INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

Soluções de broadcasting em dispositivos móveis

Tiago José dos Santos Serra

INSTITUTO POLITÉCNICO DE VIANA DO CASTELO
Escola Superior de Tecnologia e Gestão

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

Tiago José dos Santos Serra

Soluções de broadcasting em dispositivos móveis

Mestrado em Engenharia de Software

Orientador
Pedro Castro

Coorientador
Jorge Ribeiro

Setembro de 2015

Dedicatória

A redação desta tese só foi possível com o esforço que os meus pais tiveram para me dar a melhor formação possível, a eles o meu profundo agradecimento.

Á minha irmã, pela força que me deu e tem dado para enfrentar os mais diversos problemas.

Aos meus colegas que me acompanharam nesta “jornada”, especialmente ao José Rui Peixoto, Jorge Cunha e Rui Peleja - os momentos que passamos jamais esquecerei.

A todos eles o meu eterno agradecimento.

Agradecimentos

Quero começar por agradecer aos meus orientadores, o professor Pedro Castro e o professor Jorge Ribeiro, pela oportunidade de explorar um tema sobre o qual não tinha muitos conhecimentos.

Também á minha família que me apoia e tem apoiado nas minhas diversas fases da vida.

E a todos aqueles que contribuíram diretamente e indiretamente para a minha felicidade.

A todos eles, o meu muito obrigado.

Resumo

Ao longo dos últimos anos, é notória a evolução das tecnologias e sistemas de informação refletindo-se na utilização massivamente pela comunidade de aplicações e serviços, em particular através da disponibilização de dispositivos móveis com elevados recursos computacionais e custos. Cada vez mais as empresas e organizações oferecem aplicações e serviços para promoverem produtos e serviços no sentido de promoverem e rentabilizarem os seus negócios em grande escala.

Por sua vez, tipicamente as empresas de marketing e de promoção de produtos promovem as suas ações de comunicação através de eventos realizados, *online* ou fisicamente, de modo a chegar a potenciais consumidores e tentar atingir um segmento de mercado.

Com este intuito e durante este mestrado surgiu a ideia de desenvolver uma aplicação para *smartphones* onde as pessoas pudessem trocar ideias. Estas estariam associadas a um evento. Visto isto, analisamos os diferentes tipos de mensagens (*pull* e *push*) e como estas comunicam na web e nos *smartphones*. Assim sendo, o objetivo principal deste estudo é o de encontrar a melhor solução de *broadcast* para os diferentes sistemas operativos móveis existentes, neste caso o Android e o iOS. Neste contexto foi efetuada uma análise da arquitetura dos serviços atualmente existentes, e aprofundar os conhecimentos sobre os novos conceitos em que estes assentam, como por exemplo, ***push notification***. Revisto todos os novos conceitos, tendo pesado os prós e contras que cada serviço (Direct 100, Zero push e o Parse) oferecia, selecionamos aqueles que se adaptam melhor á ideia apresentada. Neste caso os serviços escolhidos foram o ***G****oogle* ***C****loud* ***M****essaging* (GCM) e o ***A****pple* ***P****ush* ***N****otification* ***S****ervice* (APNS). Após o estudo daquelas ferramentas, foram realizados testes de desempenho com dados fictícios e com dados reais em eventos por forma a comprovar que as ferramentas escolhidas são as que melhor servem o nosso propósito.

Palavras-chave: Cloud Computing, Pull e Push, Google Cloud Mensaging, Apple Push Notification Service.

Abstract

Over the past few years, it is notorious the evolution of technology and information system that reflects the massive use by the community of applications and services, in particular through the provision of mobile devices with high computing resources and costs. More and more companies and organizations offer applications and services to promote products and services in order to promote and monetize their business on a large scale. In turn, typically marketing and product promotion companies promote their communication actions through events held, online or physically, in order to reach potential customers and try to reach a market segment.

Bearing this in mind, the idea of developing a smartphone application which people could use to exchange ideas associated to an event seemed extremely suitable for this master degree. Therefore, we analyzed different types of messages (pull and push) and the way these communicate on the web and with smartphones.

Thus, the main goal of this study is to find the best broadcast solution for the different mobile systems, in this case, for the Android and iOS. Consequently, it was necessary to analyze each service regarding its architecture and deepen the knowledge of the new concepts on which these are based, such as push notification.

After having seen all concepts available, and having assessed the strong points and weaknesses from these services (Direct100, Zero push and Parse), we selected the ones that suit best the idea. In this case, the chosen services were Google Cloud Messaging (GCM) and Apple Push Notification Service (APNS). After studying these tools, performance tests were done with real and unreal data at events to ensure that the chosen tools are the ones which best serve our purpose.

Keywords: Cloud Computing, Pull e Push, Google Cloud Mensaging, Apple Push Notification Service.

Sumário

LISTA DE FIGURAS ..... **XV**

ÍNDICE DE TABELAS ..... **XVII**

LISTA DE ABREVIATURAS E SIGLAS ..... **XVIII**

CAPÍTULO 1 – INTRODUÇÃO **1**

1.1 – ENQUADRAMENTO E MOTIVAÇÃO 1

1.2 - OBJETIVO 2

1.3 - METODOLOGIA 3

1.4 - ESTRUTURA DO DOCUMENTO 4

CAPÍTULO 2 – TECNOLOGIAS DE BROADCAST EM DISPOSITIVOS MÓVEIS **5**

2.1 - CONTEXTUALIZAÇÃO 5

2.2 – DESENVOLVIMENTO WEB PARA DISPOSITIVOS MÓVEIS 7

2.2.1 – Estado de Arte - Comunicação dos dispositivos móveis na web *7*

2.2.2 – Estratégias Pull *9*

2.2.3 – Estratégias Push *10*

2.2.4 – A importância da tecnologia push *11*

2.3 – TECNOLOGIAS DE BROADCASTING 13

2.3.1 – Análise de Soluções *13*

2.3.2 – Soluções Selecionadas *16*

2.3.2.1 – Cloud Computing 16

2.3.2.1.1- História da Cloud Computing 16

2.3.2.1.2- Definição 17

2.3.2.1.3- Exemplos de aplicações em cloud computing 20

2.3.2.2 – Google Cloud Messaging (GCM) 22

2.3.2.2.1 – Descrição 22

2.3.2.2.2 – Arquitetura GCM 23

2.3.2.2.3 – Obter acesso ao GCM 26

2.3.2.3 – Apple Push Notification Service (APNS) 28

2.3.2.3.1 – Descrição 28

2.3.2.3.2 – Arquitetura de Segurança 29

2.3.2.3.3 – Implementação do APNS 33

Gerar o Certificate Signing Request (CSR) 33

Criar um App ID e Certificado SSL 34

Gerar o ficheiro Privacy Enhanced Mail 36

Criar o Provisioning Profile 37

CAPÍTULO 3 – CASO DE ESTUDO - IMPLEMENTAÇÃO DAS FRAMEWORKS NO NEARUS..... 39

3.1 – CONTEXTUALIZAÇÃO DA APLICAÇÃO NEARUS..... 39

3.2 – ARQUITETURA..... 41

3.3 – IMPLEMENTAÇÃO DO NEARUS..... 43

3.4 – DESENVOLVIMENTO DO NEARUS..... 46

3.5 – TESTES REALIZADOS NO NEARUS..... 49

3.5.1 – Testes de performance..... 50

3.5.2 – Testes realizados no NearUs em contexto real..... 54

3.6 – ANÁLISE CRITICA E TRABALHO FUTURO..... 55

CAPÍTULO 4 – CONCLUSÃO E TRABALHO FUTURO..... 57

CAPÍTULO 5 – BIBLIOGRAFIA E REFERÊNCIAS WWW..... 59

Lista de Figuras

Figura 1 – Ilustração de sistema pull [3] 10
Figura 2 - Ilustração de sistema push [3] 11
Figura 3 - Evolução do homem (imagem retirada do portal athenahealth.com)..... 17
Figura 4 - Ilustração de cloud computing (imagem retirada do portal javatpoint.com) .. 20
Figura 5 - Arquitetura do Google Cloud Messaging [7] 24
Figura 6 - Ilustração de uma mensagem recebida pelo GCM (imagem retirada do portal stackoverflow.com/questions/8473918/how-to-set-a-timeout-expiration-on-a-c2dm-message). 26
Figura 7 - Ecrã com o Projeto Criado 27
Figura 8 - Fluxo da mensagem push da apple [8] 29
Figura 9 - Service-to-Device Connection Trust [8] 30
Figura 10 - Provider-to-Service Connection Trust [8] 31
Figura 11 - Token Trust [8] 32
Figura 12 - Certificates, Identifiers & Profiles 34
Figura 13 - Identifiers 35
Figura 14 - Create Certificate 35
Figura 15 - Provisioning Profile 37
Figura 16 - Add iOS Provisioning Profile 38
Figura 17 - Cálculo de distâncias em caso de intersecção das circunferências 40
Figura 18 - Cálculo de distâncias em caso de não intersecção circunferências 40
Figura 19 - Primeira Arquitetura do NearUS 41
Figura 20 - Arquitetura Atual do NearUs 42
Figura 21 - Modelo Caso de Uso 46
Figura 22 - Teste de performance utilizando o MySql 51
Figura 23 - Ilustração da organização dos dados em memória 52
Figura 24 - Teste de performance utilizando a memória 53
Figura 25 - Lista de downloads por país, relativamente ao Android. 55

Índice de Tabelas

Tabela 1 - Síntese dos serviços encontrados 15
Tabela 2 - Caso de Uso: Enviar Mensagem 44
Tabela 3 - Caso de Uso: Procurar tokens. 45
Tabela 4 - Caso de Uso: Receber Mensagem. 45
Tabela 5 - Tabela com a função em PHP para saber a distância entre 2 pontos 49
Tabela 6 - Esquema do teste de performance 50

Lista de Abreviaturas e Siglas

Abreviatura	Descrição
GCM	Google Cloud Mensaging
APNS	Apple Push Notification Service
SaaS	Software as a Service
S3	Silpmes Storage Solution
EC2	Elastic Compute Cloud
ARPA	Advanced Research Projects Agency
C2DM	Cloud to Device Messaging Framework
TCP	Transmission Control Protocol
TLS	Transport Layer Security
SSL	Segurança da Camada de Transporte
PHP	PHP: Hypertext Preprocessor
PEM	Privacy Enhanced Mail
CSR	Certificate Signing Request
SDK	Software Development Kit
IDE	Integrated Development Environment
SQL	Structured Query Language
CRUD	Create, Read, Update e Delete
REST	Representational State Transfer
JSON	JavaScript Object Notation
PECL	PHP Extension Community Library

Capítulo 1 – Introdução

1.1 – Enquadramento e Motivação

Nos últimos anos cada vez mais os *smartphones* e os *tablets* fazem parte do nosso dia-a-dia. Com efeito é de salientar que, há uns anos atrás estes aparelhos continham pouca ou quase nenhuma relevância na nossa vida, situação que pode ser facilmente comprovada pelo facto que o primeiro anúncio de uma operadora de telemóveis em Portugal apareceu na televisão por volta do ano de 1991/1992. Por esta altura todos nós poderíamos viver e conviver sem os telemóveis e se alguém afirmasse que passado 20 anos iriamos ficar dependentes destas tecnologias, acharíamos que seria um cenário irreal.

Contudo, com o passar dos anos e com a rápida evolução da tecnologia os telemóveis tornaram-se uma extensão do nosso ser, para além da visão, audição, paladar, olfato e tato. Lembremos que no ano 2000 era lançado pela Nokia, o telemóvel modelo 3310, na altura um telemóvel topo de gama e muito comercializado. Volvidos quinze anos este modelo é considerado ultrapassado. Porém, esta rápida evolução trouxe as suas desvantagens, sendo uma delas a "não-socialização" com os que nos rodeiam, pois passamos muito do nosso tempo "agarrados" a estes dispositivos. Quantos de nós já não iniciou uma conversa com uma pessoa completamente desconhecida enquanto esperava numa fila para a aquisição de bilhetes para um jogo de futebol ou de um espetáculo de música, questionando-a sobre o estado atual da equipa, ou sobre as suas músicas favoritas em relação ao cantor em causa.

Atualmente, durante essa espera grande parte das pessoas começam a ter tendência de utilizar o seu dispositivo para jogar, verificar o *e-mail*, trocar mensagens, etc. Aliás, torna-se absolutamente caricato verificar que, mesmo durante as refeições as pessoas não falam entre si e não convivem com aqueles que consigo estão sentados à mesa, preferindo a companhia destes aparelhos.

Tendo isto em conta, e durante uma troca de opiniões com um colega que frequentava o mesmo mestrado surgiu a ideia de criar e desenvolver uma aplicação que permitisse às pessoas interagirem umas com as outras, utilizando para o efeito o seu dispositivo. Assim podiam "socializar" umas com as outras no contexto de um evento, como por exemplo, um jogo de futebol, um concerto de música, etc.

Com esta aplicação as pessoas poderiam utilizar na mesma o seu *smartphone* e em simultâneo trocar ideias e imagens sobre um acontecimento que tivesse a acontecer num dado espaço e tempo. Ou seja, não necessitariam de abdicar do seu *smartphone* para poder socializar com o meio envolvente. Com efeito, e com esta aplicação conseguir-se-ia dar uma excelente utilização a estes aparelhos evitando assim o isolamento social a que por vezes a sua utilização conduz.

A essa ideia demos o nome de NearUs.

1.2 - Objetivo

No seguimento do enquadramento e motivação descrito na seção anterior, o objetivo principal deste trabalho de dissertação centra-se no desenvolvimento de uma aplicação informática para dispositivos móveis com o intuito de permitir a partilha de imagens e de mensagens entre várias pessoas num mesmo espaço físico, utilizando para o efeito os dispositivos móveis.

Porém para a execução desta aplicação informática, surgem várias particularidades na sua implementação, como por exemplo questões associadas ao armazenamento e difusão de mensagens, sendo necessário proceder à análise e estudo de técnicas, métodos e tecnologias para efetuar a partilha, comunicação e difusão de mensagens entre um repositório de informação (tipicamente armazenado num servidor

web) e os dispositivos móveis. Por outro lado, estas comunicações, entre o servidor web e o dispositivo móvel requer a utilização de tecnologias para a troca de informação com a particularidade de ser necessário disseminar uma mensagem ou imagem pelo conjunto dos dispositivos dos utilizadores com condições de receberem mensagens num grupo predefinido de dispositivos/utilizadores. Esta característica leva á necessidade de se implementar uma solução de *broadcasting* para dispositivos móveis.

Assim, o objetivo principal desta dissertação de Mestrado centra-se no estudo de soluções *broadcasting* para dispositivos móveis materializando-se através da implementação das tecnologias mais adequadas no desenvolvimento de uma aplicação informática orientada para o envio em massa de mensagens e de imagens entre dispositivos móveis.

1.3 - Metodologia

Neste trabalho o método experimental utilizado orientou-se pelos passos metodológicos que se iniciaram, pela identificação do problema e a motivação, no sentido de formalizar um objetivo passando-se à recompilação e organização da informação, construindo-se uma proposta para resolver o problema identificado. Finalmente, foram elaboradas conclusões que refletiram os resultados obtidos na avaliação da operacionalização da aplicação desenvolvida. Neste sentido este trabalho foi desenvolvido em três fases:

1. Revisão da literatura existente sobre as técnicas, metodologias e tecnologias utilizadas para o tratamento da difusão de mensagens.

