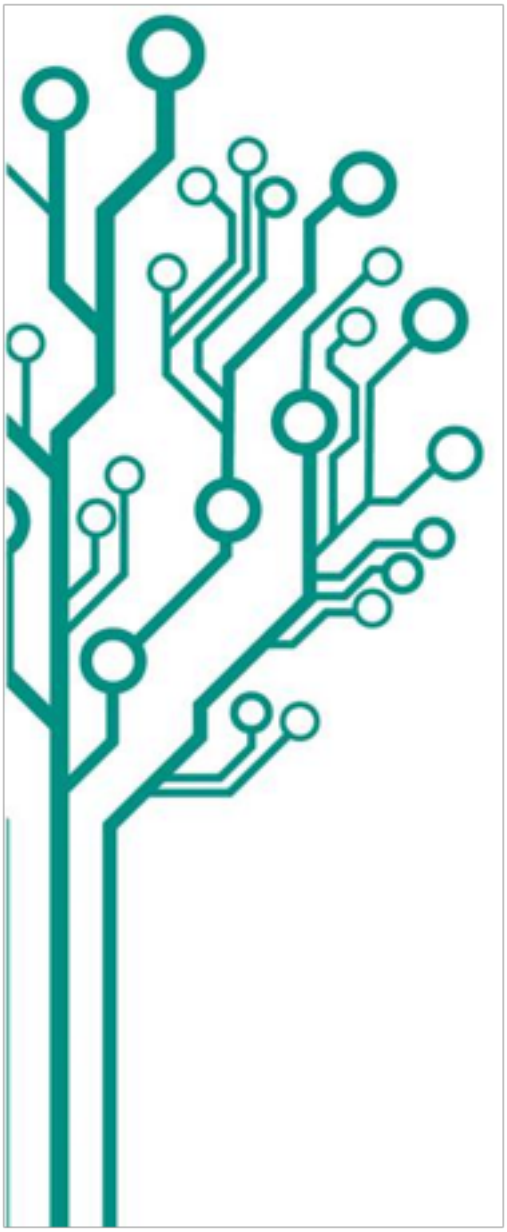
INSTITUTO TECNOLÓGICO VALE

VALE

Mestrado Profissional

Uso Sustentável de Recursos Naturais em Regiões Tropicais



Arquitetura Computacional para Manuseio de Dados de Clima e Movimentação de Abelhas com Etiquetas Eletrônicas

Helder Moreira Arruda

Dissertação apresentada ao Programa de Mestrado Profissional em Uso Sustentável de Recursos Naturais em Regiões Tropicais do Instituto Tecnológico Vale (ITV).

Orientador: Paulo A. de Souza Jr.

Belém-PA

Outubro de 2015

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca do ITV - Belém-PA

A773a

Arruda, Helder Moreira
Arquitetura Computacional para Manuseio de Dados de Clima e Movimentação de Abelhas com Etiquetas Eletrônicas / Helder Moreira Arruda. – Belém-PA, 2015.
107 f. : il. (algumas coloridas) ; 30 cm.

Dissertação (Mestrado) – Instituto Tecnológico Vale, 2015
Orientador: Paulo A. de Souza Jr.

1. Estrutura de Dados. 2. Sistemas de Informação. 3. Microssensores. 4. Abelhas. Título.

CDD 23. ed. 005.73

Helder Moreira Arruda

Arquitetura Computacional para Manuseio de Dados de Clima e Movimentação de Abelhas com Etiquetas Eletrônicas

Dissertação apresentada ao Programa de Mestrado Profissional em Uso Sustentável de Recursos Naturais em Regiões Tropicais do Instituto Tecnológico Vale (ITV).

Trabalho aprovado. Belém-PA, Outubro de 2015:

Prof. Paulo A. de Souza Jr., Dr. rer. nat.
Orientador - CSIRO/ITV

Prof. Everaldo Barreiros de Souza, Ph. D.
Membro interno - ITV

Prof. Gutemberg Hespanha Brasil, Ph. D.
Membro externo - UFES

Belém-PA
Outubro de 2015

Este trabalho é dedicado às abelhas.

Agradecimentos

Ao meu professor e orientador Paulo de Souza, por suas aulas repletas de motivação, pelo tempo, paciência e principalmente pela oportunidade ofertada em um momento difícil.

Ao professor Gustavo Pessin, pela confiança, dedicação e apoio desde a minha entrada no projeto.

Ao professor Cleidson de Souza, por ter me aberto as portas de seu laboratório.

Ao CNPq, pela bolsa de Desenvolvimento Tecnológico e Científico concedida em momento crucial para o desenvolvimento deste trabalho.

Ao ITV e à Vale, por tornar possível a realização do mestrado profissional.

À CSIRO, em especial à equipe do projeto *Swarm Sensing*, pela possibilidade de utilizar as etiquetas eletrônicas neste trabalho.

À Cooperativa de Produtores de Cacau e Mel de Santa Bárbara, pela concessão do campo de pesquisa.

Ao fotógrafo Diego Ventura, pelas fotos das Figuras 2, 4 e 13a.

Aos colegas do Laboratório de Computação Aplicada do ITV, em especial ao Bruno Ferreira, Gerson Serejo e Leon Cardoso, pelo *esprit de corps* no dia a dia e ao longo dos trabalhos de campo.

À minha família, pela compreensão de algumas ausências em nome da ciência.

À minha noiva Manuella, por todo o apoio, compreensão e pelo empréstimo de seus ouvidos ao longo de tantas horas.

Aos meus pais Heliodoro (*in pectore*) e Virginia, por todas as formas de educação.

A Deus, por me iluminar as veredas e sugerir a melhor forma de percorrê-las.

"Then you better start swimmin'.
Or you'll sink like a stone.
For the times they are a-changin'."
(Robert Allen Zimmerman)

Resumo

Processar dados de diversas fontes e formatos é um problema recorrente em sistemas de informação. Facilitar a busca de relações entre atividades de abelhas (monitoradas vinte e quatro horas por dia) e registros de temperatura do ar, umidade relativa e radiação solar direta e difusa (medidos a cada segundo) é o principal objetivo deste trabalho. As abelhas carregam microssensores que contem um código único para cada indivíduo e geram dados a cada entrada ou saída da colmeia. Próximo a colmeia ficam os sensores responsáveis por coletar os dados meteorológicos. Foi implementada uma arquitetura computacional constituída por um banco de dados, estruturas que facilitam a extração de informação (*data marts*) e uma interface responsável por importar, processar e tratar esses dados para uma melhor visualização da informação.

Palavras-chaves: estrutura de dados. sistemas de informação. microssensores. abelhas.

Abstract

To process data from different formats and sources is a common problem in information systems. The primary objetive of this work is to facilitate queries based on relationship between bee activity (constantly monitored) and registries of air temperature, relative humidity and solar radiation (measured in 1Hz frequency). Bees fitted with micro-sensors recorded with an unique identifier have their passage through the hive door registered. Meteorological data were collected near the hive. A computational architecture was implemented. It is made of a data base with structures that enables data extraction (i.e., data marts) and an user interface to facilitate data storage, processing and treatment for a better data visualization.

Key-words: data structure. information systems. micro-sensors. bees.

Lista de Figuras

Lista de tabelas

Lista de abreviaturas e siglas

CSIRO	*Commonwealth Scientific and Industrial Research Organisation*
CSV	*Comma Separated Values*
DER	Diagrama de Entidades e Relacionamentos
EPI	Equipamento de Proteção Individual
ITV	Instituto Tecnológico Vale
OLAP	*Online Analytical Processing*
RFID	*Radio Frequency Identification*
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SQL	*Structured Query Language*
SSH	*Secure Shell*
TI	Tecnologia da Informação
USB	*Universal Serial Bus*

Sumário

1 Introdução

O primeiro capítulo deste trabalho contextualiza (Seção 1.1) e justifica (Seção 1.2) a utilização de etiquetas eletrônicas (microssensores) em abelhas da espécie *Apis mellifera*, apresentando em seguida os objetivos almejados (Seção 1.3) e a estrutura da dissertação (Seção 1.4).

1.1 Contextualização

1.1.1 Projeto *Swarm Sensing*

O projeto *Swarm Sensing* ocorre no formato de cooperação internacional entre o Instituto Tecnológico Vale (ITV) e a *Commonwealth Scientific and Industrial Research Organisation* (CSIRO), tendo experimentos envolvendo insetos sendo realizados tanto no Brasil como na Austrália. A meta do projeto *Swarm Sensing* é criar um microssensor do tamanho de um grão de areia, para que pequenos insetos como mosquitos, formigas e cupins, possam ser estudados com um maior nível de detalhamento ou utilizados em monitoramento ambiental. Atualmente uma prova de conceito é realizada no município de Santa Bárbara, com uma colmeia de *Apis mellifera*. A prova de conceito teve início em maio de 2014 com a instalação semanal de microssensores nos insetos. O protótipo de software desenvolvido neste mestrado utiliza os dados desta prova de conceito.

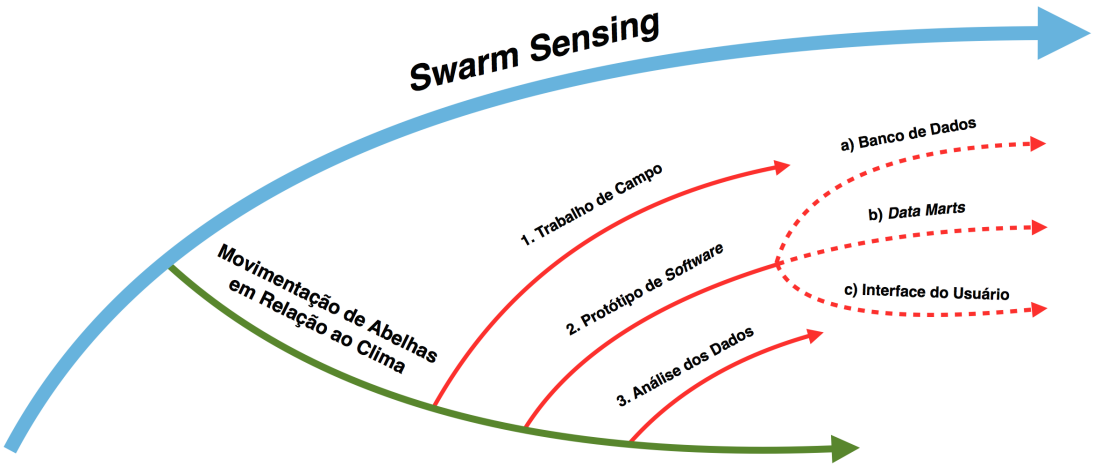


Figura 1 – Escopo da dissertação enquanto parte do projeto *Swarm Sensing*. Na figura, a linha azul representa o eixo principal do projeto, que é o desenvolvimento de microssensores. A linha verde representa o eixo de aplicação da atual tecnologia em monitoramento de insetos. As linhas vermelhas representam os eixos nos quais esta dissertação encontra-se inserida, onde as partes tracejadas representam os resultados e as contribuições mais relevantes.

1.1.2 Escopo da Dissertação

Esta dissertação é uma ramificação do projeto *Swarm Sensing* que trata da proposta e do desenvolvimento de um protótipo de *software* para o estudo da movimentação de abelhas e sua relação com variáveis climáticas (temperatura, umidade relativa e radiação solar). A pesquisa apresenta três grandes eixos, como mostra a Figura 1:

1. Trabalho de campo semanal (Figura 2) com a instalação de microssensores nas abelhas e a coleta dos dados de atividade dos insetos e das variáveis climáticas;
2. Proposta de arquitetura e desenvolvimento de um protótipo de *software*, capaz de importar e tratar os dados obtidos, possibilitando a geração de gráficos para análise de comportamento e correlação. Neste eixo encontra-se o objetivo principal do trabalho, que se ramifica em três eixos menores:
   a) Proposta e desenvolvimento de um banco de dados, visando possibilitar um melhor controle e organização dos dados para a extração de informação;
   b) Geração de *Data Marts*, visando auxiliar as consultas por meio de *Structured Query Language* (SQL) e facilitando a criação de critérios para busca de dados;
   c) Desenvolvimento de uma interface amigável ao pesquisador, proporcionando maior autonomia ao pesquisador que utilizará o sistema, no sentido de diminuir a dependência da área de Tecnologia da Informação (TI);
3. Proposta de ferramenta para análise dos dados, que pode ser realizada por meio de gráficos estatísticos gerados pelo sistema, além da possibilidade de integração com *softwares* desenvolvidos por terceiros, uma vez que o sistema é capaz de exportar os dados armazenados.

Figura 2 – Trabalho de campo semanal realizado em Santa Bárbara do Pará, com a instalação de microssensores e a coleta de dados. Foto: Diego Ventura.

