João Manuel Torres Delgado

Sistema de faturação eletrónica e de pagamentos móveis

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Setembro de 2015

João Manuel Torres Delgado

Sistema de faturação eletrónica e de pagamentos móveis

Dissertação submetida à Faculdade de Ciências da
Universidade do Porto como parte dos requisitos para a obtenção do grau de
Mestre em Engenharia de Redes em Sistemas Informáticos

Orientador: Prof. Michel Ferreira (FCUP)
Co-orientador: Prof. Luís Damas (*Geolink*)

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Setembro de 2015

Este trabalho foi financiado por I-CITY - ICT for Future Mobility (NORTE-07-0124-FEDER-000064).

Agradeço também a hospitalidade que encontrei na Geolink.

Resumo

Este trabalho é um contributo para o desenvolvimento de um sistema de faturação eletrónica a ser usado em ambiente de mobilidade, mais especificamente em serviços de táxi.

Contém o enquadramento legal de um sistema de faturação eletrónica e a identificação de requisitos gerais de um tal sistema. São também identificados desafios específicos associados a sistemas de faturação a ser usados em ambiente de mobilidade, onde não há garantia de comunicações estáveis.

Apresenta uma arquitetura possível para o sistema de faturação, descrevendo como é que esta pode ser usada para a emissão de faturas num táxi de forma a mitigar e contornar os problemas de comunicação associados a uma rede móvel.

Foi feito também trabalho inicial para integrar pagamentos móveis num serviço de táxi tradicional. Foi desenvolvido um esquema de comunicação entre as várias partes envolvidas por forma a proporcionar um método de pagamento mais conveniente.

Abstract

This work is a contribute to the development of an electronic invoicing system to be used in a mobility enviroment, specifically during taxi services.

It contains the legal framework of an electronic invoicing system and the identification of the general requirements of such systems. This work also identifies specific challenges associated with invoicing systems to be used in a mobility enviroment, where of the stability of the communications is not guaranteed.

This thesis also presents a possible architecture to the invoicing system, describing how it can be used to issue invoices in a taxi so as to mitigate and bypass the problems related to a mobile network.

Inicial work has been made to integrate mobile payments in the tradicional taxi service. We developed a communication scheme with all the involved parties in order to provide the customer with a convenient payment method.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Este capítulo começa com uma secção em que se pretende motivar o desenvolvimento do sistema de faturação aqui considerado e que se destina a ser usado em táxis. Segue-se uma secção em que se explicitam os objetivos e se fazem breves referências à forma como o projeto foi abordado. Para terminar inclui-se uma visão global da estrutura da tese.

1.1 Motivação

A mobilidade em zonas urbanas é muito constrangida pela ineficiência com que os transportes são usados, tanto os públicos como os privados. Entre outros aspetos, isto traz grandes problemas de congestionamento, poluição e gasto desnecessário de tempo, bem como as consequentes perdas económicas.

Esta ineficiência deve-se, em parte, à baixa taxa de ocupação dos veículos. Por exemplo, cerca de 82% dos serviços de táxi da cidade do Porto destinam-se a transportar um único passageiro [13].

Qualquer contribuição no sentido de diminuir a ineficiência referida seria útil, sendo esta uma das motivações deste trabalho. Mais precisamente, procuraremos que este trabalho seja uma contribuição para tornar mais eficiente uma rede de táxis.

1.2 Objetivos e Abordagem

O objetivo deste projeto é estudar, em parceria com a *Geolink*, aplicações que contribuam para uma utilização eficiente de uma rede de táxis.

O projeto passa pelo desenvolvimento de um sistema de faturação que poderá, desde logo, cumprir o objetivo de disponibilizar um serviço de emissão de faturas em táxis. Além disso deverá poder servir como ferramenta de apoio (*testbed*) para estudar a viabilidade de modelos de *taxi-sharing* na rede de táxis atual.

O sistema desenvolvido deve então conseguir o grau de flexibilidade e controlo necessário ao estudo, assim como cumprir os requisitos técnicos e ter a certificação necessária à emissão de documentos comerciais, de acordo com a legislação em vigor.

A implementação de um serviço de *taxi-sharing* implica também uma agilização dos métodos de pagamento. Neste sentido devem ser estudadas formas de integração e possível implementação de pagamentos móveis num serviço de táxi.

1.3 Visão Global

Depois deste capítulo introdutório faz-se, neste trabalho, uma contextualização do documento "fatura eletrónica", qual a sua função e características. É também feito o enquadramento legal de um sistema de faturação eletrónica e são identificados os principais desafios associados ao desenvolvimento de um destes sistemas para táxis.

No Capítulo 3 são apontados requisitos que uma base de dados de um sistema de faturação precisa de satisfazer e como isso pode ser conseguido.

No quarto capítulo é abordado o tema da emissão de faturas num ambiente móvel. É feita a comparação de duas estratégias para esta emissão, analisando vantagens e limitações.

O Capítulo 5 é dedicado a sistemas de pagamentos móveis. Começa com uma análise geral sobre o funcionamentos destes sistemas e termina descrevendo uma possível arquitetura para a integração destes nos táxis.

Capítulo 2

Enquadramento

Este capítulo começa com uma breve descrição de uma fatura. Explica em que é que este documento consiste, assim como a sua função. Faz a ponte entre uma fatura tradicional e uma fatura eletrónica, destacando características específicas desta última.

É um requisito legal a emissão de faturas em *software* certificado, assim como a sua comunicação à Autoridade Tributária. A estes aspetos são dedicadas as duas secções seguintes.

Por fim, são identificados alguns dos desafios associados a um sistema de faturação, em especial num ambiente móvel como é o caso de um táxi.

2.1 O que é uma Fatura?

Uma fatura é um comprovativo da venda de bens ou serviços. Trata-se de um documento comercial emitido pelo prestador de serviços ou transmissor de bens e indica os produtos transacionados, as quantidades, preços acordados, etc. É um importante instrumento no combate à economia informal, fraude e evasão fiscais, tendo a sua emissão e comunicação à Autoridade Tributária e Aduaneira (AT) tomado caráter obrigatório a 1 de Janeiro de 2013 [10].

De forma a cumprir estes fins, uma fatura tem de conter obrigatoriamente:

- número de identificação fiscal do emissor;
- número identificador do documento;

- data de emissão;
- tipo de documento (*e.g.* fatura ou fatura simplificada);
- número de identificação fiscal do adquirente;
- valor tributável da prestação de serviços ou da transmissão de bens;
- taxas aplicáveis;
- montante de IVA liquidado.

2.2 Fatura Eletrónica

Uma fatura eletrónica é semelhante a uma fatura tradicional, com o mesmo valor legal, mas desmaterializada num formato eletrónico. Este formato possui óbvias vantagens a nível de comunicação, processamento e armazenamento. Não surpreende, portanto, que a emissão e comunicação de faturas em formato eletrónico fosse tornada obrigatória a 1 de Janeiro de 2013 [10, Artigo 3.º].

Estes documentos têm, à semelhança das faturas tradicionais, de conter os itens obrigatórios referidos anteriormente, assim como satisfazer as condições específicas adicionais exigidas para garantir a autenticidade e integridade do seu conteúdo. Isto é conseguido com recurso ao já existente sistema de identificação rígida (identificador único da fatura), juntamente com assinaturas digitais e com a obrigatoriedade de utilização um sistema de emissão de faturas certificado pela AT.

2.2.1 Identificador Único da Fatura

Um dos itens mais importantes de uma fatura é o seu número identificador. Este permite identificar unicamente todos os documentos emitidos por um sistema de faturação e é constituído por:

- o código interno do tipo de documento atribuído pela aplicação;
- o identificador da série do documento;
- número sequencial do documento dentro da série.

Um exemplo de um possível número de fatura é a sequência `FT 001/14`.

O código interno deve identificar o tipo de documento dentro do sistema de faturação, não podendo ser utilizado o mesmo código para documentos de tipo diferente.

A série de cada documento é um campo que pode ser preenchido de acordo com as necessidades de cada aplicação.

O último campo do identificador de uma fatura é um número que deve evoluir sequencialmente de acordo com a data e a hora de emissão de documentos da mesma série. É essencial garantir esta sequencialidade uma vez que é uma propriedade fundamental para a validação das faturas emitidas, em particular para a verificação da assinatura digital de cada documento. Esta verificação será discutida na secção seguinte.

2.2.2 Assinatura Digital

A assinatura digital de uma fatura é o mecanismo que permite garantir a integridade e não repúdio de uma fatura e é conseguido com recurso a criptografia de chave pública e a funções de *hash*.

Esta assinatura é constituída por uma mensagem contendo [11]:

- data de emissão;
- data de entrada no sistema;
- número identificador;
- total do documento, com impostos;
- assinatura do documento anterior, na mesma série, se existir.

Estes campos são concatenados e separados por um "`;`". Um exemplo de uma destas mensagens, usando o identificador do exemplo anterior, pode ser:

```
2013-07-03;2013-07-03T14:25:00;FT 001/14;1500.00;mYJEv4iGwLcnQbRD7d
Ps2uD1mX08XjXIKcGg3GEHmwMhmmGYusffIJjTdSITLX+uujTwzqmL/U5nvt6S9s8ij
N3LwkJXsiEpt099e1MET/8y3+Y1bN+K+YPJQiVmlQS0fXETsOPo8SwUZdBALt0vTo1V
hUZKejACcjEYJG6nI=
```

Num segundo passo, esta mensagem é processada por uma função de *hash* criptograficamente segura.

Uma função de *hash* é uma função que toma um *input* (pré-imagem) de comprimento variável e o converte para um *output* de tamanho fixo (valor de *hash*) [30, Capítulo 2]. Para que seja considerada criptograficamente segura, esta função deve também satisfazer também as seguintes condições [20, Capítulo 9]:

Resistência à pré-imagem É impraticável encontrar qualquer x tal que $f(x) = y$ para qualquer valor de *hash* y para o qual não se conheça a pré-imagem correspondente.

Resistência à segunda pré-imagem É impraticável encontrar qualquer segunda pré-imagem que tenha o mesmo valor de *hash* que qualquer outra pré-imagem especificada.

Resistência à colisão É impraticável encontrar duas pré-imagens distintas que produzam o mesmo valor de *hash*.

Uma função que satisfaça estas propriedades permite produzir uma "impressão digital" de uma pré-imagem. Ou seja, é possível validar, com um grande grau de confiança, que uma determinada mensagem foi usada para produzir um certo valor de *hash* [30, Capítulo 2].

No processo de assinar uma fatura, a função de *hash* usada é a SHA-1 [14], como estipulado pela AT em [11].

O valor de *hash* calculado é então cifrado com recurso a criptografia de chave pública. Neste tipo de criptografia é utilizado um par chaves (tipicamente chamadas de chave pública e chave privada) e tem a propriedade de que uma mensagem cifrada com uma das chaves pode ser decifrada apenas com a outra [30, Capítulo 3].

No processo de certificação de *software* de emissão de faturas, é gerado um par de chaves. Uma das chaves é partilhada com a AT (chave pública) enquanto a outra é mantida secreta, sendo conhecida apenas pelo *software* certificado. Depois desta partilha de chaves, é possível cifrar uma mensagem com a chave privada (assinar) e transmitir a mensagem cifrada à AT que, por sua vez, a pode decifrar com a chave pública.

Esta assinatura criptográfica traz várias vantagens, em particular:

- impossibilita que sejam comunicadas à AT faturas que não tenham sido emitidas no *software* desenvolvido;
- permite à AT ter garantias de que as faturas comunicadas tiveram origem em *software* previamente certificado.

No processo de assinar uma fatura, a algoritmo de criptografia de chave pública usado é o RSA [28], como estipulado pela AT em [11].

O valor de *hash* assinado criptograficamente é finalmente codificado em Base64 [19]. Isto converte a informação binária resultante da assinatura numa representação textual. O resultado destas operações constitui a assinatura do documento.

Os documentos assinados nos termos descritos devem ter impresso, além do número identificador do documento, um conjunto de quatro carateres da assinatura calculada, correspondentes à 1.ª, 11.ª, 21.ª e 31.ª posições, assim como o número do certificado atribuído ao programa utilizado[11].

Quando uma fatura é declarada, é transmitida à AT, entre outras coisas, a sua assinatura, toda a informação que nela foi codificada (com a exceção da assinatura do documento anterior), assim como informação sobre o *software* que foi usado para emitir a fatura.

A AT pode então decifrar a assinatura da fatura com recurso à chave pública associada ao *software* que a emitiu, obtendo o valor de *hash* que foi cifrado. Pode também reconstruir a mensagem inicial usada para assinar o documento, concatenando a assinatura do ultimo documento transmitido. Com isto, pode comparar o valor de *hash* da mensagem reconstruida com aquele cifrado na assinatura do documento e verificar se são iguais, assim como cruzar informação com os dados presentes na fatura impressa.

Com isto a AT tem garantias de que a origem e valores de cada fatura estão corretos, assim como que todas as faturas (até à última transmitida) foram declaradas.

