UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Implementação de um sistema de inventariação com arquitectura distribuída e visão computacional

Benvindo da Cruz Rocha

Orientação: José Miguel Gomes Saias

Mestrado em Engenharia Informática

Dissertação

Évora, 2013

Mestrado em Engenharia Informática

Implementação de um sistema de inventariação com arquitectura distribuída e visão computacional

Benvindo da Cruz Rocha

Orientação:

José Miguel Gomes Saias

Sumário

O presente trabalho consiste no desenvolvimento de um sistema de inventário baseado em visão computacional e arquitectura distribuída. Foi desenhado para dispositivos móveis Android para que pudesse efectuar contagem física dos artigos em Stock de forma eficiente, sem a necessidade do registo das quantidades em folhas de papel. Para o desenvolvimento do aplicativo utilizou-se a linguagem de programação Java, linguagem nativa para sistemas Android e com persistência dos dados em SQLite. A sincronização com o sistema de gestão é feita por via de um serviço *middleware* WCF criado exclusivamente para o projecto usando a linguagem C#. Utiliza protocolos comunicação HTTP ou TCP e o formato de dados JSON. Integrou-se a biblioteca ZXing para efectuar a leitura de código de barras estampados nos artigos. Permite o utilizador a leitura do código de barras com câmara fotográfica do dispositivo e após o reconhecimento retorna informação detalhada do artigo, bem como introduzir as quantidades da contagem física. Após o desenvolvimento foi preparado um cenário para testes usando o software de gestão Primavera. Assim, efectuou-se uma análise comparativa com os métodos tradicionais e os resultados demonstraram o quanto é o ganho em tempo e dinheiro com a utilização num ambiente organizacional. No processo de contagem física o leitor reconheceu em mais de 90% dos casos em que foi submetido demonstrando a viabilidade do sistema.

Palavras-chave

Inventariação; Leitura Código Barra; Visão Computorizada, Computação Móvel

Implementation of an inventory system with distributed architecture and computer vision

Abstract

This work is to develop an inventory system based on computer vision and distributed architecture. It was designed for Android mobile devices that could efficiently perform the physical count of items in Stock, without the need to record the quantities on the paper sheets. The Java programming language was used to develop the application, being the native language for Android systems and SQLite was chosen for data persistence. The synchronization with the management system is done via a WCF middleware service created exclusively for the project using the C# language. The system uses HTTP or TCP communication protocols and JSON as the data exchange format. ZXing library was included for bar code reading, from each article. Allows the user to read the bar code with the device camera and after recognition returns detailed information about the article, as well as introducing the quantities of physical count. Afterwards, a testing scenario was prepared with the management software Primavera. So, was carried out a comparative analysis with traditional methods and the results showed how much is the gain in time and money using an organizational environment. In the process of physically counting, this system reader correctly recognized over 90% of the cases, demonstrating this system viability.

Keywords

Inventory, Bar Code Reading, Computer Vision, Mobile Computing

Lista de abreviaturas e siglas

ERP - *Enterprise Resource Planning*

RFID - *Radio-Frequency IDentification*

JSON- *JavaScript Object Notation*

WCF - *Windows Communication Foundation*

COM+-*Microsoft Component Object Model*

XML - *eXtensible Markup Language*

HTTP - *Hypertext Transfer Protocol*

TCP - *Transmission Control Protocol*

IP - *Internet Protocol*

IIS – *Internet Information Service*

SQL - *Structured Query Language*

EAN13- *European Article Number*

Conteúdo

Lista de Figuras

Lista de Quadros

Capítulo 1

1 Introdução

Este capítulo faz um breve enquadramento do objecto de estudo aliado às motivações que levaram à escolha do tema, a descrição do problema e os objectivos que se pretendem atingir, bem como o esquema conceptual do presente trabalho.

1.1 Enquadramento e Motivação

A arquitectura clássica de um sistema de visão computacional inclui um computador e vários periféricos. Nos últimos anos, a capacidade de aquisição, visualização e comunicação de imagens foram integrados nos telemóveis mas o poder computacional ainda era uma limitação clara. No entanto, esta situação está a mudar com o surgimento de equipamentos mais sofisticados, designados de *smartphones e Tablets*. Por um lado, o *hardware* desses dispositivos está aumentando o poder de processamento e a capacidade de armazenamento, reduzindo assim a dependência dos convencionais computadores de secretária ou portáteis. Por outro lado, a popularidade dos sistemas operativos, com por exemplo Android [1] e iOS[2], voltados para dispositivos móveis tem aumentado, trazendo assim novas oportunidades de desenvolvimento.

Além disso, o rápido crescimento do mercado de aplicações produtivos, como o AppStore[3] e Android Market[4], está a atrair muita atenção aos desenvolvedores e consumidores.

O recente aumento da capacidade computacional tem possibilitado a integração de ferramentas de visão computacional, aproveitando assim as funcionalidades dos diversos

[1] www.android.com
[2] www.apple.com/ios/
[3] http://www.apple.com/osx/apps/app-store.html
[4] https://play.google.com/store

dispositivos periféricos integrados nos equipamentos nos sistemas de processamento de imagem.

Os dispositivos móveis oferecem a funcionalidade de aceder e introduzir a informação de forma rápida, em qualquer momento e em praticamente qualquer lugar. Dotados de câmara fotográfica capazes de obter imagem digital em boa qualidade, e usando lentes cada vez mais elaboradas. Devido às funcionalidades e ao facto de terem um preço menos proibitivo que no passado, os dispositivos móveis são cada vez mais usados em vários sistemas, nomeadamente sistemas de inventariação. Uma aplicação bastante útil é integração de sistemas de leitura de imagem de código de barras estampados nos artigos da empresa, que implica tarefas de campo, onde a aquisição e análise imediata das imagens constitui um requisito fundamental.

Por isso, a integração com sistemas de gestão de base dados constitui um elemento importante e elegível no âmbito de gestão, particularmente na contagem física dos artigos em Stock. A utilização desses dispositivos em sistemas de inventariação tem sido objecto de várias iniciativas, motivadas pela crescente cultura de utilização de dispositivos móveis mais sofisticados por parte dos colaboradores.

Os tradicionais métodos de contagem têm constituído uma preocupação por acarretarem uma tarefa árdua e demorada afectando o negócio. Assim, uma das principais razões da escolha do tema resulta da necessidade, identificada nas principais empresas cabo-verdianas, de adequar essa metodologia a novas tecnologias de visão computacional.

1.2 Descrição do problema

Geralmente, num processo de contagem física dos artigos em Stock envolve a deslocação dos operadores a locais de exibição ou depósito, nomeadamente armazéns, onde nem sempre existe a possibilidade de utilização de computadores convencionais para acesso a informação detalhada dos artigos e introdução da contagem física directamente no sistema. Perante esta limitação de mobilidade do computador, os operadores, em geral, utilizam pilhas de folhas de papel onde efectuam o registo da informação da contagem física e em seguida é introduzida no sistema computacional, perfazendo assim uma dupla introdução de informação, degradando a eficiência no processo de contagem.

Figura 1- Contagem física -método tradicional

Figura 2-Armazém

Logo, com o intuito de solucionar o problema, propõe-se a utilização de dispositivos móveis, integrando uma interface intuitiva e amigável com o utilizador e com possibilidade de comunicação em tempo real com o sistema de gestão de base dados e automatização da identificação do artigo, reduzindo o trabalho manual do operador.

1.3 Objectivos

O objectivo geral é conceber um sistema para gestão de inventário com arquitectura distribuída, visão computacional e interfaces móveis.

Como objectivo específico pretende-se:

- ✓ Desenvolver um aplicativo protótipo para dispositivos móveis integrado com o sistema Gestão mais precisamente o módulo inventário, que permite a contagem física dos artigos em Stock de uma forma eficiente usando um sistema de processamento de imagem de código de barras.
- ✓ Avaliar o desempenho, precisão e nível de usabilidade;
- ✓ Identificar e mensurar os factores que influenciam a usabilidade e o desempenho do protótipo;

- ✓ Efectuar uma análise comparativa em termos de produtividade com os métodos tradicionais de contagem.

1.4 Esquema conceptual do trabalho

O presente documento foi estruturado de modo a permitir uma leitura fácil, seguindo um encadeamento de ideias de forma lógica, coerente e objectiva, incluindo dados essências ao trabalho. Assim, o desenvolvimento do trabalho a apresentar, compreende:

O capítulo 1 é a introdução onde já foram referidos assuntos como: enquadramento do tema e sua relevância na actualidade; as motivações; a formulação do problema de investigação; a definição do objectivo geral e os objectivos específicos; neste ponto menciona-se o esquema conceptual do trabalho;

No capítulo 2 apresenta-se a revisão de literatura e o estado da arte, onde se procura apresentar alguns conceitos de visão computacional, as áreas de aplicação, bem como o levantamento de algumas ferramentas. Seguida de um levantamento de sistemas de inventariação usados nos dispositivos móveis e uma descrição da plataforma Android. Abordam-se ainda tecnologias para arquitecturas distribuídas, finalizando com trabalhos relacionados com o problema.

No capítulo 3 é apresentado um modelo protótipo e os seus requisitos, subsequente da revisão da literatura. Em seguida apresenta-se a metodologia seguida para a sua concepção.

No capítulo 4 faz-se uma apresentação e discussão dos resultados obtidos com a realização dos testes.

No capítulo 5 apresentamos as conclusões do estudo, as dificuldades sentidas na implementação do sistema e o trabalho futuro com o objectivo de aperfeiçoar o protótipo.

Capítulo 2

2 Revisão Teórica e Estado da Arte

Neste capítulo serão apresentados os fundamentos, conceitos e as relações entre os mesmos para a compreensão do sistema que será apresentado neste trabalho. Será feita uma análise da mais recente obra científica de visão computacional bem como as áreas de aplicação, seguida de um levantamento, análise e comparação das ferramentas de visão computacional. Em seguida, é feito um levantamento específico dos trabalhos relacionados com os sistemas de inventariação com arquitectura distribuída aliados a dispositivos móveis.

2.1 Visão Computacional

2.1.1 Conceitos

Importa aqui definir um dos conceitos base do trabalho, visão computacional que para Solem (2012) é a "extracção automatizada de informações a partir de imagens. As informações podem significar qualquer coisa a partir de modelos 3D, posição das câmaras, detecção de objectos e reconhecimento para agrupar, procurar o conteúdo da imagem ". Ballard & Brown (2010) são mais genéricos na definição considerando como a construção e descrição de forma explícita os objectos do mundo real a partir das imagens. Um sistema de visão computacional processa imagens obtidas a partir de uma câmara (ou conjunto de câmaras), e a extracção de informações relevante a partir da imagem adquirida (Solem, 2012) (Forsyth, 2012) (DAVIES, 2012). Tenta imitar o sistema de visão humana, onde o cérebro processa a imagem captada pelos olhos (ComputerVisionOnline, 2013).

Portanto, a visão computacional procura emular a visão humana, possui como entrada uma imagem, porém, a saída é uma interpretação da imagem como um todo, ou parcialmente.

Os elementos basilares de qualquer sistema de visão computacional incluem: câmara, objectiva, computador, software e a iluminação que é considerado o elemento chave para obtenção de resultados satisfatórios (Infainmon, 2010).

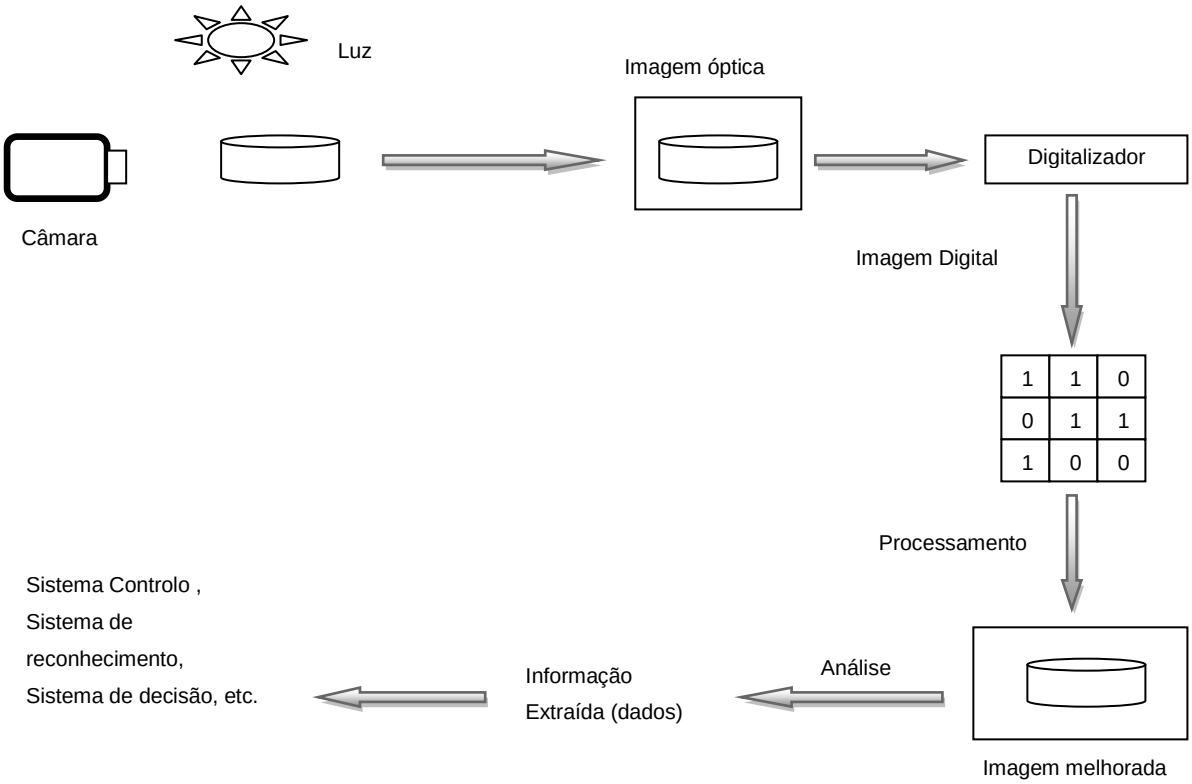

Figura 3-Visão Geral de um Sistema de Visão Computacional

Conforme a mesma fonte, existe a possibilidade de prescindir do computador e trabalhar somente com as câmaras inteligentes onde estão embutidos softwares de tratamento de imagens. As câmaras permitem a transformação de energia luminosa em eléctrica e transferir o resultado dessa conversão. As câmaras podem-se classificar segundo vários critérios, tais como:

- Resolução em pixel

-Tamanho do Sensor

-Pela sensibilidade do sensor a diferentes ângulos

As câmaras, segundo Infainmon (2010) são considerados imprescindível no processo de captação de imagem, contudo, os investigadores de Glasgow prescindiram das câmaras, ao criarem um sistema que é capaz de captar imagens 3D sem utilização de câmaras. De acordo com os investigadores, o sistema utiliza apenas um projector e quatro foto-sensores de 1 pixel cada, colocados em posições distintos. Projectando diferentes padrões sobre o objecto a capturar e vão mensurando a luminosidade reflectida, que depois de processada é integrado com todos os resultados dos sensores e logo a reconstrução da imagem em 3D (Barker, 2013). Portanto, com esta nova tecnologia prevê-se, num futuro próximo, dispositivos móveis do tipo smartphone com câmara de 4 pixéis com aplicação dessa tecnologia chamada pela mesma equipa de computação de imagens em 3D ou "*Ghost Image*". No âmbito do projecto serão utilizadas câmaras convencionais para testes ficando para trabalho futuro a aplicação dessa técnica para reconhecimento integral do produto e não somente na leitura do código de barras.

