UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

Agentes autônomos na simulação da biologia do Aedes aegypti

Pablo Chitolina

Passo Fundo

2018

UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

AGENTES AUTÔNOMOS NA SIMULAÇÃO DA BIOLOGIA DO AEDES AEGYPTI

Pablo Chitolina

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.

Orientador: Prof. Dr. José Maurício Cunha Fernandes

Coorientador: Prof. Dr. Willingthon Pavan

Passo Fundo

2018

CIP – Catalogação na Publicação

C543a Chitolina, Pablo
Agentes autônomos na simulação da biologia do Aedes Aegypti / Pablo Chitolina. – 2018.
51 f. : il. color. ; 30 cm.

Orientador: José Maurício Cunha Fernandes.
Coorientador: Willingthon Pavan.
Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 2018.

1. Agentes inteligentes (Software). 2. Aedes aegypti. 3. Simulação (Computadores). I. Fernandes, José Maurício Cunha, orientador. II. Pavan, Willingthon, coorientador. III. Título.

CDU: 004.4

Catalogação: Bibliotecário Luís Diego Dias de S. da Silva – CRB 10/2241

UPF Universidade de Passo Fundo | PPGCA Programa de Pós-Graduação em Computação Aplicada
Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

PABLO CHITOLINA

Aos vinte e quatro dias do mês de abril do ano de dois mil e dezoito, às 14 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Agentes autônomos na simulação da biologia do Aedes Aegypti"**, de autoria de Pablo Chitolina, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores José Maurício Cunha Fernandes, Willingthon Pavan, Carlos Amaral Hölbig e Alexandre Tagliari Lazaretti. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. José Maurício Cunha Fernandes - UPF
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Willingthon Pavan - UPF
(Coorientador)

Prof. Dr. Carlos Amaral Hölbig – UPF
(Avaliador Interno)

Prof. Dr. Alexandre Tagliari Lazzaretti – IFSul
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

AGRADECIMENTOS

Aos meus pais, por todos os ensinamentos e por tudo a mim dedicado. Sei que muitas vezes deixaram de realizar seus desejos em detrimento aos meus.

À Karine, minha namorada, que com muito amor e compreensão soube me apoiar de maneira correta nos momentos em que mais precisei.

À minha família como um todo, que de forma muito especial me incentivou dando apoio e suporte a cada dia.

Ao meu Orientador Prof. Dr. José Maurício Cunha Fernandes que me conduziu sabiamente nesta minha caminhada.

Ao meu Coorientador Prof. Dr. Willingthon Pavan pela parceria e apoio fundamentais para a conclusão do trabalho.

À todos os amigos e colegas do Grupo Mosaico que sempre estiveram a disposição para me ajudar na realização de minhas tarefas. Em especial ao amigo Felipe Borella pelas dicas e conversas relacionados ao Mestrado.

Aos colegas de aula que tornavam os dias e noites mais leves e divertidos.

A todos os professores e profissionais da educação que estiveram presente nestes 2 últimos anos dentro do PPGCA.

E a todos que contribuíram de alguma forma a minha formação.

AGENTES AUTÔNOMOS NA SIMULAÇÃO DA BIOLOGIA DO AEDES AEGYPTI

RESUMO

O mosquito *Aedes aegypti* é capaz de transmitir a dengue, além da zika e da chikungunya, doenças conhecidas como arboviroses que causam sérios problemas de saúde pública no Brasil. O mosquito *A. aegypti* é originário da África e, atualmente, se encontra amplamente disseminado por todos os continentes. As maiores populações do mosquito são encontradas em países tropicais, uma vez que a existência de períodos chuvosos acompanhados de altas temperaturas favorecem o desenvolvimento deste inseto. Para atenuar os problemas de saúde pública causados pelo mosquito é necessário implementar políticas que se fundamentam em um melhor entendimento na dinâmica do crescimento da população dos mosquitos em um determinado ambiente. Este trabalho tem como objetivo criar um modelo de simulação baseado em Agentes para simular a proliferação do *Aedes aegypti* em um ambiente computacional. Fazendo uso de dados de previsão climática, o modelo será capaz de prever diariamente as possíveis variações populacionais do mosquito para um determinada área. Os resultados serão distribuídos em um gráfico através de uma página web e um aplicativo Mobile. Sendo que o aplicativo também será uma ferramenta onde os usuários poderão enviar e visualizar informações em tempo real sobre os focos identificados na sua cidade.

Palavras-Chave: autônomo, agente, sistema, aedes aegypti.

AUTONOMOUS AGENTS IN AEDES AEGYPTI BIOLOGY SIMULATION

ABSTRACT

The mosquito Aedes aegypti is capable of transmitting dengue, zika and chikungunya, diseases related to arboviruses that cause serious public health problems in Brazil. The *A. aegypti* mosquito originates in Africa and is now widespread in all continents. As the largest mosquito populations are found in tropical countries, the largest mosquito populations are most often found in the ocean. Attention to public health problems by the mosquito is a policy implementation tool that can help improve understanding of the mosquito situation in certain environments. This work aims to create an agent-based simulation model to simulate the proliferation of *Aedes aegypti* in a computational environment. Making use of weather prediction data, the model will be able to predict daily the possible variations of the mosquitoes population for a certain area. The results will be distributed on a chart through a web page and a Mobile application. Since the application will also be a tool where users can send and view information in real time about the outbreaks identified in your city.

Keywords: autonomous, agent, system, mas, aedes aegypti.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

IAD – Inteligência Artificial Distribuída

IA – Inteligência Artificial

ANM – Aedes Na Mira

SMSPF – Secretaria Municipal de Saúde de Passo Fundo

SUMÁRIO

1. INTRODUÇÃO

Adotar a abordagem de Agentes para a resolução de problemas têm sido um assunto muito explorado nas últimas duas décadas. Trabalhos como **Modelos Dinâmicos Acoplados para Simulação da Ecologia do vetor Aedes aegypti** [1] têm apresentado conceitualizações, formalizações, protocolos, técnicas e métodos para aplicação deste tipo de abordagem na concepção de software. Isso tem acontecido pelo fato deste tipo de abordagem possuir algumas características que viabilizam a resolução de problemas de outra forma que não a tradicional, adequando-se à problemas complexos [2] [3] e de natureza descentralizada [4].

Este paradigma adota o conceito de Agente para caracterizar uma unidade autônoma de resolução de problemas. A partir disso, uma solução é criada através do agrupamento de Agentes que trabalham cooperativamente, cada um deles resolvendo parte do problema. A este agrupamento é dado o nome de Sistemas Complexos.

A dengue é um vírus humano transmitido principalmente pelo mosquito *Aedes aegypti*, que é comumente encontrado em casas e locais de trabalho. O ciclo de transmissão da dengue é humano - mosquito - humano. Ou seja, para haver a infecção, o mosquito deve primeiramente picar alguém infectado para então torna-se o hospedeiro que irá transmitir o vírus para outras pessoas que ele picar. Zika e chikungunya possuem o mesmo ciclo de infecção.

Sendo uma doença complexa devido ao amplo espectro de apresentações clínicas, que muitas vezes não são reconhecidas ou até mesmo erroneamente diagnosticada como outras doenças causadoras de febre. Após o período de incubação do vírus, a maioria dos pacientes apresenta um início súbito de febre que pode permanecer por 2 a 7 dias e é frequentemente acompanhada de sintomas como dor de garganta, dores de cabeça e uma erupção cutânea [5]. Devido a isso que a diferenciação da dengue perante outras doenças febris se mostra problemática pois a maioria das as pessoas experimentam um curso clínico autolimitante, que não progride para a forma grave da dengue, a dengue hemorrágica. Em casos graves, devido ao aumento da permeabilidade vascular o quadro pode evoluir para complicações hemorrágicas e de choque [6].

Mais de 40% da população mundial (Fig. 1), em mais de 100 países, estão em áreas de risco de infecção por dengue. As epidemias mais significativas de dengue nos últimos anos ocorreram no sudeste da Ásia, nas Américas e no Pacífico Ocidental. Todo ano, estima-se que ocorram 390 milhões de infecções por dengue no mundo. Deste total, 500.000 casos evoluem para o estágio de febre hemorrágica da dengue, que é uma forma mais grave da doença, resultando em até 25.000 mortes anuais [7].

Figura 1. Regiões ou áreas em risco de infecção por dengue no mundo [8].

A dengue é endêmica no Brasil desde o seu ressurgimento em 1981. Casos normalmente aumentam durante o pico da estação chuvosa, que dura de janeiro a maio. Devido as condições climáticas favoráveis e a falta de eficiência dos programas de combate ao mosquito, o Brasil atingiu níveis alarmantes de proliferação do *Aedes* nos últimos anos [9]. Porém uma grande diminuição foi percebida em 2017 e inicio de 2018. Após um longo período de severa epidemia, estamos virando o jogo e devemos nos esforçar mais do que nunca para erradicar de vez o mosquito.

As taxas de dengue, zika e chikungunya caíram em nosso pais durante a maior parte de 2017, em comparação com o mesmo período de 2016 (Fig 2). Até 11 de novembro, o número de casos suspeitos de dengue registrados em 2017 era de 239.076, em torno de 83,7% menor do que no mesmo período de 2016, segundo o Ministério da Saúde. O número de casos reportados de zika também caiu abruptamente em 2017 com 16.970 casos relatados até meados de novembro de 2017, representando uma queda de 92,1% em relação ao mesmo período de 2016. Os casos suspeitos de chikungunya apresentaram uma queda de 32% até meados de novembro de 2017 em comparação com o mesmo período de 2016 [10].

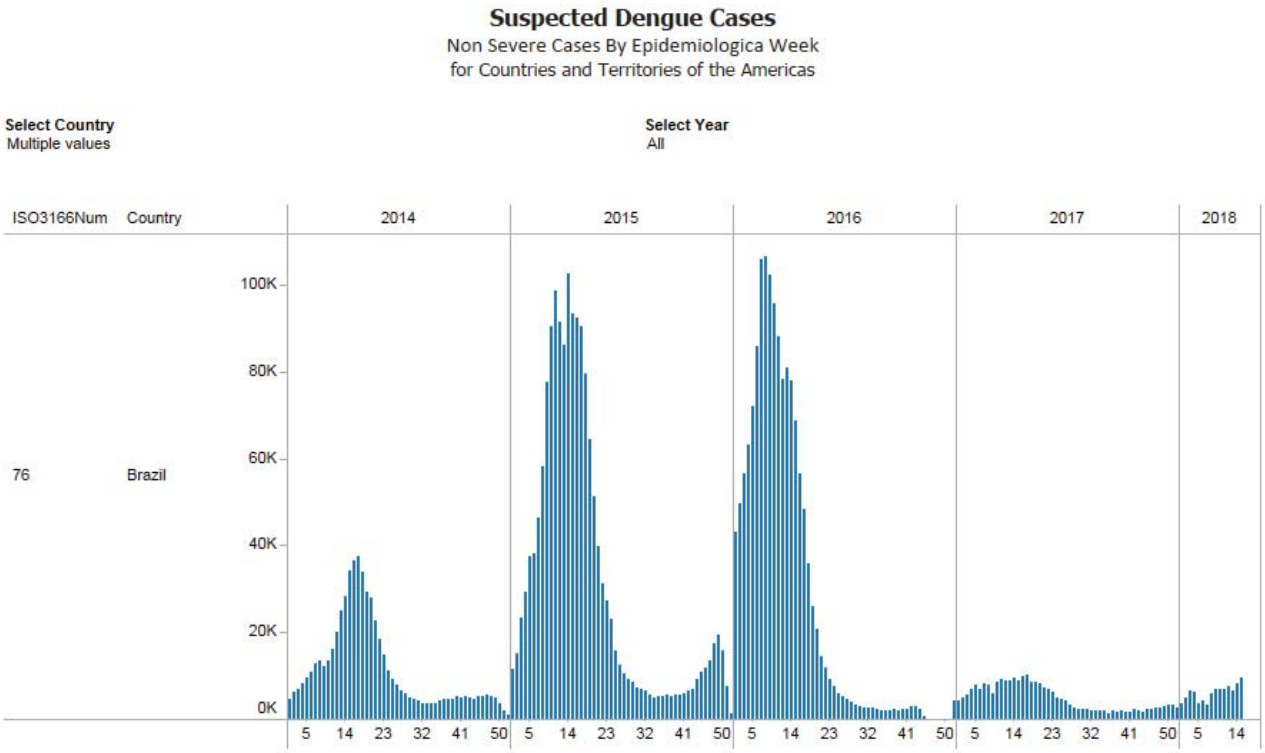

Figura 2. Número de casos de dengue reportado semanalmente no Brasil [11].

