05 Cloud Platform-as-a-Service



PEDRO NEVES DAVID CUNHA PEDRO SOUSA

PALAVRAS CHAVE
Cloud Computing, Service Broker, Platform-as-a-Service, APIs, Interoperabilidade

O *Cloud Computing* tem emergido como sendo um novo paradigma para entrega de serviços através da Internet. Neste mercado em expansão, o serviço de PaaS (*Platform-as-a-Service*) tem sido objeto de grande interesse por parte das mais variadas organizações permitindo o fácil *deployment* de aplicações sem necessidade de uma infra-estrutura dedicada, instalação de dependências ou configuração de servidores. No entanto, cada fornecedor de soluções PaaS acaba por gerar um *lock-in* do utilizador às suas características proprietárias, tecnologias ou APIs (*Application Programming Interfaces*). Além disso, dando como garantida a conectividade até aos clientes, a rede de operadores como seja o caso da PT (Portugal Telecom) acaba por servir apenas de *dumb-pipe* entre o fornecedor e os seus clientes.

Resumidamente, os principais objetivos deste artigo são apresentar os resultados do projeto CSB (*Cloud Service Broker*), realizado em colaboração com o Instituto de Telecomunicações de Aveiro, que tem como intuito resolver as questões de interoperabilidade entre fornecedores de PaaS; e igualmente apresentar o trabalho desenvolvido no projecto FP7 Cloud4SOA no qual a PTIN fez parte do consórcio.

1. INTRODUÇÃO

Além de um *buzzword* como o próprio termo *web* o é, *Cloud Computing* é uma evolução de vários paradigmas tecnológicos das últimas décadas transformando o sonho do *computing-as-a-utility* numa realidade. O potencial de tal modelo preconiza um grande impacto na indústria das TIs onde os recursos computacionais e o *software* são entregues aos utilizadores finais através de um paradigma *pay-per-use* [1].

Neste mercado em expansão, o serviço de PaaS tem sido objecto de grande interesse por parte das mais variadas organizações permitindo o fácil *deployment* de aplicações sem necessidade de infra-estruturas dedicadas, instalação de dependências ou configuração de servidores. Nos últimos anos, sensivelmente desde de 2009, surgiram diversos fornecedores e *startups* de soluções PaaS que competem entre si arduamente focando-se nos aspetos inovadores que os possam diferenciar face ao utilizador. Portanto, cada fornecedor tem intrinsecamente associadas diferentes linguagens de programação, *frameworks*, base de dados, taxonomias, modelos de negócio, ferramentas de desenvolvimento e APIs (*Application Programming Interfaces*) para interação por parte dos seus clientes. Por um lado, esta mescla de fornecedores favorece o processo de seleção por parte do utilizador. Mas por outro, surge a possibilidade de existir um *lock-in* do cliente a um fornecedor específico e às suas características proprietárias.

A Portugal Telecom Inovação (PTIN) pretende fornecer os alicerces para a entrada no mercado de *cloud* de uma plataforma mediadora entre os utilizadores e os fornecedores de PaaS de forma a posicionar-se firmemente na cadeia de valor deste mercado. Assim sendo, a PTIN teria a capacidade de oferecer aos clientes a possibilidade de operar sobre várias plataformas de forma centralizada e abstraindo os seus utilizadores das diferenças intrínsecas na interação com cada fornecedor suportado.

2. *PLATFORM-AS-A-SERVICE*

Até este momento, o IaaS (*Infrastructure-as-a-Service*) continua a ser o modelo de serviço *cloud* mais utilizado e com mais sucesso na área. Todavia o PaaS, sendo orientado ao desenvolvimento de aplicações, possui o potencial para abstrair as organizações de todos os processos de manutenção e configuração de servidores, bem como de instalação de dependências [2]. Um fornecedor de PaaS oferece um ambiente integrado simples e intuitivo para desenvolver, testar, monitorizar e hospedar aplicações *web* e respetivas bases de dados. Apesar do mercado de PaaS ainda se encontrar numa fase precoce, a Forrester[1] estimou que em 2016 o volume gerado devido ao mercado de PaaS poderia atingir os 15,2 mil milhões de dólares [3]. No seio de uma organização, o impacto destes ambientes, tanto a nível técnico e económico, é enorme. Os vários profissionais das TIs, nomeadamente programadores, administradores de sistemas e até gestores empresariais, são abrangidos por esta revolução. Em poucos dias um simples protótipo de um serviço pode ser lançado para produção sem existir um grande investimento inicial por parte da organização detentora. Subitamente, a inovação cessa de estar só no horizonte de quem possui avultados recursos económicos para transformá-la num produto comercializável.