1.2 Justificativa

1.2.1 Importância das Abelhas como Polinizadores

De todo o alimento consumido pela humanidade, estima-se que 35% depende da ação das abelhas (MESSAGE; TEIXEIRA; JONG, 2012). Isto quer dizer que cerca de um terço dos alimentos que chegam às nossas mesas são provenientes da ação desses insetos exercendo o papel de polinizadores. O processo de polinização ocorre quando abelhas operárias visitam flores e o pólen dessas flores fica retido junto aos pelos dos insetos. Conforme visitam diferentes flores, promovem a troca de gametas entre as plantas através do pólen. As operárias de uma única colmeia podem visitar mais de cem mil flores no mesmo dia (HENEIN; LANGWORTHY; ERSKINE, 2009). Michael Pollan[1] se refere às abelhas como sendo as pernas das plantas, exercendo papel fundamental na proliferação de seus genes. A Figura 3 mostra uma abelha coletando pólen de uma flor na ilha de Mosqueiro, norte do Brasil.

Figura 3 – Abelha prestando o serviço de polinização em uma ilha da Amazônia.

1.2.2 Declínio do Número de Abelhas

Apesar do crescimento global do número de colmeias domesticadas, a quantidade de abelhas vem diminuindo nos Estados Unidos desde a década de quarenta e em alguns países da Europa desde a década de sessenta (POTTS et al., 2009). Essa diminuição afeta não só as plantas oriundas das florestas nativas mas também tem um impacto significativo na agricultura e, consequentemente, na economia (POTTS et al., 2010).

[1] Michael Pollan em entrevista concedida ao documentário *"Queen of the Sun: What Are the Bees Telling Us?"*, produzido e dirigido por Siegel (2010).

Apesar da maioria dos apiários brasileiros possuir abelhas africanizadas[2], que apresentam uma maior resistência a certos patógenos e parasitas em relação as abelhas europeias puras, um enfraquecimento repentino tem sido notado nas colmeias das regiões Sul e Sudeste do Brasil. Apesar da inexistência de evidências científicas concretas, as suspeitas apontam para um conjunto de fatores, como novos tipos de parasitas, utilização de pesticidas, alterações climáticas, cultivo de monoculturas, ondas eletromagnéticas geradas por torres de telefonia celular, vegetação alterada geneticamente e o manejo inadequado de colmeias (MESSAGE; TEIXEIRA; JONG, 2012; RATNIEKS; CARRECK, 2010), ou uma combinação desses fatores.

1.2.3 Auxílio Tecnológico

Um melhor entendimento do comportamento das abelhas é fundamental para tentar elucidar a redução desses polinizadores, sendo que a utilização de tecnologia de forma adequada é uma poderosa ferramenta a favor do combate ao declínio de tais insetos. O microssensor utilizado no experimento, aqui referenciado também por etiqueta eletrônica, vem contribuir para o levantamento de dados de movimentação das abelhas devido permitir a marcação e o monitoramento de um grande número de indivíduos sem grandes alterações nos ninhos (SOUZA, 2013). A Figura 4 mostra duas abelhas do experimento no Brasil, sendo que a do lado esquerdo encontra-se equipada com uma etiqueta eletrônica.

Figura 4 – Abelhas do experimento no Brasil, sendo que a da esquerda carrega uma etiqueta eletrônica de 2,5 x 2,5 x 0,4 mm e 5,4 mg. Foto: Diego Ventura.

A utilização de sensores para extração de dados do meio ambiente e o tratamento desses dados para geração de informação é uma contribuição tanto do ponto de vista

[2] As abelhas africanizadas são uma espécie híbrida e surgiram no Brasil após o cruzamento de abelhas africanas e europeias (SOMERVILL, 2008; WIESE, 2005).

biológico como tecnológico, o que demonstra o aspecto interdisciplinar deste trabalho e o enquadra na área de tecnologia da informação.

1.3 Objetivos

O objetivo geral do trabalho é prover meios de facilitar o entendimento do comportamento das abelhas em relação às condições meteorológicas, sendo o papel desses polinizadores fundamental para a sustentabilidade dos ecossistemas do planeta. Existe atualmente a busca por um "caminho do meio" entre otimistas e pessimistas do desenvolvimento sustentável (VEIGA, 2010), somente assim considerado se as futuras gerações tiverem condições de usufruir dos recursos naturais hoje disponíveis (BANERJEE, 2006).

O objetivo específico do trabalho é estabelecer uma arquitetura de *software*, composta de um banco de dados, de alguns *Data Marts* e de uma interface para o usuário. Essas três estruturas compõem um protótipo de *software* que gera gráficos estatísticos e exporta os dados para integração com ferramentas de terceiros.

1.4 Estrutura

O documento é estruturado em sete capítulos, que apresentam o contexto, a justificativa e os objetivos do trabalho, passando pela revisão bibliográfica e abordando em detalhes o trabalho de campo e a estrutura de *software* proposta. São mostrados ainda os resultados, a discussão, as conclusões e os trabalhos futuros. A seguir são apresentados breves resumos de cada um dos capítulos:

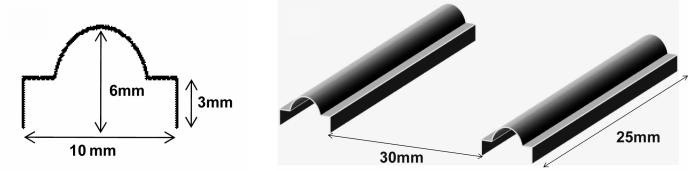
- O Capítulo 2 apresenta a revisão bibliográfica, percorrendo tanto os trabalhos envolvendo o monitoramento de insetos como as tecnologias utilizadas para fazê-lo.
- O Capítulo 3 trata dos materiais e métodos utilizados, detalhando a prova de conceito e a proposta de arquitetura e desenvolvimento de um protótipo de *software*.
- O Capítulo 4 esmiúça os resultados obtidos, mostrando o diagrama de entidades e relacionamentos, o dicionário de dados, os *data marts* e as telas do protótipo. Alguns gráficos gerados pelo protótipo e a possibilidade de exportação de dados também são mostrados.
- O Capítulo 5 mostra a discussão, detalhando e explorando as qualidades do produto.
- O Capítulo 6 conclui a proposta de arquitetura e desenvolvimento.
- O Capítulo 7 aponta os trabalhos futuros, incluindo a integração da interface criada com outros sistemas, a construção de um *Data Warehouse* (armazém de dados) e o desenvolvimento de um aplicativo para aparelhos móveis.

2 Revisão Bibliográfica

A revisão bibliográfica teve dois focos: (i) a utilização de abelhas, apontando algumas das tecnologias utilizadas atualmente no monitoramento destes insetos, e (ii) as estruturas de dados utilizadas, mostrando o que vem sendo feito em relação ao tratamento dos dados.

A dificuldade por parte dos seres humanos em monitorar abelhas a olho nu levou à utilização de etiquetas eletrônicas passivas, que funcionam por meio da tecnologia de rádio frequência (CHEN et al., 2012). Esse tipo de tecnologia para o monitoramento de insetos vem sendo utilizada há pouco mais de uma década. Desde então, a utilização de computadores equipados com bancos de dados tem um papel fundamental na análise dos dados coletados a partir das etiquetas eletrônicas para a geração de informação (STREIT et al., 2003). Além das etiquetas eletrônicas, um equipamento chamado radar harmônico também tem sido utilizado no monitoramento de abelhas. No entanto, o radar harmônico exige que seja acoplada uma antena de 16 mm ao inseto, o que pode interferir em seu comportamento usual (DECOURTYE et al., 2011).

Mais recentemente, pesquisas envolvendo abelhas marcadas com etiquetas eletrôncias vem ocorrendo com o intuito de acompanhar o comportamento destes insetos em relação a pesticidas (HENRY et al., 2012). Um estudo desenvolvido com foco no comportamento de abelhas em contato com inseticidas neonicotinoides utilizou etiquetas de rádio frequência coladas ao tórax dos insetos para monitorar sua movimentação. Utilizando uma colmeia e um alimentador artificial, localizados a sete metros de distância um do outro, as abelhas etiquetadas para entrar ou sair da colmeia ou do alimentador, precisam transpor túneis, como mostra a Figura 5. Cada um dos túneis permite a passagem de apenas uma abelha por vez para que o movimento seja registrado. Os dados coletados são então importados para um software estatístico desenvolvido por terceiros para análises futuras (SCHNEIDER et al., 2012).



(a) Corte (b) Túneis

Figura 5 – Via de passagem das abelhas no trabalho de Schneider et al. (2012).

Grande parte dos dados obtidos por meio de experimentos científicos são armazenados de forma não integrada, geralmente em arquivos de texto simples. A utilização

de bancos de dados possibilita melhores formas de consulta e visualização dos dados armazenados, facilitando a análise por parte dos pesquisadores. Para suprir a necessidade de integração e auxiliar as respostas das questões de pesquisa, o desenvolvimento de um *data warehouse* se apresenta como uma boa prática (MCGUIRE et al., 2008).

Um *data warehouse* tem como uma de suas funções armazenar dados obtidos de diferentes fontes, organizando grupos de dados de acordo com seus propósitos. A tecnologia de Processamento Analítico *Online* (OLAP), utilizada em *data warehouses*, permite a análise de dados através de múltiplas dimensões, o que é aplicável a problemas ambientais e extremamente necessário em áreas como a ecologia. A vantagem do OLAP é permitir ao usuário final a possibilidade de analisar os dados obtidos através de múltiplas dimensões de forma eficiente e confiável (WANG; GUO, 2013).

Costa e Cugnasca (2010) fizeram uso de um *data warehouse* em conjunto com uma rede de sensores sem fio, que mede temperatura do ar e umidade relativa, visando auxiliar o monitoramento de abelhas. O trabalho destaca o processo de extração, transformação e carga de dados (ETL), partindo da rede de sensores sem fio e tendo como destino o banco de dados que abriga o *data warehouse*.

Apesar de iniciativas pontuais, o trabalho de Saraiva e Canhos (2012) aponta uma carência de ferramentas computacionais em relação ao estudo de polinizadores. Uma das situações mencionadas é o fato da maioria dos dados disponíveis não ter origem de coleta e sim observatória, fato este que pode comprometer a qualidade da informação gerada por ser subjetiva. A utilização de um sistema automático de coleta de dados poderia minimizar os problemas existentes atualmente, eliminando-se os erros humanos e contribuindo para uma melhor qualidade da informação. Um outro ponto destacado é a necessidade do desenvolvimento de ferramentas de análise de dados, para que os resultados das análises possam auxiliar novos processos de coleta.

Portanto, o uso da tecnologia de RFID (identificação por rádio frequência) proposta neste trabalho permite monitorar um número grande de insetos com as menores alterações de estrutura de colmeia. Além disso, como a bibliografia indica, *softwares* e estruturas de dados para organização e análise são extremamente importantes, porém os especialistas das áreas praticamente não têm acesso a esses *softwares*.

3 Materiais e Métodos

As atividades desenvolvidas nesta dissertação se resumem ao trabalho de campo em formato de uma prova de conceito realizada no Brasil e a proposta de arquitetura e desenvolvimento de um protótipo de *software*. A Seção 3.1 deste Capítulo detalha os itens inerentes ao trabalho de campo, como a descrição da colmeia utilizada, os aparatos eletrônicos, o procedimento de instalação dos microssensores nas abelhas e a coleta dos dados. Na Seção 3.2 são expostos os itens que compõem o protótipo de *software*, representando o objetivo principal do trabalho.

Um esquema resumindo o funcionamento geral da arquitetura proposta é apresentado na Figura 6. Uma vez por semana, um conjunto de microssensores é instalado nas abelhas. Em seguida, é realizada a coleta dos dados de movimentação das abelhas e de temperatura do ar, umidade relativa e radiação solar. Os arquivos coletados são posteriormente importados para um banco de dados por meio da interface do usuário, responsável também pela geração dos *data marts*.

