INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

PROJETO GLOBAL

SMARTPHONES COMO AUDIOGUIAS

LICENCIATURA EM INFORMÁTICA

Nuno Miguel Vieira Mendes
Nº 1949

Coordenador:
Prof. Doutor Pedro Brandão

Lisboa, 2016

Índice

Agradecimentos

À minha família e amigos pelo apoio, paciência e compreensão.

Ao meu colega de curso Ricardo Nunes pela disponibilização do alojamento online do produto.

Aos meus colegas Duarte Bizarra, Gonçalo Abreu, Nuno Azevedo, Tiago Soveral e Ricardo Morais que sempre deram a motivação e apoio nos momentos mais complicados.

Resumo

Os audioguias têm vindo a melhorar a experiência dos visitantes a museus desde há vários anos. Com o aparecimento de novas tecnologias como *Bluetooth Low Energy* e de dispositivos capazes de emitir informação usando esta tecnologia, como os *beacons*, surgem novas formas de melhorar os audioguias tradicionais dependentes de *hardware*.

Recorrendo a essa tecnologia, o propósito deste trabalho é de construir um produto capaz de substituir os audioguias tradicionais pelos *Smartphones* pessoais.

Palavras Chave: Audioguias, Bluetooth, BLE, Beacons, Smartphones

Abstract

Audio guides have been improving the experience of visitors to museums for several years now. With the emergence of new technologies such as Bluetooth Low Energy and devices with the ability to send information using this technology, such as beacons, there are new ways to improve the traditional audioguides dependent of hardware.

Using this technology, the purpose of this work is to build a product capable of replacing the traditional audio guides by the personal Smartphones.

Keywords: Audio guides, Bluetooth, BLE, Beacons, Smartphones

Lista de Abreviaturas

REQ	Requisito
ET	Especificação Técnica
CT	Caso de Teste
SDK	*Software Development Kit*
API	*Application Programming Interface*
IoT	*Internet of Things*
BLE	*Bluetooth Low Energy*
URL	*Uniform Resource Locator*
CRUD	*Create, Read, Update, Delete*
IDE	*Integrated Development Environment*
BD	Base de Dados

Índice de Figuras

Índice de Tabelas

1 Introdução

Desde há vários anos a esta parte que os audioguias têm vindo a melhorar a experiência dos visitantes a museus, pois permitem fornecer informação descritiva acerca das peças em exposição. Ao longo dos anos têm vindo a variar no seu formato e/ou tecnologia, mas todos eles têm a grande desvantagem de necessitarem de *hardware* próprio, quer seja para detetarem a presença do visitante, no sentido de reproduzir o áudio, quer pela intervenção do próprio visitante para selecionar o número correspondente à zona onde se encontra.

Neste seguimento, o objetivo deste trabalho é criar uma Prova de Conceito[1] que englobe a elaboração de um produto capaz de substituir os audioguias tradicionais pelos *Smartphones* pessoais, dispensando assim o uso de *hardware* próprio, o que irá consequentemente traduzir-se na aquisição de valor por parte dos museus ou outras instituições que utilizem este produto nas suas exposições, na medida em que se elimina o custo de obtenção e manutenção do *hardware*, e providencia-se ainda todas as vantagens que a IoT[2] nos dá, tal como, indicar por exemplo, qual o conteúdo que tem mais interesse para os clientes finais, pela soma de tempo que os utilizadores dispensaram junto do mesmo e, pode assim, cada museu tomar as suas próprias decisões de negócio de forma a potencializá-lo.

A fim de dar resposta a estas necessidades, no capítulo dois deste documento será dado a conhecer as tecnologias atualmente disponíveis para a conceção de um produto deste género, nomeadamente as tecnologias utilizáveis por sistemas de localização *indoor*, os conceitos fundamentais do sistema Android, e ainda uma abordagem às tecnologias existentes para a construção de *webservices* e *websites*. No terceiro capítulo é identificada a contextualização do projeto com os requisitos gerais para a conceção do produto. No quarto capítulo, será pormenorizada toda a fase de conceção, desde as tecnologias utilizadas, passando pela arquitetura de sistemas, até à explicação do funcionamento de cada sistema implementado. Por fim será apresentado no capítulo cinco as considerações finais e futuros desenvolvimentos deste produto.

[1]Uma prova de conceito, ou PoC (sigla do inglês, Proof of Concept) é um termo utilizado para denominar um modelo prático que possa provar o conceito (teórico) estabelecido por uma pesquisa ou artigo técnico. Pode ser considerado também uma implementação, em geral resumida ou incompleta, de um método ou de uma ideia. É considerado habitualmente um passo importante no processo de criação de um protótipo.
[2] Sigla do inglês *Internet Of Things* (IoT) é um conceito utilizado para designar a conectividade de objetos físicos com o mundo digital por meio da web.

2 Estado da Arte

Neste capítulo será documentada a investigação científica das tecnologias ou produtos similares do que se pretende desenvolver. Esta é uma fase crítica do projeto na medida em que fornece informação essencial na tomada de decisão sobre as tecnologias a utilizar durante o desenvolvimento do produto, permitindo assim uma melhor gestão do trabalho a realizar.

2.1 Sistemas de localização indoor

Deve ser definido como qualquer sistema que tenta fornecer um posicionamento preciso no interior de uma estrutura coberta. Atualmente existem vários tipos de sistemas de deteção de localização, cada um com as suas próprias forças, mas também limitações. Os sistemas de deteção de localização *indoor* tornaram-se muito populares nos últimos anos, porque muitas aplicações precisam de saber a localização física dos objetos. Estes sistemas fornecem uma nova camada de automação e utiliza conceitos como: sem fio, reconhecimento ótico ou técnicas de ultra-som. A relevância de *context-awareness* para utilizadores móveis tem sido demonstrada em várias aplicações, tais como museus, planeamento de rotas, bibliotecas e turismo. Há três tecnologias vulgarmente utilizadas para sistemas de localização interior: *ultrasonic*, infravermelho e rádio frequência. Estas podem ser complementadas por sistemas de inércia que são geralmente utilizados para a predição. Um dos sistemas baseados em localização e mais conhecido é o GPS, que é amplamente utilizado para seguir objetos localizados ao ar livre, em movimento e determina a posição (altitude, longitude e latitude) de um objeto, em qualquer ponto do planeta por triangulação. No entanto, GPS, como é dependente de satélites, tem um problema inerente de precisão, que é quando a determinação da localização de objetos é no interior de edifícios. (Puertolas-Montanes, Mendoza-Rodriguez, & Sanz-Prieto, 2013)

2.1.1 QR-Code

Graças aos dispositivos móveis e à Internet das Coisas, soluções simples podem ser muito importantes em situações difíceis, como a rápida identificação de um produto. *QR-code* (abreviado de *Quick Response Code*) é um rótulo de duas dimensões que contém informações

sobre o item ao qual ele está ligado. Dependendo da sua finalidade, existem vários tipos de *QR-codes*:

- Texto ou códigos – o mais antigo e mais usado;
- QR geográfico – que corresponde a uma geo-localização;
- QR URL – *link* web com detalhes de objetos específicos ou para uma apresentação em vídeo;

Além disso, existem outros tipos de *QR-codes*, tais como contactos, número de telefone, correio eletrónico, etc.(Cată, 2015)

2.1.1.1 *Inconvenientes de QR-Code*

A principal desvantagem dos métodos com base em *QR-code* deve-se ao facto de a posição de determinado objeto ser somente determinada através da posição do rótulo, pois este só pode ser lido alinhando o dispositivo de leitura ao mesmo. *QR-codes* foram projetados especificamente para a transferência de informação rápida. Ler dados de *QR-codes* depende de fatores como qualidade da câmara, ângulo da imagem ou luminosidade, bem como sobre a área do código, pois esta depende do tamanho dos dados. (Puertolas-Montanes et al., 2013)

Apesar de algumas estatísticas mostrarem que mais de 38% dos adultos norte-americanos com menos de 35 anos tenham já feito alguma digitalização de *QR-codes*, a maioria dos proprietários de *smartphones* não se envolve com eles de maneira regular ou significativa. E mesmo sendo bastante versáteis, os principais casos de uso de *QR-codes* estão agora ameaçados por tecnologias mais recentes, como *Beacons*. A menos que o "*sweet spot*" de marketing de *QR code* seja encontrado em breve, esta outrora promessa de ferramenta de marketing está a terminar. (Sterling, 2014)

2.1.2 NFC

NFC, ou *Near Field Communication*, permite que informações sejam trocadas entre um *smartphone*que suporte NFC e outros objetos em curtas distâncias. A principal característica da NFC é que ela é um *interface* de comunicação sem fios com uma distância de trabalho

apenas limitada a cerca de 10 cm. Ao usar a tecnologia NFC, não há necessidade de lançar uma aplicação. O utilizador tem acesso imediato a informações adicionais simplesmente tocando ou passando o dispositivo sobre o rótulo. Mais, os dispositivos NFC também foram projetados para funcionar em áreas sem rede móvel. (Cată, 2015)

2.1.3 BLE Beacons

Bluetooth Low Energy (BLE) é uma tecnologia de rede sem fio que pode ser usada para a comunicação intrapessoal entre dispositivos. *Smart Bluetooth* destina-se a fornecer um consumo de energia consideravelmente reduzido, mantendo um alcance de comunicação similar ao comum Bluetooth. Prevê-se que quase dois biliões de *smartphones* serão lançados globalmente em 2018, quase o triplo da quantidade que em 2011 e mais de 90% deles vai suportar *Smart Bluetooth.* BLE *Beacon* é uma peça de baixo custo de *hardware*, pequeno o suficiente para anexar a uma parede ou mesa, que utiliza conexões Bluetooth de baixa energia, tornando-o amigável com a bateria para transmitir mensagens ou avisos diretamente a um *smartphone* ou *tablet* com uma aplicação compatível. A duração normal das baterias é de 2 anos, operando continuamente 24 horas / 7 dias por semana. Já existem *beacons* BLE com uma vida de bateria de três anos (em breve chegará a mais de 5 anos). A faixa de deteção de *beacon* BLE vai desde os 5 cm aos 50 m. (Cată, 2015)

2.1.3.1 Casos de uso de Beacons

A ShopAdvisor fez uma campanha com *beacons* no outono passado para a Levis. Os *Beacons* foram colocados nas lojas e ofertas especiais foram enviados para os utilizadores da ShopAdvisor. Dos consumidores que receberam a oferta, 16% visitaram a loja. Na Austrália, a Adshel está a lançar mais de 3.000 *beacons* em toda a sua rede de OOH[3] nacional para reforçar as suas capacidades de segmentação e de dados no espaço de publicidade ao ar livre. Adshel indica que os *beacons* serão utilizados inicialmente para "ouvir e aprender" e construir conhecimentos sobre os consumidores. As empresas que trabalham com os aeroportos de

[3] Sigla do inlês Out-of-home é uma forma de publicitar que chega aos consumidores quando estes estão fora das suas casas.

Londres estão a instalar 200 *beacons* por 8 aeroportos, incluindo Londres Gatwick, para entrega de conteúdos direcionados, ofertas e recompensas para 100 milhões de passageiros. A TouchTunes está a introduzir *beacons* na sua rede de Jukebox Digital, atingindo até 60.000 locais na América do Norte e Europa. (Hendrix, 2015)

Desde o National Geographic Museum em Washington até ao SXSW (evento de tecnologia e música que tem lugar em Austin, anualmente), locais, atrações e eventos estão a instalar *beacons* para melhorar a experiência dos convidados. Quando integrados com aplicativos móveis, os *beacons* enriquecem perfis de consumidores, fornecendo dados sobre áreas visitadas, tempo de permanência em exposições e outros aspetos granulares. Depois de ganhar esses *insights* da audiência, estes perfis enriquecidos permitem que os anunciantes alcancem melhor os visitantes com cartazes relevantes, *displays* e outros meios de comunicação OOH que correspondem estreitamente aos dados demográficos e interesses dos participantes, bem como ao seu comportamento no local. (Hendrix, 2015)

2.1.3.2 Requisitos para uso de Beacons

Para participar em campanhas e em locais que têm *beacons*, o utilizador deve:

1. Ter um dispositivo móvel com Bluetooth 4.0;
2. Ter o Bluetooth ligado;
3. Ter um ou mais aplicativos nos seus dispositivos móveis capazes de interagir com *beacons*;
4. Optar por receber alertas, mensagens e ofertas da (s) aplicação(ções) emparelhada(s).

(Hendrix, 2015)

2.1.3.3 Publicidade com recurso a Beacons

Em estabelecimentos comerciais, na rua ou no trânsito e em muitos outros locais, *Out-Of-Home* (OOH) media está por toda a parte. Ao oferecer uma grande variedade de locais e conteúdos de diversos formatos, permite que anunciantes e consumidores se envolvam por novas formas na sua vida do dia-a-dia, colocando-o deste modo entre os melhores canais de

publicidade. As tecnologias móveis e os *Beacons* estão a transformar o OOH media, na medida em que proporcionam relevantes e até mesmo personalizados anúncios, mensagens e ofertas. OOH alimentada por *Beacons* oferece vantagens significativas, pois permite obter dados mais ricos e com maior precisão na segmentação, traduzindo-se assim em resultados mais mesuráveis. (Hendrix, 2015)

2.2 Android

É um sistema operacional móvel que é executado no kernel do Linux[4]. O desenvolvimento aplicacional para Android é baseado em linguagem Java[5]. Esses códigos podem controlar dispositivos móveis através de bibliotecas Java disponibilizadas pela Google. É uma importante plataforma para desenvolver aplicações móveis usando a pilha de *software* fornecido no Google Android SDK. O sistema operacional Android fornece um ambiente flexível para desenvolvimento de aplicações móveis, pois os programadores podem utilizar não só bibliotecas Java Android, como também é possível usar IDEs[6] Java normais. O desenvolvimento de aplicações móveis pode ser usado para criar aplicações inovadoras e dinâmicas de terceiros. (Holla & Katti, 2012)

2.2.1 Arquitetura do sistema Android

Android é um sistema baseado em Linux que utiliza os padrões de arquitetura de pilha de *software*. Como é apresentado na Figura 1 da página seguinte, a arquitetura android é composta por quatro camadas: kernel Linux; bibliotecas e Android Runtime; *framework* para aplicações; e aplicações. Cada camada inferior fornece uma espécie de encapsulamento, enquanto é também uma interface de chamada para as camadas superiores.(Jose, Lakshmi, Priyadarshini, & Singh, 2015)

[4] Sistema operativo de computadores desenvolvido sobre o modelo de software livre e código aberto.
[5] Linguagem de programação orientada a objetos. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um *bytecode* que é executado por uma máquina virtual.
[6] Em inglês *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.

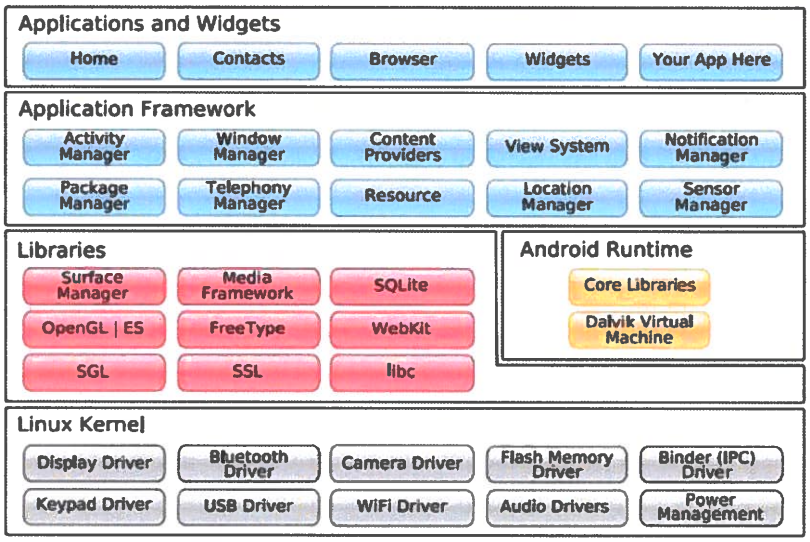


Figura 1 - Arquitetura Sistema Android

As aplicações para Android são escritas em linguagem de programação Java. No entanto, é importante lembrar que elas não são executadas utilizando o padrão de máquina virtual Java (JVM). Em vez disso, a Google criou uma VM customizada chamada Dalvik, que é responsável pela conversão e execução de código Java.(Holla & Katti, 2012)

2.3 Desenvolvimento de aplicações Android

O Android SDK[7] fornece um amplo conjunto de interfaces de programação de aplicações (APIs[8]) que é moderno e robusto. Os serviços do sistema nuclear Android estão expostos e acessíveis a todas as aplicações. Quando concedidas as permissões apropriadas, elas podem compartilhar dados entre si e ter acesso a recursos geridos pelo sistema de segurança. (Holla & Katti, 2012)

[7] *Software Development Kit* – conjunto de ferramentas que ajudam na programação de *software*.
[8] Provém do inglês *Application Programming Interface* e trata-se da disponibilização de um conjunto de funções e tipos que ajudam na construção de programas de *software*.

2.3.1 Xamarin

> *"Anything you can do in Objective-C, Swift, or Java you can do in C#"*("Xamarin," 2016)

Recentemente adquirida pela Microsoft, Xamarin é uma empresa de desenvolvimento de software para multiplataformas móveis, pois permite a criação de aplicações Android, IOS e Windows Phone com apenas uma linguagem de programação – C#.(Greene, 2016)

Completamente integrado no Visual Studio 2015, o desenvolvimento das aplicações poderá ser feito de uma forma nativa para cada uma das plataformas suportadas, ou através de Xamarin.Forms. Ainda que seja desaconselhado para aplicações de complexidade alta, Xamarin.Forms disponibiliza uma forma de utilizar controlos de *User Interface* suportáveis nas diferentes plataformas.("Xamarin," 2016)

2.4 SQLite

A base de dados incorporada SQLite é amplamente aplicada na gestão de dados de ambiente incorporado tais como os dispositivos móveis, controle industrial, etc., e tornou-se o foco do desenvolvimento de áreas relacionadas. É muito usado devido às suas vantagens de estabilidade e confiabilidade, eficiência e portabilidade. (Bi, 2009)

Hoje em dia, a tecnologia de bases de dados incorporadas tornou-se um campo de pesquisa importante, pois os principais fornecedores de base de dados lançaram os seus próprios produtos, embora estes produtos sejam incorporados com orientação ao campo, eles têm as suas próprias características sobre especificações técnicas. SQLite tem atraído fortemente a maioria dos programadores pelas suas vantagens, tais como o peso leve, fácil inclusão e sem restrições de direitos de autor. (Bi, 2009)

O sistema de base de dados incorporada é um sistema que suporta computação móvel ou um modo de computação particular, que normalmente é integrado com o sistema operacional e aplicações específicas, e executado em *smartphones*. (Bi, 2009)

2.4.1 Características do SQLite

SQLite é um tipo de base dados incorporada *open-source*, que é escrito em linguagem C. Comparando com as bases de dados habituais, tais como SQL Server, Oracle, etc, SQLite é uma base de dados leve que pode ser integrada, sem componentes adicionais, e é especialmente adequado para aplicações móveis. (Bi, 2009)

SQLite não precisa ser instalado e configurado, e não necessita de processo para iniciar e parar. Pode-se restaurar automaticamente após o colapso ou perda de energia do sistema. SQLite fornece *easy-to-use* API, esta liga-se à base de dados diretamente através da função da API e pode suportar linguagens avançadas de acesso. É muito adequado para base de dados incorporadas. (Bi, 2009)

SQLite pode ler e gravar os ficheiros de base de dados no disco rígido diretamente e não precisa de um processo de serviço adicional. Uma base de dados completa corresponde a ficheiros no disco. Os mesmos ficheiros de base de dados podem ser usados em máquinas diferentes, mas também podem ser livremente compartilhados entre máquinas com diferentes ordens de byte. (Bi, 2009)

2.5 Android & SQLite

Todos os principais sistemas operacionais móveis incluem a API para operações na base de dados. Os programadores podem ligar-se à base de dados usando a API nativa ou usando bibliotecas de *wrapper*.