2. Depois do levantamento do estado de arte, o passo seguinte foi o de identificar a melhor solução para efetuar o *broadcast* usando dispositivos móveis, optando-se por aquele que mais e melhor se adapta e se adequa dentro da ideia do caso de estudo, tendo em conta critérios como custo e desempenho.

3. Avaliação – No sentido de avaliar o desempenho e eficiência das tecnologias escolhidas para o caso de estudo, foram analisados testes com dados fictícios e reais que

permitem monitorizar de forma quantitativa os resultados da aplicabilidade dos referenciais em estudo.

1.4 - Estrutura do documento

Este documento está estruturado em cinco capítulos, iniciando-se introdução ao tema, o seu enquadramento e motivação, o objetivo e a metodologia seguida no trabalho desenvolvido. No Capítulo 2 fazemos uma descrição da ideia NearUs, enquadramos com o desenvolvimento web, isto é, como se processa a troca de informações entre o telemóvel e a web. Para além disso, introduzimos e descrevemos as estratégias *pull* e *push*, explicitando melhor porque esta última é importante no funcionamento da aplicação. Também elaboramos uma contextualização do levantamento dos serviços encontrados que permitem o *broadcast* em dispositivos móveis. Ainda no mesmo capítulo descrevemos o conceito do *cloud computing* (pois a arquitetura das soluções encontradas assentam sobre este conceito); um pouco da sua história e damos exemplos de aplicações de assentam no *cloud computing*. No final do levantamento dos serviços, escolhemos o GCM (*Google Cloud Mensaging*) e o APNS (*Apple Push Notification Service*) como a melhor solução encontrada para o problema, por isso analisamos a arquitetura de cada uma das soluções, organizadas em subcapítulos dentro do capítulo 2.

No Capítulo 3 explicamos os conceitos em que a aplicação NearUs assenta, assim como a sua arquitetura e como esta se interliga com os serviços escolhidos. Ainda no mesmo capítulo utilizamos os casos de uso para descrever o funcionamento do *broadcast* no NearUs. Descrevemos, também, as bibliotecas e as linguagens de programação em que o NearUs foi desenvolvido. Apresentamos os testes realizados para provar a eficácia da aplicação e o trabalho a realizar no futuro no NearUs.

No final deste trabalho elaboramos uma conclusão, onde refletimos e comentamos o que aprendemos com este trabalho, com o repto lançado e o contributo do mesmo para uma aprendizagem que se quer permanente, especialmente o *broadcast* nos dispositivos móveis.

Capítulo 2 – Tecnologias de BroadCast em Dispositivos Móveis

2.1 - Contextualização

Antes de explicar o funcionamento da comunicação dos dispositivos móveis com a web convém referir como surgiu a ideia NearUs. Assim, durante o período letivo em que decorreu a cadeira de Laboratório de Projeto 2, contida no Mestrado de Engenharia de Software (cujo objetivo centra-se na aplicação dos conhecimentos adquiridos nas unidades curriculares do mestrado no desenvolvimento de um projeto de *software*) eu e o colega da referida disciplina tínhamos que escolher um tema sugerido pelos docentes da cadeira, ou então propor um tema que pudesse ser desenvolvido.

Perante isto, achámos que seria importante desenvolver uma ideia só nossa, que posteriormente designámos por NearUs.

O NearUs é descrito como: “NearUs is a mobile communications system enabling messaging, chatting and image sharing. The system is to be applied in locations with spatial and temporal boundaries defined by an induced network of non-registered users.

NearUs is a location and temporal based service inspired and enhancing the traditional “walkie talkie” experience. As in the “walkie talkie” world, users do not face registration barriers but can be identified through their phone number. In addition, they can be spatially and temporally located.

By activating Wi Fi, cellular and/or GPS connections, an estimate of the user’s location is defined. The location is represented by a dot in the screen. The dot may be seen as a node in an imaginary network. By selecting a radius, each user selects his or her “communication space”. By activating this space, he or she discovers the other active users represented by nodes.

Each space is associated to an event. Events can be limited in time (a rock concert, a sports event) or unlimited (chatting at a circumscribed space such as a university or shopping mall) There will always exist a default communication space defined by the spatial boundaries associated with the event.”.[1]

Por outras palavras, colocámo-nos na ótica de um utilizar de dispositivos móveis, que, com a nossa aplicação pudesse fácil e rapidamente falar com pessoas que estivessem próximas de si.

Pensemos num utilizador no contexto de um espetáculo de música, ou num jogo de futebol, onde muitas vezes se discutem aspetos próprios do caracter dos eventos com a pessoa que está ao lado. Contudo, como sabemos, mais do que isto não é possível, pois embora estejamos todos no mesmo espaço é fisicamente impossível fazê-lo com outras pessoas para além das que estão ao nosso lado. E eis que surge a ideia central do NearUs: e porque não falar com todas as pessoas presentes no evento?

Tornar esta ideia possível seria ampliar a possibilidade de troca de opiniões e comentar o evento que estamos a assistir com diversas pessoas.

Para além desta funcionalidade que acabamos de descrever, o NearUs pode revelar-se útil em casos extremos como num cenário de resgate e salvamento, proteção civil, etc. E ainda em outras áreas tais como Marketing e Negócios.

2.2 – Desenvolvimento Web para dispositivos móveis

2.2.1 – Estado de Arte - Comunicação dos dispositivos móveis na web

Uma aplicação instalada nos nossos *smartphones* comunica com um servidor (um computador que fornece serviços, onde estão guardados dados que podem ser consultados a partir de vários dispositivos), através de mensagens, utilizando para o efeito uma ligação com a internet.

Perante a descrição da aplicação, surgiu um problema: "Como iria o utilizador receber as mensagens no seu dispositivo?".

Estas mensagens podem ocorrer com base em duas formas de comunicação: o telemóvel envia uma mensagem ao servidor a informar que necessita de um ou vários dados (*pull*) ou o servidor comunica com o telemóvel informando da existência de um dado novo (*push*).

Mas qual destes dois será o ideal para o desenvolvimento correto da aplicação?

Para dar resposta a esta questão e melhor entendermos a diferença entre estes dois sistemas, socorremo-nos da história e dos vários sistemas que existem em outras áreas que não apenas no mundo das telecomunicações, para daí tirar ensinamentos e conceitos que não são totalmente "novos", conforme descrito em [12].

Se reportarmo-nos para a época anterior à revolução industrial, mais concretamente nas antigas fábricas de materiais, qual seria o método de produção utilizado para a criação dos novos materiais? Nesta altura tudo, ou quase tudo era fabricado por encomenda e em pequenas quantidades, indo de encontra aos pedidos dos clientes. Os pedidos de roupa, artesanato, novos equipamentos, etc eram produzidos conforme a encomenda. Fazendo um paralelismo com a realidade atual, os alfaiates fazem a roupa por medida dos clientes, sendo isto um exemplo do sistema pull. Contudo o fabrico por encomenda tem 2 desvantagens: o elevado custo de produção e o desenvolvimento de poucas unidades.

Ao escrever o segundo capítulo do Tratado sobre a origem da Riqueza das Nações, em 1777, Adam Smith mudou um pouco este sistema, ao introduzir, numa fábrica de alfinetes, uma nova caraterística nos processos de produção destes, que permitiu aumentar de poucas unidades para milhares de alfinetes produzidos por dia. Para o conseguir introduziu o conceito da divisão das tarefas. Adam Smith notou que nessa fábrica um único profissional realizava todas as tarefas de produção do alfinete, por isso sugeriu que fossem criadas divisões de tarefas e assim cada pessoa realizava uma única atividade especializada. [12]

Wislon Frederich Taylor [2] desenvolveu o princípio de "Administração Cientifica", em que se proponha gerir uma fábrica como se fosse uma ciência exata, ou seja, estudar os tempos e os movimentos necessários para concluir uma atividade e realizá-la no menor tempo e movimentos possíveis, em cada elo do processo. Tudo isto levou a um aumento inimaginável da produtividade das empresas a que veio a chamar-se de "Revolução Industrial".

Depois disto, era preciso criar uma linha de produção, onde os vários processos e divisões tivessem em pleno funcionamento e completamente dominados pelo Homem. Esta revolução aconteceu tanto no meio industrial como na prestação de serviços. A revolução industrial veio assim revolucionar todos os processos de produção de materiais.

Mediante isto, constatamos que o paradigma estava a mudar para um sistema de *push*, onde os principais objetivos era produzir barato e em grandes proporções para atingir o máximo de população possível. Antes disto, como já vimos, produzir qualquer coisa seria extremamente oneroso. Contudo, com a revolução industrial foi possível mudar esta paradigma e produzir muito e barato, para que toda a população pudesse usufruir dos vários serviços e produtos. Foi na sequência desta filosofia que no século XX, Henry Ford desenvolveu um processo de produção de baixo custo de um automóvel, neste caso um Ford, para que todos os americanos e funcionários dele pudessem ter um veículo.

Mas neste século ocorreram muitas alterações e a pouco a pouco as carências de cada consumidor foram alteradas. Exemplo disso foi o aumento do pedido de atualizações de um item, produto ou serviço quando ainda contava com pouco tempo de vida (grandes exemplos disso são os automóveis, telemóveis, etc), ao invés do modelo antigo que demorava muito tempo para estes serem atualizados. [2]

Com este exemplo já podemos verificar a diferença entre estes dois sistemas. O mais utilizado antes da revolução industrial era o método de pull, e o mais utilizado após esta revolução era o sistema *push*.

Dando como exemplo a compra de um par de calças, no caso do sistema pull teríamos que ir ao alfaiate para obtê-las, e no caso do sistema *push*, teríamos que ir a uma loja para compra-las.

Nos subcapítulos seguintes descrevem-se cada uma destas formas de comunicação e explicita-se a arquitetura destas.

2.2.2 – Estratégias Pull

Transcrevendo o que foi referido no subcapítulo anterior para o mundo da tecnologia, mais especificamente para o mundo da informática, podemos descrever mensagens pull como sendo aquelas que são requeridas pela aplicação, ou seja pelo cliente. Isto acontece naquelas situações em que necessitamos que a aplicação, ou o cliente, faça poucos pedidos ao servidor, para que este não fique sobrecarregado, permitindo assim menos custos no lado do servidor. [18]

Esta técnica é muito utilizada por grande parte das aplicações, porém para o nosso problema não seria a estratégia mais aconselhada pois sempre que existe-se uma mensagem nova precisamos que o utilizador tome conhecimento desta, instantaneamente. Assim sendo, se adotássemos esta técnica a aplicação teria que fazer requisições de x em x segundos e assim não garantiríamos dois pontos fundamentais para

o bom funcionamento da aplicação: receber mensagens instantaneamente e a utilização elevada dos recursos por parte do servidor. [17]

Na figura 1 apresenta-se um exemplo ilustrativo do sistema pull no processo de vendas, ou seja, o cliente é que vai "desencadear a ação ". No nosso exemplo de aquisição de um par de calças era o cliente que teria que se dirigir ao alfaiate e pedir umas calças novas.

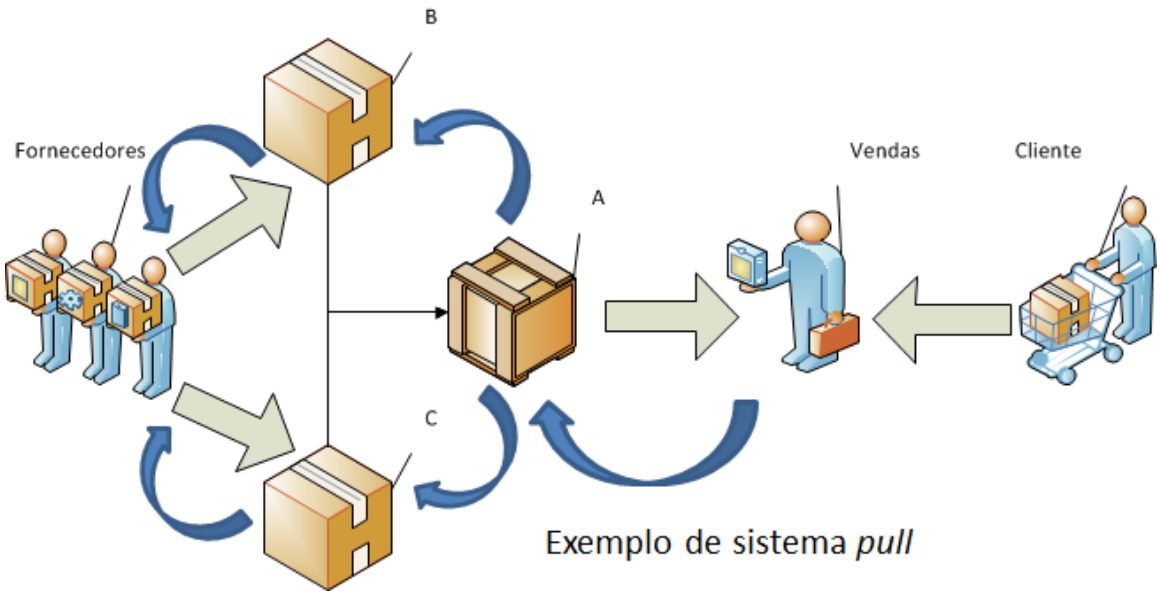

Figura 1 – Ilustração de sistema pull [3]

2.2.3 – Estratégias Push

Paulo Silvestrin [4] escreveu que "Mensagens *Push* é o nome dado ao tipo de comunicação via Internet onde a requisição para uma dada transação é iniciada pelo servidor e não pelo cliente como normalmente acontece. O objetivo desta tecnologia é fazer com que o cliente receba dados que são destinados a ele sem ele ter que consultar o servidor. Mensagens, ou notificações *push*, são muito utilizadas em aplicações de e-mail, notícias, resultados de esportes, monitoramento de rede de sensores, entre outros.

Para a implementação de mensagens push em aplicações para dispositivos móveis os fabricantes do sistema operacional, normalmente, oferecem frameworks. Por

exemplo: *Google Cloud Messaging* para aplicações Android, *Apple Push Notification Service* para aplicações iOS, Windows Azure para aplicações Windows."

Como podemos verificar na figura 2, e voltando ao exemplo de aquisição de um par de calças, neste caso é o cliente que se dirige a uma loja para comprar a peça de vestuário e comunica ao funcionário que se encontra na caixa que deseja comprar umas calças. De seguida, este verifica se existe em stock e responde ao cliente se o produto se encontra disponível ou se tem que esperar pois naquele momento não existe em stock.[12]

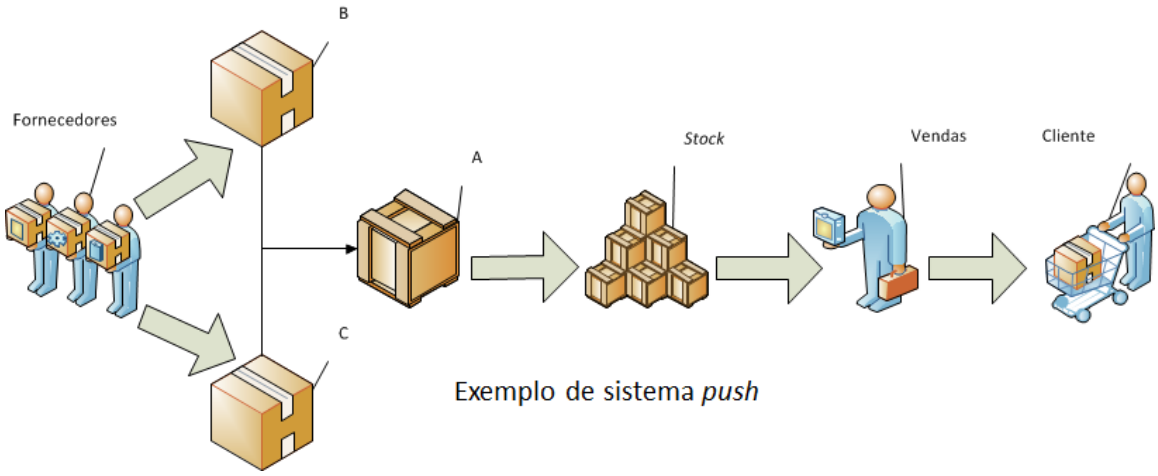

Figura 2 - Ilustração de sistema push [3]

2.2.4 – A importância da tecnologia push

Com a evolução dos anos, tornou-se cada vez mais importante a utilização do *push* nos sistemas operativos móveis, pois estes precisam cada vez mais de ser atualizados ao

segundo, na sequência dos novos tipos de utilização como por exemplo, a receção de notificações de *emails*. Para tal é preciso criar um canal de comunicação entre o dispositivo e o servidor, como por exemplo, quando nos sintonizamos com uma estação de radio é criado um canal de comunicação entre os dois. [17]

Através desse canal de comunicação o dispositivo obtém informações atualizadas instantaneamente uma vez que utiliza o canal para comunicar com o servidor. Assim, a criação de uma comunicação não implica um elevado consumo de energia do equipamento móvel, para além de não ser necessário aguardar que o servidor envie uma resposta para um pedido realizado.

