UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

WAGNER JACOBI LAMP

APLICAÇÃO DE WEBSERVICES NO ACOMPANHAMENTO E CONTROLE DO ÍNDICE DE MORTALIDADE DE AVES EM AVIÁRIOS

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2015

WAGNER JACOBI LAMP

APLICAÇÃO DE WEBSERVICES NO ACOMPANHAMENTO E CONTROLE DO ÍNDICE DE MORTALIDADE DE AVES EM AVIÁRIOS

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Everton Coimbra de Araújo.

MEDIANEIRA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

Aplicação de webservices no acompanhamento e controle do índice de mortalidade de aves em aviários

Por

Wagner Jacobi Lamp

Este Trabalho de Diplomação (TD) foi apresentado às 15:50 h do dia 09 de junho de 2015 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. Os acadêmicos foram argüidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Everton Coimbra de Araújo
UTFPR – Câmpus Medianeira
(Orientador)

Prof. Dr.. Evando Carlos Pessini
UTFPR – Câmpus Medianeira
(Convidado)

Prof. Me. Juliano Rodrigo Lamb
UTFPR – Câmpus Medianeira
(Convidado)

Prof. Me. Juliano Rodrigo Lamb
UTFPR – Câmpus Medianeira
(Responsável pelas atividades de TCC)

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

LAMP, Wagner Jacobi. Aplicação de web service no acompanhamento e controle do índice de mortalidade de aves em aviários. 2015. 47 f. Trabalho de conclusão de curso (Tecnologia em Análise e Desenvolvimento de Sistemas), da Universidade Tecnológica Federal do Paraná. Medianeira, 2015.

Este trabalho apresenta uma solução para a dificuldade encontrada em acompanhar e arquivar dados de mortalidade no período em que o lote de aves está em fase de crescimento no aviário, pois atualmente os dados são coletados diariamente pelo responsável por meio da ficha de mortalidade, que são somente enviadas ao setor responsável no momento da entrega do lote de aves para o abate, impossibilitando as medidas corretivas durante o processo, como ajustes na temperatura e umidade do ar no aviário para o valor considerado ideal de acordo com a idade do lote de aves, isso se for encontradas irregularidades de valor fora da faixa considerada ideal. O objetivo desse estudo foi aplicar tecnologias tais como web services, platafoma .NET, linguagem de programação C#, ASP .NET Web API, ADO.NET *Framework* e *Windows Phone App*, para desenvolver uma aplicação de web service para dispositivos com sistema operacional *Windows Phone 8.0*, melhorando assim a maneira de se coletar e enviar os dados referentes à mortalidade para o setor responsável. Agora com o software, os mesmos são transmitidos diariamente por meio do web service, o que possibilitou as ações de medidas corretivas acima citadas como também, foi possível medicar o lote de aves contra peste em tempo hábil, consequentemente diminuindo a perda de aves para o abate.

Palavras-chave: Aves. Mortalidade. Web service. Pintainhos.

ABSTRACT

LAMP, Wagner Jacobi. Service web application monitoring and control of bird mortality in aviaries. 2015. 47 f. Trabalho de conclusão de curso (Tecnologia em Análise e Desenvolvimento de Sistemas), da Universidade Tecnológica Federal do Paraná. Medianeira, 2015.

This paper presents a solution to the difficulty in tracking and archiving mortality data in the period when the batch of birds is growing in the aviary because currently the data is collected daily by the head through mortality form, which are only sent to the responsible department at the time of delivery of the batch of poultry for slaughter, preventing corrective measures during the process, as adjustments in temperature and humidity in the aviary for the considered ideal value according to the age of lot of birds, if this value of irregularities found outside the healthy range. The aim of this study was to apply technologies such as web services, platafoma .NET programming language C #, ASP .NET Web API, ADO.NET Framework and Windows Phone App, to develop a web service application for devices with Windows Phone OS 8.0, thus improving the way to collect and send data about mortality for responsible industry. Now with the software, they are transmitted daily via web service, which enabled the actions of the aforementioned corrective measures as well, it was possible to medicate the lot of birds against plague in a timely manner, thus reducing the loss of birds to slaughter.

Keywords: Birds. Mortality. Web service. Chicks.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Programming Interface*
ASP	*Active Server Pages*
CEP	Código de Endereçamento Postal
CLR	*Common Language Runtime*
CLS	*Common Language Specification*
CORBA	*Common Object Request Broker Architecture*
CRUD	*Create, Read, Update e Delete*
DCOM	*Distributed Component Object Model*
HTTP	*Hyper Text Transfer Protocol*
IDE	*Integrated Development Environment*
IIS	*Internet Information Services*
JIT	*Just in Time*
JSON	*JavaScript Object Notation*
LINQ	*Language Integrated Query*
MSIL	*Microsoft Intermediate Language*
MVC	*Model View Controller*
ORM	*Object Relational Mapping*
RAD	*Rapid Application Development*
REST	*Representational State Transfer*
RMI	*Remote Method Invocation*
SDK	*Software Development Kit*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*

URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*
W3C	*Word Wide Web Consortium*

SUMÁRIO

1INTRODUÇÃO

O acesso a Internet e a rede de telefonia tanto fixa como móvel, não se limitam mais aos grandes centros, também se expandiram para o interior e áreas rurais, facilitando e dando praticidade para se comunicar e realizar tarefas. Este fato está atraindo empresas para a busca de soluções tecnológicas para desenvolver seus processos, favorecendo não somente a mesma, mas também a seus clientes.

Observando que no ambiente onde foi realizado o estudo, a ficha de controle de mortalidade para aviários atualmente é preenchida pelo produtor, de maneira manuscrita, ao longo da incubação do lote de aves, até a entrega do mesmo para o abate. E que só nesse momento estas informações são enviadas pelo fomento avícola para o setor responsável de uma empresa. Depois da entrega da ficha é realizada uma análise, onde é possível a empresa acompanhar o rendimento do lote. Desta maneira as medidas corretivas quanto a regulagens de temperatura, umidade e medicações contra pestes, podem ser tomadas somente a partir do final do processo de criação das aves.

Este trabalho apresenta uma aplicação móvel desenvolvida, que se enquadra neste contexto, que possivelmente substituirá a atual ficha de controle de mortalidade para aviários, visando, desta maneira, acompanhar ou até mesmo tratar problemas que possam ocorrer no período de alojamento das aves nos aviários.

O aplicativo desenvolvido possui conceitos cliente/servidor, onde o cliente é executado em dispositivos móveis (celulares/*smartphones*), pois é a tecnologia de maior facilidade de acesso, uso e menor custo se comparando a *notebooks* (MANIA DE CELULAR, 2012).

A aplicação móvel a que este trabalho se refere, foi elaborada para atuar no sistema operacional *Windows Phone*, não descartando a possibilidade do desenvolvimento de novas versões para outros sistemas operacionais para também consumir os recursos do web service confeccionado. Neste aplicativo o produtor registra dados referentes à mortalidade das aves, no seu próprio dispositivo móvel e se já estiver conectado a rede, ou quando estiver, poderá entrar em sincronia, enviando os mesmos para o servidor que fica hospedado na empresa já se integrando com outros sistemas da mesma.

1.1 OBJETIVO GERAL

Apresentar uma solução desenvolvida com ASP.NET Web API (*Application Programming Interface*) e Windows *Phone App* para melhorar o acompanhamento e o processo de coleta de dados da ficha de controle de mortalidade para aviários.

1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, foram necessárias as seguintes etapas:

- Levantamento e apresentação das tecnologias que foram utilizadas no desenvolvimento do sistema.
- Desenvolvimento e apresentação de uma análise de requisitos para o sistema;
- Implementação e realização de testes funcionais no sistema em um ambiente virtual;

1.3 JUSTIFICATIVA

A aplicação móvel desenvolvida e apresentada neste trabalho visa acompanhar não só os registros de mortalidade, mas também a temperatura, umidade do ar encontradas no aviário e algumas observações relevantes para o bom desenvolvimento do lote de pintainhos, no período de alojamento dos mesmos no aviário. É desejável de que estas informações sejam enviadas pelo produtor do seu *smartphone* por meio do web service uma vez ao dia, sendo que estas informações serão acompanhadas pelo setor de controle de qualidade em outros sistemas, isso devido a integração realizada pelo web service.

Após o envio das informações referente à mortalidade do lote de aves, o setor responsável pelo acompanhamento do rendimento do lote de aves poderá acompanhar as ocorrências praticamente em “tempo real” se comparado à ficha de mortalidade entregue no momento do envio lote de aves para o abate, podendo encontrar irregularidades quanto a fatores que possuem uma faixa de valor considerada ideal para um bom desenvolvimento e crescimento das aves, desta maneira será possível comunicar o produtor ou incumbir um colaborador para executar ações corretivas como regulagem de temperatura, na umidade de ar do aviário, intervir com medicações necessárias ou até mesmo manutenções no aviário que busquem reduzir a perda de aves no lote, diminuindo assim o prejuízo, e com base nessas informações buscar reduzir, ou até mesmo inibir o índice de mortalidade de aves em aviários.

Caracterizado pela existência de vários computadores clientes e servidores conectados entre si por um meio de comunicação, o paradigma de construção de sistemas distribuídos cliente servidor é comandado principalmente pelos computadores clientes, sendo que estes recorrem ao servidor quando for necessário executar alguma ação específica (CASETTI; AKAMATU; KIRNER, 1993). O aplicativo, de certa maneira, poderá ser uma parceria entre o produtor e a empresa de abate, cuja intenção é aumentar a produtividade e manter a empresa informada do rendimento dos lotes, consequentemente melhorando os lucros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos teóricos considerados relevantes, para uma melhor compreensão do funcionamento do ciclo de produção de aves para o abate como também de algumas tecnologias.

2.1 AS INFLUÊNCIAS NO PROCESSO DE PRODUÇÃO DA AVICULTURA

De acordo com Bueno et. al. (2007), o processo de criação de aves se inicia no Avozeiro, onde estão as galinhas avós geradas por ovos importados e de linhagem que irão produzir ovos para originar os pintainhos para o abate, já os ovos para produção dos pintainhos são gerados no Matrizeiro, e no Incubatório ficam armazenados os ovos que resultarão nos nascedouros, primeiro lugar onde ficam os pintainhos de corte, sendo que após algumas horas os pintainhos são destinados ao aviário.

Durante o período em que permanecem no aviário os frangos recebem vacinas, água, ração e medicamentos. O desenvolvimento do lote é acompanhado diariamente por meio da ficha de controle de mortalidade para aviários. Nesta ficha são anotados dados como temperatura, aviário, lote, umidade do ar, data, número de mortes e grupo em que se enquadra o fato: morte (quando ocorre de forma natural), eliminadas (aves descartadas) e ataque (ataque do coração). A Figura 1 apresenta de forma simples e clara o processo de criação de frangos para o abate.