Platform	Mobile database	API
Android	SQLite	Java
iOS	SQLite	C/C++
Symbian	SQLite	C++
Windows CE/Windows Embedded Compact/ Windows Mobile	EDB	C/C++
	SQL Server CE/ Compact	C/C#
Windows Phone	SQL Server Compact	C#

Figura 2 - Motores de gestão de base de dados integrados em plataformas móveis

O Android usa o sistema de gestão de base de dados SQLite. Para realizar operações na base de dados, existem classes disponíveis como a SQLiteOpenHelper, SQLiteDatabase e Cursor. Para criar uma nova base de dados é usada uma classe derivada de SQLiteOpenHelper. Existem dois métodos que precisam de ser implementados:

- void OnCreate (SQLiteDatabase bd)- é chamado para criar a base de dados. O corpo da função contém o código para criar tabelas e outros objetos da base de dados (View, Trigger etc.);
- void ONUPGRADE (db SQLiteDatabase, int olVers, int newVers) - é chamado quando a estrutura da base de dados é modificada (tabelas e outros objetos da base de dados).

(Pocatilu, 2012)

2.6 *Web services*

São componentes de *software* que são desenhados para aplicações informáticas comunicarem entre si. Estas comunicações podem ser desde uma simples operação de verificar o saldo da conta, até processos complexos executados por um sistema CRM (*customer relationship management*) ou ERP (*Enterprise Resourse Planing*). De forma a permitir a comunicação entre aplicações de tecnologias diferentes, estas comunicações são então feitas de uma forma generalizada e baseada em *standarts* de tecnologias web, incluindo HTTP e mensagens em XML.(Cavanaugh, 2006)

Os *Web services* trabalham a um nível de abstração semelhante à Internet e são capazes de fazer a ponte entre qualquer sistema operacional, plataforma de *hardware*, ou linguagem de programação, assim como a Web o é". (Aldea, Sangeorzan, & Aldea, 2009)

2.6.1 Benefícios dos *web services*

Os *web services* disponibilizam os seguintes benefícios tecnológicos e de negócio:

- Integração aplicacional
- Integração de informação
- Versatilidade

- Reutilização de código
- Poupança de custos

A interoperabilidade que se obtém usando o XML como linguagem independente de fornecedores, plataforma e tecnologias, e tendo HTTP como transporte, significa que qualquer aplicação pode comunicar com qualquer outra aplicação usando *web services*.(Cavanaugh, 2006)

2.6.2 XML

É uma especificação W3C[9] que define a meta-linguagem para descrever informação. Em XML, a informação é delimitada por *tags* personalizadas e baseadas em texto, estruturadas de forma hierárquica e que descrevem a informação contida dentro das mesmas.(Cavanaugh, 2006)

2.6.3 WSDL

É um formato baseado em XML para descrever *web services* que também é mantido pela W3C. Os clientes que pretendem consumir um *web service*, apenas têm que ler e interpretar o seu ficheiro WSDL, e assim identificar quais as operações que o serviço tem, bem como onde pode ser acedido, quais os protocolos de comunicação que utiliza, e o formato correto para enviar mensagens.(Cavanaugh, 2006)

2.6.4 SOAP (*Simple Object Access Protocol*)

É um protocolo baseado em XML da W#C para trocar informação sobre HTTP. Disponibiliza uma forma *standart* simples de trocar mensagens XML entre aplicações. Os *web services* utilizam então SOAP para enviar mensagens entre um serviço e os seus clientes. Devido ao facto de HTTP ser suportado por todos os servidores web e todos os *browsers*,

[9]O *World Wide Web Consortium* (W3C) é uma comunidade internacional que desenvolve padrões abertos para garantir o crescimento a longo prazo da Web.

mensagens SOAP podem ser enviadas entre aplicações, independentemente da sua plataforma ou linguagem de programação. Esta qualidade oferece aos serviços da web a sua interoperabilidade. (Cavanaugh, 2006)

2.6.5 REST

O termo *Representation State Transfer* (REST) foi introduzido por Roy Fielding. A arquitetura de REST é uma arquitetura cliente-servidor em que o cliente envia pedidos ao servidor, e posteriormente o servidor processa esses pedidos e retorna respostas. Estes pedidos e respostas são feitas sobre a transferência de representações de recursos. Um recurso é algo que é identificado por um URI[10]. A representação de um recurso é tipicamente um documento que capta o estado atual ou pretendido de um recurso. REST é constituído por tipos menos fortes do que SOAP. A linguagem REST baseia-se na utilização de nomes e verbos e não exige formato de mensagem como *envelope* e *cabeçalho* que é exigido em mensagens SOAP. Assim como também não é necessária análise XML, havendo assim menos exigência de largura de banda. O padrão de desenho de REST assenta em três princípios: endereçamento de encaminhamento, sem estado e com interface uniforme. Modelos REST são os conjuntos de dados para operar como recursos onde os recursos são associados a um URI. É usado um interface uniforme e *standard* na medida em que é usado um conjunto de métodos HTTP fixos. Toda a transação é independente e não é relacionada com a operação anterior, sendo que todos os dados necessários para processar o pedido estão contidos nesse mesmo pedido. Os dados de sessão de cliente também não são mantidos no lado do servidor, portanto, as respostas do servidor também são independentes. Estes princípios tornam aplicações REST mais simples e leves. A aplicação web que segue uma arquitetura REST poderá ser chamada de *RESTful web service*. Essa aplicação usa os métodos http -*GET*, *PUT*, *POST* e *DELETE* para, respetivamente, obter, criar, atualizar e eliminar recursos. (Mumbaikar & Padiya, 2013)

[10] Em inglês *Uniform Resource Identifier*, é uma cadeia de carateres compacta usada para identificar ou denominar um recurso

2.6.6 SOAP vs REST

O interesse em *web services* teve um aumento muito rápido desde o início do seu uso. Trocar informação de uma forma *standard* é o principal objetivo do uso de *web services*. Esta comunicação entre aplicações é baseada em princípios SOAP ou REST. Comunicações SOAP causam maior tráfego na rede, maior latência e mais atrasos no processamento. Para colmatar estas limitações começou a ser usada a arquitetura REST'full. A REST é uma alternativa mais leve comparando com SOAP.(Mumbaikar & Padiya, 2013)

Num estudo efetuado por Snehal Mumbaikar e Puja Padiya do departamento de engenharia de computação da agência de investimento R.A.I.T, foram obtidos os seguintes resultados em termos de performance:

- Os tempos de atraso ponta-a-ponta de RESTful *web services* são 3 a 5 vezes menores que *web services* SOAP;
- A carga na rede de RESTful *web services* é quase 3 vezes menor que SOAP *web services*;

Ambos os autores afirmam inequivocamente que em termos de performance e carga na rede, e para operações simples de ler, atualizar e eliminar informação, os RESTful *web services* são consideravelmente melhores do que SOAP.(Mulligan & Gracanin, 2009)

2.7 ASP.NET MVC

A plataforma .NET da Microsoft disponibiliza duas formas de construir aplicações Web: ASP.NET Web Forms ou ASP.NET MVC. A primeira foi usada durante vários anos e providencia diversos *server-side controls*. A segunda, mais recente e também mais extensível, é mais leve e suporta os componentes HTML fundamentais. (Claesson & Dubray, 2009)

ASP.NET MVC segue o padrão de desenho *Model-View-Controller*, separando a responsabilidade de cada camada da aplicação. Model diz respeito às entidades usadas na aplicação, *View* à *user interface* da mesma e o *Controller* para estabelecer a comunicação entre as duas, providenciando métodos a serem utilizados pela *View* de forma a obter entidades da aplicação ou alguma alteração sobre as mesmas, como "ler" ou "gravar". (Claesson & Dubray, 2009)

3 Contextualização

Dado o desafio proposto da conceção de um produto que seja capaz de substituir os audioguias tradicionais, ele terá que obrigatoriamente responder a algumas necessidades implícitas.

A primeira e principal delas todas é criar uma aplicação móvel que ao detetar a presença de um *beacon*, inicie a ação associada ao conteúdo correspondente, tal como tocar um ficheiro áudio ou vídeo, ou mesmo abrir um URL, a fim de dar a conhecer algum tipo de publicidade ao cliente final.

Deste modo, será igualmente necessário possibilitar a gestão desses mesmos conteúdos (áudio, vídeo ou link), para que se consiga não só realizar as *CRUD operations*[11] sobre os mesmos, como também fazer a correspondência de conteúdos a *beacons*. Este gestor de conteúdos deverá ser acessível a qualquer hora e a qualquer lugar, pelo que será construído um website para esta finalidade.

Ao ser dada esta possibilidade de gestão de conteúdos, implica que a aplicação móvel seja também capaz de sincronizar essa informação que vai sendo atualizada ao longo do tempo. Deste modo, será construído um *webservice* que possibilita a realização destas operações de sincronismo.

[11] CRUD – *Create, Read, Update, Delete* – Operações base realizáveis numa base de dados

4 Desenvolvimento

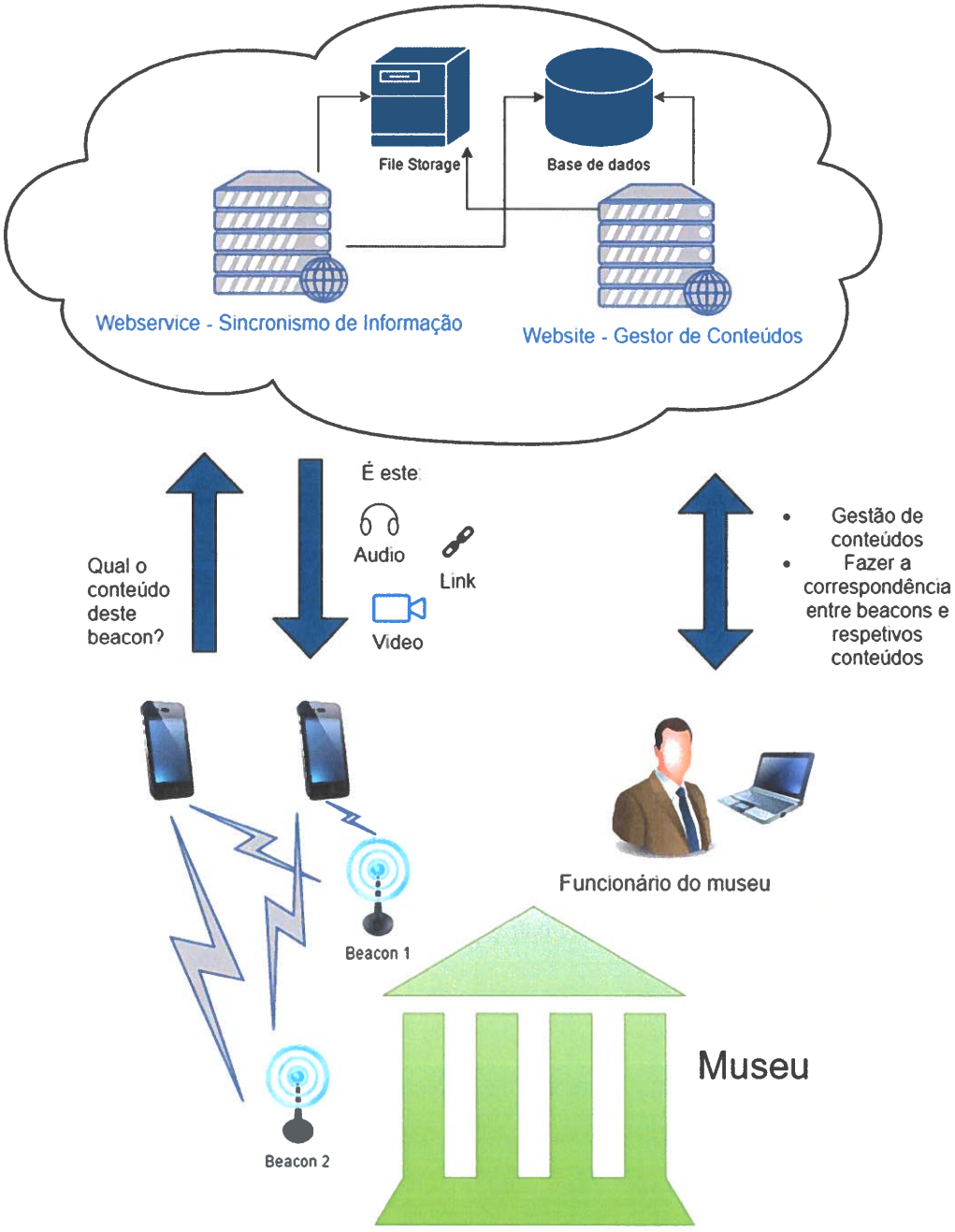
Com vista a responder às necessidades identificadas, procurou-se arquitetar da melhor forma os sistemas necessários à conceção do produto.

O primeiro sistema, e foco de desenvolvimento em todo o produto, é uma aplicação para *Smartphones* que deteta a presença de *beacons*, no sentido de começar a tocar o áudio correspondente a esse *beacon*. Optou-se pelo desenvolvimento de uma aplicação para o sistema operativo Android, dado ter sido uma das áreas abordadas no decorrer do curso. Ainda assim, e devido ao facto de o autor sentir mais apetência para desenvolvimento de aplicações .NET, optou-se por utilizar uma tecnologia emergente – Xamarin. Tal como é abordado no capítulo dois deste documento, ela foi adquirida recentemente pela Microsoft, incluída no Visual Studio, permitindo assim desenvolver aplicações para Android ou IPhone utilizando a .NET Framework em perfeita comunhão com o Android SDK ou IPhone SDK.

O segundo sistema trata-se de um website para um funcionário do museu poder gerir os seus conteúdos. Ele foi desenvolvido sob a tecnologia ASP.NET MVC e permite que um utilizador autenticado crie, atualize ou elimine conteúdos e os corresponda aos *beacons* entretanto adquiridos.

Por último, um *webservice* para a aplicação Android ter conhecimento das últimas atualizações de conteúdos de cada museu. Ele permite essencialmente fazer este sincronismo de informação através de pedidos REST, devolvendo todas as atualizações que foram feitas desde uma determinada data em todas as entidades sincronizáveis do produto, tais como Clientes, Conteúdos ou *Beacons*, entre outras.

Figura 3 - Ilustração do funcionamento dos 3 sistemas



4.1 Método

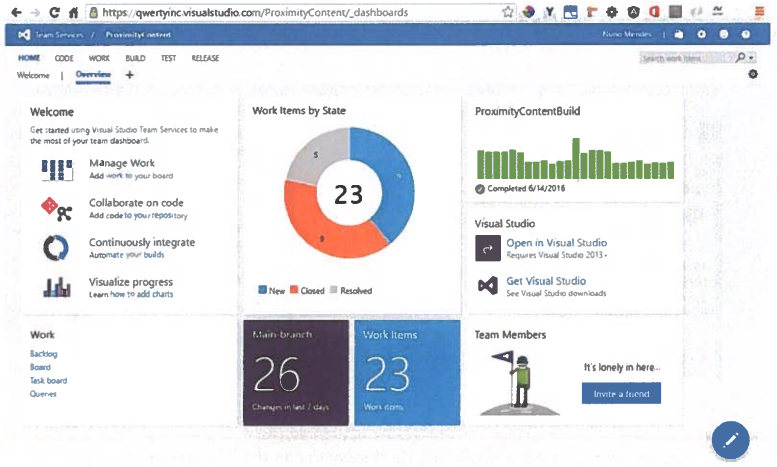


Figura 4 - Team Services dashboard

Optou-se por desenvolver o produto utilizando o *Team Foundation Server* (TFS) como repositório e controlo de versões do código fonte do produto, e também fazer uso do seu método de *Continuous Integration* (CI) que disponibiliza as seguintes vantagens:

- Manter um repositório de código fonte disponível 24h/7d.
- Automatização de build e deploy do produto para um ou mais ambientes produtivos.
- Possibilidade de execução de testes automatizados após deploy
- Ferramenta colaborativa.
- Entregas do produto de forma continuada, isto é, à medida que as novas funcionalidades são implementadas, elas estarão imediatamente disponíveis.

O TFS também disponibiliza a metodologia ágil de desenvolvimento de software[12] através de ecrãs que permitem essa gestão, nomeadamente a criação e edição de *User Stories*[13], bem como a atribuição do estado de desenvolvimento da mesma (Novo, Ativo, Resolvido e Fechado).

[12] Metodologia ágil de desenvolvimento de software pretende estabelecer um conjunto de regras para o desenvolvimento de um produto. Um exemplo desta metodologia é o SCRUM que obriga á identificação de um *backlog* (conjunto de funcionalidades a desenvolver) com estimativas de tempo para a sua implementação. Esta implementação é feita sobre a forma de Sprints, sendo este um intervalo de tempo (normalmente de duas semanas) para implementação de um conjunto de funcionalidades, permitindo assim ter uma noção de qual será a data da próxima *release* do produto.

[13] Descrição de determinada funcionalidade, com os seus requisitos e comportamentos a implementar.

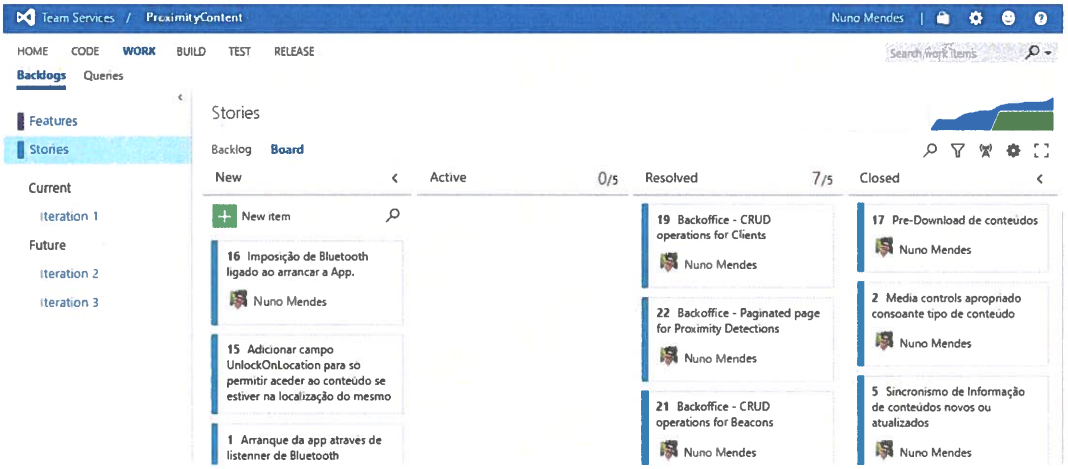


Figura 5 - Metodologia ágil de desenvolvimento de software

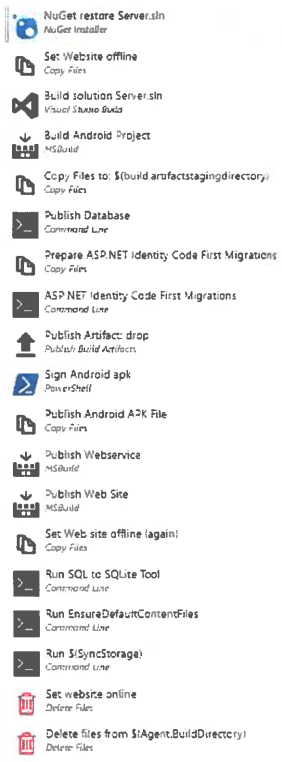


Figura 6 - Build Definition

Para a automatização do *deploy* do produto foi configurada a *Build Definition* exibida na figura ao lado.