Torna-se claro aqui a importância da numeração sequencial das faturas. Sem isto, não seria possível à AT reconstruir a mensagem usada para assinar cada fatura.

2.3 Certificação de um Sistema de Faturação

O processo de certificação de um sistema de faturação começa com a submissão ao Portal das Finanças [22] de uma declaração de certificação de programa de faturação [12]. Esta submissão é realizada via Portal das Finanças e deve ser acompanhada de uma cópia da chave pública usada pela aplicação para assinar as faturas emitidas.

Depois desta submissão, é iniciado um processo de validação da aplicação. Numa fase inicial são verificados exemplos de documentos emitidos por esta (faturas impressas assim como o ficheiro SAF-T (PT) correspondente). Na fase seguinte, há lugar a uma reunião presencial onde são realizados testes de conformidade, garantindo que a aplicação cumpre todos os requisitos técnicos exigidos [11, 24].

Terminado com sucesso este processo de validação, é atribuído à aplicação o número do certificado, podendo, a partir daí, ser instalada e comercializada.

2.4 Comunicação de Faturas à Autoridade Tributária

A comunicação à AT das faturas emitidas é obrigatória [10, Artigo 3.º] e pode ser feita de várias formas [3]:

- por transmissão eletrónica em tempo real e integrada com o sistema de faturação, utilizando o *Webservice* disponibilizado pela AT;
- através do envido do ficheiro SAF-T (PT), que deve ser comunicado com uma periodicidade pelo menos mensal;
- por inserção direta no Portal das Finanças;
- por via eletrónica, através da submissão do modelo oficial de declaração para a comunicação de elementos das faturas [25].

No contexto do desenvolvimento de um sistema de faturação eletrónica, consideraremos as duas primeiras alternativas.

Webservice

A comunicação por *Webservice* é feita com recurso ao protocolo SOAP [33], utilizando o modelo de dados definido em WSDL [7] pela AT [34] e começa assim que uma fatura é emitida ou integrada no programa de faturação.

Para que o programa de faturação consiga comunicar a fatura emitida, deve solicitar também à empresa (sujeito passivo) as suas credenciais no Portal das Finanças. Estas devem ser encriptadas recorrendo à chave pública do sistema de autenticação utilizado pelo Portal das Finanças.

Com as informações de autenticação e da fatura emitida, o sistema de faturação pode construir o pedido SOAP com estes dados e proceder ao envido, sob HTTPS [27], ao Portal das Finanças.

SAF-T (PT)

SAF-T (PT) [23] é um formato de ficheiro normalizado, na linguagem XML [4], e deve conter toda a informação relativa aos documentos emitidos.

Para comunicar os dados das faturas com recurso a este ficheiro, a empresa terá de extrair este ficheiro no sistema de faturação e, em seguida, proceder ao envio deste à AT. Isto deve ser feito pelo uma vez por mês, podendo ser fracionado em períodos mais curtos, dependendo do volume de faturas emitidas.

O sistema de faturação deve por isso possibilitar a extração deste ficheiro por mês ou por um outro período (por dia, por semana, etc.).

2.5 Desafios

Um sistema de emissão de faturas certificado necessita de estar de acordo com a legislação em vigor e de cumprir uma série de requisitos técnicos [11, 24]. Estes desafios são ampliados pelas características móveis dos táxis.

De uma forma geral, o sistema de faturação deve conseguir:

- emitir faturas de forma concorrente, garantindo a validade da numeração e da assinatura de todos os documentos emitidos;

- assegurar a validade, consistência e imutabilidade da informação armazenada relativa aos documentos emitidos;
- conseguir assegurar, com um grande grau de confiança, que todos os documentos emitidos são preservados.

Seguem-se breves explicações para cada um dos pontos referidos.

Pedidos concorrentes

Neste cenário, dois táxis distintos terminam um serviço na mesma altura e pretendem emitir uma fatura simultaneamente.

Neste caso, o sistema tem ser capaz de responder a estes pedidos concorrentes, emitindo uma fatura distinta para cada um dos serviços de táxi. Para conseguir isto, é necessário garantir que é atribuído um número identificador válido a cada documento, como descrito na Secção 2.2.1, e também que cada documento é corretamente assinado.

O estado do sistema, depois de emitidas as duas faturas, deve ser o mesmo que se tivesse sido emitidas duas faturas sequencialmente.

Imutabilidade dos documentos

Cada documento deve ser imutável, ou seja, depois emitido, nenhuma informação pode ser alterada.

Tomemos o exemplo de alterações ao IVA. Em Portugal, os vários produtos e serviços têm uma taxa de IVA associada. Esta pode ser uma de três (taxa reduzida, taxa intermédia e taxa normal) e cada uma representa uma percentagem do valor bruto do produto, a qual é adicionada ao valor bruto para a obtenção do custo total. Este valor influencia não só o custo de cada produto presente numa fatura, mas também o seu total. No entanto, nem a percentagem associada a cada escalão, nem o escalão associado a cada produto/serviço são fixos, podendo variar ao longo do tempo.

Este tipo de alteração não deve afetar os documentos já emitidos. Além disso, é importante conseguir recriar qualquer documento em qualquer altura, devendo ser possível transformar a informação guardada num documento idêntico àquele que foi impresso aquando da sua emissão.

Preservação dos documentos

Todos os documentos emitidos devem preservados, mesmo depois destes serem comunicados à AT. Isto é importante não só para análise interna e por parte das empresas que utilizem o sistema de faturação, como também no caso de uma auditoria externa.

Esta preservação de dados deve ser garantida mesmo em caso de falha inesperada do sistema, que pode acontecer tanto por erros no *software* desenvolvido como também por falha de *hardware*.

É importante então a existência de uma política de replicação ou *backups* regulares, assim como mecanismos que permitam recuperar (e se necessário reintroduzir) dados perdidos.

Capítulo 3

Base de dados

Uma base de dados é um meio de guardar informação de forma a que esta possa ser preservada, recuperada e consultada mais tarde e representa uma peça crucial de um sistema de faturação.

Este capítulo começa por identificar alguns requisitos que a base de dados de um sistema de faturação precisa de satisfazer.

É depois feita uma uma breve análise de alguns sistemas de gestão de base de dados e como estes podem satisfazer os requisitos apontados. Esta é seguida da identificação do sistema escolhido para este projeto.

É dedicada também uma secção à garantia da imutabilidade dos documentos armazenados na base de dados escolhida.

O capítulo termina com uma descrição do esquema da base de dados do sistema de faturação desenvolvido.

3.1 Requisitos

Uma base de dados para um sistema de faturação precisa de satisfazer alguns requisitos, que listamos a seguir. A cada um desses requisitos associamos algumas observações.

Durabilidade Tem de garantir a durabilidade dos dados inseridos. Por exemplo, sempre que uma fatura seja inserida na base de dados com sucesso, deve haver

a garantia de que os dados são guardados permanentemente, mesmo em caso de falha de corrente ou *crash* do sistema.

Consistência Tem de assegurar que a base de dados se mantém num estado consistente no final de cada operação. Se esta resultar em erro, deve ser capaz de mesmo reverter todas as alterações feitas. Isto é importante para garantir que todos os documentos armazenados mantêm a sua integridade.

Concorrência Precisa garantir que o acesso concorrente à base de dados não introduz erros. Deve, por exemplo, ser capaz de responder a dois pedidos de armazenamento de faturas feitos em simultâneo de forma a que cada uma seja guardada sem erros.

Um outro ponto importante é a replicação dos dados guardados. Apesar da garantia de durabilidade, o sistema de base de dados estará sempre sujeito a falhas de *hardware*. Para diminuir os problemas nestes casos, os dados deveram ser copiados para um disco redundante em tempo real. Este não é necessariamente um requisito da base de dados, uma vez que pode ser conseguido com recurso técnicas de espelhamento do disco, como por exemplo com RAID 1 [6]. No entanto, mesmo não sendo um requisito, o próprio sistema de base de dados pode disponibilizar mecanismos de replicação.

3.2 Sistema de Gestão de Bases de Dados

Um sistema de gestão de bases de dados (DBMS) é responsável por gerir como a informação é guardada, mantida e recuperada.

Estes podem ser sub categorizados de acordo com o modelo de dados que usam, ou seja, a forma como organizam a informação, dividindo-se principalmente em dois grandes grupos: bases de dados relacionais (RDBMS) e bases de dados não relacionais (*NoSQL* DBMS).

3.2.1 Sistemas de Gestão de Bases de Dados Relacionais

Sistemas de gestão de bases de dados relacionais são o tipo de DBMS mais usado[1]. São caracterizados por se basearem no modelo de dados relacional definido por E. F. Codd

[1]Baseado no ranking disponível em: `http://db-engines.com/en/ranking` (acedido em 24/09/2015)

em [8] e, na sua generalidade, implementam do *standard* SQL [17].

Neste tipo de DBMS, os dados são organizados numa coleção de tabelas, cada uma representando um conceito, entidade ou objeto. Os atributos destas entidades são dispostos pelas colunas de cada tabela e cada linha contém os vários valores guardados na base de dados.

Cada atributo, ou conjunto de atributos, está também afeto de um conjunto de restrições. Algumas destas, implícitas, definem o domínio de cada atributo, mas podem também ser definidas outras restrições como garantir a unicidade e existência (não nulo) de cada valor. É também possível relações entre as várias tabelas através de restrições de chave externa.

À lista das várias tabelas representadas na base de dados, aos seus atributos e às restrições associadas a cada uma delas dá-se o nome de esquema da base de dados [15, Capítulo 2]. É da responsabilidade do sistema de gestão de base de dados fazer cumprir este esquema, garantindo que a base de dados se encontra sempre num estado que não viole nenhuma das restrições definidas.

Este esquema é definido, assim como a generalidade das interações com a base de dados, com recurso à linguagem SQL. De uma forma simplificada, SQL permite modificar a base de dados através da criação, modificação e eliminação de tabelas, das linhas nela contidas e das várias restrições que as afetam. Permite também pesquisar, filtrar e relacionar os dados das diferentes tabelas.

Este tipo de DBMS é caracterizado também pelas suas fortes garantias de consistência e integridade, conseguidas através da implementação de transações ACID. Uma transação corresponde à execução de uma ou mais operações na base de dados como por exemplo ler, inserir ou modificar valores nela contidos. Estas transações devem conseguir satisfazer as seguintes propriedades descritas em [16].

Atomicidade Uma transação comporta-se como uma operação atómica. Ou seja, ou todas as operações numa transação são completadas com sucesso ou então nenhuma é persistida.

Consistência Esta propriedade garante que, no final de uma transação, a base de dados se encontra consistente de acordo com o esquema definido.

Isolamento Garante que duas transações concorrentes são executadas de forma independente. Num cenário de isolamento forte, as alterações de feitas por uma transação não são visíveis a outras transações concorrentes antes desta terminar.

Vejamos o exemplo em que uma transação modifica, entre outras operações, um atributo de uma linha. Esta alteração não deve ser visível a qualquer outra transação concorrente que pretenda ler essa mesma linha antes da primeira terminar com sucesso.

Durabilidade Significa que quando uma transação termina, os dados inseridos e modificados por esta foram persistidos, mesmo na eventualidade de uma perda de corrente ou *crash* do DBMS imediatamente a seguir.

Por si só, estas transações garantem que o DBMS satisfaz quase todos os requisitos apontados. Em particular, aborda os problemas de durabilidade, consistência e isolamento apontados.

Este tipo de DBMS tem, no entanto, algumas limitações. O esquema da base de dados é rígido, o que torna estes sistemas pouco flexíveis quanto à evolução dos dados guardados [1]. Têm também limitações a nível de escalabilidade quando comparadas com sistemas de gestão de base de dados *NoSQL* descritos na secção seguinte.

3.2.2 Sistemas de Gestão de Bases de Dados *NoSQL*

NoSQL, também referido de *Not Only SQL* é um termo vagamente definido e refere-se a um grupo recente sistemas de gestão de base de dados focados em escalabilidade e desempenho.

Apesar de ser um grupo bastante abrangente, podem ser identificadas algumas características comuns à generalidade dos sistemas *NoSQL* [5]:

- são capazes de escalar horizontalmente ao longo de vários nós;
- são capazes de replicar e particionar dados ao longo de vários nós;
- são capazes de adicionar novos atributos dinamicamente.
- não utilizam SQL para consulta e manipulação da base de dados;
- não implementam transações ACID.

Por estas razões, são tipicamente associadas a ambientes altamente distribuídos com um ênfase grande em escalabilidade, disponibilidade e de evolução rápida.

De uma forma geral, este tipo de bases de dados opta por um modelo de dados mais simples e relaxado que o modelo relacional, o que faz com que se adaptem melhor a sistemas dinâmicos. Podem assumir também uma arquitetura mais flexível que um RDBMS tradicional, escalando facilmente a vários nós. Isto faz com que consigam um melhor desempenho e garantias de disponibilidade, assim como facilitar a replicação de dados.

Estas vantagens são muitas vezes conseguidas comprometendo algumas propriedades das transações ACID, sendo que os compromissos feitos variam muito entre os vários *NoSQL* DBMS [5].