A criação de um modelo baseado em Agentes para simular o ciclo de vida desta espécie de inseto aliado com uma ferramenta online de uso público visa fornecer a população uma forma facilitada de apoio ao combate do *Aedes aegypti*. Baseando-se na previsão do tempo de regiões em que o sistema esta implantado, a simulação pode identificar áreas de proliferação em potencial e emitir alertas para que medidas de controle sejam tomadas.

Este trabalho tem como objetivo o desenvolvimento de uma ferramenta de acesso pela população em geral para a prevenção e combate ao *Aedes aegypti*. Dividido em basicamente 3 módulos, **Banco de Dados**, **Servidor** responsável por executar o modelo de simulação criado e que irá disponibilizar uma API para distribuição dos dados e **Front-end** composto por aplicativo mobile e página web. Informações pertinentes serão geradas diariamente e disponibilizadas de forma que qualquer pessoa possa ter acesso aos dados e se tornar um aliado no combate ao *Aedes*.

2. AGENTES AUTÔNOMOS

Os Agentes são aplicados nas mais diversas áreas que variam desde a interação homem-máquina até complexos processos de controle industrial. Devido ao elevado número de aplicações e à relativa abertura do conceito, existem diversas definições sobre o que é um Agente e devido a isso os autores da área ainda não chegaram a um consenso sobre essa matéria. No entanto, podemos simplesmente definir um Agente como sendo um sistema computacional que habita, sente e age em um determinado ambiente e com isso é capaz de realizar um conjunto de objetivos ou tarefas para o qual foi projetado [12].

O crescimento do estudo voltado para o campo dos Agentes está ligado ao fato de ser uma convicção geral de que tais entidades constituem um paradigma de software apropriado ao desenvolvimento de aplicações voltadas à ambientes abertos, heterogêneos e distribuídos como, por exemplo, a simulação do ciclo de vida de insetos. Desta forma, o crescimento do número e dimensão de ambientes com estas características constitui uma forte motivação para o estudo e análise. A adequação dos Agentes a processos de resolução de problemas cuja perspectiva centralizada não demonstra ter capacidade de os resolver satisfatoriamente é outra razão [13].

2.1 AGENTES

O Agente é uma entidade de software que exibe um comportamento autônomo e proativo orientado a objetivos, que está situado em algum ambiente sobre o qual é capaz de realizar ações para alcançar seus próprios objetivos de projeto e a partir do qual percebe alterações [14]. Wooldridge [14] visualiza um Agente como sendo uma entidade com capacidade de resolução de problemas encapsulada. Inserido nesta visão, define o Agente como tendo as seguintes propriedades:

- **autonomia** - executa a maior parte de suas ações sem interferência direta de Agentes humanos ou de outros Agentes computacionais, possuindo controle total sobre suas ações e estado interno;
- **habilidade social** - por impossibilidade de resolução de certos problemas ou por outro tipo de conveniência, interagem com outros Agentes (humanos ou computacionais), para completarem a resolução de seus problemas, ou ainda para auxiliarem uns aos outros. Disto surge a necessidade de que os Agentes tenham capacidade para comunicar seus requisitos aos outros e um mecanismo decisório interno que defina quando e quais interações são apropriadas;
- **capacidade de reação** - percebem e reagem à alterações no ambientes em que estiverem inseridos.

- **capacidade proativa** - Agentes, do tipo deliberativo, além de atuar em resposta às alterações ocorridas em seu ambiente, apresentam um comportamento orientado a objetivos, tomando iniciativas quando julgarem apropriado.

Por assim dizer, Agente é um sistema computacional que se encontra situado num dado ambiente. Este ambiente pode ser uma parte do mundo real (universidade, fábrica, hospital, campo de futebol, etc.), um ambiente simulado ou mesmo um computador. Os Agentes mais vulgares são os Agentes de Software. No entanto, os Agentes podem ter uma existência física (possuindo um corpo), designando-se nesse caso por Agentes Robóticos. Independentemente do tipo de Agente e ambiente, o essencial na definição de Agente que adotamos, é a capacidade do mesmo de se perceber do ambiente e nele agir de forma autônoma. Desta forma, o Agente deve possuir sensores e atuadores apropriados ao seu ambiente e à execução das tarefas para as quais foi projetado. A Figura 3 apresenta um esquema típico de um Agente.

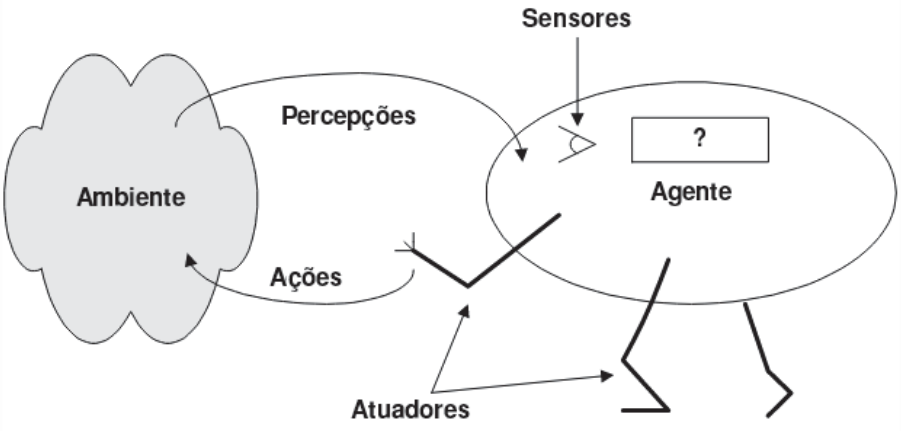

Figura 3. Esquema típico de um Agente.

Um humano interage com seu ambiente através dos seus olhos, ouvidos, olfato, paladar e do tato. E age nesse ambiente utilizando os seus atuadores: braços, pernas, cordas vocais, etc. Os sensores de um Agente robótico podem incluir câmeras, microfones, sensores de proximidade, tato e aceleração, etc. Usualmente os atuadores dos robôs são braços e pernas robóticas, motores e rodas, etc. Nos Agentes de Software é mais difícil definir o que são os sensores e atuadores do Agente. Por exemplo, para um Agente que joga uma partida de xadrez, os sensores permitem determinar a posição das peças no tabuleiro e os atuadores serão capazes de agir realizando jogadas. A definição exata da forma de funcionamento dos sensores e atuadores do Agente pode, no entanto, apresentar diversas alternativas [13].

2.2 ÁREAS CIENTÍFICAS QUE INSPIRARAM OS AGENTES

O campo de estudos designado para os Agentes Autônomos surgiu inspirado nas áreas científicas da IA, Engenharia de Software, Sistemas Distribuídos e Redes de Computadores, Sociologia, Teoria dos Jogos e Economia. A influência destas áreas no campo dos Agentes Autônomos situa-se aos níveis de:

- **IA** – Microaspectos, como a resolução de problemas de raciocínio lógico, representação e utilização de conhecimento, planeamento, aprendizagem, etc.;
- **Engenharia de Software** – O Agente como uma abstração, uma programação orientada por Agentes;
- **Sistemas Distribuídos e Redes de Computadores** – Arquiteturas de Agentes, Sistemas Multiagente, comunicação e coordenação;
- **Sociologia** – Macroaspectos como a formação de sociedades virtuais e a interação entre Agentes;
- **Teoria dos Jogos e Economia**– Negociação, resolução de conflitos e mecanismos de mercado.

Embora a IA tenha exercido inicialmente uma influência muito forte sobre o campo dos Agentes Autônomos e Sistemas Multiagente, verifica-se hoje em dia que este campo já evoluiu, para muito além de poder ser considerado uma sub-área da IA [13].

2.3 APLICABILIDADE DE AGENTES EM SISTEMAS COMPLEXOS

Esta seção apresenta uma justificativa para aplicabilidade da abordagem do processo de desenvolvimento de software vinculada à ideia de Agentes, denominado Engenharia de Software Orientado a Agentes [2], como um paradigma adequado para o desenvolvimento de soluções de software para problemas complexos (Fig. 11), tais como aqueles encontrados em aplicações de simulação populacional.

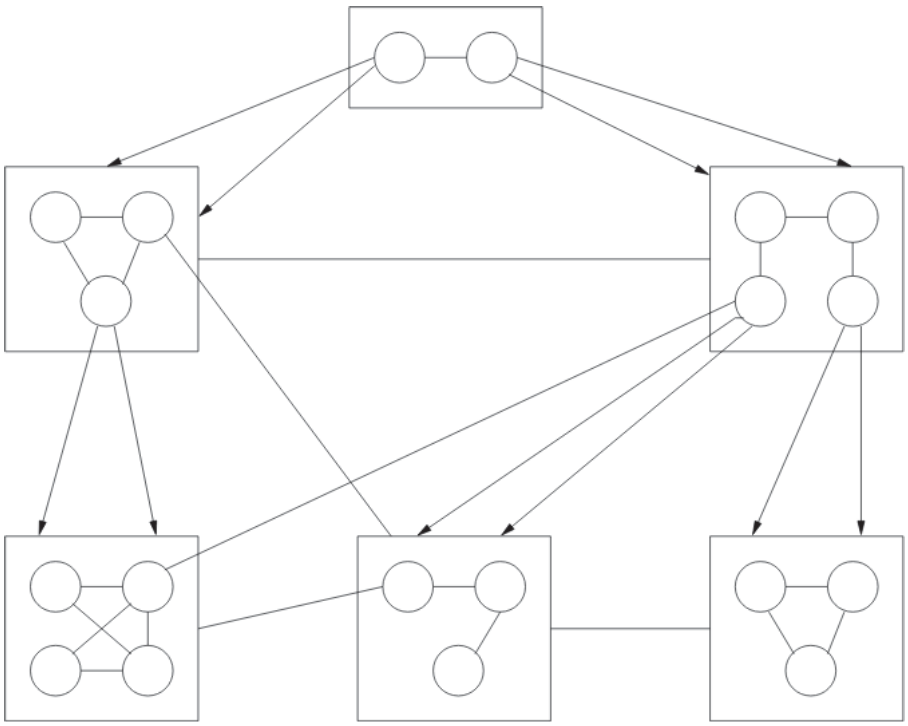

Figura 4. Visão de um Sistema Complexo [2].

- **Abstração** é uma técnica que visa considerar detalhes e propriedades relevantes ao escopo do problema em questão com o objetivo de gerar um modelo simplificado da realidade;
- **Organização** trata de identificar e gerenciar os inter-relacionamentos entre os componentes de resolução do problema.

Considerando a natureza do problema em questão, a maneira através da qual estas ferramentas são implementadas para sua resolução varia entre os diferentes paradigmas de software. Isso nos leva a concluir que a adequação de um dado paradigma à resolução de um dado problema depende da forma pela qual este paradigma implementa estas ferramentas.

Kornfield e Hewitt Apud [15] postulam que a busca cooperativa resulta no fenômeno da "implosão combinatória", que viabiliza a substituição de soluções cuja complexidade resulte numa explosão combinatória, inviabilizando o processo do ponto de vista computacional, por soluções descentralizadas e distribuídas.

