Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE ESTÁGIO

Licenciatura em Engenharia Informática

Fábio José Gomes Caramelo

agosto | 2017

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE ESTÁGIO

FÁBIO JOSÉ GOMES CARAMELO

RELATÓRIO PARA A OBTENÇÃO DA LICENCIATURA

EM ENGENHARIA INFORMÁTICA

Agosto 2017

Ficha de Identificação

Aluno

Nome: Fábio José Gomes Caramelo

Numero: 1011270

Curso: Engenharia informática

Estabelecimento de ensino

Escola Superior de Tecnologia e Gestão – Instituto Politécnico da Guarda

Morada: Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Telefone: 271220120 | **Fax:** 271220150

Duração do Projeto

Inicio: 1 de Junho de 2017

Fim: 26 de Agosto de 2017

Orientador do Projeto

Nome: António Mário Ribeiro Martins

Grau académico: Doutor

Agradecimentos

Durante o período de licenciatura vários obstáculos e adversidades foram aparecendo e como estudante tive a necessidade de as ultrapassar e com elas ganhar consciência e conhecimento para os superar e para que no futuro estivesse pronto para exceder outros novos que surgissem. Contudo foi graças à minha família, amigos e professores que foram ultrapassadas e é a essas pessoas que gostaria de agradecer pelo apoio incondicional.

Quero agradecer ao meu orientador de estágio, o Professor António Mário Ribeiro Martins, pela ajuda disponibilizada não apenas no decorrer do estágio, mas durante os anos como estudante da licenciatura em Engenharia Informática.

Deixo também um agradecimento ao Engenheiro Ricardo Almeida e ao Engenheiro José Fidalgo por toda a ajuda, acompanhamento e apoio disponibilizado durante o estágio na Coficab Portugal e também ao meu colega de estágio Mário Costa.

Resumo

Este relatório tem como objetivo apresentar o trabalho desenvolvido por Fábio José Gomes Caramelo no âmbito do projeto em contexto de estágio da licenciatura de Engenharia Informática do Instituto Politécnico da Guarda. Este descreve o projeto, Welcome2Coficab, planeado e desenvolvido na empresa Coficab Portugal.

Welcome2Coficab é uma aplicação móvel a ser disponibilizada aos visitantes da Coficab Portugal capaz de disponibilizar noticias sobre a empresa, como informações uteis sobre atividades por esta realizadas, a sua história, que certificações e homologações possui e a sua visão e missão enquanto empresa líder do mercado de cablagens para a industria automóvel. A informação disponibilizada pela aplicação é para ajudar o visitante a guiar-se pela empresa e ter acesso aos serviços que esta disponibiliza, como pode aceder à internet e aceder ao CMS (*Canteen Management System*) e contactos uteis da mesma.

Neste documento serão detalhadas as diferentes fases do projeto da aplicação Welcome2Coficab desde o seu planeamento, desenvolvimento até à fase de implementação, tendo em conta as metodologias utilizadas e o estado da arte deu suporte para o seu desenvolvimento.

Palavras Chave:

Aplicação móvel, Backoffice, Aplicação hibrida

Abstract

This report has as objective present the work developed by Fábio José Gomes Caramelo in the scope of the project in an internship context of the Computer Science degree of Polytechnic Institute of Guarda. This will describe the project, Welcome2Coficab, planned and developed in Coficab Portugal.

Welcome2Coficab is a mobile application to be provided to the visitors of Coficab Portugal being able to supply news, activities undertaken within the company, its history, which certifications and approvals owns and its vision and mission while leading manufacturer in the wiring harness market for the automotive industry. The information supplied by the application it's to help the visitors walking through the company and have access to its services, how to connect to internet, how to access CMS (Canteen Management System) and useful contacts.

In this document will be detailed all the different phases of the Welcome2Coficab application project from its planning, development until the implementation phase in consideration of the used methodologies and the supported state of art for its development.

Key words:

Mobile Application, Backoffie, Hybrid Application

Índice

Índice de figuras

Índice de tabelas

Lista de siglas e acrónimos

API – Application Programming Interface

APK – Android Application Package

CMS – Content Management System

CSS – Cascading Style Sheet

DLL – Dynamic Link Library

GPS - Global Positioning System

HTML – HypeText Markup Language

IDE – Integrated Development Environment

iOS - iPhone Operating System

IPA - iOS App Store Package

Less - Leaner Style Sheets

MIT - Instituto de Tecnologia de Massachusetts

MVC – Model View Controller

Sass - Syntactically Awesome Style Sheets

UI – User Interface

UML – Unified Modeling Language

UX – User Experience

XML - Extensible Markup Language

WYSIWYG – What You See Is What You Get

1.Introdução

O relatório apresentado descreve o projeto em contexto de estágio de fim de curso desenvolvido no âmbito da unidade curricular de Projeto de Informática, unidade curricular do 3º ano da licenciatura de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

1.1. Caracterização da Coficab

A Coficab é uma empresa do grupo Elloumi que desenvolve e fabrica cabos elétricos para o ramo automóvel criada em 1992 na Tunísia.

Devido ao alto volume de exportação, o grupo Elloumi decidiu expandir a sua empresa para outros países, onde em 1993 abriu a sua fábrica em Portugal.

A Coficab Portugal conta com diversas certificações ao nível de qualidade, ambiente, investigação e desenvolvimento. Atualmente possui homologações por diversas empresas internacionais para assegurar a qualidade do seu produto e a sua utilização no ramo automóvel.

Atualmente a Coficab Portugal está inserida no programa Portugal 2020 com um projeto "Inovação Produtiva" para a investigação e desenvolvimento de novos produtos ou de já existentes.

1.2. Motivação

A escolha deste projeto em contexto de estágio proporcionou-se devido ao facto de que à escala mundial a utilização de *smartphones* cresceu exponencialmente e consequentemente os seus utilizadores utilizem os seus aparelhos para consultar e enviar informação de uma forma mais simples e rápida, e consequentemente a criação de aplicações moveis para o uso quotidiano teve também um grande crescimento.

1.3. Objetivos do projecto

Neste caso a Coficab Portugal pretende construir uma aplicação móvel para que os seus funcionários e visitantes tenham acesso às informações sobre a empresa, notícias de interesse e serviços que esta disponibiliza. Assim, a empresa, de um ponto de vista ambiental, pode dispensar a impressão de panfletos e cartazes para proporcionar uma divulgação de informação de forma mais rápida e eficiente. Desta maneira pretende-se:

1. **Fazer o levantamento das ferramentas e tecnologias a utilizar**

   Uma vez que o projecto se desenrola num contexto nunca antes lecionado durante o meu percurso formativo, é necessário fazer uma pesquisa das ferramentas que mais se adequam ao projecto, tanto para o gestor de conteúdos, como para a aplicação móvel. As tecnologias utilizadas por estas ferramentas são linguagens ou técnicas web que foram lecionadas durante a licenciatura, porém o estudo da sua aplicação será necessário.

2. **Conceção e avaliação dos requisitos do projeto**

   Após o levantamento e adaptação das ferramentas a utilizar, o próximo objetivo passa por fazer juntamente com o orientador da Coficab o levantamento dos requisitos funcionais. Esta etapa será a de maior importância uma, pois a partir deste momento será delineado tudo aquilo que será desenvolvido.

3. **Desenvolvimento da aplicação**

   É neste objetivo que se passa grande parte do tempo de estágio, pois consiste no desenvolvimento da aplicação Welcome2Coficab com base nos requisitos levantados no ponto anterior. Segue-se a metodologia Ágil dividindo, em ciclos, o desenvolvimento da aplicação, e o acompanhamento semanal por parte do orientador da empresa.

4. **Testes**

   Pretende-se que sejam feitos testes durante do desenvolvimento da aplicação, para garantir que a implementação está de acordo com os requisitos. Pretende-se que,

na etapa final do estágio, toda a aplicação seja testada por um elemento da Coficab, garantido a imparcialidade dos teste.

1.4. Estrutura do relatório

Em termos de estrutura, este relatório é composto por 6 capítulos. Em cada um destes capítulos vai ser descrito o progresso do projeto da seguinte forma:

- A caracterização da empresa e a motivação da escolha deste projeto;
- Abordagem ao problema, onde serão estudadas as possíveis tecnologias e a sua escolha para este projecto;
- A metodologia que irá ser utilizada e a modelagem para a produção de software e a descrição das tecnologias envolvidas;
- Descrição do processo de desenvolvimento das duas partes do projeto;
- A validação de ambas as partes do projeto e testes elaborados ao mesmo;
- A conclusão onde será feita uma análise ao projeto e há importância do estágio.

2. Estado da Arte

Uma vez levantados os requisitos da aplicação a desenvolver é necessário a escolha de ferramentas que permitam a sua construção, com isto é necessária a recolha de informação de ferramentas pertinentes que permitam o desenvolvimento da aplicação ao nível de *front-end* onde é apresentada a informação ao utilizador, e *back-end* onde será introduzida a informação que irá ser fornecida ao utilizador pelo *front-end*, em que neste ultimo será também incluído um editor próprio para facilitar a formatação de texto e imagem.

2.1. 'Frameworks' para aplicações móveis

Com a grande utilização de dispositivos moveis à escala mundial onde as aplicações moveis são procuradas e utilizadas com elevada frequência debate-se entre o desenvolvimento de aplicações com compatibilidade para um determinado sistema operativo móvel, designadas de aplicações nativas, e entre aplicações compatíveis com todos os diferentes sistemas operativos moveis existentes designadas por aplicações *HyperText Markup Language* (HTML5) e aplicações hibridas [1], que podem ser desenvolvidas em plataformas ou 'Frameworks' de código livre que possuem componentes necessários, como classes, objetos e recipientes, para a interpretação de código numa determinada linguagem de programação, auxiliando o desenvolvimento e implementação de *software*.

Enquanto as aplicações nativas são desenvolvidas para um só sistema operativo móvel (Android ou iOS), e posteriormente distribuídas nas lojas de aplicações dos respetivos sistemas estas surgem como as melhores aplicações dentro de performance e design. Contudo com a evolução tecnológica e o interesse da área móvel, apareceram novas formas de criação de aplicações não nativas, onde se encontram as aplicações HTML5 e as aplicações hibridas.

A aplicações HTML5 recorrem a tecnologias como o HTML5, JavaScript e *Cascading Style Sheet* (CSS). Contudo há determinadas limitações ao nível da gestão de sessões e acesso a funcionalidades dos *smartphones* presentes nas aplicações nativas (câmara, calendário, localização Global Positioning System (GPS), etc…). As aplicações

híbridas são em tudo semelhantes às aplicações HTML5 mas encontram-se dentro de um "recipiente" nativo que fornecem acesso às funcionalidades das aplicações nativas, acima referidas, e fazem uso da loja de aplicações que eram exclusivamente utilizadas pelas as aplicações nativas dos sistemas operativos moveis, fazendo das aplicações hibridas uma plataforma acessível onde se pode aplicar linguagens para a internet na criação de aplicações para *smartphone*.

Como o projeto a desenvolver procura integrar o suporte para as duas plataformas móveis e após alguma pesquisa foi decidido a utilização de ferramentas que permitam a criação de aplicações híbridas. Dentro de uma vasta oferta de ferramentas, que foram analisadas, irá ser escolhida a que mais se adequa ao projeto. Nos próximos pontos serão descritas as funcionalidades das ferramentas.

2.1.1. Bootstrap

O Bootstrap é uma *framework front-end*, que foi criada por elementos da equipa de desenvolvedores do Twitter [2], ficando conhecida por Twitter Blueprint (Planta do Twitter). É uma ferramenta que utiliza a linguagem Syntactically Awesome Style Sheets (SASS) e Leaner Style Sheets (Less), que são pré-processadores do CSS que ajudam a reduzir a sua repetição e possibilitam a personalização, gestão e reutilização de um estilo de *design* para uma página *web*, e que disponibiliza documentação para componentes HTML, CSS e extensões para jQuery.