No mercado surgem diversos fornecedores de PaaS cada um oferecendo um serviço diferenciado

[1] http://www.forrester.com/home/

para o utilizador. Além dos titãs Amazon, Google e Windows, emergem ofertas especializadas em linguagens de programação específicas, caso do CloudBees, e igualmente soluções *open-source*, nomeadamente o OpenShift da Red Hat e o Cloud Foundry da VMware. O Heroku da Salesforce.com surge como um dos principais fornecedores de PaaS especializado em linguagens *open-source* tais como Ruby e Python, e suportando Git como ferramenta de controlo de revisão. Dessa forma será feita uma síntese desses fornecedores.

2.1 CLOUDBEES

O CloudBees [4] foi fundado em 2010 e é um PaaS inteiramente direcionado ao desenvolvimento de aplicações Java. Portanto, suporta qualquer linguagem de programação e *framework* que executem sobre uma JVM (Java Virtual Machine), nomeadamente Java, JRuby, Scala, Play, Grails, Spring, etc. O CloudBees fornece dois serviços, sendo um deles orientado ao desenvolvimento e teste de aplicações, e outro orientado apenas ao *deployment* e execução das mesmas. Através do *DEV@Cloud*, o utilizador tem acesso a ferramentas como Jenkins, Maven, Sonar e repositórios Git e SVN de modo a controlar todo o ciclo de vida da aplicação. Por sua vez através do *RUN@Cloud*, o cliente poderá efetuar rapidamente o *deployment* de uma aplicação, já criada previamente, adquirindo acesso a todas as funcionalidades de gestão, monitorização, *logging* e escalabilidade.

2.2 CLOUD FOUNDRY

O Cloud Foundry [5] é um PaaS que possui uma abordagem dissemelhante em comparação a quase todas as restantes plataformas do mercado devido a ser uma plataforma completamente *open-source* com licença Apache v2. Lançado em 2011 pela VMware, o Cloud Foundry prima por suportar diversos ambientes de execução (e.g. Java, Scala, Ruby, Node.js, etc.), *frameworks* (e.g. Spring, Sinatra, Rails etc.) e base de dados (e.g. MySQL, MongoDB, Redis, PostgreSQL), sem estar fixo a uma única infra-estrutura. O utilizador tem assim a oportunidade de alterar o código fonte do próprio PaaS e assentá-lo sobre qualquer serviço de infra-estrutura ao seu dispor, seja público ou privado. Existem diversas soluções baseadas no Cloud Foundry como o AppFog, Stackato e a extensão .NET: IronFoundry.

2.3 HEROKU

O Heroku [6] surgiu em 2007 como sendo um PaaS orientado exclusivamente ao desenvolvimento de aplicações Ruby. Após ser adquirido pela Salesforce.com em 2010, o Heroku evoluiu suportando mais tecnologias e tornando-se um dos PaaS mais utilizados e conhecidos do mercado. De momento, é estimado que suporta mais de 1,5 milhões de aplicações de todo o tipo de linguagens de programação desde de Java, Scala, Python, Ruby, PHP ou Node.js., com base de dados PostgreSQL ou Redis. O Heroku possui um amplo catálogo de *addons* que permite aos utilizadores adicionarem diversos tipos de base de dados SQL e NoSQL, serviços de monitorização, *logging*, faturação, teste, etc. O Git tem-se tornado um de facto no mercado de PaaS sendo cada vez mais utilizado como ferramenta de *deployment* e controlo de revisão de código fonte em diversos PaaS, não sendo exceção no Heroku.

3. *CLOUD SERVICE BROKER*

Cada aplicação possui dependências específicas que podem ser tão distintas como a mesma ser desenvolvida em Java ou Python até necessitar de uma base de dados SQL ou NoSQL. Para um utilizador que pretenda portar as suas aplicações para um ambiente de *cloud*, ele precisará de examinar as diversas ofertas existentes no mercado, conhecer as tecnologias suportadas, estudar e testar os interfaces cliente, as APIs fornecidas, etc. Dessa forma surgiu a plataforma CSB (*Cloud Service Broker*), que foi desenvolvida no âmbito de um projeto do Instituto de Telecomunicações de Aveiro suportado pela PTIN, com o intuito de facilitar a descoberta e interacção com diversos fornecedores de PaaS através de um único interface.

A arquitectura definida inclui além do CSB, o *PaaSManager*, que será descrito mais à frente, e as interfaces cliente: *web*, CLI e Git. Através das interfaces *web* e CLI, o utilizador tem acesso às operações dos serviços de gestão e de informação suportados pelo *PaaSManager*. No momento da criação de uma aplicação é preenchido o formulário que constitui o perfil tecnológico da aplicação, nomeadamente, os *runtimes, frameworks* e *services* (base de dados). Desta forma, o CSB invoca o *PaaSManager* para devolver as tecnologias suportadas por cada plataforma do ecossistema, comparando com a informação inserida no perfil. Por fim, é devolvida uma lista ordenada com os fornecedores recomendados. Nos diversos fluxos de operações teve-se em atenção que do ponto de vista de utilizador a interacção com esta arquitectura não se verificasse mais complexa do que interagir directamente com um fornecedor específico.

