ESTUDO DA TECNOLOGIA PUSH NA PLATAFORMA ANDROID[1]

Luiz Ângelo Tognon de Medeiros[2]
José Antônio Oliveira de Figueiredo[3]

RESUMO

No contexto atual, vivemos em uma era tecnológica, em que priorizamos cada dia mais a agilidade e desempenho na busca de informações. Diante disto, podemos notar que a tecnologia móvel cresce em uma enorme escala, demonstrando assim que os usuários estão migrando para outras plataformas em busca de velocidade e precisão a qualquer hora e a qualquer local. Este artigo tem como objetivo principal descrever o que é a tecnologia *Push* e como ela pode melhorar a experiência entre usuário e aplicação. Para melhor demonstrar esta comunicação será ilustrado por meio de um protótipo o desenvolvimento de um aplicativo no sistema Android, interligado com uma aplicação Web desenvolvida em PHP e um serviço de envio de mensagens. O cenário deste protótipo é baseado em um sistema acadêmico, em que o professor acessa uma aplicação web e nele pode enviar informações a alunos específicos filtrando pela matéria em que o mesmo está matriculado.

Palavras-chave: Push. Android. Aplicativo. Google Cloud Messaging. Aplicação Web.

1 INTRODUÇÃO

O mundo está a cada dia mais interligado; no Brasil cerca de 68 milhões de pessoas navegam na internet com Smartphones segundo matéria da revista Exame (Caputo, 2015). Por este motivo, a cada dia que passa desenvolvedores se

[1] Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet, na cidade de Passo Fundo, em 2015.

[2] Graduando em Tecnologia em Sistemas para Internet pelo IFSUL campus Passo Fundo. E-mail: luizangelomedeiros@gmail.com.

[3] Orientador, professor do IFSUL. Email: jose.figueiredo@passofundo.ifsul.edu.br

preocupam mais em como gerar “conforto” ao usuário em seu Smartphone. Navegadores que usam sistemas de voz para buscar conteúdos, leitores biométricos para a segurança de acesso no dispositivo, câmeras que detectam sorriso para uma melhor foto são alguns exemplos disto.

Já os sistemas de notificações atuam de duas maneiras. O “*pull”* ou puxar em português é o sistema mais convencional e consiste em o usuário acessar o aplicativo e buscar novas atualizações. Já as notificações do tipo “*push*”, como o próprio nome diz, são “empurradas” para o dispositivo, tendo como origem um servidor externo. E este tipo de notificação que está se popularizando será o tema de estudo deste projeto de conclusão.

Com o decorrer do texto, será possível compreender sua arquitetura, funcionamento, características com base em diagramas e referências. Neste trabalho, devido a disponibilidade de tempo e de ferramentas, o estudo será conduzido apenas sobre a plataforma Android. As outras plataformas não serão consideradas.

2 PUSH E PULL

Atualmente, todo conteúdo Web está hospedado em servidores, que fornecem seus serviços aos clientes, estes clientes fazem solicitações de dados para os servidores por meio de requisições e os servidores respondem aos respectivos clientes com os dados solicitados.

Neste modelo de comunicação, toda nova notificação somente será vista se houver uma requisição do cliente. Isto torna o modelo ineficiente, pois em diversas situações é necessário atualizar o cliente no momento em que ocorreram modificações nos dados do servidor, sob pena da informação ficar depreciada, principalmente em redes sociais, bolsa de valores, informações climáticas entre outros. Outro problema do modelo, seria de que o usuário necessitaria enviar diversas requisições para o servidor, com a intenção de saber se existe alguma atualização da informação, aumentando assim o processamento do dispositivo do usuário e também gerando um desconforto, pois na maioria dos casos não há um tempo exato para a informação ser atualizada.

Uma solução é inverter os papéis, fazendo o servidor acordar o cliente, sempre que uma nova mensagem chegar (Lecheta, 2013). Este modelo é conhecido por *Push*. *Push* refere-se a um modelo de comunicação entre cliente-servidor onde é o servidor que inicia a comunicação. Neste modelo, é possível que um servidor envie dados para um cliente sem que ele tenha explicitamente solicitado.