Outro factor importante para o bom desempenho do sistema computacional é o sensor de imagens que é constituído por uma matriz bidimensional de pixéis. Predominam dois tipos de sensores: CCD(*Charges-Couple Device*) e CMOS (*Complementary,Metal-Oxide,Semicondutor*).

Estudos realizados por (Nice, Wilson, & Gurevich, 2012) comprovam que os sensores CCD são menos ruidosos, consomem mais energia, têm menor velocidade de captura de imagens. Ora, mediante a natureza do projecto de processamento de imagens de código de barras e a velocidade processamento como um requisito essencial, então será utilizada dispositivos com sensores CMOS.

Como já referido anteriormente, outro elemento importante a ter em conta no sistema de visão computacional é a luminosidade. Nos testes efectuados a algumas ferramentas de visão computacional como o ZXing, Zbar e OpenCV, deparou-se que é fundamental encontrar a iluminação adequada para poder simplificar o processamento das imagens. A existência de iluminação fraca pode inviabilizar todo o processo.

Para finalizar, tem-se o computador e o software como partes integrantes do sistema visual computacional. São várias as alternativas para escolherem, desde dedicados até os dispositivos móveis *smartphones*, dependente da aplicação que se pretende. Quanto a categoria de computador para projecto, necessita-se de um com alguma capacidade de processamento em torno de 1Ghz. Serão utilizados dispositivos móveis *smartphones ou*

tablets que actualmente já possuem alguma capacidade de processamento e armazenamento, e também por motivo de mobilidade que o sistema impõe como requisito fundamental.

A nível de software, da análise feita da literatura encontrou-se duas opções. Primeira opção, é utilização de software especificamente desenhado para visão computacional mediante o uso de ferramentas genéricas programadas pelo fabricante. Conforme (Infainmon, 2010), a desvantagem é que apresentam um número limitado de operações para o desenvolvedor, e como vantagem é a facilidade de desenvolvimento de aplicações num curto espaço de tempo.

Outra alternativa encontrada é a utilização de bibliotecas que oferece ao programador a possibilidade de construir uma aplicação à medida. Os mais populares encontrados são o *ZXing*, *Zbar e o OpenCV*, que são bibliotecas de código aberto e serão descritos e comparados mais adiante neste trabalho.

A seguir são apresentados na figura 3 algumas das funcionalidades comuns de qualquer sistema de visão computacional, conforme indicados por Rhem e Trindade citado por (Milano & Honorato, 2010):

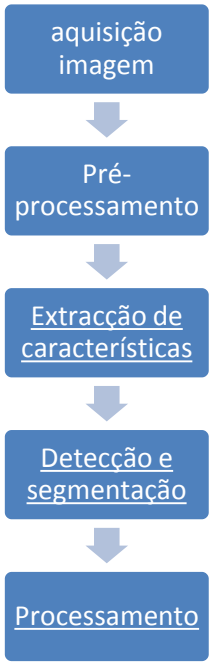

Figura 4-Etapas do processo de reconhecimento de imagem

Aquisição de imagem

É a primeira etapa trata-se do processo de aquisição de uma imagens ou de uma sequência de imagens 2D ou 3D a partir de sensores das câmaras, em que os pixéis de cada imagem obtida indicam as coordenadas da luz e as coordenadas físicas.

Pré-processamento

As imagens de onde queremos extrair alguma informação em alguns casos precisam ser convertidas para um determinado formato ou tamanho e precisam ainda ser filtradas para remover ruídos provenientes do processo de aquisição da imagem. Os ruídos podem aparecer de diversas fontes, como por exemplo, o tipo de sensor utilizado, a iluminação do ambiente, as condições climáticas no momento da aquisição da imagem, a posição relativa entre o objecto de interesse e a câmara.

Extracção de características

Extracção de características matemáticas que compõem a imagem, como textura, borda, movimentos e formatos.

Detecção e segmentação

Processo realizado para destacar regiões relevantes da imagem, segmentando-as para a fase do processamento.

Processamento

Processo que inclui a validação da satisfação dos dados obtidos, estimativa dos parâmetros sobre a imagem e classificação dos objectos obtidos em diferentes categorias.

2.1.2 Áreas de aplicação

Visão computacional tem inúmeras aplicações, como por exemplo: na segurança e vigilância; medicina; controlo de qualidade e inspecção; leitura de automática de matrículas, controlo de tráfego inteligente e leitura de código de barras (Saias, 2013).

Uma das aplicações mais promissoras é no campo de medicina que consiste na extracção de informações de imagens (raio-X, tomografia computorizada e ressonância magnética) com

objectivo de diagnosticar certas doenças. Um caso concreto é analise computorizada da papila e fibras nervosas, uma aplicação desenvolvida por VisionLazer, que permite uma análise precisa das camadas do nervo óptico, realizado através de um feixe de luz que em seguida é analisado pelo computador. Outras informações que podem ser extraídos das imagens são os tumores (VisionLazer,2011).

Outra vasta área de aplicação é na indústria, vulgarmente conhecido por "visão maquina" onde as imagens são extraídas e processadas para dar suporte aos processos industriais. Um exemplo é o sistema desenvolvido por Montrose *Techonologies Inc* no Canadá em que há uma aplicação numa padaria para o controlo de certas propriedades dos bolos como a cor e a forma, controlo o tamanho dos pães, biscoite e tortas, etc. (MontroseTechnologies, 2009).

Na mesma área de controlo de qualidade e inspecção, encontrou-se o sistema Ellips desenvolvido na Holanda que visa inspeccionar e classificar as frutas e vegetais. São aplicados algoritmos avançados de processamento de imagens para determinar com precisão certas características como a cor, peso, dimensão e a qualidade (Ellips, 2011).

No campo de reconhecimento de objectos através de dispositivos móveis, o Snaptell é uma aplicação que permite aos utilizadores efectuarem buscas de lojas *online* após o reconhecimento do objecto ou o código de barra com a câmara do *smartphone*. Por exemplo, facilmente se podem encontrar uns sapatos pretendidos nas principais lojas *online* com base na aparência do produto e através de uma fotografia digitalizada.

Login sem palavra-chave através de reconhecimento de face começa a aparecer largamente nos computadores, telemóveis e *sites*, como por exemplo a solução de (SensibleVison, 2007). Para muitos utilizadores o problema de memorização de palavra-chave é resolvido.

2.1.3 Ferramentas de Visão Computacional

OpenCV (*Open Source Computer Vision*) é uma biblioteca de programação, de código aberto, desenvolvida inicialmente pela Intel *Corporation*. O OpenCV implementa uma variedade de ferramentas de interpretação de imagens, indo desde operações simples como um filtro de ruído, até operações complexas, tais como a análise de movimentos, reconhecimento de padrões e reconstrução em 3D. O pacote OpenCV está disponível gratuitamente na Internet bem como o manual de referência. Esta biblioteca foi desenvolvida nas linguagens de programação C/C++ mas dá suporte para integração em aplicativos Java e também Python através de JNI, bem como em aplicativos Android. Desenhado para garantir eficiência no processamento e é focalizado para aplicações em tempo real, como por exemplo, seguimento de movimentos (*MotionTracking*), reconhecimento facial e identificação de objectos (OpenCV.org, 2013).

OpenTL é uma biblioteca de monitorização e acompanhamento de Imagens que consiste na integração de técnica de percepção de imagens, com metodologias de seguimento (*tracking*) com objectivo de localizar um ou mais objectos em movimento num vídeo ou sequência de imagens. Esta ferramenta tem sido aplicada em várias áreas de interesse tais como: vigilância por vídeo; sistemas de navegação; cirurgias assistidas por computadores (OpenTL, 2011).

BoofCV é uma biblioteca de código aberto disponível para aplicação Java com métodos fáceis de usar e de alto desempenho prontos para visão computacional em tempo real. Sendo escrito em Java, facilita assim a sua utilização no Android, pois não precisa de muita modificação ou código nativo. Lançado sob uma licença Apache tanto para uso académico como comercial, têm-se feito alguns estudos comparativos com outras ferramentas de visão computacional em matéria de velocidade onde tem superado bem (Boofcv.org, 2013).

No *site*[5] existem exemplos numéricos e vários tutoriais em vídeo num canal youtube e também se pode executar uma série de *applets* no browser para ver as funcionalidades antes de instalação.

***NASA Vision Workbench* (VW)**[6]**-** É uma biblioteca de visão computacional desenvolvida pelo ASR, Sistemas Autónomos e Robótica, área de divisão de sistemas inteligentes no Centro de Pesquisa Ame NASA. Foi lançada publicamente nos termos de *NASA Open Source Software Agreement* - NOSA. VW foi implementada na linguagem de programação C++.

O presente trabalho protótipo a ser desenvolvido tem como principal requisito funcional, conforme referido anteriormente, a leitura de código de barra do tipo EAN13, o mais usado nos produtos europeus. Especificando mais a análise de literatura tendo em conta objectivo especifico do trabalho, encontrou-se:

ZXing

Igualmente conhecido por "*Zebra Crossing*", é uma biblioteca de código aberto implementada em Java que descodifica imagens de código de barras 1D ou 2D usando dispositivos móveis com câmara integrada. Apresenta como principal vantagem o facto de não necessitar da ligação com um servidor para obter esses códigos descodificados. Tem suporte para maioria dos tipos de códigos de barras, tais como: UPC-A e UPC-E, EAN-8 e EAN-13, Code 39, Code 93, Code 128, QR Code, ITF,Data Matrix, etc (ZXing.org, 2008). Tem suporte a outras linguagens de programação como Net e C#. Para a sua utilização em ambiente android é preciso a instalação da aplicação *Barcode Scanner* que tem integração via *Intent* com a aplicação cliente. No *site*[7] existem dois ficheiros APK para download. Um é a biblioteca do *ZXing*, já modificada pelo autor, e outro é um projecto de exemplo onde é possível visualizar como fazer a integração da forma mais simples possível. Basta copiar tudo para o projecto e

[5] http://www.boofcv.org
[6] http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/nasa-vision-workbench/

[7] www.ZXing.org

usar. Basicamente é preciso criar um ficheiro XML com a representação da leitura, colocar a biblioteca do *ZXing* no projecto e fazer a chamada do leitor *Barcode Scanner*.

Basicamente com as mesmas funcionalidades tem-se:

Zbar

É uma solução de código aberto que suporta plataformas Linux, Windows e iOS é capaz de descodificar um código de barras presente nos artigos como também processar ficheiros de imagem de vídeo em tempo real. Esta biblioteca suporta os tipos mais populares de código de barras como por exemplo: EAN13, UPC-A , EAN8, Code 128 , Code 39 (Zbar, 2012).

Tem inúmeras aplicações, no retalho controlo de Stock, aplicações móveis e processamento automático de documentos.

Conforme a mesma fonte, possui como pontos fortes: a compatibilidade com Python,C#,.Net; alta velocidade; baixo uso de memória; não é limitado a imagens estáticas; adequado para processadores de fraca capacidade de processamento; componentes em módulos que podem ser usados em conjunto ou separados.

2.2 Mobilidade e sistemas de inventariação

2.2.1 Dispositivos móveis

O termo "dispositivos móveis" ora apresentado neste trabalho abrange os conceitos de *smartphone e tablet*. Maioria dos dispositivos móveis que são lançados ultimamente possui uma razoável capacidade de processamento e armazenamento, estão equipadas com câmara fotográfica com alguma capacidade de resolução e tem possibilidade de acesso a rede de dados Internet sem fios 3/4G. Utilizados para vários fins, estes dispositivos têm incentivado

os desenvolvedores a criarem *softwares* para ambientes negócio para optimização de tempo e dinheiro nos processos organizacionais.

Existe uma grande variedade de plataformas no mercado. Apesar de existirem vários fabricantes de dispositivo móvel, actualmente o que mais influi é o sistema operativo utilizado. A falta de padronização e de um sistema operativo único leva com que cada aplicativo tenha que ser codificado de diferente forma para cada um dos modelos existentes**.**

De entre as principais plataformas no mercado encontra-se: Android Google, Symbian Nokia, iOS Apple, Blackberry RIM, Windows Mobile Microsoft, etc.

Os estudos de mercado relatam que o Android está praticamente a atingir o valor de mercado da Apple no que respeita à popularidade no mercado dos *smartphones* e que atingiu já o mesmo nível no que respeita ao mercado dos *tablets*.

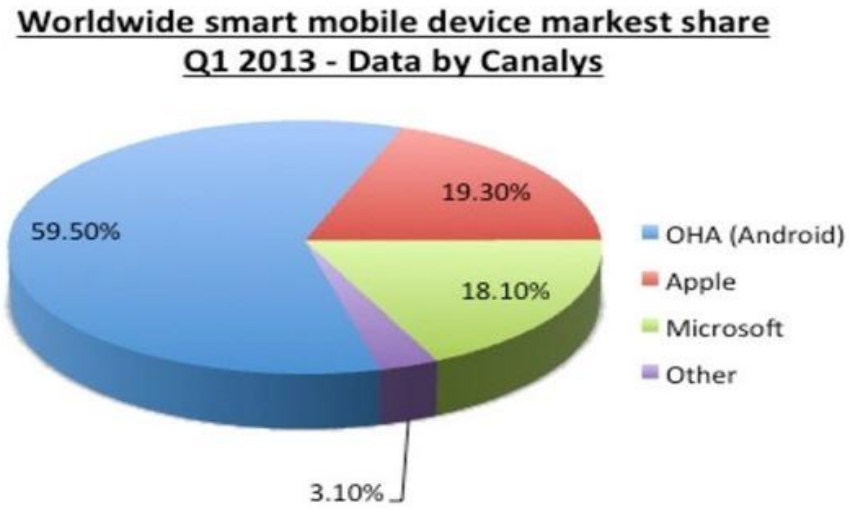

Figura 5-Mercado dos dispositivos móveis

Fonte: ZdNet- Technologies News

No entanto, conforme os dados do gráfico anterior, fica ilustrado que o Android tem maior quota de mercado em relação a Apple e Microsoft referente aos dispositivos móveis. De referir também que o Android ainda não tem presença significativa no segmento dos PC's mas, no futuro (breve), poderá ser também um forte concorrente ao Windows. O Android está presente em vários dispositivos (de referir que a Samsung lidera como marca mais

dispositivos com Android comercializa), de várias marcas, ao contrário do iOS que apenas está presente nos dispositivos da Apple.

Portanto, o mercado de sistemas operativos móveis para os mais recentes dispositivos tem vindo a crescer, levando os desenvolvedores de software ao investimento no desenvolvimento das suas aplicações onde cada sistema operativo tem as suas próprias características e os respectivos pontos fortes. Das pesquisas feitas, um dos sistemas operativos mais promissores é a plataforma Android que em seguida será objecto de uma descrição dos principais componentes.

