INSTITUTO FEDERAL DO ESPÍRITO SANTO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

OBERDAN DEBONA ALTOÉ

SISTEMA DE CONTROLE DA RAIVA E CAPTURA DE MORCEGOS

Cachoeiro de Itapemirim

2017

OBERDAN DEBONA ALTOÉ

SISTEMA DE CONTROLE DA RAIVA E CAPTURA DE MORCEGOS

Trabalho de Conclusão de Curso apresentado à Coordenadoria do Curso de Sistemas de Informação do Instituto Federal do Espírito Santo, Campus Cachoeiro de Itapemirim, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dsc. Rafael Vargas Mesquita do Santos

Cachoeiro de Itapemirim

2017

OBERDAN DEBONA ALTOÉ

SISTEMA DE CONTROLE DA RAIVA E CAPTURA DE MORCEGOS

Trabalho de Conclusão de Curso apresentado à Coordenadoria do Curso de Sistemas de Informação do Instituto Federal do Espírito Santo, Campus Cachoeiro de Itapemirim, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Aprovado em 20 de novembro de 2017.

COMISSÃO EXAMINADORA

Prof. Dsc. Rafael Vargas Mesquita do Santos
Instituto Federal do Espírito Santo
Orientador

Prof. Msc. João Paulo de Brito Gonçalves
Instituto Federal do Espírito Santo

Prof. Dsc. Eros Estevão de Moura
Instituto Federal do Espírito Santo

DECLARAÇÃO DO AUTOR

Declaro, para fins de pesquisa acadêmica, didática e técnico-científica, que este Trabalho de Conclusão de Curso pode ser parcialmente utilizado, desde que se faça referência à fonte e ao autor.

Cachoeiro de Itapemirim, 05 de Novembro de 2017.

OBERDAN DEBONA ALTOÉ

Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida, autor do meu destino, meu guia, socorro presente nas horas de angústia, ao meu pai Samuel, minha mãe Lidinalva e aos meus irmãos e namorada.

AGRADECIMENTOS

Aos meus pais, Samuel e Lidinalva, que sempre me apoiaram e incentivaram em todos os momentos de minha vida. Obrigado por tudo.

Aos meus tios, Gelson e Lucinda, que sempre estiveram à disposição para me ajudar durante está caminhada de minha vida. Obrigado por tudo.

Ao meu Orientador Prof. Rafael Vargas Mesquita pela oportunidade e apoio na elaboração deste trabalho, pelo seu grande desprendimento em ajudar e amizade sincera.

Ao João Victor Vargas Mesquita do Santos, com sua contribuição ajudou de forma vigorosa para que esse trabalho ficasse realista a sua proposta de solução.

Da mesma forma agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

Aos meus amigos que conquistei no Ifes, André, Eduardo, Erick, João Marcos, Ludivan e Matheus, pelo incentivo e ajuda para a realização deste trabalho.

A minha namorada pelo incentivo e paciência nas horas difíceis durante a realização deste trabalho.

A todos vocês, meus sinceros agradecimentos.

RESUMO

A Raiva é uma antropozoonose transmitida ao homem e animais pela inoculação do vírus rábico, contido na saliva do animal infectado; seu índice de letalidade é de 99,9%. O vírus da Raiva pode infectar todos os mamíferos até agora testados, com maior incidência em cães, morcegos e gatos, mas também com grande incidência em bovinos e equinos. Estima-se que a raiva bovina na América Latina cause prejuízos anuais de centenas de milhões de dólares, provocados pela morte de milhares de cabeças, além dos gastos indiretos que podem ocorrer com a vacinação de milhões de bovinos e inúmeros tratamentos pós-exposição de pessoas que mantiveram contato com animais suspeitos. Dada à importância da raiva bovina no país, é imprescindível o monitoramento frequente do animal, protegendo-os contra o ataque de morcegos hematófagos no rebanho. Constatado o ataque, o proprietário deve notificar o órgão competente para que sejam tomadas as devidas providências para o controle. Neste contexto, identificou-se a possibilidade do desenvolvimento de um sistema para o monitoramento das informações geradas a partir do planejamento e combate à doença, especificamente realizados pelo Instituto de Defesa Agropecuária e Florestal do Espírito Santo (IDAF). O objetivo geral deste trabalho foi a elaboração de um aplicativo para smartphones voltado ao armazenamento de informações sobre a captura de morcegos em propriedades de criação de gado. Este aplicativo pode ser utilizado por funcionários do IDAF, bem como, por quaisquer proprietários rurais interessados. Especificamente, o aplicativo traz a possibilidade de orientação aos proprietários rurais quanto à importância da vacinação do rebanho. Também permite que os proprietários rurais realizem solicitações de visitas em suas propriedades aos agentes do IDAF. As informações obtidas nestas visitas serão armazenadas pelos funcionários do IDAF em uma base de dados, incluindo as localizações geográficas das capturas realizadas. Após implementação o sistema foi avaliado, considerando as sete principais funcionalidades, e aprovado por um funcionário de perfil técnico do IDAF.

Palavras-chave: Raiva, animais, combate.

ABSTRACT

Rabies is an anthropozoonosis transmitted to man and animals by the inoculation of rabies virus, contained in the saliva of the infected animal; Its lethality rate is 99.9%. Rabies virus can infect all mammals tested so far, with a higher incidence in dogs, bats and cats, but also with a high incidence in cattle and horses. It is estimated that bovine rabies in Latin America causes annual losses of hundreds of millions of dollars, caused by the death of thousands of heads, besides the indirect costs that can occur with the vaccination of millions of cattle and numerous post-exposure treatments of people Who had contact with suspect animals. Given the importance of bovine rabies in the country, frequent monitoring of the animal is essential, protecting them against the attack of blood-sucking bats in the herd. Once the attack has been established, the owner must notify the appropriate authority so that the necessary measures for control are taken. In this context, it was identified the possibility of developing a system to monitor the information generated from the planning and fight against the disease, specifically carried out by the Institute of Agricultural and Forest Defense of Espírito Santo (IDAF). The general objective of this work is the development of an application for smartphones aimed at storing information about the capture of bats in livestock farms. This application may be used by IDAF employees as well as by any interested landlords. Specifically, the application provides an opportunity for guidance to rural owners as to the value of herd vaccination. It also allowed rural landlords to make requests for visits to IDAF properties. The information obtained from these visits is stored by IDAF officials in a database, including the geographical locations of the catches taken. After the implementation of the system, evaluated, considering as seven main functionalities, approved by an official of IDAF technical profile.

Keywords: rabies, animals, combat.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

WADL	Web Application Description Language
WSDL	Web Services Description Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
CSS	Cascading Style Sheets
GPU	Graphics Processing Unit
IDAF	Institudo de Defesa Agropecuária e Florestal
GPS	Global Positioning System
RUP	Rational Unified Process
CRUD	Create, Read, Update, and Delete
APK	Android Package
SNC	Sistema Nervoso Central
ICMBIO	Instituto Chico Mendes de Conservação da Biodiversidade
API	Application Programming Interface
HAL	Hardware Annotation Library
JVM	Java Virtual Machine
VM	Virtual Machine
SDK	Software Developers Kit
GC	Garbage Collecto
SO	Sistema Operacional
SMS	Short Message Service

ARM	Advanced Risc Machine
JDK	Java Development Kit
MVC	Model-View-Controller
RIA	Rich Internet Application

SUMÁRIO

1 INTRODUÇÃO

Segundo Murphy et al. (1999) o vírus rábico ocorre em todo o Mundo, com algumas exceções, como o Japão, Reino Unido, Nova Zelândia, Antártida, e outras pequenas ilhas como o Havaí, onde foi completamente erradicado.

O vírus da raiva é um agente causador de encefalomielite em todas as espécies mamíferas e com raras exceções, uma vez sintomáticas, a evolução nos animais e nos seres humanos é fatal.

Conforme Corrêa e Passos (2001) o Brasil possui majestosos ecossistemas com gigantesca biodiversidade florística e faunística. Lamentavelmente, a destruição desses ecossistemas ainda continua em larga escala, cuja ação degradante do homem à natureza, pode advir em consequências desastrosas à vida na Terra. O avanço da agricultura e da pecuária próximo às áreas naturais proporcionou um contato entre as populações humanas e de seus animais domésticos com as populações de animais silvestres nos seus habitats. Este estreito contato facilitou a disseminação de agentes infecciosos e parasitários para novos hospedeiros e ambientes, estabelecendo-se assim novas relações entre hospedeiros e parasitas, e novos nichos ecológicos na cadeia de transmissão das doenças .

Como dito anteriormente o estreitamento entre animais silvestres e animais domésticos facilitou muito a propagação de doenças e vírus. Com o vírus da raiva não foi diferente, se alastrou por todo o pais, matando muitas pessoas,animais silvestres e principalmente de rebanhos, pelo fato de estarem agrupados propiciando maior transmissão da doença.

O combate a raiva existe há muitos anos no Brasil que vem sempre se empenhando cada vez mais para tentar diminuir e erradicar casos de morte por Raiva, com campanhas de vacinação e combate direto aos seus principais transmissores que são os Morcegos Hematófagos, que se alimentam de sangue.

Na zona rural onde se concentram a maioria dos casos da Raiva principalmente animais de rebanhos, muitos casos da morte de animais ficam desconhecidos. Uma instituição

que ajuda no combate é o Instituto de Defesa Agropecuária e Florestal (IDAF), é a entidade responsável pela execução da política agrária do Estado no que se refere às terras públicas, pela execução da política cartográfica e pela execução da política de defesa sanitária das atividades agropecuárias, florestais, pesqueiras, dos recursos hídricos e solos bem como pela administração dos remanescentes florestais da mata atlântica, demais formas de vegetação existentes e da fauna no território nacional, que se empenha em fazer campanhas de vacinação e captura de morcegos para controle da doença.

Diante do contexto apresentado, este trabalho propõe a elaboração de um aplicativo facilitador quanto a comunicação entre agentes do IDAF e os cidadãos, principalmente criadores de gado. Por intermédio do aplicativo, qualquer cidadão comum poderá solicitar visitas dos agentes do IDAF. Várias informações sobre visitas realizadas, inclusive com identificação de localização geográfica e etc, poderão ser disponibilizadas à população. Estes dados, além de levar informação aos cidadãos, também ajudarão os agentes do IDAF no compartilhamento de registros das visitas.

2 JUSTIFICATIVA

Segundo dados da OMS (2010) a raiva mata 55 mil pessoas por ano, especialmente em países da África e Ásia onde não existe vacinação em massa dos animais. A cada 10 minutos uma pessoa morre por esta doença no mundo, principalmente crianças, num total de 30 a 50% dos casos.

Na Agricultura não é diferente, muitos produtores sofrem com essa doença principalmente em criações de rebanhos como bovinos e equinos, causando grandes prejuízos.

Neste contexto, identificou-se a necessidade de um sistema para o monitoramento das informações geradas a partir do planejamento e combate à doença.

A justificativa para tal sistematização está na possibilidade de identificar padrões de deslocamento da doença, visualizando locais nos quais está causando maiores impactos.

Dentre as várias informações a serem fornecidas pelo sistema, podemos citar, locais de captura de morcegos, quantidade de mortes de animais por raiva comprovadas por exame, localidades com realização de coleta de material, listagem de cidades e seus respectivos quantitativos de visitas e solicitações de visita para captura de morcego e etc. A possibilidade de estudo destes dados irá facilitar a identificação dos locais em que o vírus e a doença da raiva está se manifestando, podendo assim influenciar em um planejamento de combate mais eficiente.

3 OBJETIVOS

Este Projeto tem como objetivo desenvolver uma aplicativo para controle e mapeamento da Raiva em propriedades rurais com a criação de gado.

3.1 OBJETIVO GERAL

O objetivo geral deste trabalho é a elaboração de um sistema multiplataforma voltado ao armazenamento de informações sobre a captura de morcegos em propriedades de criação de herbívoros. Este aplicativo poderá ser utilizado por funcionários do Instituto de Defesa Agropecuária e Florestal do Espírito Santo (IDAF), bem como, por quaisquer proprietários rurais interessados.

3.2 OBJETIVOS ESPECÍFICOS

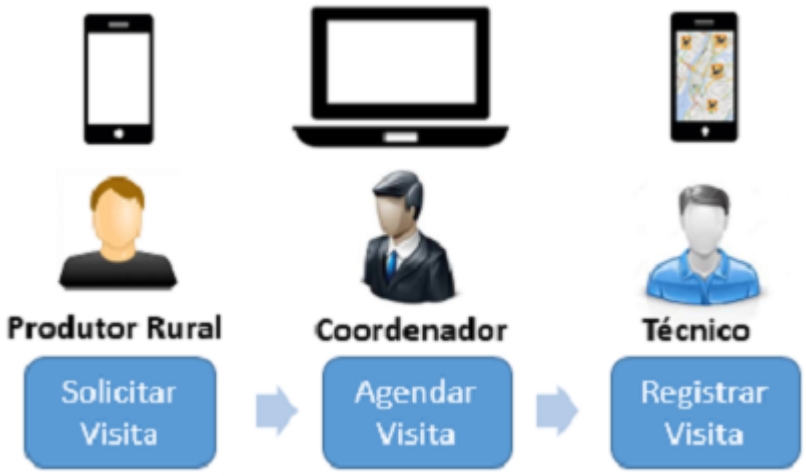
Especificamente, o aplicativo trará a possibilidade de orientação aos proprietários rurais quanto à importância da vacinação do rebanho. Também permitirá que os proprietários rurais realizem solicitações de visitas em suas propriedades aos agentes do IDAF. As informações obtidas nestas visitas serão armazenadas pelos funcionários do IDAF em uma base de dados, incluindo as localizações geográficas das capturas realizadas. Dentre os objetivos específicos deste trabalho, podem ser citados e detalhados:

- Permitir aos produtores rurais cadastrados solicitarem visitas aos funcionários do IDAF para captura de morcegos ou coleta de material (Figura 1);

- Automatizar o processo de registro de informações sobre visitas realizadas pelos agentes do IDAF, permitindo a identificação geográfica dos focos de doença via o serviço de pesquisa e visualização de mapas, Google Maps;

- Possibilitar relatórios informativos quanto às visitas e capturas visando um melhor planejamento de controle da doença;

Figura 1 – Principais funcionalidades do aplicativo por perfil de usuário



Fonte: Própria

De acordo com a Figura 1, o usuário com o perfil de Produtor Rural poderá solicitar visitas de acompanhamento em suas propriedades. O usuário com o perfil de Coordenador terá acesso a todas as solicitações e poderá agendar visitas para as mesmas, alocando os técnicos responsáveis. O usuário de perfil Técnico deverá registrar no sistema as informações das visitas, por exemplo, as coordenadas de localização referentes às possíveis capturas de morcegos realizadas.