Esta tecnologia de envio de notificações pode ser integrada com diversos sistemas operacionais como o *iOS*, utilizando o *Apple Push Notification Service*[4] (APNS); no *Windows Phone* com o *Microsoft Push Notification Service*[5] (MPNS) e no Android usando o *Google Cloud Messaging* (GCM).

2.1 SERVIÇO GOOGLE CLOUD MESSAGING (GCM)

Em meados de 2010, surgiu a versão 2.2 do Android (Froyo) e com ela veio um novo serviço conhecido como *Cloud to Device Messaging* (C2DM). Ele permitia receber mensagens do servidor conhecidas como notificações *Push* e possibilitava que aplicações servidoras enviem mensagens curtas para suas respectivas aplicações Android. O objetivo não é enviar grandes quantidades de dados, e sim notificar os clientes da existência desses dados para que eles possam fazer o download do mesmo e assim visualizá-los. Era necessário que o dispositivo Android estivesse com uma versão 2.2 ou superior, com uma conta do Google criada e com o *Google Play* instalado. O serviço de C2DM possuía algumas limitações, como limitar o tamanho das mensagens enviadas em 1KB, não garantindo a ordem de envio das mensagens e sua cota de envio diário era de no máximo 200 mil mensagens (Leal & De Vasconcelos, 2012)

Na Google I/O de 2013 - encontro anual dos desenvolvedores Google - a C2DM passou a ser chamada de *Google Cloud Messaging*. O serviço obteve uma taxa de

[4]https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html

[5] https://msdn.microsoft.com/en-us/library/windows/apps/ff402558%28v=vs.105%29.aspx

crescimento de 400% em 10 meses. No mesmo período, mais de 60% dos aplicativos mais populares do Google Play já faziam uso do GCM (Carmo, 2013).

Com a atualização da API, o serviço continuou da mesma maneira que o anterior, ajudando os desenvolvedores a enviarem dados a aplicações servidoras para App's Android. Com a atualização em relação ao C2DM, diversas tarefas que tinham que ser implementadas pelo desenvolvedor estão disponíveis em bibliotecas para download. Esta estratégia tem como objetivo facilitar o desenvolvimento e padronizar do código (Leal N. G., Push Notifications no Android, 2013). O GCM requer uma conta do Google no qual o aplicativo irá obter um número do projeto e sua chave da API. Quando a aplicação é iniciada ele faz uma operação de registro para receber mensagens de determinado servidor, gerando um ID de registro. Este ID é passado ao servidor que armazena a chave e quando houver uma nova notificação será enviado a mesma para o respectivo ID de registro (Leal N. G., Google Cloud Messaging, 2014)

Um dos grandes diferenciais da tecnologia *Push* é não necessitar estar rodando uma aplicação no dispositivo do cliente. O sistema dispara uma mensagem que é tratada por um *Broadcast Receiver* próprio do GCM. Por padrão o tamanho máximo da mensagem é de 4096 Bytes, ou seja 4KB, que fica armazenada no servidor do GCM por até 4 semanas enquanto o dispositivo estiver *offline*. Porém o serviço do Google não garante a entrega das mensagens em ordem e também não faz nenhum tipo de tratamento dos dados ou fornece uma interface gráfica (Leal N. G., Google Cloud Messaging, 2014).

2.2 ARQUITETURA DA TECNOLOGIA

Em um primeiro momento o usuário registra-se no aplicativo para receber notificações, assim que registrado, é retornado um identificador único do dispositivo, este fluxo pode ser notado na legenda 1 da Figura 1

