Mestrado em Engenharia Informática
Estágio
Relatório Intermédio

Plataforma de Gestão de Pagamentos para Redes de Turismo

Gil Valente Martins Hilário
ghilario@student.dei.uc.pt

Orientadores:
Prof. Dr. Henrique Santos do Carmo Madeira
Eng. Ricardo Machado

2 de Setembro de 2015

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

Resumo

Este trabalho tem como principal objetivo desenvolver um sistema para faturação e gestão de pagamentos, na forma de uma Interface API que faça a integração da plataforma SmartCityAPI com o *software* de faturação escolhido para fazer parte da solução.

No mundo do turismo as novas tecnologias têm cada vez mais lugar de destaque. A massificação dos *smartphones* e *tablets* veio trazer um enorme leque de novas oportunidades de interação entre o turista e o meio que o envolve. O turista está mais exigente e possui uma grande quantidade de informação à distância de um toque.

O projeto SmartCity API foi criado com o objetivo de centralizar toda a oferta e procura dos serviços prestados aos turistas. Devido à multiplicidade destes prestadores de serviços a complexidade do sistema cresce, principalmente na gestão dos pagamentos e faturação a todos os intervenientes no negócio.

Pretende-se no âmbito deste estágio que seja desenvolvido o módulo responsável pela gestão da faturação e dos pagamentos. Este módulo será responsável por realizar toda a faturação e pagamentos entre os turistas e as entidades prestadoras de serviço bem como a gestão do fluxo monetário pelas diversas entidades.

A gestão de clientes, produtos e fornecedores de serviços também é parte integrante do trabalho a realizar. No desenvolvimento do módulo em questão, será dada especial ênfase a ferramentas *OpenSource*.

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

API Application Programming Interface

CRM Customer Relationship Management

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

JSP JavaServer Pages

MVC Model View Controller

ORM Object-Relational Mapping

PCI DSS Payment Card Industry Data Security Standard

PDF Portable Document Format

RDBMS Relational Database Management System

RPC Remote Procedure Call

SAFT Standard Audit File for Tax

SQL Structured Query Language

XML Extensible Markup Language

Capítulo 1

Introdução

No presente relatório está documentado o trabalho realizado durante o período de estágio na empresa Ubiwhere no âmbito da obtenção do grau de Mestre em Engenharia Informática pela Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Estão visados neste documento os vários processos do trabalho desde o momento de contextualização do tema até á implementação da solução e seus testes, passando por processos de analise de requisitos, análise de riscos, planeamento do trabalho, conclusão e trabalho futuro.

Este primeiro capítulo tem como objetivo dar a conhecer a entidade acolhedora, o âmbito do projeto e os objetivos gerais que o motivam.

1.1 Entidade Acolhedora

A Ubiwhere é uma empresa fundada em 2007, em Aveiro, que desenvolve soluções inteligentes com as mais recentes tecnologias, faz investigação e desenvolvimento personalizado e tem como missão melhorar a vida das pessoas através do desenvolvimento de tecnologias com usabilidade.

Em 2010 conseguiu o certificado ISO 9001 pelo seu sistema de gestão de qualidade e também o certificado NP 4457 na área da investigação e desenvolvimento. É uma empresa que abrange diversas áreas, sendo que se destacam as *Smart Cities*, Internet do Futuro e Telecomunicações.

Reconhecida como uma das 100 melhores empresas para trabalhar em Portugal em 2014 segundo o estudo realizado pela Revista Exame/Accenture.

1.2 Contexto

1.2.1 Smart Tourism

O conceito de *Smart Tourism*[2] (ou *Digital Tourism*) é relativamente recente e teve origem devido à emergência das Smart Cities. Pode ser interpretado como o uso intensivo da infraestrutura tecnológica de uma *Smart City* com o objetivo de melhorar a experiência de turismo aos visitantes tornando-os conhecedores tanto da oferta local e turística de produtos e serviços disponíveis nessa cidade. Mas também pode representar o aumento de capacidade de decisão das empresas relacionadas com o turismo através do acesso à informação produzida na cidade, informação essa recolhida e gerida pelas infraestruturas tecnológicas.

O termo *Smart Cities* é aplicado a cidades que usem as tecnologias da informação para aumentar a eficiência e a inteligência com que fazem a gestão dos seus recursos, resultando em poupanças de energia e custos, melhoria dos serviços prestados e uma menor pegada ambiental.

Recentemente Tel Aviv[3] foi a galardoada com o prémio de melhor cidade no *Smart City Expo World Congress* 2014, que se realizou em Barcelona. Foi escolhida devido ao extraordinário compromisso feito pelas autoridades municipais em modernizar a forma de se relacionar com os seus cidadãos, fazendo uso das potencialidades das tecnologias de geolocalização fornecidas pela extensa rede de Internet sem fios que cobre a cidade, fornecendo informação desde obras na estrada nas imediações como serviços disponíveis e promoções consoantes o perfil do habitante.

O projeto no qual será inserida a solução desenvolvida neste trabalho, faz parte do desenvolvimento das *Smart Cities* em Portugal.

1.2.2 SmartCity API

A SmartCityAPI consiste num sistema orquestrador que fará a ponte entre o *Frontoffice* e o *Backoffice* e que dê suporte a um novo modelo de negócio de promoção turística de uma região. Através da disponibilização de uma API aberta a diversas entidades, irá aumentar o número de canais de distribuição da oferta turística. O orquestrador terá o papel central de controlar todo o fluxo de informação circulante entre todas as componentes do sistema, implementar toda a lógica de negócio, permitir a gestão de todos os componentes do sistema e disponibilizar uma API pública para que seja possível no futuro o desenvolvimento de serviços por parte de terceiros, sem que estes se tenham de preocupar com a gestão da oferta turística, conteúdos e gestão financeira de aquisição das ofertas.

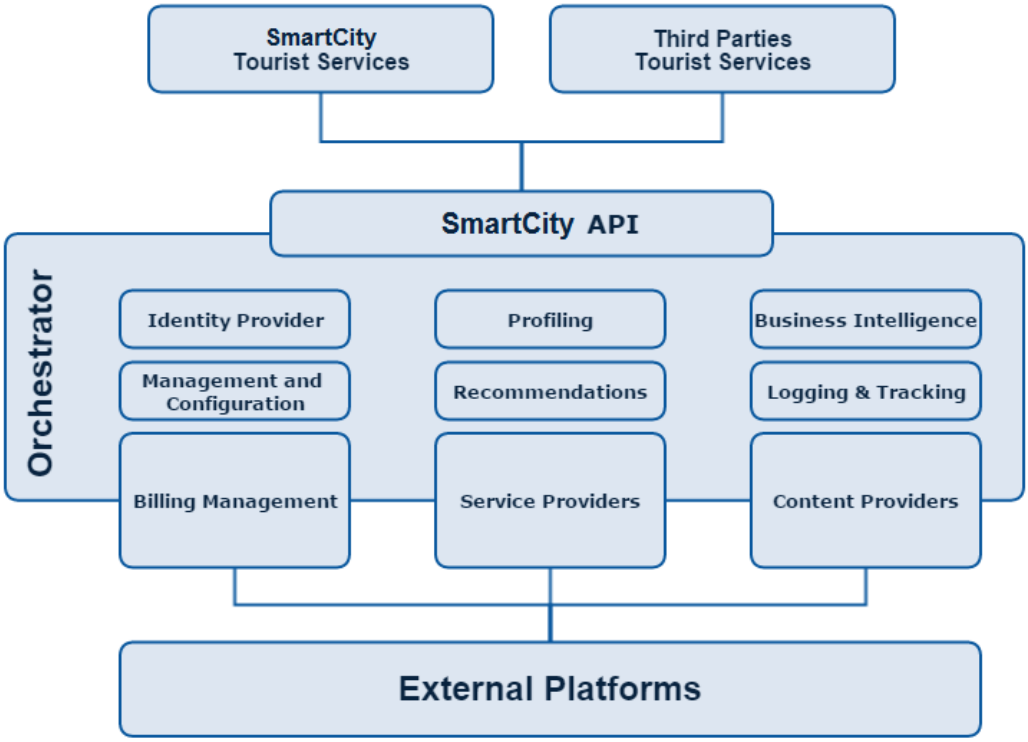


Figura 1.1: Arquitetura geral da SmartCity API

A arquitetura geral do sistema está ilustrada na figura(1.1) onde se podem observar os seguintes módulos chave:

- ***Identity Provider*** - Módulo de gestão da Identidade;
- ***Service Providers*** - Plataformas que disponibilizem serviços aos turistas;
- ***Content Providers*** - Plataformas que disponibilizem conteúdos;
- ***Billing Management*** - Sistema de gestão de pagamentos central, que será desenvolvido neste estágio.
- ***Recommendations Service*** - Serviço de recomendações de serviços e conteúdos aos turistas;
- ***Logging and Tracking*** - Componente de registo de ações;
- ***Profiling*** - Identificação de perfis dos utilizadores;

- ***Business Intelligence*** - Tratamento da informação gerada pelo sistema e apresentação de estatísticas avançadas de apoio à decisão;
- ***Management and Configuration*** - Módulo de Gestão e Configuração de parâmetros e *workflow* de processos de todo o sistema que estará disponível através de *Backoffice* web.

1.3 Objectivos

No âmbito deste estágio, será desenvolvido o módulo de faturação referente á plataforma SmartCityAPI que terá os principais objetivos:

- Gestão dos clientes.
- Gestão das fornecedores.
- Gestão dos produtos.
- Gestão da disponibilidade dos serviços e ofertas.
- Registo e gestão dos pagamentos
- Controlo do fluxo monetário entre as diversas entidades.
- Disponibilização dos dados através de uma API de Web Services

A possível integração com outros sistemas de apoio ao negócio, tais como, CRM e suporte de vendas e geração de relatórios também deve ser tido em conta no desenvolvimento da solução.

Capítulo 2

Estado da Arte

2.1 Faturação Eletrónica

A faturação é um processo de extrema importância para qualquer empresa, é através deste que os clientes recebem as faturas e realizam os respetivos pagamentos, entre outras tarefas.

A faturação eletrónica distingue-se da tradicional faturação devido à utilização dos meios digitais para agilizar os processos e poupar recursos, desde a criação de uma fatura até à confirmação do pagamento da mesma.

2.1.1 Tradicional vs Eletrónica

Na última década a crescente consciencialização em relação à escassez de recursos no planeta e os avanços tecnológicos trouxeram grandes reduções nos consumos energéticos e de recursos em diversas áreas. Os processos de faturação não foram exceção, e juntamente com a cimentação do comércio eletrónico foram adotados os primeiros sistemas de faturação eletrónica.

As principais vantagens[4] e desvantagens da adoção do sistema de faturação eletrónica estão mencionadas abaixo:

Vantagens

- **Redução de custos** - A mudança para o sistema eletrónico permite tanto às empresas como aos clientes reduzir os custos diretamente ligados às impressões em papel, e também à distribuição física dessas mesmas faturas impressas.
- **Celeridade no pagamento** - A faturação eletrónica pode fazer com que o número de dias que um cliente demora a pagar uma conta seja

drasticamente reduzido. As principais causas são a quantidade de passos eliminados no processo de faturação, fazendo não só com que a fatura chegue mais rapidamente ao cliente, mas também que o tempo a efetuar o pagamento diminua. A Striata[5] concluiu com base num relatório[6] que em média o processo tradicional tem como tempo médio de pagamento 45 dias, ao invés de que no sistema eletrónico 18% dos clientes paga em 1 dia, 41% entre 2-7 dias, 17% entre 8-15, 20% entre 16-30 dias e finalmente só 3% leva mais de 30 dias a efetuar o pagamento. Estas reduções têm um enorme impacto nas operações e orçamento das empresas.

– **Razões Ambientais** - A eliminação de apenas uma fatura em cada mês durante um ano é o suficiente para poupar quase 2 Kg em papel, prevenir cerca 185 litros de águas residuais de entrarem no lagos e rios e poupar ainda 01.5 litros de gás. Aliando a estas medidas toda a imagem positiva gerada para os clientes e incentiva-os a adotar também a faturação eletrónica.

Desvantagens

– **Velhos Hábitos** - Um dos maiores desafios para os sistemas de faturação eletrónica será a substituição efetiva dos métodos tradicionais, pois muitos dos casos que já fazem uso da faturação eletrónica ainda mantêm os antigos sistemas em simultâneo. As empresas recebem os pagamentos mais rapidamente mas ainda assim nem as consequências ambientais são atenuadas nem os gastos da empresa com o papel reduzem significativamente.

– **Custo Inicial** - As reduções de custo a médio e longo prazo são evidentes e muito vantajosas para todas as partes mas não podem ser ignorados os custos inerentes à mudança e adoção de um novo sistema de faturação. Desde investimento em software para montar o sistema, as campanhas para sensibilizar para a mudança e por fim as taxas que algumas empresas cobram para o pagamento eletrónico das faturas podem dificultar a transição para a faturação eletrónica.

– **Alcance das Tecnologias** - As discrepâncias entre utilizadores no que toca a acesso a, uso de e conhecimento sobre as tecnologias da informação, tenham elas raiz sócio-económica, na idade dos utilizadores ou noutros fatores levam a que não seja possível a todos adotar este sistema de faturação. Em Portugal[7] em 2013 cerca de 40% da população ainda não tinha acesso à Internet no seu lar.

2.1.2 Situação Atual

Para ter uma melhor ideia da dimensão do mercado da faturação a nível mundial e o impacto da faturação eletrónica pode ser consultada a tabela 2.1, onde estão representados o valor total da faturação mundial, a sua percentagem referente à faturação eletrónica e o seu aumento.

Segmento do Destinatário	**Volume anual de faturas enviadas estima-se ser pelo menos**		**Faturação eletrónica proporção estimada do volume total**		**Faturação eletrónica aumento percentual estimado**	
	Mundo	**Europa**	**Mundo**	**Europa**	**Mundo**	**Europa**
Consumidor	330 biliões	18 biliões	exceder 8%	14%	20%	15%
Empresas e Governos	170 biliões	17 biliões		24%		22%

Tabela 2.1: Faturação Eletrónica no Mundo e Europa.[1]

Embora ainda não seja um conceito muito desenvolvido em todo o mundo, já se encontram países onde as taxas de adoção dos sistemas de faturação eletrónica atingem números significativos. O panorama mundial encontra-se ilustrado na figura 2.2 onde se pode verificar que não existe ainda um padrão muito consistente de maturidade nos sistemas de faturação eletrónica.

Portugal está representado como sendo um dos países com a denominação de líderes devido principalmente à nova legislação, em vigor desde Janeiro de 2014, que veio aumentar o número de empresas obrigadas a utilizar programas de faturação certificados.

2.1.3 Softwares de Faturação e Pagamentos Online

Existem variadíssimas soluções no mercado para faturação de todo o tipo de empresas, com diversos tipos de complementações e para todas as carteiras. Devido aos requisitos não funcionais deste projeto, vão ser focados somente os *softwares open source*, evitando os custos inerentes à compra/subscrição desse mesmo software e permitindo assim o desenvolvimento de extensões para uma melhor integração na solução a desenvolver.

Tal como já foi referido anteriormente os *softwares* de faturação eletrónica não se limitam a emitir faturas e a gerir o seu pagamento. Abaixo listadas estão os principais tarefas que um *software* desta natureza deve permitir realizar[8]:

– **Provisão** - Corresponde ao processo de entregar o produto ao cliente. Mais presentes nos serviços de telecomunicações, onde através do sis-

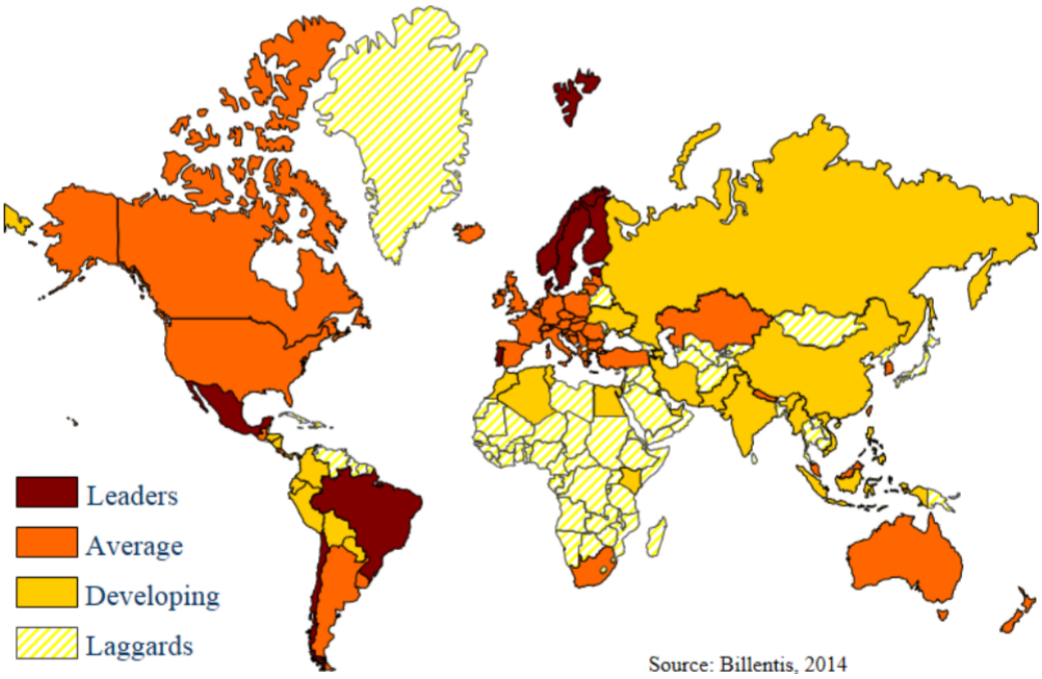


Figura 2.1: Maturidade do mercado da faturação eletrónica.[1]

tema é possível libertar serviços como por exemplo retirar o bloqueio a um telemóvel que não possa efetuar chamadas por não ter saldo, no ato do pagamento é desencadeado o processo de provisão do serviço.

