Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

PROJETO DE ESTÁGIO

Licenciatura em Engenharia Informática

Ricardo Miguel Dias Antunes
dezembro | 2012

Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Gestão de Reservas: Casa do Pastor

Ricardo Miguel Dias Antunes - Nº 1009694

Projeto Aplicado no Curso
de Engenharia Informática em contexto de estágio

2 de Dezembro de 2012

Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Gestão de Reservas: Casa do Pastor

Ricardo Miguel Dias Antunes - Nº 1009694

Projeto Aplicado no Curso
de Engenharia Informática em contexto de estágio

Supervisor: Carlos Fernando Duarte Pais - Sócio-gerente - Ideias Soberbas

Orientador: Mestre Paulo Jorge Costa Nunes, Professor Adjunto da Unidade Técnico-Científica de Informática da ESTG.

2 de Dezembro de 2012

Agradecimentos

O desempenho no desenvolvimento deste projeto em contexto de estágio não teria sido o mesmo sem o apoio, a colaboração e a paciência de algumas pessoas, às quais gostaria de aqui mostrar o meu reconhecimento.

Em primeiro lugar, como não poderia deixar de ser, quero deixar um agradecimento a todos os familiares e amigos que sempre me apoiaram e incentivaram no percurso deste projeto.

Em segundo lugar, um agradecimento aos professores da Unidade Curricular Projeto de Informática, Professor Paulo Nunes e Professor Noel Lopes.

Em terceiro lugar, e não menos importante, um agradecimento à Diretora da Escola Superior de Tecnologia e Gestão, Professora Doutora Maria Clara Silveira pelo seu apoio e disponibilidade.

Gostaria também de deixar um agradecimento ao sócio-gerente da empresa Ideias Soberbas, Carlos Fernando Duarte Pais por todo o apoio e dedicação demonstrada, bem como à Anabela Filomena Santos Pinto Pais, igualmente sócia-gerente da mesma empresa, por toda a sua disponibilidade e apoio prestado no desenvolvimento do projeto.

Por fim, um agradecimento muito especial à minha namorada pela compreensão, paciência, carinho, apoio e dedicação que sempre demonstrou.

Resumo

Este documento descreve o trabalho realizado no âmbito da Unidade Curricular Projeto de Informática na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão no Instituto Politécnico da Guarda.

A evolução tecnológica tem permitido, cada vez mais, que tudo o que tem a ver com reservas, possa ser guardado em formato digital de uma forma simples, fazendo com que a procura de aplicações para o efeito cresça cada vez mais.

O trabalho consistiu no desenvolvimento de uma aplicação web na linguagem de programação PHP e gestor de base de dados MySQL, que permitisse ao cliente consultar o calendário das casas numa determinada data, de forma a que pudesse criar a sua reserva sabendo que casas havia disponíveis nessa mesma data. Para além disso, foi também criada uma plataforma de gestão, direcionada para os gestores e administrador, onde fica guardada toda a informação das reservas, e onde estes podem editar, eliminar e inserir todos os registos de forma simplificada.

Palavras Chave

Evolução Tecnológica, Aplicação web, PHP, MySQL, reservas online, plataforma de gestão.

Abstract

This document describes the work done under the discipline Projeto de Informática in the graduation in Engenharia Informática from Escola Superior de Tecnologia e Gestão in the Instituto Politécnico da Guarda.

The technological evolution allowed, more and more, that everything that has to do with reservations, could be stored in digital format from a simple way, causing the search for applications for this purpose growing more and more.

The work consists to develope one web aplication on programming language of PHP and database manager MySQL, that allowed the clients consulting the calendar of all the houses in a certain date, so that he could creat his reservation knowing that houses were available. In addition, has also created a management platform, directed from managers and administrator, where all the information from reservations are stored, and where they can edit, delete and insert all records in a simplified way.

Key words

Technological evolution, web aplication, PHP, MySQL, online reservations, management platform.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de listagens

Glossário

bps — Bits por segundo.

CPU — A unidade central de processamento ou CPU (Central Processing Unit).

FTP — Protocolo de transferência de ficheiros (File Transfer Protocol).

IP — Protocolo de internet (Internet Protocol).

IT — Tecnologias de informação (Information technology).

Mbits — Um milhão de bits.

Página Web — Conhecida também pelo inglês webpage, é uma página na World Wide Web, normalmente no formato HTML, com ligações de hipertexto que permitem a navegação entre outras páginas ou secções.

Open Web Platform — coleção de tecnologias Web desenvolvidas pela W3C e por outros corpos de normalização, como o Unicode Consortium, Internet Engineering Task Force, entre outros.

W3C — World Wide Web Consortium é um consórcio internacional que visa desenvolver padrões para a criação e interpretação de conteúdos para a Web.

WWW — World Wide Web é um sistema de documentação em todos os formatos (textos, imagens, vídeos) que são interligados e executados na internet.

PHP — Hypertext Preprocessor, originalmente Personal Home Page.

Servidor Web — Um programa de computador responsável por aceitar pedidos HTTP de clientes, geralmente os navegadores, e servi-los com respostas HTTP, incluindo opcionalmente dados, que geralmente são páginas web, tais como documentos HTML com objetos embutidos.

Site — Um website ou site é um conjunto de páginas web, isto é, de hipertextos acessíveis geralmente pelo protocolo HTTP na internet.

HTTP — Hipertext Transfer Protocol, ou em Português, Protocolo de Transferência de Hipertexto é um protocolo de comunicação.

plugin — ou módulo de extensão, é um programa de computador usado para adicionar funções a outros programas maiores, provendo funcionalidades especiais ou específicas.

script — as linguagens de script servem para estender a funcionalidade de um programa e/ou para o controlar.

IETF — Internet Engineering Task Force, é uma comunidade internacional ampla e aberta (técnicos, agências, fabricantes, fornecedores, pesquisadores) preocupada com a evolução da arquitetura da Internet e o seu perfeito funcionamento.

Backbone — esquema de ligações centrais de um sistema amplo, tipicamente de elevado desempenho.

SGML — o Standard Generalized Markup Language é uma metalinguagem através da qual se pode definir linguagens de marcação para documentos

Unicode — padrão que permite aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente

Cookie — testemunho de conexão, ou, simplesmente, testemunho, é um grupo de dados trocados entre o navegador e o servidor de páginas web, colocado num arquivo (ficheiro) de texto criado no computador do utilizador, sendo que a sua principal função é a de manter a persistência de sessões HTTP

Capítulo 1

Introdução

O presente relatório descreve o projeto em contexto de estágio desenvolvido pelo aluno Ricardo Miguel Dias Antunes, na empresa Ideias Soberbas, no âmbito da disciplina de Projeto de Informática na Escola Superior de Tecnologia e Gestão no Instituto Politécnico da Guarda.

A Casa do Pastor está situada na aldeia rústica da Póvoa Velha, de ambiente campestre rodeado por montanhas, paisagens verdejantes e pelo "'silêncio"' da natureza, a cerca de 5 quilómetros de Seia, e é constituída por um conjunto de casas datadas de 1868, de turismo de habitação, destinada a todas as faixas etárias.

Na Casa do Pastor as reservas são feitas apenas através de contacto telefónico, e estas são guardadas numa agenda em formato de papel. Por um lado, o contacto telefónico é interessante no que diz respeito ao contacto entre cliente e gestores, por outro, este contacto exige disponibilidade durante aquele tempo por parte de ambos os lados. Quanto à agenda, cada vez se torna menos eficaz, uma vez que nem sempre tudo fica apontado, e por vezes não se sabe exatamente onde se apontaram os dados da reserva, o que pode ser bastante desvantajoso quer para o cliente quer para a Casa em si.

A relação entre evolução das tecnologias e adaptação das pessoas não evoluiu da mesma forma, o que faz com que pessoas com uma maior idade possam ter dificuldades em se adequarem ás tecnologias, mesmo sabendo que estas podem ser muito favoráveis. Neste caso em particular, a pessoa que trata das reservas que tem já uma certa idade, acredita que a agenda em formato de papel é muito mais simples e eficaz do que partir para um sistema informático; no entanto, acontece que por vezes se perdem dados das reservas na sua agenda.

O projeto consistiu em desenvolver uma aplicação web de forma a facilitar, por um lado, a criação da reserva por parte do cliente, e, por outro lado, a gestão desta por parte dos gestores.

O projeto enquadra-se no âmbito e complexidade adequada às competências adquiridas no curso:

- Autonomia e capacidade de definir objetivos;
- Capacidade de modelação de problemas;

- Saber elaborar relatórios de análise, desenho e implementação de soluções;
- Gestão de tempo e cumprimento de prazos;

O projeto obedeceu às seguintes condições:

- Ter um orientador docente da Unidade Técnico-Científica de Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda, assim como um supervisor direto na empresa;
- Ter um plano de desenvolvimento aprovado pelo diretor do curso;

1.1 Instituição de acolhimento

A Ideias Soberbas, agência de comunicação fundada a 28 de Janeiro de 2000, tem como principal objetivo a criação e implementação de páginas de Internet totalmente personalizadas e soluções de comunicação empresarial, publicidade e meios de divulgação.

1.2 Motivação

A principal motivação para o desenvolvimento deste projeto é a possibilidade em contribuir para melhoria da gestão de reservas da Casa do Pastor, quer a nível do cliente, quer a nível de quem trata destas e para fazer com que a criação de uma reserva se torne mais acessível. Também a possibilidade de poder trabalhar com novas tecnologias com as quais nunca tive oportunidade de trabalhar, principalmente a linguagem de programação PHP e gestor de bases de dados MySQL.

Para além das motivações pessoais, existiram ainda motivações a nível da empresa, nomeadamente a nível económico.

1.3 Solução

A solução proposta consiste no desenvolvimento, implementação e teste de uma aplicação web com um BackOffice para gestão dinâmica de reservas com as diversas informações, criadas posteriormente pelo cliente no respetivo FrontOffice.

Esta solução traz várias vantagens, quer para o cliente, quer para os gestores da Casa do Pastor, uma vez que todos podem agora criar/editar reservas sem terem que se encontrar cara a cara ou contactarem-se através de telefones/telemóveis.

1.4 Contribuição

A contribuição principal deste trabalho é o desenvolvimento, implementação e teste de uma aplicação web que auxilie a criação e gestão de reservas, de forma acessível e a evitar perdas desnecessárias de dados e uma má gestão do tempo.

1.5 Estrutura do documento

O documento compreende cinco capítulos para além do presente capítulo, e está organizado da seguinte forma:

- No segundo capítulo é descrita a definição do problema e objetivos previstos - apresentação do problema e dos objetivos do projeto;
- No terceiro capítulo é descrita a metodologia e a calendarização do projeto - apresentação das metodologias usadas no desenvolvimento do projeto realizado e a sua calendarização;
- No quarto capítulo são descritas todas as tecnologias utilizadas utilizadas na implementação do projeto;
- No quinto capítulo é descrito de forma detalhada todo o trabalho realizado, desde a base de dados até à aplicação web;
- No sexto capítulo são apresentadas as considerações finais sobre o trabalho desenvolvido e possível trabalho futuro;

Capítulo 2

Definição do problema e objetivos previstos

2.1 Definição do problema

Desenvolver uma aplicação web, permitindo aos clientes da Casa do Pastor efetuar reservas de casas. A aplicação deve garantir algumas funcionalidades, tais como: consultar o calendário das casas e criar a reserva com todos os dados requeridos. Desenvolver uma plataforma de gestão de acesso restrito aos gestores e ao administrador, onde serão geridos os dados das reservas, casas, atividades e gestores da Casa do Pastor. Deve ser desenvolvimento um módulo que permita consultar o histórico de todas as operações efetuadas sobre os dados da aplicação web pelos gestores.

Os principais problemas a resolver são os seguintes:

- Como obter todos os dados da Casa do Pastor;
- Como integrar a aplicação web num web site já existente;
- Como integrar toda a informação com a reserva:
  - Atividades;
  - Casas;
  - Dados do Cliente;
- Como apresentar ao cliente apenas as casas disponíveis aquando da criação da reserva;
- Como preservar todos os dados sem qualquer perda de informação;

2.2 Objetivos previstos

Segue-se a seguir a lista dos principais objetivos definidos no início do projeto, sendo que estes poderão ser ou não implementados, dependendo de vários fatores.

- Desenvolver uma aplicação web de suporte à criação e gestão de reservas;

- Integrar a informação da Casa do Pastor na aplicação web;
- Relacionar os dados dos clientes e das suas reservas para fins estatísticos;
- Integrar a aplicação no web site já existente;
- Criar um histórico de todas as operações feitas pelos gestores e administrador;

Capítulo 3

Metodologia e resultados esperados

3.1 Metodologia

A metodologia utilizada para desenvolver, implementar, testar e colocar online a aplicação web foi a seguinte:

1. Utilizar Visio para definir a arquitetura da aplicação;
2. Utilizar MySQL e phpMyAdmin para a criação da base de dados;
3. Utilizar CSS para a criação das folhas de estilo;
4. Utilizar HTML para a criação da estrutura base das páginas;
5. Utilizar Javascript para a criação de cookies e reencaminhamentos entre páginas;
6. Utilizar PHP para criar as páginas dinâmicas;
7. Utilizar Dreamweaver para a criação da aplicação;
8. Utilizar Fireworks para a criação e edição de imagens
9. Utilizar XAMPP para realizar testes e análise de eficiência e fiabilidade da aplicação na própria máquina;
10. Utilizar Core FTP LE para a transferência da aplicação para o servidor online;
11. Utilizar TeXnicCenter para a documentação do projeto num relatório;

Para reduzir e agilizar o número de reuniões com o supervisor na empresa, foi acordado que periodicamente o estagiário deveria apresentar-lhe o trabalho desenvolvido. Permitindo ao supervisor acompanhar a evolução da aplicação web e garantir que o projeto final corresponda ao que foi solicitado pela empresa.

3.2 Descrição das tarefas

As principais tarefas foram:

- Tarefa 1 — Análise dos requisitos;
- Tarefa 2 — Obtenção de dados da Casa do Pastor:
  1. Estudo sobre casas existentes e respetivas características;
  2. Estudo sobre eventuais descontos para as reservas;
- Tarefa 3 — Aquisição e estudo das várias API's necessárias para a criação do projeto;
- Tarefa 4 — Criação da base de dados (MySQL, phpMyAdmin);
- Tarefa 5 — Criação das folhas de estilo (CSS);
- Tarefa 6 — Criação dos módulos de programação (PHP, HTML, Javascript);
- Tarefa 7 — Testes da aplicação;
- Tarefa 8 — Colocação e divulgação online na aplicação web;
- Tarefa 9 — Elaboração do relatório.

O agendamento das tarefas é apresentado na figura 3.1.

ID	Nome	Data inicial	Data final	Duração
0	Tarefa 1 - Análise dos requisitos	12-03-2012	23-03-2012	10
3	Tarefa 2 - Obtenção de dados da Casa do Pastor	26-03-2012	06-04-2012	10
4	Tarefa 3 - Aquisição e estudo das várias API's para o projeto	09-04-2012	27-04-2012	15
5	Tarefa 4 - Criação da base de dados (MySQL, phpMyAdmin)	30-04-2012	21-08-2012	82
6	Tarefa 5 - Criação das folhas de estilo (CSS)	10-05-2012	21-08-2012	74
9	Tarefa 6 - Criação dos módulos de programação (PHP, HTML, Javascript)	22-05-2012	21-08-2012	66
8	Tarefa 7 - Testes da aplicação	30-04-2012	10-09-2012	96
10	Tarefa 8 - Colocação e divulgação online na aplicação web	11-09-2012	28-09-2012	14
11	Tarefa 9 - Elaboração do relatório	15-06-2012	10-10-2012	84

Figura 3.1: Tarefas.

O respetivo Mapa de Gantt é apresentado na figura 3.2.

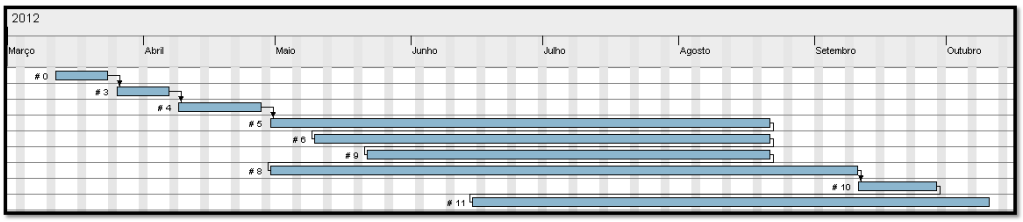

Figura 3.2: Mapa de Gant.

3.3 Resultados esperados

No final do projeto espera-se que a aplicação web esteja terminada e pronta a ser usada pelos utilizadores finais, permitindo ao cliente:

1. Consultar o calendário das casas para saber quando estão livres e ocupadas;
2. Efetuar reservas de casas;
3. Receber informações sobre o estado das reservas na sua conta de correio eletrónico;

Na plataforma de gestão online os gestores deverão poder:

1. Consultar o calendário das casas;
2. Editar, eliminar e inserir casas;
3. Editar, eliminar e inserir atividades;
4. Editar, eliminar e inserir clientes;
5. Editar, eliminar e inserir reservas:
   (a) Editar dados da reserva;
   (b) Editar dados do respetivo cliente;
   (c) Editar, eliminar e inserir casas da reserva;
   (d) Editar, eliminar e inserir atividades da reserva;
   (e) Editar o estado da reserva;
   (f) Editar o desconto da reserva;
6. Editar os seus dados pessoais;

Ainda na plataforma de gestão, os administradores poderão, para além de tudo o que os gestores podem:

1. Consultar o histórico de todas as operações feitas pelos gestores por data;
2. Editar, eliminar e inserir contas de gestores;

Capítulo 4

Tecnologias utilizadas

Para a realização da aplicação web é necessária a utilização de diversas tecnologias da web, descritas nas secções seguintes. Estas tecnologias são padronizadas pela consórcio internacional W3C com cerca de 300 membros, constituída por empresas, órgãos governamentais e organizações independentes, e que visa desenvolver padrões para a criação e a interpretação de conteúdos para a Web [4].

O conjunto de tecnologias abertas, denominadas por *Open Web Platform*, padronizadas pelo consórcio W3C permitem que qualquer pessoa possa utilizá-las para desenvolver qualquer tipo de aplicação sem ter de pedir permissão ou obter qualquer licença do consórcio W3C.

4.1 Tecnologias web

O desenvolvimento de páginas web pode ser realizado baseado num modelo de três camadas: estrutura, apresentação e comportamento [13]:

1. Estrutura — Também chamada camada de base, são produzidos os conteúdos e a sua estrutura. Com apenas esta camada os utilizadores podem ver a página através de qualquer browser;

2. Apresentação — É definida a aparência do conteúdo na página, incluindo tipo de letra, cor do texto, alinhamento do texto e das imagens, entre muitos outros aspetos estéticos;

3. Comportamento — É definida a interação com o utilizador e a página, como por exemplo os botões ou links que mudam de cor ou de imagem quando o utilizador passa o cursor por cima deles;

A figura 4.1, ilustra diversas tecnologias web agrupadas em três acamadas para o desenvolvimento de páginas web.