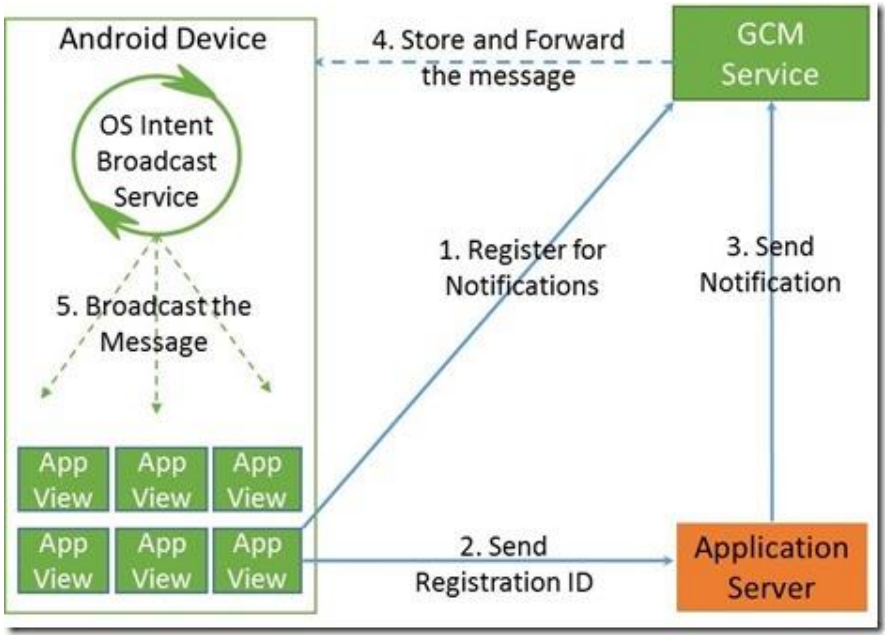
Este identificador ou também chamado de ID precisa ser enviado para o servidor da aplicação para que estes dados do dispositivo recentemente cadastrado seja salvo em um banco de dados e quando houver necessidade de envio da

notificação, possa ser escolhido qual dispositivo receberá estas informações. Este fluxo é demonstrado na Figura 1 pela legenda de número 2.

Após executar esses 2 passos a aplicação web já está preparado para enviar uma notificação. Como pode ser visto nas legendas 3 e 4 da Figura 1, a aplicação web encaminha a mensagem para o serviço que após recebido coloca a mensagem em sua fila de envio. Neste momento é feita uma cópia local do conteúdo da mensagem nos próprios bancos de dados do Serviço, para que caso o dispositivo destino tenha sido detectado com o estado de *offline*. Assim que o dispositivo é identificado como online, o serviço envia a mensagem para o destinatário.

No fluxo de número 5 da Figura 1, que ocorre no dispositivo Android, é demonstrado que após o envio da notificação ao dispositivo, o sistema Android envia uma *Intent broadcast* [6] esperando que o aplicativo para o qual a mensagem foi enviada capture esta mensagem. (Carmo, 2013).

Figura 1 – Fluxo de Notificação Push na plataforma Android



Fonte: (Hanuk, 2015)

[6] http://developer.android.com/reference/android/content/BroadcastReceiver.html

2.3 Aplicação Web

No lado do servidor, é necessário uma aplicação que faça o envio para os servidores do GCM. Esta aplicação pode ser desenvolvida em diversas linguagens, desde uma aplicação simples desenvolvida em um Smartphone, até um *Web Service* complexo em um servidor de alto desempenho.

Para enviar a mensagem para o dispositivo móvel, é necessário conhecer o código de ativação da aplicação que é fornecido para o serviço de envio quando o cliente registrar a aplicação, e sua respectiva chave criada na página do console do Google.

Na prática, para enviar uma mensagem de *Push* ao aparelho, é necessário fazer um POST no servidor do Google, enviando o corpo da mensagem, a chave do projeto, e o ID de registro do dispositivo. Também é utilizada uma classe para encapsular o código necessário para fazer essa requisição HTTP e o POST no servidor do Google. Após receber o POST, o servidor do Google utiliza um canal de comunicação que está aberto com o dispositivo Android para enviar a mensagem do *Push*.