Na Figura 1 é apresentada a arquitetura do *Cloud Service Broker* bem como a sua integração com os restantes módulos desenvolvidos.

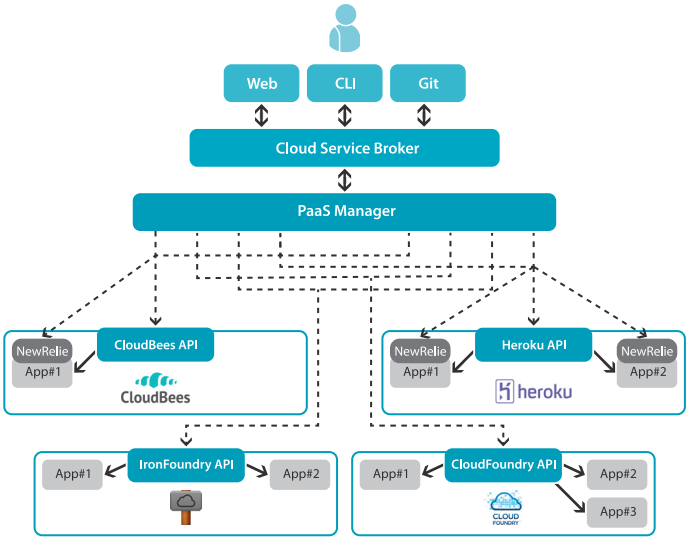

Figura 1. Arquitetura *Cloud Service Broker*

3.1 RECOMENDADOR *CLOUD SERVICE BROKER*

O *Cloud Service Broker* visa efetuar a recomendação baseada em regras definidas e analisadas pelo seu motor de regras. As suas características principais são dar assistência no processo de recomendação e fornecer todas as operações suportadas através de um único interface. Portanto, o utilizador tem assim acesso a todo o ciclo de vida de uma aplicação hospedada no PaaS recomendado pelo CSB. Além dos requisitos técnicos de uma aplicação, o algoritmo de recomendação poderá ser estendido suportando diversas exigências. São alguns exemplos a geração de notificações quando certos patamares nos valores de monitorização forem ultrapassados, de forma a respeitar os contratos definidos pelo próprio utilizador; ou então a recomendação por base nos modelos de negócio que se verifiquem mais adequados para o próprio detentor de uma aplicação.

Esta plataforma foi desenvolvida em Java, o algoritmo de recomendação em Prolog, e foi utilizada uma base de dados MySQL para registo de utilizadores e manutenção de estado.

3.2 PAASMANAGER

O *PaaSManager* é uma camada de abstração que visa unificar os processos de gestão e aquisição de informação de aplicações criadas através de diversos PaaS de modo a combater o *lock-in* existente no mercado. Inicialmente foram seleccionadas algumas plataformas que surgem no mercado com o intuito de implementar a solução que agrega as diferentes ofertas de forma transparente para o utilizador. Cada fornecedor de PaaS disponibiliza uma API. Através dessa interface são expostas diversas funcionalidades que permitem criar, gerir e obter informação sobre aplicações e bases de dados. Porém, do leque de métodos disponibilizados, foi fundamental eleger as funcionalidades idênticas ou semelhantes que são partilhadas entre as várias APIs analisadas. Apenas dessa forma seria possível obter a interoperabilidade necessária para o utilizador interagir de igual forma com diferentes PaaS que possuem diferentes APIs. Das plataformas investigadas foram selecionados para o ecossistema 4 fornecedores: CloudBees, Cloud Foundry, Iron Foundry e Heroku. Todos os fornecedores possuem diferentes APIs, ferramentas de monitorização e de *deployment*, em exceção do Cloud Foundry e do Iron Foundry que partilham uma implementação de API idêntica, porém suportando tecnologias discrepantes. Desta forma, o *PaaSManager* pode ser integrado com o recomendador *Cloud Service Broker* através de uma única interface que abstrai os vários fornecedores suportados.

No desenho da arquitetura do *PaaSManager* teve-se em atenção a modularidade preponderante que permite que todo o sistema continue em completo funcionamento mesmo que alguma API de um fornecedor ou uma API de monitorização não esteja a operar corretamente. Consequentemente, cada API foi implementada por diferentes módulos de software e gerida por entidades únicas orientadas aos grupos de serviços de gestão e de informação. Por fim, uma interface RESTFul expõe as várias operações especificadas que podem ser invocadas por um simples cliente HTTP.