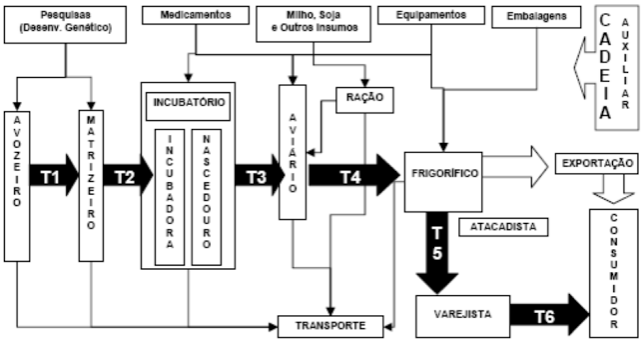


Figura 1 – Processo de criação de frangos para o abate

Fonte: Navegando e Aprendendo (2011)

Segundo Portal Suínos e Aves (2014), o manejo e o cuidado com os frangos, são as principais influências no lucro ou prejuízo no setor da avicultura. Possuindo um ciclo de produção rápido, em menos de 50 dias tem-se um lote de aves pronto para o abate, e este lote já cede seu lugar a outro.

Em muitas empresas a ficha de controle de mortalidade atualmente é um documento preenchido de maneira manuscrita pelo produtor e acompanhado por um médico veterinário responsável por orientar e acompanhar o aviário, como pode ser observado na Figura 2, a ficha preenchida com dados fictícios.

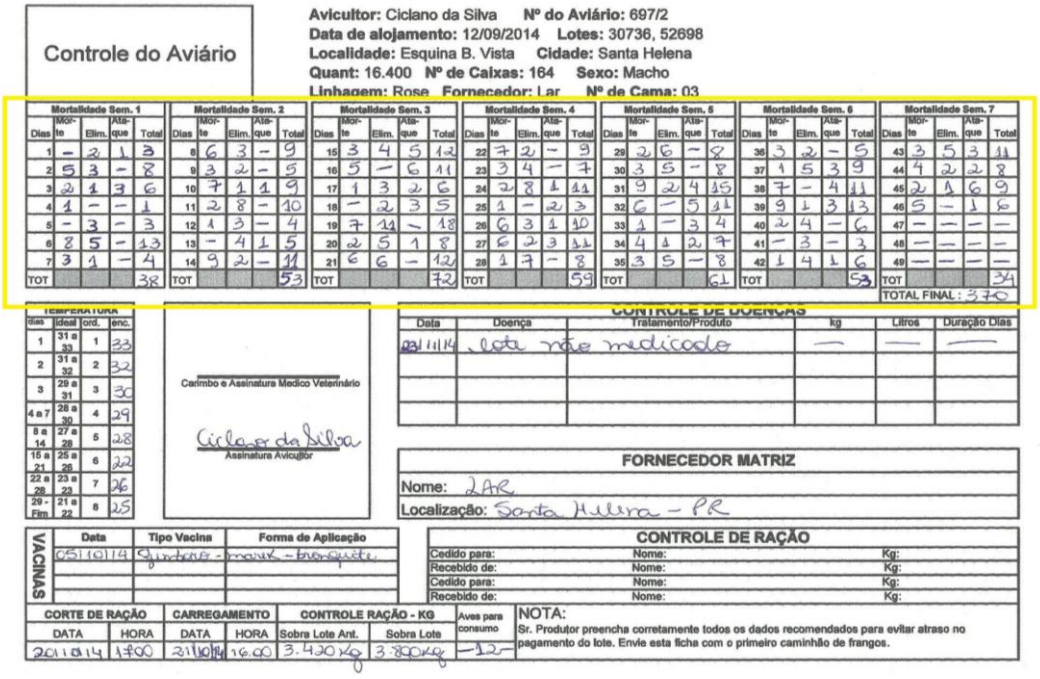


Figura 2 – Ficha de mortalidade para aviários

Fonte: Autoria Própria

A ficha de mortalidade para aviários apresenta o registro diário de mortes do lote de aves alojado no aviário, contendo também informações referentes a controle de doenças e ração. Enviada para a empresa no final do ciclo de vida do lote no aviário, estas informações são digitadas no sistema da empresa, estando suscetível a erro de digitação ou a informação estar ilegível.

2.2 PLATAFORMA .NET E LINGUAGEM C#

Criado pela *Microsoft*, a plataforma .NET teve um foco diferenciado de outras tecnologias, pois seu eixo central foi direcionado a internet para realização dos negócios, sendo isto possível por meio dos Web Services (DANTAS, 2015).

Conforme Recio e Provencio (2008), a plataforma .NET possui um conjunto de serviços e linguagens que evitam um enorme esforço para o desenvolvimento de aplicações. Os principais componentes deste ambiente são as linguagens de compilação, biblioteca de classe .NET e CLR (*Common Language Runtime*). Um *framework* é uma arquitetura representada por um conjunto de classes abstratas e concretas, com enorme potencial, que foi desenvolvida com intenção de atingir a máxima reutilização, sendo a base "a base de sustentação para o desenvolvimento de sistemas no contexto atual, segundo o paradigma de orientação a objeto" (SILVA, 2004).

Considerado o núcleo do *framework* .NET, o CLR é o ambiente onde são executadas as aplicações desenvolvidas em todas as linguagens suportadas pela plataforma. O código fonte de qualquer linguagem aceita pelo .NET é compilado pela ferramenta de desenvolvimento, tornando um mesmo código chamado MSIL (*Microsoft Intermediate Language*) ou código intermediário, regido nas especificações da CLS (*Common Language Specification*), que são as normas necessárias para criar o código compatível com CLR, sendo que este não é um código de máquina, não podendo executá-lo diretamente, necessitando do compilador JIT (*Just in Time*) que gera o código de máquina real permitindo executá-lo em qualquer plataforma, pois cada plataforma pode possuir seu compilador JIT, e por meio MSIL produz seu código fonte (RECIO; PROVENCIO,2008).

Inclusa no *Windows* SDK (*Software Development Kit*), a biblioteca de classes do .NET *Framework* disponibiliza acesso aos recursos do sistema. Foi desenvolvida com intuito de ser o alicerce para elaborar aplicativos e soluções (MICROSOFT, 2015).

Segundo Sant´anna (2000), .NET "É uma plataforma completa, que incorpora de maneira equilibrada os grandes avanços no ramo de desenvolvimento de software dos últimos anos, de uma maneira que só a maior empresa de software do mundo poderia oferecer". A seguir pode-se observar na Figura 3 a arquitetura do .NET *Framework*, a parte marrom representando a linguagens de compilação, a azul a biblioteca do .NET e o Amarelo o CLR.

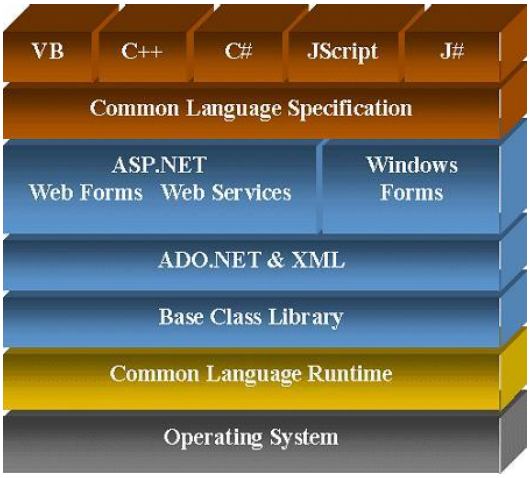


Figura 3 – Arquitetura do .NET *Framework*

Fonte: Baboo (2015)

A linguagem de programação C# é orientada a objetos e possibilita o desenvolvimento de sistemas robustos e seguros, compatíveis com o .NET *Framework*. Uma diversidade de aplicativos podem ser criados com esta linguagem (aplicações cliente-servidor, serviços Web XML e aplicativos de banco de dados) (MICROSOFT, 2013).

Segundo Microsoft (2013), conceitos de polimorfismo, encapsulamento e herança são suportados por C#, podendo uma classe derivada pode herdar mais do que uma classe, pois a herança é transitiva, mas herda diretamente somente uma classe pai. A Figura 4 demonstra um exemplo simples da estrutura da classe Aviário escrita na linguagem C#, com seus atributos (numero e lote) e os métodos *getters* e *setters* dos mesmos.

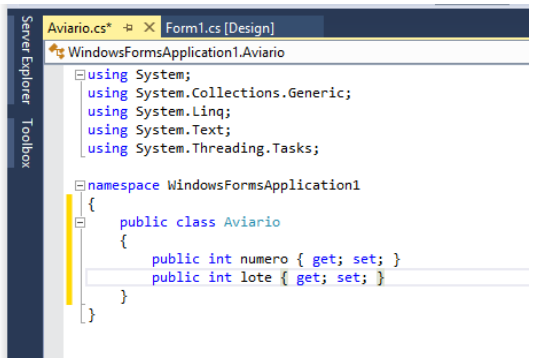


Figura 4 – Estrutura da classe Aviario em C#
Fonte: Autoria Própria

2.3 ASP .NET WEB API

Segundo Aece (2012), muitas funcionalidades do ASP .NET Web API foram aproveitadas do ASP .NET MVC (*Model View Controller*), sendo que entre elas se encontra a capacidade de envio de tipos complexos para o serviço, trabalhando na maioria das vezes com JSON e XML. Quando um formulário é submetido ao servidor todas as informações chegam por meio de um dicionário ligado a uma chave. O ASP .NET MVC instancia a classe baseada no corpo da requisição, poupando trabalho para o programador, sendo que os responsáveis por isso são os *models binders*.

Aece (2011) diz que uma classe abstrata denominada *MediaTypeFormatter*, foi criada pela Microsoft para moldar os *media types* para os serviços, com algumas implementações inclusas na API tais como *JsonMediaTypeFormatter* e *XmlMediaTypeFormatter*. De acordo com o *Content-Type* declarado na requisição, é escolhido o *formatter* para extrair as informações, continuando com o envio das mesmas para o *model binder* que cria o objeto e atribui seus devidos valores (Aece, 2012).

ASP .NET Web API é um *framework* que simplifica, o desenvolvimento de serviços REST baseados em *Hyper Text Transfer Protocol* (HTTP). Serviços que abrangem uma diversidade de clientes tais como, aplicações locais, *Browsers* ou *Mobile* (PIRES, 2013). Como pode-se notar, a Figura 5 apresenta a estrutura de uma aplicação ASP .NET Web API.