3 ESTUDO DE CASO

Para demonstrar o funcionamento da tecnologia *Push*, foi desenvolvido um estudo de caso aplicando a arquitetura abordada anteriormente. O objetivo foi explorar o funcionamento da arquitetura usando uma aplicação desenvolvida na linguagem PHP em um projeto configurado na estrutura dos servidores do Google e uma aplicação implementada no sistema Android. Este estudo de caso tem como objetivo enviar notificações quando as informações na aplicação forem alteradas, priorizando demonstrar o fluxo envio e não os dados enviados.

O cenário no qual o estudo de caso foi desenvolvido se trata de um ambiente acadêmico, em que um “aluno” hipotético faz o download do aplicativo e se cadastra como usuário, logo após um “professor”, também hipotético, acessa uma página em que o mesmo pode escolher a disciplina e os seus respectivos alunos, selecionando-

os e os enviando uma mensagem de alerta, que será entregue a cada aluno previamente selecionado.

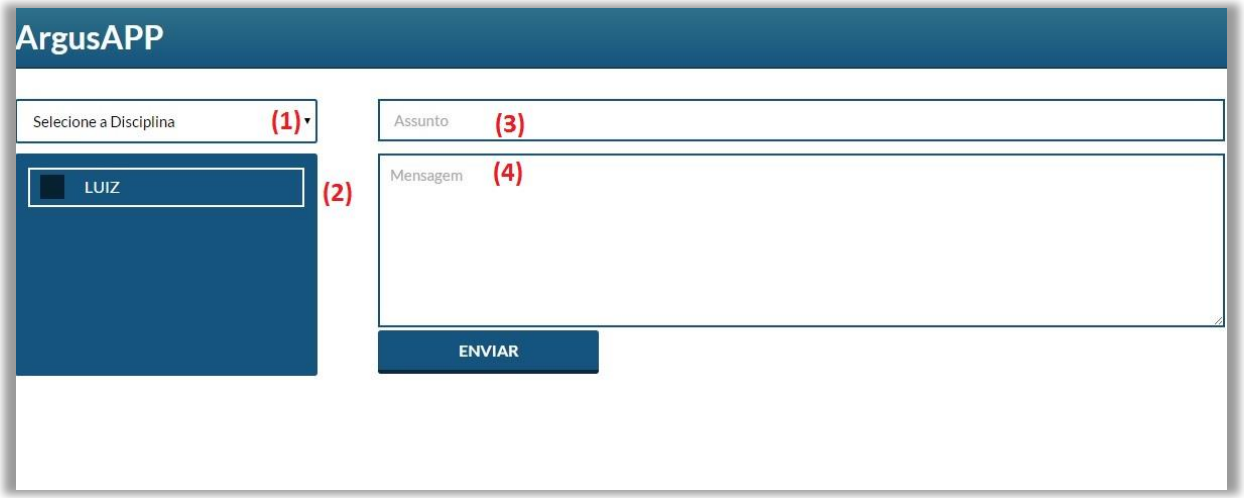
3.1 O PROJETO ARGUSAPP

Antes de iniciar o desenvolvimento do projeto, foi feito um estudo para escolher um nome que se adequasse ao tema proposto e também fosse inovador ao mesmo tempo. Assim chegamos na nomenclatura ArgusAPP[7].

3.1.1 Aplicação Web

Para enviar uma mensagem para seu alunos, o “professor” deve acessar a aplicação web via um navegador pelo link (http://argusapp.esy.es/index2.php). Após isto, o usuário “professor” irá visualizar uma website como pode ser notado na Figura 2.

Figura 2 – Tela da Aplicação Web



Fonte: Do Autor

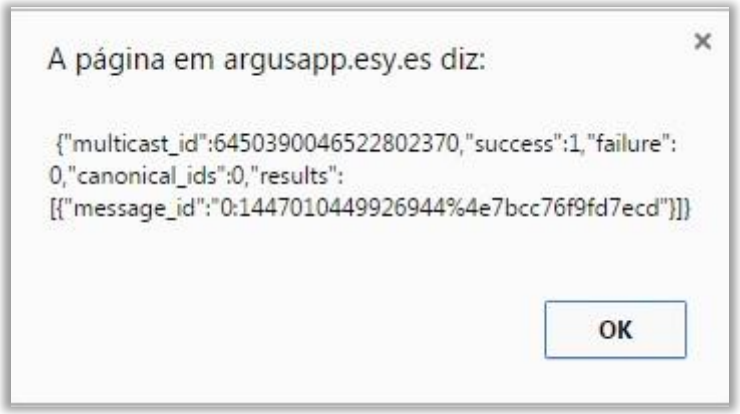
[7] Na mitologia grega, Argus ou Argos Panoptes era um gigante que tinha 100 olhos e “Panoptes” significa “aquele que tudo vê. https://en.wikipedia.org/wiki/Argus_Panoptes

