Migração de Aplicações Legacy para a Cloud

Pedro Miguel Lopes Nunes

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientador: Prof. José Manuel da Costa Alves Marques

Júri

Presidente: Prof. Alberto Manuel Rodrigues da Silva

Orientador: Prof. José Manuel da Costa Alves Marques

Vogal: Prof. Miguel Filipe Leitão Pardal

Dezembro 2019

Abstract

Cloud computing has brought an opportunity for businesses to modernize their legacy applications. Migrating legacy applications to the cloud and being able to successfully leverage its characteristics has become a major objective for several companies whose systems rely on old technologies and perform poorly when compared to what the cloud has to offer. This project provides a general state-of-the-art regarding this topic and proposes some alternative cloud-oriented architectures for a specific legacy application - EDOCLINK. It describes the process of migrating certain components to the cloud as well as a performance comparison of both its versions.

Keywords: Cloud Computing, Legacy Application, Cloud Migration, Azure, Docker

Resumo

A computação na Cloud trouxe novas oportunidades de modernização para que as organizações modernizem as suas aplicações legacy. Migrar aplicações legacy para a Cloud e permitir que se tire partido das características da mesma tornou-se um objetivo para a maior parte das organizações cujas aplicações funcionem aquém das expectativas ou utilizem tecnologias antigas comparado com o que a Cloud tem para oferecer. Esta tese providencia o estado-da-arte e propostas de metodologias de migração. É apresentado o processo de migração de alguns componentes de uma aplicação em específico, o EDOCLINK, para o ambiente Cloud e uma comparação de performance da versão atual e da versão cloudificada.

Palavras-Chave: Computação na Cloud, Aplicação Legacy, Migração para a Cloud, Azure, Docker

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

A definição de computação na nuvem ou computação na Cloud consiste tanto nas aplicações fornecidas pela Internet como na infraestrutura alocada nos centros de processamento de dados que os fornecem [1]. Permite que aplicações sejam implantadas na Internet sem a necessidade de construir ou manter o hardware necessário para as servir. Também fornece a capacidade de implantar uma aplicação num dado ambiente cujas capacidades não são estáticas, visto que o risco de construir uma enorme infraestrutura de uma aplicação que acaba por ser um fracasso ou, o contrário, adquirindo hardware de pequena escala que só é capaz de fornecer uma pequena fração dos seus utilizadores é eliminado pela elasticidade de tais recursos. As companhias tentam ativamente alavancar esta plataforma a seu favor[2], através do desenvolvimento de novo software especificamente desenhado para a Cloud ou pela migração de aplicações legacy.

1.1 Objetivos

O trabalho desenvolvido no âmbito desta dissertação concentra-se no processo de migrar uma aplicação legacy para a Cloud. O Estado da Arte apresentará alguns exemplos de metodologias e técnicas desenvolvidas para guiar o processo de migração, algumas tecnologias com pontos de semelhança com a Cloud e uma breve apresentação dos maiores players na Cloud seguido da clarificação do que é a aplicação a migrar-se e a sua arquitetura atual. O objetivo principal deste relatório passa por apresentar o trabalho realizado na migração de uma aplicação legacy para a Cloud - o EDOCLINK - e uma comparação da performance da aplicação atual com a nova versão Cloud no sentido de concluir se foi possível tirar partido das maiores vantagens da Cloud a favor da performance do sistema.

2 Estado da Arte

2.1 Conceitos

O termo computação na Cloud ou computação na nuvem é várias vezes usado como palavra-chave em diferentes contextos para transmitir a noção de que pelo menos uma parte de um determinado sistema informático reside em recursos não-locais, fazendo parte de uma infraestrutura de grandes dimensões que é gerida por uma organização e que serve vários clientes. Para clarificar a definição efetiva de computação na Cloud, o NIST, National Institute of Standards and Technology, define-o como um modelo para permitir o acesso conveniente, ubíquo e à medida que for necessário a um conjunto partilhado de recursos computacionais configuráveis sejam eles de rede, de servidores, espaço de armazenamento, aplicações ou serviços. Estes recursos podem ser rapidamente atribuídos e retirados ao utilizador de forma rápida e automática e sem interação com o provedor destes mesmo recursos. [3]

Essa definição aponta algumas características chave da computação em nuvem que são a razão da sua crescente popularidade. A computação na Cloud proporciona acesso a recursos informáticos on-demand, o que significa que mesmo as empresas de menor dimensão podem tirar partido desses recursos, uma vez que o custo dessa infraestrutura é calculado segundo uma metodologia de pay-as-you-go, ou seja, é

proporcional às capacidades do hardware e ao período de tempo em que foi efetivamente utilizado. Ao implantar uma nova aplicação, essas empresas eram frequentemente confrontadas com um dilema: ou reduziriam custos para construir uma infraestrutura tão pequena quanto possível para lidar com suas necessidades atuais e enfrentariam o risco de não serem capazes de escalar caso a aplicação aumentasse bastante o número de utilizadores ou acomodar cargas potencialmente mais pesadas e enfrentar o risco de nunca ter realmente uma utilização suficientemente grande que justifique o tamanho da infraestrutura e gerando custos desnecessários.

A ideia de combinar recursos informáticos não é nova. Por exemplo, o conceito de **Grid Computing** é um modelo de computação distribuída que permite ao utilizador aceder a recursos de rede separados entre várias unidades de computação cujas capacidades são combinadas de modo a obter um maior poder de processamento. [3] Este modelo está relacionado com a computação em nuvem, uma vez que também faz uso de vários recursos livremente distribuídos, no entanto, a computação em GRID é muitas vezes proprietária (gerida e utilizada dentro de uma organização para processar uma única tarefa) em oposição à computação em nuvem, cuja infraestrutura é propriedade de um provedor de nuvem central e pode estar em locais fisicamente distantes, podendo, no entanto, ser utilizada por qualquer cliente que assim desejar.

Algumas empresas, devido à sua grande dimensão, podem optar pela construção de uma Cloud privada que sirva apenas os interesses dessa mesma organização ao invés de utilizar um provedor público. Este tipo de Cloud é considerada mais segura uma vez que a infraestrutura é construída de acordo com as necessidades específicas dessa organização e nunca partilhada com outros utilizadores [11]. Apesar de serem mais personalizáveis, este tipo de soluções são geralmente mais caras do que as públicas devendo ser apenas opção para organizações cujas necessidades assim impliquem.

Além das suas semelhanças com a computação em GRID, a computação em nuvem baseia-se no conceito chave de "virtualização", que é uma forma de abstrair o hardware físico e fornecer recursos virtualizados para uma aplicação [3]. Isto é muitas vezes conseguido pela introdução de uma camada - o Hypervisor ou Virtual Machine Manager - que é responsável por permitir que múltiplos sistemas operativos (e as aplicações que cada um deles executa) partilhem os recursos do hardware onde residem [5]. A virtualização permite que várias aplicações sejam executadas no mesmo hardware, cujos recursos são utilizados de forma mais eficiente. Essa tecnologia é um dos fundamentos da computação em nuvem, pois fornece a capacidade de atribuir recursos de computação de forma dinâmica e on-demand a múltiplas aplicações.

Apesar desta ser a definição mais comum de Virtualização, o próprio termo ganhou um novo significado com a introdução da **Virtualização baseada em Containers**. Ao contrário das VMs, os containers não têm seu próprio hardware virtualizado; utilizam o hardware do sistema que os alberga. Os containers oferecem um nível de abstração do ambiente no qual a aplicação é executada. Os containers são enviados com todo o pacote de bibliotecas e dependências necessárias para executar a aplicação, o que evita conflitos de dependência e separa facilmente diferentes projetos ou aplicações. Eles são extremamente leves em comparação com as VMs, já que não precisam de conter todos os componentes necessários para correr o sistema operativo. Essas características tornaram a virtualização baseada em containers muito

popular nos últimos anos. Atualmente, a maioria dos provedores de Cloud públicos agilizam o processo de implantação e gestão de aplicações containerizadas. A portabilidade de aplicações containerizadas é uma da maiores vantagens desta tecnologia, que permite que estas aplicações sejam implantadas em vários ambientes diferentes com um esforço extremamente reduzido. Outra das grandes vantagens consiste no facto de um container ter a capacidade de ser movido rapidamente entre ambientes com capacidades diferentes de modo a gerir períodos com intensidades distintas de carga[12]. Estas características vão ao encontro das facilidades e benefícios oferecidos pela Cloud, razão pela qual este tipo de virtualização se massificou em ambientes Cloud.

Os provedores de serviços de nuvem - Cloud Service Providers - como são conhecidos as empresas que fornecem serviços para executar aplicações na nuvem, aproveitam essas tecnologias de virtualização para oferecer aos seus utilizadores recursos de computação que são ajustados on-demand e faturados sob um método de pagamento pay-as-you-go. No entanto, as organizações de TI, como qualquer outro tipo de organização, têm necessidades e modelos de negócios muito diferentes. Dada essa granularidade, os provedores de nuvem oferecem serviços que estão agrupados em 3 grupos diferentes:

Infrastructure as a Service é um tipo de arquitetura adaptada para a Cloud na qual o hardware ou infraestrutura é virtualizado sob a forma de VMs. Os clientes muitas vezes escolhem este tipo de modelo porque é mais fácil e barato implantar a aplicação numa infraestrutura remota ao invés de ter que comprá-la, construí-la e geri-la completamente. Exemplo: Amazon Web Services (AWS), embora vários serviços disponíveis na AWS sejam comparáveis ao PaaS.

Platform as a Service é um tipo de arquitetura adaptada para a Cloud que introduz um nível adicional de abstração em comparação ao IaaS, fornecendo um ambiente de desenvolvimento ou API que permite ao cliente desenvolver a aplicação desejada dentro de um determinado ambiente. Nesta arquitetura, o provedor alberga a aplicação na sua própria infraestrutura e disponibiliza-a através da Internet. Exemplo: Azure, Google App Engine.

Software as a Service é um tipo de arquitetura adaptada para a Cloud em que o provedor é responsável por gerir e controlar toda a infraestrutura subjacente, fornecendo um software na forma de um serviço de Internet. O cliente não controla nenhuma parte do stack tecnológico necessário para suportar a aplicação, uma vez que é controlada por terceiros. Normalmente, o utilizador final acede apenas ao software através de um navegador web ou um terminal simples e as atualizações e upgrades são geridas pelo provedor. Exemplo: iCloud, Microsoft Office.

2.2 Como migrar aplicações legacy para a Cloud?

As mudanças no negócio, no que diz respeito ao âmbito de trabalho das organizações de TI, têm vindo a acelerar continuamente a forma como as empresas agilizam os seus processos, a fim de prestar um melhor serviço da forma mais rentável possível. No entanto, na maioria das vezes, esses sistemas dependem de infraestrutura legacy, utilizando tecnologia e/ou arquitetura com décadas de existência que não atendem às necessidades modernas. Assim, no sentido de manter o produto ou serviço que disponibilizam competitivo, as organizações devem acompanhar as mudanças de paradigma do mercado e das suas necessidades, modernizando o seu stack tecnológico ou, em alguns casos mais extremos, transformando completamente

a forma como sua aplicação funciona e as tecnologias que a mesma utiliza. Como esse não é um problema novo, existe bastante literatura sobre esse tema, que aborda o processo de migração de um sistema legacy para a nuvem de uma forma geral. Essas abordagens serão apenas usadas como um guia no processo; cada aplicação é única, e assim devem ser os passos para migrá-la. Em [7], os autores desenvolvem o "modelo Ferradura SEI"que divide a migração em 3 fases principais: reconstituição do sistema legacy para alcançar a arquitetura de base, transformação da arquitetura legacy para a arquitetura mais moderna desejada e refinamento da representação da arquitetura para alcançar um novo código-fonte do sistema. O primeiro passo é pegar no código-fonte, avaliar cada componente de acordo com seu papel e projetar uma representação da arquitetura atual. Com isso, o próximo passo é representar a arquitetura desejada após a migração, remodelando a arquitetura atual para atender às especificações do sistema. O passo final é, a partir da nova arquitetura, construir modelos que representem o sistema com um nível decrescente de abstração para adquirir uma nova versão do código fonte.

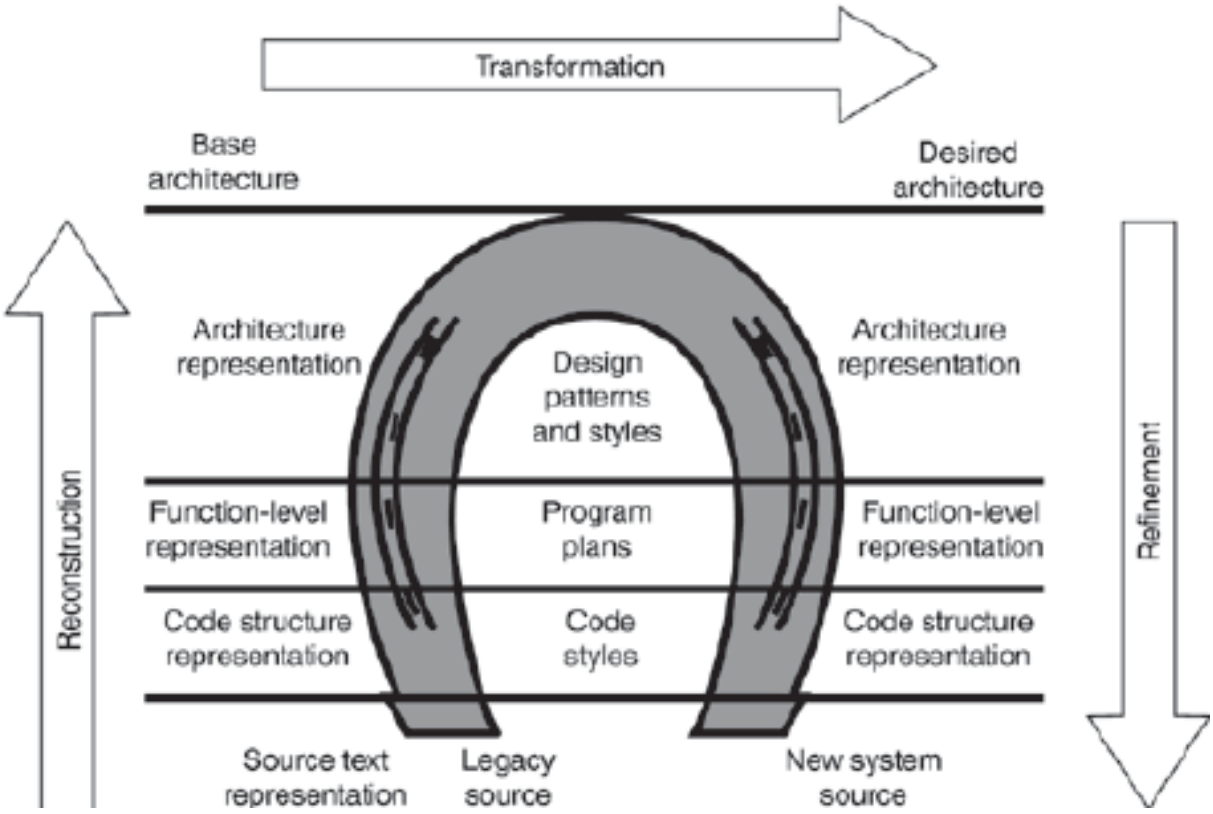

Figura 1. Modelo Ferradura SEI. [7]

Em [8] os autores separam a modernização das aplicações legacy em dois grupos principais: a black-box e a white-box. A abordagem white-box implica, à semelhança da primeira fase do modelo Ferradura SEI, um processo de reverse engineering para obter a forma como os processos internos do sistema funcionam e como eles interagem entre si. A abordagem black-box centra-se nas entradas e saídas do sistema. Basicamente, os processos internos do sistema não são relevantes nesta abordagem e a sua modernização

só deve envolver a aplicação antiga utilizando outros componentes que apenas gerem os dados que entram e saem do sistema.