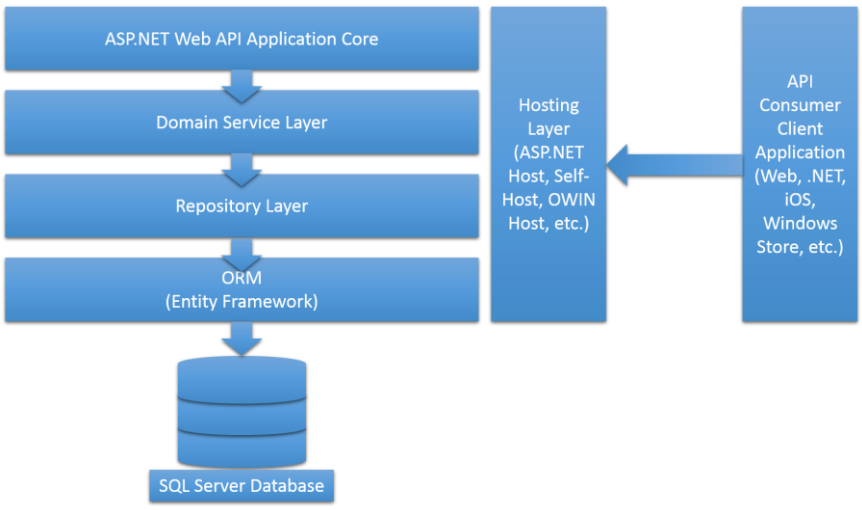


Figura 5 – Estrutura de uma aplicação ASP .NET Web API

Fonte: StackOverFlow (2013)

2.4 ADO .NET ENTITY FRAMEWORK

Este *Framework* do tipo ORM (*Object Relational Mapping*) é suportado por uma série de servidores de banco de dados, estando inclusos nesta, o Oracle, DB2 e SQL Server. A utilização *Entity Framework* no desenvolvimento de aplicações reduz o tempo do processo, pois já fornece as operações básicas sobre os dados, tendo suporte a LINQ (*Language Integrated Query*) para realização de consultas complexas. (DANRESA, 2015).

Conforme Araujo (2015), o encapsulamento do acesso ao banco de dados e geração de sentenças SQL (*Structured Query Language*) ficam por conta do ADO.NET *Framework*, deixando assim uma certa independência para a criação da aplicação quando se trata da base de dados, de certa maneira seu foco fica mais direcionado às regras no negócio.

Quando se trata do desenvolvimento de uma aplicação orientada a objetos, o mapeamento objeto relacional é realizado pelo *Entity Framework*, adaptando o banco de dados relacional ao orientado a objetos (ARAUJO, 2015).

2.5 WINDOWS PHONE APP E A IMPORTÂNCIA DA MOBILIDADE EMPRESARIAL

Windows Phone App é o tipo de aplicação destinada à plataforma *Windows Phone* desenvolvida pela *Microsoft*, é uma espécie de versão móvel do *Windows*, que atua em dispositivos tais como, *smartphones* ou até mesmo em outros dispositivos com apoio de um emulador (TECHTUDO, 2015).

De acordo com Herbert (2015), empresas líderes sabem que a mobilidade gera uma considerável diferença das mesmas perante concorrência, contribuindo para o direcionamento de inovações.

Solutis (2015), diz que conforme a pesquisa realizada por Kelton Research da *Sysbase*:

> 63% dos entrevistados disseram que a adoção de aplicações móveis pela empresa é estimulada pela redução de custos, enquanto que 51% declararam que o uso desses softwares é impulsionado pelo incremento na produtividade. Além desses motivos, outros 50% dos gerentes de TI disseram que o uso de aplicativos móveis se deve a pedidos ou necessidade dos empregados e 43%, devido ao aumento da competitividade.

O desenvolvimento de aplicações para dispositivos móveis vem crescendo de uma forma admirável, pois a facilidade e praticidade de acesso às informações que a mobilidade oferta pode melhorar a forma ou modo de realização de processos. Sendo que o melhoramento contínuo de processo hoje em dia não é mais um diferencial, mas sim uma necessidade para manter-se no mercado de trabalho.

Uma considerável quantidade de sistemas operacionais estão atuando nestes dispositivos, sendo que o mesmo pode mudar de acordo com a marca do aparelho, podendo existir escolha de sistema operacional em produtos da mesma marca. Por exemplo, aparelhos *smartphones* da *Apple* é encontrado o *iPhone OS*, já a Nokia possui aparelhos com *Windows Phone* e marcas como HTC e Samsung possuem produtos tanto com *Windows Phone* ou *Android*.

2.6 WINDOWS PHONE

Lançado em 2010 o sistema operacional *Windows Phone* vem ganhando espaço entre os concorrentes, tendo como um diferencial fortíssimo a segurança dos dados de usuários. Sendo isto alcançado por meio de medidas tomadas pelo sistema, tais como: *boot* seguro, uma resistência maior a *malware* e após *boot* ocorre uma varredura para verificar a integridade dos dados contidos no mesmo, como também se ocorreu alguma alteração mal intencionada (CANALTECH, 2014).

Com a interface muito similar a do *Windows 8* e de fácil navegação, esta plataforma não é algo “espantoso” para os usuários, pois muitos já obtiveram contato com o *Windows*, que atualmente é plataforma predominante nos computadores convencionais, tirando assim o “medo” do desconhecido.

De acordo com Canaltech (2014):

> Android e o iOS podem ser os sistemas operacionais mais usados do mundo atualmente, mas o Windows Phone, aos poucos, se consolida como a plataforma preferida dos usuários em várias partes do globo – entre elas o Brasil. Prova disso é o relatório Mobile Phone Tracker, divulgado pelo IDC (International Data Corporation), que constatou: o software da Microsoft para dispositivos móveis já é o segundo mais utilizado entre os brasileiros.

2.7 WEB SERVICES

A popularização da tecnologia de web service já tem se tornando uma realidade nos dias de hoje, as empresas estão adotando a cada dia mais esta tecnologia, devido ao aumento da confiança dos clientes em realizar transações pela internet. Para exemplificar melhor o que é um web service, podemos citar o serviço dos correios disponibilizados nos comércios eletrônicos, onde é informado o CEP (Código de Endereçamento Postal) e é retornado o respectivo endereço (LIMA, 2012).

Segundo Gomes (2010, p. 13), modelos de computação distribuídas que utilizavam tecnologias tais como RMI (*Remote Method Invocation*), DCOM (*Distributed Component Object Model*) e CORBA (*Common Object Request Broker Architecture*), obtiveram sucesso em integrar sistemas em ambientes de redes locais e homogêneos. Com o avanço da internet para o mercado coorporativo, além de expandir a área de integração de sistemas para além da rede local, surge também a necessidade de integrar aplicações diferentes.

Utilizado como solução na integração de sistemas e na comunicação entre aplicações diferentes, os web services representam uma arquitetura para criação de aplicações que podem ser acessadas remotamente, usando diferentes linguagens baseadas em XML (*Extensible Markup Language*). O que diferencia os web services dos sites web comuns é o tipo de informação que podem fornecer, pois a identificação de ambos é por meio da URL (*Uniform Resource Locator*) (KALIN, 2010).

Como cita Kalin (2010, p.1) SOAP (*Simple Object Access Protocol*) é basicamente um dialeto XML, onde documentos são mensagens. O SOAP é um protocolo que foi projeto para invocar remotamente aplicações por meio de CPR (Chamada de Procedimento Remoto) ou troca de mensagens não dependendo de linguagem de programação ou plataforma. A Figura 6 apresenta um exemplo de requisição SOAP, onde é possível observar que a estrutura do envelope possui cabeçalho e corpo.

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 6 – Exemplo de requisição SOAP

Fonte: UFRJ (2015)

De acordo com Reckziegel (2006) a arquitetura dos web services são baseadas na interação entre provedor de serviço, consumidor de serviço e registro de serviço, sendo que operações como publicação, pesquisa e ligação estão inclusas nesta interação. O provedor de serviço é aquele que fornece o serviço que pode ser utilizado por outros, consumidor é qualquer um que utilize um web service, e o registro de serviços é onde o provedor pode interagir com seus web services e os consumidores podem pesquisa-los. A Figura 7 apresenta a arquitetura dos web services.

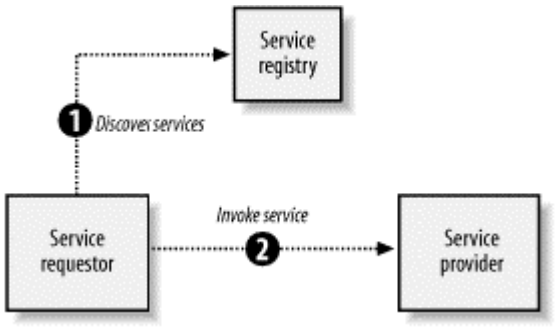


Figura 7 - Arquitetura dos web services

Fonte: Reckziegel (2006)

De acordo com Lima (2012), no ano 2000 em busca de melhores práticas com arquiteturas já existentes, com intuito de agrupa-las e formar uma só arquitetura, Roy Fielding apresentou sua dissertação de Doutorado, que resultou na arquitetura conhecida com REST, neste modelo existem duas definições o cliente e o servidor, onde o cliente é que consome e o servidor e que fornece o serviço.

Lima (2012) ainda ressalta que:

> Clientes e Servidores comunicam-se através de uma rede de computador com hardwares separados, porém o cliente e o servidor podem residir no mesmo sistema. A máquina servidora é um host que está executando um ou mais programas que compartilham os seus recursos com os clientes. O cliente por sua vez não compartilha recursos, ele solicita serviços ao servidor.

Na Figura 8 podemos observar a arquitetura de uma aplicação REST disponibilizando seus recursos a três clientes sendo um notebook e dois *smartphones*, notando também que os dispositivos que estão consumindo o serviço possuem sistemas operacionais distintos.

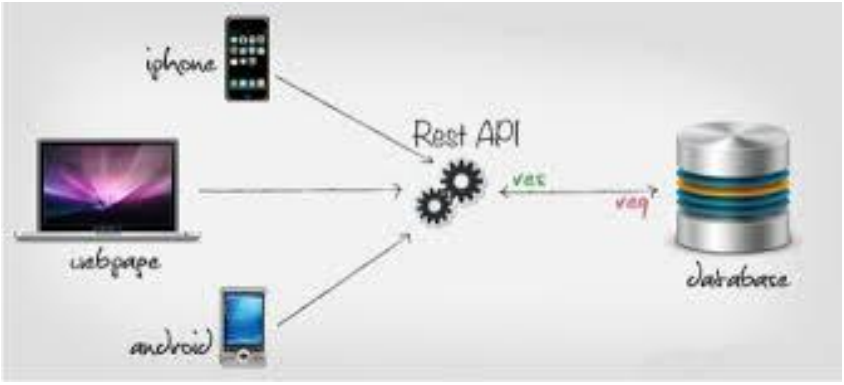


Figura 8 – Arquitetura de uma aplicação REST

Fonte: Paliari (2012)

O estilo REST também é um protocolo de comunicação, porém não possui restrições ao formato da mensagem, na maioria das vezes se comunica por XML ou JSON *(JavaScript Object Notation)*, que é um modelo para armazenamento e transmissão de informações em forma de texto. Para a demonstração de um objeto JSON utilizaremos o exemplo de um contato da agenda telefônica, que possui o nome e o número de telefone, que no formato JSON ficaria assim: {“nome”: “João Paulo”, “telefone:”: “9986-5427”}.

