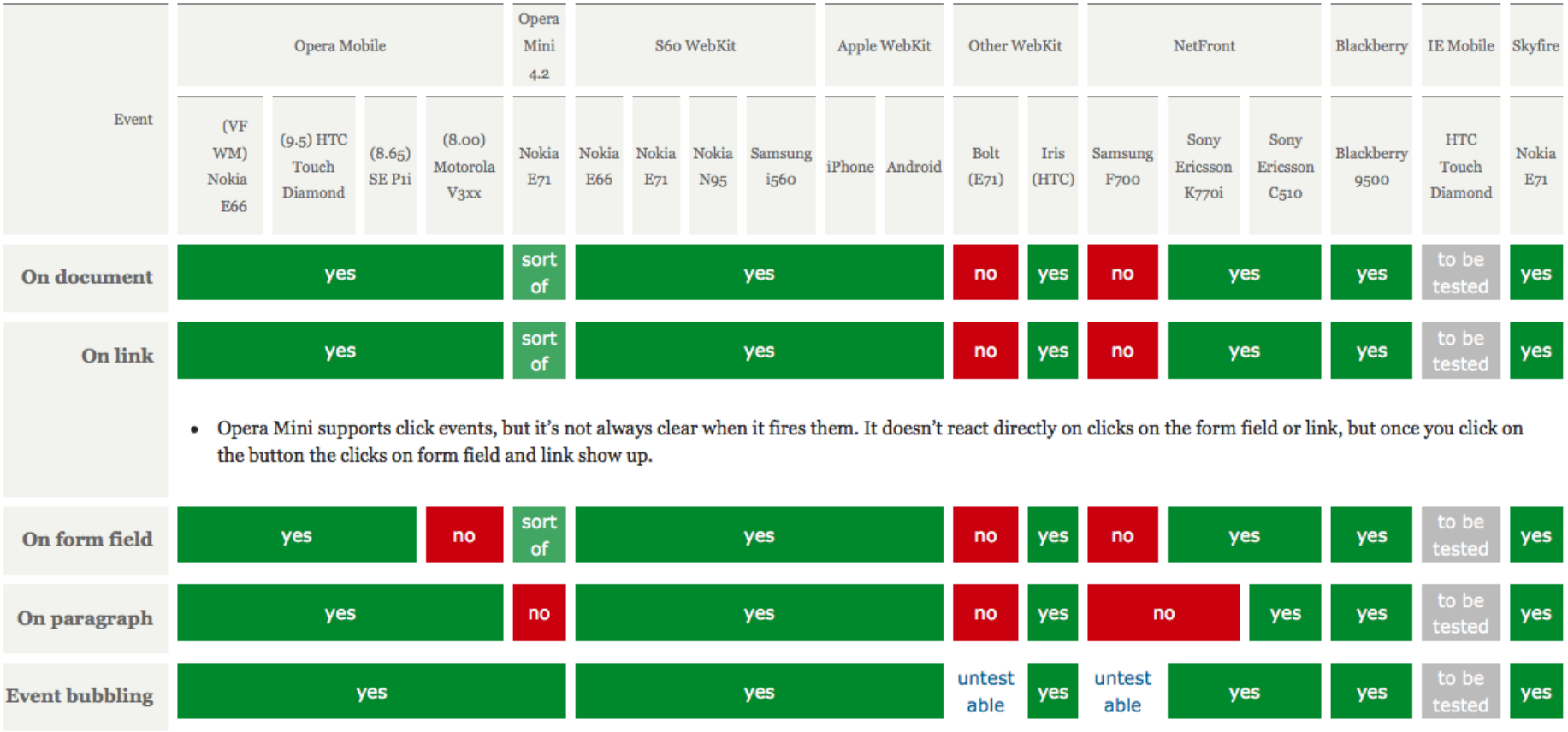
(Vaswani, 2009) Vaswani, V., 2009. *PHP: A Beginner's Guide*. USA: The McGray-Hill Companies, p. 3-16.

(Walker et al., 2005) Walker, J. et al., 2005. *The value of health care information exchange and interoperability*. Health Affairs, 24, p. hlthaff.w5.10.

(Wenz, 2007) Wenz, C., 2007. *Programming ASP.NET AJAX*. USA: O'Reilly Media, p. 3-14, 44-60, 116-130, 177-184.

