Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

VITOR MORELLI MIACRI

DESENVOLVIMENTO DE APLICATIVO PARA POCKET PC UTILIZANDO C#

ASSIS
2010

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

DESENVOLVIMENTO DE APLICATIVO PARA POCKET PC UTILIZANDO C#

VITOR MORELLI MIACRI

Trabalho de Conclusão de Curso (Qualificação) apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de Curso de Bacharelado em Ciência da Computação, analisado pela seguinte comissão examinadora:

Orientadora: Profa. Dra. Marisa Atsuko Nitto

Avaliador: Prof. Dr. Almir Rogério Camolesi

ASSIS

2010

MIACRI, Vitor Morelli
Desenvolvimento de aplicativo para Pocket PC utilizando C# / Vitor Morelli Miacri. Fundação Educacional do Município de Assis – FEMA – Assis, 2010.
66p.
Orientadora: Profa. Dra. Marisa Atsuko Nitto.
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1.C#. 2.Dispositivos móveis. 3.Pocket PC.

CDD:001.6
Biblioteca da FEMA.

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

VITOR MORELLI MIACRI

DESENVOLVIMENTO DE APLICATIVO PARA POCKET PC UTILIZANDO C#

Trabalho de Conclusão de Curso (Qualificação) apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Bacharelado em Ciência da Computação, analisado pela seguinte comissão examinadora:

Orientadora: Profa. Dra. Marisa Atsuko Nitto

Área de Concentração: Informática

Assis

2010

DEDICATÓRIA

Dedico este trabalho à minha família, aos meus professores e em especial à minha orientadora.

AGRADECIMENTOS

Primeiramente a Deus;

A Profa. Dra. Marisa Atsuko Nitto pela orientação e acompanhamento durante o desenvolvimento deste trabalho;

Á todos os professores que de uma forma ou outra contribuíram para a minha formação acadêmica;

Aos meus pais, Dora Alice Morelli Miacri e Gilberto Alvez Miacri, pelo apoio e compreensão em todos os momentos de minha vida escolar e pessoal;

A todos os companheiros de trabalho e amigos por compartilharem todos os momentos desta jornada.

"Se não puder se destacar pelo talento, vença pelo esforço."

Dave Weinbaum

RESUMO

O uso de aplicações para dispositivos móveis, como celulares, PDAs, Handhelds tem se tornado cada vez maior entre pessoas e empresas que necessitam de grande flexibilidade no acesso e troca de informações. Porém, a grande variedade de dispositivos existentes, a falta de *frameworks* e ferramentas adequadas de desenvolvimento tem dificultado muito a construção dessas aplicações.

Este trabalho apresenta a especificação, o desenvolvimento de uma aplicação para dispositivos móveis baseados na plataforma .NET e suas ferramentas. Este aplicativo tem por objetivo auxiliar a empresa a controlar a quilometragem de sua frota a partir de um Pocket PC, evitando assim o uso de papel e planilhas para a realização deste serviço. Será apresentada também a fundamentação teórica utilizada para o desenvolvimento do aplicativo. Além disso, será feita a descrição e modelagem do problema.

Palavras-chave: C#, Plataforma .NET, Windows Móbile, Pocket PC

ABSTRACT

The use of mobile applications such as mobile phones, PDAs, Handhelds have become bigger and bigger among people and companies who require greater flexibility in accessing and exchanging information. However, the wide variety of existing devices, the lack of appropriate frameworks and development tools has severely hampered the construction of these applications. This study presents the specification and development of a mobile application based on. NET platform and its tools. This application aims to help the company manage the mileage of its fleet from a Pocket PC, avoiding the use of paper and spreadsheets to perform this service. The theoretical basis used for application development willl also be presented. In addition, there will be a description and modeling of the problem.

Keywords: C # Platform. NET, Windows Mobile, Pocket PC

LISTA DE ILUSTRAÇÕES

SUMÁRIO

CAPITULO 1

INTRODUÇÃO

A mobilidade nos dias de hoje está sendo cada vez mais requisitada por empresas que procuram soluções móveis para disponibilizar recursos que possam ser usados em aparelhos portáteis.

Os primeiros aparelhos celulares eram utilizados apenas como meio de comunicação de voz. Com o avanço desta tecnologia foram surgindo novos aparelhos móveis, como os PDAs, PocketPC e SmartPhones com vários recursos que antes não possuíam, como acesso a internet, jogos, agendas, calculadoras, entre outros. O aperfeiçoamento dos componentes de hardware e sistema de software possibilitou o desenvolvimento de aplicativos cada vez mais complexos.

As empresas estão a cada dia buscando novos aplicativos que possam oferecer serviços para seus clientes capazes de possibilitar consultas de informações em seus SmartPhones, PocketPC ou PDA's. A mobilidade vem crescendo muito, e com esse crescimento, é necessário integrar as aplicações corporativas existentes nas empresas e no mundo. Esta integração não é só uma necessidade, mas sim uma meta de diversas empresas (Tonon, 2006).

Neste contexto, surgiu a ideia de criar um aplicativo para dispositivos móveis que corresponda à necessidade da empresa na área de controle da frota interna. O aplicativo será desenvolvido utilizando a plataforma .NET, a linguagem de programação escolhida foi a C#, e a base de dados SQL Server CE que é um SGBD (Sistema Gerenciador de Banco de Dados) suportado por estes dispositivos móveis. (Tarifa et al., 2005; Cembranelli, 2003); Cherry e Demichilli, 2002 e Conallen, 2003).

O desenvolvimento do trabalho foi dividido em duas fases: fundamentação teórica e desenvolvimento do aplicativo. A primeira fase consiste na documentação de toda a teoria que será utilizada no desenvolvimento do aplicativo. Estes conhecimentos foram utilizados para fazer a descrição e a modelagem do problema. A segunda fase apresenta toda a modelagem do aplicativo que será implementado. Esta divisão se justifica pelo fato de não ter muito conhecimento das tecnologias

escolhidas para desenvolver o trabalho. Com os conhecimentos adquiridos nesta primeira fase, foi possível desenvolver com mais facilidade a segunda fase, que é a implementação e documentação do aplicativo.

1.1 - Objetivos

O objetivo deste trabalho é desenvolver um aplicativo de gerenciamento de registro de abertura e fechamento de quilometragem da frota de carros da empresa. Este aplicativo é para aparelhos móveis, sendo um módulo para desktop e outro para o Pocket PC, simplificando o dia-a-dia da empresa.

1.2 - Justificativa

A demanda crescente de aplicativos para dispositivos móveis tem despertado interesse em muitos desenvolvedores, pois as empresas estão sempre buscando tecnologias que podem proporcionar um controle informatizado e centralizado, diminuindo o uso de papel em todo o processo que antes era feito manualmente. A portabilidade dos dispositivos móveis tem atraído diversas empresas, e para isso são necessários aplicativos específicos para cada segmento.

1.3 - Motivação

Adquirir conhecimento na área de desenvolvimento de aplicativos para dispositivos móveis. Perspectiva de atuação neste segmento, já que uma empresa mostrou interesse no aplicativo que será desenvolvido neste trabalho. Oportunidades profissionais promissoras no futuro.

1.4 – Estrutura do trabalho

O trabalho será dividido nos seguintes capítulos:

1. Introdução
2. Fundamentação Teórica Básica
3. Desenvolvimento do Aplicativo.

4. Conclusão.
5. Referências Bibliográficas.

CAPITULO 2

FUNDAMENTAÇÃO TEÓRICA BÁSICA

Neste capitulo será feita a fundamentação teórica necessária para o desenvolvimento do aplicativo para gerenciar a quilometragem da frota de automóveis de uma empresa.

2.1 – Tecnologia Móvel

A necessidade de fornecer serviços que suportam as necessidades do estilo de vida do indivíduo caminha para a dependência cada vez maior da mobilidade, esta por sua vez impulsiona para a convergência dos serviços em uma única solução. Neste sentido é que a evolução da comunicação móvel unida à evolução da computação móvel deu origem aos *smartphones* e consequentemente à internet móvel. A figura 1 mostra a evolução da computação móvel.

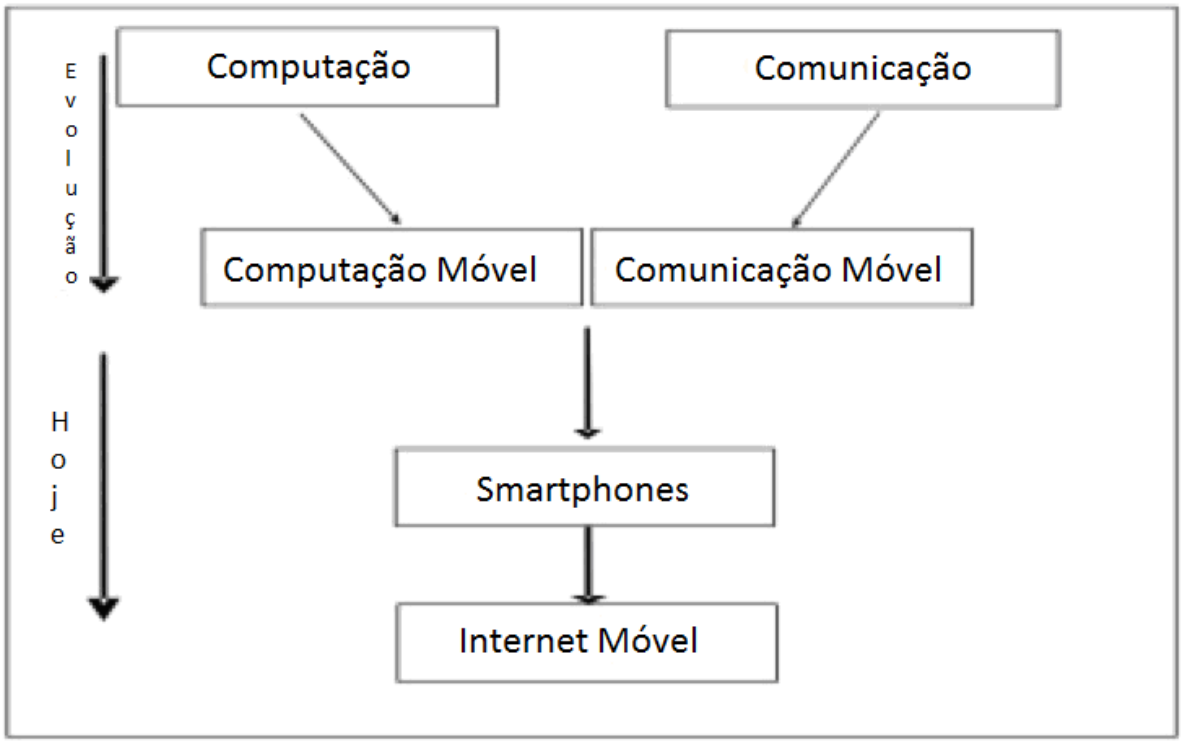


Figura 1: Evolução da computação e comunicação móvel. Fonte: (Tonon, 2006).

Neste cenário de forte crescimento da mobilidade nos próximos anos, cria-se um circulo virtuoso de crescimento tanto das necessidades quanto das soluções móveis. Isso vem ressaltar a importância das tecnologias que permitem desenvolver aplicações móveis (Tonon, 2006).

O *Personal Digital Assistant* (PDA) e o *Smartphone* são os principais dispositivos móveis do mercado e têm se tornado cada vez mais populares, devido à sua simplicidade, funcionalidade, portabilidade e facilidade de uso. Assim como acontece com todos os computadores pessoais, esses dispositivos móveis também dependem de um Sistema Operacional (SO) para poder funcionar. Também é necessário um programa que permita estabelecer a ligação do dispositivo a um computador pessoal para armazenar uma cópia de segurança dos dados e para atualizar as informações existentes. A ligação para troca de dados pode ser realizada mediante a utilização de cabos físicos, ou através de tecnologias sem fios, tais como as redes sem fios ou *Bluetooth*.

Nos últimos anos, tem se presenciado o surgimento de inúmeros aparelhos portáteis, como *notebook*, *handheld*, *netbook* e *Pocket PC*, com o intuito de auxiliar essa força de trabalho chamada de móvel. Esses aparelhos não só auxiliam na eliminação do papel nos processos comerciais, como também podem ajudar milhares de profissionais que exercem suas funções em locais não tradicionais no gerenciamento de compromissos e contatos.

Um exemplo desse tipo de profissional são os representantes de vendas que passam a maior parte de seu tempo fora da empresa, atendendo clientes, e que necessitam de informações diárias sobre o histórico de pagamentos, crédito de cada cliente, seu volume médio de compra, entre outras informações. Seria catastrófico para qualquer empresa que atua no ramo de vendas autorizar um pedido a um cliente inadimplente ou permitir que o vendedor que trabalha externamente possa negociar débitos pendentes diretamente com o cliente, sem ter maiores informações acerca da dívida (Conallen, 2003).

Mesmo com limitações em tamanho e processamento, os usuários têm buscado cada vez mais estes dispositivos para efetuar funções que antes faziam apenas em seu desktop. Isso pela facilidade do dispositivo poder ser carregado na bolsa para qualquer lugar. E isso se deve ao constante desenvolvimento da tecnologia móvel.

2.1.1 – Computação móvel

A computação móvel começou em meados de 1992, quando a Apple lançou no mercado um *handheld* chamado Newton. O Newton chegou ao mercado com uma capacidade de transmissão de dados de 38.5kbps, 1 MB de memória total e tela sensível ao toque. Entretanto, este modelo não teve muita repercussão, mas é considerado o início dos dispositivos móveis. Em 1996, a U.S. Robotics lançou o Palm Pilot 1000 e 5000 que teve uma aceitação grande no mercado e se constituiu na base de toda uma plataforma de “Palms”. A figura 2 mostra esses dispositivos.

Figura 2: Dispositivos móveis. Fonte: (Endel a, 2010 e Endel b, 2010).

No ano de 1996 surgiram também o NEC MobilePro 200 e o Casio A-10 que foram os primeiros dispositivos a utilizar o Windows CE 1.0 da Microsoft. A plataforma Windows CE não teve muita aceitação até o lançamento do Windows CE 3.0 e da plataforma Pocket PC em 2000.