2.2.2 Android

O presente tópico focaliza nos principais conceitos da plataforma móvel Android. Conforme referido anteriormente, é um sistema operativo de código aberto desenhado para dispositivos móveis que executa sobre o *kernel* do Linux. Actualmente, o Google é responsável pela gestão produto e o seu desenvolvimento. O Android permite aos desenvolvedores escreverem softwares na linguagem Java controlando os dispositivos via bibliotecas desenvolvidas pelo Google.

Na figura que se segue ilustra a arquitectura do sistema Android.

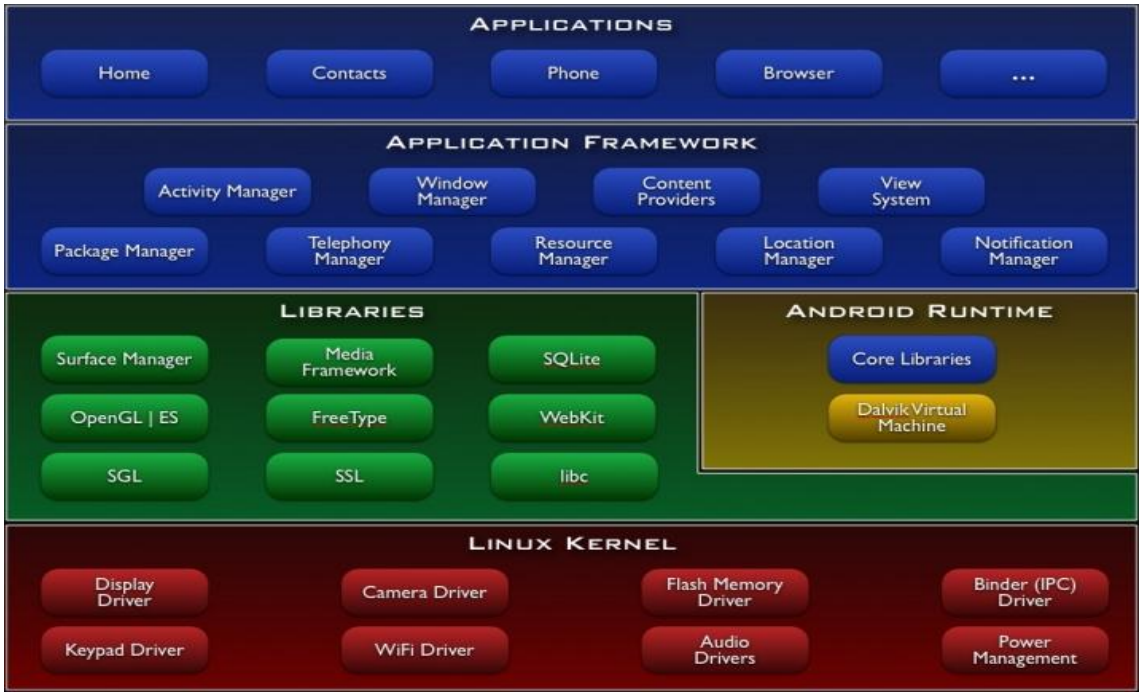

Figura 6-Arquitectura do sistema Android

Fonte: http://www.elinux.org

Fazendo uma breve descrição dos principais componentes (Meier, 2010):

O Linux *Kernel* é o nível em que nunca é usado directamente pelos desenvolvedores. No nível mais cima existe um conjunto de bibliotecas de código nativo, escritas em C e C++ e disponibilizadas para vários componentes e camadas superiores.

O *SQLite* é um motor de base de dados relacional disponível para todas as aplicações.

No Android Runtime existe:

O *Dalvik Virtual Machine* que é uma máquina virtual (VM) que executa as instruções do programa. Toda aplicação Android executa em seu próprio processo, com sua própria instância da máquina virtual Dalvik. O Dalvik foi escrito de forma a executar várias VMs eficientemente. Ele executa ficheiros .dex, de modo optimizado para consumo mínimo de memória. A VM é baseada em registos e executa classes compiladas pela linguagem Java que foram transformadas em ficheiros .dex, através da ferramenta "dx" incluída no SDK. O Dalvik VM baseia-se no *kernel* do Linux para funcionalidades subjacentes como o escalonamento e a gestão de baixo nível de memória.

O módulo *Core Libraries* fornece funcionalidades à maior parte das bibliotecas usadas na linguagem de programação Java.

Na camada superior, estão as aplicações desenvolvidas em Java que vêm junto com o Android, como: cliente *email*; contactos; agendas; mapas; navegador; programa SMS.

Activity

É uma tarefa que especifica as funcionalidades que o utilizador pode efectuar, isto é, a interface com utilizador. Uma classe que cria uma janela em que é colocado todas as componentes de interface com utilizador. Cada actividade tem o seu próprio ciclo de vida em que o programador não tem qualquer controlo sobre os estados de cada actividade. A gestão de sistema controla os estados de cada actividade e notifica os programadores quando há transição de estado. A figura 7 ilustra sequência dos estados e os métodos associados a cada um.

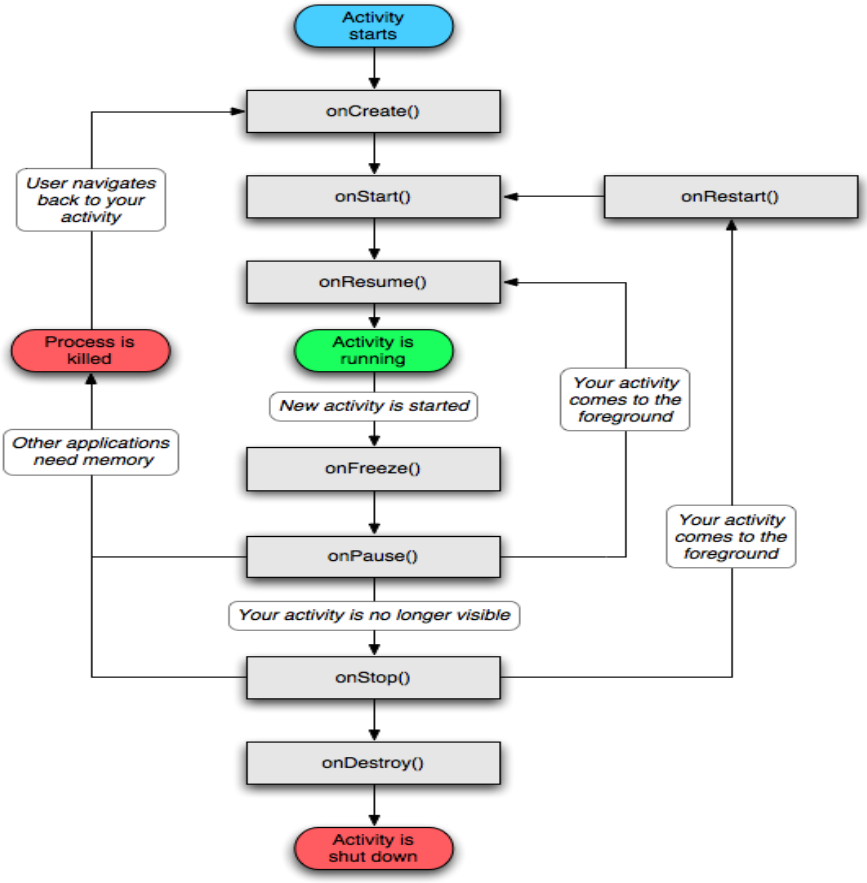

Figura 7-Ciclo vida de um Activity

Fonte: www.android.com

O diagrama acima mostra os caminhos importantes de uma *Activity*. Os rectângulos representam chamadas a métodos que se pode implementar para realizar operações quando a actividade se mude entre os estados. Os rectângulos com cantos arredondados são os estados principais que uma actividade pode ter.

Existem dois métodos que todas as subclasses de *Activity* implementam:
OnCreate(Bundle) - é executado assim que a classe *Activity* é criada. Dentro do método é chamado o setContentView(int) que inicializa o *layout* XML da interface da *Activity* e o findViewById(int) para retornar as propriedades (exemplo, valor) *widgets* daquela interface.
OnPause() - é quando a *Activity* perde o foco mas continua visível. Guarda o estado e as informações, no entanto, pode ser encerrado caso o sistema precise de liberar memoria.
OnStop() – É quando é completamente obscurecida por outra actividade, igualmente pode ser encerrada caso for precise liberar memoria.
Portanto, o ciclo de vida completo de uma actividade acontece entre a primeira chamada do onCreate(Bundle) até o onDestroy(). Uma actividade vai fazer toda a configuração do estado "global" no onCreate() e liberar os recursos remanescentes em onDestroy().

Intents

É possível enviar mensagens ao sistema operativo para que este execute uma acção, seja executar uma *Activity* ou um serviço em segundo plano.
Um exemplo tipo de utilização do *Intent* é a chamada do leitor de código barra ZXing para scannear algo e retornar o resultado a actividade chamadora.
Conforme ilustrado no código:

```
Intent  intent = new Intent(“com.google.ZXing.client.android.SCAN”);
startActivityForResult(intent,0);
```

Quadro 1-Chamada de BarCode Scanner via Intent

2.2.3 Processo de Inventário Físico

É um procedimento administrativo comum a qualquer organização e é executado pela contagem dos artigos em Stock a fim de garantir altos níveis de precisão entre os valores no sistema e a situação real. Os procedimentos são os mesmos em qualquer sistema de Gestão, apresentando diferentes níveis de eficiência. Inventário físico no (Primavera, 2013) inclui 5 etapas:

1-Preparação do Inventário - consiste na criação dos documentos de inventários;

2-Impressão do documento e introdução dos resultados da contagem no mesmo documento impresso;

3- Introdução dos resultados da contagem física no Sistema;

4-Encerramento e Lançamentos das diferenças.

5- Actualização da ficha dos artigos, Stock actual.

Os documentos de inventário são a base do inventário porque possuem a lista dos artigos marcados para a contagem, bem como o Stock actual de cada artigo. Geralmente, por questões de mobilidade, é feita a impressão dos documentos para registar os resultados da contagem.

Para evitar inconsistência, os Armazéns que estão em contagem são bloqueados logo após a preparação do inventário, isto é, o sistema não permite a realização de movimentos, tais como, entradas ou saídas de Stock, transferências de armazém e vendas a clientes, nos artigos marcados para contagem física. Após o encerramento do inventário os artigos são liberados permitindo assim movimentá-los.

Com referido anteriormente, o sistema apresenta limitações de mobilidade acarretando questões de duplicação de introdução de informação. São muitas as soluções no mercado na tentativa de agilizar esse processo por forma a diminuir o tempo gasto na contagem.

2.2.4 Imagem de Códigos de Barras

Código de barras é uma representação gráfica de dados numéricos ou alfanuméricos. A descodificação dos dados é normalmente feita utilizando um tipo de *scanner*, o leitor de código de barras, que emite um feixe de luz vermelha que percorre todas as barras. Onde a barra for escura, a luz é absorvida; onde a barra for clara, a luz é reflectida para o leitor. Os dados capturados através desta leitura são descodificados na forma de letras, números legíveis ao ser humano.

O formato mais utilizado em nível global, EAN-13, é composto pela combinação de 13 dígitos divididos em quatro partes. Os 3 primeiros dígitos representam o prefixo da organização responsável por controlar e licenciar a numeração no país (*country code*); os próximos dígitos, variando de 4 a 5, representam a identificação do fabricante ou empresa proprietária da marca do produto (*manufacture code*); a terceira parte contendo 4 dígitos identifica o produto produzido pela empresa (*product code*); e por último um dígito verificador (*check sum*) [CSB 2011], conforme mostra a Figura 8.

Figura 8-Código de barras e os seus componentes

Fonte: Code Project[8]

[8] http://www.codeproject.com/Articles/20823/Barcode-Image-Generation-Library

2.2.5 Aplicações existente no Mercado

Dispositivos móveis estão sendo utilizados em inúmeras tarefas de negócio, o que tem aumentado cada vez mais o desenvolvimento de aplicações à medida. Foi feito um levantamento das aplicações de inventariação existentes no mercado, bem como a exploração das funcionalidades e testes de usabilidade de algumas.

De entre as que mais se aproxima ao problema do presente trabalho destaca-se (GooglePlay, 2011):

Maxim mobile Inventory

Uma solução desenvolvida pela IBM que permite a gestão de inventário. As principais funcionalidades são:

- Permite a gestão dos activos com dispositivos de inventário móvel;
- Permite executar transferências de produtos entre os Armazéns;
- Utiliza leitura de código de barra e recursos de RFID para rastreamento e gestão do inventário;
- Troca dados com o Servidor usando métodos em tempo real e sem fios.

Scan to SpreedSheet

É um aplicativo que recolha de dados e armazena numa folha de cálculo. A leitura do código de barra dos produtos é feita através da ferramenta *Barcode Scanner*. Posteriormente, a informação recolhida no ficheiro é importado na aplicação central de gestão.

Inventory Droid

À semelhança da anterior, é um aplicativo de gestão de activos onde cada item no inventário contém uma lista de atributos personalizados como a localização, data compra, nº de série, valor, etc. Ainda, contém a imagem dos produtos e os seus recibos de compra. Possui suporte para base de dados online, o Google e Amazon. Podem-se consultar atributos de um produto através de uma simples leitura do código de barras. Projectado para um único dispositivo,

permite exportar para ficheiros em formato CSV. Tem uma versão que permite a "computação em nuvem" [9], suportando assim vários utilizadores em simultâneo.

Inventory count

É um aplicativo de controlo de stock que suporta sistema de contagem em ciclos, criado pela empresa ERP WMS[10]. Substitui folhas de papel para etiquetas de Stock virtual no telefone ou *tablet* a fim de fazer uma contagem rápida. É gerado um documento no **ERP**[11] que contém informações de contagem e é exportado para um *smartphone* ou *tablet* através de um serviço dedicado *Mobilize IT App Server* (*middleware*[12]). O utilizador acede às informações dos de produtos no Servidor por meio deste serviço, permitindo saber a existência do produto, bem como registar a contagem física. Após a contagem as informações são retornadas por meio do mesmo serviço e depois armazenadas na base de dados do sistema de gestão, para efeitos de relatório e actualização do Stock dos produtos. Apresenta uma particularidade em que, opcionalmente, se pode usar entrada de comando de voz na aplicação.

Retail Inventory

Aplicativo de gestão de inventário adequado para lojas.Com suporte a um telefone Android permite a contagem de forma rápida. O código de barras do produto é digitalizado, logo é inserido a quantidade e em seguida sincronizado com a base de dados.

Portanto, foram apresentadas algumas aplicações existentes na loja de distribuição de aplicações móveis Google Play. Algumas foram instaladas num smartphone android e foram feitas análises às principais funcionalidades, bem como a usabilidade das mesmas. Algumas possuem integração em tempo real com sistemas de gestão de base dados através de serviços *middleware*, enquanto outros, a integração e feita de forma *offline*, ou seja, através de

[9] O conceito de computação em nuvem (em inglês, *cloud computing*) refere-se à utilização da memória e das capacidades de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da Internet, seguindo o princípio da computação em grade.

