Mestrado em Engenharia Informática
Estágio
Relatório Final

Cosmedesk SaaS: Módulo back office

Ivan Harasym
harasym@student.dei.uc.pt

Orientadores:
Engenheiro Alcides Marques (IPN)
Professor Filipe Araújo (DEI)

03 de Julho de 2017

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

O presente estágio tinha como o objetivo o desenvolvimento de um módulo (*back office*), para o controlo e a configuração das instâncias da aplicação "Cosmedesk" alojadas na *cloud* "Azure". Também foi necessário uma investigação com possíveis alterações sobre a arquitetura à aplicação "Cosmedesk", pois esta foi construida para um cliente e agora chegou a necessidade de disponibilizar esta para novos utilizadores. O "Cosmedesk" é uma plataforma de geração e gestão de relatórios *Safety Assessment Report* (SAR) e *Product Information File* (PIF). O SAR é um relatório de avaliação da segurança dos cosméticos, que contém todo o conteúdo sobre o cosmético em causa, os seus ingredientes, as pessoas que o fizeram, as pessoas que participaram na sua avaliação, entre outros elementos. O PIF é um relatório mais abrangente que o SAR, pois o PIF contém o SAR e mais informações relevantes do produto, como detalhes dos ingredientes, os testes realizados, entre outros.

O módulo (*back office)* serve principalmente para a gestão dos novos clientes que irão usar a aplicação "Cosmedesk", realizando a configuração automática das instâncias da aplicação para estes. Também, é útil na gestão de alguns dados, que por ventura podem ser publicados nas bases de dados dos clientes, tais como aplicações genéricas, aplicações especificas, ingredientes e bibliografia. Para uma melhor gestão é necessário a existência de monitorização, que é oferecida pelo módulo, possibilitando visualizar algumas informações consideradas úteis na tomada de decisões no negócio. Uma outra parte não menos importante, são as notificações para os clientes, com o objetivo de que cada um recebe a notificação se o seu produto não satisfaz as normas da legislação, isto pode ocorrer depois de uma alteração na legislação.

Palavras-Chave

"Azure", "Blob", "Cloud", "Multi-tenant", "NoSQL", "Particionamento de dados", "Software as a Service"

Agradecimentos

Aos meus orientadores, Engenheiro Alcides Marques e Professor Filipe Araujo, pelo acompanhamento e ajuda demonstrados em todas as etapas do estágio. Ao Pedro Sousa pelo enorme contributo na minha formação e desafios oportunos. Aos restantes colegas de equipa: Hugo Almeida, Miguel Freitas, Hugo Amaro e Bruno Caceiro tal como restantes membros de Instituto Pedro Nunes por toda a amizade, companheirismo e disponibilidade para ajudar. Tal como todos os meus professores que ajudaram na minha formação. Aos meus amigos com quem sempre pude contar. Aos meus familiares, à minha mãe e à minha namorada pelo apoio continuo, perante bons e maus momentos, durante este periodo.

Índice

Lista de Figuras

Lista de Tabelas

Tabela de acrónimos

Acrónimo	Descrição
LIS	Laboratório de Informática e Sistemas
IPN	Instituto Pedro Nunes
MEI	Mestrado em Engenharia Informática
SaaS	*Software as a Service* (Software como serviço)
SAR	*Safety Assessment Report* (Relatório de Avaliação de Segurança)
PIF	*Product Information File* (Arquivo de informações sobre o produto)
REST	*Representational State Transfer* (Transferência de Estado Representacional)
CRUD	*Operaçoes: Create, Read, Update and Delete* (criar, ler, alterar e apagar)
HTTPS	*Hyper Text Transfer Protocol* Secure (Transferência de Hiper Texto Seguro)
HTTP	*Hyper Text Transfer Protocol* (Transferência de Hiper Texto)
JSON	*JavaScript Object Notation*
SSD	*Solid State Drive* (Unidade de Estado Sólido)
HDD	*Hard Disc Drive* (Disco Rígido)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
NoSQL	*Not Only SQL* (Não só *SQL)*
DTU	*Database Transaction Units* (Unidades de transação de base de dados)
GB	*Gigabyte*
TB	*Terabyte*
RAM	*Random Access Memory* (Memória de acesso aleatório)
CPU	*Central Processing Unit* (Unidade central de processamento)

Tabela 1: Tabela de acrónimos

Capítulo 1
Introdução

O presente relatório apresenta o trabalho realizado pelo aluno Ivan Harasym, no âmbito da disciplina de "Dissertação/Estágio" do Mestrado da Engenharia Informática (MEI) da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

A instituição de acolhimento do estagiário é o Instituto Pedro Nunes (IPN), uma organização sem fins lucrativos localizada em Coimbra. O estagiário esteve integrado num dos laboratórios do desenvolvimento tecnológico, Laboratório de Informação e Sistemas (LIS).

1.1 Enquadramento

O estagiário deverá desenvolver um módulo *back office* para um dos clientes do IPN, a GPH que é uma consultora que presta uma vasta gama de serviços. Alguns destes serviços são:

- Serviços de apoio laboratorial, consultoria à indústria farmacêutica, cosmética, nutracêutica, química, alimentar e formação nestas áreas.
- Prestação de serviços de controlo e certificação de produtos agroalimentares.
- Prestação de serviços de laboratório para apoio ao diagnóstico e controlo de qualidade de produtos alimentares, visando a melhoria da segurança alimentar.
- Prestação de serviços de controlo, certificação e consultoria na área ambiental.
- Análises clínicas.

O sistema a ser implementado destina-se a gerir uma aplicação já existente, utilizada pela GPH, para a gestão dos relatórios PIF e SAR entre outros menos importantes. Ambos os relatórios, referidos anteriormente, são de extrema importância, porque pode ser inspecionado pelas autoridades legais. O relatório PIF contem toda a informação sobre o produto incluindo o relatório SAR, que apesar de ser um relatório independente também têm que ser incluído nos anexos do PIF. O relatório SAR tem toda a especificação do produto e testes de segurança que foi submetido e as avaliações obtida. Contem toda a informação de apoio, para que um profissional devidamente qualificado consiga avaliar o produto até ao nível do ingrediente, isto é, este relatório tem de conter toda a informação dos ingredientes do produto, o seu nível exposição, perfil tóxico, estrutura química, as características de exposição específicas das áreas em que será aplicado, para que classe de indivíduos se destina, no caso de serem crianças, os produtores são obrigados a terem cuidados especiais, ou pessoas mais idosas, que podem ter determinados problemas de saúde, ou outra classe de indivíduos.

1.2 Motivação

A temática de certificação dos produtos da indústria cosmética tem uma vasta importância, pois os produtos podem ter uma grande diversidade de químicos que têm de ser submetidas a uma série de provas rigorosas, para demonstrar que são aptos a serem utilizados por pessoas, não impondo nenhum risco de saúde às mesmas. Cada produto obrigatoriamente, pela legislação, tem que ter o relatório PIF.

Os relatórios contêm muita informação que é acompanhada com as imagens e podem conter muitos anexos, que podem ser necessários para provar alguma das experiências realizadas. Isto implica que os documentos sejam extensos, apesar de serem todos bastante semelhantes entre si. Estes documentos contêm o mesmo formato só mudam os conteúdos relativamente ao produto e seus constituintes. A pedido da GPH, o IPN desenvolveu uma aplicação que gera os relatórios de uma forma automática, sendo possível realizar ajustes finais manualmente. O IPN surpreendeu a consultora GPH ao conseguir um nível superior ao esperado, pois conseguiu aperfeiçoar a aplicação de forma a dispensar intervenção humana. Dada a qualidade da aplicação, a GPH pensou comercializar o produto. Como o produto foi pensado para um utilizador, a configuração é realizada manualmente. Assim surge a necessidade de automatizar o processo na inserção de novos utilizadores, eliminando assim a parte trabalhosa da configuração manual. Assim surgiu o problema na gestão das instâncias da aplicação "Cosmedesk", que está alojada numa *cloud* "Azure". Deste modo pensou-se em desenvolver um módulo *back office* que irá gerir as instâncias da aplicação, automatizando a criação de todo o ambiente para cada, poupando o tempo e os recursos.

1.3 Objetivos

O estagiário ficou responsável pelo desenvolvimento de um módulo *back office*, que irá integrar com a aplicação existente "Cosmedesk", de forma a que o módulo consiga gerir todas as instâncias da aplicação. Isto é, o módulo terá que criar o ambiente para o novo cliente, a criação de uma instância, criação de uma base de dados para o cliente, aperfeiçoar o ambiente com o perfil do cliente e disponibilizar um URL para o cliente poder trabalhar na aplicação "Cosmedesk". Isto tudo deve ser realizado o mais automaticamente possível e, informando o utilizador (administrador) do estado final.

O módulo deverá também permitir monitorizar o comportamento das instâncias na *cloud* e os recursos utilizados, fornecendo os dados de forma gráfica para os diferentes tipos de dados, no qual estes dados são importantes na tomada de decisões do administrador. Os dados podem corresponder à atividade dos utilizadores, quantidade dos relatórios criados num determinado período do tempo, quantidade dos relatórios pré-visualizados num período do tempo, alterações efetuadas nos relatórios durante um período do tempo, entre outros.

O módulo também será responsável por dar acesso às instâncias, ou seja, pela gestão das licenças, que podem ser de dois tipos:

- Número de produtos: é uma licença base, que terá o efeito numa determinada data e que permitirá ao cliente ter na sua conta um determinado número dos relatórios, uma vez que estes são de conservação obrigatória durante um certo período do tempo.

Este tipo da licença é acumulativa, ou seja, se o cliente pretender comprar duas licenças para o mesmo período do tempo o número dos produtos é acumulável e representa a sua soma.

- Número dos relatórios: é uma licença que tem por base a quantidade dos relatórios que o cliente pode gerar na plataforma, este tipo de licença também tem o seu período de validade.

Deve ser possível gerir as regras de legislação para a aplicação, ou seja, o módulo irá permitir adicionar, alterar e remover regras referentes às quantidades de certas substâncias químicas permitidas pela legislação. O que permite manter os utilizadores informados sobre se os seus produtos satisfazem as necessidades da legislação, no momento da introdução de novas regras pelo administrador do módulo. Esta parte deve ser implementada na aplicação "Cosmedesk" uma vez que neste momento não existe e o estagiário deve adicionar esta funcionalidade.

É importante também efetuar o estudo das *clouds* concorrentes da "Azure", pois temos de conhecer os concorrentes e até ter o plano de migração no caso da cloud atualmente escolhida, como prioritária, deixar de existir ou no funcionamento da mesma existirem as interrupções frequentes. Para permitir fornecer um serviço mais constante e sem interrupções para os clientes, de forma a manter uma boa relação e satisfazer as necessidades dos mesmos. O que torna viral pensar na arquitetura da aplicação "Cosmedesk", pois esta pode sofrer alterações na sua evolução e a integração com o projeto a desenvolver. O estudo efetuado neste sentido sempre ajudaria ter noções do que se pode esperar, consoante as necessidades dos consumidores, em termos da arquitetura e o hardware necessário que no dado caso são as máquinas da Azure.

Ao longo da construção da aplicação devem ser cumpridos os seguintes requisitos:

- Utilização das technologias Microsoft. É importante pois a equipa do desenvolvimento da aplicação "Cosmedesk" é especializada nestas technologias.
- Utilização da *cloud* "Azure", uma vez que a *cloud* já se encontra a ser utilizada, pertencendo à Microsoft, estando, por isso, fortemente integrada com as ferramentas de desenvolvimento da Microsoft.

Capítulo 2
Conceitos Relacionados

Este capítulo apresenta o estado em que se encontra a plataforma "Cosmedesk", apresentando a sua estrutura no momento inicial do estágio. De forma a melhorar a plataforma iremos refletir sobre a pesquisa bibliográfica efetuada, onde foram identificados os artigos científicos, páginas web, testes e outros tipos de documentos relacionados com os problemas abordados no estágio, nomeadamente:

- Escolha de uma arquitetura *multi-tenant* que consiga satisfazer as necessidades dos consumidores.
- Alojar a base de dados num sítio seguro e conveniente para o cliente.
- Escolher a melhor estrutura para uma boa performance do serviço alojado no Azure.
- Realização da monitorização.
- Ajustar a utilização da plataforma para cada utilizador.
- Poupar recursos.
- Particionamento de dados.

2.1 Cosmedesk

O "Cosmedesk" é uma aplicação pioneira de geração e gestão dos relatórios SAR e PIF, entre outros, para cosméticos. A aplicação já se encontra implementada e em funcionamento, havendo alguns problemas e/ou desvantagens na construção desta. Temos de considerar que a aplicação foi feita inicialmente para satisfazer as necessidades de um só cliente daí resultando alguns problemas. A aplicação pode chegar a ter um grande potencial de mercado se os clientes, tal como o cliente principal GPH, adotarem e iniciarem o uso intensivo da mesma. Se isso acontecer, há uma grande possibilidade de sucesso, o que implica preparação forte para escalar o serviço. Neste preciso momento existem mais de 5000 fabricantes em toda a Europa, que podem chegar a ser clientes diretos ou indiretos da aplicação, havendo também intermediários, tais como consultoras, que podem tratar dos relatórios dos clientes que utilizem a aplicação. Isto deverá acontecer porque a aplicação irá estender o seu alcance a outros pontos da Europa, nomeadamente França, Holanda, Inglaterra, entre outros.

2.1.1 Arquitetura

Como podemos verificar na Figura 1, representada em baixo, a arquitetura usada na plataforma principal é constituída por várias instâncias ligadas a uma máquina virtual no "Azure". Esta contém todos os componentes necessários para o bom funcionamento da aplicação, incluindo as bases de dados. Para cada nova instância é criada uma base de dados

na máquina virtual, que contém toda a informação necessária para o cliente poder satisfazer as suas necessidades.

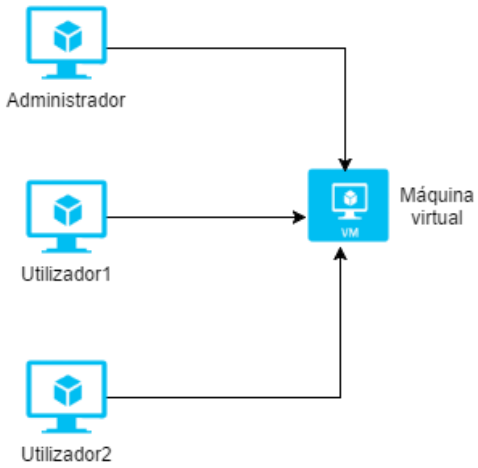


Figura 1: Arquitetura Cosmedesk no Azure

2.1.2 Visão nas máquinas

Como podemos verificar na Figura 2, apresentada em baixo, o cliente irá ter uma instância dentro da máquina e esta terá acesso a base de dados que também se encontra na máquina. Uma instância é uma *worker pool* que alimenta uma plataforma disponibilizada para um cliente. A instância também necessita do serviço do "Microsoft Office Word" para gerar os documentos. A instância está contida num servidor aplicacional que gere os domínios da aplicação do cliente. Este domínio é disponibilizado ao cliente, sendo-lhe também fornecidas as credenciais do administrador na aplicação "Cosmedesk", o administrador pode criar mais utilizadores com diferentes tipos de permissão. Consoante as permissões, os utilizadores poderão criar ou não relatórios, tipicamente não existem muitos utilizadores na aplicação, como o exemplo do cliente atual que só tem quatro utilizadores ativos. Normalmente são poucos os utilizadores que trabalham na aplicação, pois é um trabalho que não necessita de muitos utilizadores a trabalhar ao mesmo tempo, sendo este número muito baixo tipicamente até três utilizadores, a aplicação fica com reserva até cinco utilizadores por cliente na versão base depois de arranque.

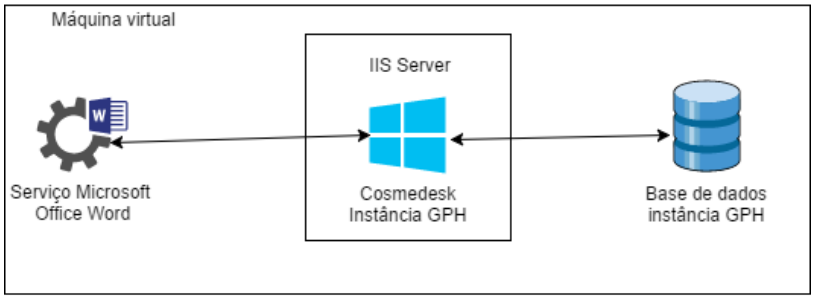


Figura 2: Visão 0 arquitetura inicial

2.1.3 Desvantagens da arquitetura atual

Aqui serão apresentados os possíveis problemas da arquitetura, identificada na Figura 1, e avaliadas possíveis soluções.

Arquitetura para um único cliente

Inicialmente a aplicação foi pensada para um cliente único, como o resultado, na altura do início da comercialização, surgiu a ideia de replicar algumas das partes da arquitetura para cada novo cliente que chega, para fornecer a segurança e o desempenho necessário para a boa satisfação das necessidades. O resultado podemos verificar na Figura 3, representada em baixo. Nesta arquitetura há lugar para várias instâncias e várias bases de dados dentro da máquina virtual, que partilham o serviço Microsoft Office Word tal como o servidor aplicacional. Sendo assim para cada cliente novo há necessidade de criar uma nova instância e uma base de dados, também é necessário adicionar o subdomínio do novo cliente. Este domínio é formado no formato "NomeCliente.Cosmedesk/funcionalidades". A aplicação existente utiliza uma base de dados por cliente, mas no presente relatório é efetuado o estudo da possibilidade de mudar esta para uma outra disposição das bases de dados, descrito mais a frente, um dos melhores concorrentes pelo estudo efetuado é a base de dados única com as *schemas* por cliente, que consegue garantir separação lógica dos dados, é uma mais valia pois os dados alojados na aplicação são considerados muito importantes para os clientes, os outros beneficios e defeitos podemos ver mais a frente, no estudo efetuado.

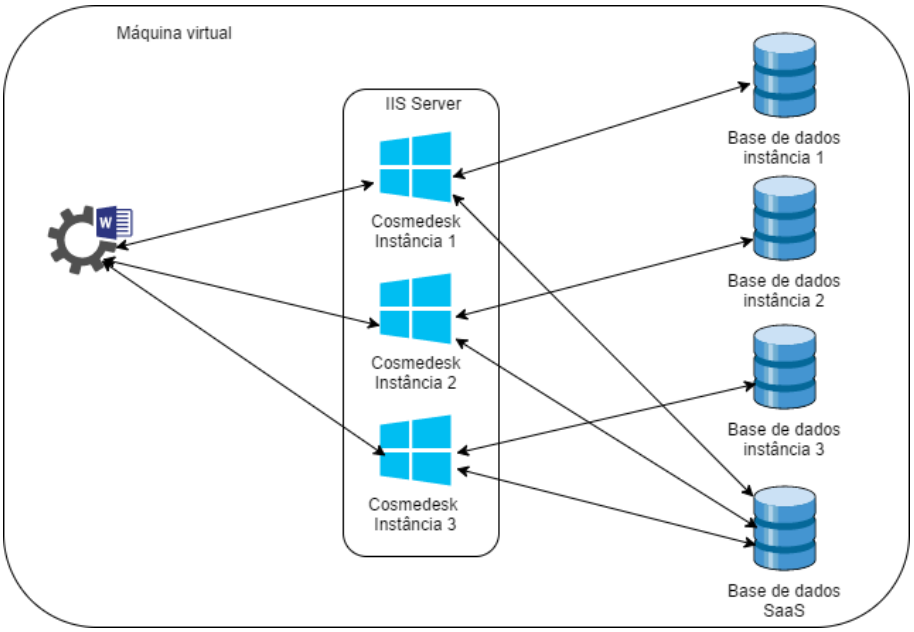


Figura 3: visão 1 da arquitetura atual

Escalabilidade

Na arquitetura não existe a possibilidade da escalabilidade automática. Para melhorar a máquina, adquirindo uma com as caraterísticas superiores à que está a assegurar o serviço, tem que se parar o serviço e iniciar este na nova máquina. Isto não é uma boa prática, porque ao impedir o acesso ao sistema, ainda que momentaneamente, pode-se incutir perdas aos utilizadores.

A possibilidade de resolver este defeito exige a construção de uma arquitetura, com um conjunto de máquinas escaláveis automaticamente. Esta arquitetura exige a existência de um *load balancer* para distribuir a carga entre as máquinas de uma forma uniforme. O *load balancer* servirá para gerir o tráfego entre várias máquinas de forma a melhorar o funcionamento do sistema.

Recursos

Para o sistema inicialmente implementado, estando o motor da base de dados numa máquina virtual, implica um maior gasto dos recursos da máquina virtual e também o processamento da informação de base de dados, ocupando assim o processador e RAM desnecessariamente.

A solução para o problema é colocar as bases de dados fora da máquina virtual e utilizar os serviços de dados da "Azure". Assim, as bases de dados ficam mais escaláveis, por ser um serviço da "Azure", este encarrega-se de toda a gestão da base de dados incluindo também a escalabilidade consoante as necessidades, assim a base de dados escala em poucos segundos, sem o utilizador se aperceber. Por outro lado, isto pode levar a outro problema, os custos, pois o sistema escala automaticamente. Isto é particularmente grave se tivermos muitos utilizadores. Como cada um destes irá ter uma base de dados só para ele, isto implica a adesão a muitas bases de dados do "Azure" e como todas elas têm um custo associado, o valor cresce em função dos clientes que temos.

A solução para este problema, pode passar pela utilização de um conjunto de bases de dados (*SQL database elastic pool*) que partilham os mesmos recursos, assim continuamos a ter bases de dados por cliente, mas partilhamos os recursos destas bases de dados. Este conjunto está limitado a um certo número de processamento que são *Database Throughput Units* (DTUs) que variam consoante as opções disponíveis, dez dos 50DTUs e até 3000DTUs, para todas as bases de dados existentes. Assim ultrapassamos o custo de ter várias bases de dados e temos um custo fixo por um conjunto de bases de dados. Este caso só é útil na existência de muitos clientes, ou seja, quando o custo das bases de dados individuais ultrapassa o custo de um conjunto de bases de dados.

Um outro problema, mais grave, que afeta de forma significativa a quantidade de memória RAM necessária, que foi verificado pela equipa do IPN, diz respeito à colocação do servidor aplicacional na máquina. Como o servidor necessita de recursos para o bom funcionamento e para conseguir responder a um número grande de clientes, por omissão ocupa uma fatia dos recursos muito significativa.

Para diminuir os recursos ocupados, uma das possibilidades é usar o servidor aplicacional da "Azure" que é o "web app". Este serviço permite colocar uma aplicação *online*. A parte mais

difícil irá ser a integração deste servidor com o "Microsoft Office Word", pois existe a necessidade de conseguir criar os relatórios na aplicação de forma automática nos formatos ".docx" e ".pdf", para isso é necessária uma integração com o "Microsoft Office Word".

2.2 Disposição da base de dados em arquitetura *multi-tenant*

A *Multi-tenant* é uma arquitetura de um serviço que satisfaz pedidos dos múltiplos clientes, onde cada um é um "*tenant*". Os *tenants* podem mudar partes da aplicação, como por exemplo as cores da aplicação ou regras do negócio, mas nunca o código. Com a arquitetura *multi-tenant* ao fazer alterações no sistema, estes são refletidos para todos os *tenants*. Por norma as arquiteturas *multi-tenant* são económicas, pois partilham recursos. Neste capítulo iremos abordar esta partilha de recursos, mais especificamente as possibilidades de partilha da base de dados por vários *tenants*.

Para satisfazer os objetivos do projeto iremos abordar três estruturas possíveis. Base de dados partilhada, em que todos os *tenants* partilham a mesma base de dados disponível. Base de dados partilhada, mas com um *schema* isolado para cada um dos *tenant,* i.e., neste caso existe uma base de dados, mas os *schemas* para lhe aceder são únicos para cada *tenant*. A última possibilidade é ter uma base de dados para cada *tenant,* o que significa que cada *tenant* irá ter uma base de dados única só para ele. Assim temos que escolher uma das três arquiteturas disponíveis, para a satisfação dos objetivos do projeto e dos clientes.

Iremos, agora, discutir com mais detalhe as diferentes disposições da base de dados perante a arquitetura *multi-tenant*.

2.2.1 Base de dados partilhada

A base de dados partilhada, é uma base de dados que irá servir todos os *tenants*. O armazenamento da informação para cada um dos *tenants* nas mesmas tabelas, indicando a identificação do *tenant* como a chave, nas tabelas correspondentes a *tenant* (cliente).

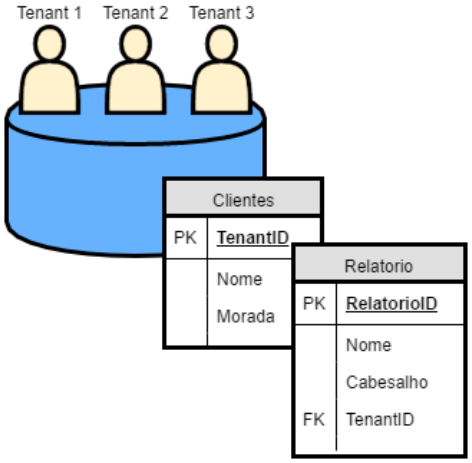


Figura 4: Base de dados partilhada

2.2.2 Base de dados partilhada e *schema* individual

A base de dados partilhada com *schema* individual para cada *tenant* armazena a informação de cada *tenant* numa estrutura diferente. Sempre que chega um novo cliente tem que se criar um novo *schema* na base de dados, para o *tenant* correspondente.

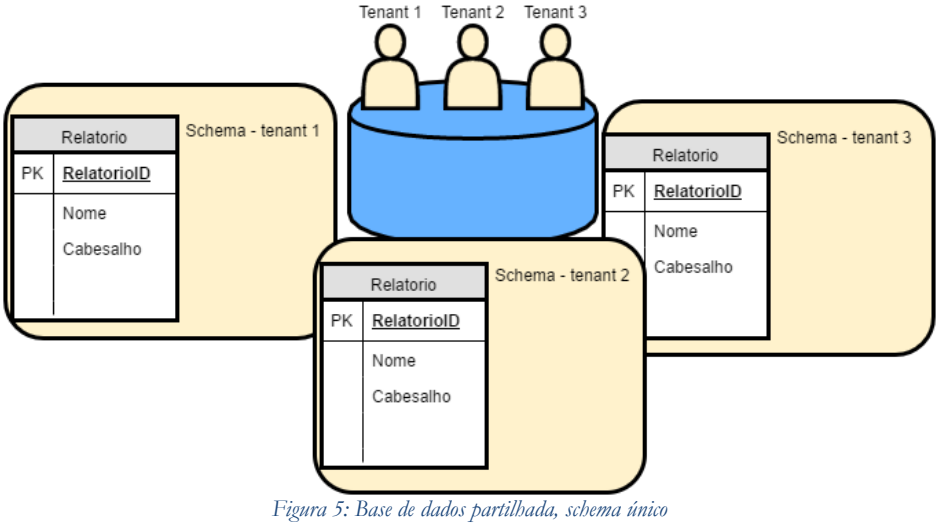


Figura 5: Base de dados partilhada, schema único

2.2.3 Base de dados por cliente

A base de dados isolada para cada *tenant*, armazena a informação de um e só um cliente numa base de dados. É uma maneira de oferecer mais performance a cada um dos *tenants* e melhorar a divisão entre os dados. Sempre que chega um novo cliente tem de se criar uma base de dados.

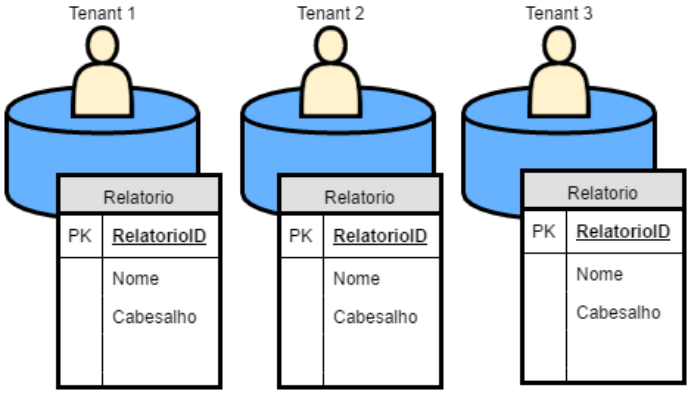


Figura 6: Base de dados por tenant

2.2.4 Isolamento

Este ponto é muito importante na maioria dos casos, porque todos queremos de ter os dados importantes bem protegidos. Este problema torna-se ainda mais importante no nosso contexto, porque os utilizadores da aplicação "Cosmedesk" de maneira nenhuma vão querer partilhar com alguém o conteúdo dos relatórios ou dados guardados na base de dados, por estes conterem toda a informação do produto e dos seus constituintes.

A estrutura com bases de dados separadas é a que melhor satisfaz o isolamento de dados, por não haver forma de um outro *tenant* aceder à base de dados do *tenant* vizinho, seja acidentalmente ou maliciosamente. Esta abordagem faz com que os dados sejam mais isolados e seguros, pois satisfaz as características da separação lógica e física.

Uma das outras possibilidades é ter arquitetura com a base de dados partilhada e *schema* individual. Estas fazem com que o grau de isolamento lógico não seja tão elevado como no anterior, mas mantém boas características de isolamento. Isto permite que cada *tenant* só aceda à sua *schema*, ou seja às suas tabelas. Isto é eficiente, apesar de ter um ponto negativo, caso o cliente insista que os dados dele são críticos e não acredite no isolamento especifico, ou seja se este prefere que os dados dele sejam arquivados numa base de dados separada.

Por fim temos a base de dados partilhada e *schema* partilhada, o que faz com que tudo esteja numa só base de dados e não exista nenhum isolamento lógico nem físico. Isto faz passar todo o processo de isolamento para a aplicação, dificultando assim o seu desenvolvimento ou aumentando a probabilidade de alguém poder aceder aos dados de outros *tenants,* seja acidentalmente ou maliciosamente.

2.2.5 Escalabilidade

A parte da escalabilidade é bastante importante pois, a aplicação, como descrito anteriormente, guarda todos os relatórios dos cosméticos no servidor, estes por vezes são "pesados", ocupam muito espaço por terem uma grande quantidade de informação armazenada que cresce de forma bastante rápida. Um cliente por um ano da utilização da aplicação, consegue gerar mais de 2 GB de dados na sua base de dados, sendo possível este obter mais procura, uma vez que a consultora GPH atua sobre este mercado da certificação dos documentos há pelo menos cinco anos, o que quer dizer que já tem a carteira estável dos clientes. O que faz pressupor que o valor para próximos anos não seja muito diferente podendo crescer um pouco. Sendo um dos principais objetivos, desta nova fase, é a angariação dos novos clientes, é importante garantir a escalabilidade e a capacidade de armazenamento de todos os documentos gerados. Considerando a arquitetura de base de dados por cliente, estamos um pouco mais estáveis porque a informação não cresce de forma acentuada e com a mesma rapidez, como nas restantes arquiteturas, já que só existe uma base de dados para todos os clientes, o que aumenta bastante o espaço ocupado com o próprio crescimento dos dados.

Há dois tipos de escalonamento "*scale-up*" ou "*scale-out*", em que o primeiro diz respeito há substituição de um componente por um outro melhor, que apresenta as caraterísticas que se consegue lidar com as necessidades do momento, e o segundo representa a criação de mais um componente com caraterísticas semelhantes e partilham a carga entre eles. Ou seja, no nosso caso considerando a arquitetura com base de dados única por tenant, podemos dizer que já estamos a fazer o "scale-out", mas não com bom balanceamento de carga. Porque alguns dos clientes podem estar mais ativos do que outros, que significa que podem precisar de mais processamento que os restantes, mas nada implica a não realização de "scale-up" para estes clientes. Assim, estes ficam bem servidos e até o processo é relativamente simples pois a base de dados só contém os dados de um tenant. Normalmente nem sentimos grandes dificuldades, pois as bases de dados são escaláveis consoante as necessidades. Nas restantes arquiteturas o processo do crescimento da base de dados é muito mais acentuado, ou seja, haverá mais necessidade em escalar, do que no caso anterior, o que implica um maior número de realização deste processo. Podem ser consideradas ambas as técnicas para a escalabilidade, mas tem que se ter mais atenção no uso da "scale-out", pois temos de tentar dividir os *tenants* de forma a que estejam equivalentes no nível do processamento necessário, isto para não criar discrepância na utilização das duas bases de dados.

2.2.6 Isolamento de dados

A segurança é sempre importante em aplicações web, mais ainda quando os dados a manter seguros incluem informações sensíveis, cujo secretismo pode determinar vantagens comerciais para os utilizadores da aplicação. Para alcançar uma segurança mais sofisticada de forma mais simples, é necessário disponibilizar uma base de dados para cada um dos clientes, assim os clientes sentiam-se mais confiantes em colocar informação relevante na aplicação, pois está garantido o total isolamento dos dados que vai dificultar e/ou diminuir estragos de um modo geral, por exemplo *SQLinjections*. A pior arquitetura seria considerar a base de dados e *schema* partilhado, o que faz com que não existirá algum particionamento lógico ou físico, ou seja a

segurança só poderá ser imposta por implementação adicional na aplicação. Por fim, considerando a arquitetura com base de dados partilhada e *schema* único por cliente, uma vez que existe a separação, existe uma separação lógica, mas não física, que apesar de garantir alguma segurança à partida, ainda que não comparável com a arquitetura de base de dados por cliente, o que significa que temos de implementar alguma segurança na aplicação, para além da disposta pela arquitetura. Ou seja, todas as arquiteturas necessitam de implementar uma segurança adicional na aplicação, mas esta implementação pode ser mais difícil ou mais fácil consoante a arquitetura escolhida.

2.2.7 Custo

Os custos da arquitetura podem variar muito, consoante a segurança e isolamento dos dados ou se as bases de dados ficam alojadas na *cloud* como o serviço, ou ficar dentro da máquina virtual. A segurança e isolamento são despesas que os clientes podem preferir pagar, mas as questões relativamente aos serviços já se referem ao próprio projeto e às necessidades da manutenção, pois as bases de dados da "Azure" são geridas pela *cloud* o que facilita a manutenção das bases de dados. Se as bases de dados ficarem como serviço da "Azure", podemos afirmar que a arquitetura com uma base de dados por cliente fica em desvantagem pois, temos que pagar por cada uma das bases de dados. As bases de dados de tipo "*basic*" oferecem pouco espaço de armazenamento, 2GB de disco, por valor de 4,20€ mensais, por isso será melhor utilizar as bases de dados de tipo "*Standart*" que já oferecem o espaço do armazenamento até 250GB por apenas 12,67€ mensais. Ou seja, se usarmos a arquitetura com as bases de dados por cliente temos de estar sujeitos a pagar o valor por cliente de 12,67€, que pode ser um número grande se tivermos grande aderência dos clientes, ou então pagar só 13,67€ utilizar uma das outras arquiteturas que são a base de dados para todos os clientes ou base de dados com *schema* por cliente. A base de dados com *schema* por cliente é favorável a base de dados para todos os clientes, pois oferece mais isolamento de dados.

Se as bases de dados permanecerem na máquina virtual os gastos serão nivelados, pois na versão "*Express*" o SQL Server é gratuito, mas limitado a 10GB de dados por base de dados, o que faz preferir a arquitetura de base de dados por cliente.

2.2.8 Conclusão

Baseado na informação discutida nos tópicos anteriores, podemos chegar à conclusão que há duas arquiteturas para o armazenamento da informação crítica, mais apropriadas. Estas, são arquiteturas com base de dados únicas por *tenant* e a base de dados partilhada, mas *schema* único por *tenant*. Isto porque os dados têm de ter algum isolamento entre si, a melhor que satisfaz é a base de dados por *tenant,* esta arquitetura também oferece melhor isolamento de dados e é mais escalável que a anterior. Estes são todos os benefícios da arquitetura da base de dados única por *tenant*. Esta arquitetura tem um ponto negativo que é custo das bases de dados, sendo o único ponto que podia levar a utilização da arquitetura com a base de dados partilhada

com *schemas* separados. Para melhor visão do que foi descrito anteriormente, a informação está exposta na seguinte tabela, que contém a classificação de cada uma das arquiteturas, a classificação é de 1 a 3 sendo o 1 o melhor e o 3 o pior resultado.

Arquitetura	Isolamento	Escalabilidade	Isolamento	Custo
Base de dados partilhada	3	2	3	1
Base de dados partilhada, *schema* individual	2	3	2	2
Base de dados individual	1	1	1	3

Tabela 2: Arquitetura multi-tenant

2.3 Particionamento de dados

Nesta seção iremos abordar o particionamento de dados, nomeadamente a parte da base de dados dos clientes. O que diz respeito aos dados dos clientes e aos respetivos relatórios gerados, como também aos dados destes mesmos relatórios. O problema que se verifica no crescimento intensivo de dados na base de dados, acontece por causa do armazenamento dos documentos diretamente na base de dados. Os documentos são armazenados em duas versões, versão ".doc" e versão ".pdf", o que implica ainda maior crescimento de dados. Como uma solução proponha-se o particionamento da informação, com isso podemos diminuir o espaço utilizado na base de dados e melhorar o desempenho do programa. Os documentos são raramente consultados, e até os dados dos mesmos documentos mantêm-se na base de dados, o que não tem nenhuma implicação no funcionamento da aplicação habitual.

2.3.1 Blob Storage

É um contentor que pode ser utilizado com um repositório da informação. Esse contentor pode chegar a um todo de 500TB de informação, é constituído por *blobs* que por sua vez podem conter até 50000 blocos de 4MB, ou seja 195GB da informação por *blob*. Este método é considerado com menor desperdício, já que é possível utilizar a ferramenta por preços razoavelmente baixos, para o uso deste serviço na "Azure". Também disponibiliza vários tipos de *blobs* tais como, página, bloco, ficheiro e fila. Estes são muito bons para armazenar os dados não estruturados ou binários. Também como podemos verificar pelos números em cima, podemos chegar a ter imenso espaço, ou seja, os *blobs* são muito escaláveis. Também não há custos de alocação do espaço pois só haverá custos com o tamanho dos dados armazenados. Têm vários tipos de redundância de dados, que por sua vez podem ou não melhorar os tempos de resposta e backup do armazenamento de dados. Pois existem quatro diferentes tipos de

redundância, esse número de redundâncias pode variar consoante o tipo do *blob***.** A redundância em *blobs* pode replicar os dados por vários *blobs* dentro do mesmo *data center*, mas também pode chegar a ser partilhado com os *data centers* mais próximos e cada um deles faz também a sua replicação dos dados.

2.3.2 Polybase

É uma base de dados NoSQL, integrada no "SQL Server 2016" que pode ser interligada com os serviços da "Azure", nomeadamente o "Hadoop" e "Blob-Storage". O polybase serve como uma ponte entre os dados guardados num repositório e o processador de dados. É otimizado para *data warehouses* e processamento de *querys* analíticas facilitando assim a junção de dados no "SQL Server". A integração com o "Hadoop" não é favorável neste caso, pois não há muita informação a transitar, não há consultas frequentes e não há necessidade de poder computacional. Assim não compensa o uso de uma ferramenta com tanto poder de processamento, logo podemos optar por uma outra possibilidade de integração com o "Blob-Storage", que é o mais adequado para guardar grandes quantidades de informação, esse pode ser dividido por todos os utilizadores uma vez que pode ter várias subpastas (*blobs*). Assim terá a separação de dados lógica, mas não física, pois os dados vão manter-se no mesmo repositório. Também pode ser interligado com uma solução de inteligência no negócio, ou seja, de monitorização de negocio, que pode facilitar a implementação de monitorização do sistema usando uma solução externa.

2.3.3 DocumentDB

É uma base de dados não SQL, com boa performance, alta disponibilidade, escalável e distribuída, o ponto mais fraco desta solução é o preço mais elevado, comparada com outras soluções disponíveis, tanto por espaço como por pedidos efetuados à base de dados. Pode ser interligada com vários componentes, tais como base de dados SQL, *hadoop*, *Service Bus* e *Worcker Role*, o serviço é desenhado para suportar os objetos JSON e o *JavaScript*, pois tem interface HTTP REST para operações CRUD (*Create, Read, Update, Delete*), também pode ser utilizado o SQL normal. O "DocumentDB" faz a replicação de dados por várias regiões na *cloud* para melhor suporte e disponibilidade de dados global, pode armazenar os dados num disco SSD.

2.3.4 Facilidade de desenvolvimento

Em termos de facilidade de desenvolvimento e manutenção o DocumentDB é mais fácil, pois é um sistema gerido pela Azure, o que quer dizer que não precisamos de nos preocupar com a gestão da base de dados, depois desta estar configurada inicialmente. O DocumentDB pode oferecer mais atrito na configuração principal, por ser um sistema mais complexo. Em termos de utilização das outras duas ferramentas, o Blob Storage oferece uma configuração simples, mas depois na utilização é relativamente mais difícil, comparando com o DocumentDB, mas podemos facilitar com a integração do Polybase. O Polybase irá fazer com que a utilização dos "Blob Storage" seja mais simples e semelhante à utilização de uma base de dados normal como por exemplo o DocumentDB.