A Figura 2 ilustra a arquitetura do *PaaSManager*, os diversos módulos que foram especificados e a sua integração com as diversas APIs dos fornecedores de PaaS.

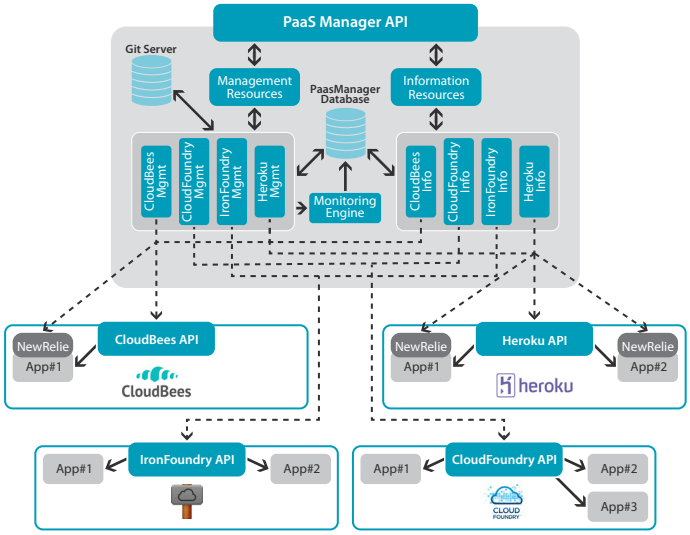

Figura 2. Arquitetura *PaaSManager*

3.2.1 Principais Operações

Nesta secção são apresentadas algumas das operações mais fundamentais que são suportadas pelo *PaaSManager*.

- ***Deployment* da Aplicação**

O *deployment* do código fonte foi unificado para que o utilizador não necessite de se preocupar com as diferentes ferramentas oferecidas pelos fornecedores. A Figura 3 ilustra o processo ocorrido durante a operação de *deployment*. O pedido recebido pela API é instantaneamente enviado ao módulo de gestão, *Management Resources*, que realiza uma busca na base de dados central com o objectivo de adquirir a identificação da plataforma onde se encontra criada a aplicação (passo prévio). Através do resultado obtido é invocado o *adapter* do PaaS pretendido, que por sua vez, executa os comandos *git-add* e *git-commit* no repositório da aplicação que se encontra no servidor Git central. Após este processo, é efetuado o *deployment* na plataforma conforme o paradigma associado. No caso do CloudBees, é feita uma pesquisa no repositório até ser encontrado o *web archive* da aplicação (.war). Para o Cloud Foundry e Iron Foundry é realizado o mesmo processo para aplicações baseadas em Java, porém, para as restantes é enviado todo o código fonte. Para o Heroku é executado um comando *git-push* para o repositório remoto criado exclusivamente para a aplicação na plataforma Heroku.

Numa situação de sucesso, é iniciado o motor de monitorização com o objetivo de recolher estatísticas em tempo-real sobre a aplicação, armazenando-as na base de dados central. Para finalizar o processo, é devolvida uma resposta em XML ou JSON ao utilizador. A estratégia de funcionamento adotada para o motor de monitorização será posteriormente analisada com mais detalhe.

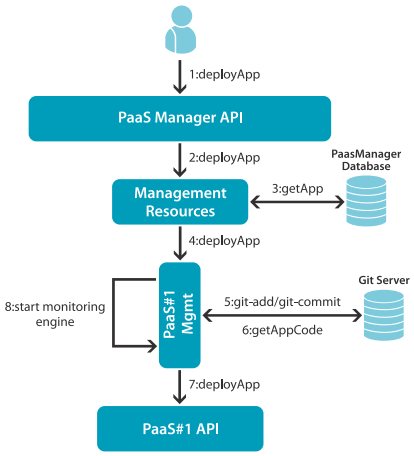

Figura 3. *PaaSManager - Deploy App*

- **Adquirir Estado da Aplicação**

O acesso à informação sobre o estado físico de uma aplicação a executar em uma plataforma é de facto fundamental para o utilizador. A Figura 4 ilustra os diversos processos que ocorrem durante esta operação. De forma a dar uma simples e intuitiva visão sobre o estado de qualquer aplicação, foram especificados 4 estados: *running, stopped, crashed* e *unknown*. No processo de obtenção de estado, o pedido efetuado à API do *PaaSManager* é instantaneamente enviado ao módulo de informação, Informação *Resources*, que direciona a mensagem

para o *adapter* do PaaS pretendido. Por sua vez, o *adapter* invoca o método da API da plataforma associada. Conforme a informação de estado devolvida, a mesma é mapeada para um dos 4 estados definidos com o intuito de unificar a resposta XML ou JSON para o utilizador.