Em [9], os autores desenharam um conjunto de atividades a serem realizadas e artefatos a serem obtidos durante o processo, conforme expresso na figura 2. As etapas A, B e C visam identificar os requisitos do sistema, listar os potenciais fornecedores de serviços na Cloud e utilizar a informação recolhida, combinando as necessidades da aplicação com os requisitos dos ambientes Cloud para criar um grupo de potenciais fornecedores de serviços na Cloud dos quais apenas um será escolhido. As etapas D, E e F dizem respeito à identificação de novas arquiteturas potenciais e à avaliação dos fornecedores de serviços na Cloud previamente selecionados em relação aos atributos de qualidade definidos para a aplicação. Após analisar cuidadosamente as restrições do sistema e dos provedores, essas três etapas devem produzir uma arquitetura final para o sistema migrado, juntamente com os provedores Cloud a serem usados no processo. Observe-se que apenas a etapa G e a etapa final consideram a implementação real, o que significa que grande parte da carga de trabalho associada à migração de uma aplicação legada para a nuvem consiste em avaliar o estado atual da aplicação, comparar os diferentes provedores da Cloud e realizar mudanças reais na arquitetura para suportar a implantação na Cloud.

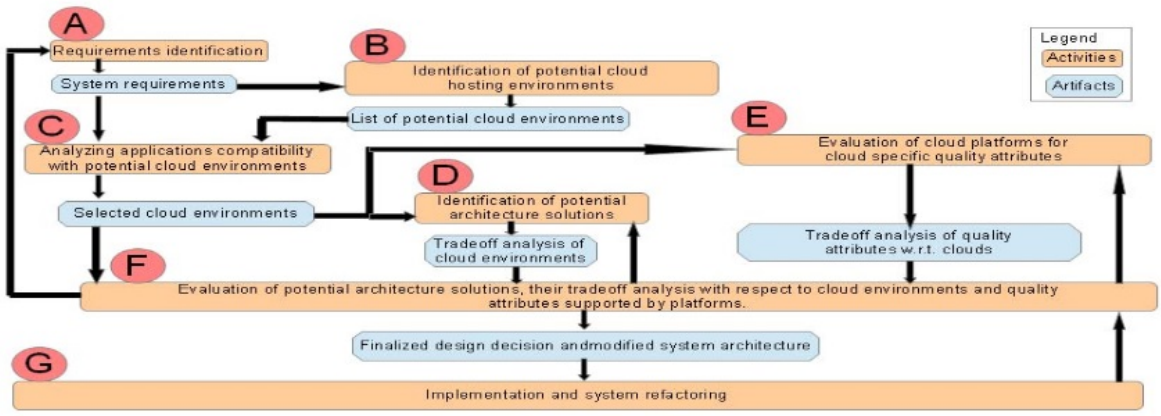

Figura 2. Processo de Migração [9]

Numa base mais geral, o [10] fornece 5 maneiras diferentes de modernizar uma aplicação, cada uma delas com características diferentes que devem ser avaliadas por cada empresa e projeto:

1. Transformação Total

Esta abordagem lida com a modernização, reconstruindo todo o sistema, começando do zero e tornando a antiga versão obsoleta.

2. Substituição Gradual

Um único componente é reconstruído usando nova tecnologia, enquanto o resto do sistema é mantido no seu estado atual. Esta abordagem lida com a modernização de cada componente separadamente e, gradualmente, todo o sistema é reconstruído.

3. Abordagem Duct Tape

Esta abordagem centra-se em mudanças de pequena escala utilizando novas tecnologias, mantendo intacta a arquitectura e a tecnologia de base.

4. Melhorar Existente

Concentra-se apenas na consolidação da aplicação existente, melhorando a manutenção do código e removendo código morto ou regras de negócio repetidas.

5. Sem alteração do sistema

Como o nome indica, nenhuma mudança é feita nesta abordagem, quando a empresa decidiu que o ROI - Return on Investment - não compensa as mudanças.

Estas abordagens diferem em termos do risco que implica, duração, financiamento, nível de modernização ou popularidade. A decisão de optar por uma delas é extremamente dependente do negócio. Uma tabela de comparação destas abordagens é apresentada na figura 3.

	1. Total Transformation	2. Gradual Replacement	3. Duct Tape Approach	4. Improve Existing	5. No System Change
Modernization	Highest	High	Moderate	Low	None
System Change	Enterprise Level	Application Level	Localized Change	Application Level	None
Risk Severity	Highest Risk	Moderate Risk	Low Risk	Moderate Risk	Risk of Falling Behind
Funding Need	Highest	High	Moderate	Moderate	Low
Duration	Long (Multiyear)	Moderate (1-2 years)	Moderate (2-6 months)	Moderate (4-10 months)	Not Applicable
Benefits	Highest	High	Moderate	Moderate	Little
Popularity	Low	High	Highest	Moderate	Low
Success Rate	Moderate	High	Highest	High	Not Applicable

Figura 3. Comparação de propostas [10]

Como conclusão, há algo que deve ser tido em conta ao selecionar que arquitetura Cloud deve ser usada para migrar uma aplicação legacy: não há escolha perfeita. A escolha da arquitetura Cloud apropriada depende fortemente de vários fatores, como a duração da migração, o nível de modernização necessário, a quantidade de financiamento necessário e as tecnologias e arquiteturas presentes no sistema legacy. Como vimos nesta secção, há, é claro, diferentes custos e restrições de tempo relacionados a cada uma das abordagens e elas não são de forma alguma imutáveis, ou seja, cada organização deve fazer as mudanças adequadas em uma dessas abordagens para melhor atender às suas necessidades. No entanto, isso é útil para entender que não existe uma metodologia livre de riscos a seguir, uma vez que a migração para a Cloud é uma tarefa muito complexa com grande impacto no negócio.

2.3 Provedores de serviços Cloud

Além do processo de seleção de metodologia para realizar a migração real, estudar e escolher o provedor Cloud adequado a partir da extensa lista de alternativas é outra tarefa que tem um enorme impacto no sucesso da migração. Amazon, Google, Microsoft, IBM, Alibaba e várias outras empresas oferecem soluções de computação na Cloud. No entanto, a Amazon é líder de mercado com 47.8% de quota de mercado em 2018 [13]. A Amazon Web Services e o Microsoft Azure serão discutidos mais adiante, já que são os 2 maiores provedores da atualidade.

2.3.1 Amazon Web Services - AWS

Um dos mais difundidos e versáteis provedores de serviços Cloud é a Amazon Web Services (AWS) [15]. Oferece dezenas de serviços de computação na Cloud, desde armazenamento de dados, até um serviço de bases de dados relacionais.

EC2, abreviação de Elastic Compute Cloud, permite que os clientes tenham uma máquina virtual na Cloud que é completamente autoescalável, ou seja, cresce e diminui sua capacidade de computação de acordo com a carga introduzida no sistema a qualquer momento automaticamente.

S3, abreviação de Simple Storage Service, como o nome indica é uma solução de armazenamento de dados na Cloud que fornece armazenamento de objetos acessível de qualquer lugar da web através de uma API REST ou AWS SDKs disponíveis para várias linguagens de programação. Possui uma das maiores taxas de disponibilidade e escalabilidade e é utilizado por vários gigantes tecnológicos como Netflix e Airbnb.

RDS, abreviação de Relational Database Service, é o serviço da AWS para implantar uma base de dados relacional na Cloud, fornecendo uma capacidade redimensionável. Inclui seis tipos conhecidos como PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server e Aurora, desenvolvidos pela própria Amazon. Também fornece um serviço de migração para migrar a base de dados de aplicações legacy para RDS.

A maior vantagem da Amazon Web Services é a sua capacidade de integrar todos os seus serviços, fornecendo soluções para todos os tipos de problemas empresariais, que foram totalmente testados e são usados por milhões de utilizadores todos os dias.

2.3.2 Microsoft Azure

Microsoft Azure [17] é o segundo maior provedor de computação na Cloud e também oferece vários serviços diferentes [14]. Como o seu maior concorrente, a AWS, é relativamente recente, o Azure tem uma vantagem quando se trata de clientes que confiam em produtos da Microsoft há muito tempo.

O Azure fornece os mais variados serviços para alojar aplicações na Cloud. Fornece VMs das mais variadas categorias no sentido de aproximar as capacidades da mesma às necessidades da aplicação que vai albergar, com um preço sempre proporcional a estas mesmas capacidades.

Para aplicações containerizadas, fornece um serviço de repositório de containers onde podem ser guardadas as imagens privadas de containers criadas para uma aplicação em específico chamado Azure Container Registry. Cada um destes containers pode depois ser implantado utilizando o serviço Azure App Service que fornece inumeras configurações, nomeadamente a possibilidade de escalar a performance do mesmo, de forma automática.

Fornece também um serviço de armazenamento de dados e de alojamento de bases de dados migradas de SQL Server.

Por razões históricas, os sistemas da Microsoft ainda são extremamente populares e muitas aplicações legacy foram criadas utilizando produtos da Microsoft. O Microsoft Azure também fornece um ambiente de desenvolvimento muito integrado com o Visual Studio. A maioria dos processos de implantação de uma nova versão de uma aplicação estão bastante ágeis entre o Visual Studio e o Azure o que facilita ainda mais estas tarefas.

3 EDOCLINK - Arquitetura Atual

Como foi explicado na secção de objetivos, o foco deste projeto consiste em migrar uma aplicação legacy para a Cloud. A aplicação em específico que vai ser alvo deste processo de migração é o EDOCLINK, uma plataforma de gestão documental e worfklow. O EDOCLINK realiza várias ações relacionadas com o ciclo de vida de um documento, como acionar um conjunto de etapas quando um documento é introduzido, controlar acessos de leitura e escrita ao documento ou associar prazos ou notificações de acordo com a data em que o documento deve ser despachado ou manuseado. Existem várias empresas que utilizam o EDOCLINK como a sua principal plataforma de gestão documental, tornando-o uma parte crítica do dia-a-dia destas organizações.

Uma instalação padrão do EDOCLINK é composta pelo Servidor Aplicacional, módulo principal da aplicação que alberga a maioria das funcionalidades do EDOCLINK, e uma base de dados SQL Server que são implantadas na mesma máquina, em ambientes de pequenas dimensões. No caso de ambientes de maiores dimensões são utilizadas várias máquinas tanto para o Servidor Aplicacional como para a Base de Dados. À data deste relatório, foi disponibilizada uma nova versão do EDOCLINK que possibilita a migração da base de dados para uma instância SQL Azure, permitindo a remoção deste componente da máquina local que frequentemente também contém o Servidor Aplicacional. Para além destes dois módulos principais, existem também 3 outros módulos satélite: o Helper Service, o Document Converter Service e o Binary File Repository.

3.1 Helper Service

Começando pelo Helper Service, este módulo é responsável por comunicar com o Servidor Aplicacional e recolher uma lista de alarmes/tarefas pendentes e executá-los. A recolha da lista de alarmes é feita através da chamada de um webservice SOAP contido no Servidor Aplicacional. De acordo com o tipo de cada alarme obtido, é chamado o respetivo webservice SOAP, também este do lado do Servidor Aplicacional, responsável por executar a tarefa em questão. Este módulo é implantado como um serviço Windows e é despoletado periodicamente de acordo com a configuração introduzida.

3.2 Document Converter Service

Relativamente ao Document Converter Service, este módulo é, como o nome indica, responsável pela conversão de documentos para o formato de imagem desejado, tipicamente JPG. Este módulo é utilizado em duas funcionalidades do EDOCLINK, sendo elas a apresentação de um thumbnail de um documento e no visualizador de documentos onde é possível navegar por todas as páginas de um documento. No caso do visualizador de documentos, o Document Converter Service é responsável, para além da conversão das páginas do documento para formato JPG, por determinar o número de páginas desse mesmo documento. A apresentação do thumbnail não é mais do que a conversão da primeira página desse documento para JPG e o redimensionamento da imagem obtida para uma resolução menor. Num pedido de conversão, o Document Converter Service obtém o documento que se encontra no Binary File Repository e efetua a conversão da página pretendida. De seguida, guarda o documento e respetiva imagem numa cache própria,

construída para impedir conversões repetidas das mesmas páginas de um documento num curto espaço de tempo. A diretoria onde são guardados os documentos e respetivas imagens tem um tamanho máximo configurável. Se ultrapassado este limite, o Garbage Collector implementado no Document Converter Service como mecanismo de gestão de cache trata de apagar os documentos e imagens que não são utilizados há mais tempo. No caso específico das conversões de ficheiros DOCX, o Document Converter Service atual trata primeiro de converter o documento por inteiro para PDF. Este passo adicional permite que em conversões subsequentes de outras páginas do mesmo documento, não seja necessário converter o ficheiro DOCX para PDF e só depois a página em concreto de PDF para JPG, visto que é possível reaproveitar o documento em PDF por inteiro já convertido anteriormente.

Na versão atual do EDOCLINK, é atribuído um GUID - Globally Unique Identifier - a cada documento inserido e a cada utilizador. O GUID é representado por uma sequência de 32 dígitos hexadecimais no formato 8-4-4-4-12, i.e., "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx", correspondendo a um total de 128 bits. A utilização de um GUID como forma de identificação de um utilizador ou documento garante a unicidade do identificador ao longo de todas as tabelas da base de dados e inclusivamente ao longo de diferentes bases de dados uma vez que, quando gerados sob os métodos padrão de geração de GUIDs, a probabilidade de se gerar um identificador duplicado é próxima o suficiente de 0 para que esta possibilidade seja descartada. Os GUIDs dos utilizadores e documentos são então utilizados para os identificar e utilizados em diversas ações como por exemplo a verificação de acesso de um utilizador a um dado documento.