A separação do código HTML, CSS e JavaScript e armazenamento em ficheiros separados apresenta as seguintes vantagens:

- Permite a reutilização;

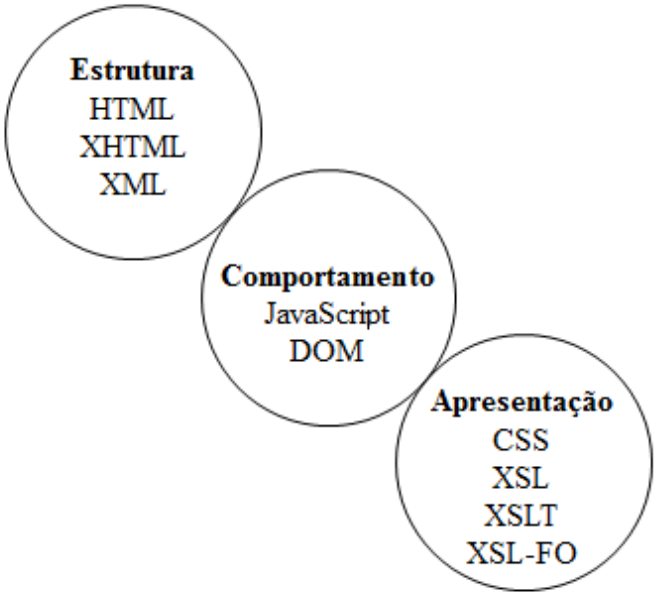

Figura 4.1: As três camadas das tecnologias web.

- Fácil de localizar;
- Fácil de manter;

4.1.1 HTML

A linguagem utilizada para publicação de informação na World Wide Web é o HTML. Esta linguagem de marcação permite escrever documentos HTML com o fim de produzir páginas web [3], que, basicamente, se trata de um conjunto de marcas (*tags*) que servem para definir a forma como se apresentará o conteúdo da página web. Tim Berners-Lee criou o HTML original e a primeira versão foi esboçada por ele e Dan Connolly, e publicada em 1993 na IETF.

O objetivo da criação da linguagem HTML foi de divulgação de informação, no entanto, nunca se pensou que a Internet chegaria a ser uma área com carácter multimédia, sendo que o HTML acabou por ser criado sem capacidades de dar respostas a todos os possíveis usos que lhe poderiam dar. No entanto, com o passar do tempo foram sendo incorporadas modificações, as quais são hoje os standards da linguagem. A última versão do HTML a ser padronizada pelo consorcio W3C foi o padrão HTML 4.01 e ocorreu em dezembro de 1999 [6]. Desde então foram realizados muitos esforços para padronizar a versão HTML5.

A primeira especificação do HTML5 foi apresentada no início de 2008, e está desde então em fase de esboço, e prevê-se que a versão final seja publicada no final de 2014. O HTML5 tem novas funcionalidades baseadas no HTML, CSS, DOM, e JavaScript; reduz a necessidade de utilização de *plugins* externos (Flash, entre outros); melhor

controlo de erros; mais marcas para substituir *scripts*, é bastante independente dos dispositivos (computador, telemóvel, tablets); permite armazenamento local de dados.

Steve Jobs em abril de 2010 publicou uma carta pública intitulada "Reflexões sobre o Adobe Flash", onde concluía que o HTML5 tornaria o Adobe Flash desnecessário, quer para assistir vídeo, quer para exibir qualquer conteúdo web [11]. Em novembro de 2011 a empresa de software Adobe anunciou a interrupção do desenvolvimento de Flash para dispositivos móveis e redirecionou os seus esforços para o desenvolvimento de ferramentas utilizando HTML5 [1].

Um documento HTML é uma hierarquia de elementos a partir de uma raiz, elemento *html*. Na primeira linha do documento, e antes do elemento *html*, existe uma declaração que contem informações sobre a versão do documento HTML (<!DOCTYPE html ...>). O elemento *html* é constituído por duas componentes:

1. Uma secção de cabeçalho declarativa — Delimitado pelo elemento `head`;
2. Um corpo, que contém o conteúdo real do documento — Delimitado pelo elemento `body`;

Na listagem 1 é ilustrado um exemplo de um documento HTML 4.01 Strict. A primeira linha contem informações sobre a versão do HTML. Na segunda linha é iniciado o documento HTML; na linha 4 é indicado o titulo do documento e na linha 6 é carregada a folha de estilos que edita o documento; na linha 7 é iniciado o corpo do documento; na linha 9 é iniciada a tabela com a ilustração das casas e dos preços. A linha 20 representa uma marca de parágrafo.

A página web resultante do documento HTML da listagem 1 é apresentado na figura 4.2. À esquerda no browser Internet Explorer e à direita no browser Google Chrome.

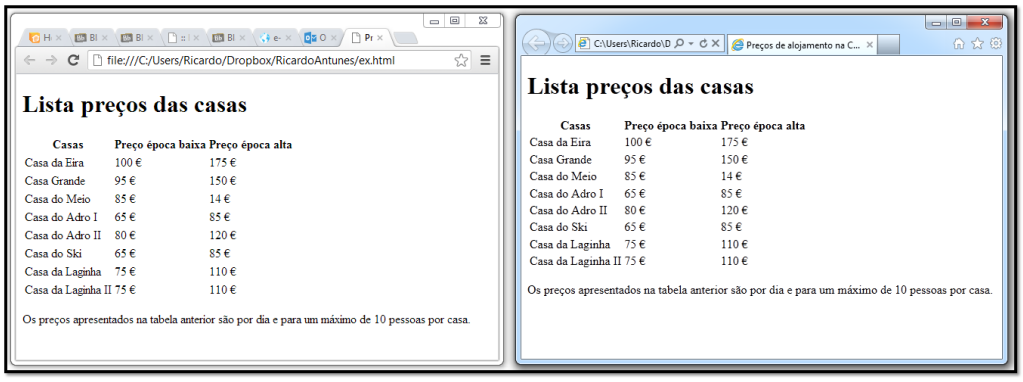

Figura 4.2: Página web. Preços de alojamento na Casa do Pastor.

Esta tecnologia foi utilizada para criar a estrutura geral de todas as páginas, tais como a inserção de tabelas, imagens e informação.

Listagem 1 Exemplo de um documento HTML.

```
 1:  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
     %"http://www.w3.org/TR/html4/strict.dtd">
 2:  <html>
 3:    <head>
 4:      <title>Preços de alojamento na Casa do Pastor</title>
 5:      <!-- <link rel='stylesheet' type='text/css' href='css1.css'> -->
 6:    </head>
 7:    <body>
 8:      <h1>Lista preços das casas</h1>
 9:      <table>
10:        <tr><th>Casas</th><th>Preço época baixa</th><th>
           %Preço época alta</th></tr>
11:        <tr><td>Casa da Eira</td><td>100 </td><td>175 </td></tr>
12:        <tr><td>Casa Grande</td><td>95 </td><td>150 </td></tr>
13:        <tr><td>Casa do Meio</td><td>85 </td><td>14 </td></tr>
14:        <tr><td>Casa do Adro I</td><td>65 </td><td>85 </td></tr>
15:        <tr><td>Casa do Adro II</td><td>80 </td><td>120 </td></tr>
16:        <tr><td>Casa do Ski</td><td>65 </td><td>85 </td></tr>
17:        <tr><td>Casa da Laginha</td><td>75 </td><td>110 </td></tr>
18:        <tr><td>Casa da Laginha II</td><td>75 </td><td>110 </td></tr>
19:      </table>
20:      <p>Os preços apresentados na tabela anterior são por dia e para um
         máximo de 10 pessoas por casa.</p>
21:    </body>
22:  </html>
```

4.1.2 XML

O XML (eXtensible Markup Language) é uma linguagem de marcação para definir linguagens de marcação específicas [5]. O objetivo foi criar uma linguagem de marcação que combinasse a flexibilidade da SGML com a simplicidade da HTML. A sua filosofia foi incorporada seguindo vários princípios, tais como:

- Separação do conteúdo da formatação;
- Possibilidade de criação de *tags* sem qualquer limitação;
- Utilização de ficheiros para validação da estrutura (DTDs e XML Schema);
- Integração com várias bases de dados;
- Concentração na estrutura da informação e não na sua aparência;

Principais vantagens do XML:

- É um padrão aberto — Facilidade para converter para formatos proprietários;
- É baseado em texto Unicode — Fácil de escrever e ler, fácil de processar, menos incompatibilidades;
- Promover a separação entre estrutura, conteúdo e apresentação;
- Pode representar as estruturas de dados relevantes da computação (listas, registos, árvores);
- É auto-documentado (DTDs e XML Schemas), sendo que o próprio formato descreve a sua estrutura e nomes de campos;
- É editável — Devido à sua popularidade nos dias de hoje, com diferentes níveis de automação, em qualquer ambiente;
- É a ferramenta mais comum para as transmissões de dados entre todos os tipos de aplicações;

No entanto, devido à sintaxe do XML ser redundante a representação de uma estrutura de dados pode tornar-se relativamente grande em relação a outras representações. Desta forma, a redundância pode afetar a eficiência das aplicações que utilizam o XML para armazenamento, transmissão e processamento e os seu custo associado.

A listagem 2 apresenta um exemplo de código XML. A primeira linha é obrigatória e deve ser escrita na primeira linha do ficheiro de texto. A referida linha indica que o documento é XML versão 1.0. O utilizador pode definir as marcas e os atributos que entender para marcar os seus dados. Por exemplo, a marca `nome`, marca a palavra "Ricardo"como sendo um nome.

Listagem 2 Exemplo de código XML.

```
 1:  <?xml version="1.0" ?>
 2:  <cartao-simples>
 3:    <foto href="ricardo.gif" />
 4:    <nome>Ricardo</nome>
 5:    <apelido>Antunes</apelido>
 6:    <endereco>
 7:      <rua>Lugar do Bacelo, Bloco B, 2º direito.</rua>
 8:      <codigo_postal>6300-000 - Guarda</codigo_postal>
 9:    </endereco>
10:    <email>antenas963@hotmail.com></email>
11:    <telefone tipo="trabalho" >
12:      <extensao>120</extensao>
13:      <numero>271084703</numero>
14:    </telefone>
15:    <telefone tipo="telemovel" >
16:      <numero>962466767</numero>
17:    </telefone>
18:  </cartao-simples>
```

4.1.3 XHTML

O XHTML (eXtensible Hypertext Markup Language) é uma reformulação da linguagem de marcação HTML, baseada em XML, que combina as tags de marcação HTML com as regras da XML [3]. Este processo de padronização tem em vista a exibição de páginas web em diversos dispositivos, com a intenção de melhorar a acessibilidade. Apesar de não existirem grandes diferenças entre o HTML e o XHTML, o XHTML consegue ser intercetado por qualquer dispositivo, independentemente da plataforma utilizada, o que não é conseguido pelo HTML, visto que as marcações do XHTML possuem sentido semântico para as máquinas.

4.1.4 CSS

As folhas de estilo CSS, siglas de Cascading Style Sheets, são uma linguagem de estilos utilizada para definir a apresentação de documentos escritos em linguagens como HTML ou XML [3], provendo assim a separação entre o formato e o conteúdo de um documento. Assim sendo, em vez de se colocar a formatação dentro do ficheiro do documento, é criada uma ligação para uma folha de estilos armazenada num ficheiro externo com extensão ".css", e todo o documento fica com as respetivas formatações. O principal benefício das folhas de estilo é que quando se pretender alterar a aparência do documento, basta alterar a folha de estilos sem ter que mudar nada no conteúdo.

A listagem 3 apresenta um exemplo de uma folha de estilos. Da linha 1 à linha 4 estão as formatações das tabelas; da linha 5 à linha 8 as formatações da borda (`th`

representa uma célula com letras a negrito e centrada por padrão e `td` representa uma célula normal), sendo que na linha 9 é atribuída à linha `th` e na linha 10 a cor da linha `td`. Da linha 11 à linha 15 são as formatações da marca `h1` com tamanho da fonte 18 pixeis, cor castanha e letras maiúsculas.

Listagem 3 Exemplo de uma folha de estilos.

```
 1:  table {
 2:    border-collapse: collapse;
 3:    border: solid 1px blue;
 4:  }
 5:  th, td {
 6:    border: solid 1px black;
 7:    padding: 4px;
 8:  }
 9:  th {color:blue; }
10:  td {text-align:center;}
11:  h1 {
12:    font-size: 18pt;
13:    color: brown;
14:    font-variant:small-caps;
15:  }
```

A página web resultante do documento HTML da listagem 4 formatada com a folha de estilos da listagem 3, é apresentado na figura 4.3. À esquerda no browser Internet Explorer e à direita no browser Google Chrome. Nota-se que a diferença entre a listagem 1 e a listagem 4 é apenas a linha 5, que permite que (no caso da segunda) a folha de estilos seja aplicada a este documento.

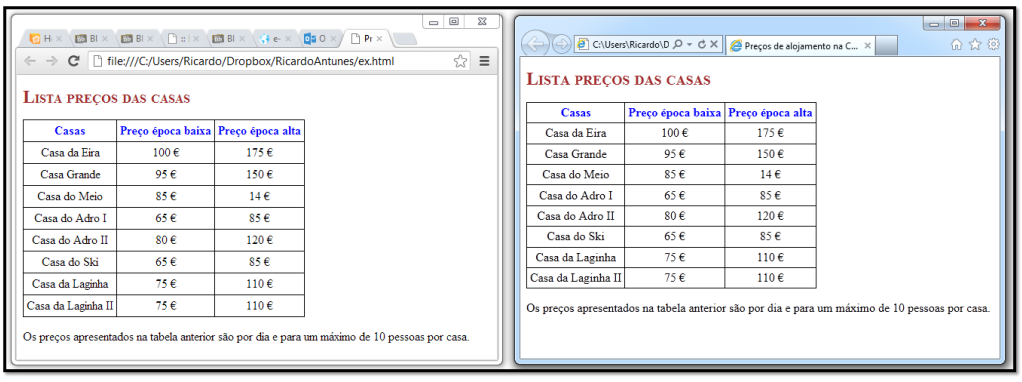

Figura 4.3: Página web formatada com CSS. Preços de alojamento na Casa do Pastor.

Esta tecnologia foi utilizada para formatar todas as páginas web. Dentro dessas formatações encontram-se, principalmente, a formatação do fundo das páginas e o tamanho, tipo e cor de letras.

Listagem 4 Exemplo de um documento HTML.

```
 1:  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
     %"http://www.w3.org/TR/html4/strict.dtd">
 2:  <html>
 3:    <head>
 4:      <title>Preços de alojamento na Casa do Pastor</title>
 5:      <link rel='stylesheet' type='text/css' href='css1.css'>
 6:    </head>
 7:    <body>
 8:      <h1>Lista preços das casas</h1>
 9:      <table>
10:        <tr><th>Casas</th><th>Preço época baixa</th><th>
           %Preço época alta</th></tr>
11:        <tr><td>Casa da Eira</td><td>100 </td><td>175 </td></tr>
12:        <tr><td>Casa Grande</td><td>95 </td><td>150 </td></tr>
13:        <tr><td>Casa do Meio</td><td>85 </td><td>14 </td></tr>
14:        <tr><td>Casa do Adro I</td><td>65 </td><td>85 </td></tr>
15:        <tr><td>Casa do Adro II</td><td>80 </td><td>120 </td></tr>
16:        <tr><td>Casa do Ski</td><td>65 </td><td>85 </td></tr>
17:        <tr><td>Casa da Laginha</td><td>75 </td><td>110 </td></tr>
18:        <tr><td>Casa da Laginha II</td><td>75 </td><td>110 </td></tr>
19:      </table>
20:      <p>Os preços apresentados na tabela anterior são por dia e para um
         máximo de 10 pessoas por casa.</p>
21:    </body>
22:  </html>
```

4.1.5 Javascript

A linguagem de script JavaScript foi criado originalmente por Brendan Eich da Netscape [3], com o primeiro nome de Mocha, e posteriormente mudado para LiveScript e atualmente JavaScript, é uma linguagem de script baseada em ECMAScript e é atualmente a principal linguagem de programação do lado do cliente em navegadores web. LiveScript foi o nome oficial da linguagem quando foi lançada pela primeira vez na versão beta do navegador Netscape 2.0 em Setembro de 1995, mas viu o seu nome alterado para JavaScript em Dezembro do mesmo ano num conjunto com a Sun Microsystems. Esta foi concebida para ser uma linguagem script orientada a objetos baseada em protótipos; possui suporte à programação funcional e apresenta recursos como funções de alto nível indisponíveis em linguagens como o Java e o C++.

A listagem 5 apresenta o exemplo de uma função JavaScript, que tem também algum código PHP incluído. A função recebe os parâmetros nome do cookie, respetivo valor e número de dias (cookieName,cookieValue,nDays), que são de seguida utilizados para criar um cookie com esses mesmos parâmetros — `document.cookie = cookieName+"="+escape(cookieValue)+"; expires="+expire.toGMTString()+"; path=/"+"; domain=.<?php echo $_SERVER['HTTP_HOST']; ?>"`.

Listagem 5 Exemplo de uma função JavaScript.

```
function SetCookie(cookieName,cookieValue,nDays) {
  var today = new Date();
  var expire = new Date();
  if (nDays==null || nDays==0) nDays=1;
  expire.setTime(today.getTime() + 3600000*24*nDays);
  document.cookie = cookieName+"="+escape(cookieValue)+";
  expires="+expire.toGMTString()+"; path=/"+"; domain=.
<?php echo $_SERVER['HTTP_HOST']; ?>";
}
```

A tecnologia Javascript foi utilizada no presente projeto essencialmente para a criação de cookies e efetuar o reencaminhamento entre páginas.

4.1.6 SQL

A história da linguagem SQL começa em Junho e 1970 com E. F. Codd [7], que propôs um modelo que é ainda hoje considerado a base de trabalho para qualquer sistema de gestão de base de dados relacional. A primeira implementação da linguagem SEQUEL foi realizada pela IBM e tinha por objetivo a implementação do modelo de Codd. A evolução desta linguagem deu por fim origem ao SQL. Nos dias de hoje SQL é considerada um standard dos sistemas gestores de bases de dados relacionais, sendo que todos os fabricantes a integram nos seus produtos.

O SQL é dividido em cinco níveis, que serão explicados nas seguintes secções.

4.1.6.1 DDL — Linguagem de Definição de Dados

Uma DDL permite ao utilizador definir tabelas novas e elementos associados. Os comandos básicos da DDL são o "Create", o "Drop"e o "Alter", que servem para criar, eliminar ou alterar, respetivamente, por exemplo, tabelas.

A listagem 6 apresenta um exemplo prático de uma DDL utilizada para a criação da tabela "casas"na base de dados.

Listagem 6 Exemplo prático de uma DDL.

```
CREATE TABLE IF NOT EXISTS ‘casas‘ (
  ‘ID_Casa‘ int(2) NOT NULL AUTO_INCREMENT,
  ‘Nome‘ varchar(50) NOT NULL,
  ‘Tipologia‘ varchar(10) NOT NULL,
  ‘Capacidade‘ int(2) NOT NULL,
  ‘Preco_Epoca_Baixa‘ int(3) NOT NULL,
  ‘Preco_Epoca_Alta‘ int(3) NOT NULL,
  PRIMARY KEY (‘ID_Casa‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=9 ;
```

4.1.6.2 DML — Linguagem de Manipulação de Dados

DML é utilizada para realizar inclusões, consultas, alterações e exclusões de dados presentes em registos. Os comandos utilizados são o "Select", "Update", "Insert"e "Delete", que servem, respetivamente, para selecionar, atualizar, inserir ou eliminar dados de uma tabela.