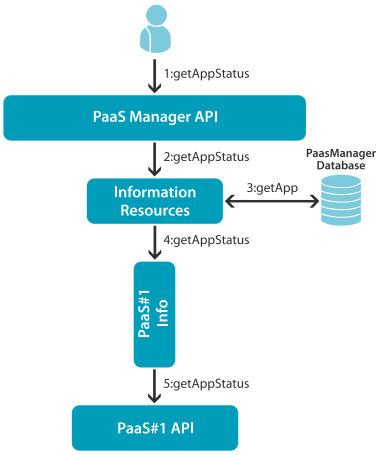

Figura 4. *PaaSManager* - Adquirir Estado da Aplicação

- **Monitorização**

A monitorização em tempo-real de aplicações permite fornecer ao utilizador a informação necessária sobre o comportamento das mesmas nos diferentes PaaS suportados. Até ao momento não existe um consenso nem nenhum modelo normalizado de monitorização partilhado entre os maiores fornecedores. Nos últimos anos, os estudos efetuados nesta área tentam definir *frameworks* de monitorização como também um modelo de métricas que se verifique apropriado para uma gestão eficiente de aplicações na *cloud* [7], [8]. Uma normalização do modelo de monitorização faria com que os SLAs definidos pelos fornecedores fossem comparáveis e o utilizador pudesse testar a mesma aplicação em 2 ambientes heterogéneos com o objetivo de obter informação sobre o comportamento dos ambientes disponibilizados. No entanto, ao nível do serviço de PaaS, cada fornecedor possui diferentes métricas e formas de monitorizar as aplicações sendo que grande parte da informação relacionada com os recursos de infraestrutura é abstraída para o utilizador. Como foi analisado, o CloudBees e o Heroku possuem parceria com APMs (*Application Performance Monitor*), nomeadamente, o New Relic que permite a monitorização de aplicações através da instalação de agentes nas instâncias onde as próprias executam. Por outro lado, o Cloud Foundry e Iron Foundry devolvem outro tipo de estatísticas diretamente através da API nativa.

Neste contexto, foi especificado um módulo na arquitetura do PaaS Manager, denominado por *Monitoring Engine* (ver na Figura 2), que efetua a recolha em tempo-real das métricas expostas pelas diferentes plataformas do ecossistema. A informação obtida é depois armazenada na base de dados central para ser devolvida ao utilizador através de pedidos à API do *PaaSManager*. O processo é definido por uma recolha síncrona efetuada todos os minutos à API do New Relic ou à API nativa, conforme o PaaS onde a aplicação se encontra hospedada. A Figura 5 ilustra os diversos passos que são executados durante o processo de monitorização.

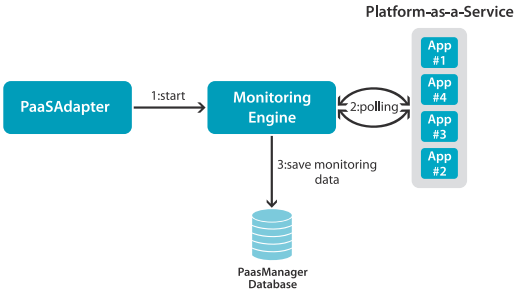

Figura 5. *PaaSManager - Monitoring Engine*

Toda a implementação do *PaaSManager* foi realizada em Java recorrendo à plataforma JavaEE6 (Java Enterprise Edition 6) devido à ótima integração com EJBs (Enterprise Java Beans), *web services* e camadas de persistência. O servidor aplicacional JBoss 7.1.1 foi o servidor selecionado para suportar o *PaaSManager* no ambiente de desenvolvimento e teste.

4. CLOUD4SOA

Context-as-a-Service traz a oportunidade de usar a informação gerada pelos clientes num ambiente móvel para selecionar e entregar conteúdo mais adequado para os mesmos (por exemplo, vídeos, fotos, etc.). Estes meta-dados podem caracterizar qualquer situação do cliente, tal como a sua posição, género, como simultaneamente interesses de musicais ou sociais. No âmbito do projeto FP7 C-CAST [9] que finalizou em 2010, a PTIN desenvolveu uma *framework* composta de diversos serviços que reagem a informação de contexto de um utilizador. Neste contexto, a proliferação de tais serviços móveis pode, em alguns casos, gerar uma sobrecarga na infra-estrutura subjacente se a mesma não for escalável.

A PTIN no projeto Cloud4SOA teve como grande objetivo migrar os vários serviços da *framework* de contexto para um ambiente *cloud*, nomeadamente, PaaS. Essa migração envolveu um extenso estudo sobre as alterações a realizar nas aplicações. A

arquitetura, apresentada na Figura 6, é composta por vários módulos que permitem recolher e processar informação relativa ao terminal móvel do cliente. Como componente central e fundamental da *framework* surge um servidor XMPP denominado por Context Broker. Este servidor tem um papel ativo nos processos efetuados pelos 3 serviços de contexto: *Context Enabler, Group Manager Enabler* e *Content Selection Enabler*.