Figura 3: Dispositivos móveis. Fonte: (Endel c, 2010 e Endel d, 2010).

Esses dispositivos só começaram a ter aceitação no mercado quando o HP Jornada e o Compaq Ipaq tiveram embutidos o Sistema Operacional Pocket PC 2000. A figura 4 mostra este dois dispositivos. A empresa Symbiam formada em 1998 por alguns dos maiores fabricantes de celulares do mundo e a PSION entregou ao mercado o Sistema Operacional Symbiam, que roda na maioria dos *smartphones* e *handhelds* da Nokia, e hoje detém a maior fatia do mercado mundial. E finalmente, o grande avanço foi em 2005, quando a Google adquiriu uma empresa de software móvel chamada Android, e em 2007 lançou um sistema operacional baseado em Linux com o mesmo nome Android direcionado a dispositivos móveis, o que permitiu à companhia fazer os aplicativos móveis da Google chegarem ao máximo de mãos possíveis. Uma grande vantagem desse Sistema Operacional é o fato de ser *open source*.

Figura 4: Dispositivos moveis. Fonte: (Endel e, 2010 e Endel f, 2010).

E foi também em 2007 que a Apple revolucionou o mercado com o lançamento do iPhone, que é um *smartphone* com funções de iPod, câmera digital, internet, mensagem de texto (SMS), *visual voicemail* e conexão wi-fi local. A interação com o usuário é feita através de uma tela sensível ao toque. A Apple registrou mais de duzentas patentes relacionadas com a tecnologia que criou o iPhone. A figura 5 mostra em detalhe o design arrojado do iPhone. O iPhone 4 foi lançado em 24 de junho deste ano de 2010 e foram vendidos 1,7 milhoes de aparelhos só nos três primeiros dias de venda. Mostrando o sucesso do aparelho que tem inúmeras funções.

Figura 5: iPhone 4. Fonte: (Endel g, 2010).

E sem dúvida, o mercado está tendendo para a convergência de recursos nos dispositivos móveis, criando equipamentos que concentram funções de *palmtops*, celular, câmera fotográfica, gps, etc., além de oferecerem excelente performance, grande capacidade de armazenamento e inúmeras possibilidades de comunicação.

A introdução de dados nos dispositivos móveis é feita com a utilização de uma pequena caneta que tem uma ponta plástica especial para escrever na tela de cristal líquido chamado *Stylus* ou de um teclado integrado no dispositivo, ou então agregado quando necessário. A introdução de dados depende do dispositivo, porque uns utilizam um alfabeto específico e outros possibilitam a introdução através da escrita “natural”, tendo associado uma aplicação específica de reconhecimento e conversão de caracteres.

Sendo uma das características destes dispositivos a mobilidade, nem sempre é possível uma conexão com a rede, para isso é preciso ter um mecanismo que permita a sincronização do dispositivo com um computador pessoal, para a troca de informação. Esta operação é responsável por trocar os dados entre o PDA e um computador pessoal, para efeitos de atualização e cópia de segurança dos dados. Para este procedimento, liga-se o PDA à base, ou ao cabo de sincronização, e inicia-se o programa de gestão de sincronização (Tonon, 2006). No momento da sincronização efetuam se as seguintes operações:

- os dados atualizados do PDA são enviados para o computador;
- os dados alterados no computador são enviados para o PDA;
- os dados externos são enviados para o PDA usando as regras adequadas às aplicações instaladas. As regras funcionam como se fossem filtros, que veiculam e convertem a informação entre os equipamentos;
- os programas são instalados no PDA;
- os programas do PDA são copiados para o computador, de modo a efetuar uma cópia de segurança. Após a sincronização, toda a informação que está no computador corresponde à informação que se encontra no PDA.

Obviamente, cada empresa escolhe o dispositivo mais adequado para solucionar seus problemas de automação de acordo com as regras de negócios estabelecidas, sempre com o intuito de facilitar e agilizar todo o processo.

Neste projeto, será escolhido o Pocket PC por ser leve, prático, fácil de usar e proporciona maior rapidez no atendimento ao cliente e maior produtividade, visto que diminui constantes retornos à empresa. Conseqüentemente, ela proporciona grande redução de custos operacionais, aumento no volume de vendas, rapidez no atendimento ao cliente e, principalmente, conhecimento antecipado do volume e valor faturado e tendências de mercado para que o departamento de marketing possa criar promoções por exemplo.

2.1.2 – Pocket PC

Será feita uma descrição mais detalhada do Pocket PC, pois foi o escolhido para desenvolver este projeto. O Pocket PC é uma arquitetura operacional dos PDAs da HP, Casio, Compaq, ASUS, entre outros grandes fabricantes e podem ser considerados verdadeiros computadores de mão devido a sua grande capacidade de processamento. O Pocket PC era a intenção original de plataforma para o sistema operacional Windows Mobile. Estes dispositivos autônomos se classificaram em dois tipos basicamente, sem capacidades de telefone celular e aqueles que incluíram telemóvel capacidades. Em ambos os dispositivos foram imbutidos o Pocket PC. Algumas características do Windows Mobile são:

- Compatibilidade com o modelo de programação baseado nas API's do Windows;
- Utilização de componentes em geral (DLL);
- Integração dos dados armazenados no PocketPC com o microcomputador;
- Dados armazenados na memória ROM como se fosse um HD.
- Fácil adaptação para usuários que conhecem o ambiente Windows;

O nome mais atual do Windows Mobile destinados à utilização em Pocket PCs é oficialmente "Windows 7 Phone Professional" para dispositivos móveis com capacidades. A figura 6 mostra a tela do Windows Mobile e serão detalhadas também algumas das opções apresentadas na tela.

Figura 6: Tela do Windows Mobile.

- A tela "*today screen*", em geral traz informações do dia de hoje, dados do dono, anotações, novos e-mails e tarefas em excecução. Ela inclui uma barra de notificação com o status do *bluetooth*, opção de instalar novos programas ou adicionar ítens na tela "*today screen*". O papel de parede pode ser personalizado através do Pocket PC ou os temas podem ser baixados pelo computador e sincronizados para o Pocket PC;
- A barra de tarefas mostra a hora atual, o volume e o *status* da conectividade atual. Quando um programa ou uma mensagem é aberta, um espaço vazio depois do relógio é preenchido com um botão OK ou um ícone para fechar;
- O Outlook Mobile que acompanha o Windows Mobile inclui agendador de tarefas, calendários, contatos e uma caixa de entrada de e-mails que pode interagir com um servidor *Microsoft Exchance* (ter conexão com a internet);
- O Windows Media Player 9 para o Windows Mobile suporta grande parte dos formatos de multimídia existentes como .WMA, .WMV, .MP3, e .AVI. Atualmente a versão 10.0 tem um suporte bem melhor a dispositivos mais modernos, mas para tocar arquivos .MPEG e .WAV, tem que baixar um programa de terceiros para conseguir abrir estes tipos de arquivos. Algumas versões conseguem tocar arquivos MPEG4 Audio (.M4A).

Projetado para ser capaz de realizar boa parte do que é possível em uma versão PC do Windows, o sistema vem com um conjunto de aplicações básicas bem conhecidas no mundo dos PCs, tais como o Word, Excel, PowerPoint, Windows Media Player Pocket. A figura 7 mostra a tela de um aplicativo do Windows Office.

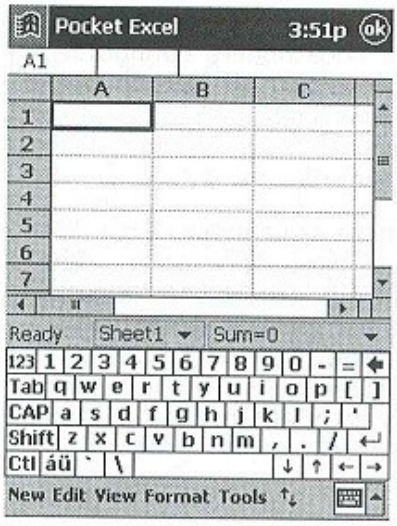
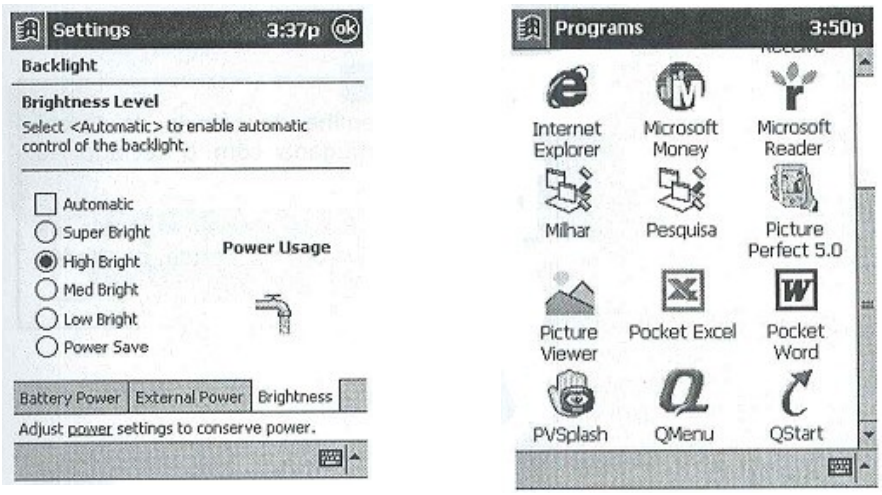


Figura 7: Microsoft Excel versão PocketPC. Fonte: (Siqueira, 2005).

O Microsoft Office que acompanha o Windows Mobile inclui o Word, o Excel para dispositivos móveis; e foi incluído o PowerPoint Mobile a partir do Windows Mobile 5.0. Nestas versões há vários recursos das versões *desktop*; entretanto, certas facilidades como inserção de tabelas e imagens não se encontravam inclusos nas versões anteriores a 5.0.

O principal recurso da barra de tarefas é o botão Iniciar, bem parecido com as das versões anteriores do Windows. O menu Iniciar tem os mesmos recursos do menu do Windows XP, permitindo acessar os programas (*Programs*) , configurações(*Settings*), procurar arquivos, ver a ajuda (*Help*) e ainda executar os programas instalados. A figura 8 mostra essas opções na tela de um Pocket PC.



Tela de Configuração

Tela de Programas

Figura 8: Telas de configurações e programas do Pocket PC. Fonte: (Siqueira, 2005).

O programa de conexão ActiveSync, que acompanha os dispostitivos, tem recursos que permitem converter os arquivos da versão *desktop* para a versão Pocket PC. O ActiveSync actua como porta de ligação entre o computador baseado no Windows e o dispositivo baseado no Windows Mobile, permitindo a transferência da informação do Outlook, dos documentos do Office, de imagens, música, vídeos e aplicações. Quando se sincroniza um item (Tarefas, Contatos, E-mail, Calendário, Arquivo etc.), o gerenciador de sincronização o compara à versão existente em seu desktop. A figura 9 ilustra algumas etapas desse processo.

As alterações mais recentes serão copiadas do dispositivo móvel para o *desktop* e do *desktop* para o dispositivo móvel. Após a sincronização, os itens do dispositivo móvel e do *desktop* serão idênticos.

Figura 9: Processo de sincronização. Fonte: (Siqueira, 2005).

Em geral, pode-se sincronizar qualquer item de seu dispositivo móvel criado pelos programas que suportam o gerenciador de sincronização.

2.2 – Linguagem de Programação C#

A linguagem C# (C *Sharp*) foi desenvolvida por dois engenheiros da Microsoft, Anders Hejlsberg e Scott Wiltamuth. Hejlsberg é conhecido pela criação do Turbo Pascal, um ambiente popular para programação de aplicativos para PC, e liderou o time que arquitetou o Borland Delphi, um dos primeiros vitoriosos Ambientes de Desenvolvimento Integrados (*Integrated Development Environments*, IDEs) para programação cliente/servidor. Baseada em princípios modernos e orientados a objeto, a linguagem foi criada para ajudar os dois desenvolvedores a conseguir melhores resultados com menos linhas de código e pouca chance de ocorrência de erros. Com o C#, desenvolvedores constroem facilmente serviços de Web que possam ser usados através da Internet - a partir de qualquer linguagem ou plataforma. Oferece produtividade aperfeiçoada, possibilita acesso completo à plataforma essencial, assim como ao controle de código de baixo nível, dando aos desenvolvedores o poder de construir sistemas corporativos complexos (Dani, 2008).

O melhor de tudo é que o C# pode ser construído sobre as ferramentas que muitos programadores e organizações já desenvolveram, possibilitando então lançamento mais rápido de seus produtos, reduzindo custos de desenvolvimento, e a habilidade de manter o ritmo com a rápida mudança do mundo da Internet.