2.4 SOFTWARE ORIENTADO A AGENTES

Algumas das características de Agentes são [2]:

- ***entidades de resolução de problemas claramente identificáveis*** com limites e interfaces bem definidos;

- ***entidades situadas em um ambiente em particular*** do qual recebem entradas correspondentes ao estado deste e sobre o qual intervêm através de atuadores;
- ***projetados para realizar um papel específico***, tendo objetivos particulares a atingir;
- ***autônomos***, têm controle sobre seu estado interno e sobre seu próprio comportamento;
- ***capazes de exibir um comportamento flexível para a resolução de problemas***, necessitam ser reativos (responder imediatamente à alterações do ambiente) e proativos (para agir de acordo com seus objetivos).

Ao adotar-se uma visão de mundo orientado a Agentes (Fig. 5), percebe-se que um simples Agente é insuficiente para resolver a maioria dos problemas. Portanto, nestes casos, envolve-se múltiplos Agentes no processo de resolução do problema para que seja representada sua natureza descentralizada, as diversas perspectivas do mundo ou os interesses conflitantes. Além disso, Agentes precisam interagir com outros para atingir seus objetivos individuais ou o acesso aos recursos do ambiente. Duas importantes considerações devem ser feitas:

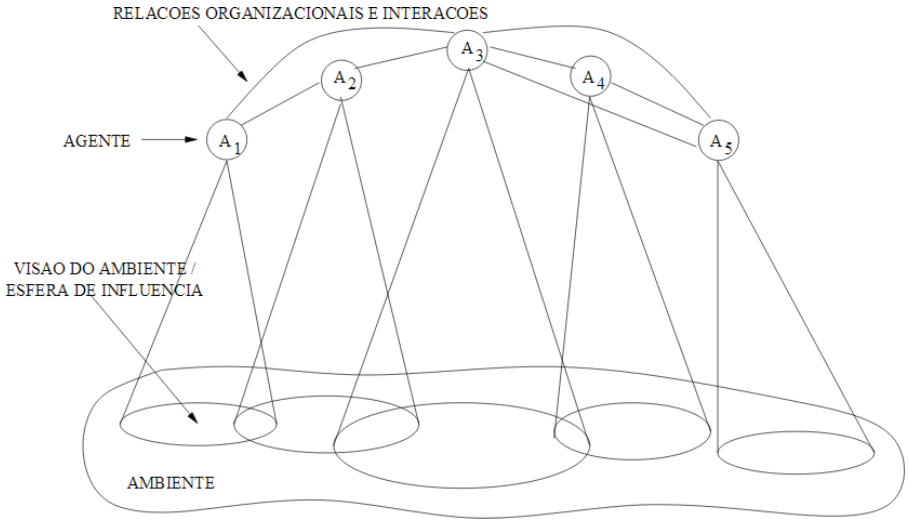

Figura 5. Visão de um Sistema baseado em Agentes [2].

- estas interações ocorrem por meio de uma linguagem de alto nível (declarativa) e, assim sendo, geralmente são conduzidas ao nível de conhecimento;
- Agentes só operam em um ambiente sobre o qual têm controle parcial.

Entende-se que interação social entre Agentes significa a possibilidade de evolução dos relacionamentos existentes e criação de novos relacionamentos.

Analisando-se os pontos acima abordados de maneira conjunta, é possível constatar que aplicar uma abordagem orientada a Agentes para a resolução de um problema significa decompô-lo em múltiplos componentes autônomos com objetivos particulares e que se inter-relacionam. Com isso, podemos enumerar as três palavras-chave desta abordagem: Agentes, interações e organizações.

2.5 APLICAÇÃO DA ABORDAGEM ORIENTADA A AGENTES

As técnicas que adotam uma abordagem orientada a Agente são bem adaptadas para o desenvolvimento de sistemas complexos pelas seguintes razões [2]:

- as ***decomposições*** da orientação a Agentes são um caminho efetivo para particionar a problemática de um sistema complexo;
- as ***abstrações*** da orientação a Agentes são uma abordagem natural para modelar sistemas complexos;
- e, a filosofia orientada a Agentes para identificar e gerenciar relacionamentos organizacionais é apropriada para a representação das dependências e interações que existem em um sistema complexo.

3. BIOLOGIA DO *AEDES AEGYPTI*

Aedes aegypti é um mosquito vetor de importância médica. Esta espécie tem uma distribuição cosmotropical entre as latitudes 20°S e 30°N e é encontrada na maioria das regiões tropicais a subtropicais do mundo, onde exibe uma preferência por habitats humanos com água parada. *Aedes aegypti* adultos são um mosquito de tamanho médio com aproximadamente 4 a 7 mm de comprimento. Os adultos têm escamas brancas na superfície dorsal do tórax e um abdômen marrom-escuro a preto que pode apresentar escamas brancas. Os segmentos tarsais das patas traseiras têm bandas basais brancas que formam o que parecem ser listras [16].

3.1 CICLO DE VIDA

O ciclo de vida do mosquito começa com uma fêmea adulta depositando ovos em ambientes propiciosos. Insetos aquáticos e imaturos chamados larvas emergem e se desenvolvem através de quatro mudas que crescem até a muda final, atingindo assim a fase de pupa não alimentar. Dentro desta pupa, o mosquito adulto se desenvolve (macho ou fêmea) e o mosquito adulto emerge. Os mosquitos adultos se alimentam, acasalam e a fêmea desenvolve ovos onde completa o ciclo dando início a próxima geração (Fig. 6).

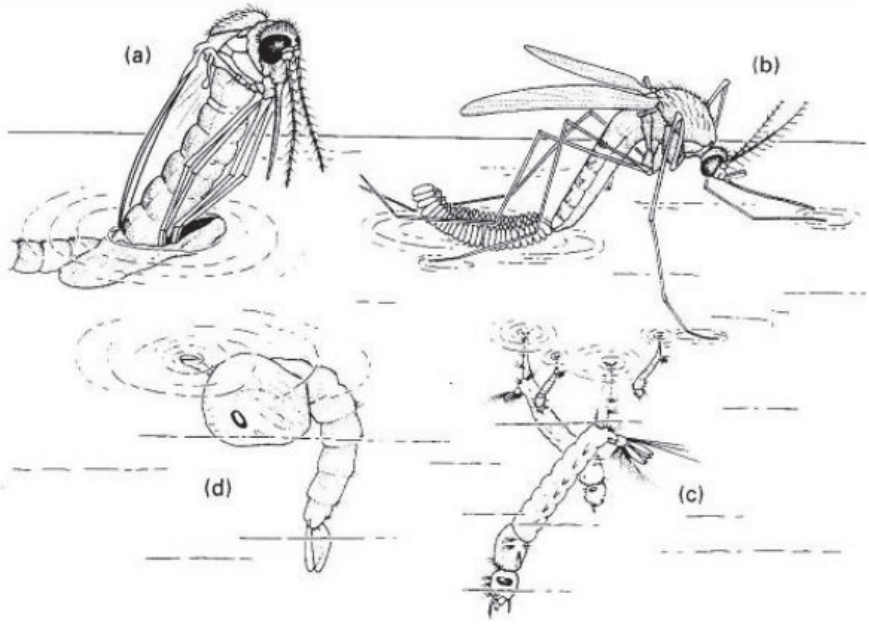

Figura 6. Ciclo de vida do mosquito. a) Adulto emergente. b) Oviposição da fêmea adulta. c) Estágios larvais. d) Pupa [17].

3.1.1 Ovo

Após uma alimentação baseado em sangue, as fêmeas do *Aedes aegypti* produzem em média 100 a 200 ovos (Fig. 7) por oviposição, mas esse número pode variar dependendo de quanto sangue foi ingerido. As fêmeas podem produzir até cinco oviposições durante toda a vida [18]. Os ovos são colocados individualmente em superfícies úmidas em áreas com probabilidade de inundação temporária, como buracos de árvores e recipientes feitos pelo homem.

Figura 7. Ovo do *Aedes aegypti* [19].

Nem todos os ovos são colocados de uma só vez, mas podem ser espalhados por horas ou dias, dependendo da disponibilidade de substratos adequados [20]. Na maioria das vezes, os ovos serão colocados a distâncias variáveis acima da linha d'água, e uma fêmea não colocará a todos os ovos em um único local, mas espalhará os ovos em dois ou mais locais [21].

3.1.2 Larva

A larva do Aedes aegypti (Fig. 8) respira oxigênio através de um sifão que é mantido acima da superfície da água, enquanto o resto do corpo fica suspenso verticalmente. A maioria das larvas do *Aedes* podem ser distinguidas de outros gêneros a olho nu pelo seu sifão curto [18].

Figura 8. Larva do *Aedes aegypti* [19].

As larvas se alimentam de partículas orgânicas na água, como algas e outros organismos microscópicos. A maior parte do estágio larval do *Aedes aegypti* é gasta na superfície da água, embora eles nadem até o fundo do recipiente se forem perturbadas ou quando estiverem se alimentando [18].

As larvas são encontradas frequentemente em casa em poças, pneus ou dentro de qualquer objeto que contenha água. O desenvolvimento das larvas depende da temperatura. As larvas passam por quatro fases, passando um curto período de tempo nas 3 primeiras e até três dias na quarta fase. Se a temperatura for baixa, o *Aedes aegypti* pode permanecer no estágio larval por meses, desde que o suprimento de água e alimento seja suficiente [21].

3.1.3 Pupa

Após o quarto estágio, o *Aedes aegypti* vira pupa (Fig. 9). As pupas dos mosquitos são diferentes de muitos outros insetos holometábolos, pois as pupas são móveis e respondem a estímulos. As pupas, não se alimentam e demoram aproximadamente dois dias para se desenvolver. Os adultos emergem ingerindo ar para expandir o abdômen, abrindo assim o casulo da pupa e emergindo de cabeça.

Figura 9. Pupa do *Aedes aegypti* [19].

3.1.4 Adulto

Depois de emergir da pupa, o mosquito adulto (Fig. 10) descansa na superfície da água por um curto período de tempo, permitindo que suas asas e corpo sequem, antes de voar em busca de um companheiro. Em geral, os machos se desenvolvem mais rapidamente do que as fêmeas e são geralmente os primeiros a emergir da larva.

A fêmea adulta, inicialmente, procura uma alimentação a base de néctar ou sucos de plantas afim de reabastecer as reservas de energia e, em seguida, acasala com um macho, geralmente perto um local de reprodução ao anoitecer. Mosquitos fêmeas acasalam apenas uma vez, sendo o suficiente para fertilizar todos lotes de ovos que ela subsequentemente produz.

Figura 10. Mosquito adulto do *Aedes aegypti* [19].

Para a produção de ovos, as fêmeas do mosquito necessitam de proteína através de uma refeição a base de sangue. Após se alimentar a fêmea procura refúgio isolado onde possa descansar sem ser perturbada para digerir a refeição de sangue e desenvolver um lote de ovos. Ela então voa em busca de refeições de sangue adicionais para repetir este processo.

4. TECNOLOGIAS UTILIZADAS NO AEDES NA MIRA

4.1 MONGODB

O MongoDB é um banco de dados NoSQL de documentos de código aberto projetado para facilidade de desenvolvimento e dimensionamento. Um registro no MongoDB é um documento, que é uma estrutura de dados composta de pares de campo e valor. Os documentos MongoDB são semelhantes aos objetos JSON. Os valores dos campos podem incluir outros documentos, arrays e arrays de documentos.

```
{
  "message": "success",
  "user": {
    "_id": "58375f8fd37c137a33ec5c1c",
    "email": "pablochitolina@gmail.com",
    "nome": "Pablo",
    "sobrenome": "Chitolina",
    "senha": "$2a$05$ppMLO6lIn.taAAhthMCx/eQ2jYabvkwzj1HKYrKbBf.fwNR7DhYw2
       ",
    "__v": 0,
    "senhaTemp": null,
    "cidade": "4154",
    "tipouser": null,
    "token": null,
    "ativo": true
  }
}
```

Trecho de código 4.1. Exemplo de um documento JSON.

As vantagens de usar documentos são:

- Os documentos correspondem a tipos de dados nativos em muitas linguagens de programação.
- Documentos incorporados e matrizes reduzem a necessidade de junções que exigem muito processamento.
- Esquema dinâmico suporta fluentemente o polimorfismo [22].