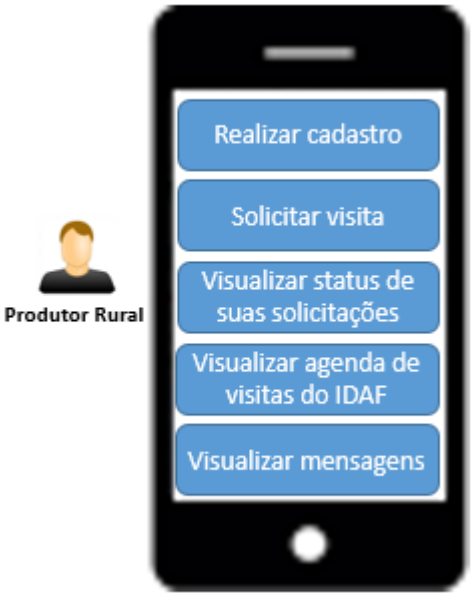
Como observado na Figura 1, os usuários com o perfil de Produtor Rural e Técnico terão acesso ao sistema por intermédio de aplicativo para smartphone (Android). Já o usuário com o perfil de Coordenador utilizará uma aplicação para computadores pessoais construída com a tecnologia JavaFX.

A seguir são apresentadas, de forma detalhada, as funcionalidades do sistema por perfil de usuário.

3.2.1 Funcionalidades de perfil Produtor rural

A Figura 2 mostra o detalhamento das funcionalidades para o perfil de usuário do Produtor Rural.

Figura 2 – Funcionalidades do perfil de usuário Produtor Rural



Fonte: Própria

- Realizar cadastro: por meio desta funcionalidade o usuário poderá realizar seu cadastro no sistema. Dentre as várias informações necessárias, o produtor rural deverá informar endereço de sua propriedade.

- Solicitar visita: permite a solicitação de visitas a sua propriedade.

- Visualizar status de suas solicitações: o produtor rural poderá visualizar o andamento de suas solicitações. Ou seja, se as solicitações estão pendentes de avaliação por parte do coordenador, se foram negadas ou aceitas.

- Visualizar agenda de visitas do IDAF: nesta opção o usuário terá acesso as informações sobre as visitas já agendadas pelo IDAF. Ele poderá fazer filtros de busca por cidades.

- Visualizar mensagens: está é uma funcionalidade que permite ao usuário receber avisos por parte do IDAF. Por exemplo, lembrete sobre período de vacinação do gado, etc.

3.2.2 Funcionalidades de perfil Coordenador

A Figura 3 mostra o detalhamento das funcionalidades para o perfil de usuário do Coordenador.

Figura 3 – Funcionalidades do perfil de usuário Coordenador

Fonte: Própria

- Solicitar visita: a funcionalidade de solicitação de visita por parte do Coordenador será utilizada para registrar no sistema solicitações de Produtores Rurais realizadas por telefone. Assim, todas as informações de solicitações e visitas serão armazenadas no sistema.

- Agendar visita: alocação dos Técnicos responsáveis para visitar a propriedade de um determinado produtor, bem como, a marcação da data da visita.

- Enviar mensagens: permite que o Coordenador envie quaisquer tipos de informativos aos Produtores Rurais.

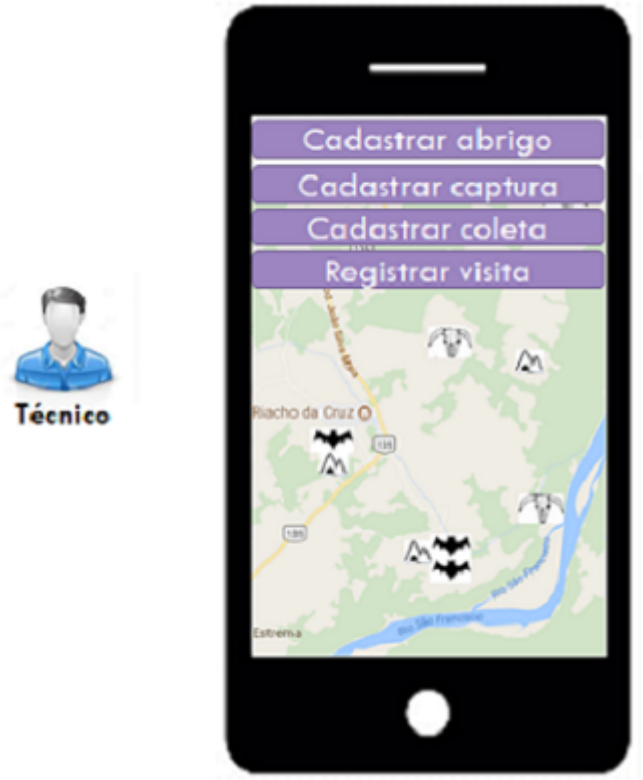
- Manutenção de cadastros: todos os cadastros serão de responsabilidade deste usuário.

- Relatórios: poderão ser gerados relatórios informativos mostrando as cidades com o maior número de casos da doença, a quantidade de visitas realizadas por municípios em um determinado intervalo de tempo, etc. Estes relatórios servirão de base para o planejamento do controle da doença por parte do órgão responsável.

3.2.3 Funcionalidades de perfil Técnico

A Figura 4 mostra o detalhamento das funcionalidades para o perfil de usuário do Técnico.

Figura 4 – Funcionalidades do perfil de usuário Técnico



Fonte: Própria

- Registrar visita: esta funcionalidade deverá ser realizada quando o Técnico possuir todos os cadastros necessários para finalizar a visita, como por exemplo, o abrigo que ele visitou ou as capturas realizadas. Nela, todas as informações relacionadas a visita serão armazenadas, por exemplo: localização GPS de capturas de morcegos, locais de identificação de abrigos de morcegos, bem como, mortes de animais.

- Cadastrar abrigo: trata-se do registro de um abrigo de morcegos contendo sua localização GPS. O cadastro de um abrigo, neste momento, não o vincula a uma visita em específico.

- Cadastrar captura: trata-se do registro de uma captura de morcegos contendo sua localização GPS. O cadastro de uma captura, neste momento, não a vincula a uma

visita em específico.

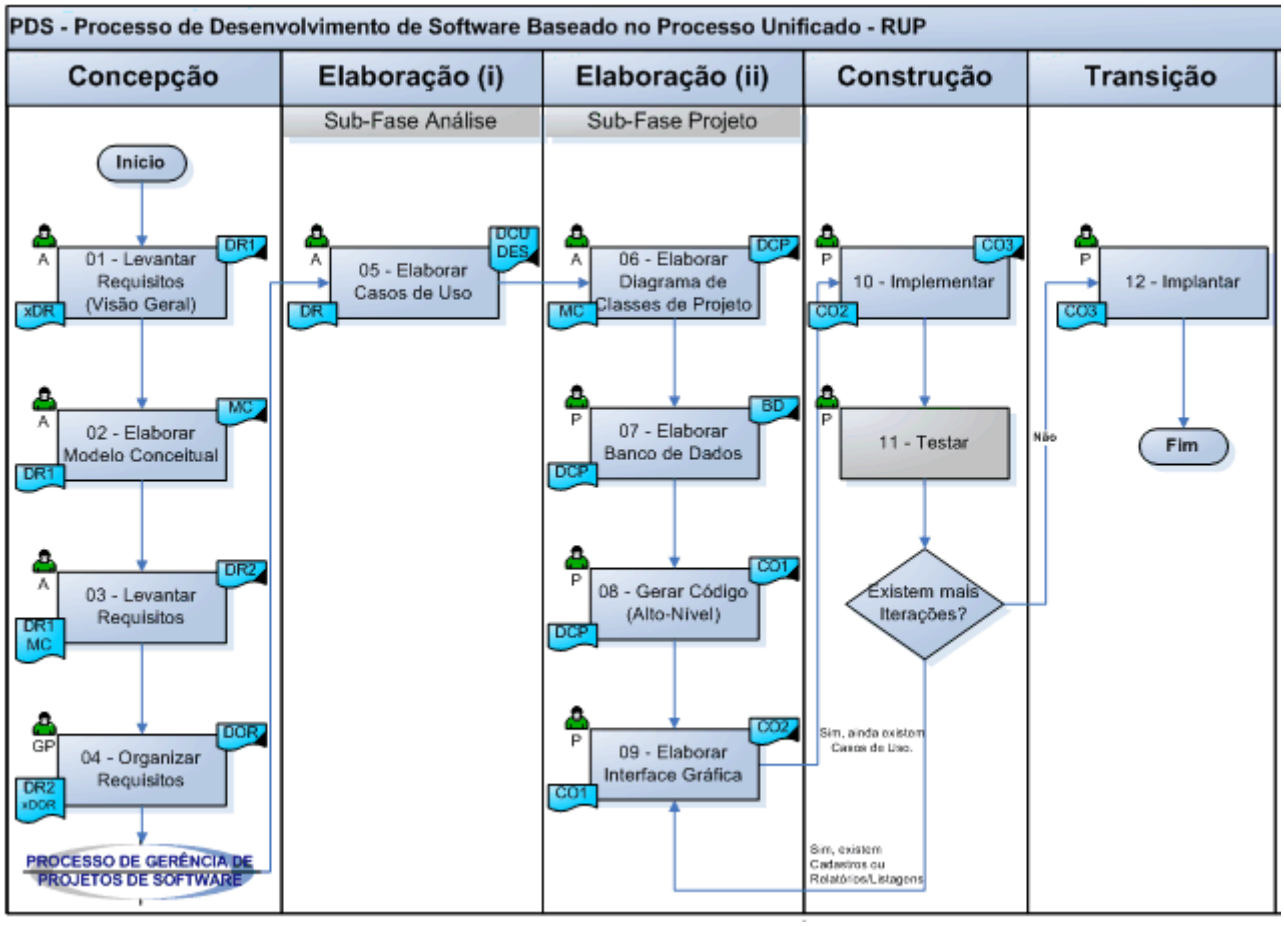
- Cadastrar coleta de material: trata-se de uma coleta de cérebro do herbívoro que morreu com suspeita de raiva, contendo sua localização GPS. O cadastro de uma coleta, neste momento, não a vincula a uma visita em específico.

4 METODOLOGIA DE PESQUISA

Para o desenvolvimento do sistema multiplataforma foi utilizado o seguinte processo baseado no Rational Unified Process (RUP).

- Processo de desenvolvimento de software (Figura 5) elaborado para a disciplina de Laboratório de Engenharia de Software do curso de Sistemas de Informação do Ifes Campus Cachoeiro (IFES-PDS-LES, 2017).

Figura 5 – Processo de desenvolvimento de software



Fonte: (IFES-PDS-LES, 2017)

Portanto, as principais atividades realizadas para elaboração do projeto foram:

1. Levantar requisitos (Visão Geral): detalhar uma visão inicial do domínio do problema relacionado ao sistema a ser desenvolvido.

2. Elaborar modelo conceitual: neste modelo foram identificadas as principais classes do sistema, bem como seus relacionamentos. Nesta etapa não será necessário a identificação dos atributos das classes.

3. Levantar requisitos: identificar as necessidades do cliente quanto ao sistema a ser desenvolvido, documentando os requisitos no documento de requisitos, o qual representa o contrato de comprometimento com o cliente.

4. Organizar requisitos: organizar todos os requisitos identificados e documentados na atividade anterior. Estes requisitos devem ser organizados em: processos de negócios, manutenção de cadastros (CRUD) e relatórios/listagens.

5. Elaborar casos de uso: elaborar o diagrama de casos de uso e a documentação de casos de uso.

6. Elaborar diagrama de classes de projeto: o modelo conceitual deverá ser atualizado, agora incluindo atributos, operações e navegabilidades.

7. Elaborar banco de dados: elaborar o script de criação do banco de dado com base no diagrama de classes elaborado anteriormente.

8. Gerar código (alto-nível): gerar código em linguagem Java automaticamente por meio da Ferramenta Astah.

9. Elaborar interface gráfica: criar os protótipos de todas as telas do aplicativo.

10. Implementar: codificar as classes necessárias para implementação das funcionalidades de processos de negócio, manutenção de cadastros e listagens.

11. Testar: realizar os testes de todas as funcionalidades desenvolvidas para o sistema.

12. Implantar: gerar o arquivo de extensão APK (Android Package), instalar o aplicativo e configurar o Web Service RESTful.