APÊNDICE A

Tabelas de compatibilidade para browsers de dispositivos móveis



Event	Opera Mobile				Opera Mini 4.2	S60 WebKit				Apple WebKit		Other WebKit		NetFront			Blackberry	IE Mobile	Skyfire
	(VF WM) Nokia E66	(9.5) HTC Touch Diamond	(8.65) SE P1i	(8.00) Motorola V3xx	Nokia E71	Nokia E66	Nokia E71	Nokia N95	Samsung i560	iPhone	Android	Bolt (E71)	Iris (HTC)	Samsung F700	Sony Ericsson K770i	Sony Ericsson C510	Blackberry 9500	HTC Touch Diamond	Nokia E71
On document	yes				sort of	yes						no	yes	no	yes		yes	to be tested	yes
On link	yes				sort of	yes						no	yes	no	yes		yes	to be tested	yes
On form field	yes			no	sort of	yes						no	yes	no	yes		yes	to be tested	yes
On paragraph	yes				no	yes						no	yes	no		yes	yes	to be tested	yes
Event bubbling	yes					yes						untestable	yes	untestable	yes		yes	to be tested	yes

- Opera Mini supports click events, but it's not always clear when it fires them. It doesn't react directly on clicks on the form field or link, but once you click on the button the clicks on form field and link show up.

Figura A.1: Resposta ao evento de clique (retirado de (Koch, 2010)).