Tem suporte em diferentes *browsers* (um programa que permite a navegação *web* como o Chrome, Firefox, Opera, Safari e Internet Explorer, em que a página desenvolvida seja responsiva dependendo do ecrã que o utilizador esteja a utilizar).

Conta com componentes de interface de utilizador (UI) consistentes, responsivos e personalizáveis, porém, este tipo de *framework* é mais utilizado para a criação de *websites* uma vez que não tem possibilidade da compilação de um ficheiro APK (Android Package) ou IPA (iOS App Store Package), para Android e iPhone respetivamente.

A sua inclusão já é pré-definida pelos Integrated Development Environment (IDE), que sejam indicados para a construção de serviços *web*, quer sejam aplicações *web* ou aplicações híbridas, onde é apenas chamada a classe correspondente para a sua inclusão.

2.1.2. Apache Cordova

É uma *framework* de código aberto para o desenvolvimento móvel [3], que permite a utilização de tecnologias *web* padrão como o HTML5, CSS3 e JavaScript para desenvolvimento multiplataforma, que se executam em código encapsulado (*wrappers*) específicos para cada plataforma (Android, iOS, Windows Phone) utilizando APIs para ter acesso às capacidades nativas de cada sistema operativo móvel (camara, sistema de ficheiros, GPS, etc.).

A sua utilização passa pela utilização de um IDE como o Visual Studio que possui as bibliotecas para esta *framework* facilitando a inclusão das mesmas.

2.1.3. Ionic2 Framework

É uma ferramenta código aberto [4], através de uma licença permissiva do Instituto de Tecnologia de Massachusetts (MIT), contruída em torno de Angular JavaScript ou AngularJS, utilizado para aplicações de página única que fornece uma experiencia de utilização ao utilizador semelhante a uma aplicação de computador, com suporte para Android e iOS. Está otimizada para dispositivos de toque, está mais focada para a construção de aplicações nativas/hibridas do que para sites dedicados para dispositivos móveis. Tem suporte para Cordova, PhoneGap e Trigger.io com a opção da utilização da linguagem SASS para gerar o código CSS e posteriormente ser editado para as preferências do programador. Disponibiliza diferentes tipos de modelos com um UI limpo e, experiencia de utilizador (UX), conciso tornando a aplicação final mais intuitiva para o utilizador fazendo desta ferramenta uma das mais utilizadas e procuradas por quem quer desenvolver aplicações *mobile*.

Apesar de ter inclusão em vários IDEs, estes ainda não estão preparados para incluir a nova versão desta *framework* e posteriores (Ionic3 agora em fase beta). Daí ser utilizada a partir da linha de comandos ou terminal do sistema operativo e o código escrito em editores de texto como o Notepad++, Sublime Text ou Visual Code.

2.1.4. Resumo das Frameworks

Após de uma breve analise entre as aplicações nativas e hibridas e de um momento de decisão, a escolha recai sobre as aplicações híbridas, uma vez que a partir de

ferramentas de código aberto é possível fazer uma programação para dois sistemas operativos moveis completamente distintos.

Após a reflecção sobre as ferramentas existentes no mercado, a mais precisa para este projeto seria a Ionic Framework, uma vez que conta com múltiplas funcionalidades do Bootstrap e do Apache Cordova, contudo não é integrável em vários sistemas de programação e devido a este detalhe, em conjunto com a empresa, foi decidido a utilização do Apache Cordova uma vez que pode ser obtido através do instalador do Visual Studio e a inclusão de bibliotecas e funcionalidades tornar-se-ão mais fáceis.

A utilização desta *framework* irá fornecer uma página em branco, onde se terá que desenvolver todo o aspeto e funcionalidades da aplicação, uma vez que não é fornecido qualquer tipo de projecto pré feito onde o programador se pode basear de forma a tornar a aplicação mais interativa e com uma estética mais agradável.

Como tal a aplicação prevê-se correr como uma aplicação nativa para dois sistemas diferentes podendo fazer uso das lojas de aplicações destes sistemas, o que irá compensar de forma monetária e programável.

2.2. Ferramentas para CMS

As ferramentas CMS são ferramentas que permitem uma entidade editar texto e imagens, para, posteriormente serem visíveis numa aplicação *web*, neste caso, numa aplicação móvel. Isto ajuda a incluir, editar ou remover a informação de uma forma mais fácil sem ter que alterar o código da aplicação a desenvolver.

2.2.1. WordPress

Esta ferramenta, antigamente exclusiva para a criação e manutenção de *blogs* WordPress, é uma ferramenta de código aberto que pode ser modificada e utilizada para diversos fins [5].

A sua construção é simples, uma vez que já possui um *template* base e a partir dai é necessário criar um sistema de segurança com *login* e definir as permissões das entidades que irão utilizar o sistema.

2.2.2. ASP.NET MVC

É uma *framework open-source* para aplicações *web* desenvolvida pela Microsoft que implementa um padrão Model-View-Controller (MVC)[6], onde cada parte representa um sistema próprio.

O Model apresenta a parte empresarial que aplica a lógica do domínio de dados como os campos de uma base de dados. O View apresenta a parte do UI que é criada a partir dos dados do modelo utilizado (Model). O Controller é o componente que lida com a interação do utilizador, enviando os dados para o Model e posteriormente a View cria a visualização dos dados.

Com a utilização deste tipo de *framework* o programador consegue dividir a aplicação em três partes para que seja mais fácil de manter o serviço e controlar por completo essa aplicação.

Para a parte da visualização esta *framework* inclui os componentes Bootstrap para que reaja em função ao ambiente em que está a funcionar.

2.2.3. Resumo das ferramentas de CMS

Ambas as ferramentas analisadas preenchem os requisitos procurados, porém o ASP.NET MVC é a ferramenta ideal uma vez que inclui a construção de um CMS de raíz podendo integrar o sistema de segurança com as permissões e dados já existentes na base de dados da empresa. É também mais adequada uma vez que pode ser estruturada de uma maneira diferente do CMS proporcionado pela WordPress, podendo assim definir uma melhor interface e gestão de conteúdos.

3. Metodologias e Modelagem

Neste capitulo irá ser feita a caracterização da metodologia chamada Ágil [7], e delinear as etapas para a produção do projeto e tem como valores fundamentais a interação entre os intervenientes, as melhores ferramentas para realizar a produção, a colaboração do cliente e a resposta às mudanças de plano.

3.1. Metodologias aplicadas

A metodologia Ágil foi escolhida devido ao seu principio iterativo em que todos os intervenientes do projeto têm um papel fundamental no mesmo.

Em todas as iterações do desenvolvimento, todos os intervenientes reúnem-se para fazer a planificação de tarefas, neste caso como a equipa é apenas composta pelo aluno, orientador e cliente, as reuniões são mais frequentes (semanalmente), criando um melhor canal de comunicação e compreensão entre os vários intervenientes havendo um eco por parte do cliente/ utilizador para que haja a reestruturação, adaptação e criação de requisitos e funcionalidades.

Esta metodologia é também importante devido ao facto de saber a evolução do projeto e também indicar novas metas para a próxima fase.

3.2. Desenho e modelagem

A aplicação irá ser dividida em duas partes sendo uma a parte administrativa (CMS) onde será feita a gestão do conteúdo, e a parte do utilizador (Aplicação Móvel) que irá receber o conteúdo posteriormente inserido através da parte administrativa.

A parte administrativa é da responsabilidade da empresa e a criação dos utilizadores para o CMS será feita pelo responsável da empresa que dará o acesso a utilizadores para a plataforma, com isto a aplicação não terá qualquer tipo de registo de utilizadores, uma vez que os utilizadores com permissão de acesso são funcionário já existentes na base de dados da empresa.

Utilizador - Funcionário:

- Inserir, editar e remover as páginas principais;
- Inserir, editar, ver e remover o conteúdo das páginas principais;

Utilizador - Administrador:

- Todas as funcionalidades do utilizador – Funcionário;
- Criação de utilizador – Funcionário;
- Gestão de privilégios e acesso.

A parte do utilizador será apenas uma interface gráfica, aplicação móvel onde o utilizador irá ver a informação que foi submetida no CMS.

Utilizador:

- Ver o conteúdo da aplicação

3.2.1. Diagrama de Contexto

Antes de começar a desenvolver o CMS com as operações descritas em cima, é necessário fazer a modelação em *Unified Modeling Language* (UML), procedendo assim a planificação dos atributos e dos atores e sistemas responsáveis pelas operações [8]. O diagrama de contexto visa planificar a interação que os atores têm com o sistema CMS e desta forma extrair os casos de uso de cada ator, como mostra a figura 1.

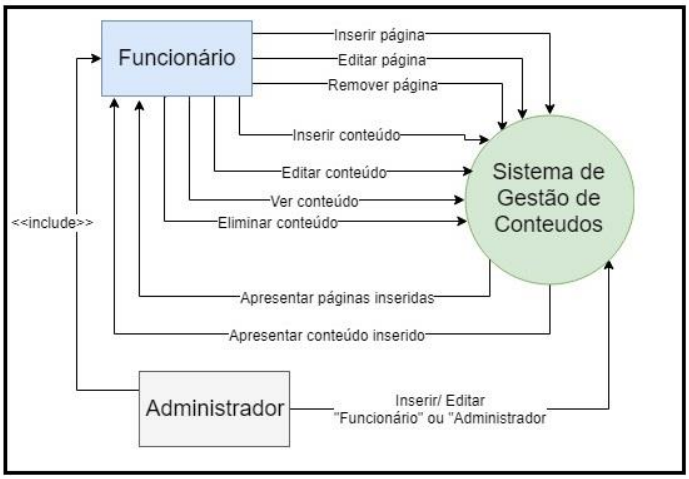

Figura 1 Diagrama de contexto

3.2.2. Diagrama de Casos de Uso

O diagrama de casos (Figura 2) de uso é composto por quatro partes: o cenário, o ator, o chamado caso de uso da aplicação e a comunicação.

Neste projecto os atores, quem interagem com o sistema, são o administrador e o funcionário, em que o administrador possui os mesmos casos de uso do funcionário acrescendo-lhe a gestão dos utilizadores e dos seus privilégios, fazendo assim uso dos casos: “Inserir funcionário” e “Editar funcionário”.

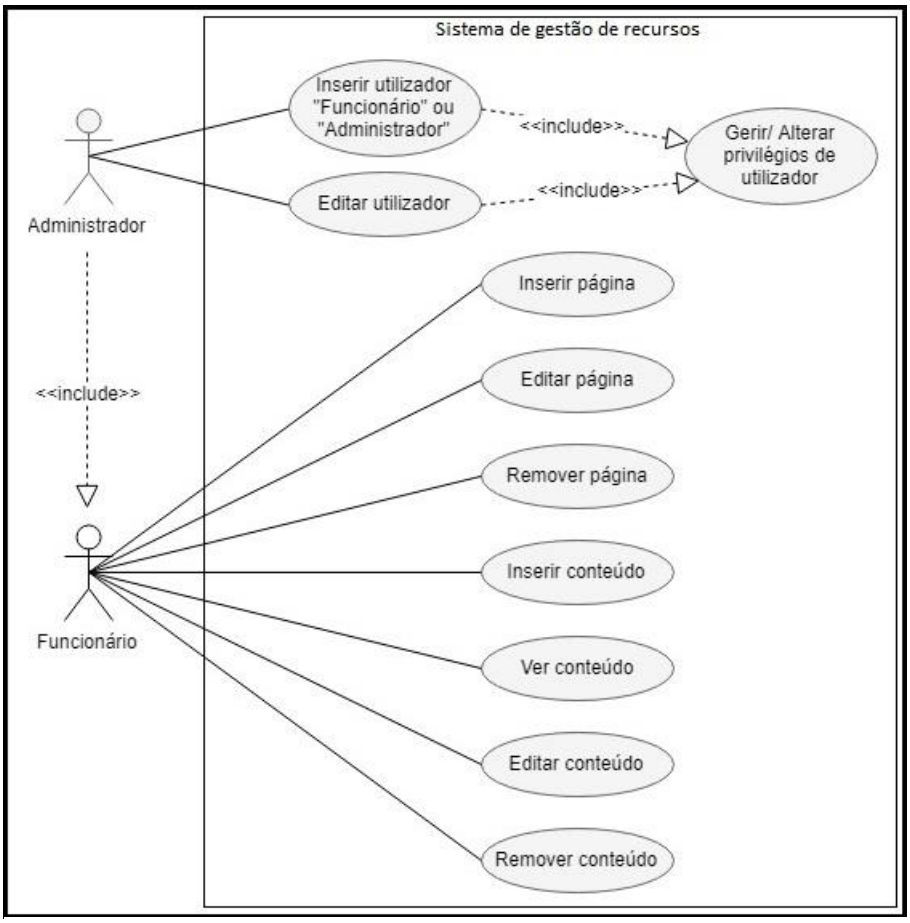

Figura 4 Diagrama de casos de uso

Com o diagrama de casos de uso pode-se iniciar o processo de criação de classes, que, ao sendo descritas, vão-se tornar em objetos caracterizados.