2.2.1 - Características

A linguagem C# possui certas características importantes que devem estar em mente na hora de escolher a linguagem de programação do aplicativo (Endel h, 2010). Estas características são:

- Primeira linguagem “orientada a componentes” da família C/C++. .NET *Common Language Runtime* é um ambiente baseado em componentes, e C# é desenhado para facilitar a criação de componentes;
- Em C#, ao contrário de linguagens como Java ou C++, tipos de dados e objetos interagem. Fornece um “sistema unificado de tipos”, onde todos os tipos são tratados como objetos, sem perda de desempenho, ao contrário de linguagens como *Lisp* ou *Smalltalk*;
- Próxima geração de softwares robustos e duráveis. Hoje, todos precisam de um software robusto e durável. Muitas aplicações são difíceis de depurar e algumas vezes trazem resultados inesperados em tempo de execução;
- C# é montado sobre a “herança” do C++, isso torna confortável a adaptação do programador C++. C# permite interoperabilidade com XML, SOAP, componentes COM, DLL e qualquer outra linguagem da Plataforma .NET, mantendo integração total com projetos existentes. Milhões de linhas de código, em C#, são encontradas no .NET, isso permite uma rápida curva de aprendizado e aumento de produtividade;
- O C# inclui idéias de várias linguagens de programação, mas é patente a influência das duas outras principais linguagens com as quais Anders trabalhou anteriormente: o Pascal do Delphi e o Java. Existem também claras influências do C++ e *Smalltalk*;

- Todas as variáveis e códigos são declarados no escopo de classes, de forma semelhante ao Java. É possível, contudo, declara tipos (“*structs*” e enumerações) fora do escopo de classes;
- Enumerações são tipos próprios e incompatíveis com inteiros e com outras enumerações. Existe um tipo lógico (*bool*) incompatível com inteiros. Os tipos intrínsecos são: lógicos, inteiros de vários tipos e com tamanhos pré-definidos (8,16,32 e 64 bits, com e sem 3 sinal), ponto flutuante IEEE de 4 e 8 bytes, string e decimal. Só existe um tipo “*char*”. Tanto o “*char*” como a “*string*” armazenam apenas caracteres Unicode (16 bits). O tipo “decimal” é armazenado como uma mantissa binária de 96 bits e expoente na base 10, para um total de 128 bits. A precisão do decimal e de 28 ou 28 dígitos decimais. Existe também “*structs*”;
- Os objetos, “*arrays*” e “string” são necessariamente alocadas dinamicamente no “*heap*” com o uso do operador “*new*”;
- O C# inicializa a maioria das variáveis com zero, verifica se uma variável foi inicializada antes de ser usada, se um *array* foi indexado fora da faixa e se um inteiro teve sua faixa violada;
- Todas as conversões de tipo “*cast*” são validadas em função do tipo real da variável em tempo de execução, sem exceções. Isto é semelhante ao “as” do Delphi, mas no Delphi só temos “as” para objetos, enquanto no C# todos os tipos têm “*cast*” seguro;
- O operador “.” é usado em diversos lugares, quando em C++ seriam usados “.”, “::” e “->”, como no Delphi;
- Existe um tipo de loop: “*forecach*”, usado para varrer todos os elementos de um *array* ou “*containe*r”;
- O “*swich*” elenca opções mutuamente exclusivas, como no Delphi. O “*break*” depois de cada opção é obrigatório;
- O único mecanismo de tratamento de erros do C# e a exception. O mecanismo e muito semelhante as do Delphi;
- Não existem macros, mas existe compilação condicional, como no Delphi. As *Templates* não são suportadas, pelo menos por enquanto. A Microsoft acena que talvez seja possível criar um mecanismo semelhante aos *templates* no

futuro. De qualquer forma, no C# suporta "*reflections*", o que pode substituir *templates* em algumas situações;

- Suporta sobrecarga de funções e de operadores, como no C++, mas não tem argumentos default como no C++ e o Delphi;
- Possui operadores de conversão, mas existe uma sintaxe para indicar se a conversão deve ser implícita ou explicita. O construtor não é usado como operador de conversão. As conversões implícitas em C++ são um grande problema, pois acabaram permitindo que o compilador inevitavelmente crie uma conversão que não faz sentido.

2.2.2 - C# x Java.

Além de conhecer as características do C# é importante saber quais delas são comuns com o Java (Dani, 2008). As características comuns às duas linguagens são:

- Modelo de orientação a objetos baseado em herança simples de classes com um ancestral comum;
- Herança múltipla de "*interfaces*";
- Gerenciamento de memória automático com "coletor de lixo";
- Tipagem forte;
- Rodam em um "ambiente gerenciado", no qual a segurança e integridade das operações efetuadas pelos programas podem ser garantidas;
- Amplo suporte a "*reflections*", um recurso também conhecido como "informação de tipos em tempo de execução".

Estas semelhanças levaram algumas pessoas a comentar que o C# era um "clone de Java", desenvolvido pela Microsoft em razão dos problemas legais, já resolvidos, ocorridos com a Sun em relação ao licenciamento da linguagem Java. Mesmo existindo alguma base nestas afirmações, é extremamente injusto com o C# chamá-lo de clone de Java. O C# não só resolve vários problemas que são notórios do Java como também traz muitas novidades. A Microsoft bem que tentou melhorar

o Java, mas a Sun não deixou (Endel h, 2010). Veja alguns problemas do Java corrigidos pelo C#:

- Os programas na arquitetura .NET são sempre compilados, ao passo que em Java eles são normalmente interpretados;
- O C# tem "*structs*", um tipo "barato" para ser usado em situações onde o custo de uma classe como alocação de memória e coleta de lixo não seria justificada. Um exemplo clássico é a representação de uma coordenada no plano (dois inteiros: X, Y);
- O C# tem enumerações, mais ou menos como versões mais recentes do C++ ou o próprio Pascal;
- Existe passagem de parâmetros por referência.

Além das características em comuns e dos problemas corrigidos do Java, o C# traz alguns recursos do C++ que foram omitidos no Java que são:

- Sobrecarga de operadores, algo muito útil nos "cálculos científicos", por permitir tratar números complexos, vetores e matrizes com a notação dos operadores aritméticos tradicionais como "+" e "*";
- Operadores de conversão, para converter valores de um tipo para outro. No C# existem tanto operadores de conversão implícitos, mais ou menos como no C++, como explícitos, que exigem o operador de "*cast*". Ao contrário do C++, o construtor que aceita um único argumento não é usado automaticamente como função de conversão.

Embora compartilhe características com outras linguagens, o C# é uma linguagem que traz vários recursos muito interessantes que não ou existem em outras ou dão muito trabalho para programar ou têm desempenho ruim. É uma linguagem muito interessante para programadores que desejam migrar para a plataforma .NET da Microsoft (Endel i, 2010).

2.2.3 – Plataforma .NET

A necessidade de fornecer serviços que suportam as necessidades do estilo de vida do indivíduo caminha para a dependência cada vez maior da mobilidade, e esta por sua vez impulsiona para a convergência dos serviços em uma única solução. Neste cenário de forte crescimento da mobilidade nos próximos anos, cria-se um circulo virtuoso de crescimento tanto das necessidades quanto das soluções móveis. Isso vem ressaltar a importância das tecnologias que permitem desenvolver aplicações móveis, como a plataforma .NET, Java, etc.

A plataforma .NET é um conjunto de tecnologias desenvolvidas para transformar a Internet em uma plataforma de computação distribuída de larga escala, como mostra a figura 10.

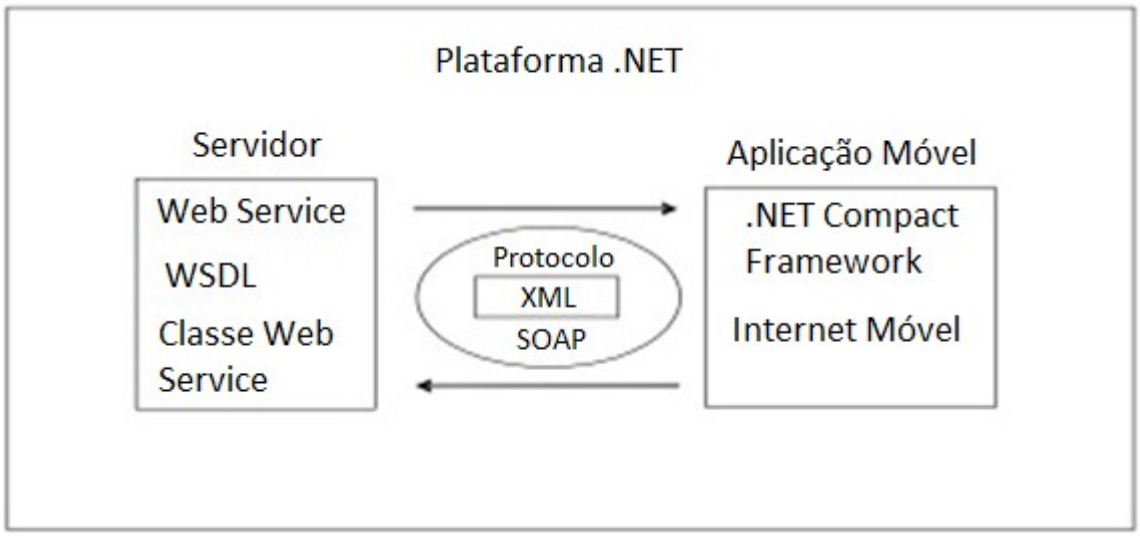


Figura 10: Plataforma .Net. Fonte: (Tonon, 2006).

O .Net é uma estratégia da Microsoft, que traz mudanças no modo como o software é feito e utilizado, e como um software “conversa” com o outro. Ele utiliza muito o XML e a colaboração entre sistemas, e também o conceito de software como serviço. Ele visa fornecer meios para criação aplicações voltados para um ambiente completamente distribuído via Web (Bonifácio, 2001).

A Plataforma .NET fornece:

- Um modelo independente de linguagem, com programação consistente em todas as camadas de uma aplicação;

- Interoperabilidade direta entre tecnologias;
- Fácil migração entre as tecnologias existentes;
- Alto suporte à plataforma neutra da Internet;
- Tecnologias baseadas em padrões, incluindo Protocolo de Transferência de HiperTexto (HTTP), Linguagem de Marcação Extensível (XML), e Protocolo de Acesso Simples a Objetos (SOAP).

2.2.4 - Versionamento

A estrutura do .NET permite que versões diferentes sejam executadas na mesma máquina usando um conjunto de versões para gerenciar isto, através da MSIL (*Microsoft Intermadiate Language*), e com isso ele resolve os problemas de dependências, que eram os grande problemas das DLLs. A figura 11 mostra a arquitetura do MSLI.

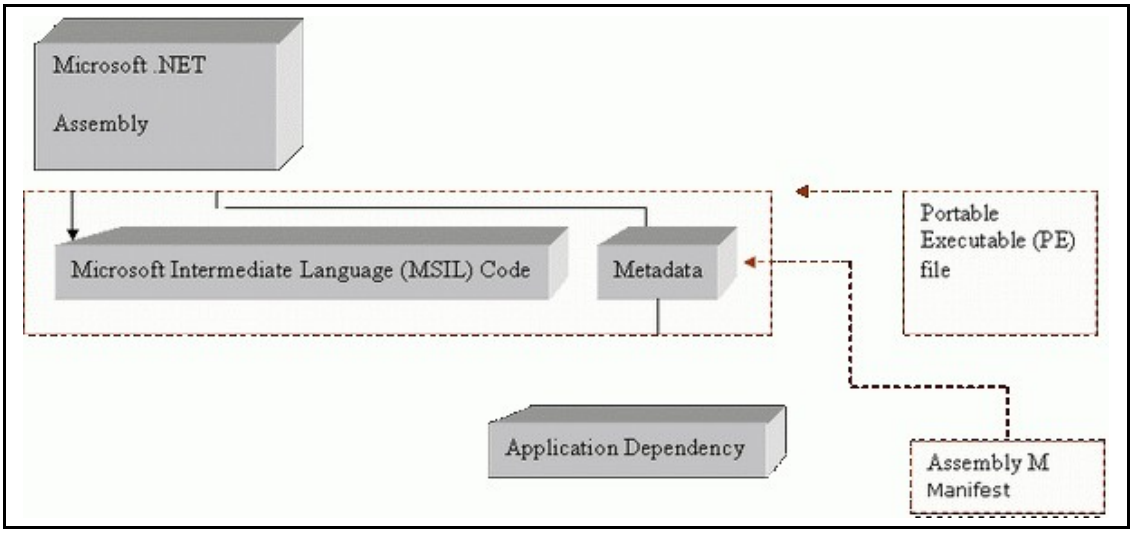


Figura 11: Arquitetura do MSLI. Fonte: (Endel j, 2010).

Os serviços em *Run-Time* dessa arquitetura são:

- Gerência da memória;
- *Type safety*;
- Segurança;

- Gerência de exceção;
- *Thread support*;
- *Debuging*.

2.2.5 – Framework .NET

A base da .NET é o *Framework*.NET. Ela é a infra-estrutura de todo o ambiente .NET e serve para criar e executar sistemas que abrangem aplicações WEB e até sistemas convencionais, como soluções de ferramentas de escritório ou o cliente-servidor. Segundo (Bonifácio, 2001), o .NET *Framework* é um componente integral do Windows que oferece suporte à criação e execução de aplicativos e Serviços XML da Web e foi criado para atender os seguintes objetivos:

- Fornecer um ambiente de programação orientada a objetos consistente, se o código objeto for armazenado e executado localmente, executado localmente, mas distribuído pela Internet ou executado remotamente;
- Para fornecer um ambiente da execução de código que minimiza conflitos de implantação e versionamento de software, que promova a execução segura do código, incluindo o código criado por terceiros: desconhecidos ou semi-confiáveis, que elimina os problemas de desempenho dos ambientes interpretados ou com scripts;
- Para tornar a experiência do desenvolvedor consistente, através dos diversos tipos de aplicativos, como aplicativos baseados no Windows e aplicativos baseados na Web;
- Para criar todas as comunicações nas indústrias padrão, para garantir que códigos baseados no .NET *Framework* possam se integrar a qualquer outro código.

Esse Framework é composto por três partes: o CLR (*Common Language Runtime*), as classes do Framework e o ASP (*Active Server Pages*) .NET. A CLR é um ambiente de tempo de execução que processa, executa e gerencia código MSIL (*Microsoft Intermediate Language*). As tarefas da CLR incluem: gerenciamento de

memória, gerenciamento de seqüência e verificação de segurança (Bonifácio, 2001).

As classes do Framework .NET contêm as ferramentas que permitem realizar todos os tipos de tarefas, desde escrever em um banco de dados até ler a partir de uma página Web. O ASP .NET é a evolução do ASP, fornecendo a facilidade de interagir com as linguagens .NET como por exemplo, VB .NET e C#. O ASP .NET funciona como gerenciador de interface das páginas Web criadas com a tecnologia .NET. O código que é executado dentro da CLR é chamado de código gerenciado (*Managed Code*) (Macdonald, 2002).

A figura 12 mostra o relacionamento do *Common Language Runtime* e da biblioteca de classes para seus aplicativos e para o sistema geral. A figura 12 também mostra como o código gerenciado opera dentro de uma arquitetura com essas três partes na arquitetura da plataforma .NET.

