Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Diogo Alexandre Fernandes Gil
novembro | 2012

Instituto Politécnico da Guarda

Escola Superior de Tecnologia e Gestão

Aplicação Web para Gestão de Ginásio

Diogo Alexandre Fernandes Gil - Nº 1008941

Projeto Aplicado no Curso de Engenharia Informática

13 de Novembro de 2012

Instituto Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

Aplicação Web para Gestão de Ginásio

Diogo Alexandre Fernandes Gil - Nº 1008941

Projeto Aplicado no Curso
de Engenharia Informática

Supervisor: Prof. José Alberto Quitério Figueiredo - Professor na Escola Superior de Tecnologia e Gestão - IPG

Orientador: Prof. José Alberto Quitério Figueiredo da Unidade Técnico-Científica de Informática da ESTG.

13 de Novembro de 2012

Agradecimentos

Gostaria de aproveitar este espaço para fazer alguns agradecimentos. Ao longo da realização do projeto e da sua complexidade, o apoio de diversas pessoas ao longo deste tempo foi essencial.

Agradecer ao Professor José Alberto Quitério Figueiredo pelo apoio prestado em todas as fases do projeto mas acima de tudo pela motivação que me foi transmitindo ao longo do tempo.

Um agradecimento aos professores da Unidade Curricular Projeto de Informática, Professor Paulo Jorge Costa Nunes e Professor Noel de Jesus Mendonça Lopes.

À Diretora da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda pela sua disponibilidade.

Ao professor José Carlos Martins Coelho Fonseca um obrigado pelo apoio e disponibilidade no desenho do modelo de Base de Dados.

Ao colega Mário Baltazar pelo apoio e sugestões que me foi facultando da sua experiência na mesma linguagem de programação que foi usada no desenvolvimento deste projeto.

Por fim à minha família mais chegada por toda a paciência, apoio, motivação e compreensão ao longo do projeto.

A todos um muito obrigado!

Resumo

A proposta de projeto insere-se nas Tecnologias da Informação e Comunicação (TIC) para a web. Em comparação com alguns anos atrás, hoje tudo ou quase tudo do mundo tecnológico passa por estar conectado em rede (Internet). Seguindo esta ideia, foi me proposto a realização de uma aplicação informática assente na web. Para que tanto o gestor, como cliente da aplicação possa aceder em qualquer parte e a qualquer hora à aplicação.

Em suma o trabalho desenvolvido e que irá ser apresentado neste mesmo documento passou por criar uma plataforma de gestão para um ginásio na qual é possível inscrever novos clientes em modalidades e atividades, consultar dados relativos às inscrições em modalidades e atividades, consultar pagamentos, consultar pagamentos em atraso tudo de modo a ter uma gestão mais eficiente. Durante o projeto foram usadas diversas ferramentas e linguagens de programação de modo a atingir o resultado final desejado. As ferramentas usadas foram essencialmente o IDE de programação Visual Studio e a ferramenta Sql Server Management Studio, ambas da Microsoft. Quanto às linguagens de programação, a linguagem C# foi a linguagem dominante no projeto, no entanto foram usadas linguagens como Javascript, Ajax, HTML, CSS, T-SQL.

Palavras Chave

C#, HTML, Aplicação Web, Javascript, CSS, T-SQL, SQL SERVER

Abstract

The proposed project is a part of the technological evolution. Compared to a few years ago, all of the technological world or most of it is networked (Internet). Following this idea, we propose the implementation of a software application based on the web. The manager and the client can have access to the application at any time or any place.

The work that will be presented in this document is about a management platform to a gym, where you can sign up new customers in modalities and activities, consult data on registrations of modalities and activities, check payments and overdue payments to achieve a more efficient management. During the project I used several tools and programming languages to produce the desired end result. The tools used were essentially the programming IDE Visual Studio and Sql Server Management Studio tool, both from Microsoft. As for programming languages, C# was the dominant language in the project, however we used languages like Javascript, Ajax, HTML, CSS, T-SQL too.

Key words

C#, HTML, Aplicação Web, Javascript, CSS, T-SQL, SQL SERVER

Conteúdo

1 Introdução **12**
- 1.1 Enquadramento . . . 12
- 1.2 Motivação . . . 13
- 1.3 Solução . . . 13
- 1.4 Estrutura do documento . . . 13

2 Estado da Arte **14**
- 2.1 InnuxSports . . . 14
- 2.2 FitGest . . . 16
- 2.3 Análise Crítica . . . 16

3 Definição do problema e objetivos previstos **17**
- 3.1 Definição do problema . . . 17
- 3.2 Objetivos previstos . . . 20

4 Metodologia e resultados esperados **21**
- 4.1 Metodologia . . . 21
- 4.2 Descrição das tarefas . . . 21
- 4.3 Resultados esperados . . . 22

5 Tecnologias utilizadas no desenvolvimento da Solução **24**
- 5.1 Serviço Web . . . 24
- 5.2 AJAX Control Toolkit . . . 24
- 5.3 API . . . 26
- 5.4 API Sapo Mapas . . . 27
- 5.5 iText . . . 28
- 5.6 CKEditor . . . 28

6 Implementação da solução **30**
- 6.1 Introdução . . . 30
- 6.2 Construção do Template . . . 30
- 6.3 Base de Dados . . . 32
- 6.4 Dicionário de Dados . . . 34
  - 6.4.1 Atividades . . . 34
  - 6.4.2 Anualidades . . . 36
  - 6.4.3 Categorias de Notícias . . . 36
  - 6.4.4 Concelhos . . . 36

6.4.5 Definiçoes . . . 37
6.4.6 Delegações . . . 37
6.4.7 Distritos . . . 37
6.4.8 Escalão de Utilizadores . . . 38
6.4.9 Escalões . . . 38
6.4.10 Escalões de Modalidades/Actividades . . . 39
6.4.11 Faturas . . . 39
6.4.12 Freguesias . . . 39
6.4.13 Inscrições . . . 40
6.4.14 Linhas de Faturas . . . 40
6.4.15 Mensalidades . . . 41
6.4.16 Modalidades . . . 41
6.4.17 Notícias . . . 42
6.4.18 Postais . . . 42
6.4.19 Presenças . . . 43
6.4.20 Requisitos . . . 43
6.4.21 Requisitos de Modalidades/Actividades . . . 44
6.4.22 Seguros . . . 44
6.4.23 Tipos de Atividades . . . 44
6.4.24 Tipos de Utilizador . . . 45
6.4.25 Utilizadores ASP.NET . . . 45
6.4.26 Utilizadores . . . 45
6.5 Desenvolvimento de Software por 3 Tier Architecture ou Modelo em 3 Camadas . . . 46
6.6 AJAX AutoComplete Extender . . . 56
6.7 Ficheiros com Códigos Postais, Freguesias, Concelhos e Distritos Nacionais . . . 60
6.8 Sapo Mapas . . . 65
6.9 Gerar Documento Comprovativo e Envio por Email . . . 68

7 Conclusões e trabalho futuro **74**
7.1 Conclusões . . . 74
7.2 Trabalho futuro . . . 75

A Listagem de programas **79**
A.1 Template da página web . . . 79
A.2 Ficheiro CSS do Template . . . 79
A.3 Classe CodigosPostais . . . 81
A.4 Classe Email . . . 85
A.5 Classe PDF . . . 87
A.6 Classe Utilizador (3 Tier Architecture) . . . 88
A.7 Classe UtilizadorDAO (3 Tier Architecture) . . . 91
A.8 Classe UtilizadorBUS (3 Tier Architecture) . . . 93
A.9 Procedimento Aniversários . . . 93
A.10 Procedimento Inserir Escalão . . . 94
A.11 Procedimento Inserir Linha de Factura . . . 94

B Manual da Aplicação 96

Lista de Figuras

2.1 Descrição do projeto da Innux Technologies 15

3.1 DFD de Contexto 17
3.2 Diagrama Casos de Uso 19

4.1 Mapa de Gantt 22

5.1 AJAX AutoComplete Extender 25
5.2 AJAX Calendar Extender 25
5.3 Exemplo de um Mapa da Sapo 28
5.4 Controlo CKEditor 29

6.1 Template: Página Principal 31
6.2 Modelo de base de dados que suporta os serviços fornecidos pela plataforma 33
6.3 Modelo ER da solução 33
6.4 Modelo em 3 camadas 47
6.5 Estrutura do modelo em 3 camadas no Visual Studio 48
6.6 Página auxiliar de carregamento dos dados para a base de dados . . 64
6.7 Solução com Sapo Mapas 65
6.8 Documento gerado 69
6.9 Template HTML do Documento 70

Lista de Tabelas

3.1 Atores e Objetivos 18

6.1 Estrutura da tabela atividades 35
6.2 Estrutura da tabela Anualidades 36
6.3 Estrutura da tabela Categorias de Notícias 36
6.4 Estrutura da tabela Concelhos 37
6.5 Estrutura da tabela Definições 37
6.6 Estrutura da tabela Delegações 37
6.7 Estrutura da tabela Distritos 38
6.8 Estrutura da tabela Escalões de Utilizadores 38
6.9 Estrutura da tabela Escalões de Utilizadores 38
6.10 Estrutura da tabela Escalões de Modalidades/Actividades 39
6.11 Estrutura da tabela Faturas 39
6.12 Estrutura da tabela Freguesias 40
6.13 Estrutura da tabela Inscrições 40
6.14 Estrutura da tabela Linha de Faturas 41
6.15 Estrutura da tabela Mensalidades 41
6.16 Estrutura da tabela Modalidades 42
6.17 Estrutura da tabela Notícias 42
6.18 Estrutura da tabela Códigos Postais 43
6.19 Estrutura da tabela Presenças 43
6.20 Estrutura da tabela Requisitos 43
6.21 Estrutura da tabela Requisitos de Modalidades/Actividades 44
6.22 Estrutura da tabela Seguros 44
6.23 Estrutura da tabela Tipos de Atividades 45
6.24 Estrutura da tabela Tipos de Utilizador 45
6.25 Estrutura da tabela Utilizadores_Asp.NET 45
6.26 Estrutura da tabela Utilizadores 46
6.27 Tabelas da base de dados e Número de registos 65

Listagens

5.1 Configuração do ScriptManager da Solução 26
5.2 Exemplo: Adicionar um Mapa ao web site 27
5.3 Linha para que o controlo seja reconhecido pela plataforma 28
6.1 Adicionar Form 30
6.2 Classe Database do modelo em 3 camadas 48
6.3 Classe Utilizador do modelo em 3 camadas 51
6.4 Classe DAO Utilizador do modelo em 3 camadas 52
6.5 Classe BUS Utilizador do modelo em 3 camadas 54
6.6 Utilização do modelo em 3 camadas 55
6.7 Exemplo do procedimento Inserir Utilizador usado pelo modelo em 3 camadas 55
6.8 Configuração do Auto Complete Extender 57
6.9 Ficheiro AutoCompleteList.asmx 57
6.10 Ficheiro AutoCompleteListSeguros.cs 57
6.11 Ficheiro AutoCompleteListSeguros.cs com WebMethod GetCompletionListMorada 59
6.12 Passar valor do código postal para o contextKey 60
6.13 Estrutura do Ficheiro Distritos 60
6.14 Distritos.txt 61
6.15 Exemplo do ficheiro Códigos Postais 61
6.16 Método da classe CodigosPostais que permite inserir na base de dados os códigos postais 62
6.17 Código de exemplo da integração dos serviços Sapo Mapas na obtenção de coordenadas geográficas 66
6.18 Código de exemplo da integração dos serviços Sapo Mapas na obtenção de direções entre pontos 67
6.19 Método criaPDFInscricao 70
6.20 Configuração do email no ficheiro web.config 71
6.21 Método que permite enviar emails com anexos 72

Glossário

API — é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicações que não pretendem envolver-se em detalhes na implementação, apenas os seus serviços.

ASP.NET — Plataforma de desenvolvimento de aplicações Web dinâmicas da Miscrosoft.

Base de Dados — Conjunto de dados estruturados e relacionados entre si.

CSS — é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos por exemplo em HTML. O objetivo é separar o formato do conteúdo.

Framework .NET — É um modelo de programação abrangente da Miscrosoft, que visa uma plataforma única para desenvolvimento e execução de sistemas. A finalidade é ter todo e qualquer código gerado para .NET poder ser executado em qualquer dispositivo com a Framework da plataforma. [9]

HTTP — Hiper Text Transfer Protocol, ou em português, Protocolo de Transferência de Hipertexto e é um protocolo de comunicação.

HTML — Hyper Text Markup Language, é uma linguagem de marcação para produção de páginas Web.

IDE — Integrated Development Environment, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

JavaScript — é uma das linguagens de programação de script mais usadas do lado do cliente em browsers. Muito útil em projetos para web sites pois permite adicionar funcionalidades muito interessantes do lado do cliente.

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

Sql Server — É um SGBD (Sistema de Gestão de Base de Dados) criado pela Microsoft e Sysbase que permite interação com os dados através da linguagem SQL.

SQL — Structured Query Language, é uma linguagem de pesquisa declarativa para base de dados relacionais.

T-SQL — Transact-SQL, linguagem que permite alterar e definir base de dados relacionais. Expande o padrão SQL e inclui procedimentos e diversas funções (Matemáticas, Datas, etc.)

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

Visual Studio — é um pacote de programas da Microsoft para desenvolvimento de software.

XML — XML (eXtensible Markup Language) é uma linguagem de marcação que permite criar estruturas de dados de forma simples e intuitiva. Assume uma grande importância na troca de dados na web.

Capítulo 1

Introdução

O presente documento serve para descrever o projeto desenvolvido pelo aluno Diogo Fernandes no âmbito da Unidade Curricular Projeto de Informática da Escola Superior de Tecnologia e Gestão da Guarda. A aplicação desenvolvida visa ajudar na gestão de um ginásio web de modo a que os dados estejam disponíveis em qualquer parte e a qualquer hora. Os dados seriam tais como, consultar o número de inscrições, clientes com mensalidades em atraso, entre outros. Mas também foi desenvolvida a pensar na gestão das páginas web que compõem a aplicação. Uma vez que é possível criar, editar e eliminar dados, alterar o logotipo da entidade entre muitas outras funções.

Hoje existem inúmeros serviços onde por exemplo podemos alojar documentos, fotografias, vídeos gratuitamente e ainda editar e visualizar os próprios ficheiros que estão em rede. Assente nesta ideia de poder aceder aos dados quando e onde quisermos é que surgiu a ideia deste projeto, de modo a que o gestor possa fazer a gestão da sua entidade quando e onde quiser. Esta ideologia é muito interessante. Basta ver, por exemplo os serviços em crescente de cloud computing ou 'nuvem' que existem no mercado que vão desde sistemas operativos a serviços de alojamento e partilha. Afinal todos nós, utilizadores de lazer ou mais profissionais gostamos de ter os serviços sempre disponíveis e interagindo com outras pessoas e serviços.

1.1 Enquadramento

O projeto enquadra-se no âmbito do curso de modo a adquirir experiência e aprofundamento dos conhecimentos adquiridos ao longo dos três anos do curso. Os ensinamentos ao longo do curso não são suficientes para a nossa preparação, desse modo a realização do projeto veio tentar colmatar essa lacuna e que para nós alunos é de certo modo importante para adquirir experiência de modo a nos sentirmos mais preparados para o nosso futuro que é o mercado de trabalho. O projeto ajudou a melhorar também algumas aptidões pessoais como o sentido de responsabilidade, cumprimento de prazos, capacidade de trabalhar sob pressão, capacidade de definir objetivos, capacidade de focar em objetivos importantes, desviando atenções de objetivos menos importantes.

1.2 Motivação

A motivação no desenvolvimento do projeto foi essencialmente ter a possibilidade de desenvolver um projeto de uma complexidade e grandeza média/grande. Aprofundar conhecimentos previamente adquiridos numa área de interesse pessoal, mas para além disso e sabendo de ante-mão o que um projeto desta grandeza envolve, o uso de novas tecnologias e linguagens de programação serviu também de motivação no desenvolvimento do projeto.

1.3 Solução

Com a evolução da sociedade e crescente ritmo em que vivemos, as pessoas procuram cada vez mais os ginásios para fazer atividade física. Com esta aplicação pretende-se que seja possível gerir toda a atividade de um ginásio. Terá de ser capaz de gerir modalidades mais especificas como por exemplo o Karaté ou Judo. Neste tipo de atividades é necessário gerir os diferentes níveis em que os praticantes se encontram, gestão de presenças, gestão de estágios frequentados e exames de passagem de nível. A solução desenvolvida vai ao encontro do pretendido tendo um Front Office que é capaz de apresentar aos visitantes da página alguma informação sobre a instituição e notícias, entre outros. O Back Office terá a capacidade de gerir utilizadores de modo a permitir os acessos às diversas áreas do web site, inscrever clientes em modalidades/atividades, consultar dados sobre as modalidades/atividades e clientes, entre outras funcionalidades já referidas em cima.

1.4 Estrutura do documento

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

- No segundo capítulo é descrito o estado da arte com duas aplicações a servir de exemplo.
- No terceiro capítulo é descrita a definição do problema, assim como os objetivos previstos.
- No quarto capítulo é descrita a metodologia usada, a descrição das tarefas e os resultados esperados.
- No quinto capítulo são descritas as tecnologias mais importantes usadas no desenvolvimento da solução.
- No sexto capítulo é apresentado de forma mais detalhada alguns aspetos da solução.
- No sétimo e último capítulo são feitas algumas considerações finais sobre o projeto e pontos a melhorar ou a acrescentar à solução.

Capítulo 2

Estado da Arte

Procurando por aplicações semelhantes à aplicação desenvolvida no âmbito do projeto, surgem algumas aplicações interessantes. Vamos apresentar aqui duas aplicações para o mesmo efeito.

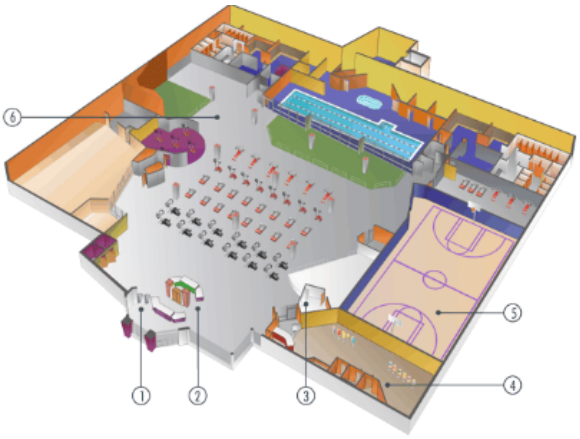
2.1 InnuxSports

A aplicação chama-se **InnuxSports** desenvolvida pela empresa Innux Technologies que se assume uma das empresas líder no mercado nacional no desenvolvimento de projetos na área da gestão de pessoas.

O InnuxSports é um sistema integrado de gestão de Ginásios e HealthClubs que permite processar a informação associada ao estabelecimento - a gestão de utentes, cartões, quotas, horários, produtos/serviços, agenda, marcações e reservas e ainda a gestão de loja/ bar.

Foi concebido a pensar em diversas atividades como ginásios, healthclubs, centros recreativos e comunitários, piscinas e centros de estética. Se possui um Ginásio com número de sócios pequeno mas em crescimento, ou um Healthclub com milhares de membros e serviços (associados, treinos, aulas, estética, reservas de espaços, um bar, entre outros) - o InnuxSports a solução completa para o seu negócio! As nossas soluções simples e eficazes vão ajudá-lo a uniformizar as operações do seu espaço, permitindo-lhe concentrar os seus esforços na angariação de novos membros e na promoção de bem-estar dos atuais. [14]

1. Entrada - Faz-se o controlo de entradas e saídas dos utilizadores ao recinto, através da leitura de cartões de associados, biometria ou pela validação electrónica de bilhetes nos torniquetes, barreiras e outros automatismos de acesso. O ginásio poderá funcionar sem colaboradores (por exemplo: em horário noturno) - neste caso, todo o controlo de entradas e saídas é efectuado electrónicamente. Após a validação de acesso dos sócios, o sistema acende automaticamente as luzes das áreas a que têm acesso.



2. Recepção - Quando os sócios se inscrevem a sua fotografia é capturada pelo software e o cartão de sócio emitido no momento, através de uma impressora de cartões. Pagam-se quotas e imprimem-se as respectivas vinhetas. Procede-se à venda de bilhetes de entrada para a utilização de equipamentos específicos, como piscinas e saunas. Os sócios e outros clientes contactam o ginásio para reservar espaços específicos dentro do mesmo. A recepcão verifica no calendário interno do InnuxSports a disponibilidade horária de cada um dos espaços e confirma a reserva.

3. Administração - Aqui são definidos os tarifários, horários e zonas de acesso. Criam-se eventos específicos definindo a sua data, hora, duração e lotação. O módulo de compras permite a gestão de compras, stocks e a verificação de contas correntes de fornecedores.

4. Loja/ Bar - A POS do InnuxSports permite a gestão de um pequeno bar ou loja através de um ecrã táctil. A sua interface amigável permite ao logista seleccionar os artigos a vender carregando com o dedo nas respectivas fotografias; o software encarrega-se do resto.

5. Recinto - O controlo de acessos de pessoas a todas as áreas do recinto é efectuado de forma automática mediante a leitura de bilhetes e cartões de associado. Este modo de funcionamento garante que cada pessoa acede apenas às areas que é tem permissão para aceder, dentro dos horários definidos pela organização.

6. Espaços - O estado de ocupação dos espaços é feito através da gestão e coordenação de horários, professores e turmas por modalidade. Em qualquer momento é possível saber quantas pessoas e que pessoas estão em cada espaço.

Figura 2.1: Descrição do projeto da Innux Technologies

2.2 FitGest

A segunda aplicação chama-se **FitGest** um produto da ng.sis e tem as seguintes características. Gestão de utentes/sócios; Gestão de aulas e modalidades; Gestão de recursos humanos; Gestão de horários; Gestão de Quotas; Gestão de Bar ou Loja; Notificações para aniversariantes; Agenda / Lembretes; Mapa de Ocupação das Salas / Aulas; Emissão Recibos, Facturas, Vendas a Dinheiro; Controlo de acessos (biometria ou cartões RFID); Multiposto e possibilidade de Acesso Remoto; Alertas de assiduídade do cliente; Estatísticas e relatórios diversos; Partilha de Ficheiros [11]

2.3 Análise Crítica

Ambas as aplicações, com base no que é conhecido, são aplicações bastante boas e completas. A aplicação desenvolvida vai ao encontro destas duas, tem funcionalidades semelhantes a ambas como a gestão de presenças (de uma forma diferente), gestão de quotas, gestão de utentes/sócios, emissão de recibos, entre outras funcionalidades que ambas as aplicações possuem. No entanto as aplicações apresentadas parecem mais completas que a aplicação desenvolvida pois têm Controlo de acessos, capacidade de gerir um bar, fazem gestão de horários entre outros pontos fortes que abonam a seu favor. No entanto nenhuma das aplicações é bastante clara quanto às modalidades ditas especiais como o caso do Judo e Karaté. Ficando a dúvida se estas aplicações são capazes de lidar com essas modalidades mais específicas que no caso da aplicação desenvolvida permite lidar com essas modalidades e outras, outro aspeto que poderá ser melhor ou pior na aplicação desenvolvida é o facto de ser uma aplicação web.

Capítulo 3

Definição do problema e objetivos previstos

3.1 Definição do problema

De maneira a compreender melhor o problema, apresenta-se de seguida o DFD de Contexto 3.1. O DFD de Contexto em Engenharia de Software representa todo o sistema como um único processo, representando os fluxos de dados que mostram as interfaces entre o sistema e as entidades externas. Resumindo, o diagrama é visto como uma forma de representar o objeto de estudo.

Figura 3.1: DFD de Contexto

Na tabela 3.1 serão apresentados os casos de uso do problema com os respetivos atores e objetivos.

Na figura 3.2, podemos consultar o diagrama genérico de casos de uso.
Em Engenharia de Software, um caso de uso é um tipo de classificador representando

Ator	Objetivos
Cliente	Consultar Inscrições Editar Dados Pessoais Receber recibos de inscrições, pagamentos
Visitante	Consultar Conteúdos
Funcionário	Inscrever Cliente Marcar Presenças Emitir Pagamentos
Administrador	Consultar Dados sobre Inscrições Consultar Dados sobre Modalidades Consultar Dados sobre Atividades
Super Administrador	Gerir Definições da Página Gerir Modalidades Gerir Mensalidades Gerir Atividades e Tipos Gerir Anuidades Gerir Mensalidades Lançar Mensalidades Gerir Utilizadores Gerir Conteúdos e Tipos Gerir Seguros Gerir Escalões Gerir Requisitos

Tabela 3.1: Atores e Objetivos

uma unidade funcional coerente provida pelo sistema, subsistema, ou classe entre os sistemas e um ou mais atores. [17]

Na conceção do trabalho proposto, os primeiros desafios a resolver foi, pensar em como tudo na aplicação seria estruturado. De seguida são apresentados alguns pontos que ajudaram a nivelar e estruturar a aplicação:

- Qual o melhor modelo de base de dados para que a aplicação seja o mais genérica possível;
- Como fazer a gestão de pagamentos;
- Como gerir as renovações;
- Como organizar os níveis de modalidades de modo a saber qual o nível em que o cliente se pode inscrever;
- Como estruturar o Front Office e o Back Office;
- Como apresentar da melhor forma os dados de análise.

Casos de Uso.png

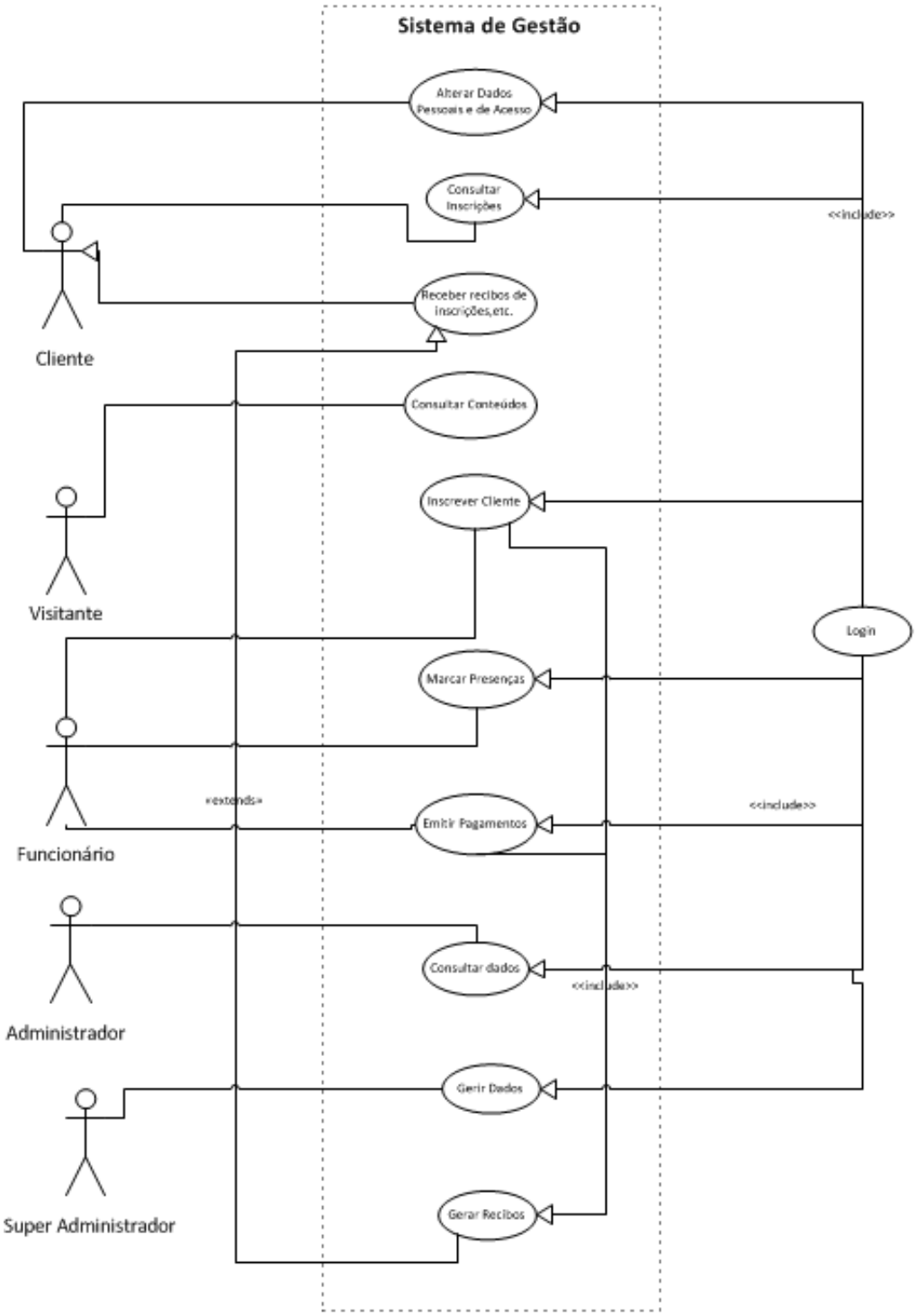


Figura 3.2: Diagrama Casos de Uso

3.2 Objetivos previstos

Os objetivos principais a atingir de um modo geral são:

- Construir um Front Office que permita à instituição comunicar com os seus colaboradores através de notícias, avisos, vídeos entre outros.
- Permitir inscrever novos clientes no sistema e gerir os acessos ao mesmo com privilégios distintos.
- Gerir pagamentos das modalidades e atividades.
- Gerir presenças dos clientes em eventos ou modalidades.
- Consultar dados sobre os clientes, como por exemplo obter a lista de inscritos e quais os clientes com pagamentos em atraso.
- Consultar dados sobre as modalidades, de modo a saber quanto é que foi faturado num determinado mês e determinado ano.
- Organizar as inscrições de modalidades e atividades.
- Organizar as subidas de níveis para determinadas modalidades para se saber quando e qual nível que o cliente atingiu.
- Facilitar os pagamentos e devolver um comprovativo ao cliente.
- Melhorar a organização dos pagamentos de modo a que os clientes não tenham mensalidades em atraso.
- Gerir as renovações a cada nova época.

Capítulo 4

Metodologia e resultados esperados

4.1 Metodologia

A metodologia no desenvolvimento da aplicação usada é o método ágil, ou seja, é um método mais virado para o código fonte e menos para a documentação, tornando todo o processo menos pesado e demoroso.
As principais tarefas foram:

1. Identificar as funcionalidades que a aplicação deverá conter no final;
2. Identificar os tipos de utilizadores da aplicação assim como os privilégios de acesso (quem pode aceder e onde);
3. Identificar os requisitos funcionais e não funcionais da aplicação;
4. Implementar a solução;
5. Realização de testes de eficiência e fiabilidade;
6. Documentação final do projeto;

4.2 Descrição das tarefas

As tarefas iniciais foram as mais complicadas uma vez que é o início de tudo. Com base na identificação dos requisitos funcionais e não funcionais é que o desenvolvimento da solução se desenrola. É com base nas informações recolhidas e imaginadas para a solução que a implementação é construida. O sucesso ou fracasso da aplicação terá muito como base a identificação e projeção correta das funcionalidades e modelo de base de dados.