A listagem 7 apresenta um exemplo prático de uma DML utilizada para atualizar os dados da tabela "casas".

Listagem 7 Exemplo prático de uma DML.

```
$sql4="UPDATE ‘casasreserva‘ SET ‘N_Pessoas‘ = ’$npessoas’,
‘N_Camas_Extra‘ = ’$ncamas’, ‘N_Bercos‘ = ’$nbercos’,
‘Data_Entrada‘ = ’$dataentrada’, ‘Data_Saida‘ = ’$datasaida’,
‘Preco_Casa‘ = ’$preco’ WHERE ID_Reserva = ’$IDReserva’
AND ID_Casa = ’$idcasa’";
```

4.1.6.3 DCL — Linguagem de Controlo de Dados

DCL é utilizado para controlar os aspetos de autorização de dados e licenças de utilizadores de forma a controlar quem tem acesso para ver ou manipular os dados dentro da base de dados. Os comandos utilizados são o "Grant"e o "Revoke", que servem, respetivamente, para autorizar a execução de operações ou para remover essa mesma capacidade de executar operações.

No presente projeto isto não foi utilizado, uma vez que todas as alterações feitas às tabelas da base de dados têm validações.

4.1.6.4 DTL — Linguagem de Transação de Dados

DTL serve para proceder a transações na base de dados, tipicamente utilizado em bancos. Os comandos utilizados são o "Begin Work"para dar início à transação, o "Commit"para enviar todos os dados da mudança de forma permanente e o "Rollback"que faz com que todas as as mudanças nos dados existentes desde o último "Commit"ou "Rollback"sejam anuladas.

Tal como as DCL, não foram usadas quaisquer transações para este projeto.

4.1.6.5 DQL — Linguagem de Consulta de Dados

DQL tem apenas um comando, o `SELECT`, que permite apenas ao utilizador fazer uma consulta a uma ou várias tabelas. Esta operação faz igualmente parte da DML.

A listagem 8 apresenta um exemplo prático de uma DML que permite obter os campos `Nome, Capacidade, Preco_Epoca_Alta, Preco_Epoca_Baixa` de todos os registos da tabela `casas`.

Listagem 8 Exemplo prático de uma DQL.

```
%$sqlca="select * from casas, casasreserva where
%casasreserva.ID_Reserva = " . $IDReserva . " and
%casas.ID_Casa = casasreserva.ID_Casa";
SELECT Nome, Capacidade, Preco\_Epoca\_Alta, Preco\_Epoca\_Baixa
FROM casas;
```

No presente projeto, a tecnologia SQL foi utilizada para aceder a dados presentes na Base de dados, bem como para editar, eliminar ou inserir novos registos nessa mesma base de dados.

4.1.7 Linguagem de programação PHP

A linguagem de programação PHP foi criado em 1995 por Rasmus Lerdorf [12], inicialmente denominado por "Personal Home Page"e agora por "Hypertext Preprocessor", era originalmente usada apenas para o desenvolvimento de aplicações presentes e atuantes do lado do servidor, capazes de gerar conteúdo dinâmico da World Wide Web. O código é interpretado no lado do servidor pelo módulo PHP, podendo este aceder a bases de dados, conexões de rede, entre outros, para criar a página final, apresentada posteriormente ao cliente. Ao longo dos anos a evolução da linguagem foi notória, passando a oferecer funcionalidades em linha de comando. É possível utilizar o PHP na maioria dos sistemas operativos de forma gratuita. Este é o concorrente direto da tecnologia ASP, pertencente à Microsoft, e é utilizado em aplicações bem conhecidas, como o Facebook, Joomla, WordPress ou Magento.

A listagem 9 apresenta um excerto de código PHP da aplicação web desenvolvida. Inicialmente a aplicação verifica se existe alguma mensagem guardada nas variáveis de sessão — `if (isset($_Request['Mensagem']))`; se existir, é visto se essa mensagem não é nula — `if ($_REQUEST['Mensagem'] != null)`, e, caso não seja nula, é guardada na variável '`$Mensagem`' o seu valor, caso contrário é guardado o conjunto vazio — `$Mensagem = ''`. No caso de não existir qualquer mensagem guardada nas variáveis de sessão, à variável '`$Mensagem`' é-lhe atribuído o conjunto vazio também.

Listagem 9 Exemplo de código PHP.

```
 1:  if (isset($_REQUEST['Mensagem']))
 2:   {
 3:     if ($_REQUEST['Mensagem'] != null)
 4:     {
 5:       $Mensagem = $_REQUEST['Mensagem'];
 6:     }
 7:     else
 8:     {
 9:       $Mensagem = '';
10:     }
11:   }
12:   else
13:   {
14:    $Mensagem = '';
15:   }
```

A tecnologia PHP foi utilizada no projeto essencialmente como plataforma de apoio à visualização, edição, eliminação e inserção de dados na base de dados. Para além disso, foi ainda utilizada para a leitura de cookies, para guardar e ler dados com recurso aos métodos '`$GET`', '`$POST`' e também o '`$REQUEST`'.

4.2 Software utilizado

No desenvolvimento do projeto foram utilizadas alguns aplicações, apresentados nas secções seguintes.

4.2.1 Adobe Systems

A Adobe System, fundada em Dezembro de 1982 por John Warnock e Charles Geschke, é uma companhia norte-americana que desenvolve programas de computador, sediada em San Jose, Califórnia. A Adobe produz programas para, por exemplo, desenvolvimento de páginas web (Dreamweaver), criação e edição de imagens (Photoshop, Fireworks), criação de animações (Flash), entre outros.

Para o presente projeto, foram utilizados dois programas da distribuição Adobe: Dreamweaver e Fireworks, que serão explicados de seguida.

4.2.1.1 Adobe Dreamweaver

O Adobe Dreamweaver (antigo Macromedia Dreamweaver) é um software de desenvolvimento de aplicações web, criado pela Macromedia (adquirida posteriormente pela Adobe Systems), e que está atualmente na versão CS6. As suas versões inicias serviam apenas como editor HTML WYSIWYG, que significa "'What You See Is Wath You Get"', mas as versões mais recentes incorporam um suporte a várias tecnologias web, tais como o XHTML, CSS, JavaScript, PHP, ASP.NET, JSP, entre outras linguagens do lado do servidor.

A versão utilizada para o projeto foi a versão CS5, lançado a 12 de Abril de 2010.

Foi com este software que toda a aplicação foi desenvolvida.

4.2.1.2 Adobe Fireworks

O Adobe Fireworks, tal como o Adobe Dreamweaver, foi desenvolvido pela Macromedia, e posteriormente adquirido pela Adobe. O Fireworks é um editor de imagens, em que as suas funcionalidades focam a publicação gráfica na Internet, incluindo assim suporte a GIF animado e PNG, para além de possuir uma ótima compressão de imagens. Foi a partir da versão MS que ganhou integração com outros produtos da mesma linha, tais como o Dreamweaver ou o Flash.

A versão utilizada para o projeto foi, tal como o Dreamweaver, a versão CS5, lançada, igualmente, em 2010.

O Fireworks foi utilizado para a criação e edição de imagens a utilizar em todas as páginas web.

4.2.2 Apache Friends - XAMPP

Uma vez que o projeto foi feito em PHP e este é executado no servidor, para o poder testar na minha máquina tive que utilizar o software XAMPP, que é um servidor independente de plataforma, de uso livre, e que consiste principalmente na base de dados MySQL, no servidos web Apache e os interpretadores para linguagens de script (PHP e Perl). O seu nome provem da abreviação de X (para qualquer sistema operativo), Apache, MySQL, PHP, Perl.

4.2.2.1 XAMPP control

O interface inicial do XAMPP é o visto na 4.4. É aqui que os serviços (Apache, MySQL, etc) são inicializados (clicando em "Start"), e a partir do qual temos acesso ás configurações desses mesmos serviços (clicando em "Config"). O Apache é o simulador de servidor, que foi utilizado para poder correr a aplicação na minha máquina, e utiliza por norma o porto 80. O MySQL é o serviço de gestão de bases de dados e usa o porto 3306. Clicando nos botões "Admin"vamos para a página de administração do Apache ou do MySQL.

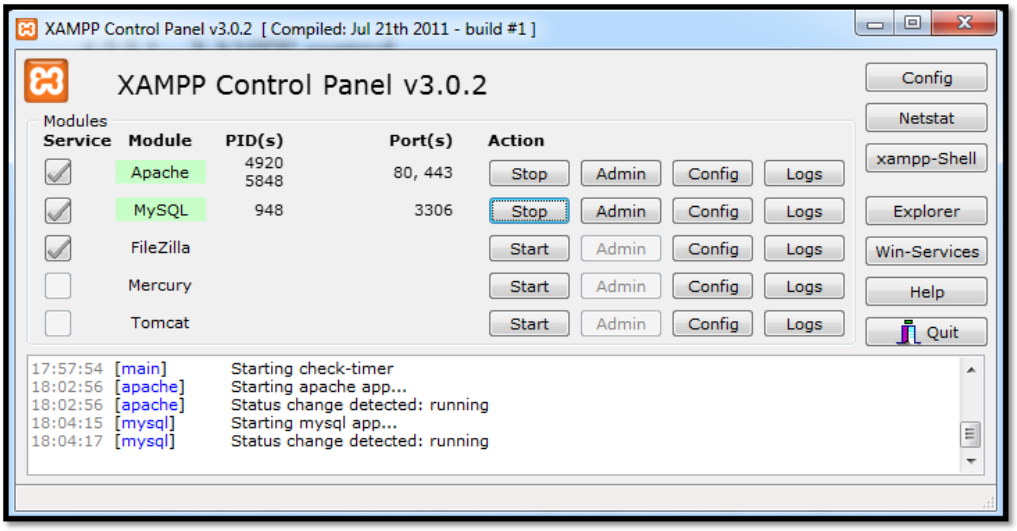

Figura 4.4: Interface inicial do XAMPP.

Nos ficheiros de configuração é possível alterar as definições e todas as propriedades de cada ferramenta, no entanto é preciso saber mexer nestes ficheiros e saber bem o que se pretende.

O botão "Admin"à frente do "Apache"levanos para a página mostrada na figura 4.5, enquanto que o "Admin"do "MySQL"nos leva para o phpMyAdmin.

Figura 4.5: Página de administração do Apache.

4.2.2.2 phpMyAdmin

O phpMyAdmin é o gestor de base de dados MySQL, que tem o interface visto na figura 4.6. Do lado esquerdo podemos ver todas as bases de dados criadas, na caixa de texto do lado direito podemos ver as características do MySQL e nas duas caixas de texto centrais podemos ver as definições gerais do interface (como por

exemplo a linguagem e o tamanho da fonte). Podemos ainda ver no menu na parte superior da janela um ícone que nos permitem criar novas base de dados ("Banco de Dados"), outro que nos permite executar comandos diretos de SQL ("SQL"), dois que nos permitem importar e exportar bases de dados ("Import"e "Export", respetivamente), entre outros.

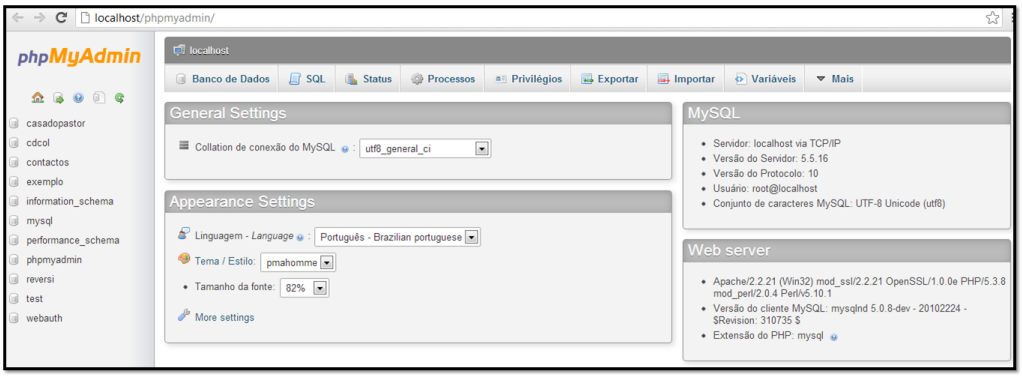

Figura 4.6: Interface principal do phpMyAdmin.

Clicando na base de dados "casadopastor", do lado esquerdo da janela, vai-nos abrir uma página com todas as tabelas desta base de dados, onde podemos consultar a estrutura desta, os seus dados, ou então inserir novas tabelas. Clicando no nome de qualquer uma delas, vemos a estrutura dos seus campos, tal como podemos ver na figura 4.7. É possível alterar um campo (clicando em "Alterar"), eliminá-lo ("Eliminar"), ou então inserir uma nova coluna, na parte inferior da figura, no botão "Executar".

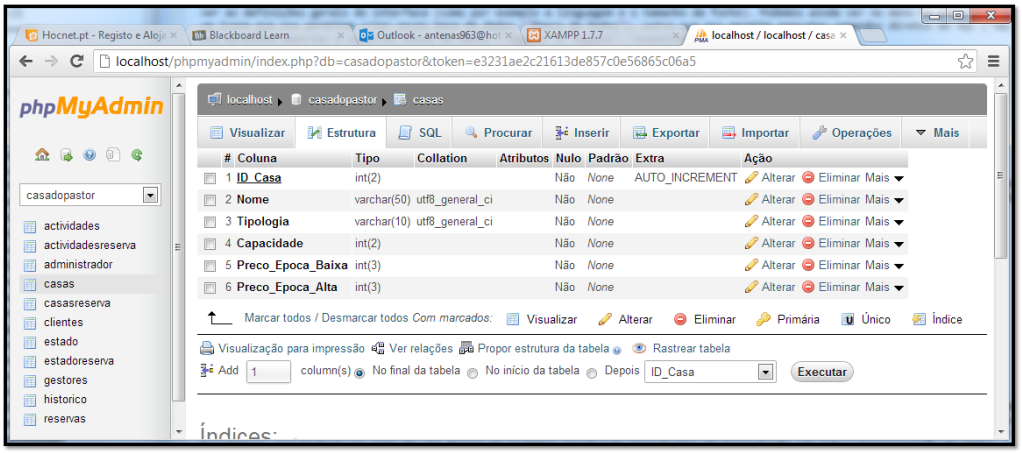

Figura 4.7: Exemplo de estrutura de tabela..

Para colocar a base de dados num servidor é necessário:

- Exportar a base de dados existente na minha máquina;
- Importá-la para a nova base de dados do servidor;

Para exportar é apenas necessário selecionar a nossa base de dados, clicar no menu superior em "Exportar", selecionar o formato pretendido (no meu caso, escolhi formato SQL), escolher as tabelas pretendidas (normalmente serão todas), escolher os dados que queremos (apenas tabelas, apenas dados, ou tabela com todos os dados), clicar em "Executar"e será recebido um ficheiro SQL (neste caso) com toda a estrutura da base de dados. Estas são as definições principais, no entanto, existem mais definições na exportação da base de dados que podem ser alterados.

De seguida, para importar a base de dados para o servidor, é apenas necessário importar o ficheiro SQL, clicando em "Importar", de seguida em "Escolher ficheiro", procurar o ficheiro SQL, e clicar em "Executar". Depois disto, toda a estrutura e dados da base de dados exportada anteriormente, ficará disponível na nova base de dados.

No presente projeto, este software foi utilizado para a criação e gestão de toda a base de dados.

4.2.2.3 Apache

O Apache é o mais bem sucedido servidor web livre, criado em 1995 por Rob McCool [8], que era na altura funcionário da NCSA (National Center for Supercomputing Applications). Segundo estudos realizados em 2007 e em 2010, o Apache serve já mais de 50% dos sites existentes, dos quais cerca de 65% dos milhões dos sites existentes mais movimentados no mundo. É a principal tecnologia da Apache Software Foundation, responsável por mais de uma dezena de projetos envolvendo tecnologias de transmissão via web e processamento de dados. As suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o utilizador escreva os seus próprios módulos utilizando a API do software.

O Apache foi utilizado para simular o servidor PHP na minha máquina e para testar todas as páginas criadas.

4.2.2.4 MySQL

O MySQL foi criado na Suécia na década de 80, por David Axmark, Allan Larsson e Michael Monty Widenius [8], que têm trabalhado juntos desde então. É um sistema de gestão de bases de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada) como interface. É atualmente uma das bases de dados mais populares, com mais de 10 milhões de instalações por todo mundo, tais como a NASA, Dataprev, HP, Nokia, Sony, U.S. Army, Cisco Systems, Google, entre outros.

4.2.3 Core FTP LE

Para proceder à transferência de ficheiros, quer para o servidor de testes, quer para o servidor final, foi utilizado o programa Core FTP LE, que é um cliente FTP (File Transfer Protocol) seguro para Windows, desenvolvido pela "CoreFTP.com"[9], que permite ao utilizador transferir ficheiros para o servidor pretendido, depois de feita a autenticação nesse mesmo servidor.

O Core FTP LE foi utilizado para a transferência de todos os ficheiros da aplicação desenvolvida para os servidores, quer de testes, quer final.

Capítulo 5

Implementação da solução

Tanto o FrontOffice como o BackOffice foram desenvolvidos no Dreamweaver, com recurso às tecnologias HTML, CSS, PHP e JavaScript, enquanto que toda a base de dados foi desenvolvida em SQL com recurso à ferramenta MySQL, como já descrito anteriormente.

Os endereços das páginas web são os seguintes:

1. Casa do Pastor — `http://www.casadopastor.com/`.
2. FrontOffice — `http://www.casadopastor.com/pt/index.php/pt/alojamento`.
3. Backoffice — `http://www.casadopastor.com/pt/reservas/`.

5.1 Arquitetura

Na figura 5.1 é apresentada a estrutura da arquitetura de navegação do site. Inicialmente o Cliente faz um pedido através do seu computador pessoal via HTTP ao Servidor Web. Este, por sua vez, verifica se a página web requerida pelo Cliente é conteúdo HTML; se for HTML, transmite de imediato a resposta HTML para o Cliente, caso contrário (se for um pedido de conteúdo PHP) faz o pedido PHP ao Processador PHP. O processador PHP pode ainda, se necessário, fazer consultas ao Servidor de Base de Dados, o qual lhe devolve os dados respetivos dessa mesma consulta, e, por fim, envia a resposta HTML para o Servidor Web e, por sua vez, ao Cliente. O mesmo acontece com os gestores e administrador, no entanto estes podem ter também acesso através da rede local.

5.2 Base de dados

5.2.1 Modelo relacional

Todos os dados são armazenados numa base de dados (casadopa_reservas) desenvolvida para o efeito. Na figura 5.2 é apresentado o modelo relacional da base de dados.