Na tela da aplicação web, podemos notar 4 elementos primordiais para a execução da mensagem que estão numerados e destacados em vermelhos, são eles:

1. Disciplina: Campo onde o usuário seleciona qual disciplina deseja entrar em contato com os “alunos”;
2. Alunos: Caixa de seleção múltipla, onde são escolhidos quais alunos irão receberam as mensagens. O sistema pode enviar a mensagem para apenas um destinatário selecionado ou para muitos alunos simultaneamente.
3. Assunto: Campo de texto onde o “professor” descreve algumas palavras que serão exibidas junto com o nome na matéria no visor do celular.
4. Mensagem: Conteúdo principal que deseja informar ao “aluno”

Após preencher todos os campos e clicando em enviar, a aplicação web processa a mensagem, faz o seu envio para o serviço do GCM e armazena em um banco de dados MYSQL as informações do envio da mensagem. E por fim exibe um alerta na tela da aplicação web com as informações geradas. Como pode ser visto na Figura 3.

Figura 3 - Alerta de envio



Fonte: Do Autor

O conteúdo exibido pelo alerta de envio de mensagem da aplicação web é basicamente uma “árvore” em JSON, que contem em cada elemento, informações referente a mensagem gerada anteriormente. Por exemplo:

- multicast_id: São os identificadores dos dispositivos que receberam esta mensagem;
- success: Atributo booleano, em que “1“ representa sucesso no envio;
- failure: Atributo booleano, no qual “1“ representa falha no envio;
- canonical_ids: Identificador da última inscrição solicitada pelo aplicativo cliente;
- message_id: Identificador da mensagem em questão;

3.1.2 Aplicativo

O principal pilar deste projeto é o desenvolvimento do aplicativo no sistema operacional Android e nele são feitas as principais operações, como o cadastramento do usuário e a leitura de uma nova notificação enviada pela aplicação web.