Nesse sentido as tarefas basearam-se da seguinte forma:

- Tarefa 1 — Recolher as todas as funcionalidades que o sistema deverá no final possuir;
- Tarefa 2 — Desenhar e conceber o Modelo de Base de Dados (Modelo ER);

- Tarefa 3 — Construção do template para a aplicação;
- Tarefa 4 — Implementar a solução final;
- Tarefa 5 — Testes à aplicação;
- Tarefa 6 — Relatório final;

O respetivo Mapa de Gantt é apresentado na figura 4.1.

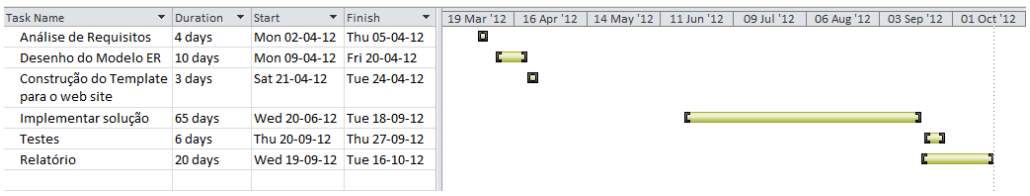


Figura 4.1: Mapa de Gantt

4.3 Resultados esperados

Pretende-se que no final, o projeto esteja implementado na sua totalidade e testada, mas acima de tudo que a implementação seja útil para aprofundar os conhecimentos e obter novos conhecimentos. Espera-se no entanto que a aplicação seja utilizada por uma entidade e que seja útil na gestão da instituição.

Espera-se que o responsável pela entidade ao utilizar a solução desenvolvida seja capaz de:

1. Consultar e obter listas de clientes com mensalidades em atraso.
2. Consultar e obter listas de clientes inscritos por modalidade/atividade.
3. Consultar e comparar dados sobre as modalidades e clientes.
4. Lançar novas modalidades/atividades.
5. Lançar as anuidades a cada nova época.
6. Lançar novas mensalidades a cada nova época.
7. Inserir novos conteúdos nas páginas de modo a interagir com o público e os clientes.
8. Registar novos clientes/utilizadores no sistema.
9. Registar e consultar subidas de níveis em modalidades.
10. Facilitar os pagamentos e emissão de recibos.
11. Efetuar renovações de inscrições a cada nova época.

12. Gerir os utilizadores do sistema, como por exemplo bloquear o acesso à aplicação, editar dados do utilizador entre outros.

Por outro lado a solução deverá ser capaz de:

1. Identificar utilizadores com pagamentos em atraso.
2. Identificar quais as modalidades em que o cliente se pode inscrever e o escalão do mesmo.
3. Não deixar renovar a inscrição do cliente se o mesmo tem pagamentos em atraso.
4. Não deixar inscrever um cliente numa modalidade/atividade se a anuidade ou data limite de inscrição já tenha sido ultrapassada.

Capítulo 5

Tecnologias utilizadas no desenvolvimento da Solução

Durante o desenvolvimento da solução foram usadas várias tecnologias de modo a obter o melhor resultado final possível. Neste capítulo foram apresentadas algumas das tecnologias mais importantes e com maior impacto na solução final. Também neste capítulo serão apresentados breves passos para a instalação de algumas dessas tecnologias.

5.1 Serviço Web

Um serviço web permite a comunicação entre aplicações diferentes, normalmente para partilha de dados. [12] Com esta tecnologia é possível que uma aplicação A numa linguagem X possa aceder aos dados da aplicação B numa linguagem Y. Isto não seria possível sem a linguagem XML que facilita todo este processo através da sua simplicidade.

5.2 AJAX Control Toolkit

Antes de falar no toolkit usado no projeto, vamos fazer uma pequena referência ao AJAX.

AJAX é o acrónimo de **A**synchronous **J**avascript **A**nd **X**ML e podemos definir como um recurso que utiliza algumas tecnologias existentes com o objetivo de promover interatividade e dinamismo para aplicações web. AJAX não é uma tecnologia, mas um conjunto de tecnologias conhecidas trabalhando juntas, cada uma fazendo a sua parte, tornando as aplicações finais mais ricas.

A citação seguinte resume perfeitamente e em poucas palavras esta tecnologia.
'AJAX is not a new programming language, but a new way to use existing standards.' [16]

O toolkit vem acrescentar precisamente novos controlos neste caso ao Visual Studio que foi o IDE usado na implementação da solução. Os controlos são extensões

aos controlos de raiz do ASP.NET. Quer isto dizer que os controlos usados são os controlos por defeito, mas com a particularidade de usar mais qualquer coisa (extensões AJAX). Entre os mais de 40 novos controlos que a toolkit tem, apenas dois foram usados na solução, tentou-se usar um terceiro elemento mas devido a alguns problemas resolveu-se retirar esse mesmo controlo. Os controlos usados foi o **AutoComplete Extender** e o **Calendar Extender**. O MaskEdit Extender foi o terceiro controlo usado mas sem sucesso, isto porque o conteúdo das caixas de texto desaparecia quando ocorria um postback na página, decidiu-se retirar a extensão das caixas de texto e resolver o problema com expressões regulares.

Passemos então a detalhar os controlos usados:

AutoComplete Extender — como o nome indica é uma extensão para ASP.NET que adiciona uma funcionalidade às caixas de texto da plataforma. Funcionalidade essa que adiciona um painel pop-up à caixa de texto com sugestões de palavras que começam com o prefixo digitado na caixa de texto. Exemplo na figura 5.1.

Calendar Extender — esta extensão fornece do lado do cliente, um painel pop-up com um calendário onde o cliente pode escolher uma data do calendário. Exemplo na figura 5.2.

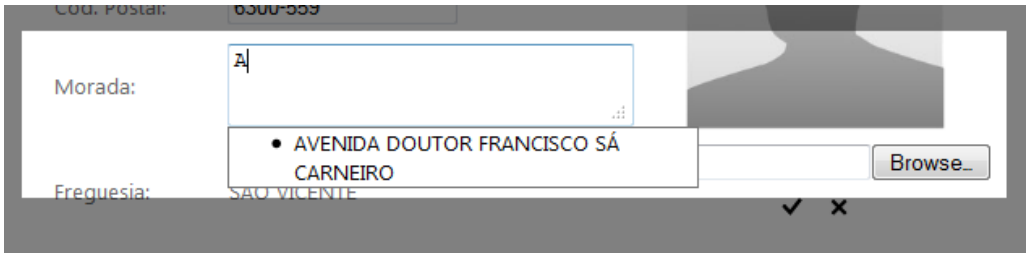


Figura 5.1: AJAX AutoComplete Extender

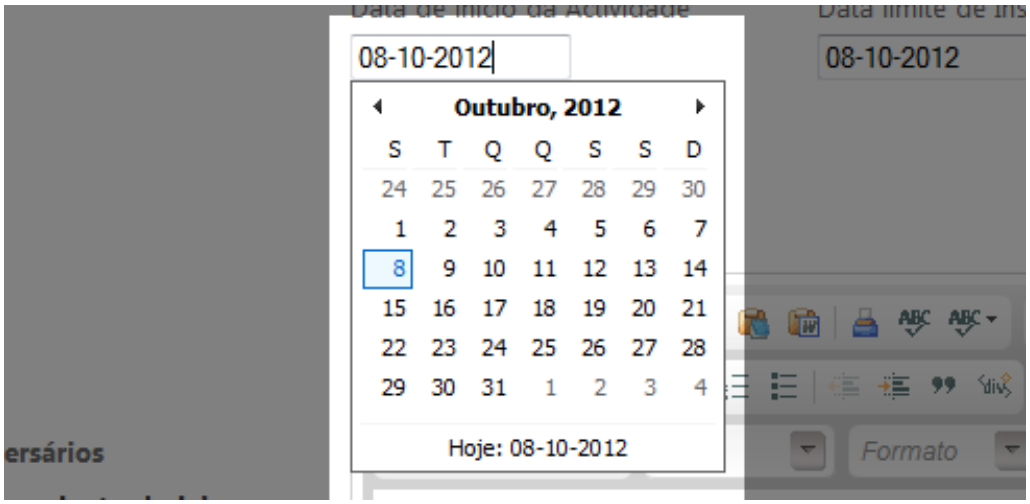


Figura 5.2: AJAX Calendar Extender

Apresentados os controlos usados, vamos apresentar de seguida como adicionar o toolkit ao IDE Visual Studio e algumas configurações que têm de ser feitas antes de se puder usar as extensões. Após uma breve descrição de como adicionar o toolkit e as configurações iniciais iremos aqui apresentar o que foi feito na solução desenvolvida com estes controlos.

O web site [8] contém as instruções gráficas como adicionar os controlos ao IDE Visual Studio, tutoriais para cada um dos controlos da toolkit e descrição das propriedades dos controlos.
Nesta página [3] podemos fazer o download dos ficheiros que compõem o toolkit, alguns exemlos de uso e o código fonte.

Para que possamos usar os controlos AJAX a página web terá de ter um ScriptManager. O ScriptManager está divido em duas fases. A primeira fase verifica se a aplicação suporta os conteúdos ASP.NET AJAX e configura o ambiente para que possamos usar os mesmos conteúdos. Na segunda fase, o ScriptManager constrói uma comunicação assíncrona com o código em execução no cliente para que o script possa executar as atualizações necessárias na página. Como é um controlo de servidor web e de programação no ASP:NET, o controlo é orientado a eventos. Podemos resumir a função do ScriptManager, como este assumindo o controlo do ciclo de renderização da página e que envia para o cliente a resposta que é facilmente interpretada e usada para atualizar os elementos no browser. [18]

No caso da solução desenvolvida, o ScriptManager (listagem 5.1) foi inserido na Master Page no Form da página, de modo a que o mesmo tivesse disponível nas outras páginas que descendem da Master Page ou Página Mestre.

Listagem 5.1: Configuração do ScriptManager da Solução

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
EnableScriptGlobalization="True">
<Services>
<asp:ServiceReference Path="AutoCompleteList.asmx" />
</Services>
</asp:ScriptManager>
```

Para os leitores mais atentos, devem estar a reparar que o ScriptManager tem algo mais na sua configuração, que é o *Services*. Pois bem é aqui que é definido o serviço web que irá permitir apresentar as sugestões nas caixas de texto com extensão AJAX. O Serviço web irá à nossa base de dados buscar a informação. No capítulo da Implementação da solução será detalhado em pormenor a implementação do AJAX AutoComplete Extender e Calendar Extender.

5.3 API

Application Programming Interface, é um conjunto de métodos e funções que são disponibilizados por alguns serviços como mapas por exemplo, que permitem aos programadores não se preocuparem com a implementação do serviço mas sim em

usar esses serviços. O uso desta tecnologia é importantes uma vez que com a mesma são feitas novas aplicações que usam funcionalidades que estão menos evidentes ao utilizador comum, surgindo aplicações totalmente novas e muito interessantes.

5.4 API Sapo Mapas

'É uma biblioteca que permite incluir os mapas do Sapo usando Javascript. Esta biblioteca disponibiliza um conjunto de utilidades e serviços, que manipulam os mapas do Sapo estendendo as suas funcionalidades base.' [13]

É assim que a página do Sapo Mapas descreve a sua API e que resume muito bem o serviço disponibilizado. É na página da API do Sapo Mapas [13] que se encontram alguns exemplos de uso e a documentação necessária para se conhecer a API. Foi de extrema importância para o projeto a consulta dos exemplos e da documentação.

Listagem 5.2: Exemplo: Adicionar um Mapa ao web site

```
<html>
	<head>
		<title>SAPO.Maps.Map on your web page!</title>
		<script type="text/javascript"
		    src="http://js.sapo.pt/Bundles/SAPOMapsAPI.js"></script>
		<script type='text/javascript'>

			window.onload = init;

			function init(){
				var map = new SAPO.Maps.Map('mapDiv');
			}
		</script>
	</head>
	<body>
		<div id='mapDiv' style='width:980px; height:400px;'></div>
	</body>
</html>
```

O Código listado em 5.2 produziria o output seguinte (figura 5.3):

Figura 5.3: Exemplo de um Mapa da Sapo

5.5 iText

O iText é uma biblioteca que permite criar e manipular documentos PDF. O iText está disponível para linguagem JAVA e para C#. [7] Os programadores podem usar muitas funcionalidades, mas a que vamos usar na solução é gerar documentos PDF. Na solução foi usado o iTextSharp, que é o iText específico para linguagem C#.

Os passos para começar a usar esta biblioteca e alguns exemplos podem ser consultados em três websites que serviram de apoio, na implementação desta função. É o web site c-sharpcorner.com [15], um artigo na página 4 Guys from Rolla.com [10] e por fim a página oficial da biblioteca [7].

5.6 CKEditor

O CKEditor é um editor de texto que foi criado com o intuito de ser incorporado em aplicações web. [1] Foi usado no projeto em páginas como a criação de conteúdos para facilitar a criação dos mesmos. A instalação e a documentação estão disponíveis na página do CKEditor [1]. Com este controlo o utilizador pode mudar a cor do texto, estruturar o texto, inserir vídeos nas páginas, entre muitas outras funcionalidades. É um controlo muito fácil de usar pois tem muitas propriedades iguais aos controlo nativos da plataforma .NET.(Figura 5.4)
Para se colocar em funcionamento basta no topo das páginas onde queremos usar o controlo colocar a seguinte linha 5.3:

Listagem 5.3: Linha para que o controlo seja reconhecido pela plataforma

```
<%@ Register assembly="CKEditor.NET" namespace="CKEditor.NET"
    tagprefix="CKEditor" %>
```

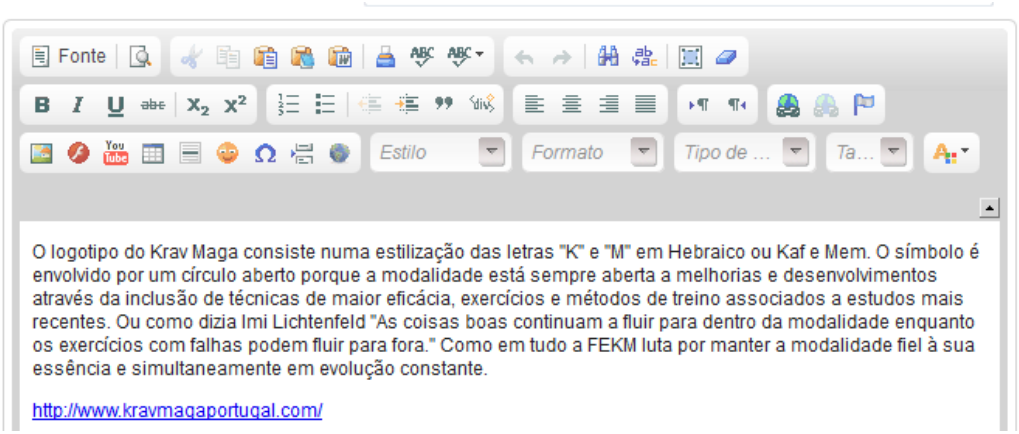


Figura 5.4: Controlo CKEditor

Capítulo 6

Implementação da solução

6.1 Introdução

A implementação da aplicação passou por diversas fases e por diversas etapas tanto no Back Office como no Front Office. Todas as páginas foram desenvolvidas na plataforma ASP.NET no IDE Visual Studio da Microsoft. Mas antes de começar a desenvolver as páginas foi necessário desenvolver um template em HTML + CSS com auxílio à ferramenta Dreamweaver da Adobe que permite o desenho de templates.
Na implementação da solução foram ainda desenvolvidos procedimentos em linguagem T-SQL recorrendo à ferramenta SQL Server Management Studio. A base de dados foi desenvolvida em SQL para o Sistema de Gestão de Base de Dados SqlServer. O desenho da base de dados foi feita numa ferramenta do Visual Studio que permite o desenho do modelo e posteriormente obter o script de criação das tabelas ou se quisermos ainda exporta logo o modelo para uma base de dados. No caso, optou-se pela exportação do modelo para script e posteriormente foi executado para uma base de dados. Falta referir o nome da ferramenta que é ADO.NET Entity Data Model.
Nos anexos deste documento poderão ser encontrados algumas classes e dois ou três exemplos de procedimentos desenvolvidos pois são muitos mais.

6.2 Construção do Template

A primeira parte da implementação como já foi referido, foi a construção do template para as páginas web. O template foi construido com recurso a HTML para a estrutura e com CSS para a parte visual da estrutura. O template construído pode ser observado na figura 6.1.

De referir que para que o template funcione na plataforma ASP.NET é necessário na tag <body> do html adicionar um form que vai permitir correr os controlos na página. (Ver Listagem 6.1).

Listagem 6.1: Adicionar Form

Figura 6.1: Template: Página Principal