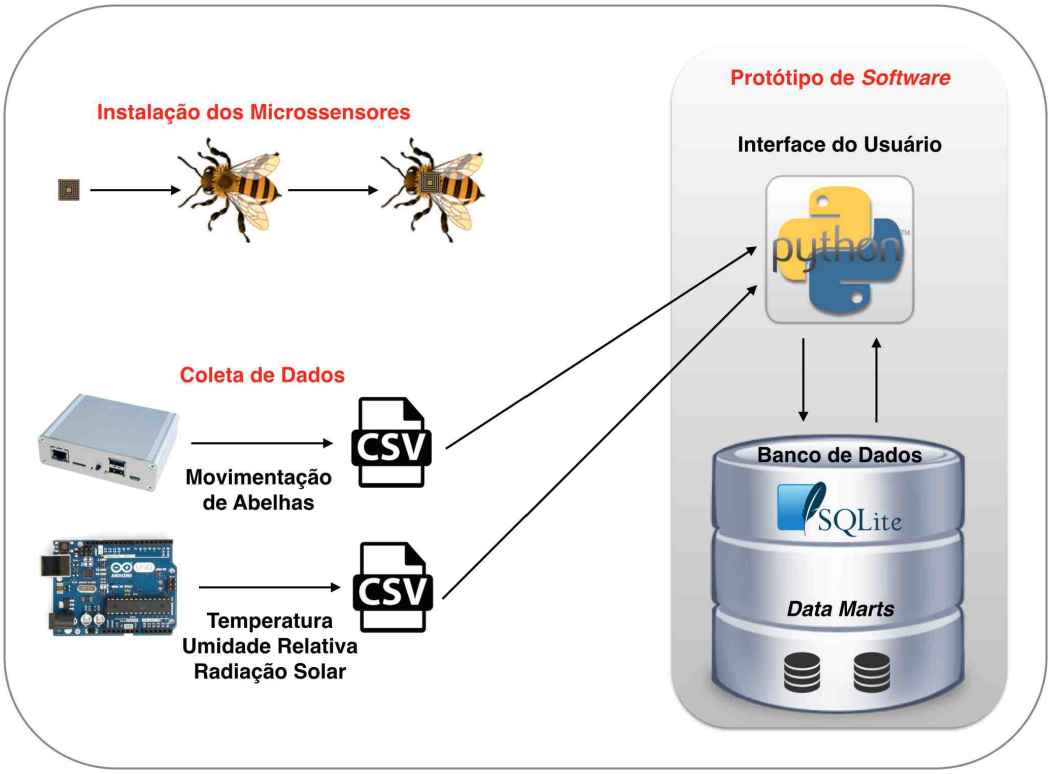


Figura 6 – Visão geral do funcionamento da arquitetura proposta, incluindo a instalação semanal dos microssensores nas abelhas, a coleta dos dados de movimentação das abelhas e dos sensores de temperatura, umidade relativa e radiação solar, e o funcionamento do protótipo de *software*, incluindo a interface do usuário e o banco de dados e seus *data marts*.

3.1 Prova de Conceito e Trabalho de Campo

A prova de conceito foi conduzida no município de Santa Bárbara do Pará, localizado na região nordeste do estado do Pará, a aproximadamente 45 km da capital Belém. A área de pesquisa foi gentilmente cedida pela Cooperativa de Produtores de Cacau e Mel de Santa Bárbara, que possui acesso terrestre através das rodovias BR-316 e PA-391.

(a) Região metropolitana de Belém. (b) Santa Bárbara do Pará, na PA-391.

Figura 7 – Imagens de satélite mostrando o local do experimento.

As Figuras 7a e 7b mostram imagens de satélite obtidas pelo *Google Maps*[1] indicando o local, por meio de uma caixa amarela, e a colmeia, em forma de um ponto vermelho. Devido ao fato das abelhas terem o ciclo de vida de poucas semanas (WIESE, 2005), o trabalho de campo ocorreu semanalmente, levando em torno de três a quatro horas por visita, incluindo o traslado do ITV ao local do experimento e o retorno. O Anexo A (Metadados) traz uma amostra dos relatórios de campo, detalhando os procedimentos realizados em cada visita.

3.1.1 Colmeia

A colmeia utilizada no experimento segue o padrão Langstroth[2] (Figuras 8 e 9), que atualmente é o mais utilizado no mundo (WIESE, 2005). A Figura 8b ilustra de forma mais clara a divisão da colmeia, que possui um ninho, uma melgueira, um alvado modificado para receber a antena RU-824 (PRINCE, 2012), responsável pela leitura das etiquetas

[1] https://maps.google.com.br

[2] Também chamada de "colmeia fria" por ter os quadros dispostos de forma perpendicular ao alvado (orifício de entrada da colmeia), permitindo assim uma melhor circulação de ar.

eletrônicas, e uma caixa plástica anexada, que abriga um pequeno computador, o Ledato NanosG20, aqui referenciado somente por Mini-PC, que é responsável por alimentar a antena e armazenar os dados de movimentação das abelhas (GMBH, 2012). O sistema experimental foi desenvolvido pela CSIRO.

(a) Vista geral da colmeia.

(b) Identificação das partes da colmeia.

Figura 8 – A colmeia da prova de conceito, com a frente voltada para o noroeste. Na Figura 8b percebe-se o alvado adaptado e a caixa plástica de proteção do Mini-PC.

A colmeia fica sobre uma mesa de madeira com os pés isolados por óleo para coibir a ação de formigas. O alvado aponta para a direção noroeste. A colmeia é protegida por duas telhas, como mostra a Figura 8a. O ninho é composto por dez quadros (Figura 9a) e a melgueira é composta por oito quadros (Figura 9b). Sob o ninho existe uma base e sobre a melgueira existe uma tampa.

(a) Ninho

(b) Melgueira

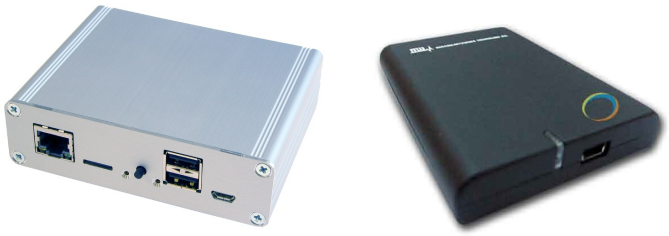
Figura 9 – A colmeia aberta, vista de cima.

3.1.2 Mini-PC e Antena

O alvado da colmeia (Figura 8b) sofreu uma modificação para receber a Antena RU-824[3] (Figura 10b). Esta é responsável por realizar a leitura das etiquetas eletrônicas

[3] http://www.mti.com.tw

acopladas ao tórax das abelhas. Essa antena é ligada por meio de um cabo USB ao Mini-PC[4] (Figura 10a), que recebe alimentação da rede elétrica local. Para proteção contra a ação do tempo, o Mini-PC fica em uma caixa plástica, que foi anexada a colmeia e pode ser aberta sem maiores dificuldades para a coleta dos dados de movimentação das abelhas.

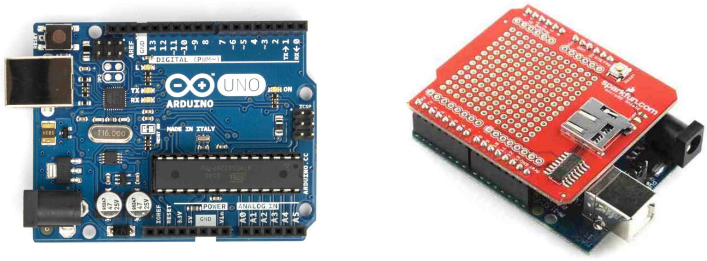


(a) Ledato NanosG20 (b) Antena RU-824

Figura 10 – O Mini-PC e a Antena que funcionam juntos para a leitura das etiquetas.

3.1.3 Miniestação Meteorológica

Abaixo da mesa fica uma miniestação meteorológica responsável por obter dados de temperatura, umidade relativa e radiação solar, construída a partir de uma placa controladora Arduino UNO R3[5] (BANZI, 2012). Sobre esta placa encontra-se encaixada a placa de extensão SparkFun microSD Shield[6] que suporta um cartão de memória para armazenagem e coleta dos dados climáticos.



(a) Arduino UNO R3 (b) MicroSD Shield encaixado sobre o Arduino.

Figura 11 – A placa controladora Arduino e a placa de expansão para o cartão MicroSD.

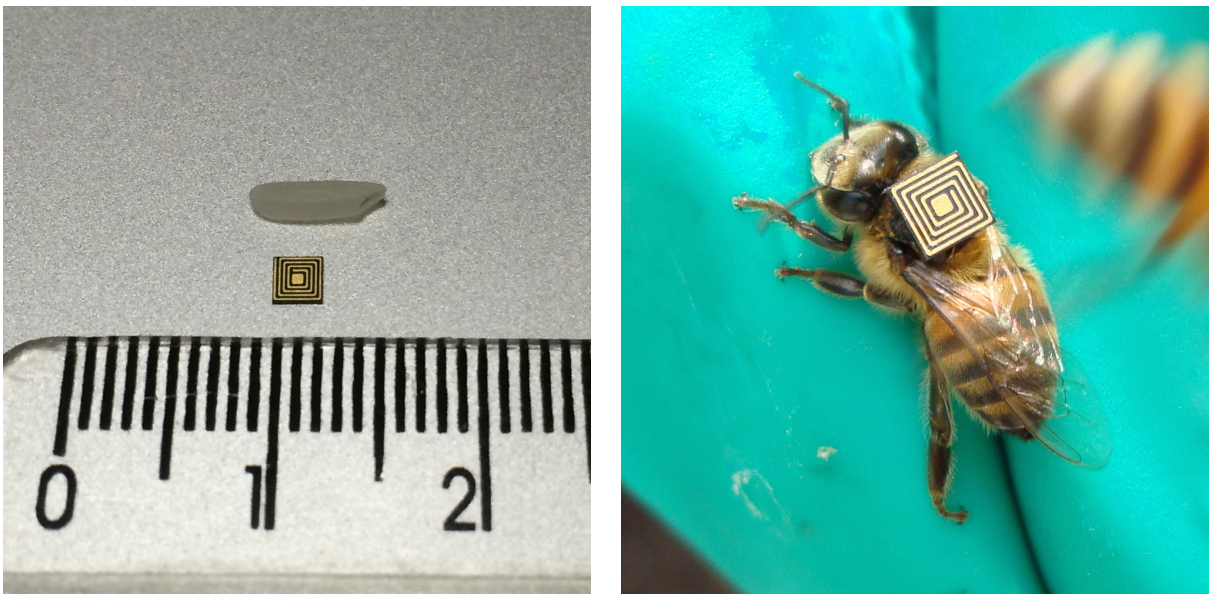
Ligados ao Arduino estão os sensores DHT11[7] (temperatura e umidade relativa), DS18B20[8] (temperatura) e uma célula solar PRT-09241[9] de 5,2 W (radiação solar direta e difusa). Há também um relógio do tipo RTC DS1307[10], soldado a placa de extensão,

[4] https://www.ledato.de
[5] http://arduino.cc
[6] https://www.sparkfun.com
[7] http://www.aosong.com
[8] http://www.dalsemi.com
[9] https://www.sparkfun.com
[10] http://www.maximintegrated.com

responsável por manter o horário de leitura. Uma bateria mantém o horário de leitura atualizado em caso de falta de energia. A miniestação de variáveis climáticas foi construída pelo aluno de Iniciação Científica do Instituto Tecnológico Vale, Bruno Ferreira, como parte de suas atividades de pesquisa.

3.1.4 Etiquetas Eletrônicas

As etiquetas eletrônicas utilizadas no experimento foram desenvolvidas pela CSIRO e fabricadas pela Hitachi. Medem 2,5 x 2,5 x 0,4 mm e pesam 5,4 mg. Cada etiqueta é gravada com um número sequencial único, por meio do qual as abelhas são identificadas. O funcionamento é por meio de identificação por rádio frequência (RFID) (WANT, 2006), e pelo fato de serem passivas, utilizam energia proveniente da antena localizada no alvado, que emana ondas eletromagnéticas capazes de alcançar as etiquetas em uma área de 30 cm^2. A etiqueta, também referenciada aqui por microssensor, pode ser comparada a um grão de arroz, como mostra a Figura 12a. Ela foi desenvolvida para ser acoplada ao tórax de abelhas da espécie *Apis mellifera*, como pode ser observado na Figura 12b.



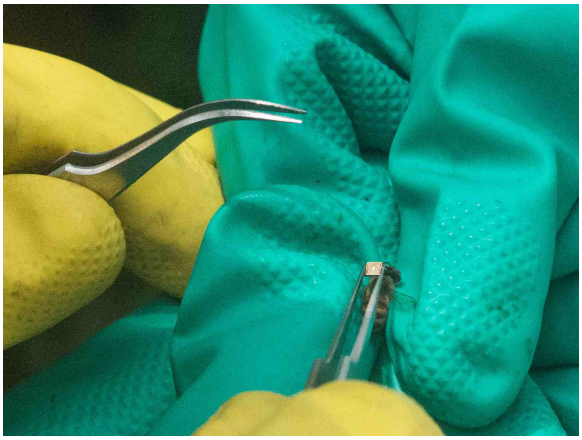
(a) A etiqueta junto a um grão de arroz e uma régua. (b) Uma etiqueta colada ao tórax de uma abelha.

Figura 12 – A etiqueta eletrônica utilizada no estudo.

3.1.5 Colagem das Etiquetas

Ao todo, foram instaladas 1.520 etiquetas entre maio de 2014 e julho de 2015. O processo de colagem das etiquetas, no Brasil, requer dois indivíduos em campo. Um deles fica sentado próximo ao alvado e é responsável por capturar as abelhas com as mãos. O outro indivíduo fica responsável por colar as etiquetas nas abelhas, utilizando uma ou duas pinças e um pouco de cola (Figura 13a). A cola utilizada no processo é de secagem

rápida, no entanto alguns minutos antes de ser iniciada a colagem, é sugerido espalhar um pouco de cola em um papelão ou em uma lâmina de plástico ou vidro, para que a cola adquira uma certa viscosidade e facilite o processo de fixação (Figura 13b). O indivíduo responsável pela colagem utiliza uma pinça para pegar a etiqueta e passar na cola. Logo em seguida, a etiqueta deve ser posicionada sobre o tórax da abelha, com o cuidado de não colar as asas do inseto. Pode ser utilizada uma outra pinça para melhor fixar a etiqueta, fazendo uma leve pressão sobre ela. Na Austrália, um único indivíduo é capaz de capturar as abelhas e colar as etiquetas, contudo por uma questão de segurança, o trabalho de campo é sempre realizado em duplas.