Kalin (2010) cita sua percepção entre os estilos SOAP e REST *(Representational State Transfer)* de web services. SOAP tem padrões no *World Wibe Web Consortium* (W3C), toolkits e nas inúmeras bibliotecas de software. REST não tem padrões, possui poucos toolkits, e escassas bibliotecas de software, o que certa forma acabou melhorando o processo de construção de web services. O estilo REST é frequentemente visto como um antídoto para a complexidade de web services baseados em SOAP (KALIN, 2010, p. 2).

3MATERIAL E MÉTODOS

Este capítulo apresenta as ferramentas utilizadas no desenvolvimento da aplicação proposta neste trabalho.

3.1 MATERIAIS

Na aplicação desenvolvida foi utilizada a análise de requisitos, o *Integrated Development Environment* (IDE) Visual Studio 2013, juntamente com o servidor web *Internet Information Services* (IIS) versão 8.5, Emulador do *Windows Phone* 8.0 e o banco de dados SQLServer 2012, descritos a seguir.

3.1.1 Análise de Requisitos

A Análise de requisitos visa relatar as funcionalidades que o sistema deve possuir como também os atributos que devem ser suportados. De maneira dissertativa ou por meio da UML (*Unified Modeling Language*), os requisitos da aplicação a que se destina este trabalho serão apresentados nos capítulos subsequentes.

3.1.1.1 Sumário Executivo

A análise representada no Quadro 1, visa estabelecer as diretrizes para elaboração de uma aplicação para acompanhamento e controle do índice de mortalidade de aves em aviários.

Sistema de Acompanhamento e Controle do Índice de Mortalidade de Aves em Aviários
Visão Geral O sistema consiste em uma aplicação destinada a dispositivos móveis, onde mediante a ocorrência de mortalidade de aves em aviários, o responsável pelo mesmo informará a data, temperatura, umidade, quantidade, grupo em que se enquadra (morte, eliminada ou ataque) e caso exista, alguma observação, como também o código do aviário e lote corrente. Esses dados serão salvos no ambiente local, e quando estiver conectado a rede poderá entrar em sincronia com o web service, enviando os mesmos para a base de dados alojada no setor de processamento de dados da empresa, já se integrando com outros sistemas por meio do web service, possibilitando um melhor acompanhamento da mortalidade como também a identificação de medidas corretivas (regulagem de temperatura, umidade e medicações contra pestes).

Quadro 1 - Visão Geral do Sistema de Acompanhamento e Controle do Índice de Mortalidade de Aves em Aviários

Fonte: Autoria Própria

3.1.1.1.1 Requisitos Funcionais e Não Funcionais

Os requisitos são objetivos ou restrições estabelecidos por clientes e usuários que definem as diversas propriedades do sistema. Os requisitos funcionais correspondem à listagem de todas as coisas que o sistema deve realizar para atender a necessidade do cliente:

a) F1 – Manter Aviário
b) F2 – Manter Usuário
c) F3 – Manter Lote
d) F4 – Manter Registro de Mortalidade
e) F5 – Enviar Registro de Mortalidade

Já os requisitos não funcionais definem restrições e propriedades do sistema tais como segurança e desempenho. Do Quadro 2 ao Quadro 6 estão sendo apresentados os requistos

funcionais e não funcionais do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários.

F1 – Manter Aviário			**Oculto( )**	
Descrição: O sistema devera realizar as operações CRUD para o Aviário				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF3.1 – Atualização dos dados	Os Aviarios serão somente atualizados no dispositivo móvel se o usuário solicitar a ação e o dispositivo estiver conectado a rede.	Facilidade de Uso	( )	(X)

Quadro 2 - Requisito Funcional Manter Aviário

Fonte: Autoria Própria

F2 – Manter Usuário			**Oculto( )**	
Descrição: O sistema devera realizar as operações CRUD para o Usuário				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF2.1 – Atualização dos dados	Os Usuários serão somente atualizados no dispositivo móvel se o usuário solicitar a ação e o dispositivo estiver conectado a rede.	Facilidade de Uso	( )	(X)

Quadro 3 - Requisito Funcional Manter Usuário

Fonte: Autoria Própria

F3 – Manter Lote			**Oculto( )**	
Descrição: O sistema devera realizar as operações CRUD para o Lote				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF3.1 – Atualização dos dados	Os Lotes serão somente atualizados no dispositivo móvel se o usuário solicitar a ação e o dispositivo estiver conectado a rede.	Facilidade de Uso	( )	(X)

Quadro 4 – Requisito Funcional Manter Lote

Fonte: Autoria Própria

F4 – Manter Registro de Mortalidade		**Oculto( )**		
Descrição: O sistema devera realizar as operações CRUD para o Registro de Mortalidade				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF4.1 – Permissão de acesso	Somente será permitido manter registro de mortalidade aos usuários autenticados.	Segurança	( )	(X)
NF4.2 – Dados do registro de mortalidade	Para cada registro de mortalidade deverá conter o aviário, lote, data temperatura, umidade, quantidade e grupo em se enquadra a morte e se necessário alguma observação.	Usabilidade	( )	(X)

Quadro 5 - Requisito Funcional Manter Registro de Mortalidade

Fonte: Autoria Própria

F5– Enviar Registro de Mortalidade		**Oculto( )**		
Descrição: O sistema devera realizar as operações CRUD para o Registro de Mortalidade				
Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF5.1 – Permissão de acesso	Somente será permitido enviar registro de mortalidade aos usuários autenticados e conectados a rede.	Segurança	( )	(X)
NF5.2 – Dados para o envio	É necessário informar somente a data ocorrência desejada para o envio.	Usabilidade	( )	(X)
NF5.3 – Compartilhamento de dados	A integração dos dados com outros sistema será realizada por meio do web service.	Integração	( )	(X)

Quadro 6 - Requisito Funcional Enviar Registro de Mortalidade

Fonte: Autoria Própria

3.1.1.1.2 Requisitos Suplementares

São requisitos que definem os detalhes de como o sistema deverá ser implementado (linguagem específica), como deverá ser apresentado (via *browser* ou aplicação remota), controle de acesso e modo de armazenamento. Os requisitos suplementares estão apresentados no Quadro 7.

Nome	Restrição	Categoria	Desejável	Permanente
S1 Tipo de Interface	O sistema deverá ser implementado para atuar em dispositivos móveis.	Interface	( )	(X)
S2 Controle de Acesso	Para manter e enviar o registro de mortalidade somente usuários autenticados poderão realizar seus afazeres no sistema.	Segurança	( )	(X)

Quadro 7 - Requisitos suplementares do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários

Fonte: Autoria Própria

3.1.1.2 Organização dos Casos de Uso

A organização dos casos de uso visa demonstrar de forma mais detalhada os casos de usos, conceitos, demonstrando as ações que cada um deverá possuir.

3.1.1.2.1 Casos de Uso

Os casos de uso devem contemplar os processos de negócio identificados na análise de requisitos. O Quadro 8 apresenta o caso de uso do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários.

Nome	Atores	Descrição	Referência Cruzada
Manter Aviário	Usuário e Sistema	Os Atores deverão realizar as operações CRUD para os aviários de acordo com as restrições.	F1
Manter Usuário	Usuário e Sistema	Os Atores deverão realizar as operações CRUD para os usuários de acordo com as restrições.	F2
Manter Lote	Usuário e Sistema	Os Atores deverão realizar as operações CRUD para os lotes de acordo com as restrições.	F3
Manter Registro de Mortalidade	Usuário	Os Atores deverão realizar as operações CRUD para os registros de mortalidade de acordo com as restrições.	F4
Enviar Registro de Mortalidade	Usuário	Os Atores realizar o envio de acordo com as restrições.	F5

Quadro 8 - Caso de uso do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários

Fonte: Autoria Própria

3.1.1.2.2 Conceitos

Compreendem os requisitos que executam operações para consultas, alterações, inserções e exclusão em arquivos ou banco de dados, são apresentados no Quadro 9.

Conceito	Inserir	Alterar	Excluir	Consultar	Observação
Aviário	X	X	X	X	
Usuário	X	X	X	X	
Lote	X	X	X	X	
Registro de Mortalidade	X	X	X	X	Todas as operações sobre o registro de mortalidade poderão ser efetuadas pelo usuário
Enviar Registro de Mortalidade	X	X	X	X	Todas as operações sobre o registro de mortalidade poderão ser efetuadas pelo usuário

Quadro 9 - Conceito dos requisitos do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários

Fonte: Autoria Própria

3.1.1.3 Diagrama de Caso de Uso

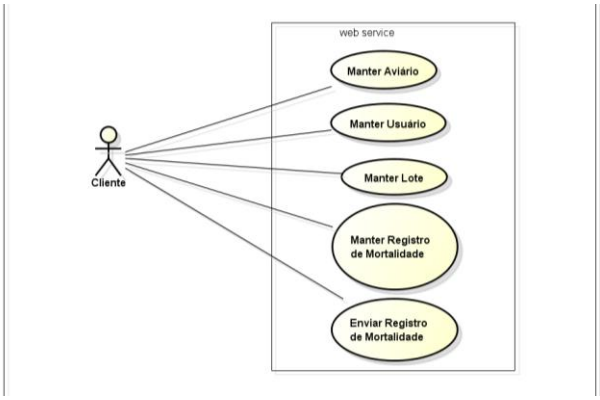


Figura 9 – Diagrama de Caso de Uso do Sistema de Acompanhamento e controle do índice de mortalidade de aves em aviários

Fonte: Autoria Própria

3.1.1.4 Descrição Formal dos Casos de Uso

Do Quadro 10 ao Quadro 14 será apresentada uma descrição formal dos casos de uso do Sistema de Acompanhamento e Controle do Índice de Mortalidade de Aves em Aviários.