```
<html>
        <head>
        //...
        </head>
        <body>
        <form id='Form1' runat='server'>
        //Controlos, Tabelas, etc...
        </form>
        <body>
</html>
```

O código HTML da estrutura e o ficheiro CSS, podem ser consultados nos anexos deste documento.

Problemas nesta fase:
Neste primeiro passo da solução não houve grande dificuldade na implementação.

6.3 Base de Dados

Por definição a plataforma recorre a uma base de dados SQL Server para guardar as contas dos utilizadores, privilégios, grupos entre outros. Essa base de dados tem um modelo ER já definido (figura 6.2) e a nossa base de dados teria outro modelo e estaria à parte. No entanto e por uma questão de facilitismo no uso das mesmas tabelas optou-se por juntar estas tabelas à nossa base de dados e obteve-se o modelo da figura 6.3. O processo foi feito seguindo um artigo que pode ser consultado na página pplware [6] e foi consultado ainda o livro ASP.NET 4.0 [4] para mais informações sobre o Membership. Mais à frente neste capítulo serão explicadas as tabelas que compõem o modelo mais em detalhe, no entanto só serão detalhadas as tabelas que foram usadas na implementação e as criadas para o efeito. Não serão detalhadas as tabelas relativas ao ASP.NET.
De referir ainda que apesar de na base de dados só serem aceites alguns tipos de dados ou tamanhos, na aplicação visual são feitas sempre validações de dados antes de inserir na Base de Dados.

Problemas nesta fase:
Este foi um dos pontos críticos do projeto. Como é a base da estrutura da aplicação, foram sentidas algumas dificuldades no desenho do modelo ER. Foi numa das tarefas onde foi dispensado algum tempo e durante a implementação mesmo assim foram surgindo pequenas alterações e detetados alguns problemas. Uma vez que com o desenrolar da implementação foram surgindo novas ideias e constatados novos factos.

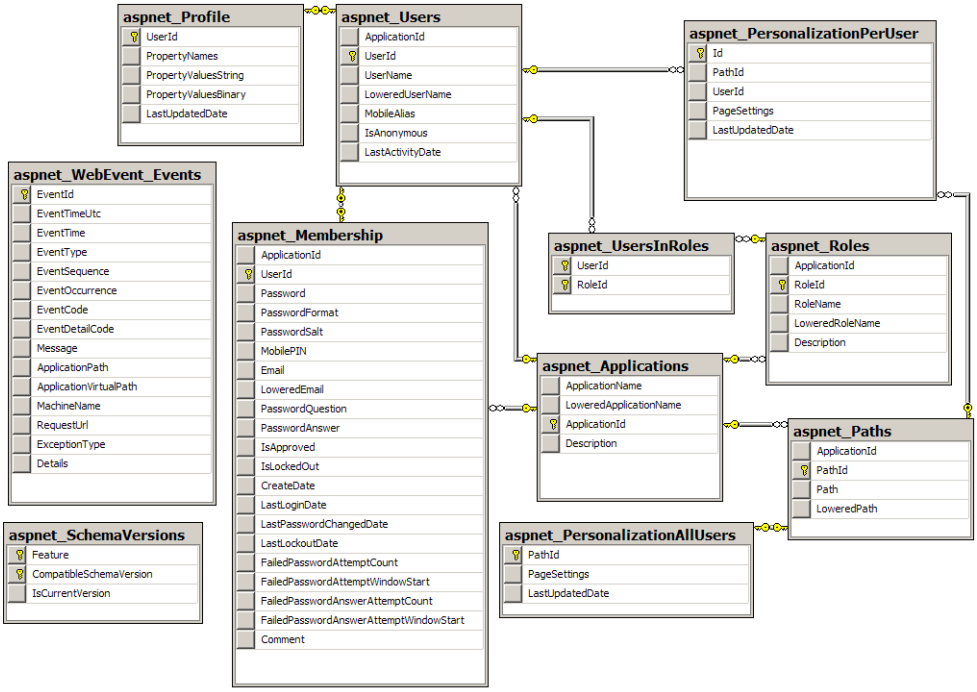


Figura 6.2: Modelo de base de dados que suporta os serviços fornecidos pela plataforma

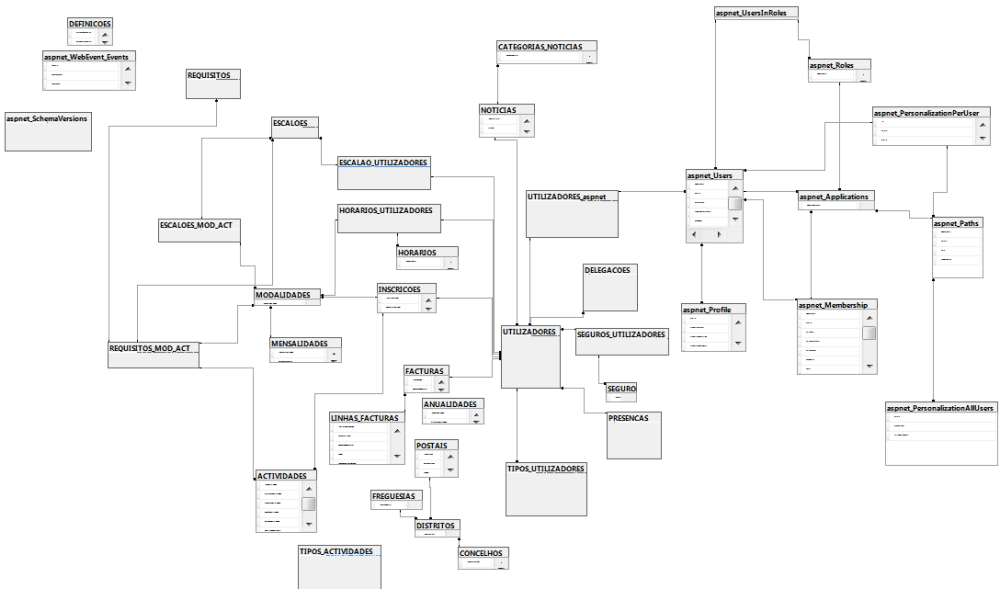


Figura 6.3: Modelo ER da solução

6.4 Dicionário de Dados

O dicionário de dados descreve as tabelas que compõem a base de dados, quanto aos seus campos e tipos de dados.
Quanto ao campo teremos:

- CP - Chave Primária
- CE - Chave Estrangeira

Quanto aos campos nulos teremos:

- N - Não
- S - Sim

Quanto ao tipo teremos:

- int - Inteiro
- nvarchar - Texto e Números
- datetime - Data e Tempo
- time - Tempo
- bit - Booleano
- float - Decimal
- uniqueidentifier - Identificador único (Cadeia de caracteres)

6.4.1 Atividades

Na tabela seguinte apresentam-se os campos relativos às atividades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_ACTIVIDADE (CP)	N	int	4	Número sequencial que representa univocamente cada atividade
NOME_ACTIVIDADE	N	nvarchar	100	Nome da atividade
PRECO_ACTIVIDADE	N	float	8	Preço da atividade
DATA_ACTIVIDADE	N	datetime	8	Data na qual a atividade terá início
HORA_ACTIVIDADE	S	time	5	Hora na qual a atividade terá início
ACTIVIDADE_VIGOR	N	bit	1	Campo que permite saber se a atividade ainda está em vigor
DESC_ACTIVIDADE	N	nvarchar	4000	Pequena descrição da atividade
TIPOS_ACTIVIDADE (CE)	N	int	4	Chave estrangeira que identifica o tipo de atividade
DISTRITOS_ID_DISTRITO (CE)	N	nvarchar	6	Chave estrangeira que identifica o distrito onde terá lugar a atividade
CONCELHOS_ID_CONCELHO (CE)	N	nvarchar	6	Chave estrangeira que identifica o concelho onde terá lugar a atividade
FREGUESIAS_ID_FREGUESIA (CE)	N	nvarchar	6	Chave estrangeira que identifica a freguesia onde terá lugar a atividade
LATITUDE	S	nvarchar	50	Latitude na qual a atividade terá lugar
LONGITUDE	S	nvarchar	50	Longitude na qual a atividade terá lugar
MORADA	S	nvarchar	100	Morada na qual a atividade terá lugar
DATA_LIMITE	N	datetime	8	Data limite de inscrição na atividade
MODALIDADES_ID_MODALIDADE (CE)	S	int	4	Modalidade associada à atividade

Tabela 6.1: Estrutura da tabela atividades

6.4.2 Anualidades

Na tabela seguinte apresentam-se os campos relativos às anualidades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_ANUALIDADE (CP)	N	int	4	Número sequencial que representa univocamente cada anuidade
NOME_ANUIDADE	N	nvarchar	100	Nome da anuidade
DATA_INICIO	N	datetime	8	Data na qual a anuidade terá início
DATA_FIM	N	datetime	8	Data na qual a anuidade terá fim
PRECO_ANUAL	N	float	8	Preço da anuidade
ANUALIDADE_VIGOR	N	bit	1	Campo que permite saber se a anuidade está em vigor

Tabela 6.2: Estrutura da tabela Anualidades

Os campos da data de inicio e fim servem para o sistema saber entre que datas são permitidas novas inscrições.

6.4.3 Categorias de Notícias

Na tabela seguinte apresentam-se os campos relativos às categorias de notícias. Esta tabela permite categorizar os conteúdos que podem ser inseridos na página.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_CATEGORIA (CP)	N	int	4	Número sequencial que representa univocamente cada categoria
PAGINA_PRINCIPAL	N	int	4	Número que identifica uma zona da página principal da aplicação
NOME_CATEGORIA	N	nvarchar	100	Nome da Categoria

Tabela 6.3: Estrutura da tabela Categorias de Notícias

O campo PAAGINA_PRINCIPAL identifica a zona da página principal a que a categoria diz respeito, uma vez que a página principal tem 4 zonas onde os conteúdos poderão ser colocados.

6.4.4 Concelhos

Na tabela seguinte apresentam-se os campos relativos aos Concelhos. Esta tabela contém informações sobre todos os concelhos nacionais.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_CONCELHO (CP)	N	nvarchar	6	Cadeia de caracteres que representa univocamente cada concelho
NOME_CONCELHO	N	nvarchar	100	Nome do Concelho
DISTRITOS_ID_DISTRITO (CP)	N	nvarchar	6	Cadeia de caracteres que representa o distrito associado ao concelho

Tabela 6.4: Estrutura da tabela Concelhos

6.4.5 Definiçoes

Na tabela seguinte apresentam-se os campos relativos às Definições. Esta tabela contém informações que são usadas na página como o nome da entidade, contactos, logotipo entre outros.

Campo	Nulo	Tipo	Tamanho	Descrição
NOME_WEBSITE	N	nvarchar	100	Nome do web site que será apresentado na página e nos recibos
COD_POSTAL_INST	N	nvarchar	8	Código Postal da entidade
DISTRITO_INST	N	nvarchar	100	Distrito da entidade
MORADA	N	nvarchar	100	Morada da entidade
CONTACTO	N	nvarchar	9	Contacto da entidade
SOBRE	N	nvarchar	MAX	Descrição da entidade
LOGOTIPO	N	nvarchar	100	Nome do logotipo da entidade

Tabela 6.5: Estrutura da tabela Definições

6.4.6 Delegações

Na tabela seguinte apresentam-se os campos relativos às Delegações.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_DELEGACAO (CP)	N	int	4	Número que identifica univocamente cada delegação
NOME_DELEGACAO	N	nvarchar	100	Nome da Delegação

Tabela 6.6: Estrutura da tabela Delegações

6.4.7 Distritos

Na tabela seguinte apresentam-se os campos relativos aos Distritos.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_DISTRITO (CP)	N	nvarchar	6	Cadeia de caracteres que identifica univocamente cada distrito
NOME_DISTRITO	N	nvarchar	100	Nome do Distrito

Tabela 6.7: Estrutura da tabela Distritos

6.4.8 Escalão de Utilizadores

Na tabela seguinte apresentam-se os campos relativos aos Escalões de Utilizadores. Com base nesta tabela o sistema sabe em que nível o praticante se encontra e as datas de subidas ou descidas de níveis.

Campo	Nulo	Tipo	Tamanho	Descrição
DATA_ESCALAO	N	datetime	8	Data em que ocorreu o evento
ESCALAO_VIGOR	N	bit	1	Campo que permite saber qual dos escalões do praticante está em vigor
UTILIZADORES_ID_UTILIZADOR (CE)	N	int	4	Identifica o utilizador
ESCALAO_ID_ESCALAO (CE)	N	int	4	Identifica o escalão

Tabela 6.8: Estrutura da tabela Escalões de Utilizadores

6.4.9 Escalões

Na tabela seguinte apresentam-se os campos relativos aos Escalões.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_ESCALAO (CP)	N	int	4	Número que identifica univocamente cada escalão
PRIMARIO	S	bit	1	Identifica o escalão como sendo o primeiro nível ou não
CINTURAO	S	bit	1	Identifica o escalão como sendo um escalão de cinturão ou não
HIERARQUIA	N	int	4	Identifica o nível anterior a este
NOME_ESCALAO	N	nvarchar	100	Nome do nível

Tabela 6.9: Estrutura da tabela Escalões de Utilizadores

O campo PRIMARIO é um campo importante pois permite ao sistema ir buscar o primeiro nível de uma determinada modalidade de modo a gerir os níveis de uma forma mais eficiente. O campo CINTURAO permite organizar os níveis de modo a saber se é um nível organizado ou seja para se ir para um determinado nível tem que se obter o último nível associado ou se é um nível desorganizado que permite

aos praticantes saltar de nível em nível.
Tomemos o seguinte exemplo, +50kg, +80kg. São dois níveis diferentes mas que não estão ligados entre si. O cliente quando é inscrito numa modalidade que tem estes escalões, irá ser inscrito no escalão do seu peso atual. Imaginemos que inicialmente o cliente tinha 78kg, quer isto dizer que se encontrava no escalão +50kg, passado algum tempo o cliente apresentou-se com 85kg, ou seja o cliente subi de nível para o escalão +80kg sem que tenha uma condição obrigatória do escalão anterior e até poderá saltar vários níveis de uma só vez. O que não é, na maioria dos casos possível em tipos de escalões que se regem por cinturão.

6.4.10 Escalões de Modalidades/Actividades

Na tabela seguinte apresentam-se os campos relativos aos Escalões de Modalidades/Actividades. Esta tabela contém os escalões associados às modalidades.

Campo	Nulo	Tipo	Tamanho	Descrição
ESCALOES_ID_ESCALAO (CE)	N	int	4	Número que identifica o escalão
MODALIDADES_ID_MODALIDADE (CE)	N	int	4	Número que identifica a modalidade

Tabela 6.10: Estrutura da tabela Escalões de Modalidades/Actividades

6.4.11 Faturas

Na tabela seguinte apresentam-se os campos relativos às Faturas.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_FACTURA (CP)	N	int	4	Número que identifica univocamente uma fatura
DATA_PAGAMENTO	N	datetime	8	Data na qual a fatura foi gerada
TOTAL_PAGAR	N	float	8	Total da Fatura
NUM_FATURA	N	nvarchar	30	Número da Fatura
UTILIZADORES_ID_UTILIZADORES (CE)	N	int	4	Número que identifica o utilizador

Tabela 6.11: Estrutura da tabela Faturas

6.4.12 Freguesias

Na tabela seguinte apresentam-se os campos relativos às Freguesias. Esta tabela contém as todas as Freguesias nacionais.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_FREGUESIA (CP)	N	nvarchar	6	Cadeia de caracteres que identifica univocamente cada freguesia
NOME_FREGUESIA	N	nvarchar	100	Nome da Freguesia
CONCELHOS_ID_CONCELHO (CP)	N	nvarchar	6	Identifica o concelho ao qual a freguesia pertence
DISTRITOS_ID_DISTRITO (CP)	N	nvarchar	6	Identifica o distrito ao qual a freguesia pertence

Tabela 6.12: Estrutura da tabela Freguesias

6.4.13 Inscrições

Na tabela seguinte apresentam-se os campos relativos às Inscrições. Nesta tabela são guardadas todas as inscrições em atividades/modalidades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_INSCRICAO (CP)	N	int	4	Número que identifica univocamente cada inscrição
DATA_INSCRICAO	N	datetime	8	Data em que ocorreu a inscrição.
INSCRICAO_ACTIVA	N	bit	1	Sinaliza se a inscrição está ativo
MODALIDADES_ID_MODALIDADE (CE)	S	int	4	Identifica a modalidade
ACTIVIDADES_ID_ACTIVIDADE (CE)	S	int	4	Identifica a actividade
UTILIZADORES_ID_UTILIZADOR (CE)	N	int	4	Identifica o utilizador

Tabela 6.13: Estrutura da tabela Inscrições

Uma inscrição só poderá ter associada uma modalidade ou atividade, daí os campos poderem ser nulos.

6.4.14 Linhas de Faturas

Na tabela seguinte apresentam-se os campos relativos às Linhas de Faturas.

As linhas de faturas são geradas quando se fazem inscrições no sistema. Vamos aqui dar alguns exemplos. Imaginemos que inscrevemos um cliente pela primeira vez na modalidade X, e o mês atual é o mês de Janeiro e a anuidade termina a 30 de abril. Quando esse cliente é inscrito, é gerada uma fatura que contém várias linhas de faturas entre elas a linha referente ao mês de janeiro e marcada como paga e as restantes linhas referentes aos meses de fevereiro, março e abril como não pagas. Será ainda gerada uma linha para o seguro e anuidade, uma vez que é a primeira inscrição são marcadas como pagas. Para os utilizadores mais atentos é por essa razão que existem 3 campos na tabela 6.14 que podem conter nulos precisamente porque a tabela é usada em diversas ocasiões distintas e em alguns casos a linha de fatura tem anuidade associada noutros casos já não tem. O mesmo para os restantes campos na mesma situação.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_LINHA_FACTURA (CP)	N	int	4	Número que identifica univocamente cada Linha de Fatura
PRECO_LINHA	N	float	8	Preço da linha
DATA_PAGAMENTO	N	datetime	8	Data que ocorreu o pagamento
PAGO	N	bit	1	Sinaliza se a linha foi paga ou não
FACTURAS_ID_FACTURA (CE)	N	int	4	Identifica a fatura associada à linha de fatura
ANUALIDADES_ID_ANUALIDADE (CE)	S	int	4	Identifica a anualidade
ACTIVIDADES_ID_ACTIVIDADE (CE)	S	int	4	Identifica a atividade
SEGUROS_ID_SEGURO (CE)	S	int	4	Identifica o seguro

Tabela 6.14: Estrutura da tabela Linha de Faturas

6.4.15 Mensalidades

Na tabela seguinte apresentam-se os campos relativos às Mensalidades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_MENSALIDADE (CP)	N	int	4	Número que identifica univocamente cada mensalidade
MES_REFERENTE	N	int	4	Número do mês da mensalidade
ANO_REFERENTE	N	int	4	Número do ano referente à mensalidade
PRECO_MENSALIDADE	N	float	8	Preço da mensalidade
ID_MODALIDADE (CE)	N	int	4	Modalidade associada à mensalidade
MENSALIDADE_VIGOR	N	bit	1	Sinaliza se a mensalidade se encontra em vigor

Tabela 6.15: Estrutura da tabela Mensalidades

6.4.16 Modalidades

Na tabela seguinte apresentam-se os campos relativos às Modalidades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_MODALIDADE (CP)	N	int	4	Número que identifica univocamente cada modalidade
NOME_MODALIDADE	N	nvarchar	100	Nome da modalidade
MODALIDADE_VIGOR	N	bit	1	Sinaliza se a modalidade se encontra em vigor

Tabela 6.16: Estrutura da tabela Modalidades

6.4.17 Notícias

Na tabela seguinte apresentam-se os campos relativos às Notícias.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_NOTICIA (CP)	N	int	4	Número que identifica univocamente cada notícia
TITULO	N	nvarchar	2000	Titulo respetivo à notícia
TEXTO	N	nvarchar	MAX	Texto da notícia
DATA_NOTICIA	N	datetime	8	Data na qual a notícia é publicada
IMAGEM_NOTICIA	N	nvarchar	1000	Imagem que associada à noticia
UTILIZADORES_ID_UTILIZADOR	N	int	4	Identifica o utilizador que publica a notícia
PREVIEW	N	nvarchar	1000	Texto resumido que poderá ser encontrado na notícia
CATEGORIA_NOTICIAS_ID_CATEGORIA	N	int	4	Identifica a categoria respetiva da notícia

Tabela 6.17: Estrutura da tabela Notícias

6.4.18 Postais

Na tabela seguinte apresentam-se os campos relativos aos Códigos Postais. Esta tabela contém todos os códigos postais nacionais.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_POSTAL(CP)	N	int	4	Número que identifica univocamente cada notícia
COD_POSTAL	N	nvarchar	8	Código Postal
LUGAR	N	nvarchar	200	Lugar respetivo ao código postal
FREGUESIAS_ID_CATEGORIA (CP)	N	int	4	Identifica a freguesia respetiva ao código postal
CONCELHOS_ID_CONCELHO (CP)	N	int	4	Identifica o concelho respetivo ao código postal
DISTRITOS_ID_DISTRITO (CP)	N	int	4	Identifica o distrito respetivo ao código postal

Tabela 6.18: Estrutura da tabela Códigos Postais

6.4.19 Presenças

Na tabela seguinte apresentam-se os campos relativos às Presenças.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_PRESENCA (CP)	N	int	4	Número que identifica univocamente cada presença
ID_UTILIZADOR (CE)	N	int	4	Número que identifica o utilizador
ID_ACTIVIDADE (CE)	S	int	4	Número que identifica a atividade
ID_MODALIDADE (CE)	S	int	4	Número que identifica a modalidade
DATA_PRESENCA	N	datetime	8	Data da presença
MES_REFERENTE	N	int	4	Mês respetivo à presença
ANO_REFERENTE	N	int	4	Ano respetivo à presença

Tabela 6.19: Estrutura da tabela Presenças

Mais uma vez as presenças só podem ser de modalidades ou atividades, podendo assim esses campos serem nulos.

6.4.20 Requisitos

Na tabela seguinte apresentam-se os campos relativos aos Requisitos.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_REQUISITO (CP)	N	int	4	Número que identifica univocamente cada requisito
NOME_REQUISITO (CE)	N	nvarchar	100	Nome do requisito

Tabela 6.20: Estrutura da tabela Requisitos

6.4.21 Requisitos de Modalidades/Actividades

Na tabela seguinte apresentam-se os campos relativos aos Requisitos de Modalidades/Actividades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_REQUISITO_MOD_ACT (CP)	N	int	4	Número que identifica univocamente cada requisito de modalidade/atividade
REQUISITOS_ID_REQUISITO (CE)	N	int	4	Número que identifica o requisito
MODALIDADES_ID_MODALIDADE (CE)	S	int	4	Modalidade do requisito
ID_ACTIVIDADE (CE)	S	int	4	Atividade do requisito
ESCALOES_ID_ESCALAO	S	int	4	Escalão permitido
REQUISITO_MIN_IDADE	S	int	4	Escalão mínimo permitido

Tabela 6.21: Estrutura da tabela Requisitos de Modalidades/Actividades

Esta tabela permite ao sistema saber se uma determinada atividade tem um mínimo de idade para o praticante se puder inscrever, no caso das modalidades permite saber se a modalidade tem algum escalão ou idade mínimo para se inscrever na modalidade.

6.4.22 Seguros

Na tabela seguinte apresentam-se os campos relativos aos Seguros.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_SEGURO (CP)	N	int	4	Número que identifica univocamente cada seguro
NOME_SEGURO	N	nvarchar	100	Nome do seguro
PRECO_SEGURO	N	float	8	Preço do seguro
SEGURO_VIGOR	N	bit	1	Sinaliza se o seguro está em vigor ou não

Tabela 6.22: Estrutura da tabela Seguros

6.4.23 Tipos de Atividades

Na tabela seguinte apresentam-se os campos relativos aos Tipos de Atividades.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_TIPO_ACTIVIDADE (CP)	N	int	4	Número que identifica univocamente cada tipo de atividade
NOME_TIPO_ACTIVIDADE	N	nvarchar	100	Nome dado ao tipo de atividade

Tabela 6.23: Estrutura da tabela Tipos de Atividades

6.4.24 Tipos de Utilizador

Na tabela seguinte apresentam-se os campos relativos aos Tipos de Utilizadores.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_TIPO_UTILIZADORES (CP)	N	int	4	Número que identifica univocamente cada tipo de utilizador
TIPO_DEFEITO	N	bit	1	Sinaliza se é o tipo de utilizadores por omissão (é o grupo onde todos os novos clientes serão inseridos)
TIPO_UTILIZADOR	N	nvarchar	100	Nome dado ao tipo de utilizador

Tabela 6.24: Estrutura da tabela Tipos de Utilizador

6.4.25 Utilizadores ASP.NET

Na tabela seguinte apresentam-se os campos relativos à tabela da base de dados intermédia que liga as tabelas criadas para esta solução com as tabelas criadas pelo ASP.NET já referidas neste documento.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_UTILIZADOR	N	int	4	Número que identifica univocamente cada utilizador nas tabelas criadas
useraspnet_id	N	uniqueidentifier	16	Identifica o utilizador do lado das tabelas criadas pelo ASP.NET

Tabela 6.25: Estrutura da tabela Utilizadores_Asp.NET

6.4.26 Utilizadores

Na tabela seguinte apresentam-se os campos relativos à tabela Utilizadores.

Campo	Nulo	Tipo	Tamanho	Descrição
ID_UTILIZADOR	N	int	4	Número que identifica univocamente cada utilizador
NOME_UTILIZADOR	N	nvarchar	100	Primeiro nome do utilizador
SOBRENOME_UTILIZADOR	N	nvarchar	100	Apelido do utilizador
MORADA_UTILIZADOR	N	nvarchar	200	Morada do utilizador
BI_UTILIZADOR	N	nvarchar	12	Número do Bilhete de Identidade
NIF_UTILIZADOR	N	nvarchar	9	Número de Identificação Fiscal
TELEFONE_UTILIZADOR	N	nvarchar	9	Contacto do utilizador
EMAIL_UTILIZADOR	N	nvarchar	100	Endereço eletrónico do utilizador
ID_DELEGACAO (CE)	N	int	4	Identifica a Delegação
NUM_UTILIZADOR	N	nvarchar	30	Número do cliente
DATA_INSCRICAO	N	datetime	8	Data em que o cliente foi criado
INSCRICAO_VALIDA	N	datetime	8	Sinaliza se a inscrição do cliente é válida
FOTOGRAFIA	S	nvarchar	100	Fotografia do utilizador
TIPOS_UTILIZADORES_ID _TIPO_UTILIZADORES (CE)	N	int	4	Identifica do tipo de utilizador
POSTAL_UTILIZADOR (CE)	N	nvarchar	8	Código postal do utilizador
DATA_NASCIMENTO (CE)	N	datetime	8	Data de nascimento do utilizador

Tabela 6.26: Estrutura da tabela Utilizadores

6.5 Desenvolvimento de Software por 3 Tier Architecture ou Modelo em 3 Camadas

Logo no ínicio do projeto foi me proposto pelo professor orientador do projeto o desenvolvimento da aplicação com base nesta arquitetura. Uma vez que não conhecia esta arquitetura, fui pesquisando sobre o assunto. E as matérias que ia encontrando sobre o assunto, falavam todas sobre a mesma questão, que era um processo mais demorado no início mas era compensado depois com a facilidade no controlo de crescimento do sistema.

O que é o Modelo em 3 camadas ou 3 Tier Architecture?
Podemos definir o desenvolvimento em 3 camadas, daí o nome. As camadas são:

1. Camada de Apresentação ou UI (User Interface)
2. Camada de Negócio, também conhecida por BUS (Business Logic Layer)
3. Camada de Dados, também conhecida por DAL (Data Access Layer) ou DAO (Data Access Object)

A figura 6.4 mostra-nos gráficamente a arquitetura por camadas.

E quais as vantagens de usar o modelo?
Apresentamos de seguida duas vantagens fortes na utilização do modelo.

1. Transformações do sistema tornam-se mais fáceis, isto porque se a lógica do negócio está separada do acesso aos dados e se a forma de acesso aos dados é alterada, só teremos que readaptar a camada de acesso aos dados e a lógica

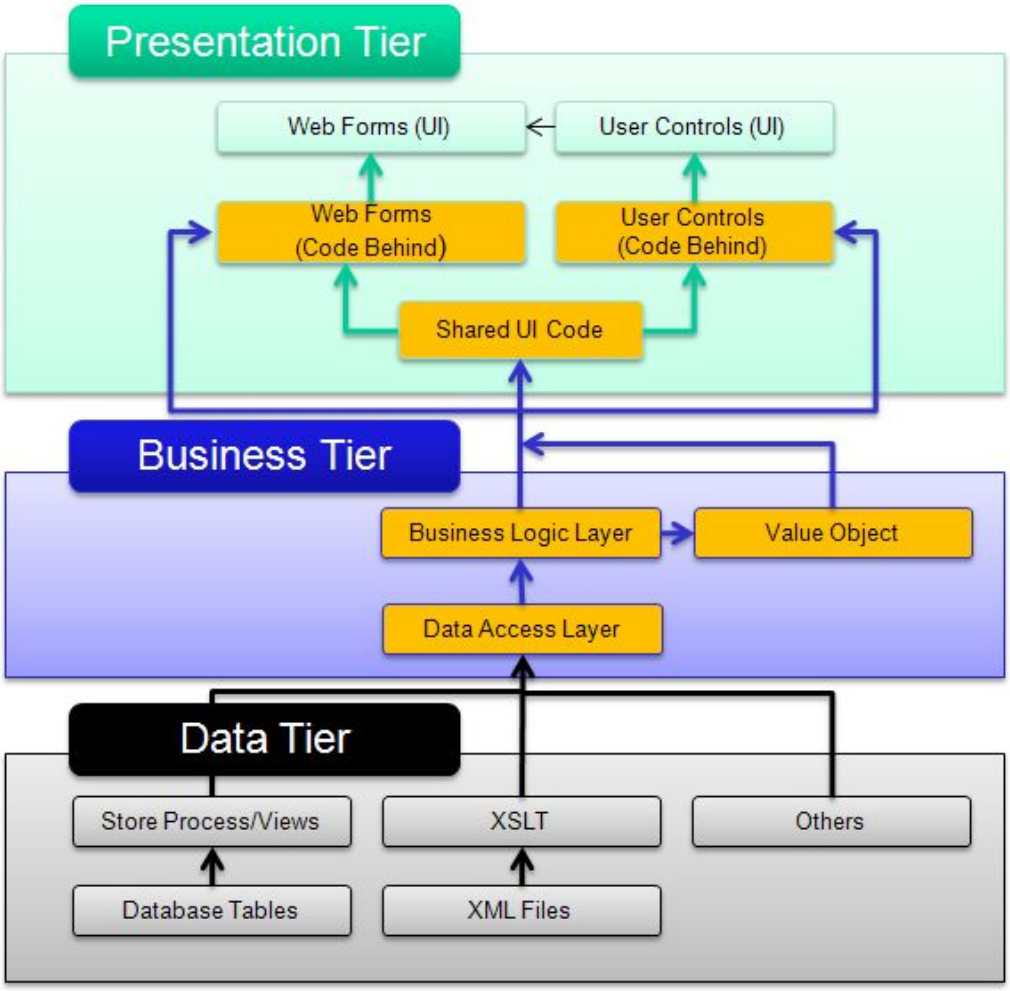


Figura 6.4: Modelo em 3 camadas

continua da mesma forma ou com pequenas mudanças tudo ficará a funcionar novamente.

2. Facilidade em reutilizar as camadas em outros projetos.

Vamos de seguida ver um exemplo concreto, retirado da solução desenvolvida. (Figura 6.5)

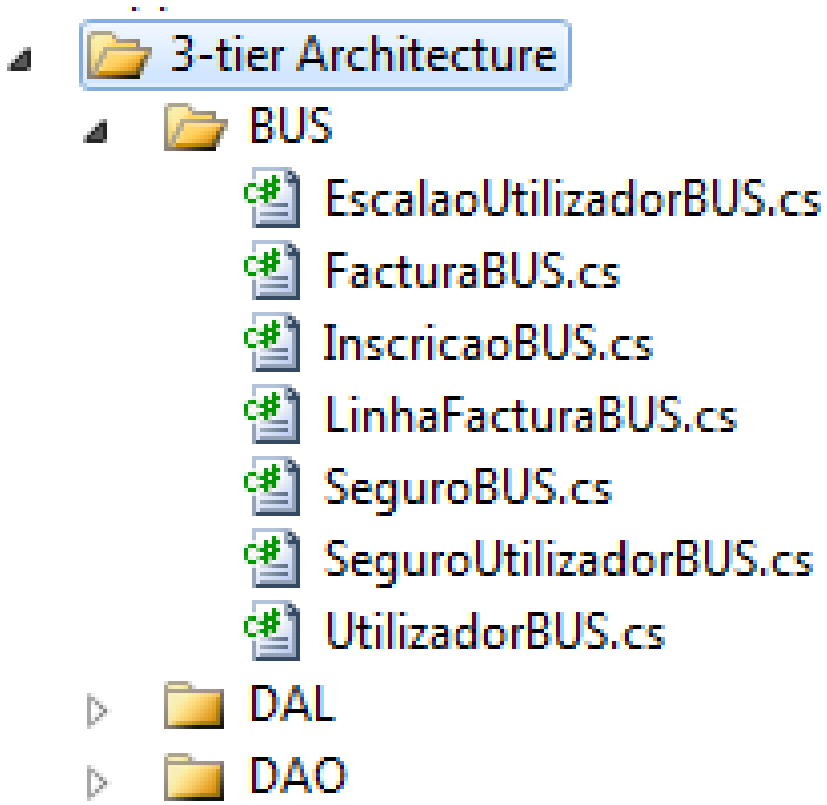


Figura 6.5: Estrutura do modelo em 3 camadas no Visual Studio

Comecemos então pela camada mais baixa, a camada de acesso aos dados (DAL). E a primeira classe que aqui será apresentada é a classe que permite aceder à base de dados desenvolvida e apresentada no ponto anterior. Esta classe tem métodos como abrir uma ligação à base de dados, fechar a ligação, iniciar transações, confirmar transações (commit), rollback de transações e execução de procedimentos. Esta é a classe principal da arquitetura pois sem esta não teriamos acesso aos dados. Na listagem 6.2 poderá visualizar a classe e alguns métodos.

Listagem 6.2: Classe Database do modelo em 3 camadas

```
public class Database
{

    private string connStr;
    private SqlConnection conn;
    private SqlTransaction trans;
```

```
    public Database()
    {
    try
    {
        foreach (ConnectionStringSettings item in
            ConfigurationManager.ConnectionStrings)
        {
            if (item.Name.Equals("WebsiteGim"))
            {

                connStr = item.ConnectionString;

            }

        }
        conn = new SqlConnection(connStr);
        conn.Open();
    }
    catch (Exception e)
    {
        conn.Dispose();
    }
    }

public SqlConnection abrirConn()
{
    if (conn.State == ConnectionState.Closed || conn.State ==
        ConnectionState.Broken)
    {
        try{
            conn.Open();
        }catch{
            conn.Dispose();
        }
    }
    return conn;
}

public SqlConnection fecharConn()
{
    if (conn.State == ConnectionState.Open)
    {
        try
        {
            conn.Close();
        }
        catch
        {
```

```
            conn.Dispose();
        }
    }
    return conn;
}

public void iniciaTrans()
{
    try
    {
        trans = conn.BeginTransaction(IsolationLevel.ReadCommitted);
    }
    catch (Exception exp)
    {
    }
}

public void rollbackTrans()
{
    try
    {
        trans.Rollback();
    }
    catch (Exception exp)
    {
    }
}

public void commitTrans()
{
    try
    {
        trans.Commit();
    }
    catch (Exception exp)
    {
    }
}

public int executeProc(String _nomeProc, SqlParameter[] sqlParameter)
{
    SqlCommand cmd = new SqlCommand();
    int valorRetorno;

    try
    {
        if (conn.State == ConnectionState.Closed || conn.State ==
            ConnectionState.Broken)
        {
```

```
                cmd.Connection = abrirConn();
            }

            if (conn.State == ConnectionState.Open)
            {
                cmd.CommandText = _nomeProc;
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddRange(sqlParameter);
                cmd.Connection = conn;
                cmd.Transaction = trans;
                cmd.ExecuteNonQuery();
                valorRetorno = (int)cmd.Parameters["@EXISTE"].Value;
                return valorRetorno;
            }
        }
        catch (SqlException e)
        {
            WebMsgBox.Show("Erro a executar (executeProc): " + _nomeProc +
                " \nException: \n" + e.StackTrace.ToString());
            conn.Dispose();
            return -1;
        }
        return -2;
    }
}
```

Apresentada a classe de acesso à base de dados, tomemos agora como exemplo o acesso em especial ao conteúdo do utilizador. Para isso desenvolveu-se uma classe que para cada campo da tabela utilizador da base de dados têm os métodos get e set. Exemplo na listagem 6.3, não será apresentada a classe completa uma vez que os métodos são iguais só muda o campo.

Listagem 6.3: Classe Utilizador do modelo em 3 camadas

```
public class Utilizador
{
    private Int32 id;
    private String nome;
    private String sobrenome;
    private String morada;


    public Utilizador()
        {
        this.id = -1;
        this.nome = "";
        this.sobrenome = "";
        this.morada = "";
```

```
        }

    public Utilizador(Int32 _id, String _nome, String _sobrenome, String
        _morada, String _bi, String _nif,
        String _telefone, String _email, String _numUtilizador, Int32
            _tipoUtilizador, Int32 _delegacao, String _postal,
        String _fotografia, DateTime _dataNascimento)
    {
        this.id = _id;
        this.nome = _nome;
        this.sobrenome = _sobrenome;
        this.morada = _morada;
    }


    public Int32 Id
    {
        get { return id; }
        set { id = value; }
    }

    public String Nome
    {
        get { return nome; }
        set { nome = value; }
    }

                //Continua
}
```

Ainda na camada DAL teremos a classe que contém então os métodos de acesso à base de dados. Na listagem 6.4 teremos o método de inserir um novo utilizador na base de dados.

Listagem 6.4: Classe DAO Utilizador do modelo em 3 camadas

```
public class UtilizadoresDAO
{
    private Database db;
    private Utilizador ut;

        public UtilizadoresDAO()
        {
        Database db = new Database();
        ut = new Utilizador();
        }

    public int inserirUtilizador(Utilizador _ut)
    {
```

```
try
{
    SqlParameter[] sqlParameters = new SqlParameter[14];

    sqlParameters[0] = new SqlParameter("@num", SqlDbType.NVarChar);
    sqlParameters[0].Value = Convert.ToString(_ut.NumUtilizador);
    sqlParameters[1] = new SqlParameter("@nome",
        SqlDbType.NVarChar);
    sqlParameters[1].Value = Convert.ToString(_ut.Nome);
    sqlParameters[2] = new SqlParameter("@sobrenome",
        SqlDbType.NVarChar);
    sqlParameters[2].Value = Convert.ToString(_ut.Sobrenome);
    sqlParameters[3] = new SqlParameter("@bi", SqlDbType.NChar);
    sqlParameters[3].Value = Convert.ToString(_ut.Bi);
    sqlParameters[4] = new SqlParameter("@nif", SqlDbType.NChar);
    sqlParameters[4].Value = Convert.ToString(_ut.Nif);
    sqlParameters[5] = new SqlParameter("@email",
        SqlDbType.NVarChar);
    sqlParameters[5].Value = Convert.ToString(_ut.Email);
    sqlParameters[6] = new SqlParameter("@morada",
        SqlDbType.NVarChar);
    sqlParameters[6].Value = Convert.ToString(_ut.Morada);
    sqlParameters[7] = new SqlParameter("@telefone",
        SqlDbType.NChar);
    sqlParameters[7].Value = Convert.ToString(_ut.Telefone);
    sqlParameters[8] = new SqlParameter("@foto",
        SqlDbType.NVarChar);
    sqlParameters[8].Value = Convert.ToString(_ut.Fotografia);
    sqlParameters[9] = new SqlParameter("@delegacao",
        SqlDbType.Int);
    sqlParameters[9].Value = Convert.ToInt32(_ut.Delegacao);
    sqlParameters[10] = new SqlParameter("@data_insc",
        SqlDbType.Date);
    sqlParameters[10].Value = Convert.ToDateTime(DateTime.Now);
    sqlParameters[11] = new SqlParameter("@data_nascimento",
        SqlDbType.DateTime);
    sqlParameters[11].Value =
        Convert.ToDateTime(_ut.DataNascimento);
    sqlParameters[12] = new SqlParameter("@postal",
        SqlDbType.NChar);
    sqlParameters[12].Value = Convert.ToString(_ut.Postal);
    sqlParameters[13] = new SqlParameter("@existe", SqlDbType.Int);
    sqlParameters[13].Direction = ParameterDirection.Output;

    db = new Database();

    int valor = db.executeProc("InserirUtilizador", sqlParameters);

    if (valor == 0)
```

```
            {
                return -1;
            }
            else
            {
                return valor;
            }
        }
        catch (Exception exp)
        {
            return -1;
        }
    }

            //Mais métodos
}
```

Na próxima camada, a camada BUS será a camada que fará a chamada da camada precedente de modo a inserir o utilizador na base de dados. As classes da camada de negócio têm o seguinte aspeto como se pode ver na listagem 6.5.

Listagem 6.5: Classe BUS Utilizador do modelo em 3 camadas

```
public class UtilizadorBUS
{
    private UtilizadoresDAO ut;
        public UtilizadorBUS()
        {
        ut = new UtilizadoresDAO();
        }

    public int NovoUtilizador(Utilizador _ut)
    {
        int valorRetorno;

        valorRetorno = ut.inserirUtilizador(_ut);
        if (valorRetorno == -1)
        {
            return -1;
        }
        else
        {
            return valorRetorno;
        }
    }
}
```

Agora que já temos a camada de acesso aos dados, a camada de negócio o que fazer com estas classes e como utiliza-las? O mais difícil já está feito. Agora

no código da aplicação onde nós quisermos que seja inserido o utilizador, basta referenciar a classe Utilizador que irá criar o objeto utilizador e referenciar a classe UtilizadoresBUS que recebe o objeto utilizador e já está.(Ver listagem 6.6). O utilizador será inserido na base de dados. Como se pode ver esta estrutura torna mais fácil qualquer mudança que possa ocorrer na aplicação. Pois um ficheiro contém o acesso aos dados separadamente de tudo o resto. O inserir na base de dados, neste caso o cliente, é feito com o auxílio de um procedimento, previamente criado. (Exemplo de procedimento em 6.7).

Listagem 6.6: Utilização do modelo em 3 camadas

```
...