Apesar de no diagrama de contexto e de classes constar a adição e edição de um funcionário, as respetivas classes não irão ser descritas, visto que essa operação não consta na aplicação CMS, uma vez que é uma operação realizada por um administrador interno da empresa.

3.2.3. Descrição de casos de uso e diagramas de sequência

Nos quadros seguintes, que descrevem os casos de uso referidos no ponto anterior estão divididos em oito secções: nome, objetivo, atores envolvidos, pré-condição, fluxo principal, fluxo alternativo, fluxo de exceção pós-condição e casos de teste.

A descrição dos casos de uso é também importante porque permite definir as interfaces necessárias para o utilizador interagir com a aplicação.

3.2.3.1. Inserir página

O objetivo deste caso de uso é a inserção de uma página e uma descrição que, desde já, permite retirar a informação que será uma das interfaces necessárias, assim como as classes necessárias: página (nome dado à página) e descrição.

Nome	Inserir página
Objetivo	Registar uma nova página na base de dados
Atores envolvidos	Funcionário
Pré-condição	*Login* válido
Fluxo principal	**1.** O ator seleciona a opção "Inserir página" na interface "Página". **2.** O sistema apresenta um campo para introduzir o nome que quer dar à página e outro para introduzir a descrição. **3.** O ator preenche ambos os campos e clica em "Submeter página". **4.** O sistema guarda o registo de uma nova página. (4A)
Fluxo alternativo	**4a)** O sistema verifica a existência de uma página com o mesmo nome e indica ao ator que a página já existe.
Pós-condição	O sistema mostra a página recém criada e a descrição na interface "Página".
Casos de teste	**1.** Verificar se a página a criar já é existente. **2.** Verificar se o campo obrigatório, nome, é preenchido. **3.** Verificar se o sistema apresenta o botão "Editar" e "Remover" após a criação de uma nova página.

Com a descrição do caso de uso já é possível identificar alguns elementos que a interface deve conter, como o botão de criação, de submissão, um para editar e outro para remover, e apresentar o nome a dar à página como obrigatório.

O diagrama de sequência (Figura 3), que resulta da descrição do caso de uso, permite identificar os campos necessários para uma tabela relacional.

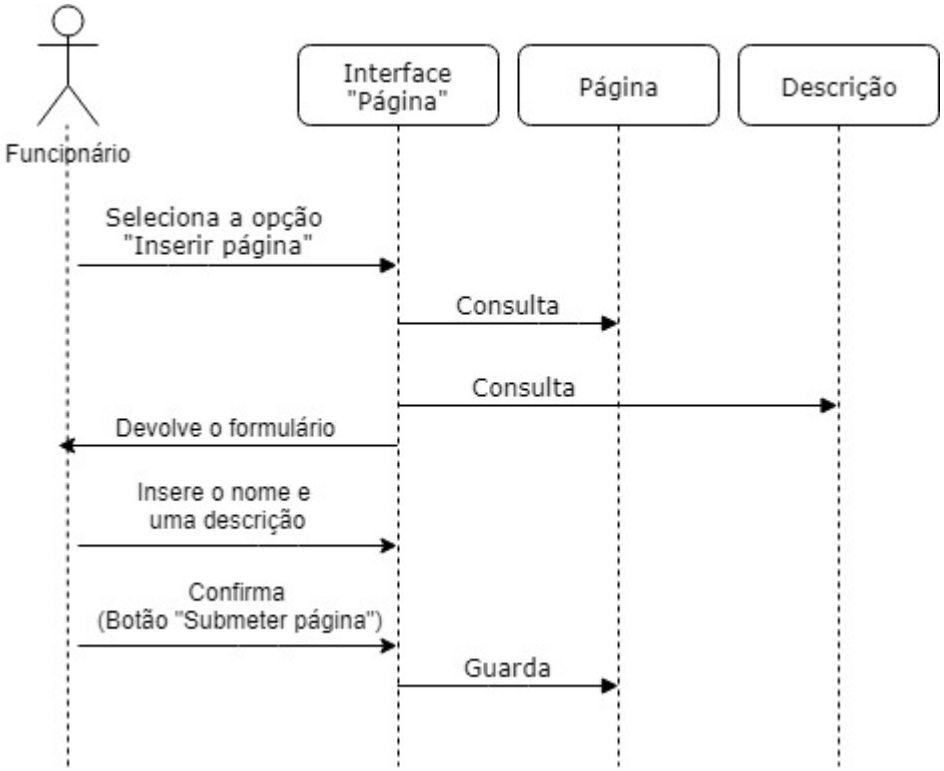

Figura 5 Diagrama de sequência "Inserir página"

3.2.3.2. Editar página

A descrição deste caso de uso permite descrever a forma como o sistema efetua a operação para editar o nome da página e a sua descrição.

Nome	Editar página
Objetivo	Editar o nome e descrição de uma página da base de dados
Atores envolvidos	Funcionário
Pré-condição	Login válido
Fluxo principal	**1.** O ator seleciona a opção "Editar" numa página já existente na interface "Página". (1A) **2.** O sistema apresenta um campo de texto com o nome atual da pagina e outro com a descrição existente. **3.** O ator edita ambos os campos e clica em "Submete" **4.** O sistema guarda a edição da página. (4A)
Fluxo alternativo	**1A.** **a)** O ator escolhe a opção "Eliminar". **b)** O sistema apresenta uma mensagem de aviso: "Pretende eliminar esta página?". **c)** O ator confirma a opção. (1Ac) **d)** O sistema volta à interface "Página".
	1Ac. **a)** O sistema deteta que a página tem conteúdo introduzido. A opção eliminar não está ativa.
	4A. **a)** O sistema verifica a existência de uma página com o mesmo nome e indica ao ator que a página já existe.
Pós-condição	O sistema atualiza a página e mostra o novo nome e descrição da página.
Casos de teste	**1.** Verificar se o novo nome para a página já é existente. **2.** Verificar se o campo obrigatório, nome, é preenchido. **3.** Verificar se o sistema apresenta o botão "Editar" e "Remover" após a edição de uma página.

Uma vez que a alteração do nome e da descrição de uma página envolve dois campos relativamente pequenos, é preferível o sistema criar o formulário de edição na interface principal (Interface "Página"). Desta forma. é possível ver se já existe alguma página com o nome que se quer dar a uma página anteriormente criada.

Como fluxo alternativo está o caso de uso “Eliminar página”, uma vez que este cenário não representa um caso de uso independente, mas sim um cenário alternativo, como demonstrado na figura 4.

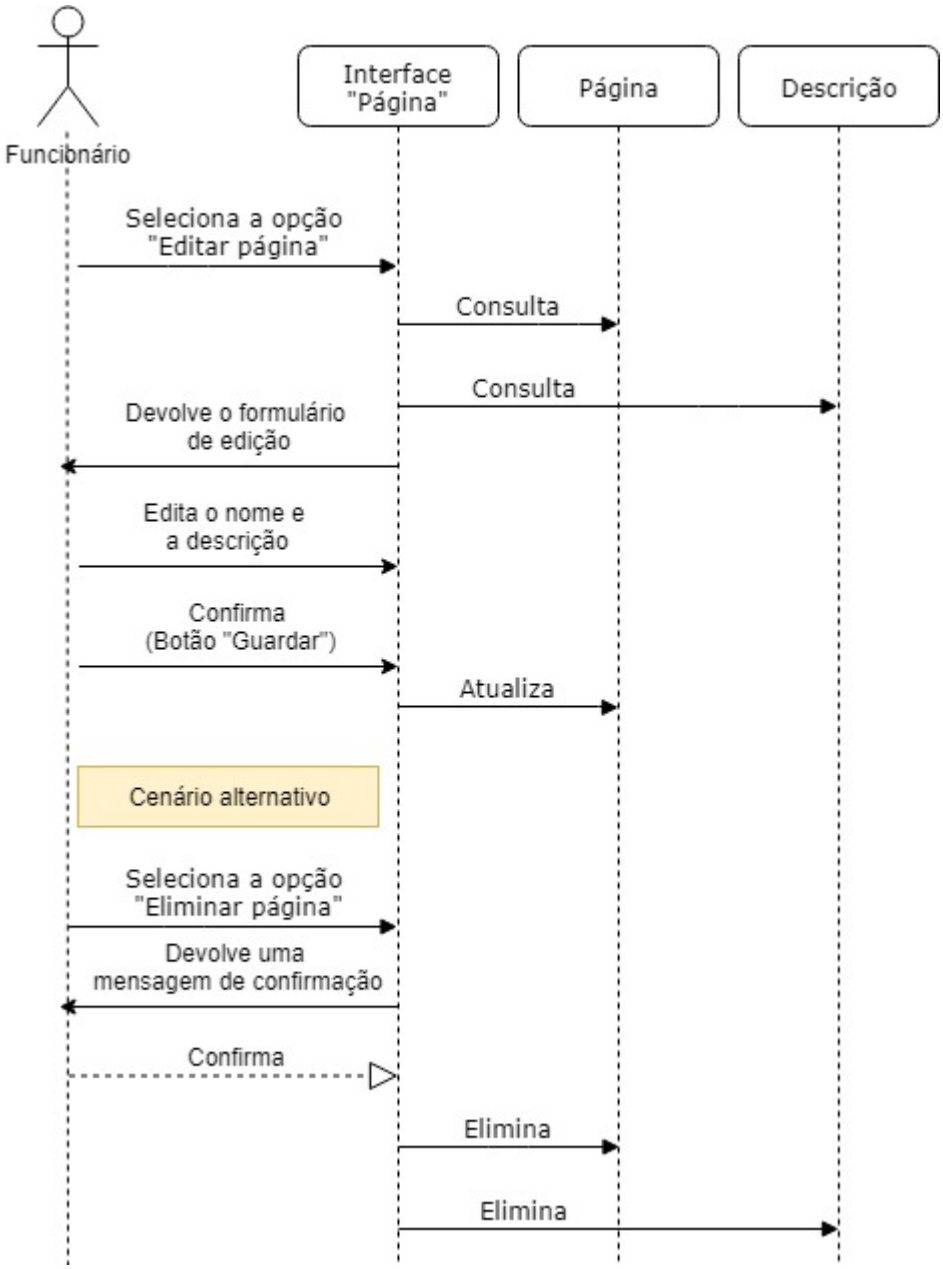

Figura 6 Diagrama de sequência "Editar página"

3.2.3.3. Inserir conteúdo

A classe “Inserir conteúdo” foca-se na inserção de dados nas páginas anteriormente criadas, desta forma será necessário a criação de mais uma tabela que relaciona a página existente na base de dados e o conteúdo inserido.