3.2.3 Sistema Escolhido

A escolha da base de dados é de grande importância neste trabalho. Pelo já descrito, esta escolha não é fácil por poderem ser detetadas vantagens e desvantagens em qualquer dos sistemas disponíveis.

Um dos aspetos que não pode ser menosprezado é a familiaridade prévia com as ferramentas existentes. Por esta, e por razões descritas à frente, foi decidido que seria usada uma base de dados relacional para este sistema de faturação.

As garantias de consistência e integridade que estes sistemas fornecem são extremamente relevantes para o armazenamento de documentos com importância legal, como é o caso das faturas. Além disso, encontram-se num estado de maturidade mais avançado que os sistemas *NoSQL*, estando em constante desenvolvimento desde os anos 80. Outro ponto relevante é o tamanho que a aplicação irá tomar. Devido à sua especificidade, é esperado que seja uma aplicação com relativamente pouco tráfego. Isto faz com que uma das grandes vantagens de sistemas *NoSQL*, a grande escalabilidade, seja menos relevante para o bom funcionamento do sistema.

Dentro dos sistemas de gestão de bases de dados relacionais, foi escolhido o *PostgreSQL* [26], um sistema desenvolvido pela *PostgreSQL Global Development Group*. Teve a seu primeiro lançamento em 1989 e continua em ativo desenvolvimento. É um projeto *open source* e está licenciado sob a *PostgreSQL License*, uma licença permissiva semelhante à família de licenças BSD.

É compatível com o *standard* SQL e, em particular, implementa transações com as propriedades ACID. Estas transações satisfazem os requisitos de durabilidade, consistência e concorrência identificados. É também possível configurar o DBMS de

forma a que haja replicação de dados em tempo real, satisfazendo os requisitos de preservação de dados referidos anteriormente.

Uma vantagem do *PostgreSQL* sobre outras DBMS é o suporte recente para colunas do tipo `JSON`, adicionado na versão `9.2` Este é um tipo de dados não *standard* e que permite guardar e fazer consultas sobre objetos codificados em JSON.

Este tipo permite então expressar objetos (ou listas de objetos), arbitrariamente complexas de forma elegante e sem a necessidade de recorrer a um esquema de tabelas complexo. Não é necessário, também, que estes objetos tenham o mesmo esquema, o que traz uma grande flexibilidade à base de dados. Esta é uma característica muito importante para este sistema de faturação uma vez que será utilizado como *testbed* a trabalhos futuros. Com este tipo de dados é possível guardar informação que não foi prevista inicialmente sem a necessidade de alterar o esquema da base de dados.

Este tipo de dados pode também ser validado com recurso a CHECK CONSTRAINTS e pode ser indexado. Em particular, é possível usar índices sobre expressões para, por exemplo, garantir a unicidade de um atributo contido no objeto JSON.

Este tipo tem, no entanto, as suas limitações. Consultas nestes objetos são, de uma forma geral, mais lentas e não suporta restrições de chave externa.

Na versão `9.4` do *PostgreSQL* foi introduzido o tipo `JSONB`, que é em tudo semelhante ao tipo `JSON`, mas é mais eficiente. O tipo `JSON` guarda uma cópia exata do objeto inserido, que tem de ser processado sempre que este é acedido. `JSONB` guarda a informação num formato binário. Isto torna a inserção um pouco mais lenta, mas torna o processamento mais rápido e suporta indexação, o que torna consultas que usem esse índice substancialmente mais rápidas.

Nos dispositivos móveis presentes em cada táxi, pela sua mobilidade e vulnerabilidade, os dados serão guardados apenas temporariamente. Estes são sincronizados com o *backend* sempre que possível, sendo este o último responsável por garantir a integridade e consistência de todos os documentos. Por esta razão, a base de dados presente nestes dispositivos não tem requisitos de robustez tão fortes como os exigidos para o *backend*.

Foi então escolhido o sistema de gestão de base de dados *SQLite* [32], um DBMS relacional *open source* e no domínio público. É um DBMS simples e otimizado para sistemas embebidos e, em particular, está presente em todos os sistemas *Android* (sistema operativo usado nos dispositivos móveis usados neste projeto), o que torna o seu muito conveniente. Pode também ser facilmente embebido numa aplicação e, assim, ser usado noutro tipo de sistemas.

3.3 Garantias de imutabilidade

Durante o desenho do esquema de uma base de dados relacional, é comum haver uma fase de normalização [8, 9], com o objetivo de minimizar redundância. Isto traz vantagens ao nível do espaço de armazenamento necessário e além disso ajuda a prevenir inconsistências resultantes de atualizações de informação.

Isto é conseguido decompondo os objetos em várias tabelas, sendo cada uma composta apenas de colunas com valores atómicos (sem valores complexos como listas, por exemplo) e definindo relações entre elas usando chaves externas. Deste modo, se dois objetos tiverem uma relação com um terceiro, este só é guardado uma vez e qualquer alteração neste é automaticamente refletida nos objetos que dele dependem.

O comportamento descrito é o desejado em muitas situações, mas não no caso de uma fatura, que é um documento imutável. Tomemos o exemplo da emissão de uma fatura da venda do produto A. De uma forma simplificada, seria guardada a seguinte informação:

FATURAS

id	...
...	
36	...
37	...

LINHAS

id	idFatura	idProduto
...		
64	37	1
65	37	2
66	36	2

PRODUTOS

id	nome	...
1	A	...
2	B	...
...		

Figura 3.1: Exemplo de tabelas não versionadas associadas a uma fatura.

Neste modelo simplificado existem três tabelas:

- a tabela FATURAS, onde está representada informação atómica da fatura, como o seu número e a série em que foi emitida, entre outros;
- a tabela PRODUTOS, que representa uma lista de todos os produtos disponíveis;
- a tabela LINHAS, que faz a ligação entre as duas anteriores, sendo que as colunas `idFatura` e `idProduto` representam relações com as tabelas FATURAS e PRODUTOS, respetivamente.

Com esta configuração, se pretendermos reconstruir a fatura com `id` 37, podemos:

1. procurar toda a informação contida na linha da tabela `faturas` cuja coluna `id` tenha o valor 37;
2. juntar todas as linhas da tabela LINHAS cuja coluna `idFatura` tenho o valor 37;
3. com as linhas obtidas no passo anterior, é possível recolher os `ids` de todos os produtos associados à fatura 37 e, fazendo uma consulta à tabela PRODUTOS, saber também outro tipo de informação, como o seu nome ou descrição.

Continuando com o exemplo da Figura 3.1, era possível determinar que na fatura 37 estão associados os produtos com nome `A` e `B`.

O problema com um desenho destes é que, se mais tarde o nome do produto A mudar, essa mudança também é refletida na fatura agora emitida, o que não deve acontecer. Na Figura 3.2 pode ver-se o estado da base de dados do exemplo anterior caso o nome do produto `A` tenha sido alterado para o produto `AA`.

FATURAS

id	...
...	
36	...
37	...

LINHAS

id	idFatura	idProduto
...		
64	37	1
65	37	2
66	36	2

PRODUTOS

id	nome	...
1	AA	...
2	B	...
...		

Figura 3.2: Exemplo de tabelas não versionadas associadas a uma fatura (depois de modificadas).

Nesta situação, se fosse repetida a mesma pesquisa do exemplo anterior chegaríamos à conclusão de que à fatura 37 estavam associados os produtos com nome `AA` e `B`, o que não era verdade quando esta foi emitida.

Uma forma de resolver este problema pode passar pela "desnormalização" da base de dados, retirando todas as relações entre a tabela das faturas e outras tabelas na base de dados. Todos os dados necessários à reconstrução do documento seriam guardados numa única tabela. Para isto podia-se tirar partido do tipo de dados `JSONB` do *PostgreSQL*, que iria permitir guardar toda a informação necessária de forma elegante e numa tabela com um esquema simples ao mesmo tempo que permitiria fazer consultas sobre os documentos guardados.

Esta abordagem tem, no entanto desvantagem de passar para a aplicação a responsabilidade de garantir que, por exemplo, todos os produtos associados a uma fatura

são válidos na altura de emissão. Para isto é preciso que tabelas auxiliares continuem a existir (como por exemplo a tabela PRODUTOS) para que sejam consultadas pela aplicação aquando da emissão de cada documento.

Um outro problema é que, como seria expectável, a "desnormalização" faz com que seja armazenada muita informação redundante. Por exemplo, todas as faturas que contenham o produto A têm de guardar toda a informação relativa ao produto, mesmo que este seja idêntico em todas. Num ambiente de faturação para táxis, não é de esperar que os objetos que contribuem para uma fatura sejam atualizados muitas vezes relativamente ao número de faturas emitidas, tornando difícil justificar o espaço que seria ocupado "desnecessariamente".

A dependência de um tipo não *standard* pode também ser uma desvantagem, uma vez que torna a transição para outro DBMS muito complicada, caso isto se venha a revelar necessário.

Outra estratégia possível passa por garantir, ao nível da aplicação, que todos os objetos necessários para recriar uma fatura são também imutáveis.

Nesta situação, todas as atualizações nestes objetos são intercetados e convertidas em inserções linhas. Deste modo, a linha original continua a existir, inalterada, e com a mesma chave primária, não modificando então os objetos que dela dependem. Ao mesmo tempo é criada uma nova linha com a informação atualizada e que pode passar a ser usada a partir desse momento.

Para conseguir isto são necessárias duas novas colunas em todas as tabela sujeitas a este versionamento:

`ativo` Coluna booleana que indica se a linha corresponde à versão mais recente do objeto.

`anterior` Coluna que referencia a linha anterior do mesmo objeto, permitindo assim conseguir percorrer todo o seu histórico.

Adaptando o exemplo anterior da alteração do nome de um produto, podemos ver na Figura 3.3 o resultado da alteração do nome de um produto de `A` para `AA`.

PRODUTOS				
id	nome	...	ativo	anterior
1	A	...	T	NULL
2	B	...	T	NULL
3	C	...	T	NULL

PRODUTOS				
id	nome	...	ativo	anterior
1	A	...	F	NULL
2	B	...	T	NULL
3	C	...	T	NULL
4	AA	...	T	1

Figura 3.3: Antes (esquerda) e depois (direita) da alteração do nome de um produto numa tabela versionada.

No entanto é preciso ter em atenção que esta "imutabilidade" não é sempre desejada. Nalgumas tabelas é preferível que as atualizações a tabelas relacionadas sejam automaticamente refletidas, como que em "cascata".

Vejamos o exemplo da relação entre a tabela PRODUTOS e IMPOSTOS. A cada produto está associado um imposto, que é representado pela coluna `idImposto` na tabela PRODUTOS, que corresponde a uma restrição de chave externa com a tabela IMPOSTOS, como representado na Figura 3.4.

IMPOSTOS				
id	%	...	ativo	anterior
1	6%	...	T	NULL
2	13%	...	T	NULL

PRODUTOS					
id	nome	idImposto	...	ativo	anterior
1	A	1	...	T	NULL
2	B	1	...	T	NULL
3	C	2	...	T	NULL

Figura 3.4: Relação entre as tabelas versionadas IMPOSTOS e PRODUTOS

No exemplo, os produtos A e B estão afetados de um imposto de 6% e o produto C está afetado de um imposto de 13%.

Suponhamos que o primeiro imposto é atualizado e passa a ter um valor de 7%. Esta atualização deve refletir-se também na lista de produtos de forma a que o produto ativo esteja relacionado com a versão mais recente do imposto.

Alteração num imposto deve então ser traduzida não só na inserção de uma nova linha na tabela IMPOSTOS mas também na atualização dos produtos que dele dependem, como representado na Figura 3.5.

IMPOSTOS

id	%	...	ativo	anterior
1	6%	...	F	NULL
2	13%	...	T	NULL
3	7%	...	T	1

PRODUTOS

id	nome	idImposto	...	ativo	anterior
1	A	1	...	F	NULL
2	B	1	...	F	NULL
3	C	2	...	T	NULL
4	A	3	...	T	1
5	B	3	...	T	2

Figura 3.5: Relação entre as tabelas versionadas IMPOSTOS e PRODUTOS (depois de modificadas)

Um exemplo completo de como este comportamento pode ser usado para manter toda a informação relativa às faturas emitidas pode ser encontrado no Apêndice C.

É importante também garantir que as linhas de tabelas versionadas não podem ser apagadas, devendo apenas ser marcadas como inativas. Isto é, no entanto, mitigado pelo facto de todas as linhas relevantes a uma fatura estarem (direta ou indiretamente) relacionadas com uma restrição de chave externa.

Esta estratégia é, no entanto, pouco eficiente se houver muitas atualizações nas tabelas versionadas, uma vez que é criada pelo menos uma nova linha por cada alteração.

3.4 Esquema

O esquema da base de dados definido no sistema de faturação desenvolvido é muito baseado na especificação do ficheiro SAF-T (PT) [23] uma vez que é neste ficheiro que estão definidos todos os atributos que devem estar associados aos documentos emitidos.