            //Inserir Utilizadores
            Utilizador ut = new Utilizador();
            ut.Bi = lbBiCc.Text;
            ut.DataNascimento = Convert.ToDateTime(lbDataNascimento.Text);
            ut.Delegacao = Convert.ToInt32(DDListDelegacao.SelectedValue);
            ut.Email = tbEmailCliente.Text;
            ut.Fotografia = lbNomeFoto.Text;
            ut.Morada = tbMoradaCliente.Text;
            ut.Nif = lbNif.Text;
            ut.Nome = tbNomeCliente.Text;
            ut.NumUtilizador = lbNumCliente.Text;
            ut.Postal = tbPostal.Text;
            ut.Sobrenome = tbSobrenomeCliente.Text;
            ut.Telefone = tbContactoCliente.Text;
            UtilizadorBUS utBUS = new UtilizadorBUS();
            int id_utilizador = utBUS.NovoUtilizador(ut);

...
```

Listagem 6.7: Exemplo do procedimento Inserir Utilizador usado pelo modelo em 3 camadas

```
ALTER procedure [dbo].[InserirUtilizador] (
@num nvarchar(20),
@nome nvarchar(50),
@sobrenome nvarchar(50),
@bi nchar(12),
@nif nchar(9),
@email nvarchar(50),
@morada nvarchar(100),
@telefone nchar(9),
@foto nvarchar(50),
@delegacao int,
@data_insc datetime,
@data_nascimento datetime,
@postal nchar(8),
```

```
@existe int=-2 output
)
as

begin
Declare @id_tipo int;
select @id_tipo=ID_TIPO_UTILIZADORES from TIPOS_UTILIZADORES where
    TIPO_DEFEITO = 1;

select @existe=count(ID_UTILIZADOR) from UTILIZADORES where BI_UTILIZADOR
    = @bi or NIF_UTILIZADOR = @nif
if @existe = 0
begin
insert into UTILIZADORES
(NUM_UTILIZADOR, NOME_UTILIZADOR, SOBRENOME_UTILIZADOR, BI_UTILIZADOR,
    NIF_UTILIZADOR,EMAIL_UTILIZADOR, MORADA_UTILIZADOR,
    TELEFONE_UTILIZADOR, TIPOS_UTILIZADORES_ID_TIPO_UTILIZADORES,
    FOTOGRAFIA, ID_DELEGACAO, DATA_INSCRICAO, DATA_NASCIMENTO,
    POSTAL_UTILIZADOR) values
(Upper(@num), upper(@nome),upper(@sobrenome),@bi,@nif, @email,
    upper(@morada),@telefone, @id_tipo, @foto, @delegacao, @data_insc,
    @data_nascimento, @postal);

SET @existe = CAST(SCOPE_IDENTITY() AS INT)
end
else
begin
set @existe = 0
end
end
```

Problemas nesta fase:
Esta fase tornou-se confusa no início do desenvolvimento do modelo, pois era um conceito completamente novo para mim e como tudo o que é novidade, custa um pouco. Depois de entender realmente a estrutura tornou-se tudo mais fácil.

6.6 AJAX AutoComplete Extender

Nesta secção será apresentado a forma como foi integrado o AJAX AutoComplete Extender no projeto e o desenvolvimento do Serviço Web associado. Este controlo AJAX foi usado para apresentar os códigos postais nacionais na página de uma maneira intuitiva e mais rápida. Para isso e uma vez que os códigos postais estão guardados na base de dados é necessário ir buscar esses dados à base de dados e apresenta-los, para isso foi usado um serviço web.
Comecemos do início. Para que a caixa de texto use o extender tem qye se adicionar o extender. Na toolbox (já ensinámos a adicionar os controlos à toolbox do IDE no capítulo Aplicação Web Para Gestão de Ginásio na secção AJAX Control Toolkit)

arrastamos o controlo AutoCompleteExtender para cima da caixa de texto que queremos que tenha esta funcionalidade. De seguida nas propriedades do controlo temos duas propriedades que temos de configurar. São elas, a propriedade ServiceMethod que é onde vamos colocar o nome do serviço que irá devolver os códigos postais e a propriedade Service Path que contém o nome para o ficheiro com o serviço web. (Listagem 6.8)
A configuração pode ser vista aqui:

Listagem 6.8: Configuração do Auto Complete Extender

```
<asp:TextBox ID="tbPostal" runat="server" Text='<%#Bind("cod_postal") %>'
    AutoPostBack="True"></asp:TextBox>
<ajaxToolkit:AutoCompleteExtender runat="server"
    ID="tbPostaisAutoComplete" TargetControlID="tbPostal"
ServicePath="~/AutoCompleteList.asmx"
    ServiceMethod="GetCompletionListPostal" MinimumPrefixLength="1"
CompletionInterval="10" EnableCaching="true" CompletionSetCount="12"
    CompletionListCssClass="autocomplete_CompletionListElement" />
```

Estas são as propriedades para que o autocomplete extender funcione. Vamos dar uma espreitadela no ficheiro AutoCompleteList.asmx que poderá esclarecer melhor este processo (Listagem 6.9).

Listagem 6.9: Ficheiro AutoCompleteList.asmx

```
<%@ WebService Language="C#"
    CodeBehind="~/App_Code/AutoCompleteListSeguros.cs"
    Class="AutoCompleteList" %>
```

Este ficheiro referencia o webservice e o seu código encontra-se no caminho indicado com o nome AutoCompleteListSeguros.cs, é este o ficheiro que contém o código do serviço web. Vamos então descobrir o que existe nesse ficheiro. (Listagem 6.10)

Listagem 6.10: Ficheiro AutoCompleteListSeguros.cs

```
[WebService]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
//A próxima linha tem que estar presente para que o web service seja
    chamada pelo AJAX e ASP.NET
[System.Web.Script.Services.ScriptService]
public class AutoCompleteList : System.Web.Services.WebService {

    public AutoCompleteList () {

    }
            [WebMethod]
    public string[] GetCompletionListPostal(string prefixText, int count)
    {
        if (count == 0)
```

```
        {
            count = 10;
        }
        DataTable dt = GetRecordsPostal(prefixText);
        List<string> items = new List<string>(count);
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            string strName = dt.Rows[i][0].ToString();
            items.Add(strName);
        }
        return items.ToArray();
    }

    public DataTable GetRecordsPostal(string strName)
    {
        string strConn =
            ConfigurationManager.ConnectionStrings["WebsiteGim"].ConnectionString;
        SqlConnection con = new SqlConnection(strConn);
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;
        cmd.CommandType = System.Data.CommandType.Text;
        cmd.Parameters.AddWithValue("@cod_postal", strName);
        cmd.CommandText = "Select top 20 * from (Select
            distinct(cod_postal) from postais where cod_postal like
            '%'+@cod_postal+'%') as postal";
        DataSet objDs = new DataSet();
        SqlDataAdapter dAdapter = new SqlDataAdapter();
        dAdapter.SelectCommand = cmd;
        con.Open();
        dAdapter.Fill(objDs);
        con.Close();
        return objDs.Tables[0];
    }
}
```

Neste momento a nossa caixa de texto já será capaz de ir buscar os códigos postais à nossa base de dados, com base no texto digitado na caixa de texto. Para os leitores mais atentos, já devem ter reparado que o nome do método do serviço web é o nome que definimos nas propriedades do controlo AJAX. De salientar ainda que sem esta linha [System.Web.Script.Services.ScriptService] no serviço web não seria possível obter os dados.
Por agora já temos a funcionar a nossa caixa de texto com os códigos postais e agora como ir buscar as moradas associada ao código postal escolhido?

Vamos seguir o mesmo processo usado para o código postal mas nas propriedades no Service Method alteramos para GetCompletionListMorada que é o nome de outro método do serviço web e que nos permitirá ir buscar a morada associada ao código postal. Adicionemos então ao ficheiro AutoCompleteListSeguros.cs as seguintes linhas (Listagem 6.11

Listagem 6.11: Ficheiro AutoCompleteListSeguros.cs com WebMethod GetCompletionListMorada

```
[WebMethod]
public string[] GetCompletionListMorada(string prefixText, int count,
    string contextKey)
{
    if (count == 0)
    {
        count = 10;
    }
    DataTable dt = GetRecordsMorada(prefixText, contextKey);
    List<string> items = new List<string>(count);
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        string strName = dt.Rows[i][0].ToString();
        items.Add(strName);
    }
    return items.ToArray();
}

public DataTable GetRecordsMorada(string strName, string contextKey)
{
    string strConn =
        ConfigurationManager.ConnectionStrings["WebsiteGim"].ConnectionString;
    SqlConnection con = new SqlConnection(strConn);
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = con;
    cmd.CommandType = System.Data.CommandType.Text;
    cmd.Parameters.AddWithValue("@lugar", strName);
    if (contextKey != "0")
    {
        cmd.Parameters.AddWithValue("@postal", contextKey);
    }
    cmd.CommandText = "Select top 20 * from (Select distinct(lugar)
        from postais where lugar like '%'+@lugar+'%' and cod_postal =
        @postal) as lugar";
    DataSet objDs = new DataSet();
    SqlDataAdapter dAdapter = new SqlDataAdapter();
    dAdapter.SelectCommand = cmd;
    con.Open();
    dAdapter.Fill(objDs);
    con.Close();
    return objDs.Tables[0];
}
```

E será que já é suficiente? Observando a query do método podemos ver que o select espera dois parâmetros, nomeadamente o @lugar e o @postal e no método para o código postal só pedia o parâmetro @cod_postal que era facultado pela caixa

de texto. Neste caso o parâmetro facultado pela caixa de texto do autocomplete extender é o parâmetro @lugar que é o que nós queremos escrever e esperamos que o controlo AJAX nos ajude fazendo as suas sugestões. Então como fazer para passar o segundo parâmetro? Agora é muito simples mas que levou horas até descobrir como. A maneira de passar o valor para o segundo parâmetro (@postal) é através do contextKey que é uma propriedade do autocomplete extender. Pode ser feito da seguinte maneira, no evento Page_Load() da página passamos o valor da caixa de texto do código postal para o contextKey do autocomplete extender da morada (listagem 6.12) que quando é chamado no método do serviço web adiciona esse valor no parâmetro @postal.

Listagem 6.12: Passar valor do código postal para o contextKey

```
tbMoradaCliente_AutoCompleteExtender.ContextKey = tbPostal.Text;
```

Problemas nesta fase:
A maior dificuldade nesta etapa foi descobrir forma de passar o segundo parâmetro para o método da morada no serviço web.

6.7 Ficheiros com Códigos Postais, Freguesias, Concelhos e Distritos Nacionais

Já aqui falámos algumas vezes neste documento, sobre códigos postais, freguesias, mas nunca foi explicado como foram obtidos esses dados. Ora bem, pesquisando um pouco de modo a obter os códigos postais e respetivos concelhos, distritos e freguesias, surgiu o web site dos CTT (Correios de Portugal) no entanto os links para download que existiam, não estavam mais disponíveis. Felizmente que há outras páginas web que disponibilizam estes recursos. Os ficheiros foram obtidos através desta página [5]. Feito o download, vamos dar uma vista de olhos nos conteúdos do mesmo. Existem vários ficheiros nomeadamente um deles que explica a estrutura que pode ser encontrada nos restantes ficheiros. 6.13

Listagem 6.13: Estrutura do Ficheiro Distritos

```
2. Formato do ficheiro de distritos

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

O ficheiro de distritos contém os códigos administrativos dos distritos

e respectiva descrição.

Cada linha tem 2 campos de dados, separados por ponto e vírgula, contendo

os seguintes valores:

1) Designação: DD
```

```
   Conteúdo:  Código do Distrito

   Tipo:      Alfa-numérico, sempre preenchido

2) Designação: DESIG

   Conteúdo:  Designação do Distrito

   Tipo:      Alfa-numérico, sempre preenchido
```

Podemos ver então que no ficheiro de texto dos distritos 6.14 teremos um campo com o código dos distritos e outro com uma designação do mesmo e até sabemos o tipo de dados que podemos esperar.

Listagem 6.14: Distritos.txt

```
03;Braga
04;Bragança
05;Castelo Branco
06;Coimbra
07;Évora
08;Faro
09;Guarda
10;Leiria
11;Lisboa
12;Portalegre
13;Porto
14;Santarém
```

Cá está a estrutura do ficheiro 6.14 como esperado. No entanto estes mesmos ficheiros CSV (Comma Separated Values) contêm informação que não interessam para esta solução principalmente no ficheiro dos códigos postais 6.15 e como passar esta informação para a nossa base de dados e normalizar a informação que queremos?

Listagem 6.15: Exemplo do ficheiro Códigos Postais

```
11;06;33;21696;Lisboa;300150611;Rua;de;;;Caribe;;Lt
   1.13.03.3E;;;1990;036;LISBOA
11;06;33;21696;Lisboa;300160611;Passeio;do;;;Cantábrico;;Lt
   1.16.04B;;;1990;057;LISBOA
11;06;33;21696;Lisboa;300160611;Passeio;do;;;Cantábrico;;Lt
   1.16.04C;;;1990;057;LISBOA
11;06;33;21696;Lisboa;300170611;Passeio;;;;Heróis do Mar;Parque das
   Nações;;;;1990;061;LISBOA
11;06;33;21696;Lisboa;300170611;Passeio;;;;Heróis do
   Mar;;;;;1990;061;LISBOA
11;06;33;21696;Lisboa;300180611;Rua;;Comandante;;Cousteau;;Lt
   4.04.02H;;;1990;303;LISBOA
```

```
11;06;33;21696;Lisboa;300200611;Largo;;;;Diogo Cão;;;;;1990;073;LISBOA
```

Uma vez que os ficheiros como já foi referido, apresentam uma estrutura CSV (Comma Separated Values) [2] podemos tirar proveito dessa situação. O que foi feito foi uma página auxiliar 6.6, que com o auxílio de uma classe desenvolvida para o efeito chama os métodos contidos na classe consoante, se queremos carregar o ficheiro dos Distritos ou o ficheiro dos Concelhos por exemplo.
De seguida é apresentado um método 6.16 que permite inserir os códigos postais na base de dados uma vez que é o caso mais complicado de todos e até o que contém mais registos. A inserção na base de dados na tabela dos códigos postais chegou a ultrapassar as 4 horas em ambiente local (quer isto dizer que a base de dados estava na máquina local). A classe CodigosPostais pode ser consultada na íntegra nos Anexos.

Listagem 6.16: Método da classe CodigosPostais que permite inserir na base de dados os códigos postais

```
public int InserirPostais(String caminho)
{
    System.IO.StreamReader file = new System.IO.StreamReader(caminho,
        System.Text.Encoding.Unicode);
    String linha;
    String id_distrito = "";
    String id_concelho = "";
    String id_freguesia = "";
    String lugar = "";
    String cod_postal = "";
    String extensao_postal = "";
    String arteria = "";
    String rua = "";
    String nome_rua = "";
    String pre_nome_rua = "";
    String dos_das = "";

    try
    {
        while ((linha = file.ReadLine()) != null)
        {

            id_distrito = "";
            id_concelho = "";
            id_freguesia = "";
            lugar = "";
            cod_postal = "";
            extensao_postal = "";
            arteria = "";
            rua = "";
            nome_rua = "";
            pre_nome_rua = "";
```

```
dos_das = "";

String[] palavras = linha.Split(';');
id_distrito = palavras[0].ToString();
id_concelho = palavras[1].ToString();
id_freguesia = palavras[2].ToString();
lugar = palavras[4].ToString();
cod_postal = palavras[15].ToString();
extensao_postal = palavras[16].ToString();
arteria = palavras[3].ToString();
rua = palavras[6].ToString();
nome_rua = palavras[10].ToString();
pre_nome_rua = palavras[8].ToString();
dos_das = palavras[7].ToString();

cod_postal = cod_postal + "-" + extensao_postal;

SqlParameter[] sqlParameters = new SqlParameter[5];

if (rua == "")
{
    sqlParameters[0] = new SqlParameter("@LUGAR",
        SqlDbType.NVarChar);
    sqlParameters[0].Value = Convert.ToString(lugar);
}
else
{
    if (dos_das != "")
    {
        rua = rua + " " + dos_das;
    }
    if (pre_nome_rua != "")
    {
        rua = rua + " " + pre_nome_rua;
    }
    if (nome_rua != "")
    {
        rua = rua + " " + nome_rua;
    }
    sqlParameters[0] = new SqlParameter("@LUGAR",
        SqlDbType.NVarChar);
    sqlParameters[0].Value = Convert.ToString(rua);
}

sqlParameters[1] = new SqlParameter("@ID_DISTRITO",
    SqlDbType.NVarChar);
sqlParameters[1].Value = Convert.ToString(id_distrito);
```

```
                sqlParameters[2] = new SqlParameter("@ID_CONCELHO",
                    SqlDbType.NVarChar);
                sqlParameters[2].Value = Convert.ToString(id_concelho);

                sqlParameters[3] = new SqlParameter("@ID_FREGUESIA",
                    SqlDbType.NVarChar);
                sqlParameters[3].Value = Convert.ToString(id_freguesia);

                sqlParameters[4] = new SqlParameter("@COD_POSTAL",
                    SqlDbType.NVarChar);
                sqlParameters[4].Value = Convert.ToString(cod_postal);

                db.abrirConn();

                db.executeProcSemRetorno("InserirPostais", sqlParameters);

            }
            db.fecharConn();
            return 0;
        }
        catch (Exception e)
        {
            return -1;
        }
        finally
        {
            file.Close();
        }
    }
```

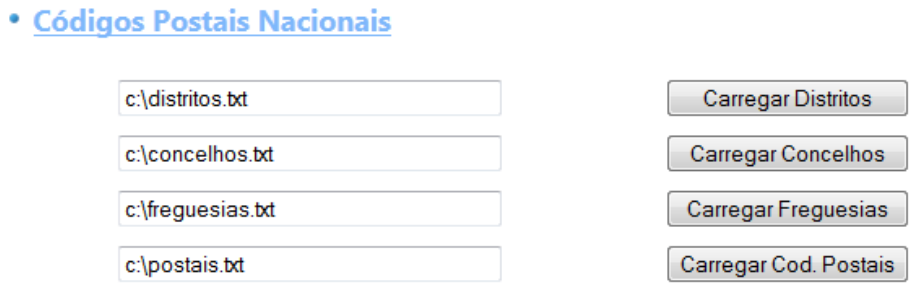


Figura 6.6: Página auxiliar de carregamento dos dados para a base de dados

No final do carregamento de todos os ficheiros para a base de dados os registos inseridos nas tabelas eram os seguintes:

Como se pode ver, limitámos-nos a juntar informação que nos convinha no caso das moradas e postais e separar alguns dados como o código da freguesia e a descrição e inserir esses dados nas tabelas de forma correta na base de dados.

Tabela	Número de Registos
Cod.Postais	245966
Distritos	29
Freguesias	4354
Concelhos	308

Tabela 6.27: Tabelas da base de dados e Número de registos

Problemas nesta fase:
Aqui maior dificuldade foi entender de que forma os ficheiros estão organizados. Nos primeiros carregamentos feitos havia um problema nos caracteres como por exemplo no caractere 'é' ou 'õ' e esse problema ocorria devido à codificação do ficheiro, alterando a codificação do ficheiro de ANSI para Unicode o problema ficou resolvido.

6.8 Sapo Mapas

Numa das páginas do projeto (figura 6.7) foi integrado o serviço de mapas de modo a que o cliente quando inscrito numa atividade lhe seja dado o caminho para essa atividade. A escolha na API dos mapas recaiu na API do Sapo Mapas, porque para além de ser um projeto nacional a documentação e os exemplos estão em português.

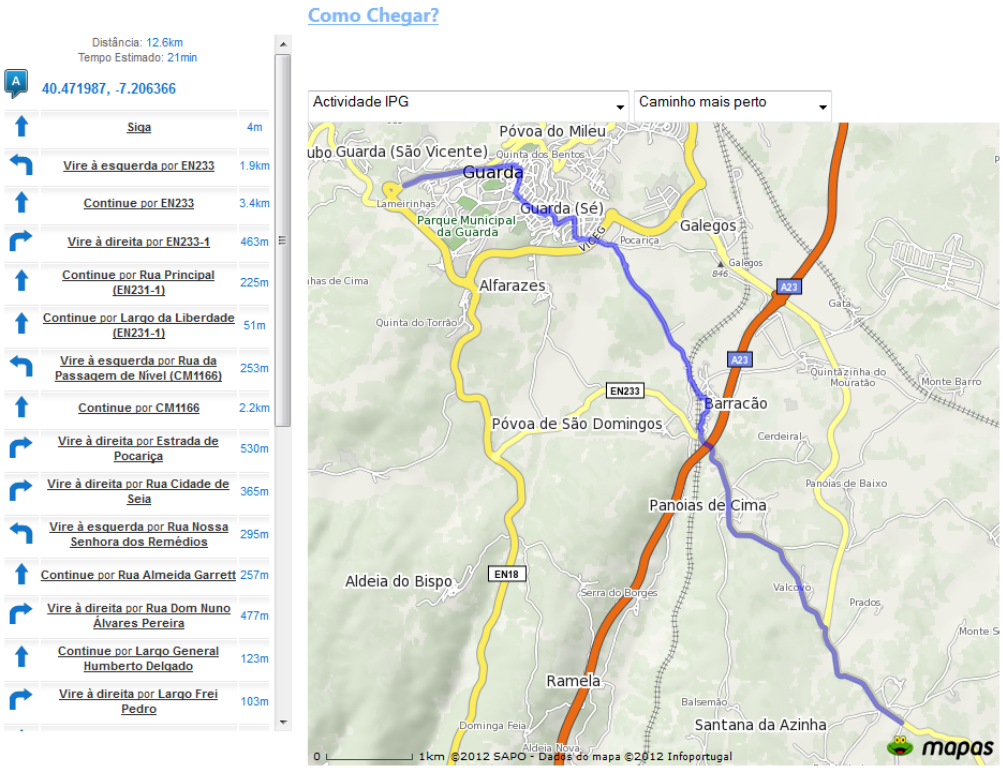


Figura 6.7: Solução com Sapo Mapas

As atividades muitas vezes são longe da residência do cliente e muitos se devem perguntar, então mas como chego lá? Foi a pensar nisso que esta página foi feita.

Quando a atividade é criada e é inserido o código postal ou a morada onde a atividade terá lugar, automáticamente é feita uma pesquisa usando este serviço de modo a obter as coordenadas geográficas (Longitude e Latitude) do local que será guardado posteriormente na base de dados com a restante informação relativa à atividade. Essa informação é obtida fazendo uma pesquisa no serviço de mapas pelo código postal, o serviço por sua vez retorna pontos de interesse (POIS) do código postal fornecido, bastando então aproveitar o primeiro elemento da estrutura dos pontos de interesse e obter a sua longitude e latitude. Aqui o mapa não é apresentado apenas se usa o serviço para obter as coordenadas geográficas.

Listagem 6.17: Código de exemplo da integração dos serviços Sapo Mapas na obtenção de coordenadas geográficas

```
<link type="text/css" rel="stylesheet"
    href="../Mapas/SintaxHighlighter/styles/shCore.css" />
<link type="text/css" rel="stylesheet"
    href="../Mapas/SintaxHighlighter/styles/shThemeDefault.css" />
<script type="text/javascript"
    src="../Mapas/SintaxHighlighter/scripts/shCore.js"></script>
<script type="text/javascript"
    src="../Mapas/SintaxHighlighter/scripts/shBrushJScript.js"></script>
<script type="text/javascript">
    SyntaxHighlighter.config.clipboardSwf =
        '../Mapas/SintaxHighlighter/scripts/clipboard.swf';
    SyntaxHighlighter.all();
</script>
<script type="text/javascript"
    src="http://js.sapo.pt/Bundles/SAPOMapsAPI.js"></script>

    <script type="text/javascript">
        window.onload = getDirections;

        function getDirections() {
            var cod_postal = '<%=tbPostal.Text%>';
            var morada = '<%=tbMoradaCliente.Text%>';
            search = new SAPO.Maps.Search();
            search.registerEvent('completed', this, completed);
            if (cod_postal != "") {
                search.cancel();
                search.search(cod_postal);
            }
            else if (morada != "") {
                searchE.cancel();
                search.search(morada);
            }
        }

        //Hanlders
        function completed(search) {
```

```
            var pois = search.getPOIS();
            var Longitude = pois[1].Longitude;
            var Latitude = pois[1].Latitude;
            var labelLat =
                document.getElementById('<%=tbLatitude.ClientID%>');
            labelLat.value = Latitude;
            var labelLong =
                document.getElementById('<%=tbLongitude.ClientID%>');
            labelLong.value = Longitude;
        }

        function debug() { }
</script>
```

Na listagem 6.17 podemos ver o código usado na página de modo a obter essa informação. Mas para quê obter estas coordenadas? Vão ser úteis onde?
Pois bem, respondendo à questão anterior, as coordenadas vão ser úteis na página que apresenta o mapa e as direções de como chegar ao destino. Nessa mesma página o que é feito é o seguinte, são obtidas as coordenadas geográficas do utilizador seguindo o mesmo raciocínio anterior e fazemos um pedido ao serviço de modo a nos devolver a direção e o mapa entre as coordenadas geográficas obtidas do utilizador e as coordenadas da atividade em questão. O código pode ser consultado na listagem 6.18.

Listagem 6.18: Código de exemplo da integração dos serviços Sapo Mapas na obtenção de direções entre pontos

```
            <link type="text/css" rel="stylesheet"
                href="../Mapas/SintaxHighlighter/styles/shCore.css" />
<link type="text/css" rel="stylesheet"
    href="../Mapas/SintaxHighlighter/styles/shThemeDefault.css" />
<script type="text/javascript"
    src="../Mapas/SintaxHighlighter/scripts/shCore.js"></script>
<script type="text/javascript"
    src="../Mapas/SintaxHighlighter/scripts/shBrushJScript.js"></script>
<script type="text/javascript">
    SyntaxHighlighter.config.clipboardSwf =
        '../Mapas/SintaxHighlighter/scripts/clipboard.swf';
    SyntaxHighlighter.all();
</script>
<script type="text/javascript"
    src="http://js.sapo.pt/Bundles/SAPOMapsAPI.js"></script>
<script type="text/javascript">

    var cod_postal = '<%=Label1.Text%>';

    window.onload = getDirections;

    function getDirections() {
```

```
            map = new SAPO.Maps.Map("map");

                search = new SAPO.Maps.Search();
                search.registerEvent('completed', this, completed);
                search.search(cod_postal);
        }

        //Hanlders
        function completed(search) {
            var pois = search.getPOIS();
            var LongitudeP = pois[1].Longitude;
            var LatitudeP = pois[1].Latitude;
            var LongitudeC = '<%=lbLongC.Text%>';
            var LatitudeC = '<%=lbLatC.Text%>';
            var modo = '<%=lbCaminho.Text%>';

            sd = new SAPO.Maps.Directions(map, "results");
            sd.registerEvent("completed", this, function (directions) {
                var x = 1;
            });
            sd.getDirections(new OpenLayers.LonLat(LongitudeP, LatitudeP),
                new OpenLayers.LonLat(LongitudeC, LatitudeC), { description:
                "text", mode: modo });
        }

        function debug() { }
    </script>
```

Problemas nesta fase:
Estaria a mentir se dissesse que esta foi uma parte fácil. Pois uma vez que a API trabalha com a linguagem Javascript e de eu não estar familiarizado com a linguagem em si, foi necessário descobrir como ter acesso aos conteúdos das caixas de texto ou label que continham informações necessárias para fazer a pesquisa no serviço. Foi necessário, fazer alguns exemplos em páginas à parte e ir integrando aos poucos na página da solução. A consulta dos exemplos e documentação disponibilizados na página respetiva à API dos mapas[13] foi essencial.

6.9 Gerar Documento Comprovativo e Envio por Email

Uma das funcionalidades desenvolvidas foi, gerar documentos em formato PDF e enviar esse documento por email para o cliente. Por exemplo, quando um cliente se inscreve numa modalidade é gerado um documento comprovativo em como o cliente se inscreveu nessa modalidade e pagou, esse documento é enviado por email para o cliente e é disponibilizado ao funcionário para guardar ou imprimir o documento. O documento tem o formato da figura 6.9.

União Karaté Shotokan das Beiras

Avenida Estádio Municipal
Contacto:925645558
N. Contribuinte:250123456

UKSB
UNIÃO DE KARATE SHOTOKAN DAS BEIRAS

N. Identificação Fiscal:000000000
N. Factura:MZZU7YDIY7
N. Cliente:UTL-16-15-Z92KV8PMY4

DIOGO FERNANDES

AVENIDA DOUTOR FRANCISCO SÁ CARNEIRO
6300-559 SÃO VICENTE
GUARDA

Modalidades/Actividades	Preço
Actividade IPG	12,50 €

Total: 12,5 €

O Funcionário:

Data de emissão: 08-10-2012

---

root

Figura 6.8: Documento gerado

Mas para que o documento tenha este formato e para facilitar a construção do mesmo, foi construído um template em HTML com posições já definidas para cada conteúdo do documento. Como se pode ver na figura 6.9 o template não nos diz muito nem se assemelha muito ao documento gerado (figura 6.9), apenas são visíveis alguns pontos em comum. No template o que está entre parêntesis retos é o que irá ser substituído pelo conteúdo que queremos, quando geramos o documento. Para isso foi criada uma classe que usa a biblioteca iTextSharp e nessa classe é gerado o documento em formato PDF.

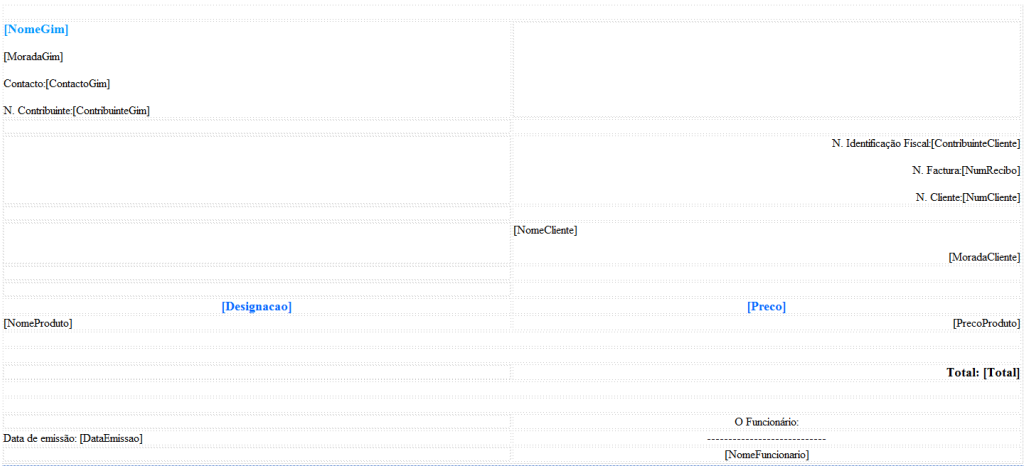


Figura 6.9: Template HTML do Documento

Vamos ver em mais detalhe a classe que cria o documento (Listagem 6.19)
O método da classe recebe como parâmetros de entrada o nome do cliente, morada, logotipo da instituição, actividades/modalidades e preços das mesmas que devem constar no documento, entre outros. A classe pode ser consultada na íntegra nos Anexos deste documento.

Listagem 6.19: Método criaPDFInscricao

```
Document doc = new Document(PageSize.A4, 50, 50, 25, 25);
MemoryStream ms = new MemoryStream();
PdfWriter pdf = PdfWriter.GetInstance(doc, ms);
                        try
                              {
    doc.Open();
    //Adicionar Elementos
    string conteudoHTML =
        File.ReadAllText(HttpContext.Current.Server.MapPath("~/templatePDF.html"));
    conteudoHTML = conteudoHTML.Replace("[NomeGim]", _nomeGim);
    conteudoHTML = conteudoHTML.Replace("[MoradaGim]", _moradaGim);
    conteudoHTML = conteudoHTML.Replace("[ContactoGim]",
        _contactoGim);
    conteudoHTML = conteudoHTML.Replace("[ContribuinteGim]",
        _nifGim);
    conteudoHTML = conteudoHTML.Replace("[ContribuinteCliente]",
        _nifCliente);
```

```
conteudoHTML = conteudoHTML.Replace("[NumRecibo]", _numRecibo);
conteudoHTML = conteudoHTML.Replace("[NumCliente]",
    _numCliente);
conteudoHTML = conteudoHTML.Replace("[NomeCliente]",
    _nomeCliente);
conteudoHTML = conteudoHTML.Replace("[MoradaCliente]",
    _moradaCliente);
conteudoHTML = conteudoHTML.Replace("[ContactoGim]",
    _contactoGim);
conteudoHTML = conteudoHTML.Replace("[Designacao]",
    "Modalidades/Actividades");
conteudoHTML = conteudoHTML.Replace("[Preco]", "Preço");
conteudoHTML = conteudoHTML.Replace("[ContactoGim]",
    _contactoGim);

foreach (System.Web.UI.WebControls.ListItem li in _modAct)
{
    produtos += "<p/>" + li;
}

foreach (System.Web.UI.WebControls.ListItem li in _precos)
{
    precos += "<p/>" + li;
}

conteudoHTML = conteudoHTML.Replace("[NomeProduto]", produtos);
conteudoHTML = conteudoHTML.Replace("[PrecoProduto]", precos);
conteudoHTML = conteudoHTML.Replace("[Total]", _total);
conteudoHTML = conteudoHTML.Replace("[DataEmissao]", _data);
conteudoHTML = conteudoHTML.Replace("[NomeFuncionario]",
    _nomeFuncionario);
```

Observando a listagem 6.19, podemos ver a maneira como os conteúdos são substituídos no template e produzem o documento final. Para permitir o anexo do documento no email o método que cria o ficheiro PDF retorna um MemoryStream que permite guardar o objeto criado em memória. Fazendo um ponto da situação neste momento, para criar o documento basta referenciar o método que cria o PDF passando os parâmetros desse método e o documento passa a estar disponível em memória.
O próximo passo é anexar esse documento e enviar por email ao cliente. Para isso foi criada mais uma classe com um método que recebe o endereço eletrónico para onde enviar o documento, o assunto da mensagem, entre outros. Mas antes de se poder iniciar o processo de envio de emails, foi necessário adicionar algumas linhas na configuração da aplicação, nomeadamente no ficheiro web.config. (Listagem6.20)

Listagem 6.20: Configuração do email no ficheiro web.config