4.1 CONFIGURAÇÃO DO AMBIENTE

Os seguintes programas serão usados pra a o desenvolvimento:

- Banco de Dados (MySQL - versão 5.0.12);
- O IDE oficial do Android (Android studio - versão 2.3.1);
- Java SE Development Kit (JDK - versão 8u131);
- Android SDK (versão 24.4.1);
- Servidor Web (GlassFish Server - versão 4.0);

5 REVISÃO DE LITERATURA

A seção 5.1 trata-se sobre o Combate a raiva, reforçando a importância do controle da doença, levando em consideração os riscos de contaminação, bem como os prejuízos econômicos. Na seção 5.2 apresenta-se a Plataforma Android, destacando seus recursos e atributos favoráveis a sua utilização. A literatura sobre Web Service é considerada na seção 5.3, mostrando a possibilidade de disponibilização de serviços para diferentes clientes. Por fim, na seção 5.4, descreve-se a plataforma JavaFX utilizada para o desenvolvimento da interface gráfica da aplicação deste trabalho.

5.1 COMBATE A RAIVA

A raiva é uma doença infectocontagiosa do sistema nervoso que tem como agente etiológico o vírus Rabdovírus, que acomete predominantemente os mamíferos. Esta infecção viral é fatal em praticamente 100% dos casos (BRASIL, 2017).

Nos bovinos, esta enfermidade representa grandes prejuízos econômicos para o produtor, bem como um grande impacto na saúde pública. A fonte de infecção sempre é um animal infectado, sendo que o método de transmissão mais comum é a mordida de um animal portador do vírus conforme a figura 6, embora a contaminação de feridas cutâneas pela saliva recente possa levar à infecção. O principal agente transmissor desse vírus para os bovinos são os morcegos, em especial, o Desmodus rotundus, porém, outras espécies de morcegos hematófagos também podem transmitir o vírus, como o Diphylla ecaudata e Diaemus youngi.

Figura 6 – Morcego mordendo o pescoço de um bovino.

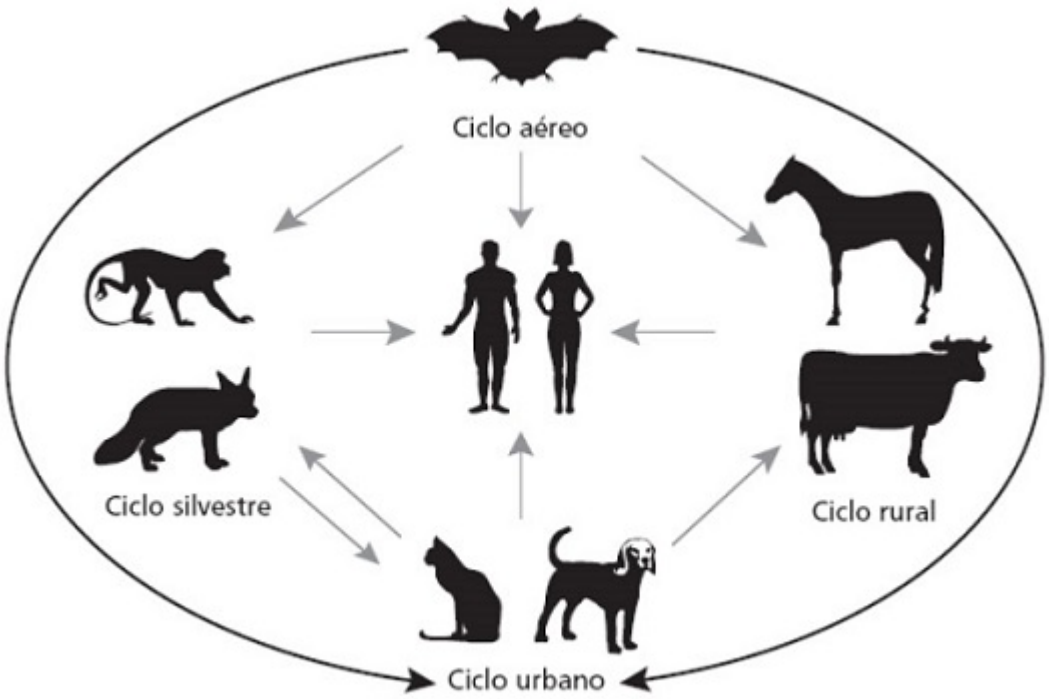
Fonte: (PERNAMBUCO, 2017)

O vírus Rabdovírus tem nos morcegos hematófagos, o melhor e o mais eficiente veículo de propagação, uma vez que estes agridem diariamente outros animais (suas presas para se alimentar, e/ou seus próprios companheiros, nas interações sociais agressivas). Conforme Bredt, Araujo e Junior (1996) descreve, essas agressões envolvem, principalmente, aplicação de mordeduras e outros tipos de comportamento interativo. Assim, um morcego hematófago infectado tem chances diárias e frequentes de transmissão, sendo, por isso, responsável ela infecção direta de animais domésticos, eventualmente, e seres humanos.

Os herbívoros, em geral, são hospedeiros acidentais do vírus da raiva, pois, embora participem do ciclo rural da raiva, contribuem apenas como sentinelas à existência do vírus. Sua participação nesse ciclo restringe-se a morte do animal, não ocorrendo envolvimento no processo de transmissão a outras espécies, apenas de forma acidental.

O ciclo epidemiológico no Brasil, representado na figura 7, pode ser dividido didaticamente em quatro ciclos: aéreo, rural, urbano e silvestre. Conforme explica Saúde (2009), no ciclo urbano os principais transmissores são cães e gatos. O morcego é o principal responsável pela manutenção do ciclo silvestre. Outros reservatórios silvestres são: macaco, raposa, coiote, chacal, gato-do-mato, jaritataca, guaxinim e mangusto. No ciclo aéreo são os morcegos. No ciclo rural os herbívoros como: bovinos, bubalinos, equídeos, caprinos, ovinos, suínos e outros também são animais de risco.

Figura 7 – Ciclos epidemiológicos de transmissão da raiva.



Fonte: (SAÚDE, 2007)

Uma vez que o animal foi infectado, o vírus pode replicar-se nas células musculares próximas ao local da inoculação, antes de invadir o sistema nervoso central (SNC). Contudo, ocasionalmente, pode acontecer a entrada direta no SNC sem replicação prévia na musculatura. Em seguida o vírus é conduzido via terminações nervosas motoras, aos nervos periféricos, transmissão célula a célula através de conexões intercelulares conforme é descrito por (BATISTA; FRANCO; ROEHE, 2007).

O tempo real entre a infecção e o aparecimento da doença varia muito e pode ser de dez dias a sete anos. Esse período é chamado de incubação, conforme diz UNESP (2014). O tempo médio corresponde a esse período, no entanto, é de três a 12 semanas.

As manifestações clínicas apresentadas pelos animais infectados são de origem neurológica. Classicamente a raiva apresenta três fases distintas: a fase prodrômica, a fase excitativa e a fase paralítica, conforme apresentado e explicado por (BREDT; ARAUJO; JUNIOR, 1996).

A fase prodrômica é a fase de mais curta duração com 2 a 3 dias, caracterizada

pelos sinais iniciais da doença, quando o animal apresenta pequenas alterações comportamentais como: hiperexcitabilidade aos estímulos externos, como luz, ruídos, deslocamentos e ar; o animal não consegue deglutir, ocasionando salivação por esta dificuldade.

Na fase excitativa os sintomas observados são os mais facilmente associados à doença. Durante esta fase, encontram-se exacerbados os sinais de hiperexcitabilidade observados durante a fase prodrômica. Além disso, o animal pode tomar-se muito agressivo, investindo contra outros animais, contra o homem, e até contra objetos inanimados. Para os animais que apresentam vidente agressividade diz-se estar cometida a raiva na sua forma furiosa. A fase excitativa pode durar de 3 a 7 dias, durante os quais a transmissão o vírus rábico se faz mais frequente.

A fase paralítica ocorre, geralmente, após a fase excitativa, caracterizando-se por paralisia progressiva que parte dos membros posteriores em direção à cabeça conforme demonstrado na figura 8. A morte, por asfixia, ocorre quando a paralisia chega à musculatura respiratória. Em herbívoros é comum não se observar a fase excitativa: o animal passa da fase prodrômica à fase paralítica "raiva paralítica".

Figura 8 – Fases da forma paralítica da raiva em bovinos

Fonte: (EMBRAPA, 2017)

O diagnóstico laboratorial é imprescindível para a determinação do foco, pois a ocorrência de um foco de raiva será reconhecida apenas quando houver um ou mais casos da doença confirmados através de testes laboratoriais.

Não apenas dos bovinos, mas como de qualquer herbívoro com suspeita de raiva, devem ser colhidas durante a necropsia, amostras do sistema nervoso central (SNC). No caso de ruminantes, o encéfalo. Em consequência do perigo de infecção para outros animais, e até mesmo para o homem, os animais infectados com o vírus da raiva costumam ser sacrificados.

Uma forma de se combater a raiva, é fazer o controle das zonas em que existam locais de abrigo dos morcegos e fazer o levantamento de localidades em que existem casos da doença.

A captura, coleta e manejo de morcegos precisa da autorização de órgãos ambientais (Instituto Chico Mendes de Conservação da Biodiversidade (ICMBio) ou Instituto de Defesa Agropecuária e Florestal (IDAF) e só podem ser executadas por profissionais habilitados e cadastrados no Sistema de Autorização e Informação em Biodiversidade conforme explica ICMBIO (2012) sobre atividade e manejo com animais silvestres.

O principal objetivo da captura é a coleta de morcegos para envio de amostras ao laboratório para realização de exames de raiva (WITT, 2012). Além disto, pode-se realizar a coleta de outros tecidos (sangue e vísceras, por exemplo) para verificação da presença de bactérias, fungos, parasitos e outras doenças transmissíveis ao homem, importante ressaltar que são tomadas vários cuidados com o material coletado, para evitar contaminações do material e do responsável pela coleta.

Pode ser realizado o controle dos morcegos, utilizando-se substâncias anticoagulantes, como a warfarina (ARRUDA et al., 2013). Esta é, por sua vez, passada no dorso de morcegos capturados por meio de armadilhas. Devido ao hábito dos morcegos de limparem uns aos outros, por meio da lambedura, estes irão ingerir a substância anticoagulante presente no corpo do animal e, deste modo, irão sangrar até morrer.

A principal medida de profilaxia da raiva é a vacinação dos animais, especialmente em áreas endêmicas. Deve ser feita em animais acima de três anos de idade e revacinados anualmente. Esta é uma vacina que contém vírus inativado, e deve ser aplicada por via subcutânea ou intramuscular, na dosagem de 2 ml por animal (CAMPO, 2017).

A raiva é uma enfermidade de notificação compulsória (obrigatória), sendo assim, cabe ao proprietário notificar imediatamente ao Serviço Veterinário Oficial, a suspeita de casos de raiva em herbívoros, bem como a presença de animais apresentando mordeduras por morcegos hematófagos, ou ainda, informar a presença de abrigo desses morcegos (como cavernas, casas abandonadas, entre outros). A não notificação expõe o rebanho da região a riscos, bem como o próprio homem. Caso o proprietário não realize a notificação, caberá sansão legal a este.

5.2 PLATAFORMA ANDROID

Hoje temos lidado cada vez mais com a tecnologia na palma da mão. Isso vem principalmente pelo conceito dos Smartphones, os quais possuem todo e qualquer tipo de aplicativo disponível para lidar com tarefas simples do dia a dia, por exemplo: ver o extrato bancário, ler um e-mail, tirar uma foto, entrar em redes sociais, registrar os dados de uma caminhada, etc.

O Android é um sistema operacional baseado em Linux projetado para dispositivos móveis touchscreen, como smartphone e tablet. Foi inicialmente desenvolvido pela Android Inc, com apoio financeiro da Google, que posteriormente comprou a empresa. Foi lançado em 2007 junto com a fundação Open Handset Alliance, um consórcio de hardware, software, telecomunicações e empresas dedicadas ao avanço dos dispositivos móveis. Fazem parte desse grupo empresas como: Google, LG, Samsung, Sony, Toshiba, HTC, Hauwei, Sprint Nextel, China Mobile, ASUS, Intel e etc, como explica o Lecheta (2013).

Esse grupo criou um padrão de desenvolvimento, e, de certa forma, padronizou a plataforma de código livre para celulares, mesmo não exigindo configurações fixas de software e hardware. Foi criado um sistema que pode utilizar os diversos recursos dos aparelhos, como: câmeras, bluethooth, jogos, GPS, acesso à internet e muito mais. O grande ponto de sucesso em todo esse processo foi ter a Google como parceiro principal, a Google tem sido revolucionária na Web.

Por ser uma plataforma *open source*, o sistema operacional do Google vem sendo cada vez mais procurado por desenvolvedores de software e empresas que desejam investir

no ramo de tecnologia móvel (KASPERBAUER, 2015). Além disso, o Android possui uma grande comunidade de desenvolvedores que criam aplicativos que estendem a funcionalidade de dispositivos, os quais são escritos principalmente em uma versão personalizada da linguagem de programação Java.

Em outubro de 2012, havia cerca de 700 mil aplicativos disponíveis para Android (WOMACK, 2017). O número estimado de aplicativos instalados a partir do Google Play, principal loja de aplicativos do Android, é de 25 bilhões. Esses fatores têm contribuído para tonar o Android plataforma de smartphone mais usado do mundo.