(a) São necessários dois indivíduos para a colagem. Foto: Diego Ventura

(b) As etiquetas eletrônicas e um pouco de cola.

Figura 13 – Processo de colagem das etiquetas eletrônicas nas abelhas.

Os participantes utilizam EPI (Equipamento de Proteção Individual), composto por um macacão de apicultura feito de *nylon*, botas de borracha, chapéu com capuz e luvas de couro ou de borracha. A utilização de luvas de borracha visa facilitar o manuseio tanto das abelhas como das pinças. Antes de chegar próximo a colmeia é imprescindível estar equipado com um fumigador[11] e tomar o cuidado de mantê-lo sempre aceso.

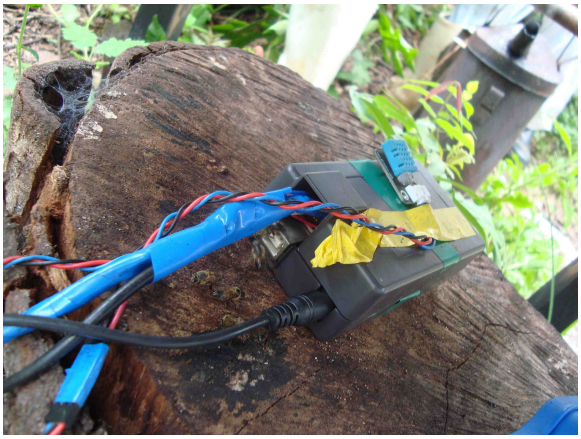
3.1.6 Leitura de Dados

A leitura dos dados de movimentação das abelhas ocorre quando uma abelha etiquetada passa sobre a antena que fica localizada no alvado da colmeia. Através de um cabo USB a antena envia ao Mini-PC os dados que correspondem a data, a hora e o número da etiqueta que encontra-se colada ao tórax da abelha. Os dados são armazenados em arquivos no formato CSV e representam o segundo exato de passagem da abelha.

Os dados de clima são lidos por uma placa Arduino, que é ligada a sensores de temperatura do ar, umidade relativa e radiação solar. Os dados são armazenados em uma

[11] Equipamento composto de fornalha, fole e bico, que através de jatos de fumaça controla a agressividade das abelhas (WIESE, 2005).

outra placa também ligada ao Arduino que possui um cartão de memória responsável por armazenar os dados, de segundo a segundo, em um arquivo no formato CSV. As placas encontram-se no interior de uma caixa plástica, para proteção contra a ação do tempo, que fica localizada abaixo da mesa da colmeia, juntamente com os sensores de temperatura do ar e umidade relativa (Figura 14a). Sobre uma das telhas que protege a colmeia, fica uma célula solar de 180 x 220 mm, responsável por medir a radiação solar (Figura 14b).



(a) Caixa plástica com a placa Arduino.

(b) Célula solar sobre uma das telhas da colmeia.

Figura 14 – Leitura de temperatura do ar, umidade relativa e radiação solar.

3.1.7 Coleta de Dados

Os dados de movimentação das abelhas são coletados por meio de uma conexão ponto a ponto entre o Mini-PC e um *laptop*. A conexão é feita por meio de um cabo de rede *crossover*. Uma vez conectado, o *laptop* estabelece uma conexão segura (SSH) com o Mini-PC para obter os dados de movimentação que encontram-se em arquivos no formato CSV (Figura 15a).

(a) Coleta de dados do Mini-PC. Foto: G. Serejo

(b) Coleta de dados do Arduino.

Figura 15 – Coleta de dados de movimentação das abelhas e de variáveis climáticas.

Os dados de temperatura, umidade relativa e radiação solar são coletados por meio de um cartão de memória que é retirado da placa ligada ao Arduino, responsável por gravar os dados, e inserido em um *laptop*. É realizada então a cópia do arquivo CSV, que contém os dados, do cartão de memória para o *laptop* (Figura 15b).

Quando da concepção da prova de conceito, a rede de dados celulares da região não funcionava a contento, então a coleta de dados ocorreu *in loco*. Caso a rede de dados celulares oferecesse cobertura, poderia ter sido montado um módulo de coleta de dados remoto.

3.2 Proposta de Arquitetura e Desenvolvimento de um Protótipo de *Software*

3.2.1 Prototipação Evolucionária

Os dados obtidos no trabalho de campo, provenientes das etiquetas eletrônicas e dos sensores de temperatura, umidade relativa e radiação solar, encontram-se originalmente disponíveis no formato de arquivo texto (padrão CSV). Para que estes dados possam ser tratados de forma mais simples, no intuito de gerar informação, é proposta uma arquitetura de dados e a prototipação de um *software*. Foi utilizada a abordagem de prototipação evolucionária, onde o protótipo serve como primeira etapa do processo de análise e foi criado a partir dos requisitos levantados no trabalho de campo (PRESSMAN, 2001).

A ideia é que o protótipo venha a servir outros experimentos que utilizem a atividade de abelhas em comparação a variáveis climáticas, além da prova de conceito mencionada ao longo desta dissertação.

A Figura 16 mostra exemplos dos arquivos CSV que são importados pelo protótipo para o banco de dados. A Figura 16a mostra algumas linhas do arquivo de atividade das abelhas e a Figura 16b apresenta algumas linhas do arquivo que armazena os dados dos sensores que medem as variáveis climáticas. Em ambos os casos, a data de ocorrência vem no título dos arquivos. Data e hora seguem o horário solar local de Belém.

```
2014.06.25-B01.csv
--------------------------------
11:34:54,B00003000000000000000000
12:19:05,B00002810000000000000000
12:31:47,B00002810000000000000000
13:28:56,B00003000000000000000000
14:51:16,B00003000000000000000000
```

(a) Atividades das abelhas.

```
2014.06.25-Clima.csv
-----------------------------------
25/6/14,11:35:15,33.00,33.63,37,8.49
25/6/14,11:35:16,33.00,33.63,37,8.45
25/6/14,11:35:17,33.00,33.63,37,8.39
25/6/14,11:35:18,33.00,33.69,37,8.37
25/6/14,11:35:19,33.00,33.69,37,8.42
```

(b) Variáveis climáticas.

Figura 16 – Exemplos de arquivos CSV.

No caso do arquivo de atividade das abelhas, cada linha inicia com a hora, minuto e segundo em que uma abelha equipada com uma etiqueta eletrônica passou pela antena. Em seguida, uma vírgula separa a hora de passagem da abelha do código gravado na etiqueta eletrônica, composto por 24 dígitos. Os 4 primeiros dígitos representam a colmeia e os 4 dígitos seguintes indicam o código único da abelha nesta colmeia. O restante dos dígitos ainda não estava sendo utilizado durante a prova de conceito.

O arquivo de variáveis climáticas inicia pela data de registro da variável, separado da hora, minuto e segundo por uma vírgula. Em seguida, a temperatura (C) do primeiro sensor, a temperatura (C) do segundo sensor, a umidade relativa (%) e a radiação solar (V), todos separados por vírgulas.

3.2.2 Banco de Dados

Visando organizar os dados, facilitar e agilizar a busca pela informação e construir o alicerce para o desenvolvimento de um sistema, foi criado um banco de dados utilizando o SQLite[12,13] como ferramenta. Trata-se de um SGBDR de domínio público e código aberto, sem qualquer tipo de restrição de uso. Pode ser embarcado no aplicativo destino de forma transparente ao usuário (ALLEN; OWENS, 2010).

Uma das grandes vantagens da utilização de um banco de dados é a facilidade apresentada pela linguagem SQL, que permite em poucas linhas de código a obtenção de agrupamentos de dados muitas vezes complexos. Pode ser atribuído a ela o grande sucesso dos bancos de dados relacionais (ELMASRI; NAVATHE, 2003).

3.2.3 Interface do Usuário

Foi desenvolvida uma interface de acesso aos dados utilizando Python, uma linguagem de alto nível, livre e portável (SWAROOP, 2003). Em conjunto com a liguagem Python, várias bibliotecas de funções, como a PyQt, que possibilita a utilização de janelas como interfaces gráficas (SUMMERFIELD, 2008), foram utilizadas visando facilitar e agilizar o desenvolvimento. O protótipo apresenta um menu principal que dá acesso a cinco telas as quais serão detalhadas no próximo capítulo [14].

O protótipo encontra-se em inglês por ser parte de um projeto que ocorre em formato de cooperação internacional entre o Instituto Tecnológico Vale (ITV) e a *Commonwealth Scientific and Industrial Research Organisation* (CSIRO).

---

[12] https://sqlite.org

[13] Embora o SQLite tenha sido o escolhido, qualquer outro banco de dados que siga o padrão SQL poderia ter sido utilizado.

[14] O protótipo será utilizado por pesquisadores nos próximos meses e assim que evoluir para sistema, um processo de registro será realizado.

4 Resultados

Neste capítulo são apresentados os resultados obtidos a partir dos dados do trabalho de campo, incluindo o diagrama de entidades e relacionamentos, os *data marts*, o dicionário de dados e o protótipo funcional.

4.1 Diagrama de Entidades e Relacionamentos

Para a criação do diagrama de entidades e relacionamentos (DER) foi utilizado o *software Valentina Studio*[1], que possibilita a geração do DER a partir de um banco de dados previamente criado no SQLite. A Figura 17 apresenta o diagrama que utiliza a notação *crow's foot* para indicar os relacionamentos (TEOREY et al., 2014).

[1] http://www.valentina-db.com/en/valentina-studio-overview

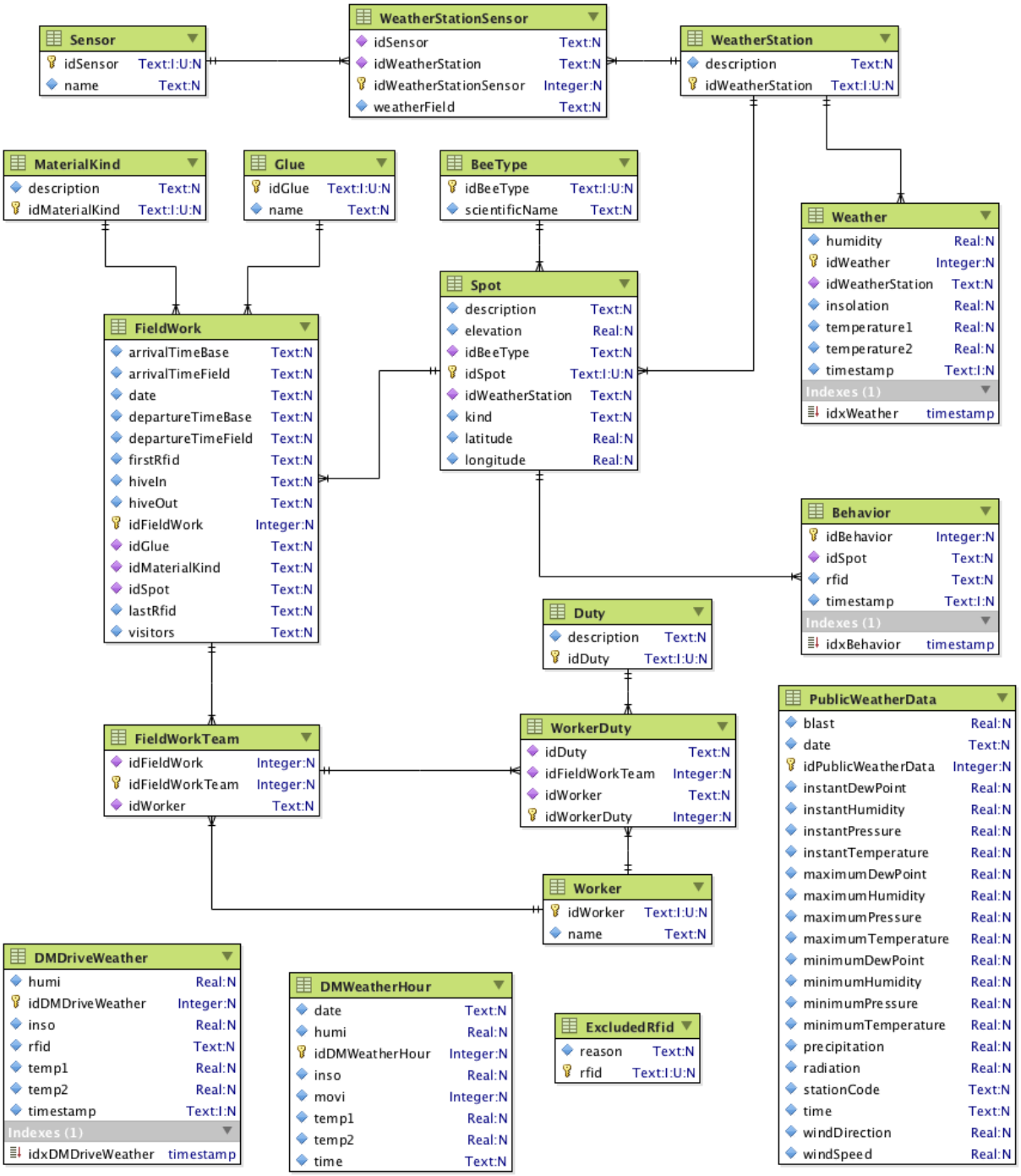


Figura 17 – O Diagrama de Entidades e Relacionamentos é a representação das tabelas existentes no banco de dados, onde cada uma das caixas do esquema representa uma entidade, que equivale a uma tabela. As linhas que ligam as caixas são os relacionamentos existentes entre as tabelas e uma de suas principais funções é garantir a integridade referencial dos dados. São representados pelo diagrama, dados como a atividade das abelhas, as variáveis climáticas, informações sobre o material utilizado na prova de conceito e o trabalho de campo.