[10] http://www.iqms.com/products/erp/manufacturing/inventory/warehousemgmt.html

[11] Sistemas Integrados de Gestão Empresarial (em inglês ERP-*Enterprise Resource Planning)*- são sistemas de informação que integram todos os dados e processos de uma organização. São integradas numa única plataforma de software as seguintes áreas funcionais: recursos humanos, contabilidade, compras, stock, vendas, marketing,etc.

[12] *Middleware* é uma camada de software que fornece uma abstracção, escondendo a heterogeneidade dos vários componentes como hardware, rede, Sistema Operativo ou linguagens de programação, oferecendo um modelo computacional uniforme para programadores e aplicações (George Coulouris, 2011).

ficheiros com formatos adequados que são importados. Para muitos desses aplicativos, não foi possível encontrar a metodologia aplicada no processo de implementação.

2.3 Arquitectura Distribuída

Actualmente, cada vez mais os serviços estão nas aplicações, seja consumindo um serviço, ou servindo uma informação. Em aplicações distribuídas, os serviços têm-se tornado uma peça fundamental, quebrando assim o paradigma de utilização das bibliotecas dinâmicas, DLL.
Nesta mesma linha de sistemas orientados a serviços, a Microsoft (2013) desenvolveu a tecnologia WCF que permite integração de sistemas com plataformas diferentes, quebrando assim a limitação dos Serviços Web.

2.3.1 WCF (*Windows Communication Foundation*)

É um sistema distribuído desenvolvido pela Microsoft e utilizado para cuidar da comunicação entre sistemas. O WCF é uma combinação de Serviços Web, WSE, *Remoting* e COM+, tudo isso em uma única plataforma, simples de usar, robusta e de fácil integração. São muitos os sistemas que comunicam via Serviços Web. Este fornece uma maneira eficiente de comunicação mas apresenta algumas limitações, primeiro, a comunicação é feita unicamente através de protocolo HTTP, outra limitação é que só permite comunicação *simplex*, enquanto o WCF a comunicação é *full duplex* e permite protocolos HTTP, TCP e IPC (Singh, 2012).
A figura 9 ilustra uma arquitectura de um sistema baseado na tecnologia WCF.

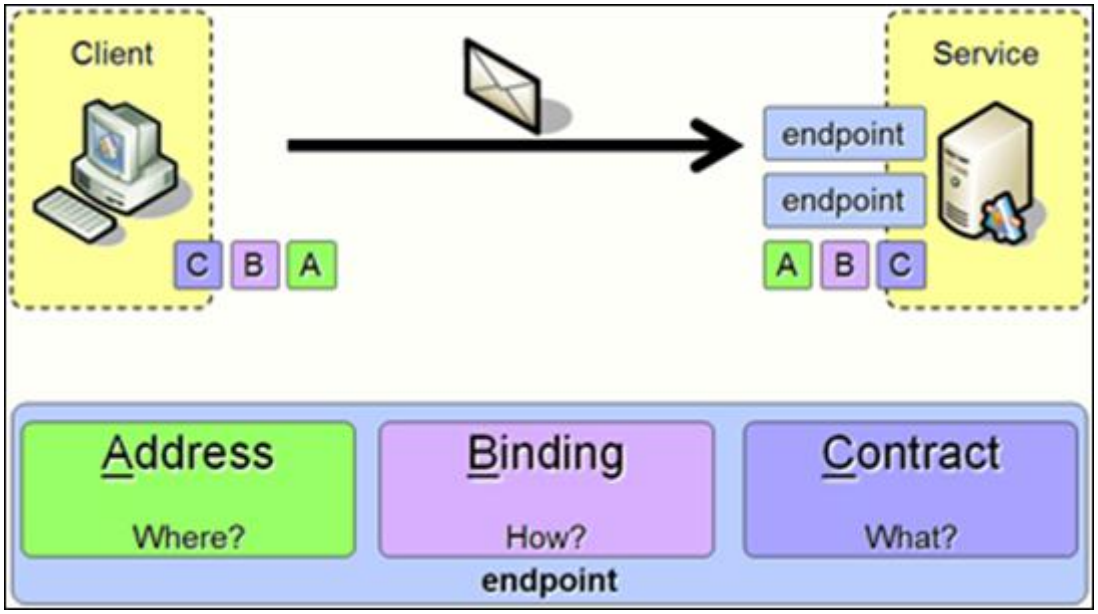

Figura 9-Arquitectura de um sistema baseado na tecnologia WCF

Conforme se pode visualizar, possui uma arquitectura de Cliente-Servidor. O cliente comunica com serviço WCF, por via de HTTP ou TCP e o formato do conteúdo das mensagens podem ser XML e JSON.O serviço é constituído uma ou mais *Endpoints*. Um *Endpoint* é o que o serviço expõe e possui três características, mais conhecido por ABC (*Address,Binding, Contract*):

Address **-** é o endereço da hospedagem do serviço. Outra das vantagens de utilização do WCF é que não há necessidade de hospedar o serviço no IIS (*Internet Information Server*). Há possibilidade de hospedar o serviço numa aplicação Windows ou um serviço Windows.

Binding **-** é como o serviço pode ser acedido e as formas mais comuns são HTTP e TCP. No contexto empresarial, HTTP é utlizado em casos em que o serviço é acedido fora da empresa, enquanto o TCP tem melhor performance de comunicação, mas limita-se apenas às comunicações dentro da empresa. No entanto, existe a possibilidade de utilizar ambas as opções. Um dos requisitos a ser implementado no protótipo é a possibilidade de ligação pelas duas vias.

Contract **–** é o que pode ser visto no serviço. No conceito de orientação a objectos, um contrato expõe os membros de uma classe que serão visíveis. No WCF o conceito é o mesmo,

e através de interfaces podemos definir um contrato entre um serviço e as aplicações que irão consumi-lo, expondo somente os métodos desejados.

Conforme referenciado anteriormente, o formato do conteúdo das mensagens trocadas entre o cliente e o serviço WCF podem ser XML e JSON. Em seguida é feita uma breve comparação entre os dois:

2.3.2 JSON (*JavaScript Object Notation*)

Actualmente, as aplicações distribuídas utilizam o formato de dados XML ou JSON para troca de dados. Ambos são os formatos de troca de dados utilizados na Web, e é preciso saber em que situações usar um ou outro. O quadro 3 ilustra exemplos do formato dos dados de JSON e XML:

	JSON		XML
1	{	1	<Artigos>
2	"Artigos": [	2	<Artigo>
3	{	3	<codigo>A001</codigo>
4	"codigo": "A001",	4	<descricao>Portatil Hp 2 Gh</descricao>
5	"descricao": "Portatil Hp 2Gh"	5	</Artigo>
6	},	6	<Artigo>
7	{	7	<codigo>A002</codigo>
8	"codigo": "A002",	8	<descricao>mesa de computador</descricao>
9	"descricao": "mesa de computador "	9	</Artigo>
10	}	10	</Artigos>
11	]		
12	}		

Quadro 2-Comparação entre JSON e XML

Numa comparação entre os dois formatos efectuada por (Mittal, 2012), o mesmo considera:

- O formato JSON é mais leve do que XML;

- JSON é reconhecido nativamente pelo JavaScript;
- A desvantagem mais importante de JSON é que o formato é muito difícil de ler para os humanos;
- JSON é melhor para o consumo de dados em aplicações web a partir de Serviços Web;
- Para efeitos de configurações, XML é o mais adequado, porque é mais fácil de ler;
- JSON é um formato melhor para troca de dados. XML é um formato melhor para troca de documentos.

Portanto, o JSON é uma opção leve e optimizada, permite representar dados estruturados, como classes e *arrays*, e é adequado para troca de dados em sistemas com arquitectura distribuída. Ao contrário de (Mittal, 2012), conforme ilustrado nos exemplos no quadro 3 acima, considera-se o JSON mais legível para o humano do que o XML.

2.4 Trabalhos relacionados

A presente subsecção aborda alguns trabalhos desenvolvidos em ambiente académico relacionados com o trabalho.

Irineu Kapietz, no âmbito do projecto de dissertação de mestrado, desenvolveu um sistema de **vendas a retalho** em que utilizou a leitura de código de barras aplicando a biblioteca ZXing. O aplicativo, desenhado para smartphone Android, comunica com um Serviço Web em formato XML para a realização de operações de compra e venda de produtos usando a ferramenta *ZXing*. Enquanto na loja o processo é feito através de RFID[13]. Fez uma análise comparativa dos dois métodos de leitura de código de barras e os dois não apresentaram diferença significativa no que toca à busca da informação do artigo na base de dados (KOPIETZ, 2011).

[13] RFID é uma sigla de "*Radio-Frequency IDentification*" originária da língua inglesa que em português significa Identificação por Rádio Frequência. Consiste num método de identificação automática através de sinais de rádio, recuperando e armazenando dados remotamente através de dispositivos denominados etiquetas RFID.

A aplicação **Atendente Móvel** segue os mesmos traços que o projecto acima resumido. É uma aplicação usada em dispositivos móveis para obter informações do produto com simples leitura de código de barras. Apresenta uma arquitectura Cliente-Servidor e com um Serviço Web que interliga a aplicação com o servidor de base de dados. Efectua a busca de produtos numa base de dados através de leitura de código de barras aplicando *ZXing*. Por enquanto, permite apenas a busca de livros na internet em que o resultado da pesquisa é uma lista de loja *online* onde existe o livro (Siqueira, 2011).

Outro projecto com uma abordagem diferente dos anteriores é o ***VisioScan***, um projecto de pesquisa apresentado no Simpósio Anual de informática no Brasil (Conte et al, 2008). É uma aplicação para o reconhecimento de imagens de código de barras com codificação EAN13 utilizando redes neuronais. Foi desenvolvido sob tecnologia Java e, para além da aplicação de leitura de imagens, foi implementado uma rede neuronal artificial do tipo *Percepton* Múltiplas Camadas, usando *backpropagation* no processo de treino. Para a integração, implementou-se um Serviço Web que recebe os pedidos dos dispositivos móveis e depois envia o pedido para o reconhecedor de imagem. Este projecto apresenta algumas particularidades, utiliza redes neuronais e o reconhecimento da imagem é feito no servidor que posterior envia o resultado para o dispositivo móvel. Conforme os resultados apresentados, o reconhecedor classificou correctamente a maioria dos casos em que foi submetido comprovando assim a eficiência das redes neuronais. Por outro lado, um trabalho divulgado na Alemanha por (Scheuermann, Werner, & Kessel, 2011) fez uma avaliação de performance do descodificador de código barra usando a biblioteca ZXing. Foi gerada uma biblioteca de imagens aplicando todo tipo de distorção que se pode esperar numa situação real. Foram examinados todos os tipos de código de barras como: *QRcode, MaxiCode*, EAN13. A mesma biblioteca apresentou boa performance, apresentando melhores resultados para código de barras do tipo *QRCode*. Também houve alguma dificuldade no reconhecimento em situações de fraca luminosidade, um problema encontrado na maioria dos sistemas de visão computacional.

Capítulo 3

3 Implementação

Neste capítulo apresenta-se a descrição do modelo teórico do sistema de inventariação com arquitectura distribuída, bem como a metodologia seguida para implementar um sistema protótipo.

3.1 Proposta de um modelo

Após a revisão literatura, levantamento e testes de uma série de trabalhos que abordam o problema de pesquisa e no intuito de solucionar o problema propõe-se:

3.1.1 Descrição do protótipo

O desenvolvimento de um protótipo de inventariação com arquitectura distribuída e visão computacional. O protótipo é desenhado para dispositivos móveis Android para garantir a mobilidade e integrada com o sistema de gestão por meio de um serviço *middleware* WCF. Terá a comunicação sem fios através de protocolos HTTP e TCP, em que o formato dos dados trocados é JSON.

3.1.2 Arquitectura do sistema

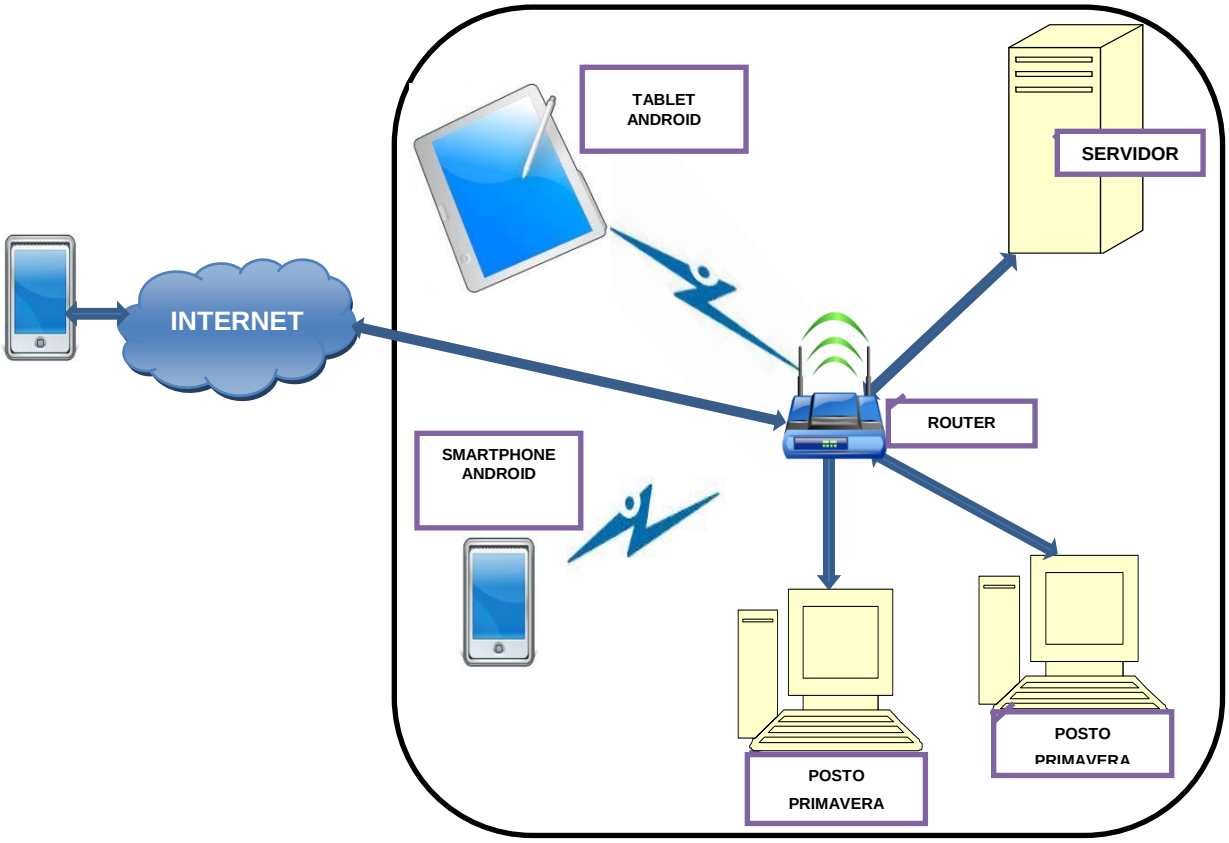

Figura 10 – Visão geral do sistema

Antes do desenvolvimento do sistema foi feito um levantamento e análise dos requisitos. Os requisitos funcionais e não funcionais que em seguida serão apresentados resultam da experiência do pesquisador em actividades de assistência técnica em sistemas de gestão numa série de empresas cabo-verdianas, bem como interacção com alguns *stakeholders* com realização de conversas informais, principalmente com operadores de logística com algum envolvimento no processo de contagem física. Nas conversas com os operadores foi dada alguma margem para que os envolvidos pudessem expor as suas ideias. E também, com base nos testes efectuados em alguns aplicativos existentes.