Nome	Inserir conteúdo
Objetivo	Inserir dados numa página da base de dados
Atores envolvidos	Funcionário
Pré-condição	Login válido
Fluxo principal	**1.** O ator seleciona a opção "Inserir conteúdo" na interface "Conteúdo". **2.** O sistema apresenta DropDownList com o nome das páginas existentes na base de dados. **3.** O ator escolhe a página em que quer introduzir dados. (3A) **4.** O sistema apresenta um campo de edição de texto. **5.** O ator introduz os dados e clica no botão "Submeter". (5A) **6.** O sistema guarda a informação na base de dados.
Fluxo alternativo	**3A.** **a)** O ator escolhe uma página com dados introduzidos. **b)** O sistema apresenta uma mensagem de aviso: "Página já em utilização". **c)** O ator volta a escolher outra página da DropDownList.
	5A. **a)** O ator clica na opção cancelar. **b)** O sistema apresenta uma mensagem de aviso: "Pretende cancelar a introdução de conteúdo? Os dados não serão guardados!". **c)** O ator confirma a ação. **d)** O sistema volta à interface "Conteúdo".
Pós-condição	O sistema atualiza a página e mostra a página da base de dados com uma breve visualização do conteúdo inserido
Casos de teste	**1.** Verificar se a página escolhida na DropDownList não tem informação adicionada. **2.** Verificar se é mostrada na interface "Conteúdos" a página em que foi introduzida informação. **3.** Verificar se o sistema apresenta o botão "Editar", "Ver" e "Remover" após o retorno à interface "Conteúdos".

A descrição deste caso de uso comprova que na nova tabela existirá uma ligação com a tabela das páginas, onde será utilizado o ID das páginas da primeira tabela e um atributo conteúdo onde será armazenada a informação submetida pelo utilizador (Figura 5).

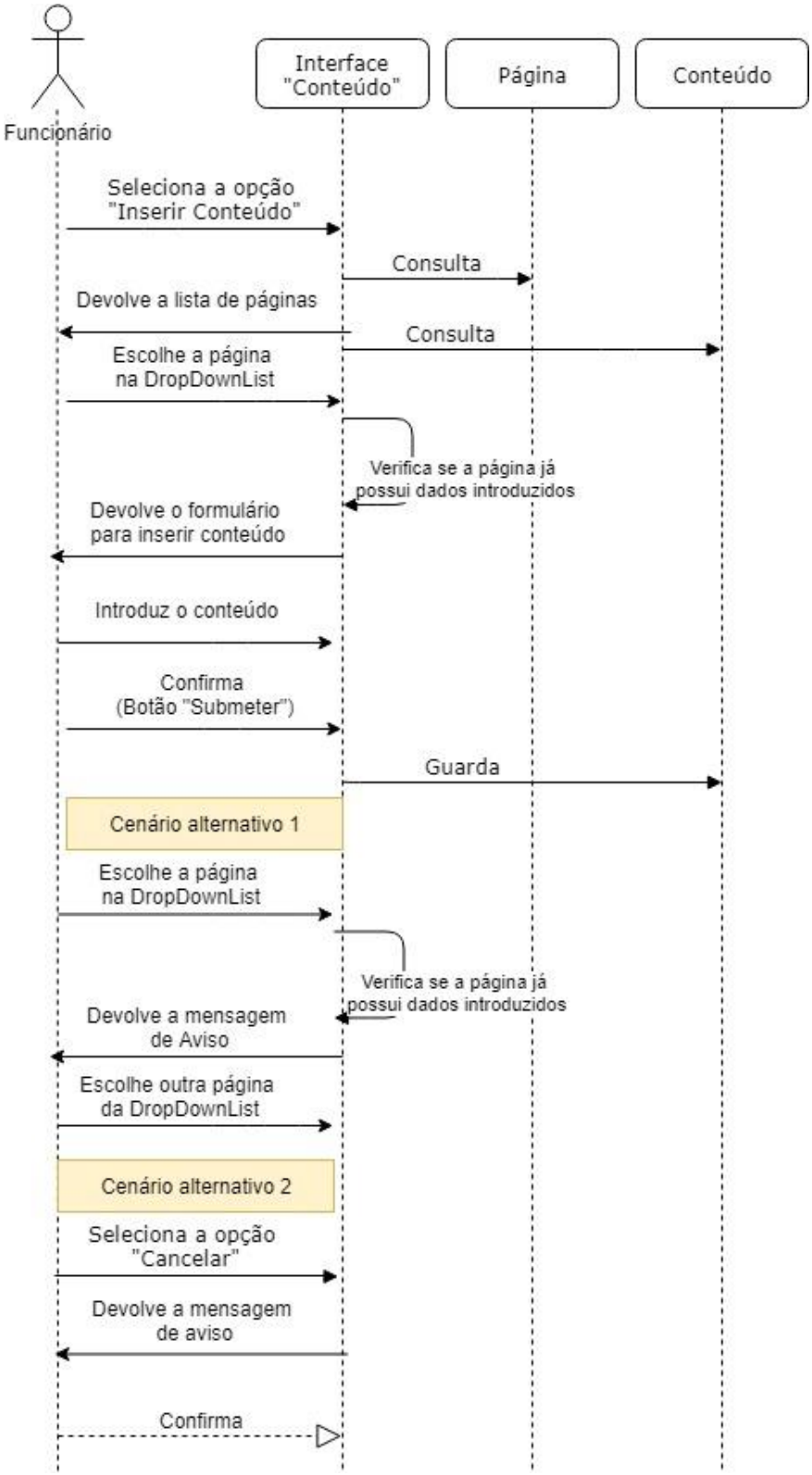

Figura 7 Diagrama de sequência "Inserir Conteúdo"

3.2.3.4. Editar conteúdo

O caso de uso “Editar conteúdo” é quando o utilizador necessita de editar, adicionar ou remover informação numa determinada página. Neste caso, não haverá verificação de páginas duplicadas, uma vez que o utilizador ao escolher a página a editar não terá hipótese de escolher de novo a página, a não ser que este cancele a edição e volte para a interface “Conteúdo”. Apenas haverá a informação se a página escolhida tiver dados para editar ou se for página vazia.

Nome	Editar conteúdo
Objetivo	Editar dados de uma página da base de dados
Atores envolvidos	Funcionário
Pré-condição	Login válido
Fluxo principal	**1.** O ator seleciona a opção “Editar conteúdo” na interface “Conteúdo”. **2.** O sistema apresenta DropDownList com o nome das páginas existentes na base de dados. **3.** O ator escolhe a página em que quer editar. (3A) **4.** O sistema apresenta um campo de edição de texto com os dados anteriormente introduzidos. **5.** O ator edita os dados e clica no botão “Submeter”. (5A) **6.** O sistema guarda a informação na base de dados.
Fluxo alternativo	**3A.** **a)** O ator escolhe uma página sem dados introduzidos. **b)** O sistema apresenta uma mensagem de aviso: “Página em branco. Escolha outra página para editar.”. **c)** O ator volta a escolher outra página da DropDownList.
	5A. **a)** O ator clica na opção cancelar. **b)** O sistema apresenta uma mensagem de aviso: “Pretende cancelar a edição de conteúdo? Os dados não serão guardados!”. **c)** O ator confirma a ação. **d)** O sistema volta à interface “Conteúdo”.
Pós-condição	O sistema atualiza a página e mostra a página da base de dados com uma breve visualização do conteúdo inserido
Casos de teste	**1.** Verificar se é mostrada na interface “Conteúdos” a página em que foi editada a informação. **2.** Verificar se o sistema apresenta o botão “Editar”, “Ver” e “Remover” após o retorno à interface “Conteúdos”.

A figura 6 demonstra como o utilizador interage com o sistema, onde o sistema pede que seja indicada a página com conteúdo a editar, e os cenários alternativos, caso o utilizador escolha uma página em branco ou cancela a operação.

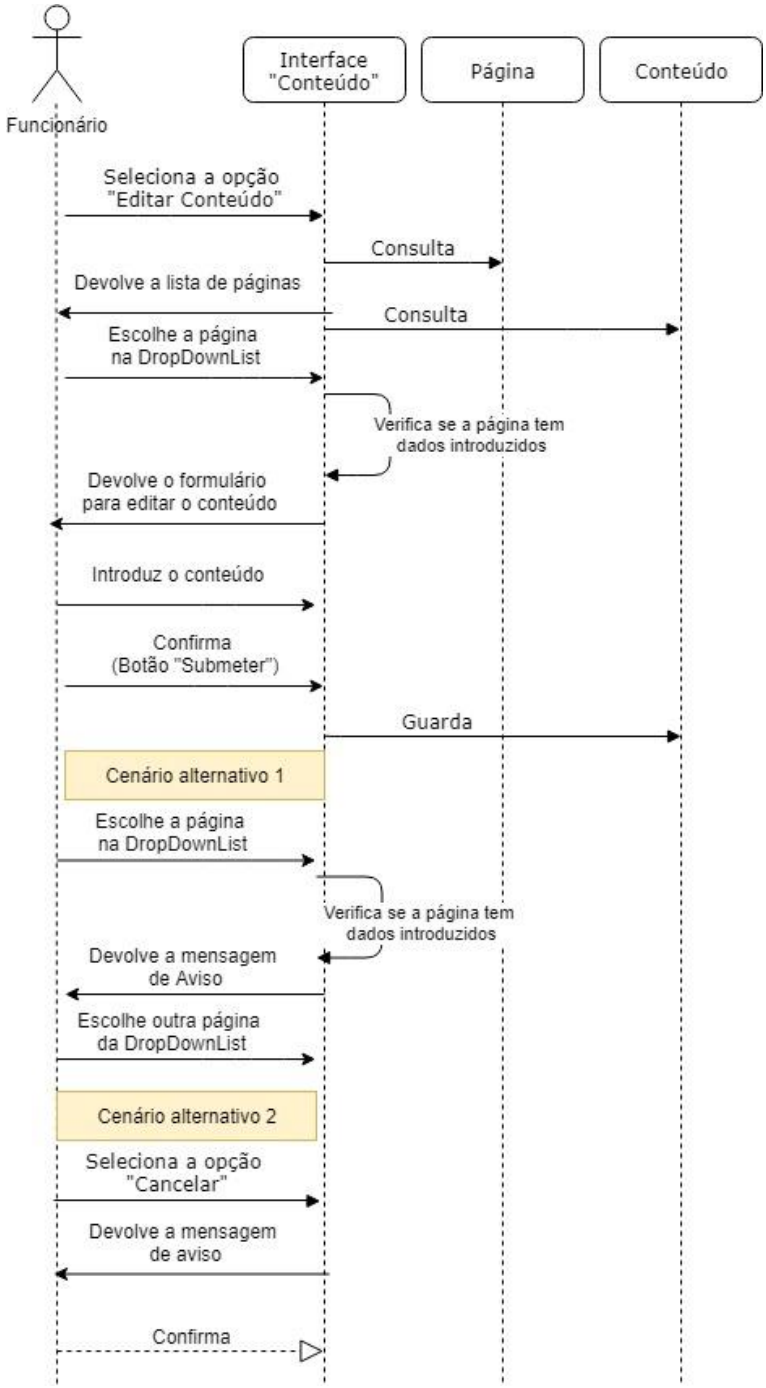

Figura 8 Diagrama de sequência "Editar conteúdo"

3.2.3.5. Ver conteúdo

Neste caso de uso, o utilizador ao escolher uma página da *DropDownList* ser-lhe-á apresentada uma área de texto estática, com a informação armazenada na base de dados onde não poderá inserir ou apagar dados.

