FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GUILHERME DE OLIVEIRA MACEDO

MOBILE PAYMENT POR APROXIMAÇÃO

MARÍLIA
2014

GUILHERME DE OLIVEIRA MACEDO

MOBILE PAYMENT POR APROXIMAÇÃO

Trabalho de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador
Profª. MSc. Giulianna Marega Marques

MARÍLIA
2014

MACEDO, Guilherme de Oliveira

Mobile Payment por aproximação / Guilherme de Oliveira Macedo; orientadora: Profª. MSc. Giulianna Marega Marques. Marília, SP: [s.n.], 2014.

75 folhas

Monografia (Bacharelado em Sistemas de Informação): Centro Universitário Eurípides de Marília.

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Guilherme de Oliveira Macedo

Mobile payment por aproximação

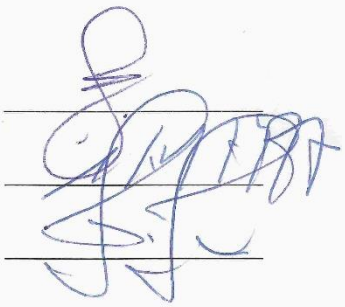
Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 10,0 (dez)

Orientador: Giulianna Marega Marques

1°. Examinador: Emerson Alberto Marconato

2°. Examinador: Geraldo Pereira Junior



Marília, 05 de dezembro de 2014.

Dedido esse trabalho à todas as pessoas, que de alguma forma contribuíram para minha formação profissional e acadêmica.

AGRADECIMENTOS

Agradeço primeiramente à Deus por ter me proporcionado a oportunidade de aprender e viver diversas experiências que contribuirão significativamente para meu futuro.

À minha família, que nos momentos de alegria, tristeza, dúvidas e incertezas estiveram ao meu lado dando-me o apoio necessário.

A Profª. MSc. Giulianna Marega Marques pelo apoio e orientação que nortearam esse trabalho e impactaram significativamente nos resultados aqui mencionados.

Aos demais professores que por meio de seus conhecimentos também contribuíram para os resultados aqui mencionados além de me preparem para os desafios que o mercado me proporcionará nos próximos anos.

Aos meus amigos, em especial o Flávio Mendes, Gustavo Zafra, Maicon Tonezi João Vitor P. Santos e Lucas Valencio pelo apoio e companheirismo durante os anos de graduação.

Por fim à todos os demais funcionários e parceiros do Univem, em especial o Giorgio Pereira, que nos apoiam diariamente para que tenhamos as melhores experiências dentro do ambiente universitário.

“A luz que me guia é mais forte do que as dificuldades que me cercam”
Autor desconhecido

RESUMO

O *mobile payment* é considerado a próxima fase da evolução entre a relação do consumidor com o comerciante ou varejista, devido as facilidades que trará ao consumidor. Compara-se esse método de pagamento como qualquer outro método tradicional, seja executada de forma presencial ou remota. O crescimento e viabilidade desse método deve-se exclusivamente a facilidade, agilidade e segurança proporcionadas ao usuário além da popularização dos dispositivos móveis, principalmente os smartphones. Também percebe-se um grande avanço das tecnologias que proporcionam o desenvolvimento desse método de pagamento, como o *Bluetooth* e *NFC* (Near Field Communication) que permitem a troca de dados entre dois ou mais dispositivos móveis, tecnologia ideal para serem aplicados nesse novo conceito de pagamento. Conforme mencionam os especialistas, os pagamentos móveis se tornarão uma das aplicações mais importantes dos próximos anos, facilitando os micros pagamentos no comércio eletrônico móvel obtendo-se uma alternativa ao uso do dinheiro (BERGONHA;HOFFMAAN, 2002; HERZBERG, 2003) Além do mais, com a popularização dessa tecnologia permite-se que as pessoas possam realizar pagamentos de forma simples e intuitiva evitando transtornos aos usuários além de fornecer dados para gerenciamento financeiro pessoal.

Palavras-Chave: aproximação, pagamentos, mobilidade, NFC, comércio, varejo, dispositivo móvel

ABSTRACT

The mobile payment is considered the next stage of evolution between the consumer's relationship with the merchant or retailer, because the facilities that will bring to the consumer. Compares this method of payment like any other traditional method, whether carried out in person or remotely. The growth and viability of this method is exclusively the ease, speed and security provided to the user in addition to the popularity of mobile devices, especially smartphones. Also, we can see major breakthrough technologies that provide the development of this payment method, such as Bluetooth and NFC (Near Field Communication) for exchanging data between two or more mobile devices, ideal technology to be applied in this new concept of payment. As mentioned experts, mobile payments will become one of the most important applications in the coming years, facilitating micro payments in the mobile e-commerce to give an alternative to the use of money (BERGONHA; HOFFMAAN, 2002; HERZBERG, 2003). Moreover, with the popularization of this technology allows people to make payments in a simple and intuitive way to avoid inconvenience to users and provides data to personal financial management.

Keywords: approximation, payments, mobile, NFC, trade, retail, mobile device

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BCEAO	Banco Central dos Estados da África Ocidental
CMN	Conselho Monetário Internacional
FEBRABAN	Federação Brasileira de Bancos
IPEA	Instituto de Pesquisa e Estatística Aplicada
NFC	Near Field Communication
NFCIP	Near Field Communication Interface and Protocol
ONU	Organização nas Nações Unidas
P2P	Person-to-Person
PIB	Produto Interno Bruto
POS	Point-of-Sales
RFID	Radio-Frequency Identification
ROI	Retorno sobre o investimento
SFN	Sistema Financeiro Nacional
SMS	Short Message Service
XML	Extensible Markup Language

SUMÁRIO

INTRODUÇÃO

A necessidade de se comunicar é tão antiga quanto o próprio homem, até hoje, poucas necessidades são tão fortes quanto essa de comunicação. A fala é um dos principais meios, mas nos últimos anos se tem utilizado o apoio da tecnologia por meio de aplicativos de mensagens instantâneas, como o *What's App*. Outra necessidade tão antiga quanto a de se comunicar é a troca de objetos de valor, o que favoreceu o surgimento do comércio. Assim os métodos de comunicação têm evoluído com o passar do tempo, o mesmo ocorre para ferramentas e métodos relacionados ao comércio, por exemplo, a invenção das moedas e bancos. Ao ponto de unir o comércio e as tecnologias, como mobilidade resulta-se em um novo conceito de relacionamento entre o comerciante e o consumidor.

Ao analisar o mercado atual, depara-se com grandes *players* como bancos, operadores de cartão de crédito, entre outros que possuem algumas características em comum que é a busca de um modelo rentável em um mercado tão saturado, sendo o mercado financeiro. Quando se fala somente de cartões de crédito, as empresas intermediadoras desse processo recebem de 1% à 3% do valor de cada transação (MICROSOFT e M-COM, 2013), o que não é bem visto por grande parte dos comerciantes, principalmente os de pequeno porte, e que tem sido alvo de ações judiciais e ações reguladoras. Qualquer outro modelo rentável, seja ele móvel ou não, deverá reduzir essas taxas ou arriscar-se a entrar em uma disputa com grandes players do mercado e os comerciantes.

No Brasil há movimentações do governo por meio do Banco Central para oficializar os micros pagamentos realizados por dispositivos móveis. Em Novembro de 2013 o Conselho Monetário Nacional (CMN), órgão superior do Sistema Financeiro Federal que tem como responsabilidade de formular a política da moeda e do crédito, regulamentou as resoluções nº 4.282 (BANCO CENTRAL DO BRASIL, 2013) e nº 4.283 (BANCO CENTRAL DO BRASIL, 2013) respectivamente, instituindo um marco regulatório inicial que disciplina a autorização e funcionamento dos arranjos e instituições financeiras em conforme com os preceitos estabelecidos na lei nº 12.865 de 09 de Outubro de 2.013. Uma das regras estabelecidas nas resoluções é a limitação das transações pré-pagas para que não ultrapassem R$ 1.500,00 e na qual o somatório dos aportes efetuados seja limitado a esse mesmo valor.

Porém a adoção desse método de pagamento tem alguns obstáculos como a grande

variedade de dispositivos móveis, tecnologias e soluções não permitindo uma padronização do *mobile payment.* Além disso, existem diversos tipos de mercado e em cada um existe uma solução que mais se adequa para a execução de serviços financeiros por meio de dispositivos móveis. Observa-se claramente isso, por meio da pesquisa realizada em 2.013 pela GFT do Brasil que entrevistou 900 pessoas em cinco países (Brasil, Alemanha, Reino Unido, Espanha e Estados Unidos), destacando o Brasil e Espanha onde mais de 60% dos entrevistados disseram que já utilizaram seus dispositivos móveis para realizarem algum pagamento, enquanto os entrevistados alemães mais cautelosos na adoção de novas tecnologias, apenas 26% disseram que já utilizaram seus dispositivos móveis para fins bancários (GFT DO BRASIL, 2014).

Para que esse novo método de pagamento se consolide no mercado é necessário convencer à todos, principalmente os comerciantes e varejistas que o ROI (Retorno sobre o investimento) é rápido e rentável. Outro ponto é apresentar os benefícios proporcionados aos negócios, como segurança, agilidade nas operações financeiras, etc.

Motivação e Justificativa

O *mobile payment* é considerado por alguns especialistas a próxima fase da evolução entre a relação do consumidor com o comerciante ou varejista devido as facilidades que proporcionará ao consumidor. De acordo com as previsões até 2.017, aproximadamente um bilhão de pessoas realizarão transações financeiras online ou utilizarão serviços baseados na mobilidade (VALOR ECONÔMICO, 2014).

Além dos números promissores, há casos de sucesso onde processos relacionados ao pagamento foram modernizados com o auxílio do *mobile payment*, como é o caso da Helsinki City Transport, empresa de transporte público localizada em uma pequena cidade no interior da Finlândia, que oferece aos seus usuários um serviço de pagamento de tickets de metrô por meio de seus dispositivos móveis. Os usuários podem adquirir seus tickets enviando um SMS (*Short Message Service*) para um número disponibilizado pela própria empresa. Aproximadamente 55% dos tickets vendidos atualmente são adquiridos dessa forma (BEGONHA, 2002).

Objetivos gerais e específicos

O objetivo geral é o desenvolvimento de uma aplicação móvel para a plataforma

Windows Phone, onde será possível aplicar os conceitos de *mobile payment* por aproximação. Devido a esse conceito, desenvolve-se com os recursos das tecnologias de troca de dados por aproximação presente em grande parte dos dispositivos móveis atuais o NFC (*Near Field Communication).*

Consideram-se objetivos específicos desse trabalho:

- Pesquisar e avaliar as API's (*Application Programming Interface)* de proximidade presentes na plataforma Windows;
- Confeccionar a documentação básica de todo o projeto obedecendo os padrões propostas pela Engenharia de Software;
- Desenvolver uma aplicação modularizada obedecendo os padrões de desenvolvimento orientado à objetos;
- Desenvolver a interface da aplicação obedecendo os conceitos propostos pela Interface Homem-Computador.

1 PANORAMA DAS APLICAÇÕES FINANCEIRAS MÓVEIS

O número de aplicações financeiras móveis (*apps)* tem crescido na mesma proporção em que a quantidade de sites cresceu no período de expansão da internet. Atualmente pequenos empresários conseguem gerenciar seus negócios por meio de seu dispositivo móvel e a ampla gama de aplicações que suportam suas atividades, proporcionando um nível de liberdade e comodidade, quanto aos consumidores, utilizam seus dispositivos móveis desde para se relacionar com outras pessoas à consulta de saldo de sua conta bancária.

Com base no estudo realizado pela Microsoft em conjunto com a M-Com pode-se elaborar um mapa conceitual, conforme demonstra-se na figura 1, do *mobile payment* demonstrando principalmente a dificuldade em categorizarmos esse novo método de pagamento (MICROSOFT e M-COM, 2013).

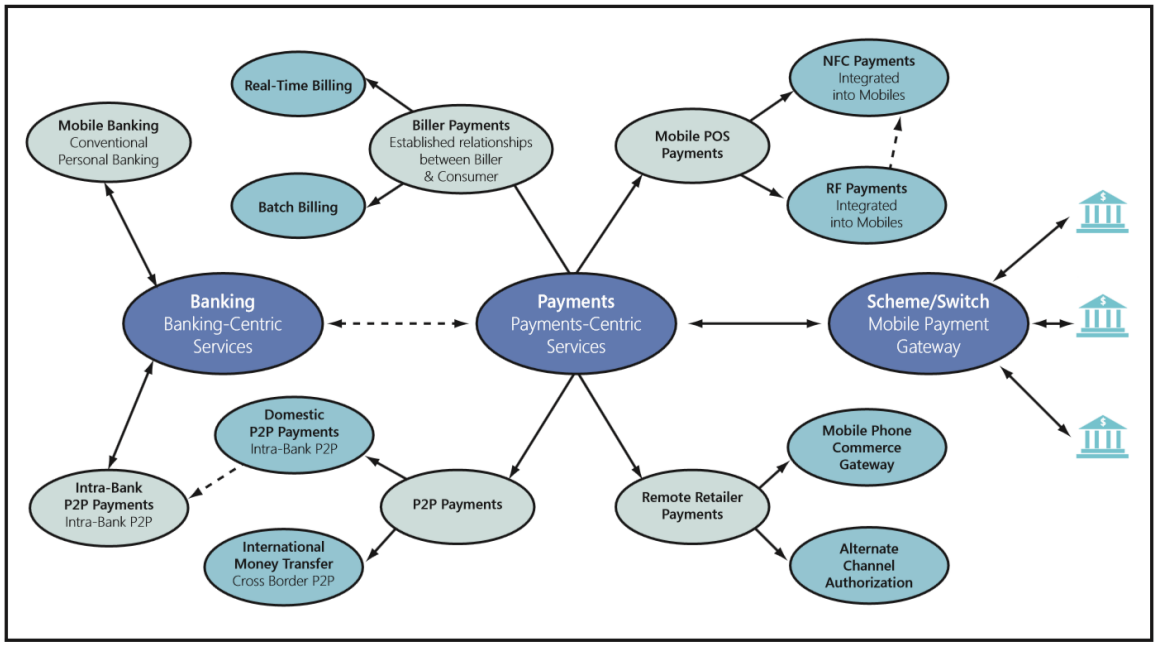


Figura 1. Mapa conceitual do *Mobile Payment* (Microsoft e M-Com, 2013)

As aplicações móveis financeiras podem ser organizadas em 5 categorias, as de mobile banking, mobile commerce, mobile point-of-sales (POS) e *mobile payment* onde subdividimos em *person-to-person* (P2P), *remote payment* e *mobile proximity payment*.

1.1 Mobile Banking

O *mobile banking* pode ser definido com uma aplicação móvel onde o usuário pode acessar e gerenciar os serviços financeiros disponibilizados pelo banco, cartão de crédito ou outros tipos de serviços financeiros (SMART CARD ALLIANCE, 2011).

De acordo com a matéria publicada no Jornal Valor Econômico no dia 11 de Abril de 2.014 menciona-se a dificuldade dos bancos em gerar receita e principalmente fazer com que seu lucro cresça. Para contornar essa situação eles têm optado em investir no *mobile banking* o que tem ser tornado uma arma fundamental na luta para manter clientes e cortar despesas. Uma transação por celular possui um custo médio de US$ 0,10 para os bancos dos Estados Unidos, cerca de metade do custo de uma transação feita por um computador e bem abaixo do custo médio de US$ 1,25 de uma operação no caixa eletrônico, segundo dados da empresa de pesquisa Javelin Strategy & Research. Ao todo, cerca de 60% dos usuários americanos que utilizam seus smartphones ou tablets mudaram de banco no quarto trimestre e que o *mobile banking* foi um fator importante na decisão, contra 7% no segundo trimestre de 2.010, segundo dados da consultoria AlixPartners (VALOR ECONÔMICO, 2014).

Outro ponto para o crescente uso dos dispositivos móveis para acessar serviços financeiros é o aumento da confiança nesse tipo de aplicativo tornando o uso mais atrativo ao usuário.

Ao mesmo tempo que os bancos investem nesse tipo de tecnologia, as operadoras de telefonia móvel fazem adequações e melhorias em suas redes para aumentar a qualidade das conexões de dados utilizadas pelos usuários.

1.2 Mobile Commerce

O *mobile commerce* ou M-Commerce refere-se ao uso dos dispositivos móveis para apoiarem uma transação comercial, ou seja, pode ser definido com qualquer transação que envolva a busca ou pagamento por bens ou serviços utilizando o navegador do dispositivo móvel, um aplicativo especializado ou mensagem de texto. Como o E-Commerce o M-Commerce inclui atividades de busca e pagamento de produtos adquiridos utilizando um dispositivo móvel. Por exemplo, o site de uma varejista como o Wal-Mart ou Americanas.com se adapta ao dispositivo móvel quando acessado permitindo que os consumidores possam realizar consultas e possivelmente a compra, porém a taxa de

conversão de um consumidor em realizar a compra é baixa o que influência diretamente no crescimento desse tipo de tecnologia.

O M-Commerce não se restringe apenas na venda e pagamento de produtos, permitindo que o usuário possa realizar diversas atividades correlatas como a recomendação de produtos aos amigos em redes sociais ou comparação de preços com outras lojas.

1.3 Mobile POS (Point of Sale)

O *Mobile POS* refere-se ao uso de um dispositivo móvel para substituir um terminal POS ou sistema tradicional e é normalmente utilizado para gestão de estoques, pagamento eletrônico, etc. Esse tipo de pagamento é que causa maior impacto no cotidiano dos consumidores pois altera radicalmente o modo como que realizam suas compras, porém os benefícios que trazem são compensadores.

Esse tipo de pagamento suporta diferentes tipos de dispositivos, desde os mais tradicionais com leitores de cartões de crédito e débito quando há transmissão de dados por meio de tecnologias de aproximação, como NFC ou Bluetooth.

Por exemplo, em um restaurante o garçom pode realizar os pedidos por um smartphone ou tablet e automaticamente os pedidos são exibidos por ordem de chegada em um televisor ou o pagamento de uma corrida de taxi por meio de adaptador para leitura de cartão de crédito acoplado ao smartphone do taxista.

2 MOBILE PAYMENT

O *mobile payment* refere-se ao pagamento realizado pelo consumidor por meio de um dispositivo móvel em troca dos serviços ou produtos. Esse tipo de pagamento pode ser feito em um POS ou entre dois ou mais consumidores.

Os consumidores podem utilizar seus dispositivos móveis para pagar serviços ou produtos, por exemplo, música, vídeos, assinaturas online para jogos ou filmes além de passagens de ônibus e metrôs.

Conforme demonstra-se a figura 2, as características do *mobile payment* leva-se em consideração algumas variáveis como tecnologia, montante da transação, localização (remota ou local) e mecanismo de financiamento.

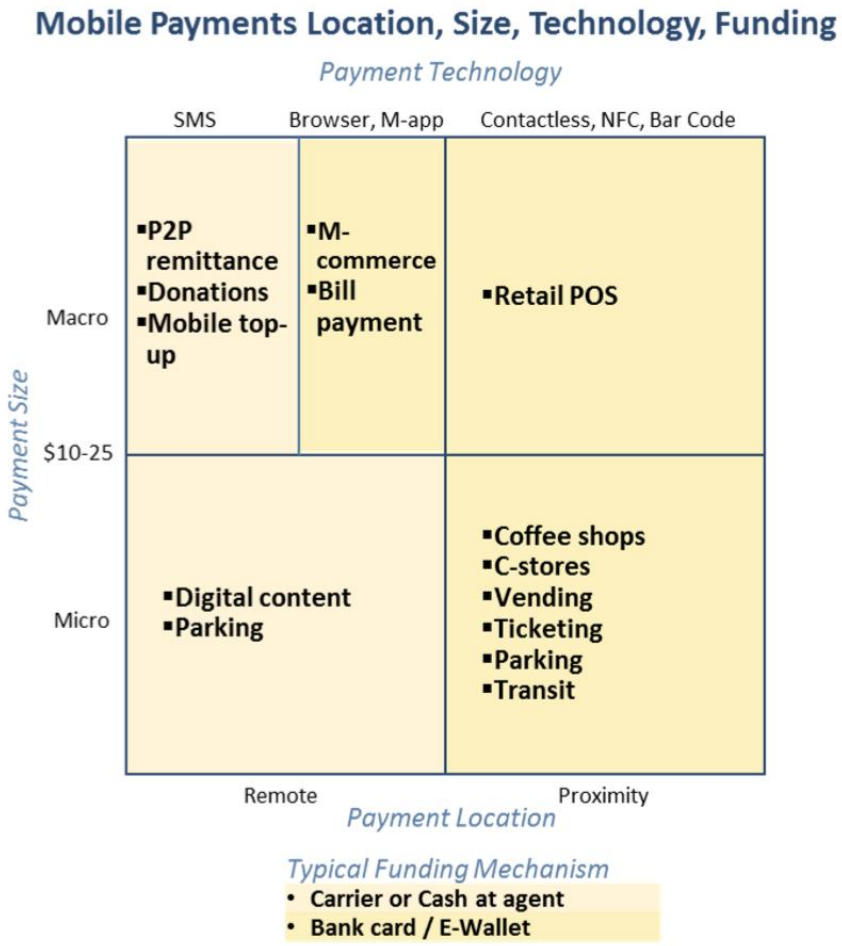


Figura 2. Características do *Mobile Payment* (Smart Card Alliance, 2011)

Tecnologia: *Mobile payments* podem utilizar diferentes tipos de tecnologia para concretizar uma transação. Os pagamentos remotos são normalmente realizados por meio do SMS, *mobile browser* ou um *mobile app*. Os pagamentos feitos por proximidade necessitam que os dispositivos móveis estejam fisicamente no mesmo lugar para que possam haver troca de dados utilizando tecnologias presentes em grande parte dos dispositivos móveis atuais (NFC e Bluetooth).