- **Mediação** - A capacidade de gerir todo o sistema sem conflitos, registando e mantendo uma única fonte de verdade, não permitindo que documentos chave sejam alteradas após a sua emissão e mantendo registo de todas as atividades registadas.
- **Avaliação** - O sistema que permite avaliar a quantidade de um produto que foi utilizado para que seja possível cobrar consoante essa avaliação.
- **Faturação** - O método de saber e declarar exatamente o que é devido à empresa, e consequente notificação ao cliente.
- **Cobrança** - Tarefa que é responsável por coletar o pagamento dos clientes, frequentemente conseguido através da integração com *gateways* de pagamentos.
- **Ajustes** - Sistema de prevenção para exceções singulares ao decorrer normal do funcionamento do programa.

– **Relatórios** - Capacidade de gerar relatórios com base na informação recolhida pelo sistema.

Após a realização da análise do mercado de *softwares* de faturação *open-source*, foram excluídos os softwares que se encontram sem contribuições de relevo pelo menos nos últimos dois anos (CitrusDB[9], phpCOIN[10], Agilebill[11]) e os que dispõem de grandes limitações nas suas versões gratuitas (Projecto Colibri[12]). Em seguida serão apresentados os *softwares* que demonstraram maior capacidade de fazer parte da solução e sua análise comparativa.

JBilling

O JBilling[13] teve origem em 2002 quando Emiliano Conde, o fundador, pensou que poderia melhorar o mundo faturação, juntou a sua equipa e começaram a escrever as primeiras linhas de código.

A missão da empresa é fornecer uma alternativa *open source* ao software de faturação empresarial que seja segura e robusta. Foram a primeira empresa da área a disponibilizar uma solução *open source* e continuam a fazê-lo sendo agora líderes do mercado de sistemas de faturação *open source*.

Conta com clientes nas mais variadas áreas, tais como, Constant Contact (Markting), FonAngle (Telecomunicações) e Silent Partner (Virtual Office). Em Portugal está presente através de uma parceria com a empresa Innowave que utilizou o JBilling para desenvolver uma solução de faturação e coleta das portagens em auto-estradas portuguesas, garantindo escalabilidade e uma rápida implementação (cerca de dois meses) do projeto que neste momento faz o registo e gestão de cerca de 20 mil registos de tráfego diariamente.

O JBilling disponibiliza 4 versões, são elas a *Telco Hosted, Telco, Enterprise e Community,* sendo que apenas a ultima é de licença comercial gratuita. As limitações incidem mais na área da mediação (especialmente para mercados das telecomunicações), disponibilização da API para integrar com outras aplicações e na personalização de preços (Ex:pacotes de produtos e descontos temporários).

O JBilling está escrito em Java (JBilling - Java Billing) permite que sejam desenvolvidos os nossos próprios *plugins* para serem posteriormente integrados, permitindo assim todo o tipo de personalização ao sistema. Existem já diversas integrações desenvolvidas que incidem essencialmente sobre as principais áreas dos sistemas de suporte ao negócio. Neste campo as mais relevantes são referentes aos sistema de gestão de clientes e à aplicação de impostos, sendo que integram os *softwares* SugarCRM e SureTax respetivamente.

Freeside

A Freeside[14] nasceu em 1996 como uma aplicação interna de um fornecedor de acesso à Internet, mais tarde após obtenção de licença sobre o código desenvolvido, Ivan Kohler o ex-trabalhador dessa empresa lançou assim a primeira versão *open source* deste *software*. Em 2005 a Freeside Internet Services Inc. foi incorporada e finalmente em 2009 Jeremy Davis juntou-se à equipa como diretor de suporte.

Neste momento a empresa oferece serviços de consultoria, hardware pré-configurado, suporte ao cliente, personalização do produto e treino para utilizadores do software de faturação Freeside e outros aspetos das operações relacionadas com fornecedores de acesso à Internet.

O sistema, escrito em Perl, está mais focado para o mercado das telecomunicações, permitindo fazer a gestão de diversas áreas do negócio, tais como, faturação, CRM, bilhética, monitorização da rede e provisão automática de *software*. Permite também que sejam escritos *plugins* para obter personalização.

Kill Bill

O Kill Bill[15] começou a ser desenvolvido por Pierre-Alexandre Meyer e Stéphane Brossier em 2010, e só em 2013 conseguiu a sua primeira licença e foi lançada a primeira versão. A sua missão é fornecer uma solução *open source* que seja robusta e flexível para gestão de faturação e pagamentos que já é adotada por algumas empresas.

O software foi pensado desde o seu início para ser *open source* e gratuito, fomentando a capacidade de personalização, integração noutras aplicações, utilização diversa de *plugins* e possibilidade de interação com a API através de várias linguagens (ao contrário do JBilling que só permite que o cliente seja uma aplicação em Java).

Os clientes principais que já usam este recente software são o Groupon, SafeKiddo e Ning. O que prova que embora esse seja um produto que ainda não está no mercado há muito tempo já se encontra num estado considerável de robustez.

2.1.4 Análise Comparativa

Posteriormente à primeira análise de onde foram selecionados três candidatos principais para fazer parte da solução a desenvolver, foi feita uma análise comparativa dos mesmos. Na tabela 2.2 estão representadas as características principais a ter em conta para o desenvolvimento deste projeto e as respetivas

capacidades de cada um destes programas.

Representadas a verde estão as características que se encontram presentes e disponíveis, a vermelho as que não se encontram presentes e por fim a amarelo encontram-se situações mais singulares que serão abordadas individualmente na análise escrita abaixo.

O Freeside é a ferramenta, das três, que há mais tempo está no mercado (desde 1998), tendo inclusive ganho por três vezes o prémio de *"Best of Show"* na ITExpo (East 2009, East 2010 e West 2013). Por outro lado contém três grandes desvantagens que levam com que seja excluída deste projeto, tais como, a enorme complexidade de fazer a instalação e configuração de todo o sistema.

O Kill Bill é um projeto consideravelmente recente, em desenvolvimento desde 2010, que ainda tem uma comunidade de pequena dimensão e que não está certificado pela autoridade tributária e aduaneira, ao contrário do JBilling.

Ambas as plataformas possuem quase todas as características chave para o desenvolvimento da solução pretendida neste estágio, mas o facto de o Jbilling já ter sido certificado para faturação pela autoridade tributária e aduaneira, possuir uma maior comunidade e inclusive uma empresa parceira oficial em Portugal, levaram a que o software de faturação escolhido, numa primeira instância, tenha sido o JBilling.

		JBilling	Freeside	Kill Bill
Características Gerais	Interface Visual (GUI)	x	x	x
	Multi-Moeda	x	x	x
	Multi-Língua	x	x	x
	Auditabilidade	x	x	x
	Escalável	x	x	x
	Certificado pela Autoridade Tributária	x	-	-
Gestão de Utilizadores	Hierarquia de Contas	x	x	x
	Grupos	x	x	x
	Clientes	x	x	x
	Fornecedores	-	x	x
	Personalização de Atributos	x	x	x
Gestão de Produtos	Categorização	x	x	x
	Promoções	-	x	x
	Subscrições	x	x	x
	Agrupamento de Produtos	-	x	x
Gestão de Faturas	Criação Automática de Faturas	x	x	x
	Suporte de Impostos Portugueses	x	x	x
	Personalização da Fatura	x	x	x
	Envio de Fatura via E-mail	x	x	x
Gestão de Pagamentos	Ciclo de Pagamentos Automáticos	x	x	-
	Suporte para Cartão de Crédito	x	x	x
	Suporte para Moedas Virtuais	x	x	x
	Sistema de Tratamento de Falhas	x	x	x
	Sistema de Reembolso	x	x	x
Integrações	Arquitetura baseada em *Plugins*	x	-	x
	Disponibilização de API	-	x	x
	Suporte para Processadores de Pagamentos	x	x	x
	Linguagem para Aplicações Cliente	Java	PHP	Java, PHP ou Ruby
Segurança	Conformidade com a DSS PCI	x	x	x
	Controlo de Acesso Baseado em Contas	x	x	x

Tabela 2.2: Análise comparativa dos softwares de faturação eletrónica

2.2 Pagamentos Eletrónicos

Hoje em dia a realização de pagamentos entre clientes e comerciantes já sofreu diversas alterações desde os modelos mais tradicionais de troca direta. As tecnologias permitem agora um maior número de alternativas para realizar essas transações.

São considerados pagamentos eletrónicos aqueles onde não existe a troca monetária física entre comprador e vendedor. Em seguida estão enumerados os principais métodos de pagamento eletrónico[16]:

- **Cartão de Crédito/Débito** - Em Portugal é um meio de pagamento extremamente comum e consiste num cartão de plástico com uma banda magnética, alguns contêm um chip, com informação do detentor do cartão, o numero da cartão e um código de segurança que são necessários quando uma compra é realizada. É utilizado tanto em compras online como no comércio tradicional.

- **Transferência Bancária** - Existem duas categorias principais, as transferências em tempo real, realizadas online, ou as *offline* onde o cliente usa um código de referência gerado pelo comerciante para efetuar o pagamento através do seu banco online ou se uma máquina Multibanco.

- **Débito Direto** - São movimentos pré autorizados que retiram automaticamente os montantes das contas bancárias dos clientes, este método só é vulgarmente utilizado em pagamentos regulares e previsíveis.

- ***E-Wallets*** - É o método de pagamento em maior crescimento no mundo, neste momento, e é muito utilizado no comércio eletrónico, onde se verifica um grande domínio do Paypal como principal fornecedor deste tipo de serviço. Serve de ferramenta para pagamento bem como, em alguns comerciantes, de comprovativo de idade ou morada, visto que requer que se forneça identificação para criar uma *e-wallet*.

- **Pagamentos Mobile** - Utilizado muito para comprar aplicações em *smartphones*, este método de pagamento usa o provedor de serviços móveis como intermediário para realizar o pagamento, sendo que o pagamento é feito através do saldo corrente ou colocado numa mensalidade futura consoante o contrato do cliente. Já vão surgindo as primeiras *mobile-wallets* mas ainda sem grande adoção por parte do público.

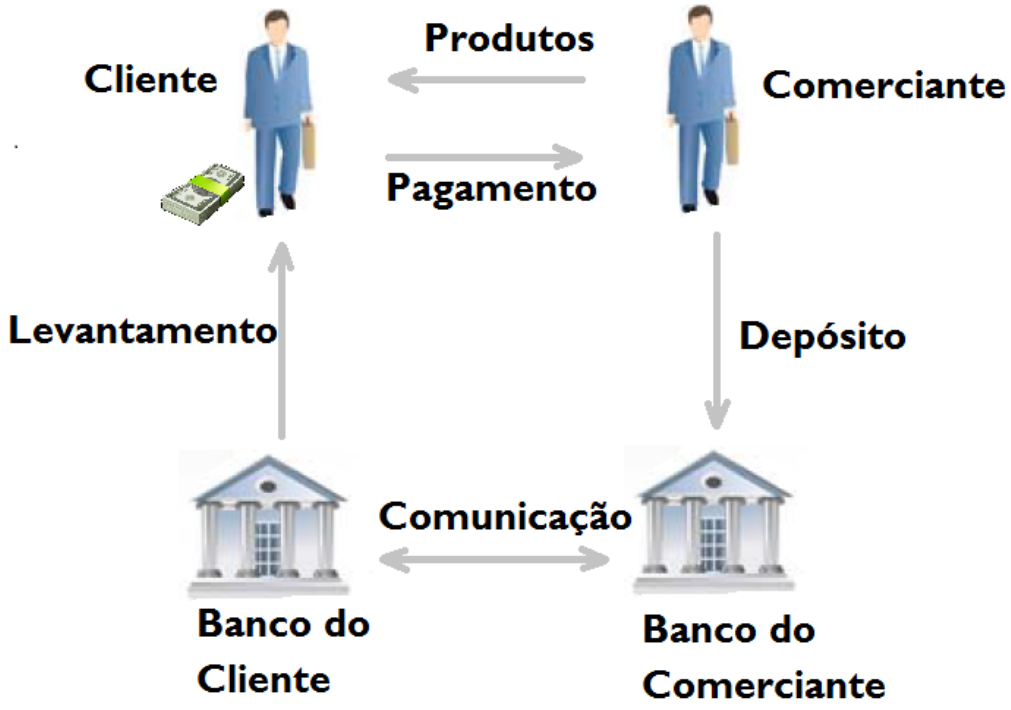


Figura 2.2: Ilustração de um exemplo do funcionamento dos pagamentos tradicionais.

2.2.1 Tradicional vs Eletrónica

As figuras 2.2 e 2.3 ilustram os diferentes processos de pagamento. Podemos observar que no pagamento tradicional o principal contacto no ato de pagamento é entre o cliente e o comerciante e que não existe um grande envolvimento de outras entidades no processo, neste caso os bancos têm uma função de armazenamento dos fundos.

Por outro lado no caso dos pagamentos eletrónicos este processo é feito através de vários intermediários que garantem a validação e segurança do pagamento. Como é possível verificar na figura 2.3 o cliente fornece a informação do seu cartão de crédito, seja presencialmente ou online, o comerciante executa o pedido de pagamento que é transmitido para o seu banco através dos processadores de pagamentos que posteriormente requerem validação ao emissor do cartão em questão. Uma vez recebida a resposta do emissor essa informação é transmitida de novo para o banco do comerciante e finalmente

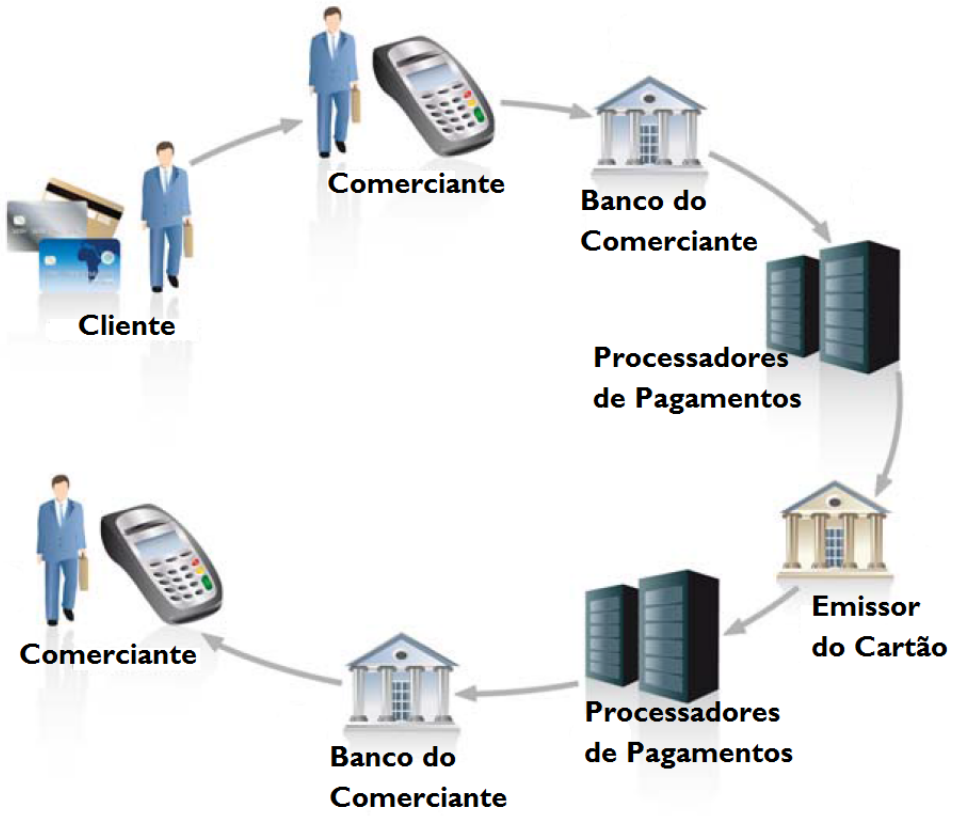


Figura 2.3: Ilustração de um exemplo do funcionamento dos pagamentos eletrónicos.

a confirmação do pagamento chega ao comerciante.