5.2.1 Arquitetura da Plataforma

O Android é uma pilha de software com base em Linux de código aberto criada para diversos dispositivos e fatores de forma. A Figura 9 a seguir mostra a maioria dos componentes da plataforma Android.

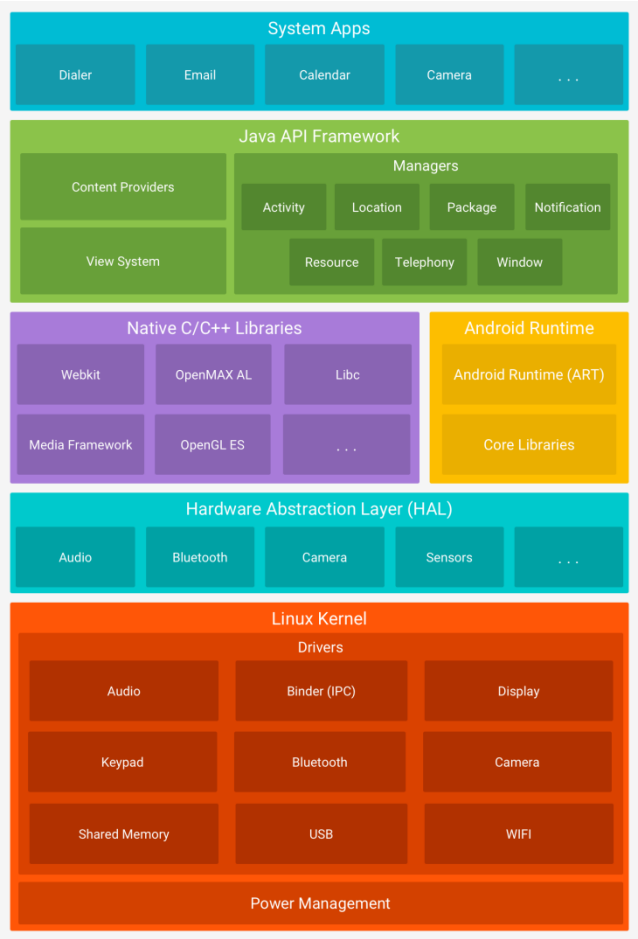
A base do Android é uma versão modificada do kernel Linux 2.6, que foi modificado de forma retirar configurações que não se aplicam em ambientes mobile, que provê vários serviços essenciais, como segurança, rede e gerenciamento de memoria e processos, além de uma camada de abstração de hardware para as outras camadas de software.

A plataforma Android é executada sobre uma máquina virtual chamada Dalvik segundo diz Pereira (2013). Máquinas virtuais emulam, via uma camada de software, as funcionalidades normalmente encontradas em hardware, garantindo assim a portabilidade e a segurança das aplicações. De um ponto de vista conceitual, Dalvik assemelha-se à JVM, a máquina virtual usada para executar programas escritos na linguagem de programação Java. As aplicações para Android são programadas em Java. O desenvolvedor escreve aplicações em Java, e essas são, posteriormente, traduzidas para um formato binário que Dalvik é capaz de emular.

A escolha de Java como a linguagem base para o desenvolvimento no mundo Android deve-se a diversos fatores, como a sintaxe voltada para a orientação a objetos, as facilidades para controle de memória e, sobretudo a grande popularidade dessa linguagem de programação. O Java tem sido nos últimos dez anos, a linguagem mais popular de

acordo com o indicador Tiobe (2017) por exemplo.

Figura 9 – A pilha de software do Android.



Fonte: (ANDROID, 2017a)

5.2.2 Kernel do Linux

O kernel do Linux 2.6 foi escolhido por já ter uma boa quantidade de drivers para dispositivos os quais são maduros e confiáveis, além do bom sistema de gerenciamento de memória e processos, pilhas de protocolos de rede e modelo de drives que são utilizados pelo Android (2017a), e como dito anteriormente, o kernel foi modificado para atuar no ambiente mobile. Além destas funções, o kernel também é usado para administrar alguns serviços de segurança e rede, e serve como uma camada de abstração entre o software, e o hardware.

O desenvolvedor de aplicações não irá programar nesta camada do Android, e só fará uso indireto do kernel através de APIs (Application Programming Interface ou Interface de Programação de Aplicativos) de níveis superiores. Por exemplo: o Android Runtime

(ART) confia no kernel do Linux para cobrir funcionalidades como encadeamento e gerenciamento de memória de baixo nível.

Conforme foi explicado no Android (2017a), o Kernel Linux possibilita o aproveitamento de recursos de segurança principais e gerenciamento de energia e monitoramento de uso da mesma, onde o driver de energia do kernel checa os dispositivos que não estão usando energia e encerra seus processos.

Conforme explica Pereira e Silva (2009) o Android é executado em kernel Linux, e toda vez que o aplicativo for instalado em uma estação Android é criado uma novo usuário Linux para aquele programa, com diretórios que são usados para o aplicativo, mas somente para aquele usuário Linux. Como os aplicativos ficam completamente isolados uns dos outros, qualquer tentativa de acessar informações de outro aplicativo precisa ser explicitamente autorizada pelo usuário, podendo ser negada a instalação do aplicativo, ou autorizada a instalação do aplicativo, mas controlando as permissões que este aplicativo poderá ter através das permissões.

5.2.3 Camada de abstração de hardware (HAL)

A camada de abstração de hardware (HAL) fornece interfaces padrão que expõem as capacidades de hardware do dispositivo para a estrutura da Java API de maior nível. Isso possibilita que a indústria faça sua variantes de versões da plataforma como adaptações de hardware como explica Android (2017b).

A HAL consiste em módulos de biblioteca, que implementam uma interface para um tipo específico de componente de hardware, como o módulo de câmera ou bluetooth. Quando uma API de *Framework* faz uma chamada para acessar o hardware do dispositivo, o sistema Android carrega o módulo da biblioteca para este componente de hardware.

5.2.4 Android Runtime

O Android Runtime (ART) é composto pela máquina virtual chamada Dalvik VM, onde as aplicações são executadas, e por um conjunto de bibliotecas que fornecem boa parte das funcionalidades encontradas nas bibliotecas padrão do Java (ANDROID, 2017a).

O ART é projetado para executar várias máquinas virtuais em dispositivos de baixa memória executando arquivos .dex, um formato de bytecode projetado especialmente para Android, otimizado para oferecer consumo mínimo de memória.

Segundo Android (2017b) existem recursos principais de ART que são:

1. Compilação "ahead-of-time"(AOT) e "just-in-time"(JIT).

2. Coleta de lixo (GC) otimizada.

3. Melhor compatibilidade de depuração, inclusive um gerador de perfil de exemplo, exceções de diagnóstico detalhadas e geração de relatórios de erros, além da capacidade de definir pontos de controle para monitorar campos específicos.

O Android possui várias bibliotecas de tempo de execução que fornecem a maioria das funcionalidades da linguagem de programação Java, inclusive alguns recursos de linguagem Java 8 que a estrutura da Java API.

5.2.5 Dalvik VM

No que se trata do ambiente de execução, a plataforma é composta pela máquina virtual Dalvik e a Core Libraries, durante a inicialização do sistema é criado uma instância especial da Dalvik que será responsável pela criação das demais instâncias onde cada uma das aplicações irá executar, ou seja, toda e qualquer aplicação em Android roda dentro de seu próprio processo, isto é, no contexto da sua instância de máquina virtual.

> A Dalvik VM não pode ser considerada uma JVM (Java Virtual Machine), pois ela não interpreta Java bytecodes. A Dalvik executa arquivos no formato Dalvik Executable, com extensão .dex, um formato de arquivo que é uma espécie de bytecodes de Java otimizados para a Android, ou seja, carrega mais rápido e ocupa menos espaço em memória. A criação do arquivo .dex se dá pela compilação em um compilador da linguagem Java que gera um arquivo .class, este arquivo é transformado em .dex pela ferramenta dx presente no SDK do Android (EHRINGER, 2010).

Outra estratégia utilizada para a otimização de aplicativos descrita por Ehringer (2010) é o compartilhamento de código entre as instâncias da VM. Isso é feito através de um processo chamado Zygote, que é pai de todas as instâncias da Dalvik VM e é inicializado juntamente com o sistema, carregando todas as classes das bibliotecas que provavelmente serão utilizadas com frequência pelos aplicativos. Assim, quando uma aplicação for iniciada, um comando é enviado ao Zygote, que faz um fork, criando um processo que se torna uma nova Dalvik VM, minimizando o tempo de inicialização.

O isolamento entre as aplicações se estende até o garbage collector (GC), que é executado em cada VM de forma separada, evitando que GCs de outros processos interfiram na execução de um determinado GC. Cada instância da VM possui a sua própria pilha de memória, além da pilha compartilhada (do Zygote) e a coleta da memória que não é mais utilizada é feita através de bits de marcação que informam ao GC quais objetos devem ser removidos, um mecanismo útil, pois mantém intocada a memória compartilhada do Zygote (MAIA; NOGUEIRA; PINHO, 2010).

Como explica Dubroy (2011), antes da versão Gingerbread, toda vez que o GC era executado, as threads do aplicativo eram paradas, inconveniência que foi removida com a introdução do GC concorrente .

5.2.6 Estrutura da Java API

O conjunto completo de recursos do SO Android está disponível pelas APIs programadas na linguagem Java. Essas APIs formam os blocos de programação que você precisa para criar os aplicativos Android simplificando a reutilização de componentes e serviços de sistema modulares e principais.

A estrutura da API do Android é composta de duas partes: uma biblioteca que reside em cada aplicativo na maquina virtual e uma implementação da API que é executada em processo (s) do sistema. A biblioteca da API é executada com as mesmas permissões como a aplicação que a acompanha, enquanto a implementação da API no processo do sistema não tem restrições (FELT et al., 2011). Chamadas de API que lêem ou mudam o estado global do telefone é processado pela biblioteca para a implementação da API no processo do sistema.

Os desenvolvedores têm acesso completo às mesmas APIs de *framework* usado pelas aplicações principais. A arquitetura da aplicação é projetada para simplificar a reutilização de componentes que se torna muito viável para o desenvolvimento. Qualquer aplicação pode publicar suas capacidades e qualquer outro aplicativo pode então fazer uso dessas capacidades (sujeito a restrições de seguranças Impostas pela estrutura). Este mesmo mecanismo permite que componentes a sejam substituídos pelo usuário (HOLLA; KATTI, 2012).

5.2.7 Aplicativos do Sistema

O Android em seu estado sem personalizações possui um conjunto de aplicativos principais para e-mail, envio de SMS, calendários, navegador de internet, contatos etc. Os aplicativos inclusos na plataforma já veem com a aplicação, o usuário que vai usar a aplicação pode instalar diversos aplicativos fabricados por terceiros (ANDROID, 2017c) .Portanto, um aplicativo terceirizado pode se tornar o navegador da Web, o aplicativo de envio de SMS ou até mesmo o teclado padrão do usuário, porém existem algumas exceções como o aplicativo de configuração do sistema que não pode mudar, dentro outras.

Os aplicativos do sistema funcionam como aplicativos para os usuários e fornecem capacidades principais que os desenvolvedores podem acessar pelos próprios aplicativos. Por exemplo: se o seu aplicativo quiser enviar uma mensagem SMS, não é necessário programar essa funcionalidade, é possível invocar o aplicativo de SMS que já está instalado para enviar uma mensagem ao destinatário que você especificar.

5.3 WEB SERVICE

Segundo o que Menéndez (2002) diz sobre Web Services, pode se afirmar que a necessidade da integração entre aplicações e a utilização unificada de processos encontrados em diferentes sistemas e escritos em diferentes linguagens são alguns exemplos de carências encontradas em empresas nos dias de hoje. A fim de sanar estas questões, a tecnologia dos Web services foi criada, permitindo assim, disponibilizar formas de integrar sistemas distintos, modularizar serviços e capacitar a integração e consumo de informações, resolvendo o problema de disparidade das linguagens.

Seguindo o que Moura e Cheiran (2016) descrevem, o Web Service trouxe vários tipos de aplicações e recursos , que rodam em diversas plataformas e sistemas operacionais. Originalmente foi criado como uma interface padronizada baseada em tecnologias já conhecidas, basicamente a especificação XML e o protocolo HTTP, tecnologias que são usadas em diversas aplicações.

Essa padronização foi desenvolvida pela W3C, um consórcio de empresas de tecnologia que tem como o objetivo criar padrões comuns para conteúdo da Web, apoiada por grandes empresas como a Microsoft, IBM, HP entre outras.

O formato XML é recomendado pela W3C para criar documento com dados organizados de forma hierárquica. Está entre suas principais características o fato de fornecer uma sintaxe básica que pode ser utilizada para compartilhar informações entre diferentes arquiteturas e aplicações.

5.3.1 Web Services SOAP x RESTFul

Os Web Services baseados no protocolo SOAP para SOA, apesar de geralmente terem desempenho inferior aos criados com RESTFul, ainda continuam sendo muito utilizados em processos distribuídos, principalmente naqueles que requerem confiabilidade e segurança.

A segurança básica relacionada Web Services RESTFul pode ser estabelecida em nível do protocolo de comunicação utilizando, por exemplo, o protocolo HTTPS (HyperText Transfer Protocol Secure) para criptografia dos dados e exigindo autenticação, por exemplo, com o auxílio do protocolo Kerberos. Em contrapartida, extensões WS-*(WS-Addressing, WS-Reliability, WS-Addressing, WS-Policy, etc) para Web Services com SOAP podem garantir não apenas segurança, mas também diversas outras funcionalidades em nível de mensagem. A Tabela 1 exibe um breve resumo comparativo entre Web Services nas arquiteturas SOA e ROA.