Inicia o processo de *deploy* com a restauração de novos ou atualizados pacotes Nuget, seguidamente coloca o website offline enquanto será feita a atualização do produto. Realiza a compilação do código fonte dos 3 sistemas, bem como do projeto de base de dados. Seguidamente, é feita a publicação dos 3 sistemas. Por fim são executadas as ferramentas de ajuda na automatização do deploy, como a ferramenta de conversão de SQL para SQLite, abordada no capítulo Ferramentas auxiliares - SQL to SQLite.

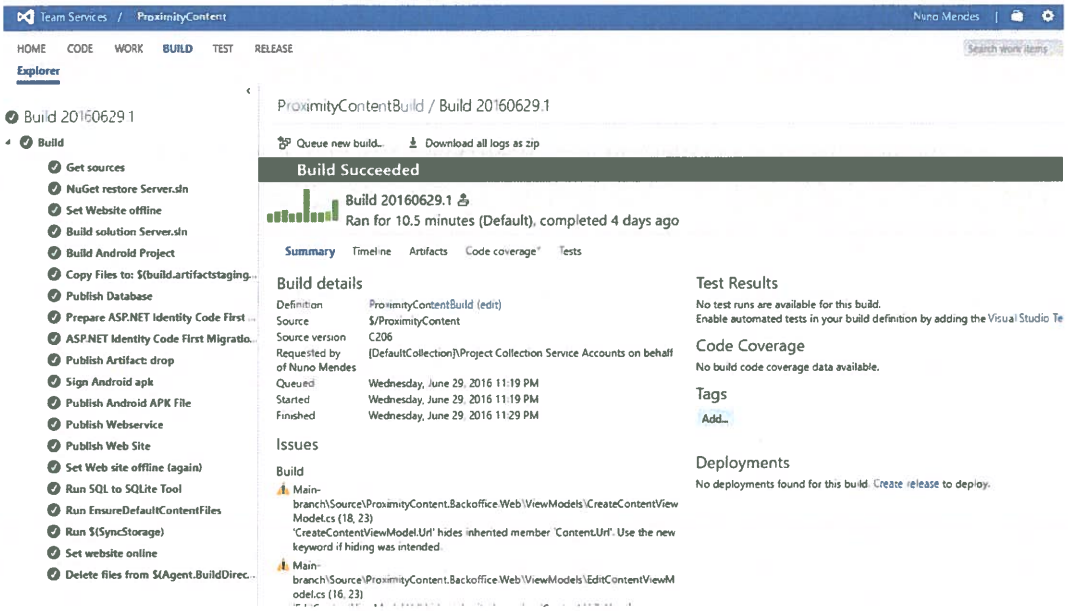


Figura 7 - Resultado da execução de um build

4.2 Base de Dados

De forma a manter o sincronismo entre os diversos sistemas, a informação tem de ser obrigatoriamente armazenada centralmente. Nesse sentido, criou-se uma base de dados para ser usada tanto pelo website de gestão de conteúdos, como pelo *webservice* de sincronismo de informação. Ela foi criada no SQL Server 2014 e com o modelo ER ilustrado.

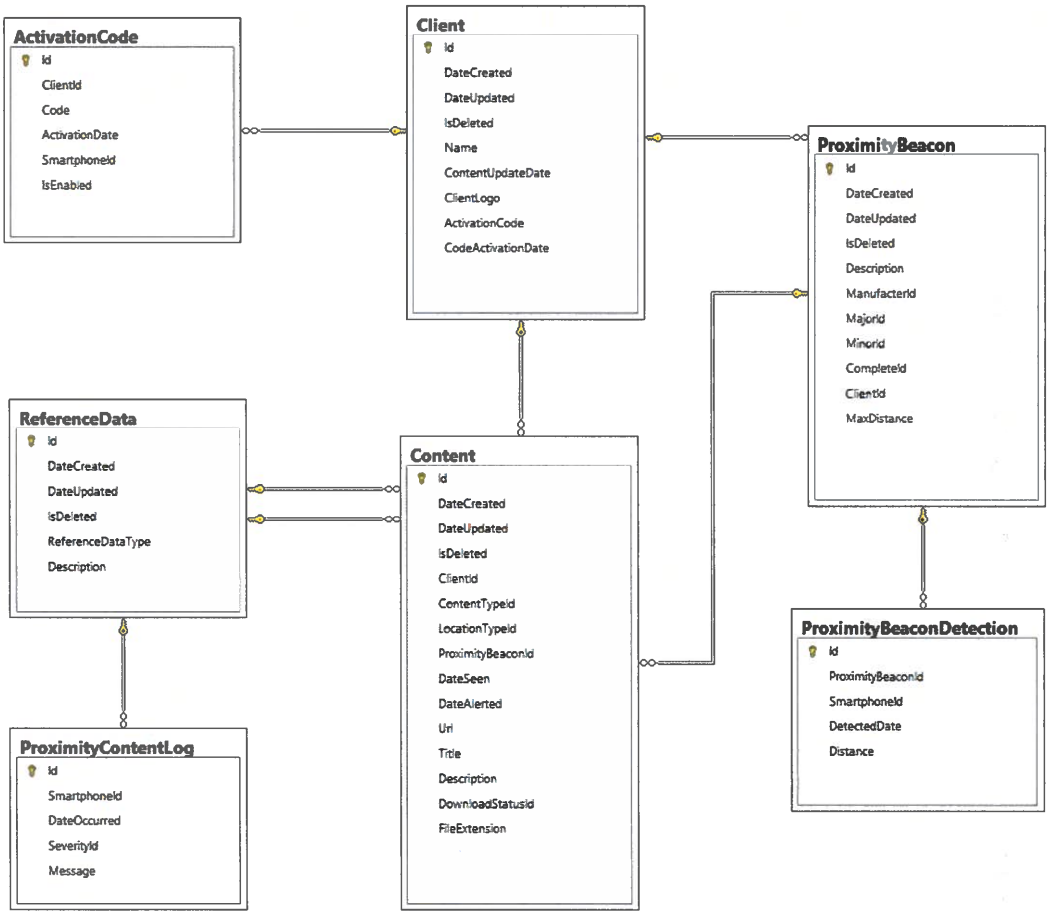


Figura 8 - Modelo ER

Tal como se pode verificar na figura acima, as tabelas Content, Proximity Beacon e Client são as tabelas principais da BD, pois são elas que guardam, respetivamente, informação relativa a Conteúdos, Beacons e Clientes. A tabela Reference Data guarda informação de

referência utilizada pelos três sistemas, tal como Tipos de Conteúdos, Tipos de Status de Download ou Tipos de Log. As restantes tabelas dizem respeito a funcionalidades específicas do produto, tal como o armazenamento de presenças detetadas pelos Smartphones ou códigos de ativação que foram gerados e utilizados nos museus.

No sentido de facilitar o sincronismo de informação das entidades que são sincronizáveis com a aplicação móvel, tal como Client, Content ou Beacon, todas estas tabelas obedecem a uma estrutura de campos fixa:

```
[Id]                UNIQUEIDENTIFIER    NOT NULL DEFAULT (newid()),
[DateCreated]       DATETIME            NOT NULL DEFAULT (getdate()),
[DateUpdated]       DATETIME            NOT NULL DEFAULT (getdate()),
[IsDeleted]         BIT                 NOT NULL DEFAULT 0,
```

Figura 9 - Estrutura base de entidades sincronizáveis

Um Id de formato UNIQUEIDENTIFIER que corresponde a um Guid para evitar problemas com ids numéricos autos incrementáveis quando se eliminassem registos em alguns dos lados da sincronização. Nas restantes três colunas, o nome é autoexplicativo, mas irão permitir dar conhecimento de quais os registos que deverão ser atualizados na aplicação móvel e de que forma – criar, atualizar ou eliminar.

Para automatizar o processo de criação ou atualização da BD no SQL Server, foi criado um projeto no Visual Studio do tipo SQL Database Project. Nele foram inseridos os scripts de criação das tabelas e os scripts de *PostDeployment* para atualização de registos. Este tipo de projeto simplifica o processo de deploy de uma base de dados, providenciando mecanismos de comparação entre o que está definido na BD e o que está definido no projeto, para deste modo gerar o script de criação / alteração da mesma.

Devido ao facto da definição das tabelas ir alterando regularmente durante o desenvolvimento da primeira *release*, e de forma a não lidar com essas modificações manualmente na construção de uma BD SQLite idêntica, foi automatizado esse processo com a construção de uma ferramenta que lê a definição das tabelas de uma BD SQL e cria uma BD SQLite com essa definição. Consulte o detalhe desta ferramenta em Ferramentas auxiliares - SQL To SQLite.

4.3 Aplicação Android

Trata-se do sistema principal para a conceção deste produto, e como tal também o mais trabalhoso, uma vez que foi desenvolvido numa tecnologia que não havia sido lecionada durante o decorrer do curso – Xamarin, e também porque obrigou a uma intensa investigação sobre a melhor forma de colocar a aplicação a detetar *beacons*.

4.3.1 Tecnologias, ferramentas e/ou referências utilizadas

- Visual Studio 2015 – IDE da Microsoft para desenvolvimento de aplicações.
- Xamarin – Tecnologia recentemente adquirida pela Microsoft para desenvolvimento de aplicações no Visual Studio para diferentes plataformas móveis, como Android ou IPhone.
- AltBeacon Library – API open source para deteção da presença de *beacons*.
- Font Awesome – Vários icons customizáveis na sua cor e/ou tamanho, e disponíveis sem qualquer custo.
- Newtonsoft.Json – API para serializar objetos em formato json e *vice-versa*.

4.3.2 Especificação de Requisitos

ID REQ	DESCRIÇÃO REQUISITO
REQ01	Ao entrar na aplicação deverão ser listados os clientes (Museus, exposições, etc..)
REQ02	Os conteúdos dos clientes são privados e são só obtidos mediante código de ativação
REQ03	O código de ativação só á válido durante 6 horas.
REQ04	Deverá ser feito o download dos conteúdos áudio e/ou vídeo quando for validado com sucesso o código de ativação.
REQ05	Ao selecionar, o cliente deverá ver listados os conteúdos disponíveis. Nesta listagem deverá ser indicado o tipo de cada conteúdo, título e descrição do mesmo.
REQ06	Ao selecionar um conteúdo deverá ser acionada a ação correspondente ao tipo de conteúdo: • Áudio – Abrir uma janela que reproduz ficheiros áudio. Deverá ser possível navegar no *timeline* do mesmo. • Vídeo - Abrir uma janela que reproduz ficheiros de vídeo. Deverá ser possível navegar no *timeline* do mesmo. • Link – Abrir uma janela com um *browser* embutido e navegar para o link indicado.

REQ07	Em qualquer que seja a janela da aplicação, deverá ser executado em background, a deteção da presença de *beacons*. Quando detetada essa presença, deverá: 1. Verificar a existência na BD do conteúdo correspondente a esse *beacon*. 2. Ser registado na BD essa presença. 3. Mediante as seguintes condições serem verdadeiras, deverá ser acionada a ação correspondente ao tipo de conteúdo, de acordo com o REQ06: a. O conteúdo ainda não foi exibido. b. Já foi descarregado o conteúdo, sendo que este descarregamento só é feito mediante o REQ03). c. Se o *beacon* tiver distância máxima configurada, o *smartphone* deverá estar a uma distância abaixo.
REQ08	Deverá existir um menu na aplicação com a opção "Sincronizar". Esta opção irá requisitar ao *web service* toda a informação que foi atualizada no gestor de conteúdos, bem como enviar registo de presenças e *logs* de erros que tenham ocorrido, de forma a ficar com essa informação guardada numa base de dados central, para posterior avaliação e tratamento.
REQ09	Ao entrar na aplicação deverá ser executado o sincronismo descrito no REQ08, mas de forma silenciosa e em *background*.

Tabela 1 - Aplicação Android - Especificação de requisitos

4.3.3 Matriz de rastreabilidade

ID REQ	ID ET	Nome Especificação Técnica	ID CT
REQ01	ET01	Listagem de clientes	CT01
REQ02	ET02	Validação de Código de Ativação	CT02
REQ03	ET02	Validação de Código de Ativação	CT02
REQ04	ET03	Listagem de conteúdos	CT02
REQ05	ET03	Listagem de conteúdos	CT02
REQ06	ET04	Reprodução de conteúdos	CT03
REQ07	ET05	Deteção de *beacons*	CT04
REQ08	ET06	Sincronismo de informação	CT05
REQ09	ET06	Sincronismo de informação	CT05

Tabela 2 - Aplicação Android - Matriz de rastreabilidade

4.3.4 Especificação Técnica

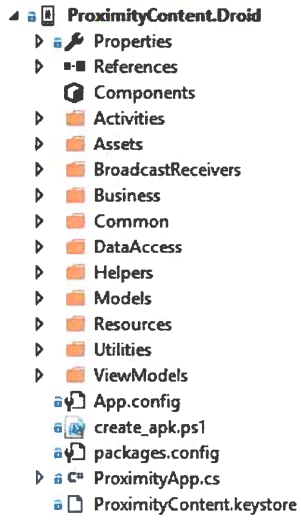


Figura 10 - Projeto da aplicação Android no Visual Studio

Tal como referido anteriormente, a aplicação Android foi desenvolvida em Xamarin, utilizando, portanto, o Visual Studio como IDE e linguagem C# como codificação da aplicação.

Para além das pastas normais de qualquer aplicação Android, como *Assets*[14] ou *Resources*[15], foram criadas novas pastas, de forma a organizar e diferenciar as diferentes camadas da aplicação, como *Business*, *DataAcces* e *Models*.

A camada *Models*, tal como o nome indica, é onde são definidas as classes que serão os modelos da aplicação, nomeadamente, pelo que é constituído um Cliente, um Conteúdo, ou mesmo um Beacon. Estes modelos são, na sua generalidade, a tradução em classes da definição das diferentes tabelas da base de dados SQLite, em que as suas propriedades correspondem à definição das colunas dessas mesmas tabelas.

Data Access é a camada responsável por fazer chamadas à base de dados SQLite da aplicação, no sentido de realizar as *CRUD operations* sobre os modelos acima referidos.

[14] Diretório onde são guardados ficheiros que são essencialmente acessórios da aplicação, tais como ficheiros xml ou txt, ou mesmo outros formatos, tal como áudio ou vídeo.
[15] Diretório onde são guardados ficheiros de recursos da aplicação, tais como ficheiros de layout para diferentes dimensões de dispositivos móveis, ou ficheiros com as mensagens de texto de cada idioma da aplicação.

Por fim, a camada Business estabelece a comunicação entre Modelos da aplicação e as diferentes funcionalidades da aplicação, pois é nela que se encontram os serviços responsáveis por realizar as operações globais a toda aplicação, tal como detetar a presença da *beacons*, ou comunicar com o *web service* de forma a sincronizar a informação.

De modo a obter tanta reusabilidade de código quanto possível, foram introduzidos alguns conceitos de herança na definição das *Activities* da aplicação. Todas as *Activities* da aplicação herdam de *ProximityBaseActivity*, uma classe abstrata que providencia algumas propriedades e métodos que serão úteis às classes filhas, nomeadamente a configuração do menu, ou *Bind* e *Unbind* dos serviços globais da aplicação. Foi ainda definida outra classe abstrata – *ContentBaseActivity*, que também herda de *ProximityBaseActivity* e servirá de base a todas as *activities* que fazem a reprodução de conteúdos.

4.3.4.1 ET01 – Listagem de clientes

Figura 11 - Listagem de clientes na aplicação Android

É a primeira janela que é exibida ao entrar na aplicação, pelo que deve ser garantido que a aplicação tem, nesse momento, toda a informação que precisa para funcionar, nomeadamente a base de dados SQLite de suporte à aplicação, deste modo, no método *Create*

da *MainActivity* é garantido que essa BD existe, no caso de não existir, faz o pedido ao web service no sentido de realizar o download da mesma.

Só após serem garantidas as condições necessárias ao bom funcionamento da aplicação, é que é feito então o carregamento da lista de clientes. De modo a providenciar algum *feedback* de que a aplicação está a carregar dados ao invés de estar bloqueada, foi criada uma classe auxiliar chamada *TaskHelper*. Ela contempla um método para utilizar uma *Progress Dialog* da Android SDK, enquanto executa uma determinada tarefa em *background*.

Com intuito de disponibilizar futuras customizações de novos campos na definição de clientes, como uma imagem, ou o horário de funcionamento do museu, ou até mesmo a geo localização do mesmo, foi criado em *Resources\Layout* um novo *layout* chamado de *ClientListItem.axml*, que corresponde a cada linha da listagem de clientes. Neste momento, o layout apenas é composto por um ícon ilustrativo de um museu e o nome do museu.

4.3.4.2 ET02 – Validação de Código de Ativação

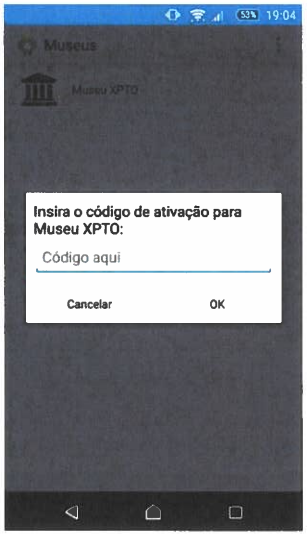


Figura 12 - Pedido ao utilizador pelo código de ativação

Ocorre quando é selecionado um cliente na listagem de clientes, e tendo em conta o requisito REQ02, nesse momento é requisitado ao utilizador o código de ativação do cliente selecionado.

Os códigos de ativação são gerados no *web site* de gestor de conteúdos, pelo que o utilizador final deverá solicitar / comprar junto do Museu um código de ativação e tendo em conta o REQ03, o mesmo será válido nas 6 horas seguintes. Durante esse tempo, e enquanto o utilizador navega pelas janelas da aplicação, se o código continuar válido, não é solicitado ao utilizador o código de ativação.

Após o utilizador introduzir o código de ativação na caixa de diálogo, é feito um pedido ao *web service* no sentido de validar o mesmo, pelo que nessa altura o utilizador necessita de ter uma ligação à internet ativa. A validação do código é então feita centralmente e as respostas possíveis por parte do *web service* são as seguintes:

- "OK" – Quando um código é válido.
- "InvalidCode" – Quando é introduzido um código vazio
- "CodeNotExists" – Quando é introduzido um código inexistente
- "CodeInUse" – Quando o código já está em uso por outro *smartphone*
- "CodeUsingTimeLimitExpired" – Quando o código já foi usado durante mais de 6h.