3.3 Binary File Repository

O Binary File Repository alberga os documentos introduzidos no EDOCLINK pelos utilizadores e providencia-os no caso de o utilizador requerer o download do mesmo ou a sua visualização. Este repositório pode funcionar em filesystem ou como um repositório na Cloud, mais especificamente o Azure, ao qual se acede remotamente.

Como ponto de partida, é importante apresentar a atual arquitetura EDOCLINK:

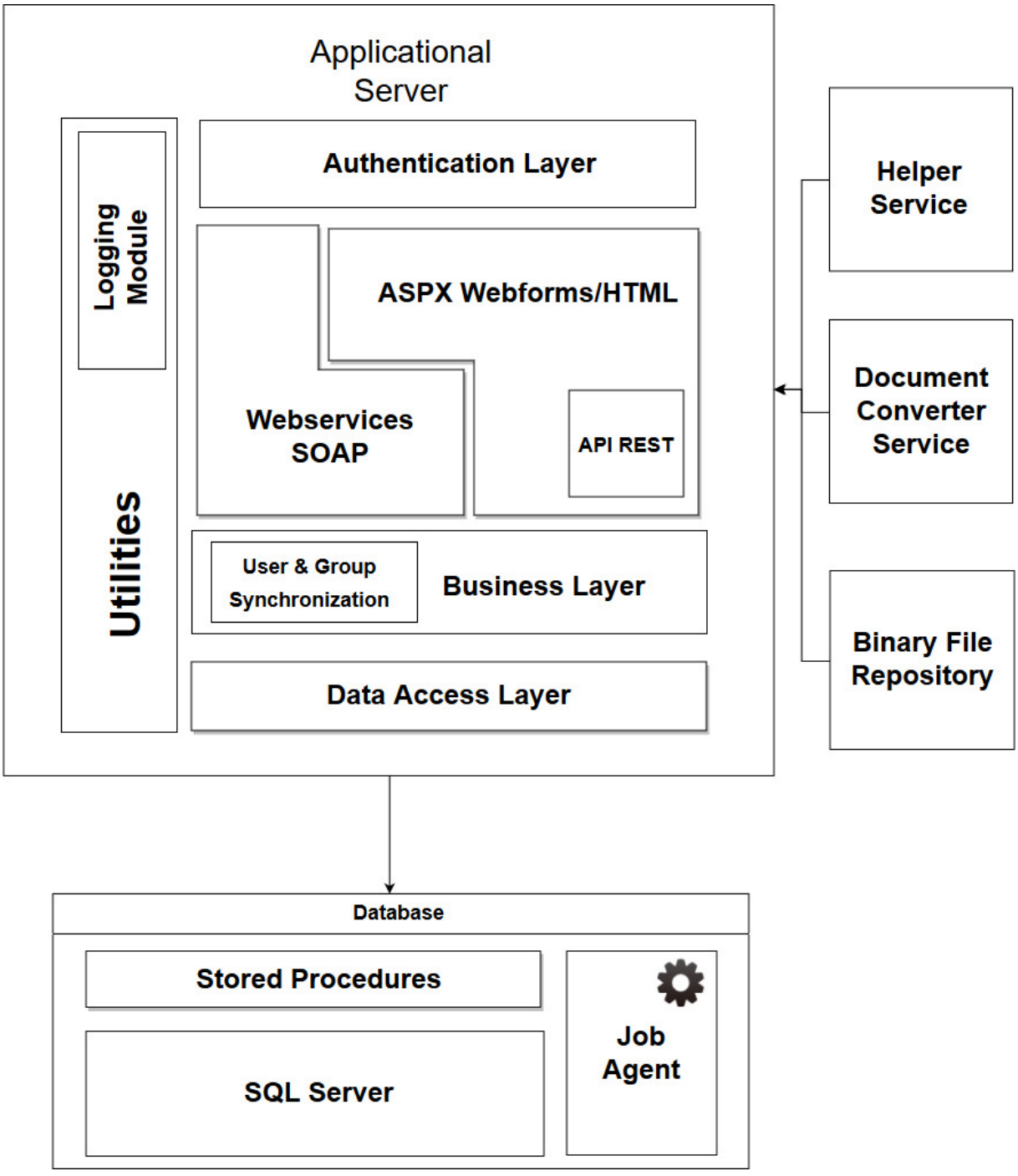

Figura 4. Arquitetura Lógica do EDOCLINK Atual

O Sevidor Aplicacional contém vários sub-módulos responsáveis por diversas funcionalidades. Começando pela camada inferior, a camada de acesso aos dados - Data Access Layer - é responsável por realizar um conjunto de funções interagem com a base de dados para obter as mais variadas informações, como por exemplo permissões de acesso a um determinado documento, autor, data de criação, etc.

A camada de negócio - Business Layer - detém a maioria das regras de negócio e é responsável pela maior parte da lógica na aplicação. A camada de Business também mantém um sub-módulo, responsável pela Sincronização de Utilizadores e Grupos. Quando o EDOCLINK é instalado num novo cliente, já existem utilizadores registados nos seus sistemas, bem como grupos que representam uma determinada equipa ou departamento que partilham configurações de permissões de acesso ou de determinadas ações específicas. Este módulo é responsável por obter estes utilizadores e grupos do sistema interno da organização e sincronizá-los com a base de dados do EDOCLINK. Além da camada de Business, existe também a camada de Web Services que expõe os serviços SOAP. Tal como referido na descrição do Helper Service, a comunicação com o Servidor Aplicacional é feita através destes serviços. Quando um destes serviços é chamado, a camada de Business despoleta os mecanismos necessários para efetuar a ação pretendida, seja ela a recolha da lista de tarefas a efetuar, a criação de um alarme ou a recolha de dados de configuração.

O ASPX Webforms/HTML é responsável por apresentar o EDOCLINK ao utilizador e lidar com as interações e pedidos que o mesmo efetue. Como qualquer outro sistema, o EDOCLINK precisa de fornecer uma maneira de autenticar os seus utilizadores. Em vez de construir um novo módulo de autenticação específico para executar esta tarefa, este módulo delega esta responsabilidade a um provedor de autenticação externo, como o Facebook ou Twitter, ou ao serviço de autenticação do Windows, usando um Active Directory. O objetivo deste módulo é conferir ao EDOCLINK a propriedade de Single Sign-On - SSO - tipicamente utilizando a Active Directory que gere os utilizadores e recursos da organização onde o EDOCLINK foi instalado e assim permitir que a autenticação dos utilizadores no EDOCLINK se faça com as credenciais com as quais os utilizadores se autenticam na organização à qual pertencem.

4 EDOCLINK - Nova Arquitetura

Tendo como objetivo a migração do EDOCLINK para a Cloud, é necessário reformular a arquitetura atual desta aplicação por forma a que seja capaz de tirar partido dos benefícios que a Cloud tem para oferecer. A arquitetura atual do EDOCLINK baseia-se numa arquitetura monolítica que contém a maior parte dos módulos e funcionalidades da aplicação. O objetivo inicial seria a migração por completo da aplicação para o ambiente Cloud. No entanto, este objetivo acabou por se revelar demasiado ambicioso dadas as restrições de tempo designado para a execução deste trabalho. Assim sendo, foi necessário definir que componentes serão alvo da migração da sua funcionalidade para a Cloud. Neste sentido, a escolha acabou por recair nos módulos externos como o Helper Service, o Document Converter Service e o Binary File Repository que comunicam com o monolito - o Servidor Aplicacional - mas executam as suas funcionalidades de forma independente. Esta característica fez com que estes componentes figurassem no topo da lista de módulos prioritários para os quais será efetuada a migração. A migração destes componentes para serviços independentes que são implantados em ambiente Cloud permitirá manter a sua funcionalidade, trazer melhorias na escalabilidade dos mesmos e flexibilidade nas alterações à forma como determinadas funcionalidades destes módulos estão implementadas.

Definidos os principais componentes alvo de alterações, é necessário determinar qual a melhor forma de os implantar num ambiente Cloud. Relativamente ao Helper Service e ao âmbito das suas funcionalidades, é necessário que exista um serviço no Azure que permita a execução das tarefas atribuídas ao Helper Service de forma periódica. As Azure Functions são um serviço Azure que permite a execução de pequenas funções na Cloud, com suporte para variadas linguagens de programação, e que adota o modelo de pagamento por utilização em que apenas é cobrado o tempo durante o qual a função é executada. Visto que este módulo está adormecido na maior parte do tempo e deve apenas ser despoletado em intervalos de tempo configuráveis, este serviço foi escolhido para efetuar a migração do Helper Service para a Cloud e assim funcionar como serviço independente.

O Binary File Repository, responsável por guardar os documentos dos utilizadores do EDOCLINK, será também migrado para a Cloud. Atualmente, este módulo é configurável para diferentes modos de funcionamento entre os quais guardar os documentos em filesystem na máquina onde está instalado o Servidor Aplicacional e a utilização de um repositório na Cloud do Azure. Assim sendo, no sentido de migrar este componente para a Cloud retirando-o do monolito, este repositório passará a funcionar apenas como repositório no Azure, utilizando o serviço Azure Blobs, concebido para armazenar grandes quantidades de ficheiros, sejam eles imagens ou documentos, que podem ser acedidos de variadas formas nomeadamente através de uma biblioteca disponibilizada pelo Azure para diferentes linguagens de programação.

A atual implementação do Document Converter Service, no entanto, revelou-se um entrave a migração *as-is* deste componente para a Cloud como serviço independente. Os documentos pedidos e respetivas imagens são, antes de apresentadas ao utilizador, guardadas em cache para posterior utilização como explicado anteriormente. A migração deste componente para um serviço independente na Cloud impossibilita desde logo que a cache exista da mesma forma e implica também a reformulação da forma como o Servidor Aplicacional envia pedidos de conversão ao Document Converter Service.

Apesar da funcionalidade de conversão e gestão de cache se manter, a migração deste componente implicará uma reformulação total da sua arquitetura. Por esta razão, este seria o momento ideal para agilizar a forma como se efetua o processo de implantação de uma nova versão do Document Converter Service. Neste sentido, e garantindo que é possível tirar partido da escalabilidade *on-demand* do ambiente Cloud, optou-se pela containerização deste componente. O processo de containerização permite o isolamento do ambiente de execução da aplicação para que contenha todas as dependências e bibliotecas necessárias à sua execução. O container é gerado a partir de um dockerfile que consiste num ficheiro de configuração do mesmo, partindo de uma imagem base de sistema operativo ao qual se adicionam as bibliotecas necessárias, no caso do Document Converter Service a instalação do .NET Core SDK. O Docker Engine, responsável pela gestão dos containers numa determinada infraestrutura, tratará então de implantar cada um destes ambientes com base no seu dockerfile. Para além da facilidade introduzida na gestão das bibliotecas e necessárias, o processo de implantação está bem integrado com a plataforma Azure e o IDE Visual Studio, utilizado no desenvolvimento deste projeto. A partir do próprio Visual Studio, é possível enviar uma nova versão do container para o serviço Azure Container Registry, responsável por armazenar imagens privadas de container personalizadas. Na primeira implantação da aplicação, partindo da imagem de container colocado neste serviço, é utilizada a funcionalidade Web App for Containers para implantar a aplicação no Azure App Service, serviço capaz de albergar aplicações Web e tratar o escalamento das mesmas, funcionalidade que será testada mais à frente. A partir da primeira implantação, sempre que partir do Visual Studio uma atualização da imagem de container personalizada para o repositório de containers no Azure, as alterações serão imediatamente propagadas à aplicação exposta pelo Azure App Service, demorando segundos até que esteja operacional.

Assim sendo, para permitir a comunicação entre o módulo principal do EDOCLINK e o Document Converter Service foi criada uma aplicação que disponibiliza uma API que recebe pedidos de conversão e de contagem do número de páginas de um documento e se responsabiliza por obter o documento em questão, efetuar a operação necessária sobre o mesmo, gerir a cache de páginas já convertidas e responder de volta ao EDOCLINK. O Document Converter Service sofreu então uma reconstrução a nível da sua arquitetura interna, tendo sido dividido em duas aplicações distintas: o DCSLogic que gere a cache e recebe os pedidos provenientes do EDOCLINK e o DCSConverter que recolhe o documento do Binary File Repository, converte páginas para imagens e efetua a contagem do número de páginas desse documento. Cada uma destas aplicações foi containerizada. Desta forma, o DCSLogic recebe na sua API um pedido de conversão proveniente do Servidor Aplicacional e reencaminha-o para a API do DCSConverter, caso não exista em cache. Para garantir a autenticidade dos pedidos recebidos pela API do DCSLogic, cada um deles contém um JSON Web Token como cabeçalho do mesmo. Os JSON Web Tokens, daqui em diantes chamados JWT, são strings que representam um conjunto de claims (informações frequentemente relacionadas com a identificação de um utilizador) que são passadas a um serviço externo que precisa de validar a legitimidade do pedido. Para garantir a não adulteração do mesmo, o JWT contém também uma assinatura criada através de uma chave conhecida por ambas as entidades que comunicam. A explicação mais detalhada do funcionamento do Document Converter Service e do funcionamento da validação dos pedidos com base nestes JWT serão feitas na secção Trabalho Desenvolvido.

Relativamente ao Servidor Aplicacional, o objetivo seria também a migração para um container que pudesse também ser implantado no Azure. No entanto, a recolha de todas as dependências e bibliotecas que a aplicação utiliza e os constrangimentos que surgiriam na criação do container que empacotasse a aplicação limitaram a sua exequibilidade no período de tempo durante o qual se estende a realização deste projeto. Assim sendo, optou-se por manter o servidor aplicacional numa VM no Azure, permitindo ter mais tempo para concluir as migrações dos módulos externos e avaliação dos resultados obtidos com os mesmos.

Quanto à base de dados, o objetivo inicial seria fazer a sua migração para uma instância no SQL Azure. Tal objetivo foi atingido pela equipa de desenvolvimento do EDOCLINK numa fase já adiantada da realização deste projeto pelo que, também por constrangimentos a nível de tempo, se optou por manter a base de dados na mesma VM que alberga o servidor aplicacional.