Este esquema pode ser dividido num conjunto de tabelas principais, onde é organizada toda a informação essencial aos documentos emitidos, e um conjunto de tabelas auxiliares, usadas maioritariamente para fins de validação.

É nestas tabelas auxiliares que são guardados os vários tipos e estados que um documento pode ter e os vários tipos de produtos, entre outros. O uso destas tabelas permite tirar partido das restrições e propriedades de consistência da base de dados, em particular restrições de chave externa, para ajudar a garantir a validade da informação contida nas tabelas principais.

São agora descritas, com mais detalhe, as tabelas principais.

Tabela Invoices

A tabela Invoices é a tabela principal de todo o sistema de faturação. Nela, é guardada toda a informação diretamente relacionada com os documentos emitidos. Esta inclui, entre outros, a data de emissão, a sua identificação e assinatura digital, assim como os valores monetários associados. Inclui também referências para os utilizadores responsáveis pela criação e modificação de cada documento, para a empresa pela qual foi emitido assim como para o cliente ao qual é dirigido o documento, relacionando-se assim com as tabelas descritas a seguir.

Possui também um atributo extra do tipo `JSONB` no qual podem ser armazenados metadados relevantes.

Tabela InvoiceLines

Tabela com a qual é estabelecida uma relação *many-to-many* entre as tabelas Invoices e Products. Nesta tabela são também guardados dados como as quantidades associadas a cada produto, assim como o seu valor.

Tabela Taxes

Tabela onde é guardada uma listagem de todas as tarifas do IVA em vigor. Esta tabela é essencial para que seja possível calcular os valores monetários associados a cada fatura.

Tabela Products

Nesta tabela é guardada a lista de produtos disponíveis. Cada um tem associados uma descrição, um valor fixo (se aplicável) e uma relação com o imposto de que está afetado.

Tabela Companies

É nesta tabela que é armazenada a informação de todas as empresas associadas ao sistema de faturação. Contém informação como o nome, contacto e morada e o NIF da empresa.

Tabela Customers

Semelhante à tabela Companies, mas onde é armazenada a informação de todos os clientes para os quais já foi emitida uma fatura. Contém informação como o NIF do cliente e, opcionalmente, nome e contacto.

Tem também um atributo extra do tipo `JSONB`, à semelhança da tabela Invoices, no qual pode ser armazenado qualquer tipo de informação que se torne relevante.

Tabela Users

Tabela utilizada para guardar informação de cada utilizador do sistema de faturação. Contém toda a informação necessária para a identificação e autenticação de cada utilizador, assim como uma relação com a empresa a que este está associado, se aplicável.

Todas estas tabelas, com a exceção das tabelas Invoices e InvoiceLines são versionadas de acordo com a estratégia descrita na Secção 3.3.

Um esquema da base de dados descrita, assim como um extrato do SQL necessário para a criar, pode ser visto no Apêndice B.

Capítulo 4

Emissão de Faturas

Neste capítulo são estudadas duas estratégias diferentes para a emissão de faturas num táxi.

Ambas as estratégias desenvolvidas baseiam-se na comunicação com um servidor central através de uma API disponibilizada sobre o protocolo HTTP. Esta API é usada para coordenar e sincronizar todas as partes do sistema e, em particular, é a interface disponibilizada para que os vários dispositivos móveis possam interagir com a base de dados, local onde são armazenados de forma permanente todos os dados.

O capítulo começa com uma parte introdutórias onde é descrita a arquitetura base pensada para o sistema de faturação, assim como pode ser feito o registo e autenticação de clientes remotos.

Continua com duas secções que constituem a parte fundamental do capítulo. Numa é estudado como um dispositivo móvel pode proceder para a emissão de uma fatura remotamente, descrevendo alguns dos problemas associados a esta estratégia. Noutra é analisada uma arquitetura onde cada dispositivo móvel é responsável pela emissão local das suas próprias faturas, discutindo as dificuldades e limitações deste método.

O capítulo acaba com uma breve descrição do como se deve proceder em caso de inoperabilidade do sistema.

4.1 Arquitetura

Para o sistema de faturação desenvolvido foi escolhida uma arquitetura relativamente simples, a qual vai ser usada como base para as secções seguintes. Esta pode ser dividida em três camadas distintas, como pode ser visto esquematizado na Figura 4.1:

- camada de dados;
- camada lógica;
- camada móvel.

A camada de dados é responsável pelo armazenamento permanente de todos os dados e é constituída por um sistema de gestão de base de dados discutido na Secção 3.2.3.

A camada lógica pode ser vista como uma camada intermédia que faz a ponte entre as camadas de dados e móvel. É nesta camada que é assegurada grande parte da lógica de todo o sistema, resolvendo problemas como autenticação e filtragem de dados. É constituída por um servidor *web* conectado diretamente ao DBMS e expondo uma API acessível remotamente. Pode ser também utilizada para outros fins como servir uma página *web* para tarefas de gestão e manutenção de todo o sistema.

A camada móvel é a camada que faz a interface entre o utilizador final (o taxista) e as outras partes do sistema. É constituída pelo dispositivo móvel presente no táxi (terminal do taxista) e é responsável pela impressão das faturas. Comunica também com o servidor *web* toda a informação necessária para a emissão, impressão e armazenamento de cada fatura.

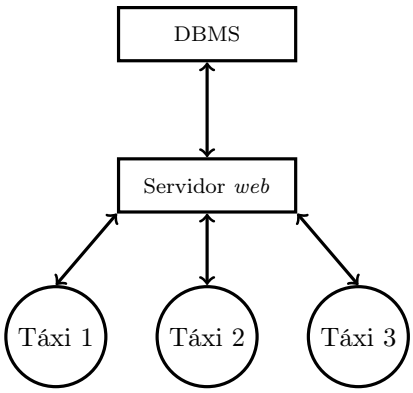

Figura 4.1: Esquema de uma arquitetura simples de um sistema de faturação.

O sistema de faturação tem também de estar preparado para conseguir resistir a falhas. Por esta razão, a arquitetura considerada, pela sua simplicidade, deve ser afinada antes de entrar em produção.

Apesar de ainda não ter sido feito muito trabalho neste sentido, consideramos que a ideia de transformar as camadas lógica e de dados em *clusters* [18, Capítulo 25] deve ser explorada. Em particular, uma configuração que envolva um *failover cluster* pode ser usada para garantir não só a essencial replicação de dados referida no Secção 3.1 mas também para minimizar o *downtime* associado à falha de algum dos nós do sistema.

4.2 Autenticação e Autorização

A API disponibilizada pelo sistema de faturação, para que possa ser acedida dispositivos presentes em cada táxi, tem, consequentemente, de estar também visível na *Internet*. Além disso, cada documento emitido tem, obrigatoriamente, de ter associado a utilizador. É necessário, portanto, que sejam previstos mecanismos de autenticação e autorização.

O sistema foi então pensado para suportar três tipos de utilizadores:

DISPOSITIVO Utilizador que representa um dispositivo presente num táxi. É usado apenas para a emissão de faturas num táxi, assim como para as anular sempre que seja necessário.

EMPRESA Utilizador que representa uma empresa. A este utilizador estão associadas informações relativas à empresa que representa, assim como algumas tarefas de administração, como a gestão dos dispositivos que lhe estão associados. Poderá também integrar faturas que, por alguma razão, tenham sido emitidas manualmente.

ADMINISTRADOR Utilizador genérico, de uso interno, com o qual é possível executar tarefas de gestão e administração.

A cada utilizador é associado, a tempo de criação, uma `API Key`. Esta é uma chave aleatória, única e que pode ser utilizada para identificar um utilizador. Assumindo que esta chave se mantém secreta, pode também ser utilizada para autenticação e é o método de autenticação usado para todos os pedidos descritos nas Secções 4.3 e 4.4.

Para o caso específico de ADMINISTRADORES e EMPRESAS, é também conveniente que se possam autenticar de uma forma mais "amigável". Para estes casos, cada utilizador tem também associado um `username` e uma `password`, os quais podem ser usados para fazer pedidos autenticados.

Caso algum dos métodos de autenticação, `API Key` ou `password`, sejam comprometidos, estes podem ser revogados e alterados a qualquer momento.

4.2.1 Configuração de um Dispositivo Móvel

Para que um dispositivo móvel possa começar a emitir faturas, é necessário que obtenha primeiro uma `API Key`. Esta deve ser gerada com recurso a um pedido `new device`, autenticado por um utilizador do tipo EMPRESA, onde deve ser indicado um nome que identifique o dispositivo.

Ao receber um destes pedido, é criado um novo utilizador do tipo DISPOSITIVO. Este é então associado à empresa usada para autenticar o pedido e é gerada uma `API Key`. É-lhe também associada uma série única na qual irão ser emitidos todos os documentos associados a este utilizador.

Como resposta é então enviada a `API Key`, que deve ser guardada e usada para autenticar todos os pedidos seguintes.

4.2.2 Geração de Chaves

As chaves usadas para autenticação devem ser imprevisíveis. Devem ser geradas aleatoriamente e não devem conter informação que permita identificar o utilizador a que está associado, como por exemplo a data de geração ou o nome do utilizador a que corresponde. Estas chaves devem também poder ser usadas em protocolos baseados em texto (como é o caso do HTTP).

Para conseguir uma chave com estas características, começa-se por gerar uma sequencia aleatória de bits. É então calculado o valor de *hash*, com uma função de *hash* criptograficamente segura como, por exemplo, o SHA-256. Este valor de *hash* pode então ser codificado em Base64, num processo semelhante ao descrito na Secção 2.2.2.

A codificação em `Base64` transforma a sequencia de bits num conjunto de caracteres alfanuméricos, `+` e `/`. Estes últimos caracteres especiais podem ser substituídos por

um par de caracteres alfanuméricos escolhidos aleatoriamente de forma a que a chave gerada possa ser facilmente incluída num *url*, por exemplo.

O resultado deste processo é uma sequencia de 38 caracteres alfanuméricos suficientemente imprevisível de forma a poder ser usada como chave de autenticação.

4.3 Emissão de Faturas Remotamente

A comunicação com os táxis, por estarem num ambiente altamente móvel, pode ser muito instável. Está sujeita a perdas de comunicação momentâneas e atrasos o que, por consequência, pode levar ao reenvio desnecessário de pedidos ou outros comportamentos inesperados. Estes fatores são agravados se tivermos em conta que uma fatura, depois de emitida, não pode ser apagada (embora possa ser anulada).

Em condições ideais, é possível emitir uma fatura através de um único pedido HTTP, utilizando a API desenvolvida. Quando um cliente remoto (táxi) pretender emitir uma fatura, pode enviar um pedido ao servidor com os dados necessários para a emissão desta. Desses dados farão parte informação como custo associado ao serviço efetuado e o NIF do cliente. O servidor emite então a fatura pedida e responde com a informação completa necessária para a impressão desta, em particular a chave resultante do processo de assinatura descrito na Secção 2.2.2

Uma consequência desta simplicidade é a não resistência a pedidos duplicados. Perdas momentâneas na comunicação podem levar a que um pedido ou resposta não cheguem ao destino. Para que isto possa ser ultrapassado, é necessário que o cliente retransmita um pedido até conseguir obter uma resposta ou até exceder um número máximo de tentativas. Isto pode levar à chegada de pedidos duplicados, o que levaria a que o servidor emitisse faturas em duplicado.

A procura de resolver estas fragilidades levou-nos a fazer diversas iterações, tendo obtido o protocolo apresentado na Secção 4.3.1, o qual incorpora mecanismos para detetar e ultrapassar as situações de erro descritas na Secção 4.3.2 sem problemas de maior.

4.3.1 Protocolo

Para a emissão de faturas remotamente foi desenhado um protocolo com o objetivo de tentar minimizar o número de faturas emitidas erradamente. Este tem como requisitos ser tolerante a perdas, atrasos e quebras na comunicação, garantindo que a base de dados onde são armazenadas as faturas emitidas se mantém num estado consistente.

A emissão de uma fatura foi dividida em três fases, sendo este protocolo constituído por três tipos de pedidos/respostas:

`start` Pedido inicial. Representa a intenção de emitir uma fatura. Tem obrigatoriamente de vir acompanhado da série em que se pretende emitir a fatura (o tipo de documento pode ser inferido). A resposta a este pedido incluiu um identificador único do documento a ser emitido, que deve ser usado nos pedidos seguintes.

`issue` Pedido usado para transmitir toda a informação necessária à emissão da fatura, como por exemplo os produtos que compõe o serviço prestado e informação sobre o cliente. Como resposta é enviado o documento completo, com todos os campos necessários à sua impressão, em particular a chave de quatro caracteres resultantes da assinatura.

`finish` Pedido usado para confirmar a receção do documento completo.

`void` Pedido que pode ser enviado pelo cliente a qualquer altura para cancelar a emissão de uma fatura. Este é usado pelo servidor para apagar ou anular a fatura correspondente, dependendo do seu estado.

A acrescentar às respostas descritas acima, a cada pedido estão também associadas respostas de erro para os casos em que existe problemas de validação ou violações do protocolo.

Já um fatura pode então estar num dos três estados:

STARTED Estado inicial de cada fatura e é resultado de um pedido `start` válido.