Poderíamos pensar que, como existe um canal aberto entre o servidor e o telemóvel, este está sempre em "alerta", gastando assim bateria e consumo de dados de rede, contudo isto não acontece porque a aplicação fica em *stand by* até receber uma notificação. [18]

Outra vantagem do sistema *push* é que permite a atualização de *software* automaticamente. Um exemplo disso é o canal de comunicação existente entre um dispositivo com o sistema operativo móvel Android e o *Google*, caso em que seja lançada uma nova versão de uma aplicação esta será atualizada no sistema sem ser necessário verificar periodicamente no servidor a existência de uma nova versão.

O mesmo acontece quando inserimos um novo contacto no telemóvel. Este é automaticamente guardado na nossa conta Gmail, que foi inserida quando o Android foi iniciado pela primeira vez. Assim quando precisarmos de trocar de Android as informações guardadas no Android antigo nunca serão perdidas.

Caso não existisse esse canal teria que ser utilizado o sistema *pull* que traz inúmeras desvantagens tanto para o servidor (uma vez que atender a vários pedidos ao mesmo tempo torna o servidor obsoleto) como para o telemóvel (consumo de energia, processador, antena para a ligação à internet, etc). [5]

Perante as características identificadas das duas opções de comunicação, optamos por selecionar o sistema *push* para o desenvolvimento do NearUs,

principalmente por esta permitir receber mensagens instantaneamente sem ser necessário fazer pedidos ao servidor.

2.3 – Tecnologias de BroadCasting

2.3.1 – Análise de Soluções

Tendo sido identificado que a metodologia ideal para o NearUs é o *push (broadcast)*, tornou-se necessário fazer um levantamento dos serviços existentes que se enquadravam melhor no estado atual do NearUs. Numa primeira fase optamos por desenvolver o NearUs para o sistema operativo Android e para o iOS, por isso o trabalho de identificação do sistema de mensagens *broadcast* centrou-se nestes dois sistemas. O desenvolvimento de uma versão para o sistema operativo Windows Phone não foi considerada nesta primeira fase, podendo no futuro vir a ser implementado.

Nesse levantamento encontramos os serviços "nativos" da Google (GCM – *Google Cloud Messaging*) e o da Apple (APNS – *Apple Push Notification Service*) e os serviços Direct 100, Zero push e o Parse. Começando por comparar os preços praticados por cada um dos serviços em que constatamos que tanto o *Google Cloud Messaging* como o *Apple Push Notification Service* são completamente gratuitos.

O serviço Parse tem um plano free (basic) que permite a realização de 1 milhão de notificações por mês. Assim sendo este plano é inadequado para o NearUs uma vez que o número de notificações que o NearUs precisa é superior a 1 milhão. O serviço Parse disponibiliza ainda um serviço PRO, com um limite de 5 milhões de notificações por mês, que poderia viabilizar a sua utilização no âmbito do projeto. Contudo esta opção tem um custo de 199$ por mês, sendo este valor insuportável para o projeto.

O mesmo acontece com o serviço Zero push, em que o custo do serviço por mês é mais baixo do que o Parse; o plano mais barato é de 9,99$ por mês mas neste caso só temos 200 mil notificações por mês, logo este serviço não serve para a aplicação.

Por fim analisamos o serviço direct 100. Este serviço é um pouco diferente dos dois já referidos, pois não tem mensalidade, apenas se paga o que se consume. Contudo, tem um custo de 1 cêntimo por cada notificação recebida, logo torna-se também incomportável a utilização deste serviço.

Em termos de aprendizagem os 3 serviços pagos oferecem tutoriais e exemplos mais simples que o GCM e o APNS. Isto deve-se ao facto que estes serviços oferecerem uma interface mais intuitiva na gestão de contas e de o seu próprio código ser mais percetível do que nos serviços nativos da Google e da Apple. Outra vantagem dos serviços pagos é que não é preciso criar e gerir contas Gmail, no caso do GCM e apple id's, chaves públicas e privadas, no caso do APNS; pois estes serviços gerem estes dados automaticamente.

Por último é de salientar que durante a pesquisa que efetuamos, reparamos que muitos dos serviços disponibilizados utilizam a ferramenta de notificações *push* do Google e da Apple. Por isso, se estes serviços utilizam o GCM e APNS com sucesso, estas tecnologias apresentam-se como adequadas para a aplicação NearUs a desenvolver.

Por estes motivos optamos pelos serviços "nativos" de cada sistema operativo móvel, havendo porém as suas desvantagens: no caso do GCM, o tamanho máximo de uma notificação é de 4kB e o tempo de espera de uma notificação varia entre 0 segundos e 4 semanas; e no caso do APNS uma notificação fica em espera no máximo de 28 dias.

Todavia não colocamos de parte a possibilidade de voltar a analisar outros serviços em evoluções futuras da aplicação, dependendo da evolução do NearUs, do cenário socioeconómico e da evolução das soluções tecnológicas e dos serviços prestados pelas empresas.

A tabela 1 resume toda a informação por nós referida supra.

	Preço	Complexidade	Suporte	Desvantagens
APNS	Grátis	Requer maior tempo de aprendizagem.	X	Notificação em espera 28 dias
GCM	Grátis	Requer maior tempo de aprendizagem.	X	Tamanho máximo da notificação de 4 kB Notificação em espera 4 semanas
Parse	• Plano grátis até 1 milhão de notificações. • Plano Pró com 5 milhões de notificações com o preço de 199$.	Fácil Aprendizagem	• Utiliza o APNS. • Tem suporte para o GCM.	Número limitado de notificações e preço
Zero Push	Plano com 200 mil notificações com o preço de 9,99$.	Fácil Aprendizagem	• Utiliza o APNS. • Utiliza o GCM.	Número limitado de notificações e preço
Direct 100	Plano de 1 cêntimo por notificação.	Fácil Aprendizagem	Não foi encontrada informação.	Número limitado de notificações e preço

Tabela 1 - Síntese dos serviços encontrados

2.3.2 – Soluções Selecionadas

Como já referimos, escolhemos o *Google Cloud Messaging* como sendo a melhor opção para enviar mensagens *broadcast (push)* para os dispositivos que utilizam o sistema operativo Android.

Visto que este serviço se encontra no *Google Cloud Platform*, indiretamente estaremos a utilizar o *cloud computing,* como podemos verificar na página oficial (https://cloud.google.com/compute/).

Nos pontos seguintes vamos contextualizar o *cloud computing* e explicar a arquitetura do GCM e APNS.

2.3.2.1 – Cloud Computing

2.3.2.1.1- História da Cloud Computing

Antes de entrarmos no estudo dos vários aspetos relacionados com o *cloud computing* importa fazer uma brevíssima resenha histórica sobre o seu início.

Assim como a sua abstrata representação também não existe uma descrição clara para o termo *cloud computing* por isso é pouco difícil dizer ao certo aonde é como tudo começou.

Em 1960, John McCarthy [6] debruçou-se sobre a ideia de *time sharing* (partilhar o tempo), onde um ou mais utilizadores poderiam utilizar um computador ao mesmo tempo para realizar várias tarefas, intercalando o tempo de uso entre um utilizador e outro.

John McCarthy percebia que esta era a melhor forma de rentabilizar o computador, que naquele tempo era um recurso muito caro, diminuindo assim os custos com este. É no fundo a ideia por detrás da *cloud*.

Nesse mesmo século, o físico Joseph Carl Robnett Licklider, fez parte da ARPA (*Advanced Research Projects Agency*) centrando a sua pesquisa na ideia de atualizar os computadores e pô-los a comunicar entre eles.

Joseph Licklider foi um dos pioneiros na comunicação de um computador com outro permitindo assim criar, o que nós chamamos, internet.

Foi especialmente nestes dois trabalhos que foi possível criar a *cloud computing* e entender a evolução desta, como podemos constatar na figura 3 [6].

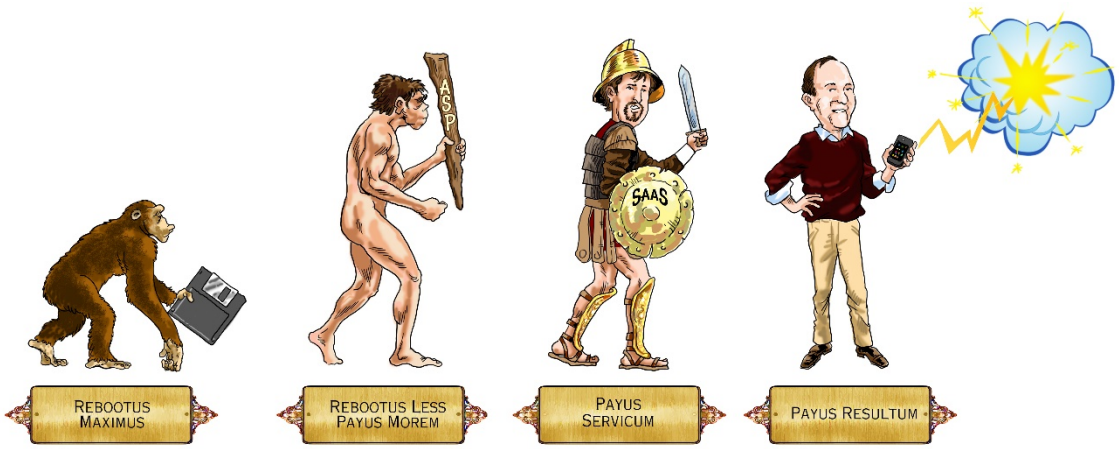

Figura 3 - Evolução do homem (imagem retirada do portal athenahealth.com)

2.3.2.1.2- Definição

Quando falamos na arquitetura do NearUs, reparamos que foram utilizadas as *frameworks* GCM e APNS para as notificações *push*. Estas assentam sobre a *cloud* por isso e indiretamente a *cloud* está representada na arquitetura da aplicação por nós desenvolvida.

Se repararmos até no próprio nome do serviço da Google, está o termo *Cloud*, pois este tema está cada vez mais presente nos dias de hoje.

Nas outras áreas do saber sempre que queremos representar algo abstrato utilizamos uma nuvem - uma imagem representativa da nossa ideia. Utiliza-se também

este conceito nos desenhos de redes, isto deve-se á representação de algo que não necessita de ser conhecido.

Assim quando queremos representar, por exemplo a internet, utiliza-se uma nuvem, podemos não entender ao pormenor como esta funciona, mas assim sabemos que existe algo lá.

Através desta ideia o *cloud computing* é representado por uma nuvem.

O conceito *cloud computing* é relativamente novo e será cada vez mais utilizado nos próximos anos, sendo que o seu objetivo é a utilização de aplicações em qualquer lugar independentemente da plataforma, utilizando para o efeito a internet, sem ter que as instalar.

Há uns anos atrás guardávamos os nossos dados nos computadores, disquetes, cd, pen-drive, entre outros, e tínhamos todas as aplicações necessárias para o nosso dia-a-dia, instaladas num computador.

Contudo este conceito foi alterado nos últimos anos, principalmente nas empresas, onde as aplicações estão num servidor e são acedidas por um computador autorizado, dentro de uma rede.

Se uma aplicação não utilizar *cloud computing* e especialmente as aplicações pagas, é necessários pagar as licenças da aplicação para cada computador, ora numa empresa esta solução poderá ser muito dispendiosa.

As aplicações ou os ficheiros que utilizam o *cloud computing* como arquitetura, não precisam de estar instalados ou guardados no nosso computador, pois o seu conteúdo fica acessível na internet. Do outro lado, ou seja, quem fornece o serviço de *cloud computing* fica encarregue de todas as operações de gestão e armazenamento de dados [6].

Um exemplo de um futuro próximo é o que se consta que a Canonical quer fazer, uniformizar o sistema operativo deles do computador, *tablet* e *smartphone* sendo assim possível aceder aos ficheiros via *cloud computing*, tendo só nestes três tipos de dispositivos instalado o sistema operativo deles. Por exemplo, estamos a assistir a um episódio de uma série em casa e precisamos de sair a meio do episódio para apanhar o

metro. Durante a viagem de metro continuamos a ver o episódio, sem que este tenha que ser transferido de um dispositivo para outro "fisicamente" [16].

Este é um dos principais objetivos do *cloud computing*, a facilidade do acesso às aplicações e aos ficheiros. Contudo, esta realidade em alguns países (incluindo Portugal) ainda está um pouco distante visto que nem toda a região de Portugal tem acesso á internet e mesmo aquelas que têm, a velocidade desta ainda é limitada.

Contudo, o acesso às aplicações e aos programas de forma "direta" não é a única vantagem do *cloud computing*, existem mais, entre elas:

- O acesso, pelo utilizador, aos variados programas, independentemente destes estarem associados a um determinado sistema operativo;
- Todas as tarefas relacionadas com a gestão das aplicações, por exemplo, necessidade de mais *hardware, backups*, manutenção e segurança, fica ao encargo de quem fornece o serviço de *cloud computing*.
- Em ambiente empresarial, esta solução torna-se muito rentável visto que qualquer utilizador (devidamente autorizado) pode aceder aos seus ficheiros de trabalho independentemente do local e do *software*, por exemplo, um trabalhador pode editar um ficheiro no local de trabalho, através do Windows e em casa através do Linux.
- Atualmente quem fornece este tipo de serviço, já vem com grande robustez, ou seja, se por acaso o servidor falhar, "entra em ação" um secundário já com os *backups* feitos do anterior.
- Outra das enormes vantagens do *cloud computing* é uma melhor gestão dos gastos por parte do utilizador, enquanto que em alguns antivírus e sistemas operativos pagamos a licença por 1 ano, neste serviço pagamos apenas o que utilizamos, ou seja, se o utilizador só quiser o serviço por 1 mês, só paga aquele mês e não o ano todo, como no sistema anterior.
- Em alguns casos, quem fornece o serviço de *cloud computing* requere que sejam instalado no cliente um programa para aceder ao mesmo, contudo, toda a gestão e *backups* de dados fica ao encargo de quem fornece o serviço.

Como já referido, a representação mais abstrata do *cloud computing* é a nuvem, pois o utilizador não precisa de saber o funcionamento desta, aonde se encontram alojados os dados, quem faz o *backup*, quantos servidores existem no serviço, etc. O maior objetivo é dar a conhecer ao cliente que existe algo lá e está disponível, como podemos constatar na figura 4 [15].