```
<system.net>
  <mailSettings>
```

```
    <smtp from="uksbguarda@gmail.com">
      <network host="smtp.gmail.com" password="uksbgu@rda"
          userName="uksbguarda@gmail.com" port="25" enableSsl="true" />
    </smtp>
  </mailSettings>
</system.net>
```

Essencialmente é necessário configurar a porta do serviço de email usado, neste caso é um email da google que usa a porta 25, configurar o host do serviço de email e depois facultar o email e a password. Agora sim já estamos preparados para enviar emails. Vamos agora conhecer a classe Email.cs (6.21) que nos permite usar a funcionalidade. Mais uma vez a classe desenvolvida pode ser consultada na sua totalidade nos Anexos deste documento.

Listagem 6.21: Método que permite enviar emails com anexos

```
public static Boolean enviaEmailComAnexo(string _nomeQueEnvia, string
    _emailQueRecebe, string _assunto, string _corpo, MemoryStream _anexos,
    string _nomeAnexos)
    {
        try
        {
            Configuration config =
                WebConfigurationManager.OpenWebConfiguration
                                            (HttpContext.Current.Request.ApplicationPath);
            MailSettingsSectionGroup settings =
                (MailSettingsSectionGroup)config.GetSectionGroup("system.net/mailSettings");
            MailMessage mail = new MailMessage();
            mail.From = new MailAddress(settings.Smtp.From, _nomeQueEnvia);
            mail.To.Add(new MailAddress(_emailQueRecebe));
            mail.Subject = _assunto;
            mail.Body = _corpo;
            mail.IsBodyHtml = true;


            mail.Attachments.Add(new Attachment(_anexos, _nomeAnexos,
                "application/pdf"));

            SmtpClient smtp = new SmtpClient();
            smtp.EnableSsl = true;
            try{
                smtp.Send(mail);
                return true;
            }catch (Exception exp)
            {
                return false;
            }
        }
        catch (Exception exp)
```

```
        {
            return false;
        }
                                return;
    }
```

Basicamente o que se pode retirar do método (6.21), é que com os parâmetros que recebe constrói uma espécie de estrutura de uma mensagem de email tem ou seja , cabeçalho, corpo e os endereços ou o endereço para onde enviar o email e um anexo, enviando de seguida a mensagem.

Problemas nesta fase:
As dificuldades sentidas nesta fase essencialmente foram na construção do documento e a forma de anexar o documento ao email. Na construção do documento e antes de usar o template como base do documento, os conteúdos saíam de sítio e tornavam o documento confuso. Esta questão foi resolvida com a introdução do template como base para o documento PDF.

Capítulo 7

Conclusões e trabalho futuro

7.1 Conclusões

Durante o desenvolvimento da solução, foram surgindo problemas e desafios para resolver. Felizmente todos os desafios foram superados com mais ou menos dificuldade. O trabalho desenvolvido serviu para melhorar os conhecimentos na plataforma ASP.NET, em particular na linguagem de programação C#, mas essencialmente para ter contacto com novas tecnologias nomeadamente as API que não é comum serem incorporadas em trabalhos mais frequentes de unidades curriculares. Serviu também para ter outras perspetivas de desenvolvimento de software, por exemplo o desenvolvimento de software por '3 Tier Architecture' Modelo em 3 camadas, o uso de procedimentos no desenvolvimento, que acaba por não ser totalmente novo uma vez que na unidade curricular Base de Dados se recorre ao desenvolvimento de procedimentos entre outros, o que aqui foi novo foi a integração desses procedimentos depois na aplicação em si. O desenvolvimento de procedimentos que era até aqui meu conhecido era na linguagem PL/SQL, a linguagem de desenvolvimento da Oracle. No projeto foi usada tecnologia Microsoft e a linguagem T-SQL foi a linguagem usada no desenvolvimento dos procedimentos, linguagem na qual foi necessário aprender as bases e o seu funcionamento.

Em suma, o trabalho desenvolvido pessoalmente é me satisfatório, nalgumas situações superei-me a mim mesmo pensando que não seria capaz de certas coisas mas fui conseguindo pequenos objetivos aos poucos. Mas acima de tudo a conclusão principal a que cheguei no final desta etapa é que, se não enfrentarmos os problemas de frente nunca serão resolvidos na sua totalidade e quanto mais tempo se arrasta o problema, pior.
Foi muito gratificante o desenvolvimento da aplicação essencialmente pelo novo conhecimento adquirido, mas também por ser uma chamada de atenção e uma pequena preparação para o futuro. Certamente que no futuro, erros que foram sendo cometidos neste projeto, serão tidos em conta em projetos futuros de modo a não cometer os mesmos erros. Quanto à aplicação em si cumpre com as funcionalidades inicialmente previstas, foram adicionadas algumas funcionalidades extra como os mapas que permitem obter as direções para uma determinada atividade. No desenvolvimento da mesma, foi notória a importância do uso de várias linguagens como Javascript em projetos para a web pois permitem-nos integrar serviços facilmente e

realizar operações de forma mais fácil. Adicionam funcionalidades do lado do cliente o que também é muito importante. Dou aqui o exemplo da integração dos controlos AJAX, que adicionaram funcionalidades do lado do cliente e tornam a aplicação visualmente mais atraente nalguns casos e facilitam o uso da aplicação por parte do utilizador.

A aprendizagem é constante e a experiência leva-nos a melhorar cada vez mais!

7.2 Trabalho futuro

Neste ponto gostaria de começar por dizer que nenhum projeto está totalmente acabado, pois existem sempre aspetos a melhorar ou funcionalidades a acrescentar. Apesar da aplicação final estar num nível em que poderá estar em total funcionamento básico e de cumprir com as funcionalidades inicialmente previstas, haverá ainda outras funcionalidades que tornariam a aplicação mais interessante. Essas funcionalidades passariam por:

- Permitir a gestão de horários de utilizadores/funcionários/instrutores;
- Permitir a gestão de seguros e atestados médicos;
- Permitir ao gestor, aceder de forma gráfica a dados: sobre seguros, sobre acessos à plataforma;
- Implementar um sistema de pesquisa na página;
- Integrar a aplicação desenvolvida com redes sociais;
- Possibilitar a gestão adaptada para plataformas móveis como tablets ou smartphones e toda a tecnologia inerente à mesma;
- Integrar um sistema de presenças, isto é em vez de ser o funcionário a marcar presenças manualmente, seria o cliente com o 'picar de cartão' num dispositivo a marcar automaticamente a presença ou outro sistema;
- Integrar um sistema de SMS onde seria possível avisar os clientes do adiamento/cancelamento de atividades/modalidades, desejar um feliz aniversário ao cliente na sua data,entre outras funcionalidades;
- Integrar um sistema de pagamentos por multibanco na aplicação, por exemplo para que o cliente não tenha que se deslocar à entidade para pagar as mensalidades ou anuidades;
- No caso do ponto anterior estar integrado, permitir as inscrições à distância uma vez que o pagamento também já seria permitido;

Como se pode ver pelos pontos anteriores, há sempre funcionalidades novas à acrescentar portanto não se pode afirmar que o projeto esteja concluído na sua totalidade e provávelmente haverá mais funcionalidades que se podem acrescentar.

Bibliografia

[1] Ckeditor for asp.net. http://docs.cksource.com/CKEditor.NET, 2012.

[2] Comma-separated values. http://pt.wikipedia.org/wiki/Comma-separated_values, 2012.

[3] Codeplex. Ajax control toolkit. http://ajaxcontroltoolkit.codeplex.com/, 2012.

[4] Luís Abreu com a colaboração de João Paulo Carreiro. Asp.net 4.0 curso completo, edição 3. Technical report, FCA, 2011.

[5] elsif. Codigos postais. http://downloads.elsif.pt/Codigos%20Postais/Modelo%20B/, 2012.

[6] Henrique Graça. Asp.net membership. http://pplware.sapo.pt/tutoriais/asp-net-%E2%80%93-membership/, Dezembro 2010.

[7] iText. itext. http://itextpdf.com/, 2012.

[8] Microsoft. Ajax library. http://www.asp.net/ajaxlibrary/act.ashx, 2012.

[9] Microsoft. O que é .net framework. social/msdn.microsoft.com/Forums/pt/504/thread/fc08f845-b3cd-4a60-bbb4-6ffb8a9891f3, Outubro 2012.

[10] Scott Mitchell. Creating pdf documents with asp.net and itextsharp. http://www.4guysfromrolla.com/articles/030911-1.aspx, 2012.

[11] ng.sis. Fitgest. http://www.ngsis.com/, 2012.

[12] Ben Rush. Scriptmanager enables ajax in your web apps. http://msdn.microsoft.com/en-us/magazine/cc163354.aspx, Setembro 2007.

[13] Sapo. Ajax. http://api.mapas.sapo.pt/, 2012.

[14] Innux Technologies. A empresa. http://www.innux.com/pt/empresa.html, 2012.

[15] Abhimanyu K Vatsa. Creating pdf files in asp.net using itextsharp. http://www.c-sharpcorner.com/UploadFile/abhikumarvatsa/creating-pdf-files-in-Asp-Net-using-itextsharp/, Junho 2012.

[16] w3schools. Ajax. `http://www.w3schools.com/ajax/default.asp`, Outubro 2012.

[17] Wikipédia. Caso de uso. `http://pt.wikipedia.org/wiki/Caso_de_uso`, Outubro 2012.

[18] Wikipédia. Webservice. `http://en.wikipedia.org/wiki/Web_service`, 2012.

Anexos

Anexo A

Listagem de programas

A.1 Template da página web

```
<!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>
<link href="styles.css" rel="stylesheet" type="text/css" />
</head>

<body>
<div id="page">
<div id="header">
  <div id="top">
    <div class="title" id="title"><span style="">Titulo</span></div>
    <div id="social">
      <div id="icons">icons</div>
      <div id="search">pesquisa</div>
    </div>
  </div>
  <div id="logo"></div>
  <div id="login">login</div>
</div>
<div id="menu"></div>
<div id="hdimage">imagem</div>
<div id="content">Content for  id "content" Goes Here</div>
<div id="footer">footer</div></div>
</body>
</html>
```

A.2 Ficheiro CSS do Template

```
#titulo {
color: #6CF;
float: left;
height: 140px;
font-family: Georgia, "Times New Roman", Times, serif;
font-size: 24px;
text-align: center;
width:150px;
}
#header {
height: 140px;
width: 768px;
}
#social {
```

```
height: 40px;
}
#header {
height: 200px;
width: 960px;
}
#menu {
height: 40px;
width: 960px;
background-image: url(../images/bar.png);
background-repeat: repeat-x;
}
#hdimage {
height: 145px;
width: 960px;
}
#content {
width: 950px;
    margin-left: 10px;
}
#footer {
height: 30px;
width: 960px;
margin-top: 3px;
}
#top {
height: 50px;
}
#title {
width: 50%;
height: 50px;
float: left;
}
#social {
height: 50px;
width: 50%;
float: right;
}
#logo
{
    background-repeat: no-repeat;
height: 140px;
width: 50%;
float: left;
}
#login {
float: right;
height: 140px;
width: 50%;
}
#icons {
float: left;
height: 50px;
width: 40%;
}
#search {
float: right;
height: 50px;
width: 60%;
}
#page {
    width: 960px;
    background-color: #FFFFFF;
    margin: 20px auto 0px auto;
    border: 1px solid #e1e1e1;
}
#barra_sep1
{
    margin-top: 5px;
}
```

```
#sepfooter
{
    background-image: url(../images/sep.png);
    background-repeat: repeat-x;
    margin-top:15px;
}
#rightContent
{
    float:right;
    width:30%;
}
#leftContent
{
    float:left;
    width:70%;
}

/* Default */
body
{
    background: #e1e1e1;
    font-size: .80em;
    font-family: "Helvetica Neue", "Lucida Grande", "Segoe UI", Arial, Helvetica, Verdana, sans-serif;
    margin: 0px;
    padding: 0px;
    margin-left: 50px;
    color: #696969;
}

a:link, a:visited
{
    color: #4c8cd4;
}

a:hover
{
    color: #588bc6;
    text-decoration: none;
}

a:active
{
    color: #6dadf5;
}

.autocomplete_CompletionListElement
{
    margin: 0px;
    background-color: White;
    cursor: default;
    overflow-y: auto;
    overflow-x: hidden;
    max-height:180px;
    text-align: justify;
    border: 1px solid #777;
    z-index:10000;
}
```

A.3 Classe CodigosPostais

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.IO;
using System.Data.SqlClient;
using System.Configuration;
using System.Data;
```

```
/// <summary>
/// Summary description for CodigosPostais
/// </summary>
public class CodigosPostais
{
    Database db = new Database();

public CodigosPostais()
{
//
// TODO: Add constructor logic here
//
}

    public int InserirConcelho(String caminho)
    {
        System.IO.StreamReader file = new System.IO.StreamReader(caminho, System.Text.Encoding.Unicode);
        String linha;
        String id_distrito = "";
        String id_concelho = "";
        String nome_concelho = "";

        try
        {
            while ((linha = file.ReadLine()) != null)
            {
                String[] palavras = linha.Split(';');
                id_distrito = palavras[0].ToString();
                id_concelho = palavras[1].ToString();
                nome_concelho = palavras[2].ToString();

                SqlParameter[] sqlParameters = new SqlParameter[3];
                int valorRetorno;

                sqlParameters[0] = new SqlParameter("@NOME_CONCELHO", SqlDbType.NVarChar);
                sqlParameters[0].Value = Convert.ToString(nome_concelho);

                sqlParameters[1] = new SqlParameter("@ID_CONCELHO", SqlDbType.NVarChar);
                sqlParameters[1].Value = Convert.ToString(id_concelho);

                sqlParameters[2] = new SqlParameter("@ID_DISTRITO", SqlDbType.NVarChar);
                sqlParameters[2].Value = Convert.ToString(id_distrito);

                db.abrirConn();

                valorRetorno = db.executeProcSemRetorno("InserirConcelhos", sqlParameters);
            }
            db.fecharConn();
            return 0;
        }
        catch (Exception e)
        {
            return -1;
        }
        finally
        {
            file.Close();
        }
    }

    public int InserirDistrito(String caminho)
    {
        System.IO.StreamReader file = new System.IO.StreamReader(caminho,System.Text.Encoding.Unicode);
        String linha;
        String id_distrito = "";
        String nome_distrito = "";

        try
        {
```

```
            while ((linha = file.ReadLine()) != null)
            {
                String[] palavras = linha.Split(';');
                id_distrito = palavras[0].ToString();
                nome_distrito = palavras[1].ToString();

                SqlParameter[] sqlParameters = new SqlParameter[2];

                sqlParameters[0] = new SqlParameter("@NOME_DISTRITO", SqlDbType.NVarChar);
                sqlParameters[0].Value = Convert.ToString(nome_distrito);

                sqlParameters[1] = new SqlParameter("@ID_DISTRITO", SqlDbType.NVarChar);
                sqlParameters[1].Value = Convert.ToString(id_distrito);

                db.abrirConn();

                db.executeProcSemRetorno("InserirDistritos", sqlParameters);

            }
            db.fecharConn();
            return 0;
        }
        catch (Exception e)
        {
            return -1;
        }
        finally
        {
            file.Close();
        }
    }

    public int InserirFreguesia(String caminho)
    {
        System.IO.StreamReader file = new System.IO.StreamReader(caminho, System.Text.Encoding.Unicode);
        String linha;
        String id_distrito = "";
        String id_concelho = "";
        String id_freguesia = "";
        String nome_freguesia = "";

        try
        {
            while ((linha = file.ReadLine()) != null)
            {
                String[] palavras = linha.Split(';');
                id_distrito = palavras[0].ToString();
                id_concelho = palavras[1].ToString();
                id_freguesia = palavras[2].ToString();
                nome_freguesia = palavras[3].ToString();

                SqlParameter[] sqlParameters = new SqlParameter[4];

                sqlParameters[0] = new SqlParameter("@NOME_FREGUESIA", SqlDbType.NVarChar);
                sqlParameters[0].Value = Convert.ToString(nome_freguesia);

                sqlParameters[1] = new SqlParameter("@ID_DISTRITO", SqlDbType.NVarChar);
                sqlParameters[1].Value = Convert.ToString(id_distrito);

                sqlParameters[2] = new SqlParameter("@ID_CONCELHO", SqlDbType.NVarChar);
                sqlParameters[2].Value = Convert.ToString(id_concelho);

                sqlParameters[3] = new SqlParameter("@ID_FREGUESIA", SqlDbType.NVarChar);
                sqlParameters[3].Value = Convert.ToString(id_freguesia);

                db.abrirConn();

                db.executeProcSemRetorno("InserirFreguesias", sqlParameters);

            }
```

```
            db.fecharConn();
            return 0;
        }
        catch (Exception e)
        {
            return -1;
        }
        finally
        {
            file.Close();
        }
    }


    public int InserirPostais(String caminho)
    {
        System.IO.StreamReader file = new System.IO.StreamReader(caminho, System.Text.Encoding.Unicode);
        String linha;
        String id_distrito = "";
        String id_concelho = "";
        String id_freguesia = "";
        String lugar = "";
        String cod_postal = "";
        String extensao_postal = "";
        String arteria = "";
        String rua = "";
        String nome_rua = "";
        String pre_nome_rua = "";
        String dos_das = "";

        try
        {
            while ((linha = file.ReadLine()) != null)
            {

                id_distrito = "";
                id_concelho = "";
                id_freguesia = "";
                lugar = "";
                cod_postal = "";
                extensao_postal = "";
                arteria = "";
                rua = "";
                nome_rua = "";
                pre_nome_rua = "";
                dos_das = "";

                String[] palavras = linha.Split(';');
                id_distrito = palavras[0].ToString();
                id_concelho = palavras[1].ToString();
                id_freguesia = palavras[2].ToString();
                lugar = palavras[4].ToString();
                cod_postal = palavras[15].ToString();
                extensao_postal = palavras[16].ToString();
                arteria = palavras[3].ToString();
                rua = palavras[6].ToString();
                nome_rua = palavras[10].ToString();
                pre_nome_rua = palavras[8].ToString();
                dos_das = palavras[7].ToString();

                cod_postal = cod_postal + "-" + extensao_postal;

                SqlParameter[] sqlParameters = new SqlParameter[5];

                if (rua == "")
                {
                    sqlParameters[0] = new SqlParameter("@LUGAR", SqlDbType.NVarChar);
                    sqlParameters[0].Value = Convert.ToString(lugar);
                }
                else
```

```
                {
                    if (dos_das != "")
                    {
                        rua = rua + " " + dos_das;
                    }
                    if (pre_nome_rua != "")
                    {
                        rua = rua + " " + pre_nome_rua;
                    }
                    if (nome_rua != "")
                    {
                        rua = rua + " " + nome_rua;
                    }
                    sqlParameters[0] = new SqlParameter("@LUGAR", SqlDbType.NVarChar);
                    sqlParameters[0].Value = Convert.ToString(rua);
                }

                sqlParameters[1] = new SqlParameter("@ID_DISTRITO", SqlDbType.NVarChar);
                sqlParameters[1].Value = Convert.ToString(id_distrito);

                sqlParameters[2] = new SqlParameter("@ID_CONCELHO", SqlDbType.NVarChar);
                sqlParameters[2].Value = Convert.ToString(id_concelho);

                sqlParameters[3] = new SqlParameter("@ID_FREGUESIA", SqlDbType.NVarChar);
                sqlParameters[3].Value = Convert.ToString(id_freguesia);

                sqlParameters[4] = new SqlParameter("@COD_POSTAL", SqlDbType.NVarChar);
                sqlParameters[4].Value = Convert.ToString(cod_postal);

                db.abrirConn();

                db.executeProcSemRetorno("InserirPostais", sqlParameters);

            }
            db.fecharConn();
            return 0;
        }
        catch (Exception e)
        {
            return -1;
        }
        finally
        {
            file.Close();
        }
    }
}
```

A.4 Classe Email

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Net.Mail;
using System.Net;
using System.IO;
using System.Configuration;
using System.Net.Configuration;
using System.Web.Configuration;

/// <summary>
/// Summary description for Email
/// </summary>
public class Email
{
public Email()
{
```

```
}


    public static Boolean enviaEmail(string _nomeQueEnvia, string _emailQueRecebe, string _assunto, string _corpo)
    {
        try
        {

            Configuration config = WebConfigurationManager.OpenWebConfiguration(HttpContext.
Current.Request.ApplicationPath);
            MailSettingsSectionGroup settings = (MailSettingsSectionGroup)config.
GetSectionGroup("system.net/mailSettings");
            MailMessage mail = new MailMessage();
            mail.From = new MailAddress(settings.Smtp.From, _nomeQueEnvia);
            mail.To.Add(new MailAddress(_emailQueRecebe));
            mail.Subject = _assunto;
            mail.Body = _corpo;
            mail.IsBodyHtml = true;

            SmtpClient smtp = new SmtpClient();
            smtp.EnableSsl = true;
            try
            {
                smtp.Send(mail);
                return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }
        catch (Exception exp)
        {
            return false;
        }
    }

    public static Boolean enviaEmailContactos(string _nomeQueEnvia, string _emailQueEnvia, string _assunto, string _c
    {
        try
        {

            Configuration config = WebConfigurationManager.OpenWebConfiguration(HttpContext.
Current.Request.ApplicationPath);
            MailSettingsSectionGroup settings = (MailSettingsSectionGroup)config.
GetSectionGroup("system.net/mailSettings");
            MailMessage mail = new MailMessage();
            mail.Sender = new MailAddress(_emailQueEnvia);
            mail.To.Add(new MailAddress(settings.Smtp.From));
            mail.ReplyTo = new MailAddress(_emailQueEnvia, _nomeQueEnvia);
            mail.CC.Add(new MailAddress(_emailQueEnvia,_nomeQueEnvia));
            mail.Subject = _assunto;
            mail.Body = _corpo;
            mail.IsBodyHtml = true;

            SmtpClient smtp = new SmtpClient();
            smtp.EnableSsl = true;
            try
            {
                smtp.Send(mail);
                return true;
            }
            catch (Exception exp)
            {
                return false;
            }
        }
        catch (Exception exp)
        {
```

```
            return false;
        }
    }

    public static Boolean enviaEmailComAnexo(string _nomeQueEnvia, string _emailQueRecebe, string _assunto, string _c
    {
        try
        {
            Configuration config = WebConfigurationManager.OpenWebConfiguration(HttpContext.
Current.Request.ApplicationPath);
            MailSettingsSectionGroup settings = (MailSettingsSectionGroup)config.
GetSectionGroup("system.net/mailSettings");
            MailMessage mail = new MailMessage();
            mail.From = new MailAddress(settings.Smtp.From, _nomeQueEnvia);
            mail.To.Add(new MailAddress(_emailQueRecebe));
            mail.Subject = _assunto;
            mail.Body = _corpo;
            mail.IsBodyHtml = true;


            mail.Attachments.Add(new Attachment(_anexos, _nomeAnexos, "application/pdf"));

            SmtpClient smtp = new SmtpClient();
            smtp.EnableSsl = true;
            try{
                smtp.Send(mail);
                return true;
            }catch (Exception exp)
            {
                return false;
            }
        }
        catch (Exception exp)
        {
            return false;
        }
   }
}
```

A.5 Classe PDF

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using iTextSharp.text;
using iTextSharp.text.pdf;
using iTextSharp.text.html.simpleparser;
using System.Web.UI.WebControls;

/// <summary>
/// Summary description for PDF
/// </summary>
public class PDF
{
public PDF()
{

}
    public static MemoryStream criaPDFInscricao(string _nomeGim, string _moradaGim,
string _nifGim, string _contactoGim, string _nifCliente, string _numRecibo, string _numCliente,
        string _logo, string _nomeCliente, string _moradaCliente, ListItemCollection _modAct,
ListItemCollection _precos, string _total, string _data, string _nomeFuncionario)
    {
        string produtos = "";
        string precos = "";

        Document doc = new Document(PageSize.A4, 50, 50, 25, 25);
```

```
        //FileStream fs = new FileStream(HttpContext.Current.Server.MapPath("~/pdfs/" + "Inscricao" + System.DateTime
        MemoryStream ms = new MemoryStream();
        PdfWriter pdf = PdfWriter.GetInstance(doc, ms);
        try
        {
            doc.Open();
            //Adicionar Elementos
            string conteudoHTML = File.ReadAllText(HttpContext.Current.
Server.MapPath("~/templatePDF.html"));
            conteudoHTML = conteudoHTML.Replace("[NomeGim]", _nomeGim);
            conteudoHTML = conteudoHTML.Replace("[MoradaGim]", _moradaGim);
            conteudoHTML = conteudoHTML.Replace("[ContactoGim]", _contactoGim);
            conteudoHTML = conteudoHTML.Replace("[ContribuinteGim]", _nifGim);
            conteudoHTML = conteudoHTML.Replace("[ContribuinteCliente]", _nifCliente);
            conteudoHTML = conteudoHTML.Replace("[NumRecibo]", _numRecibo);
            conteudoHTML = conteudoHTML.Replace("[NumCliente]", _numCliente);
            conteudoHTML = conteudoHTML.Replace("[NomeCliente]", _nomeCliente);
            conteudoHTML = conteudoHTML.Replace("[MoradaCliente]", _moradaCliente);
            conteudoHTML = conteudoHTML.Replace("[ContactoGim]", _contactoGim);
            conteudoHTML = conteudoHTML.Replace("[Designacao]", "Modalidades/Actividades");
            conteudoHTML = conteudoHTML.Replace("[Preco]", "Preço");
            conteudoHTML = conteudoHTML.Replace("[ContactoGim]", _contactoGim);

            foreach (System.Web.UI.WebControls.ListItem li in _modAct)
            {
                produtos += "<p/>" + li;
            }

            foreach (System.Web.UI.WebControls.ListItem li in _precos)
            {
                precos += "<p/>" + li;
            }

            conteudoHTML = conteudoHTML.Replace("[NomeProduto]", produtos);
            conteudoHTML = conteudoHTML.Replace("[PrecoProduto]", precos);
            conteudoHTML = conteudoHTML.Replace("[Total]", _total);
            conteudoHTML = conteudoHTML.Replace("[DataEmissao]", _data);
            conteudoHTML = conteudoHTML.Replace("[NomeFuncionario]", _nomeFuncionario);

            //Converter HTML para Pdf
            var html = HTMLWorker.ParseToList(new StringReader(conteudoHTML), null);
            foreach (var htmlElement in html)
            {
                doc.Add(htmlElement as IElement);
            }

            //Logotipo
            var logo = iTextSharp.text.Image.GetInstance(HttpContext.Current.
Server.MapPath("~/images/" + \_logo));
            logo.SetAbsolutePosition(450f, 700f);
            logo.ScalePercent(24f);
            doc.Add(logo);
            pdf.CloseStream = false;
            doc.Close();
            ms.Position = 0;

            return ms;
        }
        catch (Exception exp)
        {
            return ms;
        }
    }
}
```

A.6 Classe Utilizador (3 Tier Architecture)

```
using System;
using System.Data;
```

```
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

/// <summary>
/// Class Utilizador
/// </summary>
public class Utilizador
{
    private Int32 id;
    private String nome;
    private String sobrenome;
    private String morada;
    private String bi;
    private String nif;
    private String telefone;
    private String email;
    private String numUtilizador;
    private Int32 tipoUtilizador;
    private Int32 delegacao;
    private String postal;
    private String fotografia;
    private DateTime dataNascimento;

    public Utilizador()
{
        this.id = -1;
        this.nome = "";
        this.sobrenome = "";
        this.morada = "";
        this.bi = "";
        this.nif = "";
        this.telefone = "";
        this.email = "";
        this.numUtilizador = "";
        this.tipoUtilizador = -1;
        this.delegacao = -1;
        this.postal = "";
        this.fotografia = "";
        this.dataNascimento = DateTime.Now;
}

    public Utilizador(Int32 _id, String _nome, String _sobrenome, String _morada,
String _bi, String _nif, String _telefone, String _email, String _numUtilizador,
Int32 _tipoUtilizador, Int32 _delegacao, String _postal,
        String _fotografia, DateTime _dataNascimento)
    {
        this.id = _id;
        this.nome = _nome;
        this.sobrenome = _sobrenome;
        this.morada = _morada;
        this.bi = _bi;
        this.nif = _nif;
        this.telefone = _telefone;
        this.email = _email;
        this.numUtilizador = _numUtilizador;
        this.tipoUtilizador = _tipoUtilizador;
        this.delegacao = _delegacao;
        this.postal = _postal;
        this.fotografia = _fotografia;
        this.dataNascimento = _dataNascimento;
    }


    public Int32 Id
```

```
{
    get { return id; }
    set { id = value; }
}

public String Nome
{
    get { return nome; }
    set { nome = value; }
}

public String Sobrenome
{
    get { return sobrenome; }
    set { sobrenome = value; }
}

public String Morada
{
    get { return morada; }
    set { morada = value; }
}

public String Bi
{
    get { return bi; }
    set { bi = value; }
}

public String Nif
{
    get { return nif; }
    set { nif = value; }
}

public String Telefone
{
    get { return telefone; }
    set { telefone = value; }
}

public String Email
{
    get { return email; }
    set { email = value; }
}

public String NumUtilizador
{
    get { return numUtilizador; }
    set { numUtilizador = value; }
}

public Int32 TipoUtilizador
{
    get { return tipoUtilizador; }
    set { tipoUtilizador = value; }
}

public Int32 Delegacao
{
    get { return delegacao; }
    set { delegacao = value; }
}

public String Postal
{
    get { return postal; }
    set { postal = value; }
}
```

```
    public String Fotografia
    {
        get { return fotografia; }
        set { fotografia = value; }
    }

    public DateTime DataNascimento
    {
        get { return dataNascimento; }
        set { dataNascimento = value; }
    }
}
```

A.7 Classe UtilizadorDAO (3 Tier Architecture)

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

/// <summary>
/// Summary description for UtilizadoresDAO
/// </summary>
public class UtilizadoresDAO
{
    private Database db;
    private Utilizador ut;

public UtilizadoresDAO()
{
        Database db = new Database();
        ut = new Utilizador();
}

    public int inserirUtilizador(Utilizador _ut)
    {
        try
        {
            SqlParameter[] sqlParameters = new SqlParameter[14];

            sqlParameters[0] = new SqlParameter("@num", SqlDbType.NVarChar);
            sqlParameters[0].Value = Convert.ToString(_ut.NumUtilizador);
            sqlParameters[1] = new SqlParameter("@nome", SqlDbType.NVarChar);
            sqlParameters[1].Value = Convert.ToString(_ut.Nome);
            sqlParameters[2] = new SqlParameter("@sobrenome", SqlDbType.NVarChar);
            sqlParameters[2].Value = Convert.ToString(_ut.Sobrenome);
            sqlParameters[3] = new SqlParameter("@bi", SqlDbType.NChar);
            sqlParameters[3].Value = Convert.ToString(_ut.Bi);
            sqlParameters[4] = new SqlParameter("@nif", SqlDbType.NChar);
            sqlParameters[4].Value = Convert.ToString(_ut.Nif);
            sqlParameters[5] = new SqlParameter("@email", SqlDbType.NVarChar);
            sqlParameters[5].Value = Convert.ToString(_ut.Email);
            sqlParameters[6] = new SqlParameter("@morada", SqlDbType.NVarChar);
            sqlParameters[6].Value = Convert.ToString(_ut.Morada);
            sqlParameters[7] = new SqlParameter("@telefone", SqlDbType.NChar);
            sqlParameters[7].Value = Convert.ToString(_ut.Telefone);
            sqlParameters[8] = new SqlParameter("@foto", SqlDbType.NVarChar);
            sqlParameters[8].Value = Convert.ToString(_ut.Fotografia);
            sqlParameters[9] = new SqlParameter("@delegacao", SqlDbType.Int);
            sqlParameters[9].Value = Convert.ToInt32(_ut.Delegacao);
            sqlParameters[10] = new SqlParameter("@data_insc", SqlDbType.Date);
            sqlParameters[10].Value = Convert.ToDateTime(DateTime.Now);
            sqlParameters[11] = new SqlParameter("@data_nascimento", SqlDbType.DateTime);
            sqlParameters[11].Value = Convert.ToDateTime(_ut.DataNascimento);
            sqlParameters[12] = new SqlParameter("@postal", SqlDbType.NChar);
```