4.3.4.3 ET03 – Listagem de Conteúdos

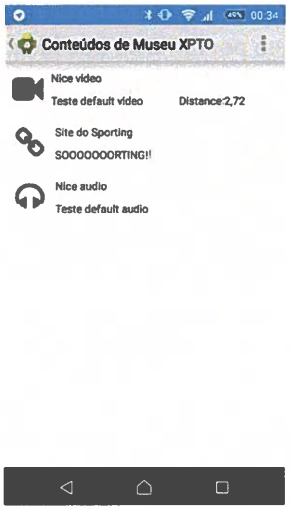


Figura 13 - Listagem de Conteúdos - Aplicação Android

Quando é selecionado um cliente na janela da listagem de clientes, e após ser validado um código de ativação com sucesso, é então criado um *Intent*[16] para dar início a uma nova

[16] Tipo definido no Andoid SDK que corresponde a uma operação a ser executada, tal como iniciar uma nova *Activity*, ou enviar mensagens a um qualquer registado recetor, ou ainda inicializar um novo serviço.

activity[17] – a ContentsActivity. No método *create* da mesma, e tendo em conta o REQ04, é despoletado o *download* dos conteúdos que estiverem com o *DownloadStatus*[18] diferente de *Downloaded*. Para desenvolver esta funcionalidade de download de conteúdos, foi necessário construir duas classes: *ProximityDownloadService* e *DownloadProgressReceiver*. A primeira trata-se de um serviço da aplicação para download de ficheiros. Ela, por sua vez, utiliza o *DownloadManager* do Android SDK para efetivamente realizar o download, uma vez que esta classe por si só, lida com as diversas preocupações que advém durante o download de ficheiros, como por exemplo quebras de conectividade e respetiva restituição do download, ou notificação de progresso. A classe *DownloadProgressReceiver* é uma classe que herda de *IntentBroadcastReceiver*, e serve unicamente para ser registado o seu uso nas *activities* que desejarem fazê-lo, neste caso, a *ContentsActivity* regista este uso de forma a ser notificada do progresso do download dos conteúdos.

Ainda no método *create* é registado um outro *Receiver*, desta feita para que a ContentsActivity seja notificada de algum beacon nas proximidades e a que distância se encontra do mesmo.

Em termos de layout, foi criado um ficheiro novo para o efeito em Resources\Layout chamado de *listview_row* que corresponde a cada linha da listagem de conteúdos. Dando resposta ao REQ05, ele é composto por um *ícon* que identifica o tipo de conteúdo (áudio, vídeo ou link) e por várias *TextViews* para dar informação sobre o mesmo, tal como "Título", "Descrição" e, se o *beacon* correspondente estiver nas proximidades, é indicada a distância do mesmo noutra *TextView*. Se estiver a ser realizado o download do conteúdo respetivo, é mostrado uma barra de progresso que vai sendo atualizada pelo *DownloadProgressReceiver*. Se ocorrer algum erro durante o download do conteúdo, é exibido noutra *TextView* uma mensagem alusiva ao efeito.

[17] Tipo definido no Andoid SDK que corresponde a uma janela da aplicação.
[18] Foram definidos em constantes da aplicação os seguintes status de download: NotDownloaded, Downloading, ErrorDownloading e Downloaded

4.3.4.4 ET04 – Reprodução de Conteúdos

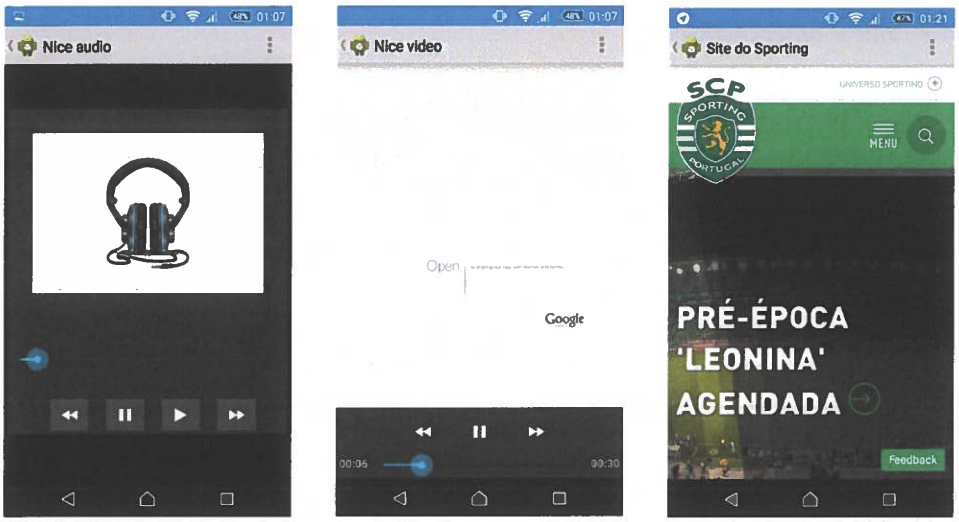


Figura 14 - Reprodução de Áudio, Reprodução de Vídeo e Browser

Existem três diferentes tipos de conteúdos que a aplicação Android suporta – Áudio, Vídeo ou navegação para um determinado URL.

Para cada um destes tipos foi criado uma *Activity* diferente, sendo que todas elas herdam de *ContentBaseActivity* a fim de evitar a repetição de código.

4.3.4.4.1 AudioContentActivity

Para a reprodução de áudio foi utilizado o *MediaPlayer* da Android SDK, que disponibiliza todas as funcionalidades necessárias a essa reprodução, nomeadamente as funções de manipulação de áudio como *Start*, *Pause*, *Stop* e *Seek*.

Um desafio interessante durante o desenvolvimento desta *activity*, foi o de lidar com a rotação do ecrã do *smartphone*, pois quando o ecrã roda, é novamente chamado o método *OnCreate* da *activity*, o que fazia com que o áudio que se estivesse a ouvir fosse reiniciado. Para ultrapassar este problema foi necessário criar uma variável (*timeElapsed*) na qual vai sendo guardado o tempo atual do áudio a cada segundo, e utilizá-la no *override* dos métodos *OnSaveInstanceState* e *OnRestoreInstanceState*, guardando no primeiro o valor no estado da instância, e no segundo restaurando esse valor. Assim, no método *onCreate* quando é configurado o *MediaPlayer* do ficheiro a reproduzir, é também chamado o método *Seek* para o

valor atual da variável *timeElapsed*, sendo que na primeira vez que é iniciada a *activity* este valor será sempre zero.

4.3.4.4.2 VideoContentActivity

Tal como foi utilizado o *MediaPlayer* da Android SDK na reprodução de áudio, de igual modo fez-se a mesma utilização para a reprodução de vídeo, pois a classe suporta os dois formatos. Apenas foi necessário incluir no layout da *activity* uma *VideoView* de forma a ser exibida a imagem do vídeo.

Nesta *activity* foi também necessário lidar com a rotação de ecrã explicada na secção anterior, pelo que foi seguido os mesmos moldes de programação na resolução deste problema.

4.3.4.4.3 WebContentActivity

Devido ao facto da Android SDK já possuir uma *WebView*, o desenvolvimento desta *activity* foi bastante simples, uma vez que foi só fazer uso da mesma, utilizando o método *LoadUrl* no método *OnCreate* da *activity*.

4.3.4.5 ET05 – Deteção de beacons

Trata-se do requisito e objetivo mais importante da aplicação. Das APIs para deteção de *beacons* mais utilizadas pelas comunidades de programadores Android e IOS, optou-se por se utilizar a AltBeacon Library, pois apresenta as seguintes vantagens em relação à Estimote SDK:

- ***Open source*** – sem segredos sobre como funciona, pode ser alterado para colmatar alguma necessidade futura que não esteja contemplada;
- **Compatível com diferentes fabricantes de *beacons*** – desenhada para ser agnóstica à marca de *beacons* e formato de transmissão, portanto é compatível com as marcas mais conhecidas como AltBeacon, Eddystone ou iBeacon, como também com qualquer outra marca, pois basta configurar na sua inicialização quais os formatos (IDs) dos fabricantes ao qual se pretende detetar a presença.
- **Bastante utilizada** – é usada por 4000 aplicações no Google Play e já foi instalada em mais de 150 milhões de dispositivos móveis.
- **Disponível em Android e IPhone** – Em comum com a Estimote SDK, também esta está disponível tanto para Android como para IPhone, ainda que atualmente

saia fora do âmbito de desenvolvimento atual deste produto a criação de uma aplicação para IPhone, futuramente será mais fácil a sua integração, pois irá usar a mesma API para comunicação com *beacons* que a aplicação Android usa.

- **Estimativas de distância** - Outra grande vantagem é o facto de esta disponibilizar a distância em metros (ainda que apenas sejam estimativas) ao qual o *smartphone* se encontra do *beacon*. Esta funcionalidade poderá revelar-se bastante interessante se, por exemplo, for colocado um *beacon* junto a um quadro específico do museu, e configurado para que o conteúdo associado comece a tocar apenas quando o *smartphone* se encontrar a 1m do mesmo, possibilitando assim uma forma de criar conteúdos para quadros específicos e não apenas a uma zona do museu.

Neste sentido, foi criada a classe *ProximityBeaconService* que herda de *Service* da Android SDK, e fez-se uso do *namespace AltBeaconOrg.BoundBeacon* que nos providência um conjunto de classes e métodos para a deteção de *beacons*.

O método *OnStartCommand* de *Service* é chamado quando alguma classe que fez *Bind* deste serviço realiza uma chamada ao método *StarService*. Sendo assim, foi feito *override* do mesmo, de modo a fazer as respetivas inicializações das propriedades da classe.

```
1 reference
public override StartCommandResult OnStartCommand(Intent intent, StartCommandFlags flags, int startId)
{
    Log.Debug(LOG_TAG, "BeaconService started");
    ActiveBeacons = new List<ActiveBeacon>();

    estimoteRegion = new Region("EstimoteBeconsRegion", Identifier.Parse("B9407F30-F5F8-466E-AFF9-25556B57FE6D"), null, null);

    CurrentActivityStateReceiver = new ActivityStateReceiver();
    CurrentActivityStateReceiver.OnActivityStateChanged += CurrentActivityStateReceiver_OnActivityStateChanged;

    RegisterReceiver(CurrentActivityStateReceiver, new IntentFilter("ActivityStateChanged"));

    IsAppInForeground = true;
    ChangeMode(CurrentMode);

    StartActiveBeaconsTimer();

    return StartCommandResult.Sticky;
}
```

Figura 15 - ProximityBeaconService OnStartCommand

Neste método é registado o identificador dos *beacons* da Estimote que foram utilizados durante o desenvolvimento deste projeto. Os *beacons* são identificados por um identificador único do fabricante (neste caso Estimote - B9407F30-F5F8-466E-AFF9-25556B57FE6D), um *Major ID* e um *Minor ID*. Ao passar *null* nos argumentos de *Major ID* e *Minor ID*, fará com que qualquer

beacon da Estimote seja detetado. Se futuramente forem utilizados *beacons* de outro fabricante, bastará adicionar mais uma *Region* com o identificador único do mesmo.

A deteção de *beacons* faz-se verificando a cada intervalo de tempo configurável se algum *beacon* está por perto. O facto de estar a fazer este processamento constantemente poderá levar a um desgaste rápido da bateria. Esta foi sempre uma grande preocupação no desenvolvimento deste projeto. De modo a responder a esta necessidade, foram implementadas as seguintes alterações:

- Na *ProximityBaseActivity*, da qual todas as *activities* da aplicação herdam, foi feito o *override* dos métodos Pause e Resume da *Activity*, no sentido de saber se a aplicação está a ser visível ou não pelo utilizador.
- Foi criado um *BroadcastReceiver* para enviar esta notificação de alteração de estado e feito o registo do mesmo no método *OnStartCommand*, implementando o *handler OnActivityStateChanged* criado para quando essa alteração de estado for feita.
- Foi criada a variável *CurrentMode* que se trata de *enum* chamado de *ProximityBeaconServiceMode* o qual contém dois modos: *Background* e *Foreground*. No *handler OnActivityStateChanged* é chamado o método *ChangeMode* para o modo correspondente ao estado da aplicação.

```
private void ChangeMode(ProximityBeaconServiceMode toMode)
{
    Log.Debug(LOG_TAG, string.Format("BeaconService going {0} mode...", toMode == ProximityBeaconServiceMode.Background ? "Background" : "Foreground"));
    CurrentMode = toMode;
    if (beaconManager != null)
    {
        //beaconManager.StopMonitoringBeaconsInRegion(estimoteRegion);
        if(beaconManager.IsAnyConsumerBound)
            beaconManager.StopRangingBeaconsInRegion(estimoteRegion);
        if (beaconManager.IsBound(this))
            beaconManager.Unbind(this);
        beaconManager = null;
    }
    InitializeBeaconManager(toMode);
}
```

Figura 16 - Método ChangeMode

A classe *BeaconManager* é a classe da AltBeacon responsável pela deteção de *beacons*. Na sua inicialização é possível indicar qual o intervalo de tempo no qual deve ser feita monitorização de *beacons*. Foi configurado um intervalo de 2s quando o modo for *Forground* e 30s quando estiver em Background. Na inicialização de *BeaconManager* é igualmente possível indicar qual o *RangeNotifier*, neste caso será a própria *ProximityBeaconService*, pois implementa a interface *IRangeNotifier*.

Esta interface obriga à implementação do método *DidRangeBeaconsInRegion* que é o método chamado aquando alguma deteção de *beacons*.

Este método recebe os *beacons* detetados e em que região. Para cada *beacon* detetado será feito o seu processamento no sentido de identificá-lo na base de dados para assim poder chamar o conteúdo correspondente, não sem antes registar a presença na base de dados e notificar aos interessados à deteção efetuada, pelo método *UpdateAndBroadcastActive* Beacon.

```
private async Task ProcessDetectedBeacon(Beacon beacon, string completeId)
{
    await Task.Run(() =>
    {
        try
        {
            ProximityBeacon proximityBeacon = ProximityBeaconFactory.Instance.GetProximityBeacon(beacon, true);

            if (proximityBeacon.Id == null)
            {
                Log.Debug(LOG_TAG, "ProcessDetectedBeacon: unknown Beacon!! " + proximityBeacon.CompleteId);
                return;
            }

            if (string.IsNullOrEmpty(androidId))
                androidId = ConfigHelper.SmartphoneId;

            ProximityBeaconDetection detection = new ProximityBeaconDetection()
            {
                DetectedDate = DateTime.Now,
                Distance = Convert.ToDecimal(beacon.Distance),
                ProximityBeaconId = proximityBeacon.Id,
                SmartphoneId = androidId
            };

            UpdateAndBroadcastActiveBeacon(detection, completeId);

            Content content = DBA.Instance.GetContentByProximityBeaconId(proximityBeacon.Id);
            if(content == null)
            {
                throw new Exception("Could not find content by beacon Id: " + proximityBeacon.Id);
            }

            //check if user needs to be notified about this beacon
            if (content.DateSeen == null && content.DateAlerted == null &&
                (lastAlertedDate == null || (lastAlertedDate < DateTime.Now.AddSeconds(-3))) &&
                (proximityBeacon.MaxDistance == null || detection.Distance < proximityBeacon.MaxDistance))
            {
                if(content.DownloadStatusId == ProximityConstants.DownloadStatus.Downloaded)
                {
                    lastAlertedDate = DateTime.Now;
                    content.DateAlerted = DateTime.Now;
                    DBA.Instance.UpdateContent(content);

                    Intent intent = PrepareIntent(content, CurrentMode);
                    StartActivity(intent);
                }
            }
        }
        catch (Exception ex)
        {
            DBA.Instance.InsertProximityContentLog(ex, LOG_TAG);
        }
    });
}
```

Figura 17- Método ProcessDetectedBeacon

4.3.4.6 ET06 – Sincronismo de informação

É feito através da classe *SyncService* que disponibiliza o método *Sync*. Neste método é invocado o método *GetSyncUpdate* do *webservice ProximityContentService.scv*, passando-lhe uma data desde o qual deve ser feito esse sincronismo.

```
public void Sync(Java.IO.File externalFilesDir)
{
    if (IsSyncing)
        return;

    IsSyncing = true;

    try
    {
        DateTime updateDate = ConfigHelper.LastSyncDate;

        SyncUpdate syncUpdate = ProximityRestService.Instance.GetSyncUpdate(updateDate);

        if (syncUpdate.ReferenceDatas != null && syncUpdate.ReferenceDatas.Any())
            ProccessSyncableEntity(syncUpdate.ReferenceDatas, updateDate);
        if (syncUpdate.Clients != null && syncUpdate.Clients.Any())
            ProccessSyncableEntity(syncUpdate.Clients, updateDate);
        if (syncUpdate.ProximityBeacons != null && syncUpdate.ProximityBeacons.Any())
            ProccessSyncableEntity(syncUpdate.ProximityBeacons, updateDate);
        if (syncUpdate.Contents != null && syncUpdate.Contents.Any())
            ProccessSyncableEntity(syncUpdate.Contents, updateDate);

        SendAllDetections();

        SendAllLogs();

        SyncDownloads(externalFilesDir);

        ConfigHelper.LastSyncDate = DateTime.Now;
    }
    catch(Exception ex)
    {
        throw ex;
    }
    finally
    {
        IsSyncing = false;
    }
}
```

Figura 18 - Método Sync

Para cada lista de entidades passíveis de serem sincronizadas, sejam elas "Clientes", "Beacons", "Conteúdos" ou "ReferenceData"[19], todas elas serão processadas no método *ProcessSyncableEntity*. Todas estas entidades herdam da classe *Syncable*, de forma a reunir um conjunto de propriedades comuns, no sentido do seu sincronismo ser uniforme e independentemente do tipo de entidade.

[19] Informação de referência como tipos de conteúdos, tipos de *status* de *download*, etc.

```
public abstract class Syncable
{
    [PrimaryKey, AutoIncrement]
    42 referências
    public System.Guid Id { get; set; }
    6 referências
    public DateTime DateCreated { get; set; }
    7 referências
    public DateTime DateUpdated { get; set; }
    8 referências
    public bool IsDeleted { get; set; }
}
```

Figura 19 - Classe abstrata Syncable

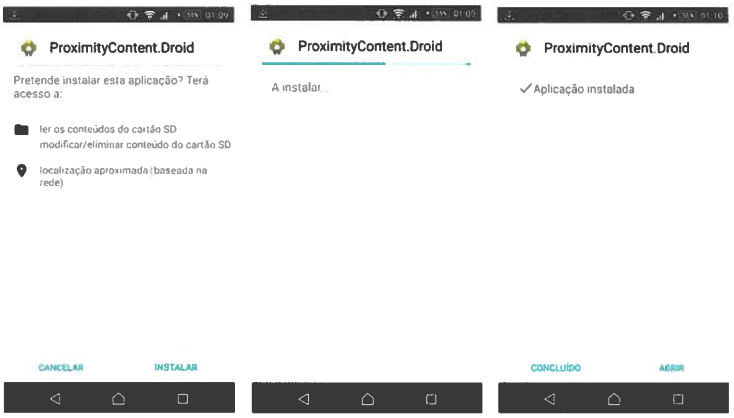
Ao conter as propriedades *Id, DateCreated, DateUpdated* e *IsDeleted*, permite tomar as ações necessárias durante o sincronismo de cada entidade, de forma a avaliar se será necessário criar, atualizar ou eliminar a mesma.