4.2 *Data Marts*

Um *Data Mart* funciona como um pequeno *Data Warehouse* voltado para uma finalidade específica. Pode ser classificado como dependente, quando os dados se originam de um *Data Warehouse* maior, ou independente, quando os dados vêm diretamente do sistema de origem (LANE et al., 2013).

Apesar de *Data Marts* independentes serem considerados por Ariyachandra e Watson (2006) como uma solução pobre em relação a arquitetura de dados, o contexto em que o sistema aqui desenvolvido encontra-se inserido apontou tal solução como a melhor para o momento, tendo em vista que a criação de um grande *Data Warehouse* demandaria um esforço maior e o tempo de desenvolvimento aumentaria demasiadamente. Dada a natureza específica de lidar com a movimentação de abelhas em relação a três variáveis climáticas, a criação de estruturas voltadas especificamente para iniciar o trabalho com essas variáveis mostra-se como aceitável de acordo com o trabalho de Watson (2002), que defende o preceito em desenvolvimento de sistemas de informação de "começar pequeno e pensar grande".

O acesso aos dados pelo usuário deve ocorrer de forma fácil e rápida. Para que isto ocorra, a ferramenta utilizada deve ser o mais simples possível. Os dados devem estar disponíveis para consulta a qualquer instante (KIMBALL; ROSS, 2013).

4.3 Dicionário de Dados

O dicionário de dados, também chamado de repositório de informação, tem como função descrever os dados e seus tipos. É composto pelas colunas "Chave", que utiliza o código PK (*Primary Key*) para identificar as chaves primárias e FK (*Foreign Key*) para identificar as chaves estrangeiras, "Atributo", que mostra os nomes dos atributos no banco de dados, "Tipo", que identifica os tipos de dados utilizados pelo SQLite, "Descrição", que detalha cada um dos atributos e "Valor de Exemplo", que traz um dado da forma que é armazenado no banco, como exemplo. Esse tipo de documento pode ser utilizado tanto pelos usuários do sistema como pelos desenvolvedores (ELMASRI; NAVATHE, 2003).

As Tabelas de 1 a 18 mostram cada uma das entidades do sistema com o seu respectivo dicionário de dados. Entidades são representações de tabelas do banco de dados.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
PK	idBeeType	text	Código de identificação da espécie de abelha a ser marcada.	amell
	scientificName	text	Nome científico da abelha.	Apis mellifera

Tabela 1 – A entidade "BeeType" representa a espécie de abelha a ser marcada.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
PK	idSensor	text	Código do sensor.	dht11
	name	text	Descrição do sensor.	Temperature's and umidity sensor.

Tabela 2 – A entidade "Sensor" representa o tipo de sensor utilizado, sendo que um mesmo sensor pode medir mais de uma variável climática, como o do exemplo acima, que mede temperatura e umidade relativa.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	description	text	Descrição do tipo de material.	Bee
PK	idMaterialKind	text	Código do tipo de material.	b

Tabela 3 – A entidade "MaterialKind" representa o tipo de material a ser marcado, pois além de abelhas poderiam ser marcados outros insetos.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
PK	idGlue	text	Código da cola utilizada para fixar as etiquetas eletrônicas.	nb
	name	text	Nome da cola.	Nice Bond

Tabela 4 – A entidade "Glue" representa a cola utilizada, tendo em vista que diferentes marcas de cola podem ser utilizadas para a marcação das abelhas. Isso permite avaliar a correlação de atividade das abelhas com o tipo de cola utilizado, permitindo avaliar a cola mais adequada para a fixação das etiquetas eletrônicas.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	description	text	Descrição da miniestação de variáveis climáticas.	Belém Hive's Zero Station
PK	idWeatherStation	text	Código da miniestação de variáveis climáticas.	w00

Tabela 5 – A entidade "WeatherStation" representa a miniestação de variáveis climáticas.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
FK	idSensor	text	Código do sensor.	dht11
FK	idWeatherStation	text	Código da miniestação de variáveis climáticas.	w00
PK	idWeatherStationSensor	integer	Código de identificação do conjunto de sensores da miniestação de variáveis climáticas.	1
	weatherField	text	Código da variável climática da tabela "Weather" correspondente ao sensor.	temperature1

Tabela 6 – A entidade "WeatherStationSensor" representa o conjunto de sensores que formam a miniestação de variáveis climáticas. Esta entidade tem como função reunir os diferentes sensores e indicar qual é a variável climática que cada um deles é responsável por medir.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	description	text	Descrição do local.	Belém Hive Zero
	elevation	real	Elevação em relação ao nível do mar medida em metros.	26
FK	idBeeType	text	Código de identificação da espécie de abelha a ser marcada.	amell
PK	idSpot	text	Código de identificação do local.	b00
FK	idWeatherStation	text	Código da miniestação de variáveis climáticas associada ao local.	w00
	kind	text	Tipo de local, podendo ser "hive" ou "feeder".	hive
	latitude	real	Latidude medida em graus.	-1.241473
	longitude	real	Longitude medida em graus.	-48.282650

Tabela 7 – A entidade "Spot" representa o local em que foi instalada a colmeia.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	arrivalTimeBase	text	Hora de chegada no laboratório após a realização do trabalho de campo.	12:35
	arrivalTimeField	text	Hora de chegada no campo de pesquisa.	10:23
	date	text	Data em que foi realizado o trabalho de campo.	2014-01-01
	departureTimeBase	text	Hora de saída do laboratório rumo ao campo de pesquisa.	09:24
	departureTimeField	text	Hora de retorno do campo de pesquisa rumo ao laboratório.	11:28
	firstRfid	text	Código da primeira etiqueta eletrônica utilizada neste trabalho de campo.	1
	hiveIn	text	Hora de chegada nos arredores da colmeia.	10:45
	hiveOut	text	Hora de saída dos arredores da colmeia.	11:18
PK	idFieldWork	integer	Código de identificação desta atividade de campo	1
FK	idGlue	text	Código de identificação da cola utilizada nesta atividade de campo.	nb
FK	idMaterialKind	text	Código do tipo de material utilizado no experimento.	b
FK	idSpot	text	Código de identificação do local.	b00
	lastRfid	text	Código da última etiqueta eletrônica utilizada neste trabalho de campo.	25
	visitors	text	Visitantes sem participação direta no trabalho de campo, separados por ponto e vírgula.	Kate;Steve

Tabela 8 – A entidade “FieldWork” representa cada ida a campo para instalação de microssensores em abelhas e coleta de dados.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	humidity	real	Umidade relativa.	56.0
PK	idWeather	integer	Código de identificação do instante de leitura da variável climática.	2068456
FK	idWeatherStation	text	Código da miniestação de variáveis climáticas.	w00
	insolation	real	Radiação solar.	7.4
	temperature1	real	Temperatura ambiente hum.	27.0
	temperature2	real	Temperatura ambiente dois.	26.19
	timestamp	text	Data e hora de leitura.	2014-06-25 17:30:01

Tabela 9 – A entidade “Weather” representa os dados coletados da miniestação de variáveis climáticas, incluindo temperatura, umidade relativa e radiação solar.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
PK	idBehavior	integer	Código da movimentação.	45556
FK	idSpot	text	Código de identificação do local.	b00
	rfid	text	Código da abelha.	709
	timestamp	text	Data e hora da movimentação.	2014-11-07 10:29:27

Tabela 10 – A entidade “Behavior” representa os dados coletados das atividades de cada uma das abelhas.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	description	text	Tipo de tarefa desempenhada.	Sensor’s gluer
PK	idDuty	text	Código da tarefa	sg

Tabela 11 – A entidade “Duty” representa as tarefas que podem ser desempenhadas por cada um dos membros da equipe de campo.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
PK	idWorker	text	Código de identificação do participante.	ha
	name	text	Nome do participante.	Helder Arruda

Tabela 12 – A entidade "Worker" representa cada um dos pesquisadores ou técnicos habilitados a desempenhar o trabalho de campo. Por meio desta informação é possível verificar em qual das tarefas o indivíduo melhor se enquadra.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
FK	idFieldWork	integer	Código de identificação do trabalho de campo.	1
PK	idFieldWorkTeam	integer	Código de identificação do time de campo.	1
FK	idWorker	text	Código do participante.	ha

Tabela 13 – A entidade "FieldWorkTeam" representa a equipe de campo presente em um determinado dia de trabalho.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
FK	idDuty	text	Código da tarefa.	sg
FK	idFieldWorkTeam	integer	Código de identificação do time de campo.	1
FK	idWorker	text	Código do participante.	ha
PK	idWorkerDuty	integer	Código da tarefa do participante.	1

Tabela 14 – A entidade "WorkerDuty" representa o papel desempenhado por um determinado membro da equipe de campo durante um dia de trabalho.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	reason	text	Motivo da exclução do sensor.	Short read interval.
PK	rfid	text	Código do sensor.	347

Tabela 15 – A entidade "ExcludedRfid" representa as etiquetas que por algum motivo devem ser desprezadas durante a análise dos dados.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	blast	real	Intensidade da Rajada do Vento.	0.0
	date	text	Data da coleta.	17/12/2014
PK	idPublicWeatherData	integer	Código de identificação do instante de leitura da estação meteorológica automática.	1
	instantDewPoint	real	Temperatura Instantânea do Ponto de Orvalho.	22.6
	instantHumidity	real	Umidade Relativa Instantânea do Ar.	93
	instantPressure	real	Pressão Atmosférica Instantânea do Ar.	1009.1
	instantTemperature	real	Temperatura Instantânea do Ar.	23.7
	maximumDewPoint	real	Temperatura Máxima do Ponto de Orvalho.	22.8
	maximumHumidity	real	Umidade Relativa Máxima do Ar.	93
	maximumPressure	real	Pressão Atmosférica Máxima do Ar.	1009.1
	maximumTemperature	real	Temperatura Máxima do Ar.	24.0
	minimumDewPoint	real	Temperatura Mínima do Ponto de Orvalho.	22.6
	minimumHumidity	real	Umidade Relativa Mínima do Ar.	93
	minimumPressure	real	Pressão Atmosférica Mínima do Ar.	1008.7
	minimumTemperature	real	Temperatura Mínima do Ar.	23.7
	precipitation	real	Precipitação acumulada no período.	0.0
	radiation	real	Radiação Solar.	-2.22
	stationCode	text	Código de identificação da estação meteorológica automática.	A201
	time	text	Hora da coleta	09
	windDirection	real	Direção do Vento.	0.0
	windSpeed	real	Velocidade Instantânea do Vento.	137

Tabela 16 – A entidade “PublicWeatherData” representa os dados obtidos de uma estação meteorológica automática.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	humi	real	Média da umidade relativa no intervalo de um minuto.	55.87
PK	idDMDriveWeather	integer	Código de identificação da movimentação da abelha em relação à média das variáveis climáticas no minuto do movimento.	16203
	inso	real	Média da radiação solar no intervalo de um minuto.	7.9
	rfid	text	Código do sensor.	300
	temp1	real	Média da temperatura hum no intervalo de um minuto.	29.0
	temp2	real	Média da temperatura dois no intervalo de um minuto.	27.08
	timestamp	text	Data e hora da movimentação.	2014-06-25 16:13:33

Tabela 17 – A entidade "DMDriveWeather" representa o *data mart* responsável por associar as médias das variáveis climáticas que ocorrem no minuto do movimento de entrada/saída da colmeia de uma determinada abelha.

Chave	Atributo	Tipo	Descrição	Valor de Exemplo
	date	text	Data da média horária das variáveis climáticas e do total de movimentos de abelhas no intervalo daquela hora.	2014-06-25
	humi	real	Média horária da umidade relativa.	52.8
PK	idDMWeatherHour	integer	Código de identificação das médias horárias e quantidades de movimentos de abelhas.	640
	inso	real	Média horária da radiação solar.	7.95
	movi	integer	Quantidade total de movimentos de abelhas naquele intervalo de hora.	7
	temp1	real	Média horária da temperatura hum.	29.0
	temp2	real	Média horária da temperatura dois.	27.96
	time	text	Hora do dia.	15

Tabela 18 – A entidade "DMWeatherHour" representa o *data mart* responsável por armazenar o total de movimentos de abelhas ocorridos ao longo de uma determinada hora do dia, juntamente com as médias das variáveis climáticas.