Figura 4 - Ilustração de cloud computing (imagem retirada do portal javatpoint.com)

2.3.2.1.3- Exemplos de aplicações em cloud computing

Apesar do termo "*cloud computing*" ser relativamente novo, muitas empresas já o adotaram, devido às inúmeras vantagens que este possui. São disso exemplo empresas como a Google (Gmail), Microsoft (Hotmail), Dropbox (partilha de ficheiros entre utilizadores, via internet) ou até mesmo portais de partilha de vídeos como o Sapo e o Youtube.

Estes são exemplos de serviços que nos são fornecidos, alguns gratuitamente, que assentam sobre o conceito de *cloud computing* aonde podemos aceder a qualquer ficheiro (dropbox) ou visualizar qualquer vídeo (Youtube/Sapo), independentemente dos dispositivos ou do sistema operativo nele instalado. Neste sentido vamos referir outros quatro exemplos de serviços que mantêm a mesma ideia:

- Google Drive: Através de uma conta Gmail, podemos ter acesso ao Google Drive, que permite ser utilizado como um Office *on-line*, ou seja, podemos partilhar documentos do tipo texto, apresentação, folha de cálculo, formulários e desenho com outras pessoas e estas editarem ao mesmo tempo. Este tipo de pacote de aplicações é muito utilizado e útil para quem deseja trabalhar em grupo.

- Amazon: Esta empresa é conhecida como umas das maiores vendedoras de artigos *on-line* do mundo, ao ponto de no pico dos pedidos ocorridos no Natal terem a necessidade de reforçar em termos de hardware os seus serviços. Contudo depois desta época festivaleira, a empresa notou que ficou com os recursos de *hardware* "parados", pelo que decidiram arrendar este espaço durante o resto do ano. Um exemplo disso são os serviços *Simple Storage Solution (S3)* para armazenar qualquer tipo de dados e *Elastic Compute Cloud (EC2)* para uso de máquinas virtuais

- iCloud: A Apple também lançou, em junho de 2011, o *iCloud*, um serviço que guardava todas as músicas, fotos, documentos, contatos, agenda, entre outros, de um utilizador Apple. Assim este podia atualizar a agenda no seu Mac book e esta seria automaticamente atualizada no iphone.

- GCM (Google Cloud Messaging): Serviço da Google que permite a comunicação do servidor com os dispositivos Android, através de *push notification*.

2.3.2.2 – Google Cloud Messaging (GCM)

2.3.2.2.1 – Descrição

Paulo Vítor Silvertrin [4] escreveu, relativamente ao *Google Cloud Messaging*: "Enviar mensagens a partir de um servidor para dispositivos móveis não é uma tarefa fácil. *Smartphones* e *Tablets* estão sujeitos a conexões com Internet instáveis, falta de bateria, entre outros problemas que tornam difícil essa comunicação. Uma solução é fazer com que o aparelho, quando estiver *online*, envie mensagens periódicas ao servidor para checar se possui alguma mensagem para ele. No entanto, essa solução é muito onerosa em termos de consumo de bateria e de envio de dados pela Internet para que o dispositivo tenha, em tempo real, as mensagens destinadas a ele.

Para resolver esse problema dos desenvolvedores de aplicativos Android, o Google disponibiliza o *Cloud Messaging* (GCM). Anteriormente, esse serviço era provido pelo Cloud to *Device Messaging Framework* (C2DM) que foi oficialmente descontinuado em Junho de 2012."

A opinião deste autor relativamente ao GCM vai de encontro com o contexto de desenvolvimento da aplicação informática NearUs, pois que em Portugal o acesso á internet tem melhorado, contudo existem ainda zonas aonde a velocidade desta é escassa. Isto seria um problema caso tivéssemos que fazer constantemente pedidos ao servidor para saber se existe uma mensagem nova.

Foi referido também que este serviço da Google funciona utilizando as notificações *push*, ou seja, o telemóvel não verifica regularmente se existem mensagens

novas, pois esta operação só traria desvantagens para o nosso problema, como a necessidade do uso da bateria, conexão com a internet, sobrecarga do servidor, etc. [18]

A ferramenta GCM veio trazer mais uma forma de comunicação entre o dispositivo Android e os servidores, relativamente ao *broadcast*, por isso é uma ferramenta muito útil, poderosa e que podemos utilizar para diversos fins.

Utilizando os exemplos que Lucas Freitas [7] referiu, "Podemos, por exemplo, desenvolver um aplicativo para o Android que, ao receber do GCM uma mensagem com o conteúdo "TirarFoto", o aplicativo simplesmente captura uma imagem através da câmera do dispositivo e envia a imagem capturada para um determinado endereço de e-mail; ou se preferir uma mensagem com o conteúdo "MostrarLocalização" para receber via e-mail ou SMS a localização geográfica de um dispositivo. Desta forma podemos facilmente criar um aplicativo que rastreie o seu dispositivo em caso de roubo. Este é apenas um exemplo dentre as inúmeras possibilidades que pode se beneficiar com o uso do GCM. O Whatsapp, por exemplo, é um aplicativo que poderia se beneficiar do GCM para troca de mensagens entre usuários."

2.3.2.2.2 – Arquitetura GCM

O GCM é o serviço da Google responsável pelo envio de dados para os dispositivos Android. No resto deste ponto iremos proceder a uma descrição da arquitetura do GCM, podendo ser encontrada informação adicional em http://developer.android.com/google/gcm/index.html.

Visto que uma aplicação NearUs irá ser implementada sobre dispositivos com o sistema operativo Android e dado que o serviço GCM é fornecido pela própria Empresa que desenvolve este sistema operativo móvel (Google) esta é a forma otimizada para enviar mensagens entre o servidor e o dispositivo.

O GCM possui várias vantagens, entre elas: utilizar poucos recursos do dispositivo, pouca utilização do servidor e não necessita de estar sempre conectado com a internet,

pois se a ligação com esta falhar, a mensagem fica em fila de “espera” para quando o telemóvel voltar a conectar-se o dispositivo recebê-la.

Para que esta solução funcione corretamente é necessário que a aplicação possua um *Broadcast Receiver* (é um componente do Android que responde a determinados “eventos” enviados pelo sistema) para que a aplicação atue quando existir uma mensagem nova.

O GCM é altamente seguro pois os servidores da Google fornecem um ID único para cada dispositivo (*token*) e só quem conhece estes ID’s pode enviar mensagens para os dispositivos [7].

A figura 5 ilustra a arquitetura do sistema. Este serviço só está acessivel para quem possui um aparelho com Android de versão superior á 2.2.

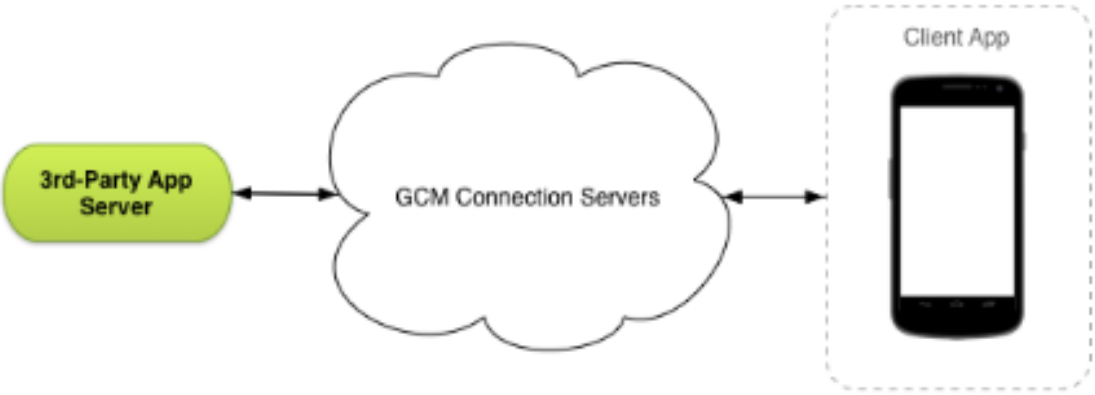

Figura 5 - Arquitetura do Google Cloud Messaging [7]

O *Cloud Message* utiliza uma conexão TCP entre o servidor e o dispositivo. O servidor está sempre em estado de *listen*, isto é, usa um socket TCP na porta 5228 e fica á espera que exista novos pedidos.

Logo que o aparelho se conecte com a internet, este abre a conexão TCP com o servidor. Mesmo quando a aplicação está em *background* ou o dispositivo está em modo suspenso, esta conexão nunca é encerrada, por isto é que o GCM é muito fíavel.

São necessários 3 ID's para que esta comunicação aconteça com uma maior segurança dos dois lados, eles são:

- *Server ID*: quando registamos na consola do GCM, este ID é fornecido para que este identifique qual o servidor que está a enviar as mensagens para os dispositivos.
- *Application ID*: como o próprio nome indica, este ID identifica a aplicação que recebe as mensagens.Por exemplo, a aplicação X tem o *Application ID* 123456.
- *Registration ID*: Este ID identifica o dispositivo com a aplicação, ou seja, é a junção do *token* do dispositivo com o *Application ID.*

O servidor GCM trata a mensagem recebida, seguindo os seguintes parâmetros:

- *Registration id's:* identifica o dispositivo que recebe a mensagem. Este id é o mesmo do *Registration ID*, acima referido.
- *Delay while idle:* este parâmetro verifica se o dispositivo está disponível para receber as mensagens.
- *Collapse key:* identifica a mensagem, por exemplo o dispositivo x vai receber as mensagens com o *Collapse key* 0,1,2,3… O *Collapse Key* é utilizado para que o telemóvel receba um conjunto de mensagens e não uma de cada vez, evitando assim um número desnecessário de notificações.
- *Time to live:* Este parâmetro indica o tempo de vida da mensagem, que pode variar entre 0 segundo e 4 semanas.
- *Data:* é o tamanho máximo das mensagens (4KB).

A figura 6 ilustra um exemplo da utilização destes parâmetros, recebido pelo GCM, em formato JSON.

Figura 6 - Ilustração de uma mensagem recebida pelo GCM (imagem retirada do portal stackoverflow.com/questions/8473918/how-to-set-a-timeout-expiration-on-a-c2dm-message).

O servidor GCM recebe as mensagens e "pede" ao servidor do NearUs os *registration id's* dos dispositivos que irão receber a mensagem. Este processo é feito por *broadcast* de mensagens (processo pelo qual se transmite ou difunde determinada informação, tendo como principal carateristica que a mesma informação está a ser enviada para muitos receptores ao mesmo tempo), que se chamam *Intents* (normalmente são criadas a partir de ações do utilizador e representam a ***intenção*** de se realizar algo, como iniciar a aplicação de e-mail do Android ou então iniciar a reprodução de uma música) no Android. Este (Android) tem a principal tarefa de verificar e tratar as mensagens [4].

2.3.2.2.3 – Obter acesso ao GCM

Já descrevemos o GCM e explicamos a sua arquitetura, faltando agora entender como se processa na prática e quais os passos necessários para que esta seja implementada no nosso sistema.

Para utilizar o GCM é preciso criar um projeto na consola do Google, para isso acedemos ao seguinte link: https://code.google.com/apis/console.

Nesta pagina é necessario definir um nome para o projeto e um identificador (sendo este ultimo preenchido por omissão que, regra geral, não deve ser alterado).

A figura 7 ilustra o resultado do Google Console, depois de criado o projeto.

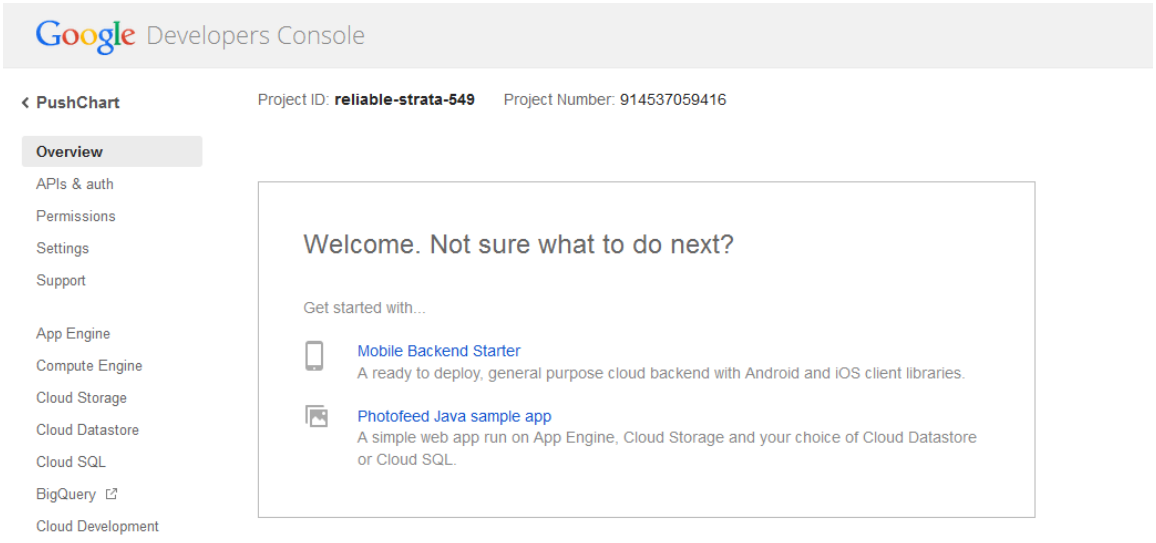

Figura 7 - Ecrã com o Projeto Criado

A fase seguinte envolve a ativação do *Google Cloud Messaging* for Android, através da criação de uma chave de acesso ao GCM por parte do telemovel. Esta chave é o Server Key referido no sub-capitulo anterior. Para a execução desta operação, necessitamos de indicar o tipo de chave que queremos (*Server Key, Browser Key, Android Key, iOS Key*). Para o presente projeto, a chave a criar é do tipo *Server Key*, uma vez que esta é que melhor se adqua á criação de um GCM, pois as restantes não se enquadram no broadcast. Por exemplo o *Android Key* é uma chave utilizada na visualização do Google Maps no Android.

Depois de criada a chave, podemos restringir o acesso ao servidor GCM, ou seja, podemos definir, através dos IP's, que só o servidor x e o servidor y podem ter acesso aquele projeto, permitindo assim uma maior segurança na troca de informação. Mesmo que alguém descubra a nossa chave, se o IP do servidor não constar nas opções não poderá aceder ao projeto [13].

2.3.2.3 – Apple Push Notification Service (APNS)

2.3.2.3.1 – Descrição

Conforme indicado anteriormente no processo de pesquisa para encontrar a melhor solução para o sistema operativo mobile iOS, optamos pelo *Apple Push Notification Service*.

Este ponto inclui uma descrição e a arquitetura de segurança do APNS.

Assim, como acontece no GCM, o APNS utiliza o *cloud computing* descrito anteriormente.

Como já referimos, o *broadcast* veio revolucionar a forma de comunicação entre dispositivos, oferecendo inúmeras vantagens face ao sistema pull, permitindo que á medida que os *tablets* e os *smartphones* ganhem popularidade as empresas que desenvolvem sistemas operativos móveis criassem um sistema de notificações para facilitar a implementação de soluções com funcionalidades cada vez mais elaboradas.