4.1.1 Mongoose

O Mongoose fornece uma solução direta e baseada em esquemas para modelar os dados de aplicativos [23]. Mongoose é uma biblioteca NodeJS que fornece mapeamento de objeto MongoDB semelhante ao ORM com uma interface familiar dentro do NodeJS. Isso significa que o Mongoose traduz os dados do banco de dados para objetos JavaScript para uso no aplicativo [24].

4.2 NODEJS

NodeJS é uma plataforma baseada no JavaScript do Chrome, com construção em tempo de execução, para criar facilmente aplicativos de rede rápidos e escaláveis. O NodeJS usa um modelo de I/O sem bloqueio, orientado a eventos, que o torna leve e eficiente, perfeito para aplicativos em tempo real com uso intensivo de dados que são executados em dispositivos distribuídos [25]. A seguir estão alguns dos recursos importantes que fazem NodeJS uma ótima escolha.

- **Assíncrono** - Todas as APIs da biblioteca NodeJS são assíncronas, ou seja, não bloqueantes. Isso significa essencialmente que um servidor baseado em NodeJS nunca espera por uma API para retornar dados. O servidor move para a próxima API depois de chamá-lo e então um mecanismo de notificação de Eventos do NodeJS ajuda o servidor a obter uma resposta da chamada da API *call*.
- **Muito Rápido** - Construído no motor de JavaScript V8 do Google Chrome, a biblioteca NodeJS é muito rápida na execução de código.
- ***Single Threaded*, mas altamente escalável** - NodeJS usa um único modelo *threaded* com loop de eventos. O mecanismo de evento ajuda o servidor a responder de forma não bloqueada e torna o servidor altamente escalável, diferentemente dos servidores tradicionais que criam threads limitadas para lidar com solicitações. NodeJS usa um único programa threaded e o mesmo programa pode fornecer serviços a um número muito maior de solicitações do que servidores tradicionais como Apache HTTP Server.
- **Sem buffer** - as aplicações NodeJS nunca armazenam quaisquer dados. Essas aplicações simplesmente devolvem os dados em pedaços.
- **Licença** - NodeJS é lançado sob a licença MIT.

4.2.1 Express

Express é um framework para NodeJS. Ele é minimalista, flexível e contém um robusto conjunto de recursos para desenvolver aplicações web, como um sistema de Views intuitivo (MVC), um robusto sistema de roteamento, um executável para geração de aplicações e muito mais [26].

4.2.1.1 Router

O Roteamento refere-se à determinação de como um aplicativo responde a uma solicitação do cliente por um endpoint específico, que é uma URI (ou caminho) é um método de solicitação HTTP específico (GET, POST, e assim por diante).

Cada rota pode ter uma ou mais funções de manipulação, que são executadas quando a rota é correspondida.

A definição de rotas aceita a seguinte estrutura:

```
1 app.METHOD(PATH, HANDLER)
```

Trecho de código 4.2. Exemplo de estrutura Router

Onde:

- **app** é uma instância do Express.
- **METHOD** é um método de solicitação HTTP.
- **PATH** é um caminho no servidor.
- **HANDLER** é a função executada quando a rota é correspondida.

4.2.2 Body-parser

Body-parser extrai a porção do *body* de um fluxo de solicitação de entrada por inteiro e a expõe no **req.body** tornando mais fácil para a interface. Ele fornece um middleware que usa *nodejs/zlib* para descompactar os dados de solicitação de entrada se for compactado e *stream-utils/raw-body* para aguardar o conteúdo completo e bruto do *body* de solicitação antes de analisá-lo.

Depois de ter o conteúdo bruto, o analisador do *body* irá analisá-lo usando uma das quatro estratégias, dependendo do middleware usado:

- ***BodyParser.raw()***: Na verdade o *body* não é analisado, mas apenas seu conteúdo armazenado em buffer é exposto a um novo buffer em **req.body**.

- ***BodyParser.text()***: Lê o buffer como texto sem formatação e expõe a *string* resultante em **req.body**.
- ***BodyParser.urlencoded()***: Analisa o texto como dados codificados por URL e expõe o objeto resultante em **req.body** para comparação;
- ***BodyParser.json()***: Analisa o texto como JSON e expõe o objeto resultante em **req.body**.

Somente depois de definir o **req.body** para o conteúdo desejável é que será chamado o próximo middleware da pilha, que pode acessar os dados de solicitação sem ter que pensar sobre como descompactar e analisar [27].

4.2.3 Bcrypt

Além de incorporar um *salt* para proteção contra ataques do tipo *rainbow table*, bcrypt é uma função adaptativa: após um determinado número de iterações, as mesmas vão se tornando, automaticamente, mais lentas, permanecendo resistente a ataques de busca de força bruta, mesmo com o aumento do poder de computação.

Enquanto bcrypt.js é compatível com o C++, porém ele é escrito em JavaScript puro e, portanto, mais lento (cerca de 2,7 vezes), reduzindo efetivamente o número de iterações que podem ser processados em um período de tempo igual [28]. O comprimento máximo de entrada é de 72 bytes (observe que caracteres codificados em UTF8 usam até 4 bytes) e o comprimento de hashes gerados é de 60 caracteres.

4.2.4 Multer

Multer é um middleware NodeJS para manipulação de *multipart/form-data*, que é usado principalmente para o upload de arquivos. É escrito sobre o *busboy* para maior eficiência [29].

4.2.5 Nodemailer

É um pacote NodeJS utilizado para enviar emails do tipo MIME sob o protocolo SMTP.

4.2.5.1 SMTP

SMTP faz parte da camada de aplicação do protocolo TCP/IP. Usando um processo chamado "*store and forward*", o SMTP move o e-mail escrito para e entre as redes. Ele

trabalha colaborativamente com o *Mail Transfer Agent*(MTA) para enviar o e-mail para a caixa de entrada do destinatário correto.

SMTP explica e direciona o e-mail pertencente ao MTA do computador origem para um MTA em outro ou vários computadores destino. Usando o recurso "*store and forward*" mencionado anteriormente, a mensagem pode se mover em etapas da origem para o destino. Em cada etapa, o SMTP está fazendo seu trabalho [30].

4.2.6 Passport

O Passport é um middleware de autenticação para NodeJS. Extremamente flexível e modular, o Passport pode ser inserido discretamente em qualquer aplicativo da Web baseado em Express [31]. Em aplicações Web modernas podem-se ter várias formas de autenticação. Tradicionalmente usuários se logam fornecendo um usuário e uma senha. Com o crescimento das redes sociais, logar-se com um provedor OAuth como o Facebook ou o Twitter tem se tornado métodos populares de autenticação. Passport reconhece que cada aplicação tem requisitos únicos de autenticação. Mecanismos de autenticação, conhecidos como estratégias, são empacotados como módulos individuais. As aplicações podem escolher qual estratégia empregar sem criar dependências desnecessárias [32]

4.3 JSON

JSON, em seu significado teórico é "*Javascript Object Notation*", do qual nada mais é que o formato mais leve conhecido de transferência/intercâmbio de dados, ele é similar ao XML, e tem a mesma utilidade, mesmo intuito, porém é mais leve, o detalhe é que não necessariamente, apesar do nome, você tem que usá-lo com Javascript. Muitas linguagens hoje em dia dão suporte ao JSON Ele é muito usado para retornar dados vindos de um servidor utilizando requisições AJAX para atualizar dados em tempo real [33].

```
1 var user = {
2     "name": "Pablo",
3     "age" : 26,
4     "hometown" : "Soledade, RS",
5     "gender" : "male"
6 };
```

Trecho de código 4.3. Exemplo de uma estrutura JSON.

4.4 IONIC

A Ionic é uma estrutura de desenvolvimento de aplicações híbridas para dispositivos móveis utilizando HTML5. Aplicativos híbridos são, essencialmente, pequenos sites executados em um *shell* de navegador em um aplicativo que tem acesso à camada de plataforma nativa. Esses aplicativos têm muitos benefícios sobre aplicativos nativos puros, especificamente em termos de suporte de plataforma, velocidade de desenvolvimento e acesso a código de terceiros.

Ionic seria,basicamente, estrutura de interface de usuário *front-end* que lida com todas as interações de aparência e interação de usuário que o aplicativo precisa para ser atraente. Semelhante ao Bootstrap para Web, mas com suporte para uma ampla gama de componentes móveis nativos, animações, lisas e design bonito.

Ao contrário de uma estrutura responsiva, o Ionic vem com elementos de interface do usuário móvel nativo e layouts que seriam obtidos fazendo uso do SDK nativo no iOS ou no Android. Ele também oferece algumas maneiras opinativas mas poderosas para criar aplicativos móveis que fazem uso das estruturas de desenvolvimento HTML5 existentes.

Como o Ionic é um framework HTML5, ele precisa de um *wrapper* nativo como Cordova ou PhoneGap para ser executado como um aplicativo nativo [34].

4.4.1 Cordova

O Apache Cordova é um framework de desenvolvimento móvel de código aberto. Ele permite que você use tecnologias web padrão - HTML5, CSS3 e JavaScript para desenvolvimento multiplataforma como visto na figura 11. Os aplicativos são executados em wrappers segmentados para cada plataforma e dependem de ligações de API compatíveis com padrões para acessar as capacidades de cada dispositivo, como sensores, dados, status da rede, etc.

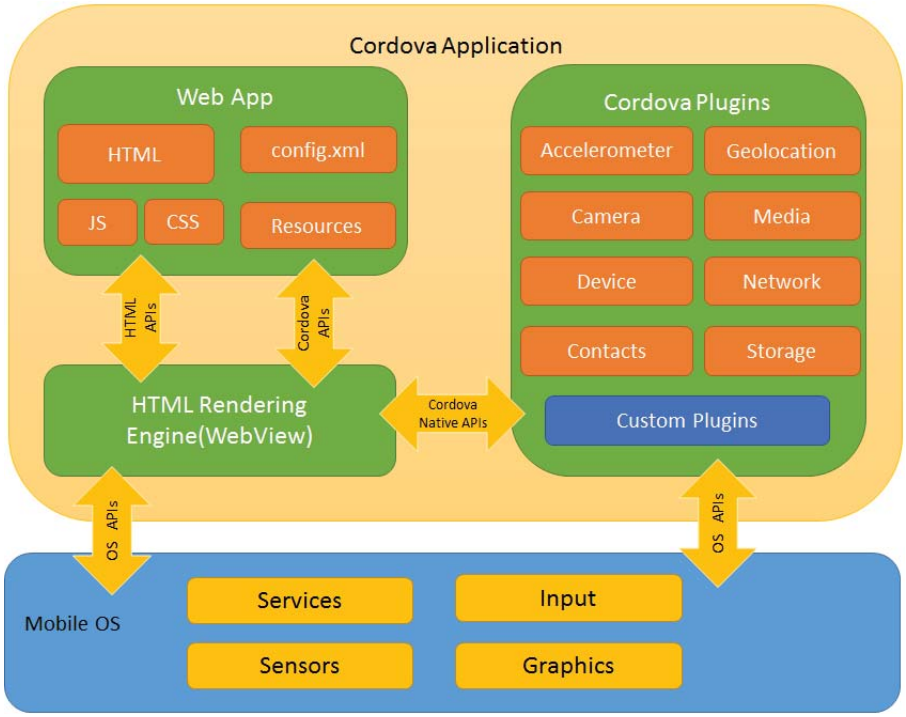

Figura 11. Arquitetura de uma aplicação Cordova [35].

Utilidade do Cordova:

- Desenvolver dispositivos móveis e estender o app criado para mais de uma plataforma, sem precisar reimplementá-lo com a linguagem e o conjunto de ferramentas de cada plataforma.
- Desenvolver web e implantar o aplicativo, que é empacotado, para distribuição em vários portais de lojas de aplicativos.
- Desenvolver em dispositivos móveis com intenção em misturar componentes nativos em um WebView que pode acessar APIs de nível de dispositivo [35].