```
            sqlParameters[12].Value = Convert.ToString(_ut.Postal);
            sqlParameters[13] = new SqlParameter("@existe", SqlDbType.Int);
            sqlParameters[13].Direction = ParameterDirection.Output;

            db = new Database();

            int valor = db.executeProc("InserirUtilizador", sqlParameters);

            if (valor == 0)
            {
                return -1;
            }
            else
            {
                return valor;
            }
        }
        catch (Exception exp)
        {
            return -1;
        }
    }

    public int inserirUtilizadorAdmin(Utilizador _ut)
    {
        try
        {
            SqlParameter[] sqlParameters = new SqlParameter[15];

            sqlParameters[0] = new SqlParameter("@num", SqlDbType.NVarChar);
            sqlParameters[0].Value = Convert.ToString(_ut.NumUtilizador);
            sqlParameters[1] = new SqlParameter("@nome", SqlDbType.NVarChar);
            sqlParameters[1].Value = Convert.ToString(_ut.Nome);
            sqlParameters[2] = new SqlParameter("@sobrenome", SqlDbType.NVarChar);
            sqlParameters[2].Value = Convert.ToString(_ut.Sobrenome);
            sqlParameters[3] = new SqlParameter("@bi", SqlDbType.NChar);
            sqlParameters[3].Value = Convert.ToString(_ut.Bi);
            sqlParameters[4] = new SqlParameter("@nif", SqlDbType.NChar);
            sqlParameters[4].Value = Convert.ToString(_ut.Nif);
            sqlParameters[5] = new SqlParameter("@email", SqlDbType.NVarChar);
            sqlParameters[5].Value = Convert.ToString(_ut.Email);
            sqlParameters[6] = new SqlParameter("@morada", SqlDbType.NVarChar);
            sqlParameters[6].Value = Convert.ToString(_ut.Morada);
            sqlParameters[7] = new SqlParameter("@telefone", SqlDbType.NChar);
            sqlParameters[7].Value = Convert.ToString(_ut.Telefone);
            sqlParameters[8] = new SqlParameter("@foto", SqlDbType.NVarChar);
            sqlParameters[8].Value = Convert.ToString(_ut.Fotografia);
            sqlParameters[9] = new SqlParameter("@delegacao", SqlDbType.Int);
            sqlParameters[9].Value = Convert.ToInt32(_ut.Delegacao);
            sqlParameters[10] = new SqlParameter("@data_insc", SqlDbType.Date);
            sqlParameters[10].Value = Convert.ToDateTime(DateTime.Now);
            sqlParameters[11] = new SqlParameter("@data_nascimento", SqlDbType.DateTime);
            sqlParameters[11].Value = Convert.ToDateTime(_ut.DataNascimento);
            sqlParameters[12] = new SqlParameter("@postal", SqlDbType.NChar);
            sqlParameters[12].Value = Convert.ToString(_ut.Postal);
            sqlParameters[13] = new SqlParameter("@existe", SqlDbType.Int);
            sqlParameters[13].Direction = ParameterDirection.Output;
            sqlParameters[14] = new SqlParameter("@tipo", SqlDbType.Int);
            sqlParameters[14].Value = Convert.ToInt32(_ut.TipoUtilizador);

            db = new Database();

            int valor = db.executeProc("InserirUtilizadoresAdmin", sqlParameters);

            if (valor == 0)
            {
                return -1;
            }
            else
            {
```

```
                return valor;
            }
        }
        catch (Exception exp)
        {
            return -1;
        }
    }

}
```

A.8 Classe UtilizadorBUS (3 Tier Architecture)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

/// <summary>
/// Summary description for UtilizadorBUS
/// </summary>
public class UtilizadorBUS
{
    private UtilizadoresDAO ut;
public UtilizadorBUS()
{
        ut = new UtilizadoresDAO();
}

    public int NovoUtilizador(Utilizador _ut)
    {
        int valorRetorno;

        valorRetorno = ut.inserirUtilizador(_ut);
        if (valorRetorno == -1)
        {
            return -1;
        }
        else
        {
            return valorRetorno;
        }
    }
    public int NovoUtilizadorAdm(Utilizador _ut)
    {
        int valorRetorno;

        valorRetorno = ut.inserirUtilizadorAdmin(_ut);
        if (valorRetorno == -1)
        {
            return -1;
        }
        else
        {
            return valorRetorno;
        }
    }
}
```

A.9 Procedimento Aniversários

```
CREATE procedure [dbo].[aniversarios](
@username nvarchar(50),
@existe int output
)

as
```

```
declare @count int;

select @count=count(*) from utilizadores, UTILIZADORES_aspnet, aspnet_Users
where UTILIZADORES_aspnet.useraspnet_id = aspnet_Users.UserId and aspnet_Users.UserName = @username
and datepart (day, SYSDATETIME()) = datepart(day, UTILIZADORES.DATA_NASCIMENTO)
and datepart (month, SYSDATETIME()) = datepart(month, UTILIZADORES.DATA_NASCIMENTO);

if @count=0
set @existe =  0
else
set @existe = 1
```

A.10 Procedimento Inserir Escalão

```
CREATE procedure [dbo].[inserirEscaloes](
@primario bit,
@cinturao bit,
@id_mod int,
@nome nvarchar(50)
)
as

Declare @id_anterior int;

if @primario = 1
begin
if @cinturao = 1
begin
insert into escaloes (CINTURAO, PRIMARIO, NOME_ESCALAO) values (@cinturao, @primario, @nome);
set @id_anterior = CAST(SCOPE_IDENTITY() AS INT)
update ESCALOES set HIERARQUIA=@id_anterior where ID_ESCALAO=@id_anterior;
insert into ESCALOES_MOD_ACT (ESCALOES_ID_ESCALAO, MODALIDADES_ID_MODALIDADE) values (@id_anterior, @id_mod);
end
else
begin
insert into escaloes (CINTURAO, PRIMARIO, NOME_ESCALAO) values (@cinturao, @primario, @nome);
set @id_anterior = CAST(SCOPE_IDENTITY() AS INT)
insert into ESCALOES_MOD_ACT (ESCALOES_ID_ESCALAO, MODALIDADES_ID_MODALIDADE) values (@id_anterior, @id_mod);
end
end
else
begin
if @cinturao = 1
begin
insert into escaloes (CINTURAO, PRIMARIO, NOME_ESCALAO) values (@cinturao, @primario, @nome);
set @id_anterior = CAST(SCOPE_IDENTITY() AS INT)
update ESCALOES set HIERARQUIA=@id_anterior-1 where ID_ESCALAO=@id_anterior;
insert into ESCALOES_MOD_ACT (ESCALOES_ID_ESCALAO, MODALIDADES_ID_MODALIDADE) values (@id_anterior, @id_mod);
end
else
begin
insert into escaloes (CINTURAO, PRIMARIO, NOME_ESCALAO) values (@cinturao, @primario, @nome);
set @id_anterior = CAST(SCOPE_IDENTITY() AS INT)
insert into ESCALOES_MOD_ACT (ESCALOES_ID_ESCALAO, MODALIDADES_ID_MODALIDADE) values (@id_anterior, @id_mod);
end
end
```

A.11 Procedimento Inserir Linha de Factura

```
CREATE procedure [dbo].[InserirLinhasFacturaMOD](
@mes int,
@ano int,
@mensalidade int,
@preco float,
```

```
@anualidade int,
@factura int,
@data datetime,
@pago bit,
@id_modalidade int,
@seguro int
)
AS
declare @id_mensalidade int;
declare c_meses cursor for
select MES_REFERENTE from MENSALIDADES where MES_REFERENTE >= @mes and ANO_REFERENTE = @ano
and MENSALIDADE_VIGOR = 1 and ID_MODALIDADE = @id_modalidade;

declare @mes_c int;

open c_meses

fetch next from c_meses into @mes_c;

while @@FETCH_STATUS = 0
begin
if @mes_c = @mes
begin
insert into LINHAS_FACTURAS(DATA_PAGAMENTO, MENSALIDADES_ID_MENSALIDADE, FACTURAS_ID_FACTURA, PAGO, PRECO_LINHA)
values
(@data, @mensalidade, @factura, @pago, @preco);
fetch next from c_meses into @mes_c;
end
else
begin
select @id_mensalidade=ID_MENSALIDADE from MENSALIDADES where MENSALIDADE_VIGOR = 1 and
MENSALIDADES.MES_REFERENTE = @mes_c;
insert into LINHAS_FACTURAS(DATA_PAGAMENTO, MENSALIDADES_ID_MENSALIDADE,FACTURAS_ID_FACTURA, PAGO, PRECO_LINHA)
values
(@data, @id_mensalidade,@factura, 0, @preco);
fetch next from c_meses into @mes_c;
end
end
close c_meses
deallocate c_meses

if @seguro <> -1
begin
insert into LINHAS_FACTURAS(DATA_PAGAMENTO, SEGUROS_ID_SEGURO, PRECO_LINHA, PAGO, FACTURAS_ID_FACTURA)
values (@data, @seguro, @preco, @pago, @factura)
end

if @anualidade <> -1
begin
insert into LINHAS_FACTURAS(DATA_PAGAMENTO, ANUALIDADES_ID_ANUALIDADE, PRECO_LINHA, PAGO, FACTURAS_ID_FACTURA)
values (@data, @anualidade, @preco, @pago, @factura)
end
```

Anexo B

Manual da Aplicação

Resumo

Este documento serve de apoio à configuração e utilização da aplicação. Serão apresentados os principais passos para uma configuração e utilização correta da aplicação, de modo a que a experiência de utilização seja o mais agradável possível. A aplicação para funcionar necessita de um servidor Web da Microsoft com sistema operativo Windows com acesso a Base de Dados Sql Server. O espaço em servidor terá de ser no mínimo 200 MB de modo a hospedar a aplicação com uma margem para uploads de fotografias.

Palavras Chave

Aplicação Web, Sql Server, Microsoft, Sistema Operativo, Configuração, Utilização

Abstract

This document serves to support the configuration and application usage. Will present the key steps to proper configuration and use of the application, so that the user experience is as pleasant as possible.
The application needs to run a web server with Miscrosoft Windows operating system with access to Database Sql Server. The server space will be at least 200 MB to host the application with a margin for photo uploads.

Key words

Web Application, Sql Server, Microsoft, Operating System, Configuration, Use

Conteúdo

1 Requisitos Mínimos da Aplicação **8**

2 Configuração e Utilização **9**
2.1 Primeiro Login e Criação de Utilizadores 9
2.2 Gestão de Atividades e Modalidades 16
2.3 Gestão de Seguros 19
2.4 Gestão Anuidade e Mensalidades 19
2.5 Gestão de Categorias de Notícias 22
2.6 Gestão de Requisitos 23

3 Área de Administrador **25**

4 Área de Funcionário **29**

5 Área de Cliente **36**

6 Como inserir vídeos do Youtube **38**

Lista de Figuras

2.1 Login . . . 10
2.2 Menu . . . 10
2.3 Gestão de Tipos de Utilizadores . . . 10
2.4 Gestão de Delegações . . . 11
2.5 Menu respetivo ao Super Admin . . . 11
2.6 Página de Gestão de Utilizadores . . . 12
2.7 Alteração da password por defeito do utilizador root . . . 13
2.8 Erro na Validação do número de Bilhete de Identidade . . . 13
2.9 Sinalização de campos obrigatórios em caso de inserção ou edição de dados . . . 15
2.10 Criar nova Modalidade . . . 16
2.11 Página de Gestão Modalidade . . . 17
2.12 Inserir escalões . . . 18
2.13 Página de Gestão Seguros . . . 19
2.14 Página de Gestão Anuidades . . . 20
2.15 Página de Gestão Modalidades . . . 21
2.16 Lançamento de Anuidades e Mensalidades . . . 22
2.17 Esquema da sub-divisão da página principal . . . 23
2.18 Modalidade com requisito de escalão . . . 24
2.19 Actividade com requisito de idade . . . 24

3.1 Exemplo de um gráfico obtido na página . . . 26
3.2 Exemplo de Gráfico de comparação de dados . . . 26
3.3 Exemplo de Lista de Clientes . . . 27
3.4 Exemplo da página Consultar dados de Actividades . . . 27
3.5 Exemplo da página Consultar dados de Modalidades . . . 28

4.1 Criar Notícia . . . 30
4.2 Inscrição em Atividades . . . 31
4.3 Inscrição em Modalidades . . . 32
4.4 Subida de Escalão . . . 33
4.5 Marcar Presenças . . . 34
4.6 Renovação de Inscrição . . . 35

5.1 Alteração dos dados do cliente . . . 36
5.2 Página Como Chegar? . . . 37

6.1 Youtube Ícone . . . 38

6.2 HyperLink Youtube 38
6.3 HyperLink Youtube 39

Lista de Tabelas

2.1 Hierarquia de Utilizadores (do mais alto para o mais baixo) 9
2.2 Dados para primeiro Login 9
2.3 Máscaras aceites pelos campos mencionados 14
2.4 Tabela de Mensalidades exemplo 20

Glossário

ASP.NET — Plataforma de desenvolvimento de aplicações Web dinâmicas da Miscrosoft.

Base de Dados — Conjunto de dados estruturados e relacionados entre si.

Framework .NET — É um modelo de programação abrangente da Miscrosoft, que visa uma plataforma única para desenvolvimento e execução de sistemas. A finalidade é ter todo e qualquer código gerado para .NET poder ser executado em qualquer dispositivo com a Framework da plataforma.

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

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 (imagens, vídeos, documentos).

Sql Server — É um SGBD (Sistema de Gestão de Base de Dados) criado pela Microsoft e Sysbase que permite interação com os dados através da linguagem SQL.

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

Capítulo 1

Requisitos Mínimos da Aplicação

- Servidor Web com suporte Framework .NET 4 ou superior

- Espaço de Alojamento superior ou igual a 200 MB

- Acesso a Base de Dados Sql Server 2008 ou superior

Capítulo 2

Configuração e Utilização

2.1 Primeiro Login e Criação de Utilizadores

Começando pela configuração da página, temos o primeiro login com o super utilizador. O login é muito importante pois limita a área a que o utilizador tem acesso. Ou seja existem utilizadores que não têm acesso a conteúdos que outros utilizadores têm. Assim os privilégios dos utilizadores estão organizados da seguinte forma (Tabela 2.1), ordenado do escalão com mais privilégios para os utilizadores com menos privilégios.

Privilégios	**Grupo**
Super Administrador	Super Admin
Administrador	Admin
Funcionários	Funcionario
Clientes	Users

Tabela 2.1: Hierarquia de Utilizadores (do mais alto para o mais baixo)

Como se pode observar o grupo Super Admin é o grupo que contém os utilizadores da aplicação com mais privilégios. Sendo assim o primeiro login e o utilizador por omissão é desse grupo e contém os seguintes dados para login: (Tabela 2.2)

Utilizador	**Password**
root	root123456

Tabela 2.2: Dados para primeiro Login

Fazendo o primeiro login. Figura 2.1.

O seguinte menu irá aparecer. Figura 2.2.

Antes de se criarem novos utilizadores convém criar vários tipos de utilizadores. Isto permite 'catalogar' de certa forma os utilizadores e é essencial para quando forem inscritos novos clientes, o sistema saber em que categoria é que o cliente vai ser inserido. Para bom funcionamento é necessário ter um grupo por defeito que é o grupo onde os clientes automaticamente são inseridos. Não confundir tipo de utilizador com privilégios de login. A gestão dos tipos de utilizadores é feito na

Figura 2.1: Login

Figura 2.2: Menu

página **Gestão de Tipos de Utilizadores**. Pode-se editar, eliminar, inserir novos tipos de utilizador. Figura 2.3.

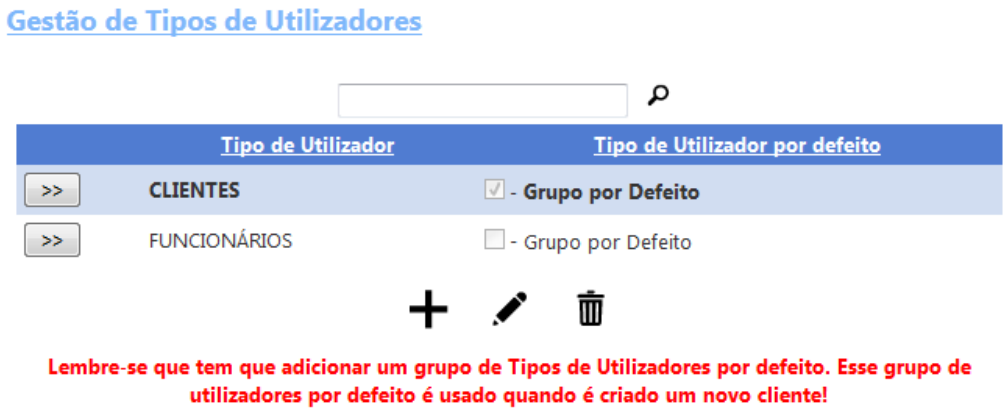


Figura 2.3: Gestão de Tipos de Utilizadores

Antes ainda de criar o novo Utilizador é essencial criar as delegações. As delegações são a instituição onde os utilizadores pertencem ou seja imaginemos que a instituição tem uma delegação na Guarda e outra em Celorico da Beira, os clientes, funcionários, etc. devem ser inseridos na delegação de Celorico da Beira, isto para que o gestor da aplicação e o funcionário tenham uma maior organização dos mesmos. A página que permite criar as delegações é a página **Gestão de Delegações**. Figura 2.4.

Agora sim já podemos criar novos utilizadores do sistema, na área do Super Administrador -> **Gestão de Utilizadores** 2.5

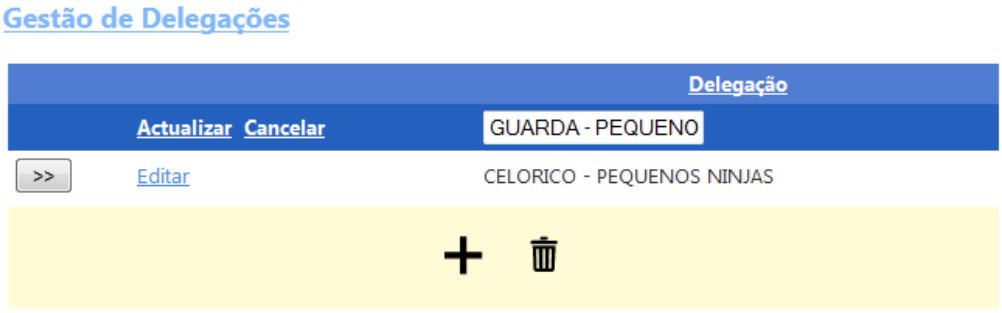


Figura 2.4: Gestão de Delegações

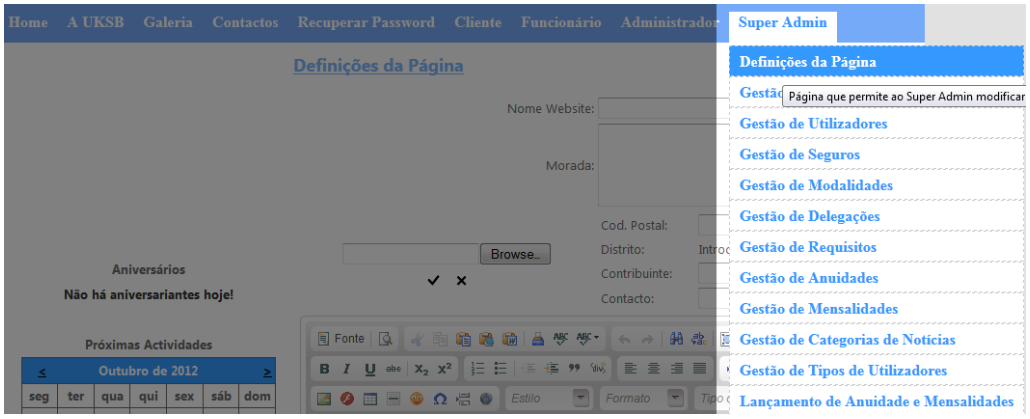


Figura 2.5: Menu respetivo ao Super Admin

Recomenda-se a criação de um novo utilizador com os mesmos privilégios que o utilizador por omissão ou seja, o utilizador é associado a todos os grupos existentes.

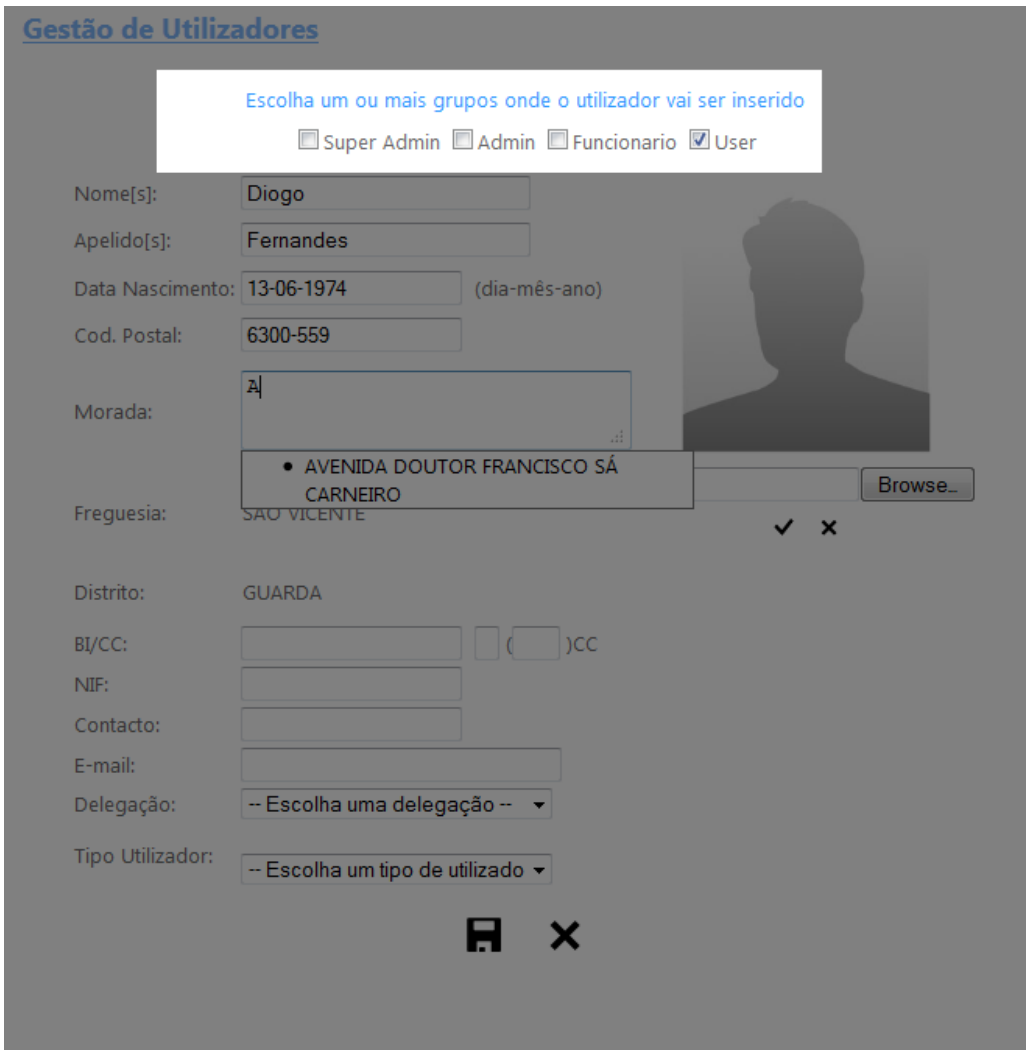


Figura 2.6: Página de Gestão de Utilizadores

Como se pode observar na figura 2.6 no topo da página temos as opções de grupos onde o utilizador deve ser inserido.
Tenha em atenção que se esquecer da password ou username deste utilizador perderá acesso a esta área da aplicação!
Outro ponto a ter em consideração é que após a inserção do utilizador é enviado uma mensagem para o email que introduziu. Essa mensagem contém um código de ativação, assim **só poderá efetuar o login com o novo utilizador após ativar a conta**.
Assim sempre que quiser criar um novo utilizador, por exemplo um funcionário novo ou um Administrador é a Página Gestão de Utilizadores que deve usar.
Sendo esta página de grande importância, a página permite ainda alterar dados dos utilizadores, como por exemplo tornar o utilizador inativo. Nesse caso o utilizador deixa de poder efetuar login com a sua conta. Outra funcionalidade associada a esta

página é a funcionalidade 'Reenviar Código de Ativação' na edição do utilizador e deverá ser usada em caso de perda dos dados de acesso por parte de um utilizador ou até mesmo o código de ativação. O utilizador quando receber o email deve seguir as instruções acompanhadas do mesmo.

Após o primeiro login e da criação de um novo utilizador com os mesmos privilégios que o utilizador root, por uma questão de segurança recomenda-se que que altere de imediato a password de acesso (root123456) para uma password à sua escolha. O utilizador root deve ser apenas usado como 'suplente' do outro utilizador criado com os mesmos privilégios. Figura 2.7.

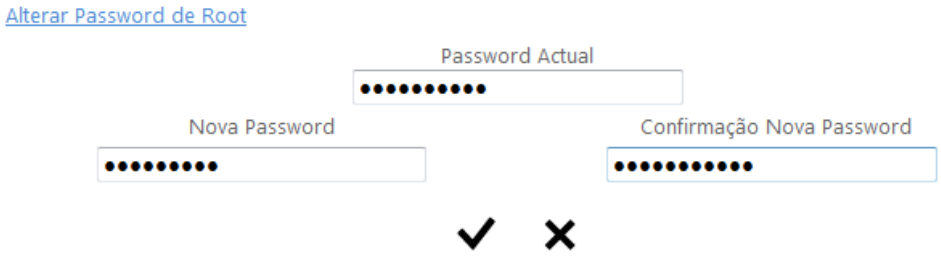


Figura 2.7: Alteração da password por defeito do utilizador root

Na criação ou edição dos dados do utilizador, a página exige a introdução de o número de identificação fiscal e cartão de cidadão/bilhete de identidade assim como o contacto, mas nos dois primeiros casos em particular e porque são de maior importância o sistema faz uma validação do número de identificação fiscal e do bilhete de identidade/cartão de cidadão introduzido. Figura 2.8.

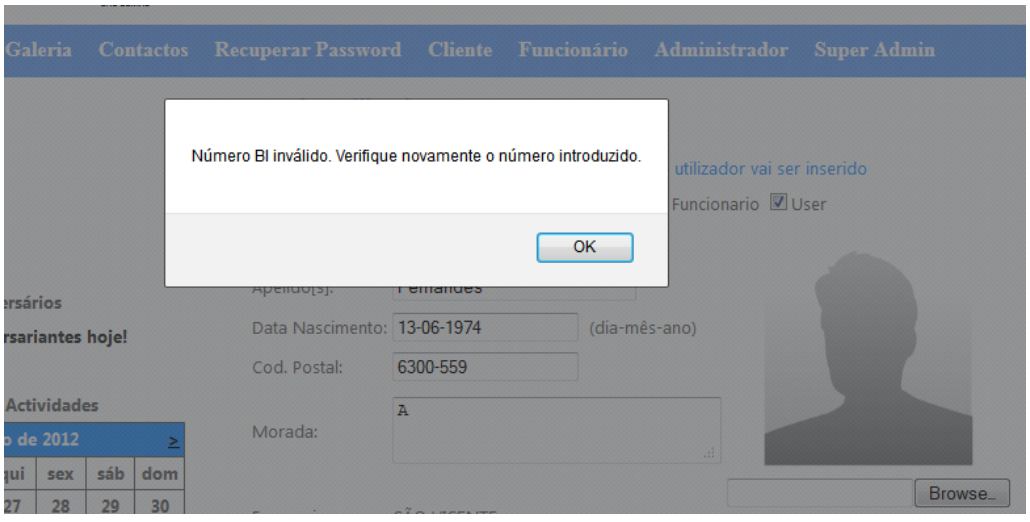


Figura 2.8: Erro na Validação do número de Bilhete de Identidade

Caso um destes números não seja aceite por qualquer motivo ou porque o utilizador a ser criado não tem essa informação disponível ou não queira facultar, são aceites as seguintes máscaras:

Campo	**Máscara**
BI	00000000 0
NIF	000000000
Contacto	222222222 ou 999999999

Tabela 2.3: Máscaras aceites pelos campos mencionados

Esta página com muitas outras têm campos obrigatórios e esses campos são sinalizados em caso de tentativa de Inserção ou Edição de dados da seguinte forma (Figura 2.9)

Figura 2.9: Sinalização de campos obrigatórios em caso de inserção ou edição de dados

2.2 Gestão de Atividades e Modalidades

Em primeiro lugar criam-se as modalidades, isto porque a página de gestão das atividades tem um campo opcional que pede a modalidade associada à atividade. Isto é as atividades podem estar relacionadas com modalidades, por exemplo, uma possível atividade seria um exame de uma modalidade, aí teria que se incluir qual a modalidade associada ao exame.

Vamos começar então por criar as modalidades, esta operação é feita na página **Gestão de Modalidades**. Figura 2.10.

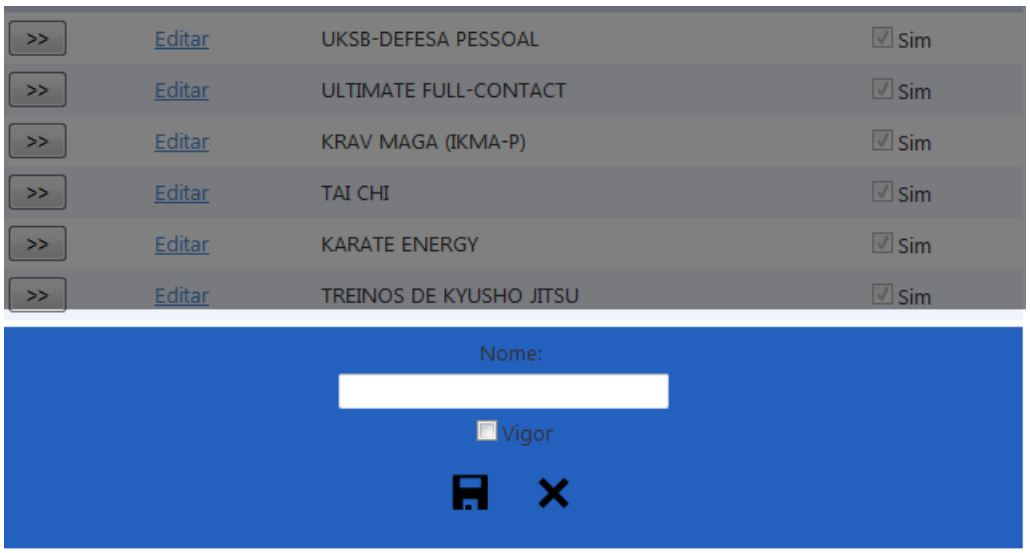


Figura 2.10: Criar nova Modalidade

Na generalidade das páginas de gestão é onde o utilizador pode editar, eliminar dados, aqui não é excepção. Figura 2.11.