3.1.3 Requisitos funcionais

O sistema deve permitir:

- Que o utilizador efectue o Login sem a palavra-chave através do reconhecimento facial;
- A visualização de informação detalhada de um artigo após leitura do código de barras;
- Sincronização do SQLite com o sistema de gestão de base dados;
- Listar os armazéns em que a situação se encontra em contagem bem como os operadores responsáveis;
- Listar os artigos que estão em contagem, para um dado armazém;
- Inserir e alterar as informações da contagem física dos artigos marcados para contagem;

3.1.4 Requisitos não funcionais

Interfaces e Usabilidade

O objectivo é fornecer uma interacção pessoa-computador o mais amigável possível, intuitiva e natural. Para isso serão integradas as mais modernas formas de interacção com computador para que o utilizador possa trabalhar em qualquer lugar com o aplicativo:

- Reconhecimento de imagens – um dos *inputs* chave do sistema é o código de barras. Esta modalidade simula a visão humana em que o utilizador não tem que digitar o extenso número associado ao código de barras para procurar informação do artigo, basta focalizar a câmara fotográfica na imagem e é logo reconhecido o número associado.
- Síntese de voz- forma natural de comunicar com o utilizador os resultados através de voz, complementando assim o *output* visual, na forma de objectos gráficos.

- Reconhecimento escrita - com esta modalidade o utilizador tem a opção de inserir as quantidades contadas com um simples manuscrito no ecrã.
- Interface tangível – simplificar e tornar mais rápida a navegação no menu associado ao programa.
- GUI – interface bastante intuitiva e natural.

Quanto a usabilidade, o tempo necessário para aprender a usar o sistema será muito baixo. Este atributo será medido em termos da velocidade de aprendizagem, por exemplo, minutos de formação necessários até ser possível o utilizador usar de forma independente o sistema. Depois de o utilizador saber usar a aplicação, será registado o nº de erros cometidos pelo utilizador e medidos os níveis de eficiência para ver em que medida o operador aumentou a produtividade e comparar com os métodos tradicionais. Em seguida, será medido o nível de satisfação dos utilizadores com a realização de um pequeno inquérito aos utilizadores directos e registar a proporção dos que gostaram de trabalhar com o sistema.

Requisitos Software e Hardware

Requisito mínimo de plataforma: Android 2.0 - Froyo.

No intuito de atingir altos níveis de desempenho, o aplicativo Android requer dispositivos móveis com uma razoável capacidade de hardware, assim como requisito mínimo será:

- Processador 1 GHz;
- Ecrã táctil de 4’’e sensibilidade *touch capacitivo;*
- Memoria interna 1 GB e externa de 8 GB para base dados SQLite;
- Com ligação a rede sem fios;
- Resolução da câmara traseira 5 MP, focalização automática;
- Autonomia mínima de 4 horas, tempo necessário para execução das tarefas.

Os outros componentes físicos e lógicos do sistema requerem:

- Servidor dentro dos padrões normais, com no mínimo *Windows Server 2003* ;
- *Router* sem fios com capacidade de redireccionamento de portas e sincronização com serviço de IP dinâmico *Dyndns*.

Portabilidade e Comunicação

O protótipo deve ser desenvolvido para plataforma Android e a estrutura dos dados trocados com o serviço de *middleware* WCF é JSON. O protocolo de comunicação será HTTP, para casos de utilização fora da empresa, e TCP para utilização interna. Os dispositivos conectados nessa rede utilizarão o serviço de WCF instalado no servidor central.

Segurança, Concorrência e Desempenho

- O sistema deve incluir um procedimento de autorização de utilizadores, onde cada utilizador se deve identificar através do processo reconhecimento facial. Apenas os utilizadores autorizados desta forma podem aceder aos dados do sistema.
- Serão considerados cenários críticos que podem afectar a integridade dos dados na aplicação e no servidor SQL. Haverá um controlo de concorrência dos dados na óptica dos dispositivos para prevenir que um dispositivo sobrescreva as modificações que outros realizaram no mesmo registo da tabela, causando eventualmente em incoerência. Todas as linhas de inventário afectadas a um dispositivo são bloqueadas pelo mesmo no serviço WCF e nenhum mais pode realizar alterações até que o bloqueio (*lock*) seja liberado. O controlo de concorrência será implementado no serviço WCF.

- O sistema deve conseguir reconhecer pelo menos 9 códigos de barras dos 10 produtos disponibilizados para o teste.
- O protótipo deve sincronizar com servidor central em menos de 3 segundos.
- Dada a natureza do problema de pesquisa, o sistema em causa deve primar pelo desempenho que por vezes poderá afectar requisitos de fiabilidade e segurança. No entanto serão feitos esforços no sentido de garantir o quanto possível esses requisitos.

3.2 Ferramentas e recursos

Visual Studio 2010

Para a codificação do serviço WCF será utilizada a linguagem de programação C# com o IDE Visual Studio 2010 juntamente com o conjunto de ferramentas e bibliotecas para ambiente de desenvolvimento Microsoft, incluindo os padrões fornecidos pelo manual MSDN.

O serviço WCF será desenvolvido na plataforma .NET Framework 4.5, sendo assim um pré-requisito para a sua instalação e execução. Não será necessário o serviço IIS para o alojamento do serviço pelo que o mesmo estará alojado numa aplicação Windows que executa no *Windows Server 2008*.

Eclipse Juno e ADT *Plugin*

O Eclipse Juno é um IDE de código aberto que possui um ambiente adequado para desenvolvimento em Java. Os aplicativos Android são codificados em Java, pelo que será necessário integração do *plugin* ADT (*Android Developement Tools***)** que possui um ambiente específico para desenvolvimento de aplicações Android. É uma extensão as capacidades do Eclipse que permite:

- A criação e configuração de projectos android rapidamente;
- Criação de interfaces com componentes *Widgets* por meio da linguagem XML;
- Adicionar componentes com base na API Android;
- Integrar bibliotecas de código nativo em C e C++ com base na ferramenta Android NDK *(Native Developement Kit)*;
- Utilização de simuladores de dispositivos móveis AVD (*Android Virtual Device*);
- Instalação de ficheiros apk nos dispositivos reais com ferramenta ADB (*Android Debug Bridge*).

Morovia[14]

É um software disponivel online para criação de códigos de barras simples e fácil de usar. O software possui um pacote de 10 fontes e um kit de ferramentas de idiomas. Pode-se escolher diferentes tipos de altura para cada família de fonte. O utilizador tem duas opções de escolha: deixar cada codigo de barras com ou sem texto legível a humanos. O kit de ferramentas possui o FontPal**,** usado para calcular os dígitos e construir mapeamentos e seu código fonte em Visual Basic e C**.**

Ambas as ferramentas são utilizadas na confecção do sistema, pelo que será necessária realização de testes de funcionamento e familiarização.

3.3 Implementação do protótipo

3.3.1 Experimentação das ferramentas e bibliotecas

O desenvolvimento do projecto iniciou-se com a experimentação e avaliação das ferramentas, bem como das bibliotecas de leitura de código de barras, reconhecimento facial e reconhecimento de escrita existentes.

O objectivo principal nesta fase foi seleccionar uma biblioteca de código de barras que pudesse ser integrada na aplicação Android. O processo de avaliação dos sistemas foi levado a cabo do seguinte modo:

Durante o processo de revisão de literatura foram avaliados a robustez, precisão dos sistemas de leitura de código de barras, compatibilidade com os dispositivos móveis e sistema operativo Android, disponibilidade do código fonte, licença, grau de actividade da comunidade de desenvolvimento e documentação disponível. A precisão do reconhecedor constitui o requisito com maior expressão dada a natureza do problema do trabalho. Também foi importante no processo de escolha a facilidade de integração com aplicativos Android.

[14] http://www.morovia.com/free-online-barcode-generator

Optou-se pelo uso de tecnologias de código aberto, compatível com a linguagem de programação Java e cujo projecto de desenvolvimento estivesse activo e actualizado.
Dois sistemas foram avaliados com base nos critérios anteriores, *Zbar* e *ZXing* (ver na secção anterior: Ferramentas de Visão Computacional). Ao final, o *ZXing* foi seleccionado devido a facilidade de integração com o Android, disponibilidade do código fonte Java, ao contínuo desenvolvimento do projecto e a documentação disponível.

Em seguida procedeu-se para codificação do *middleware.*

3.3.2 Serviço WCF

Conforme já referenciado, o WCF é uma plataforma da Microsoft utilizado para comunicações entre sistemas. Foi utilizado neste trabalho como ponto de escala, *middleware*, para troca de dados entre a aplicação Android e o SQLServer, conforme a figura 10:

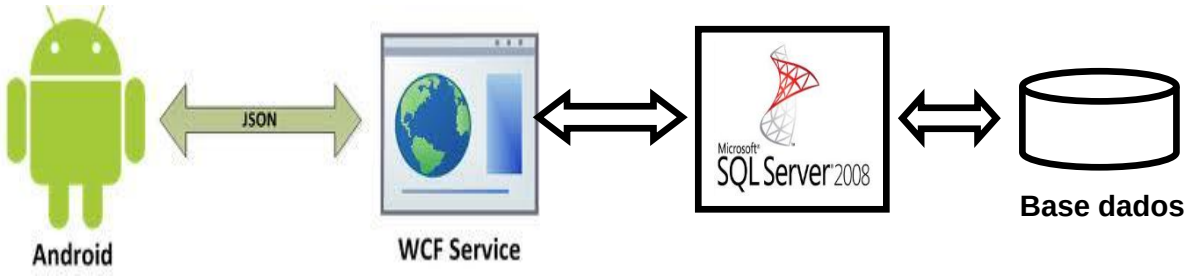

Figura 10-Interligação entre Android e o SqlServer

A codificação do serviço foi realizada na ferramenta Visual Studio 2010, linguagem C#, que já inclui alguns modelos de serviços WCF. O serviço possui dois *Endpoints*. Foram utilizados ambos os protocolos, HTTP e TCP, tomando como base os cenários corporativos em que o funcionário necessita de aceder ao serviço tanto de dentro como de fora da empresa. Assim tem-se:

1. **HTTP**- através do URL tem-se acesso ao serviço de fora da rede da empresa.
2. **TCP** - é acessível somente dentro da empresa. Este tipo de disponibilidade é mais eficiente que o HTTP, porém menos acessível já que fica restrito para a redes internas.

Contractos

Foi criada uma interface onde foram definidos os contractos entre o serviço e a aplicação Android que irá consumi-lo, expondo somente os métodos e dados desejados. O serviço WCF conta com os seguintes tipos de contractos:

- *Service Contract* - Um contrato para um serviço. Define os detalhes do serviço, e será utilizado na interface de contrato.

- *Operational Contract* - Define uma operação individual, e será aplicado na assinatura dos métodos da interface de contrato.

- *Data Contract* - Define a serialização para objectos complexos.

```
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Runtime.Serialization;

[ServiceContract]
public interface Interface1
{

[OperationContract]
[WebInvoke(Method = "GET",
BodyStyle = WebMessageBodyStyle.WrappedRequest, UriTemplate = "getLinha/{idCabec}")]
String GetLinhaInventario(string idCabec);

[OperationContract]
[WebGet(UriTemplate = "GetCabec", BodyStyle = WebMessageBodyStyle.WrappedRequest,
ResponseFormat = WebMessageFormat.Json, RequestFormat = WebMessageFormat.Json)]
String GetCabecInventario();

[OperationContract]
[WebGet(UriTemplate = "GetArtigo", BodyStyle = WebMessageBodyStyle.WrappedRequest,
ResponseFormat = WebMessageFormat.Json, RequestFormat = WebMessageFormat.Json)]
String GetArtigo();
```

```
[OperationContract]
[WebGet(UriTemplate = "GetUser/{username}", BodyStyle = WebMessageBodyStyle.WrappedRequest,
ResponseFormat = WebMessageFormat.Json, RequestFormat = WebMessageFormat.Json)]

User GetUser(string username);
}
```

Quadro 3-Declaração da Interface WCF

Esta propriedade necessita da inclusão das bibliotecas "System.Runtime.Serialization", "System.ServiceModel".

Implementação dos métodos foi na classe *ServiceWCF* que implementa a interface acima. Esta classe é responsável pela ligação à base de dados e execução dos comandos SQL para efeitos de sincronização e conversão dos dados das tabelas para o formato JSON e vice-versa.

Hospedagem

Quanto a hospedagem do serviço, criou-se uma aplicação Windows para efeitos de hospedagem, conforme ilustrado na porção de código abaixo (quadro 5):

```
public ServiceHost host;
      Uri baseAddress = new Uri("http://localhost:1977/servicoWcf/");



 ServiceHost host= (new ServiceHost(typeof(ServiceWcf),baseAddress,baseAddress2));


         host.Open();
```

Quadro 4-Hospedagem do serviço WCF

Caso fosse no paradigma de Serviços Web haveria necessidade de instalar o IIS, mas neste caso, neste ambiente escolhido, não.