Para os *smartphones* que possuem Android referimos o GCM (*Google Cloud Messaging*) existindo um sistema idêntico, em termos de arquitetura e conceito, para os sistemas operativos móveis da Apple, denominado, o APNS (*Apple Push Notification Service*)

O *Apple Push Notification Service* (APNS) é utilizado para enviar notificações *broadcast* (no ambiente iOS são chamadas de notificações *push*) para todos os dispositivos que possuem o iOS. Assim como acontece no GCM, as notificações são enviadas a partir do servidor NearUs, passando pelo servidor da Apple e por fim chegam ao dispositivo final. [18]

Uma notificação *push* é composta por: token, identifica o dispositivo que irá receber a notificação (este id é único para cada dispositivo) e o payload, conjunto de propriedades da mensagem, normalmente em formato JSON.

O fluxo de uma mensagem *push* é processado da seguinte forma: o servidor da aplicação recebe a mensagem, esta é encaminhada para os servidores da Apple; em seguida vai para o dispositivo da Apple e por fim chega á nossa aplicação, como podemos ver na figura 8.

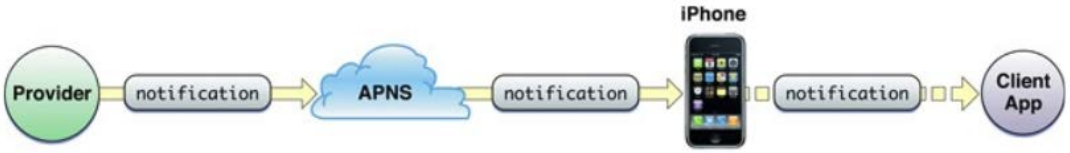

Figura 8 - Fluxo da mensagem push da apple [8]

Caso a notificação seja recusada, o APNS tem um serviço chamado *feedback* que informa o servidor do cliente que a notificação foi recusada por um certo dispositivo através do *token* do mesmo.

Como no GCM, caso o dispositivo fique sem conexão com a internet a notificação fica em espera para quando este volte a conectar-se. A este serviço a Apple deu nome de "*store-and-forward*".

Neste caso, passado algum tempo e se a mensagem não for enviada, esta será eliminada. Na documentação oficial da Apple não encontramos nenhuma referência quanto ao tempo de vida da mensagem. [8]

2.3.2.3.2 – Arquitetura de Segurança

No meio do processo de troca de mensagens, tem que existir certeza por parte do servidor e do dispositivo que estes podem e estão autorizados para a troca das mensagens.

Neste sentido a Apple dividiu este sistema em dois níveis:

- *Connection trust* – Informa que tanto o dispositivo móvel como o APNS estão autorizados para a troca de mensagens;

- *Token trust* – Para uma maior segurança encripta o *token* para que este não possa ser acedido externamente, ou seja, quando o APNS recebe a chave do *smartphone* este desencripta e verifica se o *token* é valido.

Tanto no *connection trust* como no *token trust* existe troca de informação entre o dispositivo, o APNS e o servidor cliente. No caso do *Connection trust* este está divido em duas partes:

- Service-to-Device Connection Trust – Em que existe troca de dados entre o *smartphone* e o APNS.
- Provider-to-Service Connection Trust – Onde se verifica a comunicação do APNS com o servidor do cliente.

A figura 9 diz respeito ao *Service-to-Device Connection Trust*. Neste processo são criados dois certificados, o do APNS e do dispositivo.

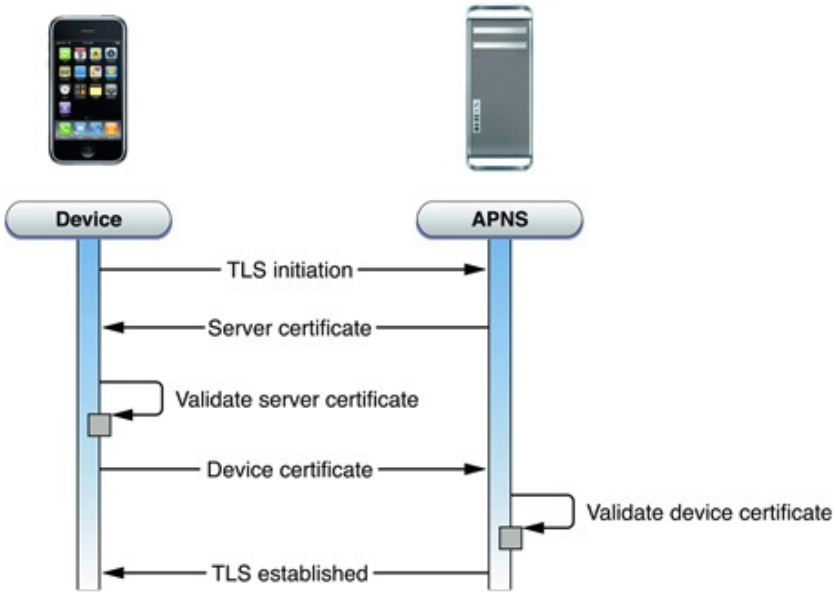

Figura 9 - Service-to-Device Connection Trust [8]

Como podemos verificar o dispositivo inicia a comunicação através de uma conexão *peer-to-peer*, utilizando uma *Transport Layer Security* (TLS) e em resposta o APNS envia o seu certificado para o telemóvel validar e enviar o dele. Por fim o APNS valida e estabelece a conexão.

A figura 10 ilustra o *Provider-to-Service Connection Trust***,** e à semelhança do *Service-to-Device Connection Trust* existem dois certificados. Mas desta vez os certificados são do servidor cliente e do APNS, sendo que quem inicia a conexão *peer-to-peer* é o servidor cliente.

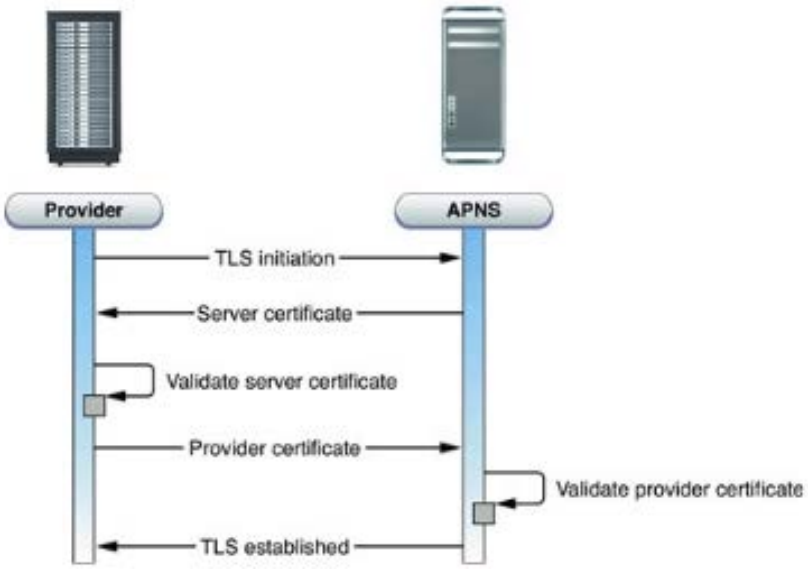

Figura 10 - Provider-to-Service Connection Trust [8]

Na figura 11 apresentamos o funcionamento do *Token Trust***,** onde existe duas situações: a primeira em que o dispositivo requere a conexão através do *token*, o APNS recebe e desencripta o *token* e valida com o certificado do telemóvel e envia a resposta ao mesmo. A segunda situação em que o servidor cliente envia a mensagem com o respetivos *tokens* autorizados para a receção das mensagens, o APNS recebe esta informação e desencripta o *token* com a sua chave e envia a mensagem com as respetivas informações para o dispositivo.

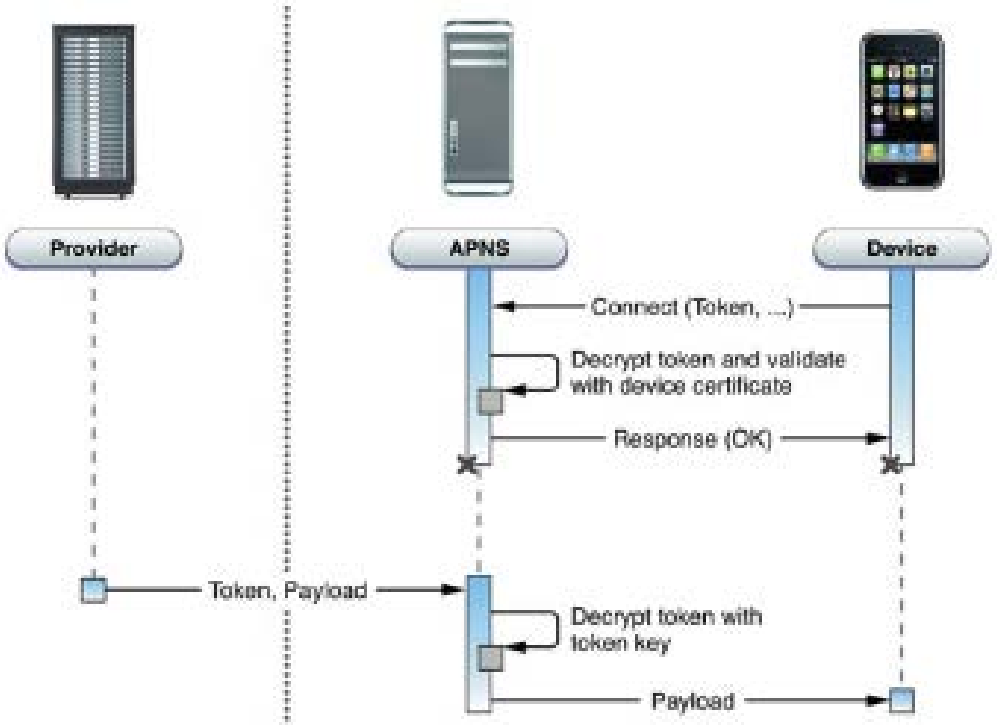

Figura 11 - Token Trust [8]

Tanto o *Service-to-Device Connection Trust* como o *Provider-to-Service Connection Trust* são executados de modo automático, não sendo necessário adicionar nenhum código á nossa aplicação para o funcionamento destes.

Nos dois processos verificamos que tanto o *payload* como os componentes do APNS e as suas credenciais são fundamentais para o bom funcionamento destes. Assim sendo convém ter presentes alguns dos seus aspetos.

Como referido anteriormente o *payload* é um conjunto de informações enviadas para o telemóvel, contudo esta informação não pode ser maior do que 256 bytes e o seu formato é um objeto em JSON.

Além de ser enviada uma mensagem ao utilizador, também é enviado um número com o qual se obtém o ícone da aplicação, assim como o respetivo som que desejamos que o utilizador ouça quando a notificação for entregue.

É ainda de referir que tanto no *Service-to-Device Connection Trust* como no *Provider-to-Service Connection Trust*, verificamos que temos 3 agentes envolvidos nas

operações: um dispositivo Apple (por exemplo: iphone, ipad,…), o servidor cliente, que envia a mensagem através do APNS e por fim o Servidor APNS, que recebe a informação do servidor cliente e envia para o dispositivo. Sendo este ultimo o intermediário destes processos.

Verificamos também que são necessários os seguintes elementos para a implementação do APNS [8]:

- ID's dos dispositivos – Código de identificação único;
- Token – Id do dispositivo com a identificação da aplicação;
- Certificado digital do servidor cliente – Identifica o servidor que envia a notificação para o servidor APNS;
- Certificado digital do servidor APNS – Identifica e garante a legalidade do servidor da Apple.

2.3.2.3.3 – Implementação do APNS

Tal como aconteceu no Android (GCM), vamos analisar a arquitetura do APNS e quais os passos necessários para a implementação da mesma no nosso sistema.

Para enviar notificações *Push*, precisamos de 4 passos: gerar o certificado *Signging Request*, criar um App ID e Certificado SSL, gerar o ficheiro PEM e criar o *Provisioning Profile*.

Gerar o Certificate Signing Request (CSR)

Conforme acabamos de referir o primeiro passo é criar/gerar um *Certificate Signing Request* (CSR). Para isto precisamos de ter instalada a aplicação *Keychain Access*,

e escolher a opção *Request a Certificate From a Certificate Authority*. Caso não tenhamos esta opção basta instalar o WWDR *Intermediate Certificate* a partir do seguinte link:

(https://developer.apple.com/certificationauthority/AppleWWDRCA.cer).

Sendo pedido o e-mail que temos registado na *Apple iOS Dev Center* e o nome da nossa aplicação.

Depois disto precisamos de obter a chave (*key*) criada automaticamente quando se produz um certificado. Para tal, basta abrir as *Keys***,** localizar a chave da nossa aplicação e clicar em *Exportar "NomeApp"*.

No passo seguinte será pedida uma palavra-chave para criptografar o certificado.

Criar um App ID e Certificado SSL

O segundo passo para a criação do *push* na nossa aplicação é a criação do App ID e do certificado SSL sendo este passo feito online, ou seja, na secção *developer* da Apple, (https://developer.apple.com/devcenter/ios/index.action), (no menu do lado direito *Certificates, Identifiers & Profiles* (Figura 12) e escolher *Identifiers* (Figura 13).).

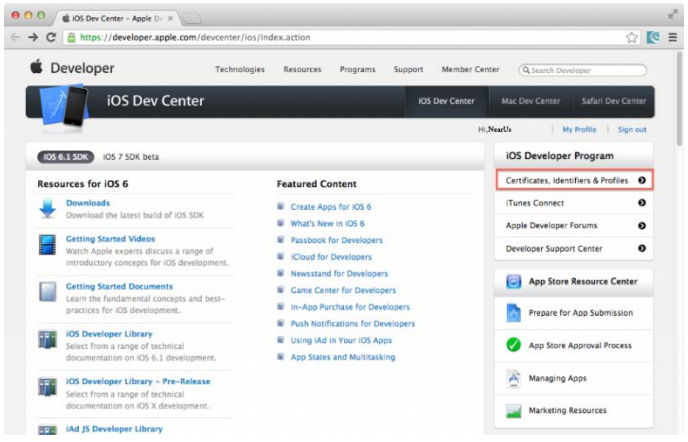

Figura 12 - Certificates, Identifiers & Profiles

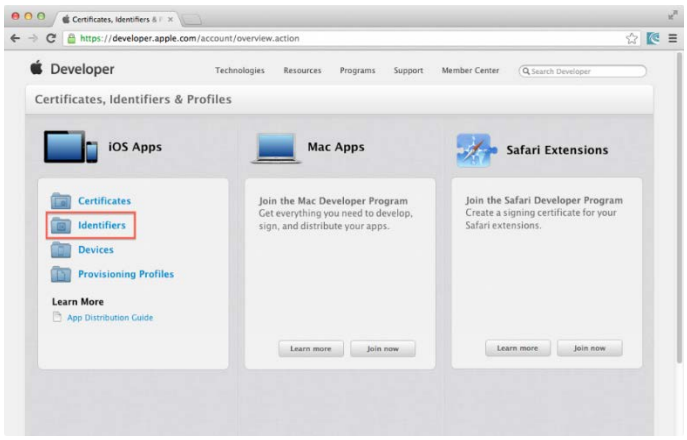

Figura 13 - Identifiers

Depois deste passo, precisamos de criar um APP ID e introduzir um nome para a aplicação assim como para o respetivo pacote e selecionar a opção *Push Notifications***.**

Feito isto, temos já criado um ID para a nova aplicação faltando agora configurá-lo. Para tal e no final do processo da criação dos ID's o portal retorna a lista dos mesmos.

Em seguida escolhemos "*Development SSL Certificate*", como verificamos na figura 14.