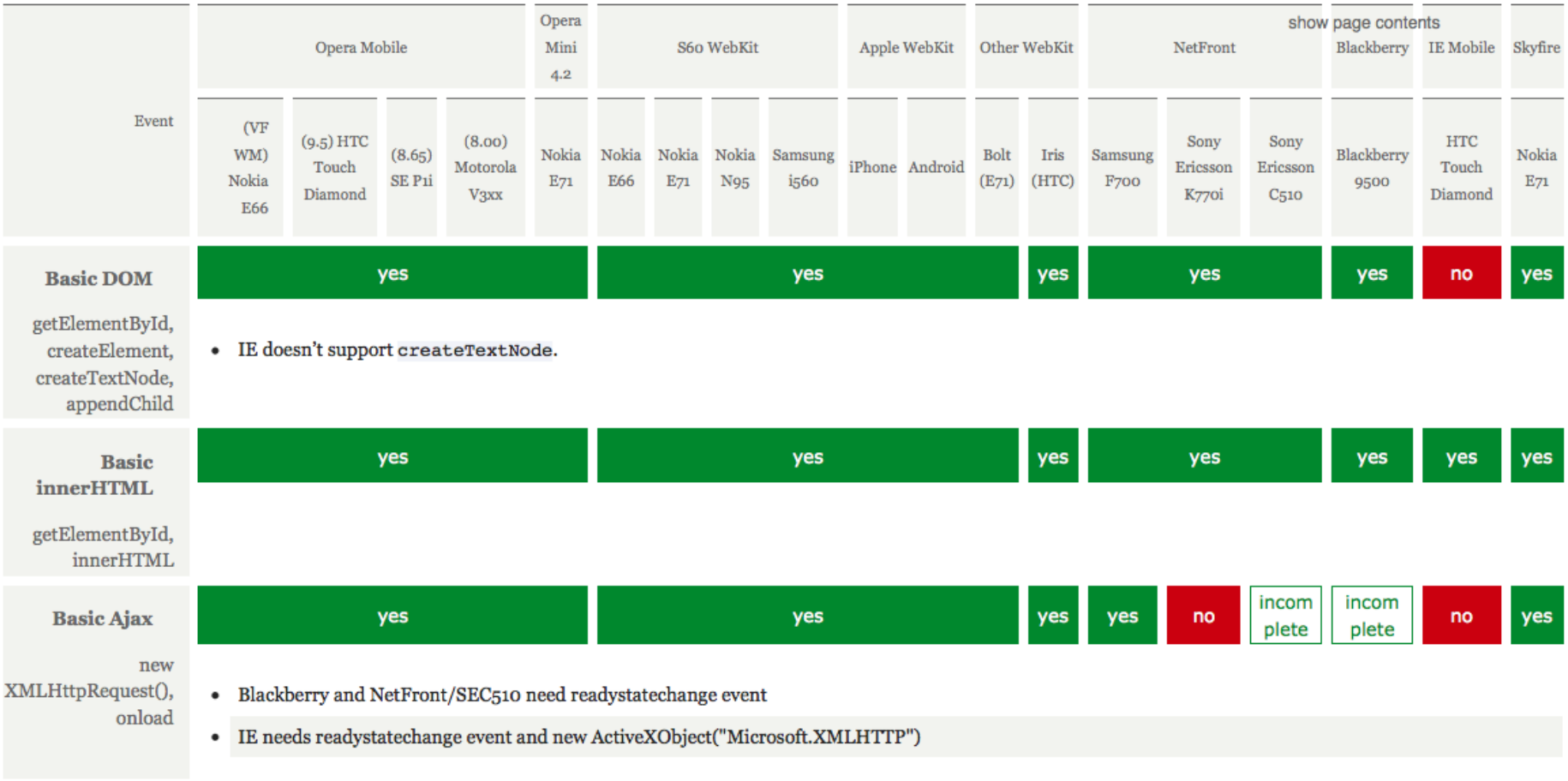


Figura A.2: Tratamento das funcionalidades DOM e Ajax (retirado de (Koch, 2010)).