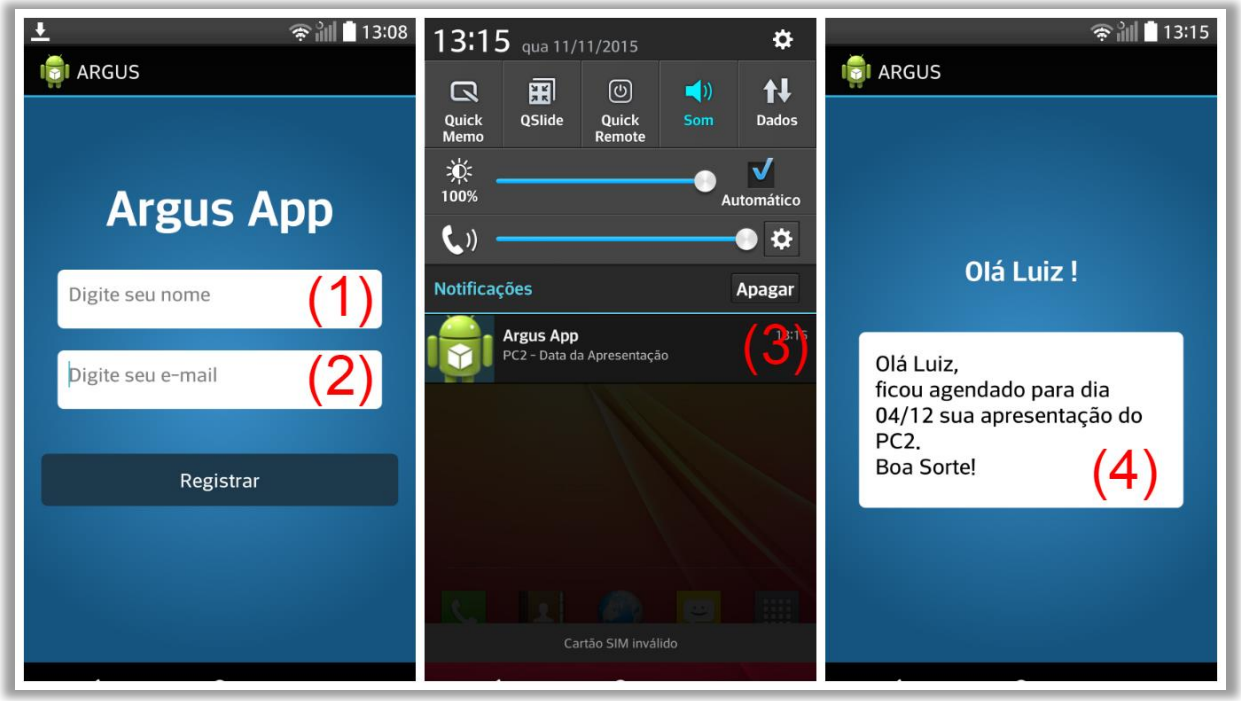
Para iniciar o desenvolvimento, foi escolhido a API 16 do Android - popularmente conhecida como Jelly Bean - como versão mínima para ser compatível com o aplicativo. Pois, segundo a matéria realizado pelo site Canal Tech, 60% dos dispositivos Android no mundo possuem a versão Jelly Bean instalada (CanalTech, 2014).

A activityPrincipal como seu próprio nome diz, é a classe principal do aplicativo, nesta classe, são inseridas as informações para o login. Nesta classe é feita a verificação se já ocorreu login anteriormente, se sim, ao abrir o aplicativo é direcionado diretamente para a tela Home.

- A classe Validacao é utilizada para verificar se os campos não estão vazios e o e-mail está com a formatação válida na tela da activityPrincipal;
- Na classe Constantes, são informadas dados úteis que são necessários para outras classes, como o id do projeto;
- A activityHome exibe as informações enviadas pela aplicação web na tela;

- GcmBroadcastReceiver é o responsável por receber o conteudo via GCM e “acordar” a aplicação;
- GcmNotificationIntentService recebe um aviso que existe uma nova notificação, processa os dados recebidos pela mesma para que sejam exibidos no topo do dispositivo, como pode ser observado na Figura 4 e legenda (3).

Figura 4 - Aplicativo Android



Fonte: Do Autor

Na Figura 4 pode ser observado em uma sequência de telas, o funcionamento do aplicativo em questão. Na qual, em sua primeira parte da imagem é exibido a tela inicial do aplicativo, logo após sua instalação, que tem como intuito digitar o nome (1) e o e-mail (2) para efetuar o registro do dispositivo na aplicação web.

Na tela localizada ao centro da imagem, podemos notar uma notificação enviada pela aplicação web e é recebido pelo dispositivo, identificada pelo nome Argus App e seu respectivo assunto (3).

Ao clicar na mesma, é aberto o aplicativo - como pode ser visto na última tela da Figura 4 – na qual é demonstrado a mensagem enviada na aplicação web (4).

CONSIDERAÇÕES FINAIS

Inicialmente, quando foi desenvolvida a ideia deste trabalho, não havia perspectivas do que era este tecnologia, como funcionava e quais aplicações faziam uso da mesma. Ao decorrer do trabalho, pode-se entender o quão importante é este tipo de comunicação para quem deseja desenvolver um aplicativo. Infelizmente, não havia muitas referências bibliográficas para este conteúdo, e algumas alternativas usadas para construir este projeto foram pesquisar em sites de outros países e vídeo aulas em línguas estrangeiras.