Montante da transação: Os valores envolvidos nas transações dependem muito de como será feito a transação, porém conseguimos categoriza-las em micro pagamentos ($10 - $25) e macro pagamentos (acima de $25).

Localização: Pode-se dividir a localização em remoto e por proximidade. Os pagamentos remotos não precisam ter interação com outros dispositivos e podem ser feitos fora do local onde está sendo feito a compra, por exemplo, quando pagamos uma compra por meio dos serviços do PayPal, em contrapartida os pagamentos por proximidade precisam ter integração com outros dispositivos para que por meio da troca de dados possam concretizar uma transação financeira com sucesso, obrigando o usuário a utilizar esse modo em lojas, restaurantes, etc.

Mecanismos de financiamento: Os pagamentos móveis podem contar com diversos mecanismos de financiamento, as transações podem ser feitas por uma conta pré-paga onde o consumidor adquire créditos por meio de um portal e realiza o pagamento pelos métodos tradicionais como boleto bancário ou cartão de crédito e débito.

O *mobile payment* devido a sua grande complexidade se subdivide-se em três categorias, o P2P, remote payments e proximity payments.

O P2P, *person-to-person* ou *peer-to-peer* permite que os consumidores possam realizar pagamentos entrem si por meio de um terceiro. Os consumidores pode realizar transações com recursos de um banco, cartão de crédito ou débito ou uma conta pré-paga.

Os *remotes payments* refere-se a transações em que os consumidores utilizam seu dispositivo móvel para realizar compras sem interagir com o POS, geralmente utilizando tecnologias como SMS, navegadores ou aplicativos móveis. Diferentemente do *remote payment*, o *proximity payment* refere-se a uma transação na qual o consumidor utiliza seu dispositivo móvel para pagamentos de bens e serviços por meio da integração entre o dispositivo móvel e um dispositivo POS utilizando tecnologias de troca de dados por aproximação, principalmente o NFC.

Como mencionado anteriormente, o processo de *mobile payment* é simples devido a fácil interação do usuário com a aplicação porém complexa quando analisa-se os

processos realizados pela aplicação para concretizar a solicitação de pagamento feita pelo usuário, devido ao grau de relevância dos dados trafegados por exemplo.

Observa-se na figura 3 o processo de maneira simplificada do *mobile payment* e as etapas fundamentais para que uma transação seja concluída com sucesso (MICROSOFT e M-COM, 2013)

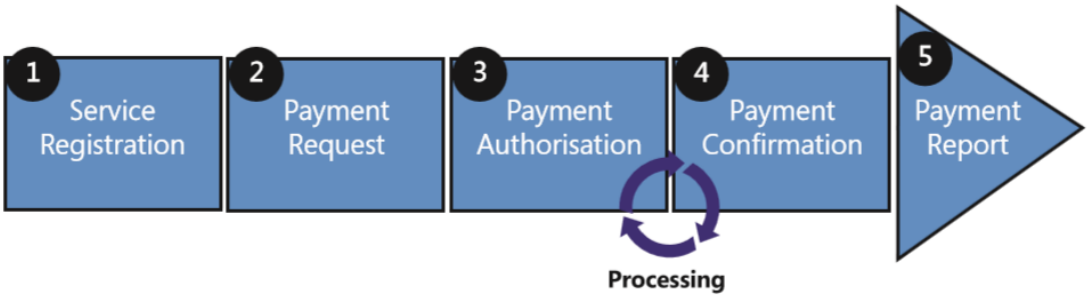


Figura 3. Processo taxonômico do *Mobile Payment* (Microsoft e M-Com, 2013)

Com base na figura 3, é possível definir as etapas do processo e seu grau de relevância para que a transação seja concluída com sucesso. Pode-se observar abaixo com mais detalhes o papel desempenhado por cada etapa:

1. **Service registration:** o usuário inicia o relacionamento com o provedor de pagamento, por meio de um cadastro feito pelo próprio dispositivo móvel ou pelos métodos tradicionais como caixa eletrônico ou por um site disponibilizado pelo próprio provedor.
2. **Payment request:** o usuário realiza a solicitação de pagamento à um terceiro (pessoa jurídica ou não) e é nessa etapa que já há os dados de relevância para conclusão do processo, como valor da transação, receptor, etc.
3. **Payment authorisation:** o usuário confirma e autoriza o pagamento antes de ser processo pelo provedor.
4. **Payment confirmation:** a solicitação de pagamento é enviada ao provedor e analisado, ou seja, é verificado se realmente o pagamento tem as condições necessárias para ser aprovado. Após alguns minutos o usuário recebe a confirmação ou não do pagamento solicitado.
5. **Payment report:** O usuário pode consultar a qualquer momento as solicitações de pagamentos feitas ao provedor.

Na transição entre as etapas *Payment Authorisation* e *Payment Confirmation* é realizado todo o processo de validação da solicitação do usuário, portando essa transição de etapas é considerado de alta criticidade para o sucesso da operação pois qualquer erro causará impacto significativo nos resultados do processo de pagamento.

Outra grande característica do processo citado e que confunde muitos especialistas dessa área é que parte do processo podem ser executados com o apoio dos métodos tradicionais, por exemplo, caixas eletrônicos. É possível observar na figura 4 a interação dos métodos tradicionais com o *mobile payment.*

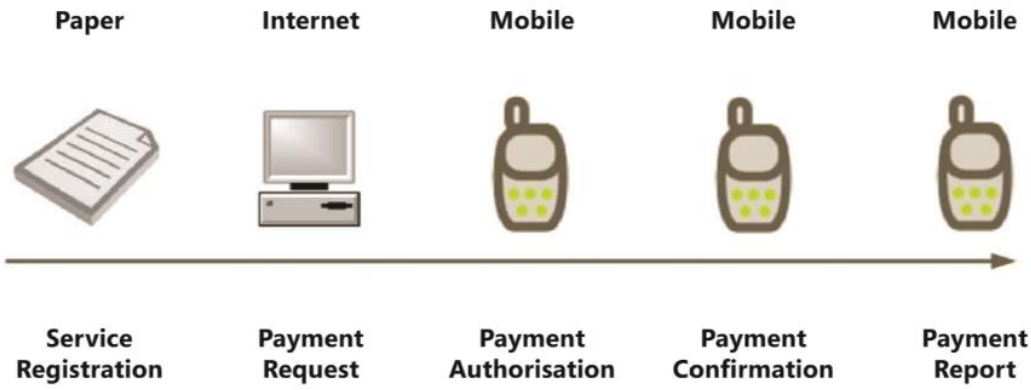


Figura 4. Integração dos métodos tradicionais com o *Mobile Payment* (Microsoft e M-Com, 2013)

A seguir é possível observar com mais detalhes o papel desempenhado por cada etapa:

1. **Service registration:** o usuário se registra em um serviço de *mobile payment* por meio de um formulário tradicional bancário.
2. **Payment request:** o usuário pela internet, pode solicitar um pagamento, como é feito atualmente em lojas on-line ou ao realizar uma compra pela internet ele pode optar em fazer o pagamento pelo seu dispositivo móvel.
3. **Payment authorisation:** o usuário autoriza o pagamento por meio de seu dispositivo móvel.
4. **Payment confirmation:** o usuário recebe em seu dispositivo móvel a confirmação da solicitação de pagamento.
5. **Payment report:** o usuário pode consultar em qualquer momento o pagamento que foi realizado.

No cenário proposto na figura 4, conclui-se que o *mobile payment* não eliminará

os métodos atuais utilizando-os em parte do processo métodos que foram aperfeiçoados com o passar do tempo como é o caso do acesso a conta bancária por meio da internet por exemplo.

Quando fala-se de *mobile payment* não há como não relacionar com a gestão de riscos associados aos serviços eletrônicos, em nenhum outro lugar as demandas por segurança são tão intensas no que dos pagamentos. O que nos obriga à considerarmos segurança como um dos fatores cruciais para a consolidação do *mobile payment* no mercado atual. Nesse contexto propõe-se o método de segurança denominado *End-to-End Security* proposta pela Microsoft e M-Com em seu estudo publicado em conjunto (MICROSOFT e M-COM, 2013).

Para desenvolver soluções móveis de pagamento precisa-se garantir a integridade dos dados desde o cadastro do usuário para uso dos serviços até as consultas das transações realizadas. Nesse contexto considera-se os itens a seguir:

1. **Autenticação:** define quais serão os requisitos para acesso ao serviço de pagamento de acordo com o modo de acesso (SMS, *Mobile Browser*, *Mobile Application*, etc.).
2. **Segurança no transporte:** define quais as tecnologias de segurança que serão aplicadas no tráfego de dados.
3. **Ferramentas de gerenciamento de risco:** é a capacidade de proteger o processo de pagamento de falhas como ataques, desastres naturais, etc.
4. **Comportamento do consumidor:** entender os anseios e dificuldades dos consumidores no uso dessas tecnologias de pagamento.
5. ***Compliance:*** realizar constantes auditorias em curtos intervalos de tempo em todo o processo com o objetivo de minimizar falhas e realizar melhorias constantes amadurecendo os serviços prestados aos consumidores. Além disso, é necessário adequar o processo de pagamento as regulamentações do governo.

Como é o caso de outros canais e meios de pagamento, as instituições financeiras devem incorporar ferramentas de gerenciamento de risco em suas soluções de pagamento móvel. Exemplos de boas práticas nesse contexto incluem:

- Possibilidade de auditar e registrar as atividades dos funcionários e dos

usuários;

- Monitoramento em tempo real e alerta em situações anormais, como *time out* do processamento de pagamentos, erros de acesso ao aplicativo, etc.;
- Gerenciamento de regras de negócio para controle dos usuários que acessam os serviços;
- Proteção contra ataques típicos, como código malicioso;
- Integração com outros sistemas de risco e fraudes que possam atuar de forma proativa preservando a integridade dos usuários que utilizam a aplicação para pagamentos;

Outro ponto interessante é a preocupação dos usuários em relação aos pagamentos móveis, algumas ações técnicas e comportamentais são recomendadas para as instituições financeiras para que esse cenário mude, entre elas cita-se:

- Envio de informações ou alertas para os usuários estabelecendo um canal de confiança e estreitando a relação entre ambas as partes;
- Fornecer e comunicar garantias aos usuários em caso de dúvidas e falhas, como por exemplo, atendimento rápido;
- Reforçar a segurança e garantir que a experiência do usuário no uso da aplicação seja a melhor possível;
- Manter o usuário sempre informado das melhorias feitas na aplicação e nos processos envolvidos deixando claro o comprometimento o usuário;

Relaciona-se o avanço do *mobile payment* com a popularização dos dispositivos móveis para uso em tarefas cotidianas. Observa-se claramente isso, no relatório divulgado em 2.014 pela GSMA, organização que representa as operadoras de telefonia de todo o mundo, onde menciona-se diversos indicadores entre eles a previsão de que em 2.017 o total de dispositivos móveis será de 2.890 milhões sendo que em 2.013 era 1.457 milhões. Percebe-se a evolução da base de dispositivos móveis, em milhões de unidades, pelos dados demonstrados na figura 5 (VALOR ECONÔMICO, 2014; GSMA, 2014).

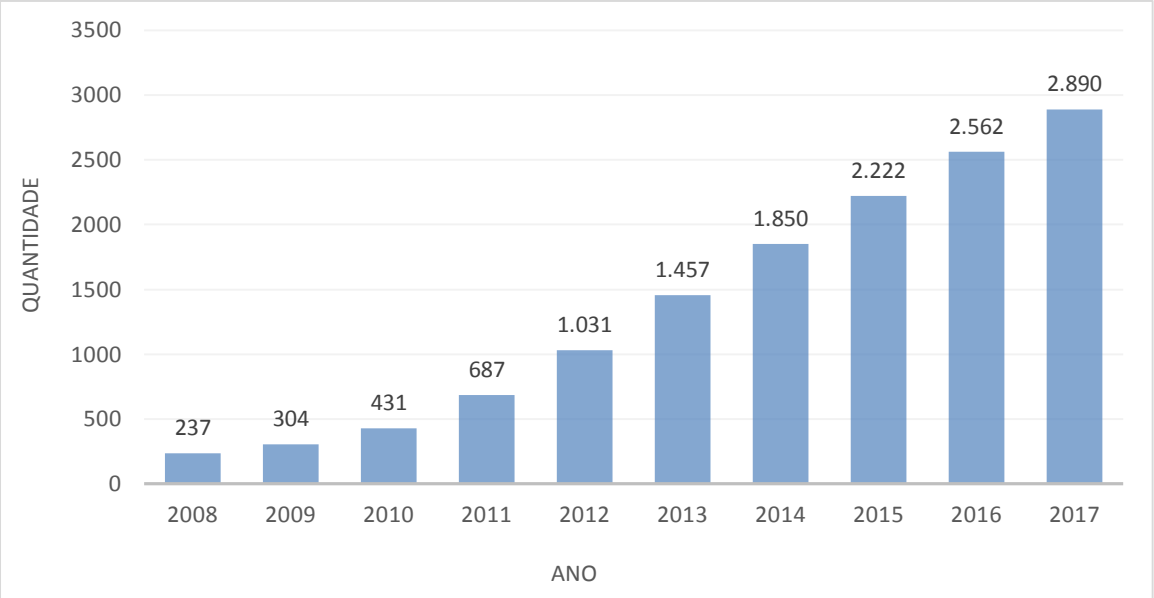


Figura 5. Evolução da base de dispositivos móveis (GSMA, 2014)

Além dessa popularização dos dispositivos móveis, nota-se as receitas obtidas com o mercado de mobilidade, que sofreram uma forte diversificação. Observa-se a composição dessa diversificação das fontes de receitas pelos números obtidos em 2.013, 205 bilhões de dólares em receita oriundos de aplicativos, conteúdo e propaganda enquanto 1,2 trilhões de dólares vieram de serviços prestados pelas operadoras de telefonia. Na figura 6 é possível observar com maior detalhe o que foi mencionado anteriormente.

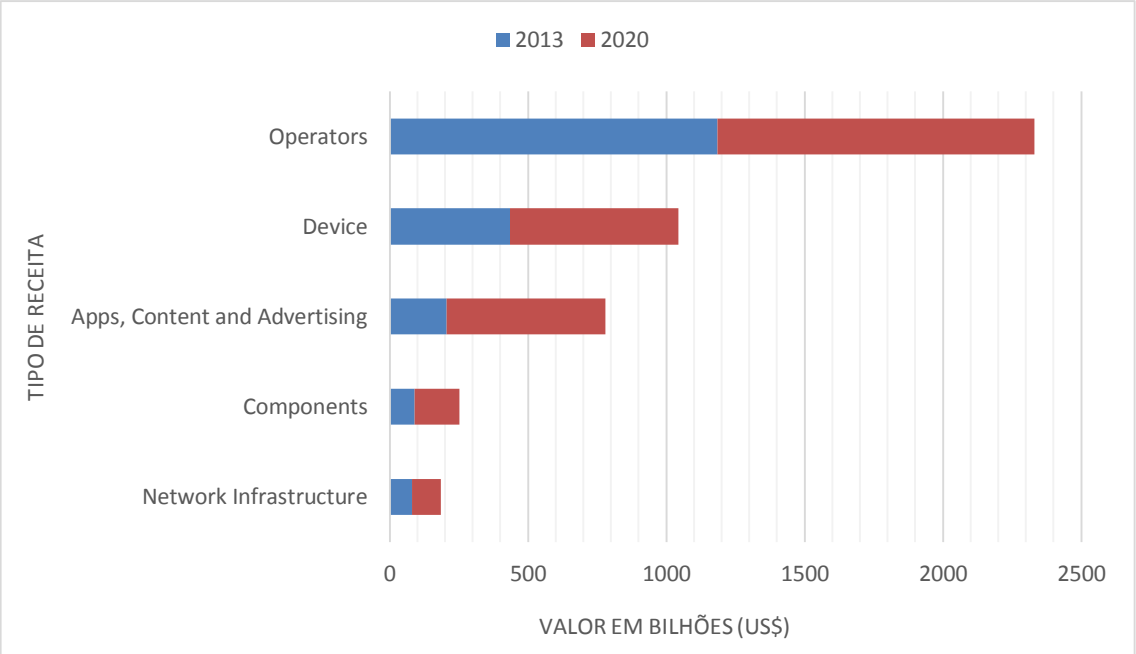


Figura 6. Receitas obtidas pelo mercado de mobilidade (GSMA, 2014)

Outro grande benefício proporcionado pelo *mobile payment* é o impacto social propiciando métodos fáceis e baratos para acesso aos serviços financeiros tradicionais tais como pagamentos, poupança, crédito, etc. Devido ao desafio de construir sistemas financeiros inclusivos em 2.005 esse assunto ganhou destaque na ONU (Organização das Nações Unidas) que definiu que esse mesmo ano seria o "Ano Internacional do Microcrédito". Entendida não só como um mecanismo de redução de desigualdades sociais, mas também como uma ferramenta para democratização do acesso ao crédito, a inclusão financeira é fundamental para o desenvolvimento econômico e social de um país além da melhoria da qualidade de vida de sua população.

Se há consenso na literatura sobre a importância que o setor financeiro desempenha um papel fundamental no processo de desenvolvimento dos países, as diferentes dimensões sobre esse tema foram abordadas de forma diversa. Grande parte da literatura sobre finanças e crescimento econômico focou no tamanho do setor financeiro e no seu impacto sobre o crescimento, como mencionado por Levine (2005;2007). Porém existem trabalhos focados na relação entre o setor financeiro e a desigualdade utilizando medidas de natureza macro (MOOKERJEE; KALIPIONE,2010).

Mesmo com literaturas com opiniões difusas, nota-se um consenso geral segundo o qual o desenvolvimento financeiro pode beneficiar os pobres (LEVINE, 2007; AKHTER; DAILY, 2009). Algumas pesquisas que se destacam nesse contexto, como Honohan (2008), cujos os resultados econométricos, embora sem robustez, segurem que o acesso financeiro é negativamente associado a desigualdade de renda.

Como não há clareza na direção da causalidade de inclusão financeira, as regressões feitas pelo autor confirmam a associação inversa entre o acesso financeiro e a pobreza. Mookerjee e Kalipioni (2010) fazem hipóteses que as barreiras para o acesso a serviços financeiros podem exercer um efeito negativo significativo na desigualdade de renda entre os países. Para explorar essa relação feita anteriormente, a partir de uma amostragem de 70 países desenvolvidos e em desenvolvimento, os autores usam como medida as barreiras, valor mínimo, acesso aos empréstimos concedidos e concluem que há suporte robusto para a ideia geral de que as medidas mais focadas no desenvolvimento financeiro são boas para as camadas mais pobres da população, ainda que as barreiras para alguns serviços, como valores mínimos para abertura de contas sejam nocivos para a distribuição de renda da população (SCHIAVINATTO;SCHIMITDT, 2011).

Contudo é possível afirmar que se o devido acesso aos serviços financeiros, a população permanece restrita as oportunidades que dependem diretamente do

desenvolvimento do país em que vivem. No Brasil, que é reconhecido pela sofisticação do seu Sistema Financeiro Nacional (SFN) ainda possui alguns desafios para a inclusão (Figura 7).