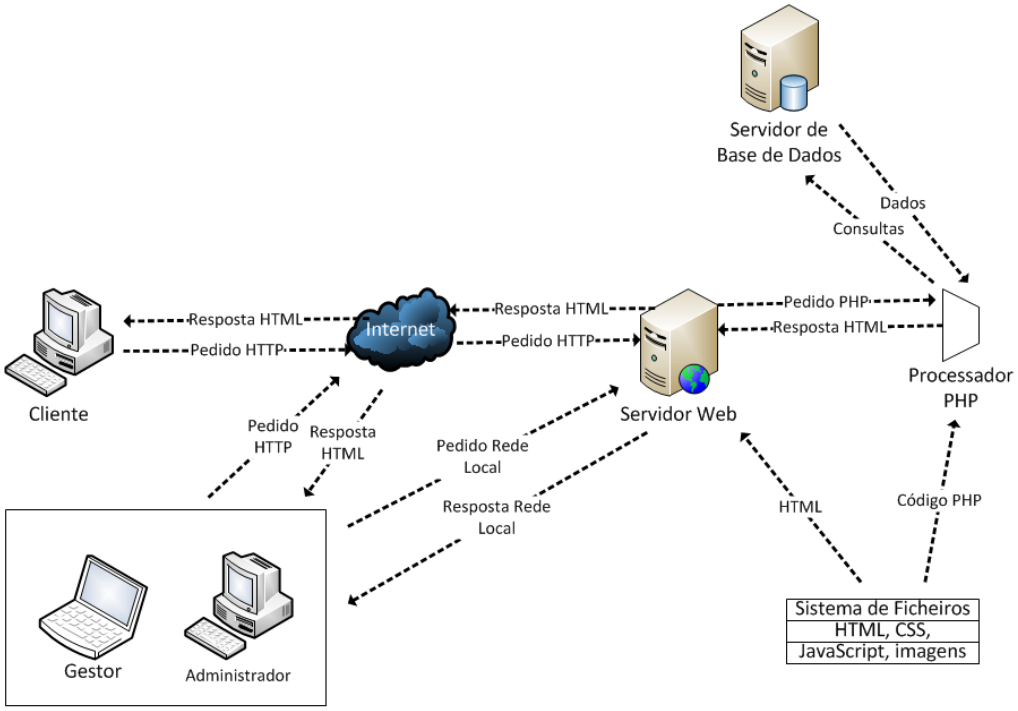

Figura 5.1: Arquitetura.

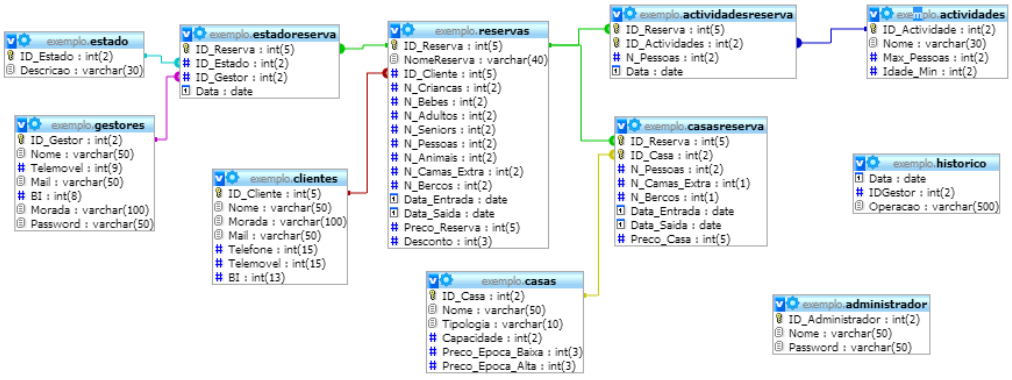

Figura 5.2: Modelo Relacional (ER).

Todas as ligações existentes indicam obrigatoriedade, à exceção da ligação entre a tabela "estadoreserva"e "gestores", em que na tabela "estadoreserva"é guardado o ID do gestor, mas este não é obrigatório, uma vez que vai ficar posteriormente guardado na tabela "historico".

5.2.2 Descrição das tabelas

Nesta secção é apresentada a descrição da estrutura das tabelas da base de dados, que inclui o dicionário de dados de cada uma das tabelas, e o significado de cada uma delas.

De forma a abreviar a descrição das tabelas, são utilizadas algumas siglas:

- D — Permite dígitos de 0 a 9.
- L — Permite letras de 'a' a 'Z' e espaço.
- CP — Chave primária.
- CE — Chave estrangeira.

5.2.2.1 Atividades

Na tabela 5.1 apresenta-se a estrutura da tabela relativa às atividades disponíveis para os clientes, onde podemos ver o ID da atividade, o seu nome, o número máximo de pessoas que podem entrar na atividade e a respetiva idade mínima.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Actividade (CP)	Inteiro	2	[1..99]	Número sequencial que identifica univocamente cada atividade
Nome	Texto	30	$\neq$ branco, [DL]	Nome da atividade
Max_Pessoas	Inteiro	2	[0..99]	Numero máximo de pessoas a participar na atividade
Idade_Min	Inteiro	2	[0..99]	Idade mínima das pessoas para participar na atividade

Tabela 5.1: Estrutura da tabela de atividades.

As atividades não são da responsabilidade da Casa do Pastor, sendo meramente informativas para o cliente, daí não terem nenhum preço associado.

5.2.2.2 Casas

Na tabela 5.2 apresenta-se a estrutura da tabela relativa às casas disponíveis para os clientes. Nesta podemos ver o ID da casa, o seu nome, tipologia e a capacidade máxima, bem como os preços da época alta e da época baixa.

A tipologia da casa serve apenas como informação, sendo que para o cálculo da capacidade e do preço não interfere.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Casa (CP)	Inteiro	2	[1..99]	Número sequencial que identifica univocamente cada casa
Nome	Texto	50	≠ branco, [DL]	Nome da casa
Tipologia	Texto	10	≠ branco, [DL]	Tipologia da casa (meramente informativa)
Capacidade	Inteiro	2	[0..99]	Capacidade máxima da casa
Preco_Epoca_Alta	Inteiro	3	[0..999]	Preço da casa em época alta
Preco_Epoca_Baixa	Inteiro	3	[0..999]	Preço da casa em época baixa

Tabela 5.2: Estrutura da tabela de casas.

5.2.2.3 Clientes

Na tabela 5.3 apresenta-se a estrutura da tabela relativa aos clientes. É aqui que podemos ver o ID do cliente, o seu nome, morada, mail, telefone, telemóvel e bilhete de identidade.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Cliente (CP)	Inteiro	5	[1..99999]	Número sequencial que identifica univocamente cada cliente
Nome	Texto	50	≠ branco, [DL]	Nome do cliente
Morada	Texto	100	≠ branco, [DL]	Morada do cliente
Mail	Texto	50	≠ branco, [DL]	Mail do cliente
Telefone	Inteiro	15	15 dígitos de 0 a 9	Telefone do cliente
Telemovel	Inteiro	15	15 dígitos de 0 a 9	Telemóvel do cliente
BI	Inteiro	13	13 ≠ branco, [DL]	Bilhete de Identidade do cliente

Tabela 5.3: Estrutura da tabela de clientes.

Os clientes da Casa do Pastor nem sempre são Portugueses, logo, o telefone e o telemóvel têm limite de 15 dígitos para precaver essas situações. Já o BI do cliente, uma vez que existem BI's que têm, para além de números, também letras, apenas se faz a validação de este não ser nulo, mas pode conter informação não correta, já que este é um dado informativo.

5.2.2.4 Reservas

Na tabela 5.4 apresenta-se a estrutura da tabela relativa às reservas. Aqui podemos ver os dados gerais da reserva, tais como o nome, o ID do Cliente inerente à reserva, o ID do estado em que esta se encontra, o número total de pessoas (e em particular de crianças, bebés, adultos e séniors), o número de animais, as datas de entrada e de saída, entre outros.

5.2.2.5 Estado

Na tabela 5.5 apresenta-se a estrutura da tabela relativa aos estados, onde podemos ver o ID do Estado e a sua descrição.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Reserva (CP)	Inteiro	5	[1..99999]	Número sequencial que identifica univocamente cada reserva
NomeReserva	Texto	40	≠ branco, [DL]	Nome da Reserva
ID_Cliente (CE)	Inteiro	5	[1..99999]	Número identificativo do cliente que efetuou a reserva
N_Bebes	Inteiro	2	[0..99]	Número de bebés
N_Criancas	Inteiro	2	[0..99]	Número de crianças
N_Adultos	Inteiro	2	[0..99]	Número de adultos
N_Seniors	Inteiro	2	[0..99]	Número de séniors
N_Pessoas	Inteiro	2	[0..99]	Número de pessoas
N_Animais	Inteiro	2	[0..99]	Número de animais
N_Camas_Extra	Inteiro	2	[0..99]	Número de camas extra
N_Bercos	Inteiro	2	[0..99]	Número de berços
Data_Entrada	Data		Data Válida	Data de entrada
Data_Saida	Data		Data Válida	Data de saída
Preco_Reserva	Real	5.2	[0,00..99999,99]	Preço da reserva
Desconto	Inteiro	2	[0..99]	Desconto da reserva

Tabela 5.4: Estrutura da tabela das reservas.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Estado (CP)	Inteiro	2	[1..99]	Número sequencial que identifica univocamente cada estado
Descricao	Texto	30	≠ branco, [DL]	Descrição do estado

Tabela 5.5: Estrutura da tabela de estados.

5.2.2.6 EstadoReserva

Na tabela 5.6 apresenta-se a estrutura da tabela relativa ao estado individual de cada reserva, onde é guardado o ID da Reserva, do Estado, do Gestor que efetuou a alteração, e a data da alteração.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Reserva (CE)	Inteiro	5	[1..99999]	Número que identifica univocamente cada reserva
ID_Estado (CE)	Inteiro	2	[1..99]	Número que identifica o estado referente à reserva
ID_Gestor (CE)	Inteiro	2	[1..99], pode ser nulo	Número que identifica o gestor que atualizou o estado da reserva
Data	Data		Data Válida	Data de alteração do estado da reserva

Tabela 5.6: Estrutura da tabela do estado de cada reserva.

Tal como referido anteriormente, o ID do Gestor pode ser nulo.

5.2.2.7 ActividadesReserva

Na tabela 5.7 apresenta-se a estrutura da tabela relativa às atividades inerentes às reservas, na qual podemos ver o ID da Reserva e da Atividade, o número de pessoas

que nela vão participar e a respetiva data.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Reserva (CE)	Inteiro	5	[1..99999]	Número que identifica univocamente cada reserva
ID_Actividade (CE)	Inteiro	2	[1..99]	Número que identifica a atividade associada à reserva
N_Pessoas	Inteiro	2	[0..99]	Número de pessoas que participarão na atividade
Data	Data		Data Válida	Data da atividade

Tabela 5.7: Estrutura da tabela de atividades da reserva.

5.2.2.8 CasasReserva

Na tabela 5.8 apresenta-se a estrutura da tabela relativa às casas inerentes às reservas. Aqui podemos ver o ID da Reserva e da Casa, o número de pessoas que nela vão habitar, as datas de entrada e saída, entre outros.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Reserva (CE)	Inteiro	5	[1..99999]	Número que identifica univocamente cada reserva
ID_Casa (CE)	Inteiro	2	[1..99]	Número que identifica a casa associada à reserva
N_Pessoas	Inteiro	2	[0..99]	Número de pessoas que irão habitar a casa
N_Camas_Extra	Inteiro	1	[0..9]	Número de camas extra necessárias para a casa
N_Bercos	Inteiro	1	[0..9]	Número de berços necessárias para a casa
Data_Entrada	Data		Data Válida	Data de entrada na casa
Data_Saida	Data		Data Válida	Data de saída da casa
Preco_Casa	Inteiro	5	[1..99999]	Preço total da casa

Tabela 5.8: Estrutura da tabela de casas da reserva.

Uma vez que o desconto não atua diretamente no preço da casa, esse valor é inteiro e não do tipo float.

5.2.2.9 Gestores

Na tabela 5.9 apresenta-se a estrutura da tabela relativa aos gestores, na qual podemos ver os dados de cada gestor, tal como o seu ID, nome, telemóvel, mail, entre outros.

A password do gestor é guardada de forma encriptada, isto é, se a password for por exemplo "‘gestor"’ não irá guardar "‘gestor"’, mas sim o resultado da encriptação.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Gestor (CP)	Inteiro	2	[1..99]	Número que identifica univocamente cada gestor
Nome	Texto	50	≠ branco, [DL]	Nome do gestor
Telemovel	Inteiro	9	9 dígitos de 0 a 9	Telefone do gestor
Mail	Texto	50	≠ branco, [DL]	Mail do gestor
BI	Inteiro	8	8 dígitos de 0 a 9	Telefone do gestor
Morada	Texto	100	≠ branco, [DL]	Morada do gestor
Password	Texto	50	≠ branco, [DL]	Password do gestor

Tabela 5.9: Estrutura da tabela de gestores.

Campo	Tipo	Tamanho	Validações	Descrição
ID_Administrador (CP)	Inteiro	2	[1..99]	Número que identifica univocamente cada administrador
Nome	Texto	50	≠ branco, [DL]	Nome do administrador
Password	Texto	50	≠ branco, [DL]	Password do administrador

Tabela 5.10: Estrutura da tabela do administrador.

5.2.2.10 Administrador

Na tabela 5.10 apresenta-se a estrutura da tabela relativa ao administrador, onde se pode ver o seu ID, nome e password.

Tal como no caso dos gestores, a password do administrador é guardada igualmente encriptada. Apesar de o objetivo da aplicação ser a existência de apenas um administrador, para evitar que um dia mais tarde se queiram mais administradores, foi dada uma margem de dois dígitos para o respetivo ID.

5.2.2.11 Historico

Na tabela 5.11 apresenta-se a estrutura da tabela relativa ao histórico, que é onde ficam guardadas todas as operações de edição e remoção de registos do sistema para poderem ser mais tarde vistos pelo administrador.

Campo	Tipo	Tamanho	Validações	Descrição
Data	Data		Data Válida	Data da realização da operação.
IDGestor	Inteiro	2	[1..99]	Número que identifica o gestor que efetuou a operação.
Operacao	Texto	500	≠ branco, [DL]	Descrição da operação realizada.

Tabela 5.11: Estrutura da tabela do historico.

O campo "'IDGestor"' é onde fica guardado o ID do Gestor que efetuou a operação, no entanto, se um administrador efetuar alguma operação, fica também aqui registado, mas com IDGestor = 0, para haver diferenciação de quem fez o quê. O campo "'Operacao"' é onde fica guardada detalhadamente, com todos os campos necessários, a operação que o gestor ou o administrador fizeram, de forma a que, caso necessário, o administrador possa repôr os dados.

5.2.3 Criação da base de dados no servidor MySQL

A criação da base de dados e tabelas no servidor MySQL (`https://b23.hocnet.org:2083/3rdparty/phpMyAdmin/index.php`) foi realizada através do script SQL apresentado em anexo A.2, com a opção *Import* do *phpmyadmin*.

5.3 FrontOffice

O FrontOffice do projeto refere-se à parte da aplicação à qual o utilizador final tem acesso. Entende-se por utilizador final todos os clientes da Casa do Pastor que irão utilizar a aplicação web para criarem as suas reservas. Esta parte da aplicação web está embebida do site oficial da Casa do Pastor, tendo sido criadas duas páginas, para:

1. O calendário, onde o cliente consulta o calendário de todas as casas.
2. Efetuar reservas.

Apesar das páginas da aplicação web estarem embebidas no projeto, quando por exemplo o cliente começa a inserir reservas, a página principal não vai atualizar, mas onde ele estará a inserir os dados irá funcionar como um site independente.

5.3.1 Mapa do site

Por um lado, temos a parte do calendário, que é apenas uma página:

- CalendarioCli

Por outro, temos a criação da reserva, que inclui já várias páginas:

- CriarReserva
  - VerificarDesponibilidade
    - InfoCasas
      - Confirmacao
        1. Finalizacao

Nas secções seguintes e apresentado com detalhe o modo de navegação e funcionamento da aplicação do lado do cliente.

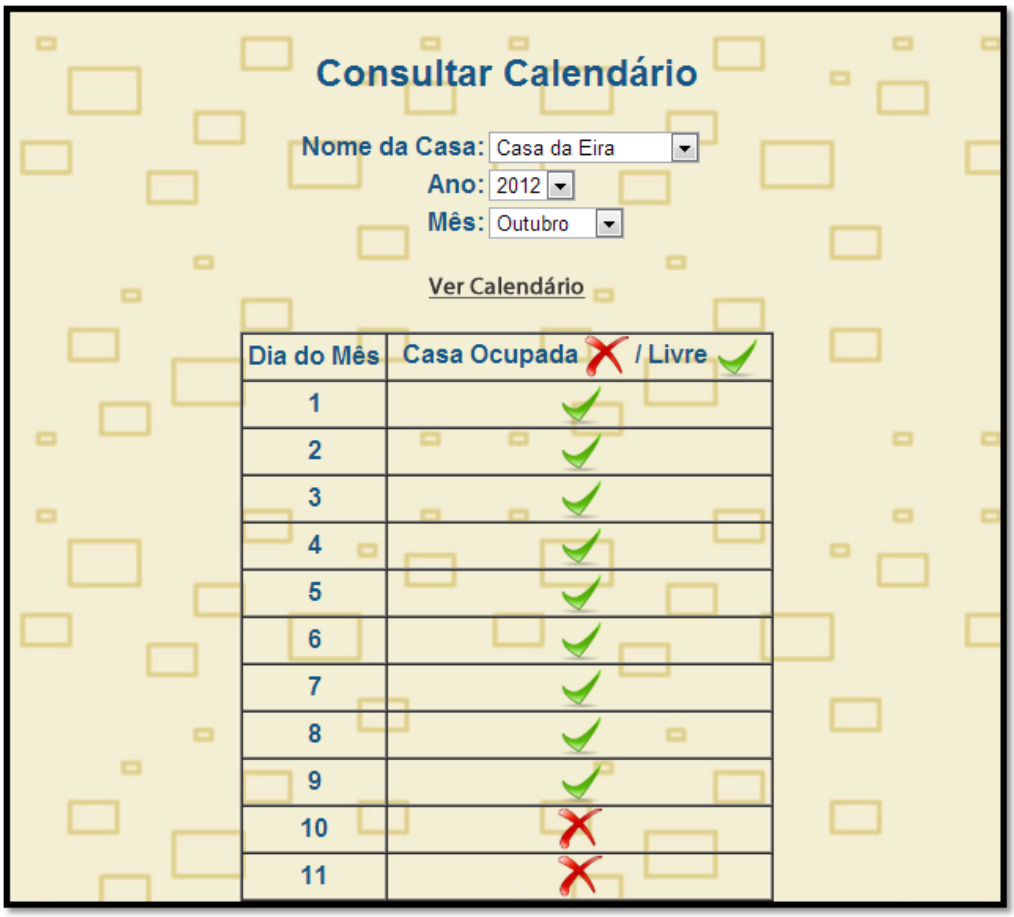

Figura 5.3: Calendário Cliente.

5.3.2 Calendário

Na figura 5.3 é apresentada a vista inicial da página onde o cliente pode consultar o calendário de todas as casas, numa determinada data, de forma a saber se uma determinada casa se encontra ocupada ou não na data escolhida. Sempre que a página é carregada é visto se existem dados guardados no método "POST"(mês, ano e casa), e, caso exista, é mostrado o calendário do mês e ano selecionado da respetiva casa, caso contrário a casa que fica selecionada é a primeira casa da lista da tabela da base de dados, vendo-se o mês e ano atual.

Esta aparência é obtida a partir do código do anexo A.1.

5.3.3 Criação da reserva

Na figura 5.4 é apresentada a vista inicial da página onde o cliente começa a criar a sua reserva. Aqui o cliente irá inserir o número de pessoas e as datas de entrada e de saída. Clicando em "Seguinte", passará (caso se verifique) para a página seguinte, continuando com a inserção dos dados da reserva, sendo que a próxima página apresentada será dentro da mesma página do site, mas dentro da criação da reserva será uma página diferente.