Assim sendo, a arquitetura final da versão implantada na Cloud do Azure do EDOCLINK está apresentada na figura seguinte:

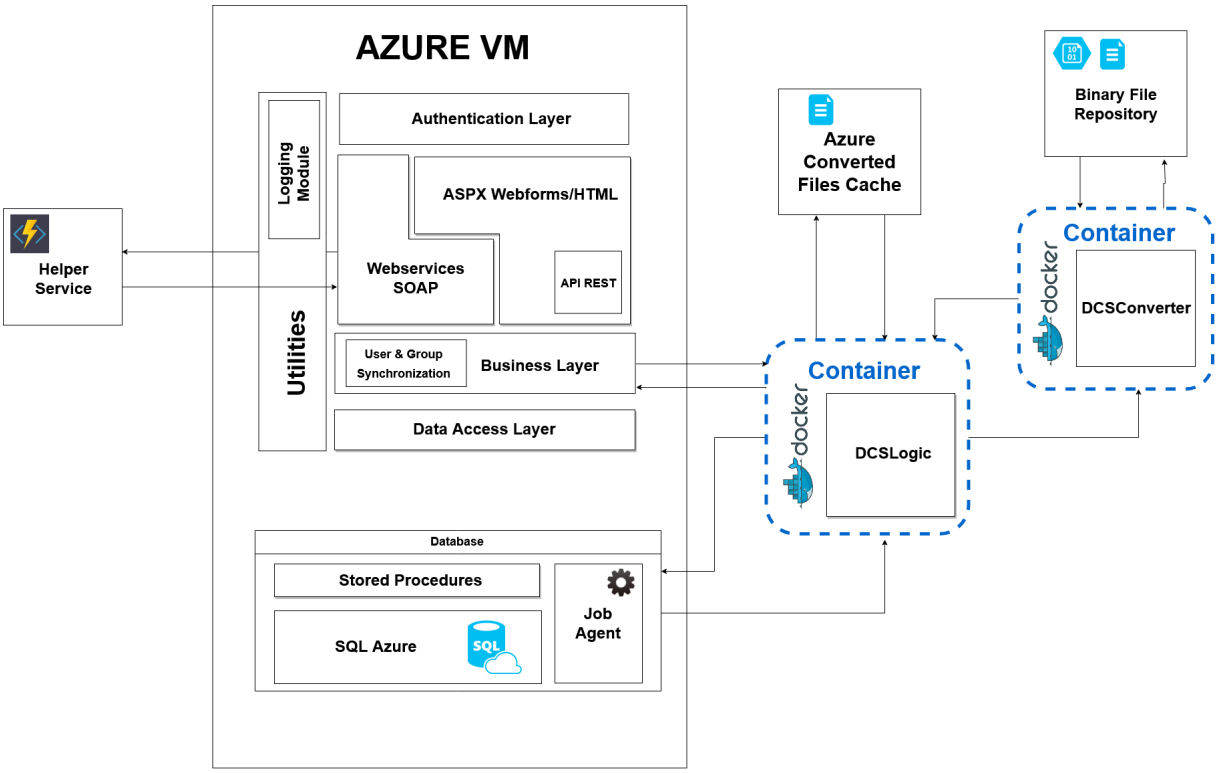

Figura 5. EDOCLINK - Arquitetura adaptada à Cloud

5 Trabalho Desenvolvido

Tendo como objetivo final possibilitar a execução de todas os módulos que constituem o EDOCLINK em ambiente Cloud e, se possível, retirar parte da complexidade do sistema do monolito que hoje alberga a maior parte do negócio, foram migrados vários componentes para a plataforma Azure. Como ponto de partida, foi necessário implantar o EDOCLINK atual numa VM alojada no Azure. Partindo desta versão, foram migrados os 3 componentes satélite explicitados anteriormente. Para o Binary File Repository foi utilizado o serviço Azure Blobs, para o Document Converter Service foi utilizado o serviço Azure App Service em conjunto com o serviço Azure Files para funcionar como um dos níveis da cache das imagens já convertidas e para o Helper Service foi utilizado o serviço Azure Functions.

5.1 Implantação da versão atual do EDOCLINK numa VM alojada no Azure

O primeiro passo realizado na migração do EDOCLINK para a Cloud foi a criação de uma máquina virtual no Azure e posterior instalação do servidor aplicacional, da base de dados e dos componentes satélite nessa mesma máquina. Desta forma foi possível obter uma versão operacional do EDOCLINK a correr em ambiente Cloud e que servirá de base para os trabalhos seguintes. No processo de criação de uma máquina virtual no Azure, são disponibilizados diferentes configurações do hardware a atribuir a essa mesma máquina. Dependendo da aplicação e das funcionalidades que esta possui, define-se qual a configuração ideal a utilizar para esse contexto em específico. Inicialmente, no caso do EDOOCLINK, a VM foi configurada na categoria A0 que corresponde, no Azure, a uma VM com 0.75GB de RAM, máximo de 300 IOPS (medida de operações de leitura e escrita em disco por segundo) e 1 vCPU, que corresponde a um núcleo virtual de processador que pode ou não correponder a um núcleo real de um processador (depende da forma como o Azure virtualiza o hardware necessário para esta determinada VM). Após a instalação do EDOCLINK nesta VM, rapidamente se verificou que a experiência de utilização estava a ser demasiado lenta. Assim, procurou-se aumentar as capacidades desta VM, alterando-a para uma configuração com mais capacidade de processamento, mais memória e, especialmente, mais IOPS para que a leitura e escrita em disco seja mais rápida. A VM foi então configurada para a categoria F2S que equivale a uma VM com 2 vCPU's, 4GB de RAM e um máximo de 6400 IOPS. Esta alteração permitiu uma melhoria substancial na utilização do EDOCLINK, aproximando-se assim de um ambiente real, sendo que os testes em capítulos posteriores à versão atual do EDOCLINK foram efetuados com base nesta configuração.

5.2 Document Converter Service - Arquitetura Técnica

Uma das capacidades que o EDOCLINK possui é a de apresentar os documentos que tenham sido carregados pelo utilizador num visualizador embutido na interface. A responsabilidade de converter um documento para um formato de imagem JPG é responsabilidade do Document Converter Service. Este componente permite, para além desta conversão, redimensionar a imagem convertida de modo a poder ser apresentada sob a forma de thumbnail. Existe também uma cache na qual são guardadas as imagens já convertidas de modo a que não sejam repetidas conversões para a mesma página de um documento. Esta cache foi reformulada face à sua versão atual e possui agora dois níveis distintos, em memória e sob a forma de ficheiros guardados num repositório Azure Files. As imagens convertidas ou acedidas mais recentemente são mantidas em memória e no repositório Azure Files. Caso o tempo máximo configurado inicialmente durante o qual a imagem deve ser mantida em memória seja ultrapassado ou o tamanho máximo da cache em memória seja atingido, a imagem em base-64 é removida da cache em memória mas mantida no repositório Azure, mantendo um url associado à página do documento em questão para o ficheiro contendo a imagem em base-64 alojado no Azure Files.

Para clarificar de que forma se processa uma conversão, definem-se os diferentes passos que devem ser executados:

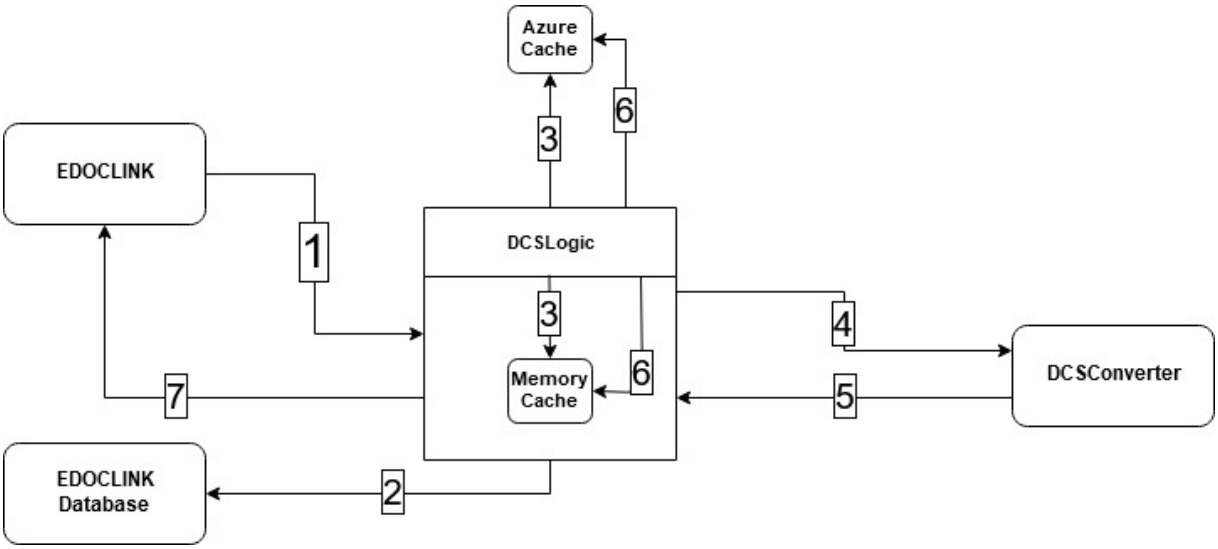

Figura 6. Passos de um pedido de conversão.

1. Receber um pedido de conversão contendo o GUID do documento a converter, uma flag que define o facto de ser ou não um thumbnail e a página do documento a converter. Este pedido deve conter um header de autenticação onde é passado o GUID do utilizador em sessão.
2. Utilizando o GUID do documento e o GUID do utilizador em seesão, aceder à base de dados do EDOCLINK, garantir a existência do documento e verificar se o utilizador em questão tem acesso ao documento.

3. Caso não tenha acesso, responder ao pedido com um HTTP 401 informando que o utilizador não tem acesso ao documento especificado. Caso o utilizador tenha acesso, verificar se existe a página do documento pedida, já convertida, na cache em memória. Caso a entrada relativa a essa página exista mas a imagem em base-64 esteja vazia, significa que a mesma está alojada no repositório Azure. Assim, é obtida a imagem em base-64 através do url para o ficheiro que a contém no repositório Azure. Dado que acabou de ser feito um pedido de conversão dessa mesma imagem, o campo que contém a imagem em base-64 na cache em memória é preenchido de forma a que esteja acessível mais rapidamente numa próxima conversão.
4. Caso exista em cache, devolver a imagem em base-64. Caso não exista em cache, enviar o pedido de conversão para o DCSConverter.
5. O DCSConverter efetua a conversão da página indicada e devolve-a sob a forma de imagem em base-64.
6. Adicionar uma entrada à cache em memória e à cache no repositório Azure contendo a imagem em base-64.
7. Responder ao pedido inicial com a página do documento especificado convertido para JPG e redimensionado, caso tenha sido pedido, codificada em base-64.

Os passos descritos permitem distinguir as responsabilidades deste módulo em dois sub-módulos com funções distintas: o DCSLogic responsável por verificar o acesso do utilizador ao documento e fazer a gestão da cache dos documentos e o DCSConverter responsável pela efetiva conversão do documento.

5.2.1 Implementação DCSLogic

A utilização de containers, como explicado nas secções anteriores, permite a abstração da aplicação e das suas dependências do ambiente onde esta é colocada. O Azure disponibiliza uma plataforma que facilita todo o processo de implantação de um container na Cloud. O passo inicial é a criação do container, contendo a aplicação e todas as suas dependências e o seu carregamento para um Azure Container Registry que não é mais do que um repositório de containers. Após este carregamento é possível utilizar a funcionalidade Web App for Containers existente no Azure e que disponibilizará o container como uma aplicação web, acessível através de um IP fornecido, utilizando o Azure App Service.

Ao receber um pedido de conversão, a primeira função deste componente será verificar se o utilizador que efetuou o pedido tem acesso ao documento em questão. Esta validação do acesso de um utilizador a um documento é feito através uma stored procedure à base de dados do EDOCLINK. Para efetuar esta stored procedure são necessários, entre outros parâmetros, o GUID do documento e o GUID do utilizador em questão. Quanto ao GUID do documento, este é obtido através de um parâmetro contido no URL do pedido:

```
https://dcslogic(...)?fileguid=7DD4DF6C-AA43-4E6B-849D-08C7D8A54190
```

No que respeita ao GUID do utilizador, este é enviado através de um JSON Web Token (JWT) que é adicionado como header do pedido. O processo de criação e a análise dos riscos associados à partilha do GUID do utilizador via JWT serão apresentados adiante.

Esta stored procedure aplicada na base de dados do EDOCLINK devolve uma série de parâmetros relativos ao documento, caso este exista na versão correta e o autor seja o utilizador em sessão. Entre estes parâmetros está a data de criação do documento. Esta data de criação é utilizada para construir o caminho deste ficheiro no repositório Azure onde está armazenado a ser passado ao DCSConverter quando lhe for enviado o pedido de conversão, caso seja necessário.

Após garantido que o utilizador em questão tem acesso ao documento, o DCSLogic irá procurar uma entrada na cache que contenha a imagem pedida já convertida. Cada entrada desta cache consiste num par chave-valor. A chave de cada entrada é constituída pela concatenação do GUID do documento, do número da página pedida e de um identificador de thumbnail caso tenha sido indicado. A cada uma destas chaves corresponde um valor que contém uma string contendo a imagem convertida codificada em base-64 e um url para a localização da imagem no repositório do Azure, caso esta lá se encontre.

Quando o limite de tempo configurado previamente durante o qual uma entrada se mantém em cache sem acessos ou modificações é atingido, uma callback é despoletada. Esta callback foi programada para que ao remover esta entrada da cache, se volte a adicionar exatamente a mesma entrada mas com o campo que contém a imagem em base-64 em branco. Desta forma, o tamanho desta entrada na cache em memória diminui substancialmente dado que a grande maioria do espaço ocupado se deve à string que contém a imagem codificada. Para além do decréscimo da memória ocupada, a reintrodução da entrada na cache permite que se mantenha sempre a localização da imagem no repositório Azure. Assim sendo,

a cache das imagens possui um sub-módulo persistente - o repositório no Azure - e um sub-módulo não persistente - a cache em memória.

5.2.1.1 Criação do JWT e Possíveis Ataques

Este JWT é criado no bloco principal do EDOC e é composto por 3 partes distintas: um header, um payload e uma signature. Tipicamente, o header é composto pelo tipo de token - neste caso JWT - e pelo algoritmo utilizado na assinatura - neste caso HS256. O payload consiste numa sequência de parâmetros através dos quais será passado o GUID do utilizador, num formato chave-valor i.e. "user_guid": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx". No pay load são também enviados outros parâmetros para garantir a validação do token. É enviada a chave "iat- issued at - com o valor do timestamp em que o token foi criado e a chave "exp- expiration - com o valor do timestamp a partir do qual o token expira e deixa de ser válido. A signature consiste na aplicação do algoritmo escolhido ao header e payload utilizando o secret.