Não interessa escalar os serviços em termos de rapidez de procura da informação, pois os dados a guardar no repositório são os dados referentes aos relatórios criados, ou seja, não há grande quantidade de registos que formam uma grande quantidade de informação, mas pelo contrário tendo poucos registos obtemos um número relativamente grande da informação. O que faz com que o tempo de procura de um determinado relatório, seja relativamente baixo, porque há poucos registos e estes vão ser procurados por chave primária, o que é o forte das bases de dados NoSQL.

2.3.5 Escalabilidade

A escalabilidade das ferramentas é um ponto muito importante, pois nunca sabemos se as necessidades futuras conseguirão ser satisfeitas da mesma forma, que são satisfeitas as necessidades presentes. Uma vez que a quantidade de dados gerados é aumentada constantemente, porque os clientes ao terem um grau maior de aprendizagem utilizam mais a aplicação, ou com o aparecimento de novos clientes. Ou seja, os dados crescem e vão continuar a crescer e se conseguimos escalar os componentes presentes temos a uma solução para longa viagem, o contrário pode dificultar a vida futura da ferramenta.

No caso com os *blob Storage* temos por defeito muito espaço para utilizar, ou seja, não se prevê que os documentos ultrapassem os 500TB de informação, mas de qualquer dos modos sempre pode ser feito *scale-out*, o que significa que em caso de necessidade de mais espaço podemos sempre criar mais um blob Storage para a informação ser alojada. Como o Polybase pode ser integrado com os "Blob Storage", servindo de ponte entre o motor da base de dados e o repositório. A Polybase tem características muito boas, para ser integrada com grandes quantidades de informação, especialmente com os "Blob Storage", uma vez que pode ser utilizada para *data warehouses*.

No caso de preferência pelo DocumentDB, este é muito escalável, pode ser escalável utilizando o método s*cale-up* até quantidade da informação não limitada, ou seja, não há limite da memória que o DocumentDB pode chegar a ter. O DocumentDB é inteiramente gerido pelo Azure, por isso não há necessidade de alguém se preocupar com a gestão da base de dados, ou seja, esta é escalada automaticamente.

2.3.6 Segurança

O Polybase não é responsável pela qualquer segurança da informação, pois só é a ponte entre o repositório da informação e o motor de base de dados, facilitando só a interação entre vários repositórios ao mesmo tempo, o que facilita o desenvolvimento, mas não traz benefícios de segurança. Enquanto os "Blob Storage" têm informação automaticamente replicada, pelo menos pelo próprio *data center* em que se encontra, são de acesso privado, podendo tornar esse público, mas neste caso isso não é desejado, pois a informação é privada e crítica para os clientes.

2.3.7 Custo

Os custos crescem, maioritariamente com o espaço, depois de se definir o que vai ser usado e quais as caraterísticas. O Polybase não tem custos adicionais, pois não armazena os dados, só é uma ponte entre o repositório e o motor da base de dados. Relativamente às outras duas ferramentas, o DocumentDB tem custos superiores aos custos da utilização dos "Blob Storage", apesar de o próprio "Blob Storage" ter uma certa variação do preço consoante a utilização de vários tipos de *blobs* e também dos vários tipos da redundância disponíveis. As diferenças de preços podem ser observadas na Tabela 4.

2.3.8 Conclusão

A informação que podemos extrair dos pontos anteriores, é que as necessidades em particionamento de dados do projeto podem ser satisfeitas com o baixo custo, utilizando os "Blob Storage", para uma melhor e mais simples utilização podemos integrar com o Polybase, uma vez que essa base de dados oferece uma utilização mais simples da ferramenta. O DocumentDB também satisfaz as necessidades, mas apresenta um custo relativamente superior e tem a arquitetura mais complexa, apesar de esta ser gerida pela "Azure", há necessidade de haver uma primeira configuração para uma boa utilização da ferramenta. Todas as ferramentas apresentam a possibilidade de escalar o serviço, mas o DocumentDB tem um custo associado a gastos de espaço superiores.

Os dados descritos anteriormente são classificados de 1 a 3, do melhor para o pior respetivamente.

	Desenvolvimento	Escalabilidade	Segurança	Custo
Blob Storage	2	3	3	1
Bolybase	3	2	2	2
DocumentDB	1	1	1	3

Tabela 3: Particionamento de dados

2.4 Soluções de *cloud*

Nesta seção iremos abordar os possíveis componentes das *clouds* que podem ser interligados, da forma a fornecer uma boa performance para todo o sistema. Analisando assim dois dos maiores concorrentes em *cloud* que são o "Azure" e "AWS", para ter alguma comparação do "Azure" e um dos seus concorrentes, apesar de um dos requisitos iniciais é a utilização da *cloud* "Azure".

2.4.1 Azure

Nesta seção iremos abordar os componentes existentes no "Azure", que podem servir na construção da arquitetura do sistema. Explorar o funcionamento dos componentes e como podem servir para o sistema e os seus benefícios, como também os preços que cada um pode chegar a ter.

Componentes disponíveis

No "Azure" temos vários componentes que nos podem ajudar a criar um sistema seguro, escalável e disponível. Para ter uma ideia do que pode ser usado na construção do sistema e também saber as caraterísticas dos mesmos, vão-se listar alguns desses componentes.

- **Máquina virtual** (1): é um componente bastante flexível no meio da virtualização, disponível com sistemas operativos "Linux" e "Windows", é escalável criando um conjunto das máquinas escalável, que permite escalar utilizando configurações relativamente a carga das máquinas.
- **Azure Load Balancer** (2)**:** é um gerenciador de tráfico, que irá distribuir o tráfego, que chega ao sistema para diferentes máquinas, melhorando a escalabilidade e aumenta o desempenho. Esta divisão será realizada na camada 4 da rede.

- **App Service** (3): é um servidor para aplicação web e mobile, disponível em ambiente "Windows" e "Linux", fornecendo alto desempenho, escalabilidade e segurança.
- **SQL Elastic Database** (4): é um conjunto de bases de dados relacionais, que permite reduzir os custos relativamente a uma grande quantidade de bases de dados independentes. Pois este componente gere várias bases de dados, disponibiliza uma quantidade certa do desempenho para o conjunto como um todo e não para cada uma das bases de dados.
- **SQL Database** (5): é uma base de dados relacional baseada no motor de SQL Server da "Microsoft", oferecendo desempenho, escalabilidade em poucos segundos e proteção dos dados.
- **Blob Storage** (6): é um repositório da informação escalável, durável e altamente disponível. Servem para guardar dados não estruturados, como um objeto, podendo estes ser em texto ou binários, como documento, música, imagem ou aplicação. Há diferentes tipos de blobs e também diferentes tipos de redundância, replicação de dados por diferentes zonas, para poderem se adequar ás necessidades de cada uma.
  Tipos de blobs:

  - **Block blob** (6): boa escolha para armazenamento de diferentes tipos dos ficheiros, este tipo dos *blobs* são otimizados para *streming* e *storing* objetos na *cloud*.
  - **Append blob** (6): são semelhantes a *Block blob*, mas são otimizados para operação escrita, sempre escrevem no final, são muito uteis para escrita de *logs*.
  - **Page blob** (6): suportam leituras e escritas aleatórias, podendo ter até um TB de tamanho.
- **Table Storage** (6): servem para guardar dados estruturados, utiliza o método de armazenamento chave valor, possibilitando o rápido acesso a grande quantidade da informação.
- **Queue Storage** (6): serve principalmente para comunicação entre componentes de serviços em nuvem, fornece mensagens confiáveis para o processamento de fluxo de trabalho.
- **File Storage** (6): serve para partilha dos ficheiros entre as máquinas e serviços da "Azure", podendo aceder a dados de arquivos em um compartilhamento por meio da API REST.
  Tipos de redundância disponível no *Blobs, Tables, Ques* ou *File Storages*:
  - ***Locally redundant storage*** (LRS) (6; 7): replica os dados, mantendo três copias de dados, num único *data center*, protegendo assim de falhas de *hardware* e não de falhas de um *data center*.
  - ***Zone-redundant storage*** (ZRS) (6; 7): replica dados três vezes seja dentro de uma única zona, mas nas instalações diferentes, ou até numa outra zona próxima, assim fornece melhore durabilidade do que o anterior.
  - ***Geo-redundant storage*** (GRS) (6; 7): replica os dados, criando três copias na região primaria e também cria outras três copias numa outra região próxima. Assim disponibiliza maior nível de durabilidade e protege das falhas no *data center* na região primária.
  - ***Read-access geo-redundant storage*** (RA-GRS) (6; 7): replica os dados, em duas diferentes zonas com diferentes localizações geográficas. Disponibilizando assim em caso de avaria numa das zonas a outra para leitura de dados.
- **DocumentDB** (8): é totalmente gerenciada a base de dados NoSQL, altamente escalável e distribuída, qua apresenta um SQL familiar de utilização com as bases de dados relacionais.

- **Application gateway** (9; 2): funciona no nível da aplicação, aplica regras de roteamento no nível da aplicação. Distribui a carga entre as máquinas virtuais e funciona com protocolos HTTP e HTTPS.

Escalabilidade

Todos os componentes são escaláveis, no entanto temos que ter certas preocupações para que isto aconteça de uma forma normal e sem prejudicar os clientes, ou seja, a configuração tem que satisfazer um certo nível de preocupação do cliente. Para isto é necessário que todas os componentes sejam escaláveis e estejam bem interligados, para que esta escalabilidade seja bem aproveitada. Como a escalabilidade dos componentes "Store blob", "SQL Database", "SQL Elastic Database" e "DocumentDB" é assegurado pela "Azure", a preocupação é com o ator principal dos componentes, que é a máquina virtual. A máquina virtual só é escalável se definirmos um conjunto de máquinas, e associando a este conjunto uma certa quantidade de máquinas existentes. Este conjunto pode escalar, utilizando o método *scale-out*, perante o horário que definimos ou perante os recursos utilizados pelas máquinas. Se a máquina escalar iremos obter mais uma máquina e assim conseguimos dividir os acessos pelas várias máquinas, existindo o componente do "Azure" chamado "Azure Load Balancer". Este irá distribuir a chegada do tráfego dos clientes por várias máquinas de uma forma uniforme.

Escalabilidade horária

A escalabilidade horária (10) corresponde à introdução de um "horário", que pretendemos a que seja executado e o número de máquinas que estejam a trabalhar. Para cada intervalo de tempo podemos definir a quantidade de máquinas a executar. Com isto podemos cobrir as horas de maior utilização com mais máquinas e horas de menor utilização com menos. Este método é útil se soubermos como o nosso serviço pode chegar a ser utilizado, ou seja, quando temos uma boa previsão da utilização do nosso serviço.

Escalabilidade pela utilização dos recursos

A escalabilidade pela utilização dos recursos corresponde a utilização de CPU (10), ou seja, se a utilização do CPU for superior a um valor definido, durante um tempo também definido irá iniciar uma nova máquina para dividir a carga entre estas. Acontece o mesmo para a diminuição das máquinas, ou seja, se a percentagem de CPU for inferior a um valor mínimo definido uma das máquinas termina a sua execução.

Custos

Nesta seção iremos observar os preços obtidos no dia 23 de Novembro de 2016, com ajuda da calculadora de preços dos componentes da "Azure" (11), para a comparação dos componentes disponibilizados, que podem chegar a ser usados na construção da solução.

Azure	Quantidade	Modelo	Características	Preço/hora	Preço mensal
Máquina virtual	1	D2	2 Cores, 7GB RAM, 100GB Disck	0,2270 €	168,77 €
App service	1	Basic	2 Cores 3,5GB RAM, 10GB Disck	0,1260 €	94,11 €
SQL Elastic Database	1	Basic	100 DTUs, 10 GB Storage	0,1700 €	125,48€
SQL Database	1	Basic	5 DTUs, 2 GB Storage	0,0057 €	4,20 €
Blob (LRS)	1	Standart	100000 Transactions, 10 GB Disck		0,21 €
Blob (ZRS)	1	Standart	100000 Transactions, 10 GB Disck		0,26 €
Blob (GRS)	1	Standart	100000 Transactions, 10 GB Disck		0,56 €
Blob (RA-GRS)	1	Standart	100000 Transactions, 10 GB Disck		0,61 €
Table and Queue (LRS)	1	Basic	100000 Transactions, 10 GB Disck		0,59 €
Table and Queue (GRS)	1	Basic	100000 Transactions, 10 GB Disck		0,80 €
Table and Queue (RA-GRS)	1	Basic	100000 Transactions, 10 GB Disck		1,01 €
File (LRS)	1	Basic	100000 Transactions, 10 GB Disck		0,70 €
File (GRS)	1	Basic	100000 Transactions, 10 GB Disck		0,88 €
DocumentDB	1		74400 Requests, 10 GB Disck	0,0070 €	7,13 €
Application gateway	1	small	1 GB data processing		17,57€

Tabela 4: Comparação do preço dos componentes da Azure

Cada um dos *blobs* contém vários tipos de redundância, representado dentro dos parêntesis. Os preços retirados da calculadora do "Azure", foram simulados com as caraterísticas semelhantes e no mesmo instante (23 de Novembro de 2016), entre o seu concorrente, para uma melhor comparação. Dos componentes listados na tabela só serão escolhidos os que são necessários para o bom funcionamento do sistema, que satisfaz os objetivos impostos.

2.4.2 AWS

Nesta seção iremos abordar os componentes existentes no "AWS", que podiam ter sido usados se a *cloud* a usar fosse esta. Também explorar o funcionamento dos componentes que poderiam ter sido usados, as funcionalidades que estes oferecem e os preços pelos quais estão disponíveis.

Componentes

Existem vários componentes que podem ser utilizados para a construção da arquitetura no "AWS", de modo a que esta seja segura, escalável e disponível. Alguns destes serão listados aqui, para termos uma ideia do que pode ser usado para construção de um sistema semelhante.

- **Máquina virtual (Windows, SQL Server)** (12): é uma máquina bastante flexível, disponível com sistemas operativos "Windows" e "Linux", damos a prioridade ao "Windows", porque este sistema operativo é necessário para o funcionamento do "Word", pode ser escalável definindo regras para a escalabilidade. Esta máquina em particular esta disponível com a integração de "SQL Server".
- **Máquina virtual (Windows)** (12): esta máquina é equivalente à anterior com uma única diferença, esta não é integrada com "SQL server", assim este serviço tem que ser requisitado à parte.
- **EBS** (13): repositório da informação para ser utilizado com a instâncias, oferece grande durabilidade e disponibilidade. A informação pode ser armazenada tanto em discos HDD ou SSD.
- **SQL Server** (14): é um motor de base de dados relacional, como o produto é da própria "Microsoft", oferece facilidade na configuração e a escalabilidade do produto.
- **DynamoDB** (15): é uma base de dados NoSQL altamente escalável, fornece bom desempenho podendo responder a qualquer quantidade de pedidos distribuindo os dados pelo número suficiente de servidores, para que estes não sejam perdidos.
- **Elastic Load Balancing** (16): é o gerenciador de tráfego, distribui o tráfego de entrada pelas várias instâncias, proporcionando assim alta disponibilidade, escalabilidade automática e forte segurança.

Custos

Nesta seção iremos abordar os custos que os componentes de "AWS" podem ter, as respetivas características, que foram obtidos pela calculadora de custos de "AWS" no dia 23 de Novembro de 2016.

AWS	Quantidade	Modelo	Características	Preço/hora	Preço mensal
Máquina virtual (Windows, SQL Server)	1	t2 large	2 Cores, 8GB RAM, 0 GB Disck	0,4500 €	329,40 €
Máquina virtual (Windows)	1	t2 large	2 Cores, 8GB RAM, 0 GB Disck	0,1500 €	109,80 €
EBS	1		500 GB Storage (HDD)		27,00 €
SQL Server	1	Express	20 GB Disck		39,34 €
DynamoDB	1	Standart	100000 Transaction, 100 GB Disck		20,27 €
Elastic Loud Balancing	1		1 GB / Mês		21,96 €

Tabela 5: Comparação dos preços dos componentes AWS

Observações: o preço de "DynamoDB" pode ser influenciado pela promoção existente na subscrição sobre o produto, o que contém os 25GB de *storage* e 2.5 milhões de pedidos efetuados à base de dados. Os preços retirados da calculadora do "AWS" (17), foram simulados com as caraterísticas semelhantes e no mesmo instante do tempo (23 de Novembro de 2016), entre os seus concorrentes, para melhor comparação entre componentes.

2.4.3 Conclusão

Como podemos verificar, temos os dois concorrentes muito similares em termos de componentes, pois existem em ambos os lados componentes que satisfazem as necessidades da aplicação, mas como um dos requisitos é trabalho com tecnologias Microsoft temos que optar pela a *cloud "*Azure". Esta também oferece mais diversidade no armazenamento da informação com os *blobs,* de diferentes tipos de redundância, também facilita a interligação com as tecnologias da "Microsoft", que já estão a ser utilizados na aplicação existente.

Capítulo 3
Método

O "Cosmedesk" é uma aplicação, existente e em execução antes do estágio iniciar, que ajuda a empresa GPH e os seus clientes a gerar os seus relatórios de uma forma mais automática.

O módulo proporciona um acompanhamento do gestor da aplicação "Cosmedesk" que irá ter umas certas vantagens na utilização da plataforma, pois o uso desta permitirá uma iteração mais automática com o "Azure" e não necessita de conhecimentos extra para manipular a *cloud* "Azure".

3.1 Levantamento e análise de requisitos

O levantamento dos requisitos será realizado de acordo com a técnica "KAOS" (18; 19). O "KAOS" é uma técnica baseada em objetivos que permite definir um objetivo principal e tentar ramificar o máximo possível. Ao ramificar os objetivos, obtemos folhas que serão os requisitos do módulo, cada folha terá um ator responsável pelo requisito. Esta técnica é bastante útil pois permite conhecemos as responsabilidades de cada ator para os seus requisitos. Mais detalhe da técnica como também a sua aplicação pode encontrar no documento em anexo, "Documento de requisitos", este fornece o detalho mais pormenorizado enquanto nesta secção só estão expostos os resultados finais.

3.1.1 Resultados da aplicação da técnica

Nesta seção iremos comentar os resultados obtidos através da aplicação da técnica "KAOS", estes resultados traduzem-se em requisitos ou responsabilidades identificadas para cada ator presente na árvore. Para facilitar a aplicação da técnica foi utilizada o programa "Objectiver 3.0" (20).

Responsabilidades do ator "Administrador"

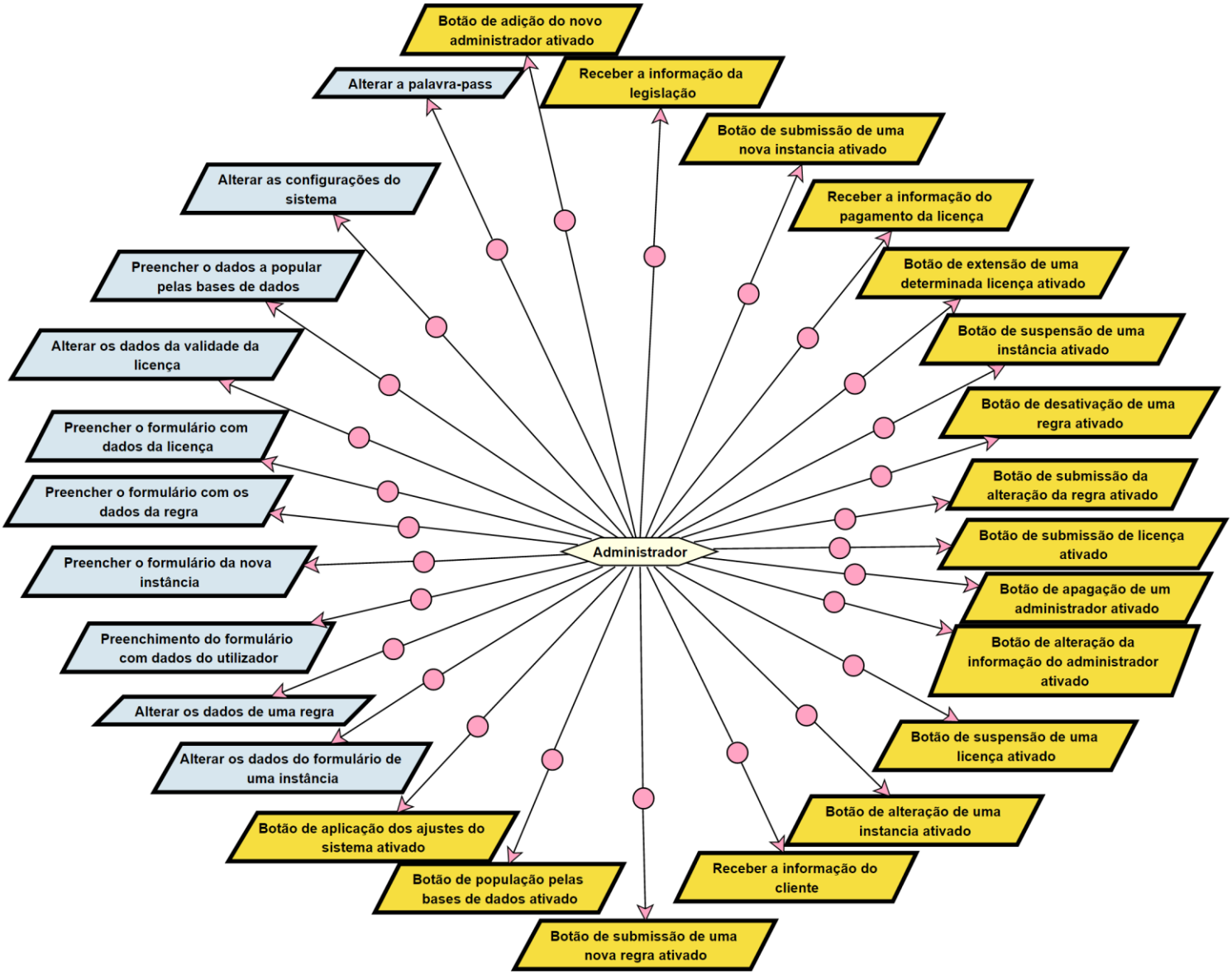


Figura 7: Responsabilidades do Administrador

Como podemos verificar, no diagrama da Figura 7,as responsabilidades do "Administrador" resumem-se em preenchimento e submissão dos formulários, a não ser que precisa de receber informação do exterior do programa. A informação do exterior corresponde a dados de novos clientes, também é importante ele verificar a legislação, pois através desta, são construídas regras para notificações, outro dado importante que o administrador tem que receber são os pagamentos das licenças antes de as criar.

3.2 Requisitos do ator "Sistema"

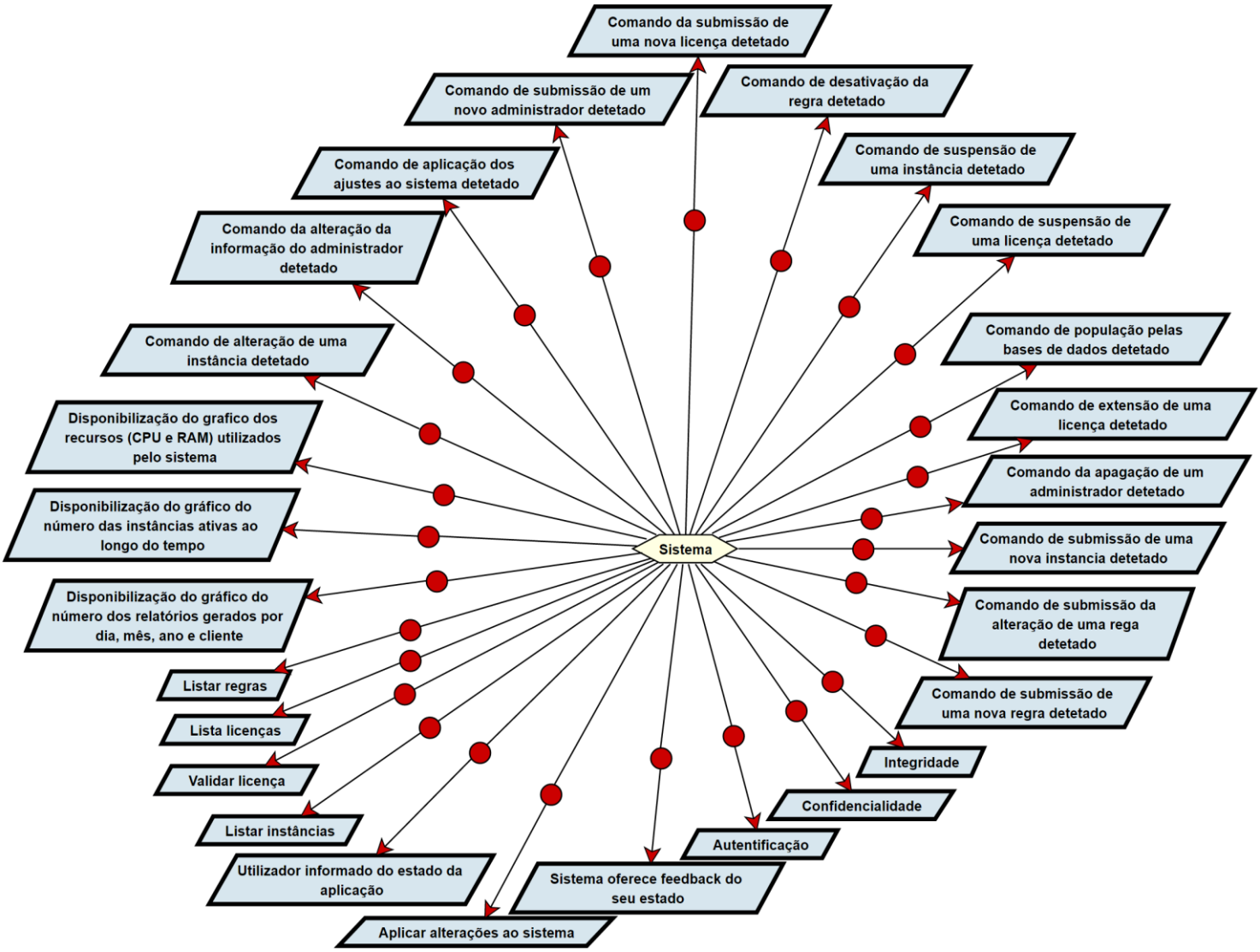


Figura 8: Requisitos Sistema

Como podemos verificar na Figura 8, para o ator "Sistema" foram identificados muitos requisitos, existindo assim os quatro não funcionais que são: "Integridade", "Autentificação", "Confidencialidade" e "Sistema oferece feedback do seu estado", os restantes requisitos são funcionais. A maioria dos requisitos funcionais correspondem a operações CRUD, ou seja, o sistema tem que conseguir detetar os comandos de criação, alteração ou de eliminação das instâncias, regras e licenças, como também listar estas todas. Existem também os requisitos que não fazem parte de operações CRUD, estes são requisitos mais específicos que têm o objetivo de fornecer ao administrador os dados ou permitir a configuração do sistema.

3.3 Requisitos do ator “Designer da interface”

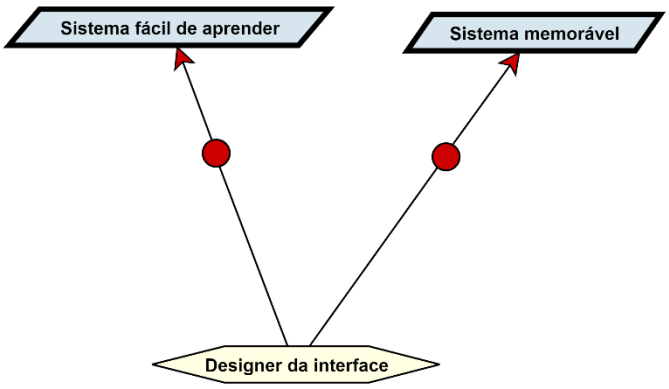


Figura 9: Requisitos "Designer da interface"

Como podemos verificar na Figura 9, o *designer* do sistema é responsável por construir uma *interface* que seja fácil de aprender e memorizar, isso ajuda o administrador melhorar a sua experiência com o módulo.

3.4 Requisitos do ator “Arquiteto do sistema”

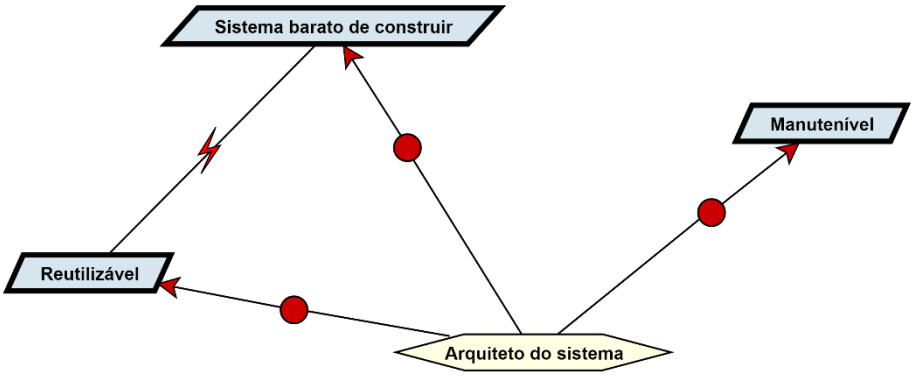


Figura 10: Requisitos "Arquiteto do sistema"

Como podemos verificar na Figura 10, o arquiteto do sistema tem que conseguir construir uma arquitetura de baixo custo, de fácil manutenção e de reutilização posterior. Também é fácil de notar que o arquiteto do sistema terá de lidar com o problema entre o reutilização e sistema barato, pois estes dois entram em conflito, não é fácil de ter um sistema reutilizável e barato ao mesmo tempo, o que implica de tentar encontrar o melhor equilíbrio entre estes requisitos.

3.5 Requisitos do ator “Azure”

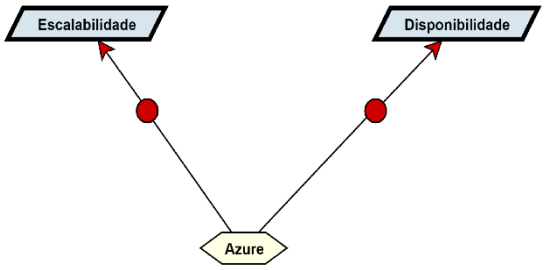


Figura 11: Requisitos "Azure"

Como podemos verificar na Figura 11, o *Azure* irá nos fornecer a disponibilidade e escalabilidade, pois são estas caraterísticas principais da *cloud*.

3.6 Modelo de dados

Nesta secção é apresentado o modelo de dados, Figura 12, na sua versão final para uma melhor visão da aplicação. No modelo de dados foram omitidas algumas das relações, nomeadamente as relações com a entidade “AspNetUser”. Estas relações foram omitidas para facilitar a visão e representação do modelo de dados, uma vez que esta entidade está relacionada praticamente com todas outras entidades. Estas relações servem para saber qual o utilizador que cria a entidade, o utilizador que fez a última edição à entidade e o utilizador que apaga a entidade. Para uma questão de coerência foram criadas estas relações até para as entidades, que não são possíveis de apagar, porque por ventura pode haver necessidade de realizar esta operação.

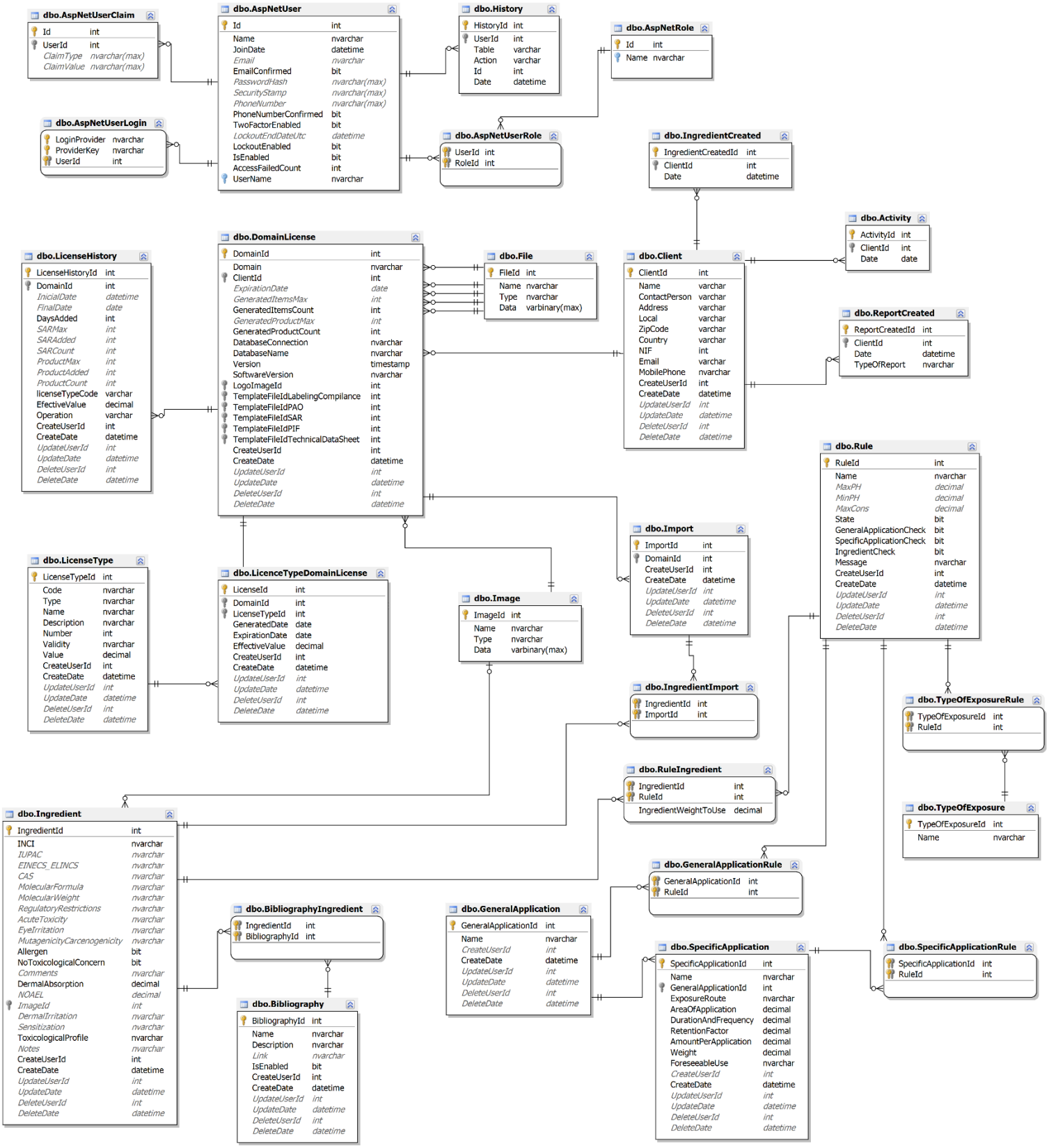


Figura 12: Modelo de dados

3.7 *Mockups*

Nesta secção serão listados alguns dos os *mockups*, os mais será possível consultar no documento de requisitos, de baixa fidelidade que foram criados para a aplicação com base nos requisitos e os objetivos do estágio, de uma forma simplificada, para demonstrar o seu possível *laylout* e estrutura. Por ventura os *mockups* também poderão servir de levantamento dos requisitos, pois o cliente poderá lembrar de algumas funcionalidades que poderão fazer sentido ser implementadas no determinado contexto. Na versão final da aplicação poderão existir mais ecrãs do que estão aqui dispostos.

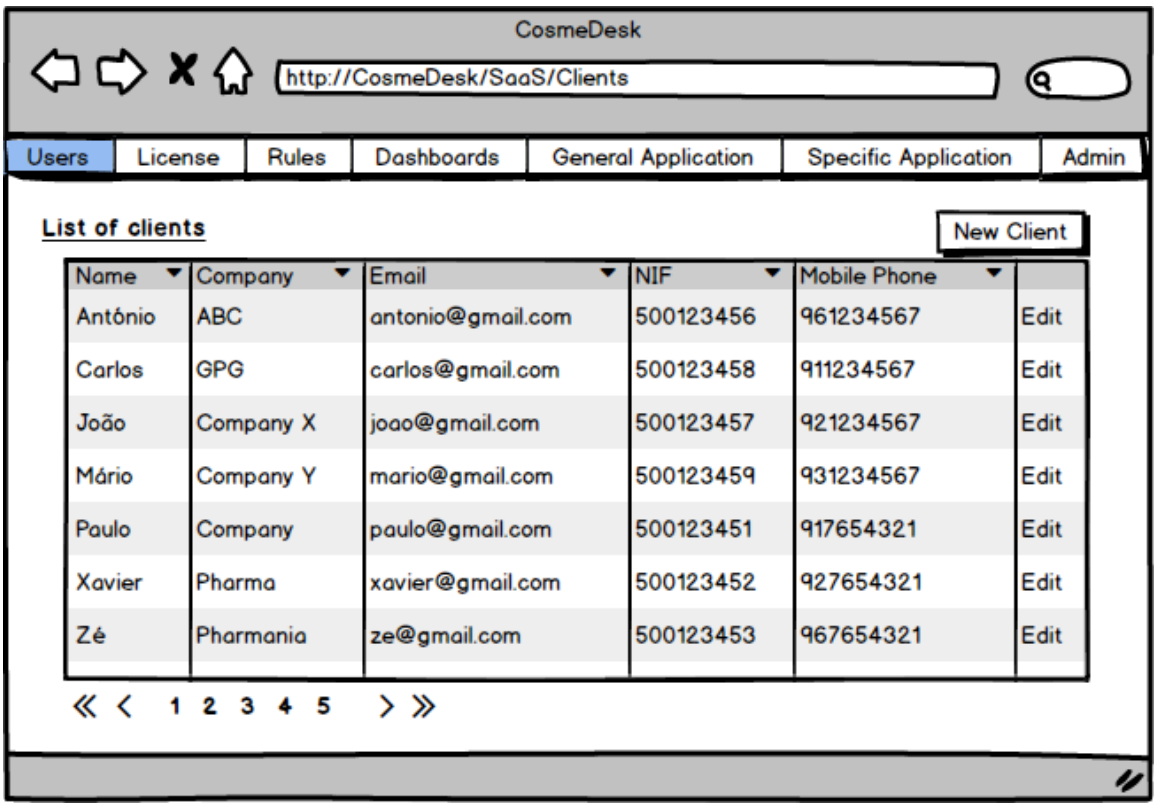


Figura 13: Lista de clientes

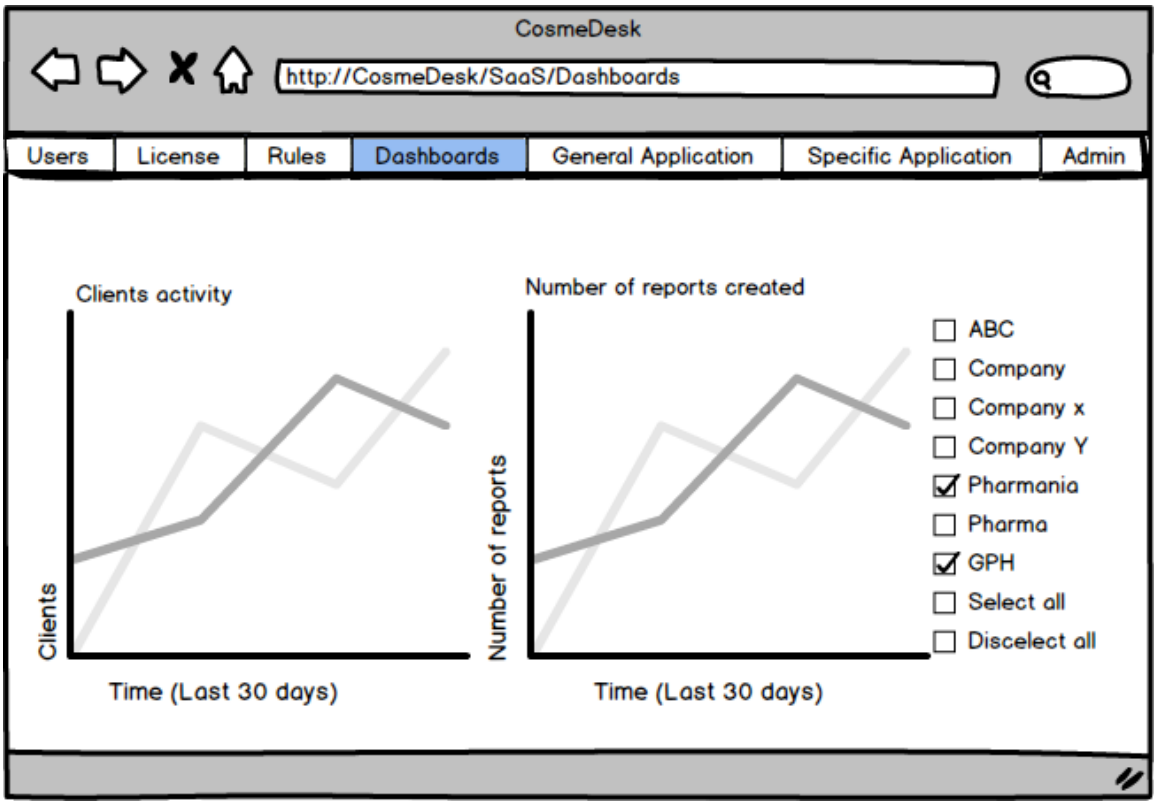


Figura 14: Dashboards

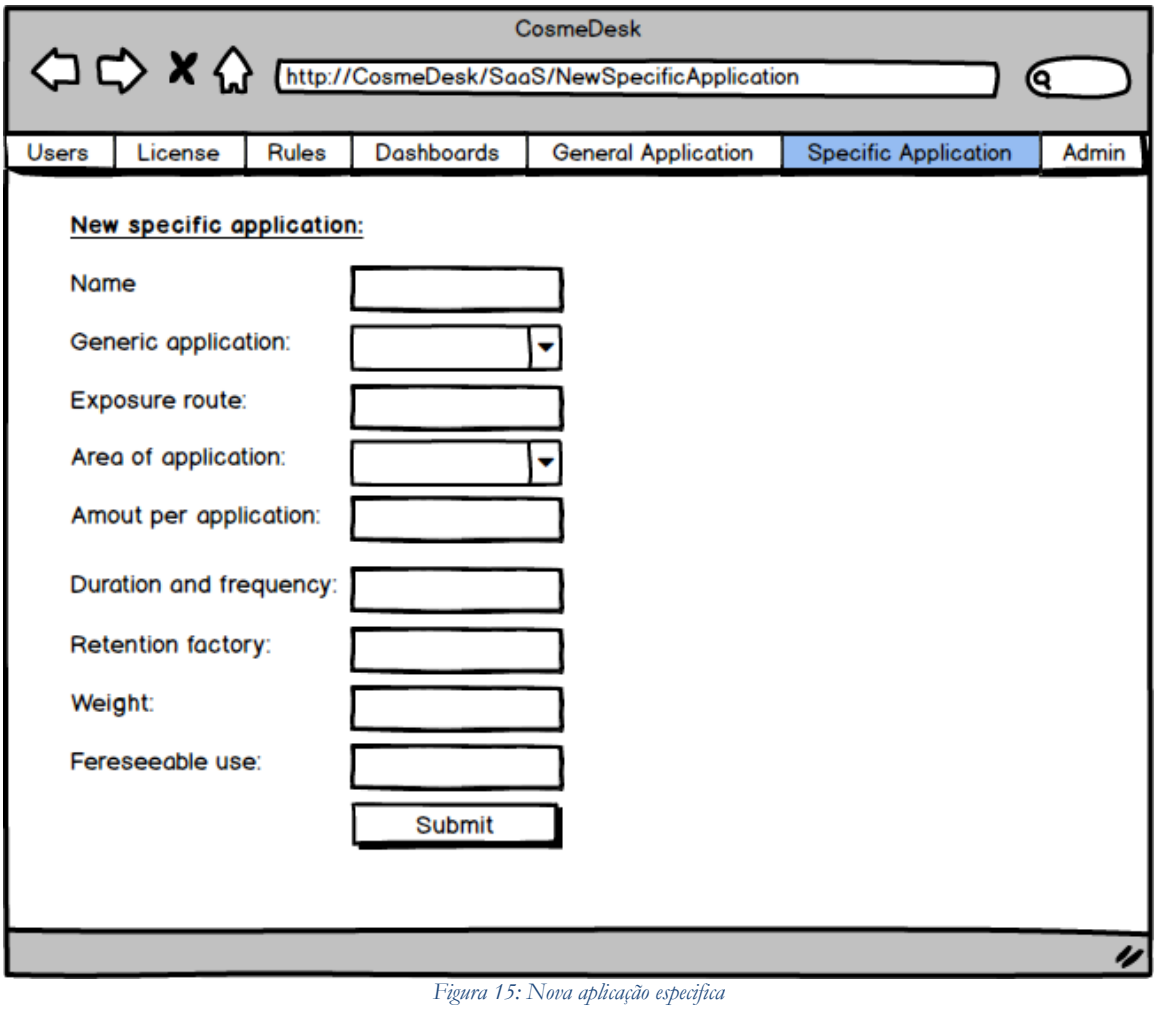


Figura 15: Nova aplicação especifica

3.8 Gestão de qualidade

Durante do presente estágio, foram adotadas alguma das normas de desenvolvimento de *software* para a organização do trabalho, de forma a manter um bom desempenho e boa qualidade. Sendo assim, esta secção irá descrever todas as metodologias/processos utilizados para estes fins.

3.8.1 Documentação

Para melhor organização dos documentos, principalmente os documentos a entregar, será adoptada uma regra para construção do nome que deverá sequir o seguinte formato:

<Nome do documento>_<Iniciais do autor>_v<versão(x.y)>.<Extenção>

Por exemplo:

Documento dos riscos_IH_v1.2.pdf

Na versão a variável "y" avança sempre que o documento seja enviado a alguém após as alterações realizadas no documento, seja isso, para verificar o documento ou demonstrar o trabalho realizado até ao momento. A variável "x" avança sempre que o documento seja aceite pelas pessoas que reverem este e consideraram de acordo com a posição atual do estágio. A posição atual do estágio não implica não existência das alterações seguintes, pois este ainda pode estar incompleto uma vez que é sempre possível melhorar a documentação ou atualizar com nova informação.

3.8.2 *Backup* da documentação

Para assegurar a segurança da documentação existente, esta será guardada no "Google Drive", da forma a manter todas ás versões da documentação será guardada. Seguindo a boa prática, o *backup* da documentação deve ser realizado no tempo inferior a uma semana. Aqui também serão armazenadas todas as versões existentes de cada documento, permitindo assim gerir as versões da documentação apesar de não ser este o principal objetivo.

3.8.3 *Backup* do código

Para assegurar a segurança do código fonte e recursos associados será realizado o *backup* do código fonte periódico num repositório do "GitHub", sendo este considerado o repositório distribuído de controlo das versões. Este sistema é distribuído pelos vários computadores que o têm subscrito, sendo assim permite o desenvolvimento colaborativo pelas versões dos ficheiros. O programa para aceder ao repositório será o "Source Tree", pois apresenta as simplificações do processo da utilização do repositório.

3.9 Gestão de riscos

A gestão dos riscos é um processo que identifica, avalia e prioriza os riscos, de forma a que o estagiário conheça os mesmos e tente evitar o acontecimento destes. Uma vez que o risco acontece, o estagiário deverá ter conhecimento de como agir, perante tal risco, de uma forma a contornar o mesmo e não prejudicar o estágio em questão.

Como os riscos podem ocorrer de uma forma natural, em qualquer das fases do projeto, o estagiário deverá conseguir lidar com os mesmos de acordo com as estratégias de mitigação. O estagiário para conhecer melhor os riscos que poderão acontecer, deverá identificar os riscos mais relevantes e ter uma estratégia para estes, que o ajudará tomar decisões perante os mesmos, caso estes se tornam problemas na realidade.

Não das tecnologias Microsoft, o que pode dificultar a construção da melhor solução

O estagiário não domina as tecnologias da Microsoft, uma vez que nunca teve a interação com as mesmas, isto poderá ser um risco porque apesar de todas as linguagens são de uma ou de outra forma semelhantes pode existir algumas dificuldades no inicio da implementação, pois coincide com a aprendizagem/interação inicial com as tecnologias.

Plano de mitigação

O presente estagiário deverá encontrar desafios e implementar exercícios de treino, para conhecer melhor as tecnologias e seguir os conselhos que recebe ao longo do estágio pelos colegas do laboratório, pois estes são muito mais experientes e podem dar conselhos úteis.

Acompanhamento

Se o presente estagiário sentir dificuldades na interação com uma tecnologia/ferramenta específica, depois de realizar várias pesquisas que não proporcionam o resultado, deve procurar ajuda nos colegas do laboratório, uma vez que estes são mais experientes, e podem já ter enfrentado um problema semelhante, podem oferecer uma ajuda relevante na resolução da dificuldade ou problema.

Plano de controlo

O estagiário sempre que sentir alguma dificuldade não pode perder muito tempo para a ultrapassar, não hesitando questionar os seus colegas do laboratório no caso da dificuldade persistir durante algum tempo.

Poucos conhecimentos do estagiário no desenvolvimento utilizando tecnologias web, o que proporciona mais esforço na implementação das tarefas

O estagiário tem pobre experiência na programação web, principalmente na parte do "Frontend", isto pode causar um relativo atraso na implementação das páginas da maneira que estes sejam mais apelativas e construtivas. Apesar da primeira iteração com o "ASP.NET" ser positiva, não se pode retirar o risco que pode crescer para um problema na criação de páginas mais complexas.

Plano de mitigação

O presente estagiário deverá conseguir encontrar e realizar os exercícios certos para ultrapassar a dificuldade encontrada, sendo estes exercícios da internet ou fornecidos pelos colegas do laboratório, sendo o segundo caso o melhor pois os colegas pela experiência passada podem facilitar a aprendizagem com exercícios simples.

Acompanhamento

O estagiário deverá notar as suas maiores dificuldades e tentar ultrapassar estas de forma a encontrar novos exercícios e exemplos de resolução para o problema.

Plano de controlo

No caso de encontrar um exercício com certa dificuldade para o estagiário, esclarecer a dificuldade com colegas, perguntar pelas possíveis resoluções do problema e encontrar exercícios semelhantes da maneira a manter e aplicar o conhecimento adquirido.

Projetos de outras unidades curriculares ocuparem o tempo em demasiado, o que proporciona atrasos da planificação do estágio

O presente estagiário tem mais quatro unidades curriculares para além da unidade curricular "Estágio/Dissertação" no primeiro semestre, o que é preocupante ao nível da gestão do tempo pelo estagiário, pois sendo este gerido mal podem surgir problemas no cumprimento da planificação.

Plano de mitigação

Construção de uma agenda para apontar todas as tarefas a realizar pelo estagiário, assim construir a visão geral da planificação e tentar integrar as tarefas de várias atividades da forma a não haver a interação entre eles.

Acompanhamento

Constante consulta da agenda, da forma a conhecer o estado atual e atualizar a agenda com as tarefas novas ou prolongamento/diminuição do tempo das tarefas já existentes na agenda.

Plano de controlo

Sempre que uma tarefa demorar mais tempo do que esperado, tentar de todas as formas possíveis enquadrar no espaço esperado, no caso de impossibilidade reagendar as tarefas consoante a nova previsão do tempo da demora para todas as tarefas.

Capítulo 4
Arquitetura

Neste capítulo é formalizada a arquitetura do sistema, descrevendo de forma detalhada o funcionamento e a interligação de todos os seus componentes. Apresentando as decisões tomadas, com justificação, para a chegada a uma solução do problema, tendo em conta não só o estudo efetuado no capítulo "Conceitos Relacionados", como também o levantamento e análise de requisitos.

4.1 Arquitetura Azure Proposta Pelo Estagiário

A arquitetura no "Azure" tem que satisfazer as necessidades dos clientes, uma vez que a aplicação poderá chegar a ter muitos clientes, havendo necessidade de gerir várias máquinas virtuais em simultâneo, distribuindo automaticamente a carga entre estas. Para isso, será usado um componente, *Application Gateway,* que irá balancear a carga entre as várias máquinas virtuais disponíveis. Em caso de sobrecarga das máquinas, que já estão disponíveis, serão adicionadas novas consoante a carga do CPU das mesmas. Este processo é gerido automaticamente pelo "Azure", não necessitando de intervenção humana. Por outro lado, poderia também existir a escalabilidade horária, para minimizar os custos, mas como o serviço irá ter uma escalabilidade das máquinas por recursos, não será necessário. Na existência da pouca atividade dos clientes as máquinas vão reduzindo automaticamente. Cada máquina virtual vai ter várias ligações, sejam estas dos clientes ou das bases de dados. Como cada cliente terá uma base de dados e um repositório partilhado, entre todos os clientes para o armazenamento dos relatórios, cada máquina virtual terá acesso a todas as bases de dados dos clientes. O *blob* será utilizado para o particionamento de dados, sendo este de custo reduzido e a utilização dos dados a armazenar nos *blobs* tenham pouca necessidade de consulta.

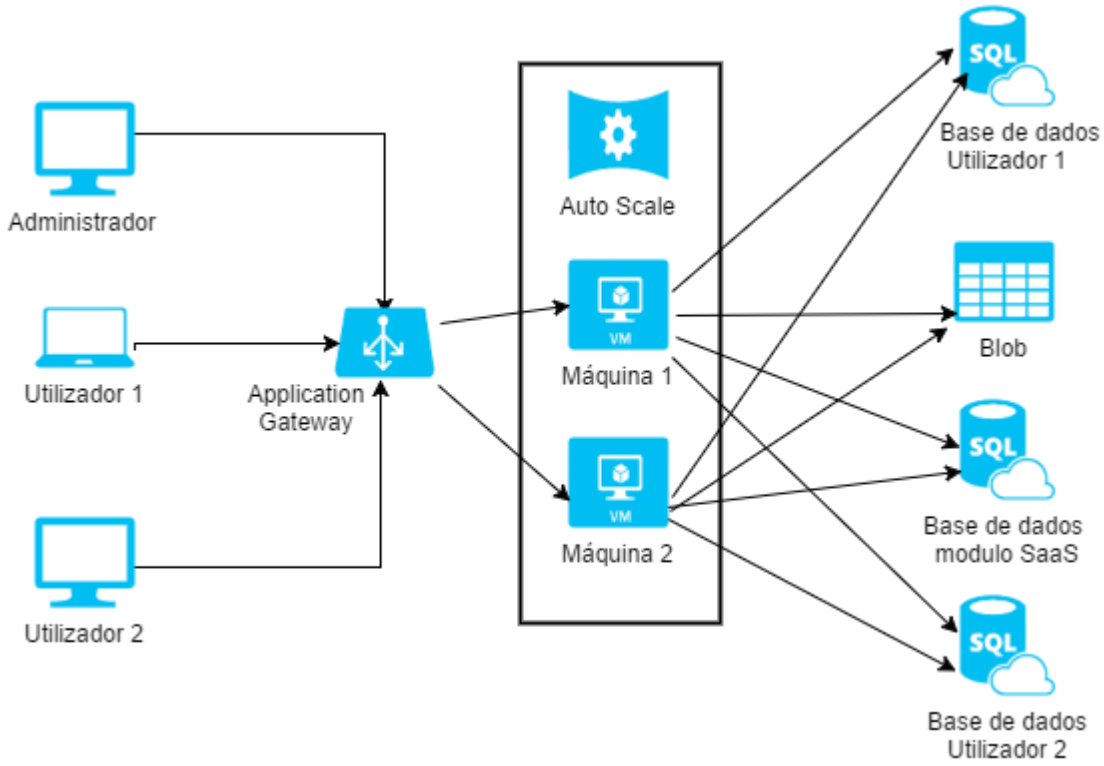


Figura 16: Arquitetura da aplicação (proposta)

4.1.1 Preço da arquitetura proposta

Na tabela seguinte, Tabela 6, estão descritos os preços para cada um dos componentes necessários para a implementação da arquitetura, representada na Figura 16. Também se apresenta o valor final mensal previsto. Estes valores foram obtidos tendo em conta os valores representados na Tabela 4

Componente	Quantidade	Preço unitário	Preço mensal
Application Gateway	1	17,57 €	17,57 €
Máquina	2	168,77 €	337,57 €
Base de dados	3	4,20 €	12,60 €
Blob	1	0,21 €	0,21
Total	7		367,92 €

Tabela 6: Preço da arquitetura proposta

Neste dado momento é suficiente uma máquina para satisfazer as necessidades de todos os clientes da aplicação, mas com a chegada de mais alguns poderá ser necessário aumentar o

número de máquinas. Precisamos de aumentar o número mínimo de máquinas a executar, pois o sistema tem de estar preparado para, no caso de todos os clientes se ligarem ao mesmo tempo, não *crashar* a aplicação até que a próxima máquina inicie. O grupo das máquinas escala ligando mais máquinas quando a percentagem de utilização da CPU ultrapassar uma certa percentagem definida, durante certo tempo. No entanto o recurso mais utilizado é a RAM, o que significa que a escalabilidade automática das máquinas pode estar comprometida, pois não inicia as máquinas quando estas são mesmo necessárias. A necessidade do sistema ter um *Application Gateway* aparece depois de nós termos muitas máquinas na arquitetura, pois necessitamos que o tráfico seja bem distribuído pelas várias máquinas, não havendo algumas destas sobrecarregadas e outras não.

Depois da discussão da arquitetura, percebeu-se, que o cliente pretende diminuir os gastos da arquitetura, representada na Figura 16, pois neste dado momento não existe grande quantidade de utilizadores na aplicação. Como o cliente é único utilizador, tem de suportar os custos da arquitetura, existindo outros na fase de teste, utilizando a versão gratuita.

4.2 Arquitetura Azure a Implementar

Como há poucos utilizadores da aplicação "Cosmedesk", neste momento, não se prevendo uma aderência na ordem das centenas de um momento para outro, o cliente pretende continuar com a arquitetura mais simples, uma vez, que tem que suportar os gastos todos sozinho. Para tal foi simplificada a arquitetura anterior, apresentada na Figura 17, havendo uma máquina virtual que contém a base de dados e o servidor aplicacional, realizando o particionamento dos dados no *blob*. Este *blob* irá conter os ficheiros correspondentes aos relatórios ".doc" e ".pdf", havendo só a referência para o local do armazenamento na base de dados. Cada cliente terá o seu *container* no *blob,* permitindo assim o melhor isolamento dos dados dos clientes.

Com esta nova arquitetura não existe o problema do escalamento, pois as máquinas podem ser escaladas utilizando o método *scale-up*. O "Azure" disponibiliza as máquinas com recursos bastante elevados, por exemplo a máquina com o modelo G5 tem: 32 Cores, 448 GB de RAM e 6144 GB de Disco.

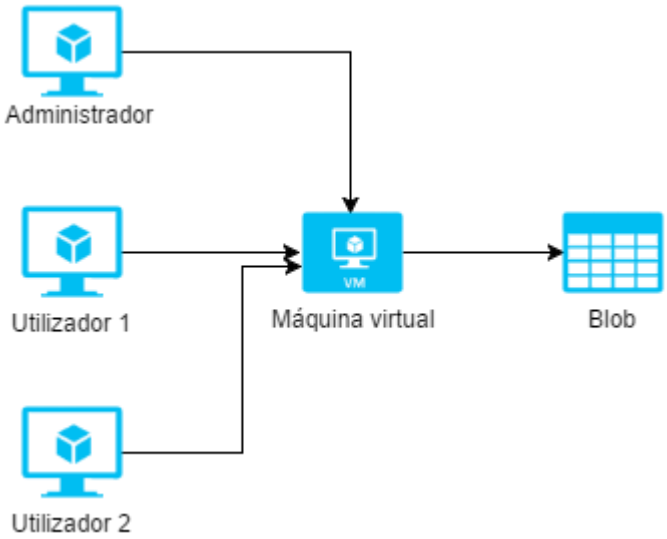


Figura 17: Arquitetura da aplicação (aceite)

4.2.1 Arquitetura visão nível 1

Arquitetura anterior, apresentada na Figura 17 descreve uma visão do nível 0, arquitetura geral do sistema com pouco detalhe, sendo necessário mais detalhe, apresenta-se nesta secção a arquitetura da visão do nível 1. Esta arquitetura representa a visão dos componentes existentes na máquina virtual. Como podemos verificar na Figura 18, representada em baixo, a máquina tem várias instâncias, cada uma com a sua base de dados, as instâncias também podem aceder à base de dados principal, que é a base de dados da instância do módulo. Isto é necessário para haver um controlo sobre as licenças e para controlar as alterações nas bases de dados. Esta ligação só é disponível para leitura dos dados. Também existe o servidor aplicacional que gere os domínios para cada instância, pois cada uma tem o seu domínio e o seu ambiente do trabalho. Deve também existir o serviço do Microsoft Word a executar, para possibilitar a geração automática dos relatórios.

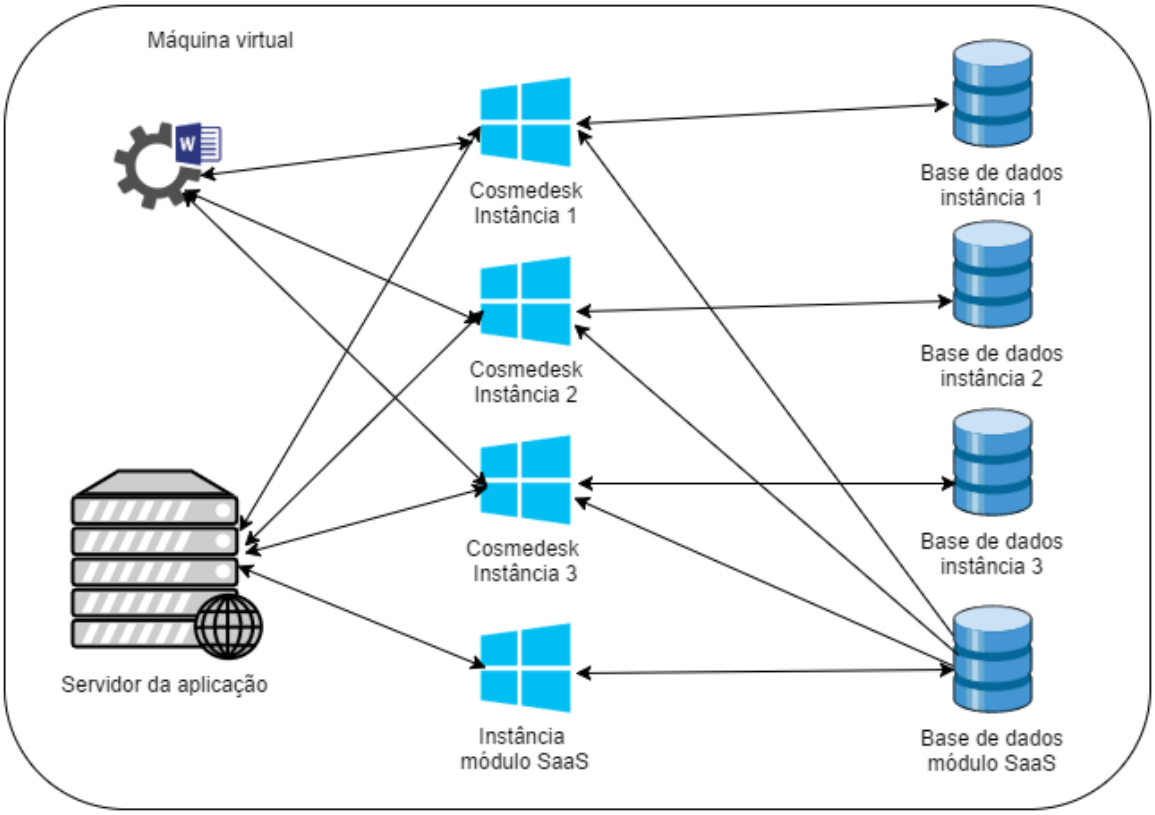


Figura 18: Arquitetura visão nível 1

4.2.2 Preço da arquitetura a implementar

Na tabela representada em baixo, Tabela 7, está descrito o preço por componente e o valor total mensal, para a arquitetura a implementar, representada na Figura 17, tendo em conta os valores representados na Tabela 4.

Componente	Quantidade	Preço unitário	Preço mensal
Máquina	1	168,77 €	168,77 €
Blob	1	0,21 €	0,21
Total	2		168,98 €

Tabela 7: Preço da arquitetura a implementar

Esta arquitetura representa o menor preço inicial, também pode ser escalada, utilizando o método *scale-up*, não apresentando dificuldades na manutenção da arquitetura. A única dificuldade visível desta arquitetura, é o gasto dos recursos da máquina pelo motor da base de dados e do servidor aplicacional. Pois a utilização destes componentes aumenta o preço da

arquitetura, por exemplo, neste dado momento como existem só três utilizadores, um dos quais é o cliente e outros dois estão em fase de testes, o cliente tem que pagar as despesas referente a quatro bases de dados, uma para cada utilizador e mais uma do módulo. Isto forma um gasto adicional da arquitetura num valor de 16,80 € mensal, não contando o gasto em *Application Gateway*, 17,57 €, que formava um gasto total de 34,37 €.

Esta arquitetura não acrescenta uma máquina igual à já existente na chegada de um número maior de clientes, ou seja, o cliente pode evoluir passo a passo, podendo adquirir uma máquina ligeiramente superior, o que pode simplificar os seus gastos. Havendo o maior problema na utilização da RAM, o cliente pode optar, por um custo ligeiramente superior, por uma máquina com caraterísticas semelhantes, oferecendo mais RAM. Por exemplo, a máquina de modelo D11 que oferece as seguintes caraterísticas: 2 núcleos, 14 GB de RAM e 100GB de Disco por 181,95 € mensais, ou seja, conseguimos duplicar a RAM por uma diferença de 13,18 € mensais.

4.3 Arquitetura de software

Nesta secção irá ser descrita a arquitetura de software implementada, fornecendo a informação sobre a organização e a interligação das ferramentas utilizadas.

4.3.1 Descrição da applicação

A aplicação esta dividida em três camadas como podemos observar na Figura 19, representada em baixo, contendo a camada de dados, a camada do negócio e a camada da apresentação que por sua vez está dividida em duas, a camada de WEB API e a camada de WEB APP.

A camada de dados é responsável pela ligação à base de dados, nesta camada é usada "*entity framework*" com *database first,* o que significa que o *entity framework* irá construir as classes com base nas tabelas existentes na base de dados. O *entity framework é "*uma ferramenta de mapeamento objeto relacional*"* (21), segundo a autor do dado artigo, com *entity framework* os desenvolvedores manipulam as classes, que estão mapeados nas tabelas da base de dados, o que facilita o trabalho aos desenvolvedores e permite-os o trabalho natural com objetos.

A camada do negócio também utiliza a *entity framework* para realizar consultas à base de dados, mas por base nesta camada esta é programada com recurso à linguagem C# da Microsoft. Esta camada é responsável por toda a lógica e pela interligação entre o módulo e o "Cosmedesk".

A última camada, camada de apresentação, está dividida em duas camadas, a camada WEB API que tem por base a programação em C# e permite a interligação da WEB APP com a camada do negócio. Enquanto a camada WEB APP esta maioritariamente construída em *framework* "Angular JS" (22) e "Kendo UI" (23) do "Telerik". Neste determinado caso a aplicação será desenvolvida maioritariamente em "Angular JS" que irá sustentar e alimentar os componentes do "Kendo UI" permitindo um melhor desempenho.

O "Angular JS" é uma *framework JavaScript* da "Google" que permite construção das aplicações web dinâmicas, funciona como uma extensão do HTML, assim diferenciando-se das outras *frameworks JawaSkript*.

O "Kendo UI" é uma *framework* de construção do *interface*, oferece várias funcionalidades e componentes que permitem uma construção mais simples do *interface web,* esse oferece muitas funcionalidades, que muitas das vezes tenham de ser variáveis consoante as necessidades de determinada apresentação, para isso existem as opções disponíveis do "Kendo UI" que são manipuladas pelo "Angular JS" oferecendo a flexibilidade na apresentação dos componentes e possibilidade de reutilizar estes componentes para grande diversidade de apresentações diferentes e até dados diferentes.

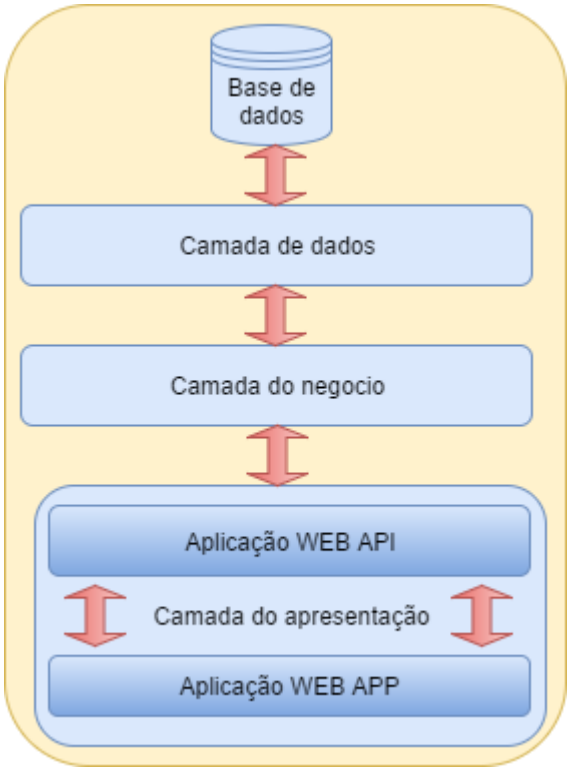


Figura 19: Arquitetura de software

Capítulo 5
Solução

5.1 Tecnologias utilizadas

Neste capítulo serão mencionadas as tecnologias utilizadas na elaboração da solução de uma a satisfazer as necessidades dos utilizadores ao melhor nível possível.

5.1.1 Angular JS

O Angular JS (24) é uma *framework javaScript*, agora mantida pela Google, foi construída mantendo a arquitetura Model-View-View-Model (MVVM) com esforço de auxiliar tanto o desenvolvimento como testes das aplicações. Os seus principais objetivos são:

- Abstrair manipulação do DOM;
- Abstrair o acoplamento entre o lado do cliente e lado do servidor da aplicação;
- Angular JS segue o padrão MVC da engenharia de software e encoraja o baixo acoplamento entre apresentação;
- O Angular JS trás os serviços, em geral, designados ao lado do servidor que diminui o peso sobre o *backend* do servidor.

5.1.2 Kendo UI

O Kendo UI (25) é uma das mais completas librarias de *user interface* para os desenvolvedores de HTML, JavaScript e Angular. O kendo UI combina de melhor forma os mais de 70 componentes que facilitam a vida ao programador. Tem vários temas já predefinidos, tendo a possibilidade de trabalhar com dados online e offline, suporta todos os browsers.

5.1.3 Entity Framework 6.0

O Entity Framework (26; 27) é uma *framework* Object Relational Maping (ORM), recomendada pelo Windows para todas as novas aplicações que trabalham com dados, suporta três métodos diferentes de criar a base de *workflows* diferentes, base de dados primeiro, modelo primeiro, código primeiro. Possibilitando ao programador trabalhar com os objetos e as suas propriedades, criar e manter a informação com menos código do que nos programas tradicionais.

5.1.4 C#

O C# (28) é uma linguagem orientada a objetos, designada para trabalhar com a plataforma .NET da Microsoft para desenvolvimento de aplicações WEB altamente portáveis, facilitando a troca de dados/serviços.

5.1.5 MVC

A estrutura Model-View-Controller (MVC) (29), é separada em componentes chaves: o modelo, a vista e o controlador. Esta separação ajuda o programador a focar em cada aspeto da aplicação. O *framework* MVC é uma estrutura de apresentação leve e altamente testável é integrada com os recursos do ASP.NET.

- **Model**: O modelo é a parte da aplicação que implementa a lógica, também interage com a base de dados.
- **View**: A vista é a parte da aplicação em que cria a interface para utilizador com base nos dados que recebe da camada do modelo.
- **Controler**: O controlador é uma parte da aplicação que lida com as interações do utilizador com a vista, interage com o modelo e seleciona a próxima página para fazer render, é uma parte que liga as outras duas.

5.1.6 Microsoft SQL Server Manager (SMO)

O Microsoft SQL Server Manager (30) é uma instância do SQL Server que permite realizar as diversas operações com o mesmo, na criação de uma instância ele conecta-se ao SQL Server e realiza todas as operações, tem a vantagem de lado do programador ser tratado como um objeto que facilita sua manipulação, também é melhor no tratamento de erros uma vez que devolve exceções que são fáceis de interpretar.

5.1.7 Web Administrator

O Web Administrator (31) é uma instância de IIS Server que permite manipulação de diversos componentes, aplicações, pool das aplicações, sites, domínios, processo trabalhador, diretório virtual, entre outros, como objetos de alto nível.

5.2 Estrutura do projeto

A estrutura real da solução resume-se a 9 projetos criados que correspondem a ao módulo *back office* representada na Figura 20. Esta divisão foi necessária para uma divisão por camadas da aplicação e também a existência de uma API, que é "CosmedeskSAS.WebApi" que alimenta o "CosmedeskSAS.WebApp" construída em Angular JS e Kendo UI. A camada de negócio corresponde ao projeto "CosmedeskSAS.BusinesLayer" e a camada dos dados é o projeto "CosmedeskSAS.DataLayer". Como o módulo *back office* é uma aplicação "Database First" tem que existir um projeto que cria a mesma para este usar, este papel é atribuído à "CosmedeskSAS.Database".

Mas também há a necessidade de a aplicação ter acesso a algumas tabelas das bases de dados da aplicação "Cosmedesk", para este fim foi criado o projeto "Shared" e o "CosmedeskDataLayer". Com a finalidade de fornecer a conectividade a todas as bases de dados dos clientes e algumas tabelas especificas do "Cosmedesk".

O projeto "CosmedeskSAS.HostedApp" tem a finalidade de fornecer a os dados da licença e a *string* da conexão à base de dados para a aplicação "Cosmedesk" de forma a possibilitar o seu correto funcionamento e limitar o utilizador perante a licença adquirida. A aplicação "Cosmedesk" é uma aplicação "Code First", ou seja, é uma aplicação que com as necessidades cria e atualiza a suas tabelas na base de dados dando lhe uma *string* de conexão à mesma, por isso existiu a necessidade de passar a *string* de conexão, mas também a licença não fica do lado do cliente, uma vez que as informações das licenças encontram-se na base de dados do módulo *back office*, o "Cosmedesk" tem que pedir informações relativamente a sua licença em cada operação realizada.

Por fim o projeto "CosmedeskSAS.BusinessLayer.Test" corresponde ao projeto com os testes unitários efetuados na camada de negócio.

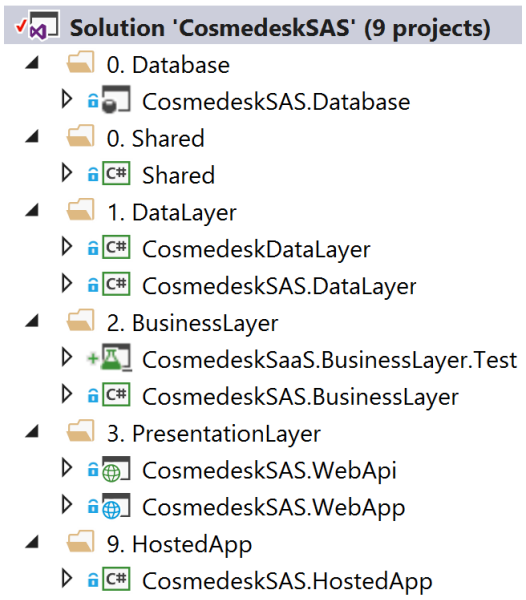


Figura 20: Solução para Cosmedesk back office

5.3 Decisões chaves

Em qualquer projeto de engenharia é importante a tomada de decisões, que podem fornecer aos clientes boas soluções, seja isso um melhor desempenho ou mais fácil interação com a plataforma. Nesta secção serão partilhadas as decisões chaves tomadas na implementação da aplicação.

5.3.1 Interligação com "Cosmedesk"

Como é de perceber do estágio existem duas aplicações, uma "Cosmedesk" que já existia antes do estágio iniciar sendo parcialmente alterada e uma outra aplicação, módulo *back office,* que foi desenvolvida durante o estágio. Para a interligação das duas aplicações tentou-se escolher a maneira mais fácil e eficiente de as interligar. Para uma boa interligação foi deixado aos clientes consultar a base de dados do módulo, enquanto que o módulo terá as permissões de manipulação de todas as bases de dados criadas. O "Cosmedesk" na sua primeira execução irá consultar a base de dados e com o seu subdomínio irá selecionar a sua *string* de ligação a base de dados, que é guardada numa variável para consultas posteriores. Depois da autentificação realizada para cada operação que a aplicação "Cosmedesk" irá realizar, terá que pedir os dados da licença, pelo seu domínio, a base de dados do módulo. O sistema foi implementado desta maneira para que a licença seja a mais atualizada possível, sendo possível várias pessoas trabalhar ao mesmo tempo, por sua vez existem poucos utilizadores que podem gerar os relatórios. Na Figura 21, representada em baixo está uma demonstração da iteração da aplicação "Cosmedesk" com base de dados do módulo.

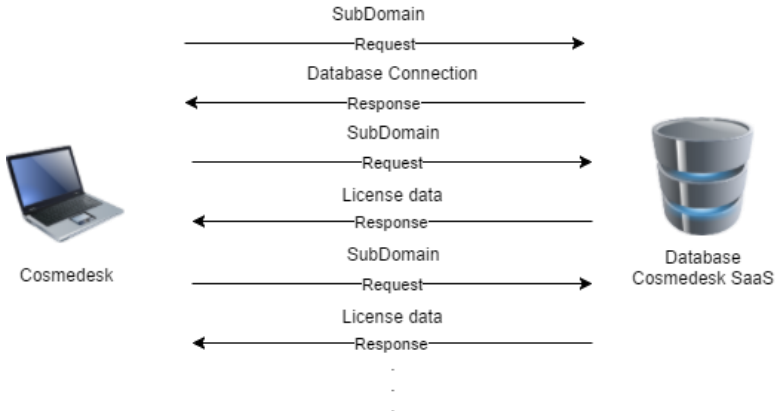


Figura 21: Iteração Cosmedesk com base de dados Back office

Havendo também a interação com a base de dados dos clientes da parte do módulo *back office,* uma vez que este terá que inicializar a base de dados dos clientes na sua criação com os dados dos "General Application" e "Specific Application", também existindo a opção de importação dos "Ingredients". Como os "General Application" e "Specific Application" não podem ser atualizados na base de dados do cliente depois da inserção, pois isto é possível fazer do lado do cliente e não podemos alterar os dados, que estão especificados por clientes, não faz sentido

ter uma tabela que guarda "ids" destas entidades. Por sua vez os ingredientes podem ser importados para um domínio especifico do cliente, uma base de dados que pertence a um certo domínio, no caso de existir um pedido da parte do cliente com que tenha a necessidade dos certos ingredientes. Os ingredientes não são atualizáveis, pois o cliente pode realizar esta operação na sua plataforma e como tal não podemos atualizar os ingredientes, uma vez que assim estamos a comprometer os relatórios que o cliente pode estar a criar com determinados ingredientes. Os ingredientes que tenham ligação com as bibliografias, para um dado ingrediente podem existir várias bibliografias, existe a necessidade de importação das mesmas para a base de dados do cliente na altura da importação dos mesmos.

5.3.2 Particionamento de dados

O particionamento de dados é muito importante, pois os relatórios como todos os constituintes para estes eram guardados na base de dados. Como os relatórios podem chegar a ter grande número de páginas (mais de 300 páginas), estes ocupam muito espaço na base de dados, tendo em conta que é sempre necessário guardar duas versões do relatório, a versão ".docx" e a versão ".pdf". Também existe uma grande parte dos constituintes do relatório que são imagens ou anexos (".pdf"), estes também ocupam muito espaço.

Sendo assim, o particionamento de dados foi realizado a pensar em conseguir separar esta informação da base de dados do cliente. Foram pensadas três soluções possíveis descritas no capítulo 2 Conceitos Relacionados na secção 2.3 Particionamento de dados. Para implementação foi escolhida a solução de utilizar o "Blob storage", utilizando os *blobs* para armazenar a informação deixando na base de dados do cliente as credenciais de acesso e o nome do ficheiro a consultar.

Como podemos ver na Figura 22, representada em baixo, é um caso real de uma base de dados de um cliente da Cosmedesk, os relatórios ocupam aproximadamente 73% da base de dados.

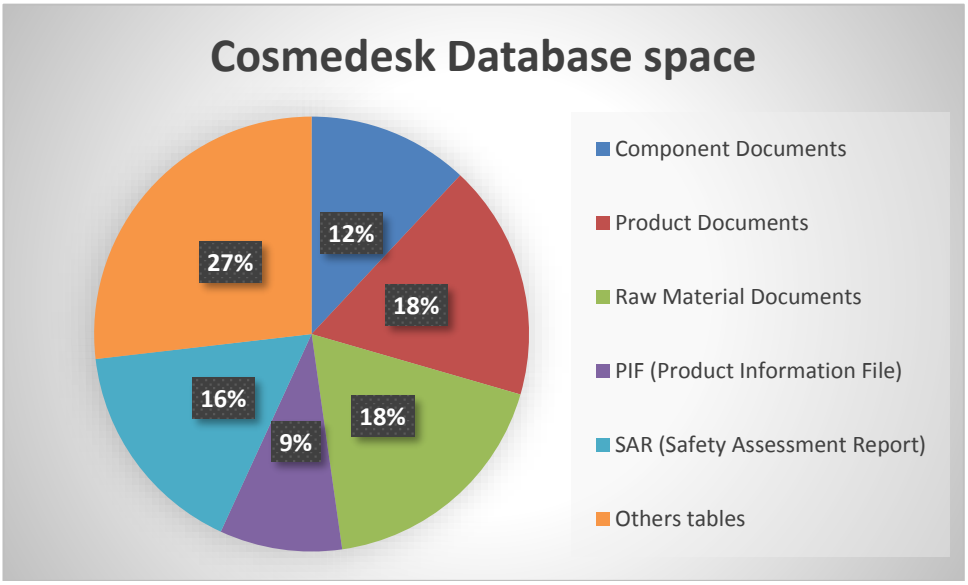


Figura 22: Base de dados do Cosmedesk de um cliente real

A base de dados selecionada tinha no seu total 3745,31 MB, depois de realizar o particionamento de dados este valor diminuiu para 935,27 MB, ou seja para cerca de 25% do tamanho inicial. Este valor foi conseguido pois não só os relatórios que foram extraídos da base de dados, mas sim também as imagens. Não se encontrava um número grande das imagens armazenadas na base de dados, por isso o valor obtido não foi assim tão diferente do valor resultado da subtração dos relatórios, mas também não podemos esquecer que as tabelas ficaram com as localizações como os nomes e tipos de dados na base de dados. Assim conseguimos reduzir o espaço das tabelas nomeadas para os documentos para um valor por volta de 1MB que é um resultado mais do que satisfatório.

O valor estimado a pagar pelo o uso de Blob da Storage na Azure é de 0,08€ mensal para 3GB da informação e até 500000 de transições. É um valor muito reduzido perante a vantagem de ter os dados fora da base de dados e estes serem replicados três vezes pelo próprio *datacenter*.

5.3.3 Suporte para única e múltiplas instâncias

O módulo *back office* está preparada para gerir os sites do Cosmedesk de duas maneiras, colocar todos os sites novos criados para os novos clientes numa especifica instância ou criar uma instância por cada site criado para cada novo cliente. Neste momento só existe uma instância que trata de todos os sites, logo o módulo *back office* tem que estar preparado para esta necessidade.

Segundo a Palvirenti Carmelo (32), o empregado da Microsoft, é mau ter vários sites numa única instância, uma vez que pode causar problemas tanto com a memória como também com o desempenho, sendo a memória e o processo alocado para a instância, não tem recursos infinitos, tipicamente ficam alocados para uma instância 2 GB da memória e um processo de 32 bits. Com o mau desempenho de um dos sites, que pode gastar muitos recursos disponíveis, os restantes serão afetados. Outro problema grave é que as aplicações ASP.NET por norma utilizam *Garbage Collector* para limpar a memória, o que na prática irá acontecer que o *Garbage Collector* irá executar várias vezes por segundo para limpar a memória para aplicações que estão na mesma instância, pois partilham a mesma memória. Isto irá afetar o desempenho das aplicações pois cada vez que o *Garbage Collector* iniciar o seu trabalho tem que parar todas as *threads* ativas, o que causará problemas ao nível da lentidão da aplicação e no elevado impacto no tempo da CPU.

Existindo também outras desvantagens, o módulo *back office* tem que estar preparado para fornecer esta mudança. Com a criação da instância para cada novo cliente, garantimos o melhor desempenho do site do cliente, pois os recursos alocados servirão um único site.

5.3.4 Múltiplas versões

O módulo *back office* permite a existência de duas versões da aplicação Cosmedesk, estas versões são "Release" e "Beta". O "Release" serve como versão sem *bugs,* que já foi suficientemente testada para ser uma versão final, enquanto o "Beta" é uma versão que acabou sair da produção e correu os testes na produção. Esta necessidade surgiu para que a versão "Release" seja mais segura e não apresente os *bugs* ao utilizador final, enquanto a versão "Beta" só será disponibilizada aos clientes que pretendem esta pelo possível desconto, ou o próprio cliente terá interesse em melhorar a versão final, como é o caso da consultora GPH, que é comerciante do Cosmedesk.

Os clientes podem ter uma só versão ou as duas ao mesmo tempo, isto é permitido pois o mesmo cliente pode ter mais do que uma instância. No caso de o cliente só ter uma única instância, pode optar por mudar para outra, isto é possível pois na prática as duas versões são duas pastas diferentes da aplicação, para mudar a versão de um cliente na prática só muda o caminho físico para a pasta correspondente e o cliente passa a usar a versão atribuída.

Para atualizar uma das versões é necessário substituir o conteúdo da pasta correspondente pela versão a atualizar, e todos os clientes que usem esta versão passam a usar a nova versão. Os controlos das publicações a realizar para substituição das versões correspondentes, são realizados pelo alguém responsável dos desenvolvedores.

5.4 Arquitetura

Nesta secção será apresentada a arquitetura do sistema implementado. Esta é a versão final da arquitetura, para demonstração da qual foi utilizada a técnica C4 Model (33) de Simon Brown, sendo esta composta por quatro níveis de visibilidade:

- **System Context**: é uma visão geral da aplicação, o foco deve ser feito nos atores principais, tais como pessoas que interagem com a aplicação e os sistemas de *software* em vez de tecnologias, protocolos e outros detalhes de baixo nível.
- **Containers**: este passo ilustra as escolhas tecnológicas de alto nível, um *container* é como uma aplicação que pode ser web, móvel, desktop, etc. Essencialmente um *container* é uma unidade desdobrável que executa código ou armazena dados. O diagrama mostra as principais tecnologias e como elas se comunicam entre si.
- **Components**: os componentes ampliam cada um dos *containers*, para visualizar os principais blocos de construção e suas interações, o que eles são, suas responsabilidades e detalhes da tecnologia.
- **Classes**: é um diagrama de classes, que representa as classes iniciais da aplicação, segundo o autor é um nível opcional.

A arquitetura representada contem três níveis uma vez que o nível quatro retrata o diagrama de classes, que o próprio autor afirma que existe uma pouca utilização deste nível, uma vez que já entra em muito detalhe e muitas das vezes é ignorado pelos utilizadores da dada técnica.

5.4.1 System Context

No *system context* , Figura 23, é representada a arquitetura do nível mais alto para demonstrar o contexto do sistema. Neste caso temos três tipos de utilizadores que utilizam a aplicação, estes utilizadores são, "Gestor de cliente", "Gestor técnico" e o "Administrador". Tem que se referir que o papel do utilizador é acumulável, isto é, uma pessoa pode ter mais que uma função. O gestor de clientes é o responsável por toda a gestão de clientes, pelas licenças e também pode importar ingredientes para os clientes. O gestor técnico é a pessoa responsável pela gestão de todos os dados mais técnicos, tais como, aplicações genéricas, aplicações especificas, ingredientes, bibliografia e regras de legislação. O administrador é a pessoa que tem acesso a todo o histórico das operações e é responsável pela gestão dos utilizadores.

O módulo também cria as instâncias da aplicação "Cosmedesk", uma aplicação exterior que o módulo usa. Para cada um dos novos clientes é disponibilizando um link, enviado por email, de registo na sua instância, possibilitando assim ao cliente escolher a sua *password* na primeira execução da aplicação.

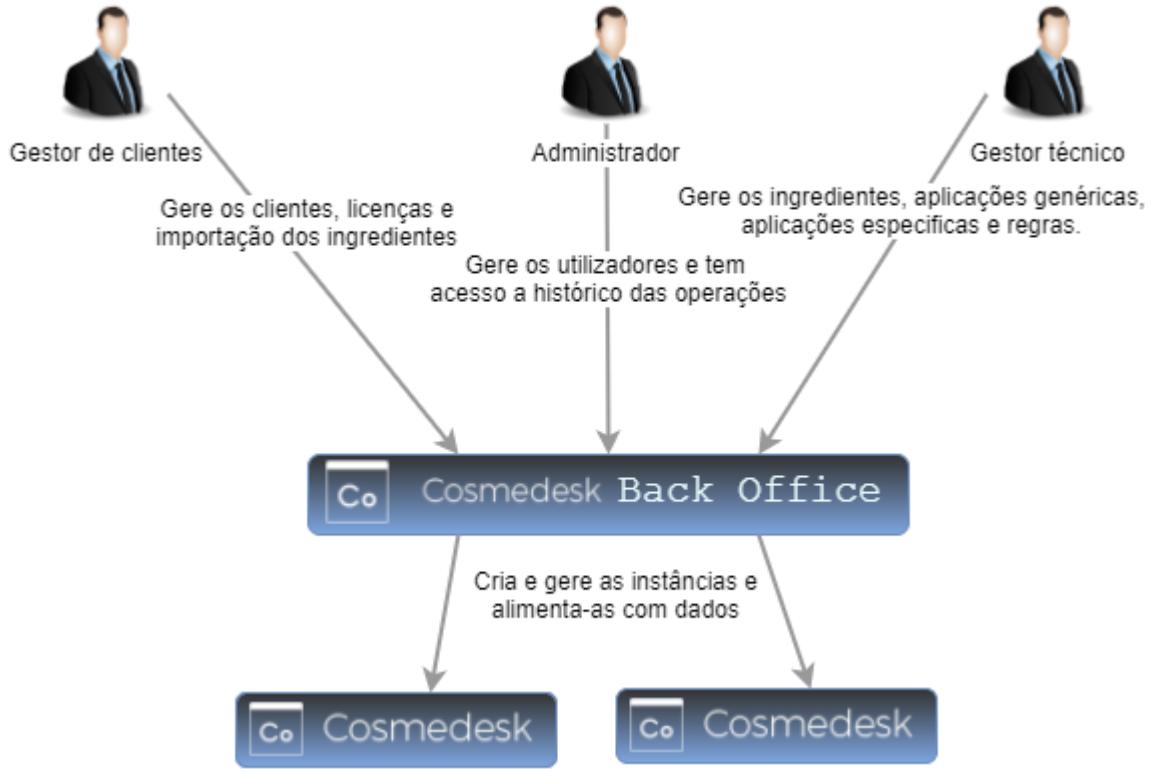


Figura 23: Arquitetura - System Context

5.4.2 Containers

O *Container* representado na Figura 24, representa a própria aplicação e as escolhas tecnológicas de alto nível, podendo-se observar as escolhas principais para o módulo. A aplicação WEB é uma aplicação construída em "Angular JS" e "Kendo UI", é a que interage com o utilizador, transmitindo as operações para a API. A API é uma aplicação desenvolvida em C#, que recebendo as operações efetuadas pelos utilizadores da aplicação WEB, efetua as alterações necessárias, criando novas instâncias ou simplesmente a atualização de uma das entidades na base de dados, que se encontra no contentor "Base de dados não relacional". O contentor "Base de dados não relacional" é um grupo de bases de dados num servidor da base de dados, neste caso foi escolhido como enunciado anteriormente SQL Server 2016, que contém todas as bases de dados das instâncias e a base de dados do módulo *back office*. O contentor "Sistema de ficheiros", serve somente para armazenar os modelos para os relatórios de uma forma estruturada e acessível às duas aplicações. O serviço "Blob Storage" é somente para o uso das instâncias da aplicação "Cosmedesk", que foi necessário para diminuir a carga sobre a base de dados perante a grande quantidade de dados existentes nestas.

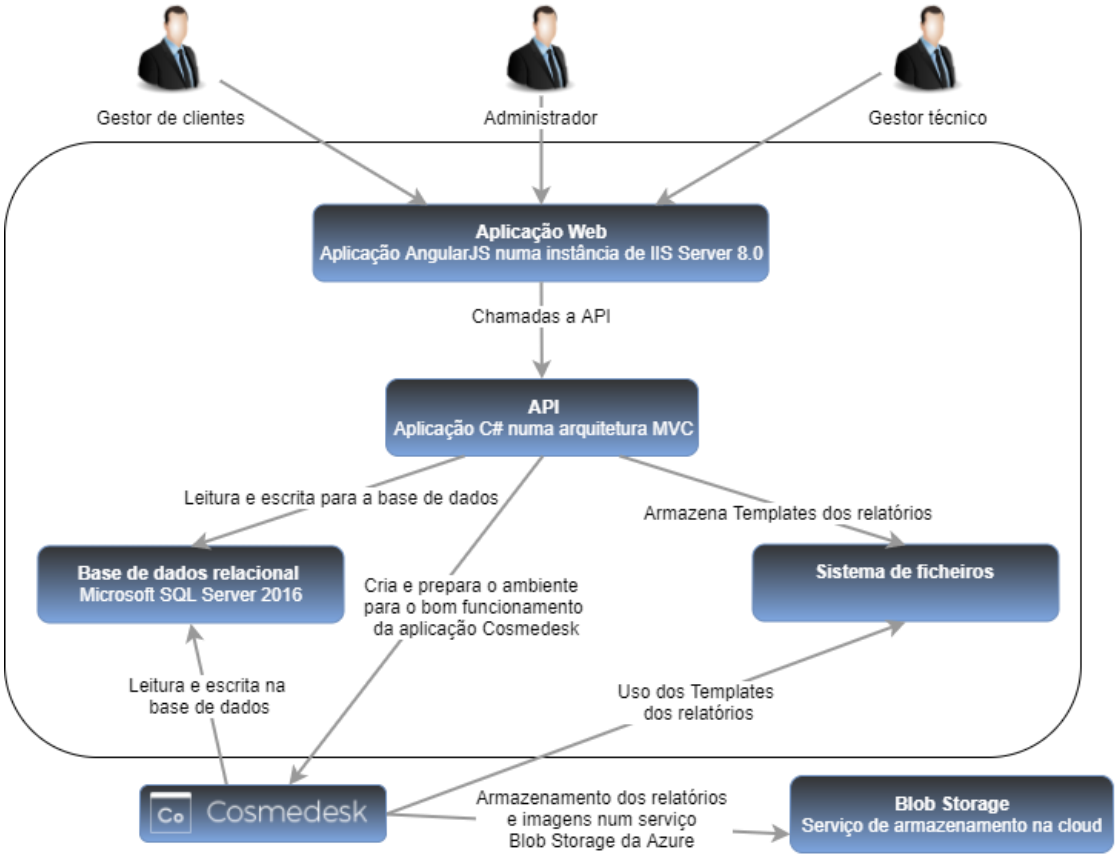


Figura 24: Arquitetura - Container

5.4.3 Components

Os componentes serão subdivididos em várias diagramas, um por cada contentor, uma vez que estes representam um diagrama significativamente maior, por isso optou-se por construir os diagramas em separado. Existindo assim quatro contentores na aplicação, podemos ver a constituição destes, mais em pormenor nos diagramas de componentes que se seguem.

Aplicação web

A aplicação WEB, Figura 25, é manipulada por três tipos de utilizadores que podem realizar operações nesta, inserindo dados nas vistas e navegando entre estas. O que define as vistas e os controladores são as *routes*, que inicializam determinado controlador consoante a vista. Os controladores comunicam com as vistas através do "$scope" que tem toda a informação necessária, esta informação transita entre os controladores e os serviços de cada entidade. Os serviços por sua vez comunicam com projeto WEB API, situado no contentor API, através do JSON.

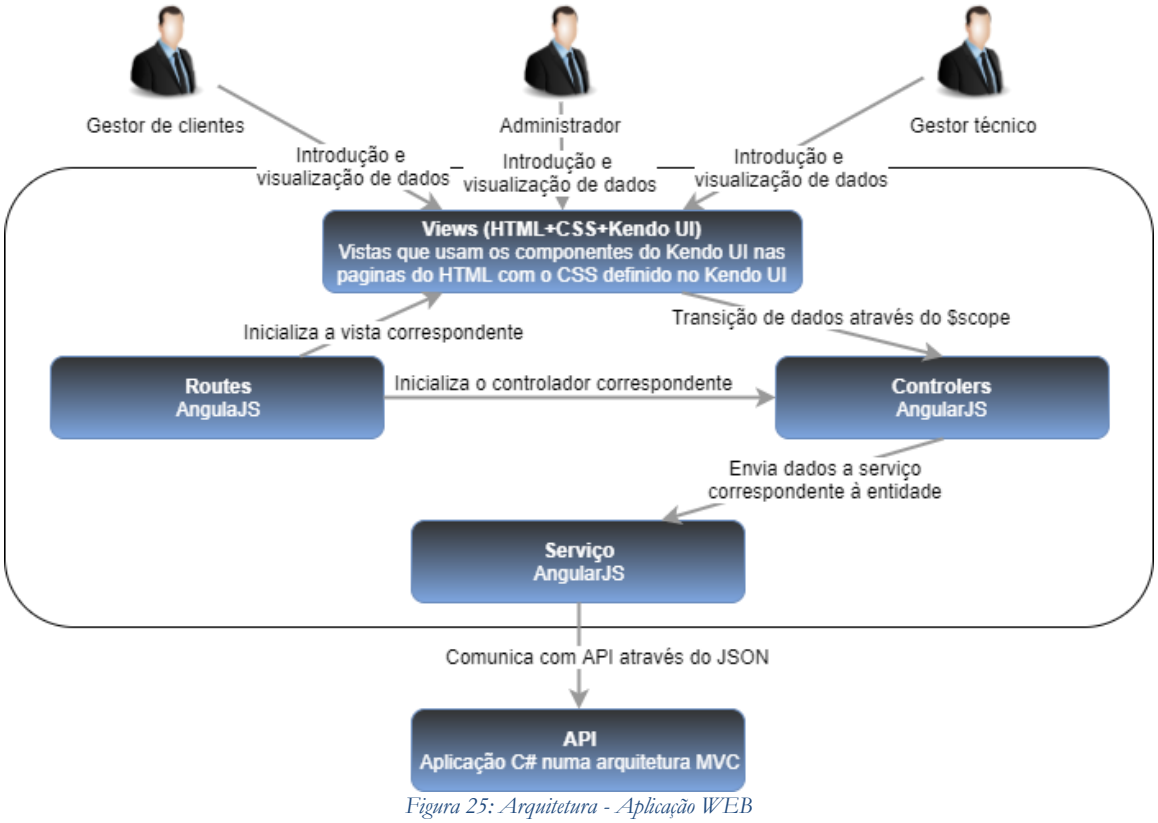


Figura 25: Arquitetura - Aplicação WEB

API

A API, Figura 26, é o conjunto dos projetos que por sua vez estão estruturados por camadas, cada uma destas é constituida por um ou mais projetos. Neste caso a camada apresentação é um projeto ASP.NET construido em C# e é um projeto baseado em API, que é uma ligação entre a aplicação WEB e a camada de negócio. A camada de negócio também é um único projeto, que trata de toda a lógica da aplicação, é este projeto que trata da criação de novas instâncias e das bases de dados para as aplicações "Cosmedesk" na criação de um novo domínio para um cliente. Tambem é este projeto que está interligado com a camada de dados, a camada de dados está dividida em dois projetos, pois existe a necessidade de comunicar com a base de dados do módulo e as bases de dados dos clientes.

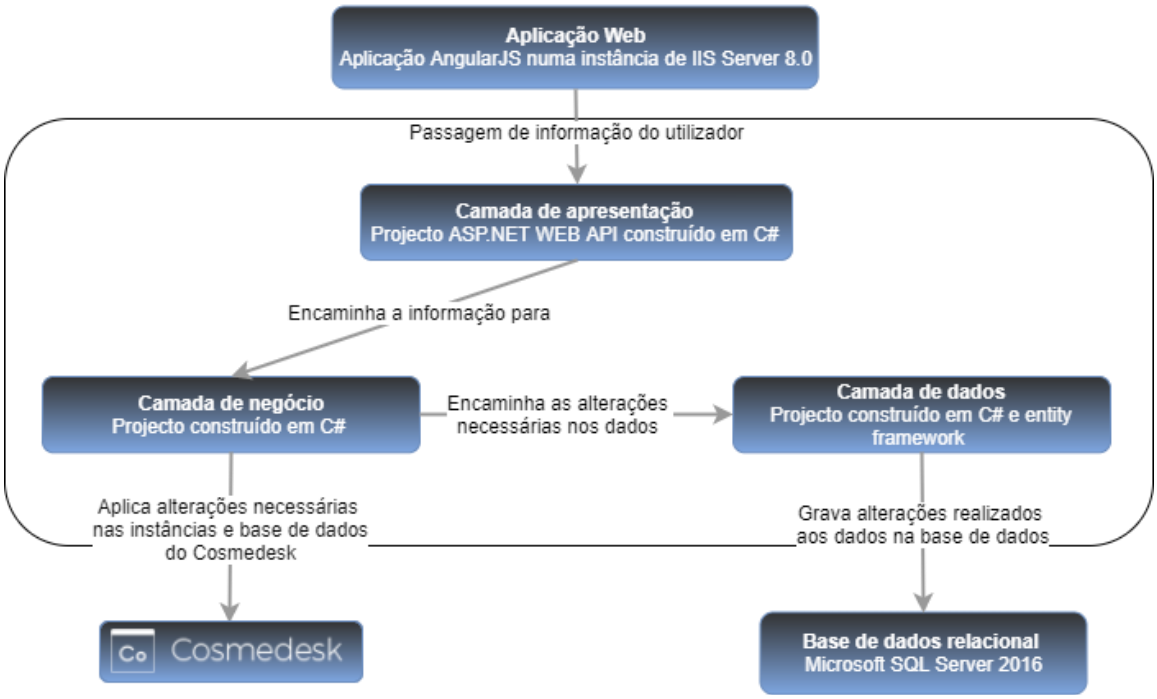


Figura 26: Arquitetura - API

Base de dados relacional

A base de dados relacional apresentada na Figura 27, contém várias bases de dados, uma destas é a base de dados do módulo *back office*, enquanto as restantes são criadas dinamicamente, para cada nova instância, porque há uma distinção entre a instância e o cliente, pois é possível criar várias instâncias para o mesmo cliente e como cada instância tem a sua base de dados, podem existem várias bases de dados para o mesmo cliente, mas sempre uma base de dados por instância. Por sua vez a aplicação também tem a necessidade de comunicar com duas bases de dados, uma das quais é a sua base de dados, que foi criada automaticamente pelo módulo, que trata por guardar toda a informação necessária, mas também tem a necessidade de consultar alguns dados da licença à base de dados do módulo.

Figura 27: Arquitetura - Base de dados relacional

Sistema de ficheiros

O contentor sistema de ficheiros, representado na Figura 28, representa as pastas que são dinamicamente criadas para cada instância, para guardar os modelos para cada um dos documentos. As pastas encontram-se na pasta "AppData" da máquina virtual, as pastas levam o nome do domínio que é único e não alterável, para fácil distinção, e dentro de cada pasta encontram-se cinco ficheiros no formato ".docx" que serão usados pela a aplicação "Cosmedesk" na geração dos relatórios.

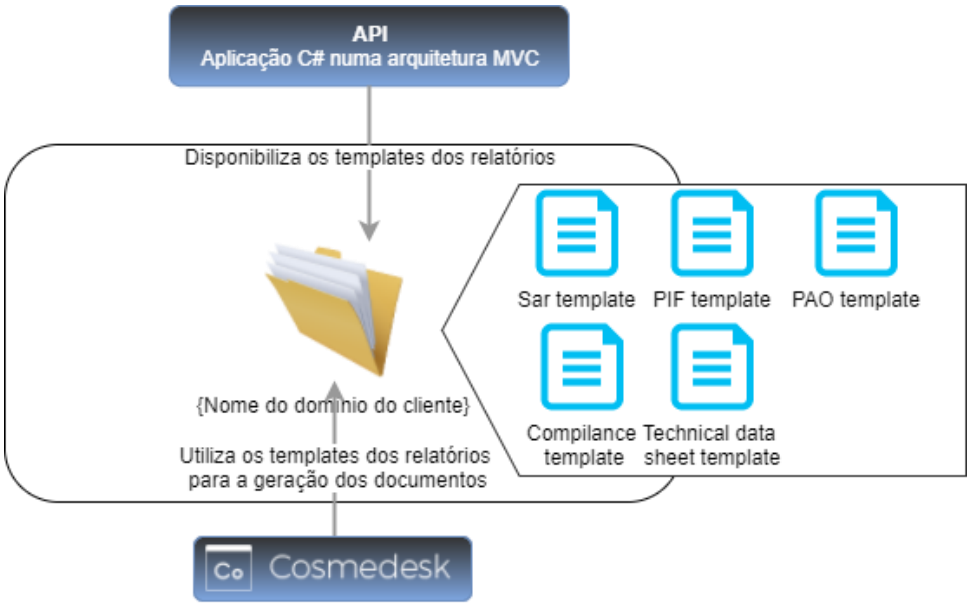


Figura 28: Arquitetura - Sistema de ficheiros

5.5 Resultado

Os resultados do presente estágio correspondem ao cumprimento dos principais objetivos que foram propostos no inicio do mesmo. Desta forma, destacam-se os seguintes objetivos alcançados:

- Desenvolvimento do um módulo *back office* nas tecnologias Microsoft.
- Integração do módulo com a aplicação existente "Cosmedesk".
- Gestão das instâncias das várias aplicações "Cosmedesk".
- Criação automática do ambiente, para as novas aplicações "Cosmedesk".
- Disponibilização do *URL* de registo na sua instância do "Cosmedesk" para o cliente.
- Gestão de licenças por instância.
- Gestão de clientes.
- Inicialização e publicação de dados para as instâncias dos clientes.
- Gestão de regras de legislação.
- Possibilidade de monitorização, disponibilizando os gráficos.

Também foram implementados alguns extras, nomeadamente algumas funcionalidades na aplicação "Cosmedesk":

- Módulo das notificações, que disponibiliza as notificações para cada cliente sobre os seus produtos.
- Validação automática dos produtos dos clientes, perante as regras, que correspondem à legislação em vigor, criadas no módulo, e notificação do cliente no caso de incumprimento de alguma legislação.

5.5.1 Requisitos cumpridos

Como uma das validações do trabalho terminado com sucesso, este será validado com os requisitos definidos inicialmente, é o cumprimento de todos os requisitos que comprovam que todas as tarefas foram realizadas com sucesso. Para a validação dos mesmos também serão apresentadas as diagramas no capítulo seguinte, que descrevem a aplicação resultante de maneira a informar as suas funcionalidades e que estes cobrem os requisitos inicialmente definidos.

Requisitos não funcionais

Código	Requisito	Prioridade	Implementado
RNF1	Manutenível	Obrigatório	Sim
RNF2	Sistema barato de construir	Obrigatório	Sim
RNF3	Reutilizável	Obrigatório	Sim
RNF4	Escalabilidade	Obrigatório	Sim
RNF5	Disponibilidade	Obrigatório	Sim
RNF6	Sistema fácil de aprender	Obrigatório	Sim
RNF7	Sistema memorável	Obrigatório	Sim
RNF8	Integridade	Obrigatório	Sim
RNF9	Confidencialidade	Obrigatório	Sim
RNF10	Autentificação	Obrigatório	Sim
RNF11	Sistema oferece feedback do seu estado	Obrigatório	Sim

Tabela 8: Requisitos não funcionais

Requisitos funcionais

Código	Requisito	Prioridade	Implementado
RF1	Aplicar alterações ao sistema	*Nice to have*	Não
RF2	Utilizador informado do estado da aplicação	Obrigatório	Sim
RF3	Listar instâncias	Obrigatório	Sim
RF4	Validar licenças	Obrigatório	Sim
RF5	Listar licenças	Obrigatório	Sim
RF6	Listar regras	Obrigatório	Sim
RF7	Comando da alteração de uma instância detetado	Obrigatório	Sim
RF8	Comando da alteração da informação do administrador detetado	Obrigatório	Sim
RF9	Comando de aplicação dos ajustes ao sistema detetado	*Nice to have*	Não

RF10	Comando de submissão de um novo administrador detetado	Obrigatório	Sim
RF11	Comando da submissão de uma nova licença detetado	Obrigatório	Sim
RF12	Comando de desativação da regra detetado	Obrigatório	Sim
RF13	Comando de suspensão de uma instância detetado	Obrigatório	Sim
RF14	Comando de suspensão de uma licença detetado	Obrigatório	Sim
RF15	Comando de população pelas bases de dados detetado	Obrigatório	Sim
RF16	Comando de extensão de uma licença detetado	Obrigatório	Sim
RF17	Comando de remoção de um administrador detetado	Obrigatório	Sim
RF18	Comando de submissão de uma nova instância detetado	Obrigatório	Sim
RF19	Comando de submissão da alteração de uma regra detetado	Obrigatório	Sim
RF20	Comando de submissão de uma nova regra detetado	Obrigatório	Sim
RF21	Disponibilização do gráfico do número dos relatórios gerados por dia, mês, ano e o cliente	*Nice to have*	Sim
RF22	Disponibilização do gráfico do número das instâncias ativas ao longo do tempo	*Nice to have*	Sim
RF23	Disponibilização do gráfico dos recursos (CPU e RAM) utilizados pelo sistema	*Nice to have*	Não

Tabela 9: Requisitos funcionais

Justificação para os requisitos não cumpridos

Os requisitos que não foram implementados e se houver alguma justificação válida, esta será apresentada aqui.

Para os requisitos com o código RF1 e RF9, que foram classificados com "*Nice to Have"*, estes não foram cumpridos, uma vez que a aplicação se encontra numa máquina virtual e está alojada no Azure, não havendo nenhuma configuração adjacente ao serviço/sistema. Sendo assim possível facilmente aplicar as alterações no próprio painel da *cloud* "Azure", num painel único e mais direcionado. A funcionalidade era colocada com a propriedade "*Nice to Have*", uma vez que o cliente não sentiu grande necessidade desta. Também os utilizadores da aplicação não vão ter um perfil informático, para aplicar as alterações a este nível. Relembrando que os utilizadores serão mais ao nível de gestão de clientes e aspetos técnicos dos produtos cosméticos/farmacêuticos, que requerem um perfil com o conhecimento químico e não informático.

Outro dos requisitos não cumprido, é o requisito com o código RF23, que não foi implementado, uma vez que trazia pouca informação para os utilizadores da aplicação e as pessoas com as necessidades podiam consultar dada informação no painel de controlo do "Azure" uma vez que este também fornece o histórico.

Extras implementados

Foram implementados alguns extras, nomeadamente na aplicação existente que sofreu alguns ajustes, como a disponibilização da nova funcionalidade que são as notificações ao utilizador. Esta funcionalidade será útil para informar os utilizadores sobre algumas novidades da aplicação ou do licenciamento, imaginando que se pretende informar o utilizador na existência de nova funcionalidade ou de um desconto em uma ou várias licenças num certo período. Também o utilizador irá receber a notificação se um dos seus produtos não passar na legislação.

No desenvolvimento do módulo teve-se em conta a possibilidade de existir a multi-língua que permite representar o módulo em várias línguas.

5.6 Diagramas

Nesta secção serão apresentados alguns diagramas que podem de certa forma validar todo o trabalho que foi realizado com sucesso. Apesar de não se mostrar todos os ecrãs da aplicação, estes são semelhantes entre si, para facilitar o uso do módulo uma vez que existe a necessidade de este ser memorável, e o utilizador poder aplicar a sua experiência dos ecrãs anteriores nos seguintes. Os diagramas que se seguem têm como objetivo, a informação geral do como a informação é apresentada aos utilizadores e as funcionalidades que o módulo tem, sendo possível ver mais na apresentação que será realizada pelo estagiário.

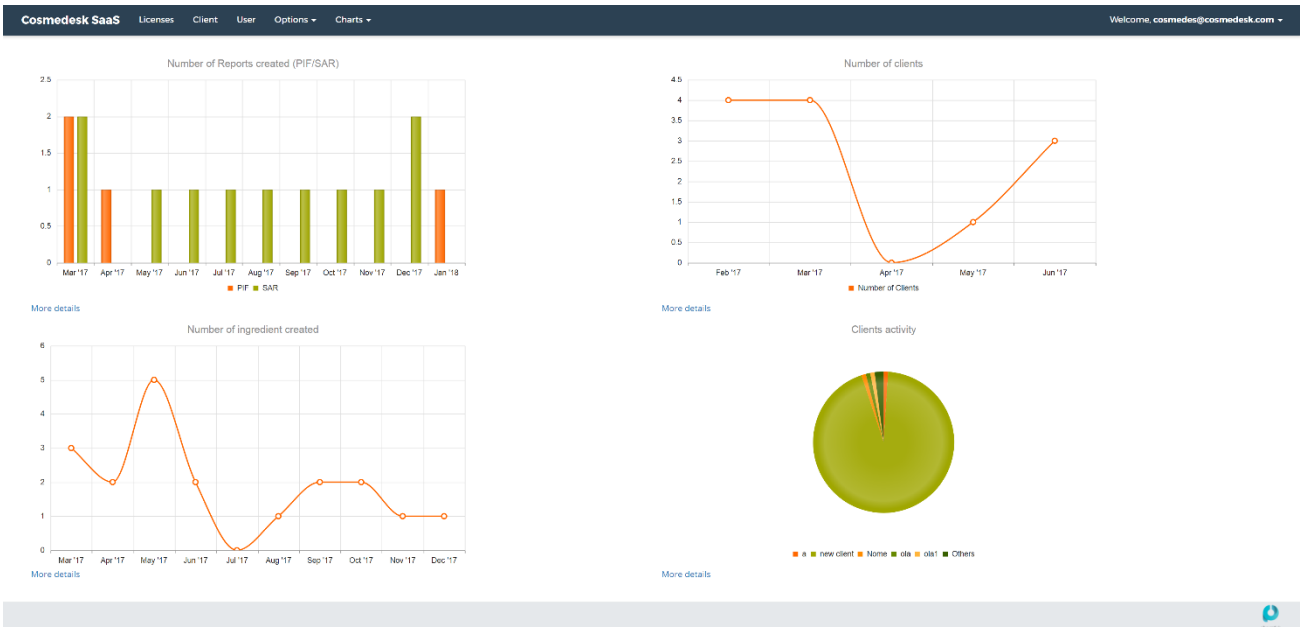


Figura 29: Diagramas – Dashboard

Como podemos ver na Figura 29, está representada a primeira página depois do utilizador realizar o login no módulo *back office*. Na parte superior da figura temos uma barra de navegação que contem as opções que o utilizador pode efetuar do lado esquerdo, enquanto do lado direito o utilizador irá ter um menu para realizar as alterações à sua conta, como por exemplo mudar password. Nesta página são apresentados quatro gráficos que disponibilizam algumas opções, como por exemplo esconder uma das partes do gráfico, isto é feito clicando sobre a legenda do item que queremos omitir. Também cada um dos gráficos tem um link por baixo que redireciona para o mesmo gráfico, mas com mais opções de manipulação.

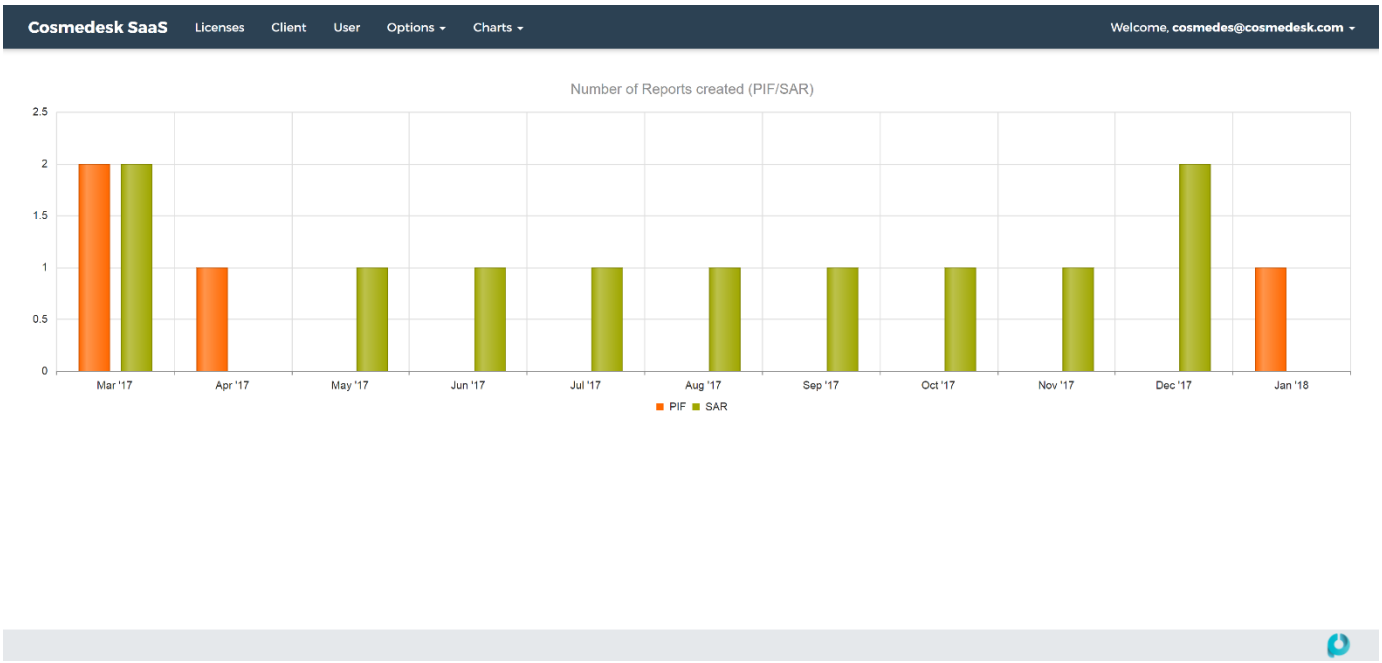


Figura 30: Diagramas - um dos graficos

A página com o gráfico representada na Figura 30, é um dos gráficos apresentados anteriormente, mas com mais opções disponíveis. Este gráfico permite alterações da exibição do seu conteúdo, como o anterior, mas também alteração da janela de vista, ou seja, podemos fazer zoom e consultar com mais detalhe o gráfico, esta janela também é movível depois de se fazer zoom, sendo possível mover essa para os lados. Esta janela é acessível através do menu dos gráficos onde podemos escolher o gráfico que pretendemos exibir, ou então a partir da página "Dashboard" e clicar sobre o link "Mais detalhe".

A barra de navegação tem uma lista de opções escondida, a Figura 31, representa as opções que o determinado utilizador pode efetuar.

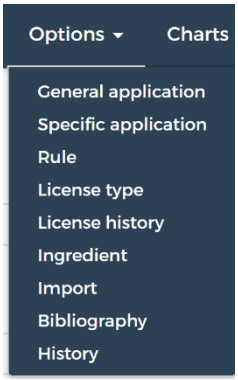


Figura 31: Diagramas - Opções

Existem mais duas listas das opções escondidas, a Figura 32, representa as opções disponíveis dos gráficos. As opções que estão apresentados na Figura 33, representam as funcionalidades sobre a conta do utilizador.

Figura 32: Diagramas - Charts

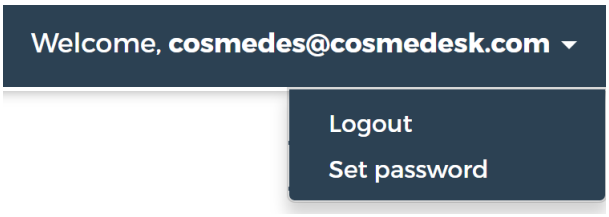


Figura 33: Diagramas - Opções sobre a conta do utilizador

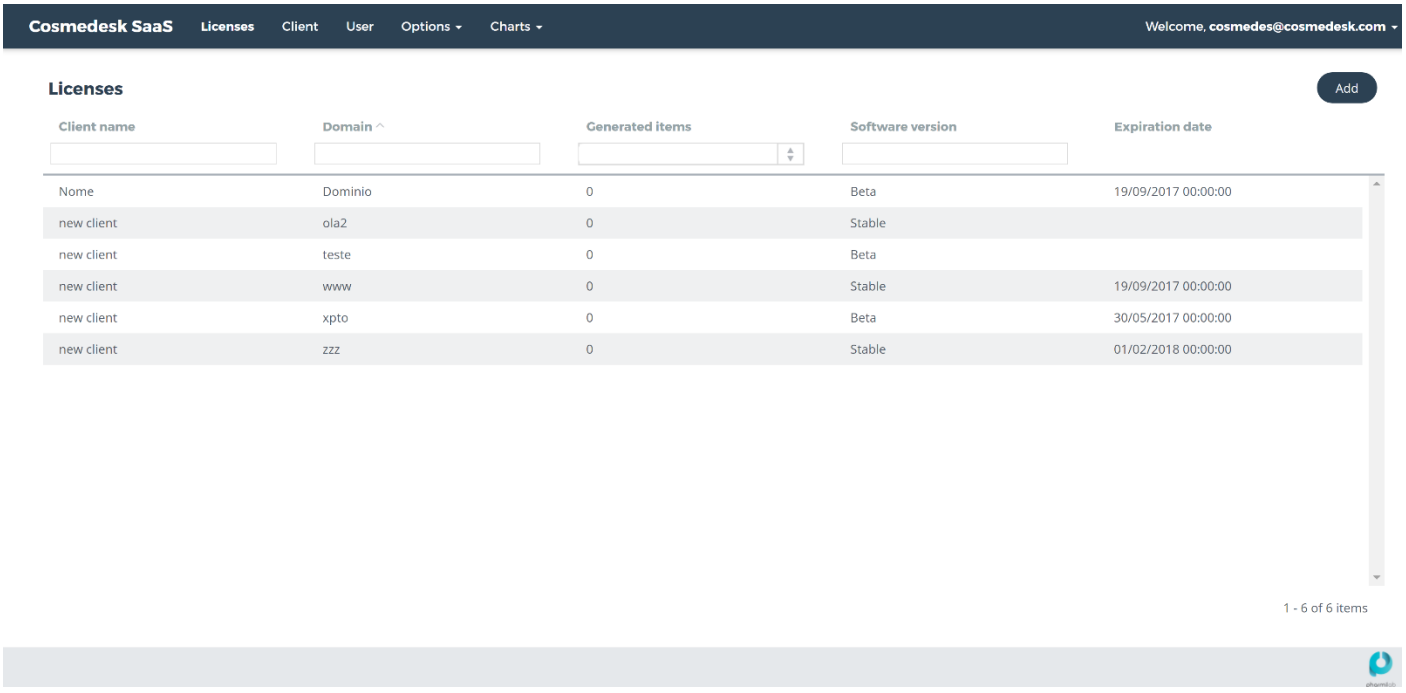


Figura 34: Diagramas - Licensas

Esta página permite visualizar os domínios e as licenças que cada utilizador tem, para mais informação é necessário clicar sobre a grelha, sobre o registo pretendido que nos irá redirecionar para uma página com a informação mais detalhada.

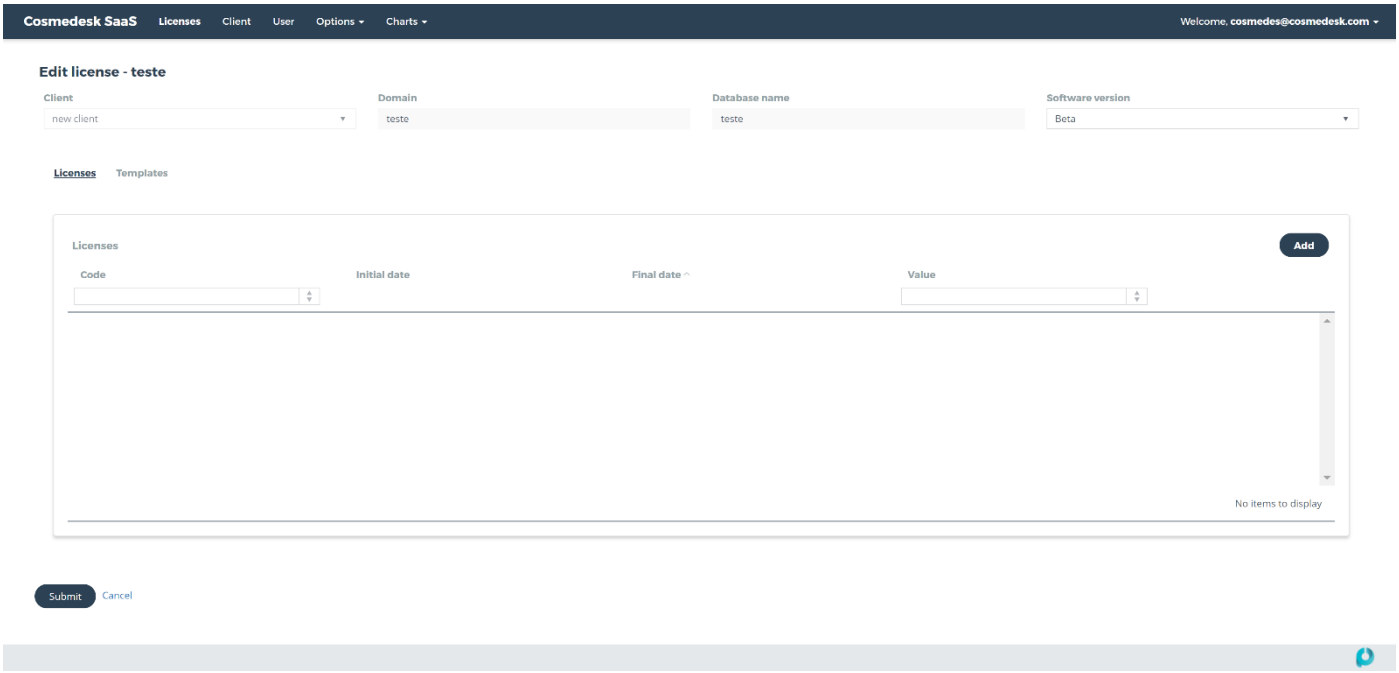


Figura 35: Diagramas - Licenças

A página apresentada na Figura 35, representa o formulário para a gestão de um dos domínios no qual existe a possibilidade de gerir todo o tipo de conteúdo relacionado com o a instância do cliente. Esta página contém alguma informação que não muda como o cliente ("Client"), o domínio ("Domain") disponibilizado para o cliente e o nome da base de dados ("Database name"). No campo domínio na realidade está representado o subdomínio da instância criada para o cliente. Os campos domínio e nome da base de dados, são únicos e não se podem repetir. O campo versão representa duas versões possíveis, a versão "Beta" e versão "Release", o utilizador pode facilmente mudar o cliente entre estas. Um pouco mais em baixo conseguimos visualizar dois separadores, "Licenses" e "Templates", o separador das licenças mostra-nos a tabela com todas as licenças que o cliente tem, uma grelha com o código das licenças, data inicial e final, como também o valor efetivo do pagamento. Enquanto o separador dos "Templates" mostra-nos os modelos dos relatórios existentes na aplicação, como podemos ver na Figura 36, coloca-nos a vista com os cinco modelos dos relatórios e o logotipo, clicando no botão "Upload" o utilizador pode escolher um novo modelo no formato ".docx" ou uma nova imagem para logotipo, enquanto o botão do "Download" descarrega o documento no formato submetido.

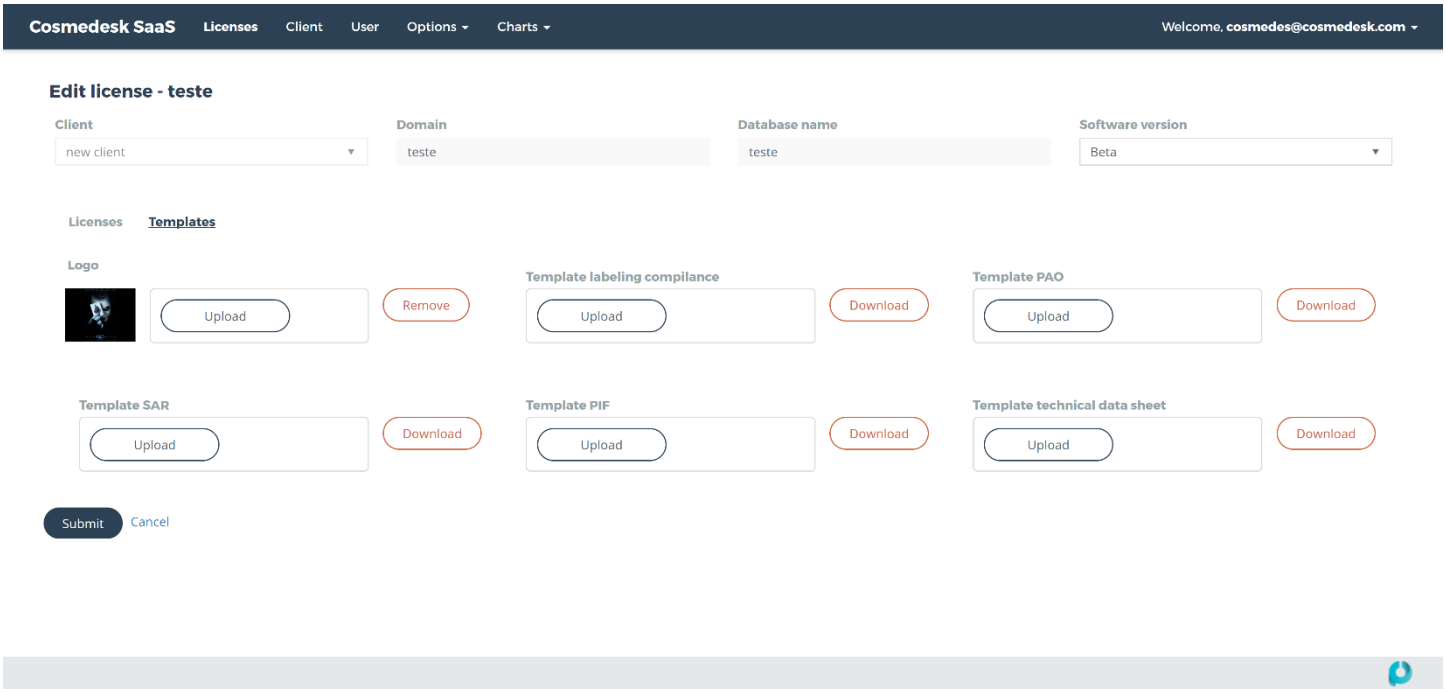


Figura 36: Diagramas - Licensas templates

Um outro formulário que é mais comum haver no módulo é o formulário do cliente, que está representado na Figura 37, este formulário é construído por campos simples, também esta entidade já permite a remoção,

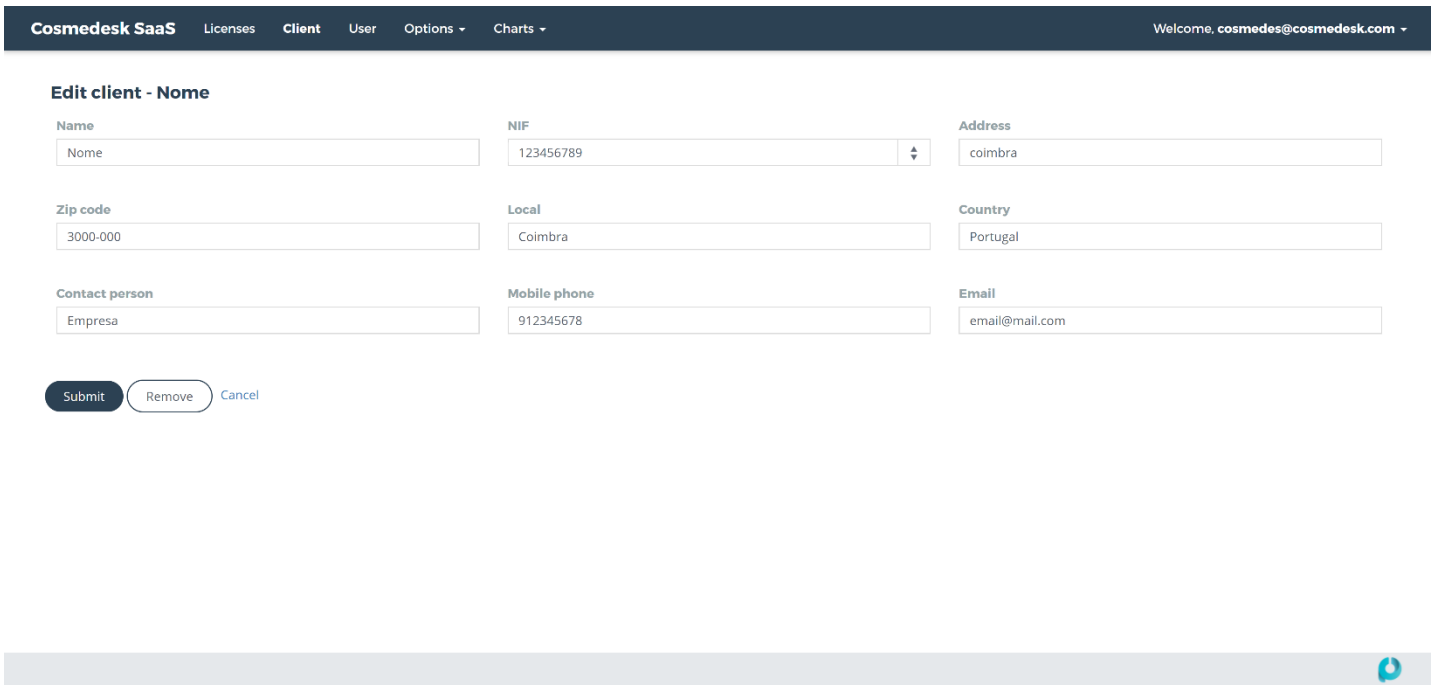


Figura 37: Diagramas - Cliente

O último formulário apresentado, é representado na Figura 38, é um formulário que só esta disponível a utilizadores com a função do administrador, disponibilizando todas as alterações efetuadas no módulo e as entidades que sofreram alterações, é possível ver as entidades clicando no link "Show entity" e será exibida a entidade correspondente.

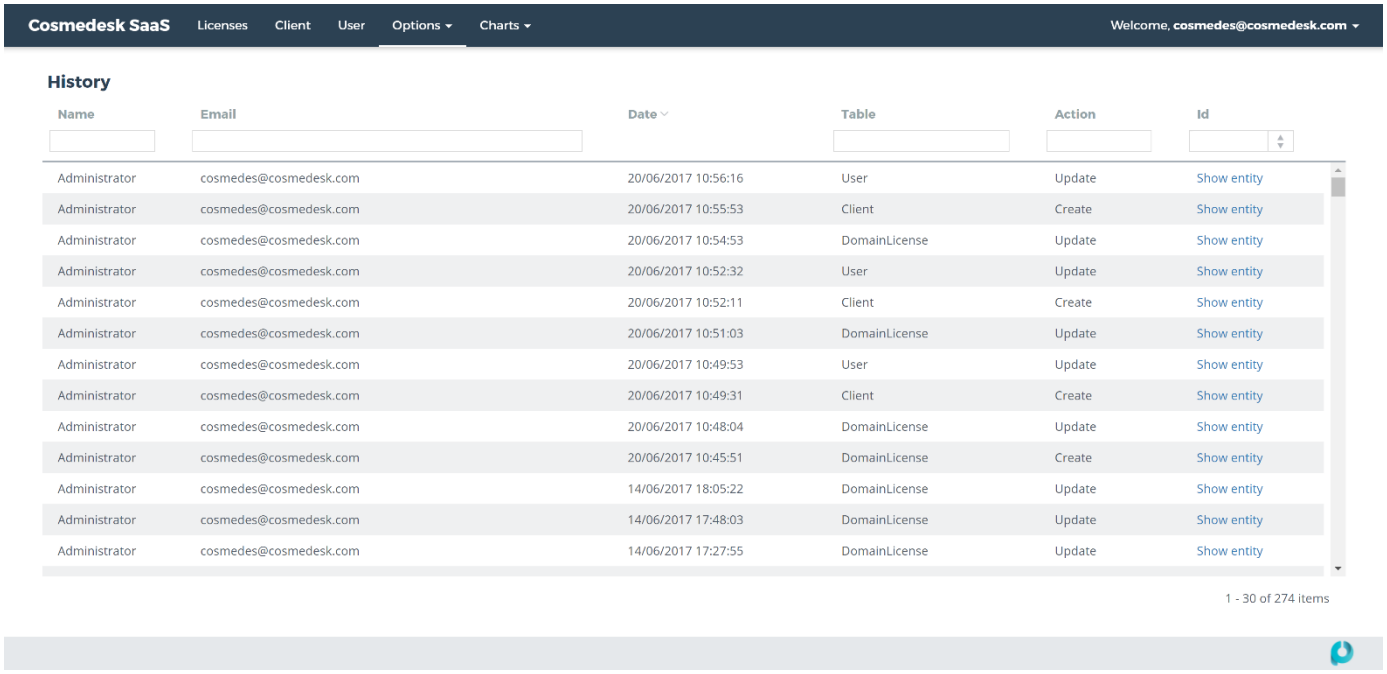


Figura 38: Diagramas - Historico

Capítulo 6 Testes

Quando se trata de alguma aplicação que está pronta para entrar em mercado esta tem de ser rigorosamente testada uma vez que os utilizadores podem procurar solução para os problemas nos concorrentes e para isso não acontecer as aplicações devem ir com o mínimo de bugs para produção. O módulo *back office* não é exceção apesar de ser uma aplicação *Back Office*, mas de qualquer modo esta irá gerir os sites dos clientes reais e lida com licenças apesar de não haver os pagamentos na aplicação os fundos são transmitidos na mesma pelos clientes pelo uso do "Cosmedesk" e o módulo têm de possibilitar concluir os desejos dos clientes em ter sua aplicação de gestão de relatórios. É possível consultar mais informação sobre os testes realizados no documento de testes situado nos anexos do relatório.

Para a máquina de testes foi utilizada uma máquina alojada na "Azure" com as seguintes caraterísticas:

- Modelo: Ds2_v2
- Sistema operativo: Windows Server 2012
- CPU: 2 cores
- RAM: 7GB
- Disco SSD: 14GB

6.1 Testes unitários

Os testes a unidade foram automatizados e escritos com ajuda da "Microsoft Unit Test Framework", que permitem realizar testes a unidade no "Visual Studio". Os testes foram feitos para funções principais que são mais complexas da camada do negocio, considerando não ser necessário a verificação das funções que façam simples redireccionamentos para outras, com poucas ou nenhumas manipulações ao(s) objeto(s).

Para verificar se o retorno era esperado foram usadas as funções de validação da *framework*, alguns exemplos das funções mais usadas são:

- Assert.IsTrue(bool condition);
- Assert.IsInstanceOfType(Object object, Type expectedType);
- Assert.IsNotNull(Object object);
- Assert.AreEqual(Object expected, Object actual);

Os testes unitários realizados foram executados com sucesso isso podemos confirmar na Figura 39, representada em baixo. Os testes unitários representam uma cobertura de 73% das classes criadas, uma vez que apesar de os testes foram realizados na camada de negocio foi possível testar os retornos das classes de camada de dados, restando assim as classes da

camada de apresentação. Uma vez que estas classes da apresentação, são de uma simplicidade extrema, como podemos ver na Figura 40, representada em baixo.

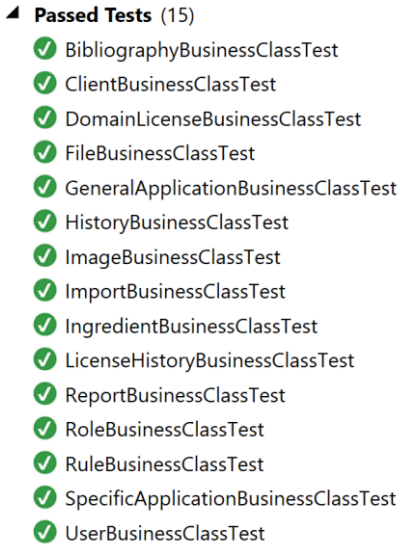


Figura 39: Testes unitários

```
namespace CosmedeskSAS.WebApi.Controllers.Entities
{
    0 references
    public class BibliographyController : BaseApiController
    {

        #region Public Lists

        0 references | 0 requests, ? live | 0 exceptions, ? live
        public IQueryable<BibliographyListDTO> Get()
                => BibliographyQuery.GetAll(DatabaseContext);

        0 references | 0 requests, ? live | 0 exceptions, ? live
        public async Task<BibliographyUpdateDTO> Get(int id)
            => await BibliographyQuery.GetBibliography(DatabaseContext, id);

        #endregion

        #region Post

        0 references | 0 requests, ? live | 0 exceptions, ? live
        public async Task<IHttpActionResult> Post([FromBody]BibliographyUpdateDTO editedObject)
        {
            await BibliographyManipulator.Update(DatabaseContext, 0, editedObject);
            return Ok();
        }

        #endregion

        #region Put

        0 references | 0 requests, ? live | 0 exceptions, ? live
        public async Task<IHttpActionResult> Put(int id, [FromBody]BibliographyUpdateDTO editedObject)
        {
            await BibliographyManipulator.Update(DatabaseContext, id, editedObject);
            return Ok();
        }

        #endregion
    }
}
```

Figura 40:. Uma classe da camada da apresentação

6.2 Testes de integração

Os testes manuais de integração foram feitos para testar a aplicação "Cosmedesk back office" no ambiente real, encontrando os possíveis bugs tentando resolver os mesmos mais breve possível uma vez que a aplicação não pode ter bugs que impossibilitam o real funcionamento da aplicação. Foram considerados três casso possíveis:

Tipo 1 Os resultados pretendidos/esperados obtidos;
Tipo 2 Os resultados não pretendidos/esperados obtidos;
Tipo 3 Os resultados não esperados, mas que não causam estragos.

No final dos 113 testes realizados foram encontrados seguintes resultados:

Tipo 1 107 testes obtiveram o resultado esperado
Tipo 2 4 testes obtiveram os resultados não esperados que podiam causar os problemas no sistema
Tipo 3 2 testes obtiveram a classificação de resultados não esperados, mas sem estragos causados no sistema.

No entanto todos os testes de Tipo 2 e Tipo 3 foram resolvidos com sucesso, dando prioridade obvia a testes do Tipo 2.

6.3 Testes de Sistema

Os testes de sistema são mais que importantes pois o sistema neste caso especifico é mais do que "Cosmedesk back office" também engloba todas as aplicações "Cosmedesk" que são criados a partir do "Cosmedesk back office". As anotações são semelhantes a notações utilizados nos testes de integração.

Foram realizados ao todo oito testes, não detetando erros, uma vez que é considerado o erro quando o módulo ou aplicação não se comportam da maneira esperada e não fazem o procedimento especifico nas operações realizados. O bom resultado nesta fase se refere a muito *debugging* executado na altura da implementação uma vez que é uma das mais importantes fazes do projeto

Com tudo é possível consultar mais detalhe no documento de testes na secção de "Testes de sistema".

6.4 Testes de Performance

Os testes de performance não foram realizados uma vez que o Cosmedesk back office irá ter poucos utilizadores, provavelmente não passará dos cinco utilizadores. Sendo assim não irá

haver a carga suficiente na máquina para causar problemas da performance. Também não existe comparação possível da configuração manual e configuração que realiza o Cosmedesk back office, uma vez que o estagiário não tem noção do tempo que podia ser necessário para configuração manual de uma nova instância, e qualquer que fosse o tempo a configuração realizada pelo Cosmedesk back office será sempre constante para todos os utilizadores, sem fator humano e quase instantânea.

Capítulo 7
Plano de Trabalho e Implicações

7.1 Planeamento

Esta secção irá descrever todo o planeamento realizado no decorrer do respetivo estágio.

7.1.1 Primeiro semestre

A calenderazição inicial prevista tinha como tarefas:

- A aprendizagem e primeira iteração com as tecnologias da microsoft
    - Aprendizagem de tecnologias
    - Realização dos exercicios práticos simples
- Escrita do relatório intermédio
    - Introdução
    - Conceitos relacionados
    - Objetivos do estágio e método de abordagem
- Documentação necessária
    - Riscos
    - Requisitos

O *Gantt,* representado em baixo, apresenta a planificação mais detalhada do planeamento, do estágio, do primeiro semestre.

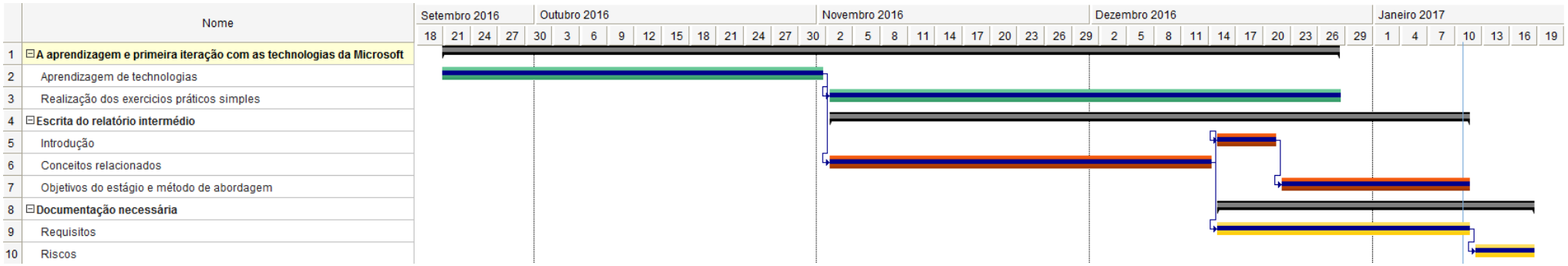


Figura 41: Gant - Planificação primeiro semestre

7.1.2 Segundo Semestre

A calenderazição inicial prevista, para o segundo semestre, tem como tarefas:

- Desenvolvimento
  - Gestão de clientes e criação das instâncias automáticas
  - Gestão das regras da legislação
  - Gestão das licenças
  - Gerir administradores
  - Monitorizar o sistema
- Testes
- Relatório
  - Trabalho atual e resultados preliminares
  - Testes
  - Conclusão

Será apresentado o Gantt relativo à planificação do segundo semestre previsto na Figura 42, tal como o que mudou da planificação do segundo semestre na Figura 43. Como o estagiário não conhecia muito bem as tecnologias e a sua interligação, também não tem muita experiência nas estimativas, como o resultado, obtém-se novas tarefas, como por exemplo a "Criação dos componentes", que resulta na interligação do "Kendo UI" com "AngularJS", mas que diminui o tempo de geração das páginas HTML para o *frontend*. No caso dos testes o estagiário no inicio não tinha a noção dos testes que eram para realizar, como o resultado a divisão da tarefa em três partes dando mais informação útil ao leitor.

O *Gantt* representado em baixo, apresenta a planificação mais detalhada do planeamento, do estágio, do segundo semestre.

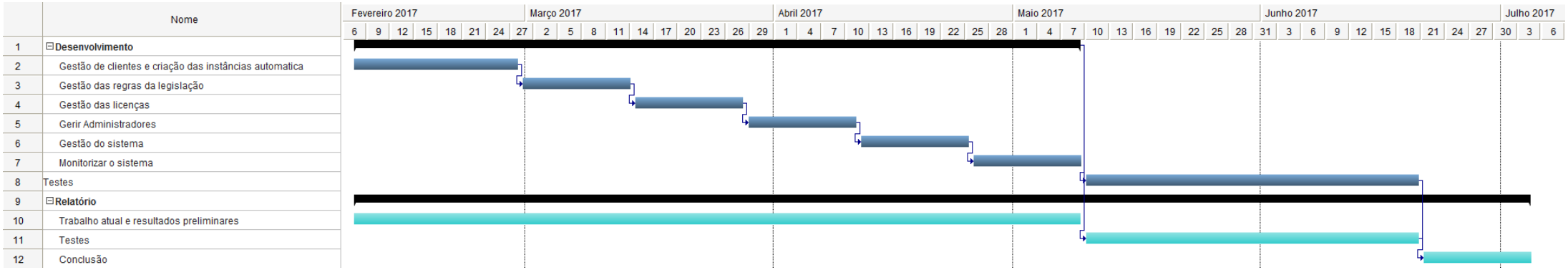


Figura 42 Gantt - Planificação segundo semestre

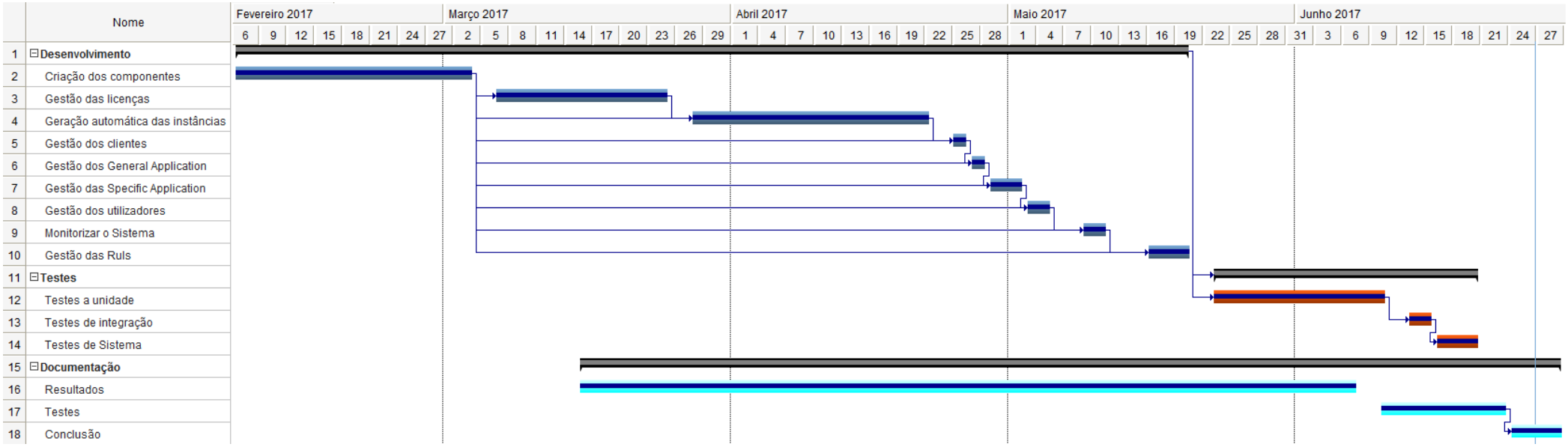


Figura 43: Gantt - Segundo semestre real

Capítulo 8
Conclusão

Terminando o período do estágio, é importante efetuar uma crítica ao trabalho realizado. Este capítulo servirá principalmente para concluir o que foi feito, o que correu bem e o que poderia correr melhor. Também retrata a experiência obtida ao longo do estágio pelo estagiário.

8.1 Trabalho realizado

É possível afirmar que o estágio foi concluído com sucesso, pois todos os objetivos propostos foram alcançados com sucesso. Apesar de no inicio do estágio terem existido algumas dificuldades, este terminou com todos os requisitos obrigatórios realizados com sucesso, desenvolvendo a solução com grande autonomia por parte do estagiário. A solução foi construída perante regras/normas internas do LIS, para que se houver a necessidade de alguém pegar e continuar no que já foi implementado, conseguir interpretar o código sem grandes dificuldades. A solução resultante foi frequentemente validada com a equipa de gestão do projeto que considerou que os objetivos do estágio foram alcançados com sucesso. As decisões foram tomadas com grande parte no que aconteceu, foram fundamentadas através de uma análise cuidada nos conceitos relacionados, mas também neste capítulo esteve presente um estudo das soluções possíveis no caso de haver a necessidade de grandes alterações no sistema, seja isso alteração da arquitetura perante o crescimento ou a migração para outra cloud concorrente perante a necessidade de melhorar a disponibilidade da aplicação ou outras necessidades. O *software* implementado foi testado e demonstrado. Todo o trabalho realizado foi devidamente documentado e disponibilizado à equipa do LIS que gerre o sistema Cosmedesk.

8.2 Contributo

O estágio deu um contributo enorme ao estagiário em diversos contextos. Talvez o principal seja a oportunidade de conhecer a realidade de um meio mais próximo a um mundo empresarial aplicando o conceito de engenharia num projeto real, que difere do que o estagiário conseguia obter anteriormente no meio do estudo. Um outro contributo de uma extrema importância para o futuro advém de todos os obstáculos ultrapassados, durante o todo projeto. A maturidade, autonomia e capacidade de lidar com adversidades, são outras caraterísticas de extremo valor, que foram adquiridos durante esta experiência, que serão uteis no futuro, não só ao nível profissional, mas também a nível pessoal.

Ao nível técnico o estágio também deu um contributo muito bom. Para alem de ter sido o primeiro contacto com diversas tecnologias, ferramentas e *frameworks* até então desconhecidas ou nunca usadas. A complexidade do problema e consequente estudo para solucionar, acabaram por enriquecer bastante as competências do estagiário.

O contributo do estagiário para o projeto Cosmedesk back office é também um aspeto a contar. A solução implementada colocou o sistema Cosmedesk num patamar acima e que abriu novas portas. O Cosmedesk agora tem a facilidade de gerir todas as suas instâncias sem a necessidade de intervenção humana na criação/alteração do seu conteúdo para estes. Também a possibilidade de gerir Clients, License, General Application, Specific Application, Ingredients, Import of ingredients, Ruls, Bibliography e possibilitando obter o estado do negocio consultando vários gráficos informativos.

Por ultimo o trabalho do estagiário contribui para encontrar alguns problemas/erros no sistema, sendo estes reportados para a equipa responsável pelo Cosmedesk e corrigidos posteriormente.

8.3 Reflexão critica

Mais importante do que falar dos bons momentos é enunciar os aspetos menos bons, que ocorreram durante o estagio, de forma a evitar a sua ocorrência no futuro.

A fase de implementação de interfaces web teve arranque lento devido a falta do conhecimento das tecnologias, para este efeito, por parte do estagiário. Esta situação poderia ter sido colmatada através da decisão sobre o uso das tecnologias a utilizar na fase pré-desenvolvimento e estudo intensivo ainda nesta fase por parte do estagiário, o que não aconteceu com Angular JS, o uso do qual foi definido só no segundo semestre.

Um outro ponto importante prende-se a estimativa de horas para a realização de tarefas e a sua ordem, principalmente a nível da implementação. A falta de experiência neste campo contribuiu para que em alguns casos, a estimativa das tarefas de baixa complexidade se tenha desviado dos valores reais obtidos e a sua ordem alterada. Apesar de esta dificuldade ao ter muito impacto no âmbito do estágio, é um aspeto que deve estar em conta para futuro.

8.4 Trabalho futuro

Dando por terminado o estagio, conseguiu-se cumprir o plano do trabalho, mas mesmo assim existe algum trabalho futuro a realizar, sendo estes aspetos não menos importantes do que o trabalho já realizado durante o estágio.

Uma das principais tarefas a realizar depois de estagio ser concluído, na minha opinião, é os testes no ambiente real. Uma vez que os testes realizados se focaram no ambiente de testes que não é igual, apesar de muito próximo, a ambiente real.

Uma outra tarefa mesmo muito importante, é repensar as regras, uma vez que as regras de maneira que estão implementadas funcionam perfeitamente, mas para os ingredientes que foram introduzidos no módulo, uma vez que estes estão relacionadas com as regras diretamente, sendo os ingredientes introduzidos no cliente, e como estes são introduzidos pelos humanos, pode haver o risco de introdução dos nomes não 100% igual ao "CosIng" (34). Desta forma podemos não encontrar um ingrediente que esteja na regra e no relatório, mas os nomes são um pouco diferentes por exemplo, um possível caso é este no CosIng um ingrediente ter o nome "HYDROXYETHYL-P-PHENYLENE-DIAMINE SULFATE" e o cliente introduzir: "HYDROXYETHYL-P-PHENYLENE-DIAMINE-SULFATE" o que já invalida o reconhecimento do ingrediente pelo nome. Sendo assim tem que haver uma nota a dizer que existem as notificações para os documentos que utilizaram os ingredientes que foram importados para o cliente. Como o cliente não demonstrou vontade de colocar esta nota para os seus clientes, uma outra hipótese era publicar os ingredientes para todos novos os clientes, como é o caso com as aplicações genéricas e especificas, aí também existe uma inconformidade com o modelo do negocio uma vez que este iria entrar em confronto com o pensamento de comercializar os ingredientes, e os seus clientes pagarem pelos ingredientes importados para eles. Mas se o módulo for utilizado e o cliente enganar no nome, estando a dizer que "Nos notificamos se o seu produto for confrontado pela nova legislação", pode ser mal-aceite se o cliente não recebera as notificações e pode reclamar.

Uma outra tarefa que pode ser realizada é mudar os CSS do módulo para que estes seja mais semelhante a Cosmedesk, uma vez que esta usa duas aplicações do mesmo grupo, tem de haver alguma semelhança entre estes.

Por último, realizar os testes de desempenho, apesar de estes podem dizer pouco uma vez que não é boa opção de comparar o trabalho manual com o módulo que tem todo o processo de criação de instâncias e a base de dados todo automático, até a publicação das aplicações genéricas e especificas é feita automaticamente.

Existindo assim claramente, uma série de novas tarefas a realizar depois de o período do estagio terminar.

Referências

1. Microsoft Azure. Virtual Machines. [Online] [Citação: 27 de 12 de 2016.] https://azure.microsoft.com/en-us/services/virtual-machines/.

2. Seen, Wheeler,and Sunny, Deng. Azure Load Balancer overview. *docs.microsoft.com.* [Online] Azure, 22 de 08 de 2016. [Citação: 05 de 01 de 2017.] https://docs.microsoft.com/pt-pt/azure/load-balancer/load-balancer-overview.

3. Communityof Azure. What is Azure App Service? *docs.microsoft.com.* [Online] Azure, 02 de 12 de 2016. [Citação: 05 de 01 de 2017.] https://docs.microsoft.com/en-us/azure/app-service/app-service-value-prop-what-is.

4. Rabeler, Carl. Elastic pool of Databese. [Online] 14 de 12 de 2016. [Citação: 27 de 12 de 2016.] https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-pool.

5. Microsoft Azure. SQL Database. [Online] [Citação: 27 de 12 de 2016.] https://docs.microsoft.com/en-us/azure/sql-database/.

6. Macy, Marsh. Microsoft Azure Storage. [Online] 08 de 12 de 2016. [Citação: 2016 de 12 de 27.] https://docs.microsoft.com/en-us/azure/storage/storage-introduction.

7. Serra, James. Redundancy option in Azure Blob Storage. [Online] 05 de 11 de 2015. [Citação: 27 de 12 de 2016.] http://www.jamesserra.com/archive/2015/11/redundancy-options-in-azure-blob-storage/.

8. Gentz, Mimi. DocumentDB. [Online] 16 de 11 de 2016. [Citação: 27 de 12 de 2016.] https://azure.microsoft.com/en-us/documentation/articles/documentdb-introduction/.

9. Microsoft Azure. Application gateway. [Online] [Citação: 29 de 12 de 2016.] https://azure.microsoft.com/en-us/services/application-gateway/.

10. Apolinário, Vinícius. Auto-Scale de Máquinas Virtuais no Microsoft Azure. [Online] 17 de 07 de 2014. [Citação: 29 de 12 de 2016.] http://admderedes.azurewebsites.net/?p=5011.

11. Microsoft Azure. Praicing calculator. [Online] [Citação: 25 de 11 de 2016.] https://azure.microsoft.com/en-us/pricing/calculator/.

12. Amazon. Amazone EC2. [Online] 2016. [Citação: 27 de 12 de 2016.] https://aws.amazon.com/pt/ec2/instance-types/.

13. Amazone. Amazon Elastic Block Store. [Online] [Citação: 27 de 12 de 2016.] https://aws.amazon.com/pt/ebs/.

14. Amazon. Amazon RDS SQL Server. [Online] [Citação: 2016 de 12 de 27.] https://aws.amazon.com/pt/rds/sqlserver/.

15. —. Amazone DynamoDB. [Online] [Citação: 2016 de 12 de 27.] https://aws.amazon.com/pt/documentation/dynamodb/.

16. —. Amazone Load Balancing. [Online] [Citação: 27 de 12 de 2016.] https://aws.amazon.com/pt/elasticloadbalancing/.

17. Amazone. SIMPLE MONTHLY CALCULATOR. [Online] [Citação: 23 de 11 de 2016.] http://calculator.s3.amazonaws.com/index.html.

18. A KAOS Tutorial. *Objectiver*. [Online] 18 de 10 de 2007. [Citação: 02 de 01 de 2017.] http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf.

19. *Goal-Oriented Requirements Engineering: An Overview of the Current Research* . Lapouchnian, Alexei. Toronto : Department of Computer Science: University Of Toronto, 2005.

20. Objectiver. *Objectiver*. [Online] Respect-IT sa. [Citação: 02 de 01 de 2017.] http://www.objectiver.com/.

21. Santos, Do Carlos dos. Fundamentos do Entity Framework 4. *Fundamentos do Entity Framework 4*. [Online] Microsoft, 05 de 2012. [Citação: 13 de 03 de 2017.] https://msdn.microsoft.com/pt-br/library/jj128157.aspx.

22. Angular. Angular JS. *Angular JS*. [Online] https://angularjs.org/.

23. Telerik. Kendo UI. *Teletik.com*. [Online] Telerik. http://www.telerik.com/kendo-ui.

24. Wikipedia Angular JS. *Wikipedia*. [Online] Wikipedia, 5 de 3 de 2017. [Citação: 22 de 05 de 2017.] https://pt.wikipedia.org/wiki/AngularJS#Objetivos.

25. Kendo UI. *Telerik*. [Online] [Citação: 22 de 05 de 2017.] http://www.telerik.com/kendo-ui.

26. Entity Framework (EF) Documentation. *Microsof*. [Online] 23 de 10 de 2016. [Citação: 22 de 05 de 2017.] https://msdn.microsoft.com/en-us/library/ee712907(v=vs.113).aspx.

27. Entity Framework. *Wikipedia*. [Online] 07 de 05 de 2017. [Citação: 22 de 05 de 2017.] https://en.wikipedia.org/wiki/Entity_Framework .

28. C#. *TechTarget*. [Online] [Citação: 22 de 05 de 2017.] http://searchwindevelopment.techtarget.com/definition/C.

29. ASP.NET MVC. [Online] https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview.

30. Server Class. *Microsoft*. [Online] [Citação: 22 de 05 de 2017.] https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.smo.server.aspx.

31. Microsoft.Web.Administration Namespace. *Microsoft*. [Online] [Citação: 22 de 05 de 2017.] https://msdn.microsoft.com/en-us/library/microsoft.web.administration(v=vs.90).aspx.

32. Carmelo, Pulvirenti. How many web applications per application pool. *Microsoft Developer*. [Online] 22 de 03 de 2013. [Citação: 04 de 06 de 2017.] https://blogs.msdn.microsoft.com/carmelop/2013/03/22/how-many-web-applications-per-application-pool/.

33. Brown, Simon. The C4 software architecture model. *structurizr*. [Online] [Citação: 10 de 06 de 2017.] https://www.structurizr.com/help/c4.

34. Cosing. *Cosmetic ingredient database*. [Online] [Citação: 26 de 06 de 2017.] https://ec.europa.eu/growth/sectors/cosmetics/cosing_pt.

35. Mary, David. Autoscale Virtual Machine. *Microsoft Azure*. [Online] [Citação: 30 de 11 de 2016.] https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-autoscale-overview.

36. Microsoft Azure. Vertical autoscale with Virtual Machine Scale sets. [Online] 06 de 12 de 2016. [Citação: 29 de 12 de 2016.] https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-vertical-scale-reprovision.

37. Carl Rabeler, Sunny Deng. Design patterns for multitenant SaaS applications and Azure SQL Database. [Online] 24 de 08 de 2016. [Citação: 29 de 12 de 2016.] https://docs.microsoft.com/pt-pt/azure/sql-database/sql-database-design-patterns-multi-tenancy-saas-applications.

38. Carraro, Frederick Chong and Gianpaolo. Architecture Strategies for Catching the Long Tail. [Online] 04 de 2006. [Citação: 29 de 12 de 2016.] https://msdn.microsoft.com/en-us/library/aa479069.aspx.

39. Frederick Chong, Gianpaolo Carraro, and Roger Wolter. Multi-Tenant Data Architecture. [Online] 06 de 2006. [Citação: 29 de 12 de 2016.] https://msdn.microsoft.com/en-us/library/aa479086.aspx.

Anexos

Anexo A - Documento de requisitos

O documento de requisitos apresenta a técnica utilizada e o resultado obtido de forma mais detalhada.

Anexo B - Documento de riscos

O documento de riscos apresenta os riscos obtidos e descrição mais detalhada do que apresentasse no relatório do estágio presente.

Anexo C - Documento de testes

O documento de testes representa todos os testes efetuados ao sistema em maior detalhe do que apresentado no relatório do estágio presente.

Anexo A

Documento de Requisitos

Levantamento e analise de Requisitos

Mestrado em Engenharia Informática

2016/2017

CosmeDesk: Módulo SaaS

Modelos KAOS

(Versão 1.1)

13/01/2017

Estagiário:
Ivan Harasym
2011168230
harasym@student.dei.uc.pt

Orientadores:
Prof. Filipe Araújo
filipius@dei.uc.pt
Departamento de Engenharia Informática, Universidade de Coimbra

Eng. Alcides Manuel de Almeida Marques
alcides.marques@ipn.pt
Instituto Pedro Nunes

Tabela de revisões

Versão	Data	Autores	Descrição
2.0	19/04/2017	IH	Ultima revisão e melhor descrição da técnica utilizada
1.5	15/01/2017	IH	Mockups
1.1	13/01/2017	IH	Descrição dos requisitos obtidos
1.0	11/01/2017	IH	O documento corrigido e aceite
0.1	02/01/2017	IH	Construção dos diagramas de responsabilidades
0.1	30/12/2016	IH	Aplicação da técnica
0.1	30/12/2016	IH	Introdução
0.1	29/12/2016	IH	Criação do documento e a sua estrutura

Índice

Introdução

Este documento descreve o comportamento do módulo a desenvolver, com o objetivo principal de identificar os principais requisitos para formar uma opinião geral do problema.
O levantamento de requisitos foi elaborado utilizando a técnica "KAOS model". O modelo KAOS é uma técnica baseada em objetivos, dando um objetivo inicial iremos subdividir esse em outros objetivos mais pequenos, que são necessários para alcançar o principal, assim, depois de várias iterações chegamos a objetivos que não se dividem, ou seja as folhas da árvore. Estas folhas podem ser de dois tipos, o primeiro são responsabilidades atribuídas a um agente especifico, e o segundo são requisitos de um agente especifico. Para a elaboração deste modelo foi utilizado o programa "Obcjetiver 3.0". Este programa facilita a elaboração do modelo, construindo a árvore, podemos construir os modelos de responsabilidades para cada agente automaticamente. Estes modelos de responsabilidade representam os requisitos ou responsabilidades para cada um dos agentes.
Depois de levantamento inicial dos requisitos foi realizada uma reunião com o cliente, para verificar se existiam requisitos importantes em falta.

Técnica escolhida

A técnica escolhida, como já referido anteriormente, é o "KAOS model" para identificação dos requisitos, de uma forma a manter imagem geral, bem construída e não faltar requisitos importantes, falta dos quais poderia implicar mal entendimento do sistema. Mas antes de mais vamos introduzir a técnica e as suas caraterísticas.

A técnica KAOS foi originada pela cooperação das duas universidades, a universidade de Oregon e a universidade de Louvain em 1990. Alguns dos investigadores da universidade de Louvain originaram uma empresa "Respect-IT" que pôs em pratica dada técnica em dezenas das industrias dos diferentes sectores. O "Respect-IT" é a distribuidora da aplicação "Objectiver" que apoia a técnica em questão, essa aplicação foi usada para aplicar a técnica na pratica.

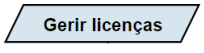
A técnica KAOS é uma técnica que se assenta nos objetivos, pois os objetivos são as metas impostas pelo (s) *stackeholder* (s). A técnica KAOS inicia definindo o objetivo mais geral possível, que se divide por outros objetivos. Os objetivos na parte superior da arvore são os objetivos mais estratégicos e mais importantes, os outros exceto as folhas, resultantes destes são os objetivos que definem como é que podemos alcançar os objetivos principais, por sua vez as folhas são os requisitos do sistema.

Na verdade, as folhas dividem-se em dois tipos, que são os requisitos e as responsabilidades. Com isto podemos construir um modelo do sistema todo não apenas do software, pois é muito importante ter em conta o ambiente que interage com o software. Os requisitos são os objetivos de baixo nível a serem satisfeitos com o software enquanto as responsabilidades são expectativas a serem realizadas por alguém que faz parte do ambiente do sistema. Daqui podemos concluir que os objetivos de mais baixo nível requisitos e responsabilidades já não precisam de ser divididas em sub objetivos, pois conseguimos associar estes a um agente que será responsável por concretização deste objetivo.

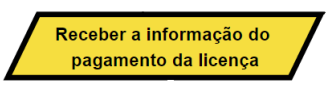
Descrição dos componentes utilizados

 Afinação representada com bola amarela na ligação representa a uma ligação dos objetivos que são necessários para satisfação do objetivo dependente destes.

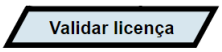
 Responsabilidade representada pela bola cor de rosa ou vermelha na ligação e apresenta uma relação entre agente e exigência, a uma agente atribuída responsabilidade de alcançar a exigência atribuída. As cores apresentam o tipo do agente, quando se trata de um agente interno, humano, é uma bola cor de rosa, enquanto a bola é de cor vermelha, significa que o agente é um sistema ou alguma entidade externa.



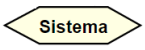
Objetivo é representado pelo um paralelogramo, é a firmação prescritiva capturar algum objetivo a atingir pela cooperação de agentes, prescreve um conjunto de comportamentos desejados.



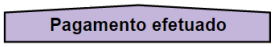
Expectativa representada pelo um paralelogramo de cor amarela, é um objetivo atribuído a um agente do sistema.



Requisito, apresentação semelhante a objetivos, mas com bordas mais grosas, é um objetivo atribuído a um agente de software.



Agente representado pelo um hexágono, é um agente a quem é atribuída os requisitos e expectativas.



Propriedade do domínio, representado por um pentágono de cor roxa, é uma firmação descritiva sobre objetos no ambiente do software. Pode ser um domínio ou uma invariante. Uma invariante de domínio é uma propriedade conhecida para segurar em cada estado de algum objeto de domínio, por exemplo, uma lei física, regulação.


Conflito, representado pelo raio vermelho, é considerado a existência de um conflito nos objetivos se algum destes não pode ser alcançado por completo.

KAOS *model*

Nesta seção iremos abordar os diferentes partes da árvore do modelo de KAOS, para demonstrar a técnica utilizada.

Parte principal

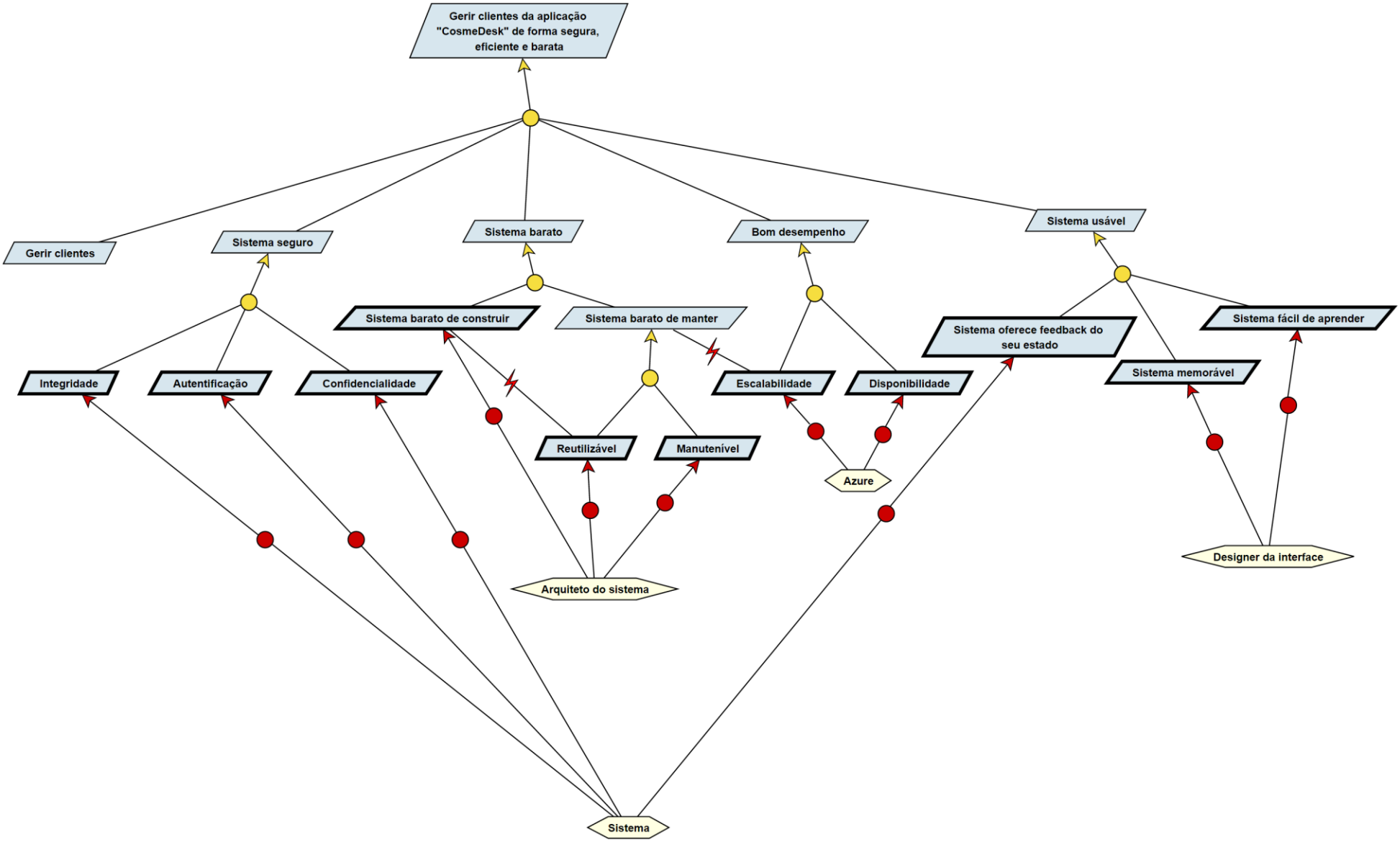


Figura 1 - Parte principal da árvore

Como podemos verificar no diagrama, representado anteriormente, o objetivo principal é "Gerir clientes da aplicação "CosmeDesk" de forma segura, eficiente e barata", este objetivo divide-se em vários ramos. O ramo mais a esquerda corresponde a objetivos que vão fornecer os **requisitos funcionais**, enquanto os restantes correspondem objetivos que nos levam a conhecer os **requisitos não funcionais**. Como podemos verificar existem conflitos entre os objetivos, pois nem sempre conseguimos ter um sistema barato e que fornece bom nível do desempenho, pois o bom desempenho muitas das vezes exige mais gastos.

Gerir clientes

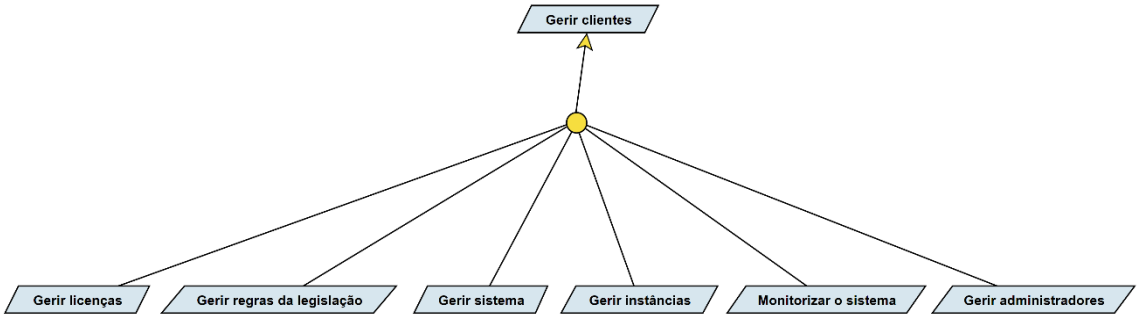


Figura 2 - Gestão dos clientes

No diagrama representado na Figura 2, o objetivo gerir clientes parte-se em mais cinco objetivos que são necessários para uma boa gestão dos clientes.

Gerir licenças

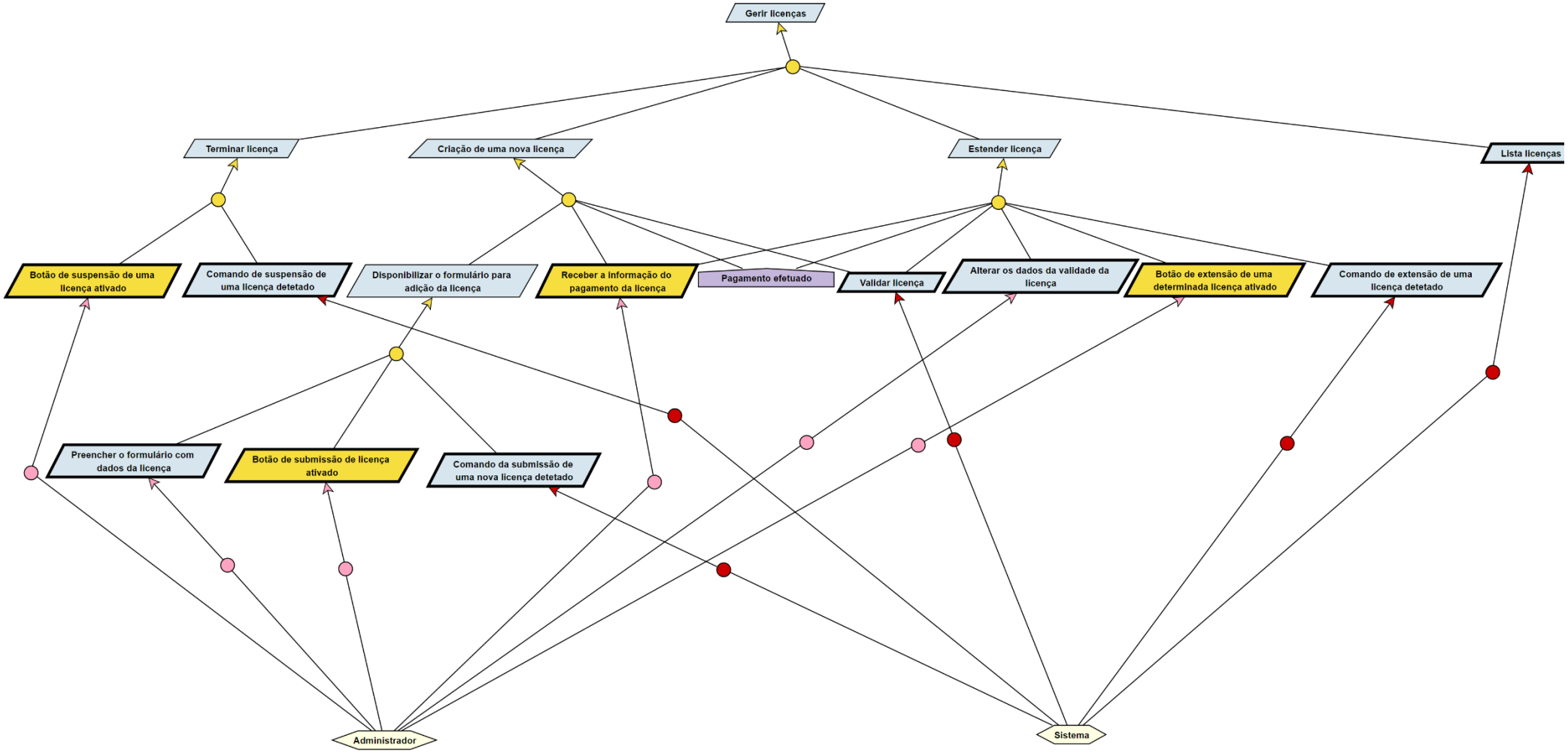


Figura 3 - Gestão das licenças

Na Figura 3 podemos verificar os objetivos da gestão de licenças, em que, objetivo principal divide-se em quatro subobjetivos. Para dois destes, "Criação de uma nova licença" e "Estender licença" é necessária haver pagamento. Também obtivemos novos requisitos para agente "Sistema" e responsabilidades para o agente "Administrador".

Gerir regras de legislação

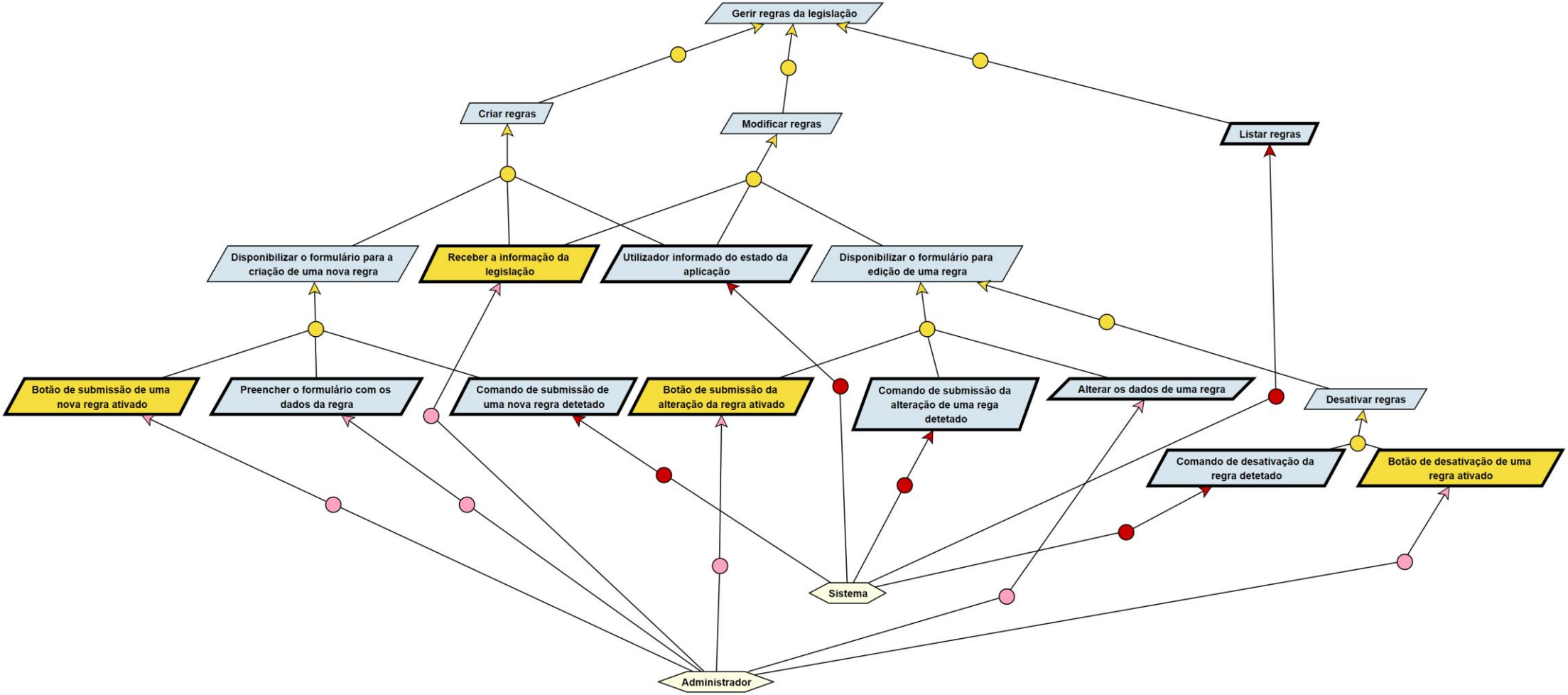


Figura 4 - Gestão de regras de legislação

Na Figura 4 temos, o objetivo de "Gerir regras de legislação", este divide-se em três objetivos que têm de estar presentes de forma a conseguir satisfazer o objetivo principal. Também podemos verificar algumas novas responsabilidades para agente "Administrador" e novos requisitos para agente "Sistema".

Gerir instancias

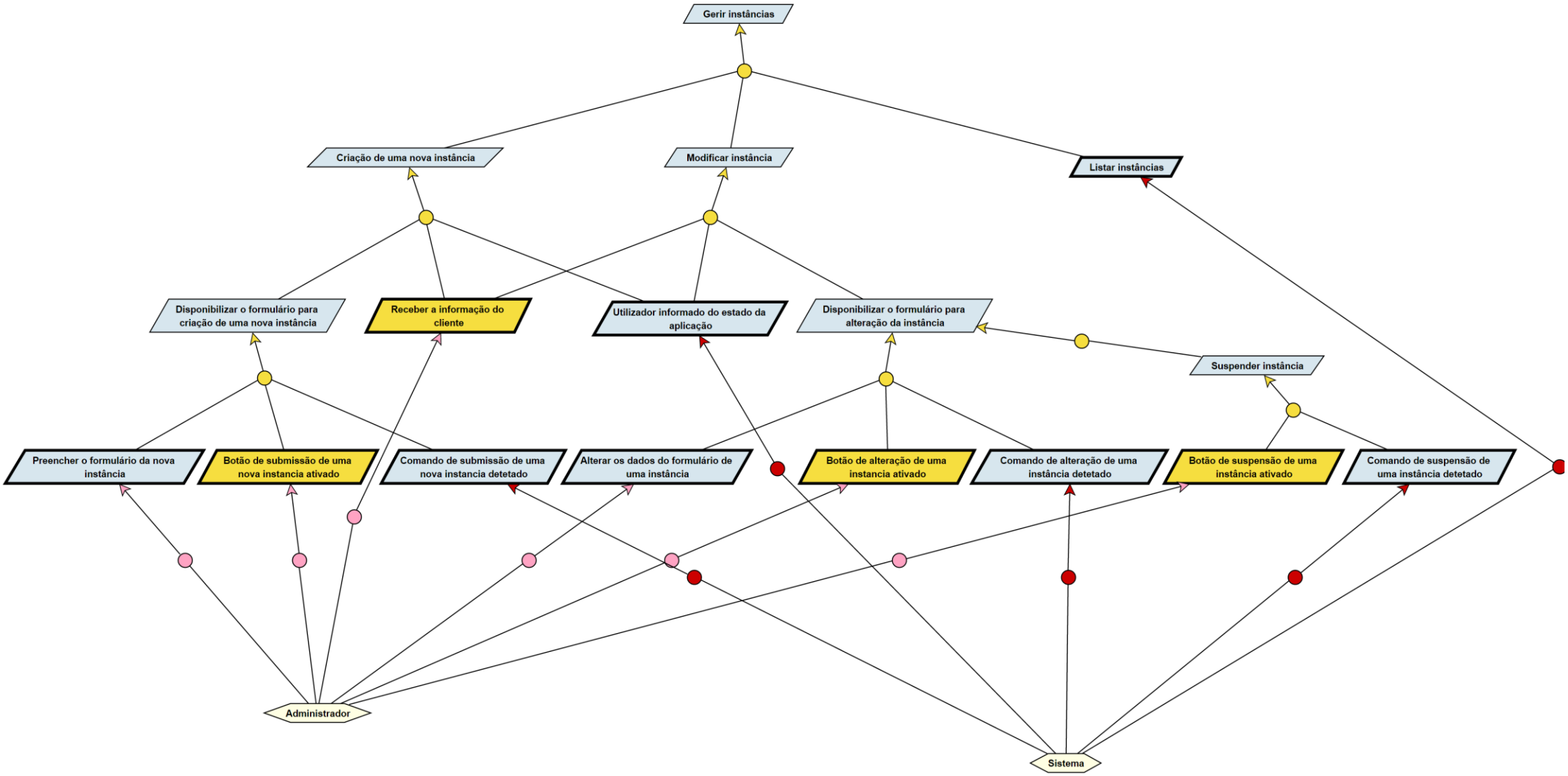


Figura 5 - Gestão das instâncias

Como podemos verificar na Figura 5, existem três objetivos para satisfação do objetivo principal da gestão das instâncias, apesar de que um destes objetivos já é um requisito do sistema.

Gerir sistema

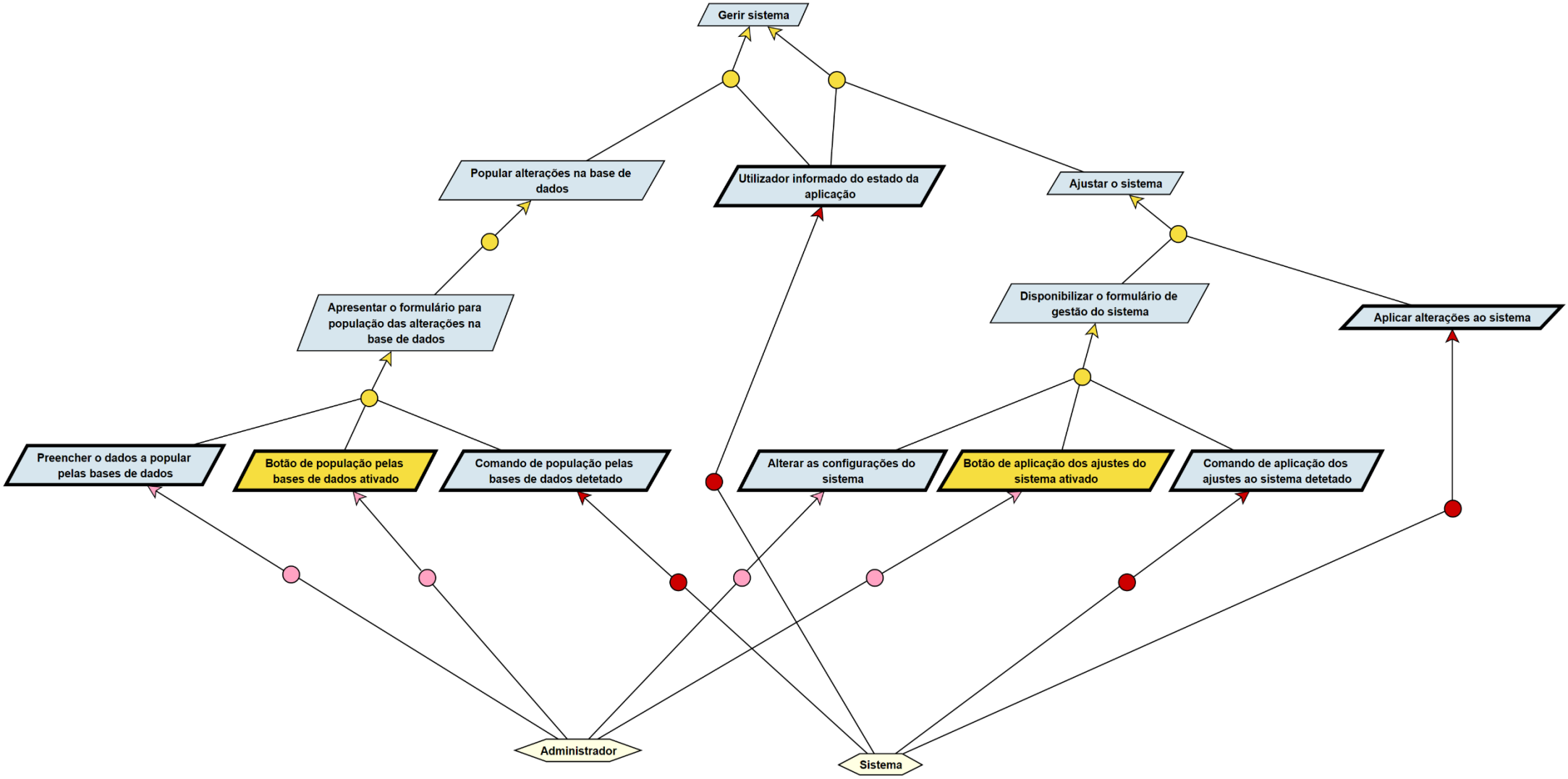


Figura 6 - Gestão do sistema

Como podemos verificar na Figura 7, objetivo principal pode ser satisfeito do conjunto de dois diferentes objetivos. Sendo estes "Popular alterações na base de dados" com o "O utilizador informado do estado da aplicação", ou "O utilizador informado do estado da aplicação" com o "Ajustar o sistema".

Gerir administradores

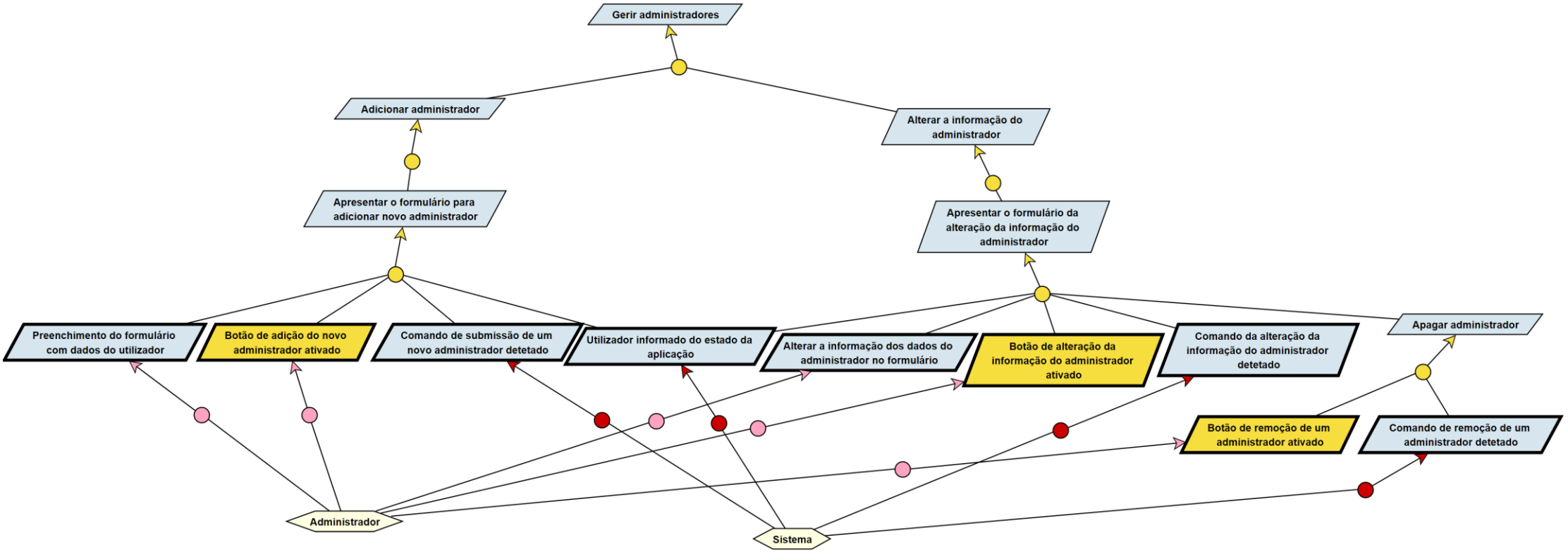


Figura 7 – Gestão dos administradores

Como podemos verificar na Figura 7, o objetivo principal divide-se em dois objetivos, estes objetivos são "Adicionar administrador" e "Alterar a informação do administrador".

Monitorizar sistema

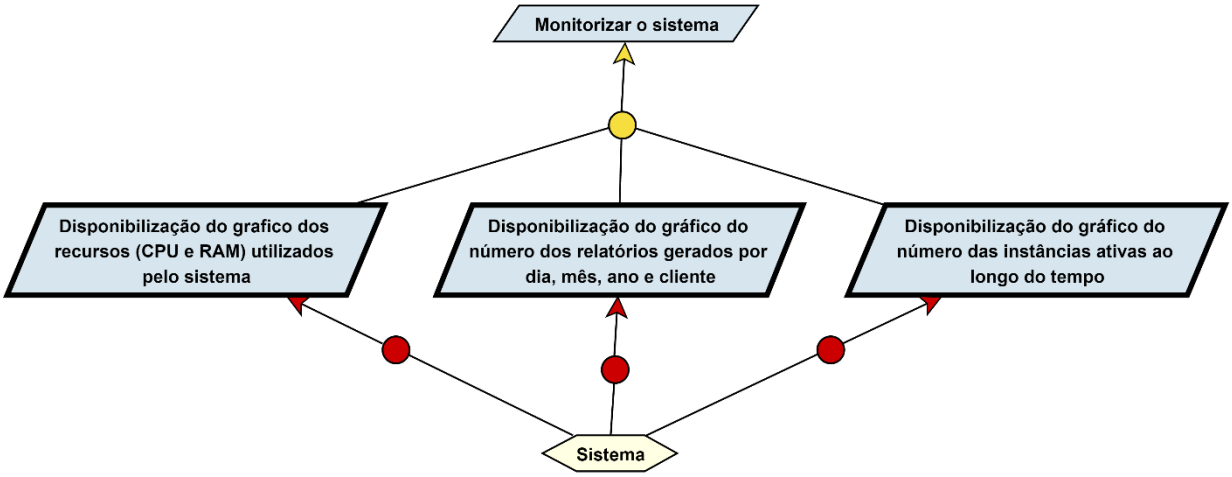


Figura 8 - Monitorizar o sistema

Como podemos verificar pela Figura 8, o sistema irá conter mais três requisitos a satisfazer.

Resultados da aplicação da técnica "KAOS"

Nesta seção iremos comentar os resultados obtidos através da aplicação da técnica "KAOS", estes resultados traduzem-se em requisitos ou responsabilidades identificadas para cada ator presente na árvore. Depois das diagramas representadas nesta secção, são apresentadas tabelas com a descrição dos requisitos, de uma forma a disponibilizar mais informação sobre cada um dos requisitos.

Responsabilidades do ator "Administrador"

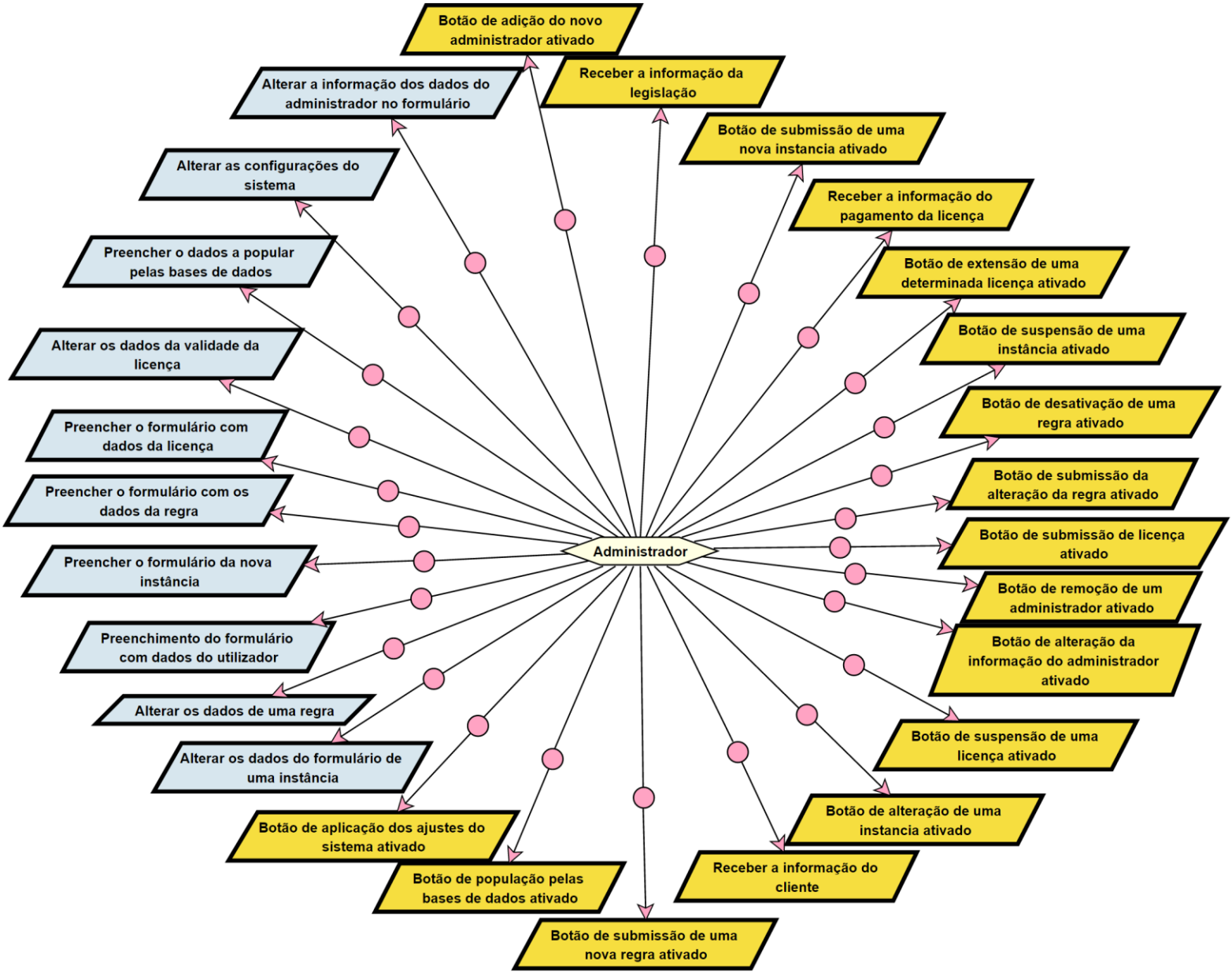


Figura 9 - Responsabilidades do Administrador

Como podemos verificar, no diagrama da Figura 9, as responsabilidades do "Administrador" resumem-se em preenchimento e submissão dos formulários, a não ser que precisa de receber informação do exterior do programa. A informação do exterior corresponde a dados de novos clientes, também é importante ele verificar a legislação, pois através desta, são

construídas regras para notificações, outro dado importante que o administrador tem que receber são os pagamentos das licenças antes de as criar.

Requisitos do ator "Sistema"

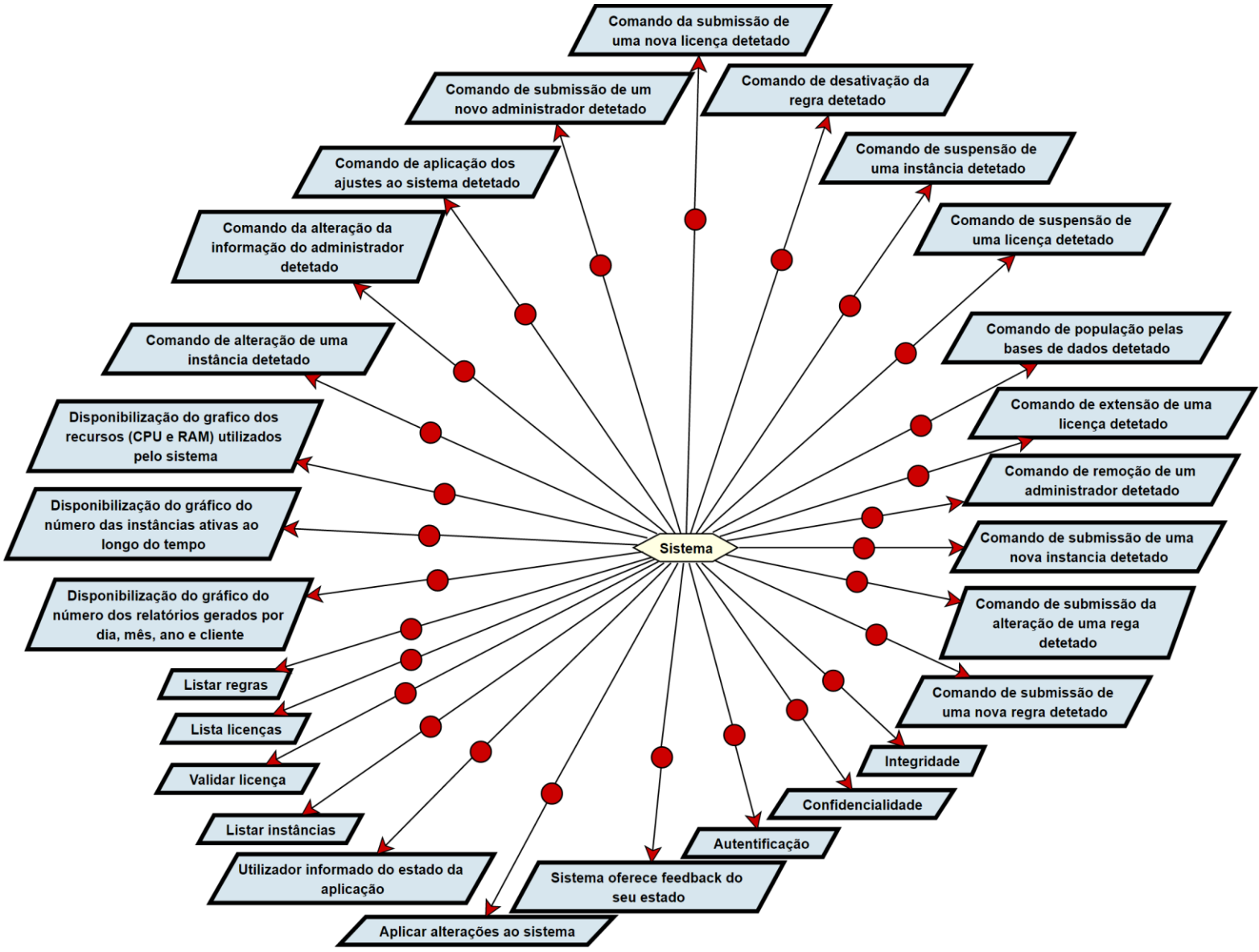


Figura 10 - Requisitos Sistema

Como podemos verificar na Figura 10, para o ator "Sistema" foram identificados muitos requisitos, existindo assim os quatro não funcionais que são: "Integridade", "Autentificação", "Confidencialidade" e "Sistema oferece feedback do seu estado", os restantes requisitos são funcionais. A maioria dos requisitos funcionais correspondem a operações CRUD (*Create, Read, Update, Delete*), ou seja, o sistema tem que conseguir detetar os comandos de criação, alteração ou de eliminação das instâncias, regras e licenças, como também listar estas todas. Existem também os requisitos que não fazem parte de operações CRUD, estes são requisitos mais específicos que têm o objetivo de fornecer ao administrador os dados ou permitir a configuração do sistema.

Requisitos do ator “Designer da interface”

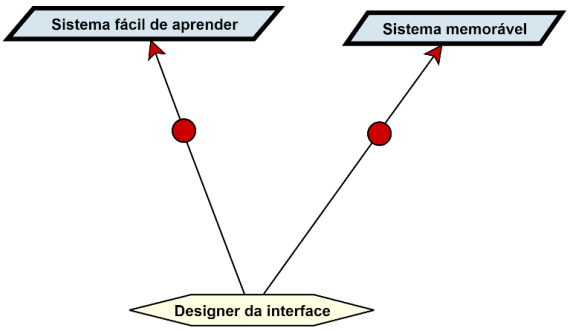


Figura 11 - Requisitos "Designer da interface"

Como podemos verificar na Figura 11, o *designer* do sistema é responsável por construir uma *interface* que seja fácil de aprender e memorizar, isso ajuda o administrador melhorar a sua experiência com o módulo.

Requisitos do ator “Arquiteto do sistema”

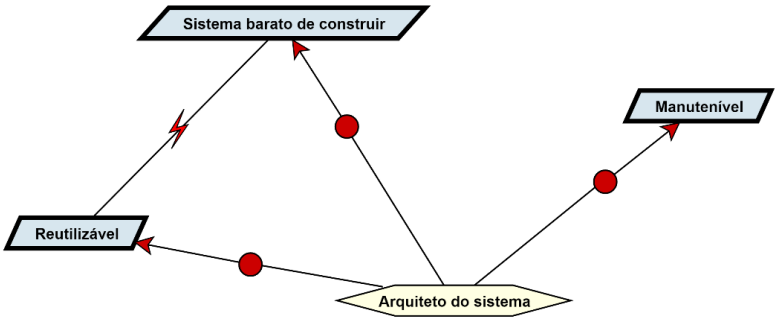


Figura 12 - Requisitos "Arquiteto do sistema"

Como podemos verificar na Figura 12, o arquiteto do sistema tem que conseguir construir uma arquitetura de baixo custo, de fácil manutenção e de reutilização posterior. Também é fácil de notar que o arquiteto do sistema terá de lidar com o problema entre o reutilização e sistema barato, pois estes dois entram em conflito, não é fácil de ter um sistema reutilizável e barato ao mesmo tempo, o que implica de tentar encontrar o melhor equilíbrio entre estes requisitos.

Requisitos do ator “Azure”

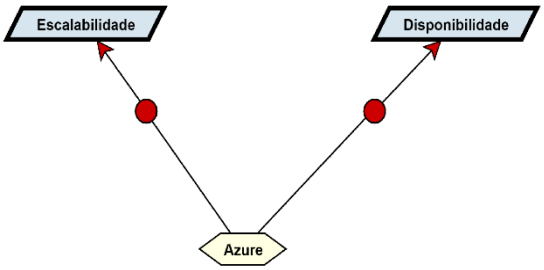


Figura 13 - Requisitos "Azure"

Como podemos verificar na Figura 13, o *Azure* irá nos fornecer a disponibilidade e escalabilidade, pois são estas caraterísticas principais da *cloud*.

Descrição dos requisitos obtidos

Nesta secção descreve-se os requisitos e as responsabilidades obtidas, para fornecer mais detalhe da importância e do modo de funcionamento do módulo.

Atores participantes

- Administrador: é um ator que terá as responsabilidades na utilização do sistema, tais como o preenchimento de formulários e submissão destes.
- Azure: é o ator que é responsável pela disponibilidade e escalabilidade do sistema, pois são as caraterísticas dos serviços da *cloud*.
- Arquiteto do sistema: é o ator responsável pela a criação da arquitetura do sistema e a sua interligação.
- Designer da interface: é o ator que é responsável pela a criação da interface usável no módulo.
- Sistema: é o ator mais importante, pois são as funcionalidades que o sistema oferece.

Requisitos não funcionais

Código	Requisito	Descrição	Prioridade
RNF1	Manutenível	O sistema de fácil manutenção.	Obrigatório
RNF2	Sistema barato de construir	O sistema não implicar grandes gastos dos recursos monetários.	Obrigatório
RNF3	Reutilizável	O sistema tem que ser reutilizável, ser possível reutilizar partes na reconstrução/manutenção do sistema.	Obrigatório
RNF4	Escalabilidade	O sistema tem de ser construído de forma a que seja escalável consoante as necessidades dos utilizadores.	Obrigatório
RNF5	Disponibilidade	O sistema tem de oferecer a disponibilidade que satisfaça as necessidades dos utilizadores.	Obrigatório
RNF6	Sistema fácil de aprender	O sistema tem de oferecer uma interface que seja simples de utilizar e de aprender.	Obrigatório
RNF7	Sistema memorável	O sistema tem de oferecer uma interface coerente, que ajuda a memorizar as funcionalidades.	Obrigatório
RNF8	Integridade	O sistema tem que garantir a integridade dos dados armazenados.	Obrigatório
RNF9	Confidencialidade	O sistema tem que oferecer a confidencialidade dos dados suficiente, para que estes não sejam observáveis por outros utilizadores ou terceiros.	Obrigatório
RNF10	Autentificação	Para poder utilizar o sistema, tem que haver uma autentificação que permite a identificar o utilizador.	Obrigatório
RNF11	Sistema oferece feedback do seu estado	O sistema tem que oferecer o feedback, mantendo o utilizador informado do seu estado e como correram as últimas operações.	Obrigatório

Requisitos funcionais

Código	Requisito	Descrição	Prioridade
RF1	Aplicar alterações ao sistema	O sistema tem que detetar o camando das alterações realizadas pelo utilizador e efetuar as alterações respetivas na plataforma da "Azure".	*Nice to have*
RF2	Utilizador informado do estado da aplicação	O sistema tem que informar sempre o utilizador do seu estado e das operações efetuadas pelo utilizador.	Obrigatório
RF3	Listar instâncias	O sistema tem que listar todas as instâncias existentes.	Obrigatório

RF4	Validar licenças	O sistema tem que realizar toda a verificação necessária para validar uma renovação de licença ou uma nova licença.	Obrigatório
RF5	Listar licenças	O sistema tem que listar todas as licenças existentes podendo estes serem filtrados de diferentes formas.	Obrigatório
RF6	Listar regras	O sistema tem que listar todas as regras existentes, podendo estas serem filtradas de diferentes formas.	Obrigatório
RF7	Comando da alteração de uma instância detetado	O sistema tem que detetar o comando de alteração de uma instância e armazenar estas alterações no sistema, podendo englobar também as alterações ao meio do trabalho do cliente.	Obrigatório
RF8	Comando da alteração da informação do administrador detetado	O sistema tem que detetar o comando da alteração do administrador e armazenar as alterações efetuadas pelo utilizador.	Obrigatório
RF9	Comando de aplicação dos ajustes ao sistema detetado	O sistema tem que detetar o comando de aplicação dos ajustes ao sistema, para poder aplicar as alterações na plataforma da "Azure".	*Nice to have*
RF10	Comando de submissão de um novo administrador detetado	O sistema tem de detetar e adicionar um novo administrador ao módulo.	Obrigatório
RF11	Comando da submissão de uma nova licença detetado	O sistema tem de detetar a submissão de uma nova licença efetuando e armazenando as alterações no sistema para um determinado cliente.	Obrigatório
RF12	Comando de desativação da regra detetado	O sistema tem de detetar a desativação de uma licença, efetuando e armazenando as alterações no sistema para um determinado cliente.	Obrigatório
RF13	Comando de suspensão de uma instância detetado	O sistema tem de detetar a alteração de uma instância, com base nos dados introduzidos pelo utilizador, Refetuar todas as alterações do ambiente de trabalho do cliente.	Obrigatório
RF14	Comando de suspensão de uma licença detetado	O sistema tem que conseguir detetar o comando de suspensão de uma licença e efetuar as alterações necessárias.	Obrigatório
RF15	Comando de população pelas bases de dados detetado	O sistema tem que conseguir detetar o comando de população pelas bases de dados e efetuar as alterações em todas as bases de dados existentes.	Obrigatório
RF16	Comando de extensão de uma licença detetado	O sistema tem que detetar o comando de extensão de uma licença e efetuar as alterações necessárias.	Obrigatório

RF17	Comando de remoção de um administrador detetado	O sistema tem que conseguir detetar o comando de remoção de um administrador do sistema e realizar as alterações necessárias.	Obrigatório
RF18	Comando de submissão de uma nova instância detetado	O sistema tem de detetar o comando da submissão de uma nova instância, com base nos dados do novo cliente, introduzidos pelo utilizador, efetuar toda a configuração automática para disponibilizar o ambiente do trabalho para o novo cliente.	Obrigatório
RF19	Comando de submissão da alteração de uma regra detetado	O sistema tem que detetar o comando de submissão da alteração de uma regra, efetuar a alteração necessária, e verificar todos os relatórios dos clientes para a validação destes, na existência de alguma regra que não seja satisfeita, será gerada uma notificação para os clientes com os relatórios que não satisfazem a legislação.	Obrigatório
RF20	Comando de submissão de uma nova regra detetado	O sistema tem que detetar o comando de submissão de uma nova regra, efetuar a alteração necessária e verificar todos os relatórios dos clientes, para a validação destes, na existência de alguma regra que não seja satisfeita, será gerada uma notificação para os clientes com os relatórios que não satisfazem a legislação.	Obrigatório
RF21	Disponibilização do gráfico do número dos relatórios gerados por dia, mês, ano e o cliente	O sistema tem que disponibilizar o *dashboard* ao utilizador com os dados importantes para seu negócio, como o número dos relatórios gerados por um certo período do tempo, podendo ser agrupado por um cliente especifico ou por grupo destes.	*Nice to have*
RF22	Disponibilização do gráfico do número das instâncias ativas ao longo do tempo	O sistema tem que disponibilizar o *dashboard* ao utilizador com os dados importantes de performance do sistema, sendo possível observar no *dashboard* o número das instâncias ativas, podendo agrupar por um certo período do tempo.	*Nice to have*
RF23	Disponibilização do gráfico dos recursos (CPU e RAM) utilizados pelo sistema	O sistema tem que disponibilizar o *dashboard* ao utilizador com os dados importantes dos recursos utilizados, disponibilizando um *dashboard* com os gastos de CPU e RAM, podendo estes serem agrupados por certos períodos de tempo.	*Nice to have*

Responsabilidades

Código	Responsabilidade	Descrição
R1	Alterar a informação dos dados do administrador no formulário	O utilizador tem a responsabilidade de alterar os dados do administrador e que estes estejam corretos.
R2	Alterar as configurações do sistema	O utilizador poderá alterar as configurações do sistema, assumindo a sua responsabilidade pelas alterações que aplica.
R3	Preencher os dados a popular pelas bases de dados	O utilizador poderá realizar populações nas bases de dados dos clientes, adicionar novos componentes, matérias primas, tipos dos produtos, entre outros, com a responsabilidade de colocar os dados corretos.
R4	Alterar os dados da validade da licença	O utilizador pode estender a validade das licenças, ficando responsável por essa alteração.
R5	Preencher o formulário com dados da licença	O utilizador pode criar novas licenças preenchendo o formulário de uma nova licença, tendo a responsabilidade de verificar o pagamento da mesma.
R6	Preencher o formulário com os dados da regra	O utilizador pode criar as regras com a responsabilidade de verificar a legislação antes.
R7	Preencher o formulário de nova instância	O utilizador pode criar nova instância preenchendo o formulário com os dados do novo cliente.
R8	Preenchimento do formulário com dados do utilizador	O utilizador pode criar um novo administrador, preenchendo o formulário do novo administrador.
R9	Alterar os dados de uma regra	O utilizador terá a possibilidade de alterar as regras existentes no sistema, verificando os dados da legislação antes.
R10	Alterar os dados do formulário de uma instância	O utilizador terá a possibilidade de alterar os dados dos clientes, alterando estes no formulário que lhe será apresentado, sempre com a responsabilidade de colocar os dados corretos.
R11	Botão de adição do novo administrador ativado	O utilizador para finalizar a criação de um novo utilizador, tem que clicar no botão de adição deste ao módulo.
R12	Receber a informação da legislação	O utilizador tem que confirmar os dados da legislação para as operações relacionadas com as regras, pois estas necessitam de estar sempre atualizadas
R13	Botão de submissão de uma nova instância ativado	O utilizador terá sempre de clicar no botão de submissão de um novo cliente no sistema, para a criação de uma nova instância no sistema.

R14	Receber a informação do pagamento da licença	O utilizador terá de confirmar sempre se o pagamento das licenças foi efetuado.
R15	Botão de extensão de uma determinada licença ativado	O utilizador terá de clicar no botão de estender licença, para que esta alteração seja aplicada.
R16	Botão de suspensão de uma instância ativado	O utilizador terá a possibilidade de suspender um cliente inativo, para poupar recursos, sempre com a responsabilidade das suas ações.
R17	Botão de desativação de uma regra ativado	O utilizador pode desativar as regras, no caso de alguma não ser necessária para o sistema.
R18	Botão de submissão da alteração da regra ativado	O utilizador para finalizar a operação da alteração de uma regra terá, sempre que clicar no respetivo botão, para que as alterações sejam feitas no sistema.
R19	Botão de submissão da alteração de licença ativado	Para finalizar a operação da alteração de uma licença, o utilizador tem que clicar no botão respetivo.
R20	Botão de remoção de um administrador ativado	Para remover um utilizador do módulo, o utilizador tem que clicar no botão que diz respeito a remoção deste utilizador do módulo.
R21	Botão de alteração da informação do administrador ativado	Para finalizar a operação da alteração da informação do administrador, o utilizador tem que clicar no respetivo botão.
R22	Botão de suspensão de uma licença ativado	Para a suspensão de uma determinada licença, o utilizador terá de clicar no botão respetivo, com a responsabilidade pelas alterações efetuadas.
R23	Botão de alteração de uma licença ativado	Para finalizar a operação o utilizador tem que clicar sempre no botão respetivo, de forma a que as alterações sejam efetuadas a sistema.
R24	Receber a informação do cliente	O utilizador terá a responsabilidade de receber os dados corretos do cliente, para que estes sejam introduzidos no sistema.
R25	Botão de submissão de uma nova regra ativado	O utilizador poderá adicionar novas regras, preenchendo o formulário e clicando no botão de adição de uma nova regra ao sistema.
R26	Botão de população pelas bases de dados ativado	Para efetuar a população pelas bases de dados dos clientes, o utilizador tem que clicar no respetivo botão.
R27	Botão de aplicação dos ajustes do sistema ativado	Depois de realizar ajustes ao sistema, o utilizador deverá clicar no respetivo botão, para aplicar as alterações efetuadas.

Mockups

Nesta secção serão listados todos os *mockups* de baixa fidelidade que foram criados para a aplicação com base nos requisitos e os objetivos do estágio, de uma forma simplificada, para demonstrar o seu possível *laylout* e estrutura. Por ventura os *mockups* também poderão servir de levantamento dos requisitos, pois o cliente poderá lembrar de algumas funcionalidades que poderão fazer sentido ser implementadas no determinado contexto.

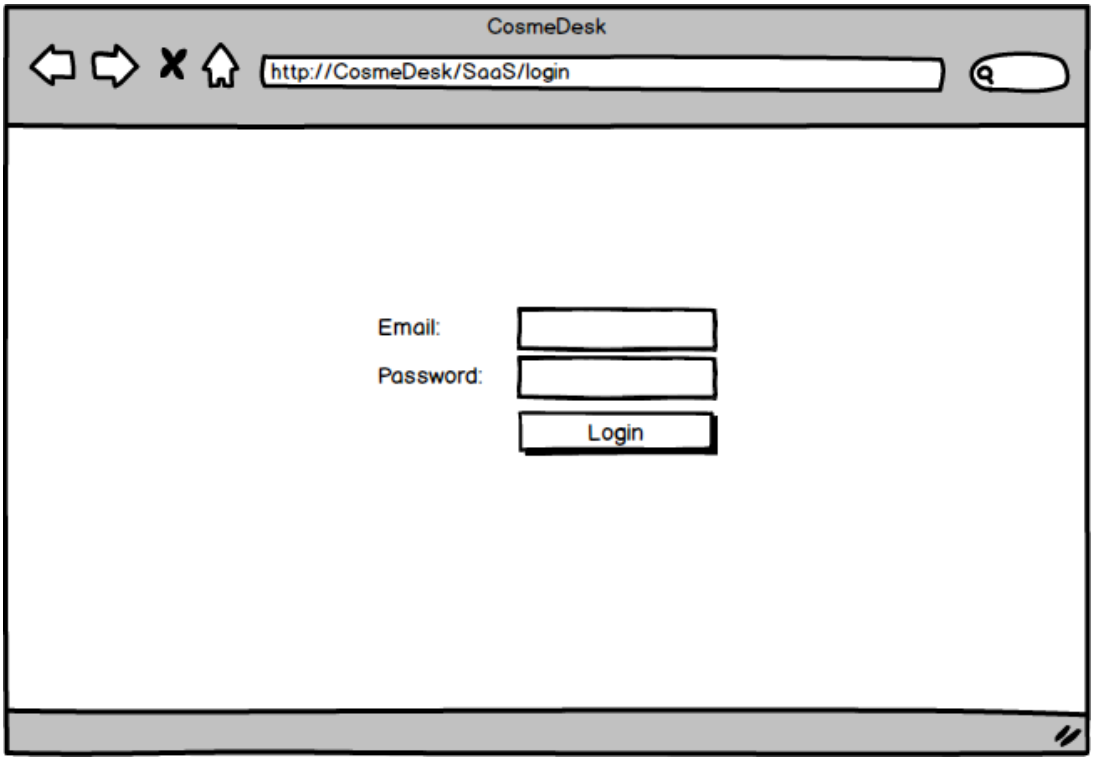


Figura 14: Login

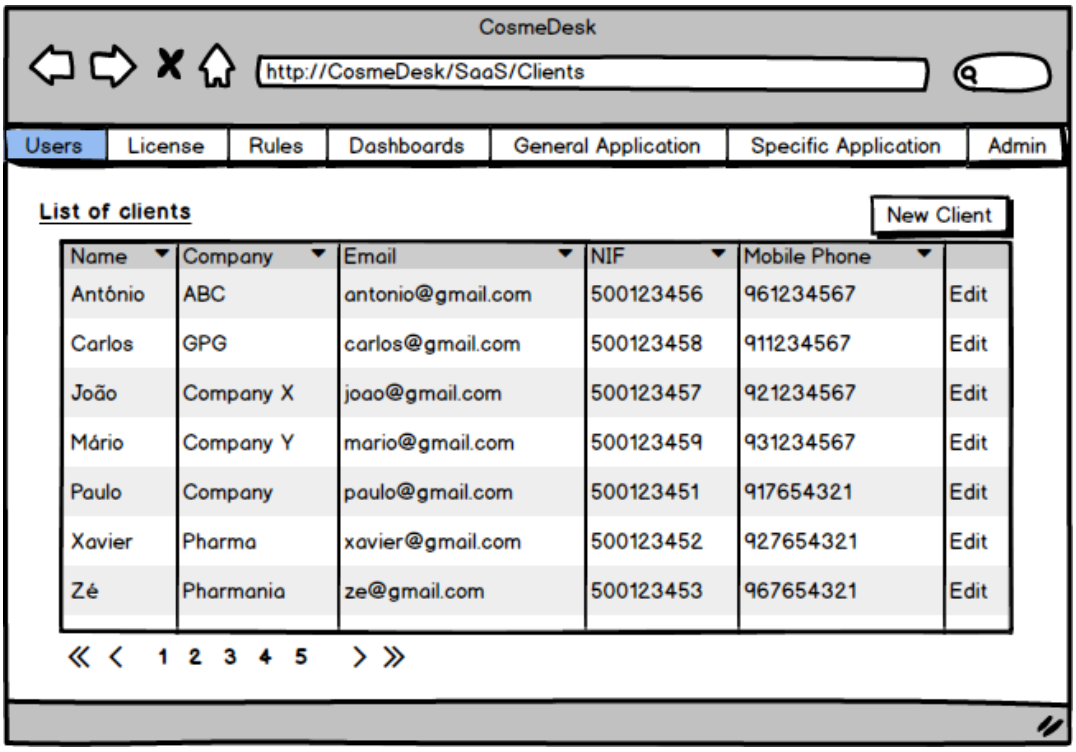


Figura 15: Lista de clientes

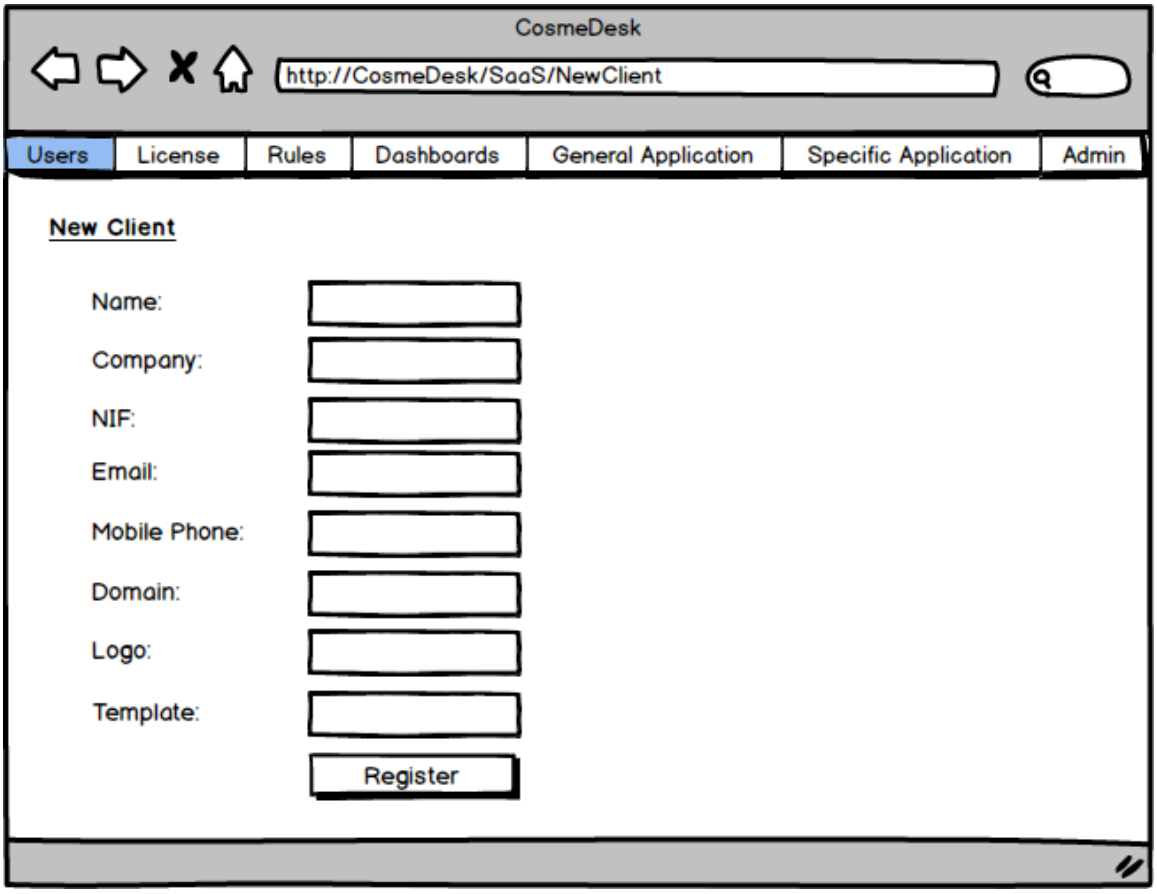


Figura 16: Novo cliente

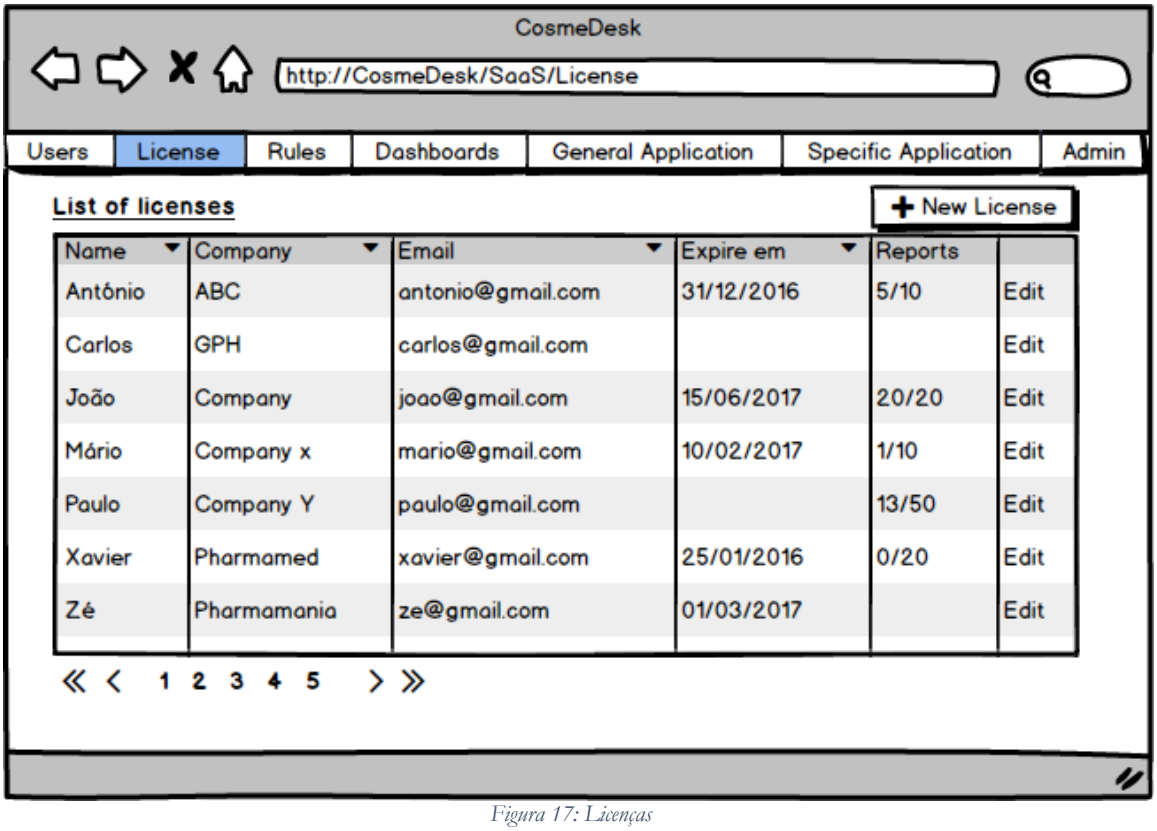


Figura 17: Licenças

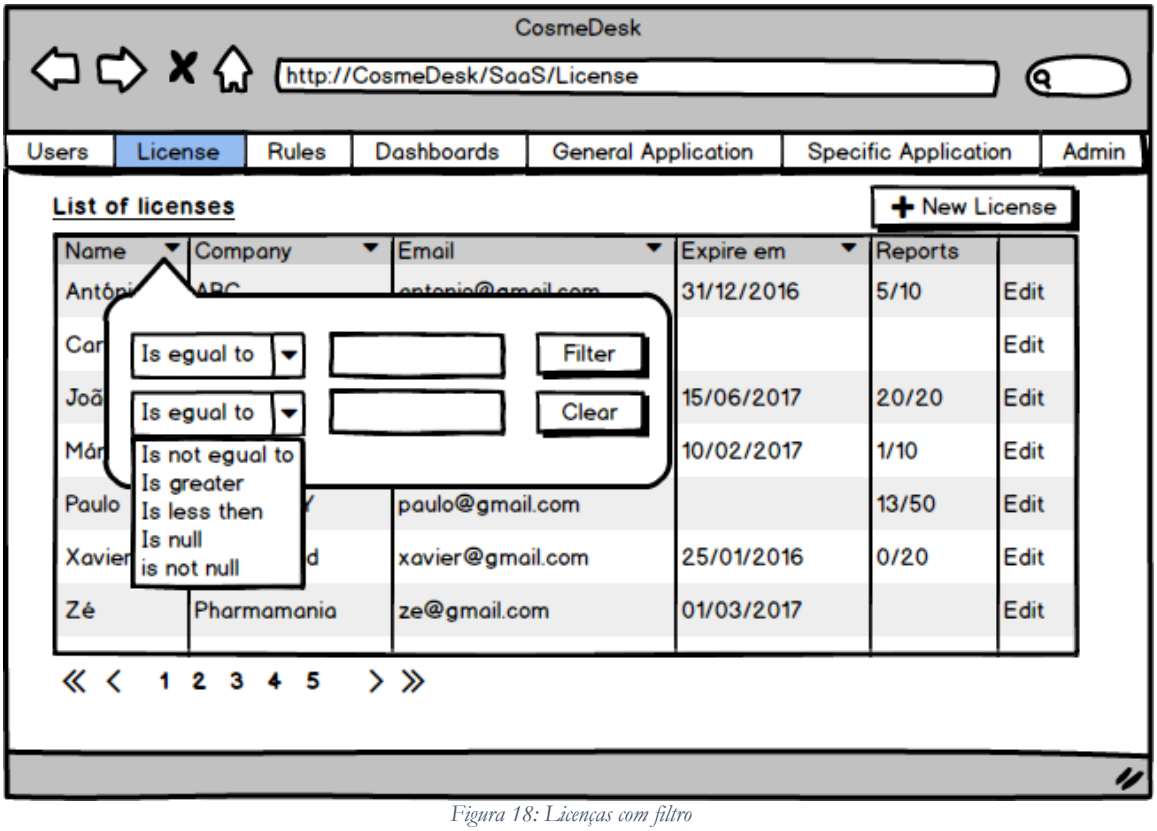


Figura 18: Licenças com filtro

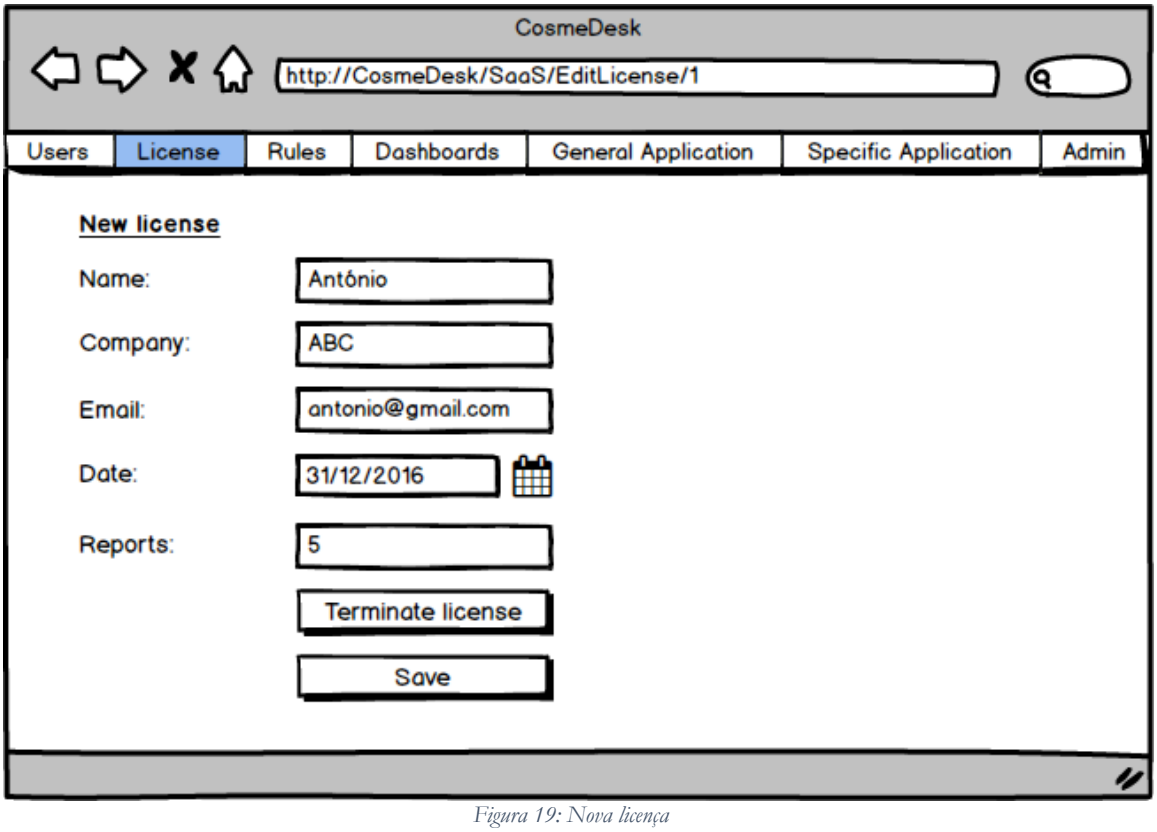


Figura 19: Nova licença

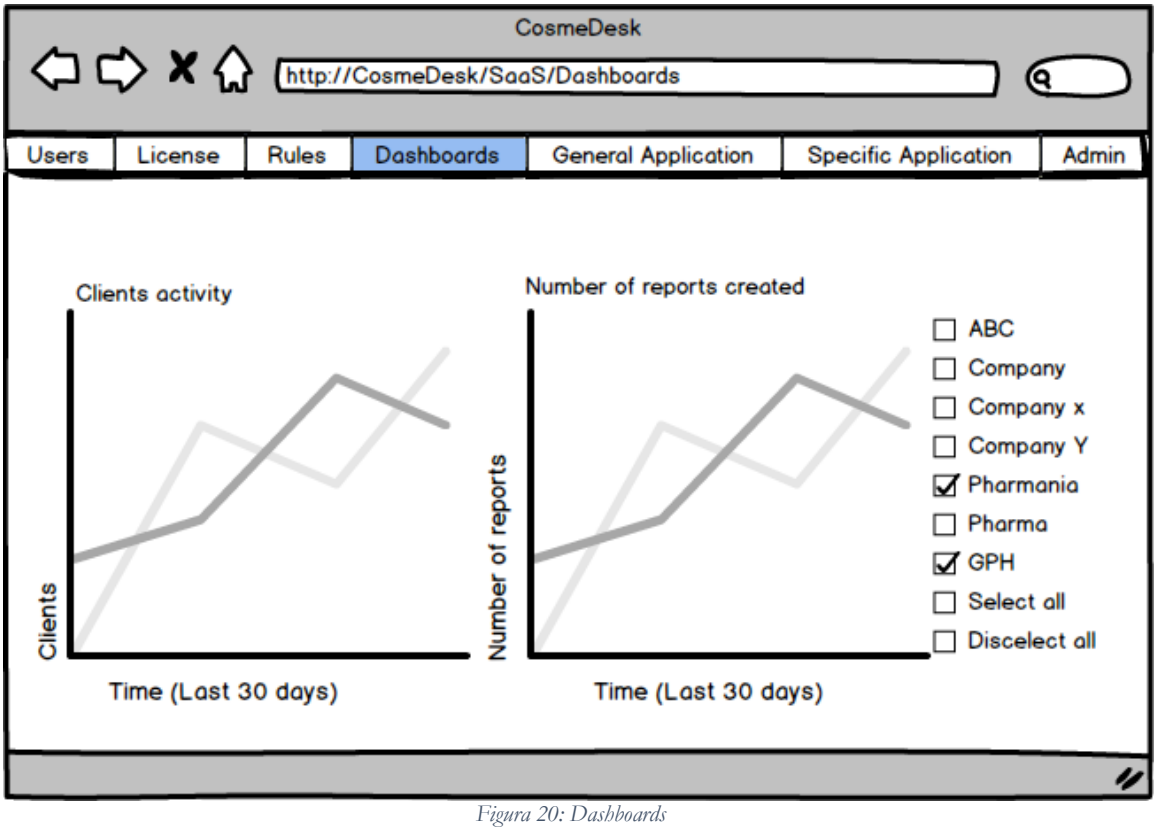


Figura 20: Dashboards

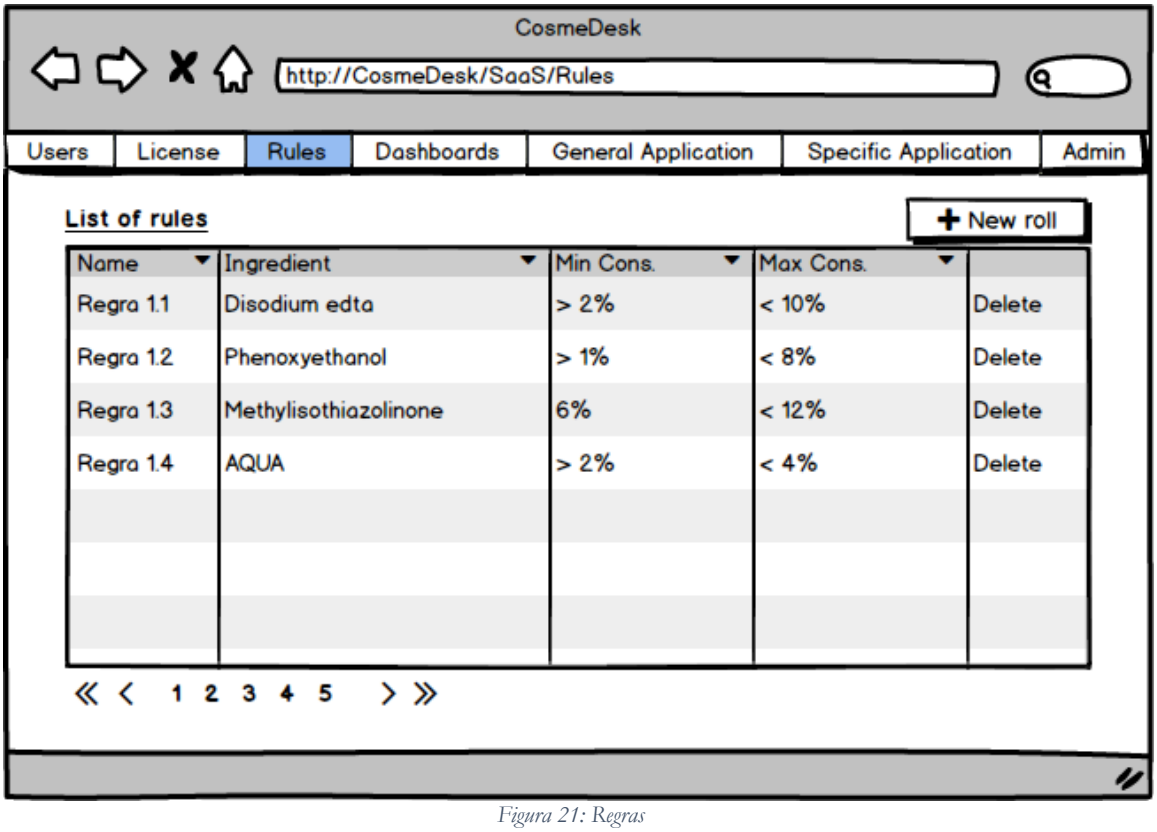


Figura 21: Regras

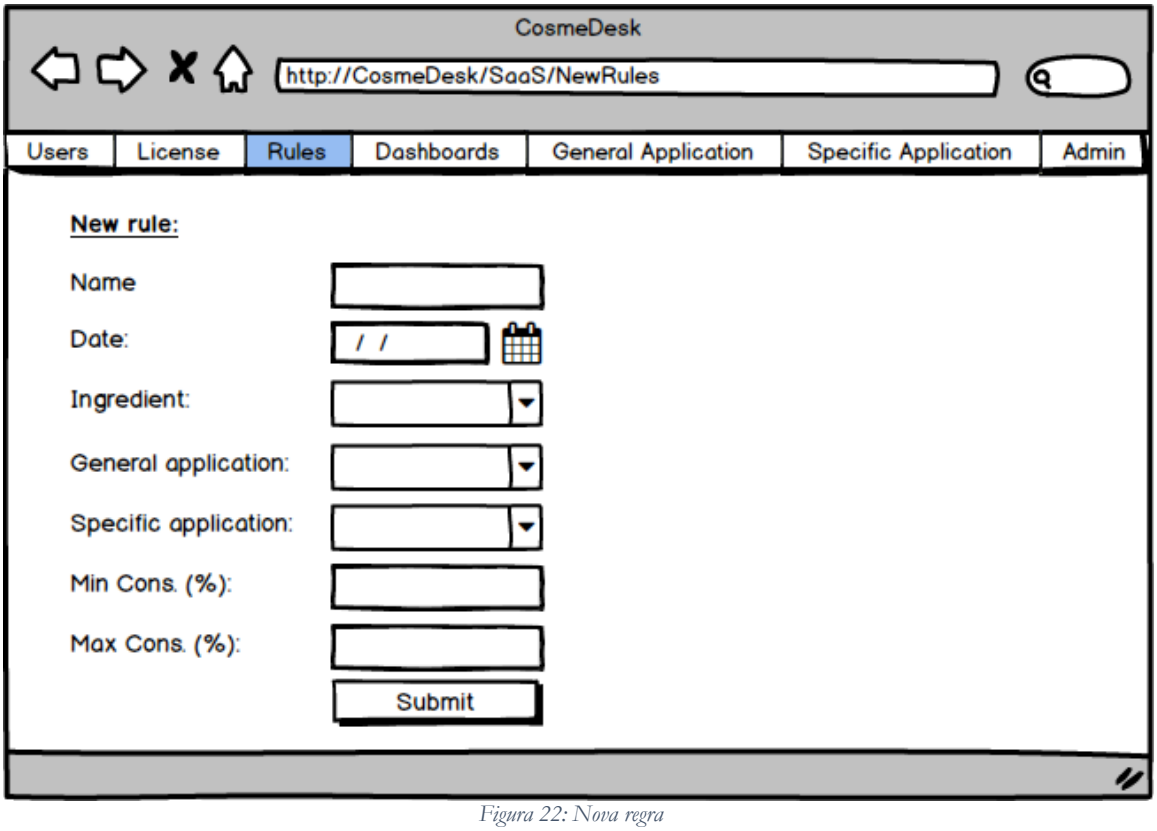


Figura 22: Nova regra

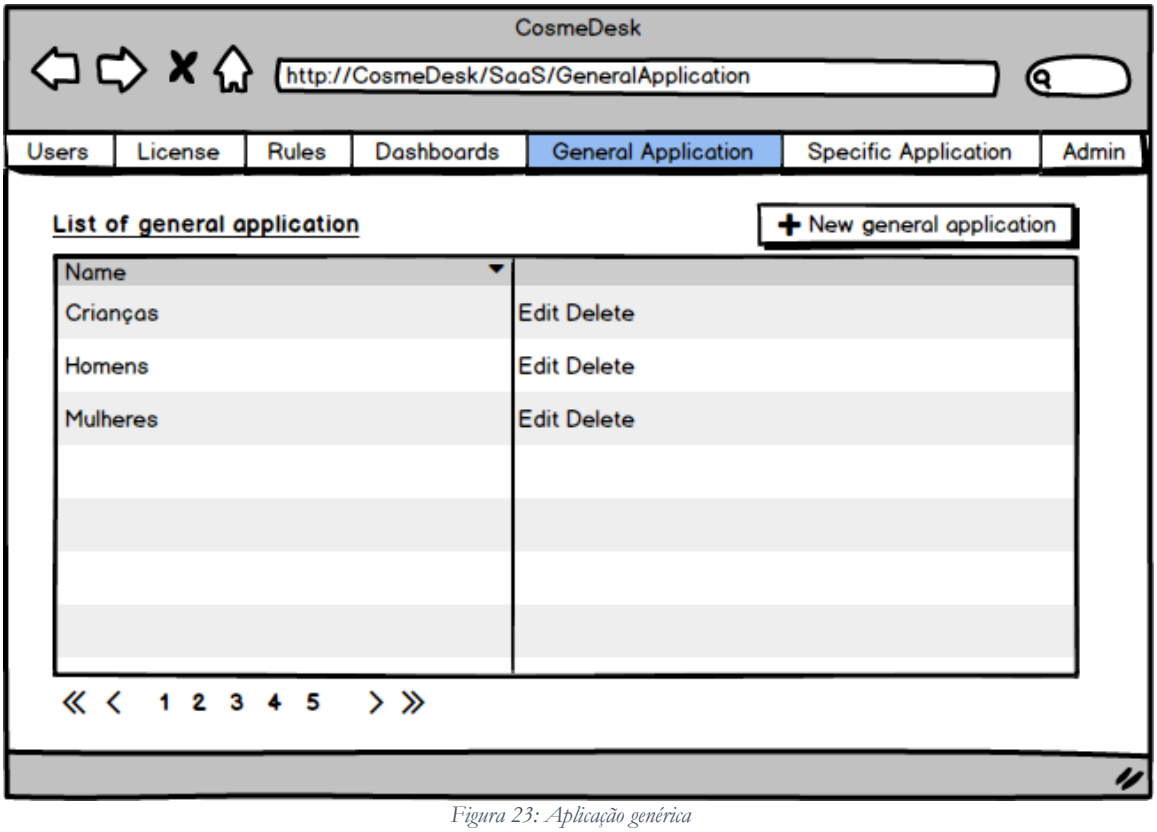


Figura 23: Aplicação genérica

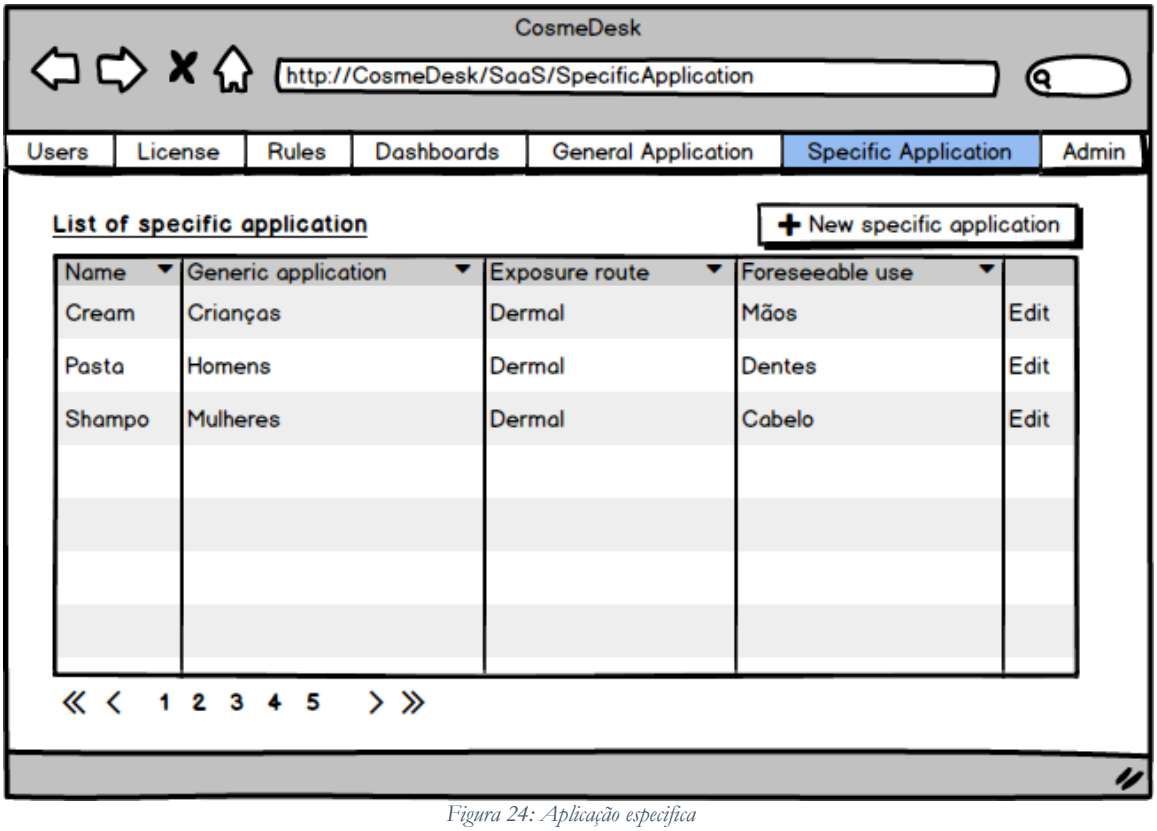


Figura 24: Aplicação especifica

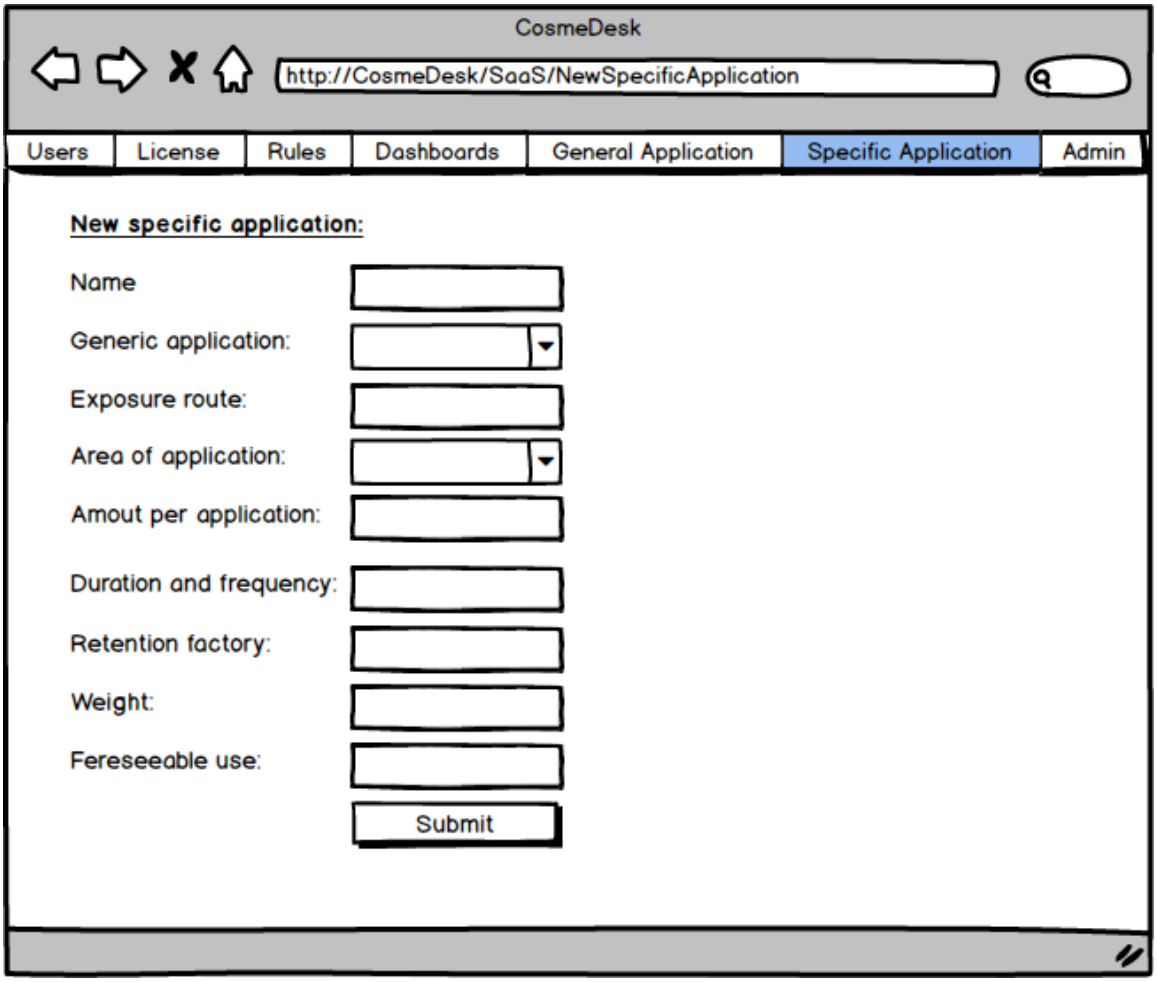


Figura 25: Nova aplicação especifica

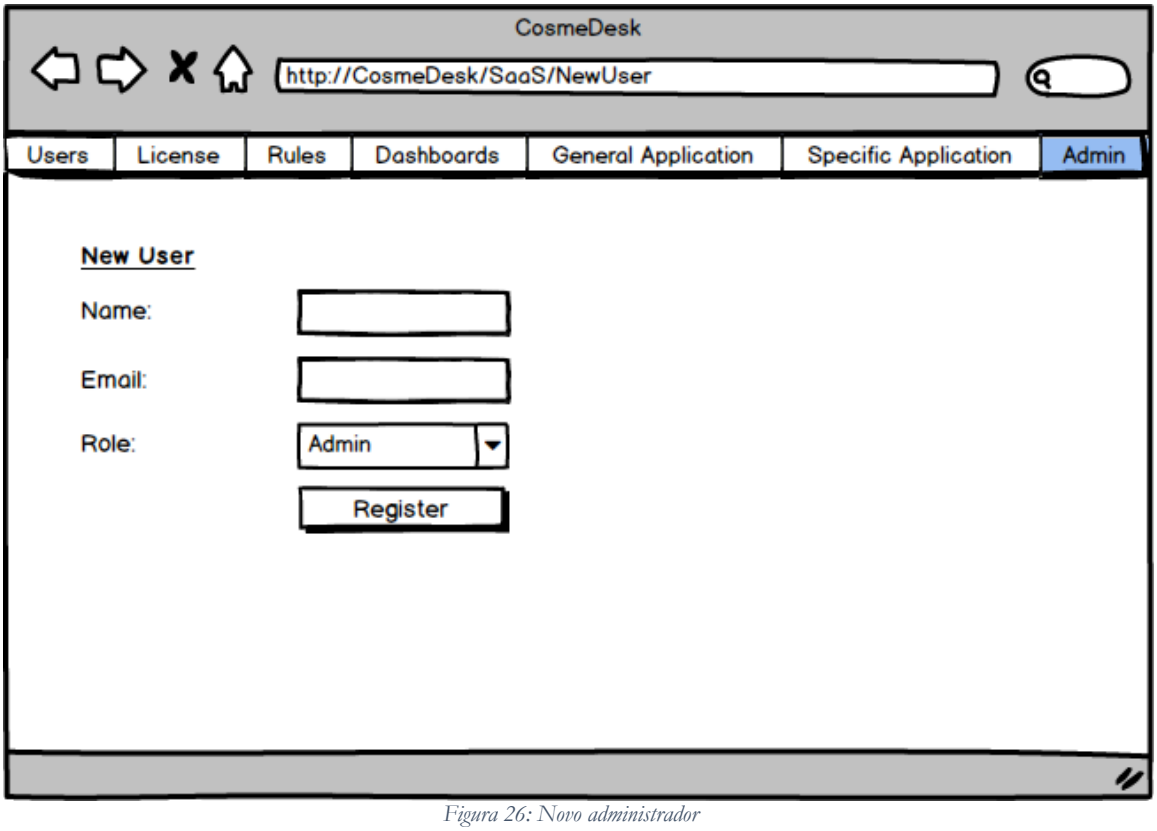


Figura 26: Novo administrador

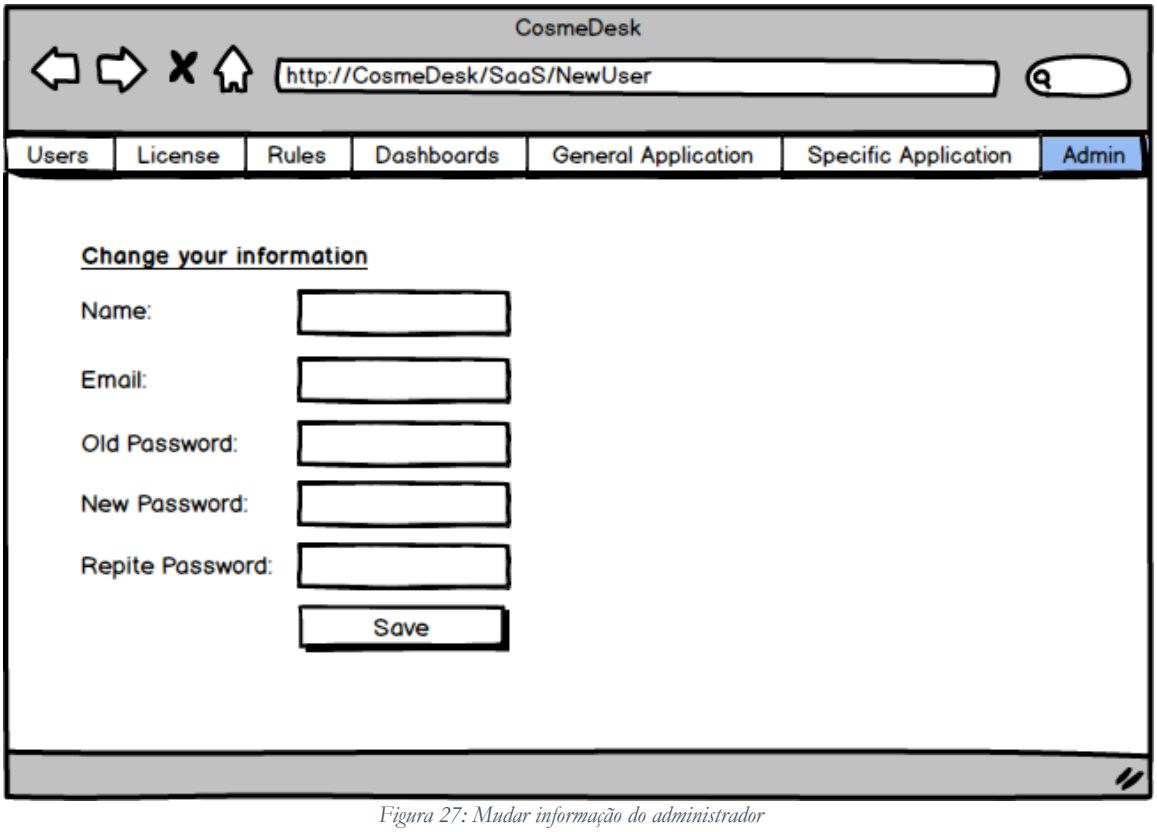


Figura 27: Mudar informação do administrador

Anexo B

Documento de Riscos

Gestão de Riscos

Mestrado em Engenharia Informática
2016/2017

CosmeDesk: Módulo SaaS

Modelos KAOS
(Versão 0.1)
29/12/2016

Estagiário:
Ivan Harasym
2011168230
harasym@student.dei.uc.pt

Orientadores:
Prof. Filipe Araújo
filipius@dei.uc.pt
Departamento de Engenharia Informática, Universidade de Coimbra

Eng. Alcides Manuel de Almeida Marques
alcides.marques@ipn.pt
Instituto Pedro Nunes

Tabela de revisões

Versão	Data	Autores	Descrição
0.1	02/01/2017	IH	Avaliação dos riscos
0.1	02/01/2017	IH	Construção do modelo de avaliação dos riscos
0.1	12/12/2017	IH	Construção dos planos de mitigação para os riscos
0.1	11/12/2016	IH	Declaração de riscos
0.1	11/12/2016	IH	Introdução
0.1	10/12/2016	IH	Criação do documento e da sua estrutura

Índice

Introdução

O presente documento servirá para a identificação dos riscos que o estagiário poderá enfrentar ao longo do estágio e maneiras como deve gerir estes.
A gestão dos riscos é um processo que identifica, avalia e prioriza os riscos, da manheira o estagiário conhecer os riscos e tentar evitar o acontecimento dos mesmos. Uma vez que o risco acontece o estagiário deverá conhecer como tem de reagir para contornar o risco de uma forma a não prejudicar o estágio em questão.
Os riscos podem ocorrer de forma natural em qualquer das fases do projeto, o estagiário deverá conseguir lidar com estes de acordo com as estratégias de mitigação. O estagiário para conhecer melhor os riscos que poderão acontecer deverá identificar os riscos de uma forma a ter uma estratégia para estes, que ajudaram lhe tomar decisões perante aos riscos que se tornam os problemas da realidade. Como a gestão de riscos não é só identificar e avaliar os processos críticos, será mantida uma vigilância, sobre os mesmos, ao longo de todo o estágio.

Avaliação

Nesta seção será descrito a forma de avaliar os riscos ao nível do impacto que eles podem chegar a ter e ao nível da probabilidade de acontecimento. A avaliação será realizada em conformidade com as tabelas representadas em baixo.

Impacto

Nível	Tipo de problema	Impacto no escalonamento
A (Mínimo ou nulo)	Nenhum	Mínimo ou nulo
B (<5%)	Algum Impacto	Necessidade de recursos adicionais
C (5% - 7%)	Impacto Moderado	Pequeno deslize nos objetivos inicialmente definidos perante a planificação
D (7% - 10%)	Grande Impacto	Grande deslize nos objetivos inicialmente definidos perante a planificação
E (>10%)	Inaceitável	O estagiário não irá conseguir alcançar os objetivos estipulados

Tabela 1 - Impacto

Probabilidade de acontecimento

Nível	Probabilidade de acontecer
A1	Remota
B1	Improvável
C1	Provável
D1	Muito provável
E1	Quase garantida

Tabela 2 - Probabilidade de acontecimento

Declaração de riscos e respetivos atributos

Declaração do risco	Id	Impacto	Probabilidade	ToS
Não domínio das tecnologias Microsoft, o que pode dificultar a construção da melhor solução	1	Algum Impacto	[0% - 20%]	Facilmente alcançados
Poucos conhecimentos do estagiário no desenvolvimento utilizando tecnologias web, o que proporciona mais esforço na implementação das tarefas	2	Algum Impacto	[0% - 20%]	Facilmente alcançados
Projetos de outras unidades curriculares ocuparem demasiado tempo, o que proporciona atrasos na planificação do estágio	3	Impacto Moderado	[40% - 60%]	Alcançados com algum esforço

Tabela 3 - Descrição dos riscos

A Tabela 3 apresenta a classificação de cada risco, classificando o "Impacto" e "Probabilidade" do acontecimento, dando como resultado o "*Threshold of Success*" (ToS), que são os objetivos estabelecidos para o estagiário alcançar.

		Impacto				
		A	B	C	D	E
Probabilidade de acontecimento	E1					
	D1					
	C1			3		
	B1					
	A1		1;2			

Tabela 4 - Matriz de riscos

Em conformidade com a classificação dos riscos, na Tabela 4 representada em cima, apresenta-se a classificação dos riscos e o impacto, dos mesmos, sobre o estágio.

Planos para mitigação dos riscos

Nesta seção serão apresentados todos os riscos e a forma de os contornar usando os planos de mitigação, controlo e acompanhamento dos respetivos riscos, da maneira a minimizar os possíveis danos causados ao estágio.

Não domínio das tecnologias Microsoft, o que pode dificultar a construção da melhor solução

O estagiário não domina as tecnologias da Microsoft, uma vez que nunca teve a interação com estes, isto poderá ser um risco pois apesar de todas as linguagens são de uma ou de outra maneira semelhantes pode existir algumas dificuldades no inicio da implementação, pois coincida com a aprendizagem/interação inicial com as tecnologias.

Plano de mitigação

O presente estagiário deverá encontrar exercícios de treino e implementá-los, para conhecer melhor as tecnologias. Deve seguir os conselhos que recebe ao longo do estágio pelos colegas do laboratório, pois estes são muito mais experientes e podem dar conselhos uteis.

Acompanhamento

Se o presente estagiário sentir dificuldades em interação com uma tecnologia/ferramenta especifica, depois de realizar varias pesquisas que não proporcionam o resultado esperado, deve procurar ajuda nos colegas do laboratório, uma vez estes são muito mais experientes podem já ter se enfrentado o problema especifico ou simplesmente explicar o funcionamento da tecnologia/ferramenta.

Plano de controlo

O estagiário sempre que sentir dificuldades, não pode perder muito tempo para ultrapassar esta dificuldade, não hesitar em questionar colegas do laboratório em caso de dificuldade se persistir durante algum tempo.

Poucos conhecimentos do estagiário no desenvolvimento utilizando tecnologias web, o que proporciona mais esforço na implementação das tarefas

O estagiário tem "pobre" experiencia na programação web, principalmente na parte do "Frontend", isto pode causar um relativo atraso na implementação das paginas da maneira que estes sejam mais apelativas e construtivas. Apesar da primeira iteração com o "ASP.NET" foi positiva, não se pode retirar o risco que pode crescer para um problema na criação das paginas mais complexas.

Plano de mitigação

O presente estagiário deverá conseguir encontrar e realizar os exercícios certos para ultrapassar esta dificuldade, sendo estes exercícios resultados de pesquisa da internet ou fornecidos pelos colegas do laboratório, sendo o segundo caso o melhor pois os colegas pela experiencia passada podem facilitar a aprendizagem com exercícios simples.

Acompanhamento

O estagiário deverá notar as suas maiores dificuldades e tentar ultrapassar estas de forma a encontrar novos exercícios e exemplos de resolução para o problema.

Plano de controlo

No caso de encontrar um exercício com certa dificuldade para o estagiário, este deve esclarecer a dificuldade com colegas, perguntar pelas possíveis resoluções do problema e encontrar exercícios semelhantes da maneira a manter e aplicar o conhecimento adquirido.

Projetos de outras unidades curriculares ocuparem o tempo em demasio, o que proporciona atrasos da planificação do estágio

O presente estagiário tem mais quatro unidades curriculares para alem da unidade curricular "Estágio/Dissertação" no primeiro semestre, o que é preocupante ao nível da gestão do tempo, pois se este for mal gerido, podem surgir problemas no cumprimento da planificação.

Plano de mitigação

Construção de uma agenda para apontar todas as tarefas a realizar pelo estagiário, assim construir a visão geral da planificação e tentar integrar as tarefas de varias atividades da forma a não haver a interação entre elas.

Acompanhamento

Constante consulta da agenda, da forma a conhecer o estado atual e atualizar a agenda com as tarefas novas ou prolongamento/diminuição do tempo das tarefas já existentes na agenda.

Plano de controlo

Sempre que uma tarefa demorar mais tempo do que esperado, o estagiário deverá refazer a calendarização de todas as tarefas, de forma a conseguir cumprir os objetivos.

Anexo C

Documento de Testes

Documento de Testes

Mestrado em Engenharia Informática
2016/2017

CosmeDesk: Módulo SaaS

(Versão 0.1)
19/05/2017

Estagiário:
Ivan Harasym
2011168230
harasym@student.dei.uc.pt

Orientadores:
Prof. Filipe Araújo
filipius@dei.uc.pt
Departamento de Engenharia Informática, Universidade de Coimbra

Eng. Alcides Manuel de Almeida Marques
alcides.marques@ipn.pt
Instituto Pedro Nunes

Tabela de revisões

Versão	Data	Autores	Descrição
1.0	16/06/2017	IH	Introdução de testes de sistema
0.5	09/06/2017	IH	Introdução de mais testes realizados, terminada a secção dos testes a unidade e testes de integração
0.3	22/05/2017	IH	Introdução de alguns testes realizados
0.1	20/05/2017	IH	Introdução
0.1	19/05/2017	IH	Criação do documento e a sua estrutura

Índice

Introdução

Este documento foi criado com a necessidade de testar o funcionamento da aplicação da aplicação desenvolvida no âmbito da disciplina Dissertação/Estágio em Sistemas de Informação, do estagiário Ivan Harasym. Este documento terá três secções que correspondem a testes a unidade, testes de integração e os testes de sistema. Os testes a unidade são feitos no "Visual Studio", e são efetuados, às classes existentes na camada de negocio e as funções que estas contenham, é testado se para uma dada entrada o resultado da classe/função o resultado é esperado. Testes de integração correspondem a testes manuais realizados a aplicação "Cosmedesk SaaS". Por fim os testes de sistema já englobam o sistema todo, ou seja, as duas aplicações que são "Cosmedesk SaaS" e as instancias criadas por esta da aplicação "Cosmedesk".

Testes a unidade

Os testes a unidade foram automatizados e escritos com ajuda da "Microsoft Unit Test Framework" que permite realizar testes a unidade no "Visual Studio". Os testes foram feitos para funções principais e mais complexas da camada do negocio, considerando não sendo necessário verificação das funções que façam simples redireccionamento para outras com poucas ou nenhumas manipulações ao/s objeto/s.

Para verificar se o retorno era esperado foram usadas as funções de validação da *framework*, alguns exemplos das funções mais usadas são:

- Assert.IsTrue(bool condition);
- Assert.IsInstanceOfType(Object object, Type expectedType);
- Assert.IsNotNull(Object object);
- Assert.AreEqual(Object expected, Object actual);

Os testes a unidade realizados foram executados com sucesso isso podemos confirmar na imagem que se seque.

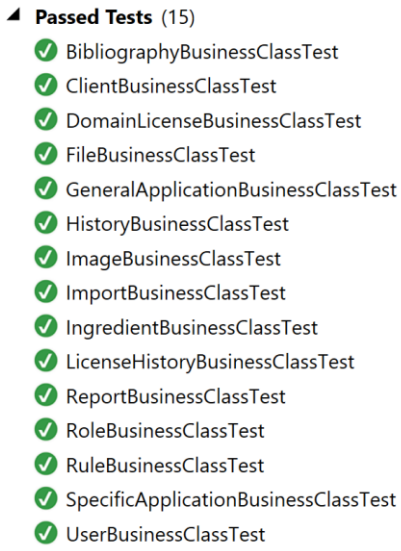
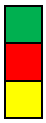


Ilustração 1: Testes a unidade

Testes a Integração

Esta parte do documento corresponde ao resultado dos testes manuais realizados a aplicação "CosmedeskSaaS".

Os testes que correspondem a ser esperados, ou seja, o resultado obtido é o resultado pretendidos serão marcados com a cor verde, os resultados não esperados e que podem causar estragos ou alterações de dados que não é suposto acontecer serão marcados com cor vermelha, os casos que não são esperados, mas também não causam estragos nem alterações de dados serão marcados com a cor amarela.



Os resultados pretendidos/esperados
Os resultados não pretendidos/esperados
Os resultados não esperados, mas que não causam estragos

Login

Teste	**Resultado**
Sem campos introduzidos	Mensagens: The User field is required The Password field is required
Introduzindo unicamente o email	Mensagem: The Password field is required
Introduzindo unicamente palavra chave	Mensagem: The User field is required
Introduzindo todos os campos	Login efetuado com sucesso transitando para pagina seguinte correspondente
Output correto	Sim

NavBar

Teste	Resultado
Clicar sobre nome do site	Transição para pagina por defeito
Clicar sobre o "Domain Licenses"	Transição para a pagina correspondente
Clicar sobre "Client"	Transição para a pagina correspondente
Clicar sobre "User"	Transição para a pagina correspondente
Clicar sobre "General application"	Transição para a pagina correspondente
Clicar sobre "Specific application"	Transição para a pagina correspondente
Clicar sobre "Ruls"	Transição para a pagina correspondente
Clicar sobre "Import"	Transição para a pagina correspondente
Clicar sobre "Bibliography"	Transição para a pagina correspondente
Clicar sobre "Ingredinet"	Transição para a pagina correspondente
Clicar sobre "License history"	Transição para a pagina correspondente
Clicar sobre "History"	Transição para a pagina correspondente
Clicar sobre "Number of clients"	Transição para a pagina correspondente apresentando o gráfico correspondente
Clicar sobre "Number of ingredient created"	Transição para a pagina correspondente apresentando o gráfico correspondente
Clicar sobre "Number of reports"	Transição para a pagina correspondente apresentando o gráfico correspondente
Clicar sobre "Set password"	Transição para a pagina correspondente
Clicar sobre "Logout"	Transição para janela login, terminando a secção
Clicar sobre "Charts"	Apresentação da lista dos gráficos disponiveis
Clicar sobre "Options"	Apresentação da lista das opções disponíveis
Clicar sobre "Welcome, #Email"	Apresentação da lista dos opções disponiveis

Domain license

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Domain license com o Domain License selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Domain License

Domain License Add new

Teste	Resultado
Submite com todos os campos vazios	Notificações sobre os campos obrigatórios
Submite com todos os campos preenchidos menos os ficheiros e imagem	Não faz submite, mas não apresenta mensagem
Submite com Domain existente	Notificação: Domain já existe
Submite com Database Name existente	Notificação: Database name existe
Submite com Todos os campos corretos	Faz submite e volta para pagina Domain Licenses
Cancelar	Redireciona para pagina Domain Licenses

Domain License Set

Teste	Resultado
Submite com todos os campos preenchidos menos a imagem	Não faz submite, mas não apresenta mensagem
Pressionar botão "Remove"	Remove a imagem da vista, mas não da base de dados
Submite com Todos os campos corretos	Faz submite e volta para pagina Domain Licenses
Cancelar	Redireciona para pagina Domain Licenses

Client

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Client com os dados do cliente selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de cliente

Client Add new

Teste	Resultado
Submite com todos/alguns campos vazios	Notificações sobre os campos obrigatórios
Submite com NIF existente	Notificação: NIF já existe
Submite com Todos os campos corretos	Faz submite e volta para pagina Clients
Cancelar	Redireciona para pagina Clients

Client Set

Teste	Resultado
Submite com alguns campos preenchidos	Notificação dos campos obrigatórios
Pressionar butão "Remove"	Marca o Cliente como removido
Submite com Todos os campos corretos	Submete e volta para pagina Clients
Cancelar	Redireciona para pagina Clients

Users

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Utilizador com os dados do utilizador selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Utilizador

Users Add

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com email já existente	Notificação do email já existente no sistema
Submite com campos "Password" e "Confirm Password"	Notificação password tem que corresponder
Submite com todos os campos corretos	Submete e redireciona para pagina Users
Cancelar	Redireciona para pagina Users

Users Set

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com email já existente	Notificação do email já existente no sistema
Submite com campos "Password" e "Confirm Password" diferentes	Notificação password tem que corresponder
Submite com todos os campos corretos	Submete e redireciona para pagina Users
Mudar para um email já existente	Submite e efetua a troca do email
Remover	Marca na base de dados como removido
Cancelar	Redireciona para pagina Users

General Application

Teste	Resultado
Clicar sobre um dos campos da tabela	Não faz nada
Clicar sobre o botão add	Apresentação de um campo novo na tabela para inserir o General Application
Clicar sobre botão edit	Seleciona o campo correspondente deixando editar o mesmo, também acrescenta dois botões para save e cancel
Clicar sobre botão save	Submete o General Application acrescentando este para a lista
Clicando sobre botão cancel	Volta a representar a lista em condições anteriores
Introduzir nome repetido na inserção	Apresenta de uma notificação com o erro

Specific Application

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Specific Application com os dados do Specific Application selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Specific Application

Specific Application Add

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com nome já existente	Notificação do nome já existente no sistema
Submite com todos os campos corretos	Submete e redireciona para pagina Specific Applications
Cancelar	Redireciona para pagina Specific Applications

Specific Application Set

Teste	Resultado
Submite com alguns campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com todos os campos corretos	Submete e redireciona para pagina Specific Applications
Remove	Marca na base de dados como removido
Cancelar	Redireciona para pagina Specific Applications

Rules

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Rule com os dados do Rule selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Ruls

Rule Add

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com nome já existente	Notificação do nome já existente no sistema
Submite com todos os campos corretos	Submete e redireciona para pagina Ruls
Cancelar	Redireciona para pagina Ruls

Rule Set

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com nome já existente	Notificação do nome já existente no sistema
Submite com todos os campos corretos	Submete e redireciona para pagina Ruls
Remove	Marca na base de dados como removido e redireciona para janela Ruls
Cancelar	Redireciona para pagina Ruls

Ingredient

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Rule com os dados do Rule selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Ruls

Ingredient Add

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com todos os campos corretos	Submete e redireciona para pagina Ingredients
Cancelar	Redireciona para pagina Ruls

Ingredient Set

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com todos os campos corretos	Submete e redireciona para pagina Ingredients
Remover	Marca na base de dados como removido e redireciona para janela Ingredients
Cancelar	Redireciona para pagina Ruls

Import

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Import com os dados do Import selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Import

Import Add

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com todos os campos corretos	Submete e redireciona para pagina Imports
Cancelar	Redireciona para pagina Imports

Import Set

Teste	Resultado
Back	Redireciona para pagina Imports

Bibliography

Teste	Resultado
Clicar sobre um dos campos da tabela	Apresentação da janela Bibliography com os dados do Bibliography selecionado
Clicar sobre o botão add	Apresentação de uma janela para criação de Bibliography

Bibliography Add

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com o nome existente	Submete e redireciona para pagina Bibliography
Submite com todos os campos corretos	Submete e redireciona para pagina Bibliography
Cancelar	Redireciona para pagina Bibliography

Bibliography Set

Teste	Resultado
Submite com alguns ou sem campos preenchidos	Notificação/notificações dos campos obrigatórios
Submite com todos os campos corretos	Submete e redireciona para pagina Bibliography
Cancelar	Redireciona para pagina Bibliography

History

Teste	Resultado
ShowEntity	Apresentação da janela com entidade corespondente

Correção

Users Set	Mudar para um email já existente	Apresentação do aviso correspondente
Bibliography Add	Submeter com o nome existente	Apresentação do aviso correspondente
Domain License Add	Submite com todos os campos preenchidos menos os ficheiros e imagem	Apresentação do aviso correspondente
Domain License Set	Submite com todos os campos preenchidos menos a imagem	Apresentação do aviso correspondente

Testes de Sistema

Esta parte do documento corresponde ao resultado dos testes manuais realizados a sistema todo, ou seja, a aplicação Cosmedesk e modulo *back office*. Estes testes mantenham a estrutura dos testes anteriores realizados e as anotações. Com uma única diferença na coluna de resultado irá ser colocado o resultado do *debug* que foi realizado, se todo o procedimento foi feito correctamente.

Teste	**Resultado**
Criação de uma licença base para um novo cliente	✓ Criação da base de dados ✓ Criação de um Site ✓ Criação de uma instância (application pool) ✓ Permissões de alteração e consulta a nova base de dados ✓ Permissões de consultas a base de dados do modulo ✓ Atribuídas as permissões para instância na pasta de versão correspondente ✓ Executado a publicação de aplicações genéricas ✓ Executado a publicação de aplicações especificas ✓ Cria a pasta para modelos dos relatórios ✓ Guarda os modelos dos relatórios na pasta correspondente para novo cliente ✓ Atribuídas as permissões na pasta criada para a nova instância
Disponibilização de um link valido para o cliente poder realizar registo	✓ Através do mail que o cliente recebe consegue realizar o registo com sucesso
Na utilização da aplicação Cosmedesk a aplicação vai buscar a licença correta	✓ Verifica a data de validade
Criação de um documento	✓ Verifica a data de validade ✓ Verifica se existe a possibilidade de gerar o relatório consoante a numero total dos relatórios para gerar ✓ Incrementa o numero dos relatórios gerados
Criação de um “General Application”	✓ General Application replicada pelas todas as bases de dados
Criação de um “Specific Application”	✓ Specific Application replicada pelas todas as bases de dados

Importação de ingredientes para um cliente especifico	✓ Os ingredientes selecionados replicados com sucesso na base de dados de um cliente no domínio escolhido
Criação de uma regra que afeta um relatório do cliente	✓ É apresentada uma notificação nas no painel das notificações para cliente