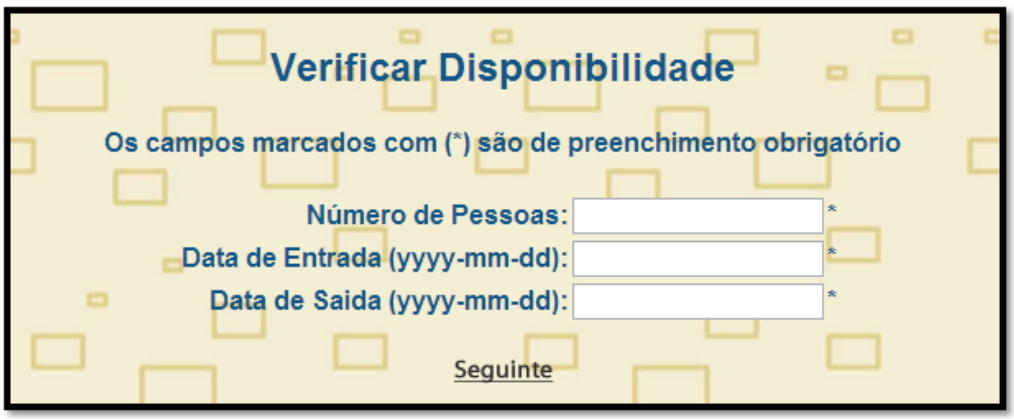

Figura 5.4: Criar Reserva.

A página para onde é seguidamente reencaminhado, é uma página onde o cliente tem que inserir dois tipos de dados diferentes:

1. Em primeiro lugar, os dados relativos ao próprio cliente, tais como o Nome, Morada, Telefone e/ou Telemóvel, email, entre outros.

2. Depois, os dados relativos à reserva no geral, bem como o Nome da reserva, o número de crianças, número de adultos, número de animais, etc.

Ambas as páginas passam por um processo de validação de todos os dados inseridos, como pode ser demonstrado pelo código presente no anexo A.7. Todos os dados têm que ser válidos, sendo que o número de pessoas tem que ser um valor numérico,

caso contrário volta para a página anterior para ser novamente inserido; da mesma forma, as datas de entrada e de saída, são igualmente validadas, e se estas estiverem inseridas de forma errada, volta igualmente para a página anterior para serem novamente inseridos. A função *ValidaData* verifica se a data inserida é válida ou não e verifica se esta é uma data superior à data atual: se tudo isto se verificar, então a data é válida, caso contrário é uma data inválida.

Tal como todos os dados são validados, o email do cliente tem que ser igualmente validado, e essa validação está demonstrada no anexo A.8. Todos os emails têm que ser do formato *exemplo@exemplo.exe*.

Se todos os dados inseridos estiverem corretos, é levado para a página que contém uma lista com as casas livres nas datas escolhidas, tal como se pode ver na figura 5.5.

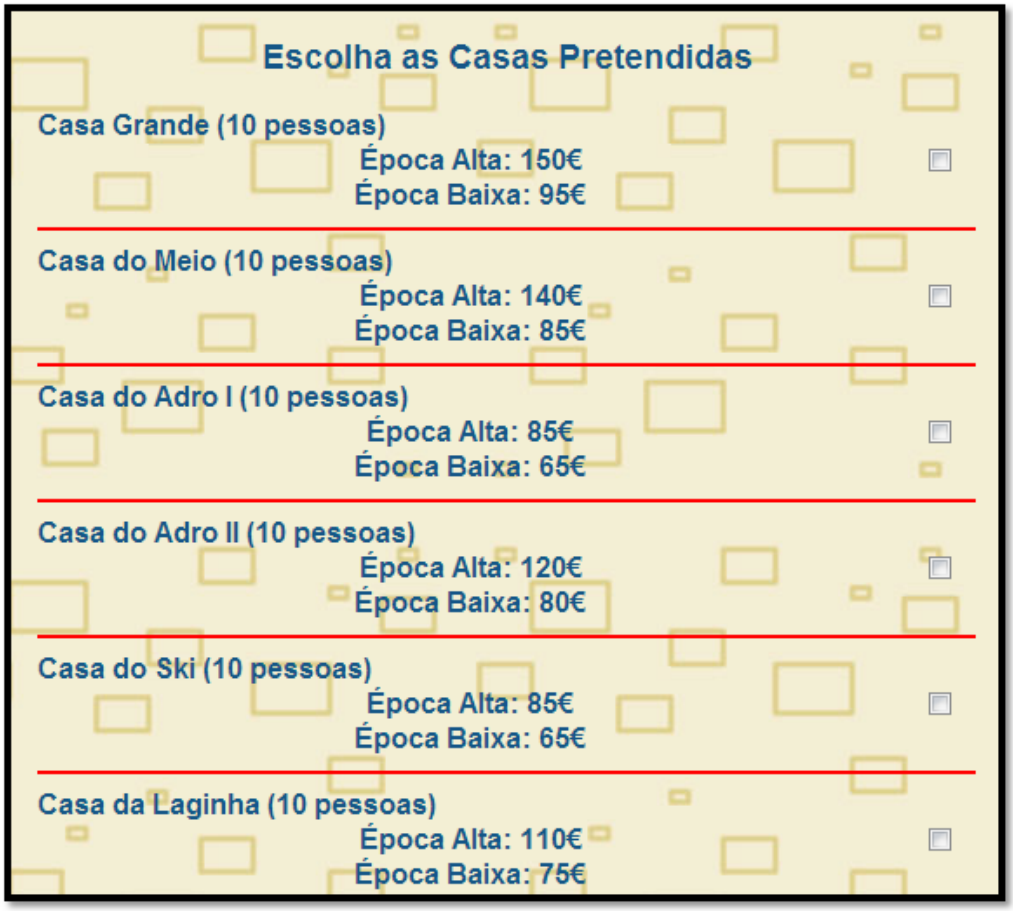

Figura 5.5: Lista das Casas disponíveis.

É aqui que o cliente escolhe as casas pretendidas (tantas quanto necessário para todas as pessoas). Nesta lista pode-se ver o nome da casa, a capacidade total, e os preços da época alta e época baixa.

Esta lista é obtida com recurso ao código mostrado na listagem 10. Inicialmente é feito um *Select* à tabela das casas da reserva de forma a guardar numa lista todas as

casas que estão a ser utilizadas entre as datas de entrada e de saída indicadas pelo cliente, sendo que se uma casa estiver a ser usada, nem que seja apenas durante um dia, entre essas datas, entra para essa lista; de seguida, é feito um *Select* a todas as casas, e apenas são mostradas ao cliente aquelas que não estiverem na lista.

A primeira tentativa de criação desta lista não foi bem sucedida, pois apenas estava a guardar os ID's das casas todos seguidos, sendo que a lista ficava neste formato: [1234567], sem qualquer separação. Poderia resultar, se houvesse garantidamente apenas 10 casas, mas se houvesse mais, este formato não funcionava, porque quando fazia a comparação entre o ID das casas e a lista na altura de as mostrar ao cliente, se o ID da casa fosse o "12", ao verificar nesta mesma lista, iria dizer que existia, uma vez que o "12"aparece naquela lista, apesar de serem ID's distintos.

A solução passou por inserir uma virgula no inicio da tabela, e uma depois de todos os ID's, ficando agora neste formato: [,1,2,3,4,5,]. Desta forma, se o ID da casa a pesquisar fosse o 12, não seria pesquisado por "12"mas sim por ",12,", o que, neste caso, não existe na lista, logo a casa com ID = 12 está livre nas datas inseridas pelo cliente.

Depois desta escolha é levado para uma página onde lhe irão aparecer todos os dados inseridos, de forma a este confirmar se todos estão corretos. Se houver erros, volta atrás e corrige, se não, clica em "'Seguinte"' e a reserva é então inserida na base de dados, mostrando uma mensagem de sucesso ao cliente.

5.4 BackOffice

O BackOffice é o espaço reservado apenas para gestores e administradores, sendo que para poderem ter acesso, é necessário efetuar um login prévio. Esta parte da aplicação web não tem acesso direto a partir da parte do site do cliente.

Este espaço é dividido em duas subcategorias:

1. A parte visível pelo gestor.
2. A parte visível pelo administrador.

Aquilo que é visível pelo gestor é igualmente vista pelo administrador, sendo que o administrador tem acesso a mais algumas operações que o gestor não tem.

Nos próximos capítulos será explicado o mapa do site para ambos os utilizadores, tal como foi anteriormente explicado para a parte do cliente.

5.4.1 Gestores

Os gestores são os responsáveis pela gestão total da Casa do Pastor, desde as reservas, até às casas, atividades, e até mesmo clientes. Eles podem editar, eliminar, ou inserir novos registos em todas estas categorias e, tal como os clientes, consultar o calendário das casas. Podem ainda editar os dados da sua conta.

Listagem 10 Verificação das casas livres.

```
 1:  $listacasasocupadas = ',';
 2:  $sqlcusadas="SELECT * FROM casas, casasreserva
     WHERE casas.ID_Casa = casasreserva.ID_Casa and (('$dataentrada' between
     casasreserva.Data_Entrada and casasreserva.Data_Saida) or ('$datasaida'
 between casasreserva.Data_Entrada and casasreserva.Data_Saida))";
 3:  $resultcusadas = mysql_query($sqlcusadas, $connection);
 4:  if(!$resultcusadas) die("Erro, registo não efectuado: " . mysql_error());
 5:  while($registocusadas = mysql_fetch_array($resultcusadas)) {
 6:    $IDCasaUsada = $registocusadas['ID_Casa'];
 7:    $listacasasocupadas = $listacasasocupadas.$IDCasaUsada.',';
 8:  }
 9:  $sqle="select * from casas";
10:  $resulte = mysql_query($sqle, $connection);
11:  if(!$resulte) die("Erro, registo não efectuado: " . mysql_error());
12:  while($registoe = mysql_fetch_array($resulte)) {
13:    $NomeCasa = $registoe['Nome'];
14:    $IDCasa = $registoe['ID_Casa'];
15:    $CapCasa = $registoe['Capacidade'];
16:    $precoalta = $registoe['Preco_Epoca_Alta'];
17:    $precobaixa = $registoe['Preco_Epoca_Baixa'];
18:    if (strpos($listacasasocupadas,','.$IDCasa.',') !== false){
19:      $casalivre = 0;
20:    }
21:    else{
22:      $casalivre = 1;
23:    }
24:    if (!isset($casalivre)){
25:      $casalivre = 1;
26:    }
27:    if ($casalivre == 1){
28:      ?>
29:        <tr>
30:          <td align="left" valign="middle"><b><?php echo $NomeCasa ?>
           (<?php echo $CapCasa ?> pessoas) </b></td>
31:          <TH ROWSPAN=3><input type="checkbox" name="casas[]"
           value="<?php echo $IDCasa ?>"/></TH>
32:        </tr>
33:        <tr>
34:          <td align="center" valign="middle"><b>Época Alta: <?php
           echo $precoalta ?></b></td>
35:        </tr>
36:        <tr>
37:          <td align="center" valign="middle"><b>Época Baixa: <?php
           echo $precobaixa ?></b></td>
38:        </tr>
39:        <tr>
40:          <td><hr size=2 noshade="noshade" color="#FF0000"/></td>
             <td><hr size=2 noshade="noshade" color="#FF0000"/></td>
41:        </tr>
42:      <?php
43:    }
44:  }
```

5.4.1.1 Mapa do site

Ao contrário da aplicação do lado do cliente, a parte do gestor não está embebida em nenhuma página já existente.

1. index
2. RecuperarPassword
   (a) Home
      i. Calendario
   (b) Actividades
      i. InserirActividades
      ii. EditarActividades
      iii. EliminarActividades
   (c) Casas
      i. InserirCasas
      ii. EditarCasas
      iii. EliminarCasas
   (d) Clientes
      i. InserirClientes
      ii. EditarClientes
      iii. EliminarClientes
   (e) Reservas
      i. InserirReserva
      ii. EliminarReservas
      iii. ExpandirReserva
         - InserirActividadeReserva
         - EditarActividadeReserva
         - EliminarActividadeReserva
         - InserirCasaReserva
         - EditarCasaReserva
         - EliminarCasaReserva
         - EditarReserva
         - EditarClienteReserva
         - EditarEstadoReserva
         - EditarDesconto
   (f) Gestores

Nas secções seguintes é apresentado com detalhe o modo de navegação e funcionamento da aplicação do lado do gestor.

5.4.1.2 Login

Para o gestor entrar na aplicação tem de efetuar o seu login, tal como visto na figura 5.6. Caso ele se esqueça da password pode sempre requisitar uma nova password para o seu email. O código da página login pode ser visto no anexo A.3.

Figura 5.6: Página de login.

Se o gestor se enganar nos dados de login, é vista uma mensagem a vermelho a informa-lo de que os dados inseridos estão errados. Caso contrário, se o gestor efetuar corretamente o login, os dados dele são guardados em cookies. A estrutura da página encarregue destas operações pode ser vista no anexo refanexo:cookiesessao.

Inicialmente, e uma vez que a passowrd dos gestores guardada na base de dados está encriptada, é feita a encriptação na password escrita pelo gestor, da seguinte forma: *$passwordgestor = md5(sha1($_POST['password']))*. De seguida, é feito um *Select* à tabela dos gestores, de forma a averiguar se existe algum registo com o nome e password inseridas pelo gestor; se existir, guarda-se o ID desse registo. De seguida é feito um *Select* exatamente igual à tabela do administrador, para averiguar igualmente se existe algum registo com o nome e passowrd inseridos, neste caso, por um administrador. Se existir, é guardado o ID desse mesmo registo.

De forma a diferenciar os gestores do administrador, aqui, se os dados de login disserem respeito a um gestor, na variável *idgestor* é guardado o ID do gestor, caso contrário, se os dados disserem respeito ao administrador, na variável *idgestor* é guardado o valor "99999"e na variável *idadmin* é guardado o ID do administrador. Desta forma, quando se for criar os cookies, se o valor guardado na variável *idgestor* for "99999", criam-se dois cookies: *gestorDados* com o valor "0"e *adminDados* com o ID do administrador, por se tratar de um administrador; caso contrário, se o valor guardado na variável *idgestor* for diferente "99999", guarda-se apenas o cookie *gestorDados* com o ID do gestor. Todos estes cookies têm uma existência de uma hora, e, depois dessa hora, a sessão expira.

No início de cada página do BackOffice é visto se o cookie *gestorDados* existe, que é feito tal como se pode ver na listagem 11. Se o cookie estiver criado, são mostrados os dados da página, caso contrário volta para a página index, com uma mensagem a informar que a sessão expirou.

Listagem 11 Código PHP para verificar se o gestor está autentificado.

```
1: if(isset($_COOKIE["gestorDados"])) {
2:   $idGestorActivo = $_COOKIE["gestorDados"];
(Respetivas operações para cada página)
3: }
4: else {
5:   ?><script language='javaScript'>window.location.href='index.php?
     Mensagem=1'</script><?php
6: }
```

Quando o gestor quiser efetuar o logout, clicando no menu do lado esquerdo "Sair", é levado para uma página que recria um cookie *gestorDados* sem qualquer valor, com tempo de duração inferior ao tempo atual, tal como podemos ver no anexo A.5. A diferença está na função JavaScript, em que antes o tempo de validade do cookie era dado pela linha *expire.setTime(today.getTime() + 3600000*24*nDays)* e aqui é dado pela linha *expire.setTime(today.getTime() -1);*.

5.4.1.3 Consultar calendário

Tal como os clientes, o gestor pode consultar o calendário de qualquer casa em qualquer data. Isso é feito na página principal, onde lhe aparecem os dados da consulta tal como do lado do cliente.

5.4.1.4 Casas, Atividades e Clientes

Para as tabelas `Casas, Atividades e Clientes` são permitidas as operações:

1. Visualizar dados;
2. Inserir Novo;
3. Editar;
4. Eliminar;

É apresentada uma tabela com todos os dados existentes na respetiva tabela da base de dados, tal como podemos ver na figura 5.7 e o gestor pode efetuar as operações ao clicar nos ícones de edição (o livro, na coluna "Editar"), no ícone de eliminação (a cruz, na coluna "Eliminar"), ou o botão "Inserir Novo..."no fundo da tabela.

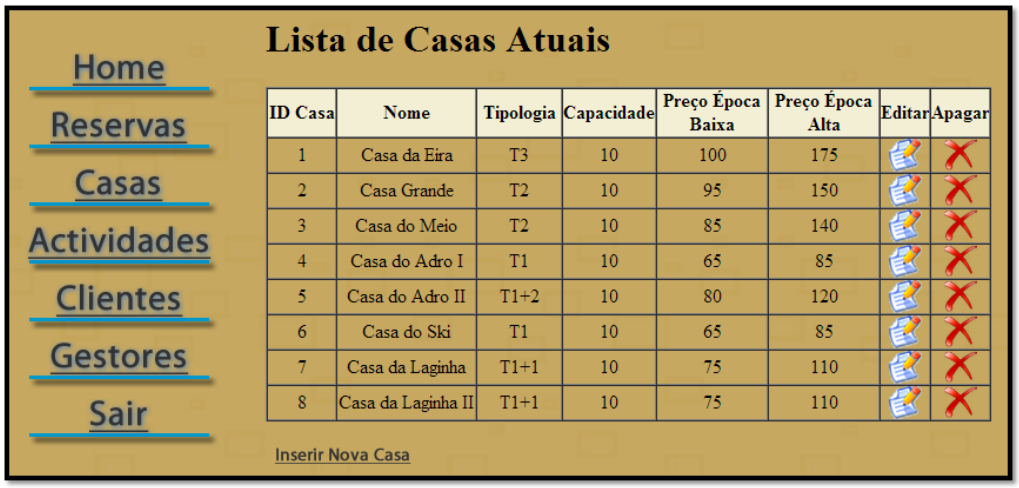

Figura 5.7: Lista de todas as casas existentes.

5.4.1.5 Reservas

No caso das reservas, é tudo igual às casas, clientes e atividades, à exceção do botão "Editar", que, neste caso, serve para expandir os dados da reserva, isto é, reencaminhar para uma página onde aparecem os dados da reserva, do cliente que a efetuou, das casas e atividades a ela associada, entre outros. Esse botão encontra-se na coluna "Detalhes"da tabela e é identificado com um ícone com o sinal "+"., tal como se pode ver na figura 5.8.

Figura 5.8: Lista de todas as reservas existentes.

Esta lista é obtida através do código presente no anexo A.9. É feito um *Select* individual a cada registo da tabela "reservas", e depois um *Select* às tabelas "clientes"e "estadoreserva"para cada uma das reservas. Todas as reservas são ordenadas por data de entrada.

5.4.1.6 Expansão da reserva

Quando o gestor expande uma reserva, pode ver todos os dados que a esta estão associados.

No caso de querer editar os dados gerais da reserva, é levado para uma página onde poderá editar todos, à exceção do ID da Reserva. Quando o gestor pretender alterar aqui o número total de pessoas e as datas de entrada e de saída, terá que ter sempre em atenção o facto de existirem ou não, casas e/ou atividades associadas a esta reserva. Para que os dados sejam devidamente editados na base de dados, estes têm que estar todos corretamente inseridos nas respetivas caixas de texto.