4.5 ANGULARJS

O AngularJS é um framework estrutural para aplicativos web dinâmicos. Ele permite o uso do HTML como linguagem para criação da estrutura da página e permite estender a sintaxe do HTML para expressar os componentes do aplicativo de forma clara e sucinta. A ligação de dados do Angular e a injeção de dependência, eliminam muito do código que deveria ser escrito. E tudo acontece dentro do navegador, tornando-se um parceiro ideal com qualquer tecnologia de servidor [36].

Angular é o que o HTML teria sido, se tivesse sido projetado para aplicações. O HTML é uma grande linguagem declarativa para documentos estáticos.

A incompatibilidade de impedância entre aplicações dinâmicas e documentos estáticos é muitas vezes resolvida com:

- **Uma biblioteca** - uma coleção de funções que são úteis ao escrever aplicativos da web.
- **Frameworks** - uma implementação particular de um aplicativo da web, onde o código criado pelo desenvolvedor preenche os detalhes [37].

Angular tem outra abordagem. Ele tenta minimizar a incompatibilidade de impedância entre HTML centrado em documentos e o que uma aplicação precisa, através da criação de novas construções do HTML. Angular utiliza uma nova sintaxe do navegador através de uma construção chamada de diretrizes. Exemplos incluem:

- Ligar de dados simplesmente utilizando {{ }};
- DOM com estruturas de controle para repetir, mostrar e esconder fragmentos;
- Suporte para formulários e validação de dados;
- Anexar novo comportamento a elementos DOM, como manipulação de eventos;
- Agrupar HTML em componentes reutilizáveis [38].

4.5.1 Defiant.js

DefiantJS fornece a capacidade para a construção de modelos inteligentes aplicáveis em estruturas JSON, com base em tecnologias comprovadas e padronizadas, como XSLT e XPath.

O DefiantJS também estende o objeto global JSON com o método "*search*", que permite pesquisas em estruturas JSON com expressões XPath e retorna partidas como um objeto semelhante a uma matriz [39].

5. AEDES NA MIRA

Aedes Na Mira é um sistema que tem como objetivo fornecer a população informações sobre o *Aedes aegypti* pertinentes à região na qual elas vivem. Tornando-se assim uma ferramenta de informação em tempo real sobre situação de sua cidade ou região.

O ANM é constituído por um Banco de Dados responsável por centralizar todas a informações geradas e coletadas, uma API para distribuição do dados e execução do modelo e o Front-end composto por um aplicativo mobile e página web.

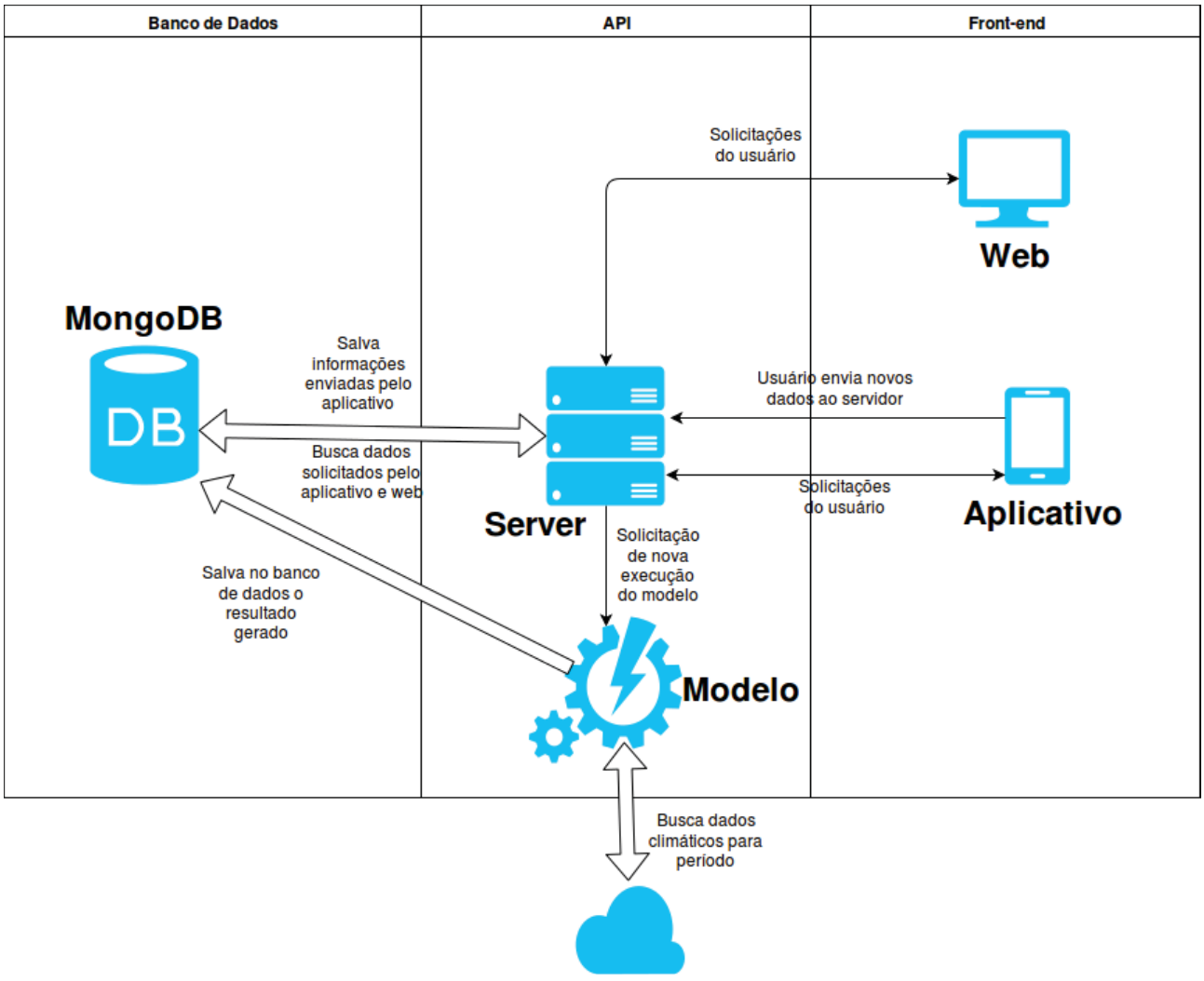

Figura 12. Diagrama demostrando o sistema ANM composto pelos 3 módulos: Banco de Dados, API e Front-end.

O usuário final terá acesso tanto ao aplicativo quanto página web de forma gratuita. Sendo que na plataforma web apenas a leitura de dados seŕa possível, enquanto que com aplicativo, todos os usuários cadastrados poderão enviar dados coletados por eles próprios, os quais se tornam visíveis para qualquer um que esteja acessando o sistema.

Através do aplicativo mobile, os usuários poderão enviar informações como localização e fotos de locais de possíveis criadouros de mosquitos, com isso, a população de determinada região poderá acompanhar em tempo real como esta a situação de sua região. Assim como, posteriormente, o modelo será rodado utilizando essas informações, de modo que o número de pontos informados fará com que diferentes resultados sejam gerados. Onde quanto maior o número de locais de risco, maior vai ser a proliferação do mosquito.

A página web tem como principal função a disponibilização dos dados gerados pelo modelo, assim como, distribuir em tempo real as informações enviadas através do aplicativo. Visando a simplicidade e usabilidade, a página terá como funcionalidade principal a demonstração do gráfico do resultado gerado pelo modelo.

Toda a informação gerada e coletada será centralizada em um único banco de dados. Tal abordagem permite que todo o sistema trabalhe de forma integrada evitando que página web e aplicativo mostrem dados incoerentes. A API criada para coleta e distribuição desses dados oferece total compatibilidade com as ferramentas disponibilizadas ao usuário final.

5.1 MODELO DE SIMULAÇÃO

O ponto central do sistema é o modelo de simulação populacional do *Aedes aegypti* que tem como base um modelo existente escrito na linguagem R [40] e que foi totalmente reescrito em NodeJS. Tal modelo faz uso do conceito de Agentes e tem como objetivo prever possíveis surtos do mosquito. Para isso o modelo tem como *input* dados meteorológicos da região em questão.

O *output* do modelo é a previsão diária da população do *Aedes aegypti* em suas 4 fases: ovo, larva, pupa e adulto. Sendo que diariamente novos indivíduos são criados, envelhecidos e trocados de fase quando necessário e mortos.

A figura 13 representa um esquema com o fluxograma de execução. Toda a simulação está compreendida em 2 funções principais: crescer e morrer. Sendo que a função morrer é responsável por controlar a taxa de mortalidade de ovos, larvas, pupas e adultos.

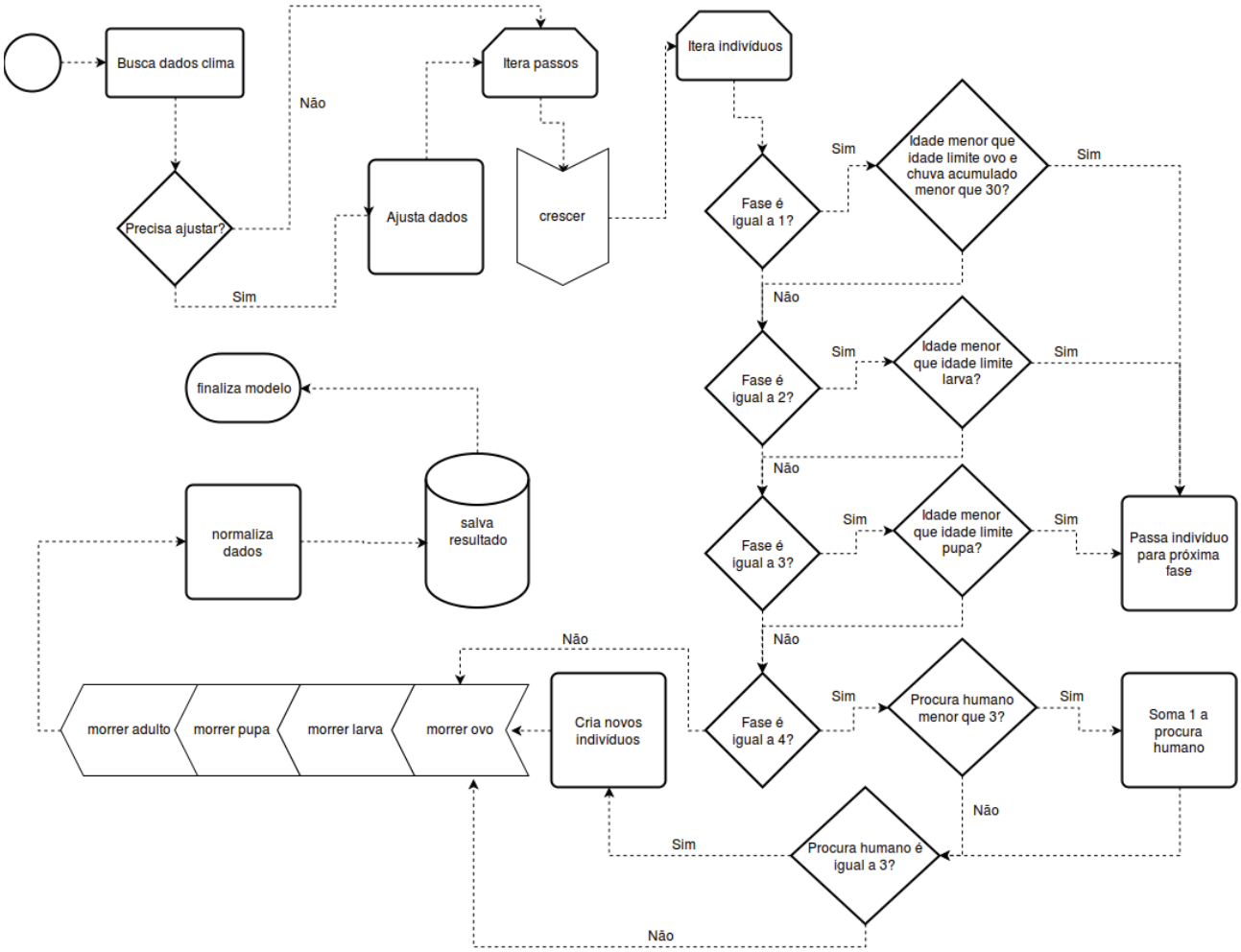

Figura 13. Fluxograma do modelo de simulação.