Eliminando assim a necessidade de efetuar os pagamentos no local, dá-se um grande aumento da oferta de produtos para os clientes que os podem adquirir através da Internet e pela mesma razão existe um grande incremento nos potenciais clientes para os comerciantes.

Os pagamentos eletrónicos não foram uma solução com uma grande adesão desde o início devido às reticências por parte dos clientes quanto à segurança e funcionamento de todo o processo. Foram então identificados num estudo[17] os principais fatores que levam os utilizadores à adoção destes sistemas:

– **Segurança** - Um dos, senão o mais importante de todos os fatores, quando se está a realizar operações monetárias é sempre ter extremo

cuidado. É necessário proteger tanto a informação pessoal dos clientes bem como as suas posses, evitar ataques de *hackers* e reduzir a as possíveis fraudes. Com a regulação implementada no presente nesta área a segurança é também cada vez melhor.

- **Usabilidade** - A facilidade e intuitividade de interação com o sistema também é um fator a ter em conta. O utilizador tende a retrair-se aos sistemas que se apresentem muito complexos.

- **Confiança** - A separação física entre o cliente e o comerciante cria alguma quebra na confiança no processo. É necessário existir uma confiança mútua entre ambos, para que o cliente acredite que a sua informação pessoal e o seu investimento vai ter o retorno esperado. As empresas recorrem a certificados de terceiros e a mecanismos de autenticação introduzidos por bancos para incrementar essa confiança por parte do cliente.

- **Interoperabilidade** - Este tipo de sistema não pode depender de uma só empresa, exige que exista uma grande quantidade de utilizadores e um dos maiores incentivos é adoção do sistema por diversas entidades como grandes empresas, bancos e o governo.

- **Regulação** - Diversos regulamentos devem ser postos em vigor de maneira a prevenir e punir eventuais casos de roubo, evasão fiscal ou branqueamento de capitais. Com medida deste género o sistema transmite mais confiança e segurança o que atrairá mais utilizadores.

- **Serviços Adicionais** - As empresas dedicadas á realização de pagamentos eletrónicos devem adicionar alguns serviços à sua oferta para a complementar, como por exemplo integração com sistemas de contabilidade ou monitorização do estado dos pagamentos.

Segundo um estudo[18] publicado em Fevereiro de 2014, em Portugal, onde 84% da população possui telemóvel e 32% *smartphone*, o métodos de pagamento eletrónico preferencial é o cartão de crédito/débito, seguido das transferências bancárias e das *e-wallets* como pode ser observado na figura 2.4.

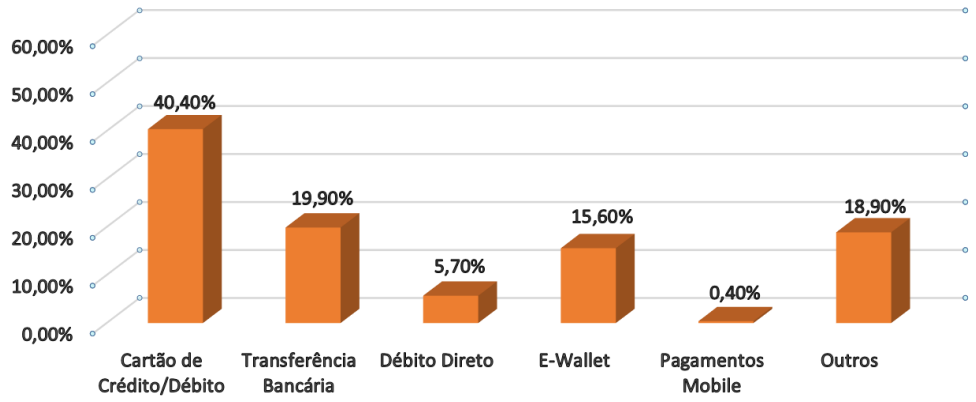


Figura 2.4: Distribuição por Tipo dos Pagamentos Eletrónicos em Portugal.

2.2.2 *Gateways* de Pagamentos

Os *Gateways* de pagamentos são um componente chave nos pagamentos realizadas online, são responsáveis por fazer a transferência de informação entre clientes, comerciantes e bancos. Podem ser equiparados aos terminais de cartão de crédito usados no comércio tradicional.

As informações do cartão de crédito que o cliente fornece são codificadas e transmitidas através do *Gateway* de pagamento que as envia para o banco do cliente com o objetivo de serem confirmadas. É verificado se o cartão é válido e se possui fundos suficientes para que se processe o pagamento, caso se verifiquem estas condições, é retornada a confirmação. O comerciante guarda então esta informação para posteriormente a apresentar ao seu banco e receber os fundos.

As garantias de segurança são dadas a ambas as partes, pois o comerciante tem a confirmação que o pagamento foi bem sucedido, e o cliente sabe que o comerciante não teve acesso direto à sua informação.

Análise comparativa

Na tabela 2.3 encontra-se a análise comparativa dos diversos *Gateway* de pagamento que foram alvo de estudo para integrarem a solução a desenvolver.

Desenvolver um *Gateway* de pagamentos na Europa[24] é de uma complexidade bastante maior do que nos EUA, pois a quantidade de entidades diferentes é muito maior, se tivermos em conta os 28 estados membros da União Europeia, podemos facilmente verificar que gerir bancos, leis e impostos de tantos locais incrementa a dificuldade do processo.

	PayPal[19]	Braintree[20]	Stripe[21]	Paymill [?]	Authorize.Net [22]	Balanced [23]
Processa em Portugal	x	x	-	x	x	-
Multi-Moeda	x	x	x	x	x	x
Multi-Língua	x	x	x	x	x	x
API para Integração	x	x	x	x	x	x
Taxa Cobrada por Transação	3.4% + 0.35	2,9% + 0,30	2.9% + 0,30	2.95% + 0.28	2,9% + 0,30	2.9% + 0,30
Capacidade para Divisão de Pagamentos	-	x	-	-	-	x
Compatibilidade com Cartão de Crédito	x	x	x	x	x	x
Compatibilidade com Bitcoins	x	-	-	-	-	-
Compatibilidade para Pagamentos Mobile	x	x	x	-	x	x
Proteção Contra Fraudes	x	x	x	x	x	x

Tabela 2.3: Análise Comparativa dos *Gateway* de Pagamento.

Por essa razão, ainda são poucos os *Gateway* de pagamentos a atuar em solo Europeu com a mesma destreza e oferta de serviços que se encontram atualmente no continente americano.

Capítulo 3

Metodologia e Planeamento

Neste capítulo, será descrita a metodologia de desenvolvimento adotada durante o estágio, as alterações ao planeamento e a comparação entre o trabalho planeado e o que realmente ocorreu.

A metodologia de desenvolvimento de *software* que foi adotada foi uma adaptação do modelo tradicional em cascata, com alguma lógica iterativa onde fases são bem definidas(figura 3.1).

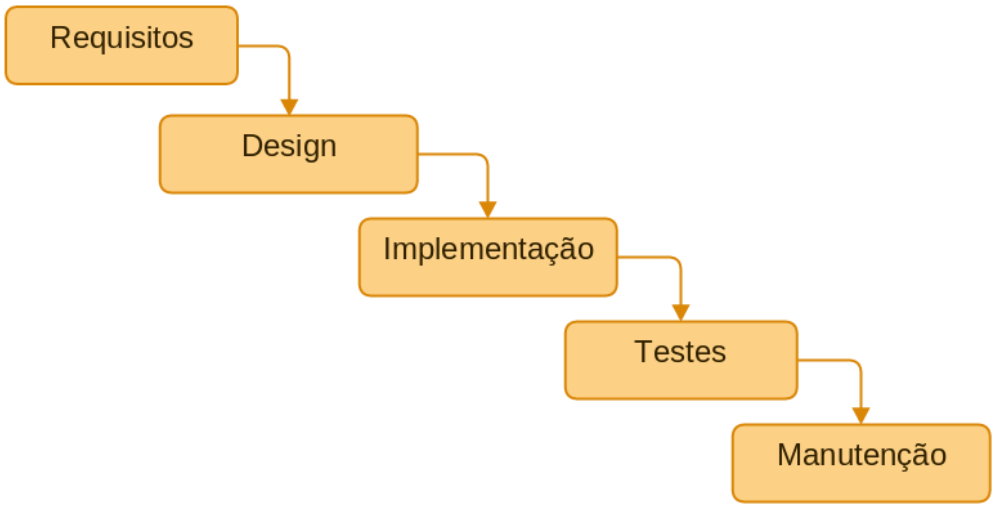


Figura 3.1: Ilustração da metodologia de desenvolvimento em cascata.

3.1 Primeiro Semestre

Durante o primeiro semestre, a alocação horária do estagiário foi de 16 horas semanais.

Abaixo está apresentado o diagrama de Gant onde está representado o plano inicial, com o intuito de pautar o trabalho do estagiário. No mesmo diagrama encontra-se também o registo do trabalho concretizado.

3.2 Segundo Semestre

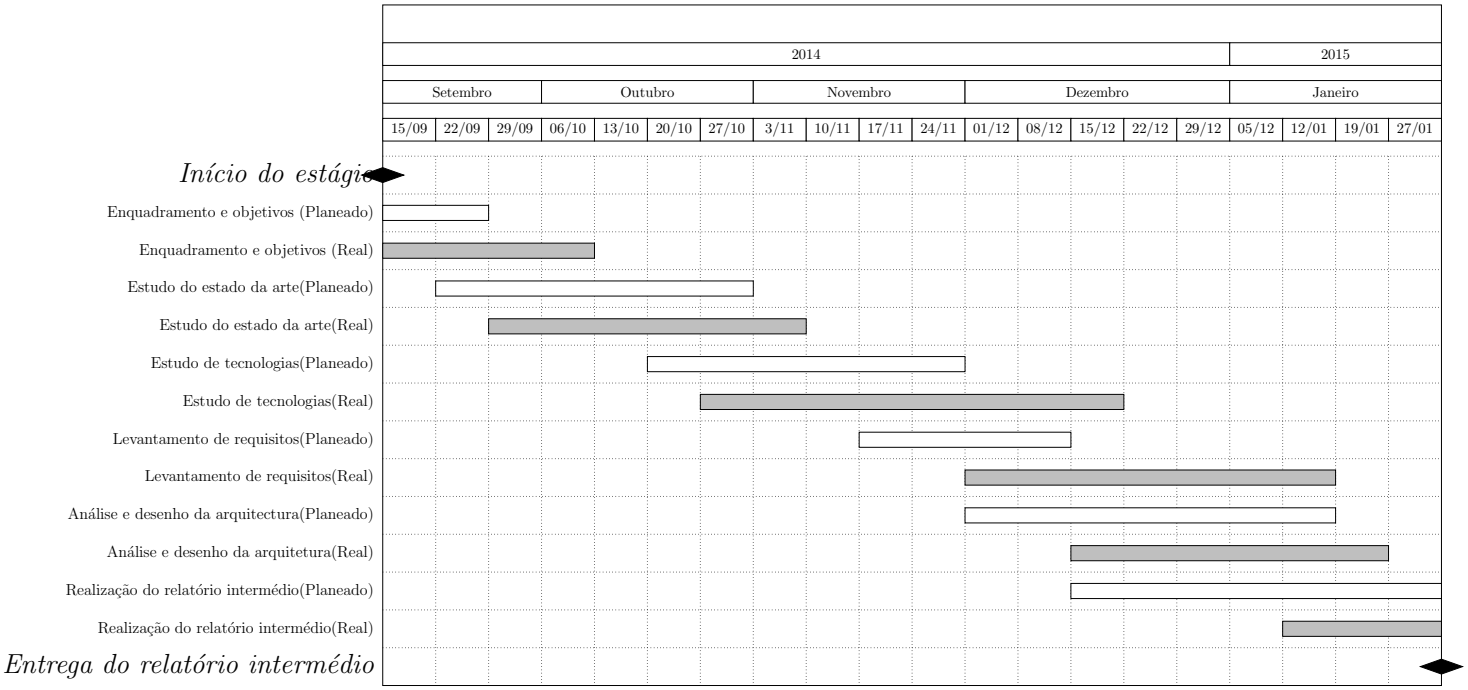
No segundo semestre, a alocação horária do estagiário foi aumentada para as 40 horas semanais. Nem sempre se verificou o seu cumprimento integral, devido apenas às incompatibilidades com o horário das duas unidades curriculares que o estagiário estava a efetuar, em paralelo com o estágio.

3.2.1 Alterações ao Planeamento

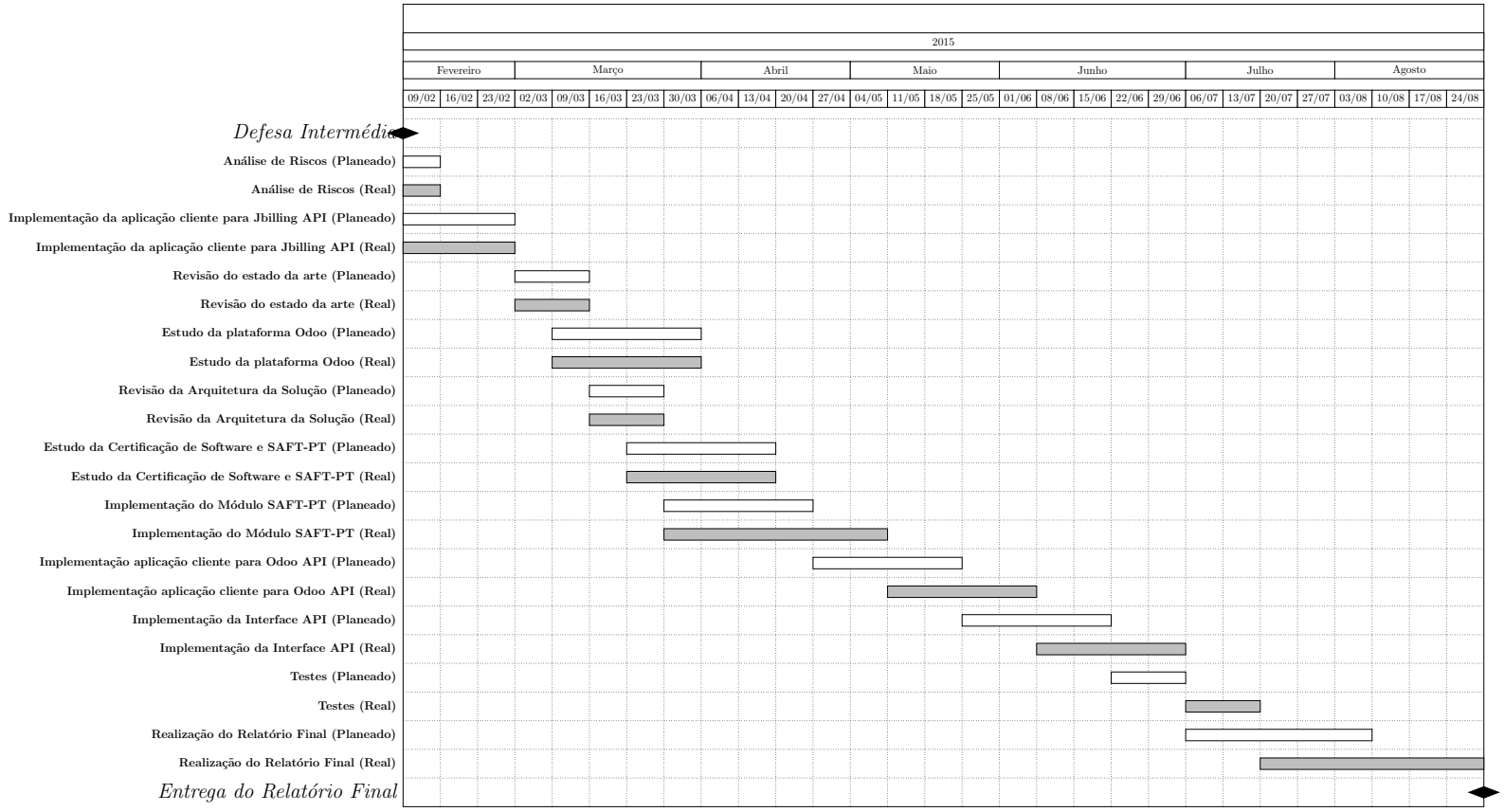
No inicio do segundo semestre, a decisão de mudar de plataforma a adotar para a solução gerou um novo planeamento das atividades. No diagrama de Gant que representa o planeamento do segundo semestre, é possível observar que, nas primeiras três semanas, o estagiário ainda estava a desenvolver a solução com base no jBilling, mas posteriormente existiu uma mudança de paradigma.