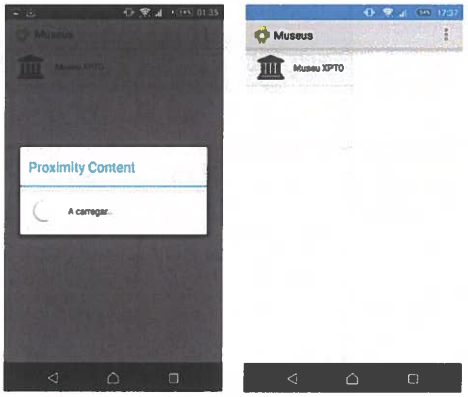
4.3.5 Casos de Teste

Em engenharia de software, os casos de teste permitem determinar se uma aplicação, ou determinada funcionalidade desta, tem o resultado esperado para o qual foi desenhada. De forma a garantir que o componente principal do produto cumpre os requisitos expostos, foram desenhados seis casos de teste, cada um deles assegura o sucesso de uma funcionalidade crítica da aplicação.

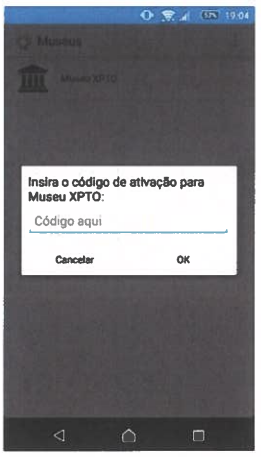
4.3.5.1 CT01 – Instalação e arranque da aplicação

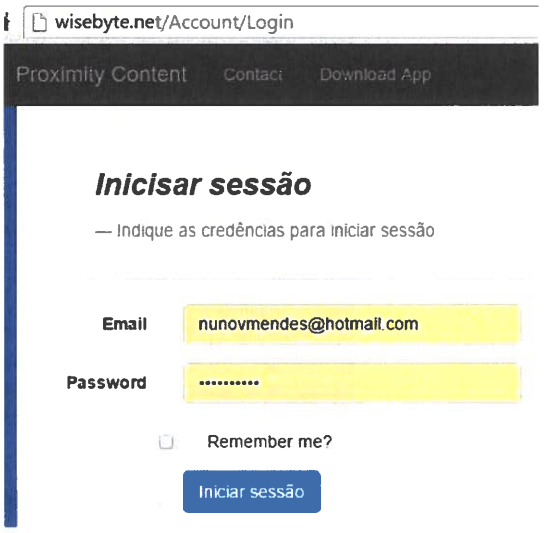
Requisitos ou dependências	• Um smartphone Android com versão de sistema operativo superior a 4.3. • Ligação à internet. • Ativar nas opções de sistema a possibilidade de instalar aplicações de origem desconhecida.	
ID Passo	Ação	Resultado Esperado
1	1. Aceder ao endereço http://wisebyte.net/Home/DownloadApp 2. Descarregar o ficheiro ProximityContent.apk	No site de gestão de conteúdos deverá estar disponível para download o ficheiro apk de instalação da aplicação.

	Resultado Efetivo	
	 	
ID Passo	Ação	Resultado Esperado
2	1. Executar o ficheiro ProximityContent.apk de forma a instalar a aplicação	Ao executar o ficheiro deverá ser perguntado ao utilizador se concorda com as autorizações de acesso da aplicação. A aplicação deverá ser instalada com sucesso.
	Resultado Efetivo	
		

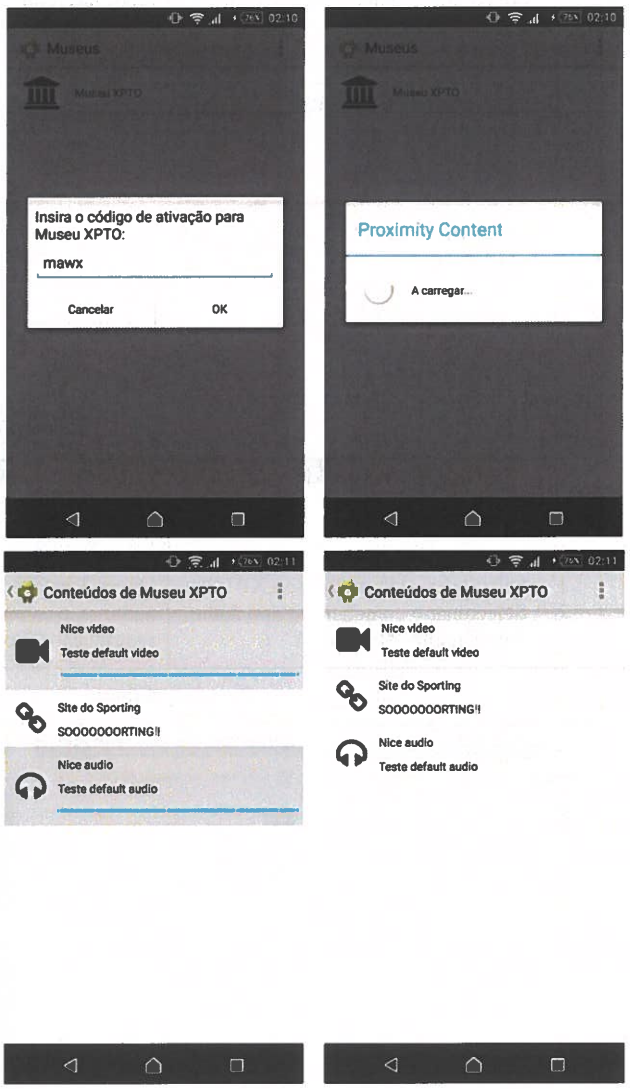
ID Passo	Ação	Resultado Esperado
3	1. Abrir a aplicação.	A aplicação deverá abrir com sucesso. No primeiro carregamento deverá ser mostrada uma mensagem de progresso, enquanto é descarregada a BD SQLite da aplicação e é feito o primeiro sincronismo da mesma. Deverão ser listados os clientes existentes na aplicação. Neste momento apenas deverá aparecer o museu com o nomoe Museu XPTO.
	Resultado Efetivo	
		

4.3.5.2 CT02 – Validação de código ativação e download de conteúdos

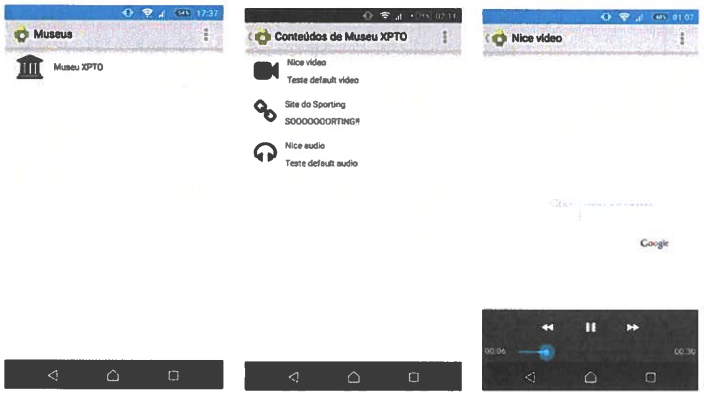
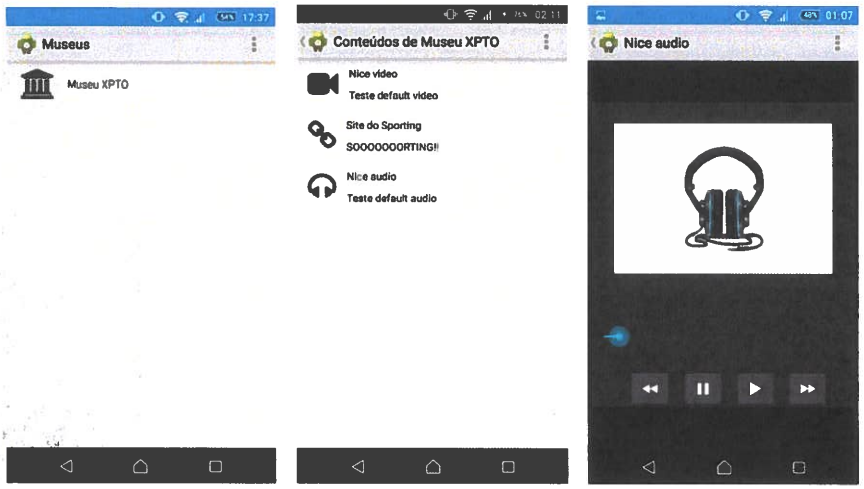
Requisitos ou dependências	• CT01 efetuado com sucesso. • Ligação á internet.		
ID Passo	Ação	Resultado Esperado	Resultado Efetivo
1	1. Abrir a aplicação. 2. Selecionar o cliente Museu XPTO	Ao selecionar o cliente deverá ser requisitado ao utilizador o código de ativação para o Museu XPTO	

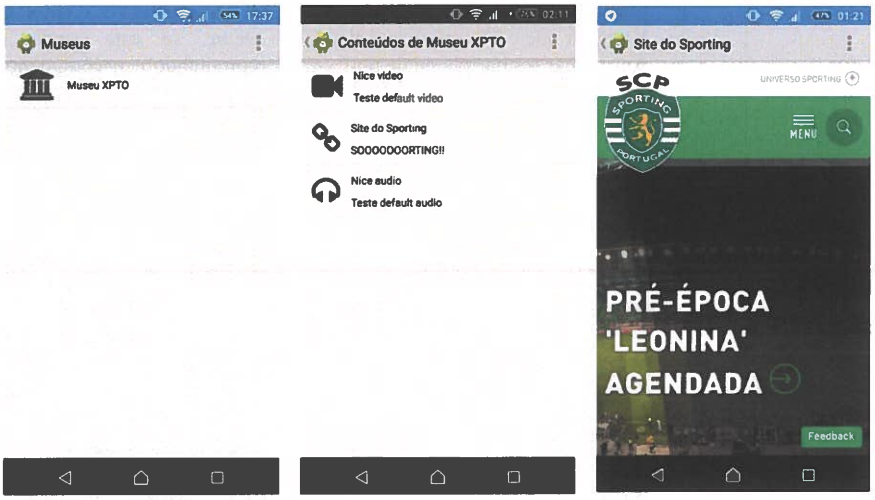
<table>
<tr><th>ID Passo</th><th>Ação</th><th>Resultado Esperado</th></tr>
<tr><td rowspan="3">2</td><td>Aceder ao site de gestão de conteúdos.
Iniciar sessão com um utilizador válido.
Na gestão de Códigos de Ativação gerar novo código.</td><td>No site gestão de conteúdos, um utilizador autenticado deverá conseguir gerar códigos de ativação.</td></tr>
<tr><td colspan="2">Resultado Efetivo</td></tr>
<tr><td colspan="2"> 


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

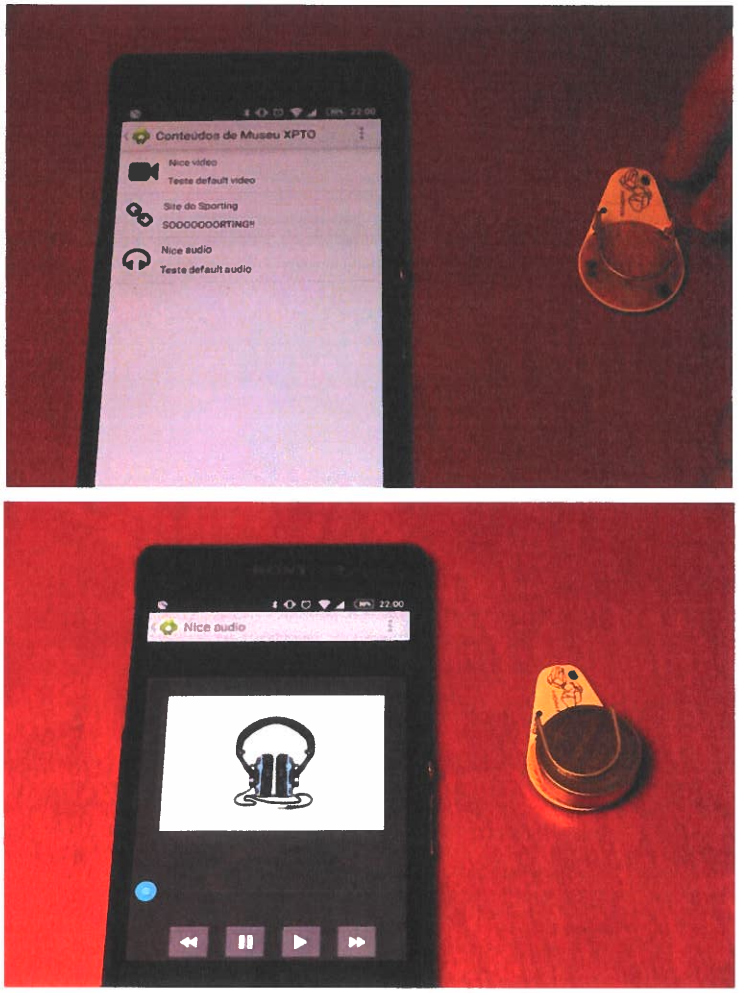
ID Passo	Ação	Resultado Esperado
3	1. Utilizar o código de ativação gerado no passo anterior na aplicação Android.	O código deverá ser validado com sucesso. Deverá ser aberta a janela de conteúdos do cliente XPTO. O descarregamento dos conteúdos derá ser inicializado nesta altura.
	Resultado Efetivo	
		

4.3.5.3 CT03 – Janelas diferentes para cada tipo de conteúdo

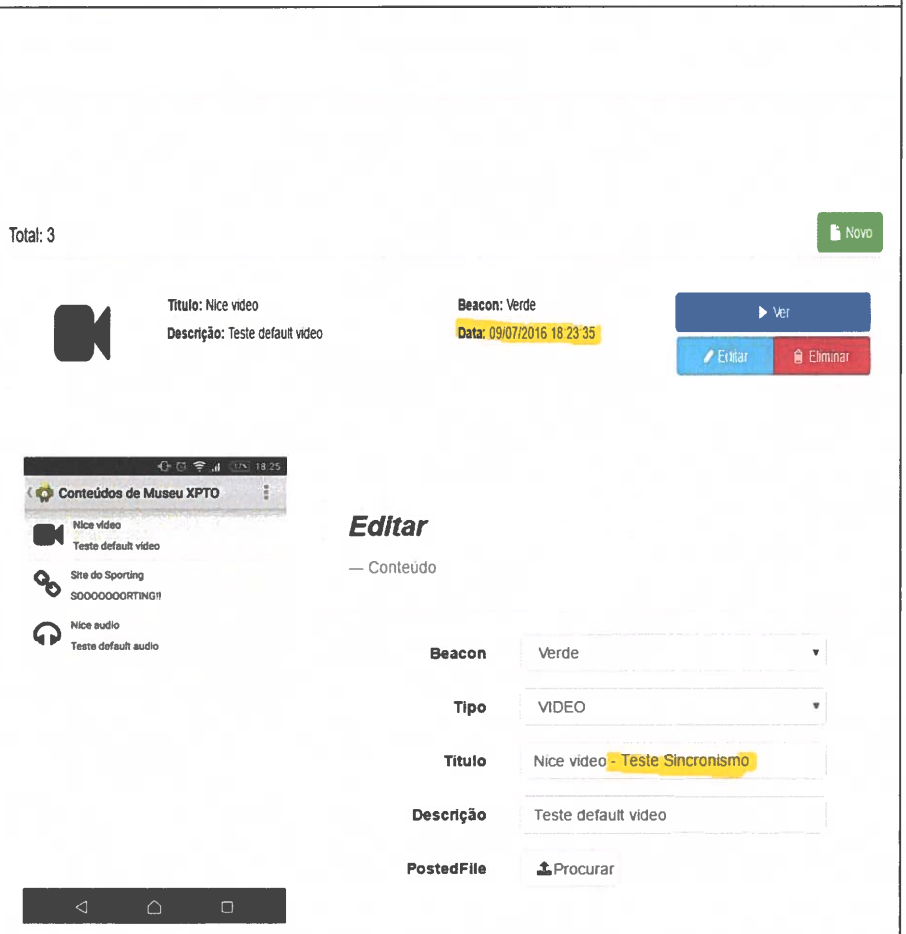
Requisitos ou dependências	• CT01 e CT02 efetuados com sucesso.	
ID Passo	Ação	Resultado Esperado
1	1. Selecionar o cliente Museu XPTO 2. Selecionar o conteúdo do tipo vídeo com o nome "Nice Video"	Deverá ser aberta a janela de reprodução de vídeo e o será iniciado automaticamente a reprodução do conteúdo.
	Resultado Efetivo	
		
ID Passo	Ação	Resultado Esperado
2	1. Selecionar o cliente Museu XPTO 2. Selecionar o conteúdo do tipo áudio com o nome "Nice audio"	Deverá ser aberta a janela de reprodução de áudio e o será iniciado automaticamente a reprodução do conteúdo.
	Resultado Efetivo	
		

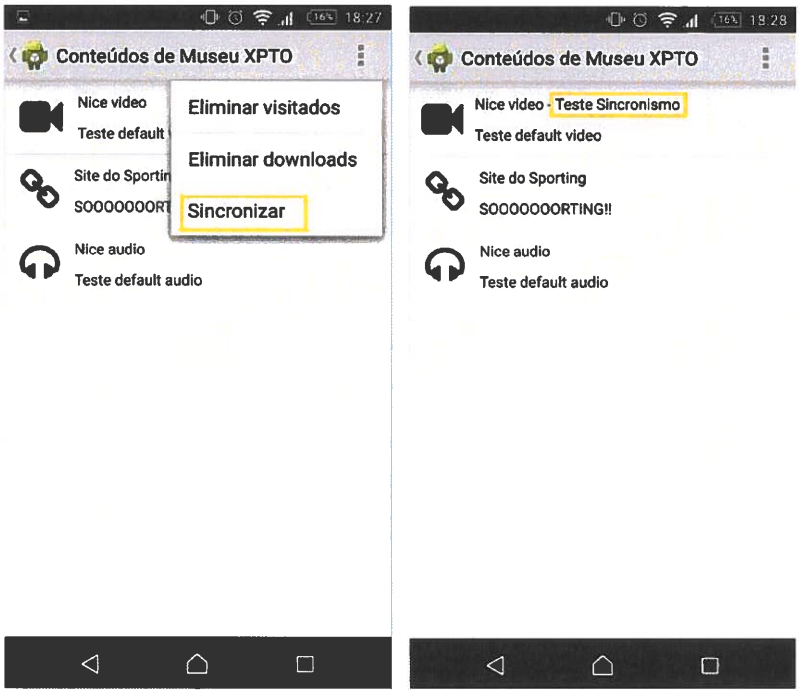
ID Passo	Ação	Resultado Esperado
3	1. Selecionar o cliente Museu XPTO 2. Selecionar o conteúdo do tipo Web com o nome "Site do Sporting"	Deverá ser aberta uma nova janela e carregado o website do Sporting C.P.
	Resultado Efetivo	
		

4.3.5.4 CT04 – Reprodução de conteúdo por proximidade de beacon

<table>
<tr><td>Requisitos ou dependências</td><td colspan="2">• CT01 e CT02 efetuados com sucesso</td></tr>
<tr><td>ID Passo</td><td>Ação</td><td>Resultado Esperado</td></tr>
<tr><td rowspan="3">1</td><td>1. Entrar na aplicação.
2. Verificar que o Bluetooth está ligado.
3. Aproximar o smartphone de um beacon</td><td>O Smartphone ao detetar a presença do becon, se este ainda não tiver sido reproduzido e o código de ativação estiver válido, deverá ser reproduzido o conteúdo correspondente ao beacon que se fez a aproximação.</td></tr>
<tr><td colspan="2">Resultado Efetivo</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