Caso de Uso: Manter Aviário
Atores: Usuário e Sistema
Interessados: Usuário e Sistema
Pré-condições: O Usuário deverá manter o Aviário solicitando a atualização dos dados para utilização do sistema.
Pós-condições: O aplicativo móvel armazena os dados dos aviários existentes no servidor, no próprio dispositivo móvel.
Fluxo Principal: **a)** O Usuário solicita a atualização dos dados. (E1) **b)** O sistema busca os dados por meio do web service. **c)** O sistema altera, exclui ou insere no banco de dados do dispositivo móvel todos os usuários armazenados no servidor.
Tratamento de Exceções **E1 –** O dispositivo não está conectado a rede. a) O sistema informa que é necessário conectar-se a rede para efetuar a ação. b) O Usuário conecta-se a rede . c) Volta ao passo "a" do fluxo principal.

Quadro 10 - Descrição formal do caso de uso manter aviário

Fonte: Autoria Própria

Caso de Uso: Manter Usuário
Atores: Usuário e Sistema
Interessados: Usuário e Sistema
Pré-condições: O Usuário deverá manter o Usuário solicitando a atualização dos dados para utilização do sistema.
Pós-condições: O aplicativo móvel armazena os dados dos usuários existentes no servidor, no próprio dispositivo móvel.
Fluxo Principal: **a)** O Usuário solicita a atualização dos dados. **b)** O sistema busca os dados por meio do web service. **c)** O sistema altera, exclui ou insere os dados no dispositivo conforme a necessidade.
Tratamento de Exceções **E1** – O dispositivo não está conectado a rede. a) O sistema informa que é necessário conectar-se a rede para efetuar a ação. b) O Usuário conecta-se a rede . c) Volta ao passo "a" do fluxo principal.

Quadro 11 - Descrição formal do caso de uso manter usuário

Fonte: Autoria Própria

Caso de Uso: Manter Lote
Atores: Usuário e Sistema
Interessados: Usuário e Sistema
Pré-condições: O Usuário deverá manter o Lote solicitando a atualização dos dados para utilização do sistema.
Pós-condições: O aplicativo móvel armazena os dados dos lotes existentes no servidor, no próprio dispositivo móvel.
Fluxo Principal: **a)** O Usuário solicita a atualização dos dados. **b)** O sistema busca os dados por meio do web service. **c)** O sistema altera, exclui ou insere os dados no dispositivo conforme a necessidade.
Tratamento de Exceções **E1** – O dispositivo não está conectado a rede. a) O sistema informa que é necessário conectar-se a rede para efetuar a ação. b) O Usuário conecta-se a rede . c) Volta ao passo "a" do fluxo principal.

Quadro 12 – Descrição formal do caso de uso manter lote

Fonte: Autoria Própria

Caso de Uso: Manter Registro de Mortalidade
Atores: Usuário
Interessados: Usuário
Pré-condições: O Usuário deverá manter o registro de mortalidade informando os dados necessários para ação.
Pós-condições: O sistema exibe o registro de mortalidade
Fluxo Principal: **a)** O Usuário solicita a inserção do registro de mortalidade. **b)** [IN] O Usuário informa os dados do registro (data, temperatura, umidade, números de mortes por grupo e observação) de mortalidade. (E1) **c)** [OUT] O registro de mortalidade é inserido.
Tratamento de Exceções **E1** - Os dados estão incompletos, fora do esperado ou errados. a) Volta ao passo "b" do fluxo principal

Quadro 13 - Descrição formal do caso de uso manter registro de mortalidade

Fonte: Autoria Própria

Caso de Uso: Enviar Registro de Mortalidade
Atores: Usuário e Sistema
Interessados: Usuário e Sistema
Pré-condições: O Usuário deverá informar a data do registro e estar com seu dispositivo conectado a rede para enviar o registro de mortalidade.
Pós-condições: O sistema exibe o resultado da ação.
Fluxo Principal: **a)** O Usuário informa a data. **b)** O sistema verifica se há conexão com a rede. (E1) **c)** O sistema envia os dados para o servidor por meio do web service. **d)** O sistema exibe o sucesso da ação.
Tratamento de Exceções **E1 –** O dispositivo não está conectado a rede. a) O sistema informa que é necessário conectar-se a rede para efetuar a ação. b) O Usuário conecta-se a rede. c) Volta ao passo "a" do fluxo principal.

Quadro 14 - Descrição Formal do Caso de Uso Enviar Registro de Mortalidade

Fonte: Autoria Própria

3.1.1.5 Diagrama de Classes e Diagrama de Componentes

O diagrama de classes apresenta uma visão sobre as classes de um sistema e suas relações, é um diagrama muito importante para o desenvolvimento de softwares, pois representa todas as classes que deverão ser implementadas no sistema, como se observa na Figura 10.

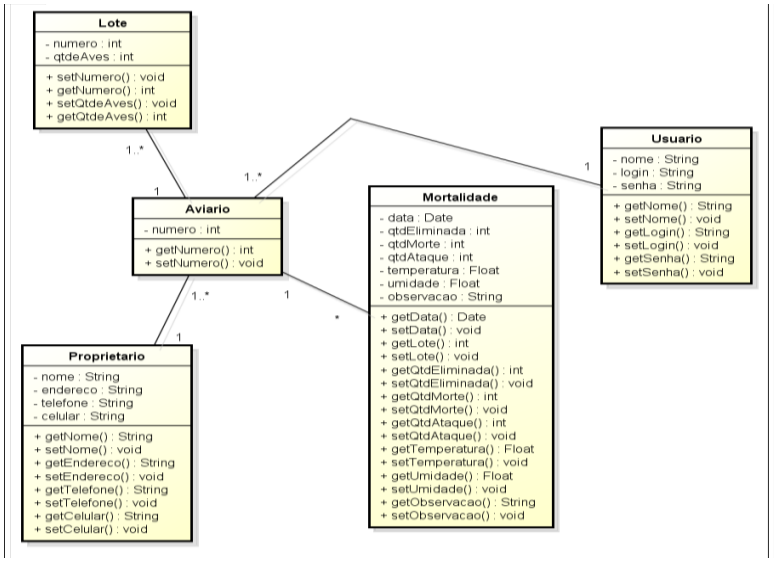


Figura 10 – Diagrama de classes da aplicação para acompanhamento e controle do índice de mortalidade de aves em aviários

Fonte: Autoria Própria

O diagrama de componentes visa demonstrar a arquitetura física da aplicação podendo identificar as principais peças do sistema e suas dependências. A Figura 11 apresenta o diagrama de componentes do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários.

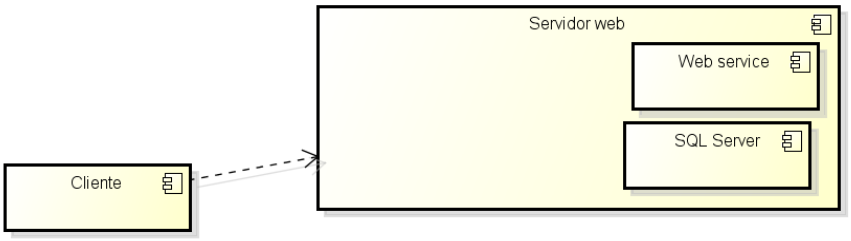


Figura 11 – Diagrama de componentes do sistema de acompanhamento e controle do índice de mortalidade de aves em aviários

Fonte: Autoria Própria

3.1.2 Visual Studio 2013 e Emulador Windows Phone 8.0

Elaborado pela empresa Microsoft, o Visual Studio 2013 é uma ferramenta de desenvolvimento do tipo *Rapid Application Development* (RAD), atuando na plataforma .NET ela oferece suporte tanto ao trabalho individual como em equipe. Possibilita o uso de várias linguagens de programação como Visual Basic, Visual C#, Visual C++ e Visual J#, que por meio do .NET *Framework* alcançam tecnologias que simplificam a criação de aplicativos sofisticados (GROFFE,2013).

A Figura 12 apresenta a interface do Visual Studio 2013, que conforme Microsoft (2015) trouxe melhorias como o desenvolvimento multiplataforma e o acesso e modificação de aplicações na nuvem por meio de *login* utilizando o navegador. O desenvolvedor pode compilar códigos para *Windows Phone*, *IOS*, *Mac* e Windows.

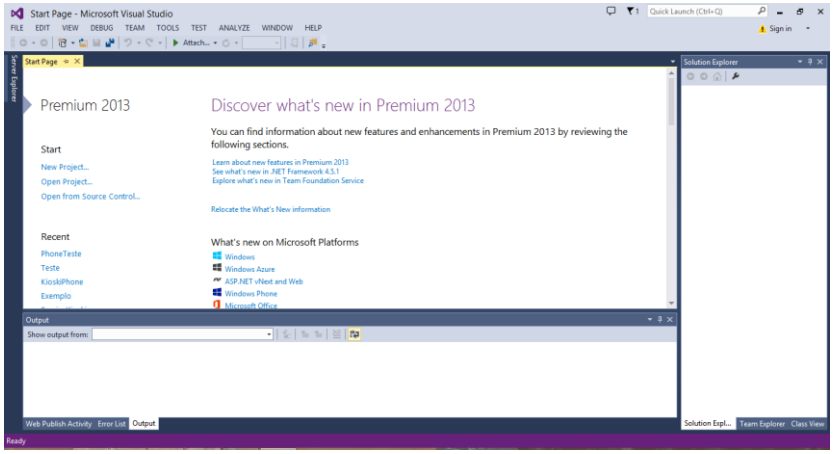


Figura 12 – Interface Visual Studio 2013

Fonte: Autoria Própria

O Quadro15 apresenta os requisitos para um bom funcionamento do Visual Studio 2013 tais como sistema operacional e *hardware*. Alguns requisitos adicionais também são necessários para o funcionamento pleno da ferramenta.

Sistema Operacional	Windows 8.1 (x86 e x64), Windows 8 (x86 e x64), Windows7 SP1 (x86 e x64), Windows Server 2012 R2 (x64), Windows Server 2012 (x64) ou Windows Server 2008 R2 SP1 (x64)
Processador	1,6 GHz ou superior
Memória	1 GB de RAM ou mais
Disco Rígido	10 GB de espaço disponível no disco e velocidade de 5.400 RPM
Vídeo	Resolução de tela 1024 x 768 ou superior, e placa de vídeo compatível com DirectX 9

Quadro 15 – Requistos mínimos para executar o visual studio 2013

Fonte: Microsoft (2015)

A aplicação desenvolvida foi executada utilizando o emulador do *Windows Phone* 8.0, oferecido pelo SDK do *Windows Phone* no Visual Studio, devido ao fato de não necessitar de um dispositivo físico com o sistema. Pois o Visual Studio possui quatro imagens com o sistema operacional sendo elas o *Emulator* WVGA 512 MB, *Emulator* WVGA, *Emulator* WXGA e *Emulator* 720P, facilitando assim o processo de teste. A Figura 13 apresenta um aparelho exibido pelo emulador.