Deste modo o planeamento foi elaborado prevendo a entrega em Setembro do relatório de estágio, de forma a garantir que a qualidade do trabalho não seria afetada. Foram consideradas três semanas a separar a data de conclusão do trabalho e a data da sua entrega para eventuais imprevistos.

Planeamento do 1ºsemestre



Planeamento do 2ºsemestre



Capítulo 4

Análise de Requisitos

O processo de definição de requisitos é crucial para o desenvolvimento de um projeto de *software*, devendo dar atenção aos requisitos funcionais bem como aos não funcionais. O levantamento dos requisitos foi feito pelo estagiário, através da análise dos diversos objetivos do projeto articulando-os com o estudo das tecnologias disponíveis, de forma a conseguir priorizar o trabalho a ser desenvolvido. Este levantamento foi complementado com reuniões com o orientador Ricardo Machado, onde foi possível validar os requisitos.

4.1 Requisitos Funcionais

Cada requisito possui um grau de prioridade (Essencial ou Desejável), consoante a sua importância para o projeto. Os requisitos funcionais, identificados na análise efetuada, foram divididos nas seguintes quatro categorias:

- **Administrador** - Representa os requisitos da solução, atingíveis através do acesso à plataforma, com as credenciais de administrador.
- **Cliente** - Indicas as ações passíveis de execução, uma vez autenticado como cliente.
- **Fornecedor** - Semelhante à anterior mas para autenticações com credenciais de fornecedores.
- **API** - Nesta categoria estão representados todos os requisitos que devem ser cumpridos através das chamadas à Interface API desenvolvida para comunicar com o Servidor Odoo, sendo na sua maioria um subconjunto dos requisitos anteriormente apresentados.

Identificador	Requisito	Prioridade
REQ_ADM_01	Criar cliente.	**Essencial**
REQ_ADM_02	Editar Cliente	**Essencial**
REQ_ADM_03	Apagar cliente.	**Essencial**
REQ_ADM_04	Criar fornecedor.	**Essencial**
REQ_ADM_05	Editar fornecedor	**Essencial**
REQ_ADM_06	Apagar fornecedor.	**Essencial**
REQ_ADM_07	Criar produto/serviço.	**Essencial**
REQ_ADM_08	Editar produto/serviço	**Essencial**
REQ_ADM_09	Apagar produto/serviço.	**Essencial**
REQ_ADM_10	Criar encomenda.	**Essencial**
REQ_ADM_11	Editar encomenda	**Essencial**
REQ_ADM_12	Cancelar encomenda.	**Essencial**
REQ_ADM_13	Gerar fatura com base na encomenda.	**Essencial**
REQ_ADM_14	Enviar fatura via email para o cliente	**Essencial**
REQ_ADM_15	Descarregar fatura em pdf	**Essencial**
REQ_ADM_16	Cancelar Fatura	**Essencial**
REQ_ADM_17	Registar pagamento	**Essencial**
REQ_ADM_18	Registar pagamento parcial	Desejável
REQ_ADM_19	Efetuar Pagamentos via PayPal	Desejável
REQ_ADM_20	Consultar relatórios de vendas	Desejável
REQ_ADM_21	Definir comissão nos produtos/serviços	Desejável
REQ_ADM_22	Exportar ficheiro SAFT-PT	Desejável
REQ_ADM_22	Gerir compras aos fornecedores	Desejável
REQ_ADM_22	Gestão de permissões para diferentes utilizadores	Desejável

Tabela 4.1: Requisitos Funcionais de Administrador

Embora muitos deles sejam auto-explicativos, para uma melhor compreensão, cada tabela está acompanhada de uma descrição mais pormenorizada dos respetivos requisitos.

Na **Tabela 4.1**, é possível observar os requisitos que formam a grande base de todo o projeto, assentes nas capacidades da plataforma escolhida à qual é possível aceder através de um *browser*, estando autenticado como administrador.

Como principais funcionalidades a ser disponibilizadas estão a gestão de clientes, fornecedores e produtos. Também de extrema relevância, estão todas as ações referentes ao processo normal de venda e faturação como a

Identificador	Requisito	Prioridade
REQ_CLI_01	Editar sua conta	**Essencial**
REQ_CLI_02	Criar encomenda.	**Essencial**
REQ_CLI_03	Editar encomenda	**Essencial**
REQ_CLI_04	Cancelar encomenda.	**Essencial**
REQ_CLI_05	Gerar fatura com base na encomenda.	**Essencial**
REQ_CLI_06	Receber fatura via email	**Essencial**
REQ_CLI_07	Descarregar fatura em pdf	**Essencial**
REQ_CLI_08	Fazer Pagamento	**Essencial**

Tabela 4.2: Requisitos Funcionais de Cliente

Identificador	Requisito	Prioridade
REQ_FOR_01	Editar sua conta	**Essencial**
REQ_FOR_02	Criar produto/serviço	**Essencial**
REQ_FOR_03	Editar produto/serviço	**Essencial**
REQ_FOR_04	Eliminar produto/serviço	**Essencial**

Tabela 4.3: Requisitos Funcionais de Fornecedor

gestão de vendas, geração de faturas e seu envio, registos e realização de pagamentos. Os requisitos com prioridade desejável incidem maioritariamente em tópicos de auxilio à gestão de negócio, como a geração de relatórios, o controlo das permissões para cada utilizador no acesso à informação presente no sistema e também no processo de certificação de *software*. Refere-se neste caso a exportação de ficheiros SAFT-PT para posterior entrega à Autoridade Tributária e Aduaneira.

Na **Tabela 4.2**, apresentam-se as funções permitidas quando se está na plataforma web da solução, na qualidade de cliente. Enquanto cliente é possível editar a sua informação, gerir as suas encomendas e faturas e efetuar os respetivos pagamentos.

Na **Tabela 4.3** estão as tarefas que os fornecedores poderão realizar na plataforma web, para gestão da sua informação e dos seus produtos, uma vez autenticados com a sua conta de fornecedor.

Na **Tabela 4.4**, estão apresentados os requisitos para a Interface API que faz a ligação entre o servidor Odoo e a SmartCityAPI. Contém vários requisitos já definidos para as categorias anteriores, exigindo que estes sejam cumpridos através dos pedidos à InterfaceAPI. Nem todos os requisitos ante-

Identificador	Requisito	Prioridade
REQ_API_01	Conectar-se ao servidor Odoo	**Essencial**
REQ_API_02	Autenticar utilizador	**Essencial**
REQ_API_03	Criar Cliente	**Essencial**
REQ_API_04	Editar Cliente	**Essencial**
REQ_API_05	Apagar Cliente	**Essencial**
REQ_API_06	Criar Fornecedor	**Essencial**
REQ_API_07	Editar Fornecedor	**Essencial**
REQ_API_08	Apagar Fornecedor	**Essencial**
REQ_API_09	Criar Produto/Serviço	**Essencial**
REQ_API_10	Editar Produto/Serviço	**Essencial**
REQ_API_11	Apagar Produto/Serviço	**Essencial**
REQ_API_12	Criar Encomenda	**Essencial**
REQ_API_13	Editar Encomenda	**Essencial**
REQ_API_14	Cancelar Encomenda	**Essencial**
REQ_API_15	Gerar Fatura com base na Encomenda	**Essencial**
REQ_API_16	Enviar Fatura via email para o Cliente	**Essencial**
REQ_API_17	Cancelar Fatura	**Essencial**
REQ_API_18	Registar Pagamento	**Essencial**

Tabela 4.4: Requisitos Funcionais da API

riormente definidos constam desta tabela, pois alguns são realizados esporadicamente não justificando a sua implementação neste cenário. Acrescentam-se ainda os requisitos referentes à autenticação de clientes e a conexão com as plataformas.

4.2 Requisitos Não Funcionais

Na análise de requisitos também foram identificados os seguintes requisitos não funcionais:

- **Escalabilidade** - A capacidade da aplicação acompanhar o crescimento da sua utilização é crucial, devido ao facto de diversos utilizadores acederem simultaneamente à plataforma, bem como os inúmeros pedidos feitos através da API. O sistema deve ser passível de expansão para acompanhar esse crescimento.

- **Segurança** - A plataforma servirá de gestão para o negócio e como tal terá diversas informações sensíveis que não poderão estar visíveis a

todos utilizadores. De igual modo, a plataforma exige um sistema de controlo de acessos acompanhado de um registo de toda a atividade no sistema, considerando que certas ações só poderão ser realizadas por determinados utilizadores, para o efeito autenticados.

- **Persistência dos Dados** - As alterações feitas no sistema devem ficar registadas. existem diversos documentos que não podem ser alvo de alterações devido a fatores de auditabilidade. Permitindo assim que sejam reconstruidos os cenários que levaram o sistema a evoluir para o seu estado atual.

- ***On-Premises*** - O software deve ser instalado localmente, de forma a garantir independência de terceiros nos capítulos de segurança, disponibilidade do serviço e confidencialidade dos dados alojados.

- ***Open-Source*** - Procurando reduzir custos, obter mais controlo sobre o software e ter a possibilidade de implementar melhorias ou diversas integrações sobre o mesmo, foi restrita a escolha aos *software open-source*.

- ***Web Services*** - A disponibilização de *web services* para interação com a plataforma é outra das restrições que devido à intenção de desenvolver uma Interface API, é de extrema importância.

- **Certificação** - Um dos requisitos mais importantes é a capacidade do software ser certificado para faturação ao abrigo da lei portuguesa, uma vez que ,de outra forma, não seria possível a adoção futura da solução desenvolvida.

- **Manutenção/Continuidade** - Visto que a escolha recai sobre um software *open-source*, é importante escolher uma plataforma que possua uma comunidade ativa e de dimensão considerável, pois isso oferece garantias de continuidade do projeto.

Capítulo 5

Análise de Risco

Nesta secção, são apresentados os riscos identificados no planeamento do projeto bem como no seu desenvolvimento, acompanhados da sua probabilidade(0-5, sendo 0 "quase impossível"e 5 "quase certo") de acontecer, o seu impacto(0-5, sendo 0 "sem impacto"e 5 "crítico") caso se verifiquem e as respetivas estratégias de mitigação:

- **Risco:** Abandono da Comunidade de Suporte

  Descrição: A comunidade que mantém o projeto ativo, desenvolve novas funcionalidades e auxilia na resolução de problemas pode deixar de apoiar o projeto. Contribuindo para o extinção do projeto.

  Probabilidade: 1 **Impacto:** 3

  Estratégia de Mitigação: A escolha do Software a utilizar deve ter em conta o tamanho e o estado da comunidade.

- **Risco:** Falta de Documentação / Documentação Deficiente

  Descrição: A documentação do *software* pode estar desatualizada, ou inexistente, principalmente em casos de criação colaborativa em que não existe uma entidade realmente responsável.

  Probabilidade: 3 **Impacto:** 3

  Estratégia de Mitigação: Perante este risco relacionado com a comunidade, deve ser avaliada a documentação no momento da adoção do *software* e monitorizar os avanços da documentação face a novas versões.

- **Risco:** Familiarização com o tema e tecnologias a usar

Descrição: O estagiário não conhecer bem o funcionamento da área de negócio em que se pretende implementar o projeto bem como algumas tecnologias a utilizar.

Probabilidade: 4 **Impacto:** 2

Estratégia de Mitigação: Estudo da informação científica sobre a área abordada no projeto e das tecnologias a adotar.

– **Risco:** Subestimação do tempo definido para cada tarefa

Descrição: A falta de experiência do estagiário neste tipo de tarefas pode levar a que sejam cometidos erros de cálculo para o tempo das tarefas, o que pode causar atrasos ao projeto final.

Probabilidade: 4 **Impacto:** 2

Estratégia de Mitigação: A avaliação das plataformas que serão requeridas para cada tarefa e a interação com os orientadores na validação do planeamento bem como com os restantes membros da empresa para obter *feedback*.

– **Risco:** Foco do Estagiário

Descrição: O estagiário estará a frequentar outras unidades curriculares no decorrer de todo o estágio o que pode levar a períodos de dispersão da atenção. Poderá ficar comprometida a concentração no trabalho do estágio devido á sobrecarga de outras responsabilidades.

Probabilidade: 4 **Impacto:** 2

Estratégia de Mitigação: O planeamento das tarefas terá de ser feito consciente destes possíveis entraves e o estagiário terá de fazer uma boa gestão do tempo dentro e fora da empresa.

– **Risco:** Certificação do Software

Descrição: A plataforma escolhida poderá não se enquadrar nos requisitos para a certificação de software de faturação em Portugal.

Probabilidade: 2 **Impacto:** 5

Estratégia de Mitigação: A cuidada avaliação na escolha da plataforma e o estudo dos requisitos para a certificação serão os passos a dar para minimizar este risco.

Capítulo 6

Abordagem jBilling

Neste capitulo encontra-se o relato do estudo e implementação efetuados na primeira abordagem ao projeto, onde a solução planeada implicava o recurso à plataforma jBilling.

6.1 Arquitetura

O JBilling é uma plataforma de faturação e gestão de pagamentos cujo Core[25] é baseado em três camadas principais:

- **Camada Cliente** - É a camada responsável pela comunicação com o utilizador, não possui lógica de negócio e só interage com a camada do servidor, nunca com a base de dados. A sua implementação segue o modelo MVC para aplicações web em Java, no qual o cliente acede através do *browser* ao *web server* onde estão as JSPs e todos os pedidos são geridos pelo controlador do Struts que levam à chamada de ações no servidor.

- **Camada Servidor** - O servidor é onde toda a lógica de negócio está aplicada. Consiste numa aplicação Java que assenta essencialmente nas *frameworks* Spring e Hibernate. A sua função principal é receber os pedidos através da interface do cliente ou dos *web services* e satisfazê-los com a aplicação da lógica de negócio e da comunicação com a base de dados.

- **Camada da Base de Dados** - É composta por um motor de base de dados que armazena toda a informação do sistema, por defeito o jBilling usa PostgreSQL mas pode facilmente ser configurado para outro sistema de gestão de base de dados.

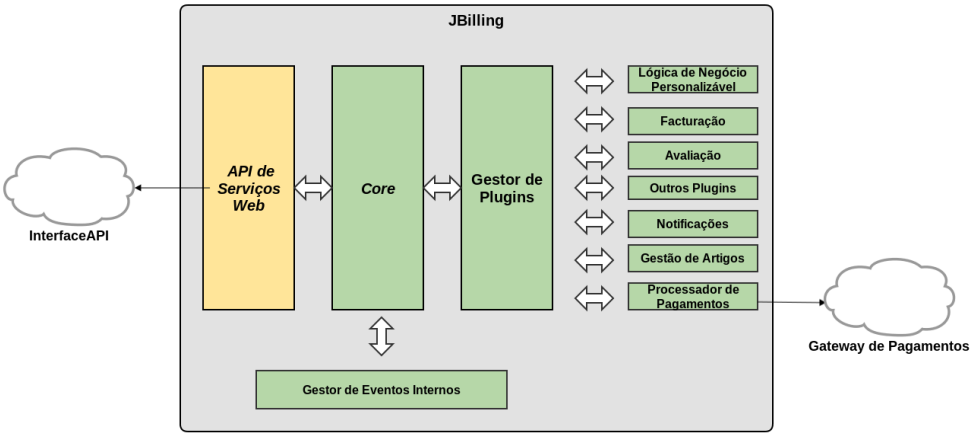


Figura 6.1: Componentes Principais do jBilling

Para compreender melhor o funcionamento do JBilling e onde vão ser integrados os componentes a desenvolver no estágio, deve ser observada a figura 6.1 onde estão ilustradas as quatro componentes principais do sistema:

- **API de Serviços Web** - É através desta API que se realiza grande parte da comunicação entre os sistemas externos e o JBilling. Com chamadas aos diversos métodos presentes na API, podem ser consultados e alterados diversos dados, efetuados pagamentos, enviadas faturas e até configurações no sistema. Infelizmente, na versão gratuita, esta funcionalidade não está disponível, por defeito(embora esse fator seja contornável).

- **Gestor de Eventos Internos** - Responsável pela gestão e criação dos eventos internos, que servem de sinal desencadeador para determinados *plugins*.