As seguintes características do protocolo de comunicação RESTful colaboraram com a escolha do mesmo no desenvolvimento deste trabalho:

Tabela 1 – Resumo comparativo: SOA x ROA.

	SOA - SOAP	**ROA- RESTFul**
Conceito	”Tudo” é um serviço	”Tudo” é um recurso
Web Service	É o próprio serviço	O serviço representa o recurso
Descoberta	WSDL	WADL
Publicação	ESB ou UDDI	APP
Pesquisa	Baseada em serviços	Baseada em recursos
Linguagens	Java, C#, C++, .NET, etc.	HTML, JSON, Javascript,Java,etc.

Fonte: (CHEN et al., 2009)

- Pequenas quantidades de conteúdo textual formatado em XML OU JSON;

- Na maioria dos casos opera sobre protocolo HTTP;

- É focado em expor recursos da aplicação de forma pública, ou seja, por meio de métodos conhecidos;

- Não é necessário suporte especifico de linguagem, uma vez que, os dados são transmitidos usando um XML simples ou uma string JSON;

5.3.2 Web Service RESTFul

REST define um conjunto de princípios arquitetônicos pelos quais você pode projetar serviços da Web que se concentram os recursos de um sistema, incluindo a forma como os estados de recursos são endereçados e transferidos por HTTP uma ampla gama de clientes escritos em diferentes idiomas. Se medido pela quantidade de serviços da Web que o utilizam, o REST surgiu nos últimos anos apenas como um design de serviço da Web predominante modelo.

Uma das principais características de um serviço RESTful da Web é o uso explícito de métodos HTTP em uma maneira que segue o protocolo como definido pelo RFC 2616. HTTP GET, por exemplo, é definido como um método de produção de dados que se destina a ser usado por um aplicativo cliente para recuperar um recurso, para buscar dados de um servidor Web ou executar uma consulta com a expectativa de que o servidor da Web procure e responda com um conjunto de recursos correspondentes (RODRIGUEZ, 2008).

O REST pede aos desenvolvedores que utilizem métodos HTTP de forma explícita e de forma consistente com o definição do protocolo. Este princípio básico de design REST estabelece um mapeamento um-para-um entre criar, ler, atualizar e excluir (CRUD) operações e métodos HTTP.

Um serviço que reconhece todos os princípios propostos pelo estilo REST são denominados RESTFul e, frequentemente, utilizam os quatro métodos usados do protocolo HTTP (GET, PUT, POST e DELETE) para manipular recursos. Partindo desse principio Ruby e Richardson (2007) completa que, ROA é a consumação de uma arquitetura RESTFul que permite transformar um problema em um Web Service RESTFul.

- POST para criar um recurso no servidor;
- GET para recuperar um recurso;
- PUT para alterar o estado de um recurso ou atualizá-lo;
- DELETE para remover ou excluir um recurso;

Com o uso de Web Services desenvolvedores tem a capacidade de desenvolver aplicações complexas, aptas a integrar dados com outras aplicações (KASPERBAUER, 2015). Isso se torna um recurso muito bom, tendo em vista o grande crescimentos de aplicações voltadas para celulares e tablets.

5.3.3 Json

Conforme explicado por Kotamraju (2014) o JSON (JavaScript Object Notation) é um canal de comunicação de dados, baseado em texto e independente da linguagem de programação, de fácil interpretação para humanos e máquinas.

A característica mais positiva do JSON está no fato de ser um formato de dados econômico, que não exige muito espaço se comparado ao XML, assim como uma rápida transferência de dados por meio de conexões lentas, sendo ótimo para uso de aplicações móveis, pois seu custo de decodificação é baixo (EULALIO; CORDEIRO; SOUZA, 2017).

O JSON pode representar dois tipos estruturados conforme descrito por Kotamraju (2014): objetos e matrizes. Um objeto é uma coleção não ordenada de zero ou mais pares de nomes/valores. Uma matriz é uma sequência ordenada de zero ou mais valores. Os valores podem ser strings, números, booleanos, nulos e estes dois tipos estruturados.

> "JSON é frequentemente utilizado em aplicações Ajax, configurações, bancos de dados e serviços web RESTful. Todos os web sites populares oferecem JSON como formato para intercâmbio de dados com seus serviços web RESTful (KOTAMRAJU, 2014)".

A API do Java para processamento JSON fornece APIs portáteis que permitem analisar, gerar, transformar e consultar JSON.

Estas são estruturas de dados universais, todas as linguagens de programação modernas as suportam. É aceitável que um formato de troca de dados que seja independente de linguagem de programação se baseia nestas estruturas (JSON, 2016).

5.4 JAVAFX

O JavaFX é uma plataforma para desenvolvimento de interfaces interativas e com muitos recursos no lado do cliente. A intenção dessa tecnologia é possibilitar a criação de aplicações RIA (Rich Internet Application) que disponibilizem aos usuários uma mesma interface, independente do dispositivo usado. Para isso, apresenta vários recursos para criar gráficos,tem recursos multimídia (como som, imagem ou vídeo),

entre outros recursos de interface. Além disso, toda essa plataforma pode ser integrada a recursos já criados em Java, possibilitando o reuso de aplicações já implementadas e, permite a integração com todos os recursos de alta complexidade da plataforma Java, mantendo o poder de desenvolvimento das aplicações e adicionando o potencial de novos recursos visuais (PREMKUMAR; MOHAN, 2010).

A aplicativos JavaFX podem ser personalizados, inclusive as partes gráficas. Cascading Style Sheets (CSS) de aparência distinta e estilo de implementação, de modo que os desenvolvedores podem concentrar-se na codificação. Os designers gráficos podem facilmente personalizar a aparência e o estilo do aplicativo através do CSS. Para separar a interface de usuário (UI) e da lógica de *back-end*, então você pode desenvolver os aspectos de apresentação da interface do usuário no idioma FXML script e usar o código Java para o aplicativo lógica. Criar UIs sem escrever código, em seguida, usar o JavaFX *Scene Builder*. Ao projetar a interface do usuário, *Scene Builder* cria marcação FXML que pode ser portado para um Ambiente de Desenvolvimento Integrado (IDE) para que desenvolvedores possam adicionar a lógica de negócios (ORACLE, 2014).

> "As APIs JavaFX estão disponíveis como um recurso totalmente integrado do *Java SE Runtime Environment* (JRE) e o *Java Development Kit* (JDK). Porque o JDK está disponível para todas as principais plataformas de desktop (Windows, Mac OS X e Linux), aplicativos JavaFX compilado para JDK 7 e mais tarde também executado em todas as principais plataformas de desktop. Suporte para plataformas ARM também foi disponibilizado com o JavaFX 8. JDK para ARM inclui a base, gráficos e controla componentes do JavaFX (ORACLE, 2014)".

Os seguintes recursos estão incluídos no JavaFX 8 e versões posteriores. Itens que foram introduzidos no JavaFX 8 Release estão devidamente identificadas:

1. APIs Java: JavaFX é uma biblioteca Java que consiste em classes e interfaces que são escritos em código Java. As APIs são projetados para ser uma alternativa amigável para linguagens Java Virtual Machine (Java VM), tais como JRuby e Scala (JAVAFX, 2014).

2. FXML e *Scene Builder*: O FXML possibilita a criação de todo tipo de interface de usuário. Adicionalmente, oferece grande facilidade na criação de interfaces com

cenas gráficas complexas e/ou numerosas, formulários de entrada de dados e animações. Ainda, pode se destacar alguns benefícios do uso do FXML para a criação de interfaces de usuário (JAVAFX, 2014):

a) Facilidade no desenvolvimento, manutenção e teste das interfaces gráficas;

b) O FXML não é compilado, portanto, não é necessário recompilar após modificá-lo;

c) O uso de mecanismos de localização para modificar o idioma corrente é mais simples;

Para conseguir de maneira rápida e eficiente de criar interfaces gráficas utilizando FXML é por meio da ferramenta JavaFX *Scene Builder*. Essa é uma ferramenta de designer que gera o código fonte da interface gráfica em FXML.Também permite fazer a ligação entre a interface gráfica e a aplicação. O uso da ferramenta segue o padrão de arrastar e soltar, possibilita que o desenvolvedor tenha uma ágil percepção do resultado da criação da interface visual (JAVAFX, 2014).

> "A criação de aplicações JavaFX usando o recurso do FXML permite explorar de maneira bastante eficiente o modelo de desenvolvimento conhecido como Model-View-Controller (MVC) (traduzido como Modelo-Visão-Controlador). Esse modelo envolve a interação de três estruturas: Model : Representa o comportamento e modelo de dados do domínio da aplicação; View : Representa a interface de interação com o usuário; Controller : Representa o controle de fluxo e lógica da aplicação.
> (MARQUEZINI; AGUINALDO; ALMEIDA, 2013)"

Estudando uma aplicação JavaFX com as características do modelo MVC, o FXML representa a interface do usuário, ou seja, o View. O Controller é definido como uma classe Java, normalmente como uma classe de inicialização, e definida no FXML como controlador. E o Model, como os dados da aplicação, são definidos como classes Java, que farão a interação entre o controlador e a interface de usuário, onde serão exibidos esses dados (JAVAFX, 2014).

Segundo o Oracle (2014) os seguintes recursos estão disponíveis nas versões do java, desde de que seja acima da versão 8 do JavaFX:

1. WebView. Um componente web que usa a tecnologia *WebKitHTML* para torná-lo possível incorporar páginas da web dentro de um aplicativo JavaFX. JavaScript em execução em WebView pode chamar APIs Java e APIs Java pode chamar JavaScript em execução em WebView. Suporte para recursos HTML5 adicionais, incluindo Web Sockets, Trabalhadores da Web e fontes da Web e capacidades de impressão foram adicionadas no JavaFX 8.

2. Interoperabilidade balanço. Aplicações Swing existentes podem ser atualizados com recursos do JavaFX, como a reprodução de gráficos rich media e conteúdo da Web incorporado. A *SwingNodeclasse*, que permite incorporar conteúdo balanço em aplicações JavaFX, foi adicionado em JavaFX 8.

3. Controles de interface do usuário e CSS. JavaFX fornece todos os principais controles de interface do usuário que são necessários para desenvolver um aplicativo completo. Os componentes podem ser explorados com tecnologias Web padrão, tais como CSS. Os controles datePicker e TreeTableView UI estão agora disponíveis com o lançamento do JavaFX 8. Além disso, os CSS aulas *Styleable* tornaram-se API pública, permitindo que os objetos sejam denominado por CSS.

4. Gráficos 3D Recursos. As novas classes de API para Shape3D (Box, Cylinder, MeshView, and Spheresubclasses), SubScene, Material, PickResult, LightBase (AmbientLighte PointLightsubclasses), e SceneAntialiasing foram adicionados à biblioteca de gráficos 3D no JavaFX 8. A classe Camera API também foi atualizado nesta versão.

5. API Canvas. A API Canvas permite desenhar diretamente dentro de uma área da cena JavaFX, que consiste em um elemento gráfico (nó).

6. API de Impressão. O javafx.print pacote foi adicionado no Java SE 8 liberação e fornece as classes públicas para a API JavaFX Printing.

7. Suporte Rich Text. JavaFX 8 traz suporte texto melhorado para JavaFX, incluindo texto bi-direcional e scripts de texto complexos, tais como tailandês e Hindu nos

controles, e multi-line, texto multi-estilo em nós de texto.

8. Suporte multitouch. JavaFX fornece suporte para operações multitoque, com base nas capacidades da plataforma subjacente.

9. Suporte Hi-DPI. JavaFX 8 agora suporta Hi-DPI, ferramenta para dimensionamento de telas.

10. Pipeline de gráficos acelerados por hardware. Os gráficos JavaFX são baseados no pipeline de renderização de gráficos (Prism). O JavaFX oferece gráficos suaves que são processados rapidamente através do Prism uando é usado com uma placa gráfica ou unidade de processamento de gráficos (GPU) suportada. Se um sistema não possui uma das GPUs recomendadas suportadas pelo JavaFX, então o Prism é padrão para a pilha de renderização de software.