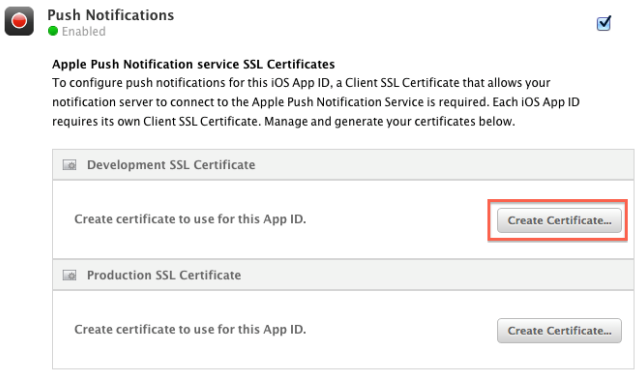

Figura 14 - Create Certificate

Por fim aparece um ecrã onde nos é pedido um ficheiro CSR, onde efetuamos o *upload* do ficheiro criado aquando da geração do CSR.

Gerar o ficheiro Privacy Enhanced Mail

O ficheiro *Privacy Enhanced Mail* (PEM) é fundamental para o funcionamento do *push* do lado do servidor, sendo este o responsável pela codificação dos dados.

Antes de mais, convém relembrar quantos ficheiros já foram criados: o CSR, ficheiro responsável pela chave pública; o *private key*, normalmente com a extensão .p12 (neste ficheiro estão as chaves publicas e privadas) e por fim temos também o certificado SSL (aps_development), com todas as informações sobre o certificado do cliente, junto com as chaves públicas e privadas.

Este passo será feito offline, ou seja, num computador MAC e para isso precisamos de abrir a consola e introduzir o seguinte comando, que irá converter o ficheiro com a extensão .cer em .pem:

```
openssl x509 -in aps development.cer -inform der -out NearUsCert.pem
```

Em seguida precisamos de converter também o ficheiro com extensão .p12 em .pem através do seguinte comando:

```
openssl pkcs12 -nocerts -out NearUsKey.pem –in NearUsKey.p12
```

A palavra-chave longa (*passphrase*) utilizada na geração do CSR, será agora utilizada no ficheiro .p12 para que o openssl possa acedê-lo. Em seguida precisamos de encriptar o ficheiro .pem.

Por fim temos que juntar o certificado com a chave num ficheiro com a extensão .pem, utilizando o comando:

```
cat NearUsCert.pem NearUsKey.pem > ck.pem
```

Para verificarmos que criamos tudo corretamente, utilizamos o comando telnet para tentar fazer uma conexão não encriptada com o servidor APNS. Sendo vezes necessário desbloquear a porta 2195:

```
telnet gateway.sandbox.push.apple.com 2195
Trying 17.172.232.226…
Connected to gateway.sandbox.push-apple.com.akadns.net.
Escape character is '^]'
```

Por fim, falta testar a ligação utilizando o certificado SSL e a chave privada, para isso iremos também utilizar o servidor APNS:

```
openssl s_client -connect gateway.sandbox.push.apple.com:2195 -cert
NearUsCert.pem -key NearUsKey.pem
Enter pass phrase for NearUsKey.pem:
```

Criar o Provisioning Profile

Este passo será feito online, tal como aconteceu na criação do App ID e o Certificado SSL. O *Provisioning Profile* é um perfil de configurações onde estão contidas um conjunto de entidades digitais que ligam exclusivamente os programadores com os dipositivos e permitem que este seja autorizado como teste.

Para criar este perfil precisamos de aceder ao iOS Dev Center e escolher o menu Provisioning Profile, como se ilustra na figura 15.

Figura 15 - Provisioning Profile

Executada a criação do perfil teremos que escolher se queremos que o provisioning profile seja de desenvolvimento ou de distribuição. Para o nosso caso optamos por iOS App Development, como verificamos na figura 16.

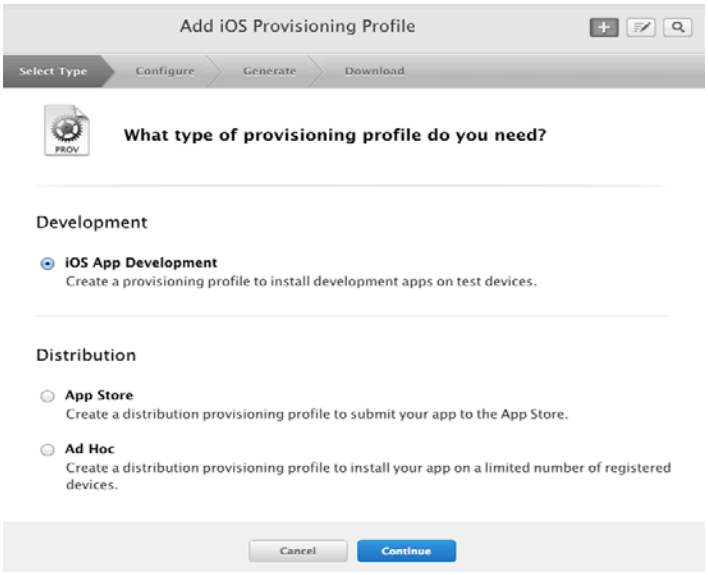

Figura 16 - Add iOS Provisioning Profile

Para executar esta operação é necessário selecionar o App ID. Em seguida escolhemos o certificado de *developer* e os dispositivos autorizados a testar a aplicação.

Por fim damos um nome ao *provisioning profile*, não esquecendo que o XCode já faz *download* deste *profile* automaticamente [14].

Capítulo 3 – Caso de Estudo - Implementação das frameworks no Nearus

3.1 – Contextualização da aplicação Nearus

Nos subcapítulos seguintes vamos descrever a arquitetura do NearUs, contudo convém referir alguns dos conceitos em que o NearUs assenta.

Como já referido anteriormente, o NearUs permite a conversação anónima num determinado espaço que por vezes está associado a um evento.

Para tal, o utilizador cria uma sala virtual que podemos defini-la como uma sala de conversação, muito utilizada nos diversos grupos de conversação *online*. A esta sala está associada a localização do utilizador, utilizando para o efeito as coordenadas GPS (latitude e longitude) do *smartphone*.

Quando o utilizador cria uma sala é-lhe perguntado o nome desta, a sua alcunha (*nickname*) e o raio de ação da sala. Este raio vai permitir a entrada da sala. Por exemplo, é criada uma sala com um raio de 300 metros com a latitude de X1 e a longitude de Y1. Em torno desta localização é criada uma circunferência virtual da sala. Um utilizador faz uma pesquisa de todas as salas em seu redor num raio de 200 metros e com uma latitude

X2 e longitude Y2. Se a diferença entre os eixos das circunferências for inferior à soma dos raios, as estas coincidem, como ilustra a figura 17.

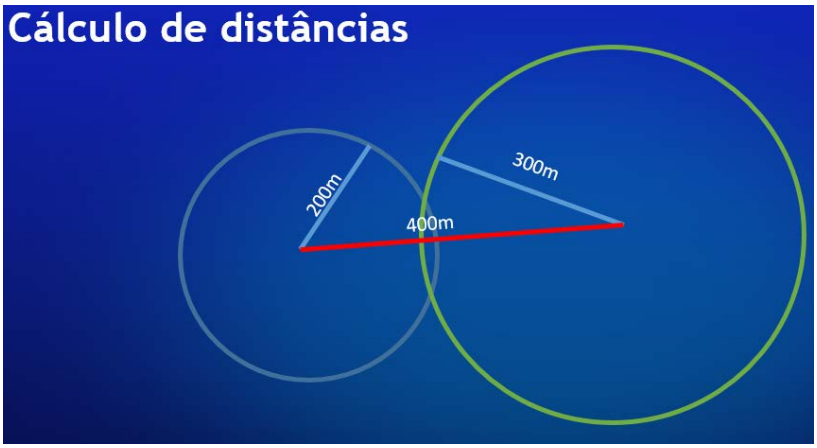

Figura 17 - Cálculo de distâncias em caso de intersecção das circunferências

Caso a diferença entre os eixos seja superior à soma dos raios, as circunferências não coincidem, como apresenta a figura 18.

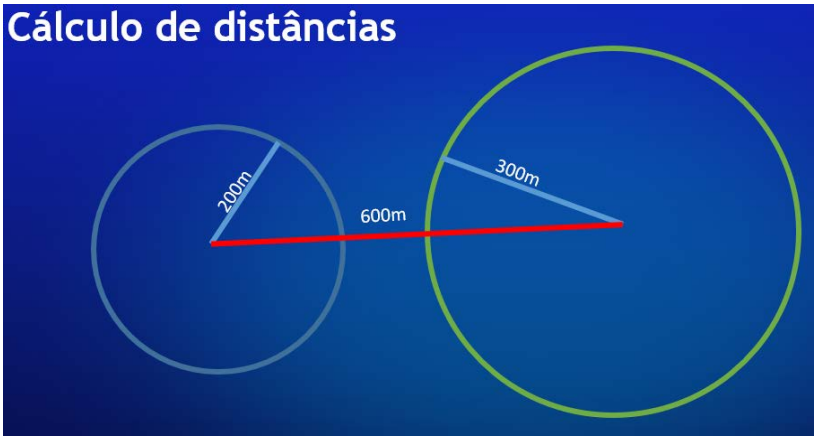

Figura 18 - Cálculo de distâncias em caso de não intersecção circunferências

Estando o utilizador numa sala poderá partilhar mensagens em formato texto ou imagem, podendo posteriormente guardar as imagens no *smartphone* ou partilha-las nas redes sociais Facebook e Twitter.

3.2 – Arquitetura

Numa primeira fase, a arquitetura pensada para o NearUs (podemos considerar como uma arquitetura em fase beta), era apresentada na figura 19.

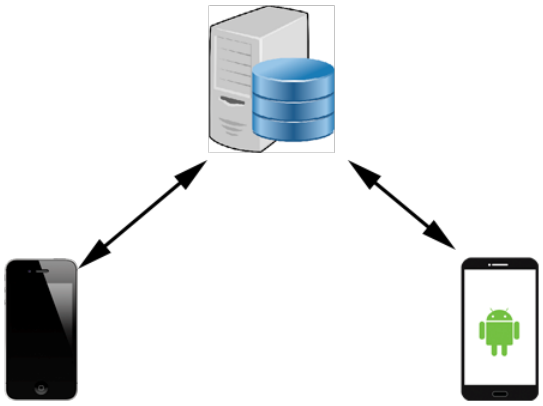

Figura 19 - Primeira Arquitetura do NearUS

Como podemos verificar, existiria um servidor web, com uma base de dados onde seriam guardados os dados relativos às salas, como por exemplo, nome da sala, coordenadas, etc. e as informações relativas aos telemóveis.

Existiriam comunicações nos dois sentidos, como por exemplo, o utilizador pediria ao servidor para listar as salas de um determinado raio; e no sentido inverso, por exemplo, o servidor comunicaria ao telemóvel que existe uma nova mensagem na sala.

Assim optamos pelas *frameworks* de *broadcasting* em que teria de existir sempre um terceiro elemento que servisse de intermediário entre o servidor e o *smartphone* e por isso que com a introdução das *frameworks* selecionadas este modelo altera-se um pouco, como podemos comprovar na figura 20:

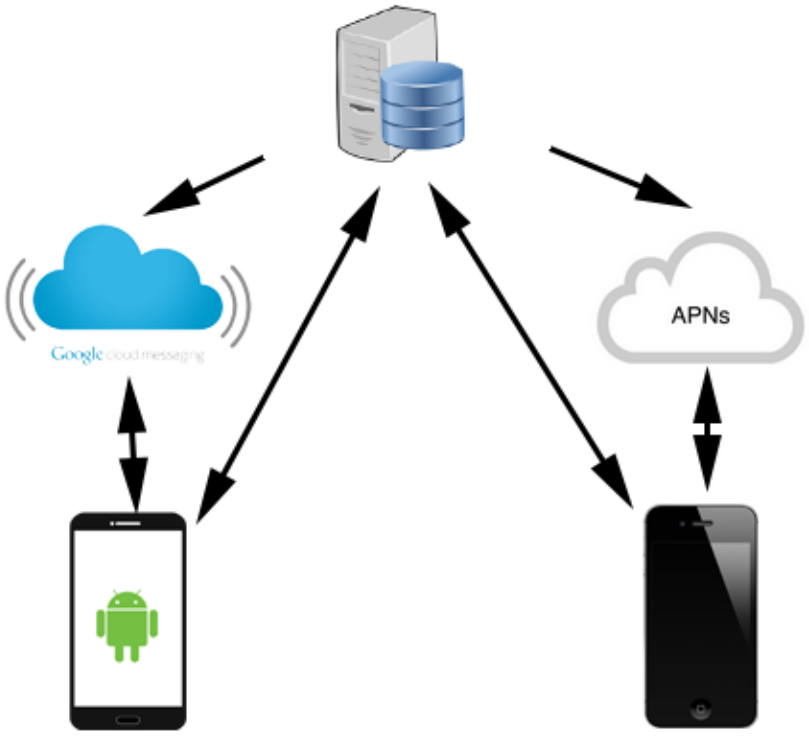

Figura 20 - Arquitetura Atual do NearUs

Como a introdução dos serviços da Google e da Apple, o funcionamento e a comunicação dos *smartphones* com o servidor altera-se, havendo no *broadcast* um elo de ligação entre os dois, partindo-se assim em três componentes: o servidor, a *framework* e o *smartphone*. Por exemplo, um iphone envia uma mensagem numa determinada sala, através de um serviço, o servidor recebe esta informação, procura os *tokens* existentes nessa sala e envia a mensagem para o GCM e o APNS e estes estão encarregues de notificar os respetivos *smartphones*.

Cabe depois, á aplicação receber essa notificação e informar o utilizador que existe uma nova mensagem na sala.

3.3 – Caso de Estudo - Implementação do NearUs

Para entendermos melhor o funcionamento do *broadcast* em relação ao NearUs, vamos socorrermo-nos dos casos de uso, utilizados no desenvolvimento de software.

Nas tabelas seguintes apresentam-se os casos de uso identificados para o NearUs e relevantes no contexto de *broadcasting,* nomeadamente os relativos ao envio de mensagens, procura de *tokens* e receção de mensagens.

Caso de uso:	Enviar mensagem.
Domínio:	Sistema.
Nível:	Objetivo do utilizador.
Actor primário:	Utilizador.
Pré-condição:	O utilizador entrou numa sala.
Iniciador:	O utilizador pretende enviar uma mensagem.
Cenário principal de sucesso:	
	1. O utilizador seleciona no sistema a área correspondente á inserção de texto. **2.** O sistema apresenta um teclado virtual para o utilizador introduzir a informação. **3.** O utilizador introduz essa informação. **4.** O utilizador seleciona o botão "Send" para enviar essa informação para os restantes utilizadores presentes nessa sala.
Extensões:	
	1.a. O utilizador seleciona o botão correspondente ao envio de uma imagem.

	1.a.1. O sistema apresenta um menu para o utilizador escolher se pretende enviar a imagem pela câmara ou pela galeria. **1.a.2.** O utilizador decide uma das opções apresentadas pelo sistema. **1.a.2.a.** O utilizador optou por enviar a foto pela câmara. **1.a.2.a.1.** O utilizador direciona a câmara para o local que pretende fotografar e seleciona a opção "tirar foto". **1.a.2.a.2.** O sistema envia a fotografia para os restantes utilizadores presentes na sala. **1.a.2.b.** O utilizador optou por enviar uma fotografia da galeria. **1.a.2.b.1.** O sistema apresenta as fotografias presentes na galeria. **1.a.2.b.2.** O utilizador escolhe a fotografia que pretende. **1.a.2.b.3.** O sistema envia essa fotografia para os restantes utilizadores presentes na sala.