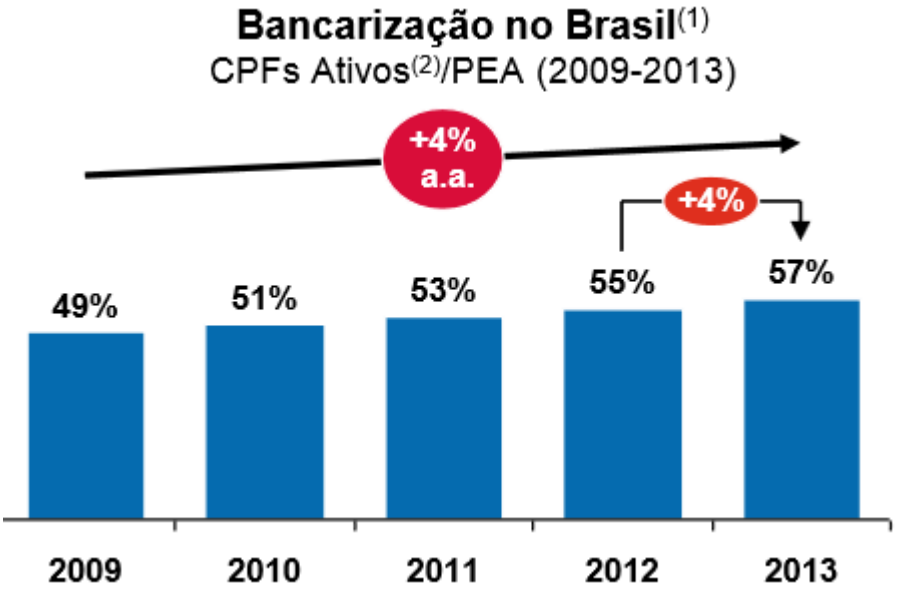


Figura 7. População bancarizada no Brasil (FEBRABAN, 2014)

Os dados revelam que 57% da população brasileira possui pelo menos uma conta em banco, conforme dados fornecidos pela FEBRABAN (Federação Brasileira de Bancos), enquanto 43% declaram não possuir nenhuma conta demonstrando um dos principais desafios enfrentados pelo mercado financeiro no nosso país. Ao analisar números mais amplos, percebe-se que o alcance dos esforços para a popularização ao acesso dos serviços financeiros básicos não é igualitário.

Em âmbito regional, revela-se que o percentual de cidadãos com conta bancária é muito maior nos estados do Centro-Oeste, Sul e Sudeste. Entretanto nota-se o fato de que as regiões Norte e Nordeste possuírem um número inferior de agências e pontos de atendimento para cada 100 mil habitantes economicamente ativas quando comparado com as demais regiões do país, mesmo com o crescimento registrado nos últimos cinco anos. De acordo com a figura 8 é possível visualizar que no último ano a região Sudeste apresentou o mesmo crescimento absoluto em agências, cerca de 340 agências, e o Centro-Oeste o maior crescimento percentual, impulsionado pela relevância da economia agrícola (FEBRABAN,2014).

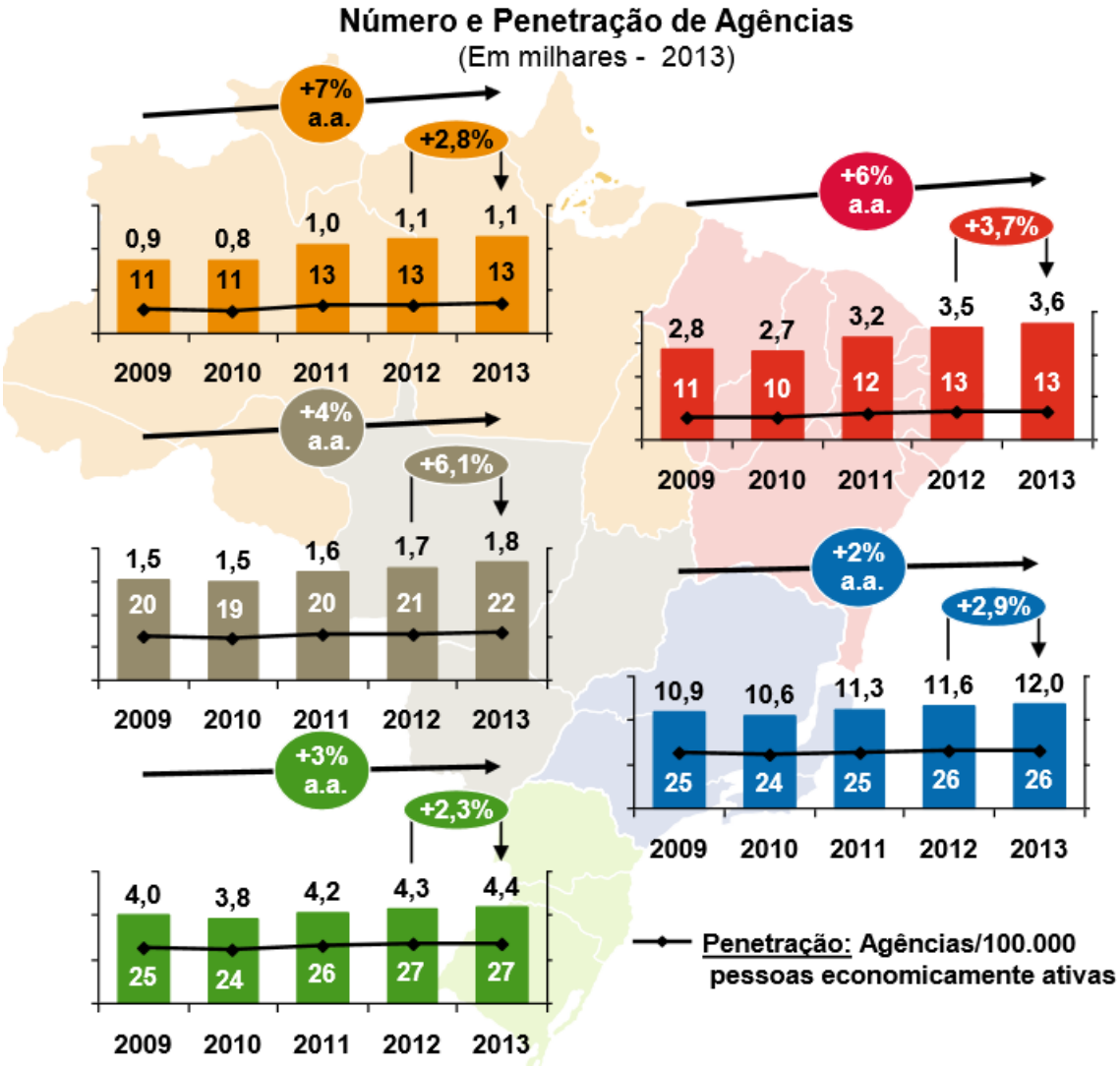


Figura 8. Distribuição numérica das agências brasileiras (FEBRABAN, 2014)

Uma das alternativas para solucionar e diminuir a inclusão financeira é por meio do *mobile payment*, conforme aponta o relatório da GSMA *Mobile Banking for the Unbanked: State of Industry 2013* (PÉNICAUD e KATAKAM, 2013) onde menciona-se diversos casos de sucesso onde o *mobile payment* permitiu a integração da população com os serviços financeiros em regiões da África e da América Latina.

Um dos casos de sucesso mencionado anteriormente ocorreu na Costa do Marfim, que em Junho de 2.013, empresas de *mobile payment* como CelPaid, Moov, MTN, Orange e Qash Services registraram em conjunto aproximadamente cinco milhões de contas, sendo que 35% dessas contas eram ativas.

Depois de um começo cheio de desafios, à primeira vista a oportunidade para o *mobile payment* na Costa do Marfim parece enorme. Com uma população de 19,8 milhões

e o maior PIB (Produto Interno Bruto) *per cápita* da região, sendo que o país apresenta-se com uma das economias mais dinâmicas do Oeste da África. Mesmo com esse dinamismo econômico, cerca de 10,7% dos adultos possuem acesso à uma instituição financeira formal, o *mobile payment* parece ser um caminho obvio para incrementar a instituição financeira. (PÉNICAUD e KATAKAM, 2013)

Com isso, o Banco Central dos Estados da África Ocidental (BCEAO) verificou que o *mobile payment* tinha potencial para incrementar significativamente a inclusão financeira. Em 2.006, o BCEAO emitiu regulações sobre o dinheiro eletrônico que qualificavam instituições não bancárias para obter uma licença como emissores de dinheiro eletrônico. Por meio dessa regulamentação, um emissor de dinheiro eletrônico pode ser um banco ou não.

Desde que emitiu-se a regulamentação, cinco empresas lançaram serviços de *mobile payment* na Costa do Marfim: Orange (Dezembro de 2.008), MTN (Outubro de 2.009, Moov (Janeiro de 2.013), CelPaid (Fevereiro de 2.011) e Qash Services (Novembro de 2.013).

Um fator externo que impulsionou a adoção do *mobile payment* foi o retorno de um país à paz civil e a recuperação econômica em 2.012. Uma década de crises políticas que culminaram em 2.010, quando os candidatos reclamavam ter ganhado as eleições presidenciais, desencadeando um conflito nacional que desestabilizou a economia e deixou a população vulnerável.

Em torno de uma semana, em Fevereiro de 2.011, quatro bancos suspenderam suas operações, criando uma importante escassez de dinheiro. Se intensificou a desconfiança pública a respeito do sistema financeiro, à qual impactou significativamente nos provedores de serviços de *mobile payment.* A limitada presença dos bancos presentes na zona rural, fez também com que a gestão da liquidez ficasse mais difícil e limitou a capacidade dos provedores de *mobile payment* em prover serviços financeiros efetivos a população. Entretanto, o retorno da paz civil tem ajudado a estimular a economia. Os provedores presentes na Costa do Marfim estão em acordo que as crises sofridas após as eleições presidenciais tiveram impacto negativo em suas operações e atribuem o crescimento obtido em 2.012 à recuperação econômica do país. (EXCHANGE, 2013)

O aumento do uso do dinheiro móvel na Costa do Marfim, entretanto, é resultado somente da estabilidade econômica vivida pelo país atualmente. Nos últimos anos, os provedores de *mobile payment* estão utilizando novas técnicas mais efetivas para aumentar o uso dessa tecnologia no cotidiano da população marfinense.

3 IMPLEMENTAÇÃO DOS CONCEITOS DE MOBILE PAYMENT

Conforme afirmou-se anteriormente, não é possível classificar a aplicação desenvolvida em um dos tipos de *mobile payment* devido à complexidade do próprio *mobile payment* como também os diferentes cenários que a aplicação desenvolvida pode ser aplicada.

Na figura 9, ilustra-se o processo desenvolvido para suportar a transação de pagamento solicitada pelo usuário, ou seja, tem-se a junção de diversas tecnologias para atender os objetivos propostos inicialmente.

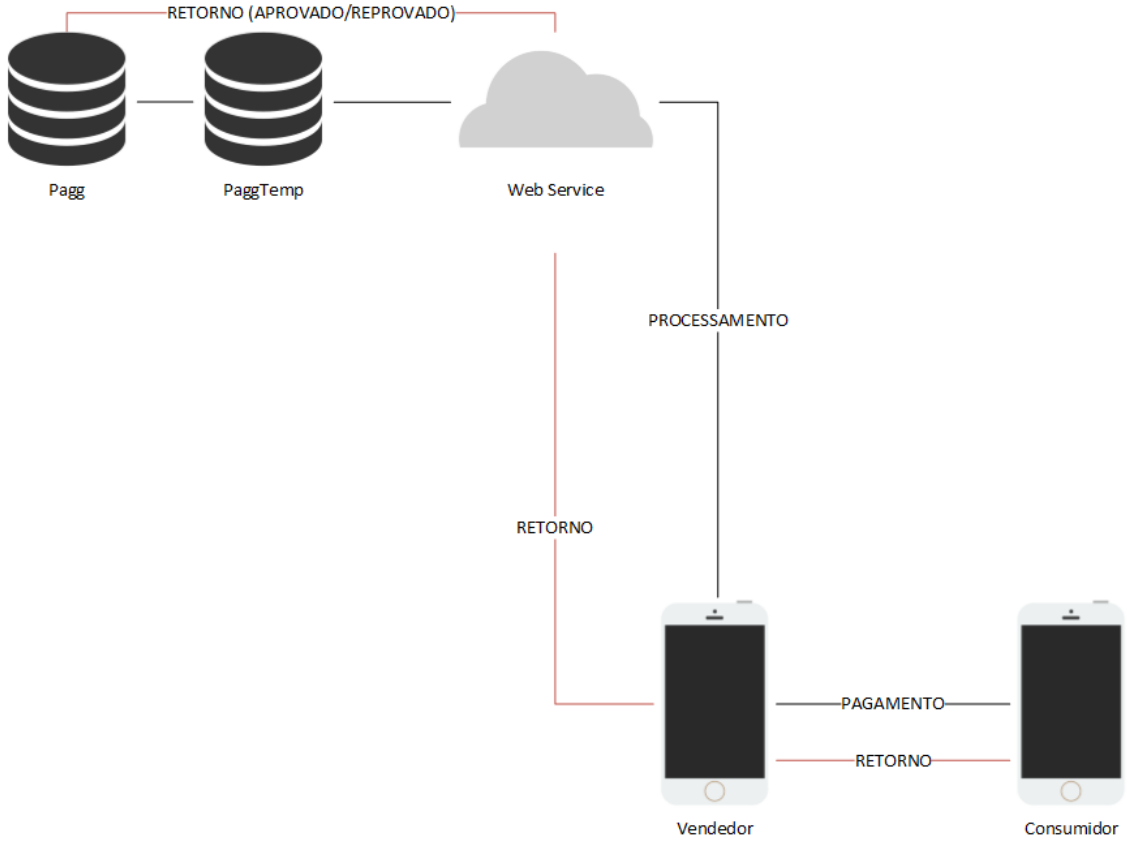


Figura 9. Diagrama conceitual implementado (Fonte Própria)

Observa-se no digrama proposto na figura 9, o quão complexo é a solicitação de um simples pagamento, mesmo que a quantidade de dados trafegados em cada etapa do processo e a relevância dessas informações tornam o processo de alta criticidade para o sucesso dessa solicitação. Além disso a preocupação com a segurança em todo o processo se torna um pré-requisito para a conclusão do processo sem problemas críticos.

O processo de pagamento inicia-se com a solicitação do consumidor, que por meio da tecnologia NFC presente em seu dispositivo, envia a solicitação de pagamento para o

dispositivo móvel do vendedor. Após, o recebimento desses dados o dispositivo móvel do vendedor envia a solicitação de pagamento para o *web service* que será o intermediador entre a aplicação e o banco de dados.

Logo após, o *web service* retornará à aplicação presente no dispositivo móvel no vendedor o resultado da solicitação de pagamento, permitindo que a solicitação se repita caso haja necessidade.

A seguir detalhe-se quais os processos realizados em cada etapa bem como a estrutura tecnológica utilizada para suportar as necessidades exigidas.

3.1 Aplicação *Front End* ou *End User*

A aplicação *front end* ou *end user* desenvolvida para os usuários finais utilizou-se a plataforma *Windows Phone* com a linguagem nativa de programação dessa plataforma, o *C#*. O *Windows Phone* é a plataforma da *Microsoft* voltada para dispositivos móveis lançada em 2.010 e que está atualmente na versão 8.1. Devido as necessidades exigidas pelo projeto, dividiu-se a aplicação em dois módulos sendo uma dedicada aos consumidores e outra dedicada aos vendedores que receberão e processarão as solicitações de pagamentos enviadas.

Na figura 10, demonstra-se o funcionamento da solicitação de pagamento enviada pelo consumidor ao vendedor utilizando a tecnologia NFC presente em grande parte dos dispositivos móveis atuais.

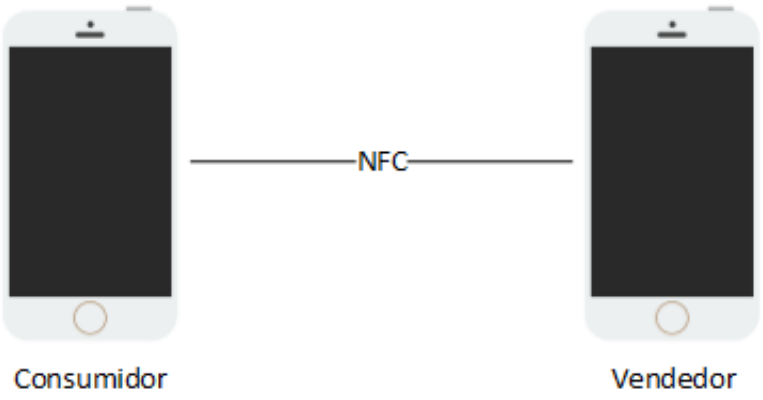


Figura 10. Diagrama simplificado do processo de pagamento (Fonte Própria)

O *Near Field Communication* (NFC), é uma tecnologia de comunicação sem fio

de curto alcance entre dispositivos eletrônicos. A comunicação somente ocorrerá quando os dois dispositivos são postos à uma distância bem pequena. Desenvolvida pela Sony e Philips, porém algumas literaturas menciona-se a participação da Nokia, o seu padrão chama-se *NFCIP-1* (*Near Field Communication Interface and Protocol*), contido nos documentos (ISO/IEC 18092:2004) e (ECMA 340). Esse padrão especifica os esquemas de modulação, codificação de bit, taxas de transmissão e formato de quadro de interface aérea, assim os mecanismos de inicialização e controle de colisão. O padrão *NFCIP-1* podem ser definidos em três tipos de comportamento que um dispositivo NFC pode apresentar: iniciador ativo, alvo ativo e algo passivo (SILVA e CAVALCANTE, 2010).

Os dispositivos que possuem essa tecnologia atualmente não só implementam o padrão *NFCIP-1*, mas também o *NFCIP-2*, que são descritos nos documentos (ISO 21841) e (ECMA 352). O padrão *NFCIP-2* especifica o mecanismo que detecta e seleciona um dos quatro modos de operações possíveis, conforme descritos abaixo:

- *Peer-to-Peer (ou Terminal-to-Terminal)*, padronizado no *ISO 18092;*
- *PCD (Proximity Coupling Device*, ou "leitor"), padronizado no *ISO 14443;*
- *PICC (Proximity Integrated Circut Card,* ou "cartão"), padronizado no *ISO 14443;*
- *VCD (Vincinity Coupling Device,* ou "leitor"), padronizado no *ISO 15693.*

Os dispositivos NFC que implementam o *NFCIP-2*, já são, portando, compatíveis com uma grande infraestrutura de leitores e tags *RFID* (*Radio-Frequency Identification)* já implementando em diversas partes do mundo, conforme observa-se na figura 11. O dispositivo NFC realiza uma varredura por dispositivos compatíveis com os protocolos que suporta em menos de 200 ms (FISHER,2009).

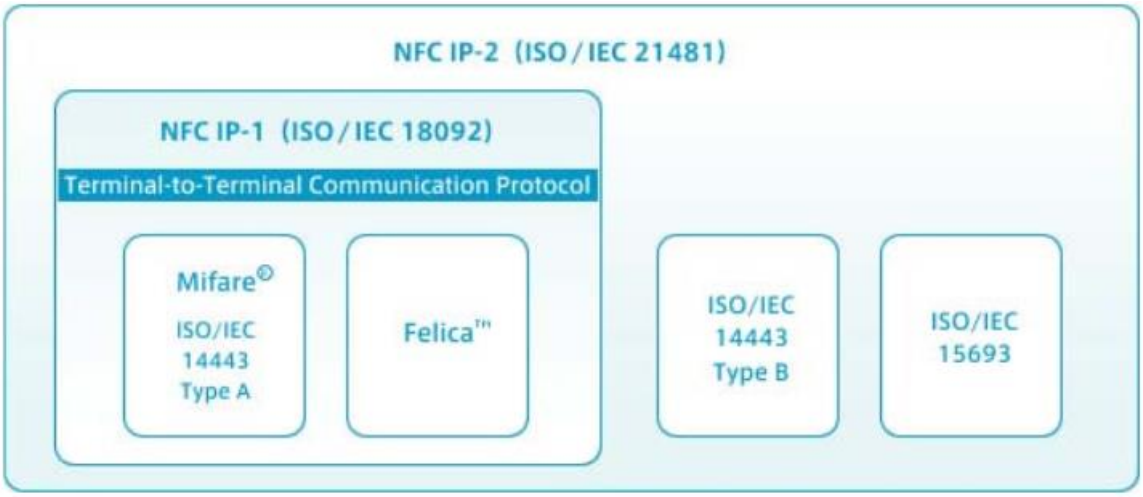


Figura 11. Encapsulamento dos padrões *NFCIP-1* e *NFCIP-2* (NFC Portal)

Por meio da tecnologia NFC presente no dispositivo móvel, enviamos para o outro dispositivo dados estruturados (Figura 12).

```
public class Payment
{
    [System.Runtime.Serialization.DataMember]
    1 reference
    public int idConsumidor
    {
        get;
        set;
    }

    [System.Runtime.Serialization.DataMember]
    2 references
    public string Nome
    {
        get;
        set;
    }

    [System.Runtime.Serialization.DataMember]
    2 references
    public string Sobrenome
    {
        get;
        set;
    }

    [System.Runtime.Serialization.DataMember]
    1 reference
    public string Login
    {
        get;
        set;
    }

    [System.Runtime.Serialization.DataMember]
    2 references
    public decimal Valor
    {
        get;
        set;
    }
}
```

Figura 12. Classe em C# com os atributos utilizados na transmissão (Fonte Própria)

Ao realizar o pagamento, o consumidor informará somente o valor da transação e os demais dados para completa-la serão coletados pela própria aplicação tornando o processo de pagamento simplificado e de fácil manuseio para usuários com pouca experiência com o uso da tecnologia, é possível observar na figura 13 o processo que envia os dados, por meio do NFC, para o outro dispositivo.

```
private void PublishCustomBinaryMessageToDevice()
{
    try
    {
        Payment payment = new Payment()
        {
            idConsumidor = Profile.idConsumidor,
            Nome = Profile.Nome,
            Sobrenome = Profile.Sobrenome,
            Login = Session.Login(),
            Valor = Convert.ToDecimal(TextBoxValor.Text.Replace("R$", "").Replace(" ", ""))
        };

        MemoryStream stream = new MemoryStream();

        var serializer = new System.Runtime.Serialization.DataContractSerializer(typeof(Payment));
        serializer.WriteObject(stream, payment);

        byte[] array = new byte[stream.Length];

        stream.Position = 0;
        stream.Read(array, 0, (int)stream.Length);

        DataWriter dataWriter = new DataWriter();
        dataWriter.ByteOrder = ByteOrder.BigEndian;
        dataWriter.WriteBytes(array);

        publishID = device.PublishBinaryMessage("Windows.urn:pagg.com:payment", dataWriter.DetachBuffer(), Message);
    }
    catch
    {
        MessageBox.Show("Ocorreu um ero ao processar sua solicitação.", "Ops!", MessageBoxButton.OK);
    }
}
```

Figura 13. Método em C# que envia os dados estruturados (Fonte Própria)

Logo no início do método *PublishCustomBinaryMessageToDevice* percebe-se que aplicação coleta os dados necessários para que a solicitação de pagamento se complete e as aloca na classe *Payment* com os atributos correspondentes.