Nome	Ver conteúdo
Objetivo	Ver dados de uma página da base de dados
Atores envolvidos	Funcionário
Pré-condição	Login válido
Fluxo principal	**1.** O ator seleciona a opção "Ver conteúdo" na interface "Conteúdo". **2.** O sistema apresenta DropDownList com o nome das páginas existentes na base de dados. **3.** O ator escolhe a página em que quer ver. (3A) **4.** O sistema apresenta um campo de edição de texto estático com os dados anteriormente introduzidos. **5.** O ator clica no botão "Voltar". **6.** O sistema volta para a interface "Conteúdo".
Fluxo alternativo	**3A.** **a)** O ator escolhe uma página sem dados introduzidos. **b)** O sistema apresenta uma mensagem de aviso: "Página em branco. Escolha outra página para ver.". **c)** O ator volta a escolher outra página da DropDownList.
Pós-condição	O sistema atualiza a página e mostra a página da base de dados com uma breve visualização do conteúdo inserido
Casos de teste	**1.** Verificar se é mostrada na interface "Conteúdos" a página que foi vista. **2.** Verificar se o sistema apresenta o botão "Editar", "Ver" e "Remover" após o retorno à interface "Conteúdos".

Neste caso o utilizador apenas escolhe uma página e o sistema devolve o texto estático ou um aviso caso a página esteja em branco, demonstrado na figura 7.

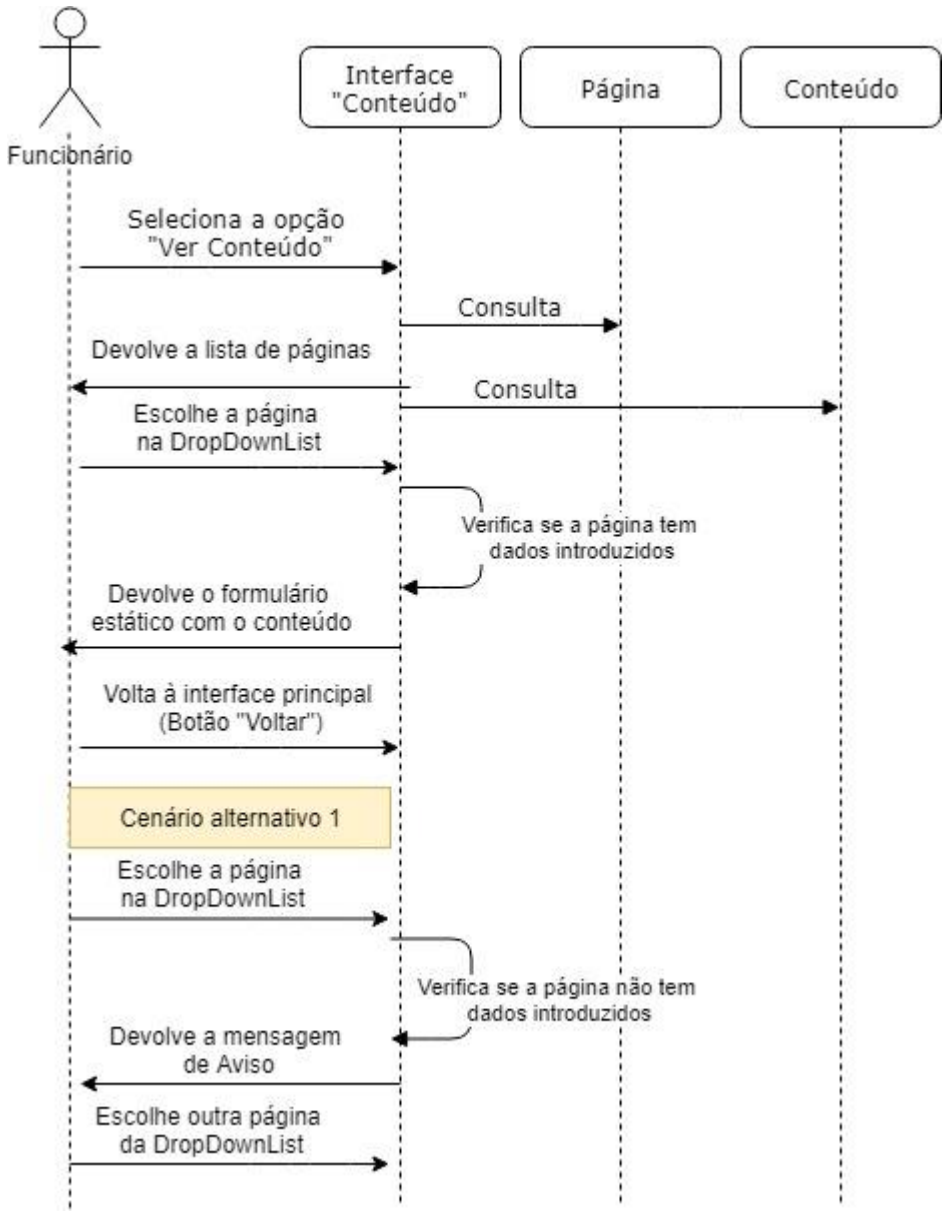

Figura 9 Diagrama de sequência "Ver conteúdo"

3.2.3.6. Eliminar conteúdo

O caso de uso “Eliminar conteúdo” apenas serve para o utilizador remover por completo o conteúdo de uma página existente. Desta maneira, a página em questão irá ficar sem conteúdo visível para, posteriormente, poder ser introduzido outro conteúdo.

Nome	Eliminar conteúdo
Objetivo	Eliminar dados de uma página da base de dados
Atores envolvidos	Funcionário
Pré-condição	Login válido
Fluxo principal	**1.** O ator seleciona a opção “Eliminar conteúdo” numa das páginas presentes na interface “Conteúdo”. **2.** O sistema apresenta uma mensagem de aviso: “Deseja eliminar o conteúdo desta página? O processo não é reversível!”. **3.** O ator confirma a opção. (3A) **4.** O sistema volta à interface “Conteúdo”.
	3A. **a)** O utilizador clica no botão “Cancelar”. **b)** O sistema retorna à interface “Conteúdo”.
Pós-condição	O sistema atualiza a página e não mostra a página na interface “Conteúdo”.
Casos de teste	**1.** Verificar se não é mostrada na interface “Conteúdos” a página eliminada.

Neste caso (Figura 8), o utilizador ao escolher a opção para eliminar, o sistema irá devolver uma mensagem, onde pode escolher a opção confirmar, e assim apagar o conteúdo da página, ou cancelar, mantendo o conteúdo.

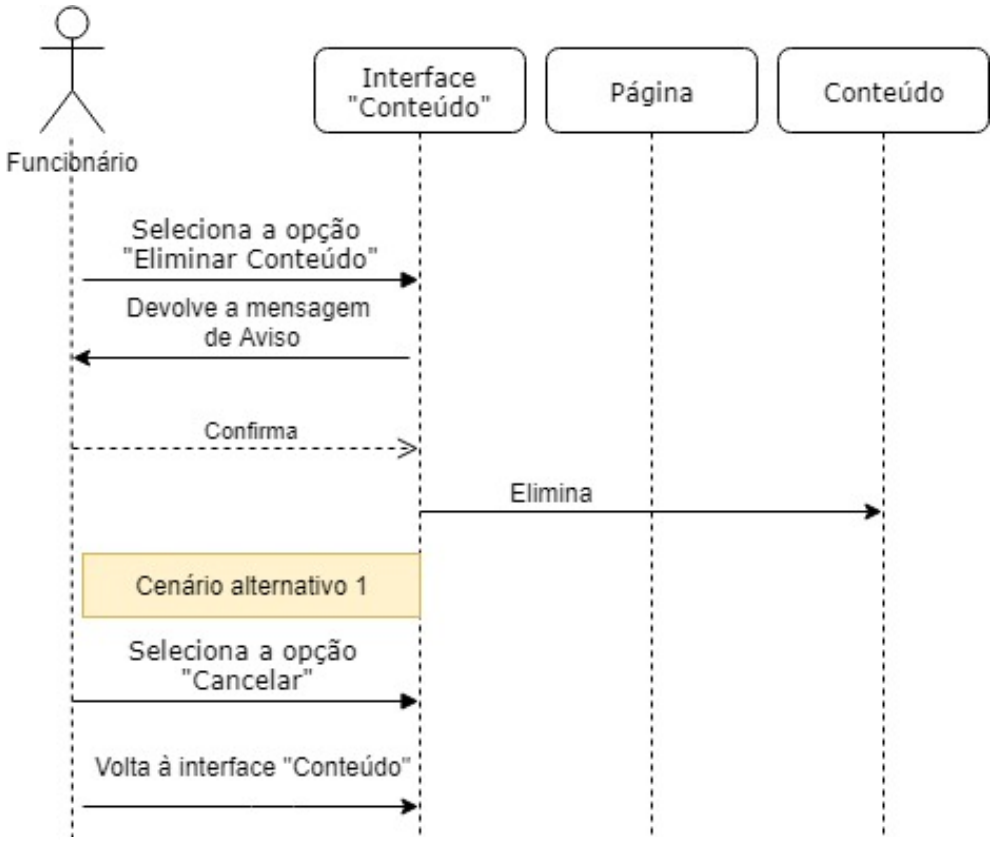

Figura 10 Diagrama de sequência "Eliminar conteúdo"

3.2.4. Diagrama de classes

O objetivo dos pontos anteriores serviu para identificar as interfaces e classes necessárias. Após a identificação das classes, foi definida a criação de duas classes: a classe “Página” e a classe “Conteúdo”. As referidas classes contêm os atributos necessários para a manipulação de dados e para a criação do modelo ER (Figura 9).

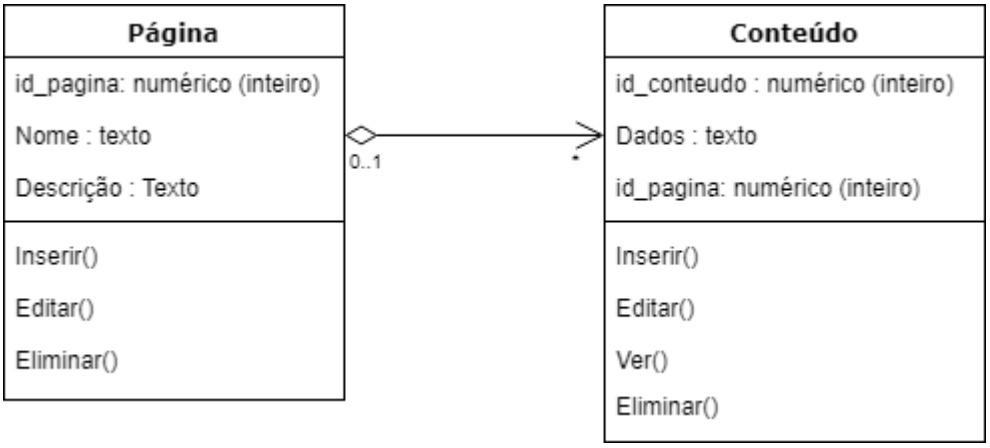

Figura 13 Diagrama de classes

3.2.5. Modelo ER e semântica de dados

O modelo entidade relacionamento, criado a partir do diagrama de classes, surge com o sentido de clarificar o relacionamento entre classes, identificar chaves primárias e estrangeiras, os tipos de dados e os seus tamanhos.

3.2.5.1. Modelo entidade relacionamento

O modelo entidade relacionamento é necessário no sentido de justificar a criação de tabelas relacionais com o intuito de servirem de modelo para a criação do diagrama da base de dados. Neste caso, como só há a existência de duas tabelas necessárias para o projeto, foram adicionadas no modelo entidade relacionamento já existente na empresa, tirando assim proveito de outras classes com a finalidade de utilização do sistema de gestão de conteúdos pelos administradores existentes na empresa.