Tabela 2 - Caso de Uso: Enviar Mensagem

Caso de uso:	Procurar tokens.
Domínio:	Sistema.
Nível:	Sumário.
Actor primário:	O Sistema (NearUs).
Pré-condição:	O utilizador enviou uma mensagem de texto ou imagem.
Iniciador:	O Sistema recebe a mensagem de um utilizador de uma sala.

Cenário Principal de Sucesso:	
	1. O sistema recebe a mensagem e o número de identificação da sala, enviada pelo utilizador. **2.** O sistema procura todos os tokens dos outros utilizadores que estão associados ao número de identificação da sala. **3.** O sistema envia a mensagem para os utilizadores. **4.** O utilizador recebe a mensagem.

Tabela 3 - Caso de Uso: Procurar tokens.

Caso de uso:	Receber mensagem.
Domínio:	Sistema.
Nível:	Sumário.
Actor primário:	O Sistema (NearUs).
Pré-condição:	O sistema recebeu a mensagem.
Iniciador:	O Sistema enviou a mensagem.
Cenário Principal de Sucesso:	
	1. O sistema envia a mensagem para o telemóvel. **2.** A aplicação recebe a mensagem e trata esta. **3.** A aplicação apresenta a mensagem ao utilizador.

Tabela 4 - Caso de Uso: Receber Mensagem.

Modelo de Caso de uso.

A figura 21 representa o modelo dos Casos de uso referidos no subcapítulo 2.6.1 de modo a entendermos os relacionamentos entre estes e o ator.

Na figura está exposto o Sistema NearUs, um ator (o Utilizador) e 3 casos de uso: Enviar Mensagem, Procurar Tokens e Receber Mensagem.

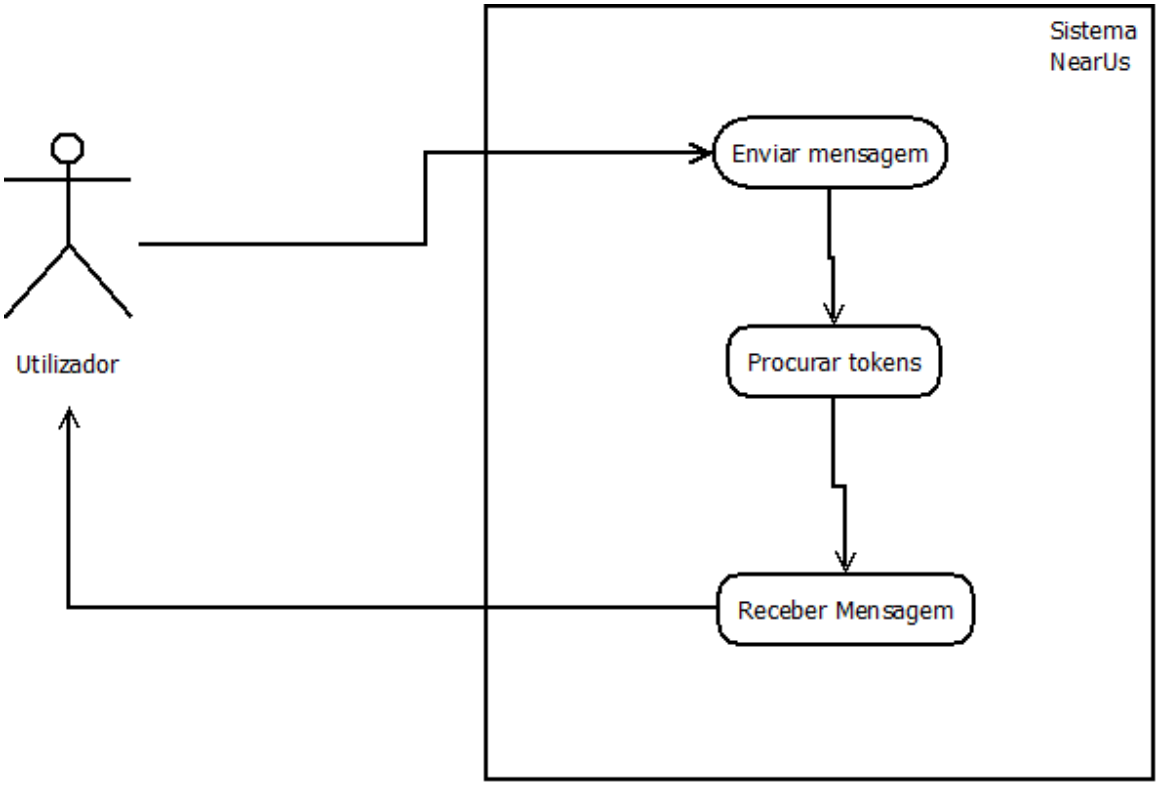

Figura 21 - Modelo Caso de Uso

3.4 – Desenvolvimento do NearUs

Atualmente o NearUs está disponível para iOS e Android; tendo sido desenvolvido, respetivamente, em Xcode e Eclipse com *Software Development Kit* (SDK) Android. Na criação do NearUs foram utilizadas as seguintes linguagens de programação:

- Objetive C – Linguagem de Programação utilizada no desenvolvimento para iOS. Recentemente actualizamos para *Swift*, a nova linguagem de programação para iOS e OS X.
- Java e XML para Android – Utilizadas na criação da aplicação no sistema operativo móvel Android. Ultimamente como a Google lançou o Android Studio, o Integrated Development Environment (IDE) official para o desenvolvimento de aplicações para Android; estamos a proceder a transição do Eclipse para o Android Studio.
- PHP – Linguagem de programação utilizada para processar as operações no lado do servidor.
- Structured Query Language (SQL) – Utilizada para o CRUD (*Create, Read, Update e Delete*) na base de dados.

Para além das *frameworks* GCM e APNS, utilizamos a *Slim Framework* para desenvolver os serviços em *Representational State Transfer* (REST).

Foi também necessário recorrer as seguintes bibliotecas:

- Ptrheads – Biblioteca em PHP para permitir executar processos em paralelo.
- Jsoup – Biblioteca em Java que permite processar os dados em JavaScript Object Notation (JSON) recebidos pelos serviços.
- GCM Library – Biblioteca em java responsável pelo *broadcast* no Android.
- Universal Image Loader – Responsável pela manipulação de imagens em Android.
- Twitter4j – Permite a integração da conta twitter com a aplicação NearUs.
- RestKit – Biblioteca para iOS que permite processar os dados em JSON.
- GPUImage – Biblioteca para iOS que permite manipular as imagens.

Como já referido anteriormente precisamos de conhecer a distância entre 2 pontos para permitir a um utilizador entrar numa sala. Para isso recorremos á função disponibilizada no portal http://www.geodatasource.com/developers/php.

```
<?php

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::*/
/*::
:*/
/*::  This routine calculates the distance between two points (given
the     :*/
/*::  latitude/longitude of those points). It is being used to
calculate     :*/
/*::  the distance between two locations using GeoDataSource(TM)
Products    :*/
/*::
:*/
/*::  Definitions:
:*/
/*::    South latitudes are negative, east longitudes are positive
:*/
/*::
:*/
/*::  Passed to function:
:*/
/*::    lat1, lon1 = Latitude and Longitude of point 1 (in decimal
degrees)  :*/
/*::    lat2, lon2 = Latitude and Longitude of point 2 (in decimal
degrees)  :*/
/*::    unit = the unit you desire for results
:*/
/*::           where: 'M' is statute miles (default)
:*/
/*::                  'K' is kilometers
:*/
/*::                  'N' is nautical miles
:*/
/*::  Worldwide cities and other features databases with latitude
longitude  :*/
/*::  are available at http://www.geodatasource.com
:*/
/*::
:*/
/*::  For enquiries, please contact sales@geodatasource.com
:*/
/*::
:*/
/*::  Official Web site: http://www.geodatasource.com
:*/
/*::
:*/
```

```
/*::         GeoDataSource.com (C) All Rights Reserved 2015
                    :*/
/*::
:*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::*/




function distance($lat1, $lon1, $lat2, $lon2, $unit) {

  $theta = $lon1 - $lon2;
  $dist = sin(deg2rad($lat1)) * sin(deg2rad($lat2)) +
cos(deg2rad($lat1)) * cos(deg2rad($lat2)) * cos(deg2rad($theta));
  $dist = acos($dist);
  $dist = rad2deg($dist);
  $miles = $dist * 60 * 1.1515;
  $unit = strtoupper($unit);

  if ($unit == "K") {
    return ($miles * 1.609344);
  } else if ($unit == "N") {
      return ($miles * 0.8684);
    } else {
        return $miles;
      }
}

echo distance(32.9697, -96.80322, 29.46786, -98.53506, "M") . "
Miles<br>";
echo distance(32.9697, -96.80322, 29.46786, -98.53506, "K") . "
Kilometers<br>";
echo distance(32.9697, -96.80322, 29.46786, -98.53506, "N") . "
Nautical Miles<br>";

?>
```

Tabela 5 - Tabela com a função em PHP para saber a distância entre 2 pontos

3.5 – Testes realizados no NearUs

Com o NearUs já implementado, procedemos a realização de testes com dados fictícios e dados reais. Pretende-se assim testar a eficácia desta aplicação para telemóveis. Para o efeito utilizamos dados recolhidos em eventos por nós analisados (testes com dados reais) e dados que criamos para testar a performance desta aplicação (testes com dados fictícios). De salientar que não foram ferramentas de *benchmarking* para analisar o desempenho, porque, até há data não existe uma ferramenta que faça isso, daí recorrermos aos testes.

3.5.1 – Testes de performance

Como já referimos, na arquitetura do NearUs temos um servidor cliente para poder processar toda a informação enviada do telemóvel. Este servidor contém um Sistema de Gestão de Base de Dados (Mysql) que armazena todos os dados da sala.

Assim sendo, pretendíamos criar um ambiente em que simulasse um caso prático. Para isto, e no caso dos testes de performance do Mysql, criamos uma base de dados com 2 tabelas, uma para guardar os dados relativos aos *tokens* do *Google Cloud Messaging* e outra para guardar os dados relativos aos *tokens* do *Apple Push Notification Service*.

Neste caso prático criamos três tipos de salas: uma com duzentos utilizadores de Android e duzentos utilizadores de IOS (consideramos esta uma sala de pequena dimensão); outra com quinhentos utilizadores de Android e quinhentos utilizadores de IOS (sala de média dimensão) e por fim outra com oitocentos utilizadores de Android e oitocentos utilizadores de IOS (sala de grande dimensão). Geramos também vinte salas para cada tipo de cenário (pequena, média e grande dimensão).

Em suma, neste teste temos sessenta salas com o somatório de sessenta mil utilizadores de Android com os utilizadores de IOS.

A tabela 6 apresenta um resumo que acima foi referido.

Tipo de sala	**Número de Utilizadores**	**Número de Salas**
Pequena dimensão	200 Utilizadores de Android + 200 Utilizadores de IOS	20
Média dimensão	500 Utilizadores de Android + 500 Utilizadores de IOS	20
Grande dimensão	800 Utilizadores de Android + 800 Utilizadores de IOS	20
Total:	60000 Utilizadores (somatório dos utilizadores de Android e IOS)	60

Tabela 6 - Esquema do teste de performance

Com este panorama fizemos um teste no qual enviou-se três mensagens ao mesmo tempo para cada tipo de sala. Visto que temos um servidor que utiliza a linguagem de programação PHP tivemos que utilizar os *thread's* para permitir executar várias operações ao mesmo tempo. Aqui foi necessário instalar um módulo PHP Extension Community Library (PECL - repositório de extensões do PHP) chamado *pthreads*, pois o servidor não tinha esta opção ativa.

Executamos este teste três vezes e no final fizemos uma média do tempo de execução que esta operação demorava. Precisamos de efetuar três vezes esta experiência pois os tempos destas foram um pouco diferentes em termos de milissegundos. Por exemplo: escolhemos uma sala de pequena dimensão e enviamos três mensagens ao mesmo tempo dentro desta sala. O Servidor recebia esta informação procurava na base de dados, os *tokens* relativos ao Android e IOS; enviava estes *tokens* para o GCM e APNS, respetivamente, e estes por fim comunicam aos respetivos dispositivos.

A figura 22 apresenta os resultados obtidos durante o teste, onde verificamos que foram precisos 3,7455 segundos para enviar 3 mensagens para uma sala de pequena dimensão; 5,8623 segundos para uma sala de média dimensão e 8,7233 para uma sala de grande dimensão.

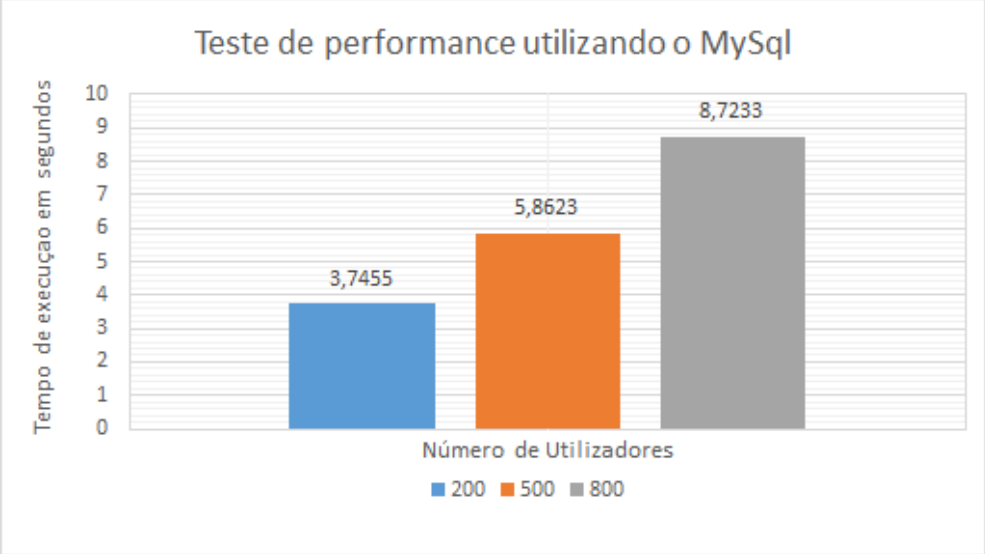

Figura 22 - Teste de performance utilizando o MySql

No sentido de ter um elemento de comparação, fizemos um segundo teste de performance utilizando dados guardados em memória, ou seja, os dados ficam armazenados em memória temporariamente em vez de ficarem numa base de dados.

Como não precisamos de guardar em definitivo os *tokens* dos telemóveis, esta ferramenta também se enquadra no NearUs. Visto que a forma de armazenar os dados em memória é diferente do Sistema de Gestão de Base de Dados escolhido (MySql), foi preciso criar um array multidimensional em PHP. Neste caso foi criado um array com a informação da sala e dentro deste array foram criados outros dois, um array com os *tokens* do Android e outro com os *tokens* do IOS.

A figura 23 ilustra a forma como estão organizados os dados em memória.