ISSUED Representa uma fatura já assinada e emitida e é resultado de um pedido `issue` válido. A partir deste estado uma fatura já não pode ser apagada, devendo ser apenas anulada caso seja necessário.

FINISHED Representa uma fatura emitida e confirmada e é resultado de um pedido `finish` válido. Uma fatura anulada também tem este estado.

Do ponto de vista de um cliente remoto (dispositivo no táxi), este pode estar num dos seguintes estados:

DEFAULT Estado por omissão. Encontra-se neste estado sempre que não estiver no processo de emissão de uma fatura. Deste estado pode emitir um pedido `start`.

WAITING START Um cliente vai para este estado sempre que logo que emita um pedido `start` e fica nele até receber uma resposta `start ok` ou cancele o processo de emissão.

STARTED Um cliente vai para este estado quando recebe uma resposta `start ok`. A partir deste pode enviar um pedido `issue`.

WAITING ISSUE Um cliente encontra-se neste estado sempre que estiver à espera de uma resposta a um pedido `issue` e fica nele até receber uma resposta `start ok` ou cancele o processo de emissão.

ISSUED Um cliente vai para este estado quando recebe uma resposta `issue ok`, devendo agora enviar um pedido `finish` a confirmar esta receção.

WAITING FINISH Um cliente encontra-se neste estado sempre que estiver à espera de uma resposta a um pedido `finish`, passando para o estado DEFAULT quando isto acontecer.

Um esquema de toda a comunicação pode ser visto na Figura 4.2.

Um esquema dos estados e as transições entre eles, numa situação normal, está representado no grafo de transições da Figura 4.3.

É de notar que o servidor, por ser um servidor *web*, não guarda qualquer estado entre os vários pedidos do cliente. Este estado é guardado em cada fatura, como um atributo desta.

Se ocorrer um erro, deve ser dada oportunidade ao utilizador de o corrigir ou, de alguma forma decidir como este deve ser resolvido. Isto pode ser feito, por exemplo quando ocorre um problema de validação num pedido `issue`, onde deve ser dada a oportunidade de corrigir os dados inseridos.

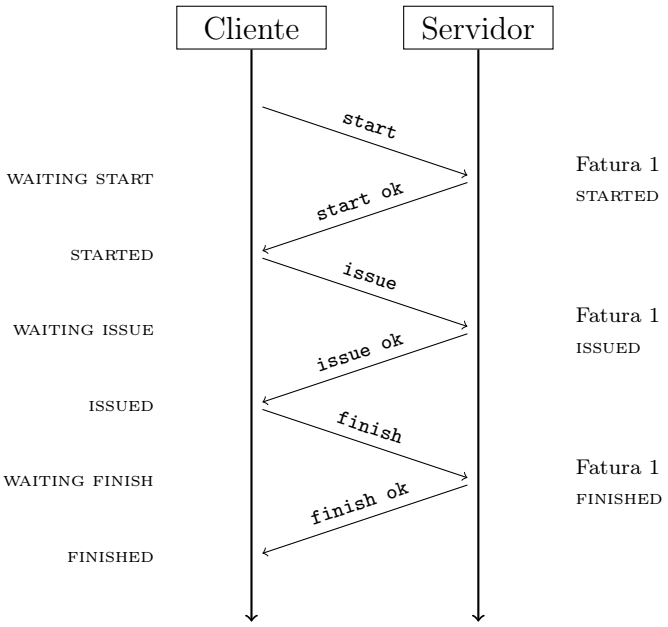

Figura 4.2: Visão global do protocolo para emissão de faturas num táxi.

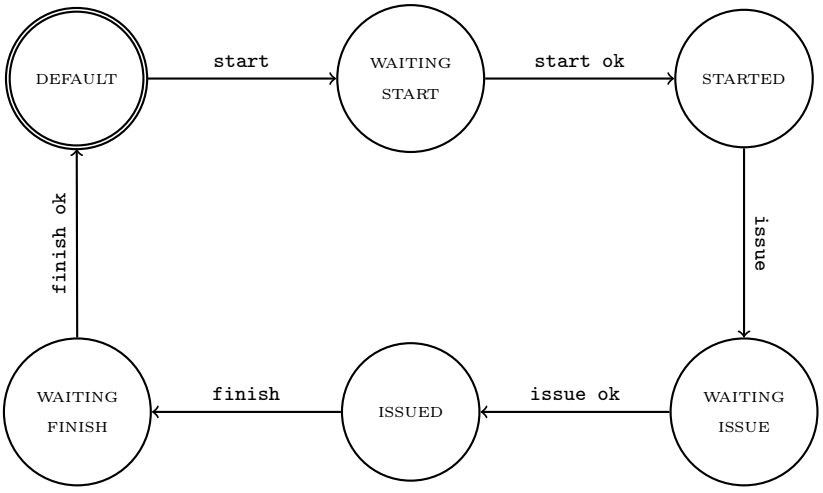

Figura 4.3: Possíveis estados de um cliente

4.3.2 Situações de Erro

A comunicação com o servidor deve ser feita usando a API desenvolvida. Como esta comunicação é feita com recurso ao protocolo HTTP, há garantias sobre a integridade da informação transmitida em cada pedido. No entanto, não resolve todos os problemas de instabilidade antes descritos, pois está ainda sujeita a perdas momentâneas da conectividade ou variações na latência da comunicação.

Para ultrapassar perdas de ligação momentâneas, o cliente deve repetir, com um intervalo razoável às condições da rede, o último pedido a que não obteve resposta. Este reenvio pode, no entanto, dar origem à repetição ou chegada desordenada de pedidos, devendo o servidor estar preparado para estes casos. Outra possível consequência é o cliente receber mensagens que não correspondem ao esperado no seu estado atual, devendo estas ser ignoradas.

Pedidos duplicados

Como descrito acima, numa situação em que o atraso na comunicação é anormalmente grande ou quando a comunicação é perdida momentaneamente, impedindo o cliente de receber a resposta esperada do servidor, podem ser enviadas repetições do mesmo pedido. Isto obriga a que todo o sistema esteja preparado para lidar com pedidos/-respostas duplicados.

Do lado do servidor, os pacotes duplicados podem ser detetados comparando com o estado do documento correspondente. Se a fatura associada ao pedido recebido já se encontrar no estado em que deveria ficar depois de processado o pedido, pode concluir-se que se trata de um pacote duplicado. Veja-se o exemplo seguinte da Figura 4.4:

1. É recebido um pedido `start`, do Cliente A, para emitir uma fatura;
2. O servidor cria a fatura correspondente e responde com o identificador que lhe foi atribuído, neste exemplo, 1. Esta resposta é no entanto perdida;
3. Passado um tempo razoável sem resposta, o Cliente A fecha a ligação que tinha com o servidor e repete o pedido `start`;
4. Ao receber este pedido, o servidor verifica que a última fatura associada ao Cliente A que já se encontra no estado STARTED. Pode então concluir que se trata de um pedido duplicado e reenviar a informação do documento já existente.

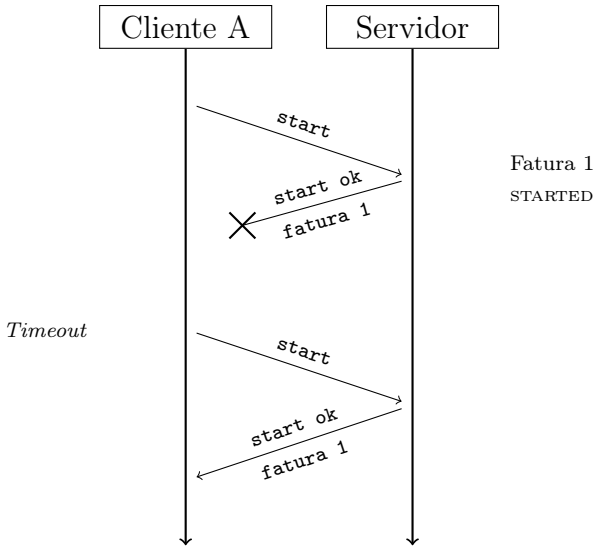

Figura 4.4: Envio de um pedido `start` duplicado.

Do ponto de vista do cliente, as respostas duplicadas podem ser ignoradas com segurança.

O procedimento é idêntico para os pedidos `issue` e `finish`.

Pedidos concorrentes

Pedidos concorrentes de vários clientes são suportados com recurso aos mecanismos de autenticação e atribuição de série descritos no Secção 4.2.

É no entanto necessário prever a situação em que dois pedidos do mesmo utilizador são recebidos simultaneamente pelo servidor, o que pode acontecer com as variações no atraso de cada ligação, como exemplificado na Figura 4.5.

Neste exemplo, são recebidos dois pedidos `start` do mesmo cliente simultaneamente. Para cada um deles, o servidor precisa de fazer uma consulta à base de dados para verificar a última fatura associada a esse mesmo cliente. A decisão de criar uma nova fatura ou reenviar uma já criada depende deste resultado. Se esta consulta for feita ao mesmo tempo, em transações isoladas, podem ser criadas dois novos documentos para o que é, essencialmente, um pedido para emissão de uma única fatura.

É então importante garantir que todos pedidos que possam alterar faturas da mesma série sejam processados de forma sequencial e isolada. Este comportamento pode ser conseguido com recurso aos mecanismos de *locking* disponibilizados pelo DBMS. Pode-

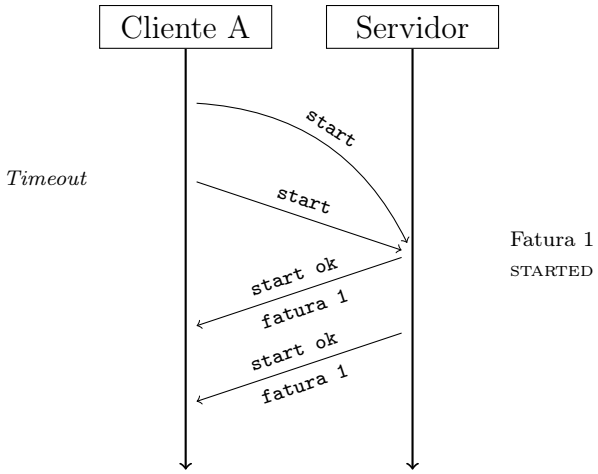

Figura 4.5: Chegada simultânea de dois pedidos `start`

se definir então, ao nível da aplicação, que todas as ações que realizem operações não atómicas sobre as faturas de uma série precisem de adquirir o *lock* correspondente. O DBMS garante que apenas uma transação pode obter este *lock* de cada vez, fazendo esperar transações simultâneas que também o tentem obter.

Pedidos fora de ordem

A chegada de pedidos fora de ordem deve ser vista como uma violação do protocolo. Estas devem ser detetadas para garantir a consistência de todos os documentos guardados.

À semelhança dos pedidos duplicados, isto pode ser conseguido verificando se o pedido recebido e o estado da fatura correspondente são o esperado. Caso a fatura associada ao pedido recebido já esteja num estado superior ao esperado, pode concluir-se que se trata de um pedido fora de ordem. Nestas situações, o servidor deve responder com a mensagem de error correspondente.

No exemplo da Figura 4.6, o cliente, por algum motivo, "desaparece" e perde o estado em que estava antes de conseguir finalizar a fatura que estava a emitir. Ao tentar emitir a uma nova fatura recebe um `start error`. Esta mensagem de erro pode ser usada pelo cliente (dispositivo no táxi) para que seja pedido ao utilizador (taxista) para tomar alguma ação, como cancelar ou finalizar a fatura anterior conforme esta tenha, ou não, sido impressa.

Esta situação podia ser evitada se não fosse requerida uma confirmação de receção

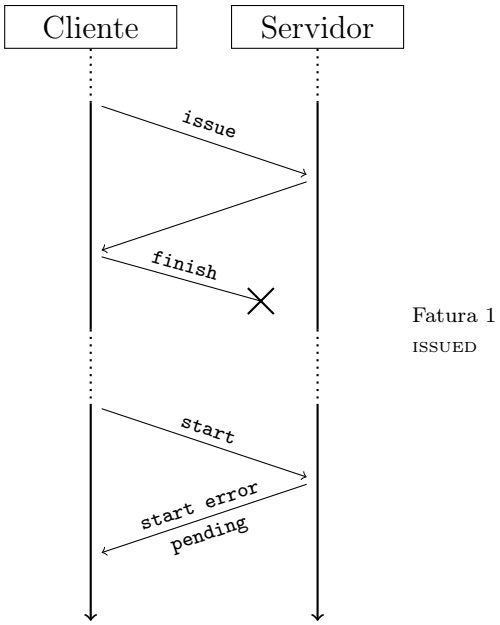

Figura 4.6: Chegada fora de ordem de um pedido `start`

por parte do cliente. Isto simplificava o protocolo, mas perdia-se a possibilidade de conseguir detetar quando uma resposta a um pedido `issue` foi perdida. Com recurso a uma confirmação por parte do cliente, apesar de continuar a não haver garantias de entrega de informação e introduzir dois novos pontos de falha, é possível detetar quando houve perda de informação.