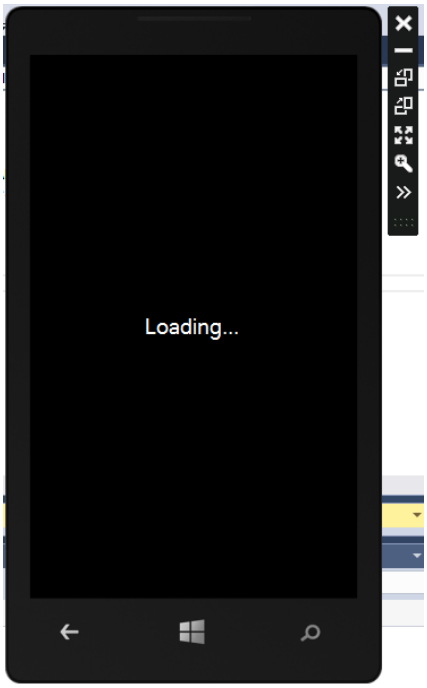


Figura 13 – Emulador Windows Phone 8.0
Fonte: Autoria Própria

3.1.3 SQL Server 2012 e SQLite

SQL Server 2012 é Sistema Gerenciador de Banco de dados Relacional desenvolvido pela *Microsoft,* e que utiliza SQL para a manipular dos dados, (WEB, 2015). "O SQL Server oferece desempenho crítico com tecnologias integradas *in-memory*, *insights* mais rápidos a partir dos dados com ferramentas de análises conhecidas, como o Excel, e uma plataforma para criar, implantar e gerenciar soluções locais e na nuvem" (MICROSOFT, 2015). É possível observar na Figura 14 a interface do Microsoft SQL Server *Management Studio.*

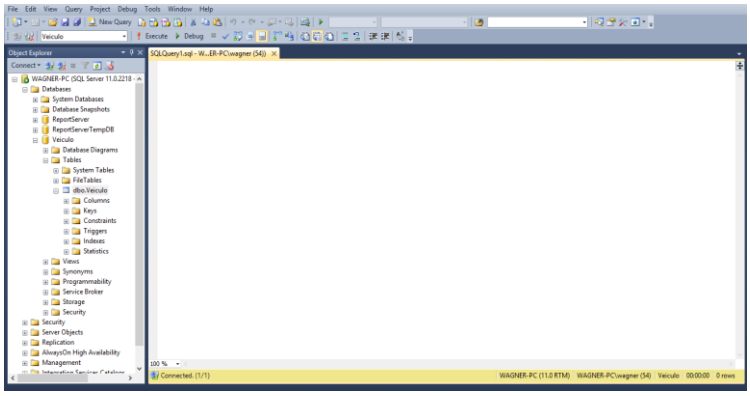


Figura 14 – Interface do Microsoft SQL Server Management Studio

Fonte: Autoria Própria

Desenvolvido na linguagem C, SQLite é um banco de dados relacional multiplataforma e *open source*, muito leve, que utiliza comandos SQL. Apesar de sua maior utilização se encontrar em aplicações pequenas geralmente destinadas a *smartphones* e outros dispositivos móveis, mas existem interfaces que dão suporte a outras linguagens com *Python*, PHP e Perl (CAVALCANTI, 2011).

3.1.4 IIS

Criado pela *Micorsoft* o IIS é um servidor web destinado aos sistemas operacionais da mesma, a geração de páginas HTML dinâmicas é uma de suas características mais utilizadas, se diferenciando de outros servidores pois possui sua própria tecnologia o *Active Server Pages* (ASP), mas nada impede que utilize tecnologias de terceiros, claro desde que adicione o módulo necessário para a utlização, gerencia ASP .NET também é uma tecnologia própria e pode ser tanto uma página web como um web service (OFICINA DA NET, 2010).

3.2 MÉTODOS

Para essa pesquisa foi utilizado o método de observação direta intensiva com entrevista, que segundo Marconi e Lakatos (2003, p. 222) se aproveita dos sentidos para obter

determinados aspectos da realidade, estudando os fatos e fenômenos que se deseja analisar. A entrevista é realizada em uma conversação já elaborada anteriormente, com o foco de se obter os dados necessários para continuidade e aperfeiçoamento do projeto.

O estudo foi realizado por meio de um acompanhamento do processo de coleta e análise dos dados referentes à mortalidade de aves em aviários, visando obter as informações necessárias para o desenvolvimento da solução.

Já a aplicação foi desenvolvida no método tradicional, utilizando a linguagem de programação orientada a objetos C# da plataforma .NET, através da *Integrated Development Environment* (IDE) Visual Studio 2013. O web service é do estilo REST, sendo um dos recursos oferecidos no ASP.NET Web API. A mesma linguagem de programação citada anteriormente foi aplicada no desenvolvimento da Windows Phone App, para persistência dos dados foi utilizado ADO.NET *Entity Framework* e como banco de dados foi utilizado SQL*Server* para aplicação REST e para a *Windows Phone App* o banco de dados SQLite.

A solução apresentada neste trabalho foi aplicada em um ambiente de teste virtual, com intuito de observar seus respectivos resultados, para que se necessário aprimorá-los. Com chances de futuras implantações em ambientes reais.

4RESULTADOS E DISCUSSÃO

O passo inicial para o desenvolvimento deste trabalho foi realizar uma análise no funcionamento do processo de criação de aves para o abate, com intuito de obter as informações fundamentais com necessidade de envio diário para o acompanhamento e controle do índice de mortalidade de aves em aviários.

Após a realização da análise, já com as informações obtidas sobre os dados e regras do negócio para elaboração do sistema, o passo seguinte foi criar o diagrama de casos de uso, o digrama de classes que apresenta as classes e as associações necessárias entre as mesmas para o funcionamento da aplicação, sendo esta uma representação de forma gráfica. Com o diagrama de classes terminado, iniciou-se a criação do web service, posteriormente o desenvolvimento da aplicação para o dispositivo móvel, chegando a fase de execuções e ajustes.

O relatório de mortalidade é obtido pelo registro diário das informações na ficha de controle de mortalidade preenchida pelo responsável. O projeto propõe que estas informações sejam registradas no *smartphone* uma vez ao dia (armazenando os dados primeiramente no próprio dispositivo, devido ao fato de alguns momentos não haver rede para se conectar) e transmitidas para o servidor por meio do web service, onde posteriormente estas informações irão se integrar a outros sistemas.

4.1 APLICATIVO CONSUMIDOR

O aplicativo consumidor do serviço possui oito telas, sendo estas: (1) a tela inicial, (2) tela de *login*, (3) tela principal, (4) tela de envio de dados, (5) tela de consulta e (6, 7, 8) para a parte de cadastro de mortalidade utilizam-se três telas. Na tela inicial o usuário poderá importar dados referentes a usuários existentes no servidor, caso não possua os mesmos em seu dispositivo, para na sequência poder acessar a tela de *login* como pode ser observada na Figura 15, sendo que para a importação dos dados é necessário a conexão do dispositivo móvel com a rede.

Figura 15 – Tela inicial da aplicação

Fonte: Autoria Própria

Os cadastros e alterações de usuários, aviários e lotes são cadastrados por outro sistema já existente no setor responsável e ficam armazenados no servidor, para atualizar os dispositivos móveis, quando a ação for solicitada pelo cliente.

Na tela de *login* o usuário terá que informar o usuário e a senha para acessar a tela principal onde terá acesso para efetuar as operações fornecidas pelo aplicativo se os dados estiverem corretos. Caso os dados estejam incompletos ou errados o sistema irá informá-lo e permanecerá na tela de *login* como pode ser notado na Figura 16.

Figura 16 – Tela de login

Fonte: Autoria Própria

Na tela principal o usuário poderá se direcionar ao cadastro de mortalidade, ao envio de dados para o servidor, a consulta de mortalidade e importar dados do servidor tais como aviários onde o responsável é o usuário logado no sistema, como pode ser visto na Figura 17, lembrando novamente que para a importação dos dados o dispositivo deve estar conectado a rede.

Figura 17 – Tela principal

Fonte: Autoria Própria

O cadastro de mortalidade se inicia em uma tela onde os aviários em que o usuário logado é responsável, são listados em um *ListBox*, onde o usuário irá selecionar o aviário que deseja registrar a mortalidade como pode ser observado na Figura 18, isso após o usuário clicar no botão registrar mortalidade na tela principal.

Figura 18 – Tela de seleção de aviário para registro de mortalidade
Fonte: Autoria Própria

Após selecionar o aviário, o usuário é direcionado a tela onde irá informar o lote, data, quantidade de aves mortas de acordo com o grupo em que se enquadra (Morte, eliminada ou ataque), temperatura e umidade, sendo necessário clicar no botão “Prosseguir” para a continuação do registro, como pode ser visto na Figura 19.

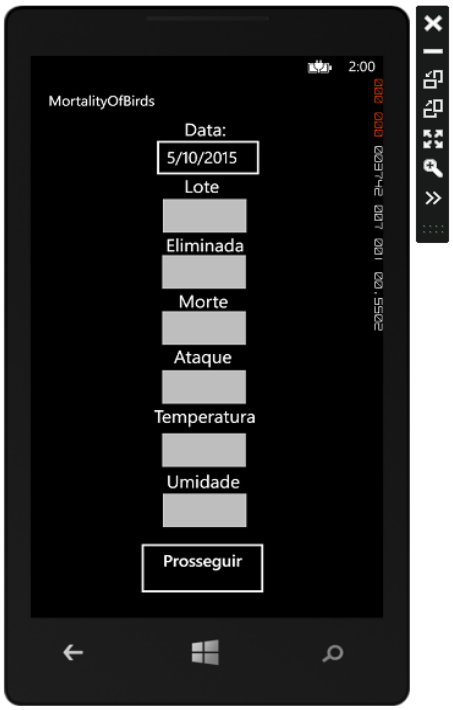


Figura 19 - Tela de informações de mortalidade

Fonte: Autoria Própria

Depois de ter inserido todas as informações e clicado no botão Prosseguir o sistema direcionará o usuário à tela onde poderá informar possíveis observações, caso seja necessário, como pode ser visto na Figura 20, finalizando o registro de mortalidade clicando no botão “Salvar” e, na sequência o sistema exibirá uma mensagem de que os dados foram armazenados com sucesso. A partir desse momento o usuário possui os dados do registro de mortalidade armazenados em seu dispositivo, podendo realizar alterações, excluir ou até mesmo envia-los.

Figura 20 – Tela de observações de mortalidade

Fonte: Autoria Própria

Clicando no botão “Consultar Mortalidade” da tela principal, o usuário é direcionado para tela onde são listados todos os registros de mortalidade realizados pelo mesmo e que estão armazenados em seu dispositivo, como pode ser visto na Figura 21. Sendo que ao selecionar o registro poderá efetuar as operações CRUD (*Create*, *Read*, *Update e Delete*) sobre o mesmo, direcionando o usuário para a tela onde poderá alterar as informações do registro de mortalidade, como também as informações ou até mesmo excluir o registro.