De seguida tem os dados do cliente, que, tal como a reserva, podem ser todos editados à exceção do ID do Cliente. A edição destes dados não interfere diretamente com nenhum dado relativo à reserva.

O gestor pode ainda editar o estado da reserva, tal como se pode ver na figura 5.9. O estado da reserva será editado sempre que se verifique, como por exemplo quando se der entrada ou saída nas casas, bem como quando o gestor aceita ou recusa a reserva.

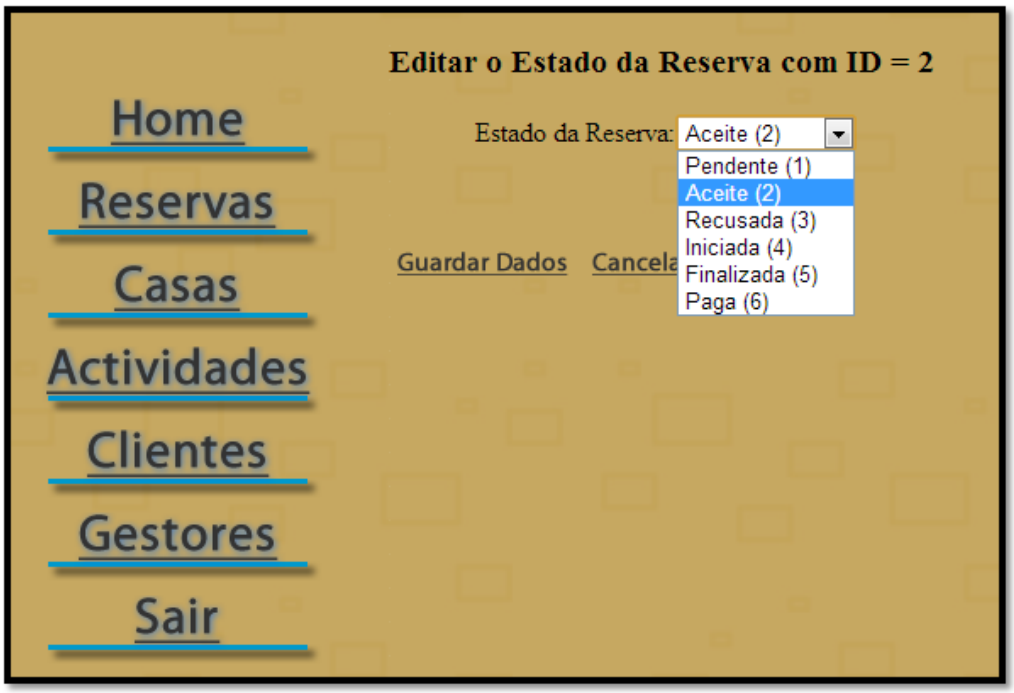

Figura 5.9: Edição do estado da reserva.

Se o gestor atualizar o estado da reserva para "aceite"ou "recusada", o cliente recebe de imediato uma mensagem no email, que indicou aquando da criação da reserva, com a informação detalhada da reserva, como pode ser visto na figura 5.10.

A estrutura deste email é criada com recurso ao código apresentado no anexo A.6. Inicialmente são adquiridos todos os dados necessários relativos à reserva e ao cliente, através de vários *Select* às várias tabelas. Depois disso, é identificado o destinatário

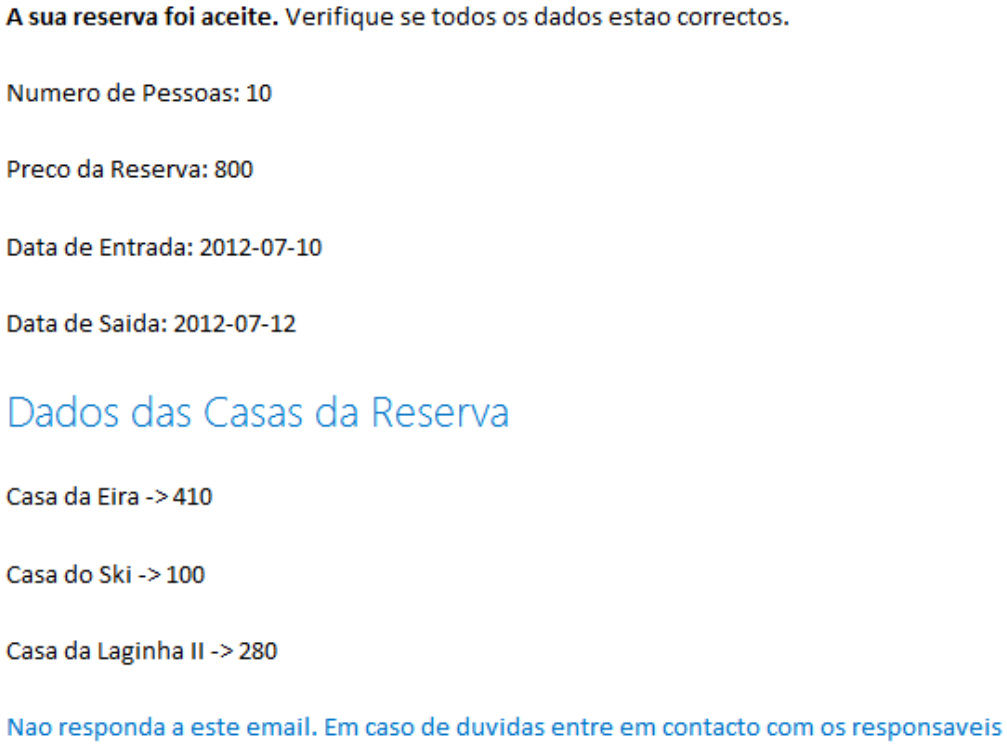

Figura 5.10: Exemplo de email de confirmação.

com o email adquirido dos dados do cliente, é escrito o assunto do email, e só depois se começa a escrever o corpo da mensagem., mensagem esta que é escrita como que se de um documento HTML se tratasse, com as respetivas *tags (head, body, etc)*. Dentro do corpo vão ficar os dados da reserva, do cliente, e de todas as casas desta reserva, bem como a informação de que a reserva foi aceite. Depois de o corpo da mensagem estar estruturado, é necessário indicar algumas especificações do email, tal como a estrutura do email, que é do tipo *text/html*, e o remetente, que, neste caso, é a Casa do Pastor (*From: casadopastor <casadopastor@hotmail.com>*. Só depois disto é criado/enviado o email, com a linha *mail(destinatario,assunto,corpo,headers)*.

O gestor tem ainda a possibilidade de editar, eliminar ou inserir uma nova casa ou atividade na reserva. Estas operações funcionam de forma semelhante ao visto anteriormente em "Páginas de visualização de dados — Casas, Atividades e Clientes".

No caso de o gestor querer inserir uma casa ou atividade, é-lhe apresentada uma página onde irá inserir todos os dados requeridos, e selecionar a casa ou atividade, que lhe é apresentada numa lista como se pode ver na figura 5.11. No caso das casas, apenas lhe aparecem as casas que estão livres nas datas da reserva.

Neste caso (*Insert*), a instrução SQL utilizada é a visível na listagem 12, sendo que todos os dados têm que ser inseridos (nenhum pode ser nulo) e ficam guardados com o ID da Reserva indicado na variável referida na instrução, na cláusula *WHERE*.

No caso de se querer editar uma casa já existente (*Update*), a instrução SQL utilizada é a visível na listagem 13, sendo que, tal como no caso do *Insert*, todos os dados

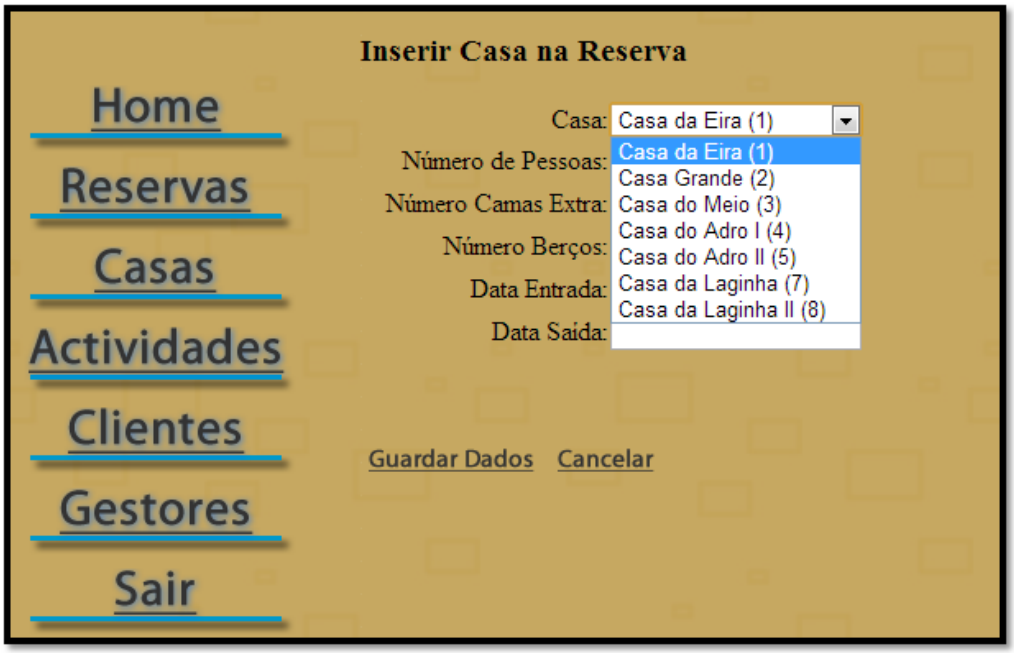

Figura 5.11: Inserção de nova casa numa reserva.

Listagem 12 Instrução SQL para inserir uma casa para a respetiva reserva.

```
INSERT INTO ‘casasreserva‘
(‘ID_Reserva‘, ‘ID_Casa‘, ‘N_Pessoas‘, ‘N_Camas_Extra‘, ‘N_Bercos‘,
‘Data_Entrada‘, ‘Data_Saida‘, ‘Preco_Casa‘)
VALUES (’$IDReserva’, ’$idcasa’, ’$npessoas’, ’$ncamas’, ’$nbercos’,
’$dataentrada’, ’$datasaida’, ’$preco’)
```

têm que ser inseridos e ficam guardados no registo com ID da Reserva e da Casa indicados nas respetivas variáveis na instrução, na cláusula *WHERE*.

Listagem 13 Instrução SQL para inserir uma casa para a respetiva reserva.

```
UPDATE ‘casasreserva‘ SET ‘N_Pessoas‘ = ’$npessoas’, ‘N_Camas_Extra‘ =
’$ncamas’, ‘N_Bercos‘ = ’$nbercos’,  ‘Data_Entrada‘ = ’$dataentrada’,
‘Data_Saida‘ = ’$datasaida’, ‘Preco_Casa‘ = ’$preco’
WHERE ID_Reserva = ’$IDReserva’ AND ID_Casa = ’$idcasa’
```

No caso de se querer eliminar uma casa da reserva (*Delete*) a instrução SQL utilizada é a visível na listagem 14, onde a linha da tabela eliminada será a que tem o ID da Casa e ID da Reserva iguais aos indicados na cláusula *WHERE*.

Listagem 14 Instrução SQL para inserir uma casa para a respetiva reserva.

```
DELETE FROM casasreserva WHERE ID_Reserva = ’$IDReserva’
AND ID_Casa = ’$idcasa’
```

Para além de tudo isto, o gestor pode ainda editar o desconto da reserva, que pode ir de 0% até 99%, sendo que este desconto não afeta a totalidade da reserva, apenas o total do valor das casas; no caso de existirem animais na reserva, esse preço é contabilizado na sua totalidade sem qualquer desconto.

5.4.1.7 Editar conta do gestor

Por fim, o gestor pode ainda editar os dados da sua conta, tais como o Nome, Morada, contactos ou password. A password tem que ser duplamente inserida, por questões de segurança, e é guardada na base de dados de forma encriptada.

5.4.2 Administrador

O administrador pode, tal como já referido, fazer tudo o que o gestor faz, mas, para além disso, pode ainda consultar o histórico de todas as operações efetuadas pelos gestores e editar, eliminar ou criar novas contas de gestores.

5.4.2.1 Mapa do site

A aplicação do lado do administrador funciona exatamente da mesma forma e com o mesmo aspeto que o lado do gestor.

1. index
2. RecuperarPassword
   (a) Home

i. Calendario
ii. Historico

(b) Actividades
i. InserirActividades
ii. EditarActividades
iii. EliminarActividades

(c) Casas
i. InserirCasas
ii. EditarCasas
iii. EliminarCasas

(d) Clientes
i. InserirClientes
ii. EditarClientes
iii. EliminarClientes

(e) Reservas
i. InserirReserva
ii. EliminarReservas
iii. ExpandirReserva
- InserirActividadeReserva
- EditarActividadeReserva
- EliminarActividadeReserva
- InserirCasaReserva
- EditarCasaReserva
- EliminarCasaReserva
- EditarReserva
- EditarClienteReserva
- EditarEstadoReserva
- EditarDesconto

(f) Gestores

Nas secções seguintes é apresentado com detalhe o modo de navegação e funcionamento da aplicação do lado do administrador.

Tal como referido anteriormente, as únicas diferenças entre o administrador e o gestor é que o administrador pode consultar um histórico de todas as operações detalhadas e editar, eliminar e inserir contas de gestores.

5.4.2.2 Histórico

Quando o administrador efetua o login, para além de, na página principal, poder consultar o calendário das casas, tem também uma opção que lhe permite consultar o histórico de operações.

A primeira vez que a página é carregada, é visto, caso exista, o histórico da data atual. O administrador pode depois selecionar o ano e o mês e, posteriormente, o respetivo dia, tal como se pode ver na figura 5.12.

Figura 5.12: Tabela de Histórico.

5.4.2.3 Contas de gestores

Quando o administrador entra no menu "'Gestores"', para além de poder, tal como todos os gestores, alterar os dados da sua conta, pode também editar, eliminar e inserir contas de gestores.

Todas estas operações funcionam de forma semelhante ao que se viu anteriormente nas casas ou nas atividades, tal como podemos ver na figura 5.13.

5.5 Colocação e divulgação online

Para colocar a aplicação web online é necessário registar um domínio, alugar o alojamento, criar a base de dados e transferir os ficheiros da aplicação web para servidor de alojamento. Nas secções seguintes são descritos todos estes passos.

5.5.1 Registo de domínios

Domínio é o nome que serve para localizar e identificar conjuntos de computadores na Internet [2]. O nome de domínio foi concebido com o objetivo de facilitar a forma de como os endereços de computadores na Internet eram memorizados, caso contrário seria necessário memorizar uma grande sequência de números. Em Portugal, o registo de domínios é feito pela FCCN (Fundação para a Computação Cientifica Nacional).

Figura 5.13: Tabela de Gestores.

A **FCCN** é uma instituição privada portuguesa sem fins lucrativos de utilidade pública, fundada em Janeiro de 1987, sendo que a sua principal atividade é o planeamento da Rede Ciência, Tecnologia e Sociedade (RCTS), o backbone de Portugal, e a gestão do serviço de registo do domínio de topo `.pt`.

O registo de um domínio significa reservar o direito de utilização de um nome na Internet durante um determinado período de tempo. É necessário entrar em contacto com o órgão competente pela gestão dos registos, preencher um formulário, envia-lo pela Internet, esperar a resposta por email e pagar as taxas correspondentes.

Existem duas classes de domínio: os genéricos, que, tal como o nome indica, são gerais, independentes de países (`.pt.com, .net, .org, .gov`), e os territoriais, que estão associados a um país ou região (`.pt` para Portugal, `.fr` para a França, `.eu` para a Europa). No entanto podem ainda ser associados domínios com parte genérica e parte territorial (por exemplo `.com.pt`).

Não existe qualquer vantagem ou desvantagem em registar um domínio genérico ou territorial, no entanto, por norma, cada um regista o domínio que mais interesse tiver:

- Se for uma empresa com atividade exclusivamente nacional pode ser mais conveniente utilizar o `.pt`;
- Se for um site com interesses para visitantes de outros países, utilizar o .com;
- Se tiver interesses numa região especifica por exemplo a Europa, utilizar um territorial do tipo .eu, ou pode-se ainda optar por restringir por países, como `.fr`, `.es`, `.be`, etc.

O custo de registo de um domínio ronda normalmente os 11 euros para domínios genéricos (`.com`, `.net`, `.org`, `.info`), 18 euros para os registos regionais (`.eu`, `.mobi`) e cerca de 21 euros para os territoriais (`.pt`, `.fr`, `.com.pt`). As renovações do

domínio são feitas anualmente e têm o mesmo custo. Em vez de comprar apenas por um ano, o proprietário do domínio pode comprar por mais tempo.

5.5.2 Servidor da empresa

A empresa Ideias Soberbas disponibilizava já de um domínio num servidor Linux para o site da Casa do Pastor, no serviço hocnet.pt [10], com o pacote "Linux Lite".

As características principais são apresentadas na tabela 5.12.

Característica	**Valor**
Pagamento Anual	1,90/Ano (+IVA)
Espaço em Disco	1 GB
Tráfego Mensal	1 TB
Nº Máx. Domínios	1
Contas de E-mail	20
Acesso Webmail	Gratuito
Acesso POP3 e SMTP	Sim
Conta Catch-All	Sim
Anti-Vírus / Anti-Spam	Sim
Respostas Automáticas	Ilimitado
Forward de E-mail	Ilimitado
Alias de Correio	Ilimitado

Tabela 5.12: Características principais do alojamento.

O servidor possui uma Base de Dados Linux MySQL/mariaDB, uma conta FTP e um subdomínio.

As linguagens suportadas são as seguintes:

- HTML;
- PHP 5;
- CGI;
- CSS;
- Flash;
- Vídeo e Áudio;
- Perl;
- JavaScript;
- Zend Optimizer;
- ImageMagick;

Na tabela 5.13 são apresentadas as características das ferramentas de gestão.

As carcterísticas do suporte técnico são apresentadas na tabela 5.14.

Característica	Valor
Painel de Controlo	Cpanel
Estatísticas	Sim
Extensões Frontpage	Não
Acesso FTP	Sim
phpMyAdmin	Sim
Backup	Sim
Protecção de directorias	Sim
SSH	Não

Tabela 5.13: Características das Ferramentas de Gestão.

Característica	Valor
24h via E-mail (ptdesk.com)	Sim
Telefone 24/h	Não

Tabela 5.14: Características das Ferramentas de Gestão.

5.5.3 Divulgação online

No caso deste projeto, a divulgação da nova aplicação não fazia parte das tarefas definidas, e, uma vez que a aplicação FrontOffice (parte do cliente) ficou embebida no site já existente da Casa do Pastor, essa divulgação, de certa forma, é feita automaticamente, uma vez que se encontra num site que os clientes, à partida, já conhecem. No entanto, poderia ter sido enviado um email a todos os clientes (que tivessem facultado o seu email à Casa do Pastor), de forma a informar desta nova ferramenta.

5.5.3.1 Criação da base de dados e transferência de ficheiros da aplicação

A primeira coisa a fazer quando se quer colocar uma aplicação online (no caso de utilizar base de dados) é criar a base de dados no servidor, importando todas as tabelas da base de dados existente no computador para a nova base de dados e, de seguida, transferir os ficheiros para o servidor.