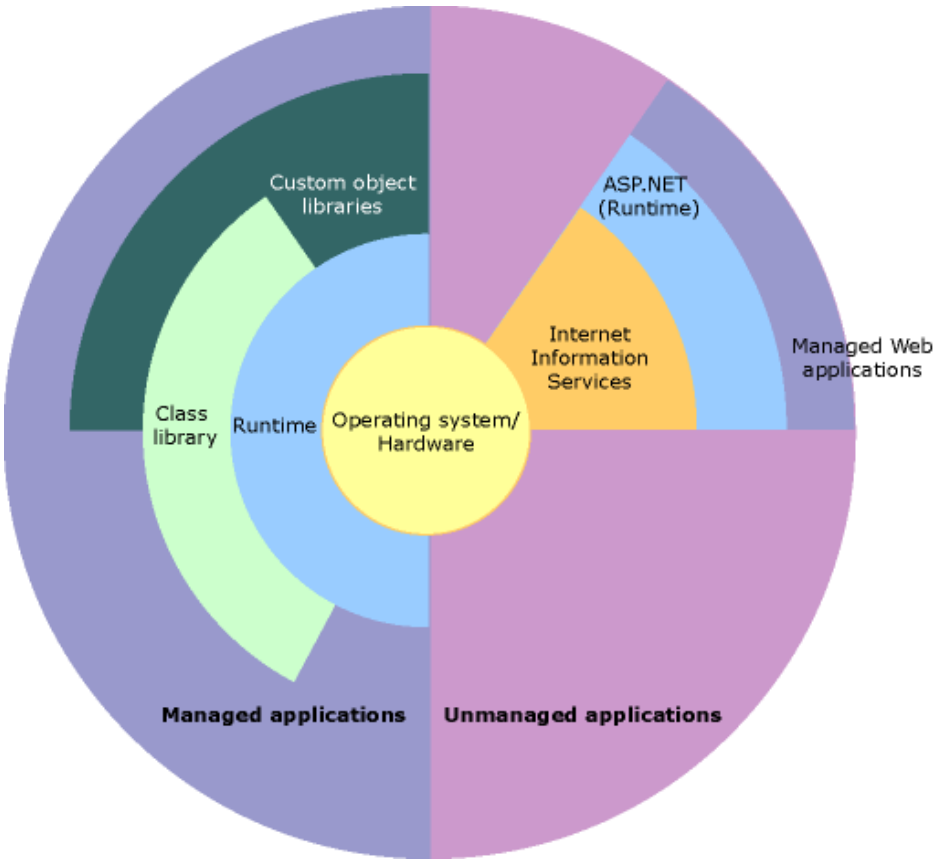


Figura 12: Relacionamento das partes do *Framework*. Fonte: (Endel k, 2010).

Na verdade, o código gerado pelas linguagens de programação que suportam o .NET é sempre código gerenciado, o que significa que ele funciona sob os serviços da CLR e opera sob a sua supervisão.

A figura 13 mostra a arquitetura do .Net *Framework*.

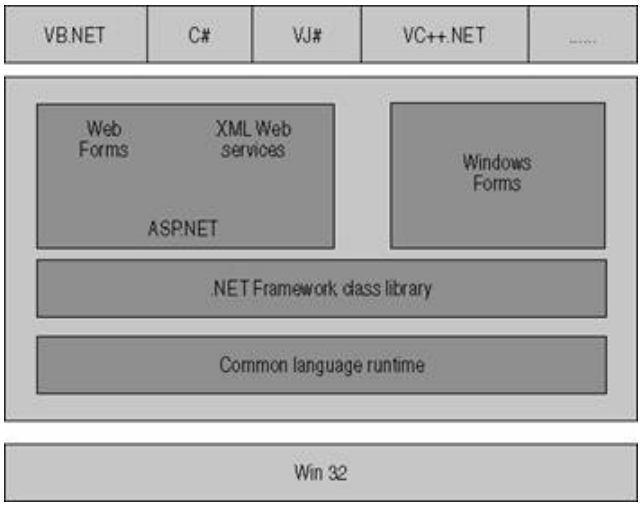


Figura 13: Arquitetura do .NET *Framework*. Fonte: (Endel j, 2010).

A arquitetura da estrutura do .NET Framework tem linguagens como o C#, VC++, VB.Net, J# com as quais os desenvolvedores criam aplicações tais como formulários de Windows, ASP.Net, Serviços Windows, WEB Services, entre outros.

2.2.5.1 – *Common Language Runtime*

O *Common Language Runtime* gerencia memória, execução de segmento, execução do código, verificação de segurança do código, compilação e outros serviços do sistema. Esses recursos são intrínsecos para o código gerenciado, que executa no *Common Language Runtime*.

Quanto à segurança, os componentes gerenciados são concedidos variando os graus da confiança, dependendo do número de fatores que incluem sua origem (como a Internet, rede corporativa ou computador local). Isso significa que um componente gerenciado pode ou não ser capaz de executar operações de acesso de arquivo, operações de registro de acesso ou outras funções

confidenciais, mesmo se ele estiver sendo usado no mesmo aplicativo ativo. Ele também impõe segurança de acesso a código.

O *Runtime* também impõe robustez ao código ao implementar uma estrita infra-estrutura *Type-and-Code-Verification* chamada *Common Type System* (CTS). O CTS assegura que todo código gerenciado é autodescritivo. Os diversos compiladores de linguagem da Microsoft e de terceiros geram códigos gerenciados que estão em conformidade com o CTS. Isso significa que códigos gerenciados podem consumir outros tipos gerenciados e instâncias, enquanto forçam estritamente a fidelidade tipo e segurança (Endel k, 2010).

Além disso, o ambiente gerenciado do *Runtime* elimina muitos problemas comuns de software. Este gerenciamento automático de memória resolve os dois erros mais comuns de aplicativos: vazamentos e referências inválidas de memória. Acelera a produtividade do desenvolvedor. Os programadores podem escrever aplicativos em sua linguagem de desenvolvimento de preferência, mas aproveitar completamente o *Runtime*, a biblioteca de classes e componentes escritos em outras linguagens, por outros desenvolvedores. Qualquer fornecedor de compilador que escolher direcionar o *Runtime* pode fazê-lo. Com isso facilitam bastante o processo de migração para os aplicativos existentes. A figura 14 mostra a arquitetura CLR.

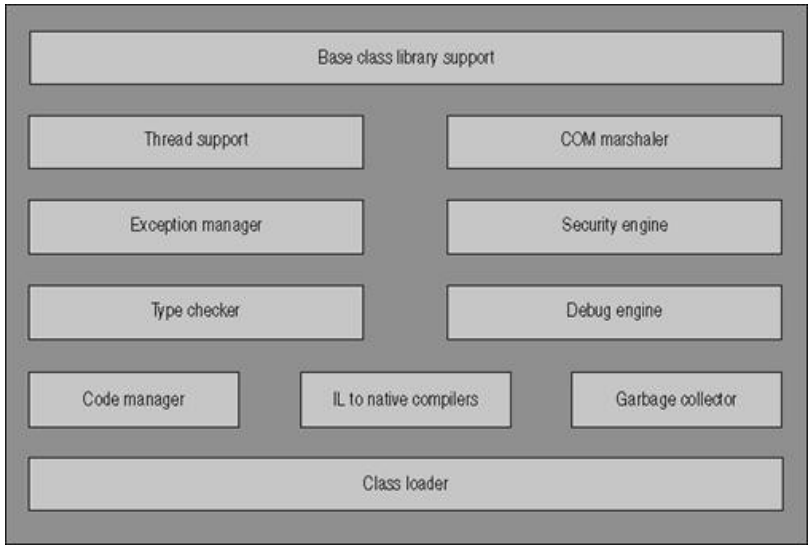


Figura 14: Arquitetura da CLR. Fonte: (Endel j, 2010).

Projetado para melhorar o desempenho, embora o *Common Language Runtime* forneça vários serviços padrão de *Runtime*, o código gerenciado nunca é interpretado. Um recurso chamado compilação *Just-In-Time* (JIT) ativa todos os códigos gerenciados para executar na linguagem nativa da máquina do sistema, no qual ele estiver em execução. Enquanto isso, o gerenciador de memória remove as possibilidades de memória fragmentada e aumenta a localidade-de-referência da memória, melhorando ainda mais o desempenho.

Finalmente, o *Runtime* pode ser hospedado por aplicativos de alto desempenho, do lado do servidor, como o Microsoft ® SQL Server ™ e Serviços de Informações da Internet (IIS). Esta infra-estrutura permite que você use código gerenciado para escrever sua lógica corporativa, enquanto aproveita o desempenho superior dos melhores servidores de empresa que suportam *runtime hosting* (Endel k, 2010).

2.2.5.2 – Biblioteca de classes do .NET *Framework*

A Biblioteca de classes do .NET *Framework* é uma coleção de tipos reutilizáveis que se integram rigidamente com o *Common Language Runtime*. A biblioteca de classes é orientada a objetos, fornecendo tipos que seu próprio código gerenciado pode derivar. Isso não só torna os tipos do .NET *Framework* fáceis de usar, como também reduz o tempo associado ao aprendizado de novos recursos do .NET *Framework*. Além disso, componentes de terceiros podem se integrar totalmente com classes do .NET *Framework*. As classes da coleção .NET *Framework* implementam um conjunto de interfaces que podem ser usadas para desenvolver suas próprias coleções de classes. Sua coleção de classes será perfeitamente combinada com as classes do .NET *Framework* (Endel k, 2010).

O que se espera de uma biblioteca de classe orientada a objetos, é que os tipos do .NET *Framework* permitam que se realize uma gama de tarefas comuns de programação, incluindo tarefas como gerenciamento de sequência de caracteres, coleta de dados, conectividade do banco de dados e acesso a arquivos. Além dessas tarefas comuns, a biblioteca de classes inclui também tipos que oferecem suporte a uma variedade de cenários especializados de desenvolvimento. O .NET *Framework* pode ser utilizado para desenvolver os seguintes tipos de aplicativos e serviços:

- Aplicativos de console;
- Aplicativos Windows GUI (Windows Forms);
- Aplicativos Windows Presentation Foundation (WPF);
- Aplicativos ASP.NET;
- Serviços da Web;
- Serviços do Windows;
- Aplicativos orientados para serviços usando Windows Communication Foundation (WCF);
- Aplicativos habilitados para fluxo de trabalho usando Windows Workflow Foundation (WF).

As classes *Windows Forms* são um conjunto abrangente de tipos reutilizáveis que simplificam vastamente o desenvolvimento do Windows GUI. Para criar um aplicativo de *Web Form* ASP.NET, deve se usar as classes *Web Forms* (Endel j, 2010).

2.2.5.3 – ASP .NET

ASP.NET é a mais nova tecnologia da Microsoft para aplicações Web, com recursos extremamente poderosos, permitindo o desenvolvimento de aplicações com facilidade e rapidez. É um componente do IIS que permite através de uma linguagem de programação integrada na .NET *Framework* criar páginas dinâmicas. Não é nem uma linguagem de programação como VBScript ou PHP, nem um servidor como IIS ou Apache. O ASP.NET é baseado no *Framework* .NET herdando todas as suas características, por isso, como qualquer aplicação .NET, as aplicações para essa plataforma podem ser escritas em várias linguagens, como C e Visual Basic .NET.

2.2.6 – O .NET Compact Framework (.NET CF)

O .NET *Compact Framework* (.NET CF) oferece um ambiente de execução, um modelo de desenvolvimento e uma biblioteca de tipos para fácil implementação

de aplicativos para dispositivos móveis do tipo *Windows Mobile*. No entanto, existem diferentes plataformas de dispositivos baseados no *Windows Mobile*. Cada uma, com suas peculiaridades, o que torna o desenvolvimento do aplicativo móvel, mesmo que baseado no .NET CF um tanto quanto diferente.

A tabela 1 apresenta as plataformas suportadas pelo .NET CF.

Dispositivo	Plataforma
Pocket PC	Windows Mobile 2003 for Pocket PC, Windows Mobile 2003 for Pocket PC SE, Windows Mobile 5.0 software for Pocket PC, Windows Mobile 6.0 software for Pocket PC
Smartphone	Windows Mobile 5.0 software for Smartphone, Windows Mobile 6.0 software for Smartphone
Windows CE Embedded	Windows CE 4.2, Windows CE 5.0, Windows Embedded CE 6.0

Tabela 1: Plataformas suportadas por .NET CE. Fonte: (Endel l, 2010).

O .NET *Compact Framework* dispõe das seguintes funcionalidades principais:

- Executa programas independentes de hardware e sistema operacional;
- Suporta protocolos de rede e conecta-se diretamente a XML *Web Services*;
- Provê aos desenvolvedores um modelo para basear suas aplicações e componentes a vário ou a uma categoria específica de dispositivos;
- Fornece benefícios de design e otimização em sistemas de recursos limitados;
- Obtém performance ótima para geração de código nativo através da compilação JIT (*Just-In-Time*) [1].

2.2.6.1 - Biblioteca de tipos do .NET CF

No que se refere à biblioteca de tipos do .NET CF, a base é a mesma para as plataformas baseadas no Pocket PC e no Smartphone. A grande diferença fica por conta do desenvolvimento da UI (*User Interface*), onde se percebe:

- O Pocket PC possui um estilo de formulário mais rico, com componentes mais sofisticados, para interação do usuário. No Pocket PC encontramos componentes como *Buttons*, *Checkboxes*, *Radio Buttons, Tool Bar*, *Status Bar*, entre outros. Alguns sequer fazem sentido no contexto do Smartphone, que não possui tela sensível ao toque;
- A entrada de informações no Pocket PC é feita através do SIP (*Soft Input Panel*) ou eventualmente através de teclado do hardware;
- No Smartphone, a navegação da aplicação é efetuada através do menu do dispositivo e do uso das *Softkeys*. A entrada de dados é feita através de teclado numérico, onde é possível utilizar modos de entrada como numérico, alfanuméricos.

2.3 – Banco de dados

O banco de dados que será utilizado para armazenar os dados do aplicativo de controle de quilometragem da frota é o SQL Server CE (*Compact Edition*) que é uma extensão da versão *desktop* do SQL Server 2000 para dispositivos, rodando no sistema operacional Windows CE.