A serialização de objetos, realizada pelo *namespace System.Runtime.Serialization* é um método presente na linguagem de programação C# *que* contém classes que podem ser utilizadas para objetos de serialização e desserialização. Serialização é o processo de conversão de um objeto ou um gráfico de objetos em uma sequência linear de *bytes* para armazenamento ou transmissão para outro local. A desserialização é o processo de converter as informações armazenadas e recriar os objetos a partir delas (MICROSOFT, 2014).

Ao final do processo, o objeto *Payment* serializado é enviado por meio de uma mensagem para o outro dispositivo. A *tag* ***Windows.urn:pagg.com:payment*** é o endereço do destinatário dos dados enviados, que neste caso, será a própria aplicação instalada no

outro dispositivo móvel.

No outro dispositivo móvel é feito o processo reverso, ou seja, a desserialização dos dados enviados em objeto, conforme demonstra a figura 14.

```
private void SubscribeForCustomBinaryMessage()
{
    subscribeID = device.SubscribeForMessage("Windows.urn:pagg.com:payment",
        (ProximityDevice sender, ProximityMessage message) =>
        {
            byte[] array = new byte[message.Data.Length];

            using (DataReader dataReader = DataReader.FromBuffer(message.Data))
            {
                dataReader.ByteOrder = ByteOrder.BigEndian;
                dataReader.ReadBytes(array);

                using (MemoryStream stream = new MemoryStream(array))
                {
                    var serializer = new System.Runtime.Serialization.DataContractSerializer(typeof(Payment));
                    Payment payment = (Payment)serializer.ReadObject(stream);

                    log += String.Format("Solicitação de pagamento de {0} {1} no valor de R$ {2} recebida.",
                                         payment.Nome,
                                         payment.Sobrenome,
                                         payment.Valor);
                }
            }
        });
}
```

Figura 14. Método em C# que recebe os dados estruturados (Fonte Própria)

Logo no início do método *SubscribeForCustomBinaryMessage* é realizado o processo de desserialização dos dados, conforme já mencionou-se é o processo de converter as informações armazenadas e recriar os objetos a partir delas, atribuindo os dados aos atributos correspondentes na classe *Payment.*

Além do processo de pagamento, a aplicação constitui-se por outras classes denominadas *Helpers*, ou seja, que contribuem ou complementam as funcionalidades que a aplicação possui. Na figura 15, é possível observar a estrutura das classes que apoiam as funcionalidades da aplicação, permitindo a alocação de recursos em métodos isolados com o objetivo de reutilizar em diferentes cenários dentro da aplicação.

Por exemplo, a classe *Sync* é utilizada para que os usuários possam se cadastrar para uso da aplicação bem como sincronizar seus dados para seu dispositivo móvel.

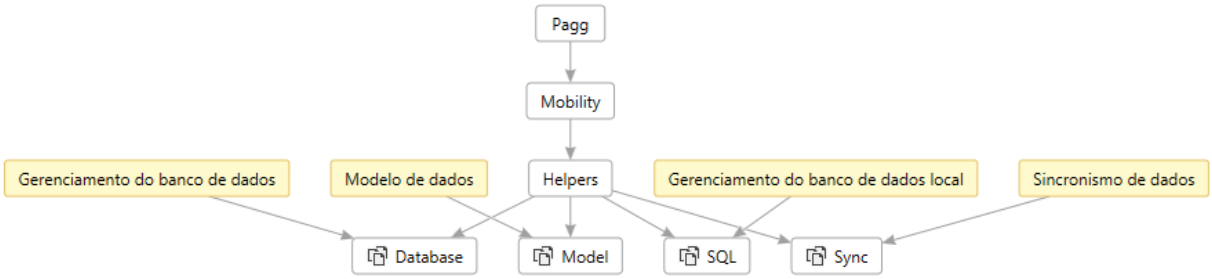


Figura 15. Diagrama de classes de apoio (Fonte Própria)

A seguir detalhe-se o papel desempenhado por cada classe bem como sua estrutura lógica.

3.1.1 Estrutura lógica

A classe *SQL* tem como principal objetivo de armazenar os métodos utilizados para gerenciamento dos dados contidos no banco de dados local da aplicação. Devido aos problemas de conexão que enfrentamos em nosso país, a aplicação foi desenvolvida com o conceito *off-line*, ou seja, mesmo sem a conexão de dados o usuário consegue utilizar a aplicação para consulta dos últimos pagamentos, dos estabelecimentos que aceitam esse método de pagamento, etc. (Figura 16).

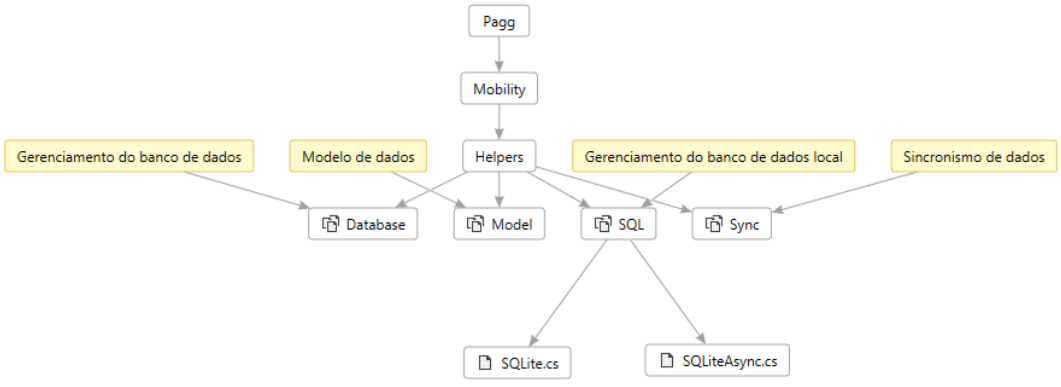


Figura 16. Diagrama de classes SQL (Fonte Própria)

Utiliza-se com o banco de dados local o *SQLite*, uma biblioteca que implementa sem a necessidade de um servidor um banco de dados SQL, transacional e autossuficiente. Essa biblioteca é de domínio público, portando, livre de qualquer fim comercial ou

instituição privada. O *SQLite* realiza a leitura e escrita diretamente em arquivos gravados nos discos comuns dos dispositivos, seja ele móvel ou não. Além disso permite criar diversos objetos que encontramos em bancos de dados mais robustos como *Microsoft SQL Server* ou *Oracle DB*, como tabelas, visões, índices, gatilhos, etc. em um único arquivo. O formato desse arquivo é multi-plataforma, sendo que se pode copiar esse arquivo livremente entre sistemas de 32 bits e 64 bits ou entre arquiteturas *big-endian* e *little-endian* (SQLITE, 2014).

A *Microsoft* possui uma versão do *Microsoft SQL Server* voltada para o uso em aplicações móveis, denominada *Microsoft SQL Server Compact* que está atualmente na versão 4.0, porém nos testes realizados durante o desenvolvimento dessa aplicação identificou-se que as leituras e escritas nesse banco exigiam um processamento elevado do dispositivo móvel do usuário além de causar lentidão na navegação entre as telas da aplicação impossibilitando o uso de certas funcionalidades, que fez optar-se pelo *SQLite.*

Para utilizar o *SQLite* junto com a plataforma móvel *Windows Phone 8.1* utilizou-se o *package SQLite for Windows Phone* disponibilizado pela própria equipe de desenvolvimento do *SQLite* para aplicativos voltados a essa plataforma. Esse *package* cria duas subclasses denominadas respectivamente *SQLite* e *SQLiteAsync* onde armazenam todos os métodos necessários para manipulação do banco de dados, ou seja, desde a criação de tabelas à exclusão de registros por exemplo.

Optou-se pelo isolamento dessa funcionalidade em uma classe específica, devido a ampla utilização em toda a aplicação, basicamente todas as outras classes e a aplicação fazem uso dessa classe para manipulação de transações com o banco de dados.

Outra necessidade exigida pelo *package* utilizado é que a compilação de todo o projeto seja feita em 32 bits, caso não esteja configurado essa opção não será possível a compilação e geração do executável.

A classe *Model* tem como principal objetivo armazenar o modelo das tabelas com seus respectivos atributos. A estrutura relacional desse banco será explicada futuramente em outro tópico (Figura 17).

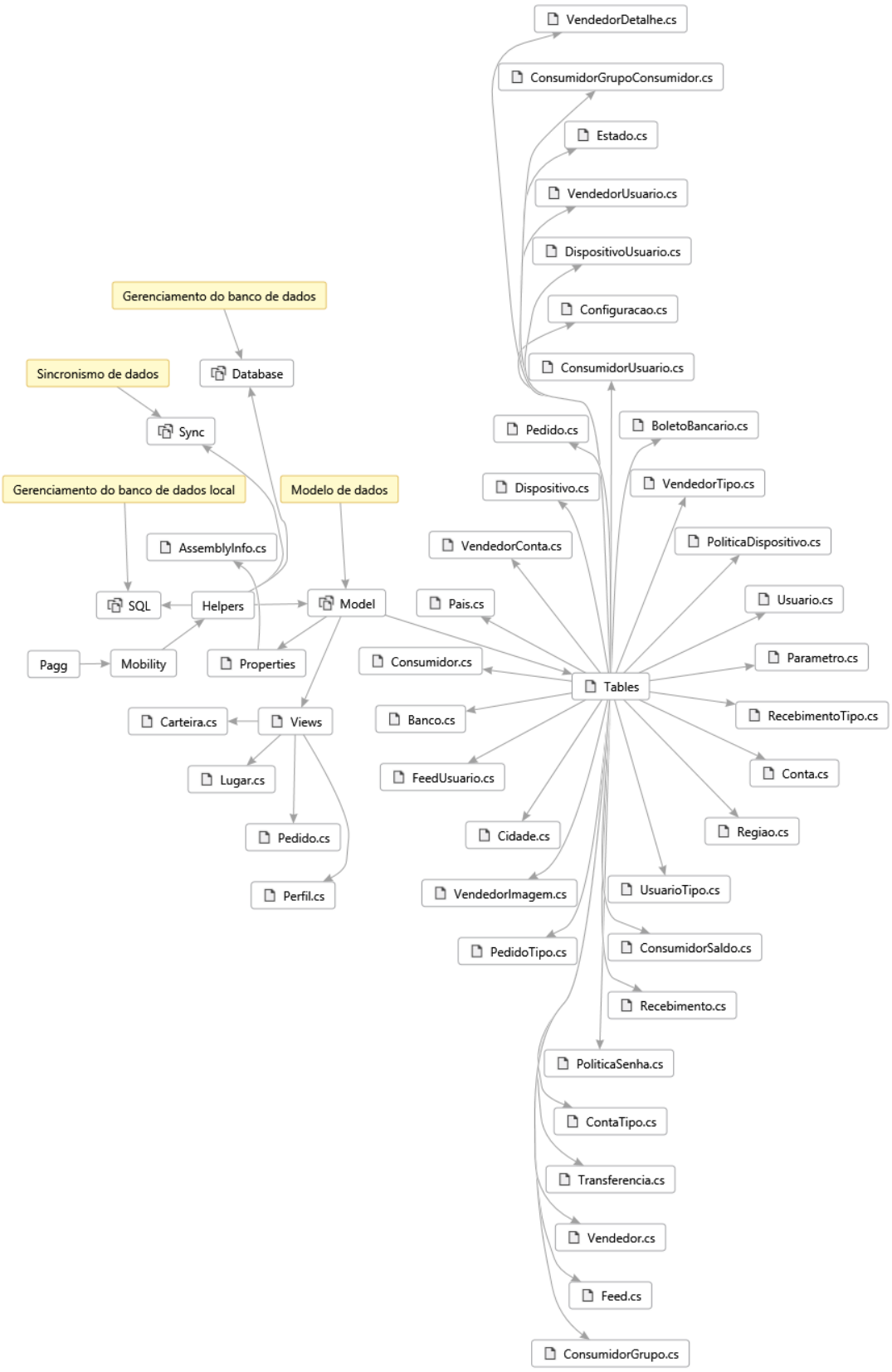


Figura 17. Diagrama de classe *Model* (Fonte Própria)

Com base nas subclasses definidas a classe *SQL* cria-se o banco de dados local respeitando todos os atributos definidos nessas classes. Demonstra-se na figura 18, um exemplo de classe que será utilizada para criação de uma tabela no banco de dados local, nela também consegue-se visualizar os atributos dessa tabela.

```
public sealed class Banco
{
    [PrimaryKey]
    0 references
    public int idBanco
    {
        get;
        set;
    }

    0 references
    public string CodigoBanco
    {
        get;
        set;
    }

    0 references
    public string Descricao
    {
        get;
        set;
    }

    0 references
    public bool Inativo
    {
        get;
        set;
    }
}
```

Figura 18. Classe em C# com os atributos utilizados na recuperação local de dados (Fonte Própria)

Além de utilizar a classe *Model* para criação de objetos no banco de dados local utiliza-se também para recuperação temporária de dados, ou seja, atribuísse um registro selecionado a classe e depois o recupera para um objeto utilizado na interface da aplicação. Observa-se isso na figura 19, onde há a atribuição dos resultados da consulta executada ao um objeto do tipo *Carteira.*

```
using (var db = new SQLiteConnection(Connection.ConnectionString()))
{
    StringBuilder query = new StringBuilder();
    query.AppendLine("SELECT DISTINCT");
    query.AppendLine("       Pedido.ukPedido,");
    query.AppendLine("       'Você esteve em ' || Vendedor.NomeFantasia AS Titulo,");
    query.AppendLine("       Pedido.Valor,");
    query.AppendLine("       Pedido.DataEmissao AS Data,");
    query.AppendLine("       VendedorImagem.Url AS Logomarca");
    query.AppendLine("FROM   Pedido");
    query.AppendLine("       INNER JOIN Vendedor        ON Pedido.idVendedor = Vendedor.idVendedor");
    query.AppendLine("       LEFT  JOIN VendedorImagem  ON Vendedor.idVendedor = VendedorImagem.idVendedor");
    query.AppendLine("                                     AND VendedorImagem.Tipo = 'Logomarca'");
    query.AppendLine("ORDER  BY Pedido.DataEmissao DESC");

    var wallet = db.Query<Model.Views.Carteira>(query.ToString());

    if (wallet != null)
    {
        foreach (var row in wallet)
        {
            row.Titulo = row.Titulo.ToUpper();
            row.Tempo = Tempo(row.Data);
        }

        ListBoxCarteira.ItemsSource = wallet;
    }
}
```

Figura 19. Método em C# para consulta e recuperação dos dados (Fonte Própria)

A classe *Sync* tem como principal objetivo a sincronização de dados entre a aplicação e o *web service*, ou seja, o envio e recebimento de dados. A sua estrutura consiste em duas subclasses denominadas *Receive*, responsável pelo recebimento e inserção dos dados no banco de dados local e *Send* responsável por enviar os dados gerados pela aplicação para processamento do *web service* (Figura 20).

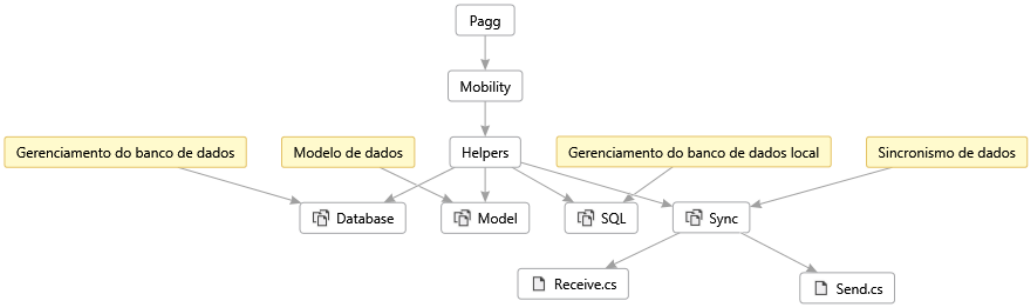


Figura 20. Diagrama de classe *Sync* (Fonte Própria)

O recebimento é o conjunto de métodos que enviam requisições ao *web service* solicitando os dados específicos para o usuário solicitante. O *web service* retornará a aplicação uma lista (*array)* com os dados a serem atualizados. Por esse motivo, essa classe necessita da classe *Model*. Na figura 21 demonstra um dos métodos que realizam a

sincronização dos dados e principalmente como é o tratamento aplicado para os dados recebidos.

```
private void VendedorTipo()
{
    ReceiveSoapClient dataSync = new ReceiveSoapClient();
    dataSync.VendedorTipoCompleted += dataSync_VendedorTipoCompleted;
    dataSync.VendedorTipoAsync();
}
```

Figura 21. Método em C# para requisição dos dados (Fonte Própria)

Em todos os métodos de requisição sempre haverá um método complementar responsável por tratar o retorno dado para a solicitação do *web service* (Figura 22).

```
private void dataSync_VendedorTipoCompleted(object sender, VendedorTipoCompletedEventArgs e)
{
    try
    {
        if (e.Result != null)
        {
            List<VendedorTipo> vendedortipo = e.Result.ToList();

            using (var db = new SQLiteConnection(Connection.ConnectionString()))
            {
                foreach (var item in vendedortipo)
                {
                    db.RunInTransaction(() =>
                    {
                        db.InsertOrReplace(item);
                    });
                }
            }
        }
    }
    catch
    {
        throw new Exception();
    }
}
```

Figura 22. Método em C# para armazenamento local dos dados (Fonte Própria)

Percebe-se que após o recebimento dos dados, que está armazenada na variável "*e*" do tipo *VendedorTipoCompletedEventArgs* por exemplo, o resultado é atribuído a uma lista do mesmo tipo. Depois da atribuição para uma lista, inicia-se o processo de inserção desses dados no banco local da aplicação. Devido ao método *InsertOrReplace* presente na classe para manipulação do banco de dados *SQLite*, conseguimos utilizar o conceito de atualização de dados incremental, ou seja, caso o registro com a chave definida já exista no banco de dados local ele atualiza caso contrário ele insere. Caso não exista o registro para a chave definida a aplicação delete o registro correspondente.

Quando ao processo de envio de dados para processamento do *web service*, criou-se um método para cada tipo de informação que precisa ser enviada, como cadastro de novos usuários, pagamentos, etc. conforme exemplifica a figura 23.

```
public void Usuario(String Login, String Senha)
{
    SendSoapClient dataSync = new SendSoapClient();
    dataSync.UsuarioCompleted += dataSync_UsuarioCompleted;
    dataSync.UsuarioAsync(Login, Senha);
}

1 reference
private void dataSync_UsuarioCompleted(object sender, UsuarioCompletedEventArgs e)
{
    Result.IsExecuted = e.Result;
}
```

Figura 23. Método em C# para envio de dados ao *web service* (Fonte Própria)

A classe *Database* tem como objetivo semelhante a classe *SQL*, ou seja, a manipulação do banco de dados local porém os métodos desenvolvidos nessa classe foram específicos para a aplicação (Figura 24).