Figura 21 – Tela consulta registro de mortalidade
Fonte: Autoria Própria

Ao clicar no botão “Enviar dados para o servidor” na tela principal, o usuário é direcionado para tela de envio, onde deverá informar a data dos registros que deseja enviar, mas para isso o dispositivo deve estar conectado a rede, e na sequência clicar no botão “Enviar”. Assim o sistema irá verificar se existe registros com a data informada para serem enviados e, caso não possua, o sistema informará que não existem dados a serem enviados, caso contrário efetuará o processo exibindo uma mensagem de êxito ao concluir o mesmo.

A Figura 22 apresenta o código da aplicação invocando o método PUT no web service, enviando como parâmetros um objeto do tipo Mortalidade e o ID. Este método verifica se o objeto já existe, caso exista ele é alterado, senão é inserido.

```
private async static void Alterar(int ID, Mortalidade mor)
{
    string Endereco = "http://192.168.1.103/Mortalidade/api/Mortalidade/"+ID;
    using (var client = new HttpClient())
    {
        await client.PutAsync(Endereco, new ObjectContent<Mortalidade>(
            mor, new System.Net.Http.Formatting.JsonMediaTypeFormatter())).ContinueWith(t =>
        {

        });
    }

}
```

Figura 22– Código responsável por invocar o método PUT

Fonte: Autoria Própria

4.2 APLICATIVO SERVIDOR

O aplicativo servidor é composto pelos controladores API web, que efetuam as operações CRUD sobre as classes modelos representadas no diagrama de classes apresentado no documento de requisitos. Cada classe modelo possui o seu controlador com seus respectivos métodos (GET, POST, PUT e DELETE). O acesso aos dados é realizado por meio do ADO .NET *Entity Framework*, sendo que estas operações são realizadas pelas requisições de clientes enviadas as URIs (*Uniform Resource Identifier*) que representam os recursos oferecidos pela aplicação.

A Figura 23 apresenta dois métodos *GetAviario()*, onde um necessita de parâmetro e outro não. O que não necessita de parâmetro tem como retorno todos os objetos Aviario armazenados no banco dados, já o outro retornará o Aviario que possuir o valor passado como *id*, caso exista.

```
// GET api/Aviario
public IQueryable<Aviario> GetAviario()
{
    return db.Aviario;
}

// GET api/Aviario/5
[ResponseType(typeof(Aviario))]
public IHttpActionResult GetAviario(int id)
{
    Aviario aviario = db.Aviario.Find(id);
    if (aviario == null)
    {
        return NotFound();
    }

    return Ok(aviario);
}
```

Figura 23 – Métodos gets do controlador da classe Aviario

Fonte: Autoria Própria

Na Figura 24 observa-se o resultado recebido em XML pela invocação do método GET, realizado por meio de um navegador web no endereço onde retorna todos os aviários armazenados no banco, que neste caso foi http://localhost/Mortalidade/api/Aviario, isso com o recurso já publicado Servidor IIS. A aplicação foi publicada pelo próprio Visual Studio 2013, sendo isso realizado por meio de um clique com botão direito do mouse na *Solution Explorer* sobre o projeto, em seguida escolhendo a opção *Publish* com o botão esquerdo do mouse, como pode ser observado na Figura 25, posteriormente foram inseridas informações referentes a publicação.

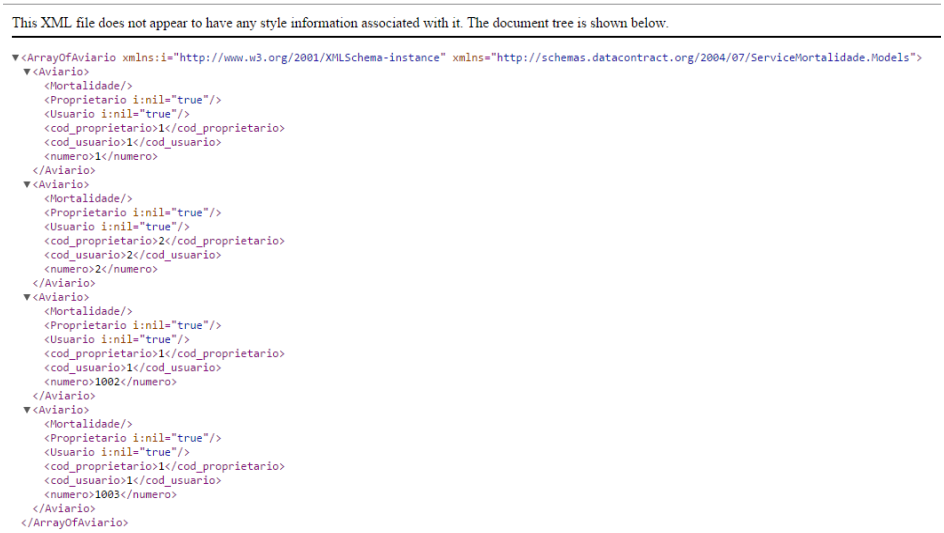


Figura 24 – Resposta do método get da classe Aviario

Fonte: Autoria Própria

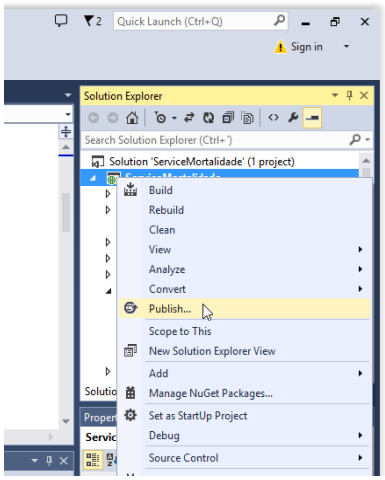


Figura 25 – Publicando a aplicação

Fonte: Autoria Própria

4.3 INTEGRAÇÃO DOS DADOS

No setor do controle de qualidade onde são acompanhados os monitoramentos dos lotes de aves nos aviários, os dados de mortalidade enviados pelos produtores de seu dispositivo móvel por meio do web service, poderão ser visualizados e analisados por outros sistemas, isto através da tela de monitoramento, relatórios ou gráficos. A Figura 26 apresenta a tela de monitoramento dos registros de mortalidade nos aviários, na Figura 27 podemos observar um relatório dos registros de mortalidade, já a Figura 28 apresenta um gráfico indicando a quantidade de morte por grupo nos aviários.

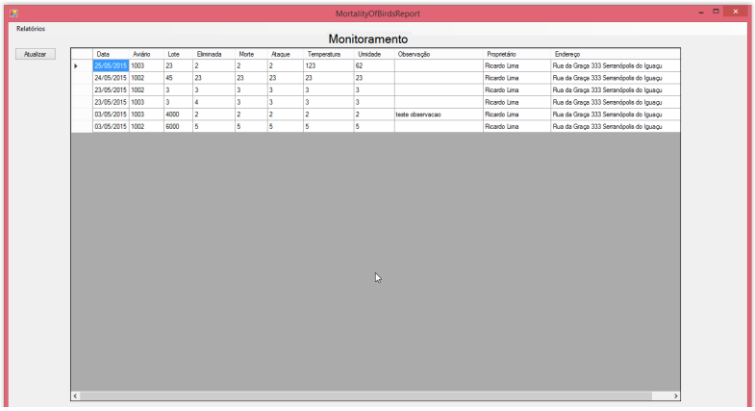
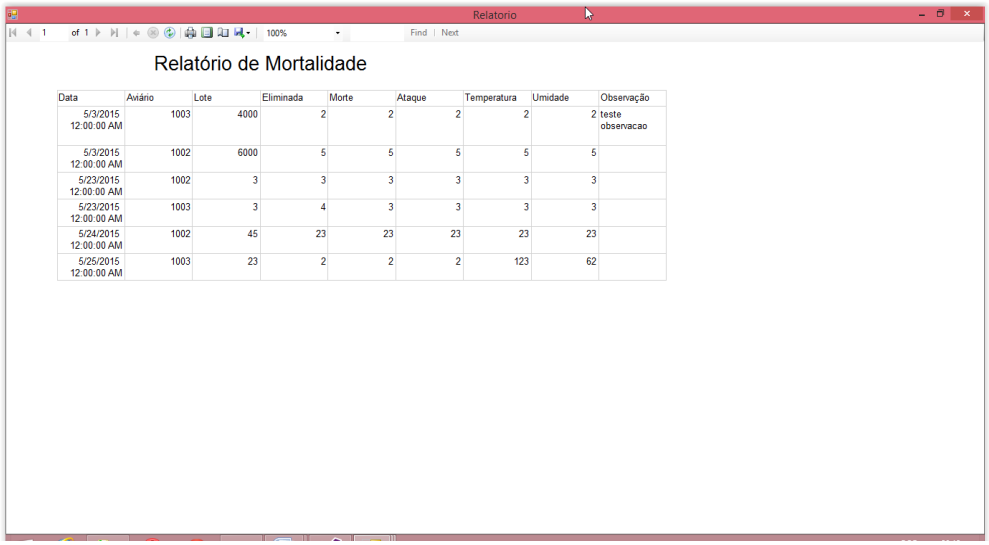


Figura 26 – Tela de monitoramento dos registros de mortalidade

Fonte: Autoria Própria



Data	Aviário	Lote	Eliminada	Morte	Ataque	Temperatura	Umidade	Observação
5/3/2015 12:00:00 AM	1003	4000	2	2	2	2	2	teste observacao
5/3/2015 12:00:00 AM	1002	6000	5	5	5	5	5	
5/23/2015 12:00:00 AM	1002	3	3	3	3	3	3	
5/23/2015 12:00:00 AM	1003	3	4	3	3	3	3	
5/24/2015 12:00:00 AM	1002	45	23	23	23	23	23	
5/25/2015 12:00:00 AM	1003	23	2	2	2	123	62	

Figura 27 – Relatório dos registros de mortalidade

Fonte: Autoria Própria

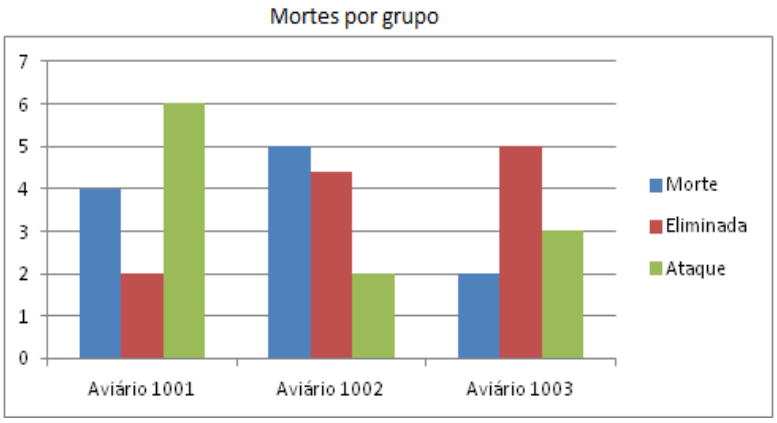


Figura 28 – Gráfico de mortes por grupo nos aviários

Fonte: Autoria Própria

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÃO

Os dispositivos móveis estão cada vez mais presentes no cotidiano da população. Sendo de fácil utilização, fornece recursos para favorecer o próprio consumidor ou corporações, por meio de aplicações desenvolvidas para atender suas necessidades, podendo ser uma aplicação local ou distribuída.