6 RESULTADOS

Utilizando como base a metodologia da disciplina de Laboratório de Engenharia de Software IFES-PDS-LES (2017) foram elaboradas os artefatos detalhados a seguir:

6.1 LEVANTAR REQUISITOS (VISÃO GERAL)

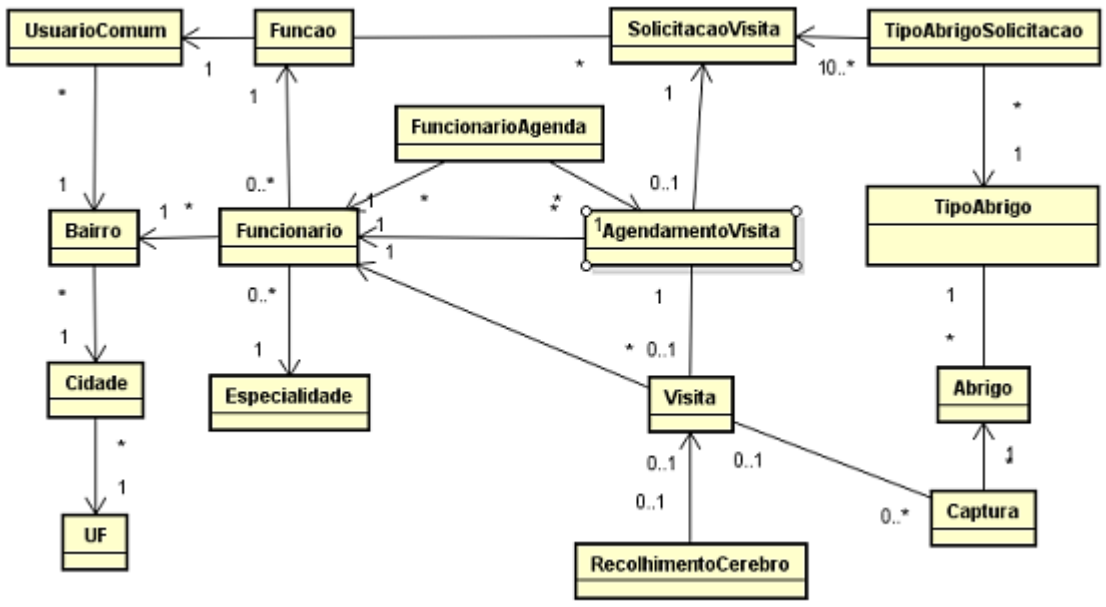
Nesta etapa foi realizado o primeiro contato com a instituição demandante (IDAF). Um funcionário com perfil técnico apresentou o processo atual de controle da raiva realizado pelo IDAF.

A partir dessa conversa identificou-se as possibilidades de sistematização de algumas atividades desse processo. Neste ponto, uma descrição geral foi elaboradora.

6.2 ELABORAR MODELO CONCEITUAL

No modelo conceitual foram identificados os principais conceitos do domínio de problema, conforme Figura 10. Nesta etapa não foram identificados os atributos das classes, mas apenas seus relacionamentos.

Figura 10 – Modelo Conceitual



Fonte: Própria

6.3 LEVANTAR REQUISITOS

Nesta atividade foi elaborado o documento de requisitos. Neste documento foram especificadas as funcionalidades do sistema. Ao todo foram levantadas 55 funcionalidades, 36 de cadastros, 12 de processos de negócios e 7 de listagens. Todas essas informações foram extraídas a partir de entrevistas que foram realizadas com um técnico do IDAF. A seguir temos os principais requisitos de negócios.

- Solicitação de Visita;
- Agendamento Visita;
- Registrar Abrigo;
- Registrar Captura;
- Registrar Recolhimento de Cérebro;
- Registrar Visita;
- Visualizar Marcações no Mapa;

6.4 ORGANIZAR REQUISITOS

Os requisitos identificados na etapa anterior foram organizados nas seguintes categorias: processos de negócio, manutenção de cadastros e listagens/relatórios.

Os requisitos de processos de negócio foram: registrar abrigo, registrar agendamento de visita, registrar captura, enviar mensagem, registrar recolhimento de material, registrar solicitação de visita e registrar visita.

Na categoria de manutenção de cadastros temos os seguintes: bairro, cidade, especialidade, função, funcionário, tipo de abrigo, unidade federativa e usuário.

Dentre as listagens podemos citar: listagem de usuários, listagem de funcionários, listagem de cidades, listagem de solicitações de visita, listagem de agendamentos de visita, listagem de visita, além da exibição de pontos no mapa utilizando a API Google Maps, listagem de tipos de abrigos.

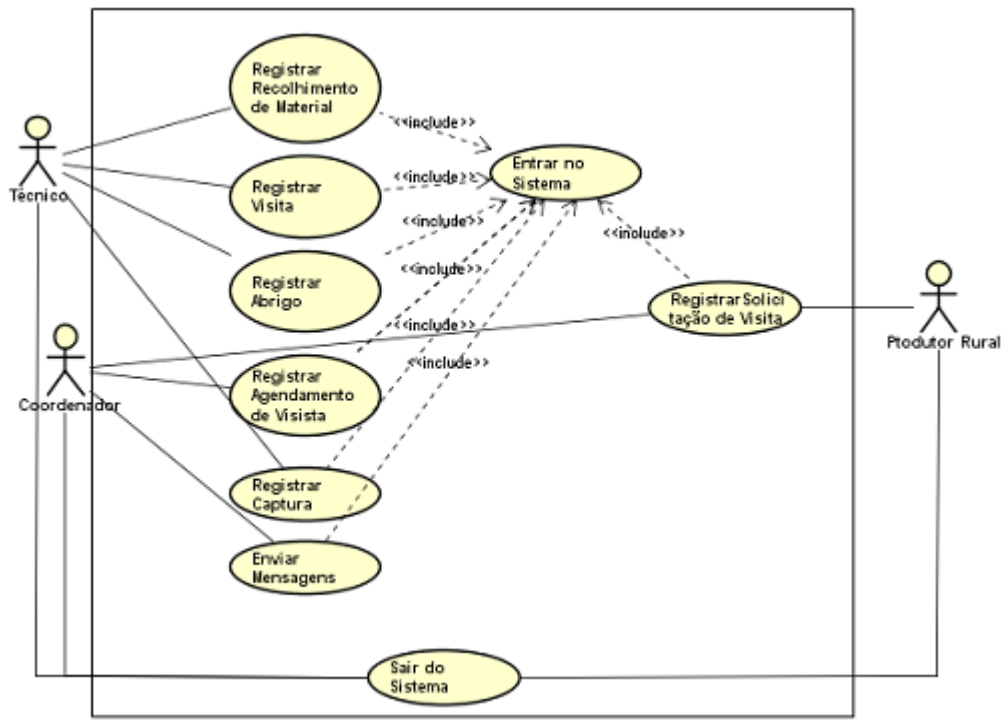
6.5 ELABORAR CASOS DE USO

Para apresentação das principais funcionalidades do sistema, bem como seus respectivos atores, foi utilizado o diagrama de casos de uso.

Este diagrama foi elaborado utilizando a ferramenta Astah (2017). O sistema contempla três diferentes perfis de usuários (atores), a saber: coordenador, técnico e produtor rural.

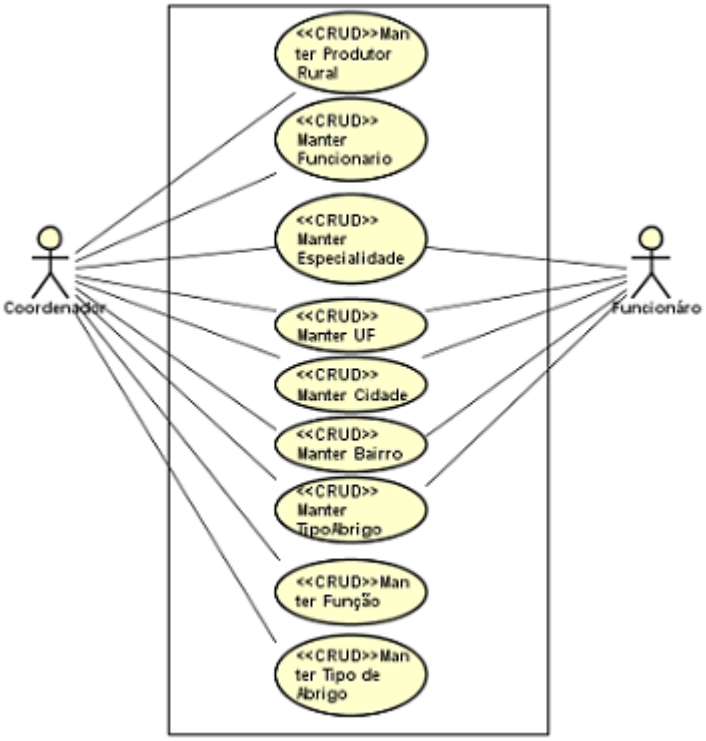
Para facilitar a visualização dos casos de uso foram elaborados três diagramas diferentes: diagrama de casos de usos dos processos de negócio (Figura 11), diagrama de casos de usos das manutenções de cadastros (Figura 12) e diagrama de casos de usos das listagens/relatórios (Figura 13).

Figura 11 – Processos de Negócios



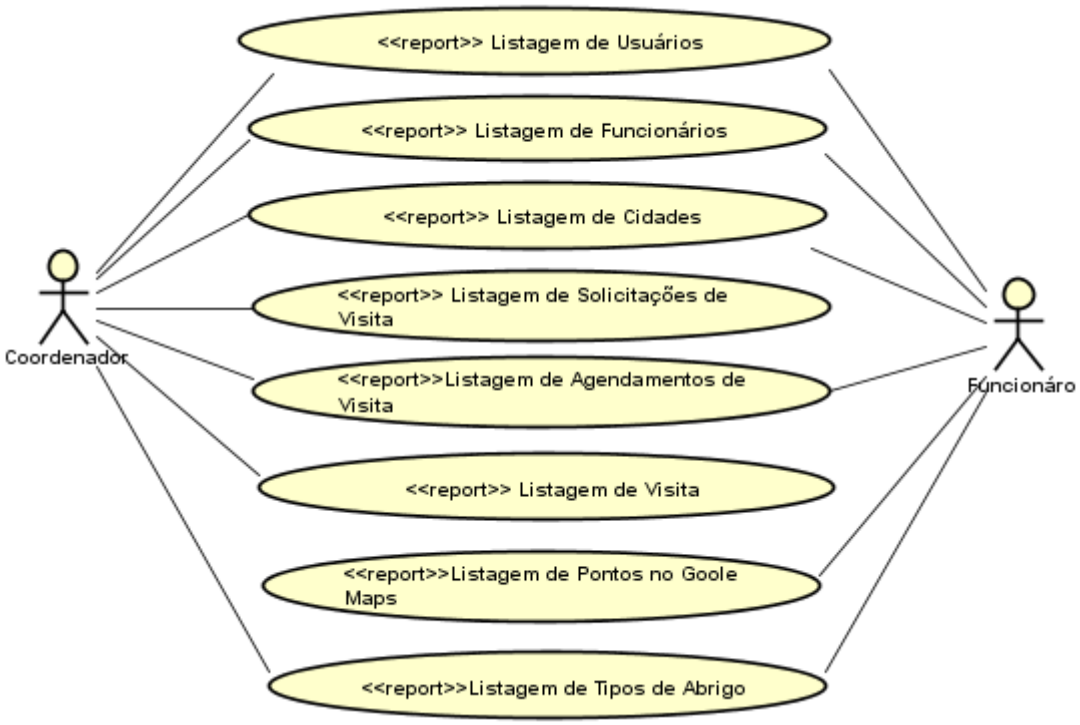
Fonte: Própria

Figura 12 – CRUDS



Fonte: Própria

Figura 13 – Relatórios

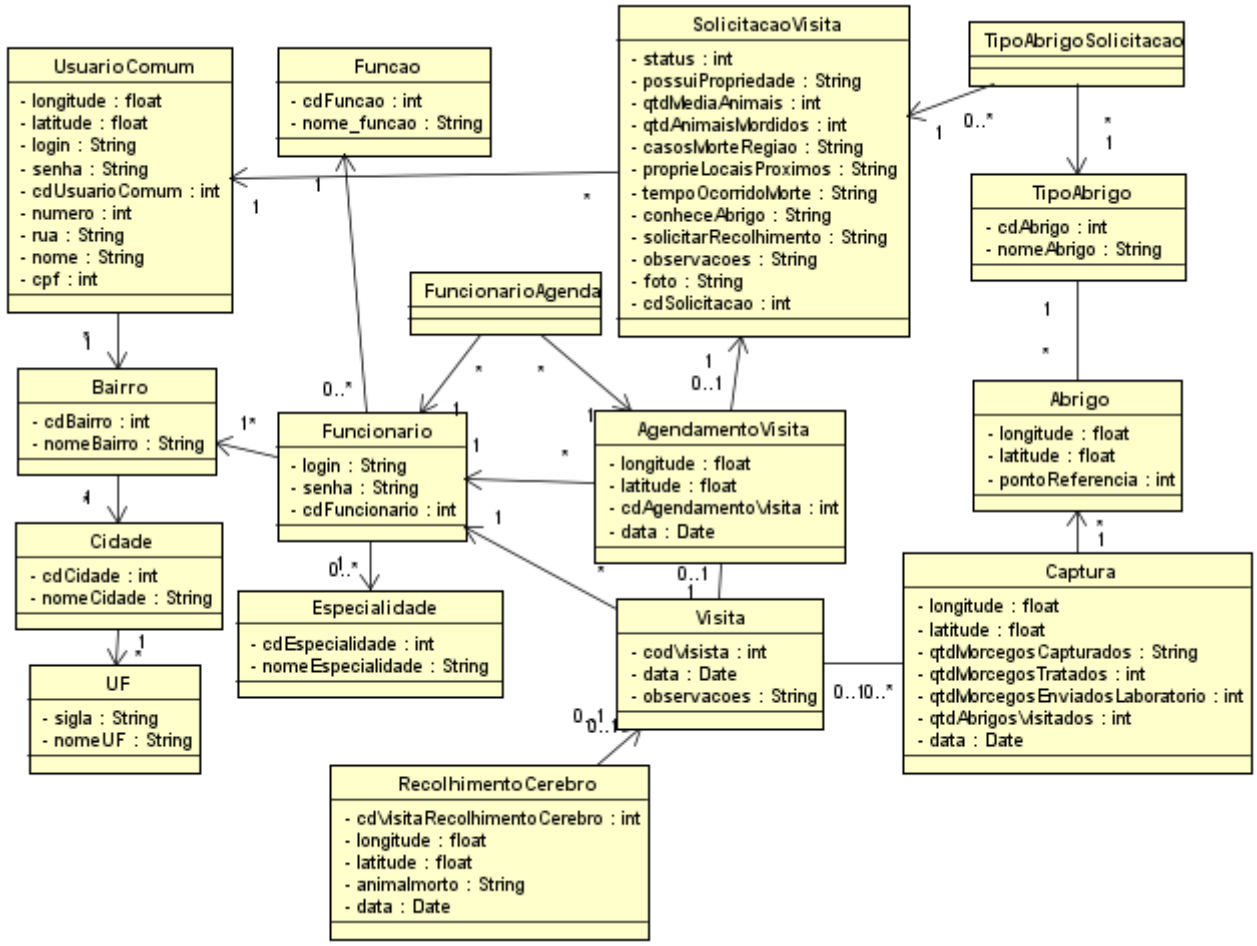


Fonte: Própria

6.6 ELABORAR DIAGRAMA DE CLASSES DO PROJETO

Nesta tarefa foi elaborada uma nova versão do modelo conceitual (Figura 14), incluindo os atributos das classes, identificados no levantamento de requisitos.