O diagrama criado, por um destes administradores, que se apresentam na figura 10:

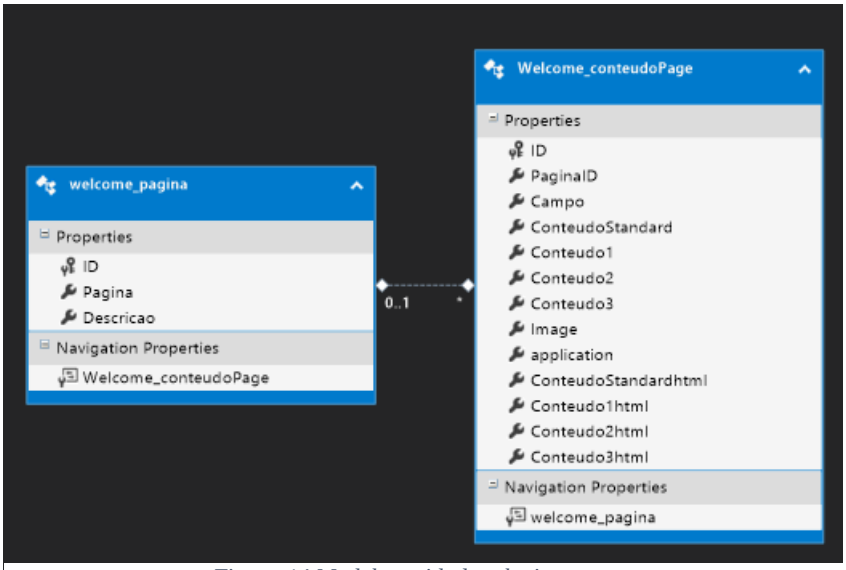

Figura 14 Modelo entidade relacionamento

Este modelo contém duas tabelas cujo nome foi atribuído pelo administrador, onde a tabela *welcome_pagina* é a tabela anteriormente referida como "Página" e a tabela *welcome_conteudoPage* referida como "Conteúdo". Nesta última tabela fora adicionado mais atributos do que aqueles que irão ser utilizados por uma questão de flexibilidade, portanto só irão ser utilizados os atributos: ID, PaginaID, Campo e ConteudoStandard.

3.2.5.2. Dicionário de dados

Neste capitulo, pretende-se descrever os campos das duas classes, apresentando o tipo de dados, valores válidos e os seus formatos. Também se irão descrever os métodos disponíveis em cada uma das classes. Na tabela 1 é feito o dicionário de dados para os campos da classe *welcome_pagina*, e a as suas operações na tabela 2. Na tabela 3 é feito o dicionário de dados para os campos da classe *welcome_conteudoPage* e as suas operações na tabela 4.

welcome_pagina					
Nome	Tipo de dados	Tamanho	Descrição	Formato	Restrições

ID	Integer	5	Chave primária da tabela	Até 5 dígitos	Maiores que 0. Não nulo. Único.
Página	Varchar2	20	Nome da página	Até 20 caracteres	Não nulo. Único
Descrição	Varchar2	20	Descrição para a página	Até 20 caracteres	Pode ser nulo

Tabela 1 Dicionário de dados da classe welcome_pagina

Operações: Classe "welcome_pagina"	
Inserir()	Conjunto de operações (set) que permitem inserir nos campos da classe.
Editar()	Conjunto de operações (get e set) que permitem manipular os campos da classe.
Eliminar()	Conjunto de operações (get e set) que permitem remover dados dos campos da classe.

Tabela 2 Operações da classe welcome_pagina

welcome_conteudoPage					
Nome	Tipo de dados	Tamanho	Descrição	Formato	Restrições
ID	Integer	5	Chave primária da tabela	Até 5 dígitos	Maiores que 0. Não nulo. Único.
PaginaID	Integer	5	Chave estrangeira que faz referencia à tabela *welcome_pagina*	Até 5caracteres	Maiores que 0. Não nulo. Único.
Campo	Varchar2	20	Descrição para o conteúdo	Até 20 caracteres	Pode ser nulo.
ConteudoStandard	Varchar2	Max	Conteúdo a ser apresentado na aplicação	Máximo possível de caracteres	Não nulo.

Tabela 3 Dicionário de dados da classe welcome_conteudoPage

Operações: Classe "welcome_conteudoPage"	
Inserir()	Conjunto de operações (set) que permitem inserir nos campos da classe.
Editar()	Conjunto de operações (get e set) que permitem manipular os campos da classe.
Ver()	Conjunto de operações (get) que permitem ver os dados dos campos da classe.
Eliminar()	Conjunto de operações (get e set) que permitem remover dados dos campos da classe.

Tabela 4 Operações da classe welcome_conteudoPage

3.3. Ferramentas e tecnologias utilizadas

Neste capitulo serão detalhadas as tecnologias para o desenvolvimento do projeto, sendo elas distribuídas gratuitamente e utilizadas em diversos campos. Deste modo há documentação especifica de forma a ajudar no desenvolvimento.

3.3.1. Visual Studio

O Visual Studio (Figura 11) é um IDE da Microsoft que, possibilita o desenvolvimento de *software* e aplicações *web* para a *framework* .NET e para linguagens mais utilizadas como Java, C, C++, C#.

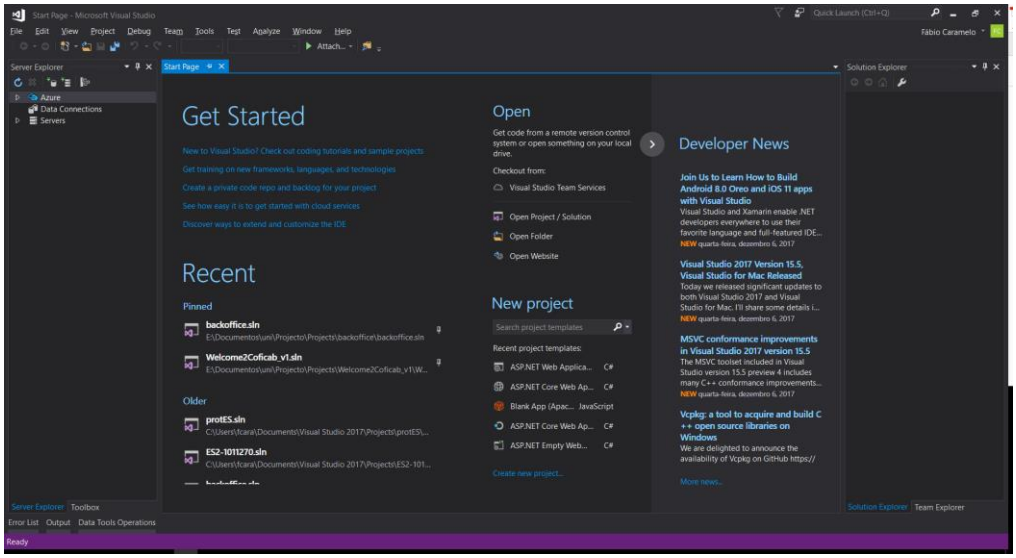

Figura 15 Microsoft Visual Studio

Este IDE é o ideal devido à inclusão do NuGet que possibilita a criação e partilha de bibliotecas e *Dynamic Link Library* (DLLs) em formato de pacotes possibilitando às empresas disponibilizarem *frameworks* de código aberto de forma simples sem ter que lançar um IDE especifico para a sua tecnologia.

O instalador deste IDE possui, também, a característica de identificar se o computador pessoal já tem instalado ferramentas necessárias para as diferentes *frameworks* funcionarem sem problemas.

3.3.2. C#

É uma linguagem de programação, desenvolvida pela Microsoft em 2000, para o desenvolvimento de soluções com base na *framework* .NET para substituir o Java. Esta

linguagem orientada para objetos é baseada nas outras linguagens de alto nível de forma a resolver os problemas destas linguagens. Esta linguagem foi utilizada para o desenvolvimento do CMS.

3.3.3. HTML

É a linguagem mais simples para a construção de uma página *web*. A sua utilização é simples e básica dependendo da caracterização escolhida através das marcações estruturais da linguagem. Esta tecnologia foi utilizada no desenvolvimento da aplicação móvel e CMS.

3.3.4. CSS

Esta tecnologia é uma linguagem *stylesheet* utilizada para descrever a apresentação de um documento HTML ou Extensible Markup Language (XML). Desta forma o CSS descreve como é que os elementos de uma página web são visualizados. Esta tecnologia foi utilizada no desenvolvimento da aplicação móvel e CMS.

3.3.5. JavaScript

É uma linguagem de programação de alto nível, que utilizada em conjunto com o HTML e CSS possibilita a criação de conteúdo para a Internet. Esta linguagem permite tornar as páginas *web* mais interativas e destacar conteúdo mais intuitivamente ao utilizador. Esta tecnologia foi utilizada no desenvolvimento da aplicação móvel.

3.3.6. jQuery

É uma biblioteca da linguagem JavaScript que interagem com as marcações HTML para a interpretar os scripts no browser do utilizador. Permite a criação de animações, manipular eventos e desenvolver aplicações AJAX, que é uma técnica para o desenvolvimento web para a criação de aplicações web interativas que atualiza assincronamente informação entre o lado do servidor e o lado do utilizador, sem que este tenha a necessidade de atualizar a página web onde se encontra. Esta tecnologia foi utilizada no desenvolvimento da aplicação móvel.

4. Desenvolvimento

Nos pontos anteriores foram determinadas as ferramentas e tecnologias a utilizar para a elaboração do projeto proposto, de forma a que os intervenientes estejam de acordo com os procedimentos a efetuar.

Neste capitulo serão descritos os passos levados para o desenvolvimento das duas componentes do trabalho, o CMS para a parte administrativa e a aplicação móvel para ser disponibilizada ao publico.

4.1. Desenvolvimento CMS

O desenvolvimento do CMS Welcome2Coficab, como descrito acima, será feito utilizando a *framework* da ASP.NET MVC. Desta maneira, é necessário definir as variáveis a utilizar dentro do Model, as operações a efetuar dentro dos Controllers e por fim a criação das interfaces apresentadas pelas Views [9].

4.1.1. Criação dos *scripts Model*

O *Model* é a parte da *framework* que utiliza os campos das tabelas, criadas na base de dados, para a criação das variáveis que irão ser utilizadas nos *Controllers*.

Para a criação do *Model* foi necessário a utilização das seguintes bibliotecas da *framework*:

- System;
- System.Collections.Generic;

Estas bibliotecas servem para chamar instâncias que possam criar uma coleção com variáveis, para serem utilizadas pelo *Controller* de forma segura.

O primeiro *Model* criado será a partir da tabela que identifica as páginas (*welcome_pagina*). Neste Model, os campos da tabela foram transformados numa coleção genérica que contem variáveis a serem utilizadas pelo *Controller*. Desta forma, as variáveis definem-se criando uma nova coleção:

```
public partial class welcome_pagina
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public welcome_pagina()
        {
            this.Welcome_conteudoPage = new HashSet<Welcome_conteudoPage>();
        }

        public int ID { get; set; }
        public string Pagina { get; set; }
        public string Descricao { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Welcome_conteudoPage> Welcome_conteudoPage {
get; set; }
    }
```

A segunda coleção é criada a partir da tabela *welcome_conteudoPage* que será desenhada da mesma maneira da coleção da outra tabela apresentando-se da seguinte forma:

```
public partial class Welcome_conteudoPage
    {
        public int ID { get; set; }
        public Nullable<int> PaginaID { get; set; }
        public string Campo { get; set; }
        public string ConteudoStandard { get; set; }
        public string Conteudo1 { get; set; }
        public string Conteudo2 { get; set; }
        public string Conteudo3 { get; set; }
        public byte[] Image { get; set; }
        public Nullable<decimal> application { get; set; }
        public string ConteudoStandardhtml { get; set; }
        public string Conteudo1html { get; set; }
        public string Conteudo2html { get; set; }
        public string Conteudo3html { get; set; }

        public virtual welcome_pagina welcome_pagina { get; set; }
    }
```

Com o *Model* criado, é agora possível utilizar as coleções nos *Controllers* de forma a criar as operações funcionais.

4.1.2. Criação dos *Controllers*

A *framework* MVC traz, por omissão, *Controllers* que podem ser utilizados para dar acesso ao utilizador às páginas construídas. Para este projeto apenas foi utilizado o *Controller AuthorizeApplicationAccess*, onde foi definida a ligação à base de dados

interna da Coficab e editada de forma a que apenas utilizadores com determinados atributos possam aceder ao CMS:

- Secção: *IT*;
- Role: *Admin*;
- Permissão do tipo: *List* e *Add*.