Esta situação não deixa, no entanto, de ser frágil uma vez que em determinadas situações acaba por ser necessária a intervenção humana.

Um dos maiores problemas relacionados com a emissão de faturas desta forma é, no entanto, a necessidade de conseguir comunicar com um servidor remoto, o que é impossível de garantir dada a mobilidade associada aos táxis. Quando esta comunicação não é possível, é necessário proceder à emissão de uma fatura manualmente e posterior integração no sistema, como descrito na Secção 4.5.

Por esta razão, é estudado na secção seguinte como se pode proceder para a emissão de faturas localmente, ou seja, no próprio dispositivo presente em cada táxi.

4.4 Emissão de Faturas Localmente

Para o bom funcionamento do sistema considerado até ao momento deve ter-se uma ligação razoavelmente estável com um servidor central. Isto faz com que a emissão de faturas não seja possível em locais onde a rede móvel é fraca, assim como introduz um ponto de falha único (o servidor remoto), sem o qual é impossível a emissão de faturas.

Para tentar solucionar este problema, pensou-se numa forma de fazer com que os dispositivos móveis presentes nos táxis consigam emitir as suas próprias faturas, sem necessidade de comunicação imediata com o exterior. Para fazer isto, o dispositivo deve ser capaz de:

- preencher uma fatura com informação válida e consistente com a presente na base de dados central;
- numerar uma fatura;
- assinar uma fatura;
- armazenar temporariamente as faturas emitidas;
- integrar na base de dados central as faturas emitidas.

Dados de uma fatura

Para conseguir emitir uma fatura é necessário que o dispositivo tenha armazenada toda informação que possa constituir uma fatura. Esta informação inclui os produtos/-serviços disponíveis, impostos que lhe estão associados, informação sobre a empresa a que o dispositivo está associado, entre outros. Devem também ser sincronizados identificadores únicos para cada "objeto" de forma a poderem ser referenciados de forma não ambígua, tanto na base de dados central, como na presente no dispositivo móvel.

Em suma, o terminal do taxista deve ter armazenada, ou ser capaz de calcular, toda a informação que precise ser impressa no documento final.

Como foi referido na Secção 3.3, não é esperado que esta informação seja alterada frequentemente, mas deve ser sincronizada com a base de dados central sempre que

necessário. É preciso ter especial atenção às mudanças de dia, uma vez que será nessas alturas que que mudanças de impostos, por exemplo, entram em vigor.

Deve-se então forçar que haja uma sincronização total antes de ser emitida a primeira fatura do dia.

Numerar uma fatura

O mecanismo de numeração de uma fatura obriga a que estas sejam numeradas sequencialmente, dentro da mesma série e de acordo com a hora de emissão. Isto pode ser conseguido atribuindo uma série única a cada dispositivo e garantindo que este apenas emite faturas dentro da série que lhe foi atribuída. Esta atribuição de série deve seguir o descrito no Secção 4.2.1.

O dispositivo ficará então responsável por guardar não só a informação descrita anteriormente e a `API Key` usada para autenticação, mas também:

- a série em que deve emitir as faturas;
- o número e assinatura da última fatura emitida na série, já que é informação necessária para numerar e assinar uma próxima fatura.

Do lado do servidor é possível validar que todas as faturas foram de facto emitidas na série correta comparando a `API Key` usada e a série que lhe foi atribuída.

Assinar uma fatura

Para assinar uma fatura é necessário que o dispositivo tenha uma cópia da chave privada que foi comunicada à AT aquando do processo de certificação.

Sendo esta chave, de uma forma geral, fixa, poderia ser instalada juntamente com a aplicação, no entanto, pode ser necessário proceder à sua alteração. Isto pode acontecer caso esta chave seja comprometida ou como uma simples medida de precaução. Nestes casos, deve ser gerado um novo par de chaves que, além de ser comunicado à AT, precisa também de ser transmitido para todos os dispositivos.

Cada par de chaves tem uma versão associada (atribuída aquando da comunicação desta à AT) que pode ser usada no processo de sincronização descrito anteriormente para verificar se é necessário atualizar a chave privada. Caso isto aconteça, a chave

privada deve ser transmitida para o dispositivo, substituindo a que já estava guardada, se existir.

Para verificar que a chave presente no dispositivo é a chave correta, e que foi transmitida sem erros, pode recorrer-se a funções de *hash* e "impressões digitais" referida na Secção 2.2.2. Comparando o valor de *hash* da chave presente no dispositivo o da chave presente no servidor, pode detetar-se se estas são diferentes.

Informação sobre a chave privada é particularmente sensível e, quando esta seja transmitida na rede, é preciso especial cuidado em garantir que a comunicação é feita sob canal seguro.

Armazenamento temporário de faturas

As faturas emitidas têm de ser armazenadas no próprio dispositivo até serem integradas na base de dados central. Estas devem estão sujeitas aos mesmos requisitos de consistência e imutabilidade descritos no Capítulo 3.

No entanto, como os documentos armazenados no dispositivo móvel têm um caráter temporário, podem ser guardados já codificados em JSON e preparados para serem sincronizados com a base de dados central, numa estratégia semelhante à descrita na Secção 3.3.

Estas faturas, com exceção da última emitida, podem ser descartadas assim que se houver a garantia de que foram integradas com sucesso na base de dados central.

Integração de faturas

A integração das faturas emitidas deve ser feita sempre que possível, idealmente assim que esta seja emitida.

Esta integração é feita enviando ao servidor uma lista de faturas ordenada cronologicamente pela data de emissão. O servidor, ao receber esta lista, valida cada fatura, verificando, em particular, a série, número e assinatura desta. Caso seja validada com sucesso, é persistida na base de dados central.

Finalmente, o servidor responde com uma lista indicando, para cada fatura recebida, se foi integrada com sucesso ou não.

Receber a mesma fatura mais do que uma vez não traz problemas uma vez que se

pode facilmente verificar se já existe uma fatura com o mesmo número. Isto significa que é tolerado o envio de pacotes duplicados. É, no entanto, preciso ter um cuidado especial quando se trata da anulação de uma fatura já emitida e integrada.

4.4.1 Limitações

Apesar das grandes vantagens associadas à emissão local de faturas, como conseguir contornar os problemas de conetividade e mitigar a existência de pontos de falhar únicos, não deixam de existir algumas limitações.

É necessária a sincronização com o servidor central pelo menos uma vez por dia, por forma a garantir que a informação armazenada nos dispositivos é suficientemente atual. Isto implica também que, se houver uma falha nalguma das camadas lógica ou de dados, esta deve ser resolvida no próprio dia, para que o sistema continue operacional.

Pode também ocorrer que, por defeito no terminal do taxista ou falha na rede móvel, não seja possível sincronizar as faturas emitidas. Isto não é, numa situação normal, um problema uma vez que o sistema deve tentar fazer esta sincronização sempre que possível, idealmente assim que cada documento é emitido. É no entanto necessário ter especial atenção quando estas falhas de comunicação são mais prolongadas, sendo necessário agir em conformidade para que não seja perdida nenhuma informação relativa aos documentos não sincronizados. Outro momento crítico, por razões semelhantes, é a reinstalação do sistema. Antes de se proceder a esta reinstalação é essencial garantir que todas as faturas emitidas foram integradas na base de dados central.

De uma forma geral, a emissão de faturas de forma distribuída é mais frágil e complexa que a emissão remota. Isto é, no entanto, compensado pelas grandes melhorias que traz em relação à disponibilidade de todo o serviço de faturação.

4.5 Inoperabilidade do Sistema

Em caso de inoperabilidade de todo o sistema de faturação, deve proceder-se à emissão manual das faturas, numa tipografia autorizada e, assim que este esteja disponível, à integração no sistema de faturação.

Para proceder a esta integração deve ser processado um novo documento que recolha todos os elementos do documento emitido manualmente. Documentos integrados desta

forma devem utilizar uma série específica e serem identificados e assinados com o processo normal.

É no entanto necessário preservar a identificação do documento original. Esta deve ser indicada aquando da comunicação do documento à AT, assim como a indicação de que resulta de uma inserção manual.

É necessário proceder de forma semelhante sempre que houver a necessidade de integrar um documento que, por alguma anomalia do sistema, tenha sido perdido e que não integre nenhuma cópia de segurança. Nestes casos deve ser criado documentos duplicados, também em série específica e com numeração sequencial própria, onde esteja contida a mesma informação do documento original.

Detalhes sobre como se deve proceder à integração de faturas nestes casos podem ser encontrados nos requisitos técnicos dos programas de faturação [11, Artigo 2].

Capítulo 5

Pagamentos Móveis

Neste capítulo são feitas algumas considerações relativas ao pagamento de um serviço de táxi, o qual se pretende que possa ser efetuado da uma forma ágil e cómoda, e como o uso crescente de *smartphones* pode ser aproveitado neste sentido.

O capítulo começa com uma secção em que é feito um enquadramento dos pagamentos móveis, com referências às características esperadas deste serviço e à arquitetura geral que o suporta.

É dedicada depois uma secção a pagamentos móveis em táxis, com referência a aplicações concretas já existentes.

5.1 Contextualização

Pagamento móvel é qualquer pagamento em que um dispositivo móvel é usado para autorizar e confirmar uma troca de um valor financeiro por bens ou serviços [2]. Para que isto seja possível, é necessário todo um ecossistema de pagamentos móveis. Este é constituído por clientes, comerciantes, provedores de pagamentos móveis (MPP) e várias instituições financeiras, responsáveis por aprovar e certificar a segurança das transações, assim como efetuar essas transações (transferências bancárias).

A utilização de um dispositivo móvel, como um *smartphone*, para realizar pequenos pagamentos tem-se tornado cada vez mais relevante. É uma área que tem recebido grande atenção por parte de empresas de sector de financeiro, assim como de empresas do mundo online e móvel (*e.g. PayPal, Google, Apple, Visa e PT*). Tem também

recebido especial atenção em países em desenvolvimento como forma de estender os serviços financeiros (*e.g. Mobile Money Africa*).

Este método de pagamento pode ser caracterizado pela sua conveniência quando comparado com métodos tradicionais. Além de utilizar dispositivos que grande parte dos consumidores já carregam consigo (*smartphones*), faz uso de tecnologia já amplamente distribuída (*e.g. QR Codes*). Num contexto de transportes, onde a movimentação rápida é muito importante, a conveniência de conseguir pagar uma viagem em segundos, apenas com o telemóvel, é difícil de ser superada.

5.1.1 Características de um Sistema de Pagamentos Móveis

Para que um serviço de pagamentos móveis se torne atrativo, tanto para o comerciante como para o cliente, deve obedecer a algumas caraterísticas [31]:

- deve ser de fácil utilização;
- deve ser capaz de efetuar transações entre vários clientes e de valores monetários variáveis;
- deve conseguir interagir com outros sistemas;
- deve merecer a confiança das entidades envolvidas tanto a nível de segurança como de privacidade;
- deve competir com outros métodos de pagamento a nível de custo, conveniência e velocidade.

5.1.2 Arquitetura de um Sistema de Pagamentos Móveis

A Figura 5.1 descreve a interação entre os vários intervenientes num pagamento móvel.

Este processo começa com o registo, tanto do cliente como do comerciante, na plataforma fornecida pelo MPP, onde são partilhados detalhes financeiros (como cartão de crédito e/ou conta bancária) e formas de autenticação.

No momento do pagamento, o comerciante obtém a identificação do cliente (*e.g.* o número de telefone) e inicia a transação, comunicando ao MPP a quantia associada, quem é o cliente e qualquer outro tipo de informação necessária (*e.g.* detalhes sobre

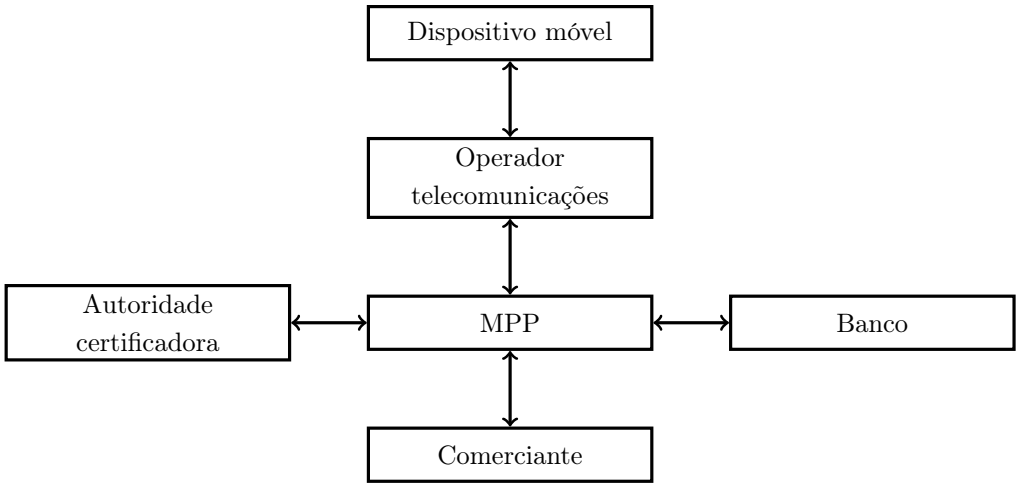

Figura 5.1: Arquitetura genérica de um sistema de pagamentos móveis. Adaptado de [31].

os produtos transacionados). Esta informação é também transmitida ao cliente, por exemplo com recurso a tecnologias como NFC ou *QR Codes*.