Concatenando estas 3 partes distintas usando um '.' como delimitador obtém-se o JWT a enviar como header no pedido de conversão. Ao receber um pedido de conversão, o DCSLogic tratará de ler o header contendo o JWT do pedido. Caso não exista, o pedido é imediatamente descartado. Caso exista, é em primeiro lugar obtido o valor da chave "exp". Se o valor deste timestamp for inferior ao atual, isto significa que o token já expirou e que portanto o pedido deve ser descartado. Caso o token exista e seja considerado válido tendo em conta o timestamp de expiração, o GUID do utilizador em sessão será então obtido, garantindo sempre que o JWT não foi adulterado recorrendo para isso ao SECRET utilizado na sua criação e que é partilhado pelo módulo principal do EDOCLINK e pelo DCSLogic. O SECRET é uma chave partilhada pelo DCSLogic e pelo Servidor Aplicacional que é utilizada no cálculo do hash do header e do payload pelo algoritmo HMACSHA256 [25]. A adição dos parâmetros acima mencionados ao JWT confere-lhe um maior grau de robustez na resistência a possíveis ataques. Dois tipos de ataque passíveis de serem executados a esta comunicação são apresentados de seguida. No ataque representado na figura seguinte, um atacante malicioso é capaz de intercetar um pedido de conversão efetuado pelo EDOC ao DCSLogic. Ao intercetar o pedido o atacante tem acesso a uma série de informações: o GUID do documento a converter, juntamente com a página e indicação de thumbnail, e o GUID do utilizador que efetuou o pedido de conversão. Nenhuma destas informações é capaz de por si só por em causa a segurança do sistema. No entanto, o atacante poderá optar por efetuar um ataque por repetição e repetir o envio do mesmo exato pedido ao DCSLogic com o mesmo JWT. Apesar de o pedido conter GUIDs reais tanto do utilizador como do documento e ter sido feito para o endereço correto, ao chegar ao DCSLogic, este comparará o timestamp de expiração do token com o timestamp atual. Caso o timestamp atual seja superior ao de expiração, o JWT considera-se expirado e portanto o pedido é descartado. No entanto, caso o atacante seja capaz de replicar o pedido rápido o suficiente para que este seja tratado pelo DCSLogic antes de o JWT expirar, o pedido será tratado como se de um pedido válido se tratasse e portanto o atacante receberá a página do documento convertida. É portanto de vital importância que o intervalo de tempo entre a geração e expiração do JWT seja o mais pequeno possível para impedir a replicação dos pedidos mas ao mesmo tempo permitir que não expirem antes de serem processados.

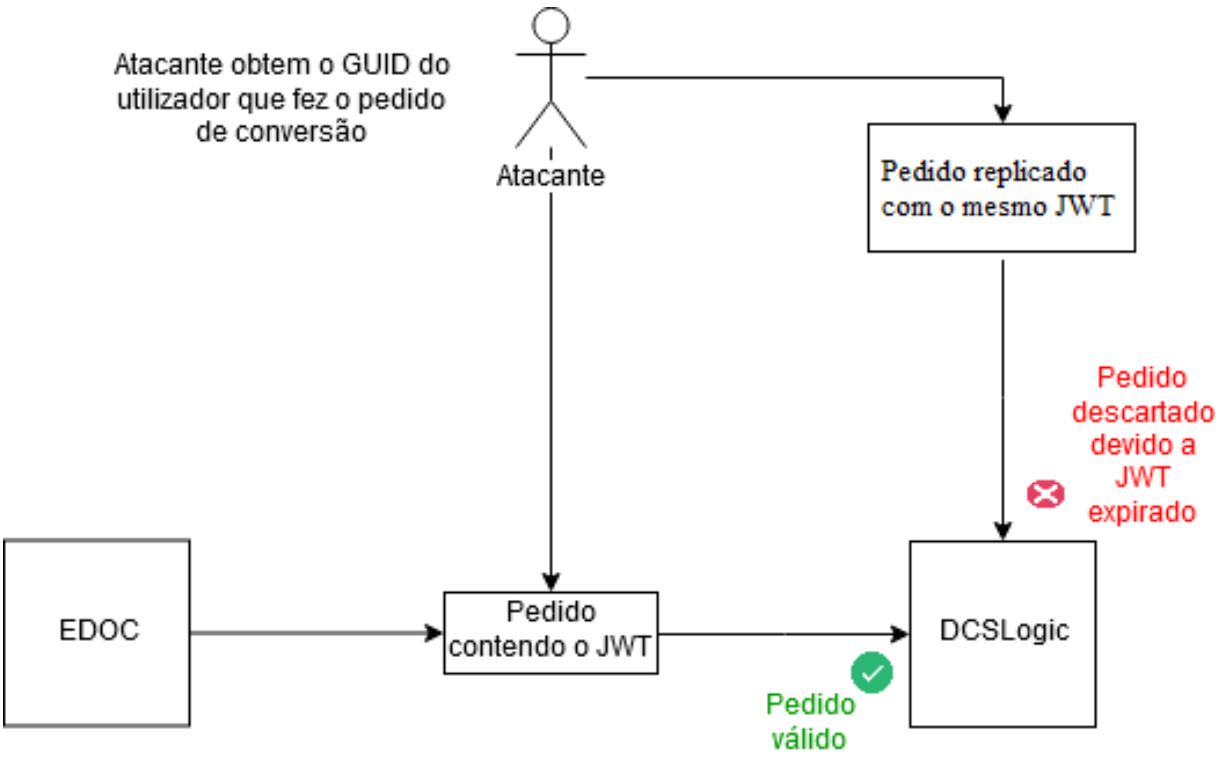

Figura 7. Repetição do Pedido com o mesmo JWT

Um outro possível ataque que poderia ser efetuado caso um atacante interecetasse um pedido de conversão é o representado na figura seguinte. Ao tentar efetuar o ataque apresentado na figura anterior e apercebendo-se do facto de o JWT estar já expirado, o atacante poderá optar por tentar efetuar o mesmo pedido mas reconstruindo o JWT. Para isso, obtém o GUID do utilizador através do JWT do pedido intercetado e gera um novo token com um timestamp de criação e expiração atualizados. No entanto, quando o DCSLogic recebe o pedido contendo este token e tenta obter o GUID do utilizador para verificar o seu acesso ao documento, verifica, comparando com a assinatura contida no JWT, que o mesmo foi adulterado visto que o payload não contém os mesmo dados que o payload da assinatura. Desta forma, o pedido é descartado impedindo portanto este tipo de ataque.

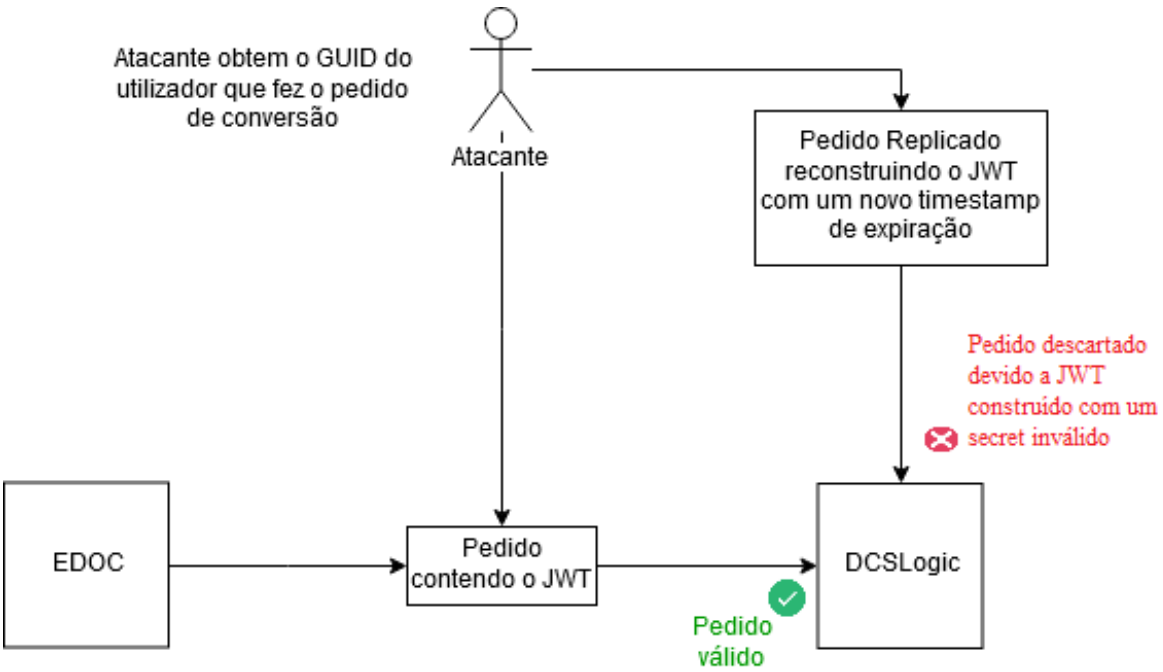

Figura 8. Repetição do pedido reconstruindo o JWT com um novo timestamp de expiração

5.2.2 Implementação DCSConverter

A implementação no Azure do DCSConverter foi em tudo semelhante ao DCSLogic. Foi criado um container que alberga o código desta aplicação que posteriormente foi implantado no Azure por via da funcionalidade Web App for Containers. O objetivo deste módulo é exclusivamente o processo de conversão. Assim, o pedido de conversão contém 3 parâmetros: o caminho completo juntamente com o nome do ficheiro, o número da página a converter e uma flag que indica se é um thumbnail. Ao receber o pedido de conversão, o DCSConverter tenta obter o documento que está guardado num repositório Azure no caminho indicado no parâmetro. Em caso de sucesso e com base nos metadados do documento que contêm a extensão do ficheiro transferido, a página pedida do documento é convertida para o formato JPG utilizando a biblioteca Aspose. Esta biblioteca permite vários processos de manipulação e conversão de imagens, sendo que era já utilizada na versão antiga deste componente. Após efetuada a conversão, o ficheiro é codificado em base-64 e enviada de volta para o DCSLogic, sob o formato JSON.

Desta forma, o DCSConverter é completamente stateless, isto é, cada pedido é tratado de forma completamente independente dos anteriores e dos posteriores, não retendo qualquer tipo de informação entre pedidos.

O DCSConverter e o DCSLogic completam assim a arquitetura total do conversor de documentos. Estes dois sub-módulos possuem, no entanto, complexidades completamente distintas, sendo que a grande parte do processamento do pedido - verificação de acesso do utilizador ao documento e gestão da cache - é efetuada pelo DCSLogic, simplificando a implementação do DCSConverter que efetua apenas uma tarefa de conversão para o formato JPG sem qualquer outra responsabilidade.

5.3 Helper Service

Como descrito no capítulo 4, dadas as características do Helper Service e as suas funcionalidades, este módulo será migrado para a Cloud do Azure utilizando o serviço Azure Functions. É necessário, para cada uma destas Functions, associar um trigger que despoletará a execução da mesma. Estes triggers podem ser de diversos tipos para a eventualidade de, por exemplo, se efetuar uma determinada ação quando um pedido HTTP é recebido ou quando um determinado ficheiro guardado num repositório de ficheiros Azure é modificado ou criado. No âmbito deste módulo específico, o trigger utilizado foi o Timer Trigger que permite, através de uma expressão CRON, definir um período de repetição de execução desta função. Uma expressão CRON consiste numa string de 7 campos que representam diferentes detalhes relativos à periodicidade desejada, sendo eles, segundos, minutos, horas, dia do mês, mês, dia da semana e ano. A expressão pode ser configurada de forma a regular a execução do Helper Service.

Relativamente à implementação, na génese do Helper Service, este foi implementado utilizando o modelo de Providers. Este modelo, que se assemelha ao padrão de desenho Factory, permitia ter diferentes módulos capazes de tratar os alarmes recolhidos. De acordo com o tipo de alarme, era decidido em tempo de execução qual o provider a utilizar para, por exemplo, alertar um utilizador por e-mail ou por SMS de um determinado prazo. No entanto, tal padrão já não se verifica hoje em dia devido às alterações que foram sendo feitas a este componente. Assim, para além da migração do código do Helper Service para uma Azure Function, foi também efetuada a remoção deste modelo de providers. Desta feita, as tarefas do Helper Service passam simplesmente a ser a chamada de um webservice presente no Servidor Aplicacional do EDOCLINK para que este recolha, na base de dados, os alarmes que precisam de ser tratados e, de seguida, a iteração ao longo dessa lista de alarmes para que, de acordo com o seu tipo, seja chamada um outro webservice também existente no Servidor Aplicacional do EDOCLINK responsável pelo tratamento do alarme, seja ele o envio de e-mail, notificação interna ou a geração de relatórios.

Outras das alterações efetuadas a este componente, foi a mudança da framework utilizada. As Azure Functions possuem duas versões distintas: a versão 1.x que permitem a execução de funções cuja framework alvo seja a .NET Framework e a versão 2.x cuja framework alvo deve ser a .NET Core 2.0 . A .NET Core é a framework sucessora da .NET Framework e permite a criação de aplicações multi-plataforma. Por defeito, novas Azure Functions deverão ser implementadas utilizando a versão 2.x. visto que é nesta versão que as melhorias e alterações vão sendo feitas, sendo a versão 1.x mantida apenas por motivos de compatibilidade com aplicações que ainda não suportem .NET Core. Visto que o Helper Service utilizava .NET Framework, e no sentido de modernizar este componente para que tire partido das melhorias de performance trazidas pelo .NET Core, decidiu-se efetuar a migração para esta framework.

A migração do Helper Service de .NET Framework para .NET Core 2.0 implica que algumas bibliotecas podem não ser suportadas. Este problema surgiu no momento de utilização de uma biblioteca existente na versão atual do HelperService que facilita a chamada dos web services SOAP presentes no Servidor Aplicacional. Dada a inexistência de alguns packages presentes nesta biblioteca no .NET Core, a solução passou pela criação manual do cliente SOAP responsável pela chamada do web service, eliminando a necessidade de utilização desta biblioteca.

Ultrapassado este problema, a migração ocorreu sem nenhum outro percalço de maior, pelo que existe atualmente uma versão do Helper Service implantada como uma Azure Function que trata os alarmes existentes, tendo a versão antiga sido desativada.

6 Avaliação dos Resultados

Neste capítulo, o objetivo principal é analisar de que forma é que a migração dos componentes para serviços Cloud impactou a performance dos mesmos. No que respeita ao Document Converter Service e considerando que o mesmo foi separado em dois módulos distintos para separar a gestão da cache e o processo de conversão propriamente dito, é necessário perceber se é possível tirar partido de uma das maiores vantagens da Cloud que é a facilidade de escalamento. Em primeiro lugar, definam-se as questões que precisam de ser respondidas:

1. Que tipo de escalamento é mais apropriado para cada um dos componentes do serviço de conversão?
2. De que forma é que a aplicação do tipo ou tipos de escalamento mais apropriados a cada um dos componentes resulta numa melhoria de performance? De que forma podemos quantificá-la?
3. Comparando com os resultados do serviço atual, é possível concluir que a migração deste componente se traduziu num upgrade face ao estado atual do mesmo ?