A partir desta edição, para a autenticação do utilizador, foi feita a criação dos Controllers *ConteudosPageController* e *ConteudosDescriptionController*. Para que ambos os Controllers funcionem adequadamente e corretamente foram utilizadas as seguintes bibliotecas:

- backoffice.Models;
- PagedList;
- System;
- System.Data.Entity;
- System.Linq;
- System.Net;
- System.Web.Mvc;

A biblioteca *backoffice.Models* foi utilizada para chamar o *Model* criado anteriormente, para que possa ser utilizadas as coleções.

A biblioteca *PagedList* permitiu a organização da interface apresentada, escolher a organização da informação consoante o nome dado a uma página ou descrição e mostrar o número de itens por página.

As restantes bibliotecas utilizadas como *namespaces*, são recipientes que fornecem um contexto para os itens que estes armazenam. Nos *Controllers* criados, estas bibliotecas são utilizadas para medidas de segurança e para conceder ligação à base de dados da empresa.

4.1.2.1. Criação do *Controller ConteudosPageController*

Este primeiro *Controller*, foi criado para a geração das páginas onde depois de criada o conteúdo irá ser inserido, utilizando um outro *Controller*. A sua construção foi feita a

pensar em como se relacionaria com as páginas do menu da aplicação móvel e desta maneira foi decidido que para a sua criação seria apenas necessário o nome equivalente ao daquele que iria aparecer na aplicação e uma breve descrição.

Para a criação da página é necessário verificar o *Model* a utilizar, caso este seja válido é então criada a página pelo utilizador:

```
//Create
public ActionResult Create(welcome_pagina model)
        {
            if (ModelState.IsValid)
            {
                DB.welcome_pagina.Add(model);
                DB.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(model);
}
```

Após a criação da página o utilizador tem ainda a opção de a editar ou eliminar. Desta forma, é possível evitar enumeras páginas com o mesmo nome que possam depois não ser chamadas de forma correta pela aplicação móvel.

4.1.2.2. Criação do *Controller ConteudosDescriptionController*

Utilizando o mesmo processo para a criação do *Controller* anterior, este irá servir para inserir conteúdo dentro das páginas anteriormente criadas.

No método de criação o código utilizado identifica se o *Model* é válido e nesse ponto irá possibilitar a introdução de dados com a identificação da página criada pelo *Controller* anterior, tendo também a possibilidade para editar ou eliminar.

4.1.3. Criação das aplicações *View*

Neste ponto é descrito como as Views foram criadas para fazer a apresentação e inserção da informação. As Views, como estão ligadas aos Controllers, significam que cada Controller tenha uma View especifica e cada View terá quatro páginas que permitem fazer a apresentação da página de Index, para visualizar as páginas criadas ou a informação nelas contida, e outras três páginas para a inserir, editar ou eliminar.

Também foram utilizadas Views que, por defeito, compõe a *framework* sendo ela a *Index.cshtml* e *Login.cshtml* responsável pela apresentação da página inicial e de *login*.

4.1.3.1. *Views* para o Controller *ConteudosPageController*

As Views criadas para este *Controller* são identificadas pela utilização da referência ao Model *welcome_pagina*. (`@model backoffice.Models.welcome_pagina`).

4.1.3.1.1. *View Index*

Esta *View* é utilizada para dispor as opções que o utilizador pretende efetuar para adicionar, editar ou remover uma página da aplicação móvel à sua escolha, podendo filtrar as páginas por nome. Nesta *View*, o utilizador pode criar uma página para a aplicação nova ou selecionar uma já existente, alterar a sua descrição e o seu nome ou remover essa página.

O botão *Create* irá redirecionar o utilizador para a respetiva área de criação, enquanto que os botões *Edit* e *Delete* irão deixar o utilizador alterar os campos na *View Index*, ou eliminar após uma mensagem de confirmação, uma vez que as páginas apenas contêm o campo Nome e Descrição. Foi decidido não criar uma *View* própria para a edição e remoção face aos poucos atributos e detalhes para especificar uma página.

4.1.3.1.2. *View Create*

Esta *View* é onde o utilizador irá introduzir o nome de uma nova página e uma descrição para a mesma.

4.1.3.2. *Views* para o *Controller ConteudosDescriptionController*

As seguintes *Views* foram criadas de forma a que o utilizador tenha uma boa perspetiva daquilo que esteja a criar, dai ter sido utilizado para a edição de texto um editor HTML WYSIWYG, o TinyMCE [10]. Com este editor o utilizador pode fazer formatações de texto, inserir imagens armazenadas no computador, vídeos e *links* como se fosse um editor de texto normal, podendo fazer uma pré-visualização de como fica o texto formatado ou se preferir uma pré-visualização com marcações em HTML.

Estas *Views* utilizam uma referencia ao Model a que estão ligadas (`@model backoffice.Models.Welcome_conteudoPage`).

4.1.3.2.1. *View Index*

Nesta *View*, é possível a criação de conteúdo para as páginas criadas anteriormente, assim como a sua edição ou a remoção do conteúdo de uma página.

As operações de *Create* e *Edit* nesta *View* irão reencaminhar o utilizador para as áreas especificas, apenas com a exceção da *View Delete* que será apresentada nesta mesma página.

4.1.3.2.2. *View Create*

Nesta *View* o utilizador escolhe a página criada anteriormente, para adicionar conteúdo a partir do TinyMCE *Editor* utilizado com uma *TextArea*, invocado pelos seguintes *scripts*:

```
<script src="~/Scripts/tinymce/tinymce.js"></script>
<script src="~/Scripts/tinymce/main.js"></script>
```

O *tinymce.js* é utilizado para substituir a *TextArea* predefinida pela *framework*, no que toca à estética e comportamento do editor e o *main.js* que fornece os *plugins* necessários para a formatação do texto e introdução de elementos como imagens, vídeos e *links*.

Após o utilizador inserir o conteúdo na página submete-o e irá ser armazenado na base de dados

4.1.3.2.3. *View Edit*

Esta *View* é uma página idêntica à anterior, reencaminhando o utilizador para a página de edição, para que consiga editar o conteúdo anteriormente inserido numa página.

Da mesma maneira que o TinyMCE é invocado, substituindo a *TextArea* predefinida o conteúdo é carregado para o editor podendo ser modificado em diferentes aspetos, dependendo da vontade do utilizador.

4.1.3.2.4. *View Delete*

Esta *View* apenas fornece uma mensagem ao utilizador se realmente pretende eliminar o conteúdo da página e o utilizador poderá então remover a informação existente na página.

4.2. Desenvolvimento da aplicação Welcome2Coficab

Com o sistema de CMS finalizado, o desenvolvimento da aplicação móvel para as plataformas móveis é o próximo passo. Neste processo foram utilizados vários componentes de código aberto uma vez que a *framework* utilizada, Apache Cordova [11], é muito limitada com o *design* que fornece aos seus utilizadores.

Todo o *design* da aplicação móvel foi concebido por um designer da Coficab, fornecendo posteriormente os códigos para as cores e tipo de letra.

4.2.1. Menu lateral

Um dos problemas causados pelo Apache Cordova, como mencionado em cima, é a falta de *designs* que possibilitam criar uma aplicação fácil de utilizar e perceber, e como esta aplicação iria ser utilizada pelo publico em geral, foi decidido utilizar o *design* geral que quase todas as aplicações utilizam; uma visualização quase completa do ecrã do telemóvel e as páginas guardadas num menu lateral utilizando um movimento de deslize horizontal para o abrir e fechar, ou tocar no ícone do menu lateral.

Para conceber esta ideia foi utilizado o plugin *JQuery.MMenu* [12], que possibilita menus encadeados e personalizar o menu conforme seja necessário.

Na figura 12, podemos ver o menu escondido e na figura 13 as opções dentro do menu.

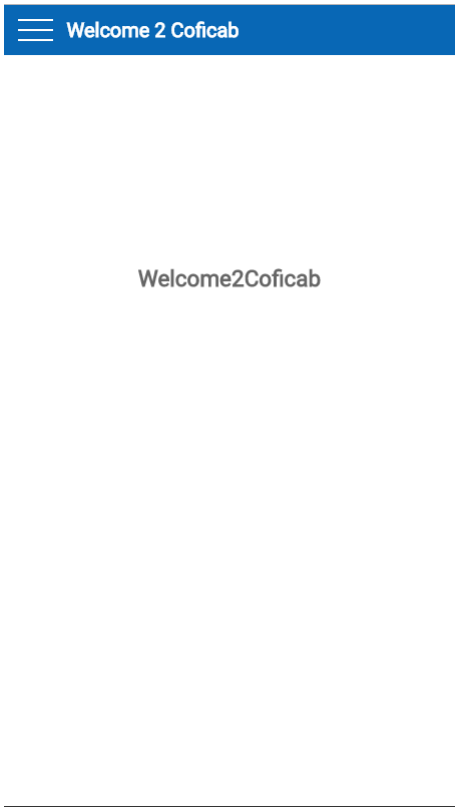

Figura 16 Aplicação com menu escondido

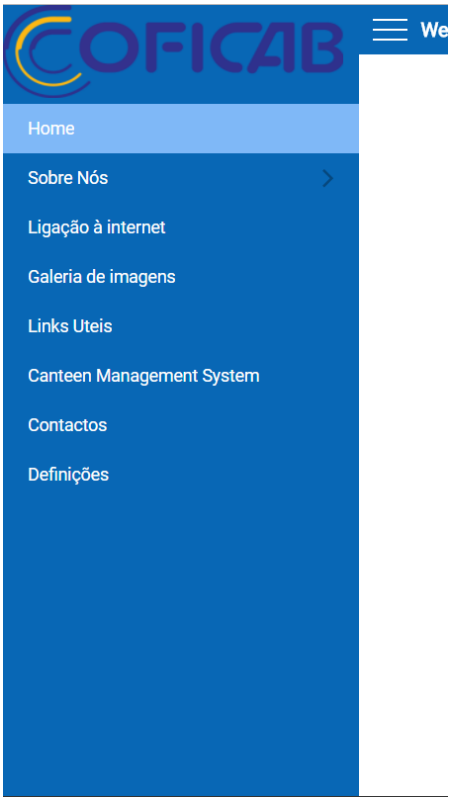

Figura 17 Aplicação com o menu exposto

O menu também possibilita itens em cascada, podendo, assim, criar subitens e consequentemente não sobrecarregar o menu com opções e ficar visualmente pesado, como demostrado na figura 14 e figura 15.

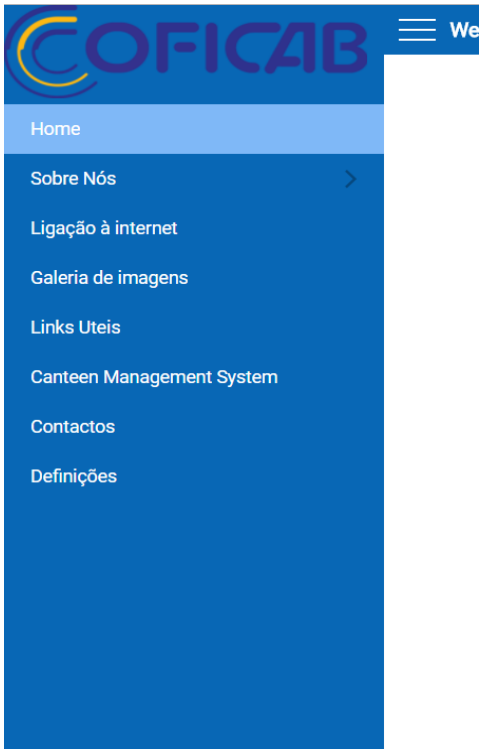

Figura 14 Aplicação no menu principal

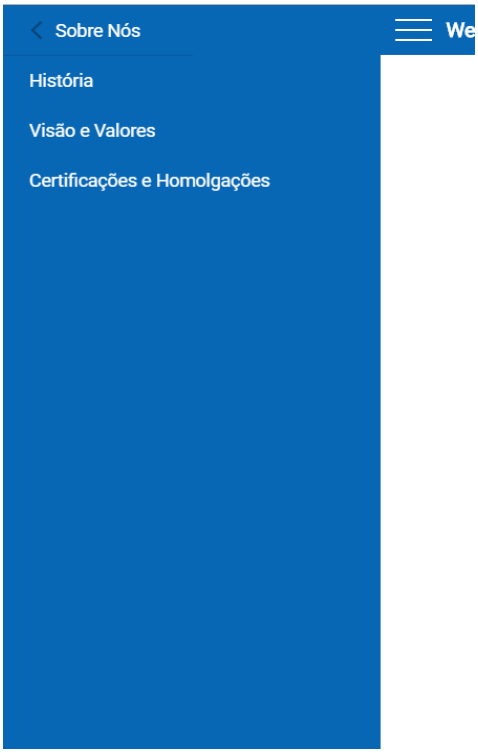

Figura 15 Aplicação no submenu "Sobre Nós"

4.2.2. Galeria de imagens

Um dos requisitos propostos pela empresa era a integração de uma galeria de imagens de forma a que o utilizador conhecesse alguns dos espaços interiores e exteriores das plantas.