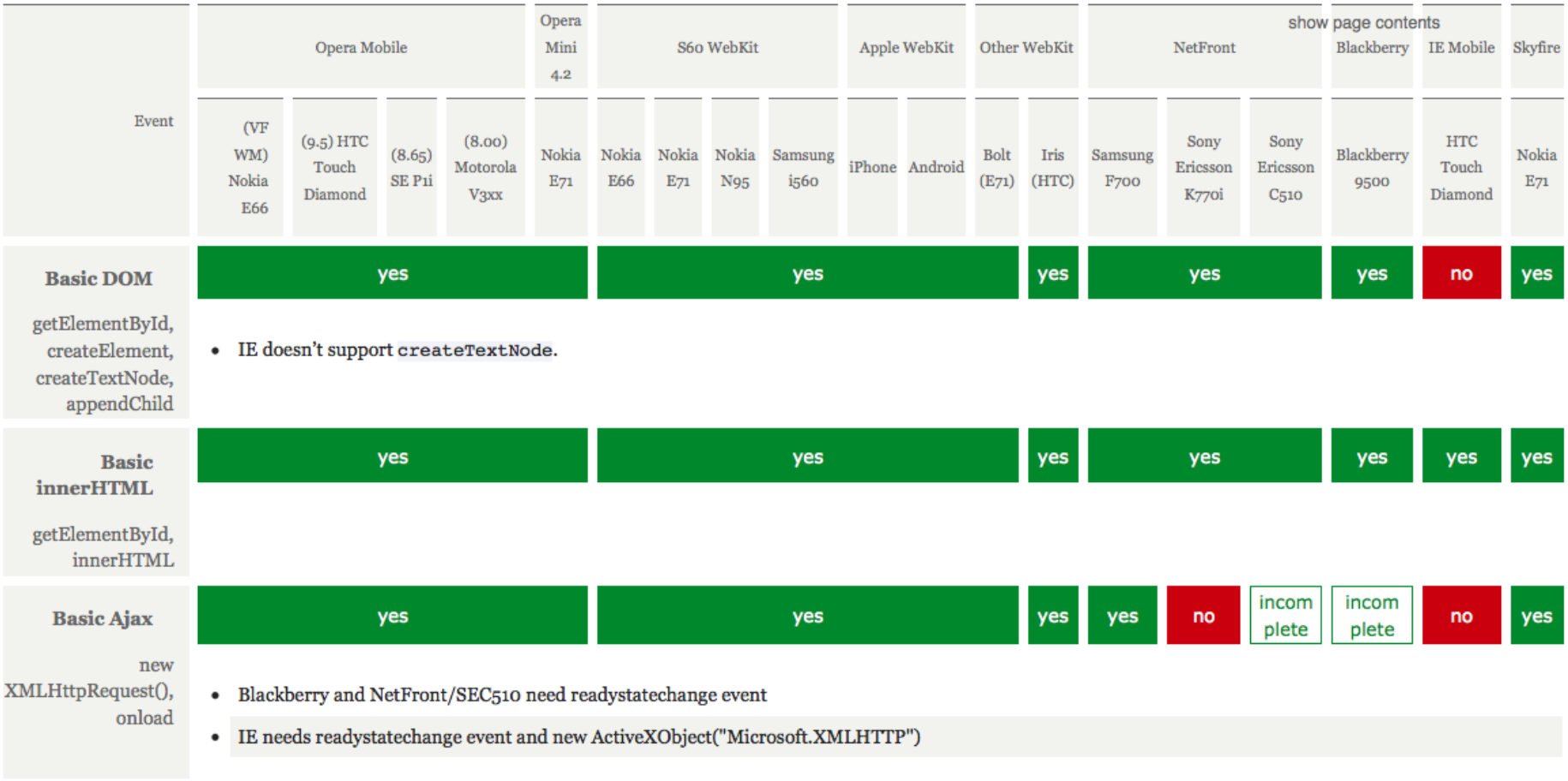


Figura A.3: Tratamento das funcionalidades DOM e Ajax (Continuação) (retirado de (Koch, 2010)).

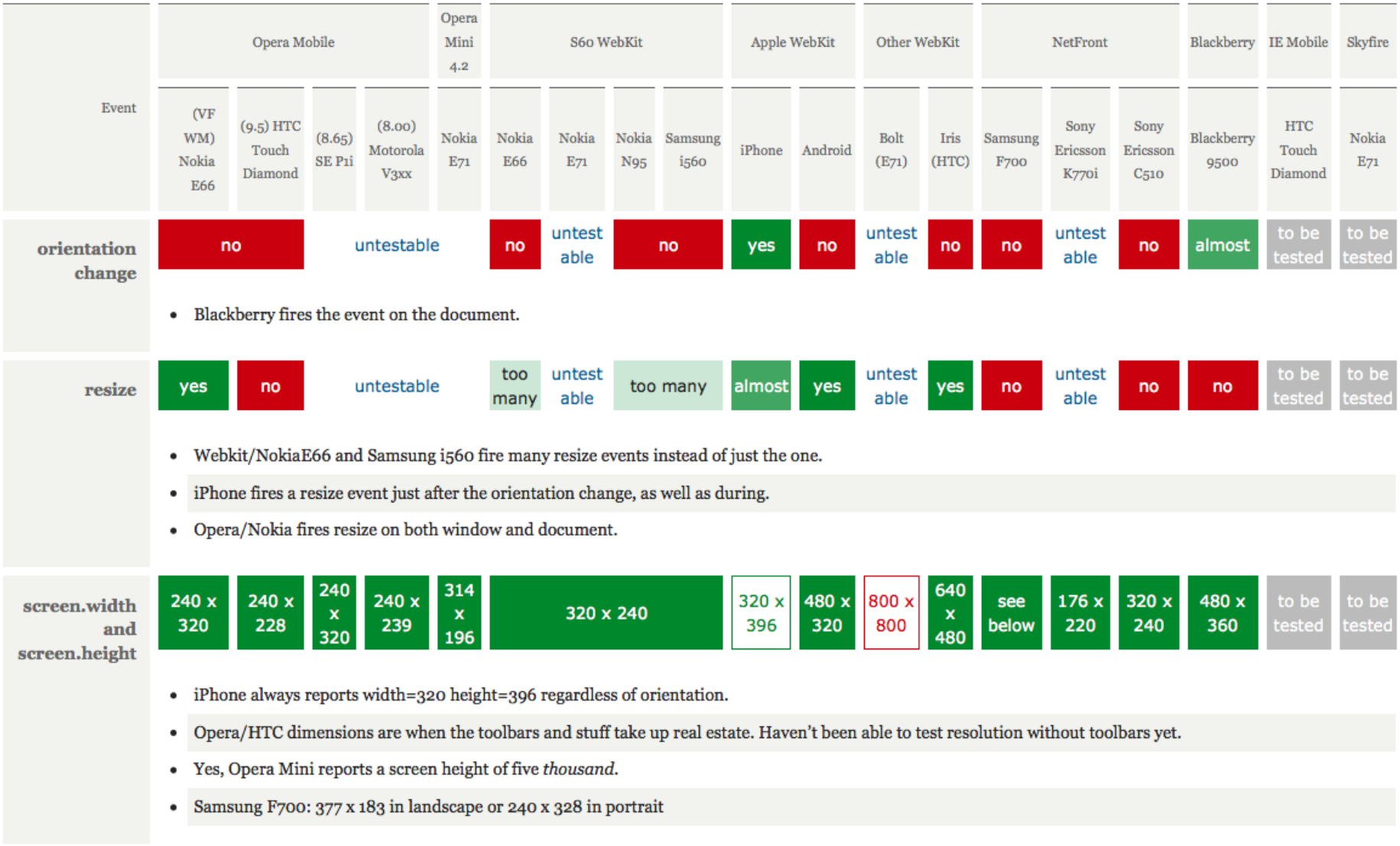


Figura A.4: Resposta aos pedidos *orientation change*, *screen width* e *height* (retirado de (Koch, 2010)).