4.3.5.5 CT05 – Sincronismo de informação

Requisitos ou dependências	• CT01 e CT02 efetuados com sucesso	
ID Passo	Ação	Resultado Esperado
1	1. Aceder ao site de gestão de conteúdos. 2. Editar o conteúdo de forma a alterar o título "Nice Viideo" para "Nice Video – Teste Sincronismo".	Será alterado o título do conteúdo no site de gestão de conteúdos.
	Resultado Efetivo	
		

ID Passo	Ação	Resultado Esperado
2	1. Na aplicação aceder ao menu no topo. 2. Pressionar o botão "Sincronizar".	O título do conteúdo será alterado de "Nice Video" para "Nice Video – Teste Sincronismo"
	Resultado Efetivo	
		

4.4 Website – Gestor de Conteúdos

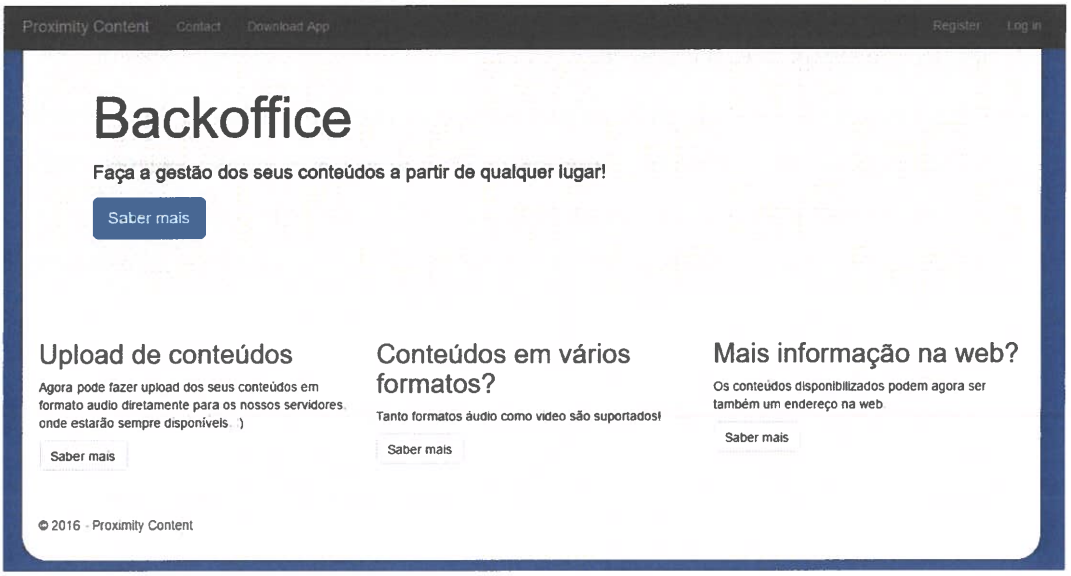


Figura 20 - Homepage do site de gestão de conteúdos

O facto de alguns museus mudarem com alguma regularidade as suas exposições, quer seja no seu conteúdo ou na sua disposição, e ainda que não seja o foco do projeto, um website para o cliente (museu) poder gerir os seus conteúdos, tornou-se uma necessidade implícita no desenvolvimento deste produto.

Optou-se por se utilizar a tecnologia ASP.NET MVC devido a esta ser mais leve e mais extensível em relação à ASP.NET Web Forms, tal como verificado no capítulo Estado da Arte - ASP.NET MVC deste documento.

4.4.1 Tecnologias, ferramentas e/ou referências utilizadas

- Visual Studio 2015 – IDE da Microsoft para desenvolvimento de aplicações.
- ASP.NET Identity – API para gestão de utilizadores.
- MVC Foolproof Validation – API que estende a Data Annotation de ASP.NET MVC para providenciar mais mecanismos nas validações de campos dos formulários.
- Elmah – API para logging.
- MediaTypeMap – API que providencia um enorme dicionário de mapeamento entre tipos de ficheiros e o seu mime type.

- Font Awesome – Vários icons customizáveis na sua cor e/ou tamanho, e disponíveis sem qualquer custo.
- Bootstrap – Framework de CSS.
- JQuery – Biblioteca de extensão do Javascript.
- EntityFramework – API para mapeamento de objetos relacionais que permite programadores .NET trabalhar com informação relacional usando classes .NET. Elimina a necessidade para grande parte do código de acesso a bases de dados.
- AutoMapper – API para mapeamento de objetos, usada neste caso para mapeamento de objetos da entity framework em DTOs[20].
- PagedList – API para paginação de objetos.
- Newtonsoft.Json – API para serializar objetos em formato json e *vice-versa.*

4.4.2 Especificação de Requisitos

ID REQ	DESCRIÇÃO REQUISITO
REQ01	CRUD operations sobre Conteúdos por um funcionário do museu.
REQ02	Associação de beacons a conteúdos por um funcionário do museu.
REQ03	Gerar códigos de ativação por um funcionário do museu.
REQ04	CRUD operations sobre Beacons por um administrador do site

Tabela 3 - Website - Especificação de requisitos

4.4.3 Matriz de rastreabilidade

ID REQ	ID ET	Nome Especificação Técnica
REQ01	ET01	Gestão de Conteúdos
REQ02	ET01	Gestão de Conteúdos
REQ03	ET02	Gestão de Códigos de Ativação
REQ04	ET04	Administração de beacons

Tabela 4 - Website - Matriz de rastreabilidade

[20] Data Transfer Object – É um objeto que contém unicamente as propriedades necessárias para serem usadas na comunicação entre camadas da aplicação ou diferentes aplicações.

4.4.4 Especificação Técnica

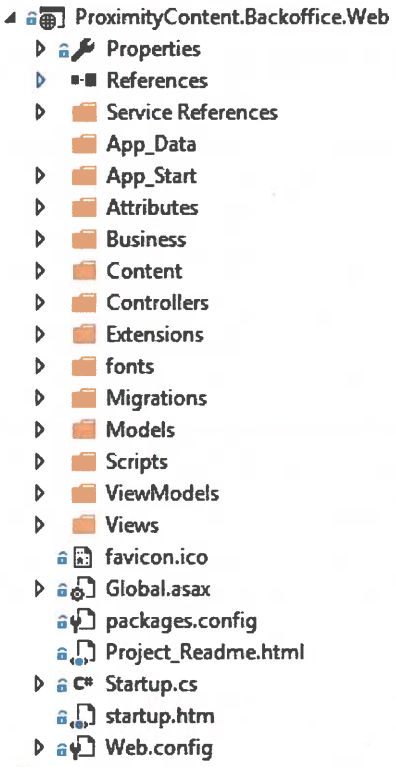


Figura 21 - Projeto ProximityContent.Backoffice.Web no Visual Studio

O website de gestão de conteúdos foi desenvolvido utilizado a tecnologia ASP.NET MVC. As pastas Models, Views e Controllers correspondem à separação de conceitos deste paradigma. Para além destas pastas próprias do MVC, foi criada a pasta Business que corresponde à camada de negócio da aplicação, onde se encontra por exemplo a classe DbHelper, para chamadas à base de dados por via da Entity Framework, e a classe StorageService que lida com a gestão de ficheiros dos conteúdos áudio ou vídeo.

De modo a providenciar restrição de acessos a determinadas funcionalidades do site, foram criadas três áreas distintas:

- Pública – Acedido por qualquer pessoa, tem unicamente acesso a páginas que não necessitem de identificação, como a Homepage, Download App e Contactos.

- Restrita – Acedido por funcionários dos museus, para além das páginas da zona pública, tem também acesso às páginas "Beacons", "Conteúdos" e "Códigos de Ativação".
- Administração – Acedido pelos administradores do site, providencia acesso às páginas "Clientes", "Beacons", "Deteções" e "Logs".

Esta restrição de acessos foi feita através do atributo *Authorize*, especificando qual o Role que terá acesso. Ao incluir este atributo na declaração do *Controller*, automaticamente só utilizadores autenticados é que terão acesso às *Actions* nesse *Controller*.

```
namespace ProximityContent.Backoffice.Web.Controllers
{
    [Authorize(Roles = "Admin")]
    0 references | Nuno Mendes, 48 days ago | 1 author, 2 changes | 1 work item
    public class ClientsController : BaseController
    {
```

```
namespace ProximityContent.Backoffice.Web.Controllers
{
    [Authorize]
    0 references | Nuno Mendes, 22 days ago | 1 author, 11 changes | 2 work items
    public class ContentsController : BaseController
    {
```

Figura 22 - Restrição de acesso a Administradores e a utilizadores autenticados

4.4.4.1 ET01 - Gestão de conteúdos

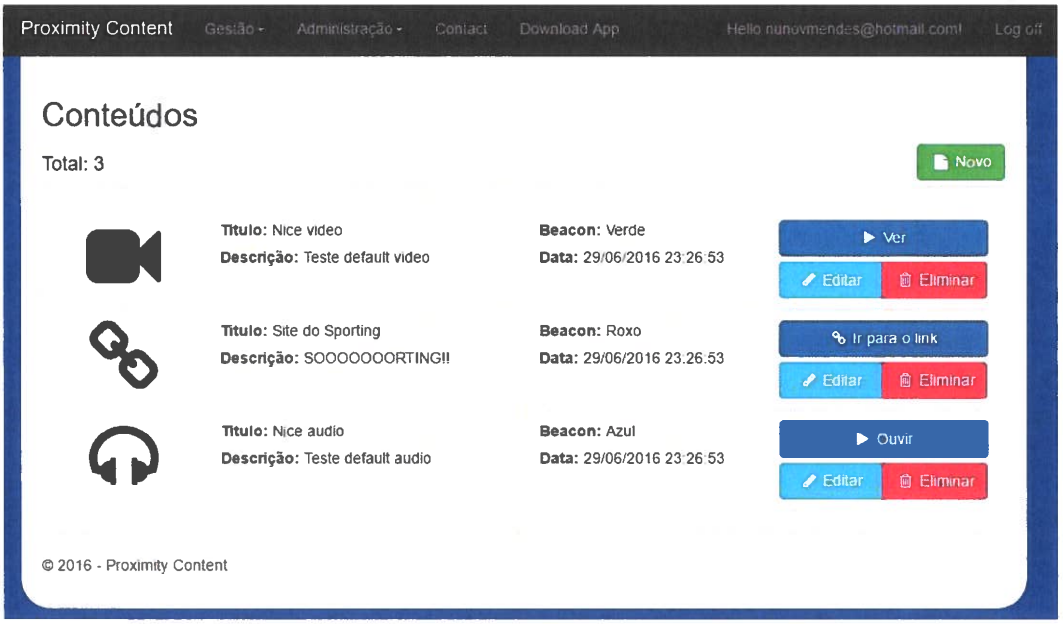


Figura 23 - Página de listagem de conteúdos

4.4.4.1.1 Listagem de conteúdos

Acedida unicamente por utilizadores autenticados, a página inicial é a que lista os conteúdos de determinado cliente. A apresentação de cada linha difere consoante o tipo de conteúdo ao qual diz respeito, deste modo, se for um vídeo, é exibido o ícon correspondente e nos botões de ações é disponibilizado o botão "Ver". Ao pressionar este botão a linha é expandida para a visualização do respetivo vídeo.

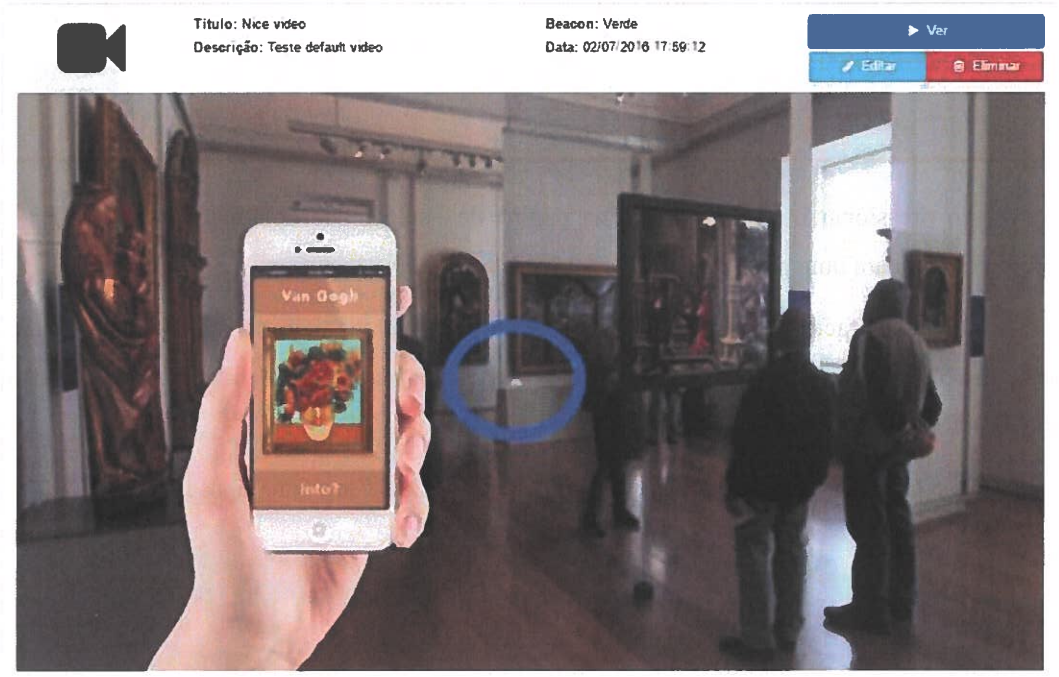


Figura 24 - Exibição de vídeo

De igual modo, para o tipo áudio, o botão "Ouvir" expande a linha para a audição do respetivo conteúdo.

Figura 25 - Audição de conteúdo

4.4.4.1.2 Criação de conteúdos

Figura 26 - Criação de conteúdo

Ao pressionar o botão "Novo" na página de listagem de conteúdos, o utilizador é redirecionado para uma nova página de criação de conteúdo.

Devido ao facto de um conteúdo ter obrigatoriamente que ficar associado a um beacon, nesta criação o utilizador terá que indicar qual o beacon correspondente, dando assim resposta ao REQ02 desta aplicação.

Se o cliente não dispor de mais beacons, será exibida a seguinte mensagem.

Figura 27 - Aviso para indisponibilidade de beacons

Em termos de validações de campos, foram utilizadas DataAnnotations que permitem validação tanto em cliente side como server side. Apenas terá que ser declarado o atributo correspondente à validação que se pretende o modelo, nomeadamente atribuir-lhe por exemplo o atributo "Required".

```
[Required]
[Display(ResourceType = typeof(ModelsLocalization), Name = "Beacon")]
0 references | Nuno Mendes, 36 days ago | 1 author, 2 changes | 1 work item
public System.Guid? ProximityBeaconId { get; set; }
```

Figura 28 - Validação de campos utilizando DataAnnotations

Foi ainda utilizado a API FoolproofValidation que extende estas anotações providenciando outros atributos como o RequiredIf ou RequiredIfNot

```
[RequiredIfNot("ContentTypeId", "2b607f7c-24b6-44af-b474-1a6357d4fedd", ErrorMessage = "Ficheiro é obrigatório")]
3 references | Nuno Mendes, 36 days ago | 1 author, 2 changes | 1 work item
public string UploadTempFilename { get; set; }
```

Figura 29 - Validação de campos utilizando a API FoolproofValidation

4.4.4.1.2.1 Referências

- Controller: ContentsController
- Action:
  - ActionResult Create(Guid? id) para disponibilização da view.
  - ActionResult Create([Bind(Include = "...")]content) para gravar o conteúdo na base de dados.
- Model: CreateContentViewModel

4.4.4.1.3 Edição de conteúdos

Figura 30 - Página de edição de conteúdo

A página de edição de um determinado conteúdo é muito semelhante à de criação apenas as validações mudam, pois passa a não ser obrigatório o carregamento de um ficheiro, de modo a que o cliente possa por exemplo apenas alterar o título a descrição do conteúdo sem que necessite alterar o próprio ficheiro.

4.4.4.1.3.1 Referências

- Controller: ContentsController
- Action:
  - ActionResult Edit(Guid? id) para disponibilização da view.
  - ActionResult Edit([Bind(Include = "...")]content) para gravar o conteúdo na base de dados.
- Model: EditContentViewModel

4.4.4.1.4 Eliminação de conteúdos

Poderá ser feito através do botão "Eliminar" que se encontra na página de listagem de conteúdos. Ao ser pressionado, é questionado o utilizador se realmente pretende eliminar o conteúdo, de forma a prevenir eliminações indesejadas de conteúdos.

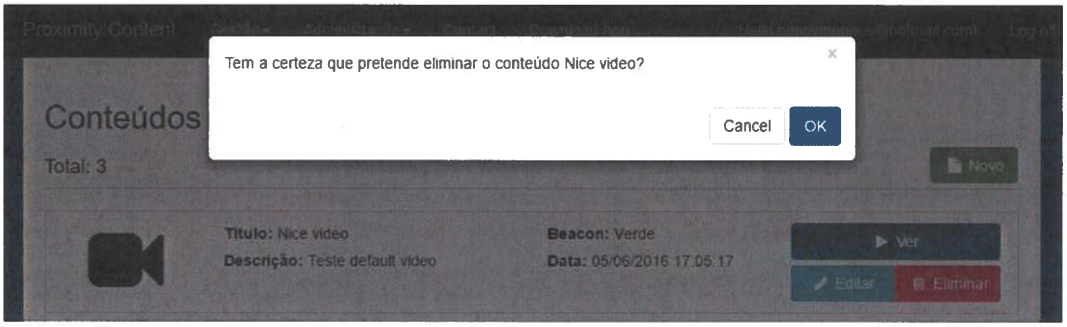


Figura 31 - Mensagem para confirmação de eliminação de conteúdo

Figura 32 - Mensagem de sucesso na eliminação do conteúdo

4.4.4.1.4.1 Referências

- Controller: ContentsController
- Action: DeleteConfirmed(string id)
- Javascript na view Index: function deleteContent(idToDelete, title) para realizar um pedido ajax ao controller e action indicados.

4.4.4.2 ET02 - Gestão de códigos de ativação

Figura 33 - Página de listagem de códigos de ativação

Igualmente acedido unicamente por utilizadores autenticados, esta página permite que um funcionário de um museu tenha a possibilidade de gerar códigos de ativação para disponibilização dos conteúdos a um determinado *smartphone*. Isto irá permitir ter não só uma forma de cobrança pelos conteúdos de determinado museu, como também uma forma de contabilizar a utilização da aplicação pelos utilizadores finais.

Os códigos são constituídos por quatro caracteres alfanuméricos e são gerados utilizado o RNGCryptoServiceProvider do namespace System.Security.Cryptography pertencente à própria .NET Framework.