2.3.1 - SQL Server Compact Edition (SSCE)

Em um nível conceitual, é possível pensar no SSCE como uma versão significativamente diminuída do mecanismo de banco de dados do SQL Server 2005. Entretanto, trata-se de um mecanismo de banco de dados separado criado de forma a maximizar os principais recursos necessários para um armazenamento de dados relacionais transacionais simples e seguro, que minimiza os requisitos de disco, memória e instalação do aplicativo *host*.

Segundo (Endel m, 2010), o .NET Framework fornece um Provedor (*Provider*) para acesso ao SQL Server CE através do *namespace System.Data.SqlServerCe*. Ele usará todo o poder do DDL (*Data Definition Language*) para criar um Banco de Dados, alterar tabelas e fazer a definição de valores padrões e também usará a DML (*Data Manipulation Language*) para inserir (*insert*), deletar (*delete*) e atualizar (*update*) os dados. Ele suporta criptografia até 128 Bits para senhas e arquivo. O SQL Server CE suporta Banco de Dados de até 2GB (*GigaBytes*) e Blobs (termo usado em certos projetos *open source* para descrever um código objeto para o qual não se disponibiliza o seu código-fonte) de até 1GB (*GigaByte*).

2.3.2 - Histórico do SSCE

A herança do SSCE leva ao SQL Server CE 1.0, que foi lançado em 2001. Esse foi o primeiro mecanismo de dados relacionais lançados pela Microsoft para sistemas operacionais de dispositivos móveis, baseado nos recursos de banco de dados do SQL Server 2000. Seguiram-se as versões 1.1 e 2.0, que aprimoravam a experiência do usuário, sendo que a 2.0 fornecia integração com aplicativos do .NET *Compact Framework*. Juntamente com o .NET 2.0 e o SQL Server 2005, o SQL Server 2005 Mobile Edition foi lançado como o mecanismo de banco de dados móvel da próxima geração (ENdel n, 2010). O SQL *Server* 2005 *Mobile Edition* oferecia vários novos recursos, incluindo confiabilidade e desempenho aprimorados, melhores opções de sincronização e mais integração com o SQL Server 2005 e o Microsoft Visual Studio 2005.

Quando o SSCE foi anunciado pela primeira vez no TechEd 2006, o novo nome dessa versão foi anunciado como SQL *Server 2005 Everywhere Edition*. Esse nome existiu apenas por um breve período, entre sua primeira divulgação e o momento em que a Microsoft refinou seus planos e lançou os recursos do SSCE, anunciando-os e o novo nome no TechEd Europe 2006, em novembro de 2006. Assim, é possível encontrar material que permanecerá por algum tempo na Web com o nome de SQL *Server 2005 Everywhere Edition* ou, abreviadamente, SQL *Everywhere*; ambos são sinônimos de SSCE (Endel m, 2010). É importante observar que o SSCE consiste em um mecanismo de dados completamente diferente e significativamente mais capacitado que as edições anteriores do SQL Server CE; assim, é necessário diferenciá-los cuidadosamente.

2.3.3 - Principais recursos do SSCE

A principal função do SSCE é permitir o armazenamento e o acesso seguros a dados relacionais transacionais. Você pode executar consultas SQL que incluem consultas DDL (*Data Definition Language*) e DML (*Data Manipulation Language*) através do mecanismo do SSCE. Com o SSCE, a instância do banco de dados é criada como um único arquivo.sdf. Nesse banco de dados, você pode definir tabelas com chaves primárias e restrições. O SSCE dá suporte à integridade referencial completa por meio de restrições de chaves estrangeiras, exclusões e atualizações em cascata (Endel m, 2010).

Além disso, o SSCE dá suporte aos seguintes recursos:

- Várias conexões simultâneas para acesso a dados *multithread*;
- Proteção por senha e criptografia de 12 bits do arquivo de dados .sdf do SSCE;
- Uma grande variedade de tipos de dados de coluna;
- Cursores atualizáveis e roláveis para um acesso rápido e fácil aos dados conectados;
- Bancos de dados que podem crescer até 4 GB;
- Sincronização com o SQL Server por meio de replicação de mesclagem e RDA (*Remote Data Access*).

2.3.4 - Novos recursos do SSCE

Na verdade, todos os principais recursos do SSCE faziam parte do SQL *Server* 2005 *Mobile Edi*tion, que foi lançado com o SQL Server 2005 e o .NET 2.0 em outubro de 2005. O SSCE possui vários recursos periféricos adicionais que o distinguem do *Mobile Edition*:

- Agora, o SSCE pode ser executado em qualquer sistema operacional Windows com suporte, incluindo dispositivos móveis, Tablet PCs, computadores portáteis, *desktop*s e servidores.

- O SSCE pode ser atualizado com a Microsoft Update, os *Systems Management Server* ou a *Microsoft Windows Server Update Services*.
- O SSCE pode ser implantado usando o *ClickOnce* (Endel m, 2010).

CAPITULO 3

DESENVOLVIMENTO DO APLICATIVO

Neste capítulo, será apresentada a modelagem do problema, divido por módulos para poder facilitar o entendimento de todos os processos a serem desenvolvidos e especificados.

3.1 – Definição do problema

O problema que será abordado neste trabalho consiste no desenvolvimento de um software para o gerenciamento da quilometragem dos veículos da empresa, utilizando um Pocket PC. Para a implementação aplicativo será usado à linguagem C#, será implementado dois tipos de softwares, um voltado para desktop e outro para dispositivo móvel. Este dispositivo será utilizado para agilizar todo o processo do controle que é feito manualmente com o auxilio de planilhas Excel.

3.2 – Modelagem do problema

O usuário irá interagir com o sistema através de toques na tela com uma interface simples e prática. Primeiramente é exibida uma tela para informar nome de usuário e senha onde será verificado se ele é cadastrado ou não no sistema. Em

seguida é mostrada a tela de abertura de Quilometragem do veículo junto com mais alguns dados para informar, por exemplo, o destino.

A próxima tela será a do menu principal, onde será possível escolher entre: fechar a quilometragem, verificar mapas, inserir alguma informação ou sair do sistema. Na tela para fechar a quilometragem apenas é necessário inserir o quilometro atual do veículo, só assim o será permitido sair do sistema. Todos dados serão armazenados no próprio aparelho para depois serem transferidos para o módulo que fica instalado em um computador de mesa para que, através de uma conexão com a internet, seja feita as alterações no banco de dados remoto.

A figura 15 representa a modelagem do problema.

Figura 15: Modelagem do problema.

O aplicativo foi dividido em módulos para facilitar a implementação e melhor entendimento de sua modelagem.

Módulo 1: Criação da estrutura das tabelas

Neste módulo é criada toda a estrutura das tabelas utilizando o banco de dados SQL *Server Compact Edition*, pois foi o que apresentou melhor compatibilidade, desempenho e resposta junto ao *Windows Móbile* usado com os dispositivos móveis.

Módulo 2: Criação da Interface com o Usuário

Este módulo foi desenvolvido dois aplicativos, um para desktop e outro para o Pocket PC, ambos utilizando o *Visual Studio* 2008. A interface para o Desktop foi criada para realizar todos os cadastros básicos, como usuários, carros, senhas e também para a emissão de relatórios e coletar as informações do dispositivo para a base de dados instalada localmente na máquina. A interface para o Pocket PC é para o registro da quilometragem do carro e o envio das informações para o aplicativo instalado na maquina local.

Módulo 3: Comunicação Pocket PC e computador desktop

Neste módulo foi implementado a comunicação do dispositivo móvel com a maquina local para ser efetuada a troca de informações.

Módulo 4: Comunicação Desktop e Base de Dados

Este módulo demonstra o modelo de comunicação entre a máquina local e o banco de dados. Inicialmente a base de dados será criada em um servidor remoto utilizando o SQL *Server* CE para que o aplicativo instalado em uma máquina local possa acessar e realizar os processos de inserção, exclusão e alteração de dados.

3.3 – Especificação

Para fazer a especificação deste aplicativo foi utilizada uma metodologia orientada a objetos, representada em diagramas UML (*Unified Modeling Language*), construídos através da ferramenta o StarUML. Para a especificação do aplicativo foram utilizados o diagrama de caso de uso, o diagrama de classes, o diagrama de atividades e os diagramas de sequência.

3.3.1 – Diagrama de casos de uso

A figura 16 ilustra o caso de uso do aplicativo.

- **Manter Usuários:** Responsável por todas as operações relacionadas ao usuário.
- **Manter Carros:** Responsável por todas as operações relacionadas ao carro.
- **Manter Quilômetros:** Responsável por todas as operações relacionadas ao quilometro.
- **Aplicativo Pocket:** Responsável pela interação do usuário com o aplicativo instalado no Pocket PC.
- **Aplicativo Desktop:** Responsável pela interação do usuário com o aplicativo instalado no Desktop.

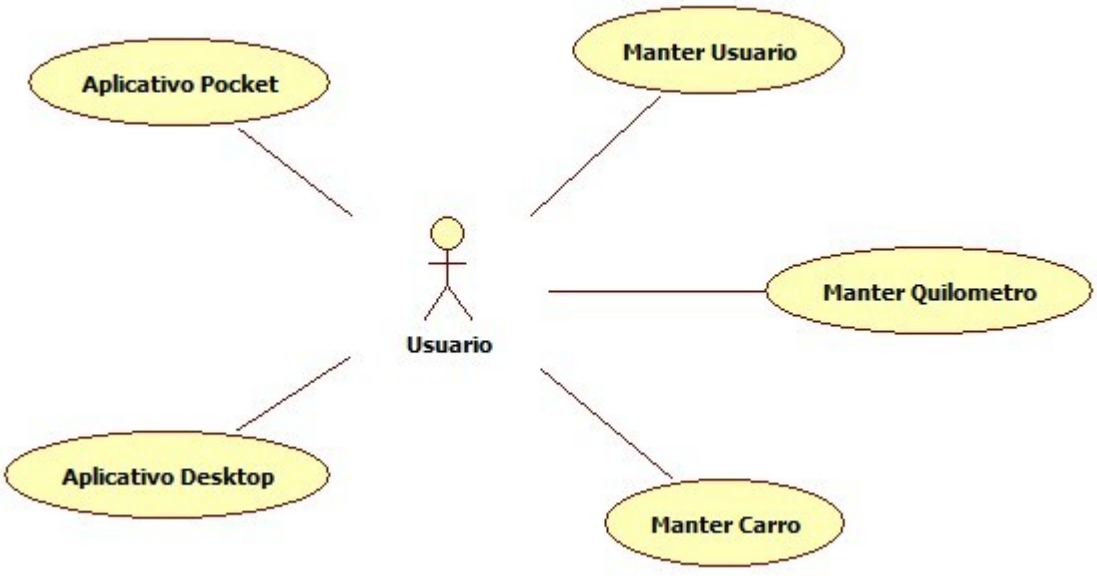


Figura 16 – Casos de Uso.

3.3.2 – Diagramas de classes

O desenvolvimento do aplicativo foi desenvolvido em camadas que foram definidas da seguinte maneira: Modelos que conterá os atributos dos campos de cada tabela. A camada DAL (Data Access Layer) conterá os métodos de manipulação das tabelas no banco de dados e a camada BLL (Business Logic Layer) onde será implementado todas as regras de negócios para o aplicativo.

A figura 17 mostra o diagrama de classe do pacote Modelos.

- **Modelos**
  - **UsuarioVO:** classe responsável por instanciar os usuários do aplicativo.
  - **CarroVO:** classe responsável por instanciar os carros do aplicativo.
  - **QuilometrosVO:** classe responsável por instanciar os quilômetros do aplicativo.
  - **LoginVO:** classe responsável por instanciar os nomes de usuário.

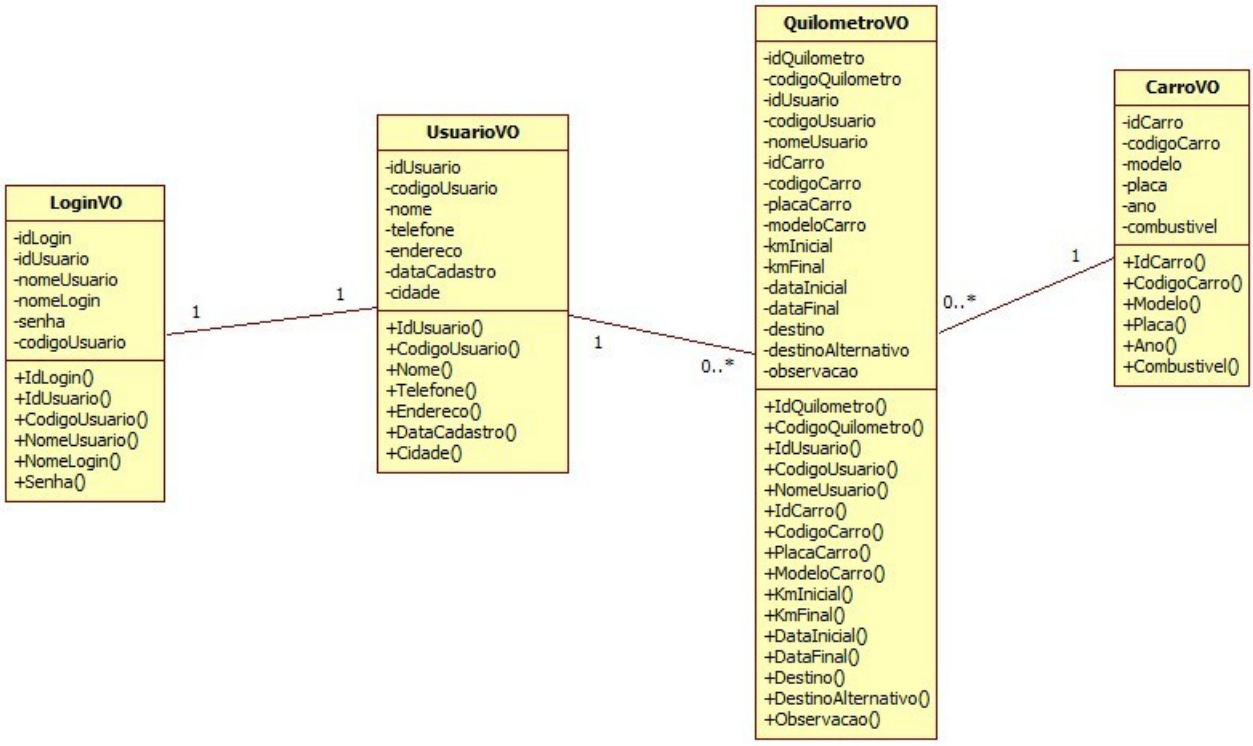


Figura 17 – Diagrama de Classes - Pacote Modelo.