Na página **Gestão de Tipos de Actividades** é onde o Super Administrador pode inserir, editar e remover Tipos de Actividades. Por tipos de Actividades entende-se a categoria a que a atividade irá pertencer. Tipos de atividades, seriam por exemplo, Jantar, Workshop, Exame, etc. Permite categorizar de alguma forma as actividades.

De seguida, já podemos criar novas atividades. Isso pode ser feito na página **Gestão de Atividades**. Nesta página temos vários campos importantes para o funcionamento da aplicação.

Data Início da Atividade— Como o nome indica é o primeiro ou único dia que tem inicio a atividade.

Data Limite de Inscrição— Esta data permite definir um limite até quando os clientes podem ser inscritos na atividade.

Código Postal— Este Campo é obrigatório uma vez que a atividade tem lugar algures, ao inserir o código postal automaticamente o distrito, concelho e freguesia são preenchidos além de, com auxilio dos Mapas da Sapo, ser automaticamente preenchidas as coordenadas (longitude e latitude) do evento.

Figura 2.11: Página de Gestão Modalidade

Isto permitirá o funcionamento de uma funcionalidade da aplicação que será explicada mais à frente. Existem situações em que as coordenadas não são devolvidas, nesse caso não se preocupe não é alarmante, pois não são obrigatórias.

Por fim resta-nos falar sobre a página dos escalões de modalidades. Nesta página podemos inserir, editar, remover níveis de modalidades. Esta página permite criar dois tipos de níveis distintos. Os níveis cuja modalidade tem cinturão e os escalões sem cinturão. O que são escalões com cinturão? Normalmente, nestes casos para se passar para o próximo cinturão é necessário possuir o cinturão anterior. Desse modo tem que se organizar os níveis hierarquicamente. Como se pode observar na Figura 2.19, os escalões que estariam a ser criados seriam da modalidade Karaté Shotokan sendo o 'Nível 1' o primeiro Nível. A página permite ordenar os níveis ao nosso gosto, sendo estes organizados do primeiro para o último sempre. No exemplo da figura e uma vez que estamos a falar de um escalão cuja modalidade é por cinturão, os níveis estariam organizados da seguinte forma.
O Nível 1 seria o primeiro nível quer isto dizer que um praticante se for a primeira vez que se inscreve na modalidade Karaté Shotokan iria ser sugerida a inscrição nesse nível.
O nível 2 seria o filho do nível 1 quer isto dizer que o sistema irá sugerir a inscrição ou a subida de escalão na modalidade referida se o cliente possuir o nível 1. Deste modo temos os escalões organizados corretamente e podemos inserir quantos níveis quisermos pois o sistema saberá qual será o próximo escalão do utilizador.

Os escalões que não tem cinturão são escalões que não têm interligação entre eles. Por exemplo escalões de peso cujo utilizador sobre de nível consoante o seu peso atual.
Tomemos o seguinte exemplo, +50kg, +80kg. São dois níveis diferentes mas que não

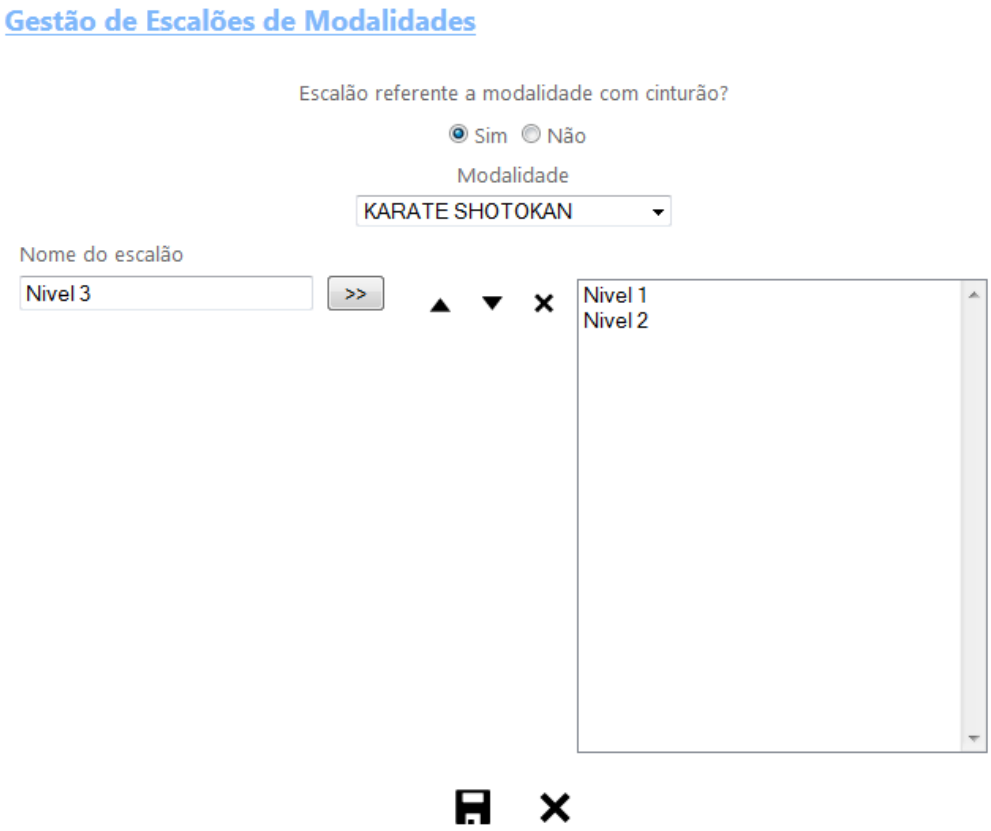


Figura 2.12: Inserir escalões

estão ligados entre si. O cliente quando é inscrito numa modalidade que tem estes escalões, irá ser inscrito no escalão do seu peso atual. Imaginemos que inicialmente o cliente tinha 78kg, quer isto dizer que se encontrava no escalão +50kg, passado algum tempo o cliente apresentou-se com 85kg, ou seja o cliente subi de nível para o escalão +80kg sem que tenha uma condição obrigatória do escalão anterior e até poderá saltar vários níveis de uma só vez. O que não é, na maioria dos casos possível em tipos de escalões que se regem por cinturão.
Deixamos aqui ainda uma nota, apresar de ser possível editar e eliminar escalões, o mesmo só deve ser feito se não estragar a hierarquia que é criada no caso dos escalões com cinturão uma vez que o sistema poderá deixar de gerir eficientemente em que escalão o utilizador se encontra e quais os escalões seguintes na subida de níveis.

2.3 Gestão de Seguros

Antes de se puder iniciar com a inscrição de clientes, é importante criar um seguro ou ter um seguro ativo, para que as inscrições de clientes funcionem corretamente. Aqui mais uma vez como é uma página de gestão é possível editar, eliminar, criar Seguros. A página que permite realizar as tarefas descritas é a página **Gestão de Seguros**. Figura 2.13.

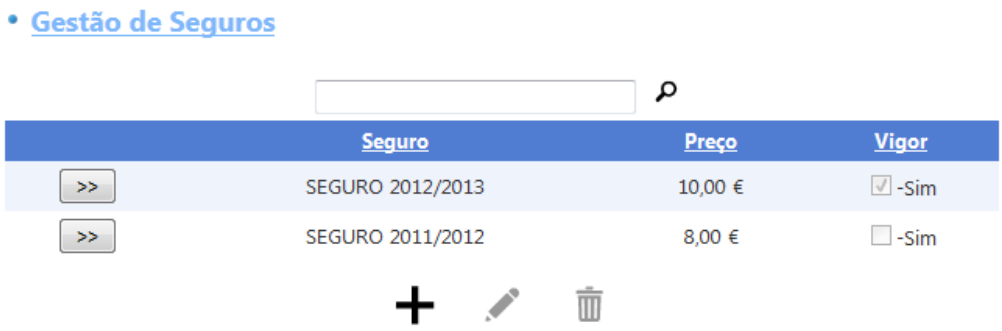


Figura 2.13: Página de Gestão Seguros

2.4 Gestão Anuidade e Mensalidades

A página **Lançamento de Anuidade e Mensalidades** (Figura 2.16) é sem dúvida a página mais importante de todas até aqui já apresentadas. Isto porque é com base nesta página que a aplicação irá funcionar. Esta página permite fazer o lançamento da anuidade que tem dois campos muito importantes além do preço da anuidade que são a **data de início e fim da Anuidade**.
Com base nestas datas é que o sistema sabe se deixa inscrever clientes ou não, mas esta página também serve para gerar as mensalidades respetivas às modalidades previamente inseridas. Isto é se a data de início escolhida for 01/10/2012 e a data de fim for 30/1/2013. As inscrições de novos clientes ou inscrições em modalidades só

podem ser feitas entre essas datas e as mensalidades geradas para cada modalidade ativa, seria por exemplo as apresentadas na tabela 2.4.

Mensalidades
Outubro
Novembro
Dezembro
Janeiro

Tabela 2.4: Tabela de Mensalidades exemplo

Nesta página só é possível fazer o lançamento das anuidades e mensalidades, caso o preço de uma mensalidade tenha que ser alterada, a página **Gestão de Mensalidades** (Figura 2.15) permite que tal seja feito. Há que ter aqui em atenção um aspeto muito **importante** que é o seguinte, se por algum motivo alguma mensalidade é eliminada é provável que venha a causar problemas nos pagamentos ou nas inscrições. A mensalidade só deve ser eliminada caso a modalidade dessa mensalidade por exemplo já não esteja em uso. O mesmo há que ter em atenção na página web **Gestão de Anuidades** (Figura 2.14).

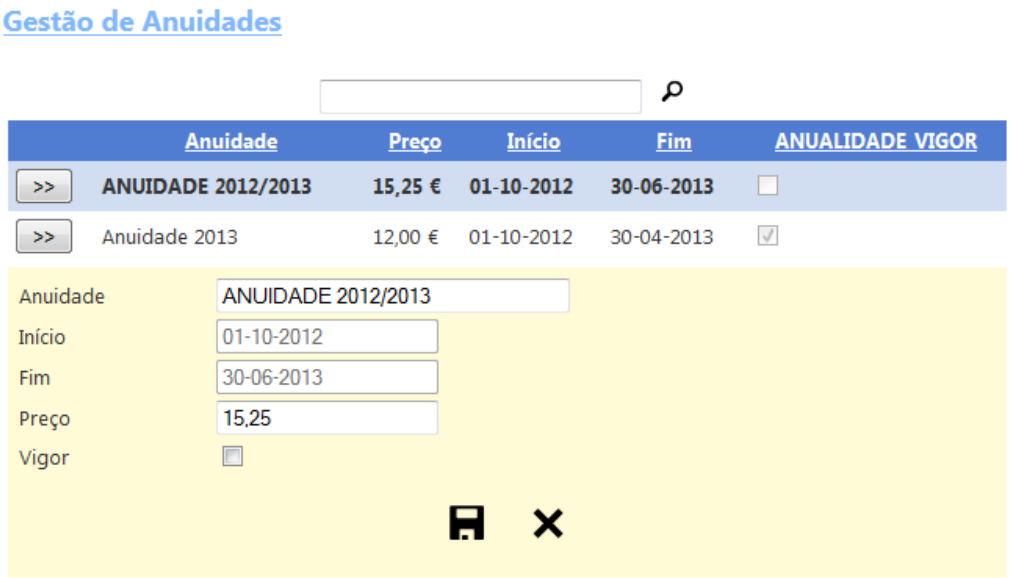


Figura 2.14: Página de Gestão Anuidades

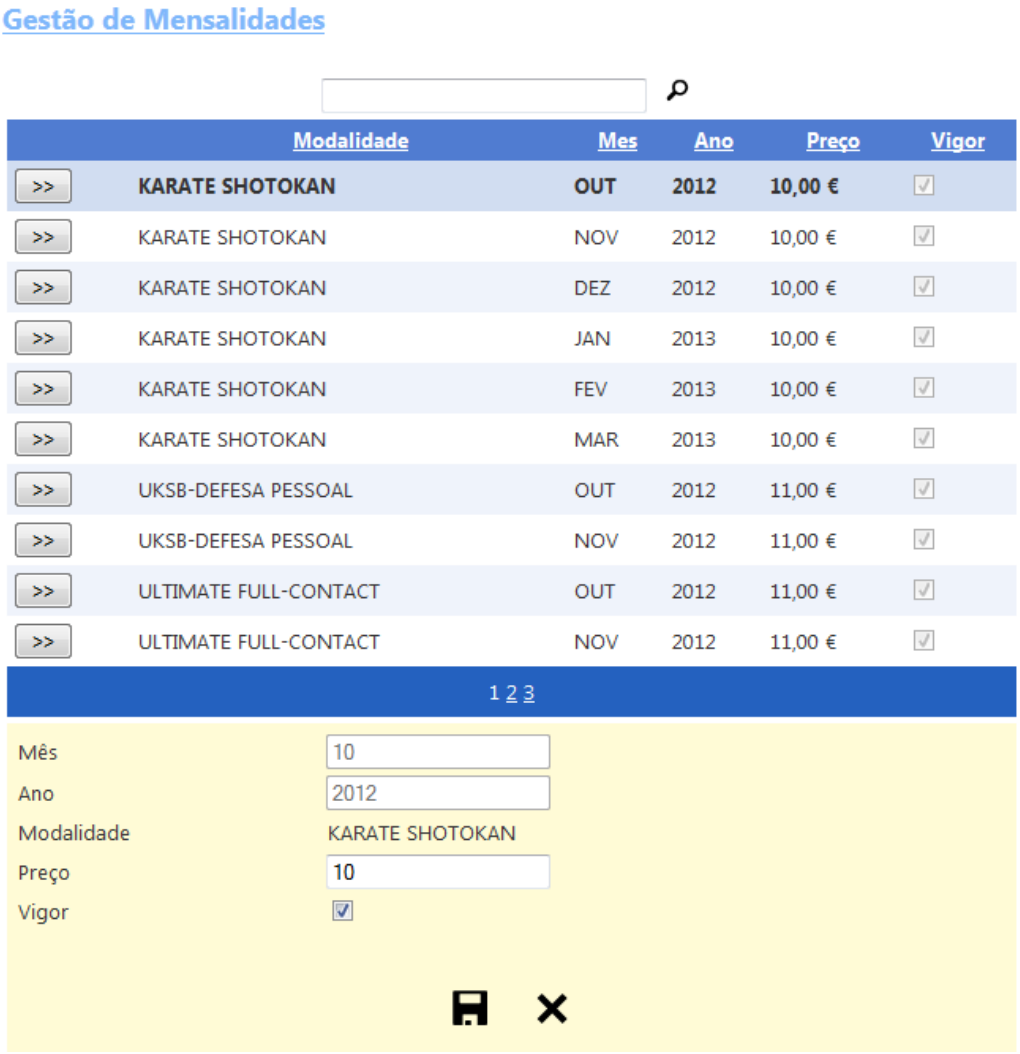


Figura 2.15: Página de Gestão Modalidades

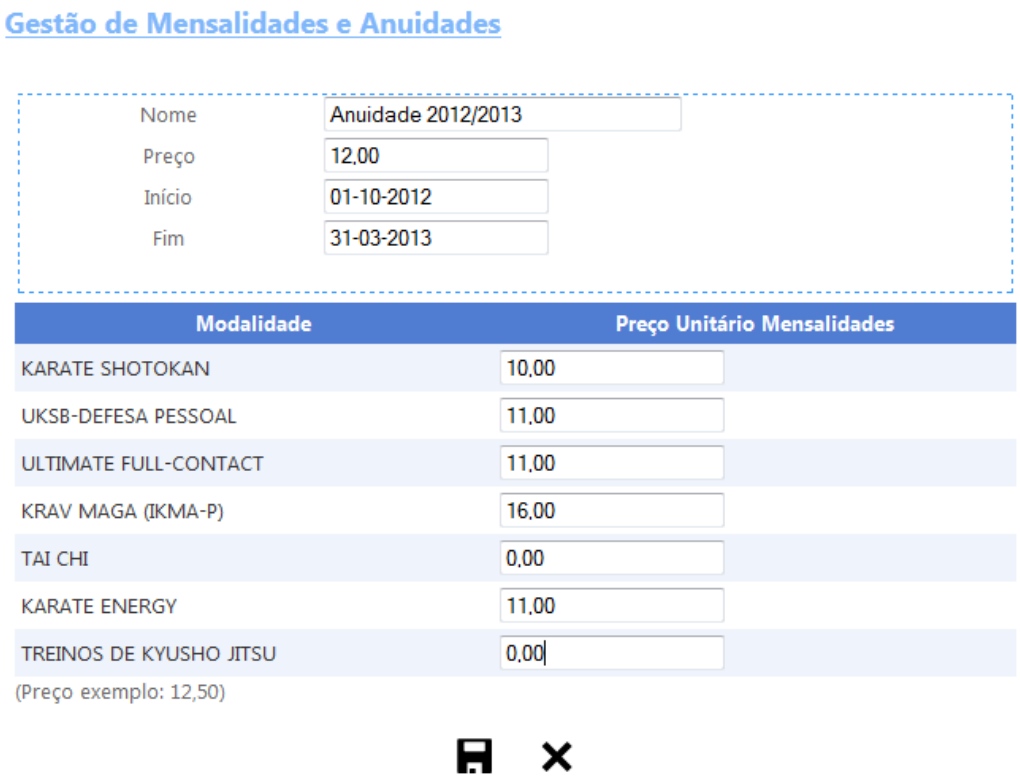


Figura 2.16: Lançamento de Anuidades e Mensalidades

2.5 Gestão de Categorias de Notícias

Esta é a página onde é permitido criar as categorias onde as Notícias vão ser inseridas. Isto é a página principal tem 4 áreas. Quando são criadas as categorias podemos definir a zona onde as notícias vão aparecer na página principal. E quando as notícias são criadas escolhe-se a categoria a que pertence a notícia. As zonas estão distribuídas pela página principal como se pode observar na Figura 2.17.

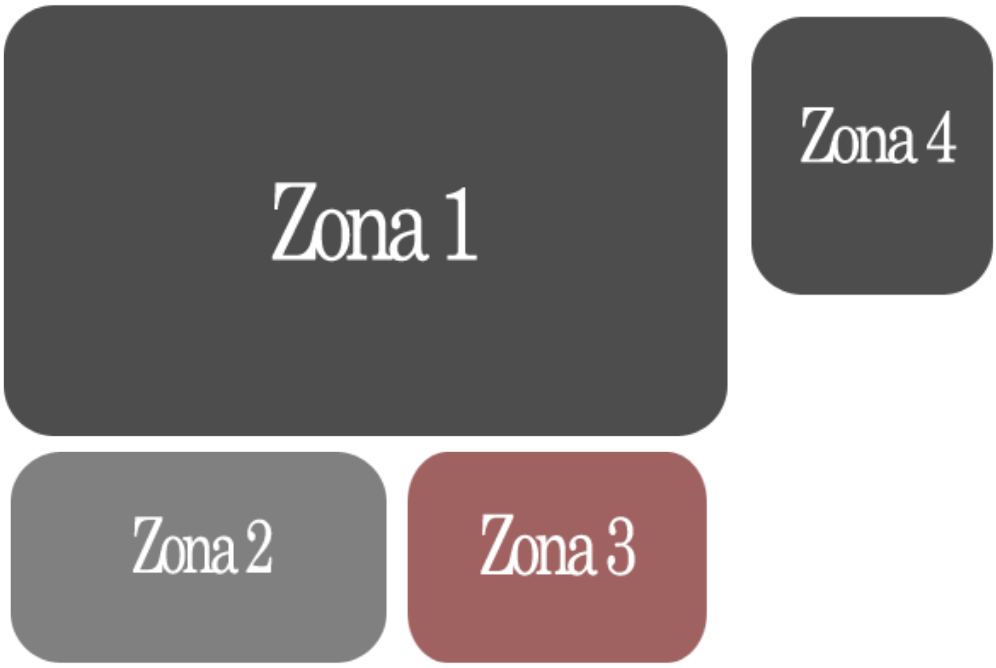


Figura 2.17: Esquema da sub-divisão da página principal

2.6 Gestão de Requisitos

Esta página permite que haja um maior controlo sobre onde e o quê o utilizador se pode inscrever. Imaginemos uma modalidade ou atividade que só pode ser praticada por um cliente com mais de 18 anos. Nesta página é possível configurar esse aspeto. Outra funcionalidade que esta página tem é configurar qual o escalão mínimo para se poder inscrever ou subir para o outro escalão. Resumindo esta página permite configurar os requisitos mínimos para a pratica de uma modalidade ou atividade, ou ainda o requisito mínimo de escalão necessário para se poder inscrever ou subir para outro escalão. Na Figura 2.18 e 2.19 podem ver um exemplo.

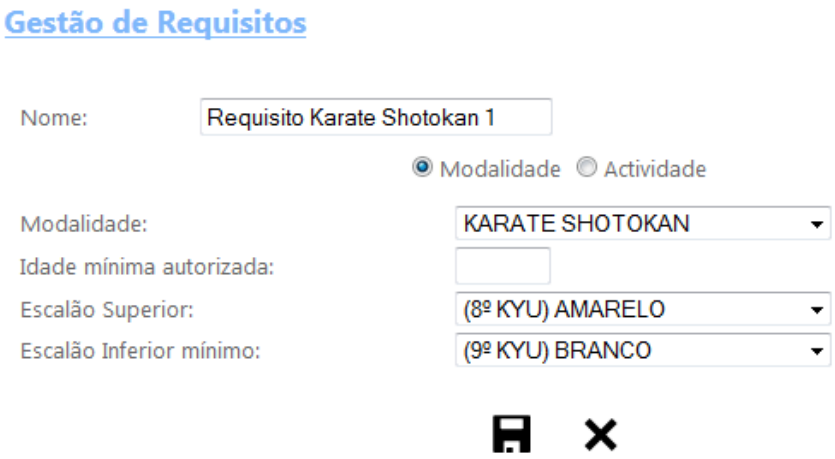


Figura 2.18: Modalidade com requisito de escalão

Figura 2.19: Actividade com requisito de idade

Capítulo 3

Área de Administrador

Como já foi referido, existem 4 grupos de utilizadores,nomeadamente os Administradores. Após o login feito o administrador terá acesso à sua área que contém algumas páginas de visualização de dados. Aqui o administrador pode consultar quais os valores envolvidos nas inscrições de clientes, qual a delegação com mais utilizadores, quais os clientes que têm mensalidades em atraso, etc.
Para responder a estas perguntas tem-se algumas páginas. Nomeadamente a página **Consultar dados de Inscrições** , **Consultar dados de Modalidades**, **Consultar dados de Atividades** e por fim a página Consultar dados de Clientes.
Comecemos então a explicar resumidamente o que pode ser obtido em cada uma das páginas.

Consultar dados de Inscrições (exemplo na Figura 3.1)— Esta página permite ao administrador graficamente consultar as inscrições ativas por modalidade e atividade assim como as inscrições ativas por delegação.

Consultar dados de Clientes (exemplos na Figura 3.2 e 3.3)— Aqui o administrador pode consultar a lista de utilizadores que com inscrição ativa ou inativa, consultar a lista de inscrições em modalidades e atividades e exportar a mesma lista para Microsoft Word ou Excel. A página ainda permite ao administrador graficamente visualizar o número de inscrições por delegação/modalidades/Actividades para comparação de resultados.

Consultar dados de Atividades (exemplo na Figura 3.4)— Esta página permite consultar as Atividades quanto ao número de inscritos e o total Faturado. Permite ainda exportar essa consulta para Microsoft Excel e Word.

Consultar dados de Modalidades (exemplo na Figura 3.5)— Muito semelhante à página anterior, mas aqui com uma informação adicional que é saber que utilizadores têm pagamentos em atraso (por pagamentos em atraso entende-se inscrições em modalidades e que o cliente tenha estado uma ou mais vezes presente, caso contrário não é considerado mensalidade em atraso).

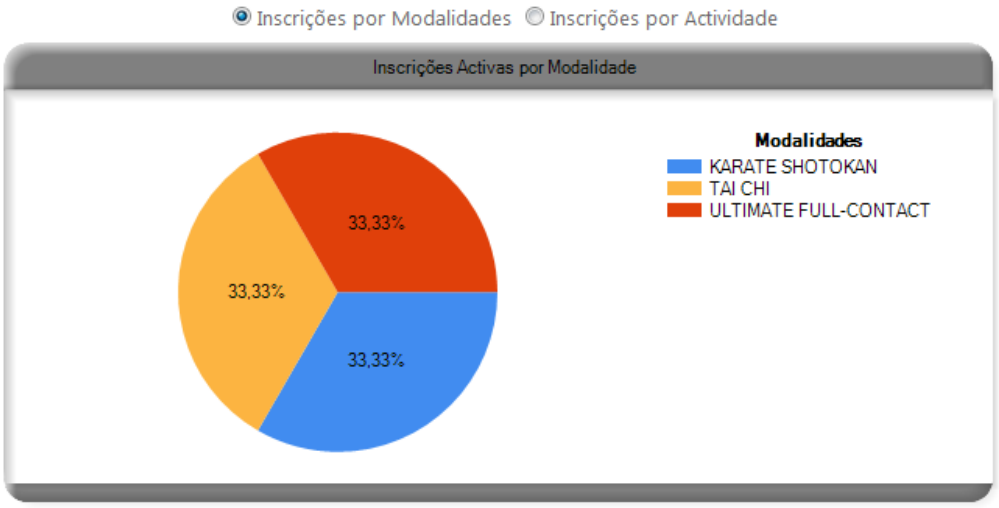


Figura 3.1: Exemplo de um gráfico obtido na página

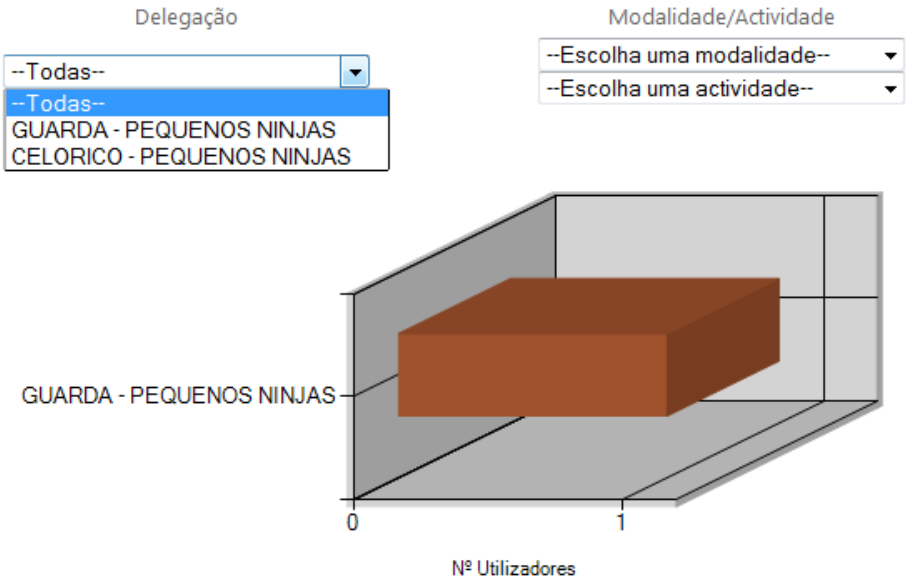


Figura 3.2: Exemplo de Gráfico de comparação de dados

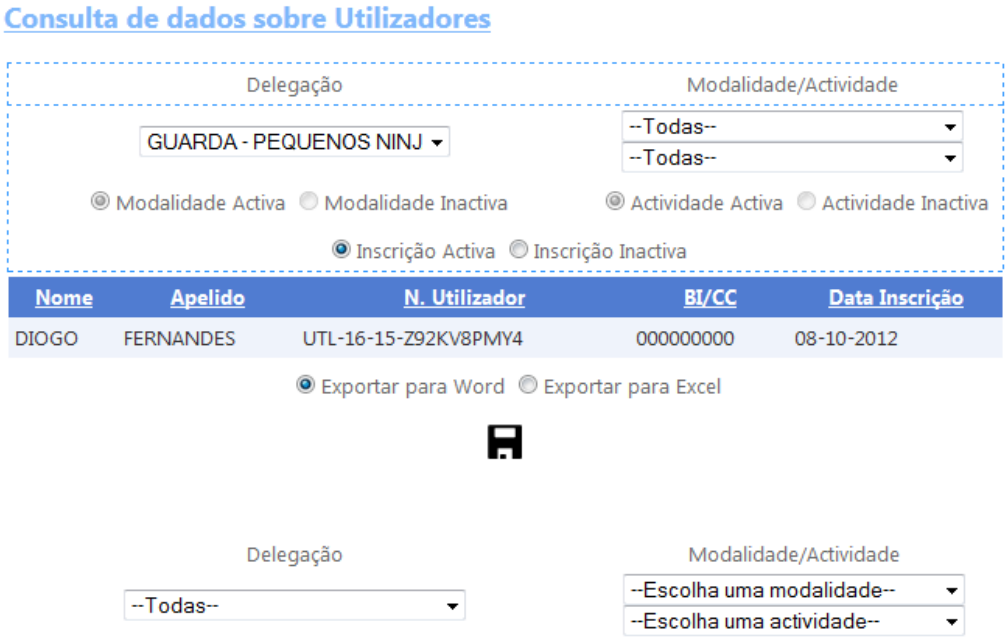


Figura 3.3: Exemplo de Lista de Clientes

Figura 3.4: Exemplo da página Consultar dados de Actividades

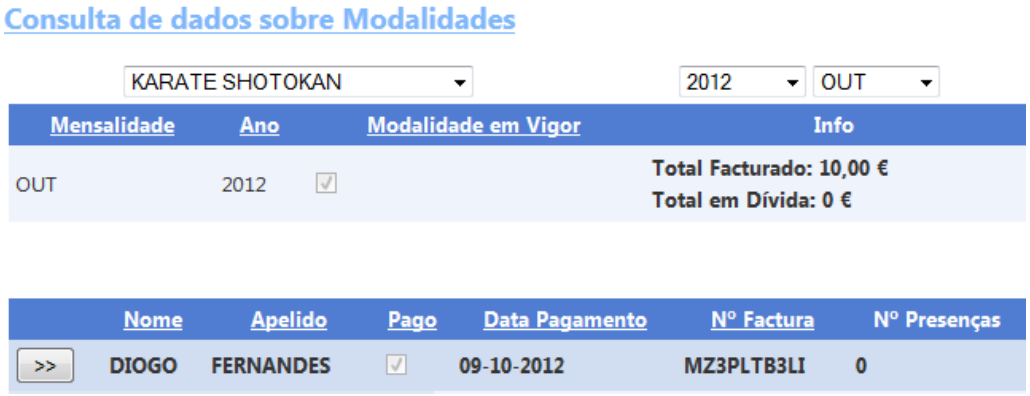


Figura 3.5: Exemplo da página Consultar dados de Modalidades

Capítulo 4

Área de Funcionário

Esta área esta reservada ao grupo dos Funcionários. É a área que contém as páginas de Novas Inscrições, Inscrições em Modalidades, Inscrições em Actividades, Renovação de Inscrições, Gestão de Presenças, Gestão de Conteúdos, Gestão de Presenças e Pagamentos. As páginas mencionadas são as mais importantes da aplicação uma vez que é a volta destas páginas que tudo irá funcionar. Comecemos então por apresentar resumidamente cada uma das páginas referidas.