APÊNDICE B

Determinação da resolução e do tamanho de letra

```javascript
function get_width() {
    var screenW = 640;
    if (parseInt(navigator.appVersion) > 3) {
        screenW = screen.width;
    }
    else if (navigator.appName == "Netscape" && parseInt(navigator.appVersion) == 3 && navigator.javaEnabled()) {
        var jToolkit = java.awt.Toolkit.getDefaultToolkit();
        var jScreenSize = jToolkit.getScreenSize();
        screenW = jScreenSize.width;
    }
    var res = (screenW * 1) - 1;
    screenW = res.toString() + 'px';
    return screenW;
}
```

Figura B.1: Função Javascript para determinar a largura da resolução do ecrã.

```javascript
function get_height() {
    var screenH = 480;
    if (parseInt(navigator.appVersion) > 3) {
        screenH = screen.height;
    }
    else if (navigator.appName == "Netscape" && parseInt(navigator.appVersion) == 3 && navigator.javaEnabled()) {
        var jToolkit = java.awt.Toolkit.getDefaultToolkit();
        var jScreenSize = jToolkit.getScreenSize();
        screenH = jScreenSize.height;
    }
    var res = (screenH * 1) - 1;
    screenH = res.toString() + 'px';
    return screenH;
}
```

Figura B.2: Função Javascript para determinar a altura da resolução do ecrã.

```javascript
function tamanho_letra() {
    var altura = get_height() + "";
    altura = altura.substring(0, altura.length - 2);
    var largura = get_width() + "";
    largura = largura.substring(0, largura.length - 2);
    var percent_largura = Math.round(((largura * 1) / 1280) * 100);
    var str_largura = percent_largura.toString() + '%';
    var percent_altura = Math.round(((altura * 1) / 800) * 100);
    var str_altura = percent_altura.toString() + '%';
    document.body.style.fontsize = str_largura;
    if (percent_largura < percent_altura) {
        document.body.style.fontSize = str_largura;
    } else {
        document.body.style.fontSize = str_altura;
    }
}
```

Figura B.3: Função Javascript para determinar o tamanho da letra em função da resolução do ecrã.

APÊNDICE C

Matriz de distâncias

		TR01	ESPE	ESPI	SM	ST	SOBS	S1	S2	S3	S4	SG	SIMG	ANALIS	RECEP
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
TR01	1	0	1	1	1	5	6	3	3	4	5	4	-4	-6	2
ESPE	2	1	0	1	2	4	5	2	2	3	4	3	-3	-5	2
ESPI	3	1	1	0	1	1	1	1	1	1	1	1	1	1	1
SM	4	1	-2	1	0	2	3	-3	-3	2	2	2	-4	-5	-3
ST	5	-5	-4	1	-2	0	2	-3	-3	-3	2	-2	-4	-5	-4
SOBS	6	-6	-5	1	-3	-2	0	-4	-4	-3	-2	-3	-5	-6	-4
S1	7	-3	-2	1	3	3	4	0	0	2	3	2	-3	-4	3
S2	8	-3	-2	1	3	3	4	0	0	2	3	2	-3	-4	3
S3	9	-4	-3	1	-2	3	3	-2	-2	0	2	0	-4	-5	3
S4	10	-5	-4	1	-2	-2	2	-3	-3	-2	0	-2	-4	-5	-3
SG	11	-4	-3	1	-2	2	3	-2	-2	0	2	0	-4	-5	3
SIMG	12	4	3	1	4	4	5	3	3	4	4	4	0	2	4
ANALIS	13	6	5	1	5	5	6	4	4	5	5	5	-2	0	5
RECEP	14	-2	-2	1	3	4	4	-3	-3	-3	3	-3	-4	-5	0

Figura C.1: Matriz de distâncias entre as salas e o serviço de urgência.