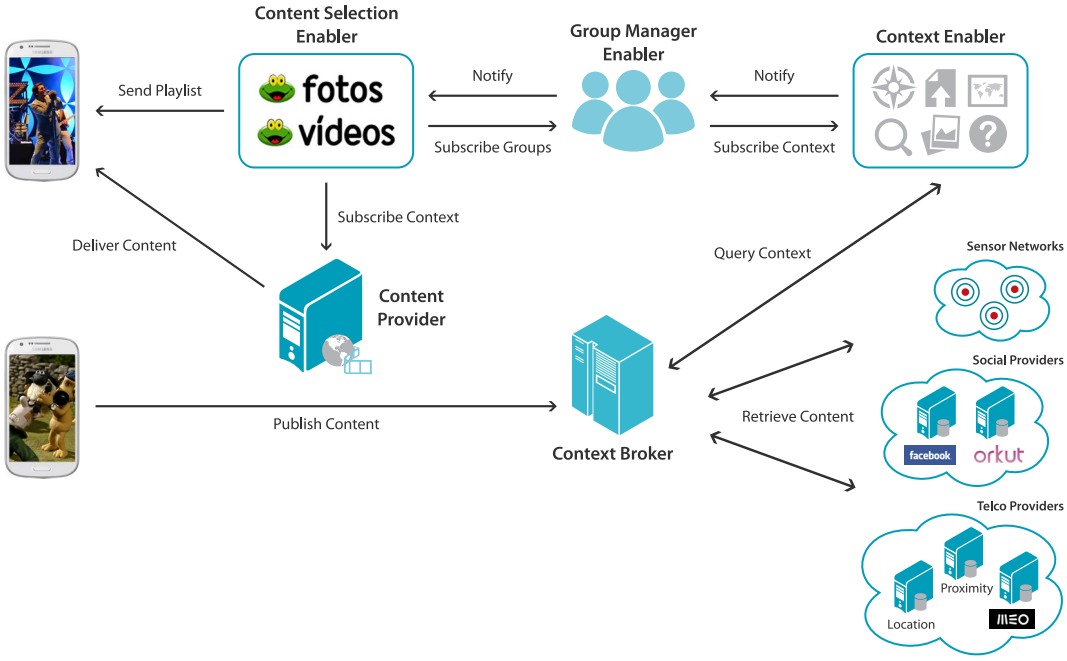

Figura 6. Cloud4SOA - Arquitetura *Framework* Contexto

Dos vários componentes foram migrados os seguintes 3 serviços:

- ***Context Enabler***: Este serviço foi desenhado durante o projeto Cloud4SOA. É um *web service* SOAP com o propósito de servir de interface para o servidor XMPP e permitir a reutilização segundo os conceitos SOA.
- ***Group Manager Enabler:*** Este serviço é responsável por estabelecer grupos de utilizadores baseado em condições pré-configuradas.
- ***Content Selection Enabler***: Este serviço tem por base um motor de regras que selecciona conteúdo (fotos, vídeos, etc.) mais adequados para os grupos de utilizadores.

Todas as aplicações foram preparadas para executarem sobre o servidor Apache Tomcat devido a ser um dos servidores Java mais suportados pelos fornecedores de PaaS existentes no mercado. Para o processo de migração foi utilizado o sistema desenvolvido pelo restante consórcio do Cloud4SOA tendo sido realizados alguns testes que provaram uma boa escalabilidade dos serviços como igualmente custos mais reduzidos para instanciar e manter a mesma infra-estrutura que existia *on-premise*.

5. ENSAIOS E AVALIAÇÃO

Nesta seção são descritos alguns dos ensaios e avaliações realizadas no projeto *Cloud Service Broker*.

5.1 INTERFACE *WEB*

O portal *web* do *Cloud Service Broker* foi desenvolvido em *Ruby on Rails* tendo sido utilizado *HTML, CSS, JQuery* e a *framework Twitter Bootstrap* para o *frontend*.

Na Figura 7 é possível observar a página inicial do portal onde o cliente pode iniciar o processo de recomendação em *New App*.

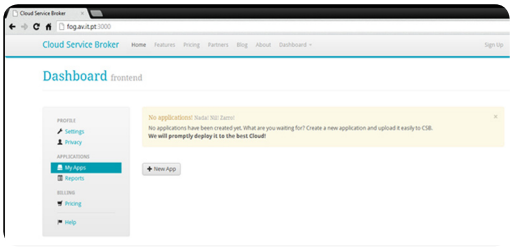

Figura 7. *Cloud Service Broker - Home Page*

De seguida, na Figura 8, é apresentado ao cliente um formulário onde o mesmo pode preencher as características técnicas da aplicação que deseja migrar para a *Cloud*. Como resultado é apresentada uma lista de fornecedores mais adequados.