4.4 Protótipo Funcional EiruSoft

A ideia da prototipação evolucionária, ou funcional, é trazer os requisitos iniciais e agregar novos requisitos que possam surgir ao longo do processo de desenvolvimento do protótipo. Este tipo de abordagem permite interagir com uma amostra do produto final, uma vez que este já encontra-se parcialmente em funcionamento (IIBA, 2011).

As subseções 4.4.1 (Sobre o Sistema), 4.4.2 (Importação dos Dados), 4.4.3 (Criação dos *Data Marts*) e 4.4.4 (Geração dos Gráficos) já encontram-se em pleno funcionamento. Foram desenvolvidas nas linguagens *Python* e *SQL*, tendo o *SQLite* como banco de dados.

A seção 4.4.5 (Exportação de Dados) teve a lógica de consulta e extração de dados do banco implementadas, no entanto o *front end* não foi programado, ficando então a tela representada por um *sketch* desenvolvido no software *Pencil*[2].

4.4.1 Sobre o Sistema

Esta tela é chamada através do menu *"About the System"* e traz informações sobre o sistema, como versão e o contexto no qual o mesmo encontra-se inserido (Figura 18).

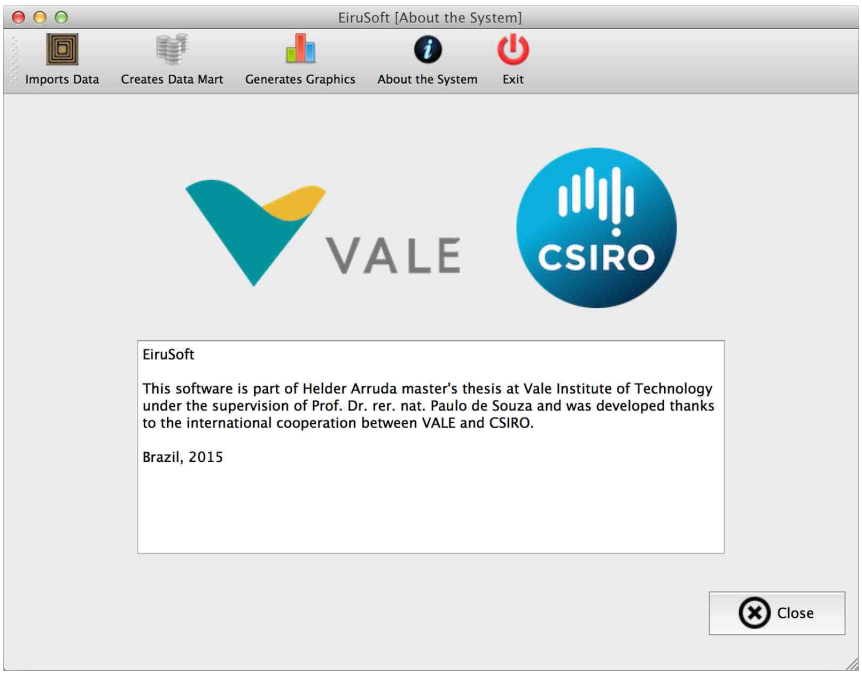


Figura 18 – Tela de informações do sistema. O menu superior se repete em todas as telas do protótipo, com as opções de importação de dados, criação dos *data marts*, geração de gráficos, informações sobre o sistema e saída do sistema.

[2] http://pencil.evolus.vn

4.4.2 Importação dos Dados

A tela de importação dos dados é chamada através do menu *"Imports Data"*. Por meio do botão *"File"*, arquivos de texto no formato CSV são importados para o banco de dados. O sistema verifica se o arquivo contém dados de movimentação de abelhas ou dados de variáveis climáticas (temperatura, umidade relativa e radiação solar), fazendo a importação para a tabela correspondente. A tela permite que o processo de importação seja acompanhado em tempo real pelo usuário para que seja identificada qualquer anomalia (Figura 19).

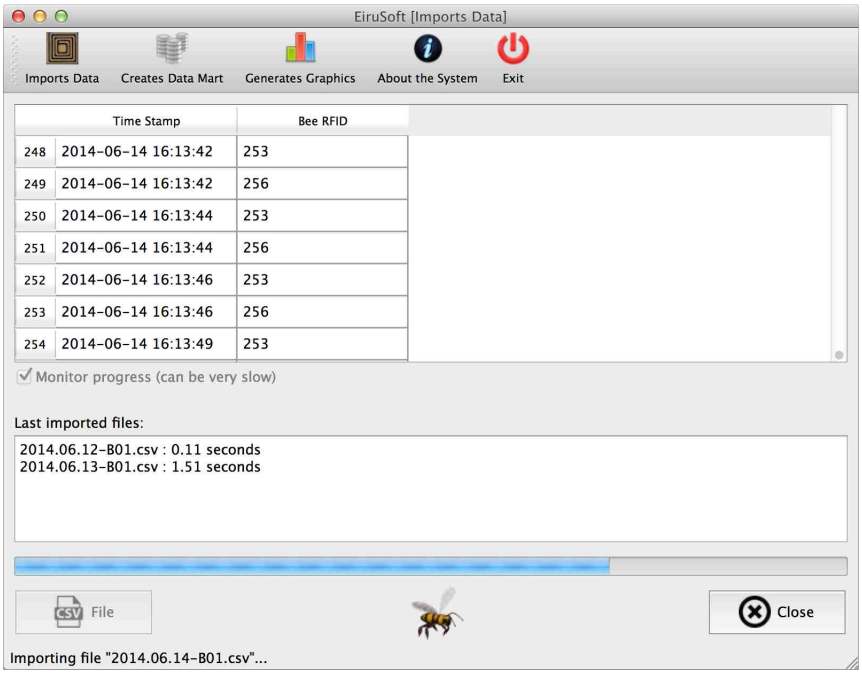


Figura 19 – Tela de importação dos dados de clima e de movimentação de abelhas. A tabela na parte superior indica o evento de saída ou entrada da abelha organizado cronologicamente. A coluna *"Time Stamp"* apresenta a data em ano, mês, dia, hora, minuto e segundo no horário solar local. A coluna *"Bee RFID"* é o valor utilizado para identificar cada abelha. O campo *"Last imported files"* é útil porque permite ao operador do sistema visualizar os últimos arquivos importados e quanto tempo durou cada importação. Na parte inferior, a barra de progresso e a animação de uma abelha em vôo permitem verificar o progresso da importação.

4.4.3 Criação dos *Data Marts*

O menu *"Create Data Mart"* é responsável pelo processo de ETL (*Extraction, Transformation and Loading*), onde cada uma das tabelas que representam *Data Marts* é alimentada a partir dos dados já coletados pelo sistema (MOODY; KORTINK, 2000). Atualmente existem duas estruturas com propósito de *Data Mart* sendo utilizadas, "DM-DriveWeather" (movimentação da abelha em relação à média das variáveis climáticas no minuto do movimento) e "DMWeatherHour" (média horária das variáveis climáticas e do total de movimentos de abelhas no intervalo daquela hora) (Figura 20).

O processo de criação de cada *data mart* inicialmente apaga todos os registros da tabela e em seguida insere os dados atualizados. Este processo pode levar algum tempo dependendo da quantidade de dados existentes na base, contudo agiliza as consultas e gerações de gráficos após o término de sua execução.

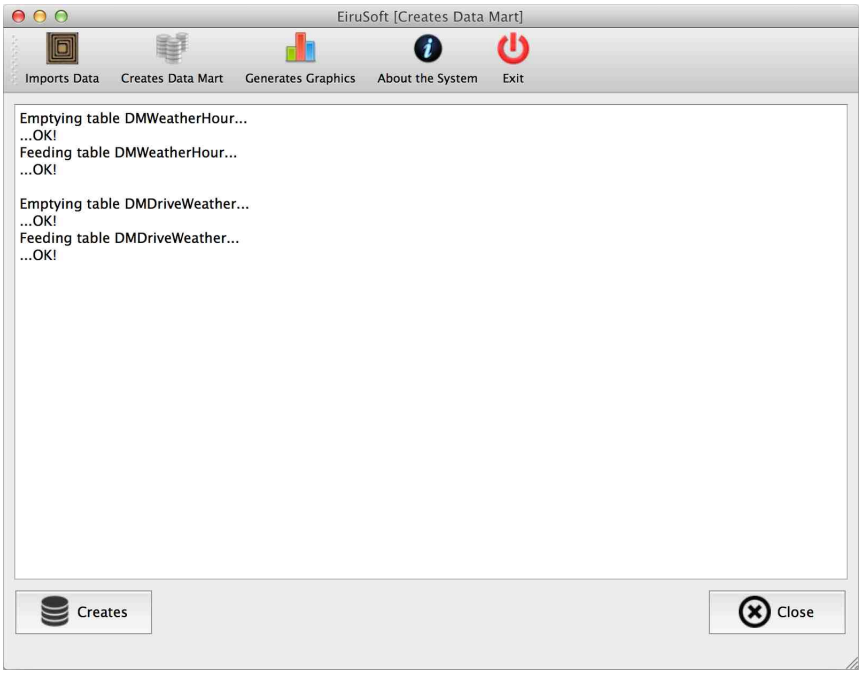


Figura 20 – Tela de criação dos *Data Marts*. Mostra o andamento dos processos de criação de cada um dos *data marts*, informando quando a tabela responsável tem seus registros excluídos e quando volta a ser preenchida.

4.4.4 Geração dos Gráficos

O menu *"Generate Graphics"* aciona a tela com gráficos disponíveis e seus respectivos parâmetros. A Figura 21 mostra um exemplo de chamada de gráfico, com o botão *"Graphic"* ao lado direito do nome do gráfico e dos parâmetros necessários. Através deste botão o gráfico é gerado e mostrado conforme o exemplo da Figura 22.

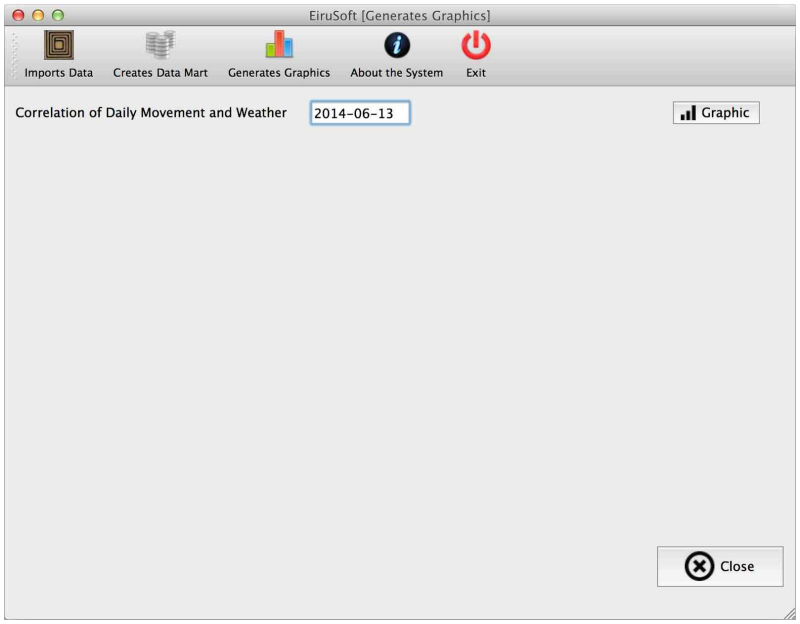


Figura 21 – Tela com o menu de geração dos gráficos. Ao lado do nome do gráfico de exemplo há espaço para o parâmetro data e o botão *"Graphic"* que aciona a tela com o gráfico gerado.

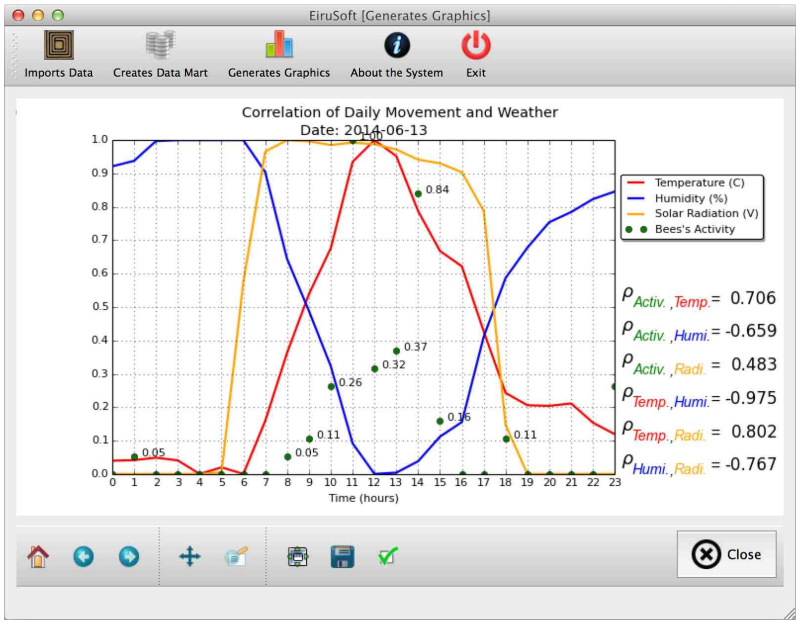


Figura 22 – Tela com o gráfico de correlação da movimentação diária das abelhas em relação ao clima. Pode ser notada uma forte correlação positiva entre a atividade das abelhas e a temperatura.