- **Gestor de *Plugins*** - A arquitetura do jBilling é baseada em *plugins* e este gestor é uma das peças fundamentais para que o processo individual de cada *plugin* seja realizado sem que existam interferências entre eles, garantindo a estabilidade do sistema. Permite também a criação de novos *plugins* e sua integração no sistema.

- **Core** - Já descrito anteriormente, este componente é onde se encontram as principais funcionalidades base da plataforma.

6.2 Solução

Os componentes a serem desenvolvidos pelo estagiário estão representados, na figura 6.2, a vermelho, onde é possível também ver as interações entre as diferentes plataformas.

Representado a azul, a SmartCityAPI, ira utilizar o jBilling, a verde, como plataforma de gestão da informação, faturação e processamento de pagamentos. Os clientes e fornecedores interagem apenas com esta plataforma.

As comunicações efetuadas entre estes dois sistemas serão realizadas através da InterfaceAPI que pretende disponibilizar as funcionalidades do jBilling através de *Web Services*(uma vez expostos) para que possam ser chamados pela SmartCityAPI.

Por outro lado, existirá também a necessidade de criar um *plugin* que integre um *gateway* de pagamentos, para que estes possam ser efetuados os pagamentos.

Para realizar o controlo de fluxo monetário entre o pagamento do cliente e as diversas entidades envolvidas, caso o *gateway* de pagamento não forneça esse serviço, será usado o motor de regras de negócio presente no JBilling.

Está contemplada ainda a possibilidade de integração com um *software* de CRM para melhor gestão de clientes, mas essa não é uma prioridade do projeto.

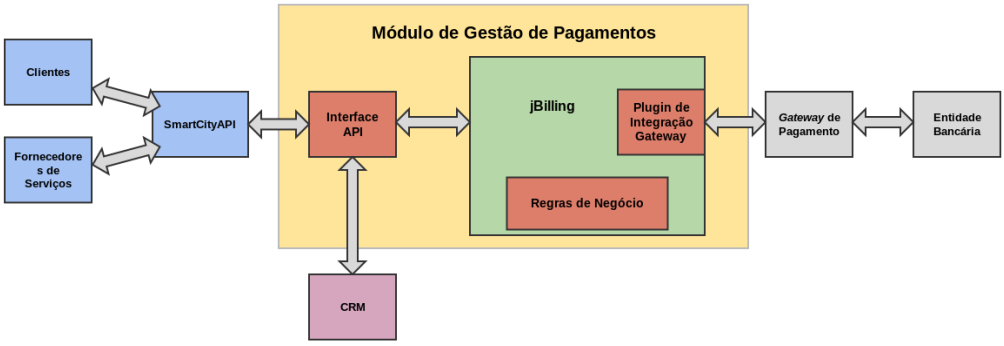


Figura 6.2: Visão Geral da Solução a Implementar.

6.3 Implementação

A InterfaceAPI é o principal componente a desenvolver durante o período de estágio, pois é responsável por toda a comunicação entre a SmartCityAPI e o jBilling.

Após alguma pesquisa.[26] foi possível concluir que o jBilling possui uma interface de *web services* e a implementação dessa mesma interface.

Com o uso do **HttpInvoker**, uma ferramenta específica da *framework* **Spring** para efetuar RPC através de HTTP, efetuou-se o acesso remoto à API a partir de uma aplicação cliente em Java desenvolvida para o efeito.

Na figura 6.3 é possível ver o funcionamento do HTTPInvoker, desde que a chamada do método é feita na aplicação cliente, passando pela sua conversão para um RPC através de HTTP, até à sua interpretação por parte da Spring *framework* do lado do servidor e a consequente chamada ao método pretendido.

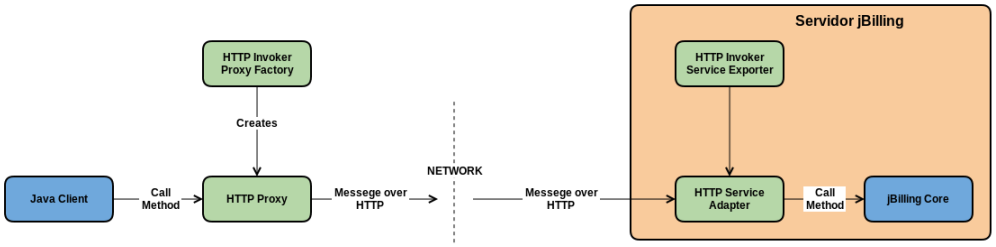


Figura 6.3: Funcionamento do HTTPInvoker

Com as condições necessárias encontradas, deu-se o início, ao desenvolvimento da aplicação cliente para executar as tarefas pretendidas no projeto.

A aplicação cliente nunca chegou a ficar concluída porque durante o processo, devido a diversos fatores, o jBlling deixou de ser a plataforma a adotar para a solução. Ainda assim no capítulo 8 é possível ver os requisitos que já se encontravam funcionais aquando do abandono da plataforma.

6.4 Problemas

Durante o período de desenvolvimento inicial, várias foram as dificuldades em encontrar documentação completa e atualizada.

A versão utilizada do jBilling(3.3.1) tinha sido lançada há pouco tempo(23/12/2014) o que terá originado a falta de documentação mas possuía funcionalidades que eram importantes para o projeto.

No fórum da comunidade, as questões e problemas acumulavam-se e o tempo de resposta era elevado. Começaram a surgir as primeira preocupações.

Uma das razões pela qual a escolha do jBilling fora tida em conta, prendia-se com o facto de já existir uma empresa em Portugal com este software certificado para faturação, pela Autoridade Tributária e Aduaneira. Após conversações com essa entidade, foram obtidas informações de que tinha sido necessário um trabalho considerável de personalização do software para estar de acordo com os requisitos da certificação.

Uma vez analisado o modelo de dados com mais pormenor, também foi verificado que a classe *partner* que o estagiário tinha assumido incorretamente que seria o equivalente a um fornecedor, na verdade não tinha essa função. Consequentemente seria necessário adaptar o modelo da base de dados para que resolver esse entrave.

Com o estudo crescente e cada vez mais profundo da plataforma, foi-se evidenciando que a necessidade de personalizações ao *software* ultrapassava em quantidade e complexidade o previsto.

Impôs-se então que se desse inicio a uma revisão do estudo do estado da arte. Face à troca de informação durante a defesa intermédia de estágio, foi revisitada a plataforma Odoo com o objetivo de avaliar o seu carácter de solução open-source, que se veio a confirmar. Deste modo foi possível optar por esta plataforma cujas potencialidades se adequam plenamente aos interesses do projeto.

Capítulo 7

Abordagem Odoo

O Odoo[27] é um *software open-source* de gestão empresarial e foi a plataforma escolhida, após uma segunda análise aos *softwares* disponíveis para construir a solução pretendida neste trabalho.

Numa primeira instância, este *software* tinha sido incorretamente rotulado como não sendo *open-source*, devido a um erro na consulta do site respetivo. Após o *feedback* recebido na defesa intermédia e uma segunda avaliação do *software*, foi possível concluir que, o que anteriormente se tinha identificado como preços de aquisição do *software*, eram na verdade preços referentes ao *hosting* remoto.

Cumprindo a esmagadora maioria dos requisitos necessários para o projeto, foi então a solução adotada.

7.1 Arquitetura

Esta secção pretende dar a conhecer um pouco do funcionamento interno da plataforma Odoo, bem como alguns protocolos de comunicação entre os diversos componentes da mesma.

É um sistema *multi-tenant* composto essencialmente por três camadas - a camada de dados, a camada da aplicação e a camada da apresentação - que seguem o modelo de arquitetura Model-View-Controller.

Como é possível observar na figura 7.1, a plataforma está dividida em três componentes principais, sendo eles:

- **Base de Dados** Por defeito, utiliza o PostgreSQL, devido às suas potencialidades. É nesta camada que se encontra toda a informação e a maioria das configurações do sistema. Embora seja possível fazer *queries* SQL diretamente a partir dos módulos do Odoo, grande parte dos

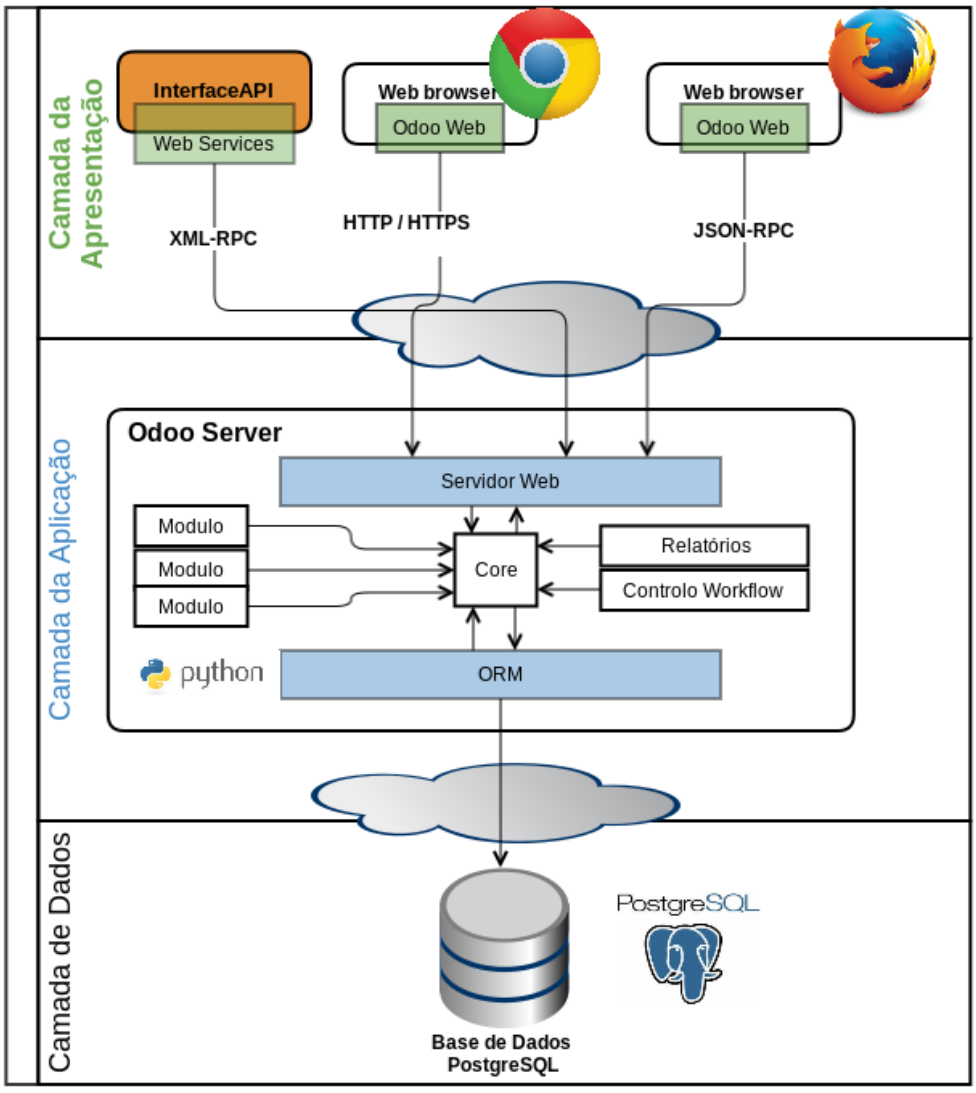


Figura 7.1: Arquitetura do Odoo

acessos à base de dados são feitos através do motor de ORM do servidor do Odoo, o que garante que sejam efetuadas diversas verificações de segurança.

- **Cliente Web** - Devido à lógica de negócio estar maioritariamente do lado do servidor, é uma simples aplicação *JavaScript* a correr no *browser* do cliente que fornece a interface de interação com os utilizadores. Faz pedidos ao servidor para efetuar alterações nos dados ou consultas dos dados, para mostrar ao cliente quando recebe a resposta do servidor. Por defeito, a comunicação com o servidor é feita através de JSON-RPC.

- **Odoo Server** Responsável por executar a grande parte das funcionalidades e do processamento. Toda a lógica de negócio é aplicada aqui, possuindo um *core* que garante todo o funcionamento integrado das diversas partes que o constituem.

  Na sua constituição encontram-se três elementos essenciais ao funcionamento do sistema - **Servidor Web**, **Módulos** e o motor de **ORM**.

  - O **Servidor Web** é a camada que faz a interface entre os *browsers* e o servidor Odoo. É capaz de tratar normais pedidos HTTP, pedidos JSON-RPC feitos a partir do *browser* ou pedidos XML-RPC vindo das aplicações que consomem os dados da API do Odoo.
  - Motor de **ORM** é a camada que faz toda a comunicação com a base de dados PostgreSQL, extremamente útil para transformar os modelos de dados escritos em Python e criar as tabelas correspondentes, na base de dados, aliando a flexibilidade do Python aos benefícios das RDBMS.
  - Os **Módulos** são o componente que oferece todo o valor ao negócio, pois o Odoo, por si só, é apenas um *core* que permite correr todos estes módulos. Consoante as necessidades e os requisitos de cada projeto, são instalados diversos módulos que interagem com todo o sistema. Qualquer versão oficial do Odoo já trás diversos módulos prontos a instalar, mas é muito fácil encontrar junto da comunidade Odoo milhares de módulos já desenvolvidos. Na secção 7.3.2, estão descritos com detalhe os seus componentes e funcionamento.

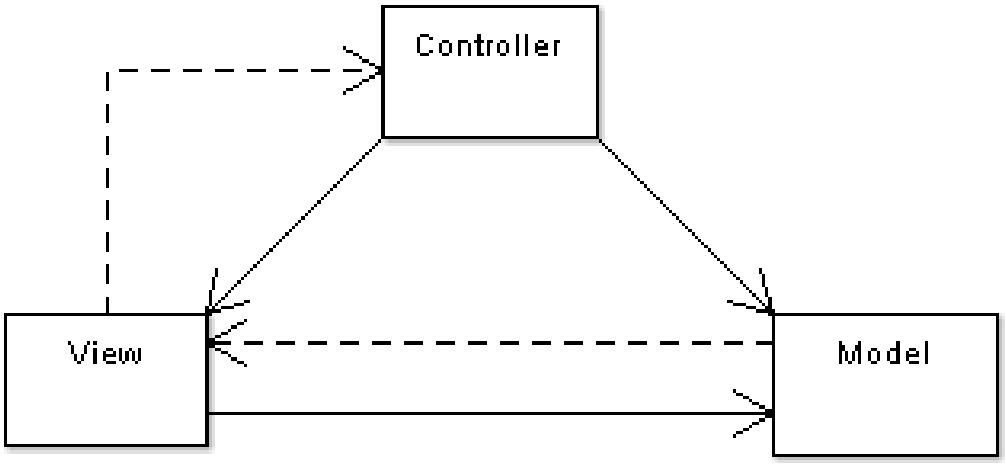


Figura 7.2: Diagrama do Model-View-Controller

O diagrama ilustrado na figura 7.2 representa o funcionamento da arquitetura MVC aplicada ao Odoo. Desta forma é possível separar os dados da interação com o utilizador, através da adição de um *controller*. Os componentes do sistema que têm as funções definidas nesta arquitetura são os seguintes:

- **Model** -> As tabelas na base de dados PostgreSQL.
- **View** -> Os ficheiros XML que definem as vistas(parte integrante de cada de cada módulo).
- **Controler** -> Os objetos do Odoo (parte constituinte dos módulos).

As linhas contínuas representam o total acesso aos componentes, enquanto que o tracejado corresponde ao acesso parcial.

No caso do *Model–>View* é enviado uma notificação para o *View*, sempre que se verifique alguma alteração, para que possa ser atualizada para o utilizador. Ao *Model* basta-lhe um miímo acesso para tomar essa ação.

No caso do *View–>Controller* justifica-se o acesso limitado devido ao facto do controller poder ser substituído a qualquer momento. Reduz-se assim as dependências do View face ao Controller.