Gestão de Conteúdos— É a página que permite ao funcionário inserir novos avisos, notícias no sistema e na página principal. A Página permite ainda aos utilizadores editar ou remover os seus conteúdos.

Inscrição em Actividades— Esta página deve ser usada para inscrever clientes já existentes no sistema em novas actividades.

Inscrição em Modalidades— Esta página deve ser usada para inscrever clientes já existentes no sistema em novas modalidades.

Pagamentos— É esta a página que deve ser usada quando um utilizador quer fazer pagamentos. É gerado um recibo para o funcionário que também é enviado para o cliente por email. O Funcionário só tem que escolher o cliente da lista e clicar num dos botões de pagamentos. Se o cliente tiver pagamentos em atraso esses pagamentos serão todos agrupados. O objetivo é que o cliente não tenha mensalidades em atraso.

Subida de Escalões— Aqui o funcionário poderá subir um cliente de escalão nas modalidades inscritas. Pode consultar todas as subidas de escalões e a data que ocorreram. Para efetuar uma subida de escalão, o funcionário tem de selecionar o cliente, depois selecionar a modalidade na qual vai ser efetuada a mudança de escalão e escolher a data na qual essa mudança teve lugar.

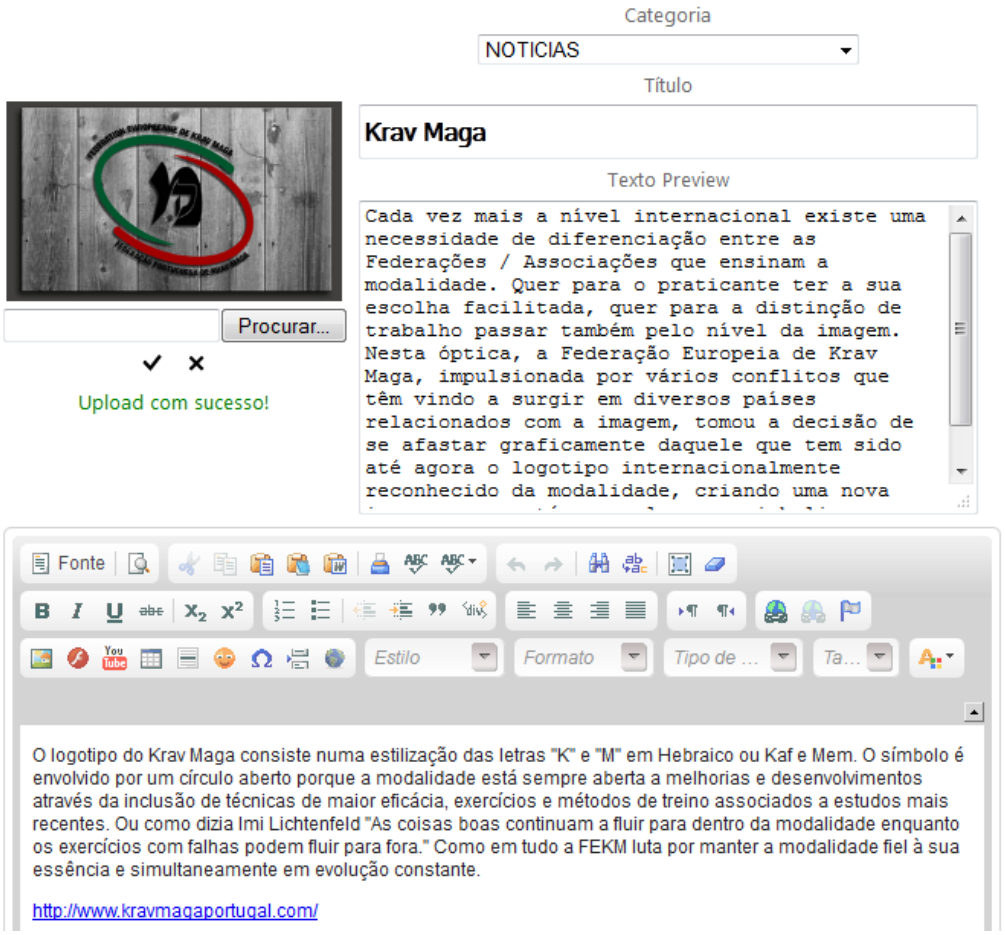


Figura 4.1: Criar Notícia

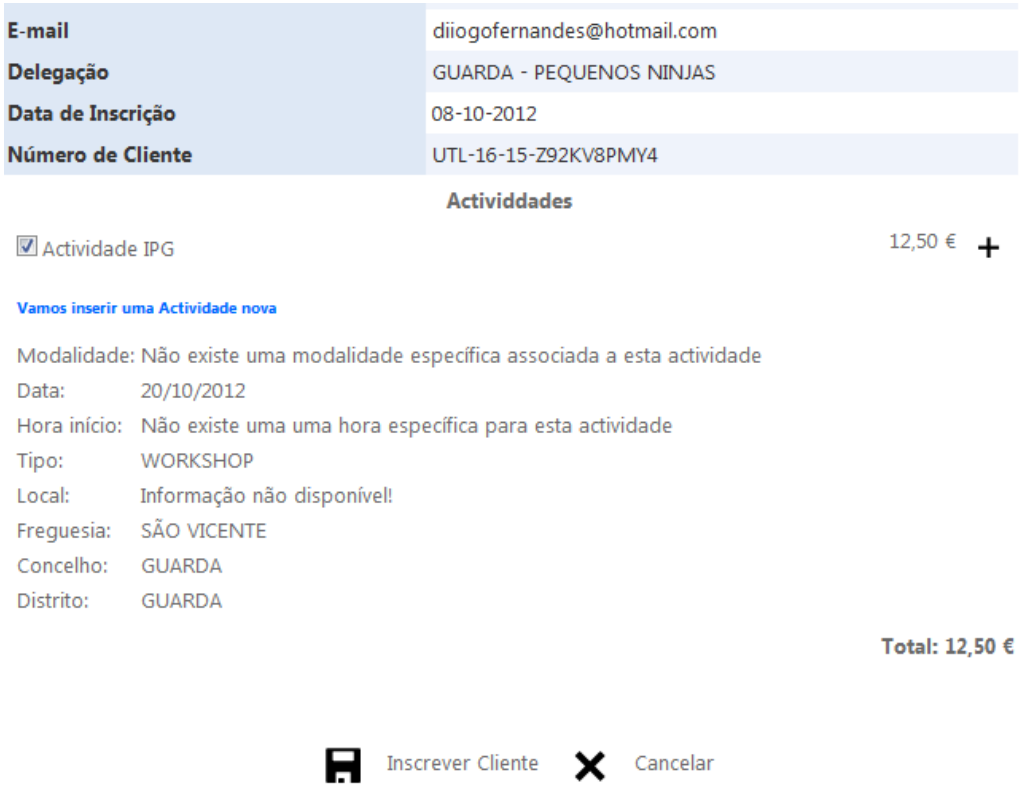


Figura 4.2: Inscrição em Atividades

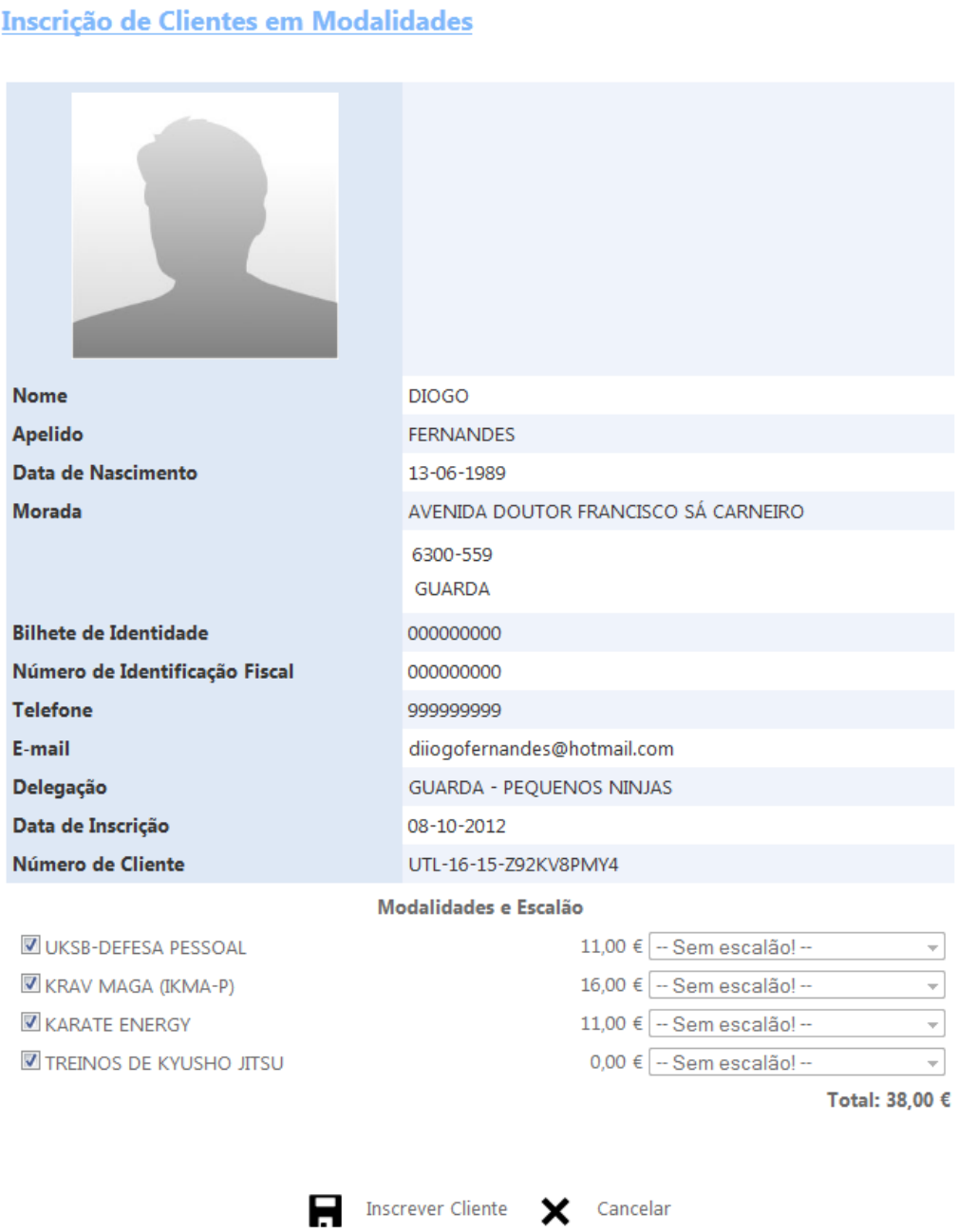


Figura 4.3: Inscrição em Modalidades

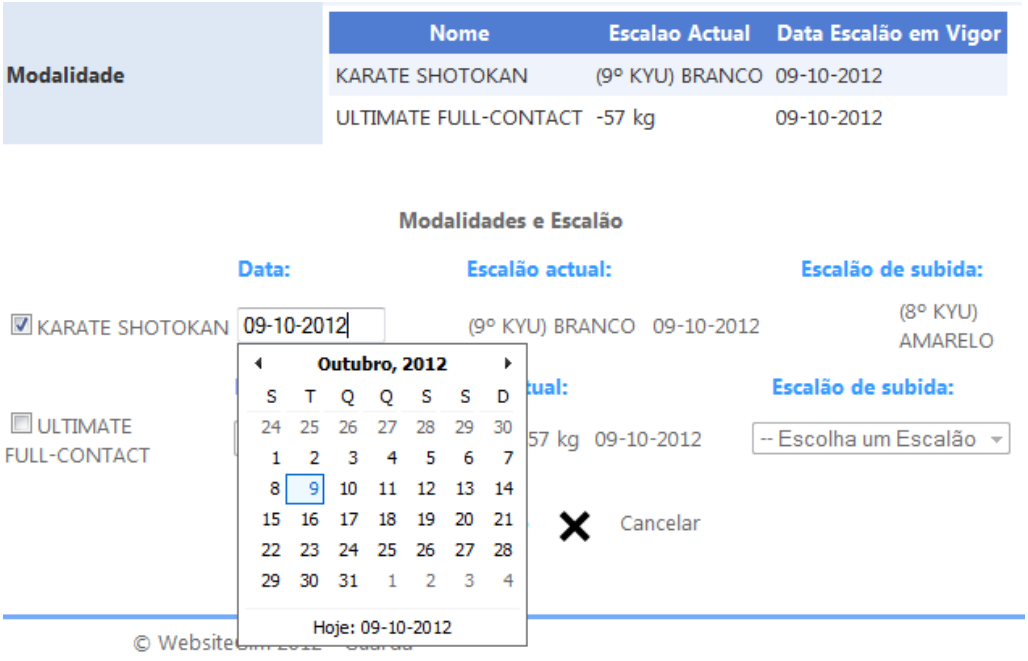


Figura 4.4: Subida de Escalão

Gestão de Presenças— Nesta página o funcionário pode marcar presenças aos clientes, para isso só tem que selecionar a modalidade ou atividade e o dia da presença, de seguida aparecerá uma lista com todos os clientes inscritos naquela modalidade, bastando escolher os clientes que estiveram presentes e clicar no botão **Inserir Presenças**. A página também permite a eliminação de presenças no entanto esta funcionalidade só está disponível para os utilizadores que pertençam ao grupo Administrador ou Super Administrador.

Renovação de Inscrição— Aqui o funcionário pode renovar a inscrição dos clientes já existentes, ou seja, a cada época nova todos os clientes terão de renovar a sua inscrição caso queiram continuar a ser considerados praticantes de uma ou mais modalidades. É nesta página que isso é feito.

Inscrição de Novos Clientes— Quando o funcionário tiver que inscrever um novo cliente deve usar esta página. Aqui o funcionário poderá inscrever novos clientes que não estejam no sistema, em atividades ou modalidades. Terá de preencher os dados pessoais do cliente e escolher as modalidades. Após estes passos e quando finalizar a ação será gerado um recibo para o funcionário e enviado por email ao cliente o mesmo recibo. No email vai contido um código de ativação e respetivas instruções que é necessário para o cliente poder efetuar login na sua conta.

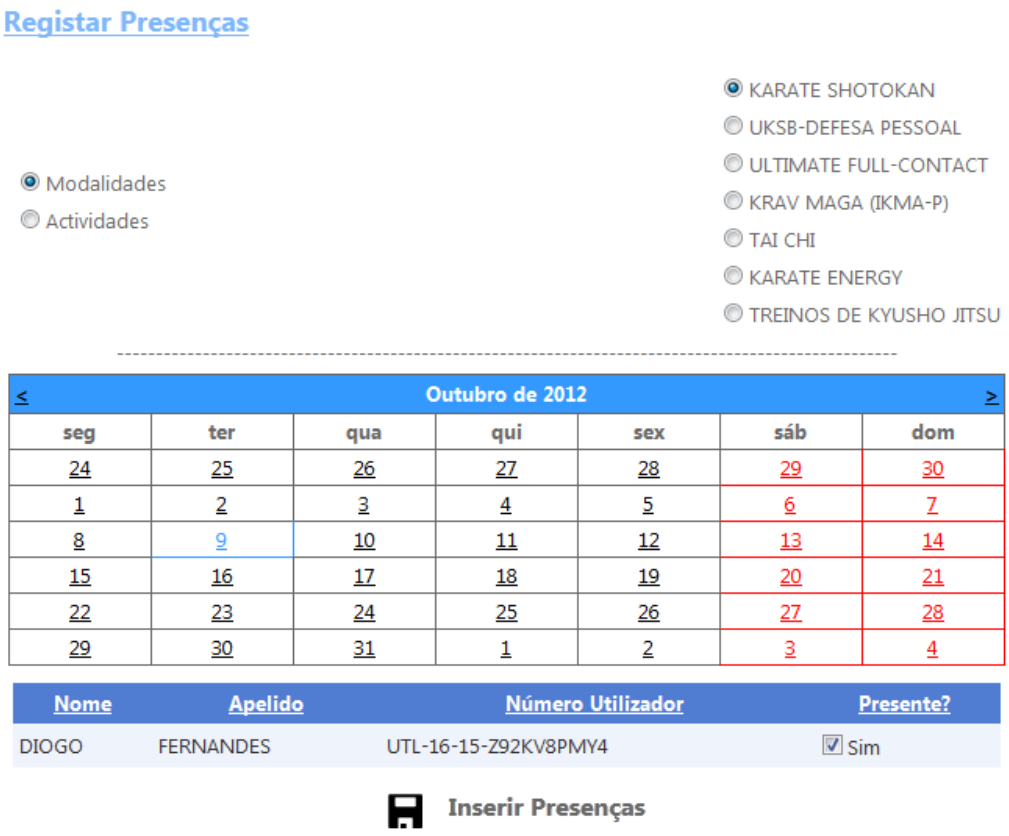


Figura 4.5: Marcar Presenças

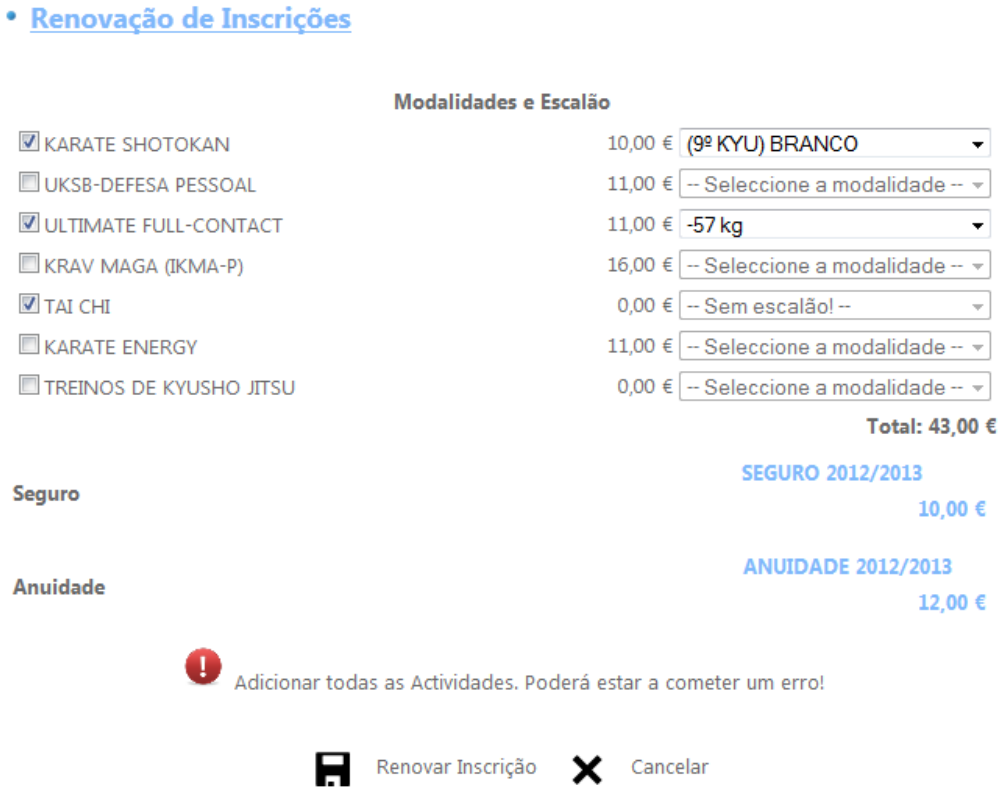


Figura 4.6: Renovação de Inscrição

Capítulo 5

Área de Cliente

A Área de Cliente é a área respetiva dos clientes (os que pertencem ao grupo Users). Aqui o cliente pode alterar os seus dados (Figura 5.1) pessoais e a password de acesso. E pode consultar a página **Como Chegar?** (Figura **??**) que ajuda o cliente a encontrar um caminho para as actividades em que o utilizador está inscrito.

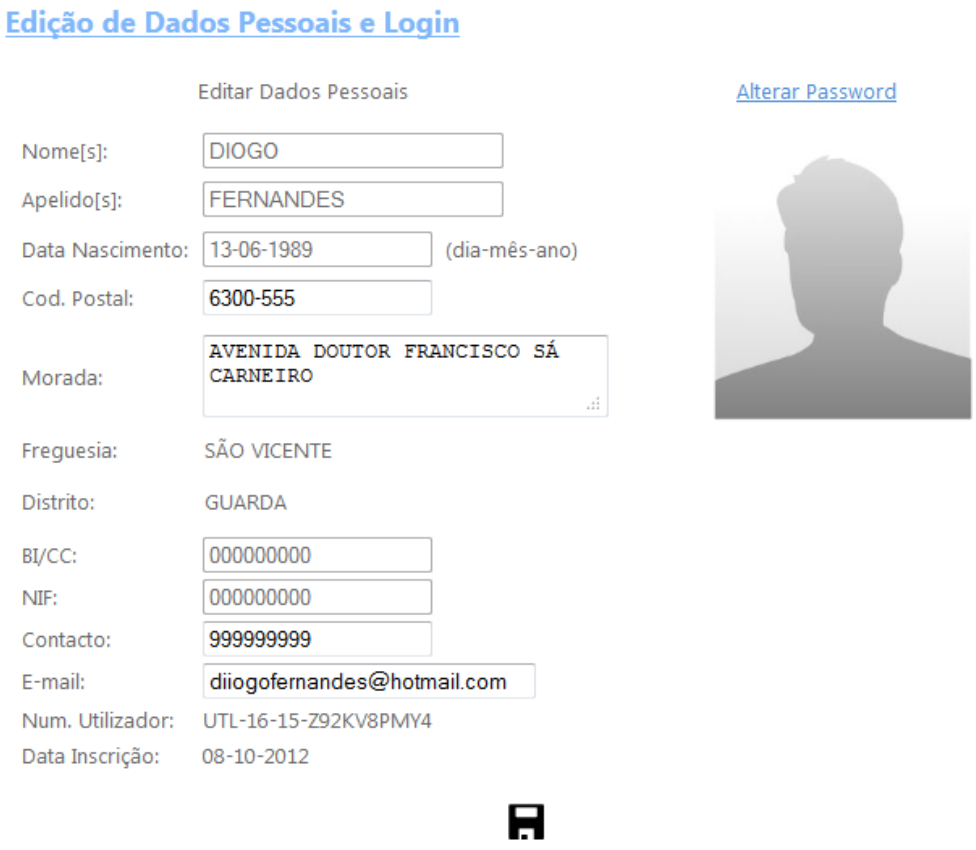


Figura 5.1: Alteração dos dados do cliente

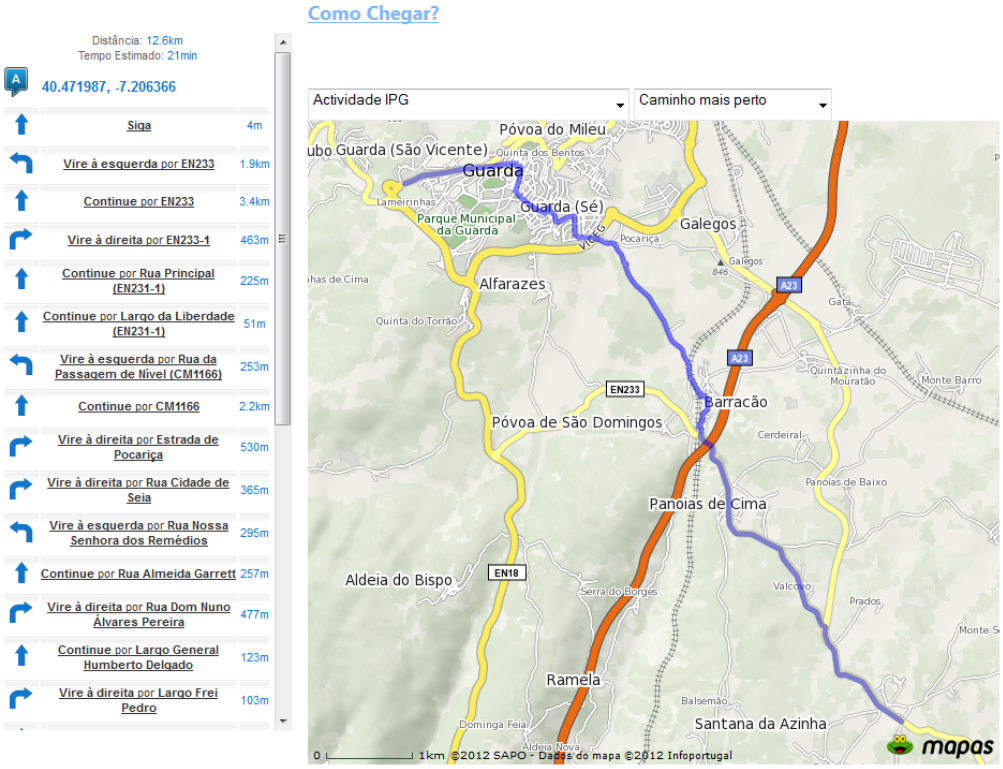


Figura 5.2: Página Como Chegar?

Capítulo 6

Como inserir vídeos do Youtube

Vamos aqui ensinar como inserir vídeos do youtube na Gestão de Conteúdos, Definições da Página etc. Em primeiro lugar clicar neste ícone (Figura 6.1)

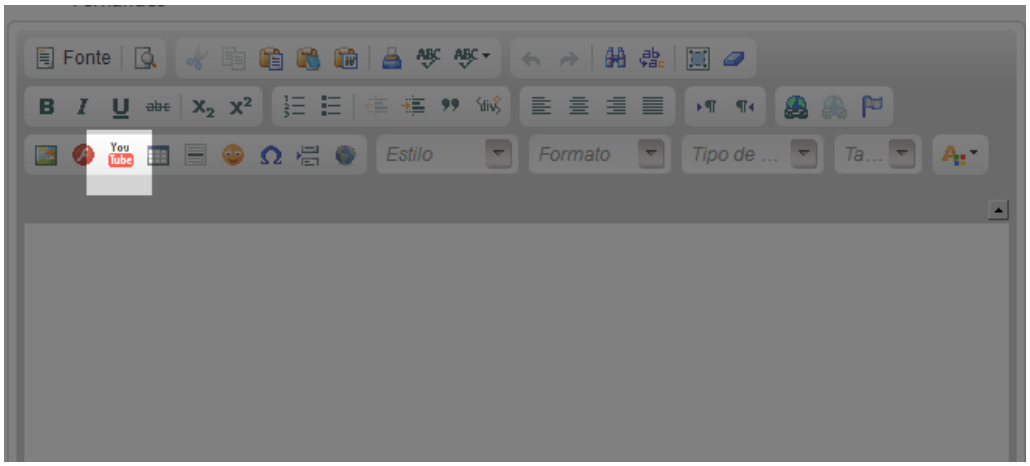


Figura 6.1: Youtube Ícone

De seguida ir ao site do youtube e escolher o vídeo. Copiar a parte selecionada do hyperlink (Figura 6.2)

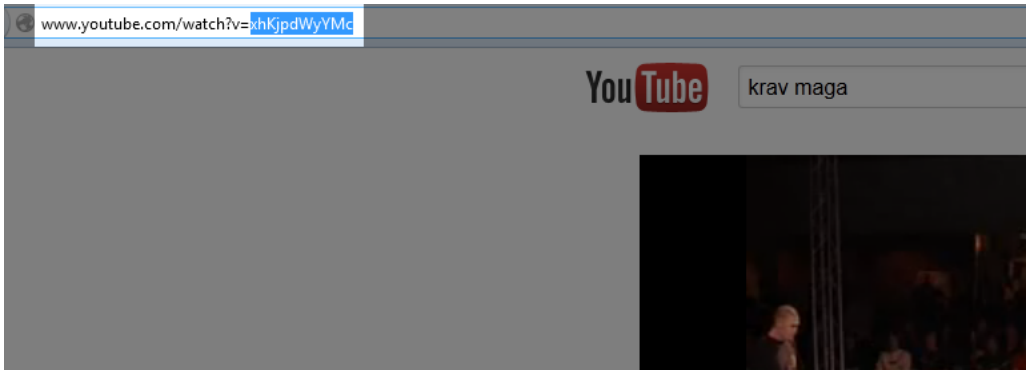


Figura 6.2: HyperLink Youtube

Por último, colar a parte seleccionada do hyperlink da página do youtube para a janela da nossa página (Figura 6.3)

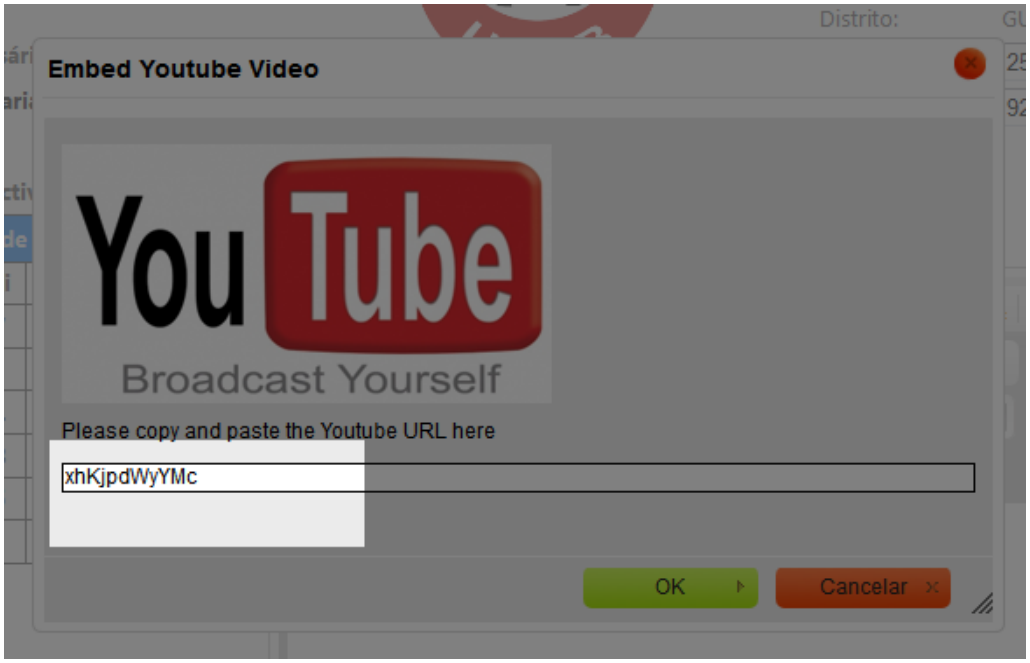


Figura 6.3: HyperLink Youtube

Clicar em **OK** e já está o nosso vídeo na nossa página!
O controlo usado tem outras opções e finalidades, não permite só inserir vídeos do youtube, deixamos a sugestão de explorar as outras funcionalidades ao nosso leitor.