Em primeiro lugar, relativamente ao módulo que contém toda a lógica associada ao Document Converter Service, devido ao facto de este ser responsável por toda a cache das imagens já convertidas, a criação de novas instâncias deste componente implicaria a criação de um mecanismo capaz de partilhar o estado atual da cache por todas as instâncias num dado momento. Neste sentido, optou-se por não escalar horizontalmente.

Relativamente ao outro módulo - DCSConverter - responsável pela conversão efetiva dos ficheiros para imagens, devido à forma como este está construído, não é necessária a manutenção de estado entre diferentes pedidos de conversão. Desta forma, qualquer pedido de conversão é independente dos anteriores e posteriores pelo que se estudará tanto o escalamento horizontal como o vertical deste módulo por forma a averiguar a forma como a sua performance se altera.

O serviço App Service onde ambos os módulos do Document Converter Service estão implantados possui várias categorias que se distinguem pelas suas diferentes capacidades de processamento, memória e número máximo de instâncias. Estas categorias podem ser alteradas a qualquer momento de acordo com as necessidades e estão compartimentadas em diferentes planos: Gratuito e Partilhado, Básico, Standard e Premium. Apenas os planos Standard e Premium oferecem a capacidade de escalamento automático, sendo que o plano Standard foi concebido com vista a albergar aplicações em produção e o Premium existe como uma versão melhorada do plano Standard. Assim sendo, foi decidido utilizar a categoria S1 para ambos os módulos, que corresponde à categoria com menos capacidades do plano Standard. Os cenários de teste seguintes foram efetuados com base nesta configuração à exceção dos último cenário onde se avalia o impacto do escalamento vertical do DCSConverter no tempo de processamento de pedidos de conversão, em que a categoria foi alterada para a P1v2, a primeira categoria do plano Premium que oferece capacidades de processamento e memória melhoradas.

De seguida apresenta-se uma tabela de comparação entre as categorias utilizadas do App Service e da categoria utilizada na VM.

Tabela 1. Comparação entre VM e Categorias App Service

	Capacidade de Processamento	Memória
VM F2S	2 vCPU	4GB
App Service S1	100 ACU	1.75GB
App Service P1V2	210 ACU	3.5GB

A comparação da capacidade de processamento entre VMs e aplicações em App Service no Azure não é direta. Para as diferentes categorias App Service, a capacidade de processamento é definida em ACU - Azure Compute Units. Segundo a tabela de equivalência entre ACU e categorias de VM no Azure [18] [19], 100 ACU correspondem a uma VM da categoria A e 210 ACU a uma VM da categoria Dv2. A categoria F2S em que foi colocada a VM contendo o servidor aplicacional do EDOCLINK juntamente com a base de dados, é equivalente em termos de capacidade de processamento à categoria Dv2 [20] e, por conseguinte, equivalente em termos de capacidade de processamento à categoria P1V2 do App Service. Assim, para efeitos de comparação de resultados, a categoria S1 de um App Service é inferior à categoria F2S da VM. A categoria P1V2 do App Service é a que mais se assemelha em termos de capacidade de processamento à categoria F2S pertencente à VM que contém a versão atual do EDOCLINK, ainda que possua ligeiramente menos memória. Esta correspondência será especialmente útil no momento de comparação de performance entre o Document Converter Service atual e a nova versão cloudificada.

Os cenários de teste seguintes dividem-se em 3 categorias com diferentes objetivos:

1. Avaliação do impacto da cache do DCSLogic nos tempos de processamento de pedidos
2. Avaliação do impacto do escalamento horizontal do DCSConverter no tempo de processamento de pedidos
3. Avaliação da performance da versão atual face à versão Cloud com escalamento horizontal automático e escalamento vertical

Os pedidos efetuados nestes cenários de teste visam a conversão da primeira página de um documento DOCX com um tamanho de 1MB em tamanho original. Nos cenários pertencentes às duas primeira categorias, foram efetuados 20 pedidos de conversão. O total de 20 pedidos foi definido por forma a que o número de pedidos total fosse superior ao número máximo de instâncias do DCSConverter (10). Assim, será possível verificar se os pedidos são distribuídos por cada uma das instâncias ativas. No que toca ao formato do ficheiro, a decisão recaiu num DOCX pelo facto de este ser um tipo de ficheiro bastante comum. O ficheiro utilizado com um tamanho total de 1MB possui 5 páginas com texto, gráficos, imagens e tabelas para que se assemelhe a um ficheiro que possa existir num ambiente real.

6.1 DCSLogic

Como referido na secção de implementação, o DCSLogic é responsável pela gestão dos acessos do utilizador ao documento que pretende aceder e pela gestão da cache das páginas de documentos já convertidos. A

introdução da cache de documentos teve como objetivo melhorar a performance deste componente no tratamento de pedidos de conversão que tenham sido efetuados recentemente. A avaliação da performance deste componente prende-se então com a medição das melhorias de tempo de tratamento de pedidos de conversão quando utilizada a cache face a um cenário em que a cache esteja inativa.

6.1.1 Impacto da introdução de cache no tempo de processamento de pedidos

Para quantificar o impacto da cache no tempo de processamento de pedidos de conversão, foram criados dois cenários de teste.

Cenário 1 - Cache de documentos desativada

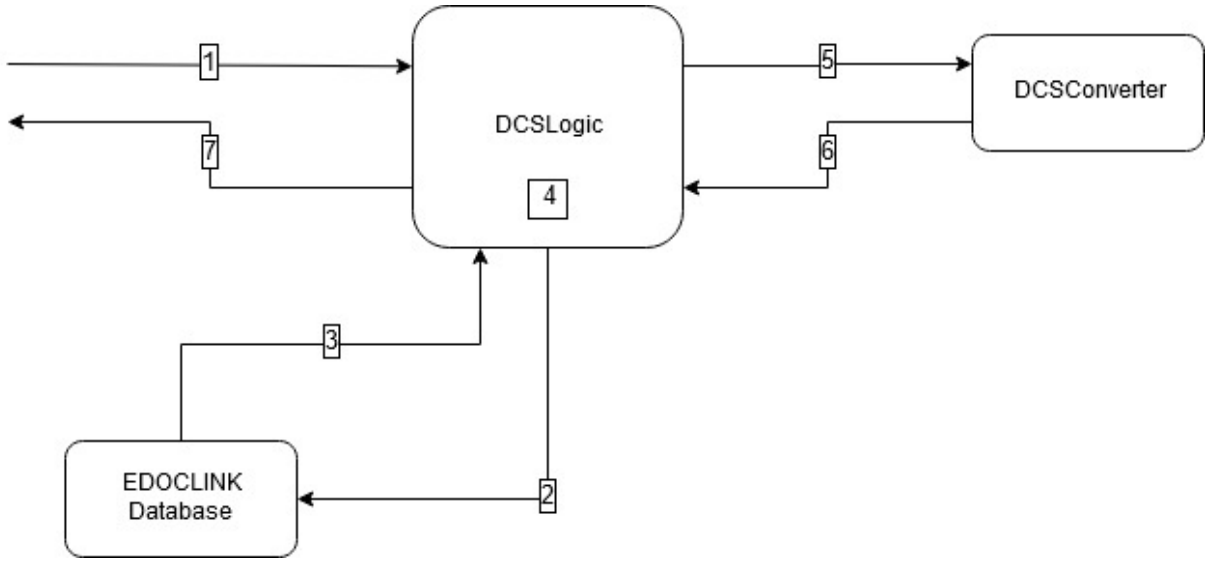

Figura 9. Primeiro cenário de teste ao DCSLogic

No primeiro cenário de teste, foram criados 20 pedidos de conversão para documentos cuja página ainda não tenha sido convertida como representado na figura seguinte. Estes 20 pedidos são lançados em rajada no início da execução do teste (1) e recebidos pelo DCSLogic. O DCSLogic valida o acesso do utilizador ao documento em questão através da base de dados do EDOCLINK (2 e 3) e procura em cache a existência da imagem já convertida para a página do documento em questão (4). Após verificar que ainda não foi efetuada uma conversão para aquela página em específico daquele documento, o DCSLogic encaminha o pedido para o DCSConverter. O balanceador de carga existente no App Service no qual o DCSConverter foi implantado responsabilizar-se-á por atribuir o pedido a uma das 10 instâncias ativas e devolver a imagem convertida em base-64 (5 e 6). O DCSLogic recebe esta reposta, guarda a imagem na cache e responde ao pedido inicial (7).

Os tempos apresentados para este cenário representam a diferença entre o instante no qual o pedido recebeu uma resposta (7) e o instante em que foi lançado (1).

A análise dos resultados mostra que os 20 pedidos demoram um total de 19 segundos a serem processados. As 10 instâncias ativas do DCSConverter neste cenário permitem que sejam tratados no máximo 10 pedidos em simultâneo. À medida que uma instância se liberta da conversão de um documento, começa a tratar de um pedido em espera até não restar nenhum dos 20 iniciais. Este comportamento permite que os primeiros 10 pedidos sejam tratados em 8,5 segundos. Dada a inexistência de imagens em cache neste

cenário, o tempo que o pedido demora a ser convertido no DCSConverter (5 e 6) representa a grande maioria de tempo total de processamento do pedido. Assim, os tempos aqui recolhidos servirão como base de comparação representativos de um cenário hipotético de não existência da cache que implicariam sempre um processo de conversão de um documento para uma imagem em cada pedido recebido.

Cenário 2 - Cache de documentos ativa

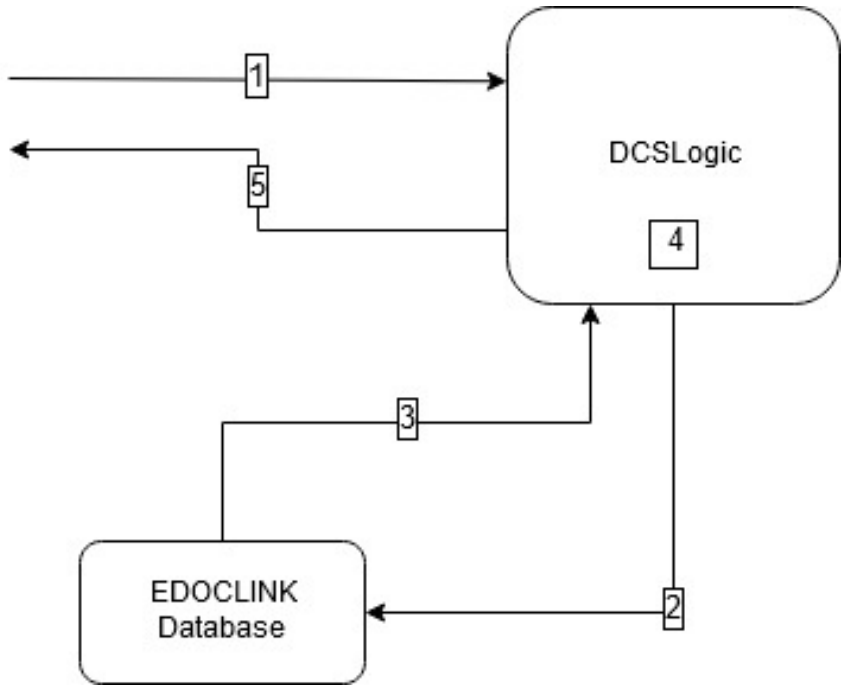

Figura 10. Segundo cenário de teste ao DCSLogic

No segundo cenário de teste criado, as condições de teste mantiveram-se. São enviados 20 pedidos em rajada (1) para a conversão de uma página de um documento. O DCSLogic verifica o acesso do utilizador ao documento (2 e 3) e a existência da imagem em questão em cache (4). E é aqui que reside a grande diferença para o cenário anterior: neste cenário a imagem pedida já se encontra em cache pelo que não é necessário o reencaminhamento do pedido para o DCSConverter, sendo o pedido imediatamente respondido pelo DCSLogic (5).

Os resultados obtidos corroboram a expectativa de que a existência do ficheiro já guardado em cache, diminui significativamente o tempo total de processamento de um pedido. A tabela abaixo representa a média de tempo de processamento de um pedido e o tempo total de processamento dos 20 pedidos em ambos os cenários. Como se pode verificar, a introdução da cache, e consequente eliminação da necessidade de enviar o pedido até ao DCSConverter, permite uma melhoria de aproximadamente 95% no tempo de processamento de um pedido.

Tabela 2. Métricas Obtidas para Avaliação do Impacto da Cache

	Tempo Total (ms)	Média de cada pedido (ms)
Cenário 1 - Sem Cache	19050	9778
Cenário 2 - Com Cache	845	427
Percentagem de Melhoria	95	95

6.2 DCSConverter

A análise feita ao DCSLogic na subsecção anterior permitiu concluir que a introdução da cache de documentos já convertidos se traduziu numa melhoria de quase 95% no tempo de processamento face a pedidos que ainda tenham de ser reencaminhados para o DCSConverter. Apesar desta melhoria, é necessário garantir que no caso de existir um pico de carga (vários pedidos concorrentes não contidos em cache que precisam de ser convertidos), o DCSConverter é capaz de escalar e tratar vários pedidos em simultâneo por forma a não degradar a performance. O objetivo desta subsecção é avaliar a forma como a performance do DCSConverter se altera com diferentes níveis de carga e, com base nos resultados obtidos, criar uma configuração que permite escalar a performance deste componente de acordo com a carga introduzida.

Por forma a quantificar as alterações na performance deste componente sob diferentes condições de teste, foram criados vários cenários de teste. De realçar que o App Service onde o DCSConverter está implantado se mantém na categoria S1 para estes cenários de teste.

6.2.1 Impacto do aumento do número de instâncias na performance do DCSConverter

Nesta fase, o objetivo passa por concluir e quantificar o impacto que o aumento do número de instâncias do DCSConverter aptas a converter documentos concorrentemente tem no tempo de processamento de um pedido de conversão.