Assim que a base de dados é colocada no servidor, é necessário atualizar as ligações à base de dados em todas as páginas que necessitem essa ligação, obtendo como resultado final o visível na listagem 15. Por questões de segurança, a password foi colocada apenas com asteriscos. Podemos ver que é criada uma conexão, a qual é seguidamente testada (linhas 4 e 5, respetivamente). Apesar de esta ser a versão final da ligação, quando foi para o servidor de testes, uma vez que tinha dados diferentes, a ligação tinha diferentes dados.

A transferência dos ficheiros para o servidor foi feito com recurso à ferramenta Core FTP LE, como referido no capítulo 'Tecnologias Utilizadas'. Todos os ficheiros têm que ser enviados para o servidor, desde páginas, a folhas de estilo, templates, imagens, etc.

Listagem 15 Ligação á base de dados do servidor final.

```
1:  $servidor = "localhost";
2:  $utilizador = "casadopa_reserva";
3:  $password = "********";
4:  $connection = mysql_connect($servidor, $utilizador, $password);
5:  if (!$connection) die("Sem ligação à base de dados: " . mysql_error());
6:  $BD = "casadopa_reservas";
7:  $db = mysql_select_db($BD , $connection);
```

5.5.3.2 Deteção e correção de erros

Inicialmente toda a aplicação foi testada na minha máquina, com recurso ao software XAMPP para simular o servidor, no entanto, e tal como esperado, assim que se testou a aplicação online apareceram vários erros, alguns deles inesperados, como por exemplo:

- Reencaminhamentos entre páginas não funcionavam;
- Cookies e variáveis de sessão não ficavam guardados;

Anteriormente ao servidor, quando tudo era testado na minha máquina, criar uma variável de sessão era tão simples quanto o seguinte código PHP: `$_SESSION['npessoas'] = $npessoas`, tal como para guardar cookies era apenas necessário utilizar o código PHP seguinte: `setcookie("gestorDados", $idgestor, time()+3600)`.

No entanto, como nos dias que correm, nenhum browser (por definição) guarda cookies ou variáveis de sessão, foi necessário arranjar uma alternativa. Essa alternativa passou por criar uma função em JavaScript, tal como se pode ver na listagem 16.

Listagem 16 Função JavaScript para guardar cookies.

```
1: function SetCookie(cookieName,cookieValue,nDays) {
2:   var today = new Date();
3:   var expire = new Date();
4:   if (nDays==null || nDays==0) nDays=1;
5:   expire.setTime(today.getTime() + 3600000*24*nDays);
6:   document.cookie = cookieName+"="+escape(cookieValue)+";
7:   expires="+expire.toGMTString()+"; path=/"+"; domain=.<?php
8:   echo $_SERVER['HTTP_HOST']; ?>";
9: }
```

Assim, para criar o cookie, era apenas necessário utilizar o seguinte código:

`<script>SetCookie('gestorDados', $idgestor,(1/24))</script>`.

Por fim, o problema com o reencaminhamento entre páginas. Inicialmente era utilizado apenas o simples código em PHP `header("location: Home.php")`, mas

como no servidor não funcionava, teve que se recorrer mais uma vez ao JavaScript, da seguinte forma: `<script language='javaScript'>window.location.href='Home.php' </script>`.

Capítulo 6

Conclusões e trabalho futuro

6.1 Conclusões

Durante a elaboração do projeto em contexto de estágio tive a oportunidade de trabalhar com um conjunto diversificado de tecnologias que estão a ser cada vez mais utilizadas na realização de aplicações web, uma vez que estas são completamente gratuitas.

Quase todas as funcionalidades previstas no início do projeto foram implementadas, sendo que foram umas mais difíceis que outras, mas, resumidamente, estou satisfeito com tudo aquilo que consegui implementar neste projeto.

Uma das maiores dificuldades encontradas foi, uma vez que o produto final seria colocado num servidor Linux, ter que fazer todo o projeto em PHP; ao início foi complicado pois nunca tinha utilizado esta linguagem de programação, que, apesar de ser muito interessante e fácil de implementar, os passos inicias são complicados, principalmente quando se tem que aprender o grosso da linguagem sozinho. No entanto, apesar desta pequena dificuldade, penso ser uma mais valia, pois é mais uma ferramenta que fico de certo modo a dominar e que posso muito bem vir a utilizar futuramente.

Com este projeto aprendi que nem sempre tudo o que funciona quando testado na nossa máquina, funciona no servidor final, como por exemplo a questão dos cookies, que inicialmente feitos simplesmente em PHP funcionavam perfeitamente na minha máquina, mas quando testado no servidor, o web-browser não guardava qualquer cookie nem variável de sessão, obrigando-me assim a recorrer também ao JavaScript.

6.2 Trabalho futuro

Na generalidade, todas as funcionalidades previstas inicialmente foram implementadas no projeto, mas, como a aplicação web oficial da Casa do Pastor contém conteúdo em português e em inglês, uma opção futura, será a possibilidade de o cliente poder ver as páginas de calendário e criação da reserva (páginas do lado do cliente) em qualquer língua.

Uma outra funcionalidade a implementar será uma pequena página de pesquisas, que devido à falta de tempo não pôde ser implementado (apesar de nem ter sido inicialmente pensado), mas poderá facilitar aos gestores e ao administrador encontrar uma certa reserva (por exemplo) de forma mais simples.

Estas foram algumas das ideias que tive para uma implementação futura na aplicação web já desenvolvida, que poderão ser apenas implementadas ao continuar a trabalhar conjuntamente com a empresa Ideias Soberbas.

Bibliografia

[1] Adobe. Interrupção do flash para mobile. http://blogs.adobe.com/digitalmedia/, Novembro 2011.

[2] arsys. Domínios - informação geral. http://www.arsys.pt/ajuda/directorio/produtos/dominios/informacao-geral-dominios.htm, 2012.

[3] Pedro Alexandre Coelho. *HTML 4 & XHTML*. FCA, Editora de Informática, Lda., Av. Praia da Vitória, 14 A - 1000-247 Lisboa, 2 edition, 2001.

[4] World Wide Wib Consortiun. About w3c. http://www.w3.org/Consortium/, 2012.

[5] World Wide Wib Consortiun. Extensible markup language (xml). http://www.w3.org/XML/, Janeiro 2012.

[6] World Wide Wib Consortiun. HTML5. http://www.w3.org/TR/html5/introduction.html#history-1, Março 2012.

[7] Luís Manuel Dias Damas. *SQL Structured Query Language*. FCA, Editora de Informática, Lda., Av. Praia da Vitória, 14 A - 1000-247 Lisboa, 7 edition, 2005.

[8] Ricardo Oliveira / Nuno Fernandes. *Apache: Instalação, Configuraçãao e Gestão de Serviços Web*. FCA, Editora de Informática, Lda., Av. Praia da Vitória, 14 A - 1000-247 Lisboa, 2006.

[9] Core FTP. Core ftp. http://www.coreftp.com/, Junho 2012.

[10] hocnet.pt. alojamento partilhado. http://www.hocnet.pt/alojamento/index.php?action=compararPlanosPartilhadoLinux, Setembro 2012.

[11] Steve Jobs. Thoughts on flash. http://www.apple.com/hotnews/thoughts-on-flash/, Abril 2010.

[12] Carlos Serrão / Joaquim Marques. *Programação com PHP5*. FCA, Editora de Informática, Lda., Av. Praia da Vitória, 14 A - 1000-247 Lisboa, 2007.

[13] Kevin Yank. Simply javascript. http://www.sitepoint.com/simply-javascript/, Junho 2007.

Anexos

Anexo A

Listagens

A.1 Página de consulta do calendário do FrontOffice

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
<style type="text/css">
<link rel='stylesheet' type='text/css' href='CSS/csscliente.css'>
</style>
</head>
<body>

<table width="100%" border="0" cellspacing="0" cellpadding="0" align="center">
  <tr>
    <td align="center">

    <h2>Consultar Calendário</h2>
    <?php

    $anoactual = date("Y");
    $mesactual = date("m");

    $servidor = "localhost";
    $utilizador = "casadopa_reserva";
    $password = "********";

    $connection = mysql_connect($servidor, $utilizador, $password);
    if (!$connection) die("Sem ligação à base de dados: " . mysql_error());
    $BD = "casadopa_reservas";
    $db = mysql_select_db($BD , $connection);
    if (isset($_POST['idcasa']))
    {
      $IDCasa = $_POST['idcasa'];
      $Ano = $_POST['anoescolhido'];
      $Mes = $_POST['mesescolhido'];

      if (($Ano == $anoactual) && ($Mes < $mesactual))
      {
        $smsalerta = 1;
        $Mes = date("m");
      }
    }
    else
```

```
{
  $Ano = $anoactual;
  $Mes = $mesactual;
  $sql3="select min(ID_Casa) from casas";
  $result3 = mysql_query($sql3, $connection);
  if(!$result3) die("Erro, registo não efectuado: " . mysql_error());

  while($registo3 = mysql_fetch_array($result3)) {
    $IDCasa = $registo3['min(ID_Casa)'];
  }
}

$sqlcasas = "select * from casas";
$resultcasas = mysql_query($sqlcasas, $connection);
if(!$resultcasas) die("Erro, registo não efectuado: " . mysql_error());

?>

<form method="post" action="CalendarioCli.php">

<table width="50%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td align="right" valign="middle"><b>Nome da Casa: </b></td>
    <td align="left" valign="middle" ><select name="idcasa">

    <?php

    while ($registocasas = mysql_fetch_array($resultcasas)) {
      $NomeCasa = $registocasas['Nome'];
      $ID_Casa = $registocasas['ID_Casa'];

      if ($ID_Casa == $IDCasa)
      {
        ?>
        <option value="<?php echo $ID_Casa; ?>" selected="selected"><?php echo $NomeCasa; ?></option>
        <?php
      }
      else
      {
        ?>
        <option value="<?php echo $ID_Casa; ?>"><?php echo $NomeCasa; ?></option>
        <?php
      }
    }
    ?>
    </select> </td> </tr>
  <tr>
    <td align="right" valign="middle"><b>Ano: </b></td>
    <td align="left" valign="middle" ><select name="anoescolhido">

    <?php

    for ($i = $anoactual; $i <  $anoactual+5; $i++)
    {
      if ($i == $Ano)
      {
        ?>
        <option value="<?php echo $i; ?>" selected="selected"><?php echo $i; ?></option>
        <?php
      }
      else
      {
        ?>
        <option value="<?php echo $i; ?>"><?php echo $i; ?></option>
        <?php
      }
    }
    ?>
    </select>
    </td>
```

```
</tr>

<tr>
  <td align="right" valign="middle"><b>Mês: </b></td>
  <td align="left" valign="middle" ><select name="mesescolhido">
  <?php
  if ($Mes == 1)
  {
    ?> <option value="1" selected="selected">Janeiro</option> <?php
  }
  else
  {
    ?> <option value="1">Janeiro</option> <?php
  }

  if ($Mes == 2)
  {
    ?> <option value="2" selected="selected">Fevereiro</option> <?php
  }
  else
  {
    ?> <option value="2">Fevereiro</option> <?php
  }

  if ($Mes == 3)
  {
    ?> <option value="3" selected="selected">Março</option> <?php
  }
  else
  {
    ?> <option value="3">Março</option> <?php
  }

  if ($Mes == 4)
  {
    ?> <option value="4" selected="selected">Abril</option> <?php
  }
  else
  {
    ?> <option value="4">Abril</option> <?php
  }

  if ($Mes == 5)
  {
    ?> <option value="5" selected="selected">Maio</option> <?php
  }
  else
  {
    ?> <option value="5">Maio</option> <?php
  }

  if ($Mes == 6)
  {
    ?> <option value="6" selected="selected">Junho</option> <?php
  }
  else
  {
    ?> <option value="6">Junho</option> <?php
  }

  if ($Mes == 7)
  {
    ?> <option value="7" selected="selected">Julho</option> <?php
  }
  else
  {
    ?> <option value="7">Julho</option> <?php
  }

  if ($Mes == 8)
```

```
    {
      ?> <option value="8" selected="selected">Agosto</option> <?php
    }
    else
    {
      ?> <option value="8">Agosto</option> <?php
    }

    if ($Mes == 9)
    {
      ?> <option value="9" selected="selected">Setembro</option> <?php
    }
    else
    {
      ?> <option value="9">Setembro</option> <?php
    }

    if ($Mes == 10)
    {
      ?> <option value="10" selected="selected">Outubro</option> <?php
    }
    else
    {
      ?> <option value="10">Outubro</option> <?php
    }

    if ($Mes == 11)
    {
      ?> <option value="11" selected="selected">Novembro</option> <?php
    }
    else
    {
      ?> <option value="11">Novembro</option> <?php
    }

    if ($Mes == 12)
    {
      ?> <option value="12" selected="selected">Dezembro</option> <?php
    }
    else
    {
      ?> <option value="12">Dezembro</option> <?php
    }
    ?>
    </select>
    </td>
  </tr>
</table>

<p><INPUT TYPE="image" SRC="Imagens/btVerCalendario.png" HEIGHT="25" WIDTH="110" BORDER="0"
ALT="Submit Form"></p>

</form>

<?php
$numDias = cal_days_in_month(CAL_GREGORIAN, $Mes, $Ano);

?>

<table width="55%" border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td align="center" valign="middle"><b>Dia do Mês</b></td>
    <td align="center" valign="middle" ><b>Casa Ocupada</b> <img src='Imagens/apagar.png'
    align="absmiddle" width='28' height='28' alt='ocupado'/> <b>/ Livre </b><img src='Imagens/livre.png'
    align="absmiddle" width='28' height='28' alt='ocupado'/></td>
  </tr>
  <?php
  for ($i = 1; $i <= $numDias; $i++)
  {
    ?>
```

```
        <tr>
          <td align="center" valign="middle"><b><?php echo $i ?> </b></td>
          <?php
          $data = $Ano.'-'.$Mes.'-'.$i;
          $sqlcasaocup = "select count(*) from casasreserva where Data_Entrada <= '" . $data . "'
          and Data_Saida >= '" . $data . "' and ID_Casa = " . $IDCasa;
          $resultcasaocup = mysql_query($sqlcasaocup, $connection);
          if(!$resultcasaocup) die("Erro, registo não efectuado: " . mysql_error());

          while ($registocasaocup = mysql_fetch_array($resultcasaocup)) {
            $casaOcupada = $registocasaocup['count(*)'];
          }
          if ($casaOcupada == 1)
          {
            ?> <td align="center" valign="middle"><img src='Imagens/apagar.png' width='28' height='28'
            alt='ocupado'/></td> <?php
          }
          else
          {
            ?> <td align="center" valign="middle"><img src='Imagens/livre.png' width='28' height='28'
            alt='ocupado'/></td> <?php
          }
          ?>
        </tr>
        <?php
      }
      ?>
    </table>

    </td>
</tr>
</table>

<?php
if ($smsalerta == 1)
{
  ?><script language='javaScript'>window.alert('Não pode consultar datas anteriores.')</script><?php
}

function formatadata($data, $tipo = 1) {
  $data = str_replace('-', '/', $data);
  $dividir = explode("/", $data);
  $parte1 = $dividir[0]; $parte2 = $dividir[1]; $parte3 = $dividir[2];
  $data = $parte3.'-'.$parte2.'-'.$parte1;
  if ($tipo == 1) $data = str_replace('-', '/', $data);
  return $data;
}
?>

</body>
</html>
```

A.2 Script SQL para criação da base de dados no servidor MySQL