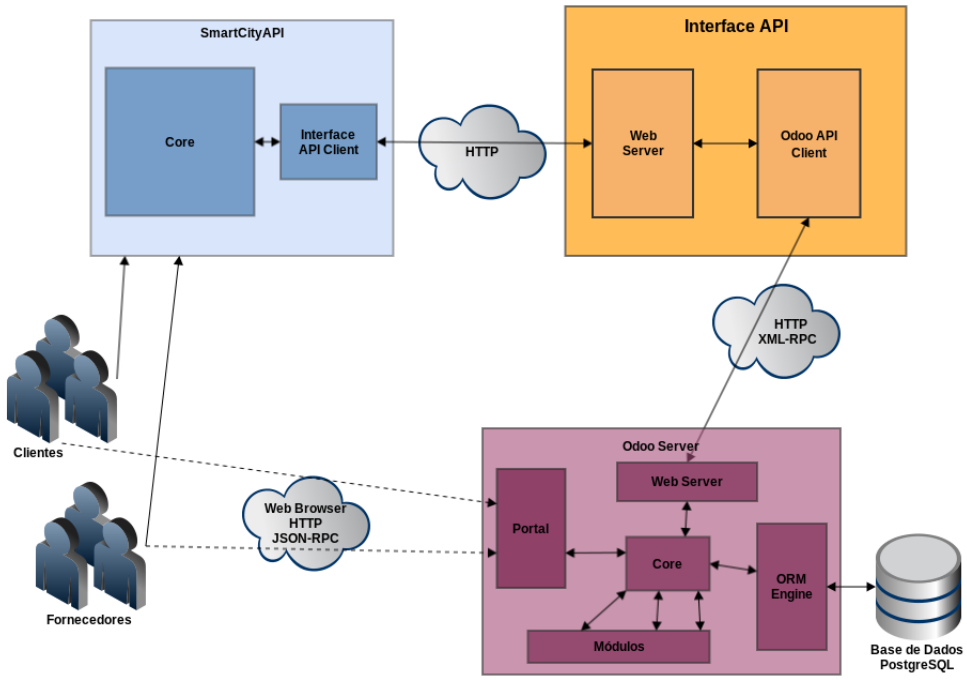


Figura 7.3: Contexto da Solução (Interface API + Odoo Server)

7.2 Solução

A solução para este projeto consiste no *software* Odoo e a exploração de todas as suas potencialidades, bem como, numa InterfaceAPI que faz a comunicação ente a SmartCityAPI e o servidor do Odoo.

Para uma melhor compreensão, é possível observar a figura 7.3, onde as duas componentes da solução estão ilustradas a laranja(Interface API) e roxo(Odoo). A azul está representada a SmartCityAPI que é o principal meio de interação entre os utilizadores e o resto do sistema.

Devido à grande variedade de módulos já desenvolvidos pela Comunidade Odoo, uma grande parte dos requisitos é cumprido apenas com a instalação e configuração de alguns desses módulos.

Os utilizadores também dispõem de uma interface gráfica acessível, através do *browser*, para interagir com o servidor Odoo. Esta característica não fazia parte dos requisitos mas foi vista como uma mais-valia para o projeto, e como tal, implementada.

7.3 Implementação

Devido à natureza dos requisitos do projeto, a implementação foi dividida em três fases distintas:

- **Instalação e Configuração de Módulos** Através da vasta oferta disponível na comunidade Odoo, foi possível definir que as tarefas mais gerais, a qualquer negócio, como a gestão de clientes, fornecedores, produtos, vendas e faturação seriam implementadas através da instalação e configuração de módulos já existentes, permitindo assim o acesso imediato através da plataforma web. A consulta de relatórios sobre as atividades de negócio e a integração com o *gateway* de pagamentos PayPal também foram conseguidos desta forma.

  Esta foi a primeira fase da implementação, pois só depois destes módulos estarem todos operacionais foi possível começar o desenvolvimento da Interface API para disponibilizar esta informação para a SmartCityAPI.

- **Implementação do módulo de Exportação SAFT-PT** Como é possível verificar na secção 7.4, a exportação de um ficheiro SAFT-PT com toda a informação sobre a faturação da empresa para ser entregue à Autoridade Tributária é essencial no processo de Certificação e Legalização dos *softwares* de faturação em Portugal. Como tal, após uma pesquisa de um módulo com esta finalidade, sem sucesso, foi decidido que seria necessário implementar um módulo no Odoo para fazer a exportação.

- **Desenvolvimento da Interface API** Nesta fase, foi desenvolvida uma RESTful API que disponibiliza, através de *web services*, a informação do servidor Odoo e permite realizar as tarefas chave do projeto. Desta forma, é possível comunicar com o servidor Odoo de uma forma definida pela empresa. Possibilita a implementação de mecanismos de autenticação e autorização, da restante infraestrutura, mantendo coerência entre os diferentes serviços e retira às aplicações clientes, que consumirão a informação da Interface API, a preocupação com possíveis alterações da API do Odoo ou outras dependências externas.

  Embora o módulo de faturação a desenvolver neste estágio esteja inserido no projeto SmartCityAPI, é também uma importante característica que seja genérico o suficiente para poder ser adaptado para outros projetos na empresa, com características semelhantes.

Nome do Módulo	
Social Network	Portugal - Chart of Accounts
Online Billing	Portugal : implementação do saft
Accounting and Finance	Payment Acquirer
Sales Management	Paypal Payment Acquirer
Warehouse Management	Transfer Payment Acquirer
WMS Accounting	Portal
Instant Messaging	Portal Sale
Purchase Management	Portal Stock
eInvoicing	Procurements
Template of Charts of Accounts	Products & Pricelists
Analytic Accounting	Report
Password Encryption	Sales Teams
Signup	Sales and Warehouse Management
Base	Share any Document
Base import	Web
Initial Setup Tools	Web Calendar
VAT Number Validation	OpenERP Web Diagram
Dashboards	Web Gantt
IM Bus	Graph Views
Decimal Precision Configuration	Base Kanban
Electronic Data Interchange (EDI)	Gauge Widget for Kanban
Email Templates	Sparkline Widget for Kanban
Email Gateway	Tests
Odoo Live Support	View Editor

Tabela 7.1: Lista de Módulos instalados no Odoo.

7.3.1 Instalação e Configuração

Nesta secção, serão apresentados os módulos instalados, uma pequena descrição dos mais importantes e as devidas configurações-chave para que todo o sistema funcione como desejado.

Tal como foi descrito anteriormente, o Odoo é apenas um Core que permite que sejam instalados diversos módulos que implementam todas as funcionalidades desejadas. Na tabela 7.1 estão apresentados todos os módulos instalados na versão final da solução implementada.

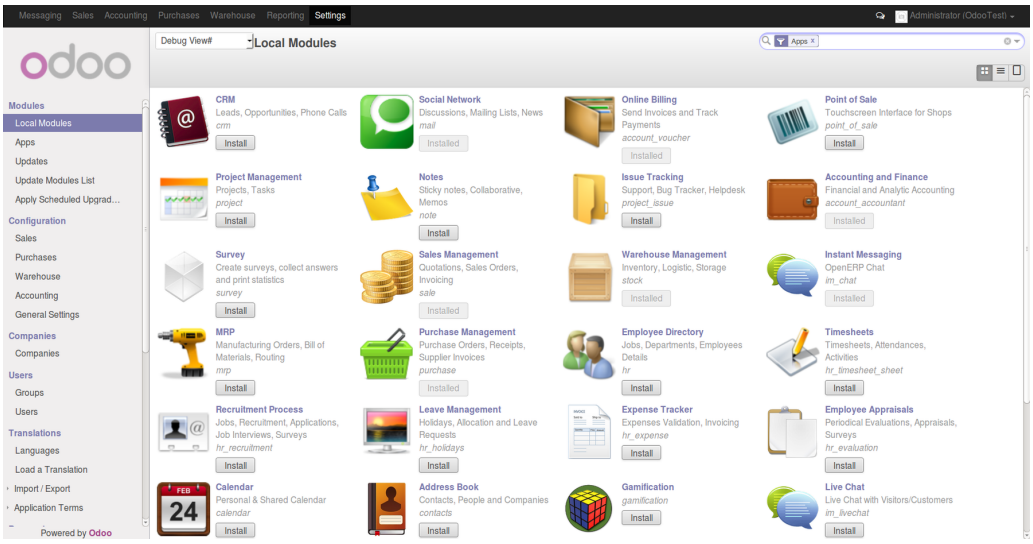


Figura 7.4: Interface Visual para Instalação dos Módulos

Na escolha dos módulos, foi dada preferência aos oficiais do Odoo, devido à sua grande funcionalidade mas também à maior garantia futura de continuação e adaptação a novas versões da plataforma, facilitando assim a manutenção da mesma. Os **módulos**-chave para o funcionamento da solução são os seguintes:

- ***Accounting and Finance*** - Módulo que permite o acesso através da conta de administrador a todos os elementos relacionados com gestão da contabilidade, essencialmente os diários de transações e os planos de contas.
- ***Sales Management*** - Peça-chave de todo o sistema, com possibilidade de criação de pedidos de cotação, gestão de encomendas e geração de faturas, entre muitas outras funcionalidades de análise de vendas.
- ***Online Billing*** - Com o propósito de enviar faturas através de correio eletrónico e de acompanhar os processos de pagamento online, foi instalado este módulo. Também adiciona funcionalidades de apoio ao pagamento aos fornecedores e ao relatório de vendas.
- ***Purchase Management*** - Possibilita a criação de pedidos de cotação aos fornecedores e as respetivas encomendas o que permite uma boa gestão de todas as compras efetuadas aos fornecedores, característica importante no projeto pois um dos requisitos é calcular o preço de venda aos clientes, aplicando uma taxa ao preço dos fornecedores.

– ***Portal*** - Plataforma por onde os clientes e fornecedores têm acesso ao sistema através de um web browser, onde podem consultar as suas encomendas e faturas recebidas bem como efetuar pagamentos. O administrador define os direitos de acesso à informação, permitindo assim controlo total, personalizando as interfaces mais indicadas para clientes, fornecedores, publico geral ou até certos membros da empresa.

– ***Paypal Payment Acquirer*** - Este módulo permite que sejam efetuados pagamentos através da plataforma PayPal. No portal dos clientes/fornecedores, ao visualizar uma fatura que ainda esteja por pagar, aparecerá um botão para realizar o pagamento através da plataforma.

Para dar resposta aos requisitos do projeto, foi necessário realizar determinadas configurações após a instalação dos módulos. Grande parte dessa informação foi conseguida através da comunidade[28] Odoo. Segue então uma lista das principais **configurações** realizadas para o funcionamento adaptado à solução:

- As **Funcionalidades Técnicas** são um enorme conjunto de potencialidades que por defeito não estão visíveis, mesmo que autenticado como administrador. Bastando apenas configurar a conta de utilizador, é possível dar acesso a estas funcionalidades, mas muito tempo foi gasto na experimentação da plataforma, à procura de menus e opções que não se encontravam visíveis devido a esta característica do Odoo. Todas as configurações listadas abaixo necessitam desta opção ativa, para poderem ser concretizadas.

- **Ativar Pricelists** permitiu implementar regras de definição de preços, fazendo com que os preços dos produtos para venda fosse calculado com base no custo e incrementando-o com base numa percentagem, permitindo assim cumprir um dos requisitos do projeto. Podem ser definidas várias regras sobre produtos de categorias diferentes e em determinados períodos de tempo, o que permite uma grande flexibilidade para diversas situações.

- Os **Pagamentos PayPal** foram conseguidos através da instalação do módulo respetivo, ficando a disponibilização dos botões de pagamento ao cliente a cargo da configuração do módulo. Uma vez definidas credenciais da conta e a permissão para mostrar o botão no portal, os clientes passam a ter a funcionalidade de efetuar os pagamentos através do PayPal.

- **Restrição de Acessos** é uma das mais importantes características do sistema para manter a sua segurança e é conseguida através de um sistema muito completo de gestão de direitos de acesso à informação dos diversos módulos existentes na plataforma.

- **Reconciliação de Pagamentos** é a funcionalidade que permite aliar os pagamentos efetuados às faturas emitidas. Por defeito, somente na versão

para contas portuguesas, esta configuração vem desativada, o que levou a algumas confusões iniciais, pois a documentação relativa a este problema era bastante escassa, e sem conhecer bem as interações do sistema, foi complicado efetuar o simples ato de pagar uma fatura na plataforma.

Após tomadas estas medidas, o sistema reunia condições para o começo do desenvolvimento da Interface API.

7.3.2 Módulo SAFT-PT

Nesta secção está descrita a forma como se conseguiu implementar o requisito de exportar ficheiros SAFT-PT através da plataforma. Embora, no final, a implementação desta funcionalidade tenha culminado numa instalação de um módulo, irá ser descrito todo o processo e estudo realizado anteriormente pois a dimensão temporal de investimento na solução foi elevada.

Como estava definido no plano de implementação , deu-se início ao estudo do desenvolvimento de módulos para expandir as funcionalidades do Odoo, bem como dos da informação contida nos ficheiros SAFT-PT[29]. Após alguma pesquisa, ficou claro que a tarefa era mais complexa do que tinha sido antecipado mas tudo foi encarado como um desafio, enquanto que, por outro lado, se estudavam possíveis alternativas. Ao fim de algumas semanas de investigação e desenvolvimento, foi encontrada uma solução num fórum online[30] onde estava mencionado um projeto que tinha como objetivo fazer a atualização de um módulo de exportação de ficheiros SAFT-PT, do antigo OpenERP(v.6) para o novo Odoo(v.8). Verificou-se ser oportuno integrar este módulo[31] na solução final deste projeto.

Módulos do Odoo

Como o processo de estudo do funcionamento interno do Odoo, SAFT-PT[32] e início de desenvolvimento do módulo ainda ocupou tempo considerável e este relatório visa descrever o estágio realizado e não só as soluções finais encontradas, de seguida, encontra-se a base da arquitetura dos módulos no Odoo, bem como as principais características de um ficheiro SAFT-PT.

Para compreender melhor a estrutura[33] de um módulo do Odoo, pode ser consultada a figura 7.5. Segue também uma explicação mais detalhada sobre os seus principais componentes:

– **Objetos** - Todos os recursos no Odoo são objetos, quer sejam clientes, faturas, menus, ações, relatórios... A sua nomenclatura é hierárquica. Como exemplo, o objeto que tem a informação dos produtos listados

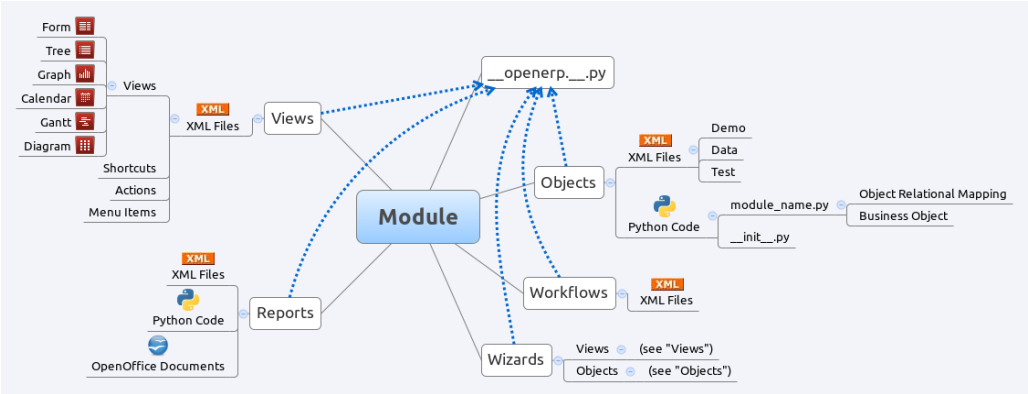


Figura 7.5: Estrutura de um Módulo no Odoo

numa fatura é representado da seguinte forma *"account.invoce.line"*. Os objetos são declarados como subclasses do osv.Model em Python e é possível através da interface ORM efetuar *queries* aos objetos, uma vez na base de dados.

- **Ficheiros XML** - São usados para inicializar ou atualizar a base de dados quando o módulo é instalado ou atualizado. A estrutura do ficheiro assenta essencialmente em *records*, que definem um novo registo num modelo da base de dados e *fields* que representam os atributos e seus valores a alterar em cada *record*.

- **Vistas** - São ficheiros XML que definem de que modo a informação que vem da base de dados vai ser mostrada no lado do cliente. Usa o QWeb, um gerador de templates, para páginas HTML. Existem vários tipos de vistas e o mesmo objeto pode conter distintas vistas para diferentes menus na plataforma.

- **Relatórios** - São escritos em HTML/QWeb tal como as vistas e têm como principal função a criação de documentos de apoio ao negócio. Os documentos são gerados em PDF ou HTML.

- **Workflow** - Para conseguir definir a dinâmica dos objetos entre os seus diferentes estados e sua evolução durante os processos, são utilizados os *workflows*. Na figura 7.6, podemos ver um exemplo de um *workflow* de uma fatura e os seus diferentes estados, apresentando-se a vermelho o estado atual em que se encontra. Não é obrigatório um objeto ter um *workflow* porém um objeto pode estar associado a mais que um *workflow*.