Cenário 1 - Apenas 1 instância do DCSConverter ativa

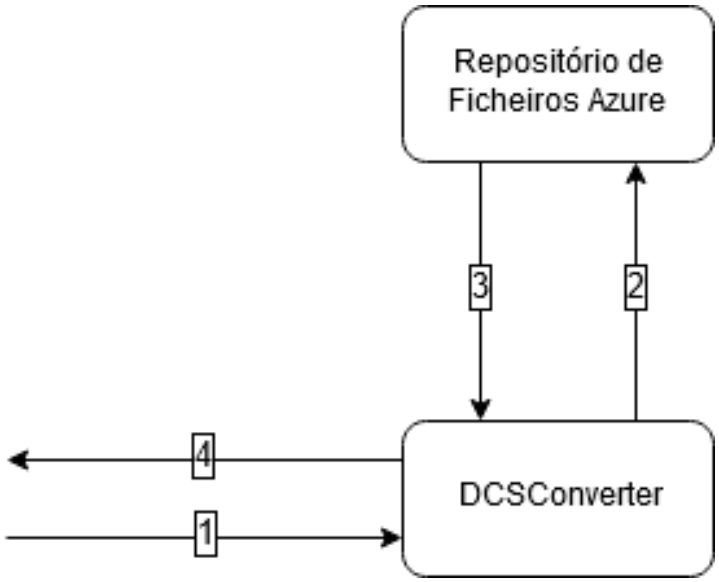

Figura 11. Primeiro cenário de teste ao DCSConverter

Neste primeiro cenário de teste, foram enviados 20 pedidos em rajada ao DCSConverter (1). Dado que existe apenas uma instância ativa do DCSConverter disponível para tratar pedidos de conversão, estes ficam em espera e são tratados de forma consecutiva. Ao receber um pedido, o DCSConverter trata de aceder ao Binary File Repository no Azure e obtém o documento localizado no caminho que foi indicado com parâmetro do pedido (2 e 3), converte-o para o formato JPG, codifica-o em base-64 e responde ao pedido (4).

Os resultados obtidos neste cenário confirmam o tratamento consecutivo dos 20 pedidos. De notar que a duração total do tratamento dos 20 pedidos foi de 77 segundos sendo que a média de tratamento de cada pedido se fixou nos 42 segundos. Um outro indicador que deve também ser tido em conta é o tempo total de tratamento dos 10 primeiros pedidos que foi de 36 segundos, sendo que para este conjunto dos primeiros 10 pedidos a média de tempo de processamento fixou em 22 segundos. Estes dados relativos aos primeiros 10 pedidos serão especialmente úteis na comparação com o cenário seguinte em que o número de instâncias disponíveis do DCSConverter será aumentado.

Cenário 2 - 10 instâncias do DCSConverter ativas

Tendo como base os resultados obtidos no cenário anterior, o objetivo deste cenário é quantificar as alterações na performance do DCSConverter causadas pelo aumento do número de instâncias disponíveis para efetuar conversões de documentos. A expectativa de resultados deste cenário é de que o aumento do número de instâncias permita o tratamento de pedidos em paralelo e consequentemente diminua o tempo de processamento do total dos pedidos.

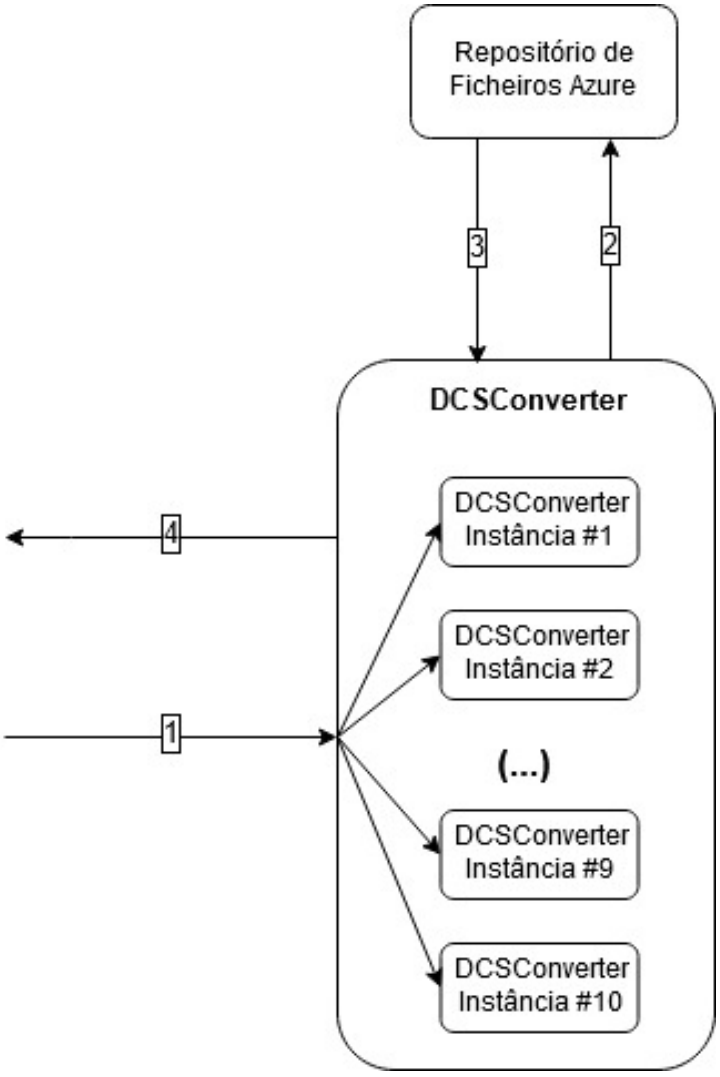

Figura 12. Primeiro cenário de teste ao DCSConverter

No entanto, os primeiros resultados obtidos neste cenário de teste eram extremamente similares ao do cenário anterior que tinha apenas 1 instância ativa. Estes resultados causaram alguma suspeição visto que era expectável que o mecanismo de gestão de carga presente por defeito no Azure conduzisse cada um dos pedidos para uma instância livre naquele dado instante. Após análise das respostas aos pedidos efetuados, concluiu-se que uma única instância estava a tratar todos os pedidos de conversão, comportamento este que não era o desejável. A causa deste componente estava numa opção, por defeito ativa, em que a

Azure Web App adiciona um cookie ARRAffinity aos pedidos aos quais responde. Este cookie tem como função transportar informação relativa à instância que tratou o pedido de forma a que o balanceador de carga direcione um próximo pedido da mesma fonte para a mesma instância. Esta função é especialmente útil em aplicações que pretendam ser stateful i.e. que pretendam manter o estado do sistema entre pedidos sucessivos do mesmo cliente. Uma vez que no caso específico do DCSConverter o objetivo seria a statelessness do sistema, ou seja, cada pedido diferente ser tratado de forma completamente independente, esta opção foi desativada. Os resultados deste cenário de teste foram obtidos após esta alteração, com a garantia que os diferentes pedidos estão a ser corretamente distribuídos pelas instâncias livres.

Existem algumas conclusões que podem ser retiradas partindo dos resultados obtidos neste cenário. Em primeiro lugar, é de realçar que o tempo total de processamento sofre uma queda abrupta passando de cerca de 77 segundos para 16 segundos.

Tabela 3. Métricas Obtidas para Avaliação do Impacto do Número de Instâncias na Performance do DCSConverter

	Tempo Total (ms)	Média de cada pedido (ms)
Cenário 1 - 1 Instância Ativa	77210	42630
Cenário 2 - 10 Instâncias Ativas	16401	7351
Percentagem de Melhoria	78	82

Para além dos resultados expostos na tabela acima, é importante também realçar que, no caso do cenário 2, os 10 primeiros pedidos são enviados para cada uma das 10 instâncias ativas. A capacidade de processamento concorrente destes pedidos permite diminuir significativamente os tempos de resposta a este grupo de primeiros 10 pedidos, sendo que estes resultados foram expostos na tabela abaixo.

Tabela 4. Métricas Obtidas para Avaliação do Impacto do Número de Instâncias na Performance do DCSConverter - Primeiros 20 Pedidos

	Tempo Total (ms)	Média de cada pedido (ms)
Cenário 1 - 1 Instância Ativa	36777	22125
Cenário 2 - 10 Instâncias Ativas	5204	4520
Percentagem de Melhoria	85	80

Enquanto que no primeiro cenário, o facto de existir apenas uma instância ativa implica o tratamento consecutivo de cada pedido, as 10 instâncias existentes no cenário 2 permitem o tratamento concorrente destes pedidos, fazendo com que o tempo total de processamento diminua de 36777ms para 5204ms. Também a média de processamento de cada pedido sofre uma queda abrupta de 22125ms para 4520ms. É

possível então concluir que a existência de mais instâncias ativas num dado momento permite aumentar a performance deste componente.

6.2.2 Configuração do Escalamento Automático do DCSConverter

Concluído então que o aumento do número de instâncias do DCSConverter diminui significativamente o tempo de resposta a um dado pedido, é necessário garantir que o número de instâncias ativas se altera dinamicamente de acordo com a carga introduzida no sistema. Para maximizar a performance deste componente, o ideal seria manter ativas a qualquer momento o número máximo de instâncias possível. Ainda assim, manter um número elevado de instâncias ativas a todo o momento implicaria uma desperdício enorme de recursos nos intervalos de tempo em que a carga introduzida no sistema não o justificasse. Assim sendo, o objetivo será potenciar umas das maiores vantagens da Cloud que é o escalamento automático on-demand. Neste cenário de teste específico, para iniciar o estudo de qual será a configuração ideal para este caso em específico, foi configurado o escalamento automático do DCSConverter com base na métrica da média de percentagem de utilização do CPU acima dos 60%, com base em valores recolhidos nos últimos 5 minutos. A cada 5 minutos o valor da percentagem média de utilização do CPU é calculado e caso atinja o limiar definido é adicionada uma nova instância do DCSConverter até um máximo de 10 instâncias. As condições de teste mantêm-se: DCSConverter na categoria S1 e 20 pedidos em rajada de conversão de um DOCX com 1MB.

Os resultados obtidos neste cenário de teste são extremamente semelhantes ao cenário com apenas 1 instância. Após análise dos dados fornecidos pelo Azure, foi possível verificar que em nenhum momento ao longo da execução dos 20 pedidos a condição-limite foi atingida e, por conseguinte, nenhuma instância foi adicionada, ficando apenas 1 instância a tratar todos os pedidos.

Na perspetiva de encontrar a melhor configuração de escalamento automático e respetiva condição-limite, foram criados cenários de teste em tudo semelhantes a este, excetuando a métrica utilizada para o aumento do número de instâncias. No entanto, em nenhum dos cenários testados acima a condição-limite foi atingida, facto que despoletou uma análise no sentido de perceber por que razão o mecanismo de escalamento automático não estaria a despoletar o aumento do número de instâncias.Este comportamento deveu-se ao facto de o total tempo de execução dos 20 pedidos ser bastante inferior ao intervalo de 5 minutos que foi definido para cálculo da média de percentagem do CPU utilizada. Este intervalo é o mínimo permitido pelo Azure pelo que picos de carga com duração inferior a este valor dificilmente despoleterão a condição-limite, fazendo com que o número de instâncias não seja aumentado impedindo a melhoria da performance.

Com base no fracasso que foram estes cenários de teste onde se pretendia tirar partido do escalamento automático providenciado pelo Azure, concluiu-se que de forma a poder estudar os seus reais benefícios seria necessário aumentar o número de pedidos por forma a aumentar o tempo total de execução do teste, e assim, permitir o correto aumento ou decréscimo do número de instâncias com base na carga introduzida num dado momento.

6.3 Comparação da versão antiga atual e da versão Cloud

Com o intuito de concluir se a migração do Document Converter Service para a Cloud se traduz numa melhoria de performance, nesta secção proceder-se-á à comparação destas duas versões. A secção anterior mostrou que quando o pico de carga é demasiado estreito no tempo, i.e. o número de pedidos em paralelo é reduzido, o escalamento automático não é despoletado a tempo de ter impacto no tratamento dos pedidos. Dado que a elasticidade do sistema à carga é uma das maiores vantagens da Cloud, a comparação com o sistema atual tem de ser feita sob condições de carga suficientemente grandes para tirar partido desta mesma elasticidade. Para avaliação de performance, os testes desta secção implicam a introdução de carga no Document Converter Service no sentido de manter uma média constante de 20 pedidos concorrentes de conversão durante 30 minutos. Estes pedido de conversão foram parametrizados para um documento DOCX de 1 Megabyte em tamanho original. Nestes cenários de teste, a cache foi desligada para que a conversão do documento seja efetuada repetidamente em cada pedido.

6.3.1 Versão Cloud

Para a configuração do escalamento automático do DCSConverter, é necessário escolher que métrica será utilizada para analisar se a carga é suficientemente grande e consequentemente aumentar o número de instâncias. A função deste componente é a conversão de documentos para o formato JPG, função esta que bastante dependente da capacidade do CPU do sistema. Assim sendo, optou-se por utilizar a métrica baseada na média da percentagem de utilização do CPU na configuração do escalamento automático. Sempre que esta métrica, calculada pelo Azure em intervalos de 5 minutos, esteja acima dos 60% o aumento do número de instâncias é despoletado. Inicialmente, o número de instâncias será de apenas 1. Para que existam 4 patamares de performance distintos, cada vez que o mecanismo for despoletado, adicionará 3 instâncias novas até um máximo de 10. Assim sendo, o DCSConverter poderá ter ativas num dado momento 1, 4, 7 ou 10 instâncias.

A condição de escalamento automático foi atingida ao minuto 1, aumentando para 4 instâncias, aos 7 minutos, aumentando para 7 instâncias, e aos 13 minutos, aumentando para o máximo de 10 instâncias. A estes aumentos do número de instâncias correspondem decrécimos progressivos do tempo de resposta destes pedidos. Este comportamento era expectável dado que os pedidos em espera têm um maior número de instâncias do DCSConverter para o qual podem ser encaminhados, aumentando a capacidade de processamento concorrente.

Repetiram-se os testes executando o mesmo cenário mas alterando a categoria do App Service onde o DCSConverter está implantado de S1 para P1V2. Esta alteração representará um aumento da capacidade de processamento para o dobro do cenário anterior. Mantendo todas as restantes configurações, o objetivo seria verificar se o aumento da capacidade de processamento inerente a cada instância do DCSConverter permitiria diminuir o tempo de conversão de um documento. Dado que a performance do processo de conversão é ditada pela capacidade do processador, é esperado que a duplicação das unidades de computação disponíveis do DCSConverter diminua o tempo médio de conversão.

Neste caso, a condição-limite para o aumento do número de instâncias foi atingido ao minuto 1, aumentando as instâncias de 1 para 4, aos 8 minutos, aumentando de 4 para 7 instâncias, e por fim aos 15 minutos, aumentando para o limite de 10 instâncias.

No entanto, a conclusão principal a retirar deste cenário de teste é a de que o aumento das capacidades de processamento das instâncias do DCSConverter diminui substancialmente o tempo necessário para efetuar uma conversão. Todos os diferentes patamares visíveis no gráfico da figura 19, associados aos diferentes números de instâncias do DCSConverter disponíveis para efetuar conversões sofreram uma diminuição significativa do tempo médio que leva a efetuar uma conversão, o que permite concluir que também o escalamento vertical deste módulo tem impacto na performance deste componente.