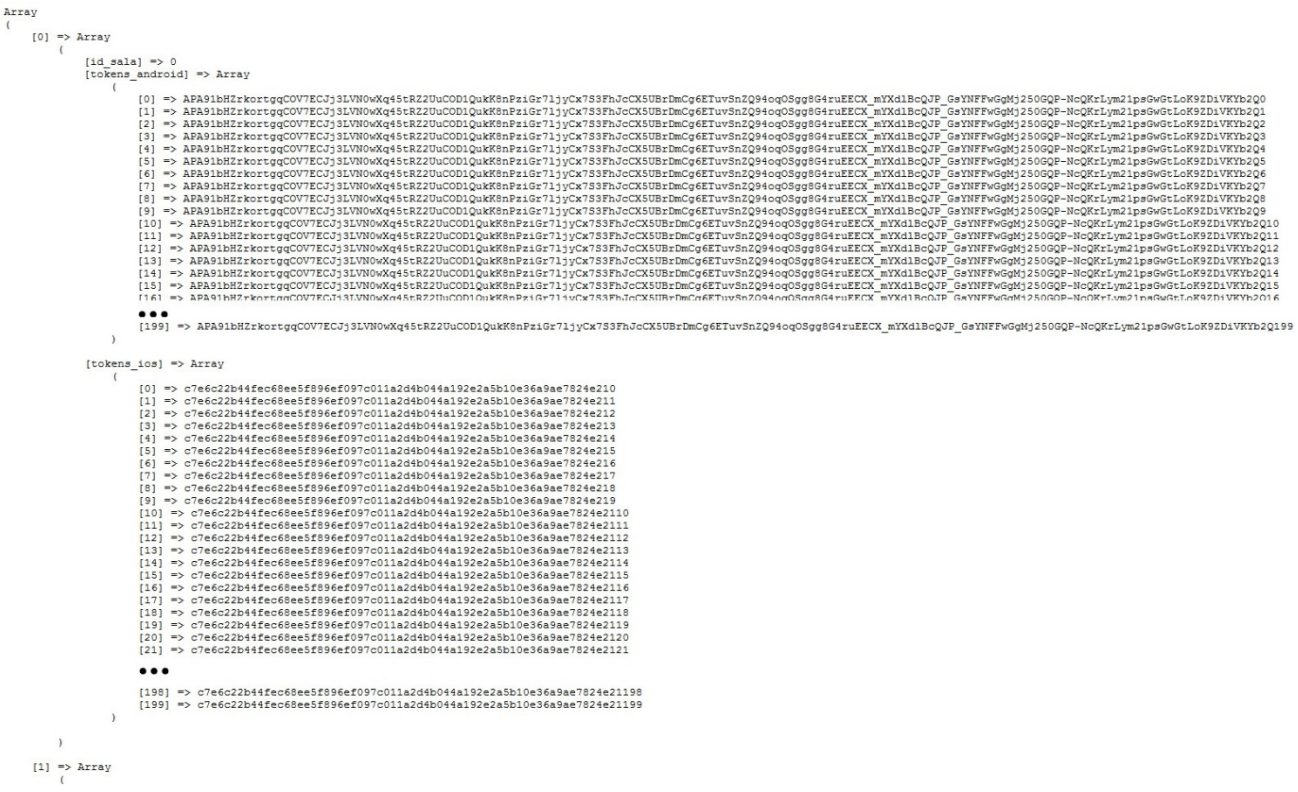

Figura 23 - Ilustração da organização dos dados em memória

À imagem do Mysql neste teste de performance também enviamos três mensagens em simultâneo dentro de uma sala de pequena, média e grande dimensão.

Executamos, também, este teste três vezes e no final apresentamos uma média do tempo de processamento da mesma.

A figura 24 apresenta os resultados obtidos durante o teste de performance de memória, onde verificamos que foram precisos 3,5697 segundos para enviar 3 mensagens para uma sala de pequena dimensão; 5,0903 segundos para uma sala de média dimensão e 7,2685 para uma sala de grande dimensão.

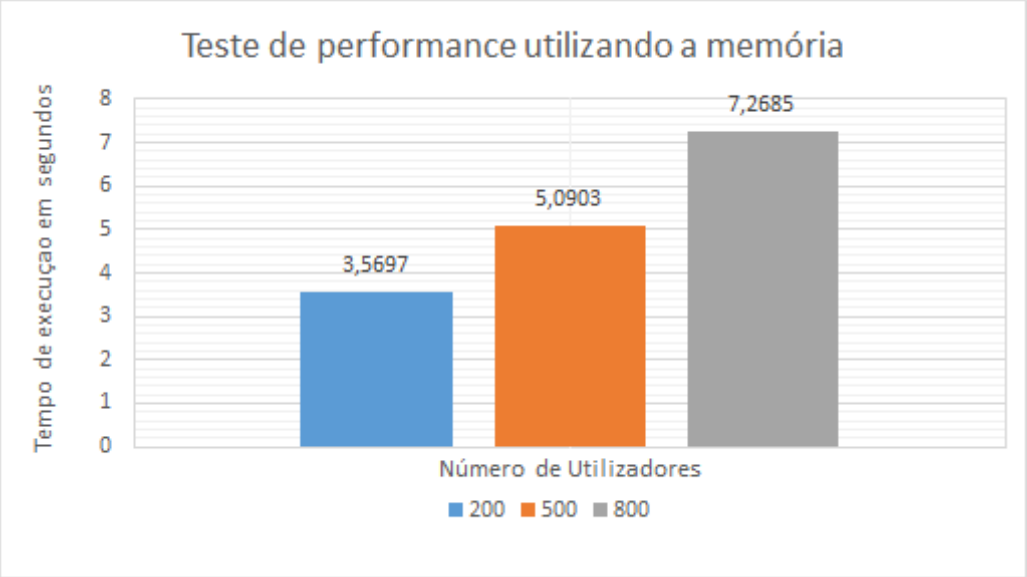

Figura 24 - Teste de performance utilizando a memória

Em síntese, onde existe uma maior diferença de tempo de execução é numa sala de grande dimensão, pois foi necessário usar 8,7233 segundos nesta operação empregando o Mysql e foram necessários 7,2685 segundos utilizando a memória. A diferença pode ser pequena, mas na implementação do NearUs torna-se muito significativa, pois é necessário que as mensagens cheguem o mais rápido possível ao utilizador.

Assim, considera-se que os tempos são aceitáveis nesta fase de desenvolvimento mas, no futuro, e num contexto de crescimento da utilização do NearUs, irá ser necessário

realizar mais testes e encontrar formas alternativas de armazenamento de dados para reduzir tempos.

3.5.2 – Testes realizados no NearUs em contexto real

Revistos que foram os conceitos do NearUs importa agora realizar testes para verificar se o mesmo está a funcionar de acordo com o requisitos definidos.

O primeiro grande teste que tivemos foi no "Open Days" ocorrido na Escola Superior de Tecnologia e Gestão de Viana do Castelo, organizado pela Licenciatura em Engenharia Informática, no dia 27 de Maio de 2014 onde foi criada uma sala com cerca de 40 utilizadores e que por parte do servidor como da Consola do *Google Play Developer Console* (onde estão alojados os registos relativos ao GCM) não houve registo de incidentes.

Em Julho de 2014 tivemos outro teste, em conjunto com a Câmara Municipal de Viana do Castelo, na aliança das artes onde tivemos uma excelente receção e onde foram partilhadas dezenas de mensagens em formato de imagem. Também aqui o *push* em ambos os sistemas operativos móveis funcionou corretamente, segundo os relatos dos participantes neste evento; contudo não conseguimos obter o número exato de participantes.

Em Outubro de 2014, a aplicação foi muito utilizada nos protestos em Hong Kong, pelo facto da China ter interdito o Instagram, como podemos verificar na notícia divulgada na reportagem do Dinheiro Vivo onde é referido que "A utilidade da aplicação foi demonstrada no mês passado, quando começaram os protestos em Hong Kong. A China bloqueou o Instagram no país e a NearUs teve um boom de downloads." [10]

Tivemos também outra prova em Outubro de 2014 com a saída da notícia do NearUs no tek sapo [11] onde foi criada uma sala com duzentos utilizadores e tanto quanto se sabe o *broadcast* funcionou corretamente.

Temos também salas criadas que vão desde os Estados Unidos da América, Arábia Saudita, Africa e Holanda, sendo que neste último país a aplicação tem sido muito utilizada em faculdades e em conferências de negócios.

Atualmente contamos com mais de mil *download's* e com mais de mil e setecentos gostos no facebook. Outro indicador relevante relaciona-se com os *download's* efetuados por país, podendo verificar-se que o NearUs já é utilizado numa diversidade de países, conforte apresentado na figura 25 onde se apresenta as instalações realizadas por país até ao dia 8 de Novembro de 2014.

Figura 25 - Lista de downloads por país, relativamente ao Android.

3.6 – Análise Critica e Trabalho Futuro

Finalizado este projeto de mestrado, (que foi importante para o desenvolvimento de competências e do aproveitamento do conhecimento adquirido ao longo do Mestrado); podemos assinalar futuros itens a desenvolver no NearUs, para que este não se torne obsoleto, tais como:

- Incluir ferramentas de monitorização para verificar o comportamento do servidor em contexto real;
- Otimizar o *broadcast* para salas com milhares de utilizadores;
- Implementar uma solução para *smartphones* com Windows;
- Utilizar Inteligência Artificial que nos permita conhecer métricas sobre a aplicação, como por exemplo: saber qual ou quais as palavras mais utilizadas numa sala; qual a altura do dia em que existe um maior número de utilizares, etc.;
- Partilhar ficheiros com vários formatos;
- Aumentar a capacidade de armazenamento, processamento e largura de banda do servidor.

Todas estas evoluções visam o aperfeiçoamento da aplicação NearUs para permitir ao utilizador uma agradável experiência e para que possa ter uma opinião favorável em relação ao NearUs.

Capítulo 4 – Conclusão e Trabalho Futuro

Como diria António Gedeão “Sempre que o homem sonha o mundo pula e avança”.

O mesmo acontece no mundo tecnológico, sendo necessário ousar duvidar e sonhar para que surjam as verdadeiras revoluções e inovações indo de encontro às necessidades das comunidades cada vez mais exigentes.

Neste sentido, assume principalmente importância o papel dos informáticos, especialmente os que desenvolvem *software*, uma vez que, num mundo cada vez mais exigente, a capacidade para acompanhar estas mudanças e para inovar é o que nos distingue e diferencia.

Tendo isto em mente surgiu a ideia de criarmos uma aplicação *smartphone* para que as pessoas pudessem trocar opiniões sobre um determinado evento em tempo real, num determinado espaço físico.

Para tal foi efetuado um levantamento do estado de arte no que diz respeito á troca de mensagens para dispositivos móveis.

Assim estudámos os diferentes tipos de mensagens (*pull e push*) e como estas funcionam no meio das comunicações entre a web e os *smartphones*.

Analisamos também o *cloud computing* (pois este está presente nos serviços estudados), a sua descrição, arquitetura, os seus conceitos derivados.

Através do levantamento do estado de arte foi possível identificar os serviços que dão resposta á troca de mensagens para dispositivos móveis e estudar a arquitetura dos mesmos e analisar a sua adequação para a arquitetura do NearUs, através de uma comparação das soluções disponíveis.

Nesta comparação analisamos os custos, a aprendizagem e a complexidade. Com efeito quase todos os serviços encontrados utilizam e com sucesso as ferramentas oferecidas pela *Google* e a *Apple*, razão pela qual, e após o estudo dos diversos serviços, também o fizemos, ou seja, optamos de igual forma pelas ferramentas oferecidas por estas empresas.

Mediante isto chegamos á conclusão que dado o presente estado do NearUs, a melhor solução encontrada foi utilizar os serviços "nativos" que cada empresa de sistema operativo móvel oferece, o *Google Cloud Messaging* para Android *e o Apple Push Notification Service* para IOS.

Após a implementação do *broadcast* no NearUs, foram realizados testes com dados fictícios (performance) e com dados reais (eventos) a fim de testar a eficácia da aplicação.

Esta aprendizagem permitiu-nos, nesta matéria, acompanhar o avanço "tecnológico" que tem acontecido nestes últimos anos.

E isto é o que um engenheiro de *software* deve fazer, acrescentar conhecimento novo a cada dia pois assim estará em condições de acompanhar o avanço do Mundo.

Pois cada avanço acontece com o surgimento da dúvida - neste caso, soluções de *broadcasting* em dispositivos móveis - e tentar resolvê-la da melhor forma possível.

Capítulo 5 – Bibliografia e referências WWW

[1] – Portal com a descrição do NearUs, "O que é o NearUs?", consultada em http://www.nearus.mobi

[2] – FLORES, Rogério – *Estratégias PULL e PUSH.* 26 De Maio de 2010. Consultado em http://fundamentosdelogstica.blogspot.pt/2010/05/5-estrategias-pull-e-push.html

[3] – DANIEL, Jorge – *A diferença entre sistemas push e pull, na cadeia de fornecimento.* 4 De Maio de 2012. Consultado em http://melhorar-negocios.blogspot.pt/2012/05/diferenca-entre-os-sistemas-push-e-pull.html

[4] – SILVESTRIN**,** Paulo – *Sistema para gerenciamento de Dispositivos Moveis baseado em Android.* Porto Alegre, Novembro de 2013. Consultado em http://www.lume.ufrgs.br/bitstream/handle/10183/86437/000910067.pdf?sequence=1

[5] – KUMAR, Vijay – **Mobile Database Systems**, New Jersey: John Wiley & Sons Inc. Publication, Julho 2006. 978-0-470-04828-3

[6] – ALECRIM, Emerson – *O que é cloud computing (computação nas nuvens)?.* 10 De Janeiro de 2013. Consultado em http://www.infowester.com/cloudcomputing.php

[7] – FREITAS, Lucas – *Google cloud Messaging (GCM) – Enviando mensagens da Nuvem para dispositivos Android*. 7 De Março de 2013. Consultado em http://blog.hachitecnologia.com.br/mobile/google-cloud-messaging-enviando-mensagens-da-nuvem-para-dispositivos-android

[8] – *Apple Push Notification Service (APNS).* Consultado em http://exmo.github.io/APNS.pdf

[9] – GRONER, Loiane – *Tutorial: Notificações em Push no iOS.* 8 De Julho de 2013. Consultado em http://www.loiane.com/2013/07/tutorial-notificacoes-push-no-ios/

[10] – Artigo do dinheiro vivo sobre o NearUS http://www.dinheirovivo.pt/buzz/tech/interior.aspx?content_id=4227480&page=1

[11] – Noticia tek sobre o NearUs http://tek.sapo.pt/tek_mobile/android/o_chat_do_irc_esta_de_volta_numa_app_que_nao_1419524.html

[12] – LEIDNER, Dorothy, et al. – **Tecnologia da Informação para Gestão: Transformando os Negócios na Economia Digital**, São Paulo: Artmed Editora S.A, 2010. 978-0-471-78712-9.

[13] – DUTSON, Phil, el al. – **The Android Developer's Cookbook: Building Applications with the Android SDK**, Estados Unidos: Addison-Wesley Professional, 2013.

978-0-321-89753-4.

[14] – LEE, Wei-Meng – **Beginning iOS 5: Application Development,** Indianápolis: Wrox, 2013. 978-1-1181-4425-1.

[15] – TAURION, Cezar – **Cloud Computing: computação em nuvem: transformando o mundo da tecnologia da informação**, Rio de Janeiro: Brasport, 2009.

978-85-7452-423-8.

[16] – WOODS, Ben – *Shuttleworth says Ubuntu's future is more exciting than space travel.* 6 de Março de 2015. Consultado em http://thenextweb.com/gadgets/2015/03/06/shuttleworth-says-ubuntus-future-is-more-exciting-than-space-travel/

[17] – Kapyla, Tuula, el al. - Towards an Accessible Web by Applying PUSH Technology. Fourth ERCIM Workshop on ``User Interfaces for All'' - Stockholm: Sweden 1998.

[18] – Na Li; Yanhui Du; Guangxuan Chen – "Survey of Cloud Messaging Push Notification Service," *Information Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on* , vol., no., pp.273,279, 7-8 Dezembro de 2013.