Figura 14 – Modelo Conceitual



Fonte: Própria

6.7 ELABORAR BANCO DE DADOS

Com base no diagrama de classes de projeto foi elaborado o script de criação do banco de dados. O SGBD escolhido para o armazenamento das informações do sistema foi o MySQL (2017). A ferramenta utilizada para a geração do banco de dados foi o phpMyAdmin (2017). O banco de dados ficara hospedado juntamente ao Web Service após implementação.

6.8 GERAR CÓDIGO ALTO-NÍVEL

O código alto-nível (classes contendo atributos e assinaturas dos métodos) foi gerado utilizando o recurso de exportação para linguagem Java na ferramenta Astah (2017).

6.9 IMPLEMENTAR

A implementação do sistema se deu por meio da ferramenta NetBeans (2017), para a aplicação JavaFX e Web Service RESTFul, e Android Studio (2017), para os aplicativos na visão do produtor rural e técnico. Conforme mencionado anteriormente, a linguagem utilizada para o desenvolvimento foi Java. É importante ressaltar que para exibição de mapas com marcações de georreferenciamento trabalhou-se com a API Google Maps (2017).

A seguir serão apresentadas as telas implementadas para as principais funcionalidades do sistema.

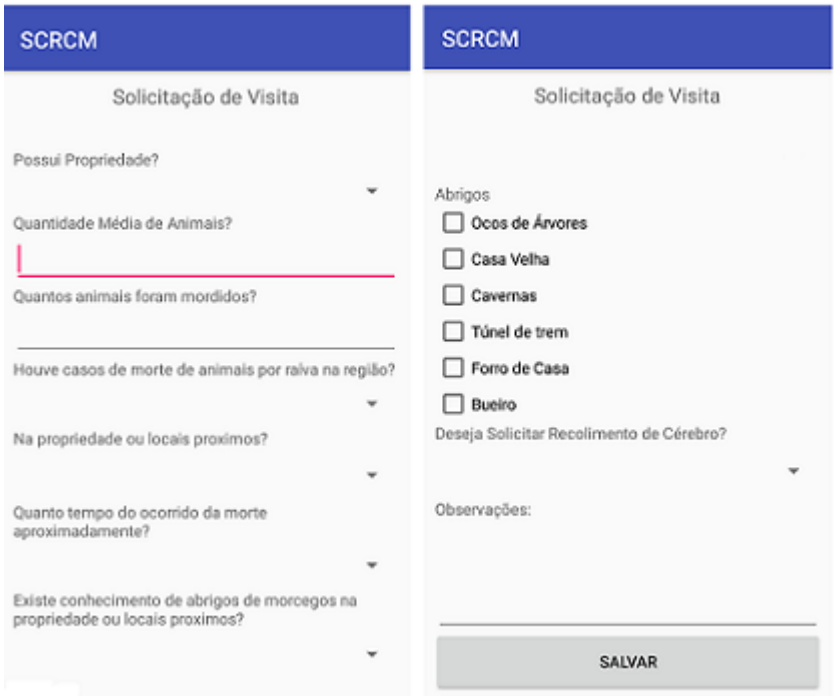
- Solicitação de Visita (Figura 15): a imagem está dividida em duas partes para possibilitar a visualização de todo o conteúdo da tela referente a esta funcionalidade.

- Agendamento Visita (Figura 16): são duas telas que se sobrepõem ao clicar no botão cadastrar, abrindo nova tela para agendamento.

- Registrar Visita (Figura 17): a imagem está dividida em duas partes para possibilitar a visualização de todo o conteúdo da tela referente a esta funcionalidade.

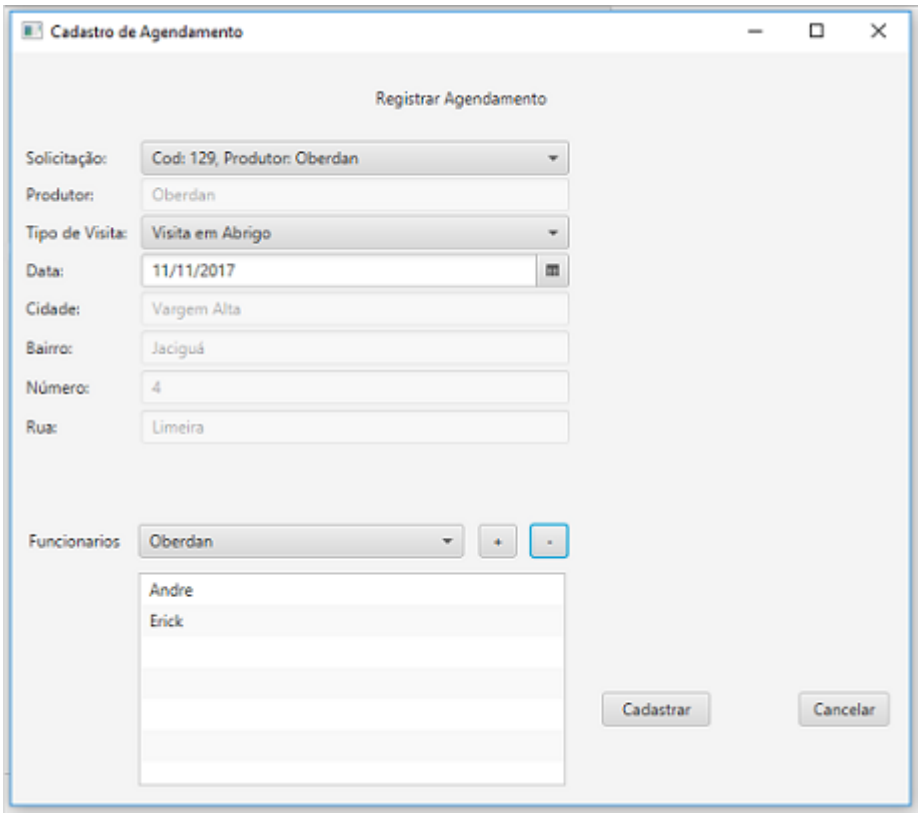
- Google Maps (Figura 18): no mapa pode-se observar pontos de marcação, sendo que os verdes são abrigos cadastrados, os violetas são as capturas de morcegos e os cianos são recolhimentos de materiais.

Figura 15 – Aplicativo Android: Usuário Produtor Rural: Funcionalidade de Solicitação de Visita



Fonte: Própria

Figura 16 – Aplicação JafaFX: Usuário Coordenador: Funcionalidade Agendamento de Visita

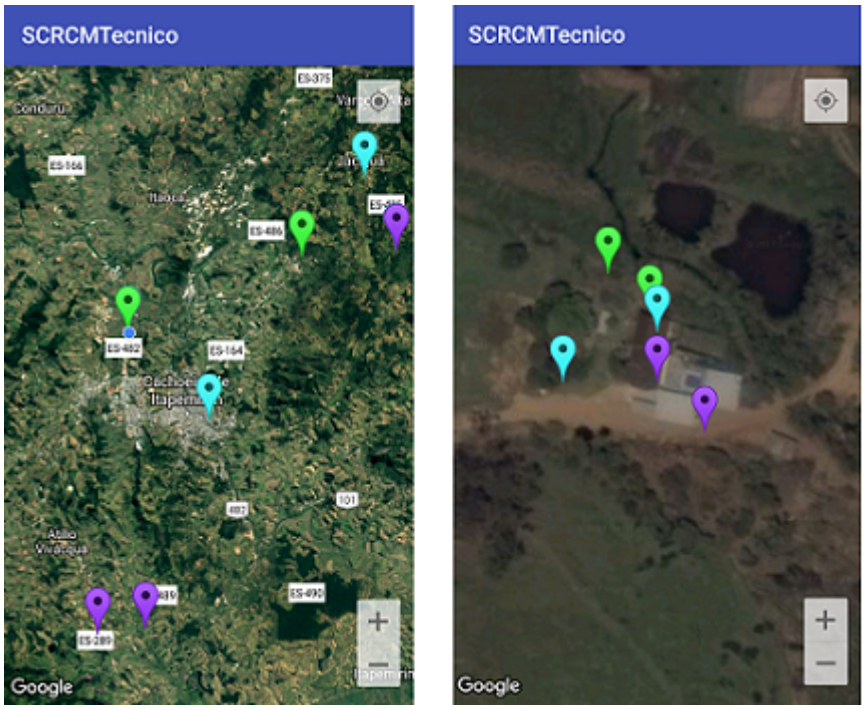


Fonte: Própria

Figura 17 – Aplicativo Android: Usuário Técnico: Funcionalidade Registro de Visita

Fonte: Própria

Figura 18 – Aplicativo Android: Usuário Técnico: Funcionalidade Google Maps com Marcações



Fonte: Própria

6.10 IMPLEMENTAÇÃO E FUNCIONAMENTO DO WEB SERVICE

O Web Service foi todo desenvolvido e implantado na linguagem Java e foi separado em pacotes, pacote DAO, Model, Service e Database.

DAO: contem as clases que fazem o intermediário com o banco de dados, essas classes fazem conexão e transações com este.

Model: todas as classes que fazem parte do modelo estão nesse pacote.

Service: chama um ou mais métodos de DAOs para obter as informações necessárias e retorna os dados.

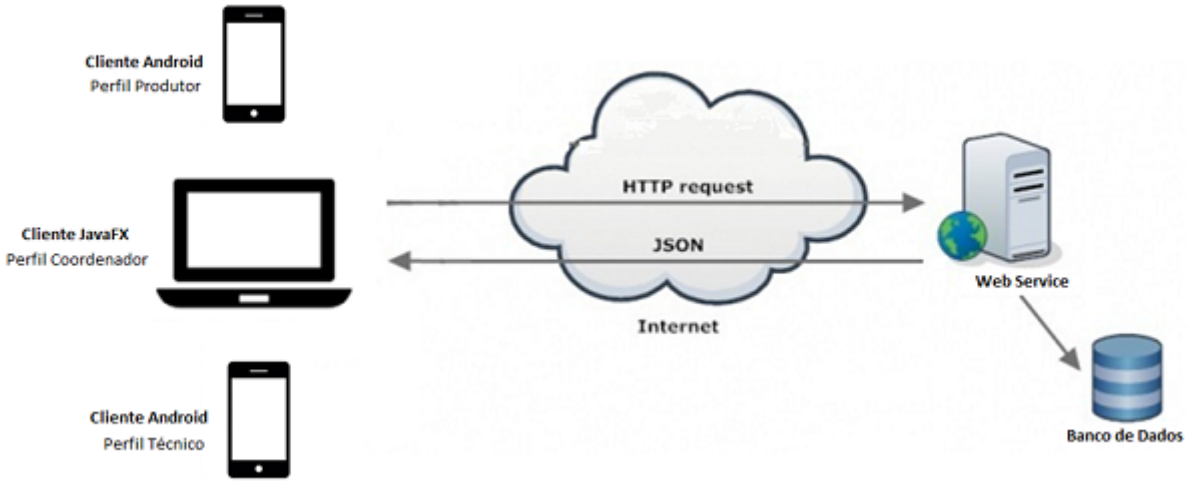
Database: onde se localiza os dados para conexão com o banco de dados, como senha e endereço.

6.10.1 Funcionamento do WebService RestFul

A Figura 19 representa o funcionamento do Web Service Restful. O Web Services de arquitetura orientada a recursos ROA (Resource-Oriented Architecture) baseado no protocolo REST (Representational State Transfer) e linguagem JSON (JavaScript Object Notation). Em resumo, um Web Services RESTful JSON.

Quando um cliente deseja acessar um recurso do Web Service ele faz uma requisição via protocolo HTTP que é transmitida no formato JSON, quando essa requisição chega ao servidor ela é convertida ao seu formato de origem, a partir dai ele vai identificar qual foi a requisição, sendo ela, GET, PUT, POST ou DELETE. Tendo identificado qual foi a requisição ele irá atendela e executa-lá, e retorna-la no formato JSON, ao chegar no cliente a respostas será convertida para o formato de origem(boolean, string, numerico e etc).

Figura 19 – Funcionamento do Web Service RestFul com seus clientes Android e cliente JafaFX



Fonte: Própria

6.11 TESTAR

Durante o desenvolvimento foram aplicados testes de unidade (SAUVE, 2017). O propósito destes testes visa diminuir os defeitos, garantir a qualidade do software e o andamento da aplicação. São testes efetuados pelo próprio programador.

Ao término do desenvolvimento da aplicação foram realizados testes beta, conforme (SAUVE, 2017), na sede do IDAF com a presença de um funcionário com o perfil técnico. Ao final destes testes o mesmo respondeu a um questionário de avaliação. Nesta etapa foram testadas as três aplicações, considerando os diferentes perfis de usuários. Na (Figura 20) é exibido o questionário de avaliação, para as principais funcionalidades, respondido ao final do teste.

Figura 20 – Questionário de avaliação respondido pelo funcionário do IDAF

Questionário de avaliação referente ao projeto Sistema de Controle da Raiva e Captura de Morcegos

Sobre as funcionalidades do sistema, marque X na opção que você considera com o estado da funcionalidade.

1. Solicitação de Visita.
   - Muito Bom
   - **Bom**
   - Regular
   - Ruim
   - Péssimo
2. Cadastro de Abrigo
   - **Muito Bom**
   - Bom
   - Regular
   - Ruim
   - Péssimo
3. Cadastro de Visita
   - **Muito Bom**
   - Bom
   - Regular
   - Ruim
   - Péssimo
4. Cadastrar Captura
   - **Muito Bom**
   - Bom
   - Regular
   - Ruim
   - Péssimo
5. Ver Mapa
   - **Muito Bom**
   - Bom
   - Regular
   - Ruim
   - Péssimo
6. Cadastro de Recolhimento de Material
   - **Muito Bom**
   - Bom
   - Regular
   - Ruim
   - Péssimo
7. Agendamento de Visita
   - Muito Bom
   - **Bom**
   - Regular
   - Ruim
   - Péssimo

Fonte: Própria

6.12 IMPLANTAR

Gerar o arquivo de extensão APK (Android Package), instalar o aplicativo e configurar o Web Service RESTful.

A etapa de implantação deste trabalho considerou a geração de dois arquivos de extensão APK (Android Package) para os aplicativos na visão do produtor rural e técnico. Também foi gerado um arquivo de extensão JAR (Java ARchive) para a aplicação JavaFX a ser utilizada pelo usuário de perfil coordenador.

Para o Web Service RESTful foi gerado um arquivo de extensão WAR (Web Application Archive). Este arquivo foi utilizado no servidor web GlassFish (ORACLE, 2017) para

disponibilização dos serviços.

7 CONCLUSÕES GERAIS E TRABALHOS FUTUROS

O presente trabalho proporcionou a elaboração de um sistema multiplataforma voltado ao armazenamento de informações sobre a captura de morcegos em propriedades de criação de herbívoros.

Este sistema multiplataforma é constituído por um Web Service RESTful e três aplicações clientes: um aplicativo Android para o perfil de usuário produtor rural, uma aplicação JavaFX para o perfil de usuário coordenador e outro aplicativo Android para o perfil de usuário técnico.

Foram contempladas funcionalidades que permitem ao produtor rural solicitar visitas ao IDAF, bem como, visualizar a agenda de visitas. O registro de informações sobre visitas realizadas pelos técnicos do IDAF também foi automatizado, sendo possível ainda o armazenamento de localizações geográficas de abrigos, capturas e recolhimentos de material. Conforme respostas do questionário de avaliação, o sistema atendeu aos objetivos propostos.

Como trabalhos futuros verificou-se a possibilidade de implementação de solicitações de visita via plataforma web. Além disso, entende-se que para uma avaliação mais completa do sistema seria necessária uma utilização do mesmo por todos os perfis de usuários, durante um maior período de tempo.

REFERÊNCIAS

ANDROID. Arquitetura da plataforma. In: ____. USA: www.developer.android.com, 2017. Disponível em: <https://developer.android.com/guide/platform/index.html>. Acesso em: 07 jun. 2017.

ANDROID. Arquitetura da plataforma. In: ____. USA: www.developer.android.com, 2017. Disponível em: <https://source.android.com/devices/>. Acesso em: 07 jun. 2017.

ANDROID. Arquitetura da plataforma. In: ____. USA: www.developer.android.com, 2017. Disponível em: <https://developer.android.com/ndk/guides/index.html>. Acesso em: 08 jun. 2017.

ARRUDA, R. C. et al. Desmodus rotundus capture in forest (amazon biome) and mangrove areas in the state of maranhão, brazil: a longitudinal study. *Pesquisa Veterinária Brasileira*, SciELO Brasil, v. 33, n. 5, p. 571–574, 2013.

ASTAH. Free uml tool for non-commercial use. In: ____. USA: http://astah.net/editions/community, 2017. Disponível em: <http://astah.net/student-license-request>. Acesso em: 01 abr. 2017.

BATISTA, H. B. d. C. R.; FRANCO, A. C.; ROEHE, P. M. Raiva: uma breve revisão. *Acta scientiae veterinariae. Porto Alegre, RS. Vol. 35, n. 2 (2007), p. 125-144*, 2007.

BRASIL, P. Conheça os principais sintomas da raiva. In: ____. BRA: http://www.brasil.gov.br, 2017. Disponível em: <http://www.brasil.gov.br/saude/2014/10/conheca-os-principais-sintomas-da-raiva>. Acesso em: 24 abr. 2017.

BREDT, A.; ARAUJO, F. A. A.; JUNIOR, J. C. Morcegos em áreas urbanas e rurais: manual de manejo e controle. In: *Morcegos em áreas urbanas e rurais: manual de manejo e controle*. [S.l.]: Fundação Nacional de Saúde, 1996.

CAMPO, A. do. Raiva doença infectocontagiosa em bovinos. In: ____. BRA: http://www.clubeamigosdocampo.com.br, 2017. Disponível em: <http://www.clubeamigosdocampo.com.br/artigo/raiva-doenca-infectocontagiosa-em-bovinos-1336>. Acesso em: 24 abr. 2017.

CHEN, N. et al. Resource oriented architecture for heterogeneous geo-processing workflow integration. In: IEEE. *Geoinformatics, 2009 17th International Conference on*. [S.l.], 2009. p. 1–5.

CORRÊA, S. H. R.; PASSOS, E. d. C. Wild animals and public health. *Biology, Medicine, and Surgery of South American Wild Animals*, Wiley Online Library, p. 493–499, 2001.

DUBROY, P. Memory management for android apps. In: *Google I/O Development Conference*. [S.l.: s.n.], 2011.

EHRINGER, D. The dalvik virtual machine architecture. *Techn. report (March 2010)*, v. 4, p. 8, 2010.

EMBRAPA. Doenças que afetam o sistema nervoso central. In: ____. BRA: http://www.cnpgl.embrapa.br, 2017. Disponível em: <http://www.cnpgl.embrapa.br/sistemaproducao/book/export/html/382>. Acesso em: 10 mai. 2017.

EULALIO, A. D.; CORDEIRO, D.; SOUZA, R. de. Web services: Integração de sistemas orientado a serviços com uma proposta de aplicação na ead. *Revista de Informática Aplicada*, v. 12, n. 2, 2017.

FELT, A. P. et al. Android permissions demystified. In: ACM. *Proceedings of the 18th ACM conference on Computer and communications security*. [S.l.], 2011. p. 627–638.

HOLLA, S.; KATTI, M. M. Android based mobile application development and its security. *International Journal of Computer Trends and Technology*, v. 3, n. 3, p. 486–490, 2012.

ICMBIO. Doenças transmitidas por cães e gatos. In: ____. BRA: Ministerio do Meio Ambiente, 2012. Disponível em: <http://www.icmbio.gov.br/sisbio>. Acesso em: 28 mar. 2017.

IFES-PDS-LES. Processo de desenvolvimento de software da disciplina de laboratório de engenharia de software. In: ____. BRA: ftp://ftp.ci.ifes.edu.br, 2017. Disponível em: <ftp://ftp.ci.ifes.edu.br/informatica/mesquita/LES/IFES-LES-Processo_01.zip>. Acesso em: 01 jun. 2017.

JAVAFX. In: TECNOLOGIA JAVAFX. EUA: JAVAFX, 2014. Disponível em: <http://docs.oracle.com/javafx/2/fxml_get_started/jfxpub-fxml_get_started.pdf>. Acesso em: 25 set. 2016.

JSON. In: JSON.ORG. EUA: Json.org, 2016. Disponível em: <http://json.org/json-pt.html>. Acesso em: 02 out. 2016.

KASPERBAUER, M. Android: Consumindo web services. In: ____. BRA: http://www.devmedia.com.br, 2015. Disponível em: <http://www.devmedia.com.br/android-consumindo-web-services-revista-mobile-magazine-37/22040>. Acesso em: 28 mar. 2017.

KOTAMRAJU, J. Web service json. In: ____. *JSON*. Json: Oracle, 2014. Disponível em: <http://www.oracle.com/technetwork/pt/articles/java/api-java-para-json-2251326-ptb.html>. Acesso em: 01 out. 2016.

LECHETA, R. R. *Google Android-3ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. [S.l.]: Novatec Editora, 2013.

MAIA, C.; NOGUEIRA, L. M.; PINHO, L. M. Evaluating android os for embedded real-time systems. In: *6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*. [S.l.: s.n.], 2010. p. 63–70.

MAPS, A. G. Google maps android api. In: ____. USA: https://developers.google.com/maps/?hl=pt-br, 2017. Disponível em: <https://developers.google.com/maps/android/?hl=pt-br>. Acesso em: 05 mai. 2017.

MARQUEZINI, A. de S.; AGUINALDO, E. C.; ALMEIDA, O. C. P. Desenvolvimento de aplicação de ponto de venda usando javafx. *Tekhne e Logos*, v. 4, n. 3, p. 111–126, 2013.

MENÉNDEZ, A. I. M. Uma ferramenta de apoio ao desenvolvimento de web services. 2002.

MOURA, E. J. R.; CHEIRAN, J. F. P. Acessibilidade em jogos: Diretrizes para promoção da acessibilidade em jogos para dispositivos móveis. *Anais do Salão Internacional de Ensino, Pesquisa e Extensão*, v. 7, n. 2, 2016.

MURPHY, F. A. et al. *Veterinary virology*. [S.l.]: Academic press, 1999.

MYSQL. Mysql downloads. In: ____. USA: https://dev.mysql.com/, 2017. Disponível em: <https://dev.mysql.com/downloads/installer/>. Acesso em: 05 abr. 2017.

OMS. Raiva. In: ____. EUA: Organização Mundial da Saúde, 2010. Disponível em: <http://www.who.int/rabies/en/>. Acesso em: 23 mar. 2017.

ORACLE. Javafx overview. In: ____. *TECNOLOGIA JAVAFX*. EUA: Oracle, 2014. Disponível em: <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>. Acesso em: 28 set. 2016.

ORACLE. Oracle glassfish server. In: ____. BRA: http://www.oracle.com, 2017. Disponível em: <http://www.oracle.com/technetwork/pt/middleware/glassfish/overview/index.html>. Acesso em: 15 mai. 2017.

PEREIRA, F. M. Q. Uma breve introdução à plataforma android. In: ____. BRA: http://dcc.ufmg.br/dcc/?q=pt-br, 2013. Disponível em: <http://homepages.dcc.ufmg.br/~fernando/classes/android/android.html>. Acesso em: 28 mar. 2017.

PEREIRA, L. C. O.; SILVA, M. L. da. *Android para desenvolvedores*. [S.l.]: Brasport, 2009.

PERNAMBUCO, G. do Estado de. Raiva. In: ____. BRA: http://www.adagro.pe.gov.br, 2017. Disponível em: <http://www.adagro.pe.gov.br/web/adagro/raiva>. Acesso em: 15 mai. 2017.

PHPMYADMIN. Trazendo o mysql para a web. In: ____. USA: https://www.phpmyadmin.net, 2017. Disponível em: <https://www.phpmyadmin.net/downloads/>. Acesso em: 05 abr. 2017.

PREMKUMAR, L.; MOHAN, P. *Beginning JavaFX*. [S.l.]: Springer, 2010.

RODRIGUEZ, A. Restful web services: The basics. *IBM developerWorks*, 2008.

RUBY, L. R. S.; RICHARDSON, L. *RESTful web services*. [S.l.]: Copyright, 2007.

SAÚDE, M. da. Guia de vigilância epidemiológica. In: ____. *Conheça os principais sintomas da Raiva*. BRA: Secretaria de Vigilância em Saúde, 2007. Disponível em: <http://www.medicinanet.com.br/conteudos/conteudo/2185/raiva.htm>. Acesso em: 25 mar. 2017.

SAÚDE, M. da Saúde e Secretaria de Vigilância em. *Guia de vigilância epidemiológica*. [S.l.]: Ministério da Saúde Brasília, 2009.

SAUVE, J. Testes de unidade. In: ____. BRA: https://developers.google.com/maps/?hl=pt-br, 2017. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/apoo/html/impl/impl3.htm>. Acesso em: 05 oct. 2017.

TIOBE. Tiobe programming community index. In: ____. USA: http://www.tiobe.com, 2017. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 12 jun. 2017.

UNESP. Doenças transmitidas por cães e gatos. In: ____. BRA: UNESP, 2014. Disponível em: <https://moodle.unesp.br/ava/mod/wiki/viewversion.php?pageid=380&versionid=739>. Acesso em: 28 mar. 2017.

WITT, M. E. F. A. A. *Guia de Manejo e Controle de Morcegos Técnicas de identificação, captura e coleta*. [S.l.]: CEVS/RS, 2012.

WOMACK. Google says 700,00 applications available for android. In: ____. USA: http://www.businessweek.com, 2017. Disponível em: <http://www.businessweek.com/news/2012-10-29/google-says-700-000-applications-available-for-android-devices>. Acesso em: 26 abr. 2017.