A figura 18 mostra o diagrama de classes do pacote DAL.

- **DAL**
  - **UsuarioDAL:** classe responsável em armazenar os usuários na base de dados.
  - **CarroDAL:** classe responsável em armazenar os carros na base de dados.
  - **QuilometroDAL:** classe responsável em armazenar os quilometros na base de dados.
  - **LoginDAL:** classe responsável em armazenar os nomes de usuários na base de dados.
  - **Dados:** classe genérica de acesso entre o banco de dados e a camada DAL.

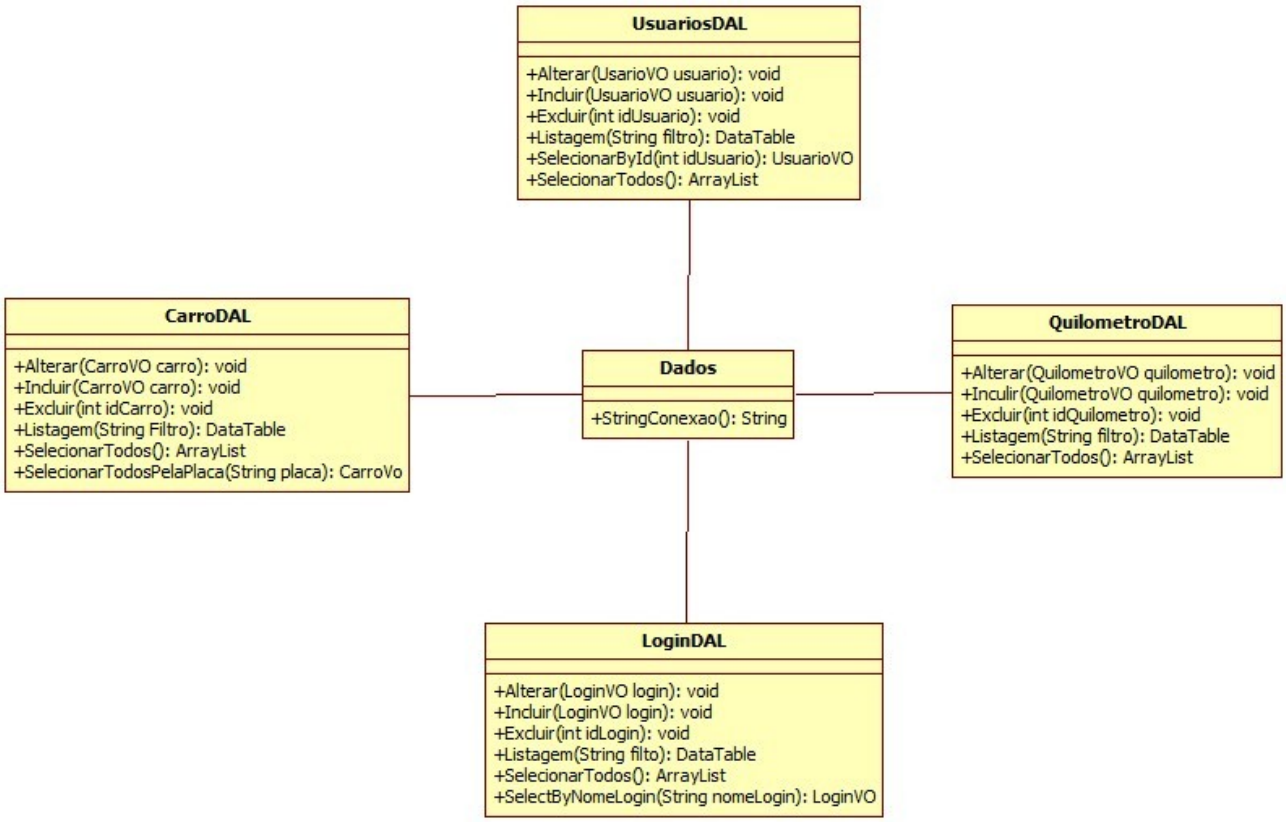


Figura 18 – Diagrama de Classes - Camada DAL

A figura 19 mostra o diagrama de classes do pacote BLL.

- **BLL**
  - **UsuarioBLL:** classe responsável pelas regras de negócio para usuários.
  - **CarroBLL:** classe responsável pelas regras de negócio para carros.
  - **QuilometroBLL:** classe responsável pelas regras de negócio para quilometros.
  - **LoginBLL:** classe responsável pelas regras de negócio para nomes de usuário.

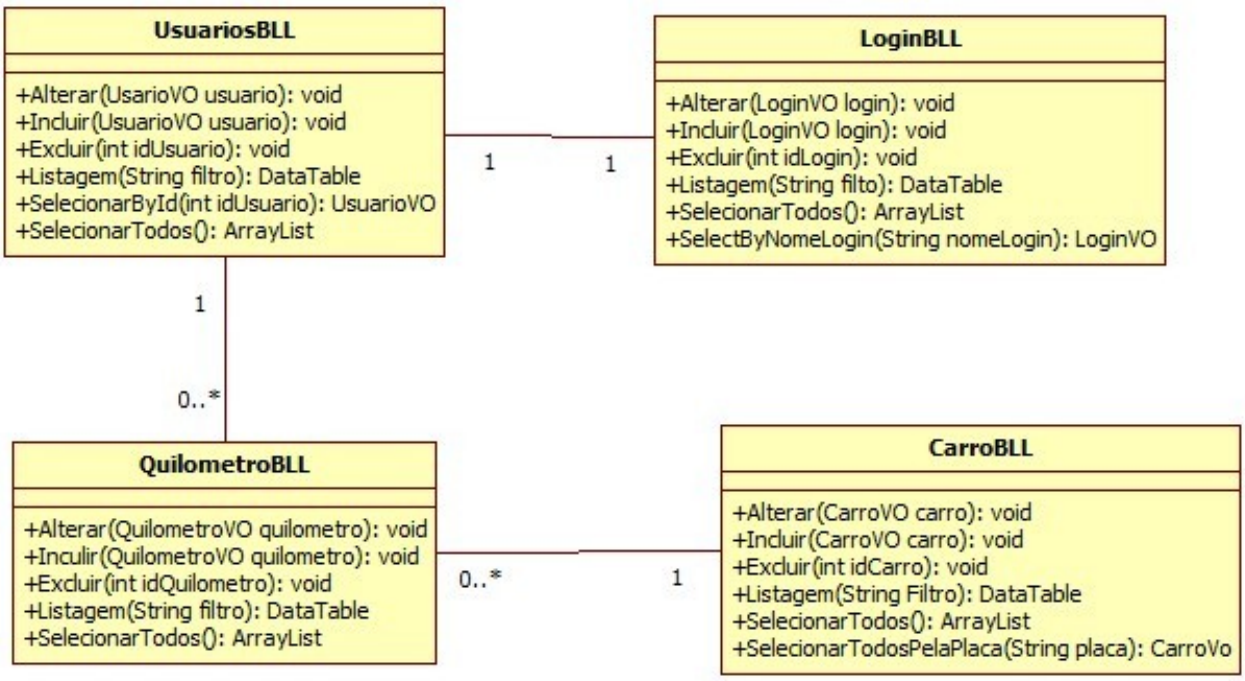


Figura 19 – Diagrama de Classes - Pacote BLL.

3.3.3 – Diagrama de atividades

O diagrama de atividades representa o controle de fluxo das informações entre as atividades de um sistema. Na Figura 20, é apresentado o diagrama de atividades do aplicativo no *desktop* e no dispositivo móvel. O diagrama mostra todos os processos para o cadastro de uma quilometragem de veículo.

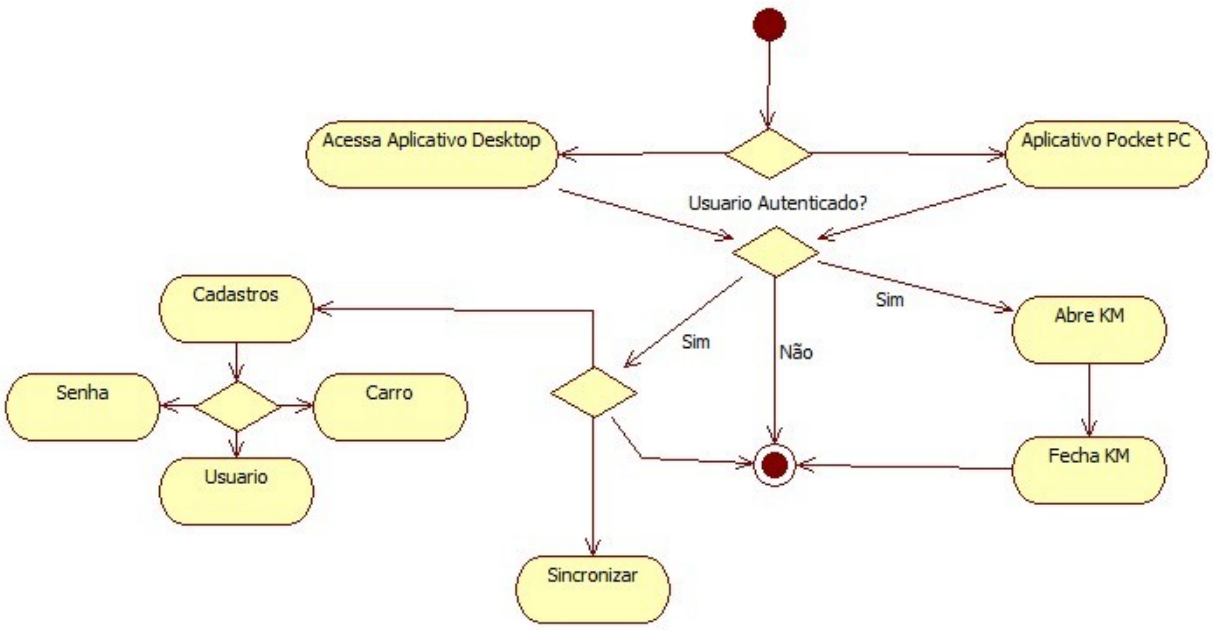


Figura 20 – Diagrama de atividades

3.3.4 – Diagramas de sequência

Os diagramas de sequência representam a sequência das ações ocorridas em um conjunto de classes, demonstrando como ocorre a troca de mensagens entre elas. Será utilizado um diagrama para cada caso de uso especificado: manter usuários, manter carros e manter quilômetros.

3.3.4.1 – Manter usuários

A figura 21 mostra o diagrama de sequência com as ações que podem ser executadas neste caso de uso: inserir, pesquisar, alterar e excluir usuário.

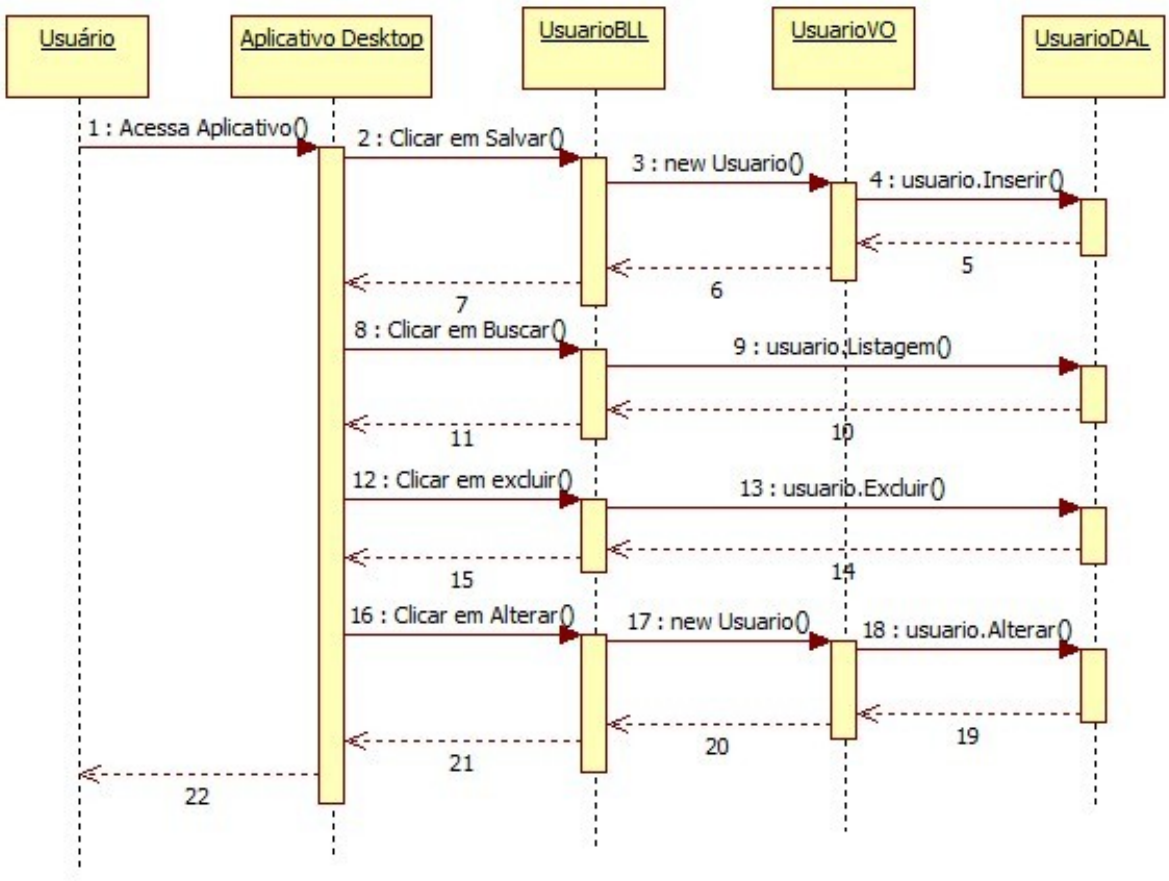


Figura 21 – Diagrama de sequência – Manter Usuários.