Finalmente, o cliente confirma e autoriza o pagamento e o MPP efetua a transferência bancária necessária, informando as duas partes de que o pagamento foi efetuado com sucesso.

Esta comunicação com o MPP é, normalmente, feita com recurso a uma *mobile wallet*, ou seja, *software* que reside no dispositivo móvel e onde são guardados os detalhes do cliente (como a identificação e cartão de crédito).

É importante assegurar que todas as comunicações que sejam feitas através da *Internet* se processem por canais seguros de forma a garantir a integridade e confidencialidade dos dados. Isto pode ser conseguido com tecnologias como certificados SSL.

5.2 Pagamentos Móveis nos Táxis

Neste trabalho foi iniciado um estudo sobre a possibilidade de introdução de um destes sistemas de pagamentos móveis nos táxis. Nesta secção será descrito como pode ser feita a integração com o sistema de pagamentos móveis *MEOWallet* [21], mas pode ser adaptada para sistemas semelhantes.

A aceitação de pagamentos através da *MEOWallet* começa com uma fase de configuração semelhante ao descrito na Secção 4.2.1, necessária para registar o terminal do

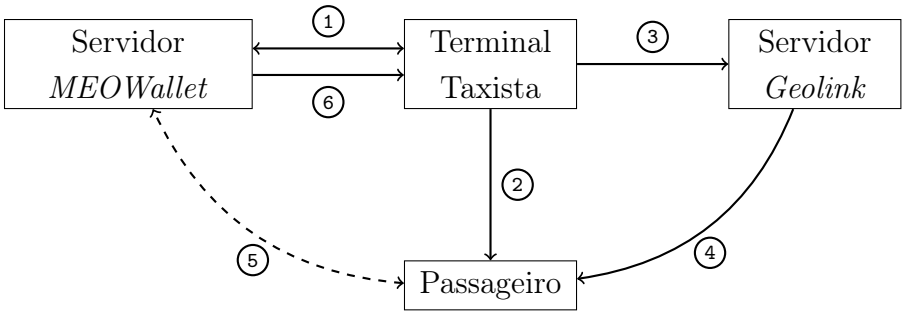

Figura 5.2: Esquema do processo de pagamento num táxi.

taxista. É nesta fase que é gerada uma `POS Key` que deve ser armazenada no terminal de taxista. Esta chave será usada pelo terminal para se autenticar perante o serviço *MEOWallet* em todos os momentos seguintes.

Terminada a fase de configuração, o terminal fica então apto comunicar com o serviço *MEOWallet* de forma a conseguir gerar *checkouts* das suas corridas de táxi. Este processo de pagamento está esquematizado na Figura 5.2.

Num primeiro passo (1), é necessário criar um *checkout*. Isto é feito enviando a informação necessária, em particular o custo do serviço de táxi, ao servidor *MEOWallet*. Este, por sua vez responde com a informação completa do *checkout* gerado, incluindo um *url* que o identifique.

Simultaneamente, o terminal de taxista deve registar-se num serviço de notificações usado para enviar atualizações sobre o estado do *checkout*. No caso do *MEOWallet*, isto é conseguido com recurso ao *message broker Sapo Broker* [29].

O *url* recebido no passo (1) deve agora ser transmitido para o *smartphone* do passageiro de forma a que este consiga efetuar o pagamento associado. Isto pode ser feito de várias formas.

O método mais simples e "universal" é com recurso a *QR Codes*. Neste caso o terminal do taxista gera e mostra um *QR Code* que codifique o *url* do *checkout* criado. Este pode ser lido pelo passageiro (2), com recurso ao seu *smartphone* e ser então redirecionado para o *site* ou aplicação da *MEOWallet* a fim de efetuar o pagamento (5).

Uma segunda opção de comunicar o *checkout* ao passageiro é enviando diretamente o *url* referido acima para o telemóvel do passageiro. O terminal de taxista pode enviar a informação do *checkout* necessária para um servidor da *Geolink* (3). Este, por sua vez, pode enviar a informação recebida para o cliente com recurso a, por exemplo,

uma SMS (4). Nesta situação é, no entanto, preciso que a *Geolink* conheça o cliente que efetuou o serviço. Isto pode ser conseguido se, por exemplo, o cliente pediu o serviço de táxi através de uma aplicação como o *MEO Táxi*. Se este for o caso, e se o telemóvel do cliente o suportar, é também possível recorrer a *push notifications* para comunicar o *url* ao cliente.

Quando o passageiro finalizar o pagamento, é enviada uma notificação ao terminal de taxista através do serviço subscrito no primeiro passo (6). Esta pode ser usada para verificar se o pagamento foi bem sucedido ou não, e agir em conformidade.

Capítulo 6

Conclusão

6.1 Trabalho Desenvolvido

Este trabalho centrou-se no desenvolvimento de um sistema de faturação eletrónica para táxis suficientemente flexível para poder ser usado como plataforma de testes para estudos futuros.

Foi desde logo feito um enquadramento legal e foram levantados os requisitos de um sistema de faturação. Identificaram-se também desafios específicos associados a uma solução de faturação num ambiente de mobilidade, em particular quais os efeitos da falta de garantias de uma comunicação estável entre os vários nós do sistema.

Apresentou-se uma arquitetura possível para o sistema de faturação, descrevendo como é que esta pode ser usada para a emissão de faturas num táxi de forma a mitigar e contornar os problemas de comunicação associados a uma rede móvel.

Este trabalho representa um contributo para o desenvolvimento do sistema de faturação referido. Foi feita a identificação de aspetos ainda não completamente tratado, o que pode contribuir para a realização de trabalhos futuros.

Iniciaram-se ainda testes de integração de sistemas de pagamentos móveis no serviço de táxi.

6.2 Trabalho Futuro

Numa próxima fase deve ser concluído o sistema de faturação eletrónica, de forma a conseguir a certificação deste. Neste sentido, é necessário ainda algum algum trabalho nas áreas de interface com o utilizador final e infraestrutura de suporte ao sistema. É também esperado que o sistema precise de ser "afinados" depois de entrar em produção.

Depois de certificado o sistema de faturação eletrónica desenvolvido, a interligação com o *sistema de despacho* (que reúne muita informação sobre o estado dos vários táxis, como por exemplo os serviços efetuados) abre grandes possibilidades ao permitir que seja reunida uma grande quantidade de informação interessante associada ao custo e pagamento de cada serviço. Irá ter-se acesso, por exemplo, ao custo real das viagens de táxi, o que pode desde logo ser usado para validar algoritmos de previsão destes custos.

A integração de pagamento móveis contribui para uma modernização da frota de táxi e possibilita um serviço de táxi mais cómodo e rápido para o passageiro. A eficácia deste método de pagamento bem como a adesão aos mesmos pode, desde logo, ser avaliada.

Os dados resultantes dos sistemas de faturação e de pagamentos móveis, quando agregados com informação recolhida do sistema de despacho, poderão também ser usados para estudos sobre uma utilização alternativa da rede de táxis, como por exemplo num cenário de *taxi-sharing*. Neste cenário, o sistema de faturação poderá ser utilizado como *testbed* para diferentes modelos de pagamento.

Seria também interessante um estudo mais aprofundado sobre a viabilidade das bases de dados *NoSQL* para um sistema de faturação. Estas representam uma área em rápida evolução e com grande potencial que sofreu bastantes desenvolvimentos desde o início do projeto.

Apêndice A

Acrónimos

ACID Atomicity, Consistency, Isolation, Durability

API Application Programming Interface

AT Autoridade Tributária e Aduaneira

BSD Berkeley Software Distribution

DBMS Database Management System

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

IVA Imposto de Valor Acrescentado

JSON Javascript Object Notation

MPP Mobile Payment Provider

NFC Near Field Communication

NIF Número de Identificação Fiscal

QR Code Quick Response Code

RAID Redundant Array of Inexpensive Disks

RDBMS Relational Database Management System

SAF-T Standard Audit File for Tax purposes

SAF-T (PT) Standard Audit File for Tax purposes — Versão portuguesa

SHA Secure Hash Algorithm

SMS Short Message Service

SOAP Simple Object Access Protocol

SQL Structured Query Language

SSL Secure Socket Layer

WSDL Web Service Definition Language

XML Extensible Markup Language

Apêndice B

Esquema da Base de Dados

B.1 Diagrama

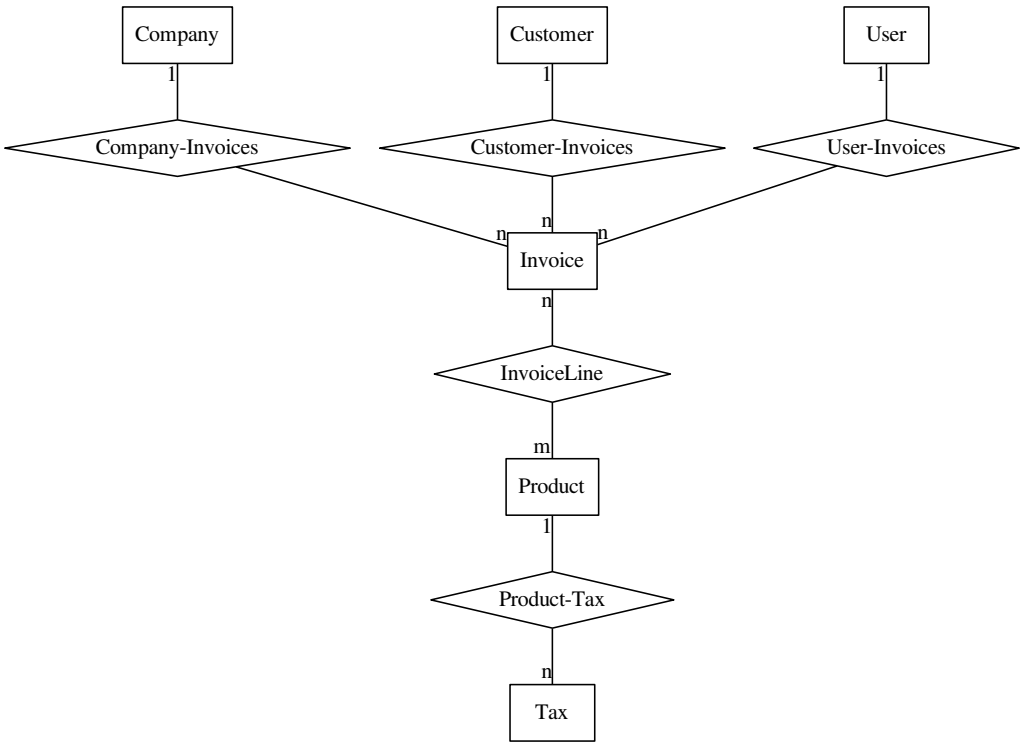

Figura B.1: Diagrama simplificado da base de dados.