Com o desenvolvimento deste trabalho, pode ser observado que sites de grande relevância já possuem uma página que visa auxiliar o usuário a configurar o *Push* em seu dispositivo. Alguns exemplos disso são os sites *Facebook*[8], *WhatsApp*[9] e *LinkedIn*[10].

Como resultado final, pode-se demonstrar as principais diferenças entre as comuniçõs Push e Pull e assim concluir que a comunicação Push é essencial para novos desenvolvimentos de aplicações mobile, pois eleva a sensação de comodidade do usuário e também facilita a verificação de atualizações dos seus conteúdos desejados. Como trabalhos futuros, deseja-se concluir um cadastro mais completo de informações referentes ao meio acadêmico para o aplicativo, e após desenvolver o mesmo para as plataformas *iOS* e *Windows Phone*. Também deseja-se fazer um estudo sobre esta tecnologia implantada em *Browsers*, como o *Google Chrome*. Sendo assim, possuir uma ferramenta completa para o estudante que irá estar sempre atualizado e o professor que conseguirá fazer um elo mais forte com seu aluno.

[8] https://www.facebook.com/help/103859036372845
[9] https://www.whatsapp.com/faq/pt_br/iphone/20950116
[10] https://ajuda.linkedin.com/app/answers/detail/a_id/31371/~/notifica%C3%A7%C3%B5es-push-em-dispositivos-m%C3%B3veis---resumo

ABSTRACT

In the current context, we live in a technological age in which we prioritize each day more agility and performance in the search for information. Given this, we can see that mobile technology grows in a huge range, thus demonstrating that users are migrating to other platforms in search of speed and accuracy at any place and at any time. Describe what is *Push* technology, how it can improve the experience between user and application and how it can be done. It is also demonstrated through a prototype, the development of an application on Android system interconnected with a Web server in PHP and a messaging service. The setting of this prototype is based on an academic system, in which the teacher accesses a web server and it can send information to specific students by filtering material on which it is registered.

Keywords: Push. Android. App. Google Cloud Messaging. Web server

REFERÊNCIAS

CanalTech. (2014). Acesso em 22 de Outubro de 2015, disponível em http://canaltech.com.br/noticia/android/Jelly-Bean-e-usado-em-60-dos-dispositivos-Android-revela-Google/

Caputo, V. (2015). Acesso em 10 de Outubro de 2015, disponível em http://exame.abril.com.br/tecnologia/noticias/68-milhoes-navegam-na-internet-com-smartphone-no-brasil

Carmo, H. V. (2013). Google Cloud Messaging: Introdução. *MobileMagazine*. Acesso em 18 de Março de 2015, disponível em Devmedia: http://www.devmedia.com.br/google-cloud-messaging-introducao/29776

Hanuk. (2015). *Introducing Windows 8 for Android Developers - Part 2*. Acesso em 2015, disponível em Microsoft Developer Network: http://blogs.msdn.com/b/hanuk/archive/2013/04/18/introducing-windows-8-for-android-developers-part-2.aspx

Leal, N. G. (2013). Push Notifications no Android. (E. Spínola, Ed.) *JavaMagazine*(113). Acesso em 31 de Março de 2015, disponível em Devmedia: http://www.devmedia.com.br/push-notifications-no-android-revista-java-magazine-113/27354

Leal, N. G. (2014). *Google Cloud Messaging*. Acesso em 16 de Maio de 2015, disponível em Debug is on the table: http://www.nglauber.com.br/2014/02/google-cloud-messaging.html

Leal, N. G., & De Vasconcelos, F. A. (2012). Mensagens da nuvem para o Android(Cloud to Device Messaging API - C2DM). (E. Spínola, Ed.) *JavaMagazine*(107). Acesso em 16 de Maio de 2015, disponível em Devmedia: http://www.devmedia.com.br/mensagens-da-nuvem-para-o-android-cloud-to-device-messaging-api-c2dm-revista-java-magazine-107/25650

Lecheta, R. R. (2013). GCM - Google Cloud Messaging. Em R. R. Lecheta, *Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK* (p. 824). São Paulo: Novatec Editora. Acesso em Maio de 2015