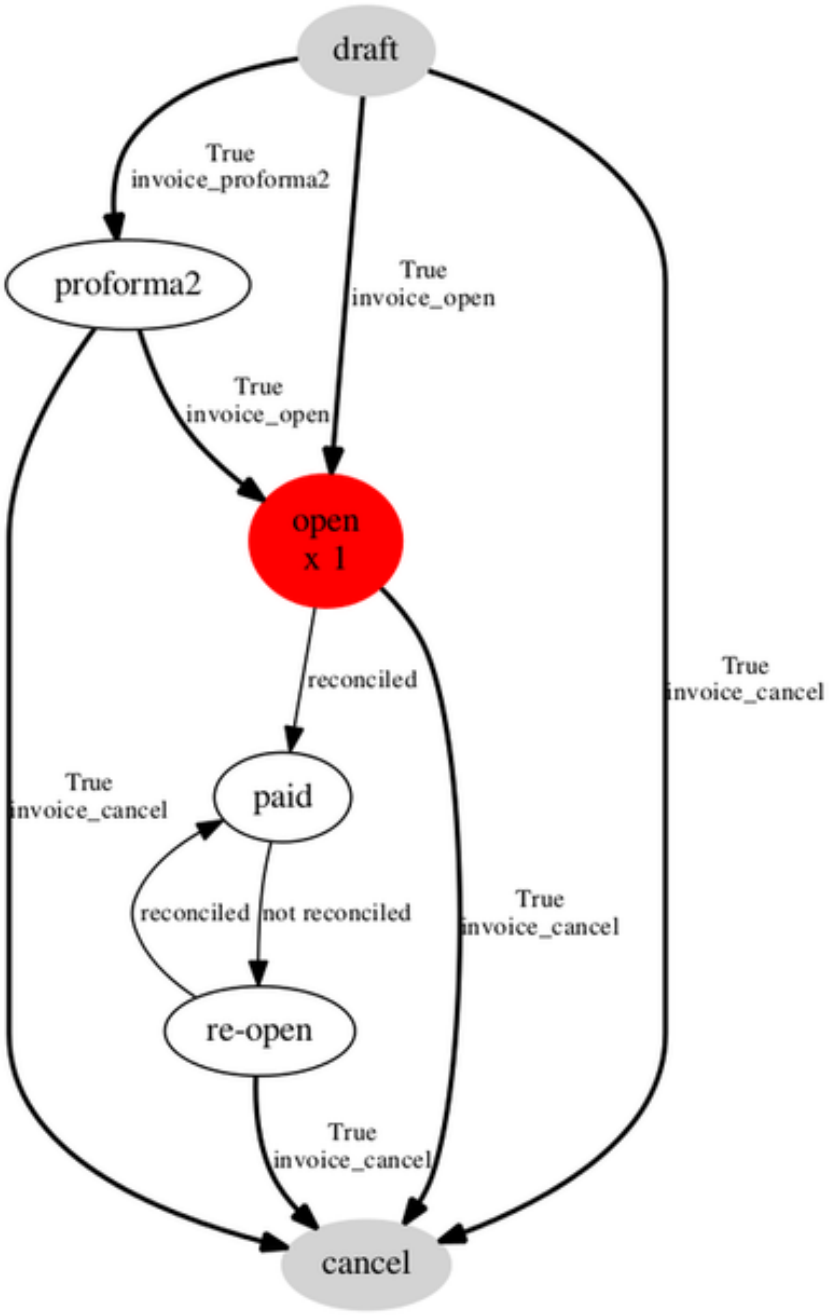


Figura 7.6: Workflow de uma Fatura

Ficheiro SAFT-PT

O SAFT-PT(*Standard Audit File for Tax* -Versão Portuguesa) é um formato de normalização para exportação de dados contabilísticos, que tem como principal objetivo facilitar às autoridades fiscais o combate à fraude e evasão fiscal bem como agilizar os processos de auditoria nas empresas.

O ficheiro deve poder ser exportado em qualquer altura, independentemente do programa utilizado e referente a um determinado período temporal.

Foi criado inicialmente pela Portaria nº 321-A/2007, de 26/03, e sofreu já algumas atualizações, sendo que a última foi registada na Portaria n.º 274/2013, de 21 de agosto.

No **Anexo A**, encontra-se esta última Portaria, para consulta dos diversos campos obrigatórios no documento.

Observações

Devido à grande dimensão e à multiplicidade dos seus componentes, a tarefa de implementar o módulo de exportação do ficheiro SAFT-PT revelou-se demasiado extensa e complexa. Surgiram algumas dificuldades em conseguir identificar todos od componentes do ficheiro que já se encontravam presentes no Odoo e os que tinham de ser acrescentados ao modelo de dados. O processo de desenvolvimento atrasou-se bastante relativamente ao previsto para esta tarefa e paralelamente foram feitos esforços para tentar encontrar uma solução.

Como já anteriormente foi referido, através da investigação, foi encontrado um módulo criado muito recentemente que substitui a solução que estava a ser desenvolvida.

Aquando da descoberta do módulo adotado, o módulo desenvolvido pelo aluno apenas recolhia as informações dos campos já existentes na base de dados para clientes, fornecedores e produtos. sem ainda executar as devidas retificações nos modelos de dados do Odoo, criando os diversos campos necessários para o SAFT-PT e não presentes no Odoo.

7.3.3 Interface API

Nesta secção, está descrito o funcionamento e implementação da Interface API usada para fazer a comunicação entre a SmartCityAPI e a plataforma Odoo.

A Interface API é composta por duas partes principais, sendo elas a aplicação cliente à API do Odoo, que executa toda a comunicação com a plataforma Odoo, e a disponibilização dessa aplicação através de uma RESTful API que recebe os pedidos da SmartCityAPI. Consequentemente também são realizadas as devidas operações de formatação nos dados para que cumpram os requisitos de ambas as API's.

Odoo API Client

O Odoo fornece[34] através do protocolo XML-RPC uma simples forma de interagir com o seu servidor.

A grande maioria das ações é feita através de dois *endpoints*. São eles o "*xmlrpc/2/common*"onde a informação acessível é bastante limitada pois o seu propósito é efetuar a autenticação dos utilizadores e o "*xmlrpc/2/object*"que já permite executar métodos que são úteis na leitura e escrita de dados para a base de dados bem como a manipulação dos workflows.

A linguagem de desenvolvimento escolhida para a aplicação-cliente foi o **Python**, com o uso da biblioteca **xmlrpclib** que transforma os objetos Python em xml, para fazer as necessárias chamadas à API.

Abaixo listadas estão as principais interações usadas na criação da aplicação-cliente, desde a conexão com o servidor até às escritas e leituras da base de dados:

- **Conexão ao Servidor** Através do primeiro *endpoint* disponível, é possível fazer a ligação ao servidor, tendo acesso apenas a um método que nos permite verificar a versão do servidor e outro que serve para autenticar o utilizador.

- **Autenticação** Os dados necessários para fazer a autenticação são o nome da base de dados, o nome de utilizador e a respetiva password. Esta ação é feita ainda no primeiro *endpoint* e devolve um uid(User Id), com o qual serão validadas as chamadas posteriores no segundo *endpoint*.

- **Execute_kw** Este é o nome da função que é chamada para efetuar quase todas as operações necessárias. A função possui os seguintes parâmetros:

  Nome da base de dados.

  Id do utilizador (recebido pela função de autenticação).

  Password do utilizador.

  O nome do modelo do Odoo.

  O método do referido modelo.

  A lista de parâmetros.

- **Métodos de Leitura** A consulta dos dados é feita através de métodos como o *search*(), que recebe um filtro e devolve uma lista de ids com os resultados correspondentes, ou do método *read*() que recebe uma lista de ids e devolve todos os campos e valores desses registos. Devido ao elevado numero de vezes em que estes dois métodos são usados em conjunto, existe o método *search_read*() que combina as duas funcionalidades. Existem ainda funções de auxílio como a *search_count*() que devolve a soma dos resultados da pesquisa e a *fields_get*() que devolve os campos de determinado registo o que é extremamente útil para a inspeção dos modelos.

- **Métodos de Escrita** Para escrever na base de dados existem duas possibilidades: ou se está a criar um novo registo e nesse caso é usado o método *create*(), ou se está a actualizar um já existente e nessa situação usa-se o *write*().

- **Método para Apagar** Os registos podem ser apagados com o uso do método *unlink*() e com o respetivo id do registo.

- **Exec_workflow** As manipulações do *workflow* são importantes para modificar o estado de certos registos. Através desta função, é possível enviar um sinal para o registo certo, fazendo este mudar de estado.

- **Geração de Relatórios** Embora na solução implementada não se tenha usado, a API permite gerar os relatórios através do *endpoint* especifico "*/xmlrpc/2/report*". O resultado desta função é um PDF convertido para dados binários (base64) que deve ser descodificado e gravado para ficheiro, antes de ser usado.

Na figura 7.7, é possível ver um exemplo do código necessário para a criação, edição e eliminação de um cliente, incluindo também a necessária conexão e autenticação com a aplicação.

```python
import xmlrpclib

url = 'http://localhost:8069'
db = 'OdooTest'
username = 'admin'
password = 'admin'

        # Primeiro Endpoint
common = xmlrpclib.ServerProxy('{}/xmlrpc/2/common'.format(url))

        # Autenticação
uid = common.authenticate(db, username, password, {})

        # Segundo Endpoint
api = xmlrpclib.ServerProxy('{}/xmlrpc/2/object'.format(url))

        # Criar o cliente
newClientId = api.execute_kw(db, uid, password, 'res.partner', 'create',
    [{'name': 'Gil Hilário', 'City':'Coimbra', 'funtion': 'Intern'}])

        # Editar o cliente
api.execute_kw(db, uid, password, 'res.partner', 'write', [[newClientId],
    {'City': 'São Brás de Alportel', 'function': 'Project Manager'}])

        # Apagar o cliente
api.execute_kw(db, uid, password, 'res.partner', 'unlink', [[newClientId]])
```

Figura 7.7: Código exemplo para operações de gestão de clientes.

Web API

Uma vez criado o código para interagir com o servidor Odoo e realizar as tarefas definidas no levantamento de requisitos, foi criada uma API cujos serviços disponibilizados são compostos pelas chamadas à API do Odoo e que respeita o modelo de arquitetura REST para *web services*.

A importância desta Interface API encontra-se em quatro fatores: na separação das aplicações que consomem a informação do servidor Odoo o que elimina preocupações com eventuais mudanças na API do Odoo; na possibilidade de aplicação de mecanismos de autenticação e autorização da restante plataforma; na manutenção de coerência entre serviços; na possibilidade de adaptação a outros projetos.

De forma a coadunar ambas as aplicações, a Web API também foi desenvolvida em Python, através do Flask[35], uma *microframework* baseada em *Werkzeug* e **Jinja2**. A escolha recaiu sobre esta *framework* devido à sua simplicidade, ao facto de responder a todas as necessidades do projeto e também ao conhecimento que o estagiário detinha da ferramenta.

No **anexo B** é possível consultar a documentação da Web API, feita através da ferramenta Swagger[36], para melhor compreender os serviços disponíveis, os parâmetros de cada chamada e as respostas dadas. Os modelos de dados devem ser tidos em conta apenas como exemplos, pois os objetos da plataforma Odoo correspondentes possuem um elevado número de campos.

Relativamente à segurança, foi implementado o sistema de autenticação *HTTP Basic Auth*. Todos os *endpoints* da aplicação, com exceção do primeiro "*/api*", requerem esta validação. A autenticação é feita com o *username* e a *password* da conta do utilizador no servidor Odoo, de forma a que, uma vez autenticados, os clientes passem a ter o acesso limitado pelo controlo de acessos implementado no Odoo.

A figura 7.8 apresenta o código da API para receber, processar e responder aos pedidos HTTP feitos a partir da SmartCityAPI. Novamente como exemplo, é dada a gestão dos clientes. É possível observar, que a API vai de encontro às especificações REST, com o uso dos vários métodos fornecidos pelo protocolo HTTP, nas situações apropriadas.

```python
@app.route('/api/customers', methods=['GET'])
@auth.login_required
def getCustomers():
    # Pesquisa todos os clientes
    return jsonify({'customers': api.execute_kw(db, uid, password, 'res.partner', 'search_read',
                                    [[['customer', '=', True]]], {'fields': ['name']})})

@app.route('/api/customers', methods=['POST'])
@auth.login_required
def addCustomer():
    # Verifica se recebeu um json e se tem o campo obrigatorio
    if not request.json or not 'name' in request.json:
        abort(400)
    # Cria o Cliente
    id = api.execute_kw(db, uid, password, 'res.partner', 'create', [request.json])
    return str(id), 201

@app.route('/api/customers/<int:customer_id>', methods=['GET'])
@auth.login_required
def getCustomerById(customer_id):
    # Pesquisa pelo cliente com o id fornecido
    customer = api.execute_kw(db, uid, password, 'res.partner', 'search_read', [[['customer',
                                '=', True], ['id', '=', customer_id]]], {'fields': ['name']})
    if len(customer) == 0:
        abort(404)
    # Devolve o Registo do customer
    return jsonify({'customer': customer})

@app.route('/api/customers/<int:customer_id>', methods=['PUT'])
@auth.login_required
def editCustomer(customer_id):
    # Verifica se recebeu um json
    if not request.json:
        abort(400)
    # Pesquisa pelo cliente com o id fornecido
    customer = api.execute_kw(db, uid, password, 'res.partner', 'search_read', [[['customer',
                                '=', True], ['id', '=', customer_id]]], {'fields': ['name']})
    if len(customer) == 0:
        abort(404)
    # Edita o cliente
    api.execute_kw(db, uid, password, 'res.partner', 'write', [[customer_id], request.json])
    return 'Success', 200

@app.route('/api/customers/<int:customer_id>', methods=['DELETE'])
@auth.login_required
def deleteCustomer(customer_id):
    # Pesquisa pelo cliente com o id fornecido
    customer = api.execute_kw(db, uid, password, 'res.partner', 'search_read', [[['customer',
                                '=', True], ['id', '=', customer_id]]], {'fields': ['name']})
    if len(customer) == 0:
        abort(404)
    # Apaga o cliente
    api.execute_kw(db, uid, password, 'res.partner', 'unlink', [[customer_id]])
    return 'Success', 200
```

Figura 7.8: Código exemplo da Interface API.

7.4 Certificação da Solução

A adoção da solução encontrada depende do fator legal, ou seja, esta deve estar em conformidade com a legislação portuguesa ou em condições de ser adaptada à mesma, de modo a cumprir os critérios[37] definidos pela Autoridade Tributária e Aduaneira para a certificação de softwares de faturação.

A exportação do SAFT-PT é obrigatória para qualquer *software* de faturação enquanto que a certificação pode ser dispensável mediante diversas condições. São exemplo, o facto de nenhum cliente ser consumidor final, as vendas serem inferiores a 150 mil euros, o numero de faturas emitidas por ano ser inferior a mil unidades.

Através do site oficial do Portal das Finanças, é possível encontrar a documentação sobre este tópico, na forma de três portarias e um despacho, emitidos entre 2010 e 2014, onde são definidas as principais diretivas a cumprir para obter a certificação. Listado abaixo encontra-se um sub-conjunto dos requisitos técnicos:

- **Criação de Documentos** - Todos os documentos com implicações externas devem ser assinados(utilizando por exemplo o algoritmo RSA para criação das chaves) pelo *software*. Deve ter uma ferramenta que permita a exportação do ficheiro SAFT-PT.
- **Controlo de Acessos** - Deve ser utilizado um sistema que permita autenticação e o controlo do acesso ao sistema, protegendo assim a informação sensível.
- **Cópias de Segurança** - Devem ser garantidas periódicas criações de cópias de segurança, ou em alternativa a instalação de vários servidores a operar em simultâneo de forma a minimizar os estragos, em caso de corrupção da base de dados.
- **Numeração** - Deve garantir a sequenciação da numeração, em função da evolução da data e hora da emissão dos documentos.

Versões anteriores do Odoo(v.8) já se encontravam certificadas por algumas empresas, mas durante a duração do estágio nenhuma foi a empresa a obter o certificado para a nova versão. Durante a elaboração deste documento, porém uma empresa obteve a certificação com a versão mais recente. Aparentemente com a utilização de um novo módulo que efetua a assinatura digital nos documentos necessários e o módulo de exportação SAFT-PT o processo de certificação será facilitado. Esta é uma informação bastante valida para a Ubiwhere, no contexto da eventual adoção desta solução encontrada, no âmbito do estágio realizado.

Capítulo 8

Testes

No processo de desenvolvimento de software, é de extrema importância testar o trabalho realizado para garantir que cumpre os requisitos desejados e que o comportamento da aplicação não reserva surpresas.

Neste capitulo encontram-se os resultados do trabalho desenvolvido, onde é possível verificar quais os requisitos que foram satisfeitos e o que ficou por fazer.

8.1 Jbilling

Embora o jBilling não tenha sido adoptado para a construção da solução final, com o intuito de registar e avaliar o trabalho realizado na primeira fase do estágio, foram elaborados os testes funcionais possíveis na plataforma.