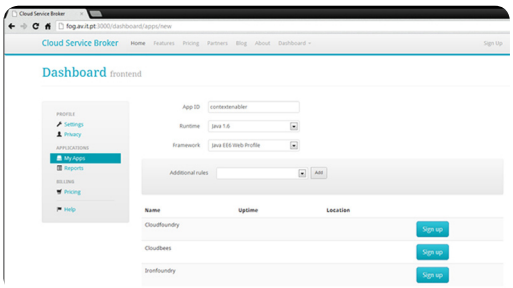

Figura 8. *Cloud Service Broker* - Recomendação

Após seleccionar o fornecedor mais adequado, a aplicação foi criada com sucesso. O processo prossegue com o *deployment* do código fonte através da ferramenta Git e para o respectivo repositório remoto.

Desta forma, a aplicação encontra-se *online* tal como demonstrado na Figura 9.

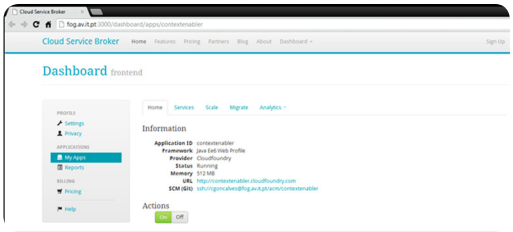

Figura 9. *Cloud Service Broker - App deployed*

Por fim, como se pode observar na Figura 10, o utilizador tem acesso às várias métricas recolhidas no fornecedor.

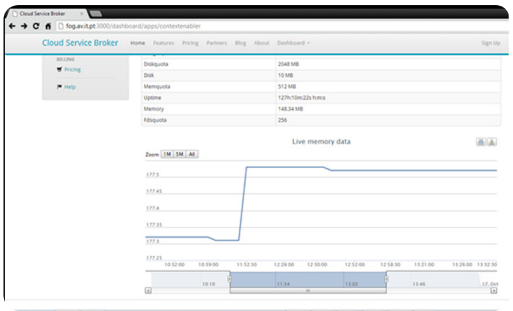

Figura 10. *Cloud Service Broker* - Monitorização

5.2 *PAASMANAGER*

Para testar a solução *PaaSManager* foram selecionados alguns dos métodos fundamentais de forma a serem avaliados diferentes aspetos e depurado onde seria possível proceder a otimizações. Através da ferramenta Apache JMeter foram realizados alguns testes de carga simulando o acesso de diversos clientes em simultâneo. Os valores obtidos serviram de referência para compreender o comportamento que o sistema detém em operações essenciais. Por um lado, de forma a se observar o *overhead* que é acrescido pelo *PaaSManager* em cada pedido efetuado, e por outro, de forma a se analisar a escalabilidade da solução com diferentes números de utilizadores. Para cada uma das operações selecionadas foram realizados ensaios com 10 e 30 utilizadores em simultâneo, obtendo-se a média e o desvio padrão associado a cada operação efetuada nas diferentes plataformas que constituem o ecossistema. Esse número de utilizadores foi escolhido devido ao *PaasManager* utilizar apenas uma conta para cada PaaS. Assim sendo, um grande número de pedidos poderia se refletir num bloqueamento da conta ou num efeito de *throttling* por parte dos fornecedores. É necessário realçar que o objetivo desta análise não foi comparar diretamente a eficiência de cada fornecedor mas sim o desempenho da arquitetura desenvolvida. Como exemplo ilustrativo apresenta-se de seguida alguns resultados relacionados com a operação de aquisição de estado da aplicação.

- **Adquirir Estado da Aplicação**

Nesta operação foram obtidos, com um total de 30 utilizadores em simultâneo, os resultados apresentados na Figura 11. A série *PaaSManager*, que representa o tempo consumido apenas no processamento interno do pedido, apresenta valores aproximados de tempo de resposta entre os 60 e os 800 ms. Esta série apenas inclui o processo de aquisição da identificação da plataforma até ser invocado o respetivo *adapter* de PaaS. Na série *PaaSManager+PaaS API*, que abrange o tempo consumido em todo o processo da operação incluindo a solicitação à API da plataforma em questão, os diversos fornecedores registaram os valores entre os 1200 e 1600 ms. No caso dos *adapters* do Cloud Foundry e Iron Foundry existe uma verificação nos *logs* de cada instância onde a aplicação executa com o intuito de detetar erros de funcionamento, explicando assim a maior contribuição do *PaaSManager* nos tempos de resposta obtidos.