O modelo será executado automaticamente todos os dias às 00:01 (GMT-3) e após a execução o resultado obtido será gravado no banco e ficará disponível para consulta tanto através do aplicativo mobile quanto página web.

5.1.1 Calibragem do modelo

Inicialmente o modelo irá realizar as simulações apenas na região de Passo Fundo/RS como sendo a fase *beta* do projeto. Tal fase será dividida em 2 etapas.

A primeira etapa será a calibragem inicial, onde o modelo fará uso de dados históricos fornecidos pela SMSPF da incidência mensal do *Aedes aegypti* no município. A figura 14 demonstra estes dados coletados.

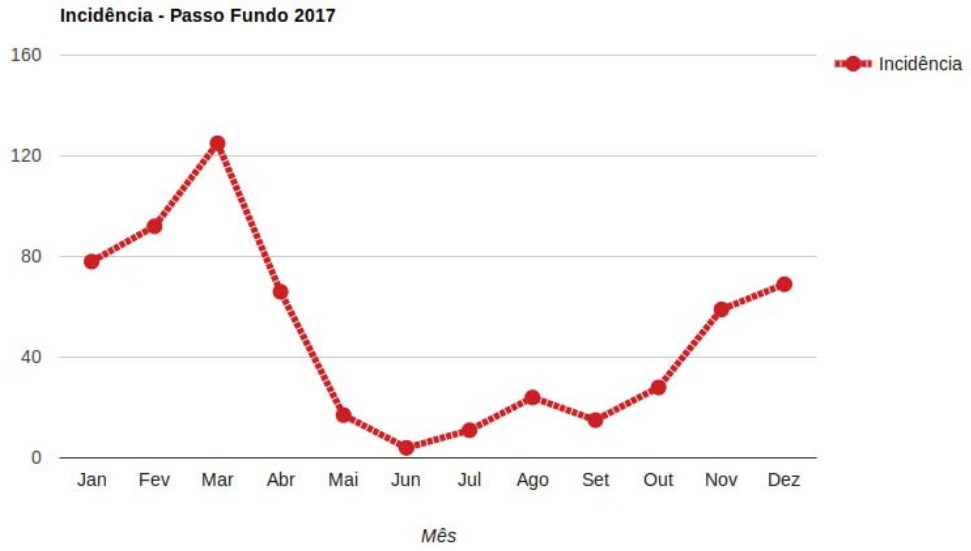

Figura 14. Incidência *Aedes aegypti* em Passo Fundo no ano de 2017.

Fazendo uso de informações climáticas referente ao ano de 2017 para Passo Fundo/RS coletados no sistema INMET [41] e cruzando estes dados com os registro de incidência fornecidos pela SMSPF, podemos perceber que os meses de maior incidência são os que apresentaram os maiores índices de precipitação e temperatura, atingindo o pico máximo de incidência no mês de Março (Fig. 15).

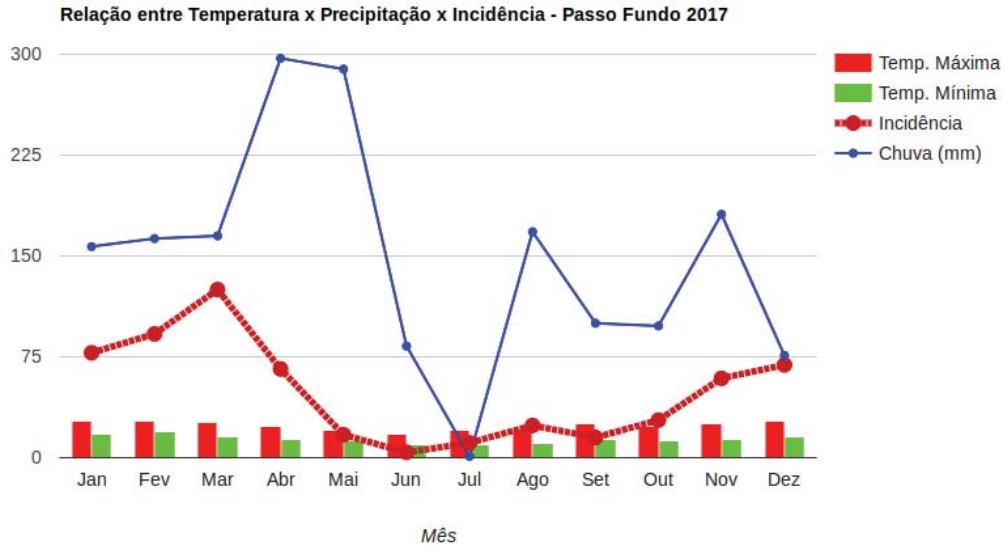

Figura 15. Incidência x Temperatura x Precipitação *Aedes aegypti* em Passo Fundo no ano de 2017.

Com os dados reais em mãos, a calibragem do modelo pôde ser mais assertiva já na primeira etapa. Onde as principais variáveis que impactaram nos resultado foram as de taxa de mortalidade do mosquito em suas 4 fases.

A segunda etapa será a de refinação da calibragem, onde fazendo uso dos dados enviados pelos usuários do aplicativo o modelo será testado diariamente para ter sua

assertividade refinada até o ponto de se mostrar confiável para fornecer resultados para qualquer região que for utilizado.

5.2 API

O lado servidor será compreendido principalmente pelo banco de dados MongoDB, como já descrito, e um Web Service RESTful rodando em NodeJS. No Web Service, os serviços ficarão disponíveis através de URLs específicas para cada tipo de requisição desejada, fazendo uso do protocolo HTTP os métodos mais utilizados serão POST e GET. Uma mesma URL irá disponibilizar variados métodos, basta apenas definir no cabeçalho da requisição qual o desejado. Exemplo: Ao executar o método POST na url: https://www.aedesnamira.com.br/api/user, estamos solicitando ao Web Service que um novo usuário seja criado.

```
$http.post("http://www.aedesnamira.com.br/api/user", {
  nome: "Pablo",
  sobrenome: "Chitolina",
  email: "pablochitolina@gmail.com",
  senha: "1234",
  cidade: "4154"
})
```

Trecho de código 5.1. Código contendo parte do método POST informando os parâmetros necessários para criar um novo usuário no BD.

O retorno obtido será uma mensagem JSON como abaixo.

```
{
  "message": "postUserSuccess"
}
```

Trecho de código 5.2. JSON contendo o resultado obtido com.

Já ao executar o método GET na mesma URL. Onde autenticação do usuário no servidor fica a encargo do Passport que faz uso do Bcrypt para criptografar a senha e diminuir os riscos de roubo de informações.

```
var auth = "Basic " + $base64.encode(email + ":" + senha);
$http.defaults.headers.common.Authorization = auth;
$http.defaults.headers.common["email"] = email;
$http.get("http://www.aedesnamira.com.br/api/user")
```

Trecho de código 5.3. Código contendo parte do metodo GET que solicita que o usuário seja retornando informando seu e-mail e senha.

O retorno obtido será uma mensagem JSON contendo o objeto usuário previamente salvo no banco de dados.

```
{
  "message": "success",
  "user": {
    "__v": 0,
    "_id": "5832e7f2f12ba4ef013a0758",
    "email": "pablochitolina@gmail.com",
    "nome": "Pablo",
    "senha": "$2a$05$qAiD0CPrUksbnSqWQ3xkcO48NI9/NY/HYcYq1AsAphhgRjdZOn0p
       .",
    "sobrenome": "Chitolina",
    "senhaTemp": null,
    "cidade": "4285",
    "tipouser": null,
    "token": null,
    "ativo": true
  }
}
```

Trecho de código 5.4. JSON contendo o resultado obtido ao consultar o usuario.

5.2.1 Banco de dados centralizado

É de extrema importância um local para armazenamento dos dados gerados pelo modelo. Sem tal armazenamento, seria necessário executar a simulação toda vez que fosse solicitado algum resultado, tornando assim inviável o projeto. Tal como o armazenamento dos usuários do sistema e os pontos de risco informados por eles. A figura 16 demonstra a estrutura básica do banco.

Para tal, o banco de dados que mais se encaixa com as ferramentas escolhidas para o desenvolvimento do sistema é o MongoDB. Este DB utiliza o novo conceito em armazenamento de dados NoSQL. Fazer uso de uma ferramenta que facilite a modelagem dos dados no banco e também forneça uma maneira mais simples de retornar as requisições é fundamental. O Mongoose, ao traduzir os dados consultados no banco em um objeto JavaScript, se torna tal ferramenta.

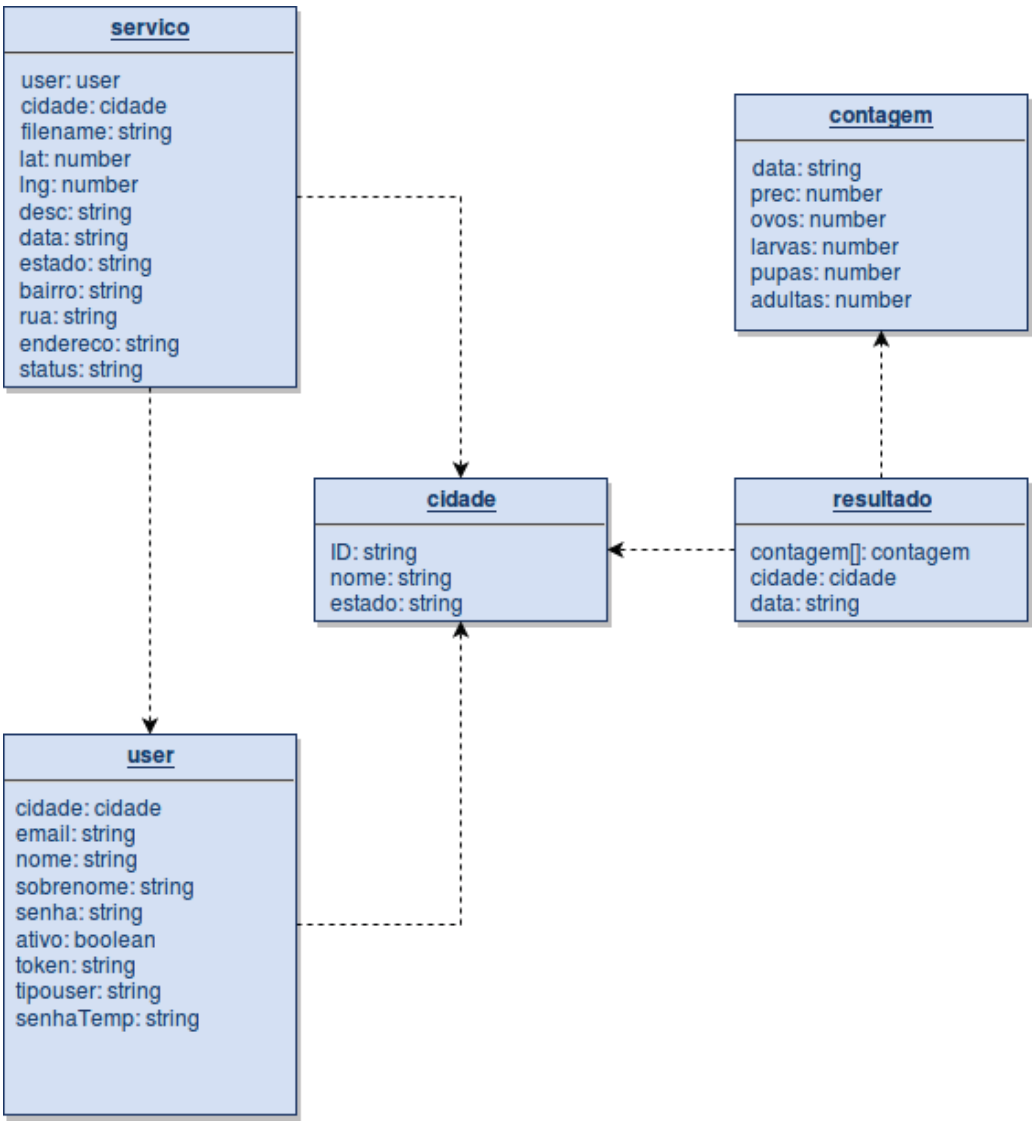

Figura 16. Estrutura Banco de Dados.