4.4.4.1 Correlação

O gráfico da Figura 22 apresenta as correlações diárias existentes referentes aos dados de atividade das abelhas, temperatura, umidade relativa e radiação solar. No eixo X encontram-se as horas do dia e no eixo Y os valores normalizados das variáveis. Temperatura, umidade relativa e radiação solar são mostrados como linhas e atividade das abelhas como pontos. As correlações podem ser observadas ao longo do gráfico hora a hora, além dos totais das combinações de cada uma das variáveis, mostrado à direita do gráfico. É observada uma forte correlação positiva entre atividade das abelhas e temperatura. E uma forte correlação negativa entre atividade das abelhas e umidade relativa. Foi utilizada Normalização Linear (GOLDSCHMIDT; PASSOS, 2005) e as correlações foram estabelecidas pelo método de Pearson (SHARMA, 2005).

4.4.4.2 Matriz de Covariância

O gráfico da Figura 23 apresenta a matriz de covariância entre atividade das abelhas, temperatura, umidade relativa e radiação solar. Os dados estão normalizados e os pontos da matriz mostram os coeficientes de determinação (SHARMA, 2005). Nesse tipo de matriz, cada ponto de interseção representa a covariância entre duas variáveis. Na diagonal principal as variáveis se repetem, logo a covariância é igual a 1. Pelo fato da matriz de covariância ser simétrica, o gráfico mostra os valores de covariância apenas na metade inferior da diagonal principal. Quanto mais alta for a covariância, mais escura é a quadra que representa o ponto de interseção entre duas variáveis (HELENE, 2006).

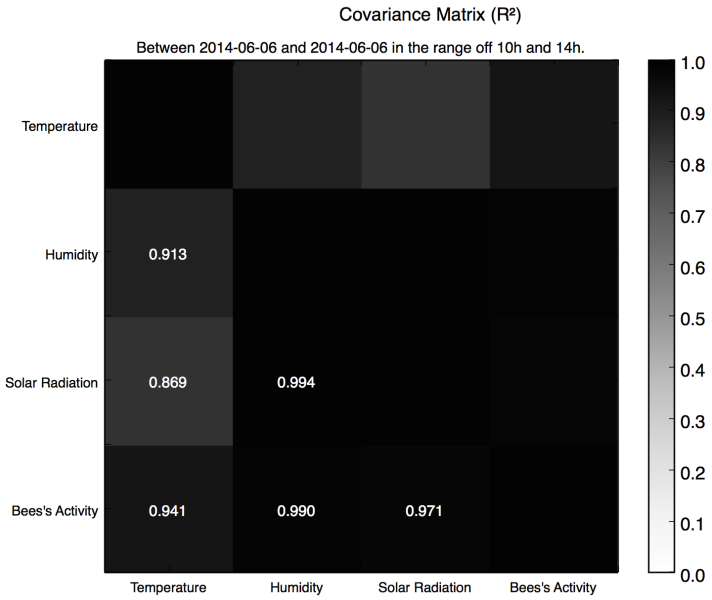


Figura 23 – Na faixa compreendida entre 10 e 14 horas do dia 06/06/2014, o gráfico mostra uma alta covariância entre atividade das abelhas e temperatura, umidade relativa e radiação solar.

4.4.5 Exportação de Dados

Uma vez que os dados estejam devidamente armazenados em um banco de dados relacional, todo e qualquer processo de consulta e extração de informação fica muito mais fácil e prático. A linguagem de consulta *SQL* facilita e agiliza o desenvolvimento de rotinas, tanto de criação de gráficos, como de geração de relatórios ou criação de arquivos para intercâmbio de dados (Figura 24).

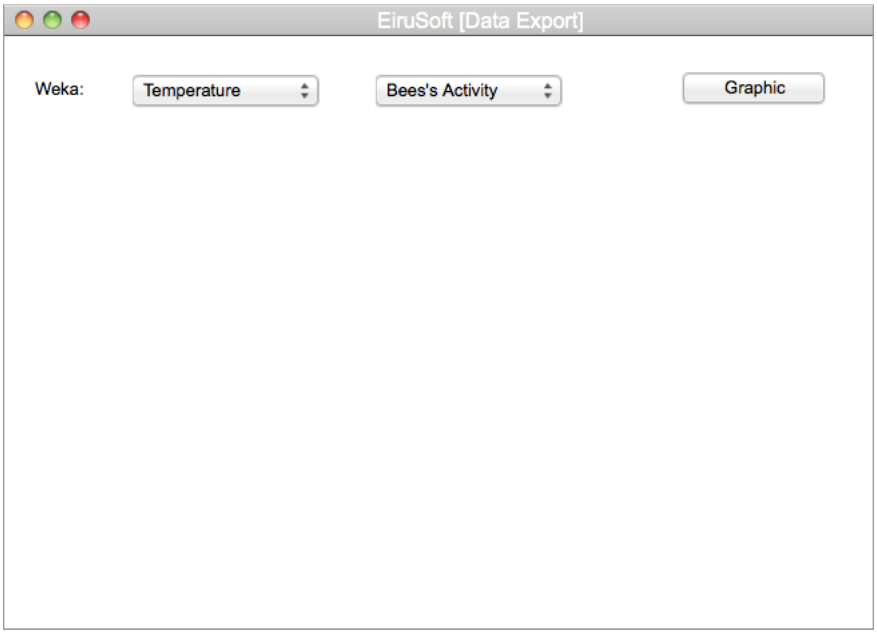


Figura 24 – Para gerar arquivos de intercâmbio de dados, uma tela será responsável por receber os parâmetros passados pelo usuário e gerar os arquivos referentes ao *software* de destino. No exemplo, os parâmetros são temperatura e atividade das abelhas e dizem respeito ao *software* Weka.

4.4.5.1 Weka

Uma das possibilidades de troca de informação é gerar arquivos no formato texto seguindo o padrão ARFF (semelhando ao CSV, porém com informações extras no cabeçalho), que podem ser importados pelo *software* Weka, que trabalha com aprendizado de máquina para geração de diversos tipos de gráficos. O *Weka* possui dezenas de algoritmos para aprendizado de máquina e é muito respeitado na comunidade. No exemplo abaixo, destacamos um gráfico gerado por meio do algoritmo *random forest*, que considera os dados das atividades das abelhas em comparação à temperatura (GARNER, 1995) (Figura 25).

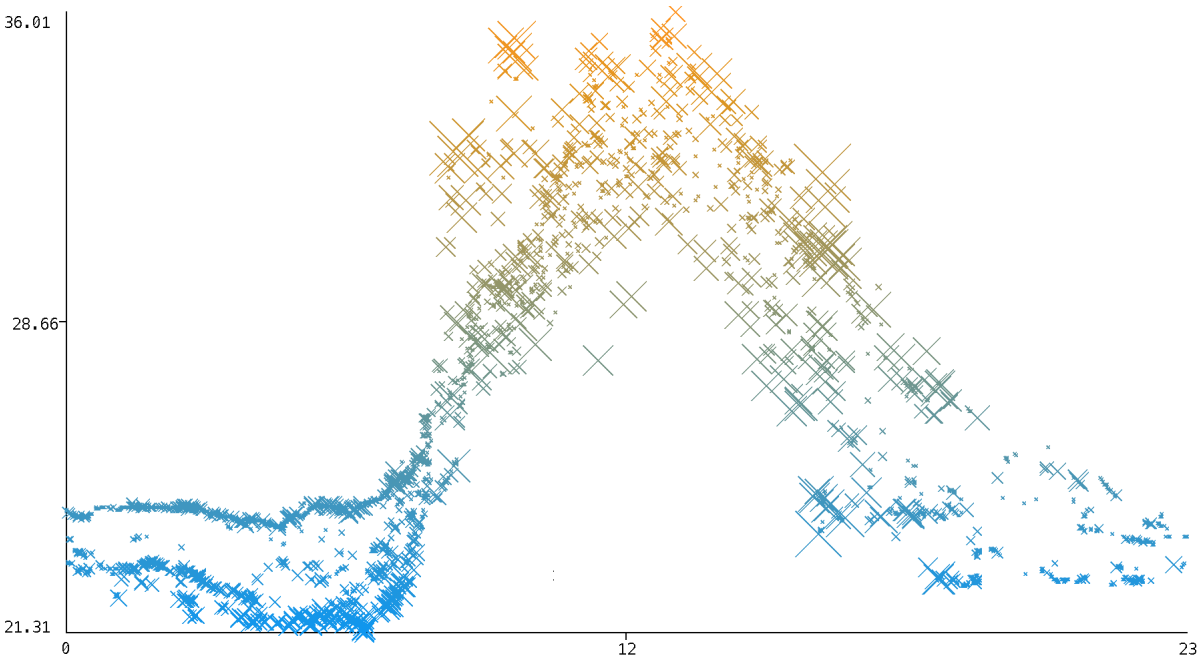


Figura 25 – Gráfico gerado pelo algoritmo *Random Forest*, incluído no *software* Weka. No eixo X encontram-se as atividades das abelhas ao longo das horas do dia e no eixo Y a temperatura medida em graus *Celsius*. Os dados gerados pelo gráfico pertencem ao intervalo compreendido entre final de maio e meados de setembro de 2014.

5 Discussão

O aspecto acadêmico dos sistemas de informação e seu valor enquanto metodologia de pesquisa científica é algo questionado por alguns grupos de pesquisadores. No entanto, uma vez que hipóteses válidas são obtidas e questões são consideradas relevantes, esse aspecto mostra-se possível (NUNAMAKER; CHEN; PURDIN, 1991). A capacidade de manipular grandes quantidades de dados para gerar informação, objetivando a análise dos dados, é discutida a seguir. Também são discutidos os aspectos de diferentes fontes de dados, a implementação do protótipo de *software* e a calibragem dos relógios do experimento.

5.1 Análise dos Dados

A geração de gráficos é um agente facilitador na interpretação de resultados, tornando o processo de análise mais claro para o pesquisador por meio da análise visual. Além disso, a partir da análise de um determinado gráfico, novas perguntas podem emergir, dando origem à elaboração de novos critérios de busca e à criação de novos gráficos.

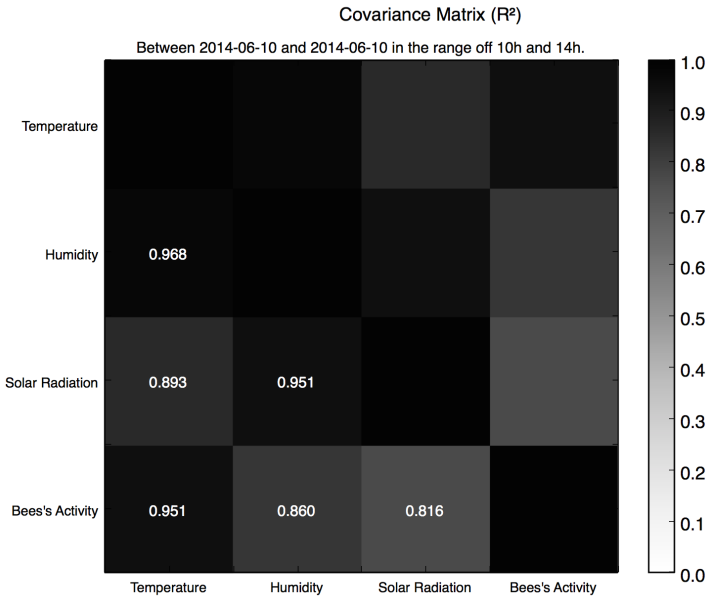


Figura 26 – A matriz de covariância mostra correlação superior a 95% entre atividade das abelhas e temperatura, no dia 10/06/2014 entre 10 e 14 horas.

No exemplo da Figura 26 foram utilizados os dados do mês de junho de 2014. Um critério de busca responsável por percorrer a base de dados procurando os dias que apresentaram correlação superior a 95% entre a atividade das abelhas e as variáveis climáticas no intervalo compreendido entre 10 e 14 horas, encontrou os dias 6, 10 e 27.

Foi esse critério de busca que facilitou a escolha dos dias 6 e 10 de junho de 2014 para utilização nos gráficos das Figuras 23 e 26, respectivamente. Na Figura 27 podemos ver a matriz de covariância do dia 27, também encontrada pelo critério de busca por apresentar correlação superior a 95% entre atividade das abelhas e qualquer uma das variáveis climáticas (temperatura, umidade relativa e radiação solar).