3.3.4.2 – Manter Carros

A figura 22 mostra o diagrama de sequência com as ações que podem ser executadas neste caso de uso: inserir, pesquisar, alterar e excluir carro.

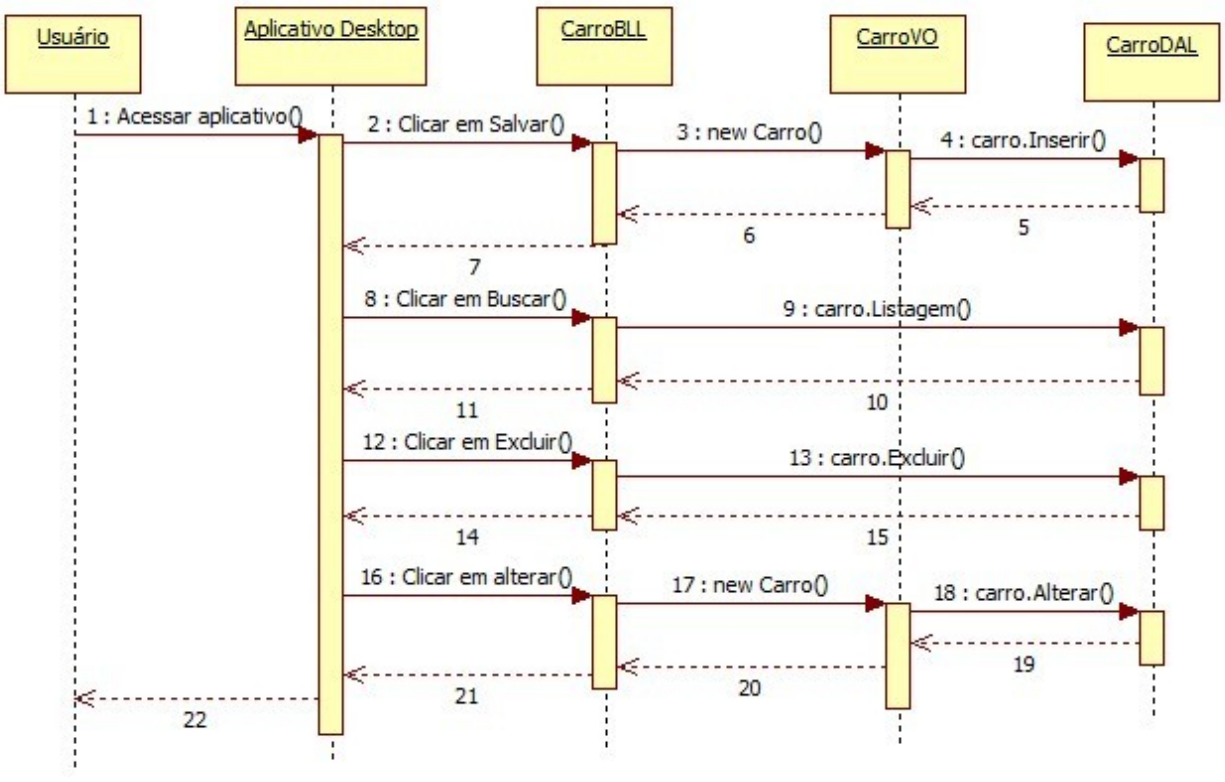


Figura 22 – Diagrama de sequência – Manter Carros.

3.3.4.2 – Manter Quilômetros

A figura 23 mostra o diagrama de sequência com as ações que podem ser executadas neste caso de uso: inserir, pesquisar, alterar e excluir quilômetro.

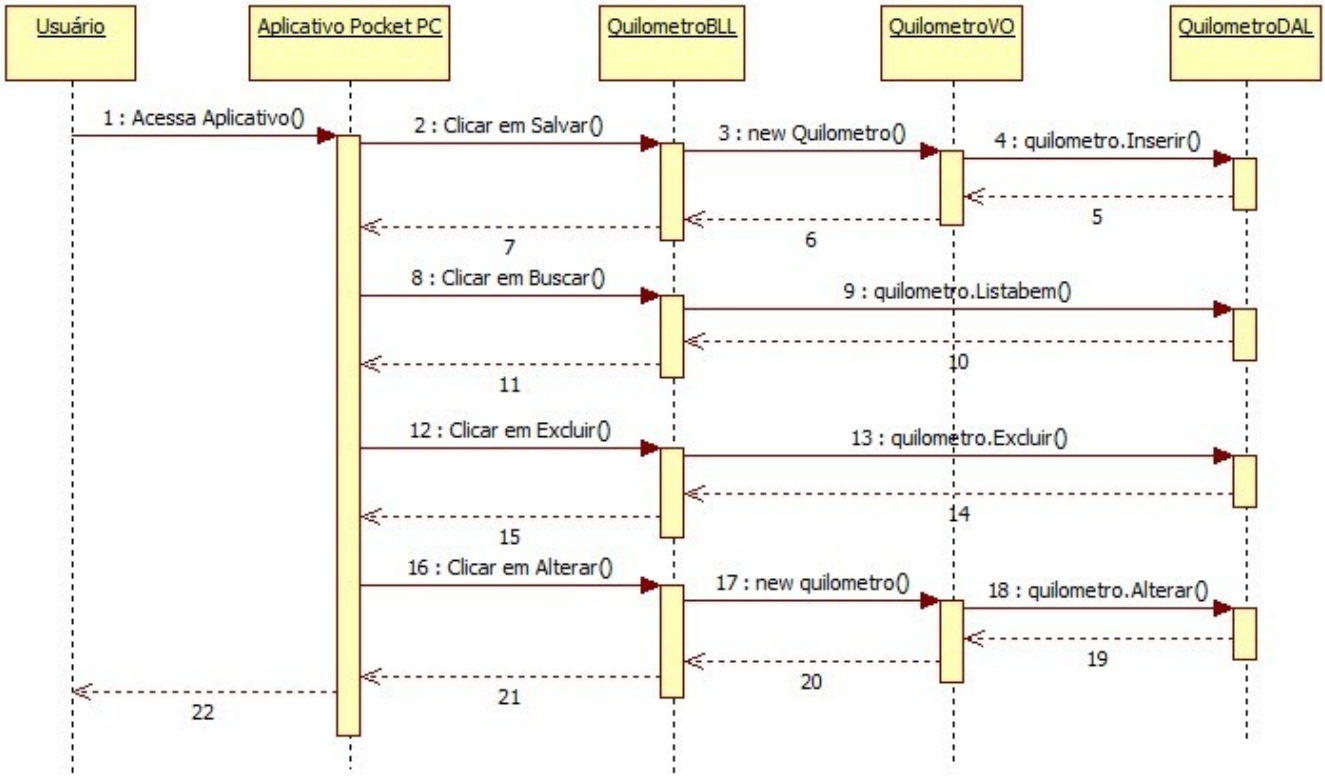


Figura 23 – Diagrama de sequência – Manter Quilômetros.

3.4 - Implementação do Aplicativo

Para a implementação do aplicativo foi utilizado o ambiente de programação Visual Studio .NET 2008.

3.4.1 – Operacionalidade da implementação

Para este aplicativo serão desenvolvidas duas interfaces para interagir com o usuário. Uma será destinada a máquina desktop e a outra para o dispositivo móvel. O ambiente apresenta um servidor de serviço onde são armazenados os serviços sobre a quilometragem do carro. As informações sobre usuários, senhas, caros e quilometragem são armazenadas em um banco de dados. A autenticação do usuário pode ser feita pelo computador *desktop* ou dispositivo móvel.

3.4.1.1 – Caso de uso “Manter Usuários”

O caso de uso “Manter usuário” é utilizado sempre que um usuário precisar incluir, pesquisar, alterar ou excluir um usuário no sistema.

Para se obtenha sucesso na inclusão de um usuário é necessário o preenchimento dos campos: Código, Nome. Abaixo dos campos para o preenchimento do cadastro será disponibilizado uma tabela onde mostrará todos os

registros já incluídos no banco, sendo que para visualizá-los nos campos de texto será preciso apenas um clique em alguma das linhas para que sejam carregadas todas as informações na tela. Para a exclusão é necessário escolher um dos registros e clicar no botão “Excluir”.

3.4.1.2 – Caso de uso “Manter Carros”

O caso de uso “Manter Carros” é utilizado sempre que um usuário precisar incluir, pesquisar, alterar ou excluir um carro do sistema.

Para se obtenha sucesso na inclusão de um carro é necessário o preenchimento dos campos: Código e Placa. Abaixo dos campos para o preenchimento do cadastro será disponibilizado uma tabela onde mostrará todos os registros já incluídos no banco, sendo que para visualizá-los nos campos de texto será preciso apenas um clique em alguma das linhas para que sejam carregadas todas as informações na tela. Para a exclusão é necessário escolher um dos registros e clicar no botão “Excluir”.

3.4.1.3 – Caso de uso “Manter Quilômetros”

O caso de uso “Manter Quilômetros” é utilizado sempre que um usuário precisar incluir, pesquisar, alterar ou excluir quilômetro do sistema.

A inclusão de um quilômetro será apenas permitida no aplicativo do dispositivo móvel onde será necessário o preenchimento dos campos: Nome do Usuário, Placa do Carro, a quilometragem inicial e final, data e hora inicial são inseridas automaticamente, data e hora final são inseridas automaticamente e o destino. No aplicativo desktop será permitida apenas a visualização dos quilômetros cadastrados, podendo ser alterados e excluídos. Onde abaixo dos campos para o

preenchimento do cadastro será disponibilizado uma tabela onde mostrará todos os registros já incluídos no banco, sendo que para visualizá-los nos campos de texto será preciso apenas um clique em alguma das linhas para que seja carregado todas as informações na tela. Para a exclusão é necessário escolher um dos registros e clicar no botão “Excluir”.

3.4.1.4 – Aplicação Desktop

A figura 24 apresenta a interface inicial do aplicativo para o *Desktop*, sendo obrigatório informar um nome de usuário e uma senha para acessar as funcionalidades do sistema.

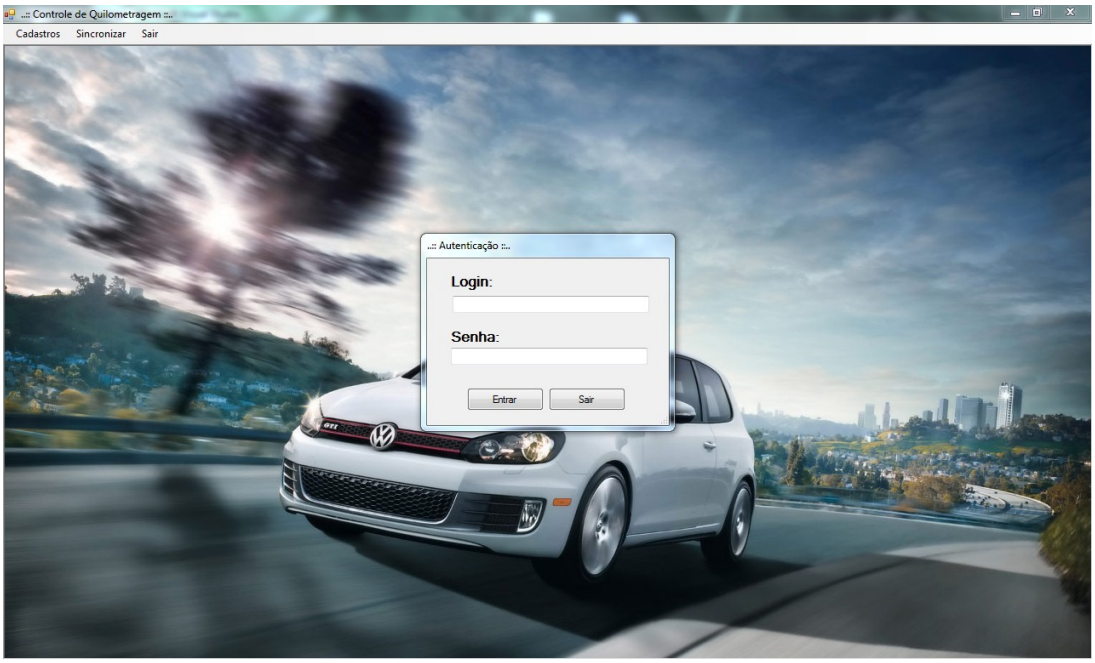


Figura 24 – Interface principal do aplicativo *Desktop*.

A figura 25 representa a interface de cadastro de um usuário com as funções de Incluir, Excluir, Alterar e buscar.

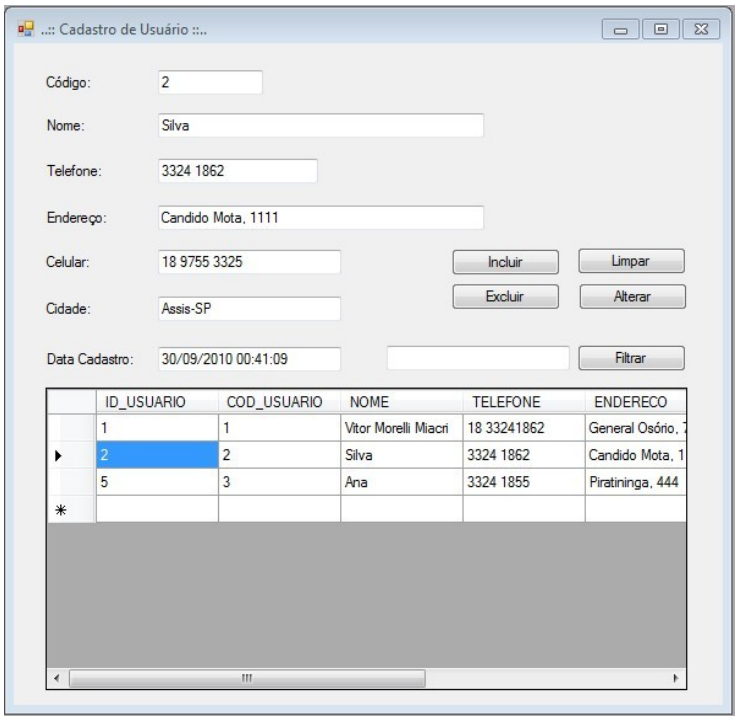


Figura 25 – Interface para as operações com o Usuário no aplicativo *Desktop*.