Esta classe proporciona uma forma de obter um subconjunto de carateres aleatórios de dimensão parametrizável, a partir de um conjunto de carateres previamente definidos. Abaixo o código necessário para essa implementação.

```
readonly char[] AvailableCharacters = {
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
    'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
    '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'
  };

1 reference | Nuno Mendes, 40 days ago | 1 author, 1 change | 1 work item
private string GenerateIdentifier(int length)
{
    char[] identifier = new char[length];
    byte[] randomData = new byte[length];

    using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
    {
        rng.GetBytes(randomData);
    }

    for (int idx = 0; idx < identifier.Length; idx++)
    {
        int pos = randomData[idx] % AvailableCharacters.Length;
        identifier[idx] = AvailableCharacters[pos];
    }

    return new string(identifier);
}
```

Figura 34 - Código para geração de carateres aleatórios

Ao pressionar o botão "Gerar novo código", são primeiramente desabilitados os códigos entretanto já expirados, de forma a não serem exibidos na lista de códigos que já não estejam em uso. Seguidamente é chamado o método acima através do *CodeActivationService*, método *GenerateCode*.

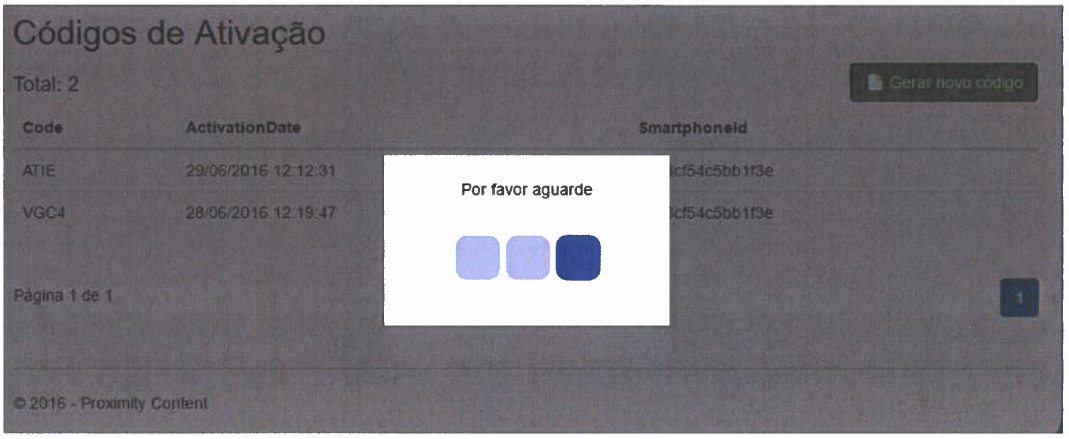


Figura 35 - Geração de código de ativação

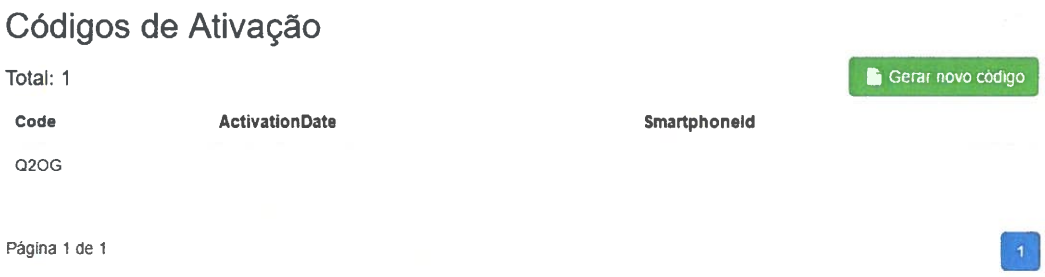


Figura 36 - Código gerado

De futuro este código poderá ser um QRCode que os utilizadores finais poderão ler de um monitor, evitando assim algum erro de introdução do código por parte dos mesmos.

4.4.4.2.1.1 Referências

- Controller: ActivationCodesController
- Action: GenerateCode ()
- Javascript na view Index: function generateCode () para realizar um pedido ajax ao controller e action indicados.

4.4.4.3 ET03 - Gestão de Beacons

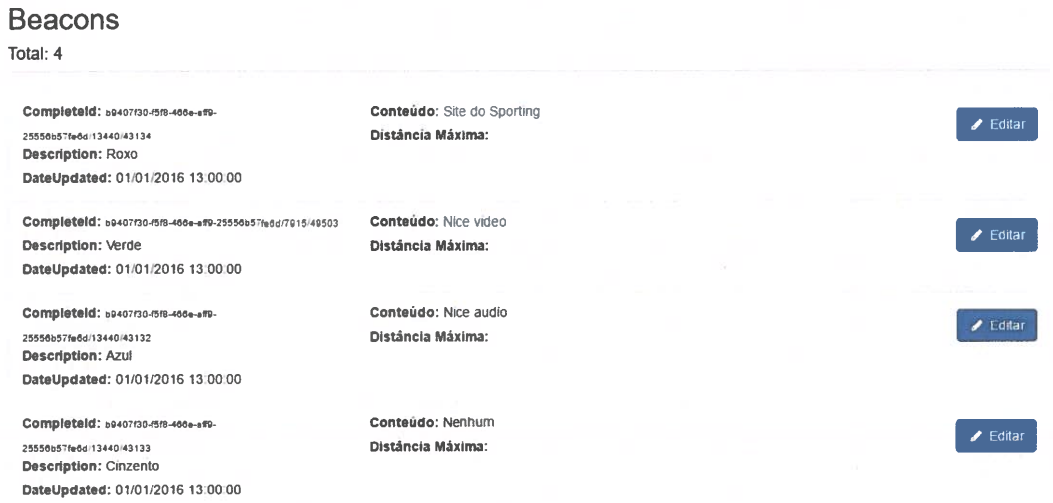


Figura 37 - Listagem de Beacons

Um funcionário de determinado museu poderá ainda consultar quais os beacons que lhe pertencem, bem como alterar a descrição do mesmo para uma melhor identificação, ou até configurar-lhe uma distância máxima.

A página de listagem proporciona ainda uma rápida visualização do conteúdo a que está associado cada *beacon*, bem como se está configurado a alguma distância máxima sobre o mesmo.

O carregamento da listagem é feito de forma assíncrona utilizando para isso não só as keyworks async e await na action correspondente, como também fazendo uso do método ToListAsync da Entity Framework.

4.4.4.3.1.1 Referências

- Controller: ActivationCodesController
- Action: GenerateCode ()
- Javascript na view Index: function generateCode () para realizar um pedido ajax ao controller e action indicados.

4.4.4.3.2 Editação de Beacon

Figura 38 - Página de edição de Beacon

A edição de um beacon por parte de um funcionário do museu apenas permite indicar a descrição e configurar uma distância máxima. Esta distância poderá ser até um máximo de 50 metros, pelo que foi utilizado na validação deste campo o atributo "Range" para definir o intervalo de números aceites, bem como indicado pelo atributo "DisplayFormat" qual o formato de texto a ser utilizado e neste caso um número com 2 casas decimais.

```
[Display(ResourceType = typeof(ModelsLocalization), Name = "MaxDistance")]
[Range(0.0, 50.0)]
[DisplayFormat(DataFormatString = "{0:0.##}", ApplyFormatInEditMode = true)]
0 references | Nuno Mendes, 34 days ago | 1 author, 1 change
public decimal? MaxDistance { get; set; }
```

Figura 39 - Validação do campo MaxDistance

4.4.4.3.2.1 Referências

- Controller: ProximityBeaconsController
- Action:
  - ActionResult Edit(Guid? id) para disponibilização da view.
  - ActionResult Edit([Bind(Include = "…")] proximityBeacon) para beacon o conteúdo na base de dados.
- Model: ProximityBeacon

4.4.4.4 ET04 – Administração de Beacons

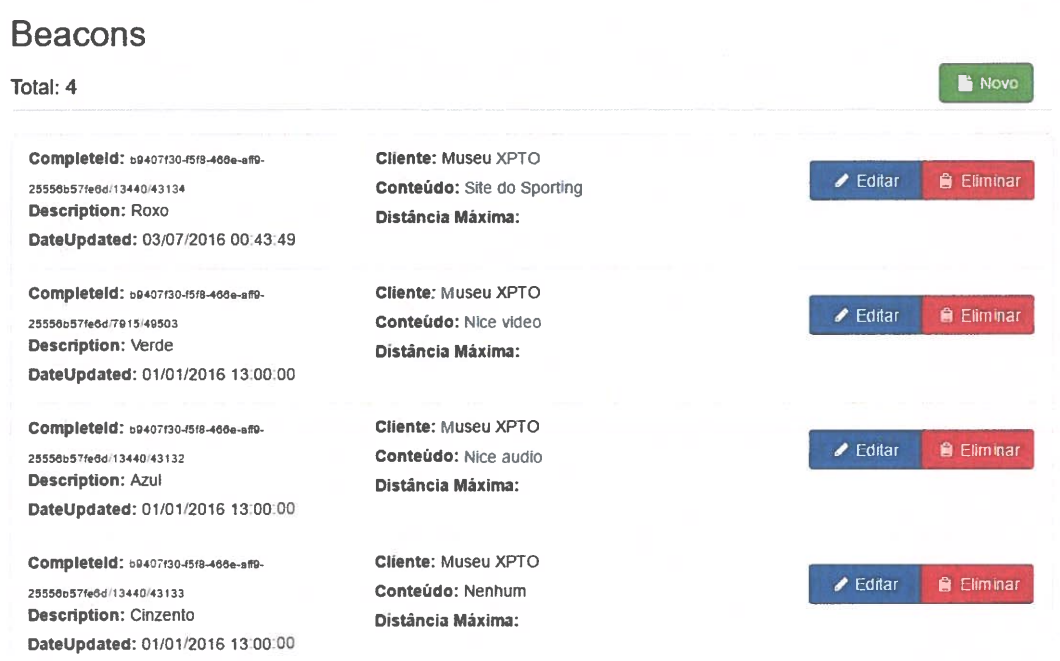


Figura 40 - Administração de beacons

Acedido unicamente por administradores do site, que permite realizar as *CRUD operations* sobre *beacons*. O seu funcionamento é bastante semelhante à gestão de beacons por parte de funcionários de um museu, apenas têm disponíveis mais campos e funcionalidades, tal como nesta listagem em que é facilmente perceptível qual o cliente ou beacon que está em uso.

Ao pressionar o botão "Novo", o utilizador é redirecionado para uma nova página de criação de *beacon*. Nesta página é solicitado não só os campos que descrevem o *beacon* como qual o cliente a que este *beacon* deverá ficar associado.

4.4.4.4.1.1 Referências

- Controller: ProximityBeaconsAdminController
- Action: IndexPartial chamada na view Index
- Model: ProximityBeacon

4.4.4.5 ET05 - Deteções

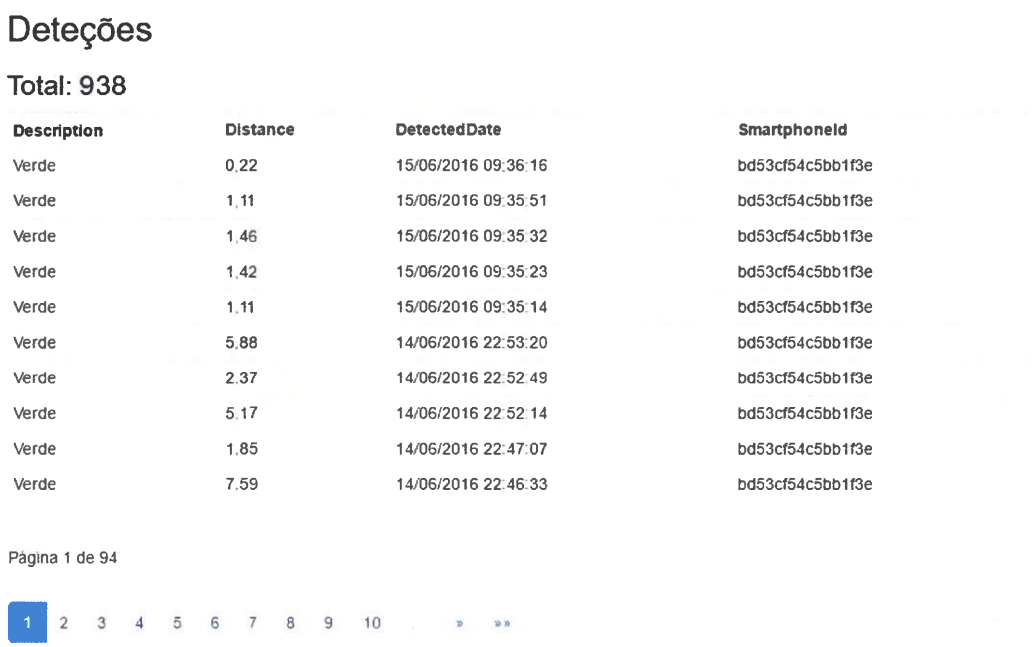


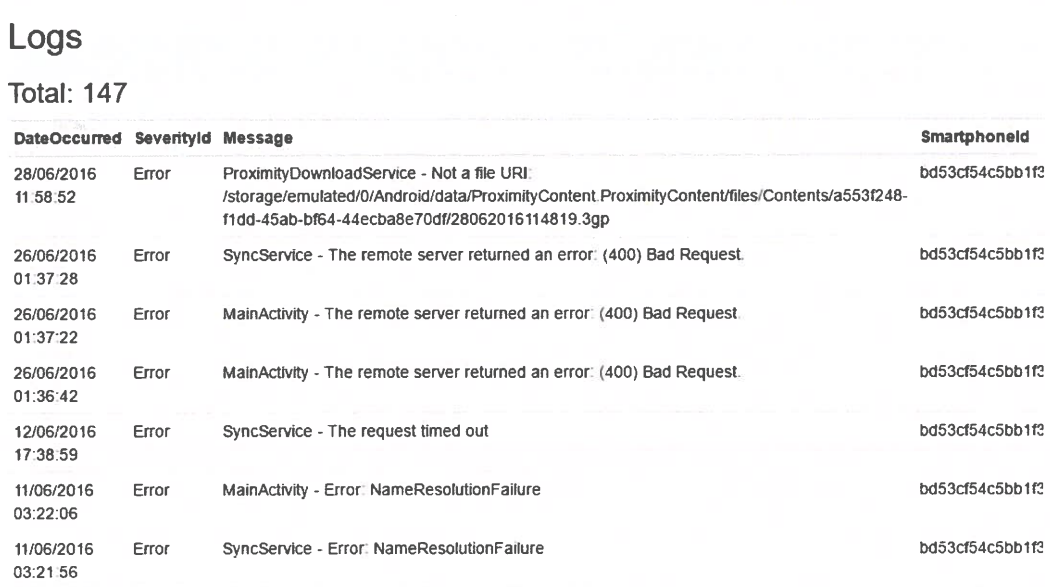
Figura 41 - Página para consulta de deteções efetuadas

Apesar de não ser um requisito expresso do produto, foi incluída a funcionalidade de consulta de deteções de *beacons* efetuadas pelos *smartphones* dos utilizadores finais. Esta funcionalidade poderá revelar-se bastante interessante na medida em que poderá avaliar quais os conteúdos que têm mais interesse para os utilizadores, bastando para isso contabilizar o tempo que estiveram junto do mesmo. De futuro, estes dados poderão ser exibidos sobre a forma de gráficos para uma melhor perceção desta avaliação.

4.4.4.5.1.1 Referências

- Controller: ProximityBeaconDetections
- Action: Index
- Model: ProximityBeaconDetection

4.4.4.6 ET06 - Logs



Logs

Total: 147

DateOccurred	SeverityId	Message	SmartphoneId
28/06/2016 11:58:52	Error	ProximityDownloadService - Not a file URI: /storage/emulated/0/Android/data/ProximityContent.ProximityContent/files/Contents/a553f248-f1dd-45ab-bf64-44ecba8e70df/28062016114819.3gp	bd53cf54c5bb1f3
26/06/2016 01:37:28	Error	SyncService - The remote server returned an error: (400) Bad Request.	bd53cf54c5bb1f3
26/06/2016 01:37:22	Error	MainActivity - The remote server returned an error: (400) Bad Request.	bd53cf54c5bb1f3
26/06/2016 01:36:42	Error	MainActivity - The remote server returned an error: (400) Bad Request.	bd53cf54c5bb1f3
12/06/2016 17:38:59	Error	SyncService - The request timed out	bd53cf54c5bb1f3
11/06/2016 03:22:06	Error	MainActivity - Error: NameResolutionFailure	bd53cf54c5bb1f3
11/06/2016 03:21:56	Error	SyncService - Error: NameResolutionFailure	bd53cf54c5bb1f3

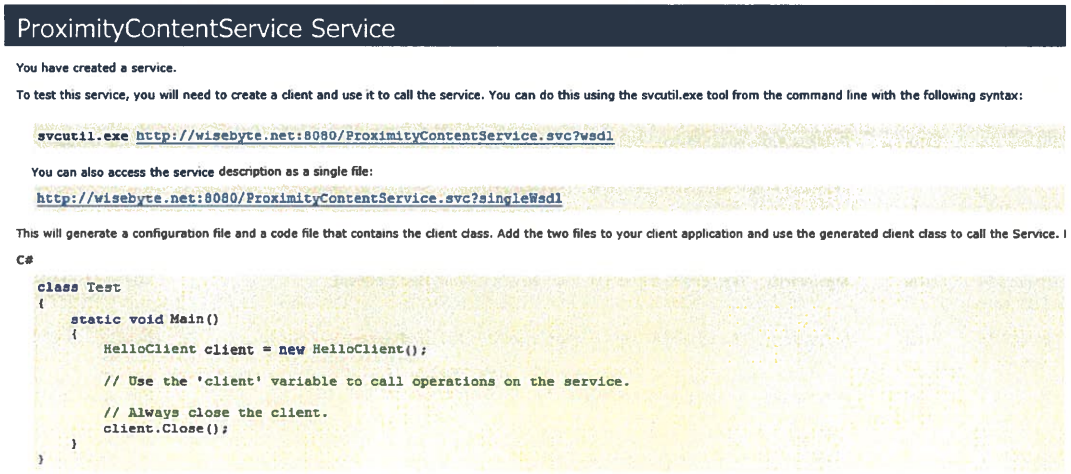
Figura 42 - Página para consula de logs

Outra necessidade identificada foi a de possibilitar a consulta de erros que ocorrem nos smarphones das pessoas. Quando eles ocorrem, são guardados na base de dados SQLite da aplicação móvel e posteriormente, quando ocorre o sincronismo de informação, os mesmos são enviados ao webservice a fim de serem guardados na base de dados central. Deste modo, é possível consultá-los a partir desta página disponível na área de administração do website.

4.4.4.6.1.1 Referências

- Controller: ProximityContentLogs
- Action: Index
- Model: ProximityContentLog

4.5 Webservice – Sincronismo de informação


ProximityContentService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://wisebyte.net:8080/ProximityContentService.svc?wsdl
```

You can also access the service description as a single file:

http://wisebyte.net:8080/ProximityContentService.svc?singleWsdl

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. I

C#

```
class Test
{
    static void Main()
    {
        HelloClient client = new HelloClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Figura 43 - Webservice ProximityContentService.svc

De forma a possibilitar o sincronismo de informação entre o que vai sendo atualizado no website de gestão de conteúdos e a informação que reside na base de dados SQLite das aplicações móveis, entretanto já instaladas nos smartphones pessoais, foi desenvolvido um webservice que providencia métodos para essas operações. Ele foi implementado utilizando a tecnologia WCF – Windows Communication Foundation e sob a forma de RESTful webservice. Para mais informação sobre esta tecnologia, consulte o capítulo Estado da Arte - Web services.