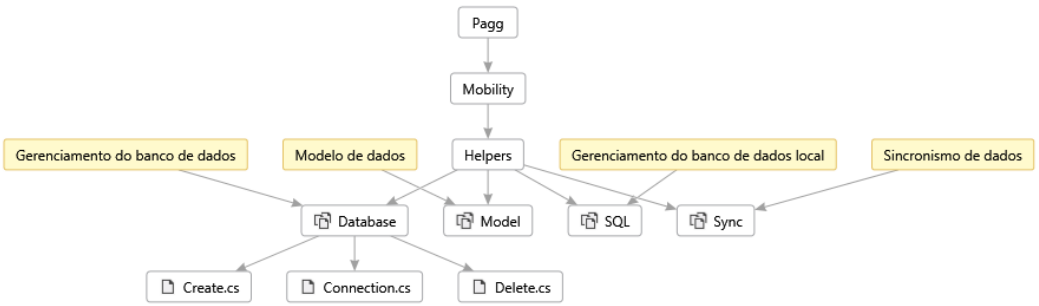


Figura 24. Diagrama de classe *Database* (Fonte Própria)

Nesse contexto, essa classe tem métodos de criação do banco de dados local, recuperação da *connection string* do banco de dados local. Na figura 25, é possível observar um exemplo da subclasse denominada *Create* responsável por criar o banco de dados local quando o mesmo não existe no dispositivo móvel do usuário.

```
private SQLiteConnection connection;

1 reference
public void All()
{
    connection = new SQLiteConnection(Connection.ConnectionString());

    connection.CreateTable<Banco>();
    connection.CreateTable<BoletoBancario>();
    connection.CreateTable<Cidade>();
    connection.CreateTable<Configuracao>();
    connection.CreateTable<Consumidor>();
    connection.CreateTable<ConsumidorGrupo>();
    connection.CreateTable<ConsumidorGrupoConsumidor>();
    connection.CreateTable<ConsumidorSaldo>();
    connection.CreateTable<ConsumidorUsuario>();
    connection.CreateTable<Conta>();
    connection.CreateTable<ContaTipo>();
    connection.CreateTable<Dispositivo>();
    connection.CreateTable<DispositivoUsuario>();
    connection.CreateTable<Estado>();
```

Figura 25. Método em C# para criação do banco de dados local (Fonte Própria)

3.1.2 Interface

Além dos recursos tecnológicos empregados, outra perspectiva considerada e de sua importância para o sucesso da aplicação foi a interface. Por tratar-se de uma tecnologia recente e que poucos usuários conhecem além de estarem seguros com os métodos de pagamentos atuais, quanto mais minimalista e fácil for a interface mais seguro e confiante o usuário se sente ao utilizar a aplicação.

O resgaste do conceito minimalista tem recebido o nome de design planificado, e seu uso pode ser notado tanto em aplicações móveis quanto em websites. Seu resgaste se faz necessário em uma época que a quantidade de informação gerada é inversamente proporcional a capacidade humana de absorve-la. É uma forma simples de quem produz o conteúdo, o sintetize e o simplifique em prol do usuário. O conceito ganhou força com o lançamento do sistema operacional *Windows 8* da *Microsoft*, por ser um software amplamente utilizado, logo sua interface passou a influenciar as aplicações. Isto ocorreu pois, para muitos, se tornou uma maneira de promover a facilidade e simplicidade por meio de uma interface minimalista (MANZOTTI, 2013).

O design planificado é uma técnica que não inclui atributos tridimensionais, efeitos como sombras, bordas e relevos, gradientes, ou outras ferramentas que possam transmitir um efeito e elevação ou criar profundidade, e que normalmente são utilizados para indicar ao usuário que ali existe uma interação. Seu visual tende a ser simples e de fácil utilização, fato que contribui para sua popularização, principalmente em interface de dispositivos móveis. A ausência dos elementos citados, em contrapartida exige mais

atenção do designer, pois ele deve compensar a falta destes elementos de outras maneiras. Seu foco deve se voltar ao uso de cores vivas, tipografia sem serifa e elementos geométricos simplificados (MANZOTTI, 2013).

A seguir apresenta-se algumas interfaces que foram desenhadas para a aplicação e que seguem o conceito de planificação como recurso para visar a objetividade e simplicidade na transmissão da informação. Sua apresentação segue o fluxo do que seria o primeiro contato do usuário com a aplicação. Partindo da abertura da aplicação, passando pelo cadastro, visualização da carteira, dos estabelecimentos que aceitam a aplicação como forma de pagamento, entre outros.

A tela de abertura é um artificio dos sistemas operacionais para indicar ao usuário que sua ação foi iniciada, e que a aplicação solicitada está sendo aberta. Sua duração é de poucos segundos e por isso a informação contida deve ser eficiente e simples (Figura 26).

Figura 26. Tela de abertura (Fonte Própria)

A tela de login é um dos primeiros contatos que o usuário tem com a aplicação, são solicitados apenas os campos de *login* e senha para acesso à aplicação (Figura 27).

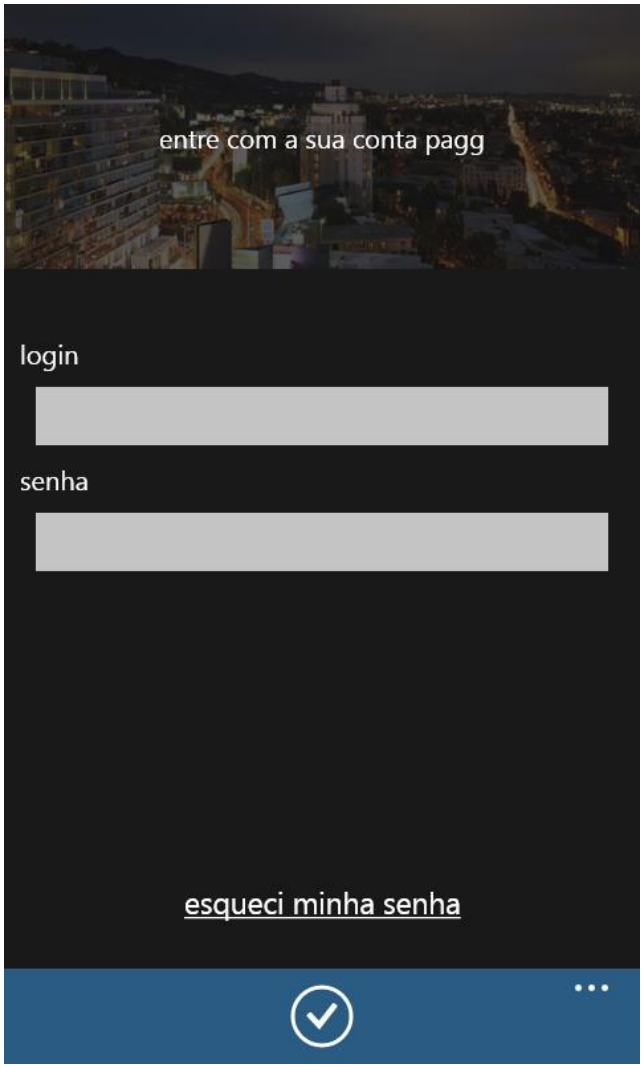


Figura 27. Tela de *login* (Fonte Própria)

A tela de inscrição, permite que novos usuários possam se cadastrar no próprio dispositivo sem a necessidade de acessarem algum *website* para se inscreverem. Logo após a finalização do cadastro, a aplicação solicitará ao usuário que já faça login (Figura 28).

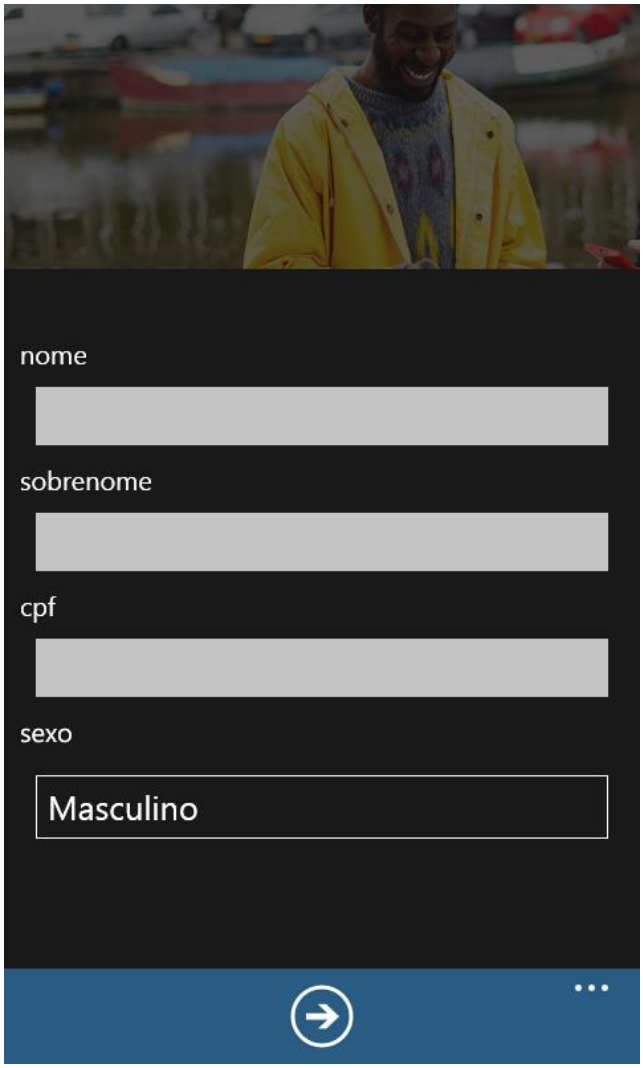


Figura 28. Tela de inscrição (Fonte Própria)

A tela principal, por meio dela é possível acessar todas as funções da aplicação. No menu lateral é possível acessar todas as funções disponíveis pela aplicação além de consultar dados do perfil do usuário e na área central é possível consultar o saldo disponível e realizar novos pagamentos (Figura 29).

Figura 29. Tela principal (Fonte Própria)

A tela de carteira permite ao usuário consultar os últimos pagamentos feitos, conforme demonstra-se na figura 30, com ênfase em três informações relevantes ao usuário: onde, quando e quanto.

Figura 30. Tela de carteira (Fonte Própria)

A tela de lugares mostra os estabelecimentos comerciais que aceitam esse meio de pagamento, ordenados pela proximidade que estão da posição geográfica (latitude e longitude) do usuário (Figura 31).

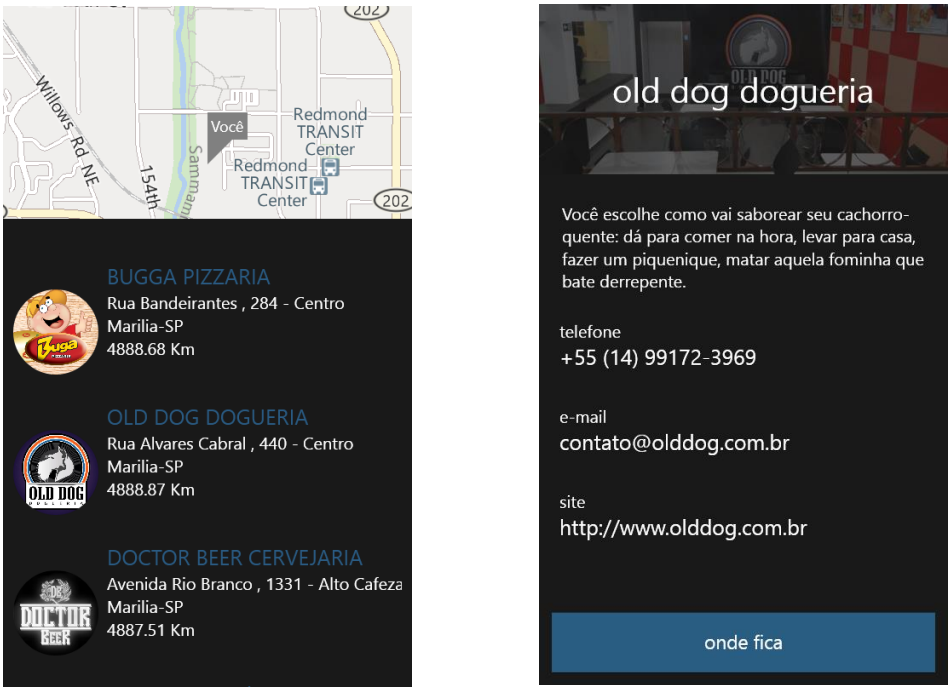


Figura 31. Tela de lugares (Fonte Própria)

Por fim, o pagamento é iniciado quando o usuário clica sobre o botão “novo pagamento” localizado na tela principal, logo após, será solicitado somente o valor a ser pago. Logo após o preenchimento do valor o usuário clicará no botão “pagar” iniciando a transmissão dos dados para outro dispositivo poder processá-lo (Figura 32).

Figura 32. Tela de pagamento (Fonte Própria)

3.1.3 Segurança

Quando fala-se em tecnologias que envolvem pagamento não tem como não associa-lo com segurança, ou seja, métodos que resguardam o usuário ao optar fazer pagamentos por meio da aplicação ou meio desenvolvido.

Verificou-se que há muitos métodos de segurança presentes no mercado, porém poucos voltados para pagamentos móveis. Outro ponto identificado foi que apenas um método de segurança não faz que aplicação seja totalmente segura podendo acarretar falhas ou brechas que possam ser alvos de ataques.

Tratou-se a segurança em dois aspectos distintos, um sobre a transmissão dos dados financeiros entre o dispositivo móvel do consumidor e do vendedor e outro sobre a garantia de autenticidade do usuário que está realizando o pagamento.

A transmissão dos dados financeiros entre os dispositivos por meio do NFC é feita em bytes, ou seja, utilizamos o método de serialização e desserialização de objetos presente na linguagem de programação *C#* o que descaracteriza a informação contida nessa sequência de bytes e dificulta o acesso à mesma.

Quanto a autenticidade do usuário, como mencionado anteriormente, verificou-se que apenas um método de segurança não é suficiente para coibir usuários que possam se passar por outros ao realizar pagamentos. Para isso, adotou-se dois meios de segurança sendo o primeiro a própria senha que o usuário utiliza para acessar a aplicação e a outra o reconhecimento facial do usuário. Para utilizar o reconhecimento facial utilizou-se o algoritmo *DSP* baseado no método de *Fast Fourier Transform*.

Esse método é uma ferramenta para processamento de dados que utiliza a decomposição da imagem nos seus componentes seno e cosseno. O resultado dessa transformação representa a imagem no domínio de *Fourier* ou de frequência, enquanto que a imagem de entrada é o domínio espacial equivalente. Na imagem de domínio *Fourier,* cada ponto representa uma frequência particular contida na imagem do domínio espacial. O algoritmo *DSP* utiliza o método mencionado para excluir as baixas frequências conforme exemplifica a figura 33 (NOKIA, 2014).

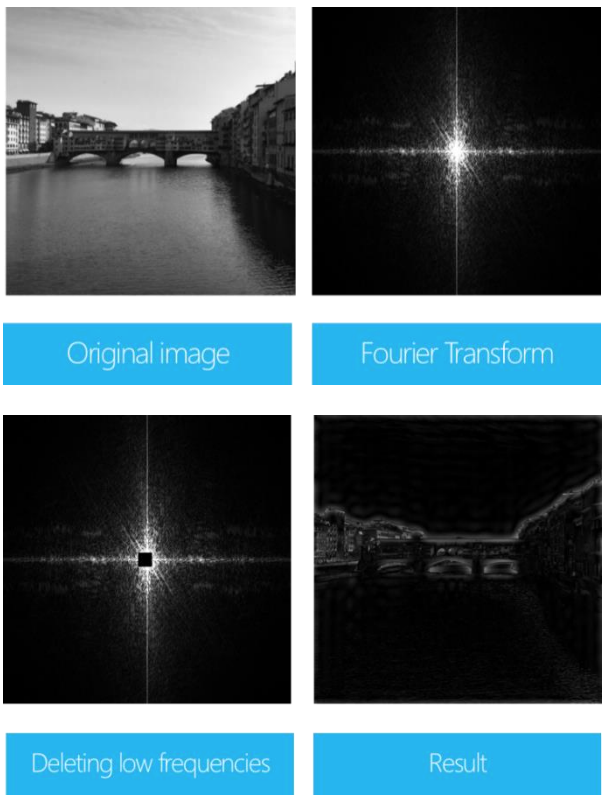


Figura 33. Processamento de imagem baseado no método de *Fourier* (Nokia)

Na figura 34 é possível observar o processo reverso, ou seja, de acordo com o método de *Fourier* realiza a exclusão das altas frequências para retornar a imagem original.

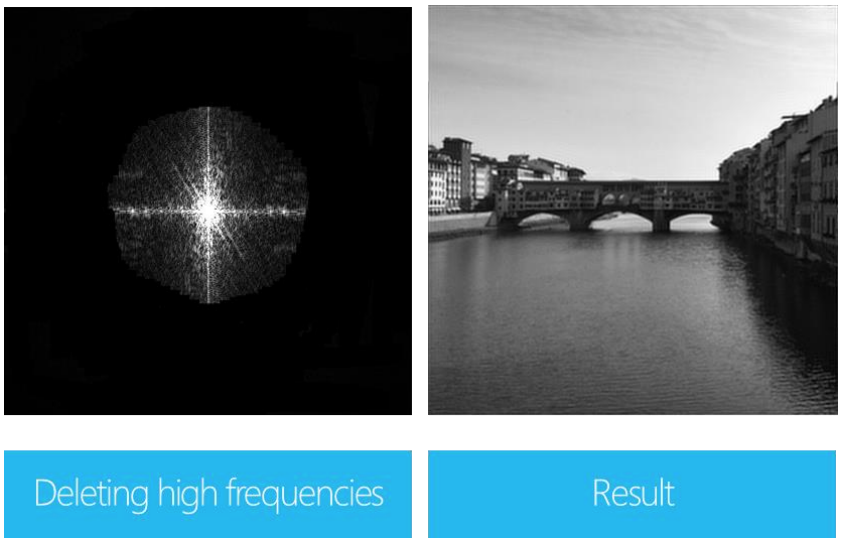


Figura 34. Processamento de imagem reverso com o uso do método de *Fourier* (Nokia)

O algoritmo *DSP* realiza as análises necessárias na imagem original e na imagem atual do usuário e retorna um valor inteiro do resultado dessa comparação. Caso esse valor seja menor que 55 as duas faces são iguais, ou seja, a mesma pessoa (Figura 35).

```
Deployment.Current.Dispatcher.BeginInvoke(delegate()
{
    capturedimage = PictureDecoder.DecodeJpeg(e.ImageStream, 256, 256);

    int[] pixel = capturedimage.Pixels;

    int color = 0;

    for (int i = 0; i < capturedimage.PixelHeight; i++)
    {
        for (int j = 0; j < capturedimage.PixelWidth; j++)
        {
            color = capturedimage.Pixels[j + (i * capturedimage.PixelWidth)];
            realIn[j + (i * capturedimage.PixelWidth)] = DSP.Utilities.ColorToGray(color) & 0xFF;
        }
    }

    Double[] match;
    double result = 0;

    DSP.FourierTransform.Compute2D((uint)256, (uint)256, ref realIn, null, ref realOut, ref imageOut, false);
    match = DSP.Utilities.triangularExtraction(ref realOut, (uint)256, (uint)256, (uint)25, 0);

    result = DSP.Utilities.MSE(ref compareSignal, ref match, (int)25);
    result /= 1000000000;
    result = Math.Round(result);

    if (result < 55)
    {
        MessageBox.Show("Você foi reconhecido! Agora podemos continuar com o pagamento.", "Ops!", MessageBoxButton.OK);
        NavigationService.Navigate(new Uri("/Views/Consumer/Pay.xaml", UriKind.Relative));
    }
    else
    {
        MessageBox.Show("Você não foi reconhecido! Por favor, tente novamente.", "Ops!", MessageBoxButton.OK);
    }
```

Figura 35. Método baseado no algoritmo DSP para verificação de imagens (Nokia)

Na figura 36, observa-se claramente os resultados obtidos na comparação de duas pessoas sejam elas iguais ou não e o resultado obtido com o algoritmo *DSP*.