As tabelas seguintes mostram os requisitos que foram satisfeitos quer através da interface gráfica ou dos métodos chamados com o uso da sua API.

8.1.1 Estado dos Requisitos

Visto que a plataforma, não permite o acesso à interface gráfica, a outros utilizadores, que não o administrador e as chamadas à API não requerem autenticação de utilizador, os testes aos requisitos de Cliente e Fornecedor não foram realizados.

Identificador	Requisito	Prioridade	Implementado
REQ_ADM_01	Criar cliente.	**Essencial**	**Sim**
REQ_ADM_02	Editar Cliente	**Essencial**	**Sim**
REQ_ADM_03	Apagar cliente.	**Essencial**	**Sim**
REQ_ADM_04	Criar fornecedor.	**Essencial**	Não
REQ_ADM_05	Editar fornecedor	**Essencial**	Não
REQ_ADM_06	Apagar fornecedor.	**Essencial**	Não
REQ_ADM_07	Criar produto/serviço.	**Essencial**	**Sim**
REQ_ADM_08	Editar produto/serviço	**Essencial**	**Sim**
REQ_ADM_09	Apagar produto/serviço.	**Essencial**	**Sim**
REQ_ADM_10	Criar encomenda.	**Essencial**	**Sim**
REQ_ADM_11	Editar encomenda	**Essencial**	**Sim**
REQ_ADM_12	Cancelar encomenda.	**Essencial**	**Sim**
REQ_ADM_13	Gerar fatura com base na encomenda.	**Essencial**	**Sim**
REQ_ADM_14	Enviar fatura via email para o cliente	**Essencial**	**Sim**
REQ_ADM_15	Descarregar fatura em pdf	**Essencial**	**Sim**
REQ_ADM_16	Cancelar Fatura	**Essencial**	**Sim**
REQ_ADM_17	Registar pagamento	**Essencial**	**Sim**
REQ_ADM_18	Registar pagamento parcial	Desejável	**Sim**
REQ_ADM_19	Efetuar Pagamentos via PayPal	Desejável	Não
REQ_ADM_20	Consultar relatórios de vendas	Desejável	Não
REQ_ADM_21	Definir comissão nos produtos/serviços	Desejável	Não
REQ_ADM_22	Exportar ficheiro SAFT-PT	Desejável	Não
REQ_ADM_23	Gerir compras aos fornecedores	Desejável	Não
REQ_ADM_24	Gestão de permissões para diferentes utilizadores	Desejável	Não

Tabela 8.1: Estado dos requisitos de Administrador, Jbilling

Identificador	Requisito	Prioridade	Implementado
REQ_API_01	Conectar-se ao servidor Jbilling	**Essencial**	**Sim**
REQ_API_02	Autenticar utilizador	**Essencial**	Não
REQ_API_03	Criar Cliente	**Essencial**	**Sim**
REQ_API_04	Editar Cliente	**Essencial**	**Sim**
REQ_API_05	Apagar Cliente	**Essencial**	Não
REQ_API_06	Criar Fornecedor	**Essencial**	Não
REQ_API_07	Editar Fornecedor	**Essencial**	Não
REQ_API_08	Apagar Fornecedor	**Essencial**	Não
REQ_API_09	Criar Produto/Serviço	**Essencial**	**Sim**
REQ_API_10	Editar Produto/Serviço	**Essencial**	**Sim**
REQ_API_11	Apagar Produto/Serviço	**Essencial**	Não
REQ_API_12	Criar Encomenda	**Essencial**	**Sim**
REQ_API_13	Editar Encomenda	**Essencial**	Não
REQ_API_14	Cancelar Encomenda	**Essencial**	Não
REQ_API_15	Gerar Fatura com base na Encomenda	**Essencial**	**Sim**
REQ_API_16	Enviar Fatura via email para o Cliente	**Essencial**	Não
REQ_API_17	Cancelar Fatura	**Essencial**	Não
REQ_API_18	Registar Pagamento	**Essencial**	Não

Tabela 8.2: Estado dos requisitos da API, Jbilling

Os requisitos da API neste caso devem ser interpretados tendo em conta só a chamada ao servidor Jbilling e não a sua interação com a SmartCity API.

8.2 Odoo

Esta secção contém os testes realizados à solução desenvolvida através da plataforma Odoo.

É composta pelos testes funcionais à aplicação, bem como algumas observações sobre os requisitos não funcionais.

8.2.1 Estado dos Requisitos

Com a solução desenvolvida é necessário fazer a validação da mesma, nesse sentido foram feitos os testes de aceitação referentes aos requisitos definidos para a plataforma.

As tabelas abaixo representam o estado dos requisitos, bem como a sua prioridade para o projeto.

O requisito de exportação dos ficheiros SAFT-PT foi validado não só pela extração do respetivo ficheiro através da plataforma do Odoo, mas também com a sua validação através da plataforma oficial do portal das finanças que permite garantir a correta formatação do documento.

O único requisito que não foi realizado foi o envio de faturas via email para os clientes através da Interface API, embora este requisito tenha sido conseguido através da plataforma Odoo diretamente a sua aplicação via Interface API não foi concluída. Embora este requisito tivesse prioridade essencial, devido a um atraso no desenvolvimento já não foi possível conseguir tê-lo pronto a tempo. No entanto, para trabalho futuro, não será certamente muito complexo adicionar esta característica ao projeto.

Identificador	Requisito	Prioridade	Implementado
REQ_CLI_01	Editar sua conta	**Essencial**	**Sim**
REQ_CLI_02	Criar encomenda.	**Essencial**	**Sim**
REQ_CLI_03	Editar encomenda	**Essencial**	**Sim**
REQ_CLI_04	Cancelar encomenda.	**Essencial**	**Sim**
REQ_CLI_05	Gerar Fatura com base na Encomenda	**Essencial**	**Sim**
REQ_CLI_06	Receber fatura via email	**Essencial**	**Sim**
REQ_CLI_07	Descarregar fatura em pdf	**Essencial**	**Sim**
REQ_CLI_08	Fazer Pagamento	**Essencial**	**Sim**

Tabela 8.3: Estado dos requisitos de Cliente, Odoo

Identificador	Requisito	Prioridade	Implementado
REQ_ADM_01	Criar cliente.	**Essencial**	**Sim**
REQ_ADM_02	Editar Cliente	**Essencial**	**Sim**
REQ_ADM_03	Apagar cliente.	**Essencial**	**Sim**
REQ_ADM_04	Criar fornecedor.	**Essencial**	**Sim**
REQ_ADM_05	Editar fornecedor	**Essencial**	**Sim**
REQ_ADM_06	Apagar fornecedor.	**Essencial**	**Sim**
REQ_ADM_07	Criar produto/serviço.	**Essencial**	**Sim**
REQ_ADM_08	Editar produto/serviço	**Essencial**	**Sim**
REQ_ADM_09	Apagar produto/serviço.	**Essencial**	**Sim**
REQ_ADM_10	Criar encomenda.	**Essencial**	**Sim**
REQ_ADM_11	Editar encomenda	**Essencial**	**Sim**
REQ_ADM_12	Cancelar encomenda.	**Essencial**	**Sim**
REQ_ADM_13	Gerar fatura com base na encomenda.	**Essencial**	**Sim**
REQ_ADM_14	Enviar fatura via email para o cliente	**Essencial**	**Sim**
REQ_ADM_15	Descarregar fatura em pdf	**Essencial**	**Sim**
REQ_ADM_16	Cancelar Fatura	**Essencial**	**Sim**
REQ_ADM_17	Registar pagamento	**Essencial**	**Sim**
REQ_ADM_18	Registar pagamento parcial	Desejável	**Sim**
REQ_ADM_19	Efetuar Pagamentos via PayPal	Desejável	**Sim**
REQ_ADM_20	Consultar relatórios de vendas	Desejável	**Sim**
REQ_ADM_21	Definir comissão nos produtos/serviços	Desejável	**Sim**
REQ_ADM_22	Exportar ficheiro SAFT-PT	Desejável	**Sim**
REQ_ADM_23	Gerir compras aos fornecedores	Desejável	**Sim**
REQ_ADM_24	Gestão de permissões para diferentes utilizadores	Desejável	**Sim**

Tabela 8.4: Estado dos requisitos de Administrador, Odoo

Identificador	Requisito	Prioridade	Implementado
REQ_API_01	Conectar-se ao servidor Odoo	**Essencial**	**Sim**
REQ_API_02	Autenticar utilizador	**Essencial**	**Sim**
REQ_API_03	Criar Cliente	**Essencial**	**Sim**
REQ_API_04	Editar Cliente	**Essencial**	**Sim**
REQ_API_05	Apagar Cliente	**Essencial**	**Sim**
REQ_API_06	Criar Fornecedor	**Essencial**	**Sim**
REQ_API_07	Editar Fornecedor	**Essencial**	**Sim**
REQ_API_08	Apagar Fornecedor	**Essencial**	**Sim**
REQ_API_09	Criar Produto/Serviço	**Essencial**	**Sim**
REQ_API_10	Editar Produto/Serviço	**Essencial**	**Sim**
REQ_API_11	Apagar Produto/Serviço	**Essencial**	**Sim**
REQ_API_12	Criar Encomenda	**Essencial**	**Sim**
REQ_API_13	Editar Encomenda	**Essencial**	**Sim**
REQ_API_14	Cancelar Encomenda	**Essencial**	**Sim**
REQ_API_15	Gerar Fatura com base na Encomenda	**Essencial**	**Sim**
REQ_API_16	Enviar Fatura via email para o Cliente	**Essencial**	Não
REQ_API_17	Cancelar Fatura	**Essencial**	**Sim**
REQ_API_18	Registar Pagamento	**Essencial**	**Sim**

Tabela 8.5: Estado dos requisitos da API, Odoo

Identificador	Requisito	Prioridade	Implementado
REQ_FOR_01	Editar sua conta	**Essencial**	**Sim**
REQ_FOR_02	Criar produto/serviço	**Essencial**	**Sim**
REQ_FOR_03	Editar produto/serviço	**Essencial**	**Sim**
REQ_FOR_04	Eliminar produto/serviço	**Essencial**	**Sim**

Tabela 8.6: Estado dos requisitos de Fornecedor, Odoo

8.2.2 Requisitos não funcionais

Acerca dos requisitos não funcionais, infelizmente o estagiário não conseguiu realizar os testes de performance que tinha planeado, mas mesmo assim ainda é possível tirar algumas conclusões sobre a performance do servidor Odoo através de informação recolhida em investigação.

Performance Servidor Odoo

A performance do Odoo está em grande parte ligada à base de dados PostgreSQL. Para implementar o Odoo em apenas um servidor, com aproximadamente 100 utilizadores ativos, as especificações de hardware recomendadas[38](dados de 2014) são as seguintes:

- Processador: Intel Xeon E5 2.5Ghz 6c/12t
- Memória: 32GB RAM
- Armazenamento: SATA/SAS em RAID-1

Com estas especificações, no serviço de alojamento remoto do Odoo o sistema consegue suportar cerca de 3000 bases de dados com uma carga média inferior ou igual a três e com tempos de leitura tipicamente a rondar os 100 milisegundos.

Para casos em que seja necessário escalar[39] o sistema para melhorar a performance é aconselhável escalar a base de dados verticalmente, e o servidor que trata dos pedidos escalar horizontalmente juntamente de um load balancer.

Segurança

Com a implementação do HTTP Basic Auth na Interface API para executar a autenticação dos utilizadores da API com as contas respetivas no servidor do Odoo, não só se verifica um sistema de autenticação que limita o acesso a utilizadores não autorizados a aceder à API, como permite que o servidor do Odoo faça o controlo do acesso à informação com base no utilizador.

Devido ao facto de a Interface API ser utilizada somente pela SmartCityAPI e não diretamente pelos clientes e fornecedores, não foi implementado um sistema de autenticação mais complexo.

Capítulo 9

Conclusão e Trabalho Futuro

Este capitulo serve de reflexão sobre o trabalho realizado, a sua futura aplicabilidade e os conhecimentos adquiridos.

O objetivo principal deste estágio foi cumprido. Foi desenvolvida a solução pretendida respeitando os requisitos necessários e com a devida documentação para uma mais fácil adoção da mesma. Apesar dos iniciais percalços com a escolha da tecnologia e alguns atrasos no desenvolvimento, a solução final foi um sucesso.

Embora a SmartCityAPI não esteja num estado muito avançado de desenvolvimento, com a conclusão deste trabalho que servia também um pouco como prova de conceito do módulo de faturação, creio que o seu desenvolvimento será acelerado e motivará outros projetos da empresa a adotar esta solução em situações de requisitos semelhantes.

O estagiário desenvolveu conhecimentos na área dos softwares de faturação, apoio à gestão de negócio e desenvolvimento e integração de APIs, mas a experiência do estágio também foi muito enriquecedora para além das fronteiras deste projeto. Devido a diversas iniciativas internas da Ubiwhere em que foi promovida a partilha de informação sobre os projetos em desenvolvimento na empresa não só entre os estagiários mas sim a empresa como um todo. Em particular a grande interação com todos os outros estagiários, permitiu que fossem debatidas temáticas referentes a todos os projetos e com isso uma maior variedade de conhecimentos foram adquiridos.

Bibliografia

[1] B. Koch, "*E-Invoicing / E-Billing International Market Overview and Forecast*," 2014.

[2] D. B. Kim Boes and A. Inversini, "*Conceptualising Smart Tourism Destination Dimensions*," 2015.

[3] S. C. E. W. Congress, "*Smart City Expo World Congress 2014 Report*," 2014.

[4] J. Kim and S. Rohmer, "*Electronic billing vs. paper billing: Dematerialization, energy consumption and environmental impacts*," 2012.

[5] M. Wright, "*Striata discusses essence of 2013 Billentis report on e-invoicing, e-billing*," 2013.

[6] B. Koch, "*E-Invoicing/E-Billing - The catalyst for AR/AP automation*," 2013.

[7] OberCom, "*A Internet em Portugal Sociedade em Rede 2014*," 2014.

[8] J. Hall, "*Open Source in the Enterprise: Billing and Payment Processing Systems*," 2012.

[9] CitrusDB, "`https://launchpad.net/citrusdb`," 2014.

[10] phpCOIN, "`https://www.phpcoin.com/`," 2014.

[11] AgileBill, "`https://github.com/tony-landis/agilebill`," 2014.

[12] P. Colibri, "`www.projectocolibri.com`," 2014.

[13] Jbilling, "`http://www.jbilling.com`," 2014.

[14] F. I. Services, "`http://www.freeside.biz/freeside/`," 2014.

[15] KillBill, "`http://killbill.io/`," 2014.

[16] WorldPay, "*Your Global Guide to Alternative Payments*," 2014.

[17] Z. Hui-min, "*The study on the influential factors of electronic payment system adoption*," 2009.

[18] Ingenico, "*Electronic payment architecture and trends in Europe*," 2012.

[19] PayPal, "`https://www.paypal.com/home`," 2014.

[20] Braintree, "`https://www.braintreepayments.com/`," 2014.

[21] Stripe, "`https://stripe.com/`," 2014.

[22] Authorize.Net, "`http://www.authorize.net/`," 2014.

[23] B. Payments, "`http://balancedpayments.com/`," 2014.

[24] Sebastien, "Payment processing landscape in europe," 2013.

[25] E. jBilling Software, "*jBilling Extension Guide*," 2013.

[26] A. Gupta, "Integrate jbilling with your own application," 2014.

[27] Odoo, "`https://www.odoo.com/`," 2015.

[28] O. Community, "`https://www.odoo.com/page/community`," 2015.

[29] A. T. e Aduaneia, "`http://info.portaldasfinancas.gov.pt/pt/apoio_contribuinte/NEWS_SAF-T_PT.htm`," 2015.

[30] P. a Programar, "`http://www.portugal-a-programar.pt/topic/57541-saft-pt-debate-de-duvidas-e-ideias/`," 2015.

[31] Communities-Communications, "`https://github.com/Communities-Communications/l10n_pt_saft`," 2015.

[32] Oracle, "*JD Edwards EnterpriseOne Applications Localizations for Portugal Implementation Guide*," 2014.

[33] OpenERP, "*OpenERP Server Developers Documentation*," 2015.

[34] N. Bessi, "*Odoo new API guideline*," 2015.

[35] Flask, "`http://flask.pocoo.org/`," 2015.

[36] Swagger, "`www.swagger.io`," 2015.

[37] A. T. Aduaneira, "`http://info.portaldasfinancas.gov.pt/pt/apoio_contribuinte/CertificacaoSoftware.htm`," 2014.

[38] Odoo, "*Improving the performance of Odoo deployments*," 2014.

[39] Odoo, "*OpenERP Benchmark: How to test performance and robustness against your volumes?* ," 2013.