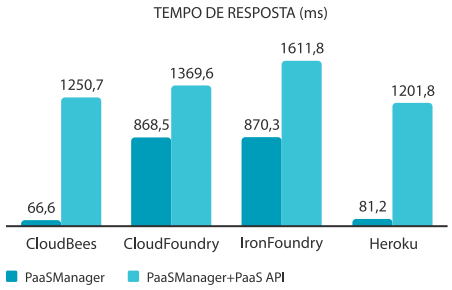

Figura 10. *PaaSManager* - Adquirir Estado App (30 Utilizadores)

6. CONCLUSÕES

O *Cloud Service Broker* e o Cloud4SOA foram projetos pioneiros na PTIN na área de PaaS. Os resultados trouxeram um considerável *know-how* sobre estes ambientes de *cloud*, originando também novas questões e desafios para futuros trabalhos.

No projeto Cloud4SOA foram migradas para um ambiente *cloud* algumas das aplicações que fazem parte de uma *framework* de contexto já existente. Para realizar essa migração com sucesso, algumas das aplicações foram modificadas sendo que os resultados obtidos mostraram as valências da utilização de serviços de *Cloud Computing*.

No projeto *Cloud Service Broker* foi desenvolvido um recomendador e uma camada de abstracção que agrega diversas soluções comerciais de PaaS. O desenho da solução envolveu inicialmente uma análise das APIs das diversas plataformas selecionadas bem como a definição das funcionalidades essenciais que deveriam ser suportadas. Os testes realizados revelaram que a arquitetura não introduziu um *overhead* significativo na maior parte das operações suportadas. Sendo assim, interagir com o *PaaSManager* não coloca mais complexidade nem origina um muito maior tempo de resposta em comparação a interagir diretamente com cada fornecedor.

Em suma, o protótipo desenvolvido no projeto *Cloud Service Broker* é neste momento um dos únicos nesta área de investigação de interoperabilidade e recomendação em ambientes PaaS que possui uma implementação estável e com resultados operacionais. Recentemente, a temática discutida ao longo deste trabalho tem vindo a receber uma grande atenção por parte da comunidade, esperando-se novas iniciativas e projetos que tencionem dar aos utilizadores a oportunidade de controlar várias plataformas de forma unificada.

REFERÊNCIAS

[1] M. Armbrust et al., "A View of Cloud Computing," Commun. ACM, vol. 53, pp. 50-58, 2010.
[2] D. Beimborn et al., "Platform as a Service," Business & Information Systems Engineering, vol. 3, pp. 381-384, 2011.
[3] S. Ried, "Platform-as-a-service market sizing," 2009.
[4] CloudBees. [Online]. Available: http://www.cloudbees.com/.
[5] CloudFoundry. [Online]. Available: http://www.cloudfoundry.com/.
[6] Heroku. [Online]. Available: https://www.heroku.com/.
[7] Q. Shao et al., "A Performance Guarantee Approach for Cloud Applications Based on Monitoring," em Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, Washington, DC, USA, 2011.
[8] T. a. E. V. C. a. M. M. a. B. I. Mastelic, "M4Cloud - Generic Application Level Monitoring for Resource-shared Cloud Environments," em Proceedings of the 2nd International Conference on Cloud Computing and Services Science, Oporto, Portugal, 2012.
[9] C-CAST. [Online]. Available: http://www.ict-ccast.eu/.
[10] Cloud4SOA. [Online]. Available: http://www.cloud4soa.eu/.

CVS DOS AUTORES

Pedro Neves, Doutorado e Mestre em Engenharia Electrónica, Telecomunicações e Informática pela Universidade de Aveiro em 2012 e 2006, respetivamente. Em 2003 tornou-se bolseiro de investigação do Instituto de Telecomunicações, onde trabalhou em projetos co-financiados pela Comissão Europeia na área de mobilidade e qualidade de serviço. Em Junho de 2006 iniciou atividade na PT Inovação na mesma área de trabalho. Desde 2010 que acumula também atividades de investigação na área de Cloud Computing. Participou em mais de 10 projetos de colaboração internacional, é co-autor de 6 livros internacionais e de mais de 30 artigos publicados em revistas e conferências internacionais.

David Cunha, M.Sc. em Engenharia de Redes e Serviços pela Universidade do Minho em 2012. Iniciou a sua actividade profissional na Portugal Telecom Inovação em 2011. Os seus interesses contemplam o open-source, desenvolvimento para a web, bem como as temáticas da *cloud*: Platform-as-a-Service e Software-as-a-Service.

Pedro Sousa, Doutorado e Mestre em Ciências da Computação pela Universidade do Minho em 2005 e 1997, respetivamente. Em 1996, juntou-se ao Grupo de Comunicações e Redes de Computadores do Departamento de Informática da Universidade do Minho, onde é Professor Assistente e executa suas atividades de investigação no Centro Algoritmi.