A incorporação da galeria foi feita de forma a mostrar as imagens em miniatura. O utilizador ao carregar numa das fotos esta iria expandir mostrando mais detalhe. Desta maneira, as imagens encontram-se dentro de uma célula (figura 16) que quando é aberta cria uma nova célula com a imagem expandida (figura 17) e ao fechar colapsa-se a célula expandida e, assim, a galeria volta ao seu formato original.

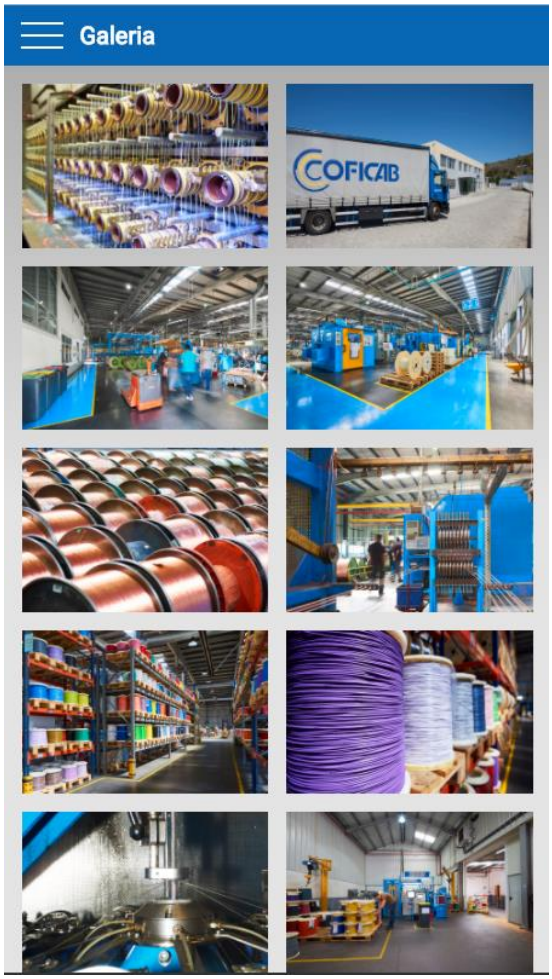

Figura 16 Galeria de imagens em grelha

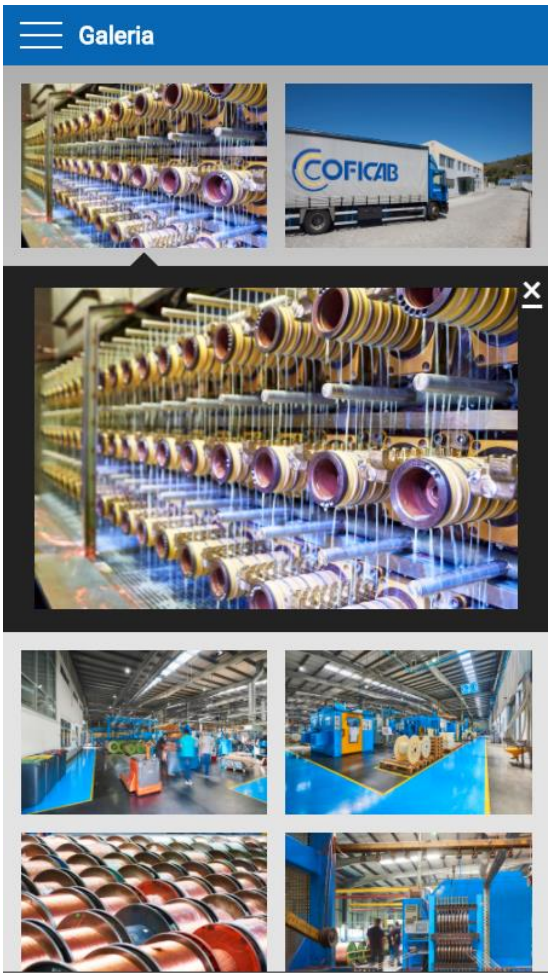

Figura 17 Galeria de imagens com célula expandida

4.2.3. Páginas Adicionais

O resto das páginas presentes na aplicação teriam um carácter informativo para o utilizador, onde estaria a informação anteriormente submetida no CMS. Desta forma o utilizador poderia ler diversas informações sobre a empresa onde constam as homologações, certificações, história da empresa e a ligação para o Canteen Management System.

A informação exposta nas páginas seria chamada via um *webservice* criado por um funcionário da empresa para que fossem implementadas funções para exibir a informação nas diversas páginas da aplicação.

5. Verificação e validação

Neste capitulo serão descritos testes feitos às duas partes do projeto tendo em conta a sua utilização normal, pela parte administrativa para o CMS e por um simples utilizador para a parte da aplicação móvel e a integração de ambas as partes.

5.1. Validação do CMS

Para o CMS, os testes efetuados foram baseados na sua utilização normal, desde o *login* por parte do utilizador, criação das páginas e inserção de conteúdo.

O *login*, uma vez que era disponibilizado por um dos administradores da empresa, funciona desde que o utilizador tenha o *User Role* e permissões atualizadas no seu perfil da base de dados, conseguindo entrar na página do CMS.

A criação de uma página no CMS é um processo simples em que o utilizador após fazer *login*, insere um nome para a página, que deverá ser equivalente a uma das páginas existente na aplicação, uma descrição simples para essa página e faz a submissão. A edição do nome e da descrição da página é apenas a alteração dos dados introduzidos fazendo a gravação na base de dados. A eliminação de uma das páginas é possível e apenas é mostrada uma mensagem ao utilizador, para que este confirme ou cancele a operação.

A introdução do conteúdo nas páginas também foi testada, para avaliar a forma como o editor TinyMCE fazia a formatação de texto e a incorporação de imagens e a forma como os dados introduzidos eram guardados na base de dados. Estes dados eram posteriormente editados, com a opção de edição de conteúdo onde iriam aparecer de forma formatada onde se poderia remover ou adicionar informação. A eliminação destes dados estava disponível e, da mesma forma como a eliminação da página, iria ser apresentada uma mensagem ao utilizador para confirmar ou cancelar a operação.

Todos os aspetos do CMS foram verificados e validados pelo orientador da empresa, onde este apenas apresentou algumas duvidas sobre as funcionalidades e utilização do editor TinyMCE, que foram esclarecidas.

5.2. Validação da aplicação

Como a aplicação seria destinada a visitantes da Coficab, esta foi testada de forma a retratar a utilização normal de uma aplicação móvel a partir da sua instalação, e execução das funcionalidades.

A instalação nas plataformas móveis apenas foi feita com sucesso na plataforma Android 4.2 ou superior, devido à falta de material físico de teste para iOS, uma vez que para compilar uma IPA é obrigatório a utilização de um computador Mac com sistema operativo macOS, o IDE XCode e uma conta de *Apple Developer*. Desta forma, foi impossível a compilação ou emulação para iPhone uma vez que não se cumpriam nenhum destes requisitos. Uma vez que o *Apache Cordova* compila a mesma aplicação para ambas as plataformas foi apenas testada em ambiente *Android* a partir da compilação de um ficheiro *Android Application Package* (APK). Após a instalação da aplicação num *smartphone* foi testada em termos de design e funcionalidades, onde o menu e a galeria de imagens passaram com sucesso nos testes realizados.

A informação que seria exibida através da utilização de *webservice* não foi apresentada. A não apresentação da informação, já disponível no CMS, poderia estar relacionada com o mau estruturamento do *webservice*, a não integração com o *Apache Cordova* ou uma falha no código da aplicação móvel.

6. Conclusão

Este projeto consistiu em duas partes, o CMS e a aplicação móvel, que utilizam algumas tecnologias trabalhadas durante os anos da licenciatura e outras novas, em que foi necessário o estudo e reflexão sobre a utilização das mesmas. Porém, a introdução de novas tecnologias no conhecimento de um engenheiro informático é uma vantagem. Mesmo que no inicio fosse difícil a sua implementação, essa informação foi retida para num futuro próximo poder aprofundar estes e outros conhecimentos.

Como foi referido no capitulo anterior, a aplicação móvel apesar de estar funcional não ficou totalmente completa, uma vez que não recebia a informação através do *webservice*. Porém, estes fatores contribuíram para a interiorização daquilo que é o mercado de trabalho na área da informática e permitiu reconhecer que, numa empresa multinacional em crescimento, o trabalho é árduo, mas com perseverança e dedicação é possível alcançar os objetivos propostos.

Apesar de este projeto não ter tido o melhor desfecho no que toca à sua utilização, para mim, foi uma experiência fundamental como aluno, para perceber a importância do conhecimento teórico envolvendo a analise de sistemas, e para adquirir experiência como programador, mostrando a importância constante da aprendizagem num percurso empresarial e pessoal.

7. Bibliografia

[1] SalesForce Developers, *Native, HTML5 or Hybrid: Understanding Your Mobile Application Development Options*. [Em linha]. Consultado em Junho de 2017. Actual. Jun. 2016. Disponível em: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[2] Bootstrap. *Bootstrap*. [Em linha]. Consultado em Junho de 2017. Disponível em: https://getbootstrap.com/

[3] Apache Software Foundation. *Apache Cordova*. [Em linha]. Consultado em Junho de 2017. Disponível em: https://cordova.apache.org/

[4] Ionic, *Ionic*. [Em linha]. Consultado em Junho de 2017. Disponível em: https://ionicframework.com/

[5] WordPress, *WordPress*. [Em linha]. Consultado em Junho de 2017. Disponível em: https://wordpress.org/

[6] Microsoft, *ASP.NET MVC*. [Em linha]. Consultado em Junho de 2017. Disponível em: https://www.asp.net/mvc

[7] *Beck, K* et al., *Manifesto for Agile Software Development*, [Em Linha]. Consultado em Julho de 2017. Disponível em: http://agilemanifesto.org/

[8] Silveira, C. (2016) *Apontamentos da Cadeira de Engenharia de Software II*. Texto não publicado, Instituto Politécnico da Guarda, Guarda

[9] Tutorials Point, *ASP.NET MVC Tutorial*. [Em linha]. Consultado em Julho de 2017. Disponível em: https://www.tutorialspoint.com/asp.net_mvc/index.htm

[10] TinyMCE, *TinyMCE Documentation*. [Em linha]. Consultado em Julho de 2017. Disponível em: https://www.tinymce.com/docs/

[11] Apache Software Foundation, *Apache Cordova Documentation*. [Em linha]. Consultado em Julho de 2017. Disponível em: https://cordova.apache.org/docs/en/latest/

[12] Fred Heusschen, *Jquery.Mmenu Plugin*. [Em linha]. Consultado em Julho de 2017. Disponível em: http://mmenu.frebsite.nl/documentation/core/