6.3.2 Versão Atual

Na subsecção anterior, foram analisados os resultados dos testes de carga efetuados ao novo Document Converter Service. Nesta fase, é necessário retirar as métricas do sistema atual para comparar de que forma este se compara com a versão migrada para a Cloud.

Para obtenção da performance do conversor atual, o mesmo cenário de teste foi aplicado de forma a tornar a comparação coerente com o que foi feito para o novo conversor. Durante 30 minutos, foi mantido um fluxo constante de 20 pedidos de conversão em paralelo para um documento no formato DOCX com 1 Megabyte. Sendo que o conversor atual não permite o escalamento da performance de acordo com a carga introduzida, é expectável que o tempo de processamento de uma conversão seja o mesmo ao longo do teste.

A figura acima representa o tempo de processamento de cada um dos pedidos ao longo dos 30 minutos de teste. Como era de esperar, a performance mantém-se inalterada ao longo do tempo total do teste dada a incapacidade de escalar a performance no estado atual do conversor.

Como foi referido na secção da Arquitetura Atual do EDOCLINK, o Document Converter Service atual converte todo o documento DOCX para formato PDF, no primeiro pedido a uma página do mesmo. Apesar deste comportamento melhorar a performance em conversões subsequentes, na primeira conversão a um dado documento, todo ele será convertido para PDF e só depois a página específica para JPG. Assim, para que as condições de teste fossem as iguais às utilizadas na nova versão do conversor em que a cache foi desligada e portanto cada pedido foi tratado como se de uma nova conversão se tratasse, optou-se por forçar a que a imagem JPG da página especificada fosse gerada novamente sob um nome diferente e assim analisar os tempos de um processo de conversão propriamente dito e não apenas um acesso a um ficheiro previamente guardado em cache, implicando assim primeiramente a conversão do documento por inteiro para o formato PDF.

Tabela 5. Comparação do Document Converter Service Atual Vs. Versão Cloud

	Total de pedidos tratados	Média de cada pedido(ms)
EDOC Atual	1314	27432
DCS sem escalamento vertical	3404	1-24000/4-18000/7-12000/10-8000
DCS com escalamento vertical	9007	1-11000/4-7000/7-4000/10-2900

Como verificado na tabela acima, existem várias melhorias causadas pela migração do Document Converter Service para a Cloud. Para além das melhorias associadas à introdução da cache, a possibilidade de introdução de escalamento horizontal automático, traduz-se numa melhoria significativa do tempo de processamento de cada pedido. A inexistência de escalabilidade no EDOCLINK atual torna o processo de conversão num período de carga bastante lento, sendo incapaz de se ajustar ao número de pedidos recebidos. Por outro lado, na versão do Document Converter Service cloudificada sem escalamento vertical, é de realçar que embora o tempo médio de cada pedido quando existe apenas uma instância do DCS-Converter ativa seja semelhante ao EDOCLINK atual, os tempos médios para os diferentes patamares do

número de instâncias vão diminuindo progressivamente, à medida que o número de instâncias aumenta. Por conseguinte, também o número de pedidos que o mesmo é capaz de tratar ao longo dos 30 minutos aumenta substancialmente.

Na versão do Document Converter Service cloudificada e com escalamento vertical as melhorias são ainda mais notórias. A capacidade de processamento do Document Converter Service neste cenário é equivalente ao da VM utilizada para implantar a versão atual do EDOCLINK como foi descrito anteriormente. No entanto, o número de pedidos tratados é praticamente sete vezes maior bem como os tempos médios de processamento de cada pedido de conversão que foram reduzidos a mais de metade do cenário sem escalamento vertical.

Apesar de não terem sido efetuados testes em que as capacidades do App Service onde o DCSConverter está implantado são ainda mais aumentadas, é de esperar que um aumento do número de instâncias máximo ou da capacidade de processamento aumentasse ainda mais a performance do mesmo, podendo mesmo diminuir ainda mais o tempo médio de tratamento de cada pedido bem como aumentar o número de pedidos tratados.

6.3.3 Análise de Custos

Em termos de custos associados à migração do Document Converter Service para a plataforma Azure, a utilização do App Service implica um custo dependente da categoria, número de instâncias e tempo de utilização [24]. Também as VM's diferem de preço consoante as suas capacidades. Tendo apenas em conta as categorias utilizadas, segue-se uma tabela de comparação:

Tabela 6. Comparação de custos dos recursos utilizados

Recurso	Preço por hora (em euros)
App Service Standard - S1	0.085
App Service Premium - P1v2	0.169
VM - F2S	0.1628

De notar que, no caso dos App Services, o preço por hora definido deve ser multiplicado pelo número de instâncias ativas. Dado que a nova versão do Document Converter Service possui um App Service onde está implantado o DCSLogic e outro onde está implantado o DCSConverter, é necessário somar ambos os custos. Para o DCSLogic, será metade do preço por hora da categoria S1, ou seja, 0.0425 euros, que terá ainda de somar ao preço da VM e do DCSConverter de acordo com o tempo que passou com cada um dos diferentes números de instâncias. Desta forma, tendo como base o teste de carga efetuado ao EDOCLINK atual e à versão cloudificada, segue-se uma tabela de comparação de custos tendo em conta o escalamento automático que foi feito ao novo Document Converter Service e os minutos passados com cada número de instâncias:

Tabela 7. Custo do teste de carga de 30 minutos

Versão	Custo Total em Euros	Custo por pedido tratado
EDOCLINK Atual	0.5 x 0.1628 = 0.0814	6.19×10^{-5}
DCS sem escalamento vertical	0.33575 + 0.0814 + 0.0425 = 0,459	1.34×10^{-4}
DCS com escalamento vertical	0.66755 + 0.0814 + 0.0425= 0,791	8.78×10^{-5}

Na versão cloudificada do Document Converter Service, conclui-se que apesar de o preço do teste total de 30 minutos ter aumentado substancialmente, a diferença para a versão cloudificada com escalamento vertical no preço por pedido processado é bastante pequena. Conclui-se também que o Document Converter Service sem escalamento vertical possui o maior custo por pedido processado, algo que deve ser tido em conta no momento de escolher a melhor categoria a utilizar. Apesar de o custo por pedido na versão com escalamento vertical ser ligeiramente superior, o Document Converter Service foi capaz de tratar 7 vezes mais pedidos no mesmo espaço de tempo.

7 Trabalho Futuro

O trabalho de migração do EDOCLINK para a Cloud efetuado ao longo deste projeto permitiu implantar uma versão totalmente funcional da aplicação em que todos os componentes estão albergados em serviços Cloud. Apesar de nem todos os objetivos terem sido cumpridos, nomeadamente a migração do Servidor Aplicacional do EDOCLINK para um container e da base de dados para o SQL Azure, os componentes satélite sofreram uma profunda remodelação para os compatibilizar com o serviço Cloud em que foram implantados.

Em relação à base de dados, o próximo passo seria a migração da mesma para SQL Azure dado que a versão mais recente de desenvolvimento do EDOCLINK à data deste relatório já o permite.

Após a migração da base de dados para a Cloud, é apenas necessário migrar o Servidor Aplicacional do EDOCLINK para que seja possível abandonar o uso de VMs na versão cloudificada. No sentido do objetivo inicial, a migração a realizar implicaria a criação de um container que o albergasse, juntamente com as suas dependências e bibliotecas, e a implantação deste container da mesma forma que foram implantados os containers do Document Converter Service. Desta forma, o processo de implantação de uma nova instância do EDOCLINK para um novo cliente seria bastante mais simplificado e permitiria ajustar as capacidades da mesma com o mínimo de esforço, de acordo com as necessidades a um dado momento. Em [21], são apresentadas algumas opções na migração de aplicações que utilizem .NET Framework para containers. Algumas das possíveis opções seriam a construção manual de um dockerfile em que se adicionasse uma a uma cada biblioteca e dependência do Servidor Aplicacional do EDOCLINK ou a utilização de uma ferramenta, o Image2Docker, que permite a criação de um dockerfile a partir de uma aplicação contida numa VM [22]. Apesar das opções apresentadas, é de esperar que esta seja a tarefa mais exigente dada a dimensão e complexidade do Servidor Aplicacional.

No que respeita ao Document Converter Service, apesar dos testes efetuados ao longo deste projeto terem permitido concluir que o escalamento automático tem de facto um impacto positivo na nova versão do mesmo, será útil no futuro complementá-los por forma a verificar de que forma é que o tamanho do

documento ou a extensão do documento a converter tem impacto nos tempos de processamento e se as melhorias de performance são da mesma ordem de grandeza que as obtidas nestes testes. Seria também útil perceber se as melhorias de performance são proporcionais ao aumento do número de instâncias ou se a partir de certo número de instâncias em paralelo se verificam alterações. Outro teste também útil seria verificar até que ponto de escalamento vertical das intâncias do DCSConverter é que se verifica um descréscimo do tempo necessário para efetuar uma conversão. Uma outra melhoria que poderia ser efetuada neste componente seria a adição de uma funcionalidade à cache atual no sentido de guardar também o número de páginas total de um documento de forma a que também nesses pedidos se pudesse evitar a comunicação com o DCSConverter. A nível da partilha do Secret utilizado no processo de hash do payload e header do JWT, deve ser utilizado o serviço Azure Key Vault [26]. Este serviço permite guardar e cifrar chaves de forma segura para permitir partilha das mesmas entre o Servidor Aplicacional e o DCSLogic.

8 Conclusões

A migração do EDOCLINK para a Cloud permitiu retirar várias ilações que poderão ser úteis tanto nesta mesma tarefa no futuro como para processos de migração de outras aplicações legacy. A existência de aplicações com tecnologias bastante antigas pode dificultar o processo na medida em que as plataformas Cloud, apesar de oferecerem serviços que as compatibilizam com estas tecnologias, se focam maioritariamente em tecnologias mais recentes e modernas. Aproveitar o processo de migração para operar melhorias no funcionamento dos módulos alvo da migração pode acabar por permitir não só que o mesmo corra em ambiente Cloud e beneficie das suas características como por modernizar as tecnologias que esse mesmo utilize. A maioria das necessidades das aplicações legacy possuem uma solução em ambiente Cloud, como se verificou para o caso do EDOCLINK. Utilizar os serviços mais adequados às características de cada módulo facilita o processo de migração dado que a maioria dos problemas encontrados é transversal à maioria das aplicações legacy. Em termos de performance, a Cloud permite que se tire partido da facilidade de escalamento das capacidades do ambiente. Remodelar os módulos alvo de forma a que isso seja possível pode revelar-se uma tarefa demorada mas que resulta em melhorias substanciais de performance face à versão legacy. Apesar das melhorias, é necessário ter em conta que as mesmas implicam, na maioria das vezes, custos adicionais. É responsabilidade da organização responsável pela aplicação gerir de que forma estas melhorias valem o aumento dos custos, devendo para isso ser feita uma análise dos mesmos face à versão anterior. Os objetivos que se focavam na migração de parte do EDOCLINK para um ambiente Cloud de forma a verificar de que forma era a performance afetada foram atingidos, tendo sido possível obter uma comparação entre ambas as versões. Foi também útil perceber de que forma é que é possível quantificar as alterações de custos da nova versão e assim quantificar a relação entre os ganhos de performance e os custos associados.

9 Bibliografia

Referências

1. A View of Cloud Computing By Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia Communications of the ACM, April 2010, Vol. 53 No. 4, Pages 50-58 https://doi.org/10.1145/1721654.1721672
2. Jamshidi, Pooyan & Ahmad, Aakash & Pahl, Claus. (2014). Cloud Migration Research: A Systematic Review. IEEE Transactions on Cloud Computing. 1. 142 - 157. https://doi.org/10.1109/TCC.2013.10.
3. Zhang, Qi & Cheng, Lu & Boutaba, R. (2010). Cloud Computing: State-of-the-art and Research Challenges. Journal of Internet Services and Applications. 1. 7-18. https://doi.org/10.1007/s13174-010-0007-6.
4. Peter Mell (NIST), Tim Grance (NIST), The NIST Definition of Cloud Computing Location (2011)
5. Simon Crosby, Xensource and David Brown, Sun Microsystems, 2006/2007 https://doi.org/1542-7730/06/1200
6. Michael Eder, Hypervisor- vs. Container-based Virtualization, Seminars FI / IITM WS 15/16, Network Architectures and Services, July 2016 https://doi.org/10.2313/NET-2016-07-1_01
7. Robert C. Seacord, Daniel Plakosh, Grace A. Lewis, :Modernizing Legacy Systems:Software Technologies, Engineering Processes, and Business Practices, 1stedition, pp.58,(2003)
8. S. Comella-Dorda, K. Wallnau, R. Seacord, and J. Robert. A survey of legacy system modernization approaches. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania
9. Chauhan, Muhammad Aufeef and Muhammad Ali Babar. "Towards Process Support for Migrating Applications to Cloud Computing." 2012 International Conference on Cloud and Service Computing (2012): 80-87.
10. Legacy Enterprise Systems Modernization: Five Ways of Responding to Market Forces, Souvik Roychoudhury, Cognizant 20-20 Insights, July 2015
11. Solanke, Vikas & Kulkarni, Gurudatt & Vishnu, Maske & Prashant, Kumbharkar. (2013). Private Vs Public Cloud. International Journal of Computer Science & Communication Networks. Vol 3(2). 79-83.
12. An Introduction to Docker and Analysis of its Performance,Babak Bashari Rad, Harrison John Bhatti, Mohammad Ahmadi,Asia Pacific University of Technology and Innovation,Technology Park Malaysia, Kuala Lumpur, Malaysia
13. Market Share Analysis: IaaS and IUS, Worldwide, Gartner, 2018.
14. Statista, Current and planned usage of public Cloud platform services running applications worldwide in 2018.
15. Amazon Web Services - https://aws.amazon.com/
16. Angular - https://angular.io/
17. Azure - https://azure.microsoft.com/
18. Azure VM Sizes - https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-memorydv2-series-11-15
19. Azure compute unit (ACU) - https://docs.microsoft.com/en-us/azure/virtual-machines/windows/acu
20. Azure VM F-Series - https://azure.microsoft.com/en-us/blog/f-series-vm-size/
21. Migração de aplicações em .NET Framework para containers - https://github.com/dotnet-architecture/eShopModernizing/wiki/02.-How-to-containerize-the-.NET-Framework-web-apps-with-Windows-Containers-and-Docker
22. Image2Docker - https://github.com/docker/communitytools-image2docker-win
23. Custos VM Azure - https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/
24. Custos App Service Azure - https://azure.microsoft.com/en-us/pricing/details/app-service/windows/

25. HMAC - https://www.ietf.org/rfc/rfc2104.txt
26. Azure Key Vault - https://azure.microsoft.com/pt-pt/services/key-vault/