```
-- phpMyAdmin SQL Dump
-- version 3.4.10.1
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Generation Time: Oct 09, 2012 at 12:16 PM
-- Server version: 5.5.21
-- PHP Version: 5.2.17
```

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Database: `casadopa_reservas`
--
CREATE DATABASE `casadopa_reservas` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `casadopa_reservas`;

-- --------------------------------------------------------

--
-- Table structure for table `actividades`
--

CREATE TABLE IF NOT EXISTS `actividades` (
  `ID_Actividade` int(2) NOT NULL AUTO_INCREMENT,
  `Nome` varchar(30) NOT NULL,
  `Max_Pessoas` int(2) NOT NULL,
  `Idade_Min` int(2) NOT NULL,
  PRIMARY KEY (`ID_Actividade`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;

-- --------------------------------------------------------

--
-- Table structure for table `actividadesreserva`
--

CREATE TABLE IF NOT EXISTS `actividadesreserva` (
  `ID_Reserva` int(5) NOT NULL,
  `ID_Actividades` int(2) NOT NULL,
  `N_Pessoas` int(2) NOT NULL,
  `Data` date NOT NULL,
  PRIMARY KEY (`ID_Reserva`,`ID_Actividades`),
  KEY `actividades@002freserva_ibfk_2` (`ID_Actividades`,`ID_Reserva`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Table structure for table `administrador`
--

CREATE TABLE IF NOT EXISTS `administrador` (
  `ID_Administrador` int(2) NOT NULL AUTO_INCREMENT,
  `Nome` varchar(50) NOT NULL,
  `Password` varchar(50) DEFAULT NULL COMMENT 'md5->sha1',
  PRIMARY KEY (`ID_Administrador`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;

-- --------------------------------------------------------

--
-- Table structure for table `casas`
--

CREATE TABLE IF NOT EXISTS `casas` (
  `ID_Casa` int(2) NOT NULL AUTO_INCREMENT,
  `Nome` varchar(50) NOT NULL,
  `Tipologia` varchar(10) NOT NULL,
  `Capacidade` int(2) NOT NULL,
  `Preco_Epoca_Baixa` int(3) NOT NULL,
  `Preco_Epoca_Alta` int(3) NOT NULL,
```

```
  PRIMARY KEY (`ID_Casa`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=9 ;

-- --------------------------------------------------------

--
-- Table structure for table `casasreserva`
--

CREATE TABLE IF NOT EXISTS `casasreserva` (
  `ID_Reserva` int(5) NOT NULL,
  `ID_Casa` int(2) NOT NULL,
  `N_Pessoas` int(2) NOT NULL,
  `N_Camas_Extra` int(1) NOT NULL,
  `N_Bercos` int(1) NOT NULL,
  `Data_Entrada` date NOT NULL,
  `Data_Saida` date NOT NULL,
  `Preco_Casa` int(5) NOT NULL,
  PRIMARY KEY (`ID_Reserva`,`ID_Casa`),
  KEY `casas@002freserva_ibfk_2` (`ID_Casa`,`ID_Reserva`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Table structure for table `clientes`
--

CREATE TABLE IF NOT EXISTS `clientes` (
  `ID_Cliente` int(5) NOT NULL AUTO_INCREMENT,
  `Nome` varchar(50) NOT NULL,
  `Morada` varchar(100) NOT NULL,
  `Mail` varchar(50) NOT NULL,
  `Telefone` int(15) NOT NULL,
  `Telemovel` int(15) NOT NULL,
  `BI` text NOT NULL,
  PRIMARY KEY (`ID_Cliente`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=42 ;

-- --------------------------------------------------------

--
-- Table structure for table `estado`
--

CREATE TABLE IF NOT EXISTS `estado` (
  `ID_Estado` int(2) NOT NULL AUTO_INCREMENT,
  `Descricao` varchar(30) NOT NULL,
  PRIMARY KEY (`ID_Estado`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;

-- --------------------------------------------------------

--
-- Table structure for table `estadoreserva`
--

CREATE TABLE IF NOT EXISTS `estadoreserva` (
  `ID_Reserva` int(5) NOT NULL,
  `ID_Estado` int(2) NOT NULL,
  `ID_Gestor` int(2) DEFAULT NULL,
  `Data` date NOT NULL,
  PRIMARY KEY (`ID_Reserva`),
  UNIQUE KEY `ID_Reserva` (`ID_Reserva`),
  KEY `estdo@002freserva_ibfk_2` (`ID_Estado`),
  KEY `estado@002freserva_ibfk_2` (`ID_Gestor`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------
```

```
--
-- Table structure for table `gestores`
--

CREATE TABLE IF NOT EXISTS `gestores` (
  `ID_Gestor` int(2) NOT NULL AUTO_INCREMENT,
  `Nome` varchar(50) NOT NULL,
  `Telemovel` int(9) NOT NULL,
  `Mail` varchar(50) NOT NULL,
  `BI` int(8) NOT NULL,
  `Morada` varchar(100) NOT NULL,
  `Password` varchar(50) DEFAULT NULL COMMENT 'md5->sha1',
  PRIMARY KEY (`ID_Gestor`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;

-- --------------------------------------------------------

--
-- Table structure for table `historico`
--

CREATE TABLE IF NOT EXISTS `historico` (
  `Data` date NOT NULL,
  `IDGestor` int(2) NOT NULL,
  `Operacao` varchar(500) CHARACTER SET utf8 NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `reservas`
--

CREATE TABLE IF NOT EXISTS `reservas` (
  `ID_Reserva` int(5) NOT NULL AUTO_INCREMENT,
  `NomeReserva` varchar(40) NOT NULL,
  `ID_Cliente` int(5) NOT NULL,
  `N_Criancas` int(2) NOT NULL,
  `N_Bebes` int(2) NOT NULL,
  `N_Adultos` int(2) NOT NULL,
  `N_Seniors` int(2) NOT NULL,
  `N_Pessoas` int(2) NOT NULL,
  `N_Animais` int(2) NOT NULL,
  `N_Camas_Extra` int(2) NOT NULL,
  `N_Bercos` int(2) NOT NULL,
  `Data_Entrada` date NOT NULL,
  `Data_Saida` date NOT NULL,
  `Preco_Reserva` float NOT NULL,
  `Desconto` int(3) DEFAULT NULL COMMENT 'Percentagem',
  PRIMARY KEY (`ID_Reserva`),
  KEY `reservas@002freserva_ibfk_2` (`ID_Cliente`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=45 ;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `actividadesreserva`
--
ALTER TABLE `actividadesreserva`
  ADD CONSTRAINT `actividadesreserva_ibfk_1` FOREIGN KEY (`ID_Reserva`) REFERENCES `reservas` (`ID_Reserva`),
  ADD CONSTRAINT `actividadesreserva_ibfk_2` FOREIGN KEY (`ID_Actividades`)
  REFERENCES `actividades` (`ID_Actividade`);

--
-- Constraints for table `casasreserva`
--
ALTER TABLE `casasreserva`
  ADD CONSTRAINT `casasreserva_ibfk_1` FOREIGN KEY (`ID_Reserva`) REFERENCES `reservas` (`ID_Reserva`),
```

```
  ADD CONSTRAINT `casasreserva_ibfk_2` FOREIGN KEY (`ID_Casa`) REFERENCES `casas` (`ID_Casa`);

--
-- Constraints for table `estadoreserva`
--
ALTER TABLE `estadoreserva`
  ADD CONSTRAINT `estadoreserva_ibfk_1` FOREIGN KEY (`ID_Reserva`) REFERENCES `reservas` (`ID_Reserva`),
  ADD CONSTRAINT `estadoreserva_ibfk_2` FOREIGN KEY (`ID_Gestor`) REFERENCES `gestores` (`ID_Gestor`),
  ADD CONSTRAINT `estdo@002freserva_ibfk_2` FOREIGN KEY (`ID_Estado`) REFERENCES `estado` (`ID_Estado`);

--
-- Constraints for table `reservas`
--
ALTER TABLE `reservas`
  ADD CONSTRAINT `reservas@002freserva_ibfk_2` FOREIGN KEY (`ID_Cliente`) REFERENCES `clientes` (`ID_Cliente`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

A.3 Página web inicial da aplicação BackOffice

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><!-- InstanceBegin template="/Templates/template.dwt"
codeOutsideHTMLIsLocked="false" -->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!-- InstanceBeginEditable name="doctitle" -->
<title>Login</title>
<!-- InstanceEndEditable -->
<style type="text/css">
body {
  background-image: url(Imagens/exfundo1.jpg);
  background-repeat: repeat;
}
</style>

<!-- Start css3menu.com HEAD section -->
<link rel="stylesheet" href="@FILES_PATH@/style.css" type="text/css" />
<!-- End css3menu.com HEAD section -->

<!-- InstanceBeginEditable name="head" -->
<style type="text/css">
.mensagemerro {
  color: #F00;
}
</style>
<!-- InstanceEndEditable -->
</head>

<body>
<table width="900px" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
<td colspan="3" height="25px" background="exfundo/superior.png"> </td>
</tr>
  <tr>
    <td colspan="3" align="center" valign="middle" background="exfundo/interior.png"><img src="Imagens/logo.png"
    alt="logo" width="800" height="60" align="middle" /></td>
  </tr>
  <tr>
    <td colspan="3" background="exfundo/interior.png"> </td>
  </tr>
  <tr>
    <td width="650" colspan="2" valign="top" background="exfundo/interior.png"><!-- InstanceBeginEditable
```

```
    name="EditRegion1" -->

    <?php
    if (isset($_REQUEST['Mensagem']) != null)
    {
      $Mensagem = $_REQUEST['Mensagem'];
    }
    else
    {
      $Mensagem = '';
    }

    if ($Mensagem == 1)
    {
      $Mensagem = "Sessão Expirada!";
    }
    else if ($Mensagem == 2)
    {
      $Mensagem = "Dados Inválidos!";
    }
  ?>
      <p> </p>
      <p> </p>
      <p> </p>

      <form method="post" action="checkuser.php" name="form">

      <table width="50%" border="0" cellspacing="0" cellpadding="0" align="center">
        <tr>
          <td align="right" valign="middle">Login: </td>
          <td align="left" valign="middle"><input name="login" type="text" width="30px"/></td>
        </tr>
        <tr>
          <td align="right" valign="middle">Password: </td>
          <td align="left" valign="middle"><input name="password" type="password" width="30px"/></td>
        </tr>
        <tr>
          <td align="right" valign="middle"><INPUT TYPE="image" SRC="Imagens/btEntrar.png" HEIGHT="25"
          WIDTH="50" BORDER="0" alt="Submit Form"></td>
          <td align="left" valign="middle"></td>
        </tr>
        <tr>
          <td align="right" valign="middle"></td>
          <td align="left" valign="middle"><h4><span class="mensagemerro"><?php echo $Mensagem ?></span>
          </h4> </td>
        </tr>
        <tr>
          <td align="right" valign="middle"><p><input type="image" SRC="Imagens/btCliqueAqui.png" HEIGHT="25"
          WIDTH="110" BORDER="0" onclick="form.action='RecuperarPassword.php?Mensagem='; form.submit()"/> </p></td>
          <!--<td align="right" valign="middle"><p><a onclick="javascript: location.href=
          'RecuperarPassword.php?Mensagem=';"><input type="image" SRC="Imagens/btCliqueAqui.png" HEIGHT="25"
          WIDTH="110" BORDER="0" /></a></p></td>-->
          <td align="left" valign="middle">Para recuperar a password</td>
        </tr>
      </table>
    </form>

      <p> </p>
      <p> </p>
      <p> </p>

  <!-- InstanceEndEditable --></td>
  </tr>
  <tr>
<td colspan="3" height="25px" background="exfundo/inferior.png"> </td>
</tr>
</table>
</body>
<!-- InstanceEnd --></html>
```

A.4 Guardar cookies dos Gestores/Administrador

```
<script type="text/javascript">
function SetCookie(cookieName,cookieValue,nDays) {
  var today = new Date();
  var expire = new Date();
  if (nDays==null || nDays==0) nDays=1;
  expire.setTime(today.getTime() + 3600000*24*nDays);
  document.cookie = cookieName+"="+escape(cookieValue)+"; expires="+expire.toGMTString()+"; path=/"+";
    domain=.<?php echo $_SERVER['HTTP_HOST']; ?>";
}
</script>

<?php

ob_start();

$nomegestor = $_POST['login'];
$passwordgestor = md5(sha1($_POST['password']));

$servidor = "localhost";
$utilizador = "casadopa_reserva";
$password = "reservas";

$connection = mysql_connect($servidor, $utilizador, $password);
if (!$connection) die("Sem ligação à base de dados: " . mysql_error());
$BD = "casadopa_reservas";
$db = mysql_select_db($BD , $connection);

$idgestor = 0;
$idadmin = 0;

$sqlgestor="select * from gestores where Nome = '$nomegestor' and Password = '$passwordgestor'";
$resultgestor = mysql_query($sqlgestor, $connection);
if(!$resultgestor) die("Erro, registo não efectuado: " . mysql_error());

while($registogestor = mysql_fetch_array($resultgestor)) {
  $idgestor = $registogestor['ID_Gestor'];
}

$sqladmin="select * from administrador where Nome = '$nomegestor' and Password = '$passwordgestor'";
$resultadmin = mysql_query($sqladmin, $connection);
if(!$resultadmin) die("Erro, registo não efectuado: " . mysql_error());

while($registoadmin = mysql_fetch_array($resultadmin)) {
  $idadmin = $registoadmin['ID_Administrador'];
  $idgestor = 99999;
}

if ($idgestor != 0) {
  if ($idgestor == 99999) {
    echo "<script>SetCookie('gestorDados', '0',(1/24))</script>";
    echo "<script>SetCookie('adminDados', $idadmin,(1/24))</script>";
  }
  else {
    echo "<script>SetCookie('gestorDados', $idgestor,(1/24))</script>";
  }

  ?><script language='javaScript'>window.location.href='Home.php'</script><?php
}
else {
  ?><script language='javaScript'>window.location.href='index.php?Mensagem=2'</script><?php
}
?>
```

A.5 Eliminar cookies dos Gestores/Administrador

```
<script language="JavaScript">

function SetCookie(cookieName,cookieValue,nDays) {
  var today = new Date();
  var expire = new Date();
  if (nDays==null || nDays==0) nDays=1;
  expire.setTime(today.getTime() -1);
  document.cookie = cookieName+"="+escape(cookieValue)+"; expires="+expire.toGMTString()+"; path=/"+";
  domain=.<?php echo $_SERVER['HTTP_HOST']; ?>";
}
</script>

<?php
if (isset($_COOKIE['gestorDados'])) {
  echo "<script>SetCookie('gestorDados', '0',(1/24))</script>";
}

if (isset($_COOKIE['adminDados'])) {
  echo "<script>SetCookie('adminDados', '0',(1/24))</script>";
}

?><script language='javaScript'>window.location.href='index.php'</script><?php
?>
```

A.6 Exemplo de envio de email de confirmação

```
$sqldadosmail="select * from reservas where ID_Reserva = '$IDReserva'";
$resultdadosmail = mysql_query($sqldadosmail, $connection);
if(!$resultdadosmail) die("Erro, registo não efectuado: " . mysql_error());
while($registodadosmail = mysql_fetch_array($resultdadosmail)) {
  $ID_Cliente_Mail = $registodadosmail['ID_Cliente'];
  $N_Pessoas_Mail = $registodadosmail['N_Pessoas'];
  $Data_Entrada_Mail = $registodadosmail['Data_Entrada'];
  $Data_Saida_Mail = $registodadosmail['Data_Saida'];
  $Preco_Reserva_Mail = $registodadosmail['Preco_Reserva'];
}

$sqldadosmail2="select * from clientes where ID_Cliente = '$ID_Cliente_Mail'";
$resultdadosmail2 = mysql_query($sqldadosmail2, $connection);
if(!$resultdadosmail2) die("Erro, registo não efectuado: " . mysql_error());
while($registodadosmail2 = mysql_fetch_array($resultdadosmail2)) {
  $Nome_Cliente_Mail = $registodadosmail2['Nome'];
  $Mail_Cliente_Mail = $registodadosmail2['Mail'];
}

$destinatario = $Mail_Cliente_Mail;
$assunto = "Confirmacao Reserva";
$corpo = '
         <html>
         <head>
           <title>Confirmacao Reserva</title>
         </head>
         <body>
         <h1>Ola '.$Nome_Cliente_Mail.'!</h1>
         <p>
         <b>A sua reserva foi aceite.</b> Verifique se todos os dados estao correctos.
         </p>
         <p></p>
```

```
          <p>Numero de Pessoas: '.$N_Pessoas_Mail.'</p>
          <p>Preco da Reserva: '.$Preco_Reserva_Mail.' Euros</p>
          <p>Data de Entrada: '.$Data_Entrada_Mail.'</p>
          <p>Data de Saida: '.$Data_Saida_Mail.'</p>
          <p></p>
          <p><h2>Dados das Casas da Reserva</h2></p>';

$sqldadosmail3="select * from casasreserva where ID_Reserva = '$IDReserva'";
$resultdadosmail3 = mysql_query($sqldadosmail3, $connection);
if(!$resultdadosmail3) die("Erro, registo não efectuado: " . mysql_error());
while($registodadosmail3 = mysql_fetch_array($resultdadosmail3)) {
  $ID_Casa_Mail = $registodadosmail3['ID_Casa'];
  $Preco_Casa_Mail = $registodadosmail3['Preco_Casa'];

  $sqldadosmail4="select * from casas where ID_Casa = '$ID_Casa_Mail'";
  $resultdadosmail4 = mysql_query($sqldadosmail4, $connection);
  if(!$resultdadosmail4) die("Erro, registo não efectuado: " . mysql_error());
  while($registodadosmail4 = mysql_fetch_array($resultdadosmail4)) {
    $Nome_Casa_Mail = $registodadosmail4['Nome'];
  }

  $corpo = $corpo.'
                <p>'.$Nome_Casa_Mail.' -> '.$Preco_Casa_Mail.' Euros</p>
                ';
}

$corpo = $corpo.'
               <p></p>
               <p></p>
               <p><h5>Nao responda a este email. Em caso de duvidas entre em contacto com os responsaveis</h5></p>
               </body>
               </html>
               ';

//para o envio em formato HTML
$headers = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html;
charset=iso-8859-1\r\n";

//endereço do remitente
$headers .= "From: casadopastor <casadopastor@hotmail.com>\r\n";

mail($destinatario,$assunto,$corpo,$headers);
```

A.7 Exemplo de validação de dados

```
function ValidaData($dat){
  $data = str_replace("/",", ",$dat);
  $stamp = strtotime( $data );
  $m = date( 'm', $stamp );
  $d   = date( 'd', $stamp );
  $y  = date( 'Y', $stamp );
  $dataActual = date("Y/m/d"); // data atual
  $res = 0;
  if (checkdate($m, $d, $y)) {
    if (strtotime($dat) >= $dataActual) {
      $res = 1;
    }
  }
  return $res;
}


if (!is_numeric($npessoas)) {
  ?><script language='javaScript'>window.location.href='CriarReserva.php?Mensagem=1'</script><?php
}
```

```
if ((ValidaData($dataentrada) == 0) || (ValidaData($dataentrada) == 0) || ($npessoas < 0) || ($npessoas > 99)) {
  ?><script language='javaScript'>window.location.href='CriarReserva.php?Mensagem=1'</script><?php
}
```

A.8 Validação Email

```
function validaEmail($mail){
  if(preg_match("/^([[:alnum:]_.-]){3,}@([[:lower:][:digit:]_.-]{3,})(\.[[:lower:]]{2,3})(\.[[:lower:]]{2})?$/",
  $mail)) {
    return true;
  }else{
    return false;
  }
}
```

A.9 Página reservas

```
<?php
if(isset($_COOKIE["gestorDados"]))
{
  $idGestorActivo = $_COOKIE["gestorDados"];
  ?>

  <h1>Lista de Reservas Atuais</h1>
  <p>
  <?php

  $servidor = "localhost";
  $utilizador = "casadopa_reserva";
  $password = "********";

  $connection = mysql_connect($servidor, $utilizador, $password);
  if (!$connection) die("Sem ligação à base de dados: " . mysql_error());
  $BD = "casadopa_reservas";
  $db = mysql_select_db($BD , $connection);

  $sql="select * from reservas ORDER BY  `Data_Entrada` ";

  $result = mysql_query($sql, $connection);
  if(!$result) die("Erro, registo não efectuado: " . mysql_error());

  echo "<table width='95%' border='1' cellspacing='0' cellpadding='0'>";
  echo "<tr align='center' valign='middle' bgcolor='#F3EFD4'>";
  echo "<td width='12%'><strong>Nome Reserva</strong></td>";
  echo "<td width='12%'><strong>Nome Cliente</strong></td>";
  echo "<td width='12%'><strong>Data de Entrada</strong></td>";
  echo "<td width='12%'><strong>Data de Saída</strong></td>";
  echo "<td width='12%'><strong>Estado da Reserva</strong></td>";
  echo "<td width='12%'><strong>Detalhes</strong></td>";
  echo "<td width='12%'><strong>Apagar</strong></td>";
  echo "</tr>";

  echo "<tr>";
  while($registo = mysql_fetch_array($result)) {
    $ID_Reserva = $registo['ID_Reserva'];
    $ID_Cliente = $registo['ID_Cliente'];
    $Nome_Reserva = $registo['NomeReserva'];
    $Nome_Cliente = '';
    $Data_Entrada = $registo['Data_Entrada'];
```

```
    $Data_Saida = $registo['Data_Saida'];
    $Estado = '';

    $sqlc="select * from clientes where ID_Cliente = " . $ID_Cliente;

    $resultc = mysql_query($sqlc, $connection);
    if(!$resultc) die("Erro, registo não efectuado: " . mysql_error());

    while($registoc = mysql_fetch_array($resultc)) {
      $Nome_Cliente = $registoc['Nome'];
    }

    $sqle="select * from estado, estadoreserva where estadoreserva.ID_Reserva = " . $ID_Reserva . "
    and estadoreserva.ID_Estado = estado.ID_Estado";

    $resulte = mysql_query($sqle, $connection);
    if(!$resulte) die("Erro, registo não efectuado: " . mysql_error());

    while($registoe = mysql_fetch_array($resulte)) {
      $Estado = $registoe['Descricao'];
    }

    echo "<tr align='center' valign='middle'>";
    echo "<td>$Nome_Reserva</td><td>$Nome_Cliente</td><td>$Data_Entrada</td><td>$Data_Saida</td>
    <td>$Estado</td><td><a href='ExpandirReserva.php?ID_Reserva=$ID_Reserva'><img src='Imagens/detalhes.png'
    width='28' height='28' alt='expandir'/></a></td><td><a href='EliminarReserva.php?ID_Reserva=$ID_Reserva'>
    <img src='Imagens/apagar.png' width='28' height='28' alt='apagar'/></a></td>";
    echo "</tr>";
  }
  echo "</table>";

  ?></p>

  <p><a onclick="javascript: location.href='InserirReserva.php';"><input type="image"
  SRC="Imagens/btInserirReserva.png" HEIGHT="25" WIDTH="100" BORDER="0" /></a></p>

  <?php
}
else
{
  ?><script language='javaScript'>window.location.href='index.php?Mensagem=1'</script><?php
}
?>
```

O resultado deste código é o apresentado na figura A.1.

Figura A.1: Resultado da página Reservas.