B.2 Tabela Companies

```
CREATE TABLE companies (
    id serial PRIMARY KEY,
    name character varying(100) NOT NULL,
    nif character varying(30) NOT NULL,
    active boolean NOT NULL,
    previous_id integer,
    ...
);


CREATE UNIQUE INDEX companies_commercial_register_unique
    ON companies USING btree (commercial_register) WHERE active;

CREATE UNIQUE INDEX companies_nif_unique
    ON companies USING btree (nif) WHERE active;

ALTER TABLE ONLY companies
    ADD CONSTRAINT companies_previous_id_fkey FOREIGN KEY (previous_id)
    REFERENCES companies(id);
```

B.3 Tabela Customers

```
CREATE TABLE customers (
    id integer NOT NULL,
    nif character varying(30) NOT NULL,
    active boolean NOT NULL,
    previous_id integer,
    "default" boolean NOT NULL,
    extra jsonb,
    ...
);

CREATE UNIQUE INDEX customers_default
    ON customers USING btree ("default") WHERE "default";

CREATE UNIQUE INDEX customers_nif_unique
    ON customers USING btree (nif) WHERE active;

ALTER TABLE ONLY customers
    ADD CONSTRAINT customers_previous_id_fkey FOREIGN KEY (previous_id)
    REFERENCES customers(id);
```

B.4 Tabela Taxes

```
CREATE TABLE taxes (
    id serial PRIMARY KEY,
    type_id integer NOT NULL,
    description character varying(255) NOT NULL,
    percentage numeric(4,1),
    country_region_id integer NOT NULL,
    active boolean NOT NULL,
    previous_id integer,
    "default" boolean NOT NULL,
    ...
);

CREATE UNIQUE INDEX taxes_default
    ON taxes USING btree ("default") WHERE "default";

ALTER TABLE ONLY taxes
    ADD CONSTRAINT taxes_country_region_id_fkey FOREIGN KEY (country_region_id)
    REFERENCES countries(id);

ALTER TABLE ONLY taxes
    ADD CONSTRAINT taxes_previous_id_fkey FOREIGN KEY (previous_id)
    REFERENCES taxes(id);

ALTER TABLE ONLY taxes
    ADD CONSTRAINT taxes_type_id_fkey FOREIGN KEY (type_id)
    REFERENCES tax_types(id);
```

B.5 Tabela Products

```
CREATE TABLE products (
    id serial PRIMARY KEY,
    unit_price numeric(18,4),
    type_id integer NOT NULL,
    tax_id integer NOT NULL,
    active boolean NOT NULL,
    previous_id integer,
    ...
);

ALTER TABLE ONLY products
    ADD CONSTRAINT products_previous_id_fkey FOREIGN KEY (previous_id)
    REFERENCES products(id);

ALTER TABLE ONLY products
    ADD CONSTRAINT products_tax_id_fkey FOREIGN KEY (tax_id)
    REFERENCES taxes(id);

ALTER TABLE ONLY products
    ADD CONSTRAINT products_type_id_fkey FOREIGN KEY (type_id)
    REFERENCES product_types(id);
```

B.6 Tabela Users

```
CREATE TABLE users (
    id serial PRIMARY KEY,
    type character varying NOT NULL,
    username character varying(64) NOT NULL,
    password_hash character varying(128),
    api_key character varying(43),
    email character varying(60),
    series character varying(20),
    company_id integer,
    created_by_id integer,
    active boolean NOT NULL,
    previous_id integer,
    CONSTRAINT user_has_company CHECK (((company_id IS NOT NULL) OR (NOT (((type)::
    text = 'manager'::text) OR ((type)::text = 'device'::text))))),
    CONSTRAINT user_has_created_by CHECK (((created_by_id IS NOT NULL) OR ((type)::
    text <> 'device'::text))),
    CONSTRAINT user_has_series CHECK ((((series IS NOT NULL) OR ((series)::text <> ''
    ::text)) OR ((type)::text <> 'device'::text))),
    ...
);

CREATE UNIQUE INDEX users_api_key_unique
    ON users USING btree (api_key) WHERE active;

CREATE UNIQUE INDEX users_email_unique
    ON users USING btree (email) WHERE active;

CREATE UNIQUE INDEX users_series_unique
    ON users USING btree (series) WHERE active;

CREATE UNIQUE INDEX users_username_unique
    ON users USING btree (username) WHERE active;

ALTER TABLE ONLY users
    ADD CONSTRAINT users_company_id_fkey FOREIGN KEY (company_id)
    REFERENCES companies(id);

ALTER TABLE ONLY users
    ADD CONSTRAINT users_created_by_id_fkey FOREIGN KEY (created_by_id)
    REFERENCES users(id);

ALTER TABLE ONLY users
    ADD CONSTRAINT users_previous_id_fkey FOREIGN KEY (previous_id)
    REFERENCES users(id);
```

B.7 Tabela Invoices

```
CREATE TABLE invoices (
    id serial PRIMARY KEY,
    intern_code character varying(20) NOT NULL,
    series character varying(20) NOT NULL,
    number integer,
    signature character varying(172),
    key_version integer NOT NULL,
    system_entry_date timestamp with time zone,
    date_created timestamp with time zone,
    last_update timestamp with time zone,
    net_total numeric(18,4),
    tax_total numeric(18,4),
    gross_total numeric(18,4),
    type_id integer,
    company_id integer,
    customer_id integer,
    status_id integer,
    source_id integer,
    creator_id integer NOT NULL,
    last_update_user_id integer,
    extra jsonb,
    ...);
ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_series_number_key UNIQUE (series, number);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_company_id_fkey FOREIGN KEY (company_id)
    REFERENCES companies(id);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_creator_id_fkey FOREIGN KEY (creator_id)
    REFERENCES users(id);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_customer_id_fkey FOREIGN KEY (customer_id)
    REFERENCES customers(id);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_last_update_user_id_fkey
    FOREIGN KEY (last_update_user_id)
    REFERENCES users(id);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_source_id_fkey FOREIGN KEY (source_id)
    REFERENCES invoice_sources(id);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_status_id_fkey FOREIGN KEY (status_id)
    REFERENCES invoice_status(id);

ALTER TABLE ONLY invoices
    ADD CONSTRAINT invoices_type_id_fkey FOREIGN KEY (type_id)
    REFRENCES invoice_types(id);
```

B.8 Tabela InvoiceLines

```
CREATE TABLE invoice_lines (
    id serial PRIMARY KEY,
    quantity numeric(18,4) DEFAULT 1::numeric,
    unit_price numeric(18,4) NOT NULL,
    net_total numeric(18,4) NOT NULL,
    tax_total numeric(18,4) NOT NULL,
    gross_total numeric(18,4) NOT NULL,
    product_id integer,
    invoice_id integer
);

ALTER TABLE ONLY invoice_lines
    ADD CONSTRAINT invoice_lines_invoice_id_product_id_key
    UNIQUE (invoice_id, product_id);

ALTER TABLE ONLY invoice_lines
    ADD CONSTRAINT invoice_lines_invoice_id_fkey FOREIGN KEY (invoice_id)
    REFERENCES invoices(id);

ALTER TABLE ONLY invoice_lines
    ADD CONSTRAINT invoice_lines_product_id_fkey FOREIGN KEY (product_id)
    REFERENCES products(id);
```

Apêndice C

Imutabilidade

C.1 Um Exemplo Completo

Neste exemplo será usado um cenário base onde existem dois impostos, três produtos e duas faturas. A primeira fatura existente, com `id 36`, está relacionada com os produtos A e B. Já a segunda fatura, com `id 37`, está relacionada apenas com o produto B. Por sua vez, os produtos A e B estão relacionados com o imposto com `id 1` e ao produto C está associado o imposto com `id 2`. O estado descrito está esquematizado nas tabelas da Figura C.1.

FATURAS

id	...
...	
36	...
37	...

LINHAS

id	idFatura	idProduto
...		
63	36	2
64	37	1
65	37	2

IMPOSTOS

id	%	...	ativo	anterior
1	6%	...	T	NULL
2	13%	...	T	NULL

PRODUTOS

id	nome	idImposto	...	ativo	anterior
1	A	1	...	T	NULL
2	B	1	...	T	NULL
3	C	2	...	T	NULL

Figura C.1: Estado inicial da base de dados

Para este exemplo, iremos agora supor que o imposto com `id 1` sofre uma alteração na percentagem que lhe está associada de 6% para 7%. Esta alteração resulta, tal como descrito na Secção 3.3, na inserção de uma nova linha na tabela IMPOSTOS (`id 3`) que reflete esta alteração. Esta é também propagada pelas tabelas versionadas, resultando na inserção de duas novas linhas (`ids 4` e `5` da tabela PRODUTOS).

Num próximo passo deste exemplo é criada uma nova fatura (`id 38`), onde é incluído o produto A, sendo que desta vez é feita a relação com a linha com `id 4` da tabela PRODUTOS.

O estado da base de dados depois das alterações descritas pode ser visto na Figura C.2.

FATURAS

id	...
...	
36	...
37	...
38	...

LINHAS

id	idFatura	idProduto
...		
63	36	2
64	37	1
65	37	2
66	38	4

IMPOSTOS

id	%	...	ativo	anterior
1	6%	...	F	NULL
2	13%	...	T	NULL
3	7%	...	T	1

PRODUTOS

id	nome	idImposto	...	ativo	anterior
1	A	1	...	F	NULL
2	B	1	...	F	NULL
3	C	2	...	T	NULL
4	A	3	...	T	1
5	B	3	...	T	2

Figura C.2: Estado da base de dados depois de uma alteração do imposto `1`

As faturas existentes antes da alteração do imposto continuam relacionadas com as versões de produtos e impostos iniciais, não sendo então afetadas pelas alterações introduzidas. Faturas emitidas depois da modificação de imposto já irão refletir estas alterações. A fatura com `id 38` é um exemplo.

Referências

[1] J Andany, M Léonard e C Palisser. «Management Of Schema Evolution In Databases.» Em: *VLDB* (1991).

[2] Yoris a. Au e Robert J. Kauffman. «The economics of mobile payments: Understanding stakeholder issues for an emerging financial technology application». Em: *Electronic Commerce Research and Applications* 7.2 (jun. de 2008), pp. 141–164.

[3] Autoridade Tributária e Aduaneira, ed. *Manual de Integração de Software*. Mar. de 2015.

[4] Tim Bray et al. *Extensible Markup Language (XML) 1.1 (Second Edition)*. 2006. URL: `http://www.w3.org/TR/xml11/` (acedido em 24/09/2015).

[5] Rick Cattell. «Scalable SQL and NoSQL data stores». Em: *ACM SIGMOD Record* 39.4 (mai. de 2011), p. 12.

[6] Peter M. Chen et al. «RAID: high-performance, reliable secondary storage». Em: *ACM Computing Surveys* 26.2 (jun. de 1994), pp. 145–185.

[7] Erik Christensen et al. *Web Service Definition Language (WSDL)*. 2001. URL: `http://www.w3.org/TR/wsdl.html` (acedido em 24/09/2015).

[8] E. F. Codd. «A relational model of data for large shared data banks». Em: *Communications of the ACM* 13.6 (jun. de 1970), pp. 377–387.

[9] E. F. Codd. «Further Normalization of the Data Base Relational Model». Em: *IBM Research Report, San Jose, California* RJ909 (1971).

[10] *Decreto-Lei n.º 198/2012*. Diário da Replública nº 164/2012 – Série I. Ago. de 2012.

[11] *Despacho n.º 8632/2014*. Diário da República n.º 126/2014 – Série II. Jul. de 2014.

[12] Direção-geral dos Impostos, ed. *Modelo 24 – Declaração de Certificação de Programa de Facturação*. Jun. de 2010.

[13] Pedro M d'Orey e Michel Ferreira. «Can ride-sharing become attractive? A case study of taxi-sharing employing a simulation modelling approach». Em: *IET Intelligent Transport Systems* 9.2 (2014), pp. 210–220.

[14] D. Eastlake e P. Jones. *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174 (Informational). Set. de 2001.

[15] Ramez Elmasri e Shamkant B. Navathe. *Fundamentals of Database Systems (6th Edition)*. 6ª ed. Pearson, 2010, p. 1200.

[16] Theo Haerder e Andreas Reuter. «Principles of transaction-oriented database recovery». Em: *ACM Computing Surveys* 15.4 (dez. de 1983), pp. 287–317.

[17] ISO/IEC 9075:2011. *Information technology – Database languages – SQL*. Rel. téc. International Organization for Standardization, dez. de 2011.

[18] Kailash Jayaswal. *Administering Data Centers: Servers, Storage, and Voice Over IP*. 1ª ed. Wiley, 2005.

[19] S. Josefsson. *The Base16, Base32, and Base64 Data Encodings*. RFC 3548 (Informational). Jul. de 2003.

[20] Alfred J. Menezes, Paul C. van Oorschot e Scott A. Vanstone. *Handbook of Applied Cryptography*. Vol. 106. 1997, p. 780.

[21] *MEOWallet*. URL: `https://wallet.pt/` (acedido em 24/09/2015).

[22] *Portal das Finanças*. URL: `https://www.portaldasfinancas.gov.pt` (acedido em 24/09/2015).

[23] *Portaria n.º 274/2013*. Diário da República n.º 160/2013 – Série I. Ago. de 2013.

[24] *Portaria n.º 340/2013*. Diário da República n.º 227/2013 – Série I. Nov. de 2013.

[25] *Portaria n.º 426-A/2012*. Diário da República n.º 251/2012 – Série I. Dez. de 2012.

[26] *PostgreSQL*. URL: `http://www.postgresql.org/` (acedido em 24/09/2015).

[27] E. Rescorla. *HTTP Over TLS*. RFC 2818 (Informational). Mai. de 2000.

[28] R. L. Rivest, A. Shamir e L. Adleman. «A method for obtaining digital signatures and public-key cryptosystems». Em: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[29] *Sapo Broker*. URL: `http://oss.sapo.pt/sapo-broker/` (acedido em 24/09/2015).

[30] Bruce Schneier. «Applied Cryptography». Em: *Electrical Engineering* 1.32 (1996). Ed. por Array, pp. 429–455.

[31] Basudeo Singh e Jasmine K.S. «Comparative Study on Various Methods and Types of Mobile Payment System». Em: *2012 International Conference on Advances in Mobile Network, Communication and Its Applications* (2012), pp. 143–148.

[32] *SQLite*. URL: `https://www.sqlite.org/` (acedido em 24/09/2015).

[33] W3C. *W3C SOAP Specification*. 2013. URL: `http://www.w3.org/TR/soap11` (acedido em 24/09/2015).

[34] *Webservice para Comunicação das Faturas*. URL: `http://info.portaldasfinancas.gov.pt/NR/rdonlyres/02357996-29FC-4F11-9F1D-6EA2B9210D60/0/factemiws.wsdl` (acedido em 24/09/2015).