5.3 APLICATIVO

Para fazer uso do aplicativo, o usuário deve primeiramente baixar na App Store da Google através do link: https://play.google.com/store/apps/details?id=br.com.droidgo.aedesnamira. Após baixado e instalado é necessário efetuar o cadastro para acessar qualquer parte do app. Com o aplicativo é possível cadastrar um ponto de risco de maneira simples seguindo apenas 2 etapas:

1. **Marcar no mapa o local**, figura 17;

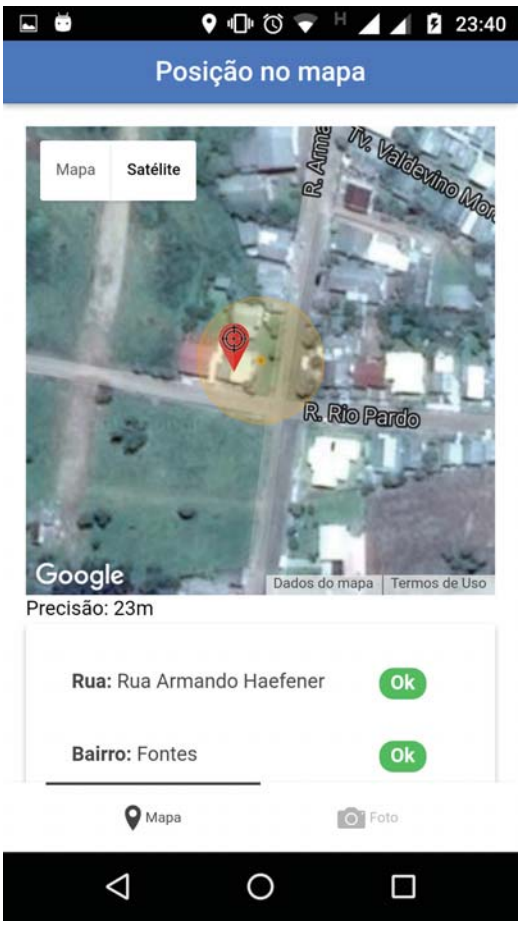

Figura 17. Automaticamente ao clicar em um local o endereço e buscado e preenchido.

2. **Adicionar foto e descrição**, figura 18.

Figura 18. Para concluir o cadastro é necessário tirar uma foto e adicionar uma breve descrição.

Após cadastrado, o chamado será mostrado numa lista na tela inicial do app. Como mostrado na figura 19 Neste ponto os chamados que foram criados através do app por todos

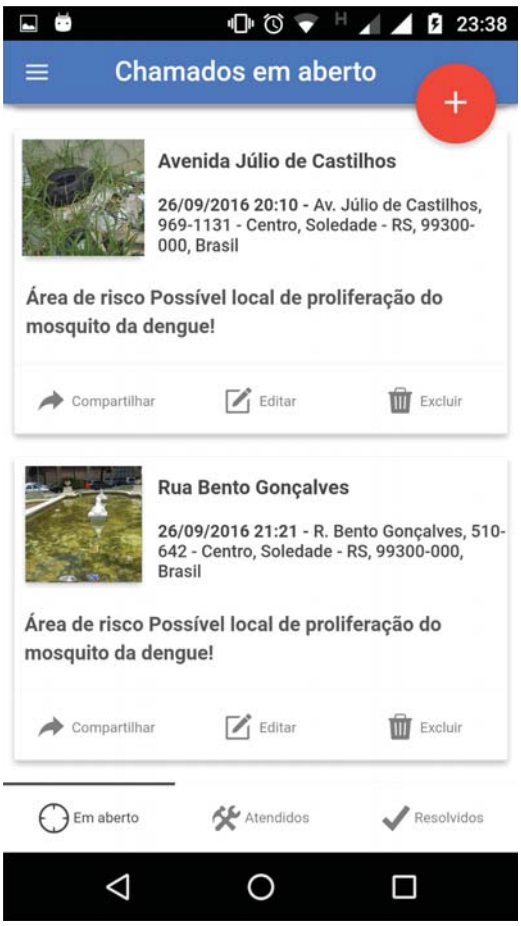

Figura 19. Chamados criados pelo usuário.

os usuários também estarão sendo mostrados na página web.

5.3.1 Coleta de dados

O aplicativo terá como função principal a coleta de dados para uso no modelo e para que os usuários possam identificar locais de infestação.

5.4 PÁGINA WEB

Na página web que se encontrará a maior parte das informações e detalhamentos dos resultados gerados pelo modelo. Ela estará disponível através do link: https://www.aedesnamira.com.br e seŕá de acesso público, onde o usuário não precisará logar-se no sistema para visualizar as informações como é mostrado na figura 20.

Figura 20. Página web de acesso público em https://www.aedesnamira.com.br.

Construída com o uso de AngularJS a página deve fornecer ao usuário uma navegação fluída e com baixo consumo de rede, já que basta apenas 1 carregamento inicial da página para ele ter acesso a todos os módulos disponíveis. Os gráficos serão gerados em tempo real, fazendo uso dos dados previamente recebidos ao carregar a página.

6. TESTES E RESULTADOS

Este capítulo trará uma análise da metodologia usada e os resultados alcançados com o modelo de simulação.

6.1 ANÁLISE DA METODOLOGIA PROPOSTA

Os principais motivos referente a escolha do Ionic como sendo a linguagem de implementação do aplicativo, inicialmente, foi o fato dela oferecer a compilação em multiplataforma (iOS, Android, Windows Phone) e também por mostrar que supriria completamente as necessidades e funcionalidades propostas pelo ANM. Porém além disso, no decorrer do trabalho, o Ionic apresentou uma alta produtividade graças as bibliotecas oferecidas e a grande quantidade de informações e suporte disponibilizados pela comunidade em sites e forums da Internet.

Tanto o lado servidor assim como o modelo de simulação foram desenvolvidos na linguagem NodeJS devido a mesma fazer uso do paradigma orientado a objetos e ser baseada em JavaScript. Com isso foi possível atingir um ótimo nível de estruturação do projeto assim como uma grande eficiência no cálculo do modelo e velocidade na comunicação entre lado cliente e servidor.

O banco de dados MongoDB possui uma completa integração com o NodeJS, devido a isso e também ao fato de ser um banco muito veloz para o tipo de dados trabalhados no projeto, toda a informação gerada pelo sistema será salva num único lugar.

O AngularJS por se tratar de um framework muito utilizado no desenvolvimento Front-end e mostrar-se muito maduro e completo em relação a outros semelhantes, possibilitou o desenvolvimento de uma interface rica e intuitiva. Fazendo uso do Bootstrap, o usuário terá uma experiência limpa durante a utilização da página.

6.2 ANÁLISE DOS RESULTADOS DO MODELO

Os dados de incidência do *Aedes aegypti* disponibilizados pela SMSPF são dados mensais, portanto o modelo teve de sofrer alguns ajustes e modificações para a calibragem e comparação de resultados. Em ambiente de produção o modelo executa simulação populacional do mosquito para 45 dias no futuro. Sendo que cada "passo"do modelo corresponde há 1 dia real. Desta forma, 45 passos simulam 45 dias.

Para que fosse possível analisar e comparar os resultados obtidos com os dados reais, o modelo executou 50 passos, sendo que cada passo representa 1 semana real. O ano tem 52 semanas, mas como os 10 primeiros dias de simulação do modelo são usados

apenas para calcular a precipitação acumulada, restaram 355 dias de histórico do clima, onde o número arredondado de semanas corresponde a 50.

Inicialmente o modelo inicia a execução com 10 indivíduos, sendo 2 na fase de ovo, 3 larvas, 3 pupas e 2 adultos.

A figura 21 demostra o resultado do modelo para a rodada baseada nos dados do clima do ano de 2017 em Passo Fundo.

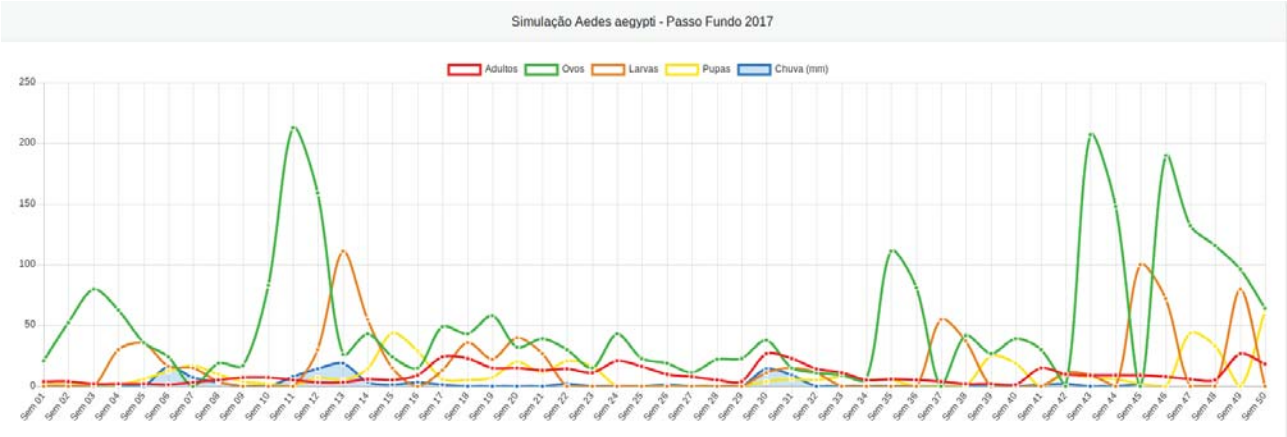

Figura 21. Resultado da simulação populacional do *Aedes aegypti* para o periódo de 50 semanas no ano de 2017 para a cidade de Passo Fundo. Semana 1 com inicio em 11/01/2017.

Pode-se perceber que nas semanas iniciais e finais do ano o número de indivíduos cresce em relação as semanas que representam o inverno em nossa região. Assim como após o aumento no número de ovos ocorre um aumento de larvas e consecutivamente o aumento na contagem de pupas resultando no consecutivo crescimento da população de adultos, a figura 22 demonstra isso.

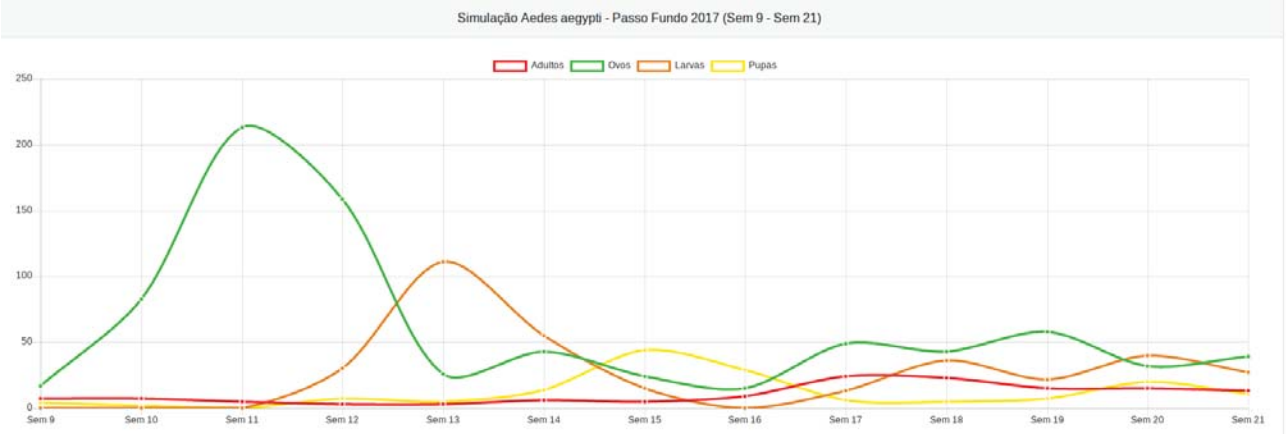

Figura 22. Resultado da simulação populacional do *Aedes aegypti* para o período entre as semanas 9 (15/03/2017) e 21 (07/06/2017) do ano de 2017.

Tais resultados demostram que a dinâmica populacional do inseto na simulação está de acordo com o esperado em um ambiente real.

6.2.1 Comparação dos dados reais com o resultado do modelo

Com o resultado do modelo e os dados disponibilizados pela SMSPF, um gráfico (Fig. 23) para comparação e melhor visualização entre ambiente real e simulado foi gerado.

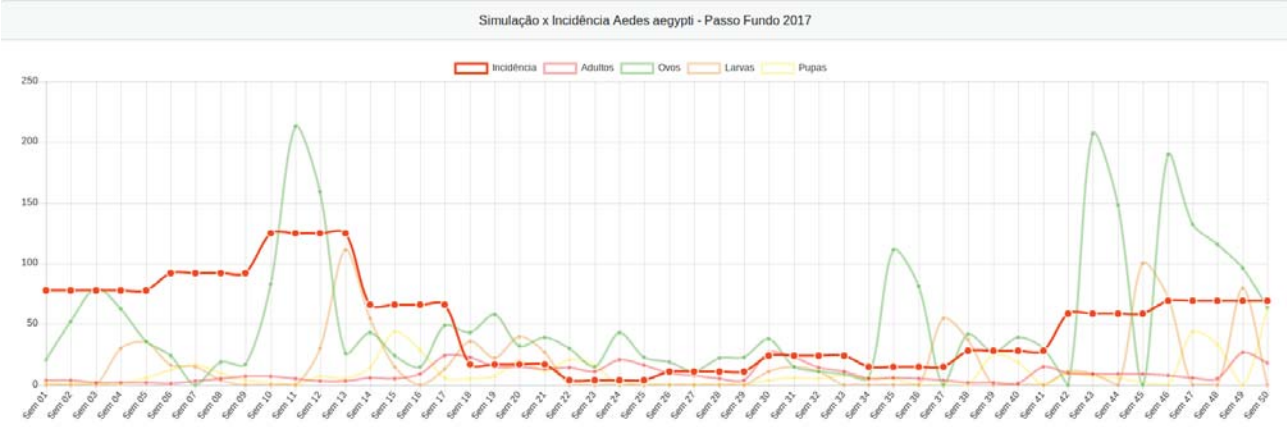

Figura 23. Comparação com os resultados da simulação populacional com os dados fornecidos pela SMSPF.

A linha em vermelho apresenta a incidência real no ano de 2017, enquanto as linhas claras demonstram a população de ovos, larvas, pupas e adultos simulados pelo modelo.

Com os resultados obtidos, o modelo apresentou satisfatória precisão na simulação populacional do *Aedes aegypti*. E portanto se apresenta apto para entrar na fase de calibragem com os enviados diariamente pelo usuários do aplicativo.

7. CONCLUSÕES E TRABALHOS FUTUROS

O ANM esta disponível em sua plataforma web no endereço http://www.aedesnamira.com.br, onde o usuário terá acesso aos resultados e previsões disponibilizadas pelo modelo e também informações sobre a situação atual de seu município, podendo visualizar em tempo real os dados obtidos pelo app.

O aplicativo está disponível em https://play.google.com/store/apps/details?id=br.com.droidgo.aedesnamira, que por meio dele serão feitas as coletas de dados para alimentar o modelo tal como fotos e levantamentos para o conhecimento da população.

O Aedes Na Mira e um sistema que visa ser um aliado tanto da população quanto de órgãos públicos no combate ao *Aedes aegypti*. Com ele, medidas de combate ao mosquito poderão ser tomadas baseadas na previsão do tempo e informações enviadas pelo app. Um exemplo deste combate poderia ser o de um terreno com acumulo de lixo onde foi identificado possíveis locais de focos do mosquito, aliando esta informação com a previsão do modelo, as autoridades de saúde poderiam realizar a limpeza da área antes da chegada da data de surto de proliferação prevista pelos resultados da simulação.

O projeto piloto será implantado inicialmente na cidade de Passo Fundo - RS. Após a validação do modelo e eventuais correções na plataforma, o ANM estará pronto para ser implantado em qualquer cidade que esteja disposta a ter uma ferramenta a mais no combate a dengue e outras doenças.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] LANA, R. M. *Modelos Dinâmicos Acoplados para Simulação da Ecologia do vetor Aedes aegypti*. Tese (Doutorado) — Instituto de Ciências Exatas e Biológicas, Outro Preto, 2009.

[2] JENNINGS, N. R. Agent-oriented software engineering. In: . Heidelberg, Alemanha: [s.n.], 1999. v. 1647, p. 1–7.

[3] JENNINGS, N. R.; WOOLDRIDGE, M. J. Applications of intelligent agents. In: . Heidelberg, Alemanha: [s.n.], 1998. p. 3–28.

[4] MOULIN, B.; CHAIB-DRAA, B. *An Overview of Distributed Artificial Intelligence*. [S.l.]: Foundations of distributed artificial intelligence, 1996.

[5] ORGANIZATION, W. H. *International travel and health*. 2013. Disponível em: <http://www.who.int/ith/diseases/dengue/en/>. Acesso em: Mai. 2018.

[6] MURRAY M. B. QUAM, A. W.-S. N. E. *Epidemiology of dengue: past, present and future prospects*. 2013. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3753061/>. Acesso em: Mai. 2018.

[7] PROGRAM, W. M. *Dengue*. 2016. Disponível em: <http://www.eliminatedengue.com/our-research/dengue-fever>. Acesso em: Mai. 2018.

[8] OMS. *Dengue*. 2014. Disponível em: <https://www.iledefrance.ars.sante.fr/zika-chikungunya-dengue-information-et-recommandations>. Acesso em: Mai. 2018.

[9] BERDJIS, N. *Dengue Epidemic in Brazil*. 2015. Disponível em: <http://www.healthmap.org/site/diseasedaily/article/dengue-epidemic-brazil-51315>. Acesso em: Mai. 2018.

[10] GARDAWORLD. *Brazil: Reported cases of dengue, Zika, and chikungunya down in 2017s*. 2017. Disponível em: <https://www.garda.com/crisis24/news-alerts/82631/brazil-reported-cases-of-dengue-zika-and-chikungunya-down-in-2017>. Acesso em: Mai. 2018.

[11] PAHO. *Reported Cases of Dengue Fever in The Americas*. 2018. Disponível em: <www.paho.org/data/index.php/en/mnu-topics/indicadores-dengue-en/dengue-nacional-en/252-dengue-pais-ano-en.html>. Acesso em: Mai. 2018.

[12] MAES, P. *Intelligent Software: Programs That Can Act Independently Will Ease the Burdens that Computers Put on People*. [S.l.]: IEEE Expert Systems, 1996.

[13] REIS, L. P. *Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico*. Tese (Doutorado) — Universidade do Porto, Porto Alegre, 2003.

[14] WOOLDDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. In: _____. [S.l.: s.n.], 1995. v. 10, n. 2, p. 115–152.

[15] JENNINGS, N. R. et al. Using intelligent agents to manage business processes. In: . Londres: [s.n.], 1996.

[16] SJ, L. W. C. *Aedes aegypti: biology and ecology*. Berkeley, CA: Pan American Health Organization, 1955.

[17] GULLAN, P. S. C. D. J. *The Insects: An Outline of Entomology*. Boston: Wiley-Blackwell, 2004. 528 p.

[18] NELSON, M. J. *Mosquitoes of North America*. Washington, D.C: University of California Press, 1986. 50 p.

[19] NUNES, M. *Super zoom no mosquito da dengue*. 2011. Disponível em: <http://faqbio.blogspot.com.au/2011/06/super-zoon-no-mosquito-da-dengue.html>. Acesso em: Mai. 2018.

[20] CLEMENTS, A. N. London.

[21] MULLEN, L. D. G. *Medical and Veterinary Entomology*. 2. ed. San Diego, CA: Academic Press, 2009. 637 p.

[22] MONGODB. *Introduction to MongoDB*. 2016. Disponível em: <https://docs.mongodb.com/manual/introduction/>. Acesso em: Dez. 2016.

[23] MONGOOSE. *mongoose - Elegant MongoDB object modeling for Node.js*. 2016. Disponível em: <http://mongoosejs.com/>. Acesso em: Dez. 2016.

[24] HERNANDEZ, M. *Node.js and MongoDB - Getting Started with Mongoose*. 2016. Disponível em: <http://blog.modulus.io/getting-started-with-mongoose>. Acesso em: Dez. 2016.

[25] POINT, T. *Node.js - Introduction*. 2016. Disponível em: <https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm/>. Acesso em: Dez. 2016.

[26] ADMIN. *Primeiros passos com Express em Node.js*. 2016. Disponível em: <http://nodebr.com/primeiros-passos-com-express-em-node-js/>. Acesso em: Dez. 2016.

[27] VLASBLOM, A. *What exactly does body-parser do with express.js and why do I need it?* 2016. Disponível em: <https://www.quora.com/What-exactly-does-body-parser-do-with-express-js-and-why-do-I-need-it>. Acesso em: Dez. 2016.

[28] DEPENDENCIES bcrypt.js Optimized bcrypt in JavaScript with zero. *axelpale*. 2016. Disponível em: <https://github.com/dcodeIO/bcrypt.js/blob/master/README.md>. Acesso em: Dez. 2016.

[29] MULTER. *expressjs*. 2016. Disponível em: <https://github.com/expressjs/multer>. Acesso em: Dez. 2016.

[30] ADDRESS, W. I. M. I. *The Mailman Inside Our Computers. Or: What Is Simple Mail Transfer Protocol?* 2016. Disponível em: <http://whatismyipaddress.com/smtp>. Acesso em: Dez. 2016.

[31] PASSPORT. *Simple, unobtrusive authentication for Node.js*. 2016. Disponível em: <http://passportjs.org/>. Acesso em: Dez. 2016.

[32] ADMIN. *Primeiros passos com Passport e Express em Node.js*. 2016. Disponível em: <http://nodebr.com/primeiros-passos-com-passport-e-express-em-node-js/>. Acesso em: Dez. 2016.

[33] LENGSTORF, J. *JSON: What It Is, How It Works, e How to Use It*. 2016. Disponível em: <https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/>. Acesso em: Dez. 2016.

[34] IONIC. *Welcome to Ionic*. 2016. Disponível em: <http://ionicframework.com/docs/guide/preface. Acesso em: Dez. 2016.

[35] CORDOVA. *Architectural Overview*. 2016. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/>. Acesso em: Dez. 2016.

[36] FARREIRA, D. *Criando uma aplicação simples com AngularJS*. 2016. Disponível em: <https://tableless.com.br/criando-uma-aplicacao-simples-com-angularjs/>. Acesso em: Dez. 2016.

[37] ANGULARJS. *0 - Bootstrapping*. 2016. Disponível em: <https://docs.angularjs.org/tutorial/step_00>. Acesso em: Dez. 2016.

[38] ANGULARJS. *What Is Angular?* 2016. Disponível em: <https://docs.angularjs.org/guide/introduction>. Acesso em: Dez. 2016.

[39] HBI99. *DefiantJS*. 2016. Disponível em: <https://github.com/hbi99/defiant.js?files=1>. Acesso em: Dez. 2016.

[40] RINKE, T. P. K. *Effects of temperature and food on individual growth and reproduction of Daphnia and their consequences on the population level*. Tese (Doutorado).

[41] INMET. *Estações Automáticas*. Disponível em: <http://www.inmet.gov.br/portal/index.php?r=est