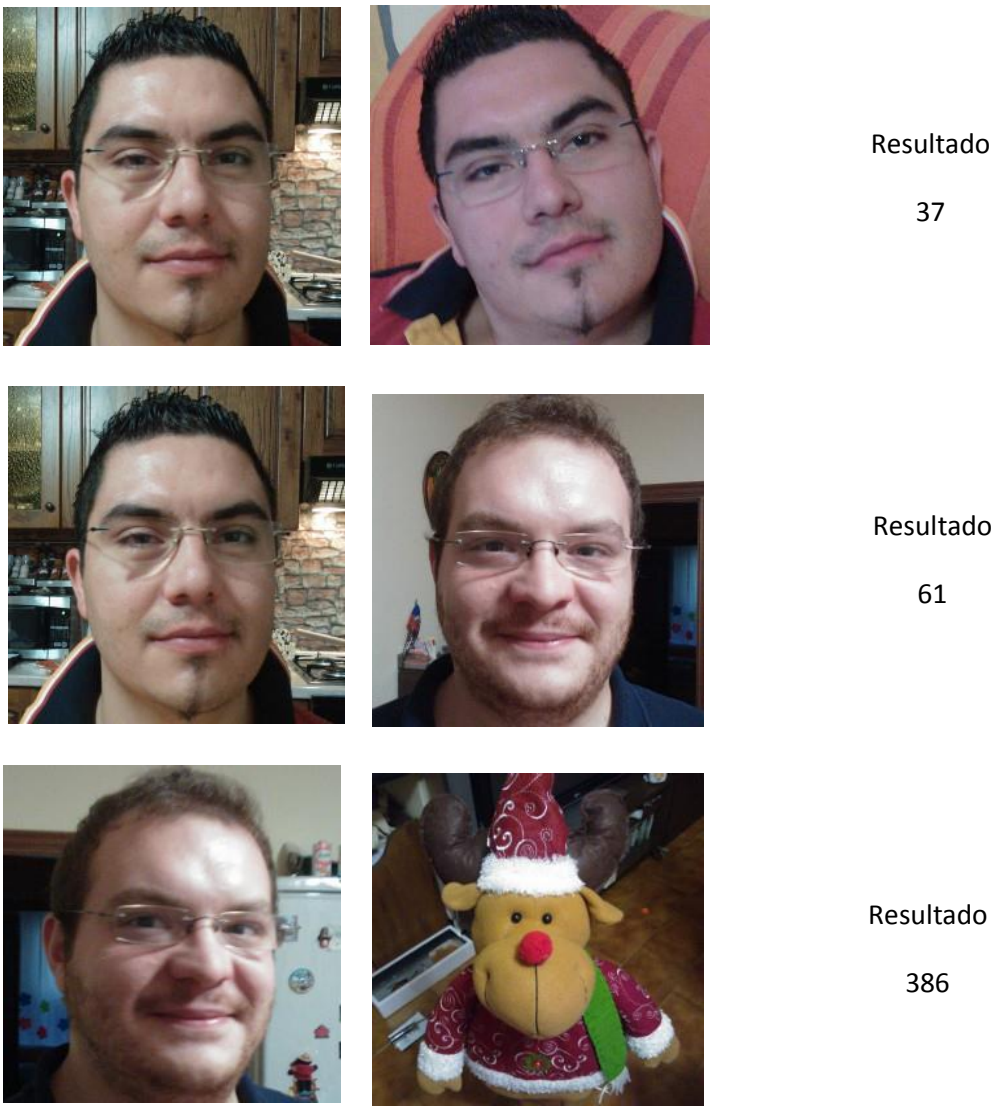


Figura 36. Exemplo dos resultados obtidos pelo algoritmo DSP (Nokia)

3.2 Aplicação *Back End*

A aplicação *back end* foi desenvolvida para gerenciar as solicitações feitas pela aplicação ao banco de dados. Utilizou-se a linguagem de programação *C#* para desenvolvimento desse serviço. Na figura 37, é possível observar a estrutura lógica da aplicação *back end* desenvolvida.

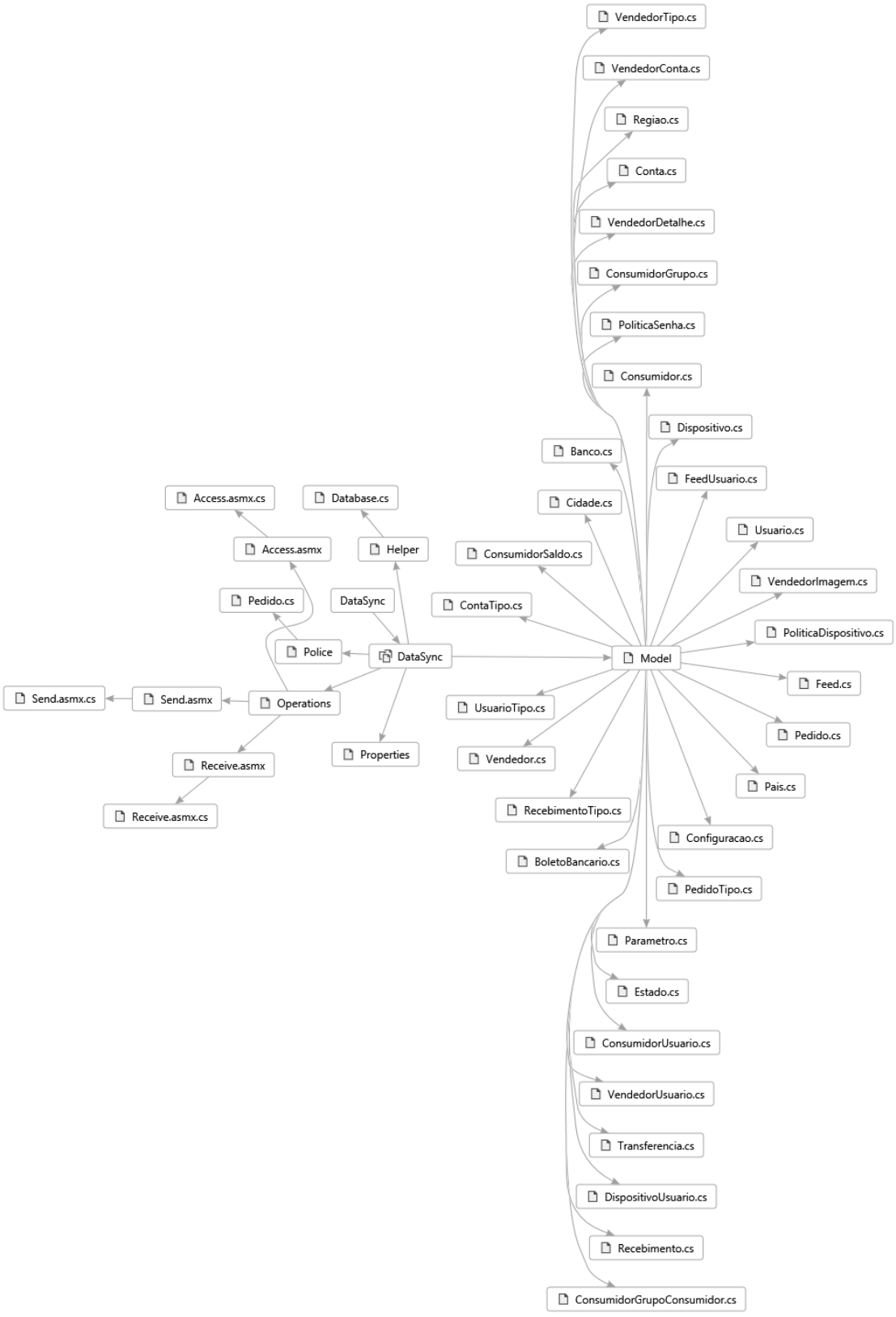


Figura 37. Diagrama de classe da aplicação *back end* (Fonte Própria)

Devido as necessidades exigidas pelo projeto, dividiu-se a aplicação *back end* constituído pelo *web service* em três grupos denominados respectivamente de *Receive*, *Send* e *Access*. O grupo *Receive* é composto pelos métodos para recepção dos dados, ou seja, cada método presente nesse grupo ao ser solicitado pela aplicação retorna uma lista estruturada com os dados pertinentes ao usuário. Classificou-se os métodos presentes nesse grupo em dois tipos, sendo os genéricos e específicos. Os métodos genéricos são métodos que retornam dados que podem ser utilizados por todos os usuários e os métodos específicos são métodos que retornar dados pertinentes ao usuário que fez a requisição (Figura 38).

```
[WebMethod]
0 references
public List<Banco> Banco()
{
    using (SqlConnection connection = new SqlConnection(Database.ConnectionStringToReceive()))
    {
        StringBuilder query = new StringBuilder();
        query.AppendLine("SELECT *");
        query.AppendLine("FROM   Banco");

        SqlCommand command = new SqlCommand(query.ToString(), connection);
        SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
        DataSet dataSet = new DataSet();
        dataAdapter.Fill(dataSet);

        List<Banco> banco = new List<Banco>();

        foreach (DataRow dataRow in dataSet.Tables[0].Rows)
        {
            banco.Add(new Banco
                {
                    idBanco = Convert.ToInt32(dataRow["idBanco"]),
                    CodigoBanco = Database.NullToString(dataRow["CodigoBanco"]),
                    Descricao = dataRow["Descricao"].ToString(),
                    Inativo = Convert.ToBoolean(dataRow["Inativo"])
                });
        }
        return banco;
    }
}
```

Figura 38. Método em C# para envio dos dados em XML (Fonte Própria)

Percebe-se que o método exemplificado na figura 38 realiza uma consulta no banco de dados e os retorna por meio do *SqlDataAdapter*, que nos fornece uma maneira de ler um fluxo unidirecional das linhas de um banco de dados *SQL Server*. Após esse procedimento, aloca-se os dados presentes no *SqlDataAdapter* em um *DataSet*, ou seja, os dados são alocados em cache de memória.

O *web service* também possui um conjunto de classes denominado *Model*, conforme demonstra a figura 39.

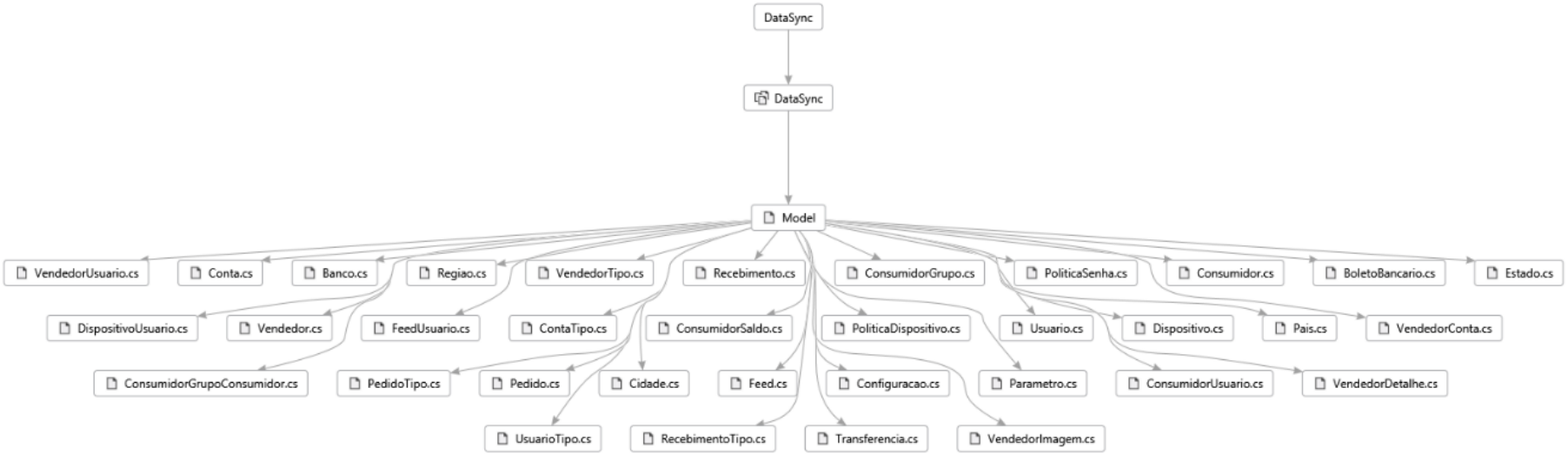


Figura 39. Diagrama de classe *Model* (Fonte Própria)

Esse conjunto de classes representam a estrutura do banco de dados e é responsável por atribuir e recuperar os dados quando necessário. Na figura 40 verifica-se a estrutura de uma das subclasses que compõe a *Model*.

```
public class Banco
{
    1 reference
    public int idBanco
    {
        get;
        set;
    }

    1 reference
    public string CodigoBanco
    {
        get;
        set;
    }

    1 reference
    public string Descricao
    {
        get;
        set;
    }

    1 reference
    public bool Inativo
    {
        get;
        set;
    }
}
```

Figura 40. Classe em C# com os atributos utilizados no envio de dados (Fonte Própria)

No momento que a aplicação realiza a requisição do método desejado, o mesmo retornará uma cadeia de dados estruturados no formato *XML* (*Extensible Markup Language*), conforme exemplifica a figura 41. O arquivo *XML* é um arquivo estruturado, ou seja, todos os dados nele apresentados são identificados por uma *tag*. Cada *tag* corresponde a um atributo pertencente a classe definida no conjunto de classes *Model*.

```
<?xml version="1.0" encoding="UTF-8"?>
- <ArrayOfBanco xmlns="http://tempuri.org/" xmlns:xsi="http://ww
  - <Banco>
      <idBanco>1</idBanco>
      <CodigoBanco>001</CodigoBanco>
      <Descricao>Banco do Brasil</Descricao>
      <Inativo>false</Inativo>
    </Banco>
  - <Banco>
      <idBanco>2</idBanco>
      <CodigoBanco>002</CodigoBanco>
      <Descricao>Banco do Bradesco</Descricao>
      <Inativo>false</Inativo>
    </Banco>
  - <Banco>
      <idBanco>3</idBanco>
      <CodigoBanco>003</CodigoBanco>
      <Descricao>Banco Itáu</Descricao>
      <Inativo>false</Inativo>
    </Banco>
  - <Banco>
      <idBanco>4</idBanco>
      <CodigoBanco>004</CodigoBanco>
      <Descricao>Banco Santander</Descricao>
      <Inativo>false</Inativo>
    </Banco>
  - <Banco>
      <idBanco>5</idBanco>
      <CodigoBanco>005</CodigoBanco>
      <Descricao>Banco HSBC</Descricao>
      <Inativo>false</Inativo>
    </Banco>
</ArrayOfBanco>
```

Figura 41. Exemplo de retorno de dados no formato XML (Fonte Própria)

O grupo *Send* é composto pelos métodos de envio dos dados, ou seja, cada método presente nesse grupo ao ser solicitado pela aplicação envia os dados correspondentes ao método solicitado e faz a inserção no banco de dados.

Na figura 42, observa-se a estrutura de um dos métodos para envio dos dados ao banco de dados.

```
[WebMethod]
0 references
public bool Usuario(String Login, String Senha)
{
    using (SqlConnection connection = new SqlConnection(Database.ConnectionStringToSend()))
    {
        try
        {
            StringBuilder query = new StringBuilder();
            query.AppendLine("INSERT INTO Usuario");
            query.AppendLine("(");
            query.AppendLine("  Login,");
            query.AppendLine("  Senha,");
            query.AppendLine("  DataInclusaoNoServidor");
            query.AppendLine(")");
            query.AppendLine("VALUES");
            query.AppendLine("(");
            query.AppendLine("  @Login,");
            query.AppendLine("  dbo.Encrypt(@Senha),");
            query.AppendLine("  @DataInclusaoNoServidor");
            query.AppendLine(")");

            SqlCommand command = new SqlCommand(query.ToString(), connection);
            command.Parameters.AddWithValue("@Login", Login);
            command.Parameters.AddWithValue("@Senha", Senha);
            command.Parameters.AddWithValue("@DataInclusaoNoServidor", DateTime.Now);
            connection.Open();

            return Convert.ToBoolean(command.ExecuteNonQuery());
        }
        catch
        {
            return false;
        }
        finally
        {
            connection.Close();
        }
    }
}
```

Figura 42. Método em C# para envio dos dados (Fonte Própria)

Todos os métodos de envio de dados retornam a aplicação se o comando foi executado corretamente ou não, permitindo assim o reenvio da informação para inserção no banco de dados.

Percebe-se, pela figura 42, que o método apenas insere os dados enviados por parâmetro no banco de dados e faz a execução do comando construído anteriormente atribuindo os valores as variáveis correspondentes.

O grupo *Access* é composto pelos métodos utilizados para gerenciamento do acesso à aplicação. É composto por apenas dois métodos, sendo o primeiro o de *Login* e o segundo *ChangePassword*.

O método de *Login*, conforme observa-se na figura 43, valida se o *login* e senha inseridos pelo usuário estão corretos. Além dessas duas validações, é validado também a data em que a solicitação de acesso está sendo feita, ou seja, caso a data esteja divergente com os servidores do *web service* o usuário não conseguirá acessar a aplicação.

```
[WebMethod]
0 references
public bool Login(String Login, String Senha, DateTime DataSincronismo)
{
    if (DataSincronismo != DateTime.Today)
    {
        return false;
    }
    else
    {
        using (SqlConnection connection = new SqlConnection(Database.ConnectionStringToReceive()))
        {
            StringBuilder query = new StringBuilder();
            query.AppendLine("SELECT 1");
            query.AppendLine("FROM   Usuario");
            query.AppendLine("WHERE  Usuario.Login = @Login");
            query.AppendLine("       AND dbo.Decrypt(Usuario.Senha) = @Senha");
            query.AppendLine("       AND Usuario.Bloqueado = 0");
            query.AppendLine("       AND Usuario.Inativo = 0");

            SqlCommand command = new SqlCommand(query.ToString(), connection);
            command.Parameters.AddWithValue("@Login", Login);
            command.Parameters.AddWithValue("@Senha", Senha);
            connection.Open();

            return Convert.ToBoolean(command.ExecuteScalar());
        }
    }
}
```

Figura 43. Método em C# para validação do *login* (Fonte Própria)

O processo de manipulação das senhas, utiliza uma função disponível no banco de dados principal que é responsável por encriptar a senha inserida pelo usuário, conforme pode-se observar na figura 44.

```
CREATE FUNCTION [dbo].[Encrypt]
(
    @Value varchar(50)
)
RETURNS varbinary(50)
AS
BEGIN
    DECLARE @EncryptValue varbinary(50) = (ENCRYPTBYPASSPHRASE('p@G$%&*',@Value));
    RETURN (@EncryptValue)
END
```

Figura 44. Função em T-SQL para encriptação da senha dos usuários (Fonte Própria)

O método *ChangePassword* permite aos usuários que alterem a senha caso a esqueçam ou caso a mesma expire (Figura 45).

```
[WebMethod]
0 references
public bool ChangePassword(String Login, String Senha, DateTime Data)
{
    if (Data != DateTime.Now)
    {
        return false;
    }
    else
    {
        using (SqlConnection connection = new SqlConnection(Database.ConnectionStringToReceive()))
        {
            StringBuilder query = new StringBuilder();
            query.AppendLine("UPDATE Usuario");
            query.AppendLine("   SET Senha = dbo.Encrypt(@Senha),");
            query.AppendLine("       DataUltimaTrocaSenha = GETDATE()");
            query.AppendLine(" WHERE Login = @Login");

            SqlCommand command = new SqlCommand(query.ToString(), connection);
            command.Parameters.AddWithValue("@Login", Login);
            command.Parameters.AddWithValue("@Senha", Senha);
            connection.Open();

            return Convert.ToBoolean(command.ExecuteScalar());
        }
    }
}
```

Figura 45. Método em C# para alteração da senha dos usuários (Fonte Própria)

3.3 Banco de dados

O banco de dados foi desenvolvido utilizando o *Microsoft SQL Server* desenvolvido pela *Microsoft* e que está na versão 2014. O *Microsoft SQL Server* é um banco de dados transacional para aplicações robustas, possui diversos recursos para gerenciamento e armazenamento dos dados.

Durante o desenvolvimento da aplicação, percebeu-se um *delay* em algumas requisições da aplicação ao *web service* para envio dos dados, ou seja, a requisição feita pela aplicação não chegava no *web service* e consequentemente não era inserida no banco de dados. Para solucionar isso, criou-se um banco de dados temporário sem integridade referencial para armazenar temporariamente os dados enviados pela aplicação. Na figura 46, consegue-se analisar com mais detalhes a estrutura desse banco de dados temporário. Eliminando os relacionamentos entre as tabelas, todos as requisições de envio de dados feitas pela aplicação foram inseridas no banco de dados temporário.