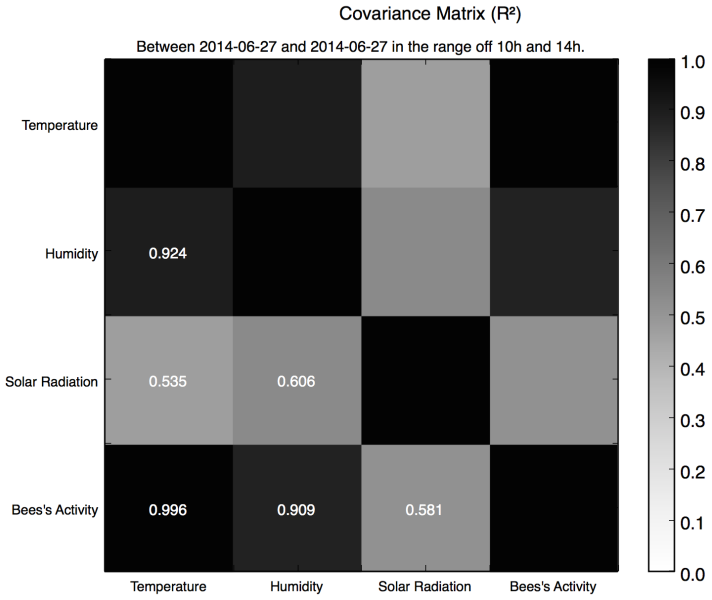


Figura 27 – A matriz de covariância indica correlação superior a 95% entre atividade das abelhas e temperatura, e superior a 90% entre atividade das abelhas e umidade relativa, no dia 27/06/2014 entre 10 e 14 horas.

A flexibilidade de alteração dos critérios de busca permitida por uma base de dados organizada apresenta um ganho de tempo e performance muito grande comparado a utilização de arquivos no formato texto. O tempo investido na organização dos dados e no desenvolvimento do *software* é compensado pela facilidade e agilidade na consulta desses dados em busca de informação.

O critério de busca utilizado pode ser uma correlação estatística no lugar de um dado bruto. Este tipo de filtro seria complicado de implementar em arquivos no formato texto isolados um do outro. Com uma estrutura de dados organizada, este tipo de procedimento torna-se mais simples.

Os gráficos apresentados são gerados por meio da biblioteca Matplotlib, utilizada em conjunto com a linguagem Python. Todos os gráficos gerados podem ser exportados como imagens no formato PNG (Portable Network Graphics), o que facilita a inclusão destes gráficos em artigos e outros documentos. A tela de geração dos gráficos permite ainda dar zoom em áreas selecionadas e redimensionar o gráfico para melhor visualização.

5.2 Diferentes Fontes de Dados

Dados de diferentes fontes constituem um desafio para os processos de extração, transformação e carga, pois para cada tipo de arquivo é necessário um processo distinto. Para os arquivos de atividade das abelhas existe um processo e para os arquivos de temperatura, umidade relativa e radiação solar existe um outro processo. O usuário do sistema visualiza uma única opção de importação por meio do botão *"File"* da tela de importação de dados (Figura 19), pois o software identifica o tipo de arquivo que está sendo importado e direciona este arquivo ao processo responsável.

De acordo com a origem do arquivo, um mesmo tipo de dado pode apresentar formatos diferentes ou aparecer em diferentes posições no arquivo. Na Figura 16a, no arquivo de atividade das abelhas, a data aparece apenas no título do arquivo, mostrando ano (com quatro dígitos), mês e dia separados por pontos ao longo dos dez primeiros dígitos. Logo, conforme o processo percorre o arquivo, a cada leitura de linha, a data deve ser buscada do título. No caso da Figura 16b, no arquivo de variáveis climáticas, a data aparece tanto no título como em cada uma das linhas, porém em formatos diferentes. No título, ano (com quatro dígitos), mês e dia estão separados por pontos. Em cada linha do arquivo, dia, mês e ano (com dois dígitos), encontram-se separados por barras. Além do formato diferente, o número de dígitos do ano também muda, o que requer atenção na implementação dos processos.

Esse tipo de situação é comum devido às diferentes origens dos arquivos. O arquivo de atividade das abelhas é gerado por um script Python, a partir das leituras das etiquetas eletrônicas. O arquivo de variáveis climáticas é gerado por uma placa Arduino, de acordo com os dados dos sensores de temperatura, umidade relativa e radiação solar. É responsabilidade do sistema, por meio dos processos de extração, transformação e carga, organizar estes dados em seus *Data Marts* para auxiliar as consultas feitas ao banco de dados.

5.3 Implementação do Protótipo

O protótipo de *software* aqui apresentado continua em desenvolvimento, pois em engenharia de software é aceitável a construção de sistemas de forma dinâmica, com a elicitação de requisitos ao longo da fase de prototipação (PRESSMAN, 2001).

A solução proposta foi implementada na linguagem Python (Anexos B.1, B.4 e B.5), uma liguagem gratuita que possui um vasto número de bibliotecas para geração de interfaces e gráficos, dispondo de métodos estatísticos validados pela comunidade de desenvolvedores. O sistema é robusto, sendo capaz de importar uma grande quantidade de dados enquanto o sistema operacional da máquina que executa o software pode executar outras tarefas em paralelo.

O software foi desenvolvido e testado em duas versões do sistema operacional OS X (Mavericks e Yosemite), apresentando pequenas alterações na interface, como o tamanho de alguns botões, que não interfere em seu funcionamento. Além do OS X, o software também pode ser utilizado em sistemas Linux e Windows, desde que a linguagem Python e as bibliotecas utilizadas estejam instaladas, incluindo o banco de dados. A sugestão é utilizar a distribuição Anaconda[1], que é gratuita e traz uma vasta coleção de bibliotecas, possibilitando ainda a instalação de bibliotecas adicionais.

O banco de dados utilizado foi o SQLite, que se integra facilmente à linguagem Python e está disponível de forma gratuita para OS X, Linux e Windows. Os processos de criação dos *Data Marts* (Anexos B.2 e B.3) foram desenvolvidos em linguagem SQL e são executados por meio de scripts acionados pelo protótipo.

5.4 Calibragem dos Relógios

A prova de conceito utiliza dois relógios, sendo um localizado no Mini-PC e outro na Miniestação de Variáveis Climáticas. Durante o trabalho de campo, um microssensor gravado com código diferenciado (aqui referenciado por FFF) é utilizado para verificar e calibrar o relógio do Mini-PC. O microssensor FFF foi colado a uma haste de madeira (Figura 28) que é introduzida no alvado da colmeia, de forma a simular a passagem de uma abelha pela antena de leitura. No momento em que o microssensor FFF é passado sobre a antena, o horário de um relógio calibrado, externo ao experimento (e.g, relógio de um celular), é anotado e utilizado como referência na comparação com o horário do relógio do experimento. Para a Miniestação de Variáveis Climáticas, o procedimento é desligar por alguns segundos e voltar a ligar em seguida, anotando o horário exato do relógio externo no momento do desligamento.

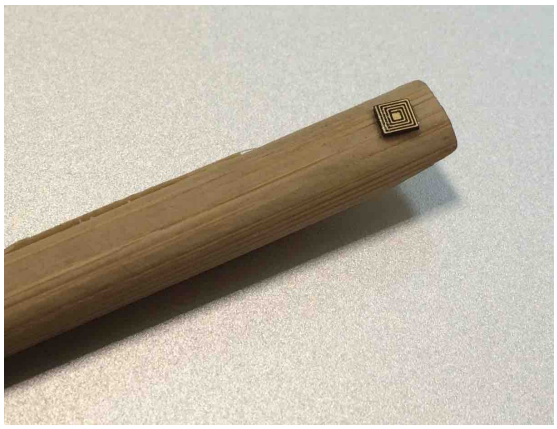


Figura 28 – Haste de madeira com uma etiqueta eletrônica de teste colada na extremidade. Ao ser introduzida na entrada da colmeia, tem como funcionalidade simular a passagem de uma abelha sobre a antena.

[1] https://store.continuum.io/cshop/anaconda/

Estes procedimentos têm por objetivo garantir a integridade dos horários das movimentações das abelhas e das medidas dos sensores, caso haja alguma diferença entre o relógio externo e algum dos relógios do experimento. No caso do Mini-PC, a diferença pode ser corrigida de acordo com a hora em que o microssensor FFF foi utilizado, somando ou subtraindo a diferença. Na miniestação de variáveis climáticas, o horário do desligamento deve ser levado em consideração, uma vez que as medidas são realizadas a cada segundo.

5.5 Interpretação dos Metadados

Os relatórios de campo (Anexo A) contém dados sobre cada visita ao local da prova de conceito e detalhes sobre as atividades semanais que constituem uma importante fonte de informação. São os chamados metadados, de onde partiu o levantamento inicial de requisitos para a implementação do protótipo de software.

Entre as possibilidades existentes na análise dos metadados está a questão do controle de qualidade da colagem das etiquetas eletrônicas nas abelhas. Os relatórios trazem os nomes dos membros da equipe de campo e qual foi a atividade de responsabilidade de cada um. Essas informações podem ser inseridas no banco de dados para servir, por exemplo, de controle de qualidade de colagem das etiquetas. Se a longevidade de um grupo de abelhas etiquetadas por um pesquisador for maior que a de outro, pode ser um indício de que aquele pesquisador desempenha melhor a função de colador de etiquetas.

Outra questão que pode ser analisada é o tipo de cola utilizado para fixar as etiquetas eletrônicas nas abelhas. A cada trabalho de campo o nome da cola utilizada é anotado e pode ser inserido banco de dados. Caso o índice de mortalidade de um grupo de abelhas etiquetadas com um determinado tipo de cola seja elevado em relação a outro tipo de cola, pode ser um indicativo que a cola apresente algum composto tóxico às abelhas.

6 Conclusões

A proposta deste trabalho foi o desenvolvimento de uma arquitetura computacional composta por um banco de dados, *data marts* e uma interface para o usuário, sendo a própria interface responsável pela criação do banco de dados, pela importação dos dados brutos e pela alimentação dos *data marts*. Foi utilizado um banco de dados relacional baseado no padrão SQL, o que facilitou e agilizou o tratamento e a manipulação dos dados, uma vez que a linguagem SQL possibilita tratar os dados como conjuntos. A organização dos dados em estruturas chamadas *data marts* tornou o processo de visualização da informação mais ágil e prático, uma vez que após tratados, os dados ficam disponíveis de forma agrupada e ordenada. Por fim, a utilização de uma interface gráfica para o usuário permite que a ferramenta seja utilizada por qualquer usuário capaz de utilizar um computador, eliminando a dependência de um especialista em computação.

A utilização de abelhas como vetores de dados mostrou-se como um desafio interdisciplinar, uma vez que o processo de manipulação da colmeia durante o trabalho de campo e o entendimento da biologia dos insetos, tanto para a captura como para a fixação das etiquetas eletrônicas, somou-se ao conhecimento em tecnologia da informação para que as estruturas do banco de dados, os *data marts* e a interface do usuário pudessem ser construídos.

O trabalho desenvolvido nessa pesquisa representa um passo essencial na tentativa de auxiliar o atingimento do objetivo geral proposto. Uma vez que os objetivos específicos foram alcançados com o desenvolvimento da interface do usuário, resta a possibilidade de dar continuidade ao desenvolvimento do sistema, incluindo novas funcionalidades e melhorando as existentes.

7 Trabalhos Futuros

7.1 Integração da Interface Desenvolvida com Outros Sistemas

A possibilidade de integração do protótipo desenvolvido com sistemas de terceiros, através do intercâmbio de arquivos, é algo que deve ser levado em consideração no intuito de explorar as diferentes funcionalidades já implementadas, por exemplo, em softwares de inteligência computacional e mineração de dados.

7.2 Criação de um *Data Warehouse* com os Dados Obtidos

Partindo dos *Data Marts* já desenvolvidos, a criação de um grande *Data Warehouse* é algo que pode contribuir com pesquisas futuras que necessitem de dados de movimentação de abelhas em relação a variáveis climáticas. Além disso, a possibilidade de correlacionar um número grande de colmeias serve como motivação para este trabalho.

7.3 Desenvolvimento do EiruApp

A utilização de um *smart phone* com um aplicativo que se conecte a um banco de dados através da *internet* pode facilitar o trabalho de campo. A proposta de desenvolvimento do EiruApp é não só facilitar o trabalho de campo, como agilizar a entrada de dados por parte do pesquisador, que durante ou logo após o trabalho de campo já poderá dar entrada nos metadados e até coletar os dados dos sensores através do aplicativo e enviar ao banco de dados central.

(a) Tela de *login* do aplicativo.

(b) Registro dos metadados do trabalho de campo.

Figura 29 – O EiruApp visa facilitar e agilizar o trabalho de campo do pesquisador.

Bibliografia

ALLEN, G.; OWENS, M. *The Definitive Guide to SQLite*. 2. ed. New York: Apress, 2010. 368 p. ISBN 978-1-4302-3225-4.

ARIYACHANDRA, T.; WATSON, H. J. Which Data Warehouse Architecture Is Most Successful? *Business Intelligence Journal*, v. 11, n. 1, p. