Mestrado em Engenharia Informática
Estágio
Relatório Final de Estágio

Estudo e desenvolvimento de uma plataforma de gestão de serviços SaaS para o sector do alojamento – subscrição e cancelamento de serviços

José Carlos Gonçalves de Miranda
jgonc@student.dei.uc.pt

Orientador IPN:
Eng. Bruno Almeida

Orientador DEI:
Prof. Pedro Martins

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

Resumo

Este estágio está inserido no projecto *Hotelcracy Apps,* que tem como objectivo o estudo e desenvolvimento de uma plataforma integradora de múltiplos serviços *Software-as-a-Service* utilizados no sector do alojamento. A plataforma permitirá ao utilizador a selecção de vários serviços que melhor se adequem ao seu negócio, podendo subcrever novos serviços, cancelar serviços subscritos e migrar de um serviço para outro. Os serviços subscritos ficarão imediatamente integrados com os restantes e serão utilizados através de uma *interface* homogénea.

A principal contribuição deste estágio para o projecto *Hotelcracy Apps* é o desenvolvimento de componentes que permitam a subscrição e cancelamento de serviços na plataforma, constituindo assim uma prova de conceito dos processos que executam. Para tal, começou-se por identificar várias categorias de serviços *Software-as-a-Service* que são utilizados no sector do alojamento e seguidamente identificaram-se e analisaram-se serviços de cada categoria existentes no mercado. Das categorias identificadas, seleccionaram-se duas essenciais no sector e estudaram-se aprofundadamente os serviços existentes no mercado dessas categorias.

O estudo realizado teve como resultados principais os processos detalhados de subscrição e cancelamento de serviços das duas categorias seleccionadas, bem como a definição dos processos genéricos de subscrição e cancelamento de serviços.

Partindo dos processos definidos, estabeleceu-se um modelo de gestão de dados que suporta e viabiliza as interacções entre a plataforma *Hotelcracy Apps* e os serviços SaaS e desenhou-se a arquitectura estática e dinâmica da prova de conceito dos componentes de subscrição e cancelamento de serviços da plataforma.

Terminada a concepção tecnológica da prova de conceito, implementaram-se os componentes que a constituem e definiram-se testes automáticos para a sua verificação e validação em ambiente laboratorial.

Índice

Lista de Tabelas

Lista de Figuras

Acrónimos

API	*Application Programming Interface*
BPMN	*Business Process Model Notation*
CRM	*Customer Relationship Management*
FCT	*Faculdade de Ciências e Tecnologias*
HTML	*Hyper Text Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IU	*Interface de Utilizador*
IPN	*Instituto Pedro Nunes*
JSON	*Javascript Object Notation*
MEI	*Mestrado em Engenharia Informáica*
MVC	*Model-View-Controller*
NIST	*National Institute of Standards and Technology*
RMS	*Revenue Management System*
OTA	*Online Travel Agency*
PMS	*Property Management System*
POS	*Point-of-Sale*
REST	*Representational State Transfer*
SaaS	*Software-as-a-Service*
SI&IDT	*Sistema de Incentivos à Investigação e Desenvolvimento Tecnológico*
SOAP	*Simple Object Access Protocol*
SSH	*Secure Shell*
SuD	*System under Development*
TDD	*Test-Driven Development*
TIC	*Tecnologias da Informação e Comunicação*
TPA	*Terminal de Pagamento Automático*
UC	*Universidade de Coimbra*
UML	*Unified Modeling Language*

URI	*Universal Resource Identifier*
URL	*Universal Rersource Locator*
XML	*eXtensible Markup Language*

Capítulo 1
Introdução

O presente relatório visa apresentar o trabalho realizado no contexto do estágio que tem por título "Estudo e desenvolvimento de um plataforma de gestão de serviços *SaaS* (*Software-as-a-Service*) para o sector do alojamento – subscrição e cancelamento de serviços", do ano lectivo 2016/2017, para a disciplina "Estágio/Dissertação" do Mestrado em Engenharia Informática (MEI) da Faculdade de Ciências e Tecnologia (FCT) da Universidade de Coimbra (UC). Este estágio está inserido no projecto *Hotelcracy Apps,* que é um projecto cofinanciado pelo sistema de incentivos à Investigação e Desenvolvimento Tecnológico (SI&IDT) do programa Portugal 2020. O trabalho aqui apresentado foi realizado no Instituto Pedro Nunes (IPN) – Associação para a Inovação e Desenvolvimento em ciência e tecnologia, segundo copromotor de uma parceria que conta também com a *Hotelcracy Software Lda*, empresa líder do projecto.

1.1. Objectivos do estágio

O principal objectivo do estágio é o desenvolvimento de componentes que permitam a subscrição (através de um *marketplace*) e cancelamento de serviços *Sofware-as-a-Service* (SaaS) para o sector do alojamento, com o propósito de realizar uma prova de conceito da viabilidade da subscrição e cancelamento de serviços SaaS através de uma plataforma de gestão, também ela SaaS.

Para a prova de conceito foram seleccionadas duas categorias de serviços SaaS, pois considera-se que será suficiente para demonstrar que mais tipos de serviços poderão ser subscritos e cancelados. Segundo a opinião da empresa líder do projecto – Hotelcracy Software, Lda – as duas categorias de serviços mais importantes para a fase em que o projecto se encontra na altura da realização deste estágio são os serviços *Channel Manager* e os serviços de facturação. Com base nesta opinião, estas foram as categorias de serviços SaaS selecionadas para a prova de conceito e, com base na análise do Estado da Arte realizada e apresentada no Capítulo 3 deste relatório, os serviços SaaS destas categorias que foram seleccionados para a concretizar são o InvoiceXpress e o InvoiceOcean (facturação) e o HotelRunner (*Channel Manager*). A descrição destas duas categorias, e também de todas as outras identificadas, será apresentada no Capítulo 3 deste relatório – Estado da Arte.

Para o primeiro semestre, as tarefas que estavam planeadas no início do estágio com vista a atingir o principal objectivo eram as seguintes:

- Análise do estado da arte: análise e síntese de características de serviços SaaS no sector do alojamento;
- Identificação de casos de uso e macro-requisitos do(s) componente(s) da subscrição e cancelamento de serviços.
- Definição detalhada de requisitos, e processos de subscrição e cancelamento de serviços.
- Análise preliminar de tecnologias para a implementação do(s) componente(s) de subscrição e cancelamento de serviços.

Para o segundo semestre, o planeamento das tarefas era o seguinte:

- Definição de um modelo de gestão de dados e suporte à interacção com os sistemas SaaS. Análise preliminar da interface com o utilizador. [tarefa colaborativa]
- Especificação tecnológica da arquitectura do(s) componente(s) de subscrição e cancelamento de serviços, no âmbito de um *marketplace* de soluções a implementar.
- Desenvolvimento/Implementação do(s) componente(s) de subscrição e cancelamento de serviços.
- Definição e desenvolvimento de testes ao(s) componente(s) de subscrição e cancelamento de serviços.
- Instalação e validação final do(s) componente(s) (prova de conceito do marketplace).

1.2. Enquadramento

Antes de apresentar o enquadramento do estágio e do projecto no qual o estágio se insere, importa clarificar o conceito *Software-as-a-Service (*SaaS*)* presente logo no título deste relatório. Segundo o *National Institute of Standards and Technology* (NIST), *Software-as-a-Service* (SaaS) é um dos três modelos de serviços que compõem o modelo denominado *Cloud computing*: *Software-as-Service*, *Platform-as-a-Service* (PaaS) e *Infrastructure-as-a-Service* (IaaS). Por sua vez, *Cloud computing* é um modelo para possibilitar o acesso via Internet a um conjunto partilhado e configurável de recursos computacionais (p.e. redes, servidores, armazenamento, aplicações e serviços). No caso do SaaS, é proporcionada ao cliente a utilização do software do fornecedor que está em execução numa infraestrutura *cloud*. O software é acessível através de clientes magros como um *web browser* (p.e. cliente de email na *web*), ou uma interface de utilização do software. O cliente não gere nem controla a infraestrutura *cloud* subjacente, incluindo redes, servidores, sistemas operativos ou armazenamento [1].

Este modelo de distribuição de software – SaaS – tem vindo a crescer ao longo dos últimos anos em todos os sectores, em particular no do alojamento, que é um sector que depende da utilização de muitos e variados softwares. Atenta a este facto e ao modelo de negócio do sector, em que os principais fornecedores procuram vender soluções integradas de vários softwares dando preferência a alguns parceiros de negócio, a equipa que idealizou o projecto *Hotelcracy Apps* propõe uma abordagem que consiga acompanhar a evolução das tecnologias através de uma solução integrada e personalizável *on-demand*.

O projecto *Hotelcracy Apps* pretende então desenvolver uma plataforma *SaaS* que dará a possibilidade a um hoteleiro de selecionar um conjunto de serviços *SaaS* que mais se adequem às necessidades do seu negócio. Os serviços ficarão integrados na plataforma de forma transparente de forma a estarem preparados para trocar informação entre si assim que são subscritos.

A plataforma permitirá ao hoteleiro subscrever novos serviços através de um *marketplace*, migrar de um serviço para outro conservando toda a informação e cancelar um serviço que deixará de fazer parte do conjunto de soluções utilizadas pelo hoteleiro. Além disto, permitirá também fazer uso dos diversos serviços sob uma *interface* homogénea que abstrairá o utilizador da complexidade associada à integração dos vários serviços subscritos e eliminará as dificuldades de lidar com diferentes modelos de interacção. O presente estágio não se ocupou dos processos de migração de um serviço para outro.

1.3. Documentação produzida

Um dos principais contributos do presente estágio para o projecto *Hotelcracy Apps* foi a documentação produzida a fim de registar o trabalho efectuado, não só para toda equipa do projecto mas também para o seu cliente (entregáveis), neste caso a entidade financiadora. A Tabela 1 lista e apresenta uma breve descrição dos documentos produzidos e que serão referenciados neste relatório.

Tabela 1 - Documentação produzida

REFERÊNCIA/NOME	DESCRIÇÃO
Anexo 1: E1_2_analise_plataformas_alojamento_v4.6 [2]	Documento em que se apresenta a identificação e descrição de categorias de serviços *SaaS* utilizados no sector do alojamento, a identificação e análise de soluções existentes no mercado dentro de cada categoria (120 serviços analisados de 14 categorias identificadas) e a análise e comparação de funcionalidades fornecidas pelos vários serviços da mesma categoria. É de salientar que este é um entregável do projecto.
Anexo 2: analise_especificação_APIs_Channel_Manager_v1.2 [3]	Documento em que são identificadas as funcionalidades principais da categoria de serviços *Channel Manager* e detalhado o formato de mensagem necessário para utilizar cada uma dessas funcionalidades através da *Application Programming Interface* (API) de seis (6) serviços *Channel Manager* diferentes. Além desse registo de trabalho essencialmente de recolha, é também apresentada neste documento uma proposta de formato de mensagem para cada uma das funcionalidades, para a API da plataforma *Hotelcracy Apps*.
Anexo 3: analise_especificação_APIs_Facturação_v0.5 [4]	Documento em que são identificadas as funcionalidades principais da categoria de serviços de facturação e detalhado o formato de mensagem necessário para utilizar cada uma dessas funcionalidades através da API de seis (6) serviços de facturação diferentes. Além desse registo de trabalho essencialmente de recolha, é também apresentada neste documento uma proposta de formato de mensagem para cada uma das funcionalidades, para a API da plataforma *Hotelcracy Apps*.
Anexo 4: mockups_subs_canc_frontoffice_sofia_v2 [5]	Documento que contém os esboços de ecrã desenhados no âmbito da análise da interface

REFERÊNCIA/NOME	DESCRIÇÃO
	com o utilizador descrita no Capítulo 6.
Anexo 5: E1_3_Inv_Def_processos_v1.1 [6]	Documento que apresenta os processos de subscrição e cancelamento definidos e o método utilizado para os definir. Este é também um entregável do projecto.

1.4. Estrutura do relatório

O Capítulo 2 deste relatório apresenta o planeamento inicial das tarefas do primeiro semestre deste estágio e depois apresenta como efectivamente decorreu o trabalho. É feita também uma análise às diferenças entre os dois. Ainda neste capítulo, é apresentado o planeamento das tarefas que foram realizadas no segundo semestre deste estágio e feita uma análise semelhante à realizada para o primeiro semestre.

Após o Capítulo 2, a estrutura deste relatório procura seguir o caminho que foi percorrido ao longo dos dois semestres do ano lectivo para serem alcançados os objectivos estabelecidos. No Capítulo 3 são apresentadas as tarefas realizadas no âmbito do primeiro objectivo do estágio, análise do estado da arte: análise e síntese de características de serviços SaaS no sector do alojamento.

No Capítulo 4, as tarefas levadas a cabo para a identificação de macro-requisitos dos componentes de subscrição e cancelamento de serviços são apresentadas e o Capítulo 5, encerrando a descrição das tarefas realizadas no primeiro semestre do estágio, descreve os processos de subscrição e cancelamento de serviços.

No Capítulo 6 dá-se início à descrição das tarefas realizadas no segundo semestre deste estágio, sendo apresentado o modelo de gestão de dados especificado para os componentes de subscrição e cancelamento de serviços. No Capítulo 7 é descrita a análise da interface com o utilizador realizada. O Capitulo 8 é dedicado à especificação da arquitectura desses mesmos componentes e o Capítulo 9 descreve o processo de desenvolvimento, de testes e de instalação da prova de conceito.

Por último, o Capítulo 10 apresenta as conclusões retiradas do trabalho realizado ao longo do primeiro semestre e como esse trabalho permitiu antever a viabilidade das tarefas que foram levadas a cabo no segundo semestre. Apresenta também as conclusões do trabalho realizado no segundo semestre e os contributos do estágio para o projecto *Hotelcracy Apps*.

Capítulo 2
Planeamento do estágio

Sendo este estágio inserido no projecto *Hotelcracy Apps*, o planeamento das tarefas a realizar pelo estagiário estão intimamente relacionadas com o planeamento das actividades constituintes do projecto. Na Tabela 2 apresenta-se então o planeamento das actividades do projecto *Hotelcracy Apps* para as quais o primeiro semestre do presente estágio contribuiu.

Tabela 2 - Planeamento projecto Hotelcracy Apps

DATA DE INICIO	DATA DE CONCLUSÃO	ACTIVIDADE/TAREFA
01-10-2016	30-04-2017	Análise e monitorização de serviços *SaaS*
01-12-2016	31-05-2017	Investigação e definição de processos automáticos de subscrição e cancelamento de serviços *SaaS*

De forma a enquadrar os objectivos do estágio com o planeamento do projecto, foi realizada uma divisão das actividades apresentadas na Tabela 2 em várias tarefas mais concretas. A Figura 1 apresenta essa divisão bem como a sua duração planeada inicialmente.

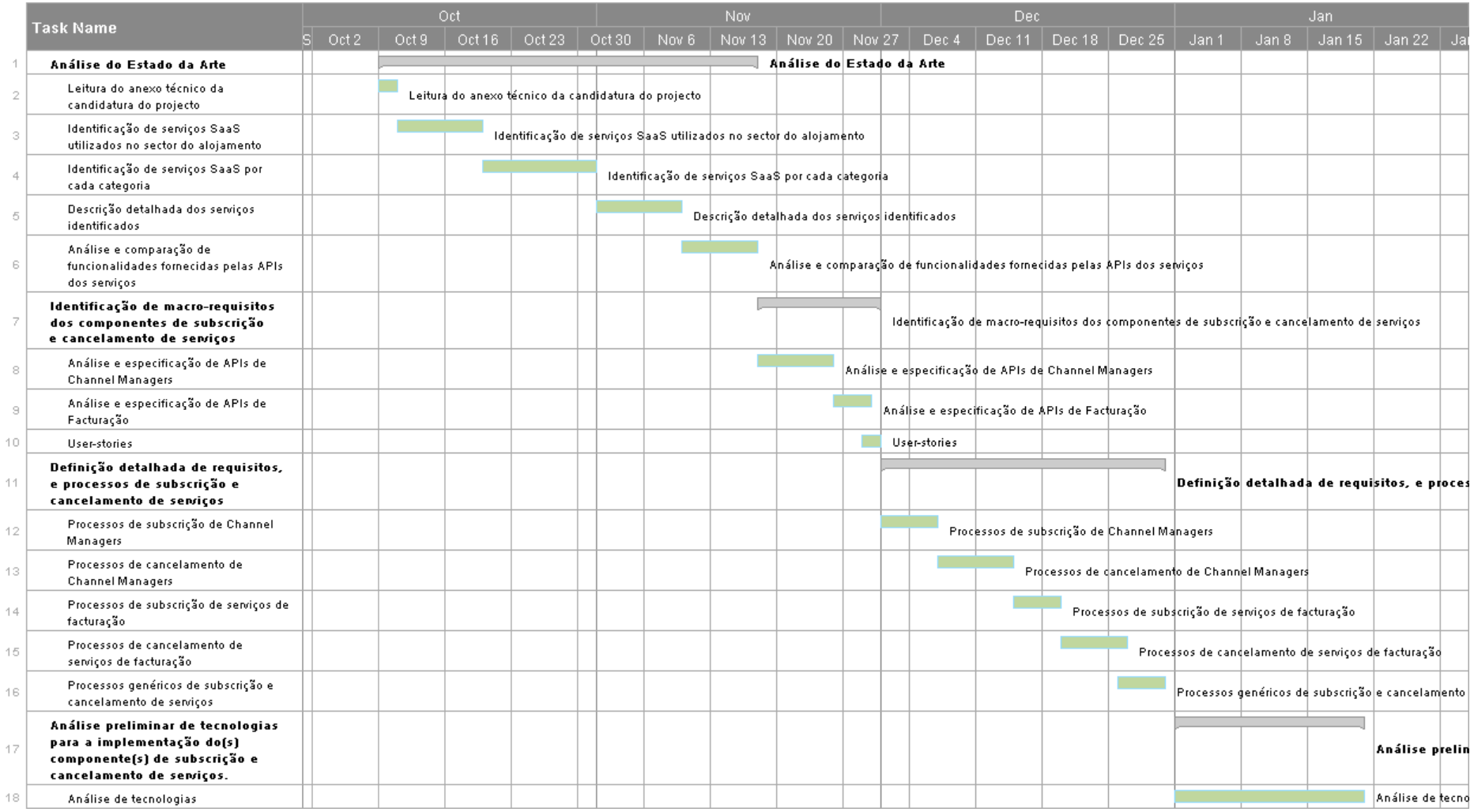

Figura 1- Diagrama de Gantt inicial do 1º semestre do estágio

A Figura 2 apresenta como decorreu efectivamente o trabalho realizado durante o semestre.

Pode-se constatar pela visualização da Figura 2, que foram concluídas as seguintes tarefas:

- Análise do estado da arte: análise e síntese de características de serviços SaaS no sector do alojamento;
- Identificação de casos de uso e macro-requisitos do(s) componente(s) da subscrição, uso e cancelamento de serviços.
- Definição detalhada de requisitos, e processos de subscrição e cancelamento de serviços.

E que não foi concluída a seguinte tarefa:

- Análise preliminar de tecnologias para a implementação do(s) componente(s) de subscrição e cancelamento de serviços.

O incumprimento desta tarefa deve-se ao facto de ao longo do semestre se ter tornado notório que as tarefas do estágio agendadas para o mês de Novembro exigiam mais do que apenas um mês para serem concluidas. Este facto não é totalmente surpreendente, pois as tarefas do estágio agendadas para Novembro de 2016 inserem-se numa actividade do projecto cuja data de conclusão é em Abril de 2017 (Tabela 2). Além disso, o número de serviços identificados foi maior que o esperado e, consequentemente, a sua análise e descrição demorou mais tempo do que o estimado.

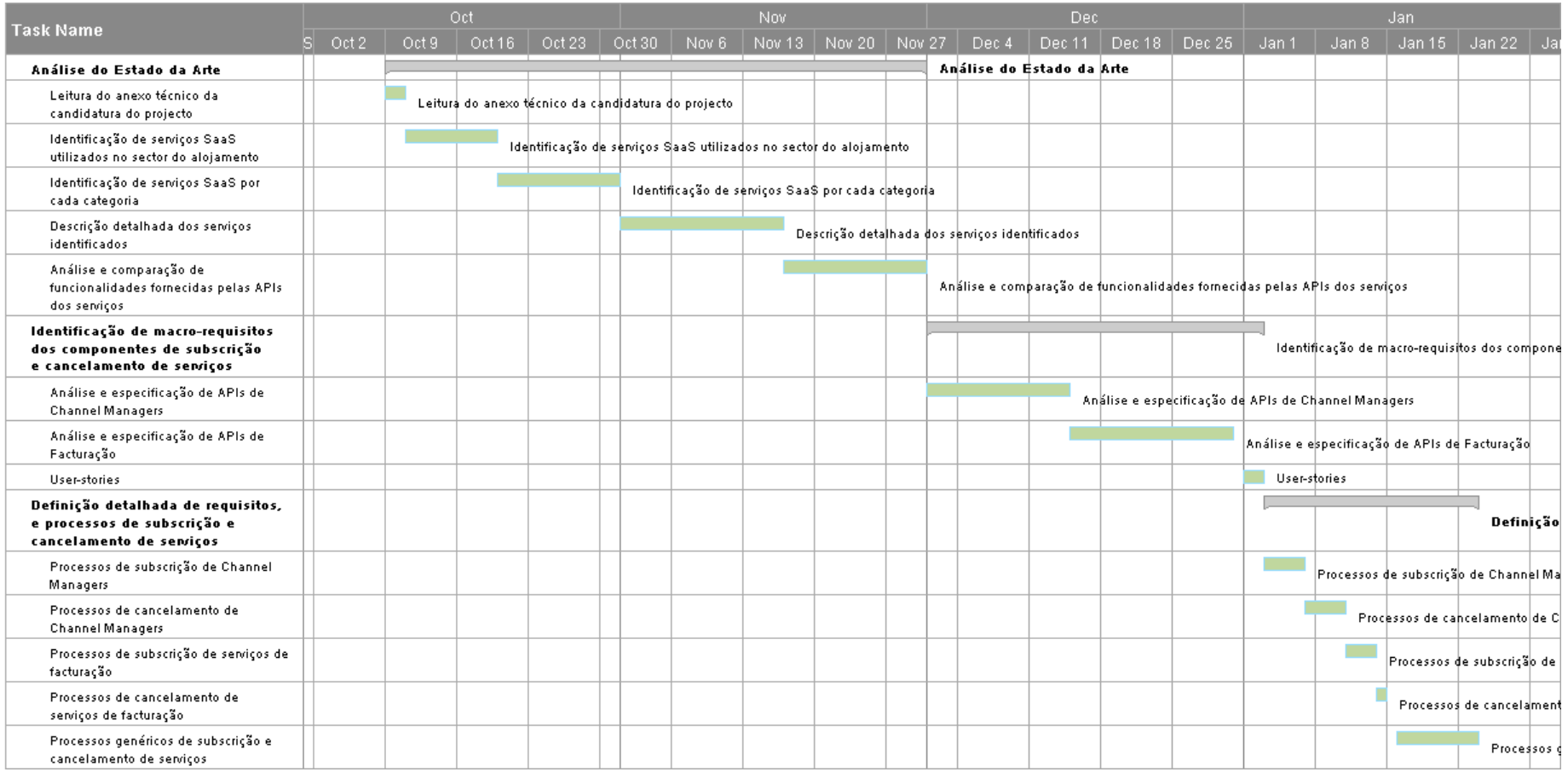

Figura 2 – Diagrama de Gantt real do 1º semestre do estágio

Para o segundo semestre, o planeamento inicial das tarefas que foram realizadas e a sua duração estimada é apresentado na Figura 3. Como se pode verificar, a tarefa inicialmente agendada para o primeiro semestre do estágio deslizou para o segundo semestre.

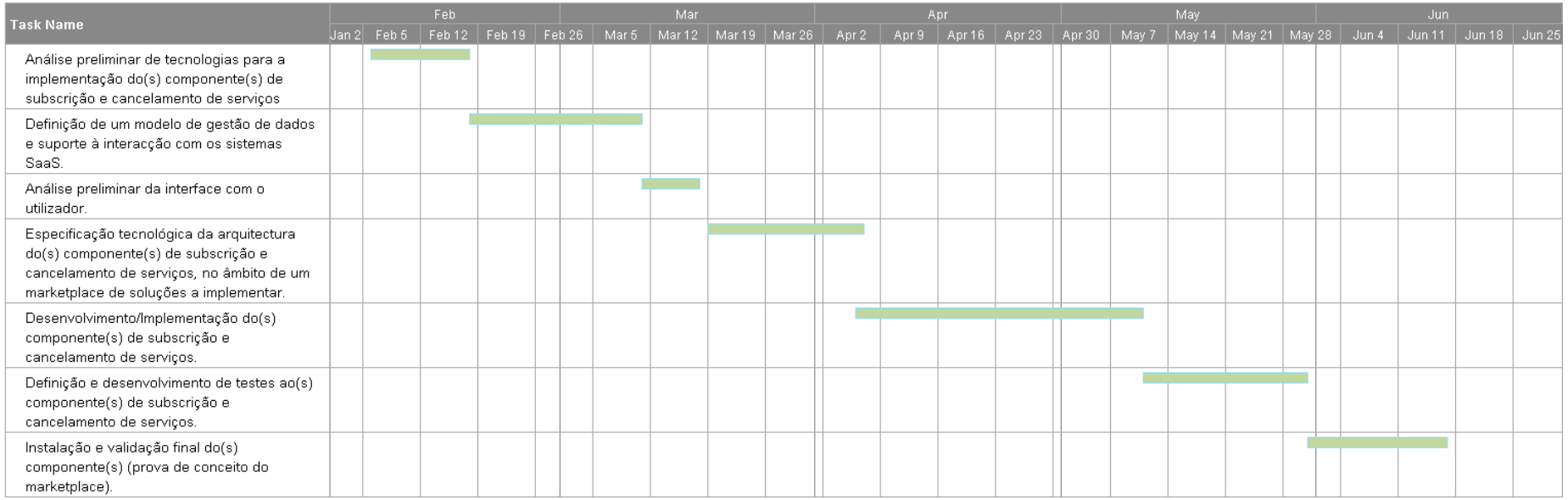

Figura 3- Gantt chart do planeamento para o 2º semestre do estágio

A Figura 4 apresenta como decorreu efectivamente o trabalho realizado durante o semestre. Como se pode verificar, a tarefa de análise de tecnologias para a implemetação dos componentes de subscrição e tecnologias foi removida. Isto deve-se às restrições tecnológicas impostas pela empresa líder do projecto – Hotelcracy Apps Lda. – e, embora na fase em que o projecto se encontra aquando da finalização deste estágio ainda estarem a ser discutidas diferentes abordagens tecnológicas para o desenvolvimento dos processos de integração e migração de serviços, estas não se aplicam aos processos de subscrição e cancelamento pois estes últimos são processos em que não ocorrem interacções entre diferentes serviços. Este tópico será abordado com mais detalhe no Capítulo 7 deste relatório onde se especifica a arquitectura dos componentes para a subcrição e cancelamento de serviços SaaS.

Pode-se verificar também pela Figura 4, que as tarefas de desenvolvimento da prova de conceito e a definição dos respectivos testes passaram a ser tarefas paralelas. Este facto prende-se com o metodologia de desenvolvimento utilizada – *Test-Driven Development* (TDD) – que será abordada no Capítulo 8.

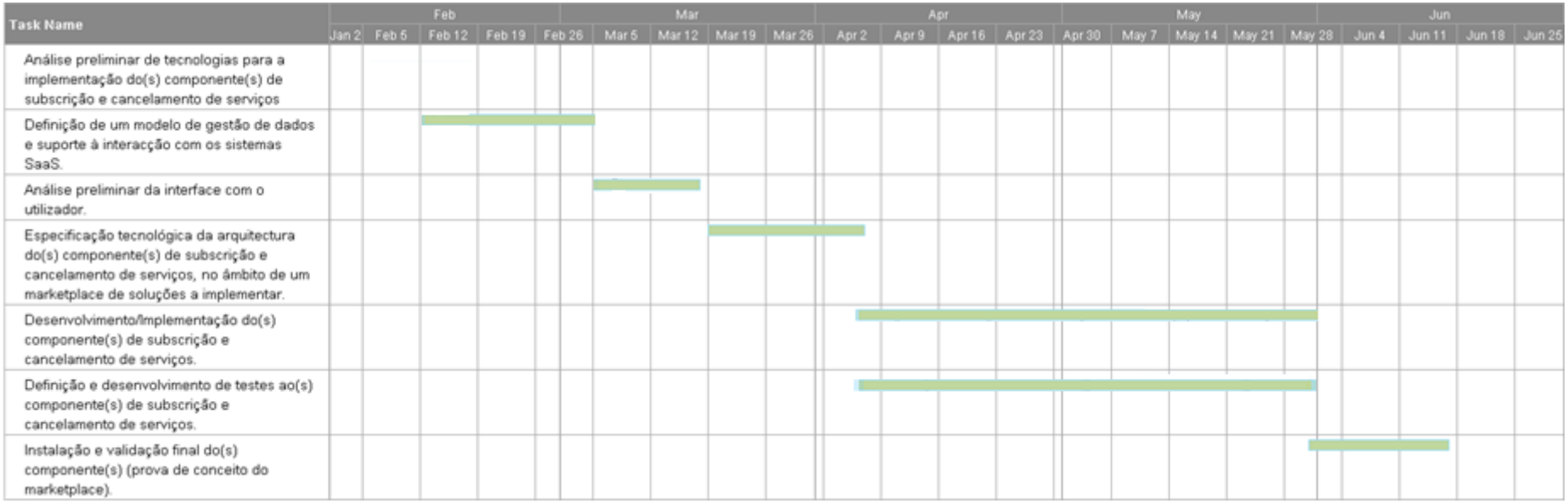

Figura 4 - Gantt chart real do 2º semestre do estágio

Capítulo 3
Estado da Arte – Serviços SaaS no sector do alojamento

Como ponto de partida para a análise do estado da arte que este capítulo descreve foi lida e interiorizada a análise do estado da arte que consta no anexo técnico da proposta de candidatura a financiamento do projeto *Hotelcracy Apps* [7].

Depois desta leitura e interiorização, a primeira tarefa essencial para a concretização dos objetivos do projeto em que este estágio se insere foi a identificação das várias categorias de serviços *SaaS* que são utilizados no sector do alojamento. Alguns destes serviços são específicos do sector enquanto outros são transversais a todos os tipos de negócio. Naturalmente o foco inicial incidiu nos primeiros.

Após a identificação das categorias de serviços utilizados, procedeu-se à pesquisa de serviços *SaaS* que se inserem dentro de cada categoria e posteriormente à análise detalhada dos serviços encontrados.

As secções seguintes visam apresentar os resultados da identificação, pesquisa e análise realizada.

3.1. Trabalho anterior

O trabalho de análise do estado da arte feito anteriormente ao início deste estágio incidiu no estado actual das Tecnologias da Informação e Comunicação (TIC) na gestão no sector do alojamento. Nessa análise conclui-se que nos últimos anos ocorreu um significativo acréscimo de maturidade na utilização das TIC neste sector, originando a dependência de ferramentas que se tornaram factores críticos de sucesso do negócio. Algumas destas ferramentas foram identificadas nessa análise e serviram de base à análise do estado da arte realizada no âmbito do presente estágio.

Com base num estudo realizado pela *Hospitality Technology Magazine* [8], a análise conclui que nos próximo anos existirá uma procura continuada de novas soluções tecnológicas e ferramentas para o sector do alojamento e que a migração para *cloud* é destacada como uma das prioridades de um número significativo de empresas do sector.

Essa mesma análise identifica um conjunto de soluções existentes no mercado para algumas categorias de serviços utilizados no sector do alojamento e algumas das suas características. Por sua vez, a investigação e análise realizada no âmbito do presente estágio que foi documentada no Anexo 1 [2] deste relatório, permitiu encontrar mais categorias de serviços e mais serviços *SaaS* de cada categoria. Além disso, permitiu descrevê-los de um ponto vista técnico, (por exemplo, estado de maturação das APIs), financeiro (por exemplo, custos de utilização do serviço e modelo de preço) e do ponto de vista de qualidade do serviço (por exemplo, *Service Level Agreement* e termos e condições de utilização).

3.2. Identificação de categorias de serviços *SaaS*

A identificação das várias categorias de serviços *SaaS* utilizados no sector do alojamento teve por base o anexo técnico da proposta de candidatura a financiamento do projeto *Hotelcracy Apps* pois nesse documento são referidos vários tipos de serviços *SaaS* específicos do sector.

Foram identificados catorze tipos de serviços distintos - e estudadas as suas principais áreas de operação - que são apresentados nas secções subsequentes.

S1 - *Property Management System*

O *Property Management System* (PMS) é o *software* usado na indústria hoteleira para gerir as operações centrais de um hotel. Algumas das suas funções básicas são: *check-in* e *check-out* de hóspedes, perfis de hóspedes, serviços de monitorização, geração de relatórios, auditoria, serviços de *front* e *back office* e sistemas de segurança. O PMS gere por exemplo toda a informação relativa aos quartos e ao serviço de limpeza.

Alguns PMS's podem ser integrados com sistemas já existentes no hotel, enquanto outros oferecem soluções completas que incluem todos ou alguns dos serviços identificados [9].

S2 - Channel Manager

O Channel Manager é o *software* que permite aos hóteis e alojamentos disponibilizarem os seus quartos pelos diversos canais de distribuição como por exemplo *websites* de reservas ou operadores turísticos (grossistas) [10]. Além da distribuição, o Channel Manager permite centralizar e gerir as reservas do hotel efectuadas em diferentes locais *online* [11]. O Channel Manager lida por exemplo com informação sobre os preços dos quartos do hotel e sua disponibilidade.

S3 - *Revenue Management System*

O *Revenue Management System* é o *software* utilizado para controlar as quantidades e os preços do inventário do hotel (neste sector o inventário corresponde aos quartos do alojamento [12]) com o objectivo de obter o máximo de retorno ou lucro gerindo a disponibilidade, os tipos de quartos, padrões de estadia, entre outros [13].

S4 - *Reputation Manager*

O surgimento de plataformas *online* especializadas na avaliação de hotéis e a crescente importância das redes sociais leva a que potenciais clientes possam ter acesso a críticas e problemas dos hotéis. Por esse motivo, é importante para o negócio acompanhar a informação partilhá-la, devendo ser analisada de forma a melhorar e/ou a corrigir determinados serviços. Os *Reputation Managers* são aplicações que automatizam esta tarefa [14]. Por exemplo se um hóspede se queixar do estado de limpeza de alguma zona das instalações, a aplicação pode alertar o proprietário deste facto que por sua vez acelera o processo de limpeza.

S5 - *Rate Shopper*

O Rate Shopper é o *software* que recolhe os preços dos concorrentes e ajuda a analisá-los [15]. Esta informação é utilizada pelo *Revenue Management System* a fim de obter o máximo retorno ou lucro.

S6 - *Customer Relationship Management*

Customer Relationship Management é o conjunto de princípios, práticas e directrizes que uma organização segue na interação com os seus clientes [16]. Naturalmente, a maioria dos negócios beneficia da implementação de estratégias de *Customer Relationship Management* mas, no sector hoteleiro, é um caso em que estas estratégias se tornam essenciais, pois todos os seus serviços giram em torno dos hóspedes, eles são a base do negócio quer este seja de pequena ou de grande dimensão, uma cadeia de hotéis ou um hotel independente [17]. Para facilitar ou até tornar possíveis estas estratégias, existem *softwares* que automatizam determinadas tarefas de CRM tais como recolher e analisar informação dos clientes, gerir reclamações e pedidos de serviços ou até desenvolver e executar campanhas publicitárias adequadas a determinado cliente [18].

S7 - *Booking Engine*

Outro dos processos do negócio hoteleiro que necessita de artefactos de *software* é o processo responsável pelas reservas *online*. É importante que os hotéis além de permitirem reservas através de *Online Travel Agencys* (OTA's) - *websites* especializados na venda de produtos de viagem como voos e hotéis [19] - também forneçam essa possibilidade aos clientes através do seu próprio *website* pois pode ter sido por este último meio que o hotel foi encontrado. Acresce a isto o facto das reservas efetuadas diretamente no *website* do hotel evitarem comissões por parte das OTA's [20]. Este *software* recolhe dados sobre o cliente, sobre as datas da estadia, sobre os quartos pretendidos, entre outros.

S8 - Facturação e pagamentos

Um *software* de facturação permite gerir o seguimento de produtos e serviços - fornecidos e prestados - e facturá-los aos clientes. Tarefas que costumavam ser demoradas, como preparar facturas e outra documentação, passam a ser automatizadas facilitando a administração de um negócio [21]. Este *software* ocupa-se de toda a informação que pode constar numa factura como por exemplo: dados do cliente, serviços utilizados, preços, dados da empresa prestadora dos serviços e assim por diante.

S9 - *Point-of-Sale*

Um sistema *Point-of-Sale* (POS) é a substituição informatizada de uma caixa registadora [22]. Como os hotéis costumam incluir serviços de bar e restauração, a utilização destes sistemas é útil na gestão das transacções realizadas no interior do hotel. Além da gestão de transacções contém ainda funcionalidades de que o negócio pode benefeciar, por exemplo, o registo do tempo de trabalho de um empregado de caixa ou a gestão de inventário [23]. Os sistemas POS poderão ainda ser utilizados nos serviços de SPA.

Os sistemas POS utilizados nos hotéis devem permitir a transferência da conta do cliente para o seu fólio (lista com todos os serviços de que o cliente usufruiu), portanto é essencial que o POS permita comunicação com o PMS [24].

S10 - Terminal de Pagamento Automático Virtual

Um Terminal de Pagamento Automático (TPA) é um sistema de pagamento que visa facilitar as transações comerciais. Um TPA é um dispositivo que aceita cartões bancários portugueses (Multibanco) e estrangeiros (p.e. Visa e Mastercard) para pagamentos eletrónicos. Existem vários tipos de TPA, um deles é o virtual que permite uma ligação ao

posto de venda via Internet, sem que exista um dispositivo físico instalado no *Point-of-Sale* ou outro sistema que precise de receber pagamentos [25].

Dado que as reservas *online* vêm acompanhadas por cartões de crédito e tendo em conta desenvolvimentos recentes como o aparecimento de novas formas de pagamento (*apple pay, google wallet, meo wallet, bitcoin etc.*) ou o movimento para aumentar a segurança dos dados de cartões de crédito, torna-se cada vez mais relevante que os negócios de alojamento disponham de soluções de pagamento virtuais.

S11 - Operações de manutenção e relacionados

Este tipo de *software* permite gerir a manutenção de infraestruturas e equipamentos de uma empresa. Oferece por exemplo a possibilidade de fazer a gestão integral de avarias ou organizar informação relativa a intervenções de reparação em edifícios e equipamentos [26]. Estando o sector hoteleiro diretamente relacionado com edifícios e seu recheio, é uma mais-valia dispor de *software* deste tipo.

S12 - Aplicações móveis

Para melhorar a retenção de clientes, a relação com os clientes e aumentar a sua fidelidade, os hotéis podem disponibilizar aplicações móveis a clientes e a não-clientes que servem como guia turístico digital e rececionista. Estas aplicações podem, por exemplo, fornecer um mapa com as direções do alojamento diretamente ao cliente. Este mapa pode conter também pontos de interesse escolhidos nas redondezas do alojamento [27].

S13 - Gestão de recursos humanos

Este é um tipo de *software* que permite gerir e processar, de forma fácil e intuitiva, toda a informação referente aos recursos humanos de uma organização [28]. Fornece por exemplo a possibilidade de agendar as horas de trabalho e comunicar com os trabalhadores de um hotel [29].

S14 - Gestão financeira/tesouraria

A gestão financeira é atividade que envolve o planeamento, a análise e o controlo de contas para que os empreendedores consigam visualizar a situação financeira da sua empresa [30]. Este tipo de *software* facilita esta atividade realizando tarefas de forma automática como a captura de movimentos bancários [31].

3.3. Visão geral das categorias de serviços identificadas

Com o objectivo de fornecer uma visão geral das categorias de serviços utilizados no sector do alojamento, desenhou-se um diagrama (Figura 5) que concentra as designações das categorias identificadas e ilustra algumas das possíveis interacções entre os serviços das mesmas.

Como se verifica pelo diagrama, o serviço com o maior número de interacções possíveis é o *Property Management System* (PMS), devendo-se isto ao facto de este ser responsável pelas operações centrais de um hotel. Os parágrafos seguintes explicam sucintamente as várias interacções.

A interacção entre o PMS e o *Channel Manager* é indispensável pois este último necessita ter conhecimento da disponibilidade de quartos do hotel e é o PMS que tem esta informação, por outro lado o PMS necessita saber que reservas foram efectuadas através das *Online Travel Agency's* (OTA's). Por sua vez, o *Channel Manager* recebe reservas para o hotel por parte das OTA's.

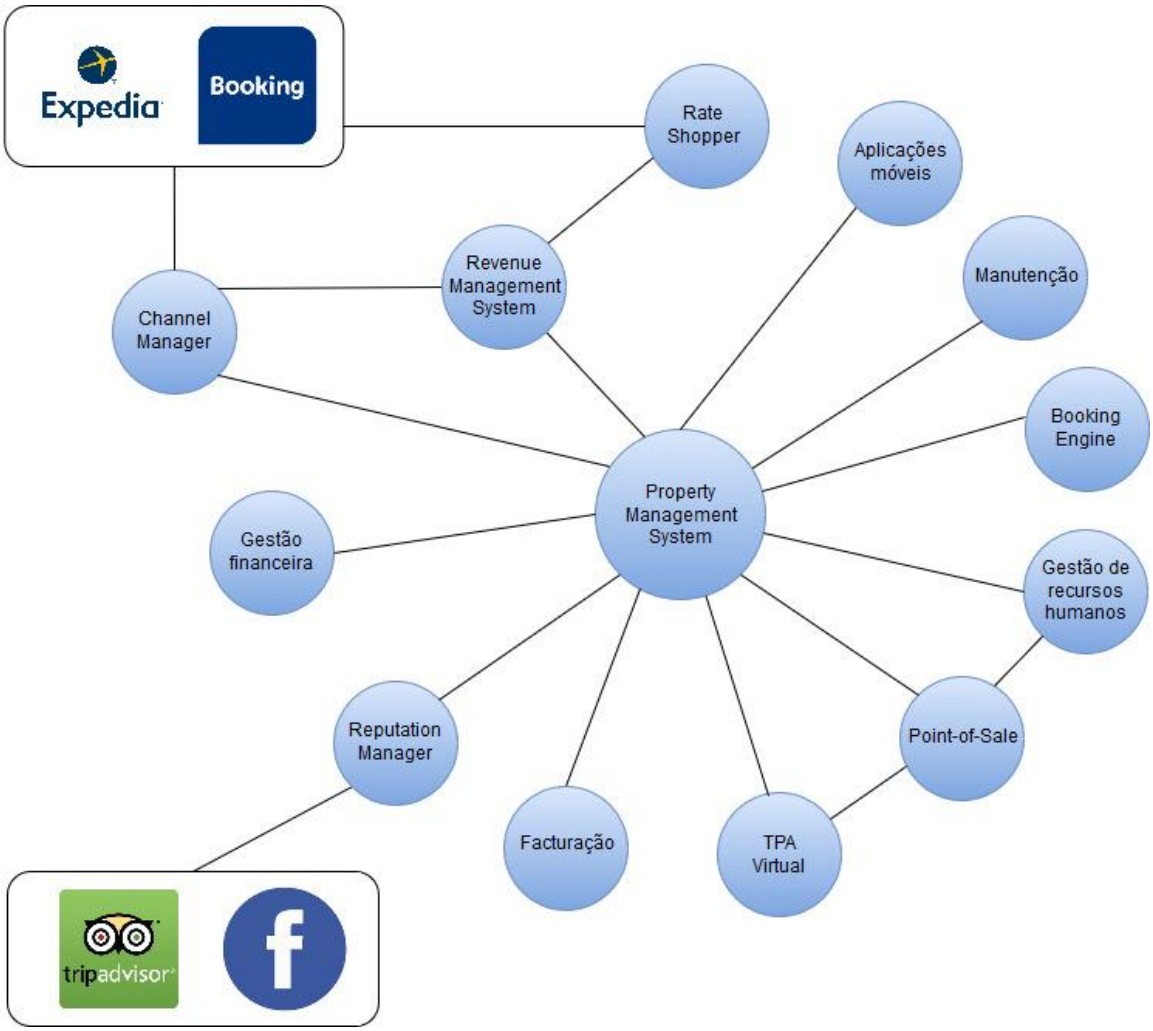

Figura 5 - Diagrama de interações entre serviços

O *Revenue Management System* (RMS) interage com o *Rate Shopper* pois é este que recolhe os preços da concorrência – por exemplo, consultando as OTA's – fornecendo informação valiosa para o RMS fixar os preços mais vantajosos. Tanto o *Channel Manager* como o PMS necessitam ter conhecimento dos preços definidos pelo RMS.

Como o serviço de Manutenção gere, por exemplo, as intervenções que os quartos do hotel possam necessitar e o *Property Management System* (PMS) gere os quartos que um hotel tem disponíveis, é importante que o serviço de Manutenção informe o PMS sobre os quartos para os quais este não pode aceitar reservas.

A *Booking Engine* de um hotel tem de estar permanentemente em contacto com o PMS, informando-o de novos pedidos de reserva efectuados e recebendo por parte do PMS a disponibilidade e preços dos quartos do hotel.

O serviço de Gestão de recursos humanos pode beneficiar da interacção tanto com o PMS como com o *Point-of-Sale,* pois estes são dois serviços que são utilizados por trabalhadores do

hotel e, como tal, têm informação – por exemplo, tempo ou carga de trabalho de um recepcionista – que pode ser utilizada para uma melhor gestão de recursos humanos.

Tal como referido em cima (Secção 3.2 – *Point-of-sale*, pag. 15), a interacção entre o *Point-of-Sale* (POS) e o *Property Management Sytem* (PMS) é indispensável porque o POS necessita transferir a conta com consumo do cliente no bar ou restaurante para o PMS.

O Terminal de Pagamento Automático Virtual interage, naturalmente, com os serviços que precisam de receber pagamentos por parte dos clientes, ou seja, o PMS (por exemplo, operações de *check-out* de clientes) e o *Point-of-Sale* (por exemplo, pagamento de uma refeição).

Os serviços de Facturação e o de Gestão Financeira podem interagir por exemplo com o PMS, pois é este que tem informação sobre os clientes do hotel - serviços e produtos consumidos.

Por último, intrínseca ao *Reputation Manager* é a interacção com diversas plataformas de avaliação e crítica de hotéis ou redes sociais como referido em cima (Secção 3.2 – *Reputation Manager,* pag. 14). Outra interacção possível deste serviço é com o *Property Management System* (PMS), podendo enviar-lhe avaliações de um cliente já conhecido permitindo ao PMS associá-las ao cliente.

Como se pode verificar, o ecossistema de ferramentas tecnológicas utilizadas no sector do alojamento é muito amplo e continua a crescer com o aparecimento de novas soluções no mercado para o sector. O leitor estará agora em melhores condições de entender a necessidade a que o projecto *Hotelcracy Apps* pretende dar resposta: proporcionar ao gestor de hotel uma forma simples de adicionar, remover e migrar serviços a utilizar no seu negócio.

3.4. Identificação de serviços *SaaS* por cada categoria

Depois de identificadas as categorias de serviços *SaaS* usados no sector do alojamento, procedeu-se à pesquisa de serviços disponíveis no mercado por categoria identificada.

O método utilizado para identificar as soluções apresentadas nesta secção baseou-se no conhecimento do Hotel Oslo e na realização de pesquisas *online* no motor da *Google* [32]. Utilizaram-se as seguintes palavras de pesquisa:

- PMS; Property Management System; Hotel Management System; Hotel Management Software.
- Channel Manager; Hotel Distribution.
- RMS; Revenue Management System.
- Reputation Manager; Hotel Reputation Management.
- Rate Shopper; Hotel Rate Shopping.
- CRM; Customer Relationship Management; Hotel CRM.
- Booking Engine.
- Invoice software; Billing software; Hotel billing software.
- POS; Point-of-sale; Hotel POS.

- TPA virtual.
- Software for maintenance; Facilities management.

Estas palavras-chave foram combinadas (manualmente) com as palavras API, Prices, License, Terms, Developers, Documentation; JSON; XML; Integration.

Para descrever os serviços encontrados, optou-se por identificar um conjunto de campos que permitissem resumir a informação de cada serviço e registá-los sob a forma de uma tabela para cada serviço. Os campos utilizados para a descrição dos serviços foram os seguintes:

- **Versão da tabela**: como as descrições realizadas são susceptíveis de sofrer alterações ao longo do decorrer do projecto, este campo é necessário para identificar a versão actual da informação contida na tabela da solução.
- **ID**: identificador atribuído ao serviço *SaaS*.
- **Nome**: nome do serviço *SaaS*.
- ***Uniform Resource Locator*** (URL): endereço web para a descrição do serviço *SaaS*.
- **API**: identifica se a solução dispõe (Sim) ou não (Não) de uma *Application Programming Interface* (API).
- **Observações sobre a API**: no caso de a solução dispor de uma API, apresenta dados relevantes sobre a mesma, por exemplo, protocolos de comunicação e documentação.
- **Licença**: identifica a informação sobre a licença de uso da API: quem tem acesso, preços de subscrição, entre outros.
- **Termos e Condições**: endereço do documento de termos e condições do serviço.

Para exemplificar a descrição dos serviços, a Tabela 3 apresenta um dos serviços identificados.

Tabela 3 - Exemplo de tabela descritiva de serviços

VERSÃO DA TABELA	1.1
ID	CM_17
NOME	MyAllocator
URL	https://www.myallocator.com/
API	Sim
OBSERVAÇÕES SOBRE A API	A API implementa o protocolo REST. A API aceita/fornece conteúdos no formato XML e JSON. Documentação da API em: http://myallocator.github.io/apidocs/
LICENÇA	Necessário ser parceiro da empresa para usar a API.
TERMOS E CONDIÇÕES	https://www.myallocator.com/terms-and-conditions/
CONTROLO DE VERSÕES	**v1.0** (19/10/2016)

	- Primeira descrição do serviço. **v1.1** (04/11/2016) - Atualização do campo Observações sobre a API. - Adicionado o campo Termos e Condições.

O conjunto completo de tabelas que descreve 120 (cento e vinte) serviços SaaS distintos encontram-se no Anexo 1 [2]deste relatório.

Além das tabelas descritivas de cada serviço, das quais a Tabela 3 é exemplo, criaram-se tabelas – uma para cada categoria – que resumem e concentram a informação recolhida, com o objectivo de proporcionar uma visão geral das características dos serviços encontrados de cada categoria identificada. Os campos utilizados nestas tabelas resumo foram:

- **Nome:** nome da plataforma/ferramenta SaaS.
- **API:** identifica se a solução dispõe (Sim) ou não (Não) de uma *Application Programming Interface* (API).
- **Protocolos de comunicação:** protocolos de comunicação que a API utiliza.
- **Formato dos dados**: formato dos conteúdos que a API aceita/fornece.
- **Acesso à API:** identifica quem tem acesso à API - o serviço pode, por exemplo, permitir o acesso à sua API aos seus subscritores (utilizadores finais) ou permitir o acesso apenas a outros fornecedores de software para estes integrarem o seu software com o serviço.
- **Documentação da API:** identifica se a documentação da API está (Sim) ou não (Não) disponível.

A Tabela 4 exemplifica as tabelas resumo das descrições dos serviços de cada categoria.

Tabela 4 - Exemplo de tabela resumo dos serviços de cada categoria

ID	NOME	API	PROTOCOLO API	FORMATO DOS DADOS	ACESSO À API	DOCUMENTAÇÃO DA API
RS_01	Rate360	Não	N/A	N/A	N/A	N/A
RS_02	Revcaster	Sim	N/D	N/D	Subscritores.	Não
RS_03	BookLogic	Não	N/A	N/A	N/A	N/A
RS_04	OTAInsight	Sim	N/D	N/D	Subscritores	Sim
RS_05	RateChecker by WuBook	Sim	N/D	XML/JSON	Subscritores	Sim

As restantes tabelas podem ser encontradas no Anexo 1 [2] deste relatório.

3.5. Análise e comparação de funcionalidades fornecidas pelas APIs dos serviços de cada categoria

Aos serviços encontrados que disponibilizam a documentação da sua API, foi feito um levantamento de funcionalidades fornecidas através da mesma. Nos casos em que mais do

que um serviço dentro da mesma categoria disponibiliza a documentação da API, foi feito também um estudo comparativo entre serviços em que se procurou identificar quais as funções transversais aos vários serviços da mesma categoria. Os resultados deste estudo são relevantes para o levantamento de requisitos do sistema que será desenvolvido e também para a selecção dos serviços a integrar.

Este foi um estudo exaustivo em que se percorreram 70 (setenta) documentações de APIs disponibilizadas publicamente pelos serviços. As funcionalidades constantes nas documentações (985) foram registadas em tabelas que se encontram no Anexo 1 [2] deste relatório.

As funcionalidades foram agregadas em módulos e cada tabela apresenta as funcionalidades de cada módulo. Nas células da primeira coluna constam os nomes das funcionalidades e as colunas seguintes correspondem aos serviços SaaS cuja API foi analisada. Caso o serviço forneça a funcionalidade, a sua célula correspondente contém o símbolo "X" ou, caso forneça a funcionalidade sob um nome diferente, o nome sob o qual a fornece. A Tabela 5, retirada do Anexo 1 [2], exemplifica a comparação de funcionalidades efectuada.

Tabela 5 - Exemplo de tabela comparativa de funcionalidades fornecidas pelas APIs dos serviços

MÓDULO PRINCIPAL									
FUNCIONALIDADES		CHANNEL MANAGER							
DESIGNAÇÃO	NOTAS	HOTELSPIDER	HOTELRUNNER	GUESTCENTRIC	SITEMINDER	WOODOO BY WUBOOK	CHANNEL RUSH	CULTSWITCH	MYALOCATTOR
Actualizar inventário de hotel	P.e. nº de quartos disponíveis.	Updating inventories	-	OTA_HotelAvailNotifRQ	OTA_ HotelAvail Notif RQ	-	Allocation Update Request Message	OTA_HotelInvCountNotif RQ	-
Actualizar restrições de estadia num hotel	P.e. mínimo de dias de estadia.	Updating stay restrictions	-	OTA_HotelAvailNotifRQ	OTA_ HotelAvail Notif RQ	-	Restrictions Update Request Message		-
Consultar inventório de um hotel	P.e. quartos disponíveis.	Retrieving inventories	-	-	-	-	-	OTA_ HotelAvailRS/ RQ	-
Consultar dados de uma transacção	P.e. o pedido de actualização de atributos de um quarto falhou.	-	Get Transaction Details	-	-	-	-		-
Enviar dados para um canal específico	-	-	-	-	-	-	Send Data To Specific Channels Only		-
Inserir dados de um hotel	Permite definir as propriedades básicas de um hotel, p.e. nome.	-	-	-	-	-	-	OTA_Hotel DescriptiveContentNotifRQ/RS	-

3.6. Conclusões

A Tabela 6 mostra alguns resultados gerais da pesquisa e análise realizada: quantas ferramentas foram analisadas, de que tipo, quantas disponibilizam API e alguma informação relativa ao acesso à API.

Tabela 6 - Resultados gerais da identificação de serviços SaaS

TIPO DE FERRAMENTA	NÚMERO DE SERVIÇOS ANALISADOS	NÚMERO DE SERVIÇOS COM API	ACESSO À API	
			CLIENTES	PARCEIROS
PMS	24	8	3	0
Channel Manager	20	15	6	3
Revenue Management	10	3	3	0
Reputation Manager	10	6	6	2
Rate Shopper	5	2	2	0
CRM System	7	3	2	2
Booking Engine	8	7	6	0
Facturação	9	10	6	0
POS	11	10	3	5
Aplicações móveis	3	0	0	0
TPA Virtual	5	4	N/D	N/D
Software de Manutenção	2	1	1	1
Gestão de Recursos Humanos	4	1	1	0
Gestão Financeira	1	1	1	1
TOTAL	**120**	**70**	**40**	**14**

A pesquisa permitiu concluir que há tipos de serviços SaaS em que é mais comum a disponibilização de API. Estes tipos de serviços são:

- Sistemas de Facturação e pagamentos: 9 em 10 possuem API para acesso.
- Sistemas POS: 10 em 11 possuem API para acesso.
- Sistemas de Booking Engine: 7 em 8 possuem API para acesso.
- Sistemas de Reputation Manager: 6 em 10 possuem API para acesso.
- Sistemas Channel Manager: 15 em 20 possuem API para acesso.
- TPA Virtual e Serviços de pagamento: 4 em 5 possuem API para acesso.

Permitiu também concluir que a maioria dos serviços disponibiliza API mediante subscrição ou compra do serviço ou produto. Há também uma parte significativa, 14 serviços, em que o acesso a API é apenas para uso de parceiros.

As conclusões retiradas da análise de comparação de funcionalidades fornecidas pelas APIs dos serviços de cada categoria podem ser encontradas no Anexo 1 [2] deste relatório. Embora muito úteis para as fases seguintes do projecto e estágio, considerou-se que sendo tão específicas dos serviços analisados não se enquadram no contexto deste relatório e portanto não são aqui apresentadas.

Capítulo 4
Identificação de macro-requisitos e casos de uso

Como referido na introdução deste relatório, o objectivo deste estágio é a realização de uma prova de conceito que demonstre a validade e viabilidade dos componentes de subscrição e cancelamento de serviços *SaaS* na plataforma *Hotelcracy Apps*.

No seguimento da investigação apresentada no Capítulo 2, a tarefa que se impõe é a identificação dos macro-requisitos destes dois componentes. Tendo em conta a duração do projecto (33 meses) e a fase inicial do mesmo em que este estágio se insere, prevê-se que os requisitos vão ter tendência a mudar e, portanto, optou-se por utilizar os artefactos dos ciclos de vida *agile* que servem para levantamento de requisitos: *user stories*. Considerou-se também o facto de as *user stories* serem artefactos de levantamento de requisitos de muito alto nível[33, Sec. 1] e por isso se adequarem à identificação de macro-requisitos.

A primeira secção deste capítulo é dedicada às *user stories* escritas e encontra-se estritamente relacionada com os componentes de subscrição e cancelamento em que estágio se foca. A segunda descreve a identificação e análise dos casos de uso – funcionalidades que serão permitidas pela plataforma através da interacção com os serviços SaaS das categorias de facturação e *Channel Manager* – sendo esta uma tarefa com um objectivo mais amplo de contribuição para o projecto *Hotelcracy Apps,* à semelhança da análise do estado da arte realizada.

4.1. *User stories*

As *user stories* fazem parte de uma abordagem *agile* que ajuda a mudar o foco da escrita de requisitos para a conversa sobre os requisitos. Para tal, uma *user story* é uma descrição pequena e simples de uma funcionalidade do ponto de vista da pessoa que pretende essa funcionalidade do sistema, normalmente um utilizador ou cliente [34].

No seu livro User Stories Applied [35], Mike Cohn propõe uma abordagem formal para a escrita de *user stories*. O formato proposto e que será utilizado na escrita de *user stories* para os componentes de subscrição e cancelamento de serviços é o seguinte:

Enquanto (tipo de utilizador/papel) quero (algum objectivo) para (finalidade).

Este formato foi escolhido para a escrita das *user stories* que serão apresentadas de seguida, pois ajuda a forçar que a escrita da *user story* responda às três seguintes questões [36]: para quem se vai desenvolver a funcionalidade, quem é o utilizador?; o que é que se vai desenvolver, qual é a intenção?; porque é que se vai desenvolver, que valor traz para o utilizador?

***User story 1* - Subscrever serviço SaaS**

Enquanto hoteleiro, quero subscrever um novo serviço SaaS e que ele fique automaticamente integrado com os restantes serviços subscritos, para que o possa passar a utilizar nas operações do meu negócio sem ser necessário passar pelo período de integração do novo serviço com os restantes.

***User story 2* – Cancelar serviços SaaS**

Enquanto hoteleiro, quero cancelar a subscrição de um serviço SaaS para que ele deixe de fazer parte dos serviços utilizados no meu negócio e para que deixem de me ser cobrados os custos associados à utilização do serviço.

4.2. Identificação de casos de uso

Após terem sido selecionadas duas categorias de serviços, *Channel Manager* e facturação, realizou-se a identificação das funcionalidades principais destas duas categorias de serviços com base no estudo comparativo de APIs referido no Capítulo 3. Considerou-se que as funcionalidades principais são as que são implementadas por um maior número de serviços SaaS diferentes e foram validadas pela empresa líder do projecto – Hotelcracy Software, Lda – que conhece o negócio.

Tendo sido identificadas as funcionalidades principais dos serviços, procedeu-se à análise e especificação de cada funcionalidade, de forma a perceber quais os requisitos a que a plataforma *Hotelcracy Apps* tem que responder para utilizar as funcionalidades nos diferentes serviços.

Nesta análise, além de se identificar o método *HyperText Transfer Protocol* (HTTP) e o formato do corpo da mensagem requeridos pelo serviço, listam-se e normalizam-se os atributos presentes no corpo da mensagem da funcionalidade para cada serviço. Para isso, optou-se por utilizar uma tabela como a Tabela 7 em que as linhas correspondem a nomes de atributos normalizados e as colunas aos serviços que fornecem a funcionalidade. Nas células de intersecção da matriz, assinala-se com um "X" caso o serviço utilize o mesmo nome de atributo ou, no caso de utilizar um diferente, insere-se o nome do atributo para o serviço. Se a célula estiver vazia, significa que o serviço não requer esse atributo para realizar a funcionalidade.

Tabela 7 - Tabela exemplo da análise e especificação de APIs

ATRIBUTOS	HOTEL SPIDER	GUEST CENTRIC	CULT SWITCH	SITEMINDER	CHANNEL RUSH	HOTEL RUNNER
OTA_HotelInvCountNotifRQ (Object)	X	OTA_AvailNotifRQ	X	OTA_AvailNotifRQ	X	
Version (String)	X	X	X	X		
TimeStamp (Date format: yyyy-mm-ddThh:ii:ss)	X	X	X	X		
Target (String)	X		X			
EchoToken (String)				X		
POS (Object)	X					
Source (Object)	X					
RequestorID (Object)	X					
ID (String)	X					hr_id
MessagePassword (String)	X					token
ID_Context (String)	X					

CompanyName (String)	X					
BookingChannelType (Integer)	X					
Inventories (Object)	X	**A**vailStatusMessages	X	AvailStatusMessages	X	
HotelCode (String)	X	X		X		
HotelName (String)	X					
Inventory (Collection[**Inventory**])	X		X		X	
StatusApplicationControl (Object)	X	X	X	X	X	
Start (Date format: yyyy-mm-dd)	X	X	X	X	X	Start_date
End (Date format: yyyy-mm-dd)	X	X	X	X	X	End_date
InvCode (String)	X	X	X	InvTypeCode	InvTypeCode	Room_code
RatePlanCode (String)		X		X		
IsRoom (Boolean)			X			
Mon (Boolean)			X		X	
Tue (Boolean)			X		X	
Wed (Boolean)			X		X	
Thur (Boolean)			X		X	
Fri (Boolean)			X		X	
Sat (Boolean)			X		X	
Sun (Boolean)			X		X	
InvCounts (Collection[**InvCount**])	X		X		X	
InvCount (Object)	X	AvailStatusMessage	X	AvailStatusMessage	X	
Count (Integer)	X	BookingLimit	X	BookingLimit	X	
CountType (Integer)	X		X			
ActionType (String)	X				X	

Esta análise encontra-se no Anexo 2 [3] e Anexo 3 [4] deste relatório que correspondem à identificação e análise de casos de uso da plataforma *Hotelcracy Apps* para os serviços *Channel Manager* e de facturação, respectivamente.

Capítulo 5
Processos de subscrição e cancelamento de serviços

Depois de identificados os macro-requisitos dos componentes de subscrição e cancelamento de serviços, a fase seguinte foi detalhar os processos destes componentes, ou seja, detalhar as *user stories* escritas. Para esta tarefa recorreu-se ao *Business Process Model and Notation* (BPMN), que é uma notação para descrever e modelar processos de negócio. Esta foi a notação considerada mais adequada, pois a subscrição e cancelamento de serviços são processos de negócio entre o utilizador da plataforma, o *Hotelcracy Apps* e os próprios serviços que o utilizador pretende subscrever e cancelar.

5.1. *Business Process Model and Notation* (BPMN)

Business Process Model and Notation (BPMN) é uma notação gráfica que descreve os passos de um processo de negócio. Esta notação foi especificamente desenhada para coordenar a sequência de processos e mensagens que são transmitidas entre diferentes participantes do processo num conjunto de actividades relacionadas [37]. Partindo desta definição, a justificação para a utilização de uma notação de modelação de processos de negócio é relativamente directa. Vejamos, como foi referido em cima, os processos de subscrição e cancelamento envolvem pelo menos três participantes: o utilizador, o *Hotelcracy Apps* e o serviço que se pretende subscrever ou cancelar. Estre três participantes trocam mensagens entre si e realizam actividades que se relacionam para alcançar um fim comum – a subscrição ou cancelamento de determinado serviço.

Apesar de existir um largo espectro de notações para modelar processos de negócio, a notação BPMN é particularmente adequada às características dos processos de negócio que serão apresentados. Uma das características é a coexistência de tarefas manuais e automáticas sendo que a notação BPMN permite representar estes diferentes tipos de tarefas. Além disso, como foi desenhada para poder ser convertida em *Business Process Execution Language* (BPEL) – linguagem que permite a execução de tarefas automáticas presentes num processo de negócio, por exemplo, invocar um *web service* [38]– a opção por esta notação deixa em aberto a utilizaçao de motores de execução e orquestração de processos de negócio, tais como jBPM ou Bizagi.

Uma outra notação para modelar processos de negócio com um nível de popularidade semelhante à BPMN é a que é designada por *Event Driven Process Chain* (EPC). Segundo Marco Fargnoli, esta é uma notação mais vocacionada para capturar diferentes elementos dos processos [39] – resultados do processo, riscos, problemas, *Key Performance Indicators* (KPIs) etc. – que não fazem sentido numa fase de construção de processos de subscrição e cancelamento de serviços. O autor refere também que BPMN é mesmo a notação mais ajustada a qualquer empresa do sector das TIC pois um dos seus focos principais é automatização dos processos de negócio [39].

Sustentada a opção por BPMN, importa também esclarecer que os elementos BPMN que foram utilizados para modelar os vários processos foram os estritamente necessários para uma primeira descrição das actividades e interacções que ocorrem nesses mesmos processos.

5.2. Processos de subscrição e cancelamento de serviços *Channel Manager* e de Facturação

Dos serviços *Channel Manager* e de facturação que foram identificados e analisados (Anexos 1, 2 e 3), seleccionaram-se aqueles que dispõem de API – os serviços que não dispõem de uma *interface* de integração, não são aplicáveis a uma plataforma integradora de serviços - e seguidamente procedeu-se à definição dos processos de baixo nível que têm em conta as especificidades de cada serviço. Partindo destes processos, chegou-se depois a uma definição genérica dos processos de subscrição e cancelamento de serviços. Os serviços selecionados constam da Tabela 8.

Tabela 8 - Serviços seleccionados para a definição de processos de subscrição e cancelamento de serviços

CATEGORIA	SERVIÇOS SELECCIONADOS
Channel Manager	HotelRunner; HotelSpider; SiteMinder; GuestCentric; ChannelRUSH; MyAllocator;
Facturação	InvoiceXpress; IncoiceOcean; HostBill; ZohoInvoice; Debitoor; Wave;

As secções seguintes visam apresentar os processos que foram desenhados e o caminho percorrido para os definir.

Processo de subscrição de um serviço *Channel Manager*

Para possibilitar a definição do processo de subscrição de um serviço *Channel Manager*, realizou-se uma subscrição de teste para cada um dos serviços selecionados e registaram-se os passos e pontos considerados relevantes para concluir cada subscrição. A Tabela 9 apresenta os resultados desta tarefa.

Tabela 9 - Registos das subscrições de teste - Channel Manager

NOME	REGISTOS EFECTUADOS
HotelRunner	Criação manual de nova subscrição gratuita. Após a subscrição, o serviço permite – manualmente - conectar uma

NOME	REGISTOS EFECTUADOS
	aplicação ao serviço a fim de esta poder utilizar a sua API.
HotelSpider	Criação manual de nova subscrição. A subscrição só é válida após a recepção, por parte do serviço, de um contrato assinado pelo utilizador.
SiteMinder	O serviço requer a existência de uma parceria com o *Hotelcracy* para permitir a utilização da API do serviço. Quando um novo utilizador pretende utilizar o serviço através do *Hotelcracy,* o *Hotelcracy* tem que efectuar um pedido de conexão de uma nova propriedade/hotel que corresponde ao novo utilizador.
GuestCentric	O serviço requer a existência de uma parceria com o *Hotelcracy* para permitir a utilização da API do serviço. Quando um novo utilizador pretende utilizar o serviço através do *Hotelcracy,* o *Hotelcracy* tem que efectuar um pedido de conexão de uma nova propriedade/hotel que corresponde ao novo utilizador.
ChannelRUSH	O serviço requer a existência de uma parceria com o *Hotelcracy* para permitir a utilização da API do serviço. Quando um novo utilizador pretende utilizar o serviço através do *Hotelcracy,* o *Hotelcracy* tem que efectuar manualmente o registo de uma nova propriedade/hotel que corresponde ao novo utilizador.
MyAllocator	O serviço requer a existência de uma parceria com o *Hotelcracy* para permitir a utilização da API do serviço. O serviço permite a criação de novas contas de cliente através da API.

Partindo dos registos apresentados na Tabela 9, procurou-se desenhar um processo que abrangesse todos os passos e decisões necessárias à subscrição dos serviços *Channel Manager* selecionados.

De forma a guiar o leitor na visualização e compreensão do processo de subscrição de um serviço *Channel Manager,* ilustrado na Figura 6, apresenta-se de seguida uma descrição do processo.

Descrição

O processo inicia-se quando um pedido de subscrição de um serviço *Channel Manager* é recebido pelo sistema. Após a verificação de que serviço o utilizador pretende subscrever, ocorre a primeira decisão: o serviço *Channel Manager* é parceiro do *Hotelcracy*? Se não, a actividade seguinte é a criação manual de uma conta no serviço por um utilizador do *back office*. Se sim, é necessário tomar uma nova decisão: o *Channel Manager* parceiro do *Hotelcracy* requer um pedido de nova conexão de propriedade? Caso requeira, esse pedido é enviado ao serviço *Channel Manager* em questão, caso contrário a conta para o novo utilizador é criada automaticamente através da API do serviço.

Voltando ao caso em que o serviço *Channel Manager* é parceiro do *Hotelcracy*: após a criação manual de uma conta para o utilizador uma nova questão se coloca, pois há um caso em que o utilizador tem que assinar e enviar um contrato assinado antes de poder começar a utilizar o serviço. Nesse caso, o contrato é enviado ao utilizador e o processo fica parado até que o utilizador devolva o contrato assinado que seguidamente é enviado ao serviço *Channel Manager* correspondente. No caso de não ser necessário um contrato assinado, é feito um registo manual do *Hotelcracy* como aplicação conectada ao serviço.

Em qualquer dos casos descritos nos dois parágrafos anteriores, o processo termina com as seguintes actividades: guardar e associar ao utilizador as credenciais de utilização da API do serviço, adicionar o serviço à lista de serviços subscritos pelo utilizador e notificar o utilizador de que o novo serviço *Channel Manager* foi subscrito com sucesso.

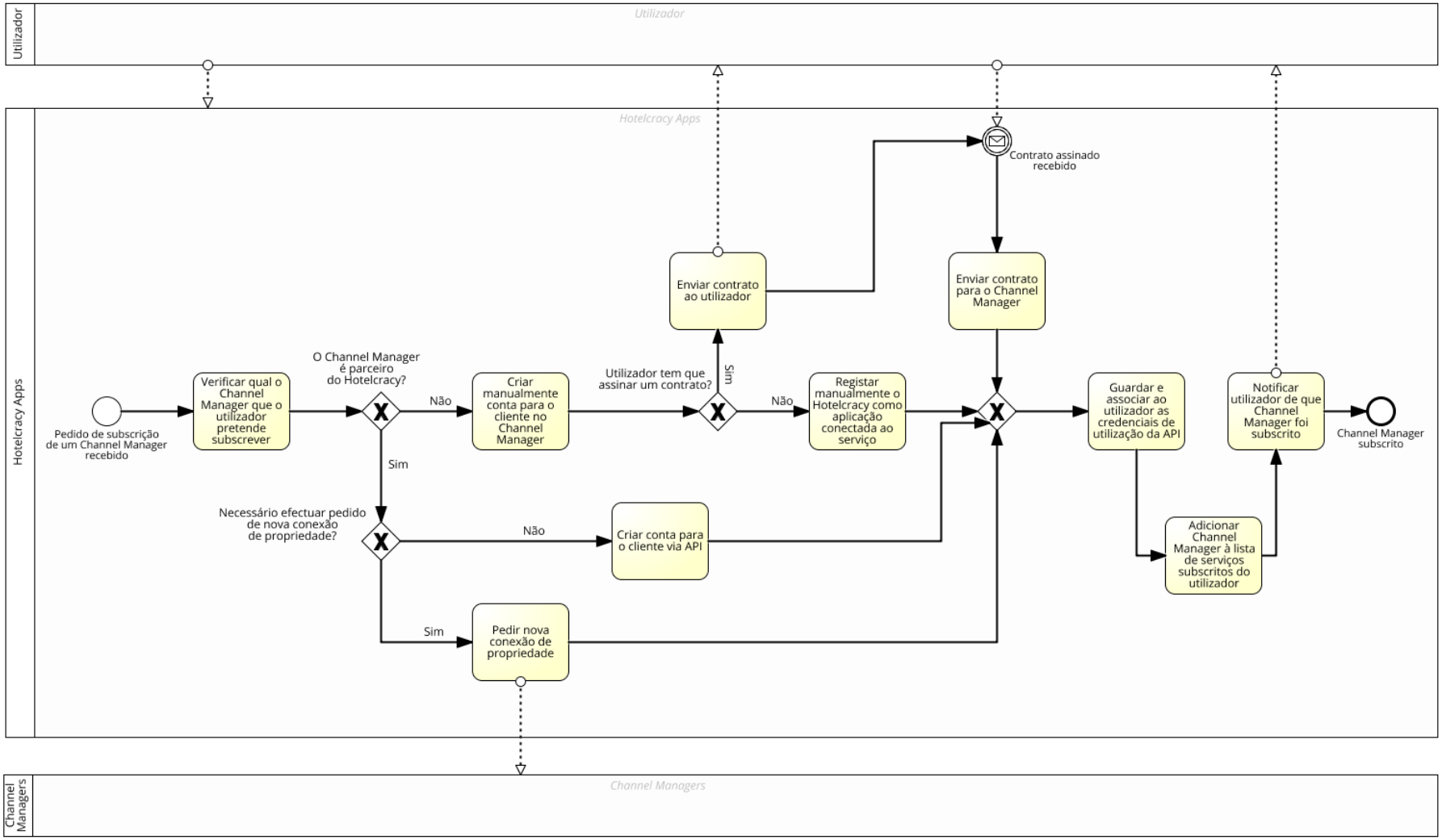

Figura 6 - Processo de subscrição de um serviço Channel Manager

Processo de cancelamento de um serviço *Channel Manager*

Não tendo sido encontradas características específicas diferentes entre os processos de cancelamento de serviços *Channel Manager,* a definição do processo de cancelamento de um serviço *Channel Manager* ficou-se pela descrição de alto nível das actividades e interacções do sistema. O aumento do nível de detalhe deste processo só será possível numa fase mais avançada do projecto. A Figura 7 apresenta o processo de cancelamento de um serviço *Channel Manager*.

De forma a guiar o leitor na visualização e compreensão do processo de cancelamento de um serviço *Channel Manager,* ilustrado na Figura 7, apresenta-se de seguida uma descrição do processo.

Descrição

O processo inicia-se com a recepção do pedido de cancelamento do serviço *Channel Manager*, de seguida o sistema envia o pedido de cancelamento ao serviço *Channel Manager* e aguarda a confirmação do cancelamento por parte do serviço. Após a recepção da confirmação as credenciais de utilização do serviço do utilizador são removidas do sistema, o serviço é removido da lista serviços subscritos do utilizador, e o sistema informa o utilizador de que o serviço foi cancelado com sucesso.

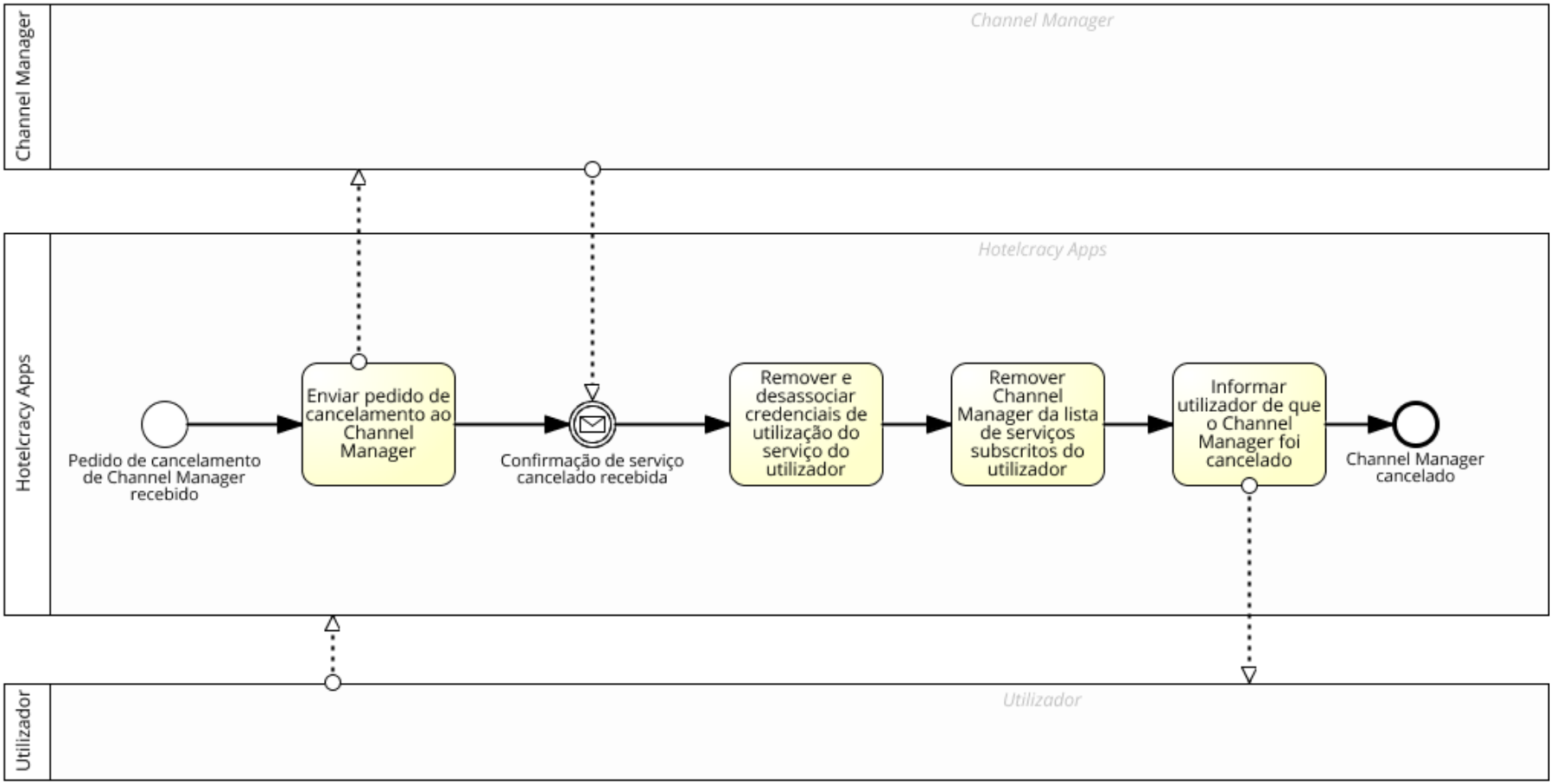

Figura 7 - Processo de cancelamento de um serviço Channel Manager

Processo de subscrição de um serviço de facturação

Para possibilitar a definição do processo de subscrição de um serviço de facturação, realizou-se uma subscrição de teste para cada um dos serviços selecionados e registaram-se os passos e pontos considerados relevantes para concluir cada subscrição. A Tabela 10 apresenta os resultados desta tarefa.

Tabela 10 - Registos das subscrições e teste - Facturação

NOME	REGISTOS EFECTUADOS
InvoiceXpress	Permite a criação de novas contas de utilizador via API, devolve uma nova API *key* e o *Uniform Resource Locator* (URL) de utilização do serviço. Ao fim de 30 (trinta) dias é obrigatório o pagamento que é feito de forma manual (a API não permite pagamentos).
InvoiceOcean	Permite a criação de novas contas de utilizador via API, devolve uma nova API *key* e o *Uniform Resource Locator* (URL) de utilização do serviço. A conta é permanentemente gratuita mas apenas permite a emissão de 3 facturas/mês.
HostBill	Subscrição manual. É necessário efectuar pagamento logo no acto da subscrição.
ZohoInvoice	O serviço disponibiliza uma conta gratuita permanente que permite a emissão de facturas para 5 clientes distintos. O serviço permite a criação de novas organizações na mesma conta, ou seja, diferentes organizações podem utilizar a mesma subscrição do serviço.
Debitoor	Conta gratuita. Subscrição manual. Para o *Hotelcracy* utilizar o serviço em nome do cliente é necessário obter o *access-token* (credencial de utilização do serviço via API) para o cliente.
Wave	Conta gratuita. Subscrição manual. Para o *Hotelcracy* utilizar o serviço em nome do cliente é necessário obter o *access-token* (credencial de utilização do serviço via API) para o cliente.

Partindo dos registos apresentados na Tabela 10, procurou-se desenhar um processo que abrangesse todos os passos e decisões necessárias à subscrição dos serviços de facturação selecionados.

De forma a guiar o leitor na visualização e compreensão do processo de subscrição de um serviço de facturação, ilustrado na Figura 8, apresenta-se de seguida uma descrição do processo.

Descrição

O processo inicia-se quando um pedido de subscrição de um serviço de facturação é recebido pelo sistema. Após a verificação de que serviço o utilizador pretende subscrever, o

processo segue diferentes caminhos consoante o serviço permita adicionar novas organizações à mesma conta de utilização ou não. Se permite, a nova organização é adicionada à conta do *Hotelcracy* e é guardado e associado o identificador da nova organização ao utilizador. Se não permite, é necessário verificar se o serviço disponibiliza a criação de novas contas de utilizador através da API e caso disponibilize, os dados do utilizador (já conhecidos pelo *Hotelcracy*) são enviados para o serviço de facturação via API, que responde com as credenciais do utilizador. Estas credenciais são guardadas e associadas ao utilizador por parte do sistema. Caso o serviço não disponibilize a criação de contas através da API, é necessário efectuar a subscrição manual do serviço e de seguida verificar se o serviço permite a sua utilização gratuita. Se sim, o *acess-token* para o utilizador é pedido e depois guardado e associado ao utilizador. Se não, a página de pagamentos é enviada ao utilizador e o processo aguarda que o pagamento seja efectuado antes de terminar. No caso de a conta gratuita ser temporária, a página de pagamentos é enviada também ao utilizador quando o período gratuito termina.

Todos os caminhos descritos anteriormente encontram-se na actividade que informa o utilizador de que o serviço foi subscrito com sucesso.

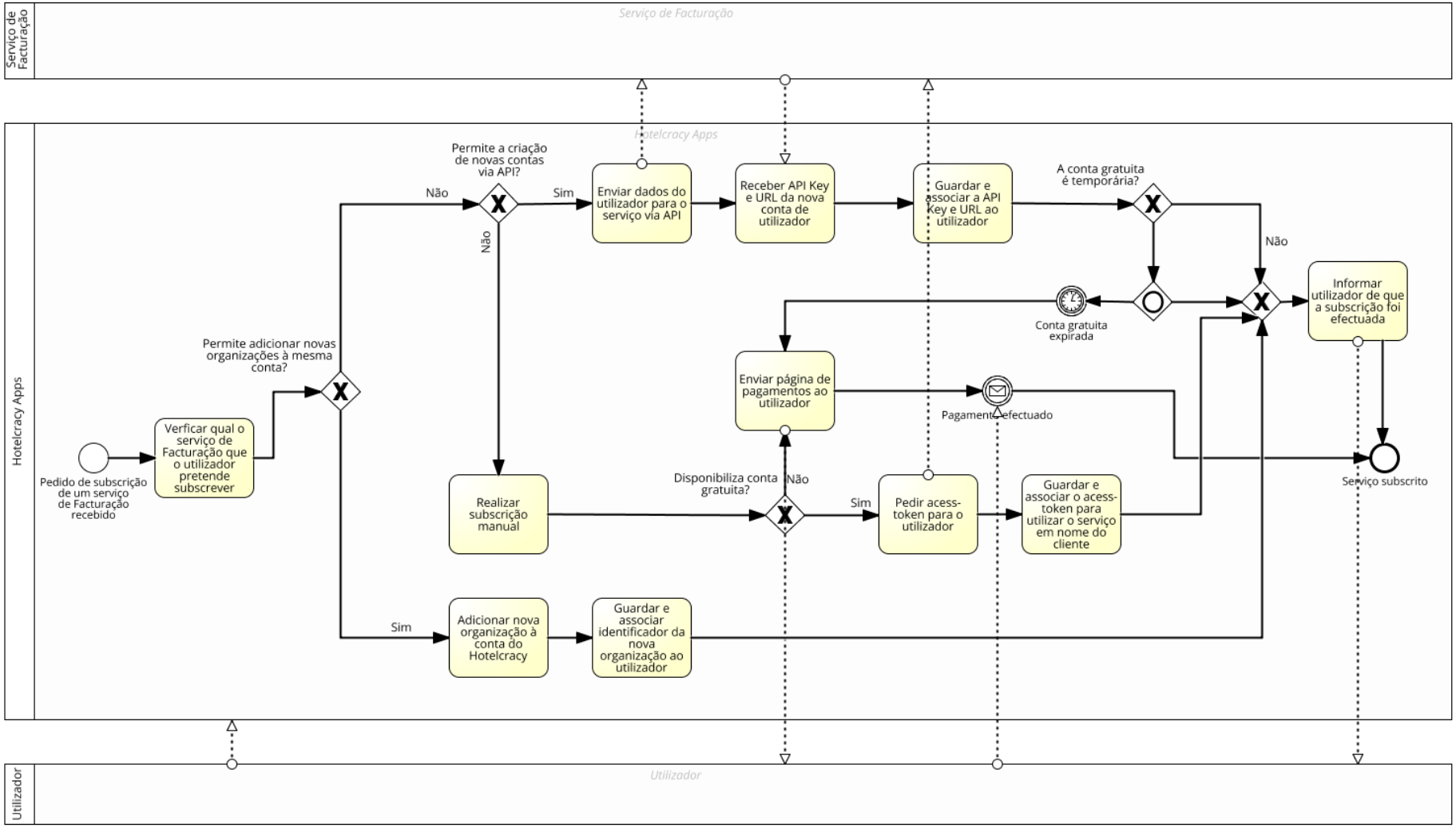

Figura 8 - Processo de subscrição de um serviço de facturação

Processo de cancelamento de um serviço de facturação

Para perceber qual o processo necessário para o cancelamento de um serviço de facturação, efectuou-se o cancelamento dos vários serviços de facturação selecionados. A Figura 9 apresenta o processo a que se chegou.

Descrição

O processo inicia-se com a recepção do pedido de cancelamento de um serviço. Após a verificação de qual o serviço que o utilizador pretende cancelar, é necessário verificar se o serviço permite adicionar várias organizações à mesma conta. Em caso afirmativo, a organização correspondente ao utilizador é removida da conta do *Hotelcracy* no serviço e, em caso negativo, a conta do utilizador no serviço é removida manualmente.

O processo termina com a remoção do serviço da lista de serviços subscritos do utilizador e com o envio de uma notificação ao utilizador informando-o de que o serviço foi cancelado com sucesso.

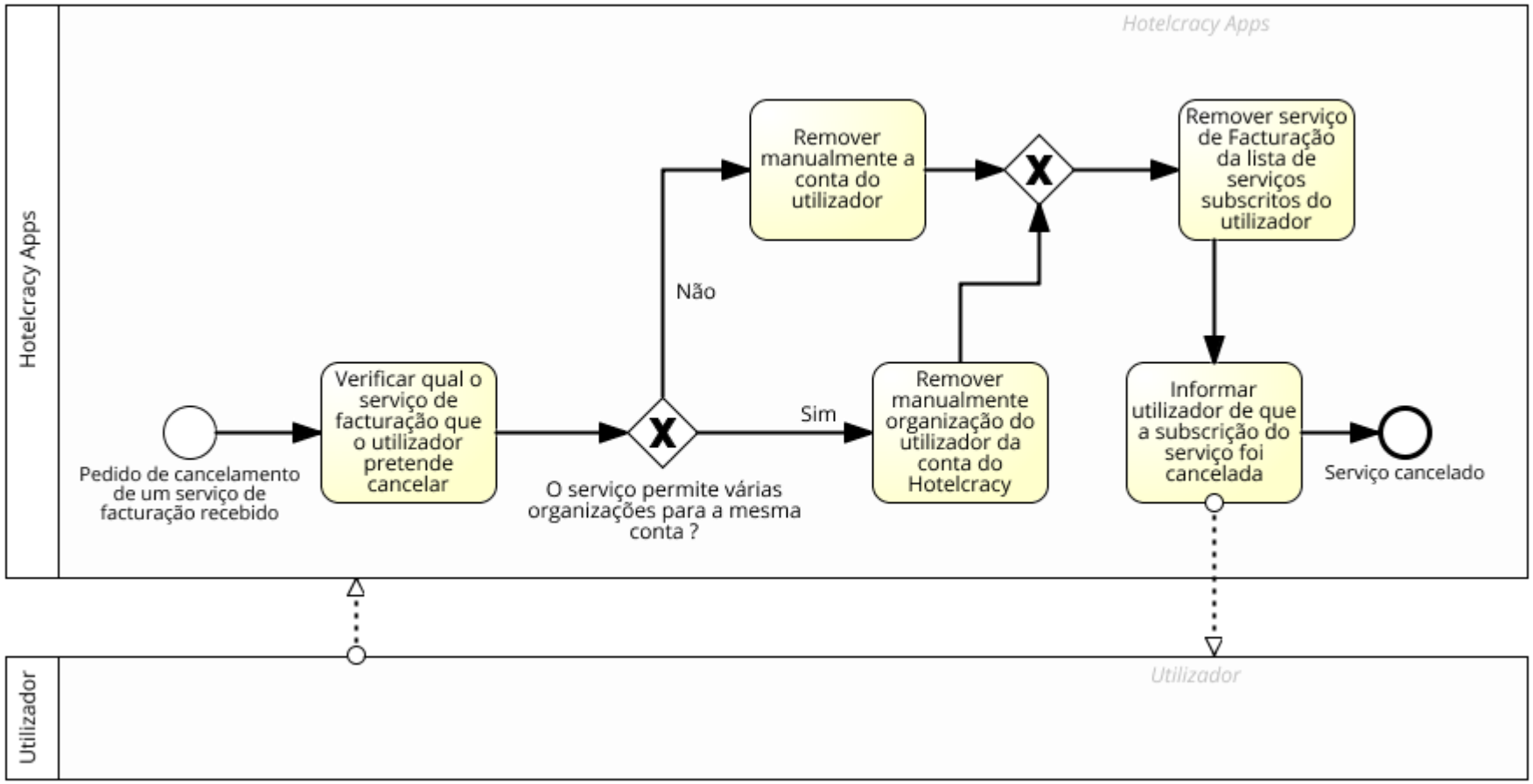

Figura 9 - Processo de cancelamento de um serviço de facturação

5.3. Processos genéricos de subscrição e cancelamento de serviços

O estudo das características individuais dos processos de subscrição de cada serviço *Channel Manager* e facturação permitiu chegar a um processo de mais alto nível que se abstrai das complexidades especificas da subscrição de cada serviço. O foco deste processo são as actividades, automáticas e manuais, que serão realizadas pelo *Hotelcracy* independentemente da categoria de serviços que o utilizador pretende subscrever. A Figura 10 ilustra o processo de alto nível da subscrição de serviços.

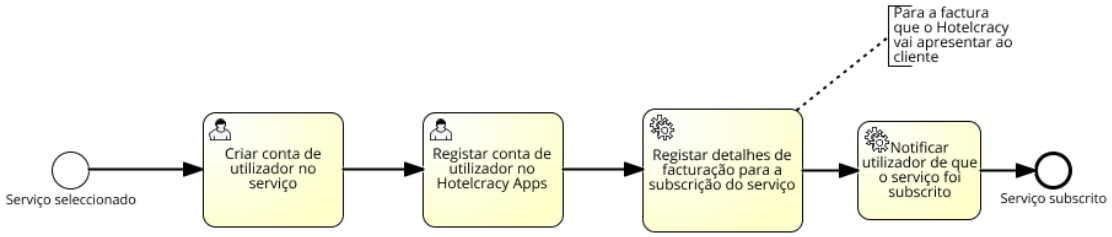

Figura 10- Processo genérico de subscrição de serviços

Tal como o estudo dos processos de subscrição de serviços *Channel Manager* e facturação, o estudo dos processos de cancelamento para essas duas categorias permitiu desenhar um processo genérico para o cancelamento de serviços. Este processo apresenta-se na Figura 11.

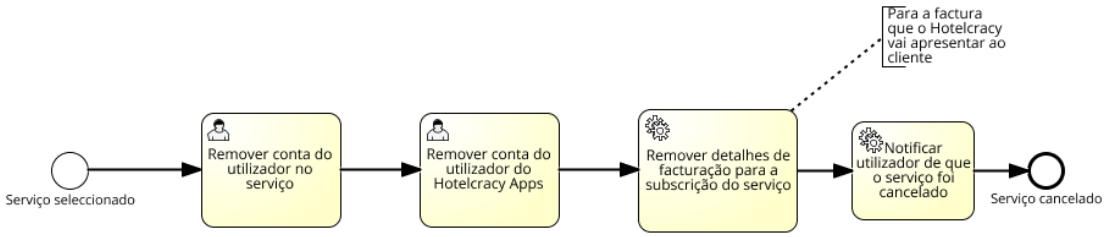

Figura 11 - Processo genérico de cancelamento de serviços

5.4. Conclusões

Neste capítulo apresentou-se uma breve introdução ao BPMN e a justificação para o seu uso na definição detalhada de requisitos e processos de subscrição e cancelamento de serviços. É descrito o método utilizado para essa definição, que consistiu na realização de subscrições e cancelamentos de teste nos vários serviços existentes no mercado que foram selecionados. Das subscrições e cancelamentos de teste, foram registados todos os passos e requisitos e depois definidos processos detalhados que abrangem todos requisitos necessários para a subscrição e cancelamento de todos os serviços.

Por último, foram apresentados processos genéricos que se abstraem dos detalhes dos processos específicos da subscrição e cancelamento de cada serviço. Estes processos genéricos descrevem as tarefas de alto nível, manuais e automáticas, que o sistema terá que realizar para permitir a subscrição e cancelamento de qualquer serviç

Capítulo 6
Modelo de gestão de dados e suporte à interacção com os serviços SaaS

Após a especificação dos processos de subscrição e cancelamento descritos no capítulo anterior deu-se início à concepção tecnológica da prova conceito objecto deste estágio.

Partindo dos processos genéricos de subscrição e cancelamento de serviços e da identificação e análise de casos de uso descrita no Capítulo 4, procurou-se definir um modelo de gestão de dados que suportasse e viabilizasse as interacções entre a plataforma *Hotelcracy Apps* e os serviços SaaS.

Este capítulo encontra-se dividido em duas secções: na primeira descreve-se o trabalho realizado com base na análise e identificação de casos de uso com o objectivo de definir uma *interface* comum a todos os serviços da mesma categoria; na segunda, o foco incide no levantamento e definição dos dados necessários para os três serviços em que se concretiza a prova de conceito: InvoiceXpress, InvoiceOcean e HotelRunner.

6.1. Definição de uma *interface* comum

A finalidade do trabalho descrito nesta secção foi a definição de uma *interface* que possa suportar a utilização das funcionalidades dos serviços SaaS em todos os serviços da mesma categoria. Neste caso, nas duas categorias selecionadas para balizar o estágio: facturação e *Channel Manager*.

Partindo da identificação de funcionalidades descrita no Capítulo 4, fez-se o levantamento dos atributos relevantes para a execução das operações nos serviços SaaS e propôs-se um corpo de mensagem para a operação interna do sistema *Hotecracy Apps*, ou seja, a *interface* comum que abrange todos os atributos obrigatórios de todos os serviços para permitir a utilização de todos os serviços da categoria. As operações descreveram-se utilizando os seguintes campos:

- **Versão da tabela**: identifica a versão da informação contida na tabela da funcionalidade. Sempre que haja alterações nessa informação, deverá ser incrementada a versão. As alterações realizadas, bem como o seu motivo, devem ser registadas no campo "Controlo de versões" da tabela respectiva.
- **Uniform Resource Identifier** (URI): identifica os endereços dos serviços onde se deve efectuar a operação.
- **Formato de corpo de mensagem**: identifica o *Content-Type* suportado pelos serviços na operação.
- **Método HTTP**: identifica o verbo HTTP que deve ser utilizado para executar a operação nos serviços.
- **Autenticação/autorização**: identifica os mecanismos de autenticação e autorização utilizados para comunicação com o serviço.
- **Pré-requisitos**: identifica os requisitos necessários para o correcto funcionamento do pedido no serviço (por exemplo, obter identificador único do hotel no serviço).
- **Headers**: identifica, se aplicável, a informação que deve ser enviada no cabeçalho da mensagem do protocolo HTTP.

- **Parâmetros de pesquisa**: identifica os *query parameters* permitidos para operações que utilizam o método GET do protocolo HTTP.
- **Descrição da funcionalidade interna**: descreve e propõe corpo de mensagem para a operação interna do sistema. Esta descrição será composta pelos seguintes campos:
    - **Descrição de atributos**: descreve os atributos normalizados que serão utilizados na proposta de corpo de mensagem.
    - **Proposta de corpo de mensagem**: proposta de corpo da mensagem do protocolo HTTP para a plataforma a desenvolver. Esta informação é descrita em *Javascript Object Notation* (JSON) para simplificar a visualização da estrutura da mensagem. Em cada atributo enviado no corpo da mensagem, será colocado um valor exemplificativo.
- **Controlo de versões**: descreve as alterações realizadas, ao longo do projecto, à tabela descritiva da operação.

A Tabela 12 exemplifica a descrição de uma das operações da *interface* comum especificada.

Tabela 11 - Tabela exemplificativa da definição de uma interface comum

VERSÃO DA TABELA	1.1
URI	Hotel Spider: https://extranet.hotel-spider.ch/ota/OTA_HotelInvCountNotif/2014A GuestCentric: http://xml.xdev.guestcentric.org/ota2011b/request http://xml.xdev.guestcentric.org/ota2011b/soap CultSwitch: http://cultswitch.cultuzz.de:8080/cultswitch/processOTA SiteMinder: https://cmtpi.siteminder.com/siteconnect/services ChannelRush: https://[SERVERNAME]/InventorySend/Index?providerId=[PROVIDERID] HotelRunner: https://app.hotelrunner.com/api/v2/apps/rooms/~
FORMATO DE CORPO DE MENSAGEM	Hotel Spider: text/xml GuestCentric: text/xml CultSwitch: text/xml SiteMinder: text/xml ChannelRush: text/xml
MÉTODO HTTP	Hotel Spider: POST GuestCentric: Não disponível (N/D). CultSwitch: POST

	SiteMinder: Não Aplicável (N/A). ChannelRush: N/A HotelRunner: PUT
AUTENTICAÇÃO/ AUTORIZAÇÃO	Hotel Spider: credenciais de autenticação no corpo da mensagem. GuestCentric: API *key* e HotelCode no endereço do pedido HTTP, credenciais de autenticação no cabeçalho do envelope SOAP. CultSwitch: credenciais de autenticação no corpo da mensagem. SiteMinder: credenciais de autenticação no cabeçalho do envelope SOAP. ChannelRush: credenciais de autenticação no cabeçalho do envelope SOAP. HotelRunner: API TOKEN e HR_ID no pedido HTTP.
PRÉ-REQUISITOS	HotelSpider: Necessário código de inventário obtido pela funcionalidade CM_FN_7; Necessário código de hotel fornecido pelo serviço. GuestCentric: Necessário código de hotel fornecido pelo serviço. CultSwitch: Necessária certificação do PMS que pretende utilizar a funcionalidade. SiteMinder: Necessário gerar um identificador único para identificar o pedido ao serviço; Necessário código de hotel fornecido pelo serviço. ChannelRush: Necessário obter um *servername* e um *provider id* fornecidos pela ChannelRush: Necessário código de hotel fornecido pelo serviço. HotelRunner: Não tem.
HEADERS	N/A
PARÂMETROS DE PESQUISA	N/A
PROPOSTA DE OPERAÇÃO INTERNA	
DESCRIÇÃO DE ATRIBUTOS	Internamente, o novo sistema deverá utilizar os seguintes atributos: - otaHotelInvCountNotifRQ: objecto principal que contém todos os outros atributos necessários. - version: versão da mensagem. - timeStamp: data e hora da mensagem. - target: target da mensagem (Test/Production). - echoToken: identificador único da mensagem. - pos: objecto necessário para a autenticação no serviço Hotel Spider. - source: objecto necessário para a autenticação no serviço Hotel Spider. - requestorId: objecto necessário para a autenticação no serviço Hotel Spider. - id: identificador de utilizador, corresponde ao *id* do serviço Hotel Spider e ao *hr_id* do serviço Hotel Runner. - messagePassword: palavra-passe do utilizador, corresponde ao *messagePassword* do serviço HotelSpider e ao *token* do serviço HotelRunner. - Inventories: lista de objectos *inventory*. - hotelCode: código do hotel. - hotelName: nome do hotel.

	- Inventory: objecto que contém a informação sobre o inventário. - statusApplicationControl: objecto requerido por um *inventory*. - start: data de inicio da actualização do inventário. - end: data de fim da actualização do inventário. - invCode: código de inventário. - ratePlanCode: código do plano de preços. - mon: especifica se a actualização é válida ou não para a segunda-feira. - tue: especifica se a actualização é válida ou não para a terça-feira. - wed: especifica se a actualização é válida ou não para a quarta-feira. - thur: especifica se a actualização é válida ou não para a quinta-feira. - fri: especifica se a actualização é válida ou não para a sexta-feira. - sat: especifica se a actualização é válida ou não para o sábado. - sun: especifica se a actualização é válida ou não para o domingo. - invCounts: lista de objectos invCount. - invCount: objecto requerido por um *inventory*. - count: número de items de inventário disponíveis. - countType: tipo de inventário. - actionType: tipo de actualização (*Allocation*/*Adjustment*).

PROPOSTA DE CORPO DE MENSAGEM	{
	"otaHotelInvCountNotifRQ":{
	"version":"1.0",
	"timeStamp":"2016-11-29T11:27:00",
	"target":"Test",
	"echoToken":"token",
	"pos": {
	"source": {
	"requestorId": {
	"id":"pmsusername",
	"messagePassword":"pmspassword"
	},
	"bookingChannelType":"5"
	}
	},
	"inventories":{
	"hotelCode":"hotelcode",
	"hotelName":"hotelname",
	"inventory":
	[
	{
	"statusApplicationControl":{
	"start":"2016-11-29",
	"end":"2016-11-30",
	"invCode":"1983",
	"ratePlanCode":"2353",
	"mon":"true",
	"tue":"true",
	"wed":"true",
	"thur":"true",
	"fri":"true",
	"sat":"true",
	"sun":"true"
	},
	"invCounts":
	[
	{"count":"1", "countType":"14",
	"actionType":"Allocation"},
	{"count":"2","countType":"14","actionType":"Allocation"}
	]
	},
	{
	"statusApplicationControl":
	{
	"start":"2016-11-29",
	"end":"2016-11-30",
	"invCode":"1983",
	"ratePlanCode":"2345",
	"mon":"true",

	<pre>"tue":"true",
	"wed":"true",
	"thur":"true",
	"fri":"true",
	"sat":"true",
	"sun":"true"
	},
	"invCounts":
	[
	{"count":"1","countType":"14","actionType":"Allocation"},
	{"count":"2","countType":"14","actionType":"Allocation"}
	]
	}
	]
	}
	}
	
	}

CONTROLO DE VERSÕES	**v1.0** (06/12/2016) - Primeira descrição da operação. **v1.1** (05/30/2017) - Actualizados campos conforme nova versão da documentação da API do serviço Hotel Spider.

Esta análise encontra-se no Anexo 2 [3] e Anexo 3 [4] deste relatório que correspondem à definição de uma *interface* comum para a interacção da plataforma *Hotelcracy Apps* com os serviços das categorias *Channel Manager* e facturação, respectivamente.

6.2. Modelo de dados para a prova de conceito

Depois do trabalho que abrange todos os serviços e que é um valioso contributo do estágio para o projecto, o trabalho que a seguir se descreve é centrado na prova de conceito da subscrição e cancelamento de serviços. Como tal, partiu-se dos processos apresentados no Capítulo 5 (Figura 10 e Figura 11) para se definir um modelo de dados que suporta e viabiliza as interacções do sistema com os serviços SaaS na execução das tarefas constituintes dos processos.

A primeira actividade do processo – "Criar conta de utilizador no serviço" – foi definida como uma tarefa manual no processo genérico porque os serviços que disponibilizam esta funcionalidade de forma automática através das suas APIs também permitem naturalmente a sua subscrição manual. Sendo assim, definindo esta tarefa como manual, consegue-se abranger todos os serviços. Este facto não significa que não se faça uso desta funcionalidade automática nos serviços que a disponibilizam, portanto, a fim de executar esta tarefa automaticamente, foi feito o levantamento dos dados necessários para utilizar a funcionalidade em dois serviços: InvoiceXpress e InvoiceOcean. A Tabela 12 descreve a informação recolhida.

Tabela 12 – Levantamento de dados para a subscrição automática

URI	**InvoiceXpress:** https://app.invoicexpress.com/invoices.xml **InvoiceOcean:** https://YOUR_DOMAIN.invoiceocean.com/invoices.json
FORMATO DE CORPO DE MENSAGEM	**InvoiceXpress:** application/xml **InvoiceOcean:** application/json

MÉTODO HTTP	**InvoiceXpress:** POST **InvoiceOcean:** POST	
AUTENTICAÇÃO/ AUTORIZAÇÃO	**InvoiceXpress:** N/A **InvoiceOcean:** *api_token* no corpo da mensagem.	
HEADERS	**No pedido (cliente):** **InvoiceXpress:** Content-Type: application/xml; charset=utf-8 **InvoiceOcean:** Accept: application/json: Content-Type: application/json	
CORPO DO PEDIDO	**INVOICE XPRESS**	**INVOICE OCEAN**

	<account>
	<organization_name>
	</organization_name>
	<email>
	</email>
	<password>
	</password>
	<terms>
	</terms>
	</account>
		{
		"api_token": "token",
		"account": {
		"prefix": "example",
		},
		"user": {
		"login": "example",
		"email": "example@mail.ex",
		"password": "password"
		}
		}

Concluiu-se, após o levantamento da informação presente na Tabela 12, que o sistema tem que obter e armazenar as suas credenciais próprias para a utilização da funcionalidade de subscrição do serviço InvoiceOcean. Note-se que o mesmo não se passa em relação ao serviço InvoiceXpress visto que não exige nenhuma forma de autenticação para a utilização desta mesma funcionalidade. Optou-se então por escolher um formato de dados de estrutura dinâmica – JSON – para o armazenamento desta informação. Esta escolha deriva da constatação de que diferentes serviços exigem diferentes formatos e campos para autenticação não podendo esta informação estar estruturada de uma forma fixa. No caso do serviço InvoiceOcean, a informação de autenticação que tem que ser obtida e guardada é a dada pela Tabela 13.

Tabela 13 - InvoiceOcean - Dados de autenticação

AUTENTICAÇÃO NO SERVIÇO INVOICEOCEAN

{
"domain": "exemplo",
"api_token": "exemplo"
}

Ainda com o foco na mesma tarefa – "Criar conta de utilizador no serviço" – mas realizada de forma manual, constatou-se que a plataforma *Hotelcracy Apps* precisa de ter conhecimento de quais as credenciais de autenticação que tem que pedir ao utilizador após ele subscrever o serviço por sua conta. O formato escolhido para armazenar esta informação foi também

JSON e o serviço em que esta tarefa manual se concretiza na prova de conceito é o HotelRunner. A Tabela 14 mostra os nomes dos campos que serão pedidos ao utilizador aquando de uma subscrição manual neste serviço.

Tabela 14 - HotelRunner - Dados de autenticação

AUTENTICAÇÃO NO SERVIÇO HOTELRUNNER
{ "names": ["Token", "HR_ID"] }

A tarefa seguinte do processo genérico de subscrição de serviços – "Registar conta de utilizador no Hotelcarcy Apps" – requer a concretização de "conta de utilizador" nos vários serviços que a plataforma vai permitir subscrever e a definição dos dados, e seu formato, constituintes deste registo na plataforma *Hotelcracy Apps*.

Para definir "conta de utilizador" nos serviços que permitem subscrição automática utilizados na prova de conceito – InvoiceXpress e InvoiceOcean – registaram-se as respostas aos pedidos descritos na Tabela 12 e com base nestas estabeleceram-se os campos guardados pela plataforma que correspondem à conta de um utilizador em cada um destes dois serviços. No caso do serviço selecionado em que a subscrição é manual – HotelRunner – o registo que é efectuado nesta tarefa do processo genérico de subscrição não é mais do que a atribuição de valores aos nomes dos campos solicitados ao utilizador na primeira tarefa deste processo (ver Tabela 13 e Tabela 14).

Um exemplo da informação que é registada nesta tarefa para cada um dos serviços seleccionados para a prova de conceito apresenta-se então na Tabela 15.

Tabela 15 - Registo de conta de utilizador nos serviços da prova de conceito

INVOICEXPRESS	INVOICEOCEAN	HOTELRUNNER
{ "id": "1" "name": "exemplo", "email": "ex@mail.ex", "password":"pw", "url": "ex.invx.com", "api_key": "exemplo" }	{ "api_token": "exemplo", "url": "ex.invo.com", "login": "exemplo", "email": "ex@mail.ex", "password":"pw" }	{ "Token": "exemplo", "HR_ID": "1234" }

É de salientar que os dados mais relevantes que são guardados nesta tarefa de "Registar conta de utilizador no Hotelcracy" são as credenciais de utilização da API do serviço para o novo utilizador (por exemplo, "url" e "api_key"), pois são estas que vão permitir ao utilizador interagir com o serviço através da plataforma. A finalidade dos restantes dados é principalmente informar o utilizador dos seus dados de acesso ao serviço directamente na *interface* gráfica disponibilizada pelo mesmo.

Após definidos os dados utilizados e armazenados para o suporte à interacção com os serviços SaaS – mais concretamente, os serviços seleccionados para a presente prova de

conceito – no âmbito da subscrição de serviços, torna-se igualmente importante estipular onde estes mesmos dados serão guardados. Como esta decisão implica a existência de uma especificação arquitectural da plataforma, este assunto será abordado no Capitulo 8 em que se descreve a arquitectura dos componentes para a subscrição e cancelamento de serviços.

Capítulo 7
Análise da *interface* com o utilizador

Os principais objectivos da tarefa do estágio a que se dedica esta secção foram compreender qual a informação que a *interface* gráfica da plataforma deve apresentar ao utilizador e quais as interacções entre o utilizador e a plataforma para a subscrição e cancelamento de serviços.

Para realizar esta análise começou-se por identificar dois contextos distintos de interacção com o sistema: quando o utilizador pretende consultar os serviços SaaS disponíveis, obter informação sobre os mesmos e subscrevê-los (*marketplace*); e quando o utilizador tenciona gerir os serviços subscritos, como por exemplo obter informação sobre determinada subscrição ou cancelar uma subscrição (gestão de serviços).

Identificados estes dois contextos, recorreu-se ao desenho de esboços de ecrã que representam não só a informação que deve ser apresentada ao utilizador em cada um dos contextos mas também as acções que lhe devem ser possibilitadas. Foi através de várias iterações sobre os esboços que se alcançou um nível de detalhe suficiente para que eles pudessem servir de base à fase de especificação da arquitectura dinâmica do sistema (diagramas de sequência) e à fase de implementação da prova de conceito.

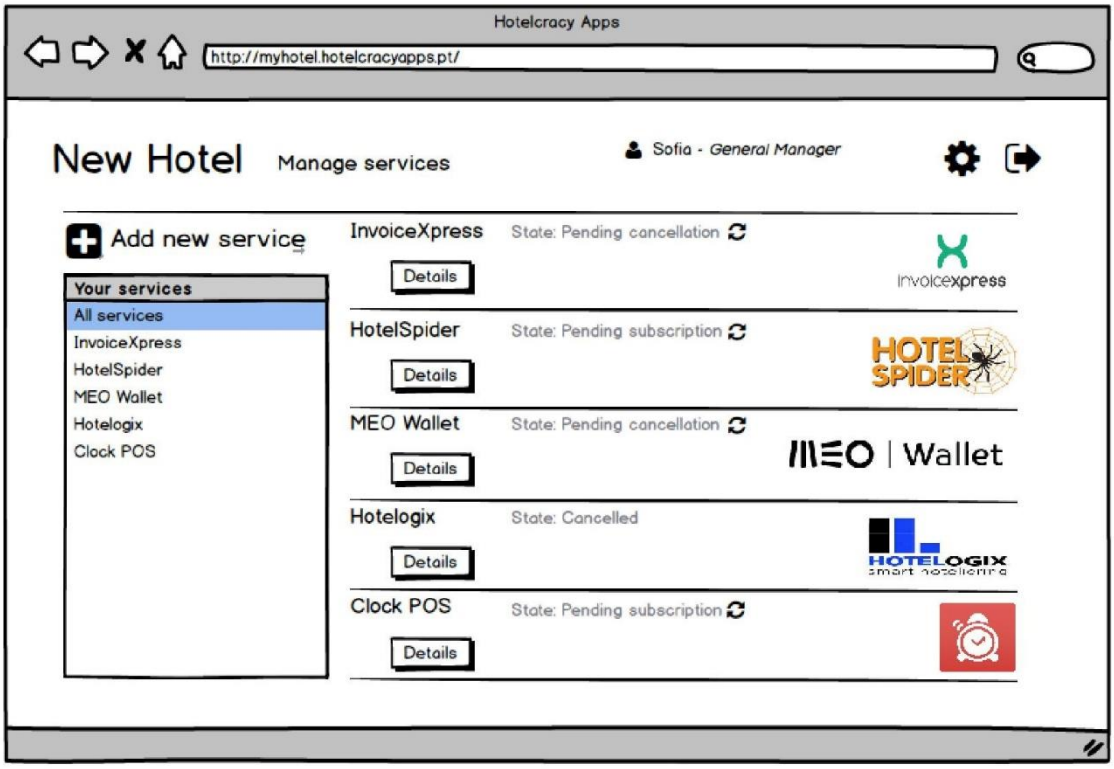

Figura 12- Exemplo de um esboço de ecrã - Gestão de Serviços

É de referir também que o encadeamento entre os vários ecrãs ficou estabelecido nesta fase, retratando a sequência de passos que o utilizador deve seguir para subscrever e cancelar um serviço.

Na Figura 12 apresenta-se um exemplo dos esboços de ecrã desenhados para o contexto de gestão de serviços e na Figura 13 um exemplo dos que foram desenhado para o *marketplace*, podendo ser visualizadas as sequências completas dos esboços no Anexo 4 deste relatório [5].

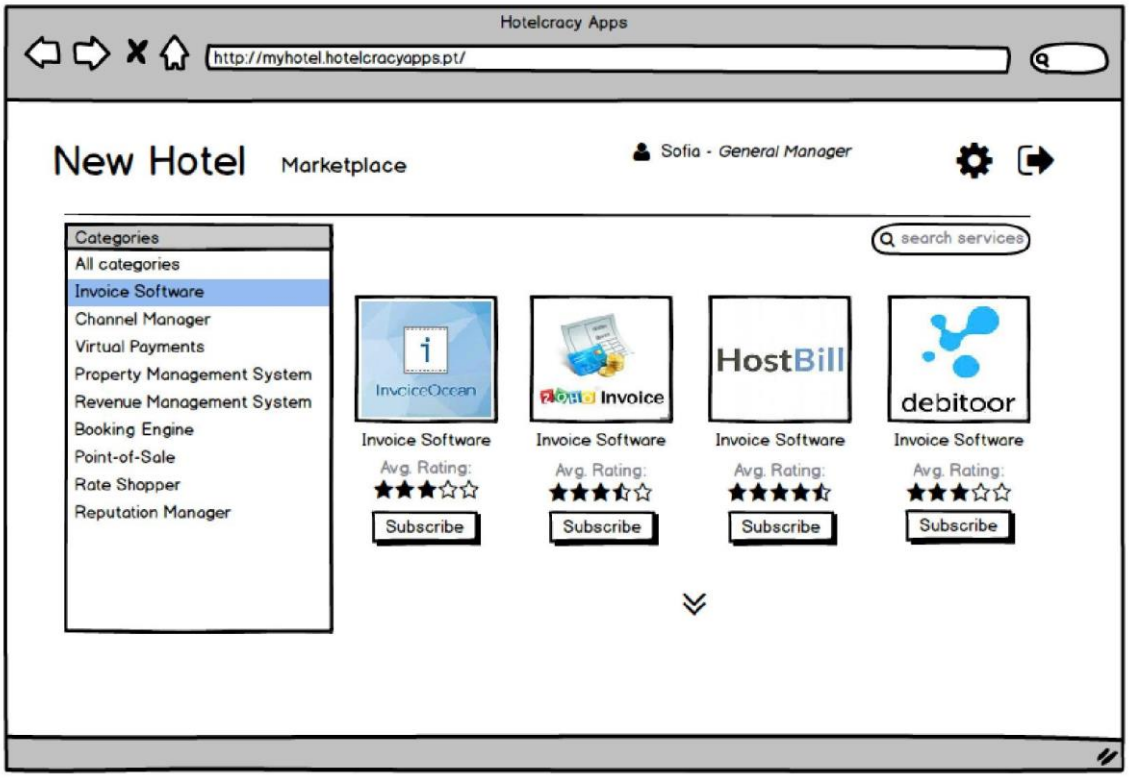

Figura 13 - Exemplo de um esboço de ecrã – Marketplace

Com a finalidade de demonstrar de que forma os esboços de ecrã desenhados serviram de base à implementação do *design* da *interface* de utilizador da prova de conceito apresenta-se também nas Figura 14 e Figura 15 as páginas *web* desenvolvidas correspondentes aos exemplos de esboços de ecrã apresentados em cima.

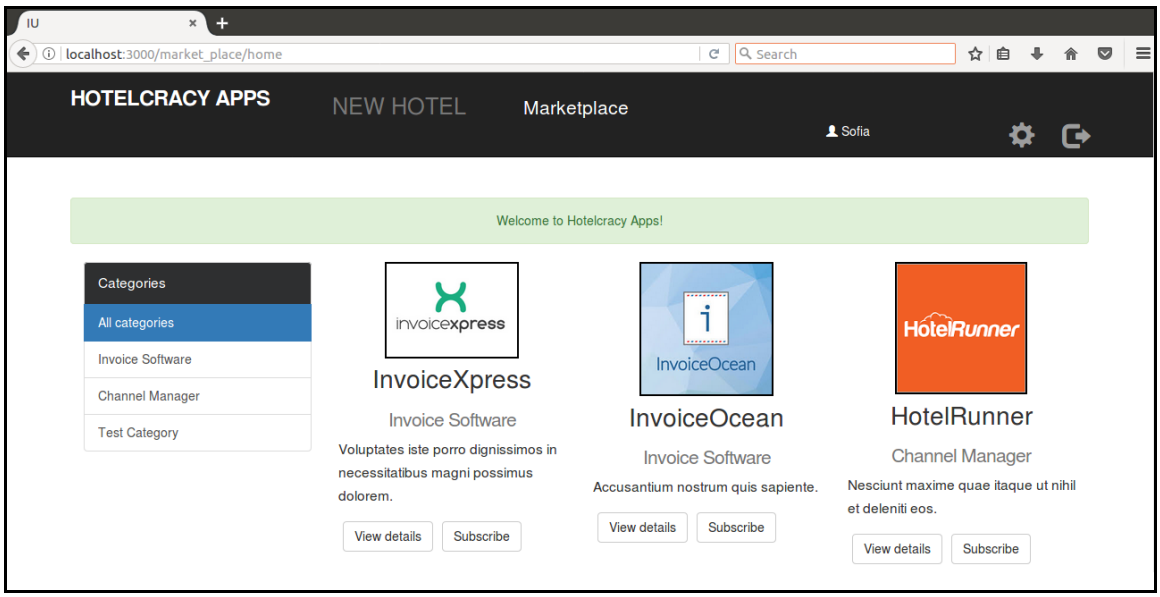

Figura 14 – Exemplo de página web real – Marketplace

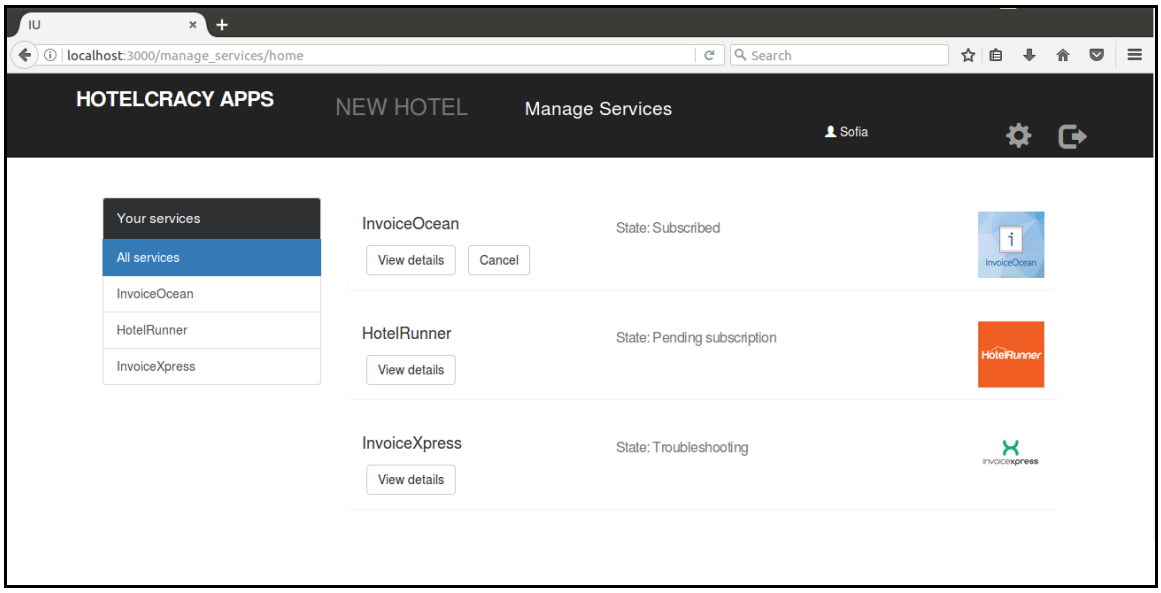

Figura 15 –Exemplo de página web real – Gestão de serviços

Capítulo 8
Arquitectura estática e dinâmica da prova de conceito

Neste capítulo descreve-se a arquitectura da prova de conceito da plataforma *Hotelcracy Apps* que permite viabilizar os processos de subscrição e cancelamento de serviços.

Para definir e apresentar a arquitectura seguiu-se o modelo C4 proposto por Simon Brown no seu livro "*Software Architecture for Developers*"[40]. Este modelo sugere uma forma hierárquica de pensar sobre as estruturas estáticas de um sistema de *software* em termos de *containers*, componentes e classes [41]. Para visualizar esta hierarquia são desenhados diagramas dos vários níveis: diagrama de contexto do sistema*,* diagrama de *Containers,* diagrama de componentes e, opcionalmente, diagrama(s) de classes e/ou outros diagramas UML de baixo nível.

Após a definição dos elementos presentes no modelo das estruturas estáticas do sistema, é sugerido no *C4 Model* que se utilizem esses elementos para criar diagramas adicionais que esclareçam o comportamento do sistema em funcionamento [41]. Na presente especificação arquitectural utilizaram-se diagramas de sequência para ilustrar as interacções entre os vários componentes do sistema com vista à subscrição e cancelamento de serviços SaaS.

O capítulo está organizado em quatro secções distintas: na primeira pretende-se apresentar e descrever o diagrama de mais alto nível do *C4 Model,* o nível 1; na segunda, expor o caminho percorrido para conceber o modelo interno do sistema e o diagrama de nível 2 que o representa; na terceira, descrever e justificar sumariamente a arquitectura dos componentes constituintes do sistema; e, por último, apresentar os diagramas de sequência referidos no parágrafo e secção anteriores, bem como o processo utilizado para a sua concepção.

8.1. Diagrama de contexto – nível 1

O diagrama de contexto é a especificação de nível mais alto da hierarquia proposta pelo *C4 Model*. Este diagrama e a sua definição pretendem ser um ponto de partida para o desenho de um sistema de software, fornecendo uma perspetiva global do sistema e sua relações com o exterior. Para tal, o *System Under Development* (SuD) – neste caso, a plataforma *Hotelcracy Apps* – é representado no diagrama de contexto por uma caixa ao centro rodeada pelos seus utilizadores e outros sistemas com os quais o SuD interage [41].

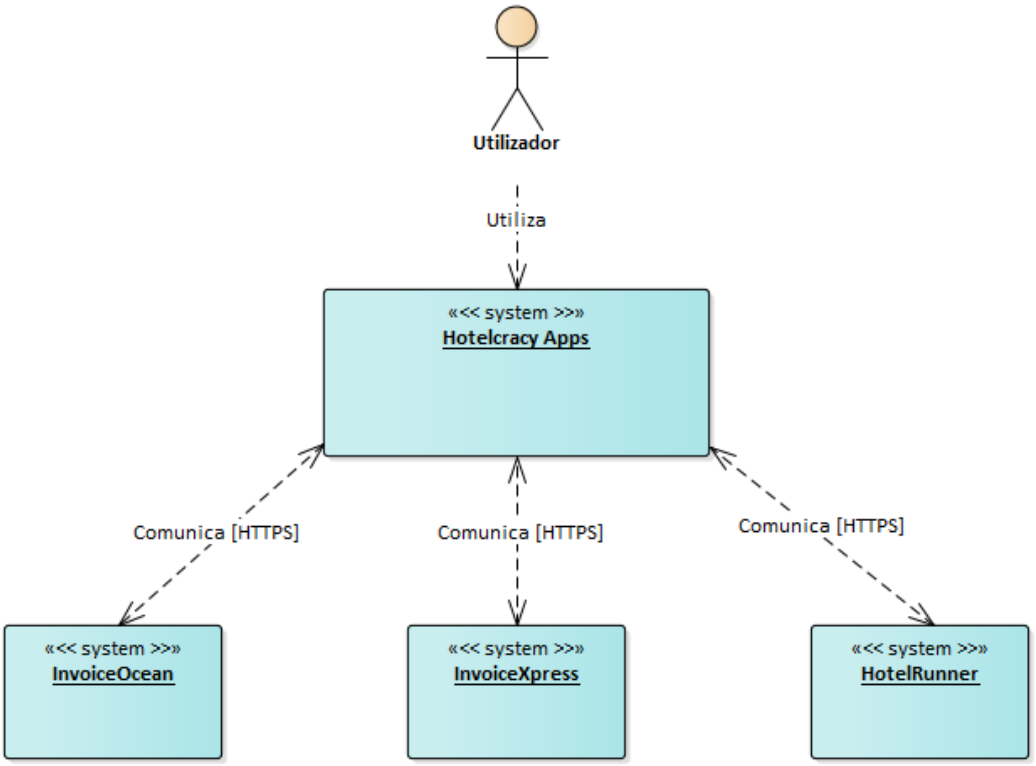

Figura 16 - Diagrama de contexto

No âmbito da prova de conceito objecto deste estágio, o utilizador da plataforma é um utilizador com permissões para navegar no *Marketplace* e para subscrever e cancelar serviços na, e através, da mesma. Como tal, é representado no diagrama (Figura 16) apenas um utilizador genérico sem detalhar o tipo de utilizador e o seu papel perante o sistema. De qualquer modo, no sistema completo de produção existirão diferentes tipos de utilizador com diferentes permissões e papéis [42].

Como referido anteriormente, os serviços SaaS externos em que se vai concretizar a subscrição e cancelamento na prova de conceito desenvolvida são o InvoiceOcean, o InvoiceXpress e o HotelRunner. O diagrama de contexto (Figura 16) apresenta então estes serviços, o tipo de relação com o sistema *Hotelcracy Apps* e a tecnologia utilizada para implementar essa relação: o protocolo de comunicação HTTPS. Nesta fase, a tecnologia já é conhecida em virtude da análise previamente realizada à documentação dos serviços que a plataforma *Hotelcracy Apps* vai integrar.

8.2. Diagrama de componentes – nível 2

Um *container,* no contexto do modelo utilizado para especificar a arquitectura, é essencialmente uma *deployable unit* separável do restante sistema que executa código ou armazena informação. Na presente especificação de arquitectura optou-se por utilizar o termo "componente" para designar o conceito de *container* do *C4 Model.*

O passo seguinte deste modelo é, então, a ilustração dos vários componentes e suas relações de alto nível que constituem o sistema de *software* através de um diagrama de componentes. O seu principal objectivo é mostrar a arquitectura de alto nível do *software* e, através da sua descrição, como as diferentes responsabilidades são distribuídas no interior do sistema.

À semelhança do procedimento utilizado para definir o modelo de dados apresentado no Capítulo 6 deste relatório, partiu-se dos processos genéricos de subscrição e cancelamento de serviços (Figura 10 e Figura 11) para identificar os vários componentes necessários para a realização das tarefas que constituem esses mesmos processos.

Na figura 13, apresenta-se na Figura 17o diagrama de *containers* do sistema onde constam os vários componentes e as relações de alto nível entre eles. Para complementar o diagrama, descreve-se brevemente cada um deles.

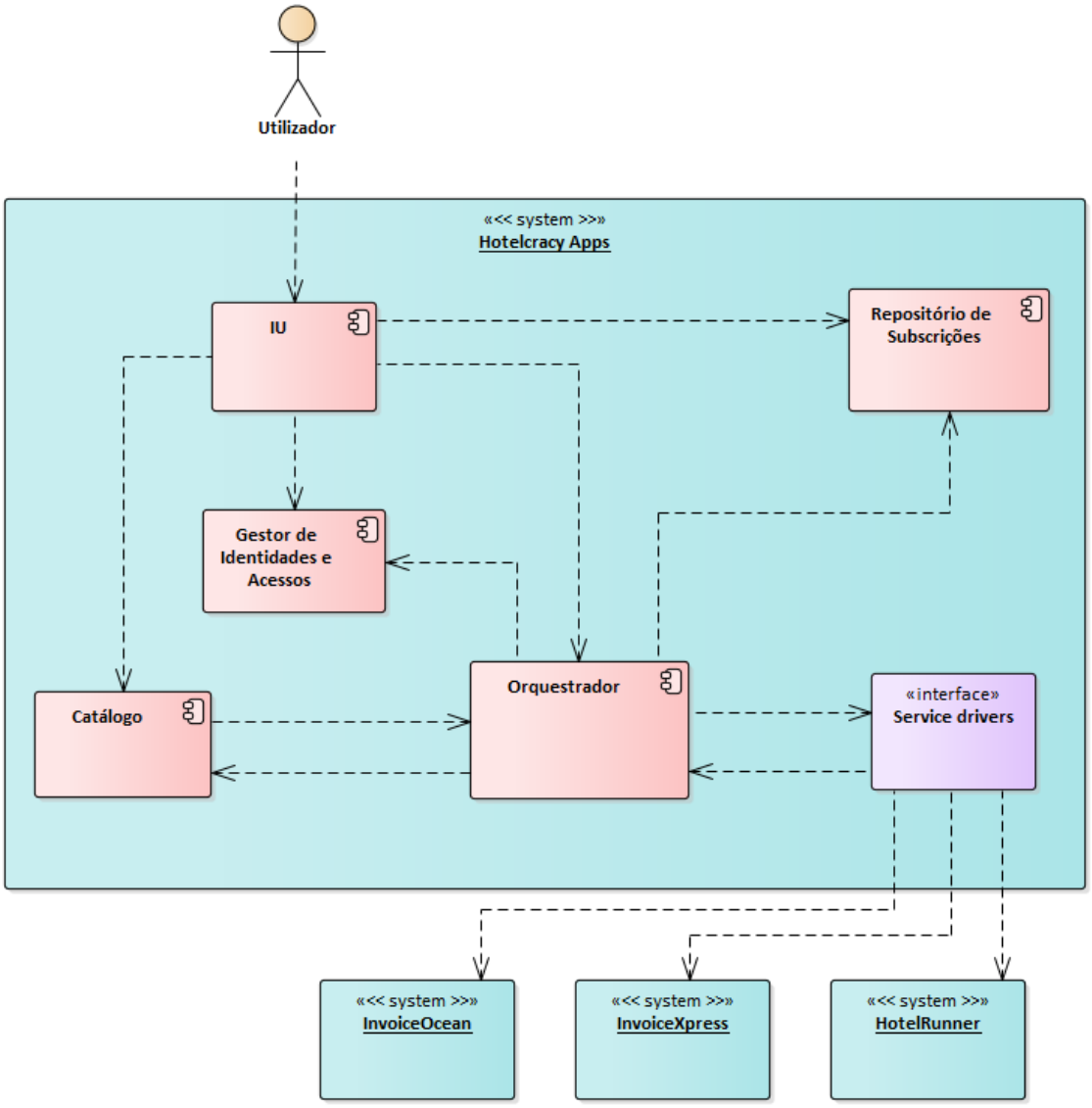

Figura 17 - Diagrama de componentes

Seguidamente apresenta-se uma breve descrição de cada um dos componentes:

- **IU** – Este é o componente que representa a interface com os utilizadores da plataforma *Hotelcracy Apps*.
- **Catálogo** – Este componente armazena e fornece toda a informação relativa aos serviços SaaS disponíveis no sistema *Hotelcracy Apps*.
- **Gestor de identidades e acessos** – Este é o componente responsável pela gestão dos utilizadores e seus acessos (autenticação e autorização).

- **Repositório de Subscrições**: Este é o componente responsável pela gestão das subscrições de serviços dos utilizadores.
- **Service drivers** – Este é o componente que implementa a interface de comunicação para a subscrição, uso e cancelamento de um serviço.
- **Orquestrador** – Este é o componente responsável pela orquestração dos processos do sistema. Comunica com os drivers dos diferentes serviços disponibilizados pela plataforma.

Após ter sido apresentado o diagrama que é resultado final deste estudo, as secções seguintes são dedicadas à descrição de cada componente identificado e das suas responsabilidades no sistema.

Interface de Utilizador (IU)

O evento inicial de ambos os processos – "Serviço selecionado" – implica a existência de um componente que permita ao utilizador selecionar o serviço que se pretende subscrever/cancelar. Esse componente será responsável pela implementação da interface uniforme e centralizada do sistema. Este componente, a que se deu o nome de Interface de Utilizador (IU), permite a recepção das mensagens identificadas previamente nos processos: "Notificar utilizador de que o serviço foi subscrito" e "Notificar utilizador de que o serviço foi cancelado".

Catálogo

Segundo a análise efectuada e os objectivos do projecto em que este estágio se insere, o sistema terá que conter um *Marketplace* que contenha serviços de várias categorias. Este componente irá conter informação técnica e comercial dos serviços para os clientes da *Hotelcracy* (p.e. preços, funcionalidades e críticas) e os dados sobre os serviços necessários para os restantes componentes do sistema, como por exemplo os dados identificados no Capitulo 6 (Tabela 13 e Tabela 14).

Optou-se por chamar a este componente de Catálogo, porque permite a gestão de um conjunto de serviços que serão apresentados aos utilizadores.

Orquestrador

No momento em que o utilizador executa uma acção (subscrever/cancelar) será preciso orquestrar as tarefas necessárias para a correcta execução da operação. Para isso, será necessário desenvolver um componente que execute operações como verificar se o utilizador pode subscrever o serviço, identificar a categoria e o serviço a subscrever/cancelar, mapear o tipo de serviço com o correcto componente de comunicação com o serviço e preparar os dados necessários para esse componente e, por último, notificar a componente UI da realização da operação com sucesso.

Este componente, a que se deu o nome de Orquestrador, faz a orquestração dos processos implementados pelo sistema.

Service drivers

Como referido, os serviços possuem características próprias que terão de ser endereçadas aquando da comunicação com os mesmos. Por esse motivo, é necessário criar um

componente que conheça as especificidades e instruções para a criação de uma nova conta de utilizador no serviço em causa e as execute servindo de *interface* de comunicação do sistema com o serviço. Fica então definida a localização dos dados sobre as credenciais que devem ser pedidas ao utilizador num cenário de subscrição manual identificados no Capítulo 6.

Como um dos objectivos do sistema é a integração de novas soluções que surjam no mercado [7] – extensibilidade – este componente deverá ter uma arquitectura que permita, com facilidade, a inclusão de novos *drivers* para o serviço.

A este componente deu-se o nome de *Service drivers*.

Repositório de subscrições

Voltando aos processos uniformes de subscrição e cancelamento, emerge a necessidade de um componente dedicado à gestão das várias subscrições que um utilizador realizou. Esta gestão passa, por exemplo, por adicionar ou remover um novo serviço à lista de serviços de um utilizador. Este componente, a que se deu o nome de Repositório de Subscrições, armazena dados sensíveis relativos às subscrições (p.e. credenciais de utilização), sendo, portanto, uma componente que merece especial atenção em termos de segurança. Por esta razão, o Repositório de Subscrições é um componente à parte e não um módulo constituinte de outro dos componentes do sistema.

Fica desta forma definida também a localização dos dados relativos a contas de utilizador identificados no Capitulo 6 (Tabela 15).

Gestor de Identidades e Acessos

Apesar de todas tarefas dos processos uniformes de subscrição e cancelamento de serviços já estarem asseguradas pelos vários componentes já enunciados, estes ainda não são suficientes para que se alcance com sucesso os eventos finais dos processos ("Serviços subscrito" e "Serviço cancelado"). Isto porque os processos pressupõem que o utilizador ao selecionar um serviço para subscrever ou cancelar, já se encontra identificado no sistema. Assim sendo, torna-se necessário a criação de um componente que lide com a autenticação e autorização dos utilizadores e que, deste modo, saiba qual é o utilizador que pretende subscrever o serviço – isto é de particular importância, pois o mesmo cliente da plataforma *Hotelcracy Apps* pode ter vários utilizadores com permissões diferentes [42], levantando a necessidade de centralizar a gestão dos vários utilizadores de um cliente.

Este componente, a que se deu o nome de Gestor de Identidades e Acessos, armazena os dados dos utilizadores do sistema e fornece-os aos restantes componentes.

8.3. Arquitectura dos componentes

Uma das restrições tecnológicas impostas pela empresa líder do projecto – *Hotelcracy Software Lda* – foi a utilização da *framework Ruby On Rails* na implementação da plataforma. Esta é uma *framework* fundamentada no padrão arquitectural *Model-View-Controller* (MVC) que organiza a lógica de programação em três camadas principais: a camada *Model* é onde reside a lógica de negócio de uma aplicação [43] ou, em termos de programação orientada a objectos, corresponde ao conjunto de classes que modelam e suportam a aplicação [44]; a camada *Controller* é aquela que recebe pedidos de acções por parte do utilizador, as executa

interagindo com a camada *Model* e depois "renderiza" as vistas apropriadas presentes na camada *View*.

Considerando a restrição supra, a arquitectura dos diversos componentes constituintes do sistema é baseada no padrão MVC referido. Todos os componentes são aplicações *Ruby On Rails* que implementam os seus próprios modelos, os seus controladores e as suas vistas. A principal, e única, diferença arquitectural entre os vários componentes é a existência ou não de uma base de dados: existem componentes que são responsáveis pelo armazenamento de informação, pelo que dependem de uma base de dados própria (p.e. Catálogo); e existem componentes cuja função é apenas operacional ou de interacção com o utilizador, recorrendo aos restantes componentes do sistema para obterem a informação de que necessitam (p.e. IU).

Embora não seja uma diferença arquitectural, é pertinente tornar claro que o formato das vistas da IU – *HyperText Markup Language* (HTML) - é diferente do dos restantes componentes. Isto deve-se ao facto da IU ser o único componente que interage directamente com o utilizador do sistema, apresentando-lhe vistas que são interpretadas pelo navegador *web*. O formato de vistas utilizado pelos restantes componentes é JSON pois todos são componentes internos do sistema que trocam mensagens entre si através das suas *interfaces Representational State Transfer* (REST).

8.4. Arquitectura dinâmica da prova de conceito

Com objectivo de estudar as interacções entre o utilizador e o sistema e entre os componentes do sistema nos processos de subscrição e cancelamento de serviços, criaram-se diagramas de sequência que ilustram estas interacções. Foram criados seis diagramas, sendo que dois deles se focam na definição das interacções entre os componentes do sistema independentemente do serviço que se pretende subscrever ou cancelar e os restantes ilustram as interacções do sistema - para realizar a subscrição de um serviço - com dois actores externos: a API do serviço InvoiceXpress e a do HotelRunner. Estes são dois dos serviços seleccionados para a implementação da presente prova de conceito, o InvoiceXpress foi selecionado para estudar as interacções necessárias no caso de uma subscrição automática e o HotelRunner no caso de uma subscrição manual.

Este estudo, realizado através da construção de diagramas de sequência, consistiu em várias iterações. Em cada iteração foram surgindo novos problemas que se procuraram solucionar na iteração seguinte e assim sucessivamente até se chegar a uma versão final.

Estados de uma subscrição de serviço

Um dos problemas que emergiu prende-se com o processo de subscrição manual: como é que o sistema terá conhecimento de que um utilizador iniciou e terminou a tarefa que lhe é endereçada "Criar conta de utilizador no serviço" (ver Figura 10)? Para resolver esta questão, concluiu-se que uma subscrição (neste contexto, a sua representação no sistema) deve encontrar-se num de vários estados que permitem ao sistema saber em que etapa o processo de subscrição ou cancelamento se encontra. Além dos estados, as transições e restrições entre eles também têm que ser estabelecidas, possibilitando ao sistema saber, por exemplo, que a subscrição num serviço só pode ser cancelada se estiver activa. A Figura 18 apresenta um diagrama de estados que ilustra então os vários estados de uma subscrição e as transições entre eles.

No estado "Initial", a subscrição de serviço ainda não existe e é criada - passando para o estado "Pending subscription" - quando o utilizador subscreve o serviço. A subscrição de serviço encontrar-se-á nesse estado até que o processo de subscrição termine, transitando para o estado "Subscribed" caso a subscrição tenha sido efectuada com sucesso. Caso não tenha sido, a subscrição de serviço transitará para o estado "Troubleshooting" e aguardará a intervenção da *Hotelcracy* que, após a resolução do problema, alterará manualmente o estado da subscrição de serviço. Os restantes detalhes do diagrama podem ser consultados a seguir ao mesmo.

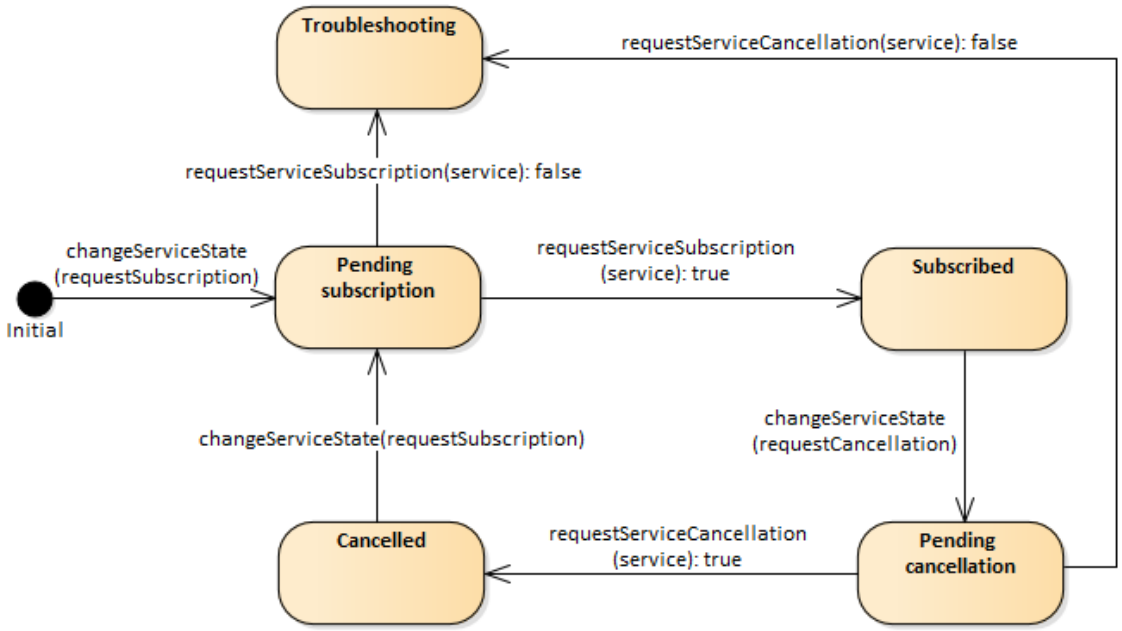

Figura 18 - Diagrama de estados de uma subscrição

Seguidamente, descreve-se cada estado e as transições do diagrama. As designações de estado estão formatadas a negrito e as transições a itálico, de forma a diferenciar as suas descrições.

- **Initial** – Estado inicial de uma subscrição de serviço: inexistente.

  changeServiceState(requestSubscription) – Quando o utilizador pretende subscrever o serviço, é feito um pedido de mudança de estado da subscrição consoante o estado em que esta se encontra. Esta transição representa esse pedido.

- **Pending subscription** – Quando a subscrição de serviço se encontra neste estado significa que a subscrição se encontra pendente e o pedido de subscrição no serviço está a ser processado.

  requestServiceSubscription(service): true – Esta transição corresponde ao resultado positivo do processo de subscrição no serviço, ou seja, esta foi realizada com sucesso.

  requestServiceSubscription(service): false – Esta transição ocorre quando não foi possível concluir a subscrição.

- **Subscribed** - Quando a subscrição de serviço se encontra neste estado significa que está subscrito e em operação.

  changeServiceState(requestCancellation) – Quando o utilizador pretende cancelar o serviço, é feito um pedido de mudança de estado da subscrição de serviço consoante o estado em que esta se encontra. Esta transição representa esse pedido.

- **Pending cancellation** - Quando a subscrição de serviço se encontra neste estado significa que o cancelamento no serviço se encontra pendente e o pedido de cancelamento está a ser processado.

  requestServiceCancellation(service): true – Esta transição corresponde ao resultado positivo do processo de cancelamento do serviço, ou seja, o cancelamento foi realizado com sucesso.

  requestServiceCancellation(service): false - Esta transição ocorre quando não foi possível conluir o cancelamento do serviço.

- **Cancelled** - Quando a subscrição de serviço se encontra neste estado significa que o serviço já esteve subscrito, mas de momento a subscrição está cancelada.

Sequências de execução da operação de subscrever serviço

Como referido, foram criados seis diagramas de sequência que podem ser visualizados no Anexo 5 [6]. Destes seis, quatro deles representam a operação de subscrever serviço. No presente relatório optou-se por apresentar e descrever apenas dois destes quatro: o diagrama que representa o pedido de uma nova subscrição num serviço SaaS por parte do utilizador independentemente do serviço; e o que representa o processamento de um pedido de subscrição que tem que ser feita de forma manual. O primeiro, como se abstrai de comunicações mais complexas que ocorrem no interior do sistema durante o processamento do pedido, é o que melhor se adequa a proporcionar uma visão geral do estudo realizado. O segundo é aqui descrito exactamente para demonstrar alguma dessa complexidade.

A Figura 19 mostra o diagrama de pedido de nova subscrição e a Figura 20 mostra o processamento desse pedido no caso de uma subscrição manual (HotelRunner).

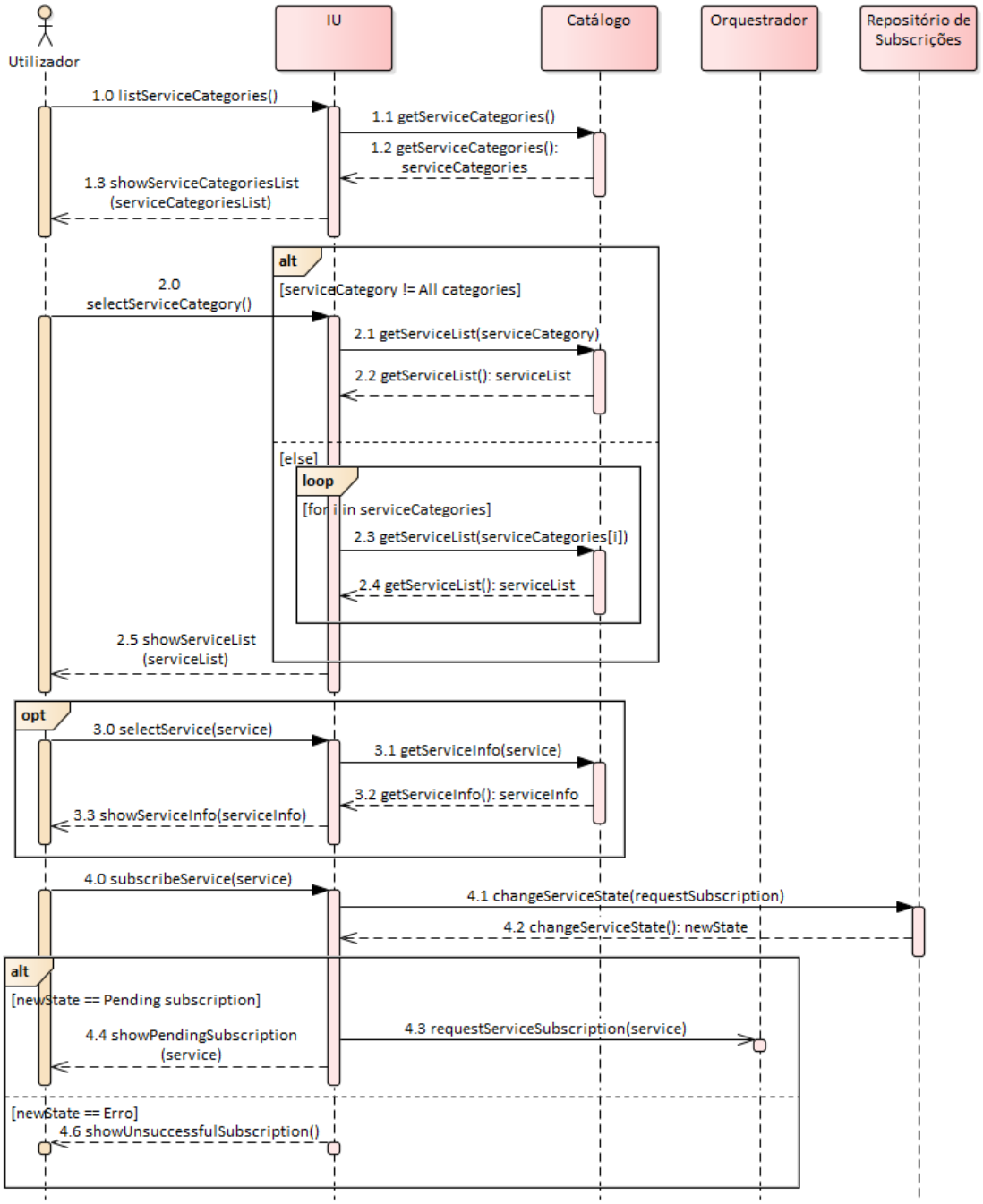

Figura 19 - Diagrama de sequência do pedido de subscrição

De forma a guiar o leitor na visualização e compreensão do diagrama de pedido de nova subscrição (Figura 19), apresenta-se de seguida uma descrição do processo:

1. O Utilizador pede à IU a lista de categorias de serviços disponíveis no sistema.
    1.1. Por sua vez, ao receber este pedido, a IU faz um pedido ao Catálogo para que este devolva a lista de categorias de serviços existentes no sistema.
    1.2. O Catálogo responde com a lista de categorias de serviços.
    1.3. Como resposta a 1., a IU mostra ao utilizador a lista de categorias de serviços disponíveis no sistema.
2. O Utilizador seleciona uma categoria de serviços das que lhe são mostradas pela IU. A categoria selecionada pode ser "Todas as categorias".

2.1. Caso a categoria selecionada não seja "Todas as categorias", a IU faz um pedido ao Catálogo para que este envie a lista de serviços da categoria selecionada disponíveis no sistema.

2.2. O Catálogo responde com a lista de serviços da categoria selecionada.

2.3. No caso da categoria selecionada ser "Todas as categorias", a IU percorre todas as categorias de serviços e pede ao Catálogo a lista de serviços de cada categoria.

2.4. O Catálogo responde coma a lista de serviços de cada categoria.

2.5. A IU, em resposta a 2., mostra ao Utilizador a lista de serviços da categoria selecionada pelo Utilizador.

3. O Utilizador pode, opcionalmente, selecionar um serviço com o intuito de obter informação sobre o serviço (funcionalidades, custo, crítica editorial e crítica de utilizadores). Esta mensagem representa este pedido à IU por parte do Utilizador.

3.1. Ao receber este pedido, a IU faz um pedido ao Catálogo para que este devolva a informação sobre o serviço selecionado pelo Utilizador.

3.2. O Catálogo devolve à IU a informação do serviço solicitado pela mesma.

3.3. Como resposta a 3., a IU mostra ao Utilizador a informação sobre o serviço selecionado por este.

4. O Utilizador confirma através da IU que pretende subscrever determinado serviço.

4.1. A IU, por sua vez, envia um pedido ao Repositório de Subscrições para que este altere o estado da subscrição de serviço para o Utilizador (ver Figura 18).

4.2. O Repositório de Subscrições responde com o novo estado da subscrição de serviço ("Pending subscription" ou "Pending subscription approval") ou com uma mensagem de erro caso a subscrição de serviço não se encontre num estado em que possa ocorrer um pedido de subscrição.

4.3. No caso do Repositório de Subscrições devolver "Pending subscription" como novo estado da subscrição de serviço, a IU, envia um pedido ao Orquestrador para que este trate da subscrição solicitada. Note-se que esta é uma **mensagem assíncrona** e que, portanto, a IU não fica à espera da resposta por parte do Orquestrador e segue a sua execução.

4.4. No caso do Repositório de Subscrições devolver "Pending subscription" como novo estado da subscrição de serviço e após o envio do pedido de subscrição ao Orquestrador, a IU em resposta a 4., mostra ao Utilizador que o processo de subscrição foi iniciado e que a subscrição no serviço se encontrada pendente.

4.5. No caso do Repositório de Subscrições reportar um erro, a IU informa o Utilizador de que a subscrição no serviço falhou e que portanto se encontra no estado "Troubleshooting" a aguardar intervenção de um utilizador do *backoffice* do sistema.

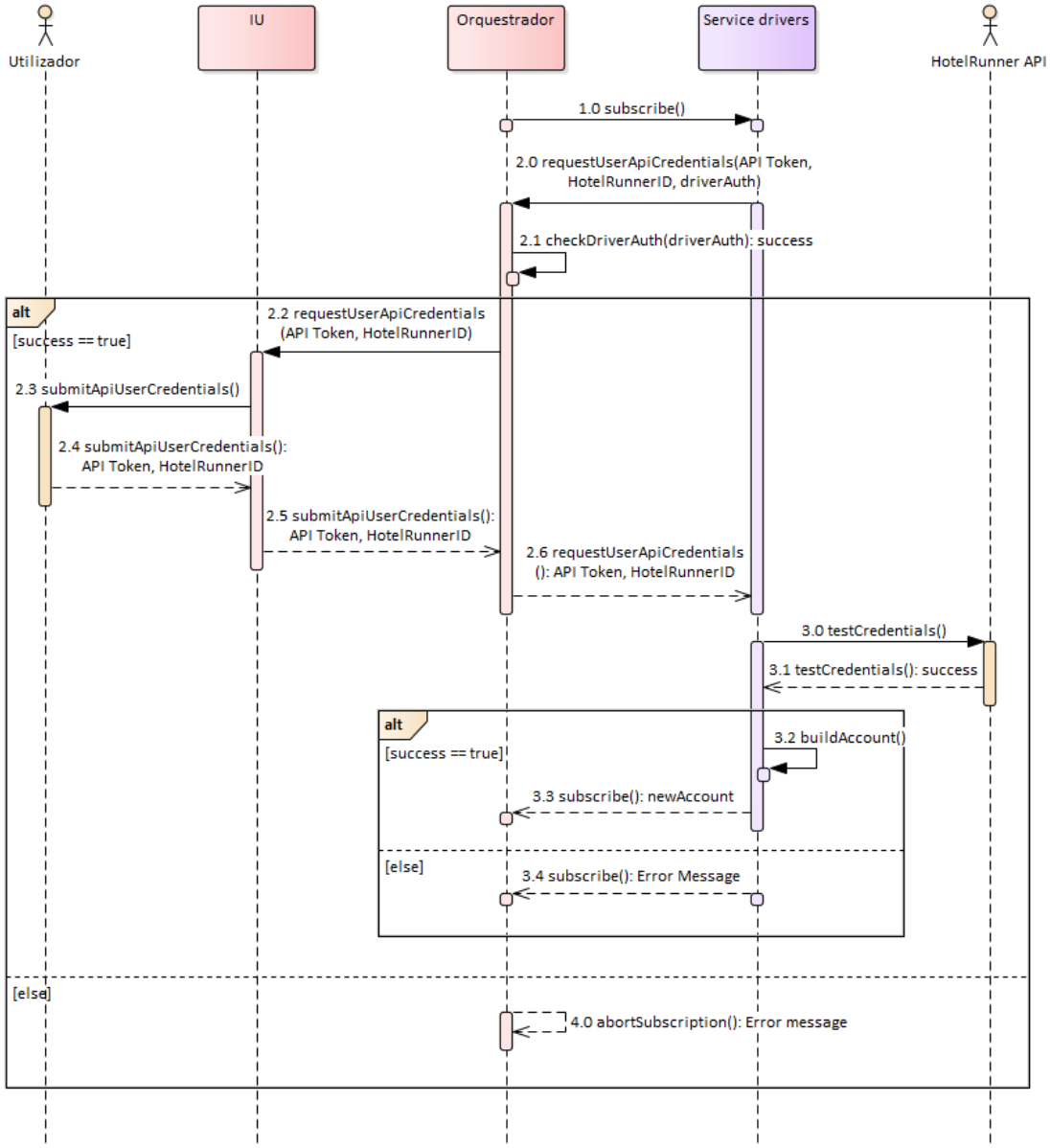

Figura 20 - Diagrama de sequência do processamento de subscrição manual

À semelhança do diagrama anterior, apresenta-se de seguida uma descrição do processo:

1. O Orquestrador envia um pedido de subscrição ao Service driver.
2. O Service driver pede ao Orquestrador as credenciais do Utilizador para a utilização da API do HotelRunner.
    2.1. O Orquestrador, antes de reencaminhar este pedido, verifica a autenticidade do Service driver.
    2.2. Caso o Service driver seja autêntico, o Orquestrador reencaminha o pedido das credenciais do Utilizador para a IU.
    2.3. Por sua vez, a IU pede as credenciais ao Utilizador e aguarda que este as submeta.
    2.4. O Utilizador submete as suas credenciais para a utilização da API do HotelRunner.

2.5. Em resposta a 2.2, a IU devolve as credenciais ao Orquestrador.

2.6. Em resposta a 2., o Orquestrador envia as credenciais para o Service driver.

3. Após a recepção das credenciais do Utilizador para a utilização da API do serviço, o Service driver faz um pedido à API do HotelRunner utilizando as credenciais recebidas a fim de verificar a sua validade

   3.1. A API do HotelRunner responde com sucesso ou com uma mensagem de erro.

4. No caso da API do HotelRunner responder com sucesso, o Service driver cria uma nova conta no formato que o Orquestrador espera.

   4.1. O Service driver, em resposta a 1., devolve a nova conta de utilizador criada.

5. Caso a API do HotelRunner tenha respondido com um erro, o Service driver, em resposta a 1., devolve ao Orquestrador uma mensagem de erro.

6. No caso de a verificação de autenticidade efectuada em 2.1 ter falhado, a subscrição é abortada pelo Orquestrador que lança uma mensagem de erro.

Sequências de execução da operação de cancelar serviço

Dos seis diagramas de sequência criados [6], dois representam a operação de cancelar serviço. Nesta secção apresentam-se e descrevem-se os dois diagramas: o primeiro representa a interacções entre os componentes que actuam quando o utlizador efectua um pedido de cancelamento de um serviço (Figura 21); o segundo representa as interacções que ocorrem quando o processo de cancelamento é iniciado (Figura 22).

O diagrama apresentado na Figura 18 representa então as interacções do utilizador com o sistema quando este pretende cancelar um serviço, bem como as mensagens trocadas entre os componentes constituintes do sistema em consequência desse pedido.

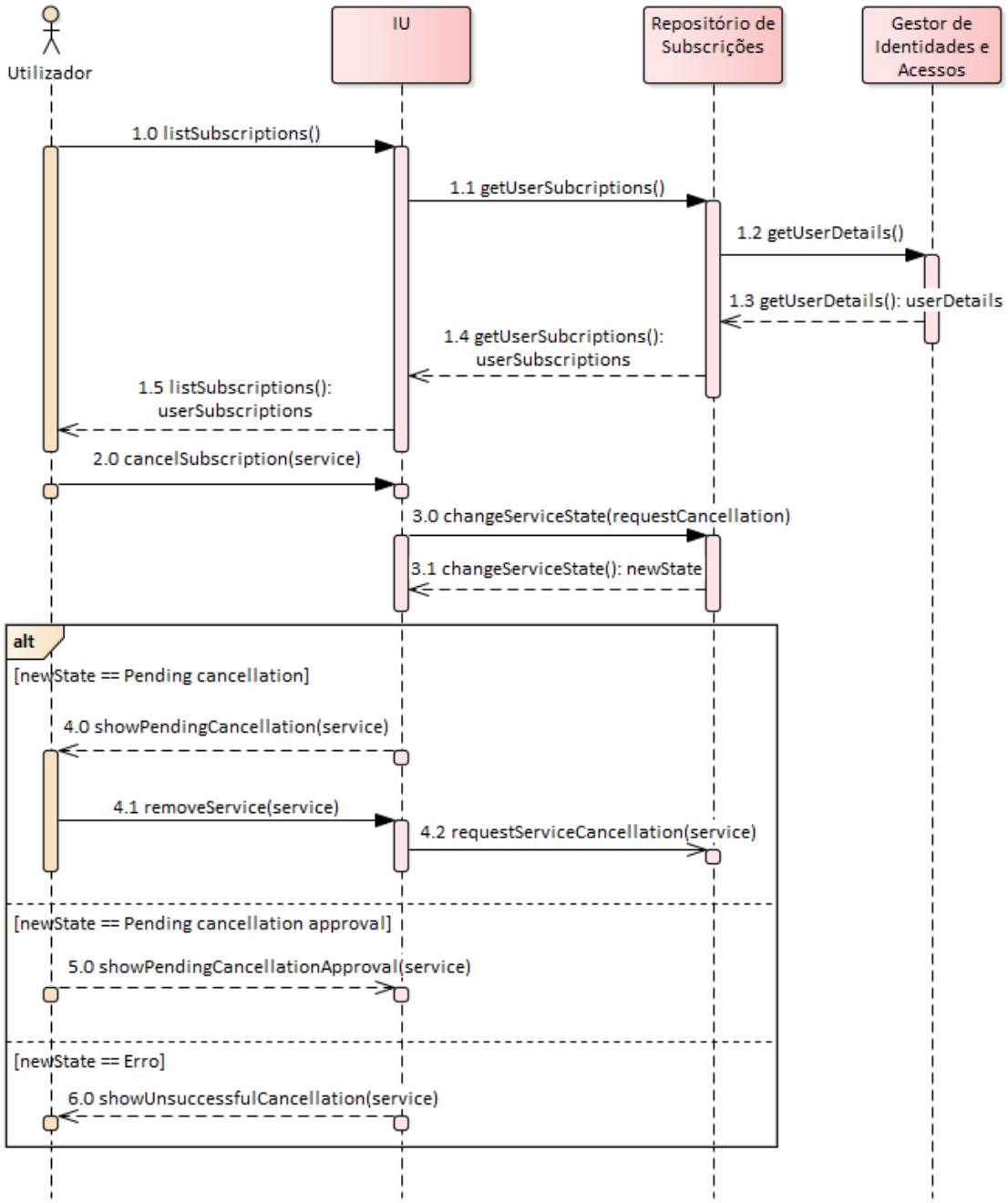

Figura 21 - Diagrama de pedido de cancelamento de serviço

De forma a guiar o leitor na visualização e compreensão do diagrama de pedido de cancelamento de serviço (Figura 21), apresenta-se de seguida uma descrição do processo:

1. O Utilizador, através da IU, informa o sistema de que obter a sua lista de subscrições.
    1.1. A IU pede ao Repositório de Subscrições a lista de subscrições do Utilizador.
    1.2. O Repositório de Subscrições pede ao Gestor de Identidades e Acessos os detalhes do Utilizador pois necessita de saber qual é o utilizador que pretende gerir as suas subscrições.
    1.3. O Gestor de Identidades e Acessos devolve os detalhes do Utilizador ao Repositório de Susbcrições.

1.4. O Repositório de Subscrições, em resposta a 1.1, envia à IU as subscrições do Utilizador. São enviadas todas as subscrições independentemente do estado em que se encontrem.

1.5. Por sua vez, a IU, mostra ao Utilizador todas as suas subscrições.

2. Após serem mostradas ao Utilizador as suas subscrições, o Utilizador envia à IU um pedido de cancelamento da subscrição de determinado serviço.

3. A IU, por sua vez, envia um pedido ao Repositório de Subscrições para que este altere o estado da subscrição de serviço para o Utilizador (ver Figura 18).

   3.1. O Repositório de Subscrições responde com o novo estado da subscrição de serviço ("Pending cancellation" ou "Pending cancellation approval") ou com uma mensagem de erro caso a subscrição de serviço não se encontre num estado em que possa ocorrer um pedido de cancelamento.

4. No caso de o Repositório de Subscrições devolver "Pending cancellation" como novo estado da subscrição de serviço, a IU informa o Utilizador deste novo estado.

   4.1. O Utilizador, após cancelar a sua subscrição junto dos prestadores do serviço em causa, envia um pedido à IU para que a subscrição do serviço no sistema seja removida.

   4.2. A IU, após receber o pedido do Utilizador para remover a subscrição do serviço no sistema, envia um pedido ao Repositório de Subscrições para que este trate do cancelamento solicitado. Note-se que esta é uma mensagem assíncrona e que, portanto, a IU não fica à espera da resposta por parte do Repositório de Subscrições e segue a sua execução. Esta é a mensagem que despoleta o processo representado no diagrama de sequência que será apresentado na Figura 22.

5. No caso do Repositório de Subscrições devolver "Pending cancellation approval" como novo estado da subscrição de serviço, a IU em resposta a 2., mostra ao Utilizador que o pedido de cancelamento se encontra à espera de aprovação por um utilizador com permissões.

6. No caso do Repositório de Subscrições reportar um erro, a IU informa o Utilizador de que o cancelamento do serviço falhou.

O diagrama ilustrado na Figura 22 representa o diagrama de sequência do processamento de cancelamento de serviço, que ocorre após o Utilizador ter cancelado a sua subscrição junto dos prestadores do serviço em causa e ter confirmado que pretende também cancelar a subscrição no sistema.

Ao contrário do processamento de nova subscrição (ver Figura 20), o processamento de cancelamento de serviço não necessita da interacção com nenhum actor externo (a API do serviço que se pretende cancelar) porque a responsabilidade de cancelar a subscrição junto do serviço é do utilizador, ficando a cargo do sistema cancelar a subscrição apenas internamente. Consequentemente o componente *Service driver* não é utilizado neste processo, dispensando também o componente Orquestrador cuja função é interagir com os diversos Service drivers.

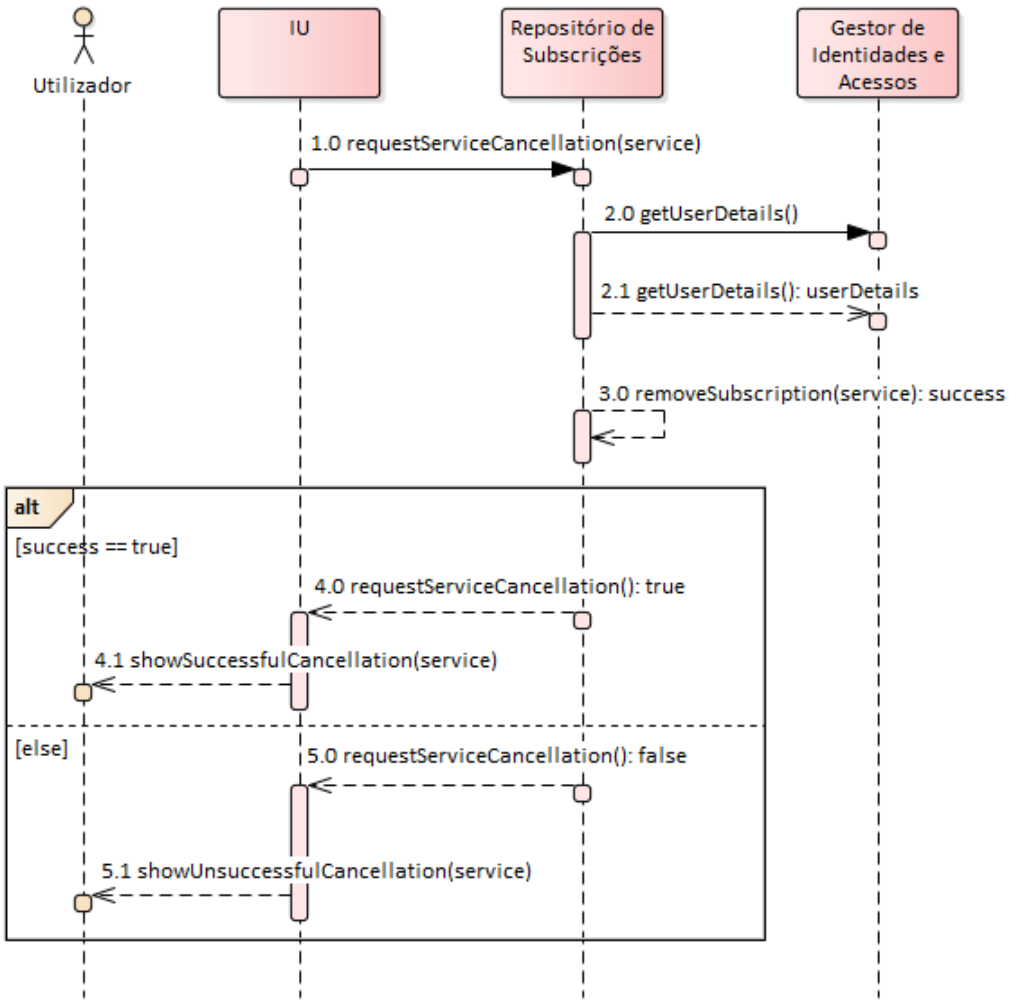

Figura 22 – Diagrama sequência do processamento de cancelamento de serviço

De forma a guiar o leitor na visualização e compreensão do diagrama de processamento de cancelamento de serviço (Figura 22), apresenta-se de seguida uma descrição do processo:

1. A IU faz um pedido ao Repositório de Subscrições para remover do sistema a subscrição no serviço selecionado.
2. O Repositório de Subscrições pede ao Gestor de Identiddades e Acessos os detalhes do utilizador que pretende cancelar a sua subscrição no serviço.
3. O Repositório de Subscrições remove a subscrição no serviço do Utilizador removendo credenciais e alterando o estado da subscrição de serviço para "Cancelled".
4. Caso a remoção da subscrição no serviço tenha sido efecutada com sucesso, o Repositório de Subscrições em resposta a 1., informa a IU que o cancelamento foi realizado com sucesso.
   4.1. Por sua vez, a IU informa o Utilizador de que o cancelamento foi realizado com sucesso.
5. No caso da remoção da subscrição no serviço tenha falhado, o Gestor de Susbcrições em resposta a 1., informa a IU deste facto.
   5.1. Por sua vez, a IU informa o Utilizador de que o cancelamento falhou.

Capítulo 9
Desenvolvimento da prova de conceito

Desenhada a solução para a prova de conceito da subscrição e cancelamento de serviços SaaS através de uma plataforma de gestão, o passo seguinte foi a sua implementação. Neste capítulo descrevem-se alguns pontos considerados importantes desta etapa, como o processo de desenvolvimento utilizado, os principais desafios ultrapassados e os testes efectuados ao sistema.

9.1. Preparação para o desenvolvimento

Como referido na secção 3 do Capítulo 6 deste relatório, a implementação da plataforma *Hotelcracy Apps* tem uma restrição tecnológica: a utilização da *framework Ruby On Rails*. Sendo que o estagiário nunca tinha tido contacto com esta *framework* nem com a linguagem *Ruby* sobre a qual ela é implementada, tornou-se essencial a realização de uma fase de preparação através do estudo - teórico e prático - tanto da linguagem *Ruby* como da *framework Ruby On Rails*.

Como esta *framework* assenta na linguagem *Ruby*, o primeiro passo da preparação foi, logicamente, o estudo da linguagem. Para tal, realizaram-se os cursos "*Ruby Primer*" e "*Ruby Primer: Ascent*" disponibilizados pela firma *C42 Engineering* na sua plataforma de ensino interactivo de programação *CodeMonk* [45].

Dominados os princípios básicos da linguagem *Ruby*, estavam reunidas as condições para começar o estudo de *Ruby On Rails*. Para uma iniciação nos conceitos, fundamentos e teorias desta *framework*, leu-se o artigo *The Rails Doctrine* [46] que, além da interiorização desses mesmos conceitos, proporcionou forte motivação para o estudo prático que se seguiu. Estudo prático esse que consistiu então na leitura e resolução dos exercícios propostos no livro *The Ruby On Rails Tutorial* [47], escrito por Michael Hartl.

9.2. Processo de desenvolvimento

O processo de desenvolvimento utilizado para a implementação da prova de conceito foi *Test-Driven-Development* (TDD). Este é um processo de desenvolvimento "redescoberto" por Kent Beck que escreveu o livro "*Test-Driven Development by Example"* [48] sobre este tópico. O autor afirma que o seu papel foi a "redescoberta" do TDD pois as verdadeiras origens deste processo remontam a 1957, ano em que D.D. McCracken publicou o livro "*Digital Computer Programming"* referindo-se a esta prática como "*Program Checkout*" [49].

O TDD é, essencialmente, um processo que inverte a sequência normal da programação na qual primeiro o programador escreve o programa e depois o testa. Kent Beck defende que a escrita de testes antes da escrita do código que os satisfaz permite ao programador: trabalhar com confiança, pois à medida que vai avançado os testes já escritos vão-lhe assegurando que o seu trabalho é válido; trabalhar em séries de passos alcançáveis em vez de atacar o problema maior todo de uma vez; assegurar que o *design* do software atende à necessidade do código; e, por último mas não menos importante, deixar para traz uma *suite* de testes que ajuda a preservar a integridade do código desenvolvido [50]. Este processo de desenvolvimento, assente em ciclos de desenvolvimento curtos e repetitivos, está associado aos ciclos de vida *agile*, mais concretamente à metodologia *Extreme Programming* criada

também por Kent Beck. A Figura 23 ilustra estes ciclos curtos de desenvolvimento e fornece uma ideia geral do processo.

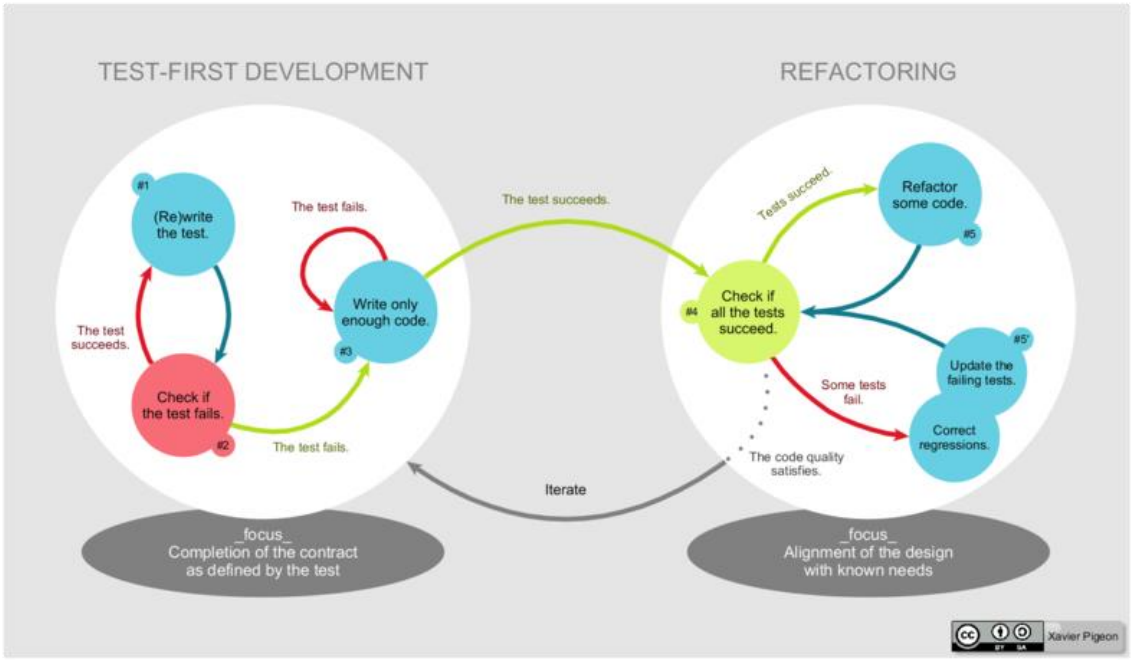

Figura 23 - Ciclo de vida do Test Driven Develpment[1]

A escolha da utilização do *Test-Driven Development* para o desenvolvimento da prova de conceito assentou nas várias vantagens já referidas e em particular porque é direccionado para ciclos de trabalho de trabalhadores individuais, como é o caso do estagiário. Uma das vantagens já referidas – uma *suite* de testes automática que cobre todo o trabalho desenvolvido até ao momento – provou ser essencial para o método de desenvolvimento incremental baseado em componentes – e não em funcionalidades - utilizado.

Como grande parte das funcionalidades a implementar envolvem mais do que um componente, impôs-se a necessidade de adoptar um método de desenvolvimento em que primeiro é implementado cada componente do sistema [Cap. 7, sec. 7.1], desenvolvidos os seus módulos e métodos necessários à interacção com os restantes componentes, e só depois são implementadas as funcionalidades do sistema como um todo. A ordem com que foram implementados os diversos componentes foi a seguinte:

1) **Catálogo**: Este foi o primeiro componente a ser implementado por ser o mais autocontido de todos, ou seja, não necessita de informação que reside noutro componente para cumprir com a sua responsabilidade de armazenar e servir informação sobre os serviços SaaS.

2) **IU** (*Marketplace*): Estando o componente Catálogo já implementado e pronto a disponibilizar a informação comercial dos serviços SaaS, o passo seguinte foi a implementação do contexto *Marketplace* do componente IU [Cap. 6, sec. 6.2] que apresenta essa mesma informação ao utilizador. Além do facto de esta informação já estar disponível para ser apresentada, pesou também na decisão de implementar este elemento em segundo lugar o facto de ficar desenvolvido desde logo algum trabalho visível para o exterior do sistema.

[1]"Lifecycle of the Test-Drive Development method" by Xarawn is licensed under CC BT-SA 4.0

3) **Repositório de Subscrições**: Apesar de este componente pressupor a existência de utilizadores – são estes que subscrevem serviços e o componente responsável pela gestão de utilizadores (Gestor de Identidades e Acessos) ainda não estava implementado – optou-se por desenvolver de seguida o Repositório de Subscrições, estabelecendo a existência de apenas um único utilizador do sistema sem necessidades de autenticação. Esta decisão foi baseada no diagrama de sequência de pedido de subscrição (Figura 19) no qual este componente é o primeiro a intervir quando um pedido de subscrição é realizado pelo utilizador.
4) **Orquestrador**: Nesta fase, iniciou-se a implementação da essência do processamento de subscrições de serviços SaaS por parte do sistema, e portanto, antes de implementar os Service Drivers foi necessário construir o componente que administra este processamento interagindo com eles.
5) **Drivers:** Com o Orquestrador desenvolvido, o passo seguinte foi a implementação dos Service Drivers. Primeiro os que comunicam com os SaaS que permitem subscrições automáticas utilizados para a presente prova de conceito (InvoiceXpress e InvoiceOcean) e depois o que comunica com o serviço SaaS em que o processo de subscrição é manual (HotelRunner).
6) **IU** (Gestão de Serviços): Integrando todos os componentes desenvolvidos para efectivar a subscrição e cancelamento de serviços, implementou-se o contexto Gestão de Serviços do componente IU, onde o utilizador pode subscrever serviços SaaS e cancelar as suas subscrições.
7) **Gestor de Identidades e Acessos**: Com os processos de subscrição e cancelamento para um utilizador único assegurados pelos componentes da prova de conceito do sistema desenvolvidos, foi a altura de desenvolver e integrar o componente Gestor de Identidades e Acessos para o sistema permitir a autenticação, autorização e utilização do sistema por múltiplos utilizadores.

Graças ao processo de desenvolvimento utilizado – *Test-Driven Develpment* (TDD) – foi possível utilizar esta abordagem componente a componente, uma vez que após a integração de um novo componente, existia já uma *suite* automática de testes que permitia identificar rapidamente se (e que) código já desenvolvido nos outros componentes necessitava de ser actualizado. Um exemplo prático em que o TDD se mostrou essencial foi na conversão de um sistema de utilizador único para um sistema de múltiplos utilizadores.

Importa também referir que ao longo desta etapa foram sendo feitas apresentações sobre o ponto de situação do desenvolvimento da prova de conceito nas reuniões do consórcio.

9.4. Testes

Os testes aos componentes do sistema escritos consoante o processo de desenvolvimento descrito na secção anterior podem ser classificados em quatro tipos distintos, tendo em conta a arquitectura MVC que a *framework Ruby on Rails* implementa: os testes aos modelos, os testes aos métodos dos controladores, os testes às vistas apresentadas ao utilizador e os testes de integração. Além destes quatro tipos de testes relacionados com a arquitectura MVC, foram também escritos testes funcionais do tipo *black-box* que despoletam uma acção através da *interface* de utilização e verificam o seu resultado através daquilo que é apresentado ao utilizador de seguida, abstraindo-se de todo processamento que está a ocorrer dentro do sistema. Os pontos seguintes descrevem sucintamente cada tipo de testes desenvolvido.

- Os testes aos modelos verificam se um modelo está correctamente definido. Por exemplo, se um modelo "Service" pertence a um modelo "Category" ou se o modelo valida a presença de determinado atributo.
- Os testes aos métodos dos controladores incidem sobre os métodos que não necessitam de interagir com outros componentes da platforma *Hotelcracy Apps*. Verificam se para os vários *inputs* possíveis de um método, este devolve o *output* esperado e se ocorre o comportamento esperado no contexto do componente que é exterior ao método.
- Os testes às vistas executam acções dos controladores e verificam se é mostrado ao utilizador o que, e como, é esperado. Naturalmente, este tipo de testes foi escrito apenas para o componente IU.
- Os testes de integração são feitos de um ponto de vista externo ao componente, em que é realizado um pedido a determinada funcionalidade que um componente expõe para o exterior e é verificada a resposta devolvida consoante o pedido enviado.

Para o desenvolvimento dos testes foi utilizada a ferramenta de testes *RSpec* [51]. Esta ferramenta proporciona uma linguagem de utilização muito expressiva e orientada ao comportamento, simplificando a prática de TDD e dando origem a testes que se descrevem a si próprios.

Essencialmente, foram desenvolvidos três testes funcionais automáticos que validam a subscrição dos três serviços SaaS seleccionados para a prova de conceito: InvoiceXpress, InvoiceOcean (subscrição automática) e HotelRunner (subscrição manual). Devido à sua natureza, estes testes - ao contrário de todos os outros - não foram desenvolvidos antes da funcionalidade que testam, mas sim quando todo o sistema já se encontrava desenvolvido. O cancelamento de serviços SaaS foi coberto apenas por testes de integração dado que o processo é o mesmo qualquer que seja o serviço SaaS que o utilizador pretende cancelar. Pela observação da Figura 24 - que mostra o *output* da execução destes três testes - pode-se verificar a propriedade de auto-descrição referida no parágrafo anterior, que resulta em *outputs* elucidativos e estruturados.

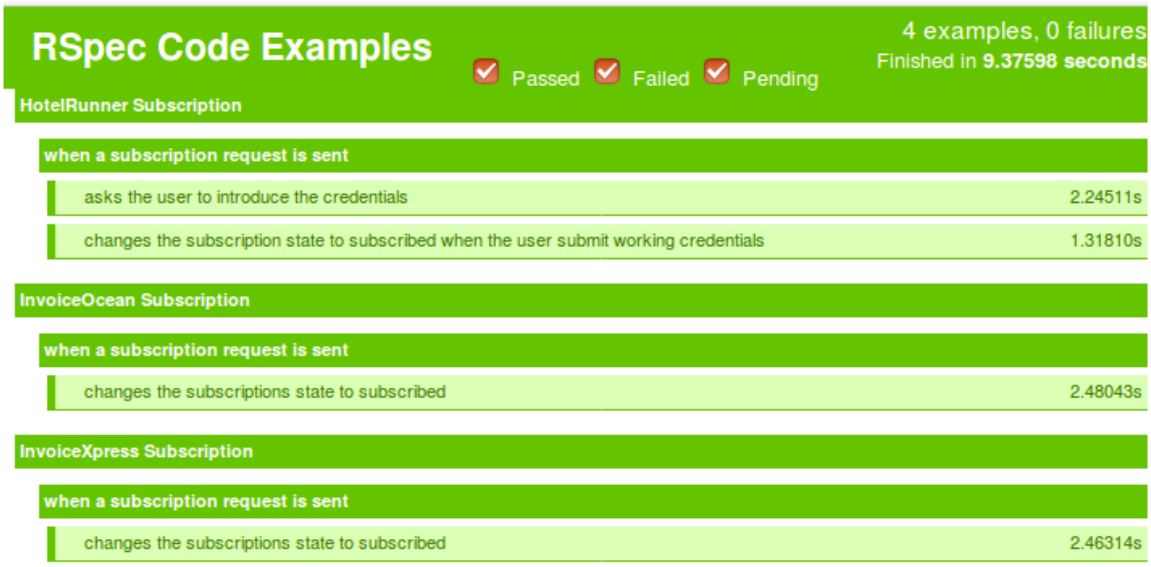

Figura 24 - Output dos testes funcionais

Durante a especificação dos testes foram surgindo algumas necessidades relacionadas com a interacção com serviços externos. Por exemplo, se se pretende testar isoladamente uma funcionalidade do Orquestrador que envolve pedidos a um Service Driver (que por sua vez realiza pedidos a um serviço externo), os testes realizados não podem depender do correcto funcionamento, nem do Service Driver nem do serviço externo. Como tal, surgiu a necessidade de implementar dois Service Drivers que são utilizados neste tipo de testes: um que responde como se de uma subcrição automática num serviço externo se tratasse e outro como se de uma subscrição manual, não realizando nenhum processamento real ou pedido ao exterior. Outra necessidade que surgiu, também ela relacionada com as interacções com serviços externos, foi a definição de *mocks* que simulam as respostas dos serviços externos com que cada um dos Service Drivers desenvolvidos comunica. Desta forma, ao executar os testes automáticos – que vão sendo executados várias vezes ao longo do ciclo de desenvolvimento – não são enviados pedidos reais aos serviços SaaS, caso fossem, estariam a ser criadas várias subscrições para utilizadores inexistentes nos serviços externos em causa.

Em jeito de conclusão da presente secção, apresenta-se a Tabela 16 que pretende resumir a *suite* de testes automáticos produzida. As colunas que a compõem são:

- **Componente:** identifica o componente a que se refere a linha da tabela.
- **Número de testes:** número total de testes realizados ao componente, englobando os diferentes tipos de testes referidos no ínicio desta secção.
- **Tempo de uma execução:** tempo decorrido entre o ínicio e o fim de uma execução da *suite* de testes ao componente. Note-se que este não é um valor médio mas meramente um valor indicativo.
- **Cobertura:** percentagem de código fonte do componente que é executado quando a *suite* de testes ao componente é executada.

Tabela 16 - Resumo dos testes

Componente	Número de testes	Tempo de uma execução	Cobertura
Catálogo	145	9.45 seg	99.43%
IU	127	29.65 seg	90.28%
Repositório de Subscrições	81	4.51 seg	98.56%
Orquestrador	89	8.63 seg	85.27%
Test Driver Auto	16	0.45 seg	97.37%
Test Driver Manual	20	0.79 seg	94.23%
InvoiceXpress Driver	35	0.72 seg	93.75%
InvoiceOcean Driver	36	0.65 seg	93.75%
HotelRunner Driver	20	1.69 seg	96.43%
Gestor de Identidades e Acessos	24	1.66 seg	93.97%
TOTAL	593	58.2 seg	92.66%

Como se pode verificar pela Tabela 16, em virtude do processo de desenvolvimento seguido – *Test-Driven Development* (TDD) – a percentagem de cobertura total alcançada é bastante satisfatória encontrando-se acima dos 90%. Verifica-se que existem inclusive componentes cuja suite de testes tem uma cobertura muito próxima dos 100%. A suite de testes ao

Orquestrador é a que tem um valor de cobertura mais baixo – embora também ele satisfatório – devido ao seu maior nível de complexidade em relação aos restantes, dificultando o desenho de testes que cubram todos os caminhos possíveis da sua execução.

Ainda pela Tabela 15, note-se que, em cerca de 1 minuto, é possível validar e verificar a integridade de todo o trabalho desenvolvido.

9.4. Instalação

Encontrando-se a prova de conceito para a subscrição e cancelamento de serviços através da plataforma *Hotelcracy Apps* desenvolvida e testada em ambiente de desenvolvimento, ficou apenas a faltar realizar a sua instalação em ambiente de produção, ou seja, num servidor remoto.

Para tal, utilizou-se a ferramenta de *Capistrano* [52] que permite a automatização de tarefas de instalação dos componentes num servidor remoto. Algumas das tarefas da instalação foram feitas manualmente através de SSH, tarefas estas relacionadas com a preparação do servidor para executar a prova de conceito desenvolvida:

- Instalação da linguagem *Ruby*
- Instalação *framework Ruby on Rails*
- Instalação de servidor MySQL
- Instalação das ferramentas tecnológicas utilizadas por cada componente
- Criação das bases de dados necessárias no servidor MySQL

A Figura 25 apresenta o diagrama de *deployment* da prova de conceito. São representados os vários servidores *web,* neste caso *Rails,* que estão a correr na máquina onde foi instalada a prova de conceito; o servidor MySQL e as bases de dados criadas para os componentes que dispõem de base dados; os endereços que são utilizados pelos vários servidores *web* para comunicarem entre si; e o endereço em que o servidor disponibiliza a prova de conceito para o exterior.

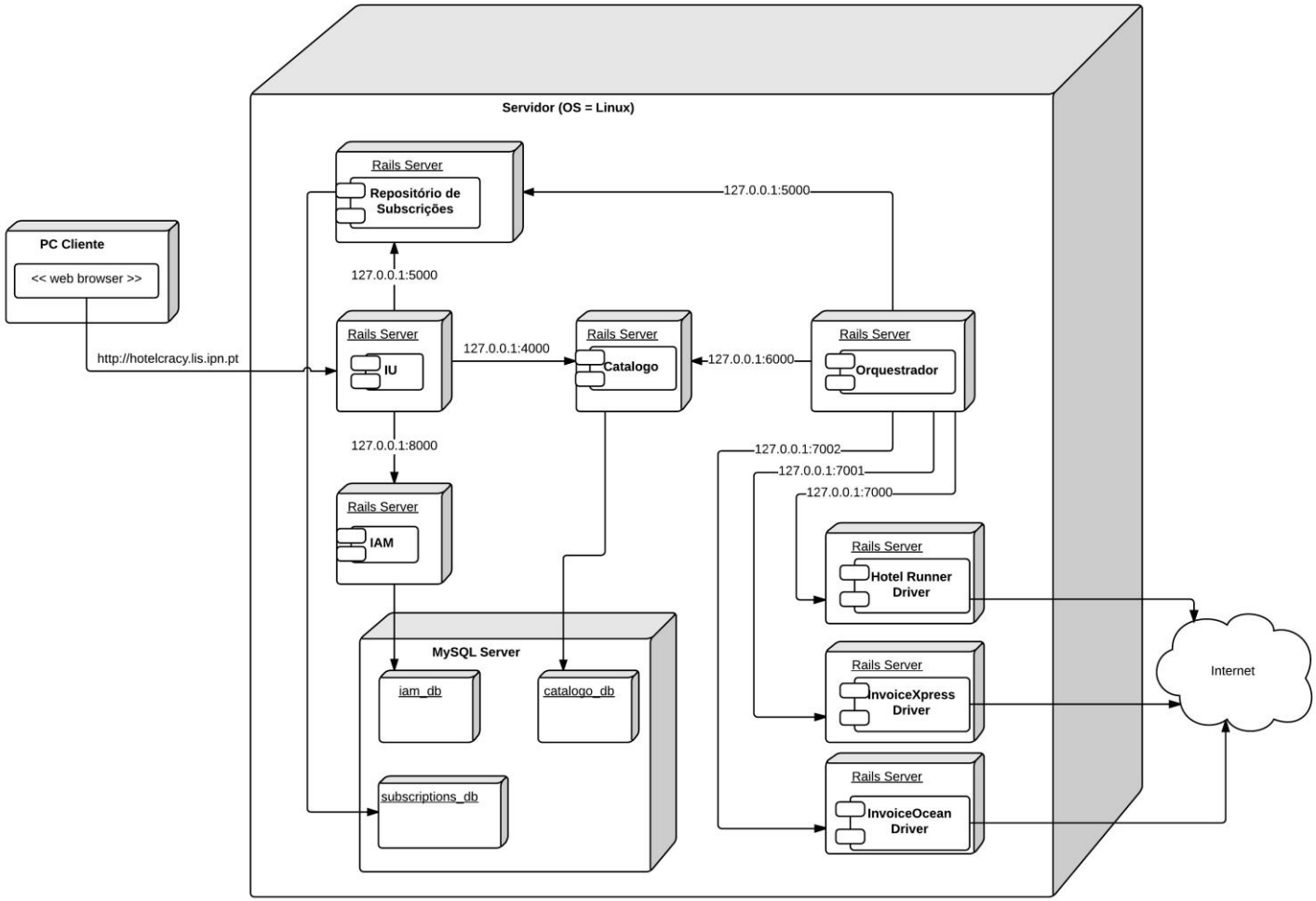

Figura 25 - Diagrama de deploymentda prova de conceito

Capítulo 10
Conclusões

O estágio que o presente relatório acaba de descrever, inseriu-se no projecto *Hotelcracy Apps* que pretende ser uma plataforma integradora de serviços SaaS utilizados no sector do alojamento. O principal objectivo do estágio foi desenvolver os componentes de subscrição e cancelamento de serviços através da plataforma, constituindo uma prova de conceito da viabilidade dos processos que os suportam. Complementando este objectivo principal, o trabalho realizado teve também como finalidade sustentar as próximas fases do projecto em que o estágio se inseriu, através das várias investigações, análises e especificações levadas a cabo. Nos parágrafos seguintes é apresentada uma reflexão sobre os resultados do trabalho realizado ao longo do estágio e como estes suportaram o trabalho realizado no segundo semestre e suportarão o trabalho que será realizado no futuro no âmbito do projecto.

O trabalho apresentado neste relatório permitiu atingir todos os objectivos estabelecidos para o primeiro semestre do estágio, com a excepção da análise preliminar de tecnologias para a implementação dos componentes de subscrição e cancelamento de serviços. Este facto deveu-se à carga de trabalho inerente ao primeiro objectivo do estágio: análise do estado da arte. Ainda assim, com o decorrer do estágio, concluiu-se que esta tarefa deveria ser descartada devido às restrições tecnológicas impostas pela empresa líder do projecto.

Por outro lado, a análise do estado da arte com a abrangência e detalhe com que foi realizada, constituiu uma base sólida para as restantes tarefas do estágio pois permitiu antecipar possíveis entraves às mesmas e seleccionar com maior probabilidade de sucesso os serviços SaaS a utilizar na implementação da prova de conceito dos componentes de subscrição e cancelamento de serviços da plataforma *Hotelcracy Apps*.

Como os processos definidos na primeira metade do estágio, foram definidos de forma detalhada - através de subscrições e cancelamentos de teste automáticos e manuais – a viabilidade das tarefas que foram realizadas no segundo semestre ficou mais próxima de estar assegurada.

O trabalho realizado ao longo de todo o estágio atingiu e efectivou o seu principal objectivo: provar a viabilidade da subscrição e cancelamento de serviços SaaS através da plataforma *Hotelcracy Apps*. Mas, além deste contributo, o estágio forneceu vários benefícios ao projecto no qual se inseriu: a análise de serviços SaaS utilizados no sector do alojamento e a sua documentação é uma ferramenta muito útil para todos os envolvidos no desenvolvimento do projecto, não só pela agregação de muita informação dispersa sobre bastantes serviços SaaS existentes no mercado, mas também pela introdução às categorias de serviços usados no sector do alojamento; a análise e especificação das APIs realizada, bem como a *interface* comum definida, será também uma ferramenta muito útil para a fase de desenvolvimento dos processos de integração e utilização de serviços SaaS; a produção de dois entregáveis, documentos que serão entregues ao cliente como resultados da evolução do projecto; e a selecção e priorização, de um ponto de vista técnico, dos serviços SaaS a serem integrados.

Na fase futura de implementação do projecto *Hotelcracy Apps*, os componentes desenvolvidos para a prova de conceito da subscrição e cancelamento de serviços SaaS poderão ser utilizados como ponto de partida para os restantes processos da plataforma e serão mesmo utilizados para subscrição e cancelamento de serviços. Graças à *suite* de testes automáticos de alta cobertura que foi implementada, será possível realizar testes de regressão verificando a integridade da primeira versão dos componentes desenvolvida no estágio que agora termina.

Referências

[1] P. Mell e T. Grance, «The NIST Definition of Cloud Computing». U.S. Department of Commerce, Set-2011.

[2] J. Miranda e B. Almeida, «Relatório de análise de serviços SaaS no sector do alojamento». 30-Mai-2017.

[3] J. Miranda e B. Almeida, «Análise e especificação de APIs de serviços: Channel Manager». 30-Mai-2017.

[4] J. Miranda e B. Almeida, «Análise e especificação de APIs de serviços: Facturação». 12-Mai-2017.

[5] J. Miranda, «Análise preliminar da interface com o utilizador: subscrição e cancelamento de serviços». 17-Mar-2017.

[6] J. Miranda e B. Almeida, «Descrição dos processos automáticos de subscrição, migração e remoção de serviços SaaS». 29-Mar-2017.

[7] «Proposta de Candidatura, Parte B (Anexo Técnico) - Hotecracy Apps», 2015.

[8] HospitalityTechnology, «2016 Lodging Technology Study», 2015.

[9] «property management system». [Em linha]. Disponível em: http://www.allpropertymanagement.com/faq/property-management-system_p69.html. [Acedido: 18-Out-2016].

[10] «What is a Channel Manager? Definition and list of 40 Channel Managers». [Em linha]. Disponível em: http://rentalsunited.com/en/what-is-a-channel-manager.html. [Acedido: 22-Jun-2017].

[11] «Why use a Channel Manager - Hotel Distribution». [Em linha]. Disponível em: http://www.clerkhotel.com/blog/use-channel-manager/. [Acedido: 22-Jun-2017].

[12] «What Types of Inventory Does the Service Industry Use? | Chron.com». [Em linha]. Disponível em: http://smallbusiness.chron.com/types-inventory-service-industry-use-33758.html. [Acedido: 02-Mar-2017].

[13] «Glossary of Hotel Revenue Management Terms | IDeaS». [Em linha]. Disponível em: http://ideas.com/tools-resources/glossary/. [Acedido: 02-Mar-2017].

[14] «Olery Reputation | Olery». [Em linha]. Disponível em: http://www.olery.com/reputation/. [Acedido: 19-Out-2016].

[15] «Hotel Rate Shopping - Hotel Rate Shopper». [Em linha]. Disponível em: http://www.xotels.com/en/revenue-management/revenue-management-book/hotel-rate-shopper. [Acedido: 19-Out-2016].

[16] «Customer Relationship Management - CRM». [Em linha]. Disponível em: http://www.investopedia.com/terms/c/customer_relation_management.asp. [Acedido: 11-Out-2016].

[17] J. Schweisberger e A. Chatterjee, «Small Hotels & CRM», 2002.

[18] M. Torggler, «The Functionality and Usage of CRM Systems», 2008.

[19] «What is an OTA (Online Travel Agency)?» [Em linha]. Disponível em: https://www.rezgo.com/glossary/ota. [Acedido: 02-Mar-2017].

[20] «HowToGuide_BookingEngines_v1-0.pdf». [Em linha]. Disponível em: http://www.failteireland.ie/FailteIreland/media/WebsiteStructure/Documents/2_Develop_Your_Business/3_Marketing_Toolkit/3_Market_Your_Business_Online/Choose%20Your%20Internet%20Tools/HowToGuide_BookingEngines_v1-0.pdf. [Acedido: 11-Out-2016].

[21] «What is Billing Software? - Definition from Techopedia». [Em linha]. Disponível em: https://www.techopedia.com/definition/4234/billing-software. [Acedido: 02-Mar-2017].

[22] «What is point-of-sale terminal (POS terminal)? - Definition from WhatIs.com». [Em linha]. Disponível em: http://whatis.techtarget.com/definition/point-of-sale-terminal-POS-terminal. [Acedido: 21-Out-2016].

[23] «What Is A Point of Sale System? A Guide to POS Features». [Em linha]. Disponível em: http://www.softwareadvice.com/resources/what-is-a-point-of-sale-system/. [Acedido: 02-Mar-2017].

[24] «The Point of Sale Features Your Hotel Needs | IQware». [Em linha]. Disponível em: http://www.iqwareinc.com/blog/point-sale-features-hotel-needs/. [Acedido: 18-Out-2016].

[25] «Terminal de Pagamento Automático: Como Funciona? - Economias». [Em linha]. Disponível em: https://www.economias.pt/terminal-de-pagamento-automatico-como-funciona/. [Acedido: 16-Nov-2016].

[26] «Conheça as funcionalidades do Infraspeak». [Em linha]. Disponível em: http://home.infraspeak.com/pt-pt/funcionalidades/. [Acedido: 16-Nov-2016].

[27] «Mobile Concierge App - Hotelgenius». [Em linha]. Disponível em: https://www.hotelgenius.co/index.php/mobile-concierge-app/. [Acedido: 16-Nov-2016].

[28] «Recursos Humanos| Software de Gestão| ARTSOFT». [Em linha]. Disponível em: http://www.artsoft.pt/gestao-rh. [Acedido: 16-Nov-2016].

[29] «Online Employee Scheduling Software and Time Clock | When I Work». [Em linha]. Disponível em: http://wheniwork.com/. [Acedido: 16-Nov-2016].

[30] «Gestão financeira: conheça as vantagens ao usar um software». [Em linha]. Disponível em: https://blog.contaazul.com/vantagem-software-gestao-financeira/. [Acedido: 16-Nov-2016].

[31] «MagniFinance - Home Page». [Em linha]. Disponível em: http://www.magnifinance.com/. [Acedido: 16-Nov-2016].

[32] «Google». [Em linha]. Disponível em: https://www.google.pt/. [Acedido: 29-Jun-2017].

[33] «User Stories: An Agile Introduction». [Em linha]. Disponível em: http://www.agilemodeling.com/artifacts/userStory.htm. [Acedido: 05-Jan-2017].

[34] «User Stories and User Story Examples by Mike Cohn». [Em linha]. Disponível em: https://www.mountaingoatsoftware.com/agile/user-stories. [Acedido: 05-Jan-2017].

[35] M. Cohn, *User Stories Applied - For Agile Sotware Development*. US,: The Addison Wesley Signature Series, 2004.

[36] «Writing User Stories, Examples and Templates In Agile Methodologies – Yodiz Project Management Blog». [Em linha]. Disponível em: http://www.yodiz.com/blog/writing-user-stories-examples-and-templates-in-agile-methodologies/. [Acedido: 16-Jan-2017].

[37] «BPMN Specification - Business Process Model and Notation». [Em linha]. Disponível em: http://www.bpmn.org/. [Acedido: 16-Jan-2017].

[38] «What is BPEL (Business Process Execution Language)? - Definition from WhatIs.com». [Em linha]. Disponível em: http://searchmicroservices.techtarget.com/definition/BPEL-Business-Process-Execution-Language. [Acedido: 01-Jul-2017].

[39] M. Fargnoli, «EPC vs BPMN: Reviewing Modelling Notations», *Leonardo Consulting*, Australia, 2015.

[40] S. Brown, *Software Architecture for Developers - Visualize, document and explore your software architecture*, vol. 2, 2 vols. Leanpub, 2016.

[41] «Structurizr - Help - The C4 software architecture model». [Em linha]. Disponível em: https://www.structurizr.com/help/c4. [Acedido: 22-Jun-2017].

[42] Hotelcracy, «Definição de personas e cenários de utilização». 27-Fev-2017.

[43] «Rails has everything you need | Ruby on Rails». [Em linha]. Disponível em: http://rubyonrails.org/everything-you-need/. [Acedido: 22-Jun-2017].

[44] J. Deacon, «Model-View-Controller (MVC) Architecture», *Computer Systems Development, Consulting & Training*, 2009.

[45] «RubyMonk - Interactive Ruby tutorials». [Em linha]. Disponível em: https://rubymonk.com/. [Acedido: 23-Jun-2017].

[46] D. Heinemeier Hansson, «Doctrine | Ruby on Rails», 2016. [Em linha]. Disponível em: http://rubyonrails.org/doctrine/. [Acedido: 23-Jun-2017].

[47] M. Hartl, *The Ruby On Rails Tutorial: Learn Web Development with Rails*, 4ª. US: Addison-Wesley Professional Ruby Series, 2016.

[48] K. Beck, *Test-Driver Development by Example*. US: The Addison Wesley Signature Series, 2002.

[49] «Why does Kent Beck refer to the "rediscovery" of test-driven development? - Quora». [Em linha]. Disponível em: https://www.quora.com/Why-does-Kent-Beck-refer-to-the-rediscovery-of-test-driven-development. [Acedido: 28-Jun-2017].

[50] «Test Driven Development by Kent Beck | The Pragmatic Bookshelf». [Em linha]. Disponível em: https://pragprog.com/screencast/v-kbtdd/test-driven-development#details. [Acedido: 28-Jun-2017].

[51] «RSpec: Behaviour Driven Development for Ruby». [Em linha]. Disponível em: http://rspec.info/. [Acedido: 28-Jun-2017].

[52] «What is Capistrano?» [Em linha]. Disponível em: http://capistranorb.com/documentation/overview/what-is-capistrano/. [Acedido: 28-Jun-2017].