As ferramentas utilizadas para o desenvolvimento da aplicação são completas e robustas, Sendo que o Visual Studio possui uma interface de fácil utilização e também muitos recursos já inclusos, além de permitir *downlod* de pacotes, fornece também um emulador do sistema operacional *Windows Phone 8* de boa qualidade facilitando a fase de executar a aplicação.

Com relação à aplicação desenvolvida, gerou agilidade no processo eliminando o tempo de digitar as fichas manuscritas, fornecendo um melhor acompanhamento do ciclo de lote de aves em aviários, podendo tomar não só medidas preventivas, como também corretivas.

5.2 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Como trabalhos futuros deste projeto, sugere-se a inclusão de novos recursos tanto para a aplicação móvel como também para o web service, tais como o controle de ração e vacinas que também são monitorados pela ficha de mortalidade para aviários.

Quando possuir uma grande quantidade de dados armazenados, realizar uma mineração nos mesmos para auxiliar nas tomadas de decisões, quanto as ações necessárias para o bom desenvolvimento e crescimento dos frangos para abate.

REFERÊNCIAS BIBLIOGRÁFICAS

AECE, Israel. **Características do ASP.NET Web API.** 2012. Disponível em: <http://www.israelaece.com/post/Caracteristicas-do-ASPNET-Web-API.aspx>. Acesso em: 11 fev. 2015.

ARAUJO, Rodrigo. **ADO.NET Entity Framework 4 e ASP.NET - Artigo WebMobile Magazine 32.** Disponível em: <http://www.devmedia.com.br/ado-net-entity-framework-4-e-asp-net-artigo-webmobile-magazine-32/18162>. Acesso em: 21 fev. 2015.

AECE, Israel. **WCF Web API - Media Types.** 2011. Disponível em: <http://www.israelaece.com/post/WCF-Web-API-Media-Types.aspx>. Acesso em: 26 mar. 2015.

BABOO. **NOVIDADES DO VISUAL STUDIO.NET.** Disponível em: <http://www.baboo.com.br/tutorial/novidades-do-visual-studio-net/>. Acesso em: 26 mar. 2015.

BUENO, Miriam Pineiro; ARAÚJO, Geraldino Carneiro de; SPROESSER, Renato Luiz, et al. Apropriação de Valor Bruto nas Transações Econômicas da Cadeia Produtiva da Avicultura de Corte. In.: **Anais...** ENANPAD 2007 – XXXI encontro da ANPAD, Rio de Janeiro/RJ. 22 a 26 set. 2007.

CANALTECH. **Windows Phone já é o segundo sistema operacional móvel mais usado no Brasil.** 2014. Disponível em: <http://canaltech.com.br/noticia/windows-phone/Windows-Phone-ja-e-o-segundo-sistema-operacional-movel-mais-usado-no-Brasil/>. Acesso em: 29 mar. 2015

CASETTI, Orestes; AKAMATU, Durval Makoto; KIRNER, Cláudio. **PARADIGMAS PARA CONTRUÇÃO DE SISTEMAS DISTRIBUÍDOS.** 1993. Disponível em: <http://www1.serpro.gov.br/publicacoes/tematec/pubtem13.htm>. Acesso em: 22 ago. 2014.

CAVALCANTI, Carlos. **INTRODUÇÃO SOBRE SQLITE.** 2011. Disponível em: <http://carloscavalcanti.com/2011/05/introducao-sobre-sqlite/>. Acesso em: 08 mai. 2015.

DANRESA. **O ADO.NET Entity Framework no .NET 4.** Disponível em: <http://www.danresa.com.br/fabrica-de-software/index.php/o-ado-net-entity-framework-no-net-4/>. Acesso em: 27 mar. 2015.

DANTAS, Cleber. **Série aprenda C# - Introdução a plataforma .NET e ao C#.** Disponível em: <http://www.linhadecodigo.com.br/artigo/984/serie-aprenda-csharp-introducao-a-plataforma-net-e-ao-csharp.aspx>. Acesso em: 21 fev. 2015.

GOMES, Daniel Adorno. **Web Services SOAP em Java:** Guia prático para o desenvolvimento de web services em Java. São Paulo: Novatec, 2010.

GROFFE, Renato Jose. **Uma visão geral sobre o Visual Studio 2013 Leia mais em: Uma visão geral sobre o Visual Studio 2013.** 2013. Disponível em: <http://www.devmedia.com.br/uma-visao-geral-sobre-o-visual-studio-2013/28651>. Acesso em: 09 fev. 2015.

HERBERT, Liz. **Desenvolvimento de aplicativo móvel:** como escolher seu parceiro. Disponível em: <http://www.alexandreporfirio.com/2012/07/11/android/desenvolvimento-de-aplicativo-movel-como-escolher-seu-parceiro/>. Acesso em: 29 mar. 2015.

KALIN, Martin. **Java Web Services:** Implementando. Rio de Janeiro: Alta Books, 2010.

LIMA, Jean Carlos R. **Web services (SOAP x REST)**. 2012. 41 f. Trabalho de Conclusão de Curso (Graduação) – Tecnologia em Processamento de Dados, Faculdade de Tecnologia de São Paulo, São Paulo, 2012. Disponível em: <http:// http://www.fatecsp.br/dti/tcc/tcc00056.pdf>. Acesso em: 16 jun. 2015.

MANIADE CELULAR. **Vantagens e desvantagens que Smartphones podem oferecer.** 2012. Disponível em: <http://www.maniadecelular.com.br/233131/vantagens-e-desvantagens-que-smartphones-podem-oferecer.html>. Acesso em: 20 maio 2015.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia científica.** 5. ed. São Paulo: Atlas, 2003.

MICROSOFT. **Introdução à linguagem C# e ao .NET Framework.** 2013. Disponível em: <https://msdn.microsoft.com/pt-br/library/z1zx9t92.aspx>. Acesso em: 10 fev. 2015.

MICROSOFT. **SQL Server.** Disponível em: <http://www.microsoft.com/pt-br/server-cloud/products/sql-server/try.aspx>. Acesso em: 06 abr. 2015.

MICROSOFT. **Visual Studio Premium with MSDN.** Disponível em: <https://www.visualstudio.com/products/visual-studio-premium-with-msdn-vs#Fragment_SystemRequirements>. Acesso em: 30 mar. 2015.

MICROSOFT (E.u.a.). **Biblioteca de classes .NET Framework.** Disponível em: <https://msdn.microsoft.com/pt-br/library/ms229335(v=vs.90).aspx>. Acesso em: 26 mar. 2015.

NAVEGANDO E APRENDENDO. **Ciclo de vida de um frango de corte.** 2011. Disponível em: <http://desafioxeduardoevitor.blogspot.com.br/2011/09/antes-de-todo-o-processo-comecar-todos.html>. Acesso em: 15 jun. 2015.

OFICINA DA NET. **Internet Information Services.** 2010. Disponível em: <http://www.oficinadanet.com.br/artigo/servidores/internet_information_services>. Acesso em: 19 jun. 2015.

PALIARI, Marcos. Como funciona um WebService REST. 2012. Disponível em: <http://www.matera.com/br/2012/10/22/como-funciona-um-webservice-rest/>. Acesso em: 17 jun. 2015.

PORTAL SUÍNOS E AVES. Etapas de manejo de frango de corte. , 2014. Disponível em: <http://www.portalsuinoseaves.com.br/etapas-do-manejo-de-frango-de-corte/>. Acesso em: 29 ago. 2014.

PIRES, Eduardo. **ASP.Net Web API – Meu primeiro serviço REST**. 2013. Disponível em: <http://eduardopires.net.br/2013/07/asp-net-web-api-meu-primeiro-servico-rest/>. Acesso em: 19 fev. 2015.

RECIO, Francisco; PROVENCIO, David. **Net Framework.** 2008. Disponível em: <http://www.criarweb.com/artigos/net-framework.html>. Acesso em: 25 mar. 2015.

RECKZIEGEL, Mauricio. Entendendo os webservices. 2006. Disponível em: <http://imasters.com.br/artigo/4245/webservices/entendendo_os_webservices/>. Acesso em: 16 jun. 2015.

SANTANNA, Mauro. **Plataforma .NET para desenvolvedores.** 2000. Disponível em: <http://www.portaldaprogramacao.com/artigos2.asp?n=69>. Acesso em: 21 fev. 2015.

SILVA, Rafael. **A importância de um Framework.** 2004. Disponível em: <http://rafaelsilva.net/framework/importancia-de-um-framework>. Acesso em: 17 jun. 2015.

SOLUTIS. **MAIORIA DAS EMPRESAS PRETENDE INVESTIR EM NOVAS APLICAÇÕES MÓVEIS.** Disponível em: <http://solutis.com.br/?p=1446>. Acesso em: 29 mar. 2015.

STACKOVERFLOW. **Throw or not throw the exception from the methods consumed by the ASP.NET Web API layer.** 2013. Disponível em: <http://stackoverflow.com/questions/14199837/throw-or-not-throw-the-exception-from-the-methods-consumed-by-the-asp-net-web-ap>. Acesso em: 26 mar. 2015.

TECHTUDO. **Windows Phone:** Windows Phone é a plataforma sucessora do Windows Mobile. Disponível em: <http://www.techtudo.com.br/tudo-sobre/windows-phone.html>. Acesso em: 21 fev. 2015.

TUDOCELULAR.COM. **Microsoft diz que o Windows Phone é a plataforma mais segura do mundo.** 2014. Disponível em: <http://www.tudocelular.com/windows/noticias/n41502/microsoft-windows-phone-seguro.html>. Acesso em: 29 mar. 2015.

UFRJ. **SOAP:** Simple Object Access Protocol. 2015. Disponível em: <http://www.gta.ufrj.br/grad/05_1/webservices/soap.htm>. Acesso em: 16 jun. 2015.

WEB, Treina. **Curso Online de SQL Server - Desenvolvedor.** Disponível em: <https://www.treinaweb.com.br/curso/sql-server-desenvolvedor>. Acesso em: 21 fev. 2015.