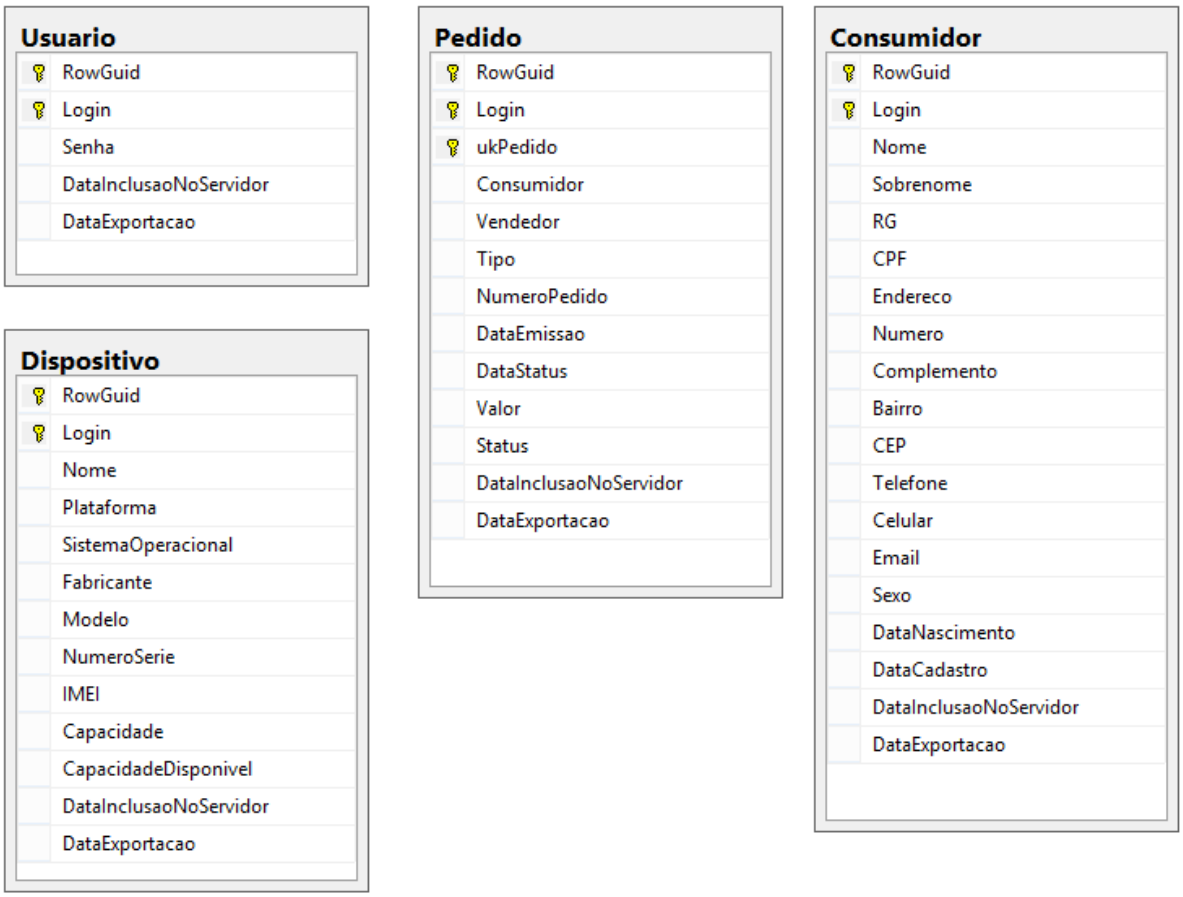


Figura 46. Diagrama Entidade Relacionamento do banco de dados temporário (Fonte Própria)

As chaves primárias de todas as tabelas desse banco são compostas pelo *RowGuid* e o *login* do usuário proprietário dessa informação. O *RowGuid* tem a função de índice na tabela, ou seja, deixar as leituras nessas tabelas um pouco mais rápidas e o *login*, conforme já dito, permite saber o proprietário dessa informação. Na figura 47 observa-se o que foi dito anteriormente.

RowGuid	Login	Senha	DataInclusaoNoServidor	DataExportacao
4857ED83-E85C-457E-9FC6-2288F4C9E73F	juliana.vasconcellos	0x01000000B090451C4BC3AABAC50FB2B78DDE061BE4A6E7...	2014-10-12 17:15:47.460	2014-10-12 17:16:00.470
60CD19D9-3AEB-41E2-97B9-5742D509BA9B	guilherme.macedo	0x01000000264C8894BB01782C87421171C80BCC53CDE77C1...	2014-10-05 15:12:53.487	2014-10-05 15:17:52.537
B4E606A0-CC66-47E1-9055-5BCECC70C6F0	marcelo.macedo	0x010000004E537D97B12BA425814C5B95527D79E602DAFA7...	2014-10-12 16:36:50.720	2014-10-12 16:38:00.373
2DBC540E-F21E-4863-8DFC-5DD604740D2A	eduardo.gomes	0x01000000DE1143B92D114C2D4752C166A09761848623E932...	2014-10-12 17:12:27.600	2014-10-12 17:14:00.457
1F0E7A3B-9D82-4133-B344-805E25E1E301	giulianna.marques	0x010000006E241F57B8B996A1C9B8C4983FB586B7244F53A3...	2014-10-12 16:55:25.423	2014-10-12 16:56:00.420
04F367C1-6A26-4016-9144-87B23337DBBC	marcia.macedo	0x010000006A9E1AE9E9C67CBE87E8E8D9611F2816F85A45A...	2014-10-12 16:32:51.250	2014-10-12 16:34:00.377
86263BA6-2957-4958-9C3B-8C5CAC1758DA	caio.macedo	0x010000004D4E9171E4534F393DA13745EB9FB5D3FD14BDF...	2014-10-07 22:42:24.067	2014-10-07 22:54:00.747
980F62A1-D422-4796-B638-8C846FE13395	benedicta.pires	0x01000000507BD20B9E91A78962A0A0021F3CF2B98FDBC62...	2014-10-12 16:52:00.893	2014-10-12 16:54:00.427
34B62DF5-B6DD-42C0-99A2-BDB16DB8D1F8	carla.novaes	0x01000000BEF8BEBA66C2EB1E7F43CA0C906248F55A56CFB...	2014-10-12 17:25:11.257	2014-10-12 17:26:00.490
730FB57C-7E7E-4AFB-9CAB-BF22BB1AA1A0	julia.silva	0x0100000019EFAA628956AE830FE021F6C77D6C4E30B78204...	2014-10-12 19:06:21.350	2014-10-12 19:08:00.647
348CC375-BE09-4849-AE47-D5BD357C86FB	lucas.rocha	0x0100000020841EDF220DFD5A449684CD84DCAA1A55AA81...	2014-10-12 16:58:41.487	2014-10-12 17:00:00.440

Figura 47. Tabela de usuários do banco de dados temporário (Fonte Própria)

O banco de dados temporário possui integração com o banco de dados principal da aplicação, para isso utilizou-se a tecnologia presente do *Microsoft SQL Server 2014* denominada *Merge*. A instrução *Merge* permite unir uma fonte de dados com uma exibição ou tabela de destino e, em seguida, executar várias ações no destino com base nos resultados dessa união, ou seja, consegue-se executar as operações de inserção, atualização e exclusão usando essa instrução presente do *Microsoft SQL Server* desde a versão 2008. Pode-se utilizar essa instrução para executar as seguintes operações (MICROSOFT, 2014):

- Inserir ou atualizar linhas em uma única tabela de destino condicionalmente. Se a linha existir na tabela de destino, atualiza uma ou mais colunas, caso contrário, insere os dados em uma nova linha,
- Sincronizar duas tabelas: insere, atualiza ou exclui linhas de uma tabela de destino com base nas diferenças em relação aos dados de origem.

Precisa-se compreender como os dados de origem e de destino são mesclados em um único fluxo de entrada, e como critérios adicionais de pesquisa podem ser usados para filtrar corretamente as linhas não necessárias. Caso contrário, poderá especificar critérios adicionais de pesquisa de uma maneira que produza dados incorretos. As linhas de origem são correspondidas com as linhas de destino com base no predicado de junção especificado na cláusula *ON*. O resultado é um fluxo de entrada combinado. Uma operação de inserção, atualização ou exclusão é executada na linha de entrada. Dependendo da cláusula *WHEN* especificada na instrução, a linha de entrada poderá qualquer uma das seguintes (MICROSOFT, 2014):

- Um par correspondente consistindo em uma linha do destino em uma linha da origem. Esse é o resultado da cláusula *WHEN MATCHED*.
- Uma linha da origem que não tem nenhuma linha correspondente no destino. Esse é o resultado da cláusula *WHEN NOT MATCHED BY TARGET*.
- Uma linha do destino que não tem nenhuma linha correspondente na origem. Esse é o resultado da cláusula *WHEN NOT MATCHED BY SOURCE*.

A cláusulas *WHEN*, presente na instrução *Merge,* especificam as ações a serem tomadas com base dos resultados da cláusula *ON* e em quaisquer critérios adicionais de pesquisa especificados nas cláusulas *WHEN*. Em muitos casos, as condições de pesquisa especificadas na cláusula *ON* produzem o mesmo fluxo de entrada necessário, conforme observa-se na figura 48 um exemplo de parte da *procedure* utilizada para integrar os dados de novos usuários cadastrados para uso da aplicação entre o banco de dados temporário e banco de dados principal (MICROSOFT, 2014).

```
MERGE INTO TargetTable
USING (SELECT * FROM @Usuario) AS SourceTable
ON
(
    SourceTable.Login = TargetTable.Login
)

WHEN NOT MATCHED BY SOURCE
     THEN DELETE

WHEN NOT MATCHED BY TARGET
     THEN INSERT
     (
        idUsuarioTipo,
        Login,
        Senha
     )
     VALUES
     (
        SourceTable.idUsuarioTipo,
        SourceTable.Login,
        SourceTable.Senha
     )

WHEN MATCHED AND
    (
        (TargetTable.idUsuarioTipo != SourceTable.idUsuarioTipo) OR
        (TargetTable.idUsuarioTipo IS NULL AND  SourceTable.idUsuarioTipo IS NOT NULL) OR
        (TargetTable.idUsuarioTipo IS NOT NULL AND  SourceTable.idUsuarioTipo IS NULL)
    )
    OR
    (
        (TargetTable.Senha != SourceTable.Senha) OR
        (TargetTable.Senha IS NULL AND  SourceTable.Senha IS NOT NULL) OR
        (TargetTable.Senha IS NOT NULL AND  SourceTable.Senha IS NULL)
    )
THEN UPDATE SET
     TargetTable.idUsuarioTipo = SourceTable.idUsuarioTipo,
     TargetTable.Senha = SourceTable.Senha;
```

Figura 48. Exemplo de instrução *merge* para integração de dados (Fonte Própria)

O banco de dados principal é responsável por armazenar todos os dados necessários para funcionamento da aplicação, desde os usuários aos estabelecimentos que aceitam esse meio de pagamento. Na figura 49, consegue-se observar a estrutura desse banco de dados e os relacionamentos entre as tabelas criadas.

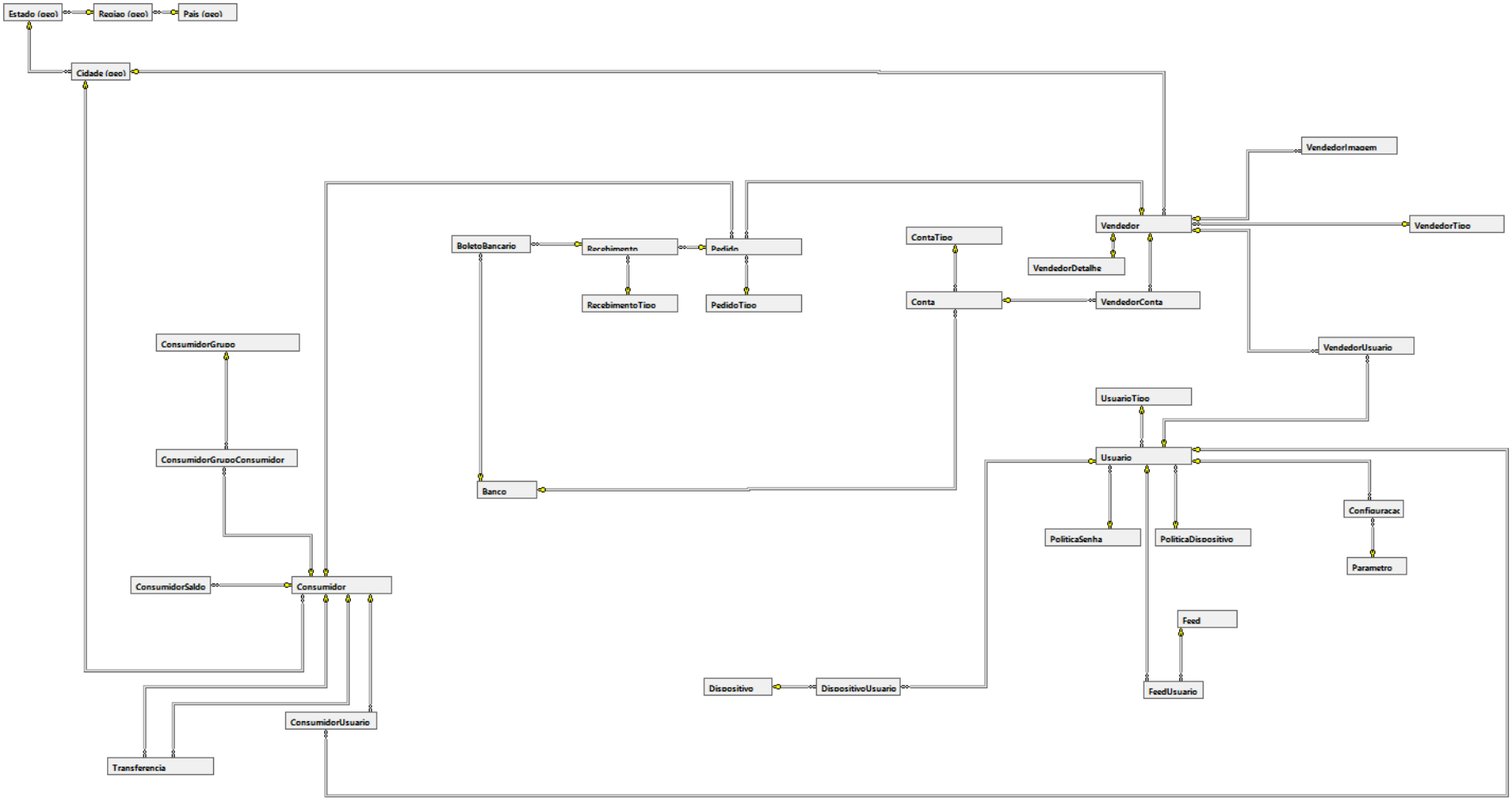


Figura 49. Diagrama Entidade Relacionamento do banco de dados principal (Fonte Própria)

No “Apêndice A – Diagrama Entidade Relacionamento”, é possível analisar com mais detalhes a estrutura de cada tabela e seus relacionamentos que compõe o banco de dados principal.

CONCLUSÕES

O presente trabalho visou o desenvolvimento de uma solução alternativa para pagamentos e a democratização dos serviços bancários.

Tal objetivo foi atingido com o desenvolvimento do protótipo de uma aplicação móvel utilizando a tecnologia NFC para pagamentos por aproximação de dois dispositivos, sendo um o emissor e o outro receptor da solicitação de pagamento.

Conclui-se que apesar do *mobile payment* ser uma tecnologia promissora ainda apresenta alguns desafios, como a simplificação e padronização dos processos relacionados ao pagamento, ou seja, diferentemente do cartão de crédito que há somente uma única forma de utilizar as aplicações dedicadas ao *mobile payment* possui infinitas possibilidades de realizar um pagamento o que pode dificultar a adoção em massa dessa tecnologia pelos usuários finais.

Conclui-se que não há métodos de segurança desenvolvidos que atendam às necessidades exigidas pelo mobile *payment*, o que também se torna um empecilho para o crescimento dessa tecnologia.

Conclui-se que apesar disso, o *mobile payment* proporcionará ganhos notáveis para sociedade, seja na simplificação do processo de pagamento contribuindo com ganho de produtividade, mobilidade além de deixar a relação entre o consumidor e o vendedor mais dinâmica e estreita.

Para trabalhos futuros, sugere-se o estudo de métodos de segurança que possam ser aplicados ao *mobile payment* permitindo maior credibilidade a essa tecnologia tão promissora. Além disso, sugere-se também o estudo de outros tipos de *mobile payment* buscando a padronização dos processos de pagamento feitos por dispositivos móveis.

REFERÊNCIAS

AKHTER, S.; DALY, K. Finance and poverty: evidence from fixed effect vector decomposition. Emerging Markets Review, v. 10 n. 3, p. 191-206, Sept. 2009.

Banco Central do Brasil, Disponível em www.bcb.gov.br/pre/normativos/busca/normativo.asp?tipo=res&ano=2013&numero=4282, Acessado em 13 de Maio de 2.014.

Banco Central do Brasil, Disponível em www.bcb.gov.br/pre/normativos/busca/normativo.asp?tipo=res&ano=2013&numero=4283 , Acessado em 13 de Maio de 2.014.

Begonha, D.B, Hoffman, A., and Melin, P.M-payments: Hang up, try again. Credit Card Management 15, 10 (Dec. 2002), 40-42.

EXCHANGE I.M The State of Financial Inclusion in Ivory Coast in the Aftermath of the Crisis. Disponível em http://www.themix.org/publications/mix-microfinance-world/2013/12/state-financial-inclusion-ivory-coast-aftermath-crisis. Acessado em 24 de Novembro de 2.014.

Face Recognition using 2D Fast Fourier Transform, Disponível em http://developer.nokia.com/community/wiki/Face_Recognition_using_2D_Fast_Fourier_Transform, Acessado em 25 de Outubro de 2.014.

FEBRABAN, Disponível em http://www.febraban.org.br/7Rof7SWg6qmyvwJcFwF7I0aSDf9jyV/sitefebraban/RPSP-6021-14%20FEBRABAN_Pesquisa%20Tecnologia%20Banc%E1ria_2013%207.5.2014_vf.pdf. Acessado em 08 de Dezembro de 2.014.

FILHO F. S. L. O. e CALVACANTE V. S Comunicação NFC (*Near Field Communication)* entre Dispositivos Ativos, UFPE, 2010.

FISHER, J. NFC in cell phones: The new paradigm for an interactive word [Near Field Communications], Communications Magazine, *IEEE*, p. 22.

GFT do Brasil: Pesquisa sobre hábitos dos consumidores em relação à aplicações móveis financeiras, Disponível em www.gft.com/br/pt/index/empresa/imprensa/Informacoes_para_a_imprensa/2013/estudo_gft_os_celulares_nao_podem_substituir_sucursais_bancarias.html, Acessado em 21 de Maio de 2.014.

GSMA Team, GSMA Annual Report: The Mobile Economy 2014, 2014.

Microsoft Corporation and M-Com, Write Paper: Mobile Payments: Delivering Compelling Customer and Shareholder Value through a Complete, Coherent Approach (Out. 2013)

Herzberg, A. Payments and baking with mobile personal devices. Commun. ACM 46, 5 (Maio 2003), 53-28.

LEVINE, R. Finance and growth: theory and evidence. Handbook of Economic Growth, Amsterdam, 1. ed., v. 1, p. 865-934, 2005.

MANZOTTI S. C. Design de Interface em Dispositivos Móveis, 2013, p. 87-116. Disponível em http://www.ensaiovisual.com.br/media/caio_manzotti_design_interface.pdf, Acessado em 10 de Novembro de 2.014.

Microsoft: Namespace System.Runtime.Serialization, Disponível em http://msdn.microsoft.com/pt-br/library/system.runtime.serialization%28v=vs.110%29.aspx, Acessado em 03 de Novembro de 2.014.

Mobile Electronic Transactions Ltd. MeT White Paper on Mobile Transactions (2003); www.mobiletransaction.org.

MOOKERJEE, R.; KALIPIONI, P. Availability of financial services and income inequality: the evidence from many countries Emerging Markets Review, v. 11, n. 4, p. 404-408, Dec. 2010.

MSDN: Inserindo, atualizando e excluindo dados usando MERGE, Disponível em http://technet.microsoft.com/pt-br/library/bb522522%28v=sql.105%29.aspx, Acessado em 06 de Novembro de 2.014.

PÉNICAUD C., KATAKAM A. GSMA Mobile Money for the Unbanked: State of the Industry 2013, 2012.

Smart Card Alliance: The Mobile Payment and NFC Landscape: A U.S Perspective (Set. 2011).

Schiavinatto F. (Organizador) Sistema de indicadores de percepção social (SIPS), 1ª Edição, 2011, Cap. 8, p. 181-210.

SQLite Documentation, Disponível em http://www.sqlite.org/about.html, Acessado em 03 de Novembro de 2.014.

Valor Econômico: Aplicativos são nova arma para manter clientes, Disponível em www.valor.com.br/empresas/3513296/aplicativos-sao-nova-arma-dos-bancos-para-manter-clientes , Acessado em 27 de Maio de 2.014.

APÊNDICE A – DIGRAMAS DE ENTIDADE E RELACIONAMENTO

Esse apêndice contém todos os diagramas de entidade e relacionamento do banco de dados principal da aplicação.

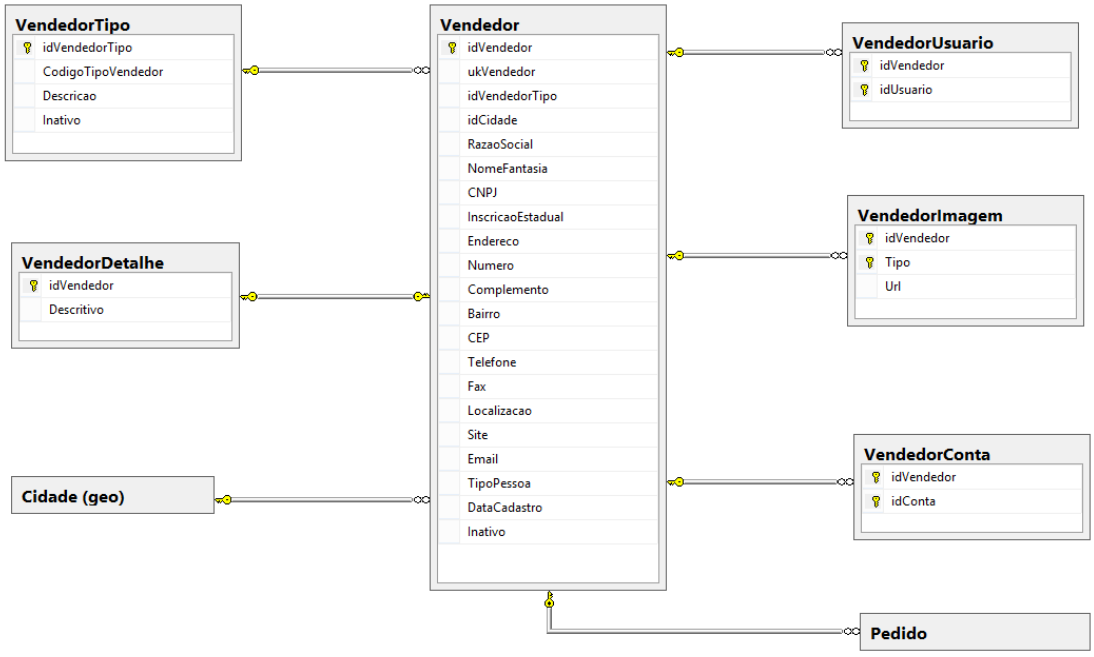


Figura 1A. Diagrama Entidade Relacionamento "Vendedor" (Fonte Própria)

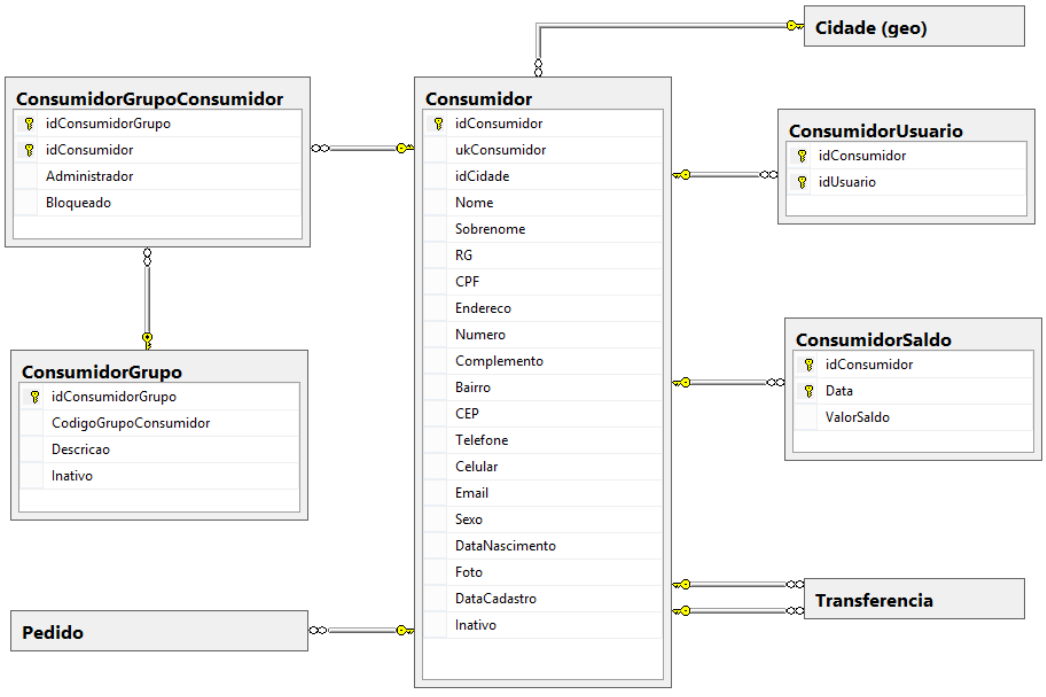


Figura 2A. Diagrama Entidade Relacionamento "Consumidor" (Fonte Própria)

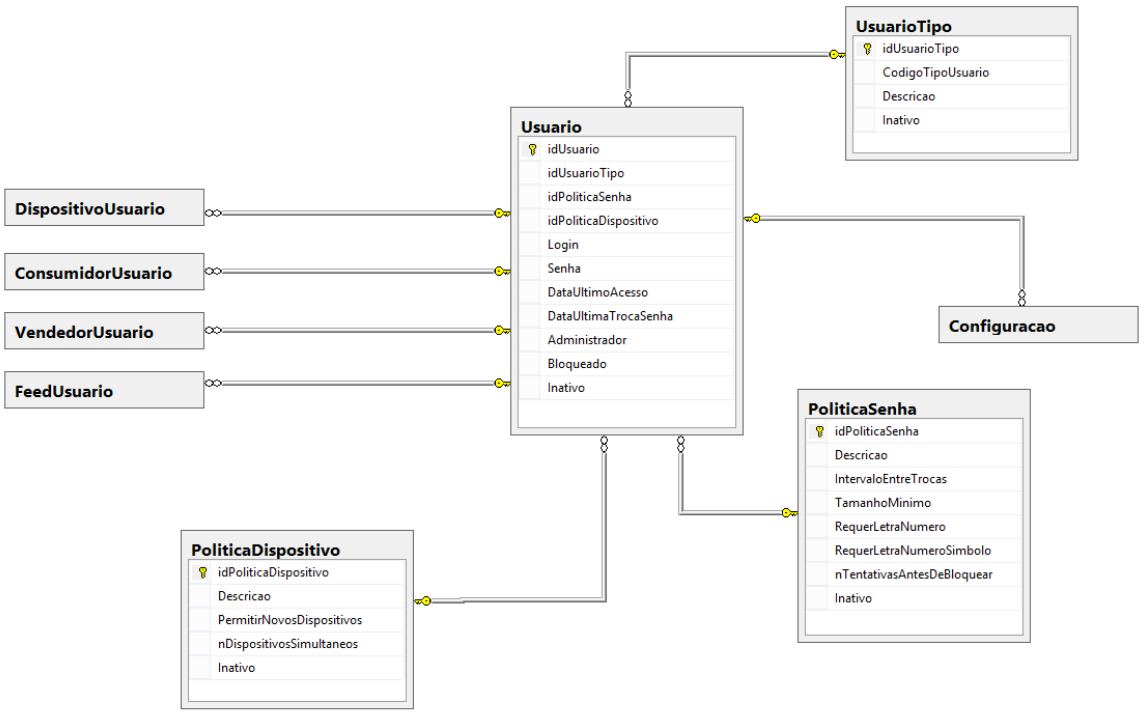


Figura 3A. Diagrama Entidade Relacionamento "Usuário" (Fonte Própria)

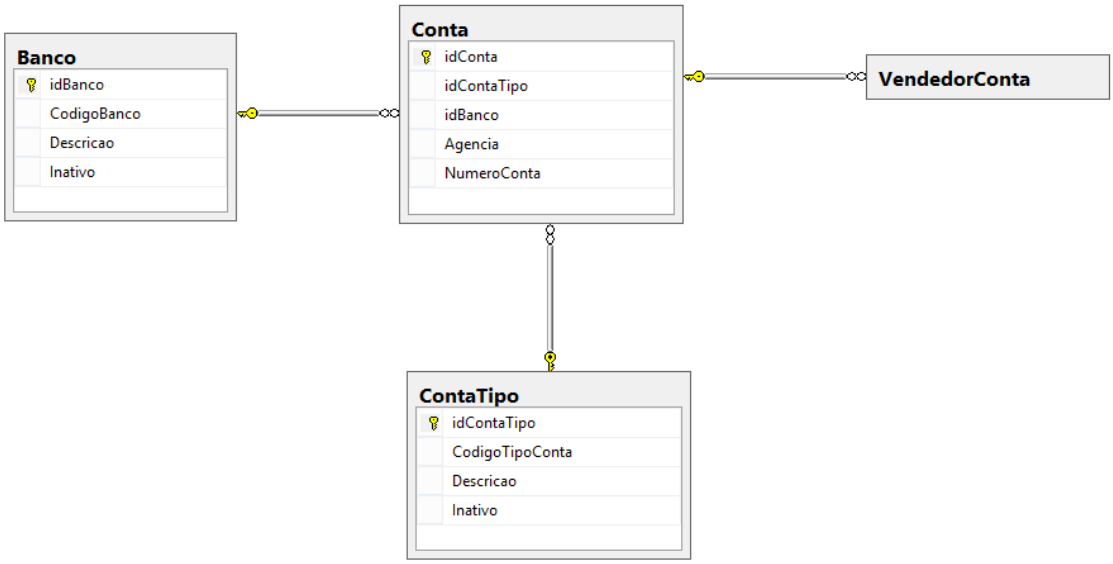


Figura 4A. Diagrama Entidade Relacionamento "Conta" (Fonte Própria)

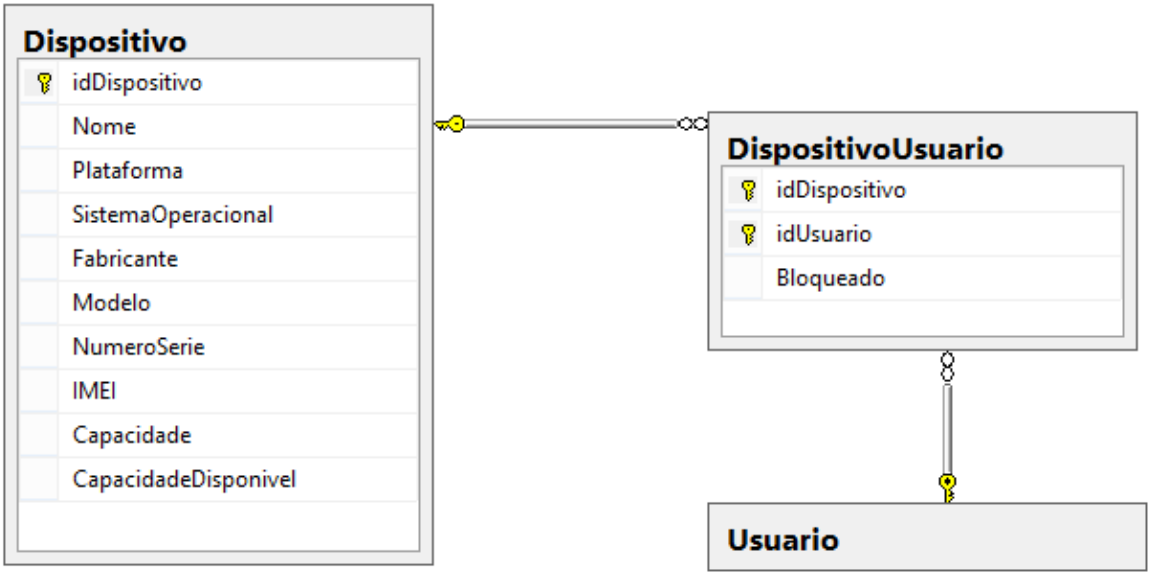


Figura 5A. Diagrama Entidade Relacionamento "Dispositivo" (Fonte Própria)

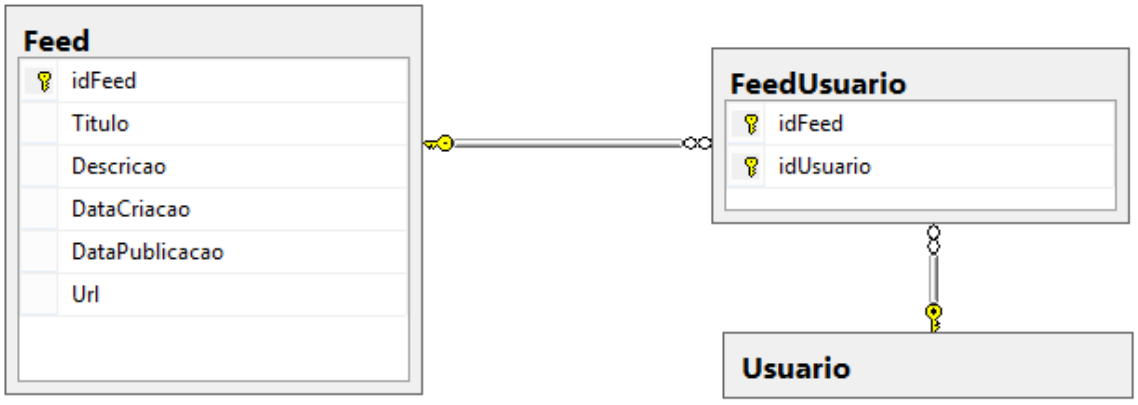


Figura 6A. Diagrama Entidade Relacionamento "Feed" (Fonte Própria)

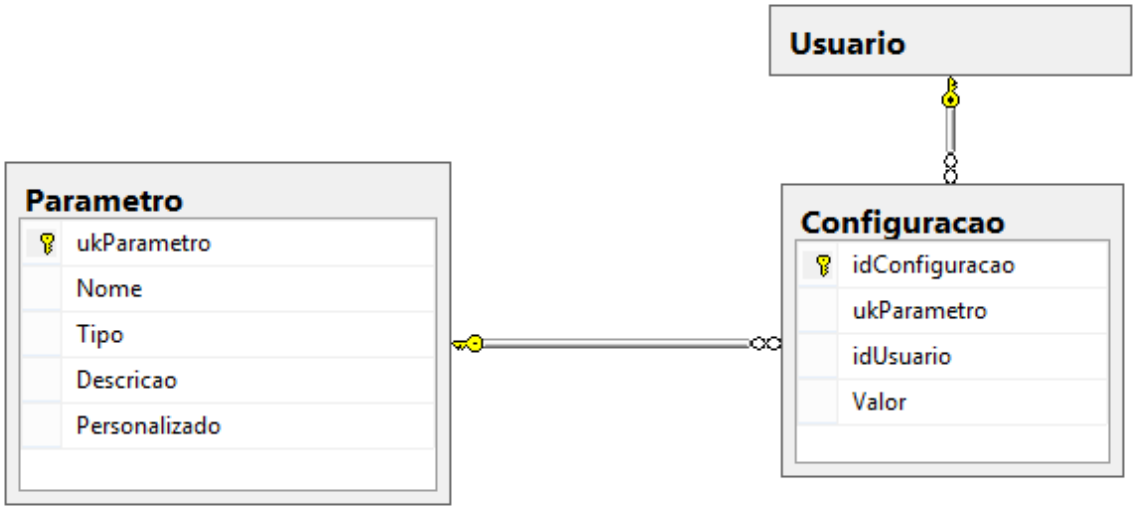


Figura 7A. Diagrama Entidade Relacionamento “Configuração” (Fonte Própria)

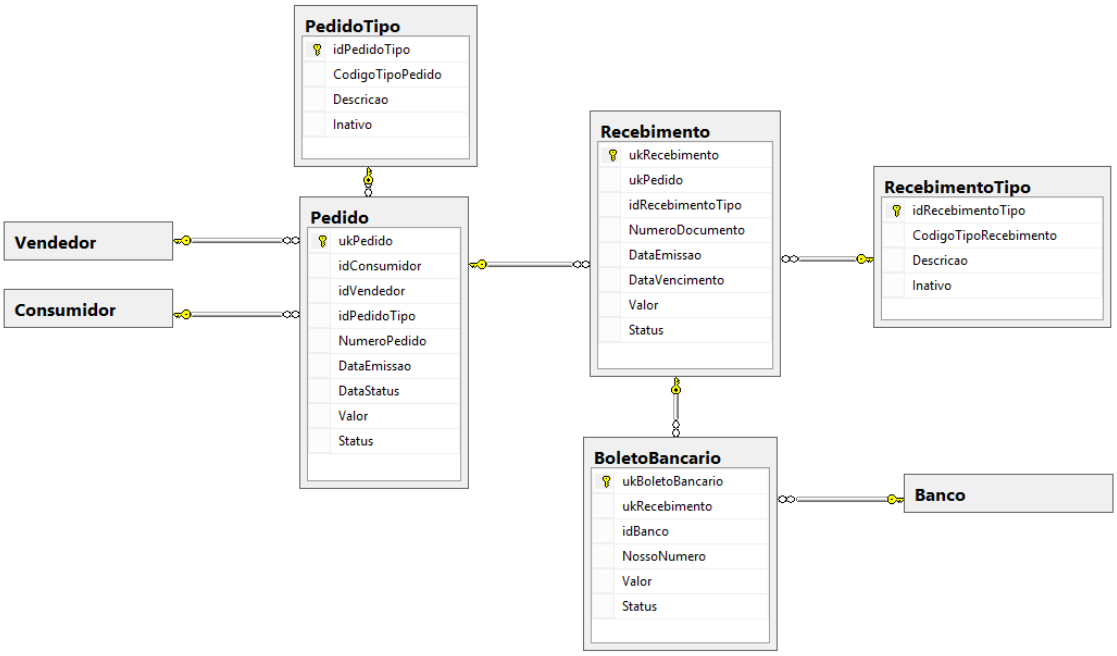


Figura 8A. Diagrama Entidade Relacionamento “Pedido” (Fonte Própria)

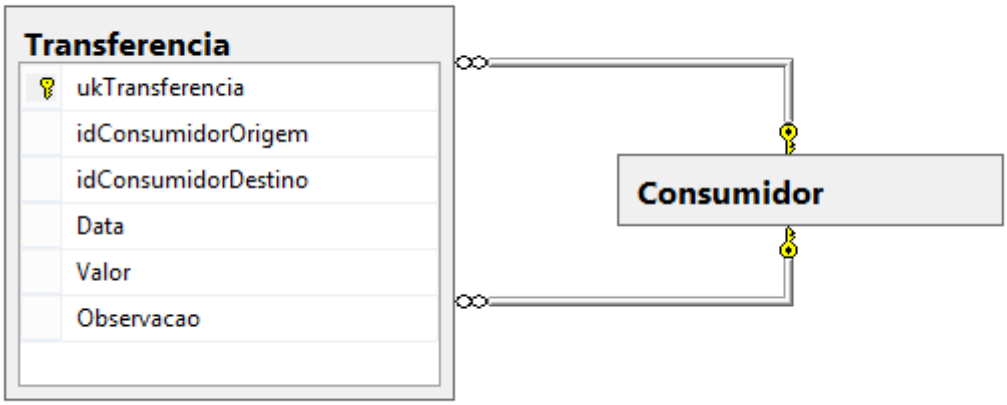


Figura 9A. Diagrama Entidade Relacionamento "Transferência" (Fonte Própria)

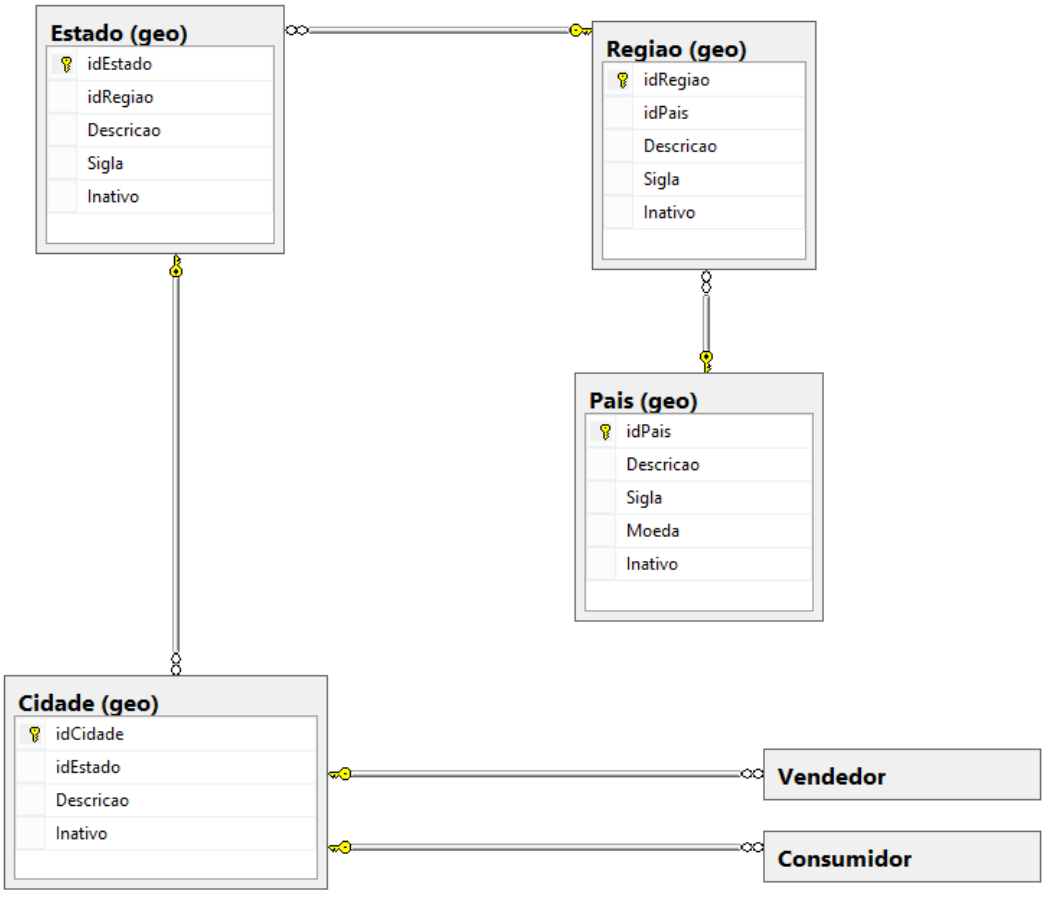


Figura 10A. Diagrama Entidade Relacionamento "Geografia" (Fonte Própria)