Configuração do serviço

O WCF conta com dois tipos de configuração, sendo o modo imperativo criado no próprio código e o modo declarativo criado num ficheiro XML. Foi utilizado o modo declarativo para evitar a recompilação do código sempre que há mudanças de configuração. Em seguida é ilustrado no quadro 6 o ficheiro XML devidamente configurado:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>

 <connectionStrings>
  <add
   name="conexaoSQL"
   connectionString="Data Source=TOSHIBA-PC\sqlexpressr2;Initial Catalog=PRIDEMO;Persist Security Info=True;User ID=sa;Password=benvindo1977" />
 </connectionStrings>


  <system.serviceModel>

   <bindings>
    <netTcpBinding>
     <binding name="NewBinding0" portSharingEnabled="true">
      <security mode="None">
       <transport clientCredentialType="Windows" />
       <message clientCredentialType="Windows" />
      </security>
     </binding>
    </netTcpBinding>
    <webHttpBinding>
     <binding name="NewBinding1">
      <security>
       <transport clientCredentialType="Windows" proxyCredentialType="Windows" />
      </security>
     </binding>
    </webHttpBinding>
```

```
    </bindings>
    <serviceHostingEnvironment
      multipleSiteBindingsEnabled="True"/>


      <behaviors>
        <endpointBehaviors>
          <behavior name="NewBehavior0">
            <webHttp defaultOutgoingResponseFormat="Json" />
          </behavior>
        </endpointBehaviors>
        <serviceBehaviors>
          <behavior name="compsimples">
            <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
            <serviceDebug includeExceptionDetailInFaults="false" />
          </behavior>
        </serviceBehaviors>
      </behaviors>
      <services>
        <service behaviorConfiguration="compsimples" name="WindowsFormsApplication1.ServiceWcf">
          <endpoint address="http://localhost:1977/servicoWcf" behaviorConfiguration="NewBehavior0"
            binding="webHttpBinding" bindingConfiguration="NewBinding1"
            name="httpendpoint" contract="WindowsFormsApplication1.Interface1" />

          <endpoint address="net.tcp://localhost:1978/servicoWcf/" behaviorConfiguration=""
            binding="netTcpBinding" bindingConfiguration="NewBinding0"
            name="tcpendpoint" contract="WindowsFormsApplication1.Interface1" />
        </service>
      </services>
    </system.serviceModel>
</configuration>
```

Quadro 5-Configurações do serviço WCF

Neste ficheiro encontram-se os parâmetros da instância da base de dados do sistema que são necessários para estabelecer a ligação. Caso exista necessidade de redireccionar o serviço para outra instância, basta mudar os parâmetros neste ficheiro sem que haja necessidade de recompilar.

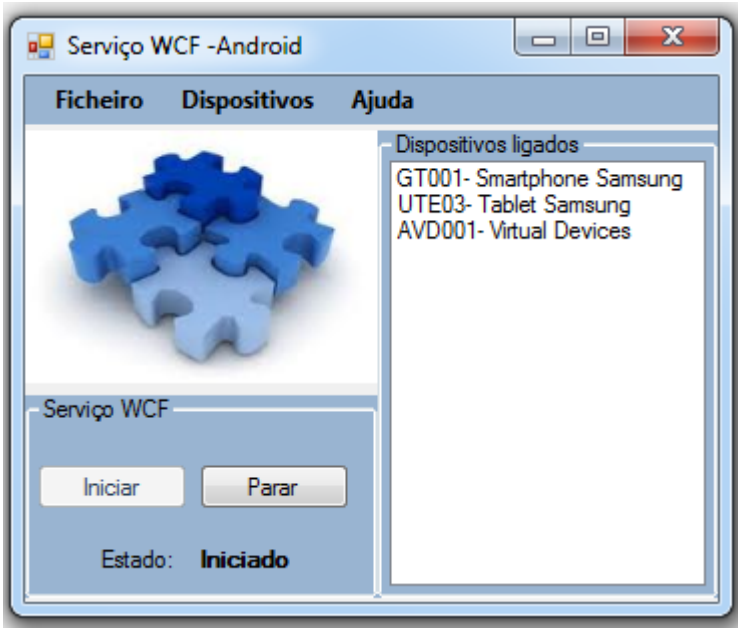

Figura 11-Servico WCF hospedado numa Aplicação Windows

3.3.3 Sincronização de base de dados

O sistema de gestão de base de dados para o Android é o *SQLite* e para o Sistema Gestão Primavera é o *SQLServer* e poderão sincronizar-se de forma:

- Automática- É especificado o intervalo de tempo para sincronizar e é aplicado no caso de o dispositivo estar ligado à internet. Possui como vantagem o facto de disponibilizar em tempo real o desenrolar das actividades de contagem aos operadores dos postos trabalho fixo e de garantir cópia de segurança dos dados do *SQLite*;
- Manual - É usada em casos de impossibilidade de acesso a rede, por vezes em locais fora do alcance de rede sem fios. Nestes casos, recomenda-se que a sincronização seja feita nas instalações antes da deslocação ao ponto de acesso fraco, ao fim do processo contagem faz-se a sincronização manual assim que estiver ligado à rede.

No ERP Primavera as tabelas envolvidas no processo de contagem física são:
CabecInventario – guarda informação dos cabeçalhos dos documentos de inventário. Associa-se a uma ou mais linhas de inventário;
linhasInventario - guarda informação das linhas dos documentos de inventário;

Artigo – possui informação referente a cada artigo, inclusive o número código de barras associado ao artigo, as suas existências e quantidade em stock.

Para efeitos de sincronização, as tabelas foram criadas na base de dados da aplicação Android com mesma estrutura e tipos de dados através da classe:

InventarioSqlHelper - A classe estende *android.database.sqlite.SQLiteOpenHelper* e é responsável pela criação, abertura e gestão da base de dados *Sqlite*. O objecto da classe é criado no momento da inicialização da actividade principal da aplicação Android.

O carregamento dos dados

A aplicação Android, através do pacote *org.apache.http*, solicita a consumo dos dados das tabelas por via dos métodos disponibilizados pelo serviço WCF. Um exemplo no quadro 7:

```
HttpGet httpGet3 = new HttpGet(HTTPADRESS+"GetArtigo");
```

Quadro 6- Chamada via HTTP do serviço

Em que, *HTTPADRESS* é o endereço do serviço e este inclui o nome endereço de IP do servidor, nome do serviço WCF e a porta. Devido ao elevado custo de aquisição de um IP público optou-se pelo serviço de IP dinâmico *Dyndns*. Exemplo:

http://silmaccv.dydns.org/servicoWCF:1977/getArtigo

Após a execução deste pedido é devolvida uma lista de artigos, em formato JSON.

JSON

```
{
"Artigos": [
{
"codigo": "A001",
"descricao": "Portatil Hp 2Gh"
},
{
"codigo": "A002",
"descricao": "mesa de computador "
}]}
```

Quadro 7-Resultado em formato JSON

O quadro 8 ilustra um exemplo da conversão da tabela *Artigo* para formato JSON, feita pelo conversor JSON do serviço WCF.
Sendo a resposta do serviço WCF em formato JSON, do outro lado da aplicação Android houve necessidade de incluir o pacote *org.json.JSONObject* no projecto para a operação inversa, isto é, percorre o objecto e para cada linha ou item são extraídos os valores dos atributos que posteriormente serão inseridos através de comandos SQL na base de dados *SQLite*.

Controlo de Concorrência

O serviço WCF guarda informações de segurança relativa as permissões de escrita e leitura nos documentos afectos aos dispositivos. Os documentos disponíveis pelo software de gestão para contagem física podem estar 3 estados: bloqueado para edição; disponível; somente leitura.

3.3.4 Integração do leitor código de barras

Para a leitura do código de barras associado aos artigos recorreu-se à biblioteca *ZXing*. Possui duas formas de integração. Pode ser incluída directamente no projecto Android e com isso não há necessidade de instalar um leitor de código de barras para que a aplicação funcione. Outra forma, a que foi utilizada no presente protótipo, é a instalação de um aplicativo nativo e exclusivo para leitura de código de barras, *Barcode Scanner*.
Foram descarregados dois ficheiros do *site* da ferramenta. Um deles é a biblioteca ZXing 2.2 já actualizada e outro foi um projecto exemplo onde foi possível integrar a funcionalidade de forma rápida.
Foi criado um ficheiro XML para representação da leitura, colocou-se a biblioteca do ZXing no projecto e fez-se a chamada do leitor via *Intent* a partir actividade principal. As bibliotecas do *ZXing* (core.jar e ZXingjar.jar) foram colocadas dentro do projecto Android.
A chamada do leitor de código de barras foi por via *Intent*, como se pode ver no quadro 9:

```
Intent intent = new Intent("com.google.ZXing.client.android.SCAN");
        startActivity(intent);
```

Quadro 8-Chamada do leitor de código de barras

E o resultado retornado por meio de:

```
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
   IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);
   if (result != null) {
     String contents = result.getContents();
     if (contents != null) {
       showDialog(R.string.result_succeeded, result.toString());
     } else {
       showDialog(R.string.result_failed,
     }
   }
 }
```

Quadro 9- Resultado retornado pelo onActivityResult

Cabo realçar que antes da execução foi necessário modificar o android manifest, ficheiro de configurações gerais do projecto, para o funcionamento correcto.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Quadro 10-Permissão de utilização da câmara

No quadro 11 ilustra como a declaração é feita para permitir que o *Barcode Scanner* aceda a câmara do dispositivo.

Integração do reconhecimento facial

Foi utilizada um processo de autenticação do utilizador no sistema sem a palavra-chave, com recurso a ferramenta de visão computacional *OpenCV*, utilizada para detecção e reconhecimento de faces. A biblioteca é escrita na linguagem de programação C++, e para sua reutilização no aplicativo android, escrito em Java, foi necessária a compilação com a ferramenta Android NDK.

Para finalizar a implementação, na figura 11 que se segue mostra a arquitectura em que o sistema foi concebido. É apresentada a visão interna dos principais componentes do sistema, incluindo uma lista de módulos chave que demonstram o funcionamento da aplicação e a forma como estes módulos comunicam entre si e com o serviço WCF.

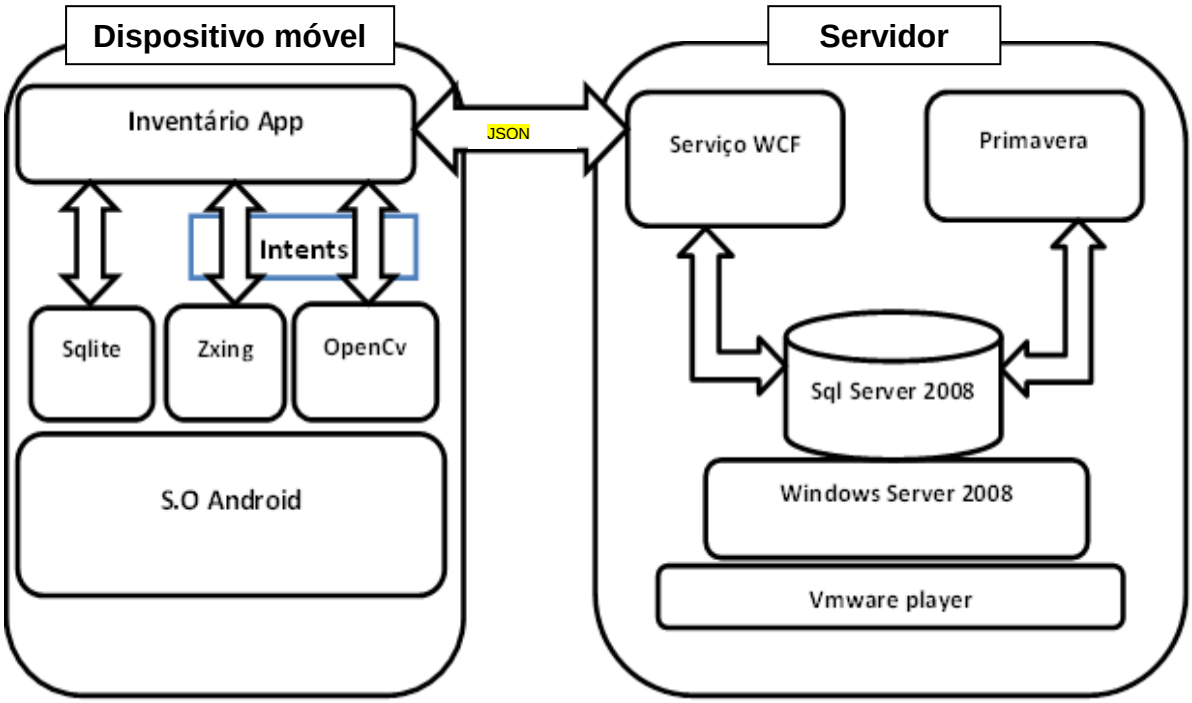

Figura 12-Visão interna dos componentes do sistema

Capítulo 4

4 Testes e Avaliação do modelo

Após a codificação do protótipo procedeu-se para a instalação, testes e avaliação do mesmo. Testes foram levados a cabo com o Software de gestão Primavera Profissional. É um software genérico, robusto desenvolvido de pela empresa Primavera Business Software Solutions [15] com sede na cidade de Braga - Portugal. Optou-se pelo software pelo facto de estar em expansão no mercado PALOP[16] sobretudo as pequenas e médias empresas com actividades ligados ao sector comercial e industrial.

4.1- Preparação do cenário de testes

Antes dos testes, foi necessária a instalação e configuração de um servidor virtual com Windows Server 2008 sobre a plataforma de virtualização VMWARE[17]. Sobre o mesmo servidor instalou-se o software de gestão Primavera versão 7.50, juntamente com o *SQLServer 2008 R2* e o Serviço WCF. Após a instalação do software de gestão Primavera, inicializou-se a plataforma para servidor de base de dados *SQLServer* e foram criados os dados de demonstração para teste do sistema inventário, relativos à empresa, perfis de utilizador e respectivas permissões. No mesmo módulo criaram-se as fichas de artigos e os armazéns e lançou-se o Stock inicial de cada artigo em cada armazém. Importa realçar a obrigatoriedade do preenchimento de um dos atributos chave, o código de barras associado ao artigo para efeitos de leitura no dispositivo móvel.

[15] www.primaverabss.com
[16] Países Africanos de Língua Oficial Portuguesa
[17] *www.**vmware**.com*

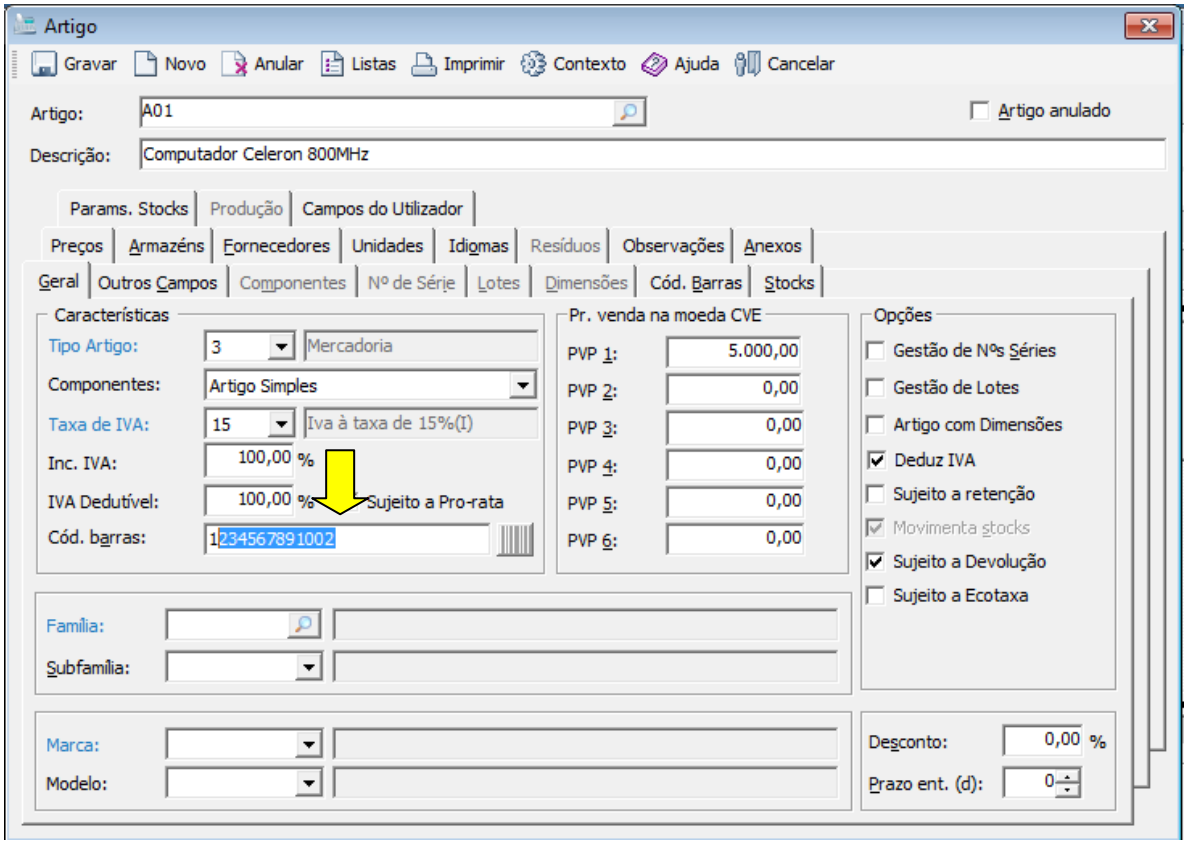

Figura 13-Primavera - Ficha de artigo

A figura 12 exemplifica a criação da ficha de um artigo do tipo mercadoria e cujo código é A01 e o preço PVP é 5000 escudos. A figura 13 mostra a imagem do código de barras associado fisicamente ao artigo.

Figura 14-Código de barras de um artigo

Para cada um dos artigos criados foram gerados os códigos de barras no formato EAN13 conforme a figura 13, com recurso à aplicação Morovia, e em seguida foram impressos e estampados nos produtos marcados para testes, conforme ilustrado na figura 14.

Figura 15-produtos com código de barras estampados

Figura 16-leitura de um código de barras de um produto

Fez-se a movimentação dos artigos com operações correntes, saída, transferência e entrada de stock.

Ao fim da simulação de uma série de operações, para efeitos de contagem física, primeiramente houve uma preparação do inventário para que pudesse gerar o documento de inventário que inclui os artigos marcados para a contagem e com stock actual, conforme a figura 16 do editor de Contagem do Inventário Primavera.

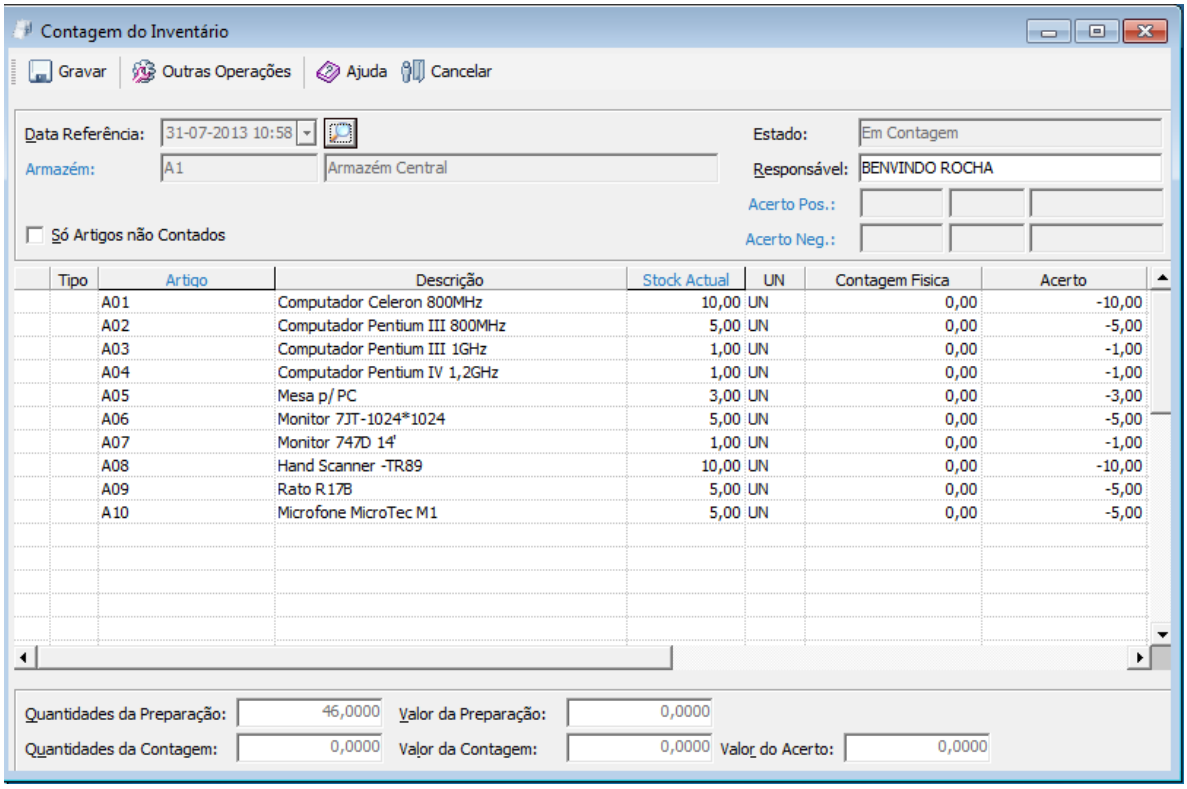

Figura 17-Primavera-Preparação para contagem física

A figura 16 ilustra o documento de inventário pronto para a contagem física. Contém informação do nome do armazém, o operador responsável, os artigos marcados para contagem e um campo específico para introduzir as quantidades, que neste caso são introduzidas no aplicativo Android. A informação contida no editor acima foi carregada na aplicação do dispositivo móvel.

Os códigos de barras foram gerados no programa Morovia, impressos e recortados em tamanho 1000X1500 milímetros. Em seguida estampados em objectos, simulando assim um produto com código de barras.

Todos os testes foram realizados em dispositivos virtuais e reais. A aplicação foi instalada no *smartphone Samsung galaxy young duo* com processador de 1Gz, câmara fotográfica de 3.2 MP e com ecrã de 3.4’’. Em seguida, os testes foram repetidos num tablet Samsung.

Figura 18-Dispositivos móveis, tablet e smartphone

Assim, após a preparação do cenário procedeu-se com realização dos testes descritos na secção seguinte.

4.2 Apresentação e análise dos resultados

4.2.1 Precisão do reconhecedor

Primeiro foram realizados testes para verificar o quanto preciso é o reconhecedor de código de barras *ZXing*. Foram seleccionados 10 produtos com código de barras e para cada um procedeu-se com a leitura por 10 vezes em situações de diferente luminosidade, distância focal e rotatividade. Ao fim foram aplicados alguma distorção nos códigos de barras para simular a degradação da imagem com o tempo. Eis o quadro 11 que ilustra os resultados:

Artigo	Nº falhas	Nº sucesso	Precisão %
Luminosidade normal (dia)	**1**	**9**	**90%**
Luminosidade fraca (noite)	**3**	**7**	**70%**
Distância focal 10 cms	**0**	**10**	**100%**
Distância focal 15 cms	**1**	**9**	**90%**
Rotatividade 90º	**1**	**9**	**90%**
Rotatividade 360º	**2**	**8**	**80%**
Curvatura	**10**	**0**	**0%**
Desgasto com tempo	**3**	**7**	**70%**

Quadro 11-Medição da precisão

Os resultados ora apresentados no quadro 11 demonstram que:

- Aumentando o valor eixo x (largura), diminui a distância focal necessária entre o sensor e o código de barras;
- Ao diminuir a luminosidade reduz a precisão;
- Aumentando curvatura da imagem de código de barras reduz-se a precisão. Ou seja, o insucesso aumenta quando o código de barras é estampado em objectos com formas cilíndricas, mesmo com aumento do tamanho de código de barras;
- Não há variação na precisão com a rotação da imagem de código de barras;
- Não existe muita diferença no que toca à altura da imagem, sendo que não reconhece abaixo de 2 milímetros de altura.

4.2.2 Velocidade de sincronização e pesquisa

Foram empregados alguns testes para verificar a velocidade de comunicação com o servidor WCF usando o formato de dados JSON. Para este tipo de teste foram gerados de forma automática uma série de fichas de artigos e em seguida a criação de documentos de inventário com diferentes quantidades de artigos. O quadro 12 que se segue demonstra os resultados obtidos de cada documento na operação de maior tráfego, sincronização manual.

Documento	**Nº Artigos**	***Download* (Tempo)**	***Upload* (Tempo)**
Nº1	10	1	1
Nº2	20	1	1
Nº3	100	3	2
Nº4	150	4	2

Quadro 12-Medição da velocidade de sincronização(em segundos)

O quadro 12 apresenta tempo que leva ao carregamento das linhas de um documento no dispositivo e para o posterior envio das mesmas ao servidor após encerramento da contagem. O tempo degrada com o aumento do número de linhas de artigos, sendo que o tempo de *upload* ligeiramente mais baixo pelo facto de no envio constar uma quantidade reduzida de campos de informação (identificadores da linhas e as quantidades inseridas).

4.2.3 Situações de Concorrência

Durante a execução dos testes foi validada a eficácia do protótipo em situações de concorrência. Foram disponibilizados dois documentos de inventário para um *smartphone*, um *tablet* e um dispositivo virtual. Os resultados foram conforme esperado, o primeiro a solicitar um documento é-lhe atribuído as permissões de escrita e leitura enquanto fica somente de leitura para os próximos.

4.2.4 Produtividade

Foram realizados testes para comparar em termos de produtividade, medindo o tempo necessário para a contagem dos 10 artigos desde da preparação até o encerramento do inventário no sistema Primavera. Foram seleccionados 3 operadores com algum envolvimento no processo de contagem física e os resultados obtidos são exibidos no quadro 13.

	Produtividade (minutos)	
Operadores	**Usando método tradicional**	**Usando o protótipo**
Operador 1	10	2
Operador 2	11	3
Operador 3	9	2

Quadro 13- Comparação de metodologias de contagem física

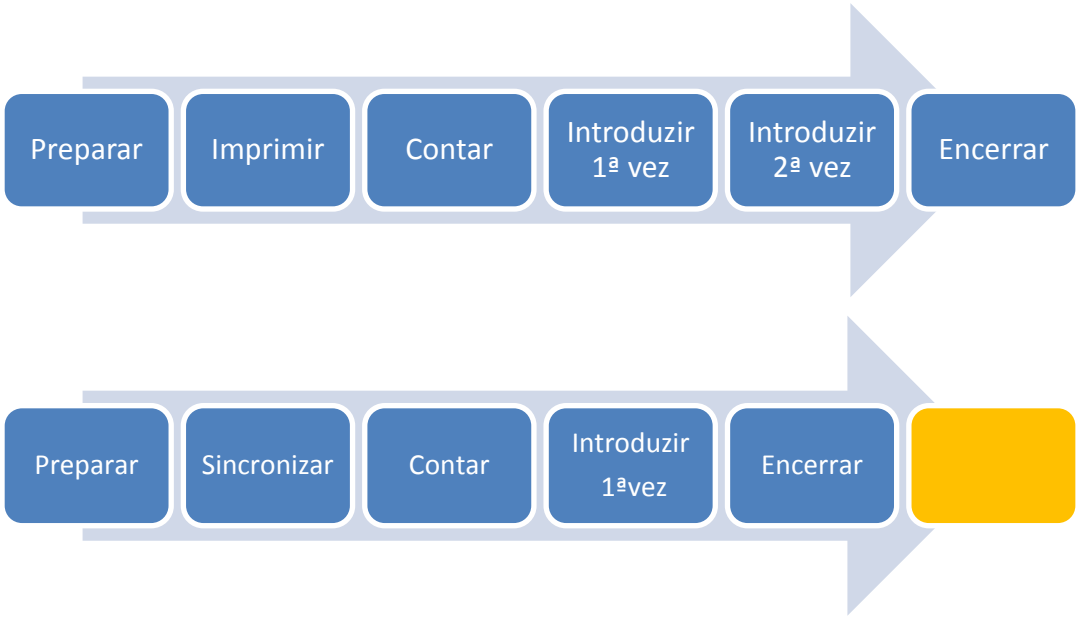

Figura 19-Cronograma de comparação

A figura 19 ilustra uma comparação em termos de passos necessários para completar o processo de contagem física. O processo começa num posto fixo do sistema Primavera com a operação comum de preparação do inventário conforme ilustrado na figura 19 - preparação inventário.

A etapa seguinte difere no modo de aquisição da lista de artigos por parte do operador. No modo tradicional ela é impressa, enquanto no protótipo basta sincronizar com base de dados que a lista fica disponível na aplicação Android.

Procede-se com contagem dos itens e a respectiva introdução das quantidades encontradas. No modo tradicional a 1ª introdução é feita em folhas de papel (quadro 14) que, ao fim da contagem, terá de ser introduzida no sistema Primavera, perfazendo assim a dupla introdução. Enquanto no modelo desenvolvido neste trabalho, a primeira e única introdução é feita no

dispositivo móvel com auxílio do teclado numérico virtual. Com a sincronização, a informação fica de imediato disponível no repositório do sistema Primavera, em tempo real dependente da ligação à rede. Finalmente, a operação comum de encerramento de inventário é executada no Primavera que regulariza o Stock dos artigos e liberta-os para as operações de correntes. É evidente o tempo gasto na execução das operações é maior no método tradicional comprovando assim a eficiência no processo com auxílio do protótipo desenvolvido neste trabalho.

Empresa de Teste - Dissertação de Mestrado

Preparação de Inventário

Data Emissão: 13-08-2013 Pág. 1

Artigo	Descrição	Stock Actual	Contagem Fisica	Observações
A01	Computador Celeron 800MHz	10,00	10	
A02	Computador Pentium III 800MHz	5,00	6	
A03	Computador Pentium III 1GHz	1,00	2	
A04	Computador Pentium IV 1,2GHz	3,00	3	
A05	Mesa p/ PC	5,00	5	
A06	Monitor 7JT-1024*1024	0,00	1	
A07	Monitor 747D 14'	5,00	5	
A08	Hand Scanner -TR89	6,00	7	
A09	Rato R17B	6,00	6	
A10	Microfone MicroTec M1	6,00	6	

Quadro 14-Metodologia tradicional de contagem física.

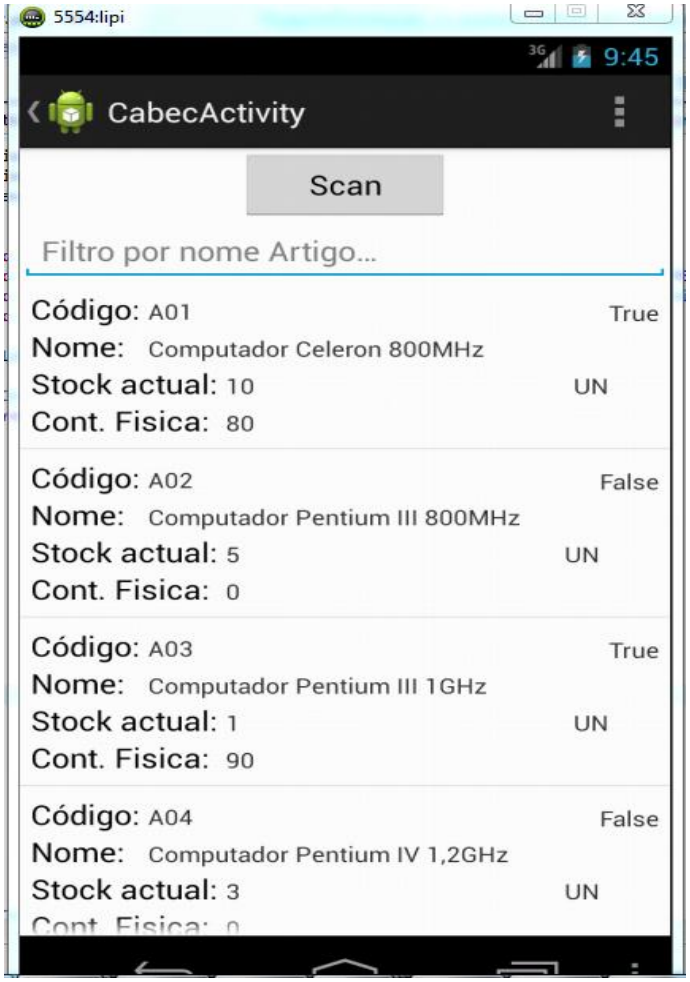

Figura 20-Contagem física no aplicativo android

Ganho de Tempo e Dinheiro

Mediante esses dados pode-se simular quanto é o ganho em tempo e dinheiro com a utilização deste protótipo, por exemplo, num estabelecimento comercial de venda a retalho.

Ora, sabendo que em média:

Protótipo possui velocidade de contagem= 10 itens/ 3 minutos;

Na loja o volume venda por hora = 10000 $/ 60 minutos.

Pretende-se calcular o ganho de tempo e dinheiro, sabendo que a loja contém 5000 produtos de várias famílias. O inventário físico é semestralmente implicando o encerramento para contagem de 1,5 dias (2160 minutos).

Ora, calculando o tempo necessário para efectuar a contagem física de 5000 itens usando o protótipo:

$$\frac{10\ itens \rightarrow 3\ min}{5000\ itens \rightarrow\ \mu\ min} \quad \rightarrow \mu = 1500\ min$$

O tempo ganho será:

$$2160 - 1500\ = 660\ min$$

Implica assim o ganho em dinheiro de:

$$\frac{60\ minutos \rightarrow 10000\ \$}{660\ minutos \rightarrow\ \mu\ \$} \rightarrow \mu = 110000\ \$$$

Portanto, usando o protótipo no estabelecimento comercial, estima-se que ganho em tempo será de 660 minutos (8 horas) e que o ganho em dinheiro será de 110000 $.

4.2.5 Usabilidade