A figura 26 representa a interface de cadastro de um novo carro. O usuário pode incluir, alterar, excluir e visualizar os carros cadastrados.

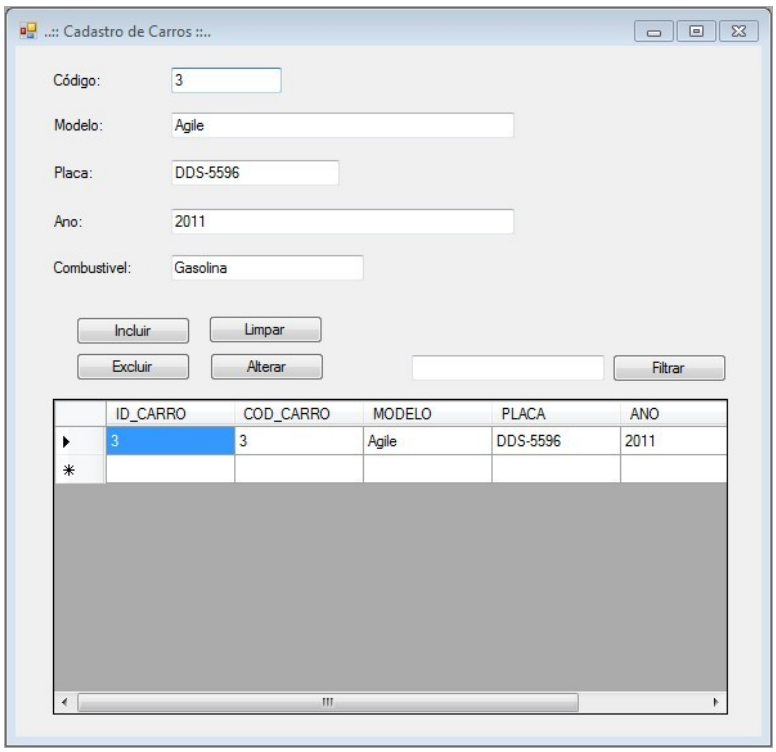


Figura 26 – Interface para as operações com o Carro no aplicativo *Desktop*.

A figura 27 representa a interface de visualização das quilometragens cadastradas. O usuário poderá somente alterar e excluir uma quilometragem.

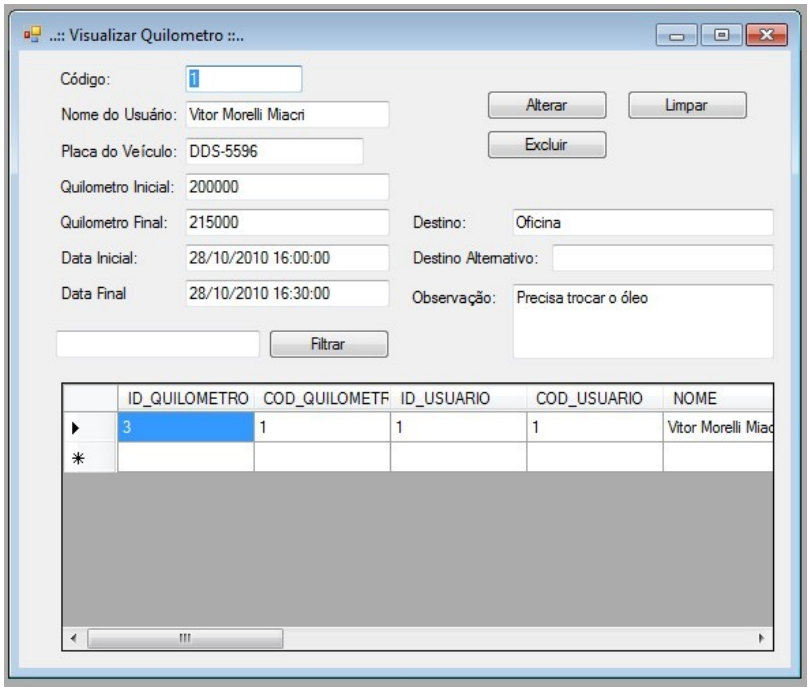


Figura 27 – Interface de visualização de quilometragem no aplicativo *Desktop*

A figura 30 representa a interface de envio dos dados para o *Pocket PC*, onde o usuário deverá conectar o dispositivo com a máquina local para transferir dados do banco local para o banco do *Pocket PC*.

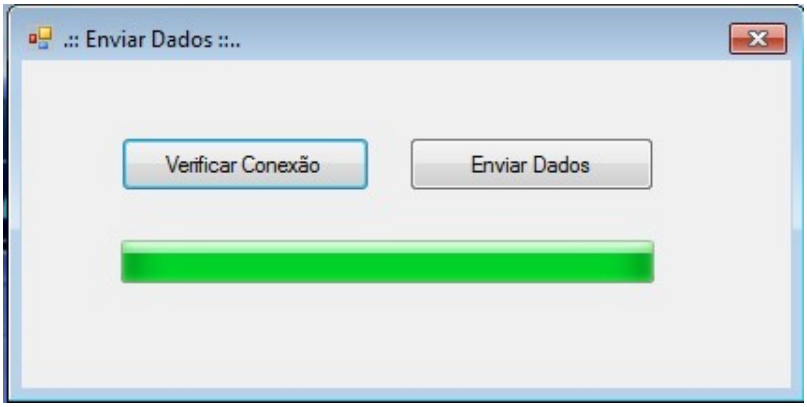


Figura 30 – Tela de visualização de envio de informações no aplicativo *Desktop*

A figura 31 representa a interface de recebimentos dos dados do *Pocket PC*, onde o usuário deverá conectar o dispositivo com a máquina local para transferir dados do banco do *Pocket PC* para o banco do *Desktop*.

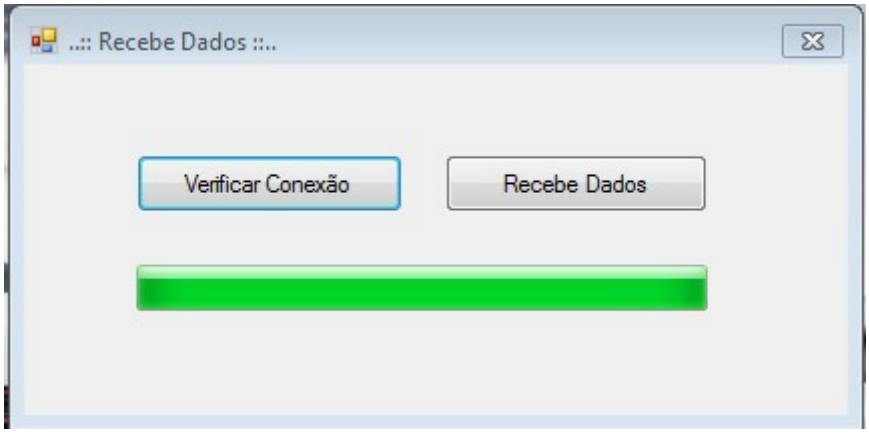


Figura 31 – Interface de visualização de envio de informações no aplicativo *Desktop*

Para implementar todo o processo de sincronização entre o aplicativo do *Pocket PC* com o aplicativo do Desktop foi usada uma biblioteca de comunicação open source chamada *OpenNETCF.Desktop.Communication* onde oferece todos os métodos para desenvolver a comunicação entre os dois aplicativos.

3.4.1.5 – Aplicação *Pocket PC*

A figura 28 representa a interface inicial do aplicativo no *Pocket PC*. Para realizar a autenticação do usuário no aplicativo será necessário o nome de usuário e uma senha. A interface de inclusão de uma nova quilometragem, onde será

mostrado o usuário que acessou o sistema, a placa do veículo e os campos para o preenchimento das informações necessárias para a inclusão. Os campos KM Atual e Destino são obrigatórios.

Figura 28 – A) Interface de *login*. B) Interface de Quilometragem.

A figura 29 representa a interface de menu do aplicativo para o *Pocket PC*, onde o usuário fecha a quilometragem aberta, com a opção de deixar uma observação sobre o veículo. Para sair do sistema será necessário selecionar a opção Sair.

Figura 29 – Interface de menu de fechamento de quilometragem

A figura 30 representa a interface de fechamento de quilometragem. O usuário deverá informar a quilometragem atual do veículo, que não pode ser menor que a quilometragem anterior. Caso o usuário necessite deixar alguma observação sobre o veiculo, basta preencher o campo de Observações.

Figura 31 – Interface de fechamento de quilometragem

CAPITULO 4

CONCLUSÃO

O uso de .NET no desenvolvimento de aplicações para dispositivos móveis pode ser considerado uma excelente opção, pois apresenta grande similaridade com o desenvolvimento de aplicações .NET desktop, o que proporciona uma curva de aprendizado bastante elevada se comparada com outras tecnologias para dispositivos móveis.

O desenvolvimento deste trabalho foi dividido em duas fases: a primeira fase que consistiu em adquirir conhecimento dos métodos e tecnologias envolvidas para o desenvolvimento do aplicativo, enquanto a segunda fase consistiu na implementação do aplicativo.

A proposta de aprender novas tecnologias, principalmente àquelas pouco difundidas no meio acadêmico foi desafiador. Foram encontradas algumas dificuldades no início, mas o resultado obtido foi bastante satisfatório. Alguns conhecimentos adquiridos já estão sendo utilizados profissionalmente.

Uma contribuição importante resultante desta pesquisa é incentivar outras pessoas a desenvolver aplicativos com estas tecnologias. A documentação foi elaborada de maneira a conter as fundamentações teóricas principais.

CAPITULO 5

REFERÊNCIAS BIBLIOGRÁFICAS

(Bonifácio, 2001) Bonifácio Jr, J. M.: *ASP. NET: Fundamentos para o desenvolvimento de aplicações Web em plataforma .NET*. 1. ed. São Paulo: Ed. Berkeley, 2001.

(Cembranelli, 2003) Cembranelli, F.: *ASP.NET: guia do desenvolvedor*. São Paulo: Novatec, 2003.

(Cherry e Demichilli, 2002) Cherry, M.; Demichillie, G.|: *A plataforma de desenvolvimento .NET*. [S.l.]: Microsoft, 2002.

(Conallen, 2003) Conallen, J.: *Desenvolvendo aplicações web com UML*. 2. ed. Tradução Altair D. C.de Moraes; Cláudio B. Dias. Rio de Janeiro: Campus, 2003.

(Dani, 2008) Dani, D.*: Linguagem C#.* 2008.

(Endel a, 2010) http://pt.wikipedia.org/wiki/Apple_Newton. Acesso em junho de 2010.

(Endel b, 2010) http://pdadb.net/index.php?...palm_pilot_1000_5000. Acesso em junho de 2010.

(Endel c, 2010) htttp://ixbtlabs.com/articles2/nec-200/-Estados Unidos. Acesso em junho de 2010.

(Endel d, 2010) http://en.wikipedia.org/wiki/Casio_Cassiopeia. Acesso em junho 2010.

(Endel e, 2010) http://en.wikipedia.org/wiki/Jornada. Acesso em junho de 2010.

(Endel f, 2010) http://pt.wikipedia.org/wiki/IPAQ. Acesso em junho de 2010.

(Endel g, 2010) http://www.apple.com/iphone/gallery/. Acesso em junho de 2010.

(Endel h, 2010) Site brasileiro da comunidade de programadores .NET, http://www.microsoft. com/brasil/msdn/. Acesso Abril 2010.

(Endel j, 2010) Sobre a plataforma .NET Framework. http://www.mcpbrasil. com/integra.asp? id=133. Acesso em abril de 2010.

(Endel k, 2010) Visão geral conceitual do .NET Framework, http://msdn.microsoft. com/pt-br/libr ary/zw4w595w(v=vs.90).aspx . Acesso Maio 2010.

(Endel l, 2010) MSDN - Devices and platforms supported by the .NET compact framework, http://msdn.microsoft.com/en-us/library/ms172550.aspx . Acesso Maio de 2010.

(Endel m, 2010) Arquitetura de armazenamento de dados com o SQL Server 2005 Compact Edition, http://msdn.microsoft.com/library/bb380177.aspx . Acesso maio 2010.

(Endel n, 2010) Introdução ao SQL Server CE. http://www.devmedia.com.br/articles /viewcomp. asp?comp=95. Acesso maio de 2010.

(Endel, i, 2010) Site Oficina da Net, http://www.oficinadanet.com. br/artigo/526/csharp_csharp_ o_que_ e_esta_linguagem. Acesso abril 2010.

(Macdonald, 2002) Macdonald, M.: VB .NET: O essencial de .NET para desenvolvimento VB. 1. ed. Rio de Janeiro: Ed. Ciência Moderna, 2002.

(Siqueira, 2005) Siqueira, J.R.: Programação do Pocket PC com eMbedded Visual Basic, Editora Novatec, 2005.

(Tarifa et al., 2005) Tarifa, A.; Facunde, E.; Garcia, M.: *Visual Basic .NET: desenvolvendo uma aplicação comercial*. Rio de Janeiro: Brasport Livros e Multimídia, 2005.

(Tonon, 2006) Tonon, U. S.*: "Medic Mobile" Aplicação Móvel para Acesso Remoto de Dados Clínicos de Pacientes Hospitalizados*, Projeto de Conclusão de Curso, Centro Tecnológico da Universidade Federal do Espírito Santo, 2006.