4.5.1 Tecnologias, ferramentas e/ou referências utilizadas

- Visual Studio 2015 – IDE da Microsoft para desenvolvimento de aplicações.
- EntityFramework – API para mapeamento de objetos relacionais que permite programadores .NET trabalhar com informação relacional usando classes .NET. Elimina a necessidade para grande parte do código de acesso a bases de dados.
- AutoMapper – API para mapeamento de objetos, usada neste caso para mapeamento de objetos da entity framework em DTOs

4.5.2 Especificação de Requisitos

ID	DESCRIÇÃO REQUISITO
REQ01	Download da base de dados SQLite para aplicações móveis.
REQ02	Sincronismo de informação a partir de uma determinada data até hoje.
REQ03	Ativar um código de ativação válido.

Tabela 5 - Webservice - Especificação de requisitos

4.5.3 Matriz de rastreabilidade

ID REQ	ID ET	Nome Especificação Técnica
REQ01	ET01	Download da base de dados SQLite
REQ02	ET02	Sincronismo de Informação
REQ03	ET03	Ativação de código

Tabela 6 - Webservice - Matriz de rastreabilidade

4.5.4 Especificação Técnica

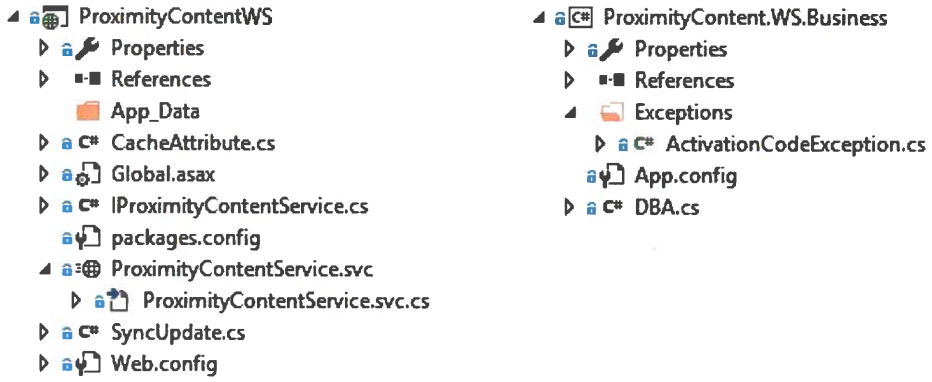


Figura 44 - Projetos ProximityContenWSe ProximityContent.WS.Business

O serviço é exposto através da interface IProximityContentService. A camada de negócio deste webservice foi separada em outra assembly, pelo que é nela que são feitas as chamadas à base de dados SQL central através de Entity Framework e expressões lambda, utilizando a classe DBA para o efeito. A interface expõe os seguintes métodos;

```
[ServiceContract]
1 reference | nunovmendes, 14 minutes ago | 1 author, 11 changes | 1 work items
public interface IProximityContentService
{
    [OperationContract]
    [WebInvoke(Method = "GET",
        BodyStyle = WebMessageBodyStyle.Wrapped,
        UriTemplate = "GetDatabaseFile")]
    [Cache(0)]
    1 reference | nunovmendes, 5 days ago | 1 author, 2 changes
    Stream GetDatabaseFile();

    [OperationContract]
    [WebInvoke(Method = "GET",
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "GetSyncUpdate?updateDate={updateDate}")]
    [Cache(0)]
    1 reference | nunovmendes, 5 days ago | 1 author, 4 changes | 1 work item
    SyncUpdate GetSyncUpdate(DateTime updateDate);

    [OperationContract]
    [WebGet(
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "ActivateCode?clientId={clientId}&code={code}&smartphoneId={smartphoneId}")]
    [Cache(0)]
    1 reference | nunovmendes, 5 days ago | 1 author, 2 changes | 1 work item
    string ActivateCode(string clientId, string code, string smartphoneId);

    [OperationContract]
    [WebInvoke(Method = "POST",
        ResponseFormat = WebMessageFormat.Json,
        RequestFormat = WebMessageFormat.Json,
        UriTemplate = "/InsertDetections")]
    1 reference | nunovmendes, 70 days ago | 1 author, 2 changes | 1 work item
    string InsertDetections(string detections);

    [OperationContract]
    [WebInvoke(Method = "POST",
        ResponseFormat = WebMessageFormat.Json,
        RequestFormat = WebMessageFormat.Json,
        UriTemplate = "/InsertLogs")]
    1 reference | nunovmendes, 12 days ago | 1 author, 1 change
    string InsertLogs(string logs);
```

4.5.4.1 ET01 – Download base de dados SQLite

Quando os utilizadores das aplicações móveis instalam a aplicação, e no primeiro arranque da mesma, é feito um pedido ao webservice para download da base de dados SQLite, de forma a ter sempre uma base de dados atualizada.

Esse pedido é feito através do método GetDatabaseFile, que devolve um stream correspondente aos bytes que compõem o ficheiro da base de dados SQLite. Utilizando "Continuous Integration" do TFS, a cada build da solução é gerado um ficheiro SQLite correspondente à base de dados SQL central, e colocado na pasta da raiz do webservice. Deste modo, quando o pedido de download é feito, é lido este ficheiro e devolvido os bytes do mesmo.

```
1 reference | nunovmendes | 1 author, 1 change
public Stream GetDatabaseFile()
{
    string path = Path.Combine(HttpContext.Current.Server.MapPath("."), "ProximityContent.db");
    byte[] byteArr = File.ReadAllBytes(path);
    return new MemoryStream(byteArr);
}
```

Figura 45 - Método GetDatabaseFile

4.5.4.2 ET02 – Sincronismo de informação

É feito através do método GetSyncUpdate que recebe uma data desde o qual deve ser feito o sincronismo e devolve um objeto SyncUpdate. Esta classe é composta por várias listas de objetos de entidades passíveis de serem sincronizadas.

```
[DataContract]
5 references | Nuno Mendes, 4 days ago | 1 author, 6 changes | 1 work item
public class SyncUpdate
{
    [DataMember]
    2 references | Nuno Mendes, 4 days ago | 1 author, 2 changes
    public List<ReferenceDataDTO> ReferenceDatas { get; set; }
    [DataMember]
    2 references | Nuno Mendes, 4 days ago | 1 author, 3 changes | 1 work item
    public List<ClientDTO> Clients { get; set; }
    [DataMember]
    2 references | Nuno Mendes, 4 days ago | 1 author, 3 changes | 1 work item
    public List<ContentDTO> Contents { get; set; }
    [DataMember]
    2 references | Nuno Mendes, 4 days ago | 1 author, 3 changes | 1 work item
    public List<ProximityBeaconDTO> ProximityBeacons { get; set; }

    1 reference | Nuno Menoes, 69 days ago | 1 author, 5 changes | 1 work item
    public SyncUpdate()
    {
        this.Clients = new List<ClientDTO>();
        this.ReferenceDatas = new List<ReferenceDataDTO>();
        this.Contents = new List<ContentDTO>();
        this.ProximityBeacons = new List<ProximityBeaconDTO>();
    }
}
```

Figura 46 - Classe SyncUpdate

Para cada uma das entidades é chamado o método correspondente na classe DBA no sentido de obter os registos que foram alterados desde a data indicada, exemplo abaixo para a entidade "Content".

```
1 reference | nunovmendes, 28 days ago | 1 author, 3 changes | 1 work item
public List<Content> GetChangedContents(DateTime updateDate)
{
    List<Content> result = new List<Content>();
    db = new ProximityContentContext("ProximityContentContext1");
    var x = db.Contents
            .Where(ct => ct.DateUpdated >= updateDate);
    if (x != null && x.Any())
        result.AddRange(x);
    return result;
}
```

Figura 47 - Método na classe DBA para obtenção dos conteúdos alterados

Posteriormente é feito o mapeamento para DTO utilizando a API AutoMapper de forma a enviar apenas as propriedades do objeto que interessam.

```
var contentsEntities = DBA.Instance.GetChangedContents(updateDate);
if (contentsEntities != null && contentsEntities.Any())
{
    sync.Contents = (from c in contentsEntities
                     select Mapper.Map<ContentDTO>(c)).ToList();
}
```

Figura 48 - Código para mapeamento entre entidades da Entity Framework e DTOs

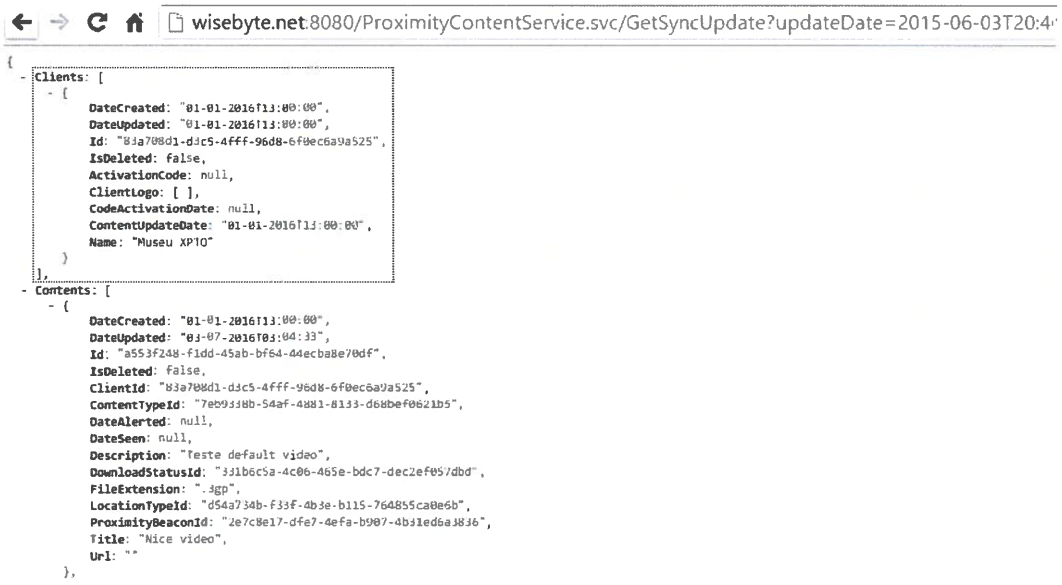


Figura 49 - Exemplo de resposta de um pedido de sincronismo

4.5.4.3 ET03 – Ativação de Código

Quando os utilizadores desejam ter acesso aos conteúdos de determinado museu, os mesmos estão disponíveis mediante código de ativação. Após o solicitarem junto de um funcionário do museu e o inserirem na aplicação, a aplicação irá validar o código através deste webservice.

O método que permite esta validação é o ActivateCode, que recebe o id do cliente, o código de ativação e o id do smartphone. De forma a obter uma maior separação de responsabilidades, a lógica de validação do código encontra-se na camada de negócio e foi criada uma Exception personalizada para esta operação – ActivationCodeException.

```
5 references | Nuno Mendes, 41 days ago | 1 author, 1 change | 1 work item
public class ActivationCodeException : Exception
{
    3 references | Nuno Mendes, 41 days ago | 1 author, 1 change | 1 work item
    public ActivationCodeException(ActivationError error) : base ()
    {
        this.ActivationError = error;
    }
    2 references | Nuno Mendes, 41 days ago | 1 author, 1 change | 1 work item
    public ActivationError ActivationError { get; set; }
}

8 references | Nuno Mendes, 41 days ago | 1 author, 1 change | 1 work item
public enum ActivationError
{
    InvalidCode,
    CodeNotExists,
    CodeInUse,
    CodeUsingTimeLimitExpired
}
```

Figura 50 - Activation Code Exception

Esta classe extende a Exception base da framework de forma a providenciar um e num com o problema que ocorreu durante a ativação do código:

- Código inválido – quando o código é vazio.
- Código Inexistente – quando não encontrado na BD.
- Código em uso – Quando outro smartphone já ativou o código.
- Código expirado – quando ultrapassado o tempo limite de usabilidade.

4.6 Ferramentas auxiliares

No sentido de automatizar o processo de *deploy* para um determinado servidor, através do método *Continuous Integration* do TFS (capitulo Desenvolvimento - Método), foram então desenvolvidas pequenas ferramentas que ajudam nesta automatização.

4.6.1 SQL To SQLite

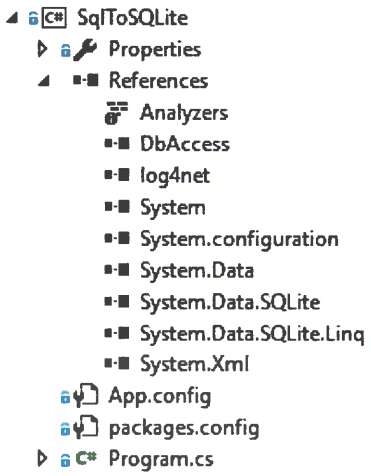


Figura 51 - Projeto SQL to SQLite

Teve por base o seguinte projeto do CodeProject.com em:

http://www.codeproject.com/Articles/26932/Convert-SQL-Server-DB-to-SQLite-DB

Esse projeto era composto por uma interface gráfica onde era possível especificar qual a base de dados SQL de onde se pretende realizar a conversão e qual o ficheiro SQLite de destino. De modo a esta conversão ser incluída no processo de automatização, foi removido a interface gráfica e incluída a lógica de configuração da base de dados a converter numa *Console Application*. Foi também alterada a lógica de parâmetros de configuração para suportar a especificação de que tabelas serão exportadas a estrutura, e quais as tabelas em que se pretende também exportar os registos inseridos.

4.6.2 Sync Storage

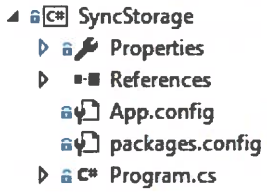


Figura 52 - Projeto SyncStorage

Outra *Console Application* para que, a cada *build* da solução no processo de *Continuos Integration,* seja realizado a eliminação de ficheiros multimédia dos clientes que já não estejam e uso, por terem sido eliminados ou alterados.

Durante sua execução, é percorrido toda a estrutura de pastas a partir de uma pasta base, por forma a comparar se o que existe em termos de ficheiros relativos a conteúdos é consistente com a informação residente na base de dados central, caso não seja, o ficheiro é eliminado.

5 Conclusão

Apresenta-se neste capítulo as conclusões do trabalho desenvolvido bem como o sucesso no cumprimento dos objetivos do projeto. São também descritas as vantagens do sistema desenvolvido, quer para os utilizadores finais, quer para as organizações que integrem esta solução.

Com base nos conceitos que foram abordados neste documento e os resultados obtidos, é permitido chegar a uma conclusão sobre a integração de sistemas móveis no sistema dos audioguias. Com o crescimento e evolução exponencial das capacidades dos dispositivos móveis e da sua diversidade, torna-se cada vez mais propícia a integração destes nas diversas áreas de atividade, devido à sua simplicidade, funcionalidades e portabilidade, mas em especial à mobilidade que confere aos seus utilizadores.

Os sistemas de audioguias tradicionais envolvem custos elevados para as organizações, não só na sua aquisição, bem como na sua constante manutenção e atualização. Conclui-se que a melhor solução para o sistema tradicional de audioguias não passaria pela sua otimização, mas sim por uma reformulação de todo o sistema. A reformulação do sistema consistiu na realização de uma análise às funcionalidades que os utilizadores necessitariam, de uma nova arquitetura, sendo introduzida a tecnologia Bluetooth Low Energy e os Beacons. Esta análise contribuiu de forma muito significativa para o aumento da real qualidade do produto. Da mesma forma, o desenho da solução mostrou-se o mais adequado para este tipo de sistemas.

Relativamente ao cumprimento das expectativas traçadas para o sistema desenvolvido, é possível aferir, de forma concreta, acerca da sua qualidade perante a nova solução. Durante o desenvolvimento do sistema, vários utilizadores tiveram a oportunidade de realizar testes à aplicação móvel, mostrando a sua satisfação pelas melhorias acentuadas comparativamente ao sistema tradicional. Em suma, pela forma como decorreu o desenvolvimento deste projeto e pela qualidade do resultado final, poderá concluir-se que este projeto trouxe um enriquecimento do sistema de audioguia e introduziu o conceito de mobilidade, através dos sistemas móveis.

Bibliografia

Aldea, C., Sangeorzan, L., & Aldea, A. (2009). Web Services and Enterprise Games. In *Proceedings of the 9th WSEAS International Conference on Simulation, Modeling, and Optimization* (pp. 298–303). Retrieved from http://www.wseas.us/e-library/conferences/2009/budapest/SMO/SMO53.pdf

Bi, C. (2009). Research and application of SQLite embedded database technology. *WSEAS Transactions on Computers*, *8*(1), 83–92. Retrieved from http://www.wseas.us/e-library/transactions/computers/2009/31-846.pdf

Cată, M. (2015). APPLYING QR CODES , NFC TAGS OR BLE BEACONS ON OBJECTS FROM A UNIVERSITY, *XVIII*(1), 6–7. Retrieved from https://www.anmb.ro/buletinstiintific/buletine/2015_Issue1/FCS/306-307.pdf

Cavanaugh, E. (2006). Web services: Benefits, challenges, and a unique, visual development solution. *Product Marketing Manager, Altova® WhitePaper*. Retrieved from http://www.altova.com/documents/whitepaper_webservices_2006.pdf

Claesson, A., & Dubray, C. (2009). *Development of support web applications in . NET*. University of Gothenburg.

Greene, J. (2016). Microsoft Agrees to Acquire Xamarin. *The Wall Street Journal*. Retrieved from http://www.wsj.com/articles/microsoft-agrees-to-acquire-xamarin-inc-1456340494

Hendrix, P. (immr). (2015). Watch this space. *Gimbal*, 4. Retrieved from https://www.gimbal.com/wp-content/uploads/immr-Mobile-Beacons-and-OOH-Whitepaper_Gimbal.pdf

Holla, S., & Katti, M. M. (2012). Android Based Mobile Application Development and its Security. *International Journal of Computer Trends and Technology*, *3*(3), 486–490.

Jose, D. V, Lakshmi, P. C., Priyadarshini, G., & Singh, M. (2015). International Journal of Advanced Research in Computer Science and Software Engineering. *International Journal*, *5*(1), 811 – 814. Retrieved from http://www.ijarcsse.com/docs/papers/Volume_5/1_January2015/V5I1-0464.pdf

Mulligan, G., & Gracanin, D. (2009). A COMPARISON OF SOAP AND REST IMPLEMENTATIONS OF A SERVICE BASED INTERACTION INDEPENDENCE MIDDLEWARE FRAMEWORK. In Intergovernmental Panel on Climate Change (Ed.), *Proceedings of the 2009 Winter Simulation Conference* (pp. 1423–1432). Cambridge: Cambridge University Press.

Mumbaikar, S., & Padiya, P. (2013). Web Services Based On SOAP and REST Principles. *International Journal of Scientific and Research ...*, *3*(5), 1–4.

Pocatilu, P. (2012). Building Database-Powered Mobile Applications, *16*(1), 132–142. Retrieved from http://revistaie.ase.ro/content/61/12 - Pocatilu.pdf

Puertolas-Montanes, J. A., Mendoza-Rodriguez, A., & Sanz-Prieto, I. (2013). Smart Indoor Positioning/Location and Navigation: A Lightweight Approach. *International Journal of Interactive Multimedia and Artificial Intelligence*, *2*(2), 43–50. http://doi.org/10.9781/ijimai.2013.225

Sterling, G. (2014). Will In-Store Bluetooth Beacons Marginalize QR Codes? Retrieved from

http://marketingland.com/will-store-bluetooth-beacons-marginalize-qr-codes-77455

Xamarin. (2016). Retrieved from https://www.xamarin.com/