Os mesmos operadores/utilizadores que foram seleccionados para realização dos testes de velocidade de contagem participaram nos testes de usabilidade e os resultados encontram-se no quadro seguinte:

	Usabilidade		
Operadores	**Velocidade de aprendizagem**	**Nº erros cometidos**	**Nível satisfação (0-5)**
Operador 1	10	1	4
Operador 2	13	1	4
Operador 3	5	2	4

Quadro 15- Usabilidade do sistema.

O tempo médio para consolidação da utilização do sistema (velocidade de aprendizagem) é relativamente baixo, à volta dos 9 minutos. Embora em ambos os casos tenham completado as tarefas, o número de erros cometidos situou-se à volta de 1,5 nº de falhas cometidas, isto é,

pedidos de assistência técnica. Pode-se constatar que os operadores que mais tempo dispensaram na aprendizagem, menos erros cometeram.

Ambos os operadores alegaram da importância da mobilidade no processo de contagem física e também da aquisição de informações detalhadas do produto. Indagados sobre a necessidade de integração do comando por voz e reconhecimento de manuscrito, ambos acharam pertinentes essas novas modalidades de interacção. Alguns apresentaram sugestões na colocação dos *widgets* por forma a facilitar a navegação.

De modo geral, os operadores fizeram uma boa avaliação do modelo concebido e são unânimes nos benefícios que o mesmo pode acarretar para uma organização.

4.3 - Comparação com trabalhos relacionados

O quadro ilustra uma comparação com os trabalhos relacionados com base nalguns requisitos.

	Trabalhos			
Requisitos	**Protótipo**	**Atendente Móvel**	**VisioScan**	**Vendas a retalho**
Reconhecimento	ZXing	ZXing	Redes Neuronais	ZXing
Middleware	Serviço WCF	Serviço Web	Serviço Web	Serviço Web
Sincronizado	Sim	Não	Não	Não
Comunicação	TCP e HTTP	HTTP	HTTP	HTTP
S.O Móvel	Android	Windows Mobile	Symbian	Android
Formato Dados	JSON	XML	XML	XML

Quadro 16-Comparação com trabalhos relacionados

Para análise comparativa com os trabalhos relacionados, foram utilizados requisitos funcionais e não funcionais considerados fundamentais num sistema com arquitectura distribuída. Foi detectado que a ferramenta ZXing tem sido usada em todas as soluções que implicam de reconhecimento de código de barras, talvez pela facilidade da integração, exceptuando o VisioScan que aplicou redes neuronais. O protótipo desenvolvido neste trabalho marca diferença no que toca ao serviço *middleware* e no formato dos dados trocados.

A maioria das outras soluções tem optado pelos Serviços Web e o formato de dados XML, enquanto neste trabalho escolheu-se a tecnologia desenvolvida pela Microsoft WCF especificamente desenhada para integração de sistemas de diferente plataforma e com formato de dados mais leve e optimizado JSON.

Capítulo 5

5 Conclusões

Subsequente à apresentação e discussão dos resultados obtidos, neste capítulo fazem-se as considerações finais do trabalho, apontando as principais limitações encontradas ao longo da sua execução e os pontos de desenvolvimento futuro de investigação que permitirão complementar o trabalho apresentado.

5.1 Objectivos alcançados

A propagação do uso de dispositivos móveis com maior poder computacional abriu uma nova gama de aplicações de visão computacional. No presente trabalho fez-se uma aplicação nos processos de inventário, nomeadamente contagem física para solucionar o problema de ineficiência dos métodos de tradicionais de contagem.

Neste contexto foi proposto um sistema de inventário baseado em visão computacional e com arquitectura distribuída. O trabalho tinha como principal objectivo o desenvolvimento de um aplicativo protótipo para dispositivos móveis integrado com software de gestão mais precisamente módulo inventário, para que pudesse agilizar o processo de contagem física usando sistema de processamento de imagens de código de barras. Outros objectivos eram avaliar os níveis de desempenho, precisão e usabilidade do aplicativo, bem como efectuar uma análise comparativa com os métodos tradicional de contagem.

Os objectivos deste trabalho foram alcançados com resultados satisfatórios, pois foi possível a implementação de quase todos os requisitos planeados. Foi levado a cabo uma investigação exaustiva por forma a escolher a mais moderna tecnologia: plataforma para dispositivo móvel; leitor de código de barras e de comunicação de sistemas. Ao fim, optou-se pela plataforma Android em que foi integrado o leitor de código de barras ZXing e com um serviço de *middleware* WCF e o formato de dados JSON para cuidar da comunicação entre os sistemas.

Verificou-se a robustez do aplicativo Android como cliente numa arquitectura distribuída. A velocidade das comunicações com o serviço WCF apresentou resultados aceitáveis graças à

representação compacta no formato JSON. Ficou dentro dos limites tempo especificados para o sistema.

A precisão do leitor de código de barras apresentou resultados gerais na ordem dos 95%, indicando a viabilidade do sistema. Os problemas apresentados pelo aplicativo ocorreram principalmente nas imagens de código de barras estampados em produtos cilíndricos e em condições de fraca iluminação, constituindo assim factores que influenciam o desempenho do sistema.

Em termos de usabilidade, apresentou-se uma interface simples e bastante amigável em que nos testes submetidos, apresentou um tempo de aprendizagem bastante reduzido e, consequentemente, alguma satisfação dos utilizadores.

É explícito o contributo do trabalho de investigação ora apresentado para a comunidade empresarial. Da análise comparativa com os métodos tradicionais, ficou provado o quanto é o ganho de tempo e dinheiro para a organização com a utilização do sistema de inventário baseado na visão computacional. Contribui para extensibilidade dos softwares de gestão, sobretudo nas tarefas de campo como a contagem física, quebrando assim a dependência dos computadores convencionais.

Do ponto vista criativo, o trabalho mostrou-se bastante inovador no mercado a que foi projectado devido a combinação de tecnologias modernas, pois não foi possível encontrar aplicações com os mesmos requisitos.

5.2 Principais limitações

Sentiu-se necessidade de efectuar os experimentos de precisão do reconhecedor num dispositivo com uma câmara com capacidade superior ao utilizado nos testes (3.2 MP) para verificar se o problema da curvatura das imagens de código de barras dependia da resolução da câmara fotográfica.

O maior e último dos constrangimentos encontrados foi com relação ao servidor WCF. Possui altos níveis de parametrização. Este servidor disponibiliza os serviços por via dos contractos. Foram criados e disponibilizados tanto os métodos HTTP do tipo GET como POST. Não houve grandes dificuldades nas consultas para os serviços GET, mas as inserções com métodos POST não tinham o mesmo resultado, gerando um erro na passagem de parâmetro

para o método. Ao fim, foi detectado que o problema residia no formato de dados que não correspondia ao formato JSON.

5.3 Trabalhos futuros

O protótipo teve resultados satisfatórios, mas para torná-lo um sistema robusto é necessário continuar o desenvolvimento do mesmo. Logo, sugere-se a continuidade na integração de interfaces multimodais como trabalho de extensão:

- Reconhecimento integral do produto, bem como a identificação de certas características como a cor, tamanho, etc.
- Reconhecimento facial para a autenticação do utilizador.
- Integração de comandos voz para optimizar a navegabilidade, como por exemplo a chamada por comando voz do leitor de imagem de código de barras e de pesquisa de informações dos artigos com base na descrição.
- Reconhecimento de manuscritos para introdução das quantidades de forma natural.

Relativamente as comunicação sugere-se a optimização em termos de velocidade das transferências de dados entre o dispositivo e o serviço *middleware*.

Por fim, considera-se importante um estudo da portabilidade de aplicativos Android para diferentes plataformas como Windows Mobile, iOS.

Referências

A9. (s.d.). *Visual Search*. Obtido em 31 de 5 de 2013, de a9: http://www.a9.com/whatwedo/visual-search/

Barker, R. (12 de 5 de 2013). *Captação de imagens 3D sem camera*. Obtido em 7 de 7 de 2013, de University Of Glascow: http://www.gla.ac.uk/news/headline_277930_en.html

Boofcv.org. (6 de 6 de 2013). *open source Java library*. Obtido em 28 de 6 de 2013, de Boofcv: http://boofcv.org/

ComputerVisionOnline. (2013). *Computer Vision Online*. Obtido em 25 de 6 de 2013

Conte, M., Mattos, M., Nicoleit, E., & Lazzari, R. (2008). *Reconhecimento de padrões de Código de Barras a partir Redes Neuronais.* Brasil: Universidade do Extremo Sul Catarinense.

DAVIES, E. R. (2012). *Computer and Machine Vision: Theory, Algorithms,Practicalities.* University of London, Egham, Surrey, UK: Elsevier Inc.

Ellips. (7 de 7 de 2011). *Vision systems for inspecting and grading fruits and vegetables*. Obtido em 6 de 5 de 2013, de http://www.ellips.nl/: http://www.ellips.nl/

Flanagan, D. (1996). *Java in a Nutshell.* USA: O´Reilly & Associates.

Forsyth, D. A. (2012). *COMPUTER VISION-A MODERN APPROACH.* University of Illinois at Urbana-Champaign: Pearson Education.

George Coulouris, J. D. (2011). *Distributed Systems - Concepts & Design.* 5th Edition, Addison Wesley,.

GooglePlay. (2011). *Aplicaçoes Android*. Obtido em 1 de 2 de 2013, de GooglePlay: https://play.google.com/store

Infainmon. (7 de 2010). *Sherlock- Machine Vision Software*. Obtido em 3 de 2 de 2012, de www.infainmon.com: Manual de praticas de visão por computador

KOPIETZ, I. (2011). *Automação de vendas no varejo: Comparando RFID x Decodificação de imagens.* Brasil:curitiba: UNIVERSIDADE POSITIVO.

Meier, R. (2010). *Professional Android Application Development.* USA: Indianalopis: Wiley publishing Inc.

Microsoft. (1 de 1 de 2013). *Windows Communication Foundation*. Obtido em 6 de 6 de 2013, de msdn: http://msdn.microsoft.com/

Milano, D., & Honorato, L. B. (2010). *VISÃO COMPUTACIONAL.* Brasil: Universidad Estadual de Campinas.

Mittal, P. (1 de 11 de 2012). *Article: diference betwen GSON & XML*. Obtido em 3 de 6 de 2013, de codebucket: http://codebucket.co.in/which-one-is-better-xml-or-json/

Monteiro, C. (2009). *Metodologia de investigação Cientifica.* Brasil.

Nice, K., Wilson, T. V., & Gurevich, G. (1 de 1 de 2012). *How Digital Cameras Work*. Obtido em 6 de 6 de 2013, de electronics.howstuffworks.com: http://electronics.howstuffworks.com/cameras-photography/digital/digital-camera2.htm

OpenCV.org. (1 de 1 de 2013). *Open Source Computer Vision*. Obtido em 4 de 6 de 2013, de OpenCV: http://opencv.org/

OpenTL. (2011). *Visual Tracking*. Obtido em 2 de 2 de 2013, de OpenTL: http://www.opentl.org/library.html

Primavera. (1 de 1 de 2013). *Primavera Knowledge Base*. Obtido em 12 de 12 de 2012, de primaverabss: http://www.primaverabss.com/pkb

Reis, F. (2010). *Como elaborar uma dissertação de Mestrado segundo Bolonha.* Portugal: Pactor.

Saias, J. (12 de 2012). Slides - visao computarizada. Universidade de Evora, Evora, Pt.

Scheuermann, C., Werner, M., & Kessel, M. (2011). Evaluation of Barcode Decoding Performance using ZXING. (p. 6). Germany: Ludwig-Maximilians-University.

SensibleVison. (2007). *Login Without Password*. Obtido em 3 de 5 de 2013, de ensibleVison: http://www.sensiblevision.com/pt-br/p%C3%A1ginainicial.aspx

Singh, R. R. (12 de 2 de 2012). *Article: Understanding WCF*. Obtido em 6 de 6 de 2013, de Code Project: http://www.codeproject.com/Articles/406096/A-beginners-tutorial-for-understanding-Windows

Siqueira, A. (2011). *Atendente Movel: uma solucção de mobilidade para atender os consumidores.* Brasil: Universidade Federal do Rio Grande Sul.

Solem, J. E. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images.* USA: O´Reilly Midia Inc.

Technologies, M. (12 de 2009). *Vision systems for the baked goods industry*. Obtido em 1 de 5 de 2013, de montrose-tech: http://www.montrose-tech.com/

Wikipedia. (s.d.). *Visao Computarizada*. Obtido em 4 de 4 de 2013, de http://en.wikipedia.org: http://en.wikipedia.org/wiki/Computer_vision

Zbar. (2012). *ZBar bar code reader*. Obtido em 1 de 1 de 2013, de Zbar: http://zbar.sourceforge.net/

zdnet. (2013). *Tecnologies News*. Obtido em 5 de 7 de 2013, de zdnet: http://www.zdnet.com/

ZXing.org. (1 de 1 de 2008). *ZXing decoder Online*. Obtido em 7 de 6 de 2013, de ZXing.org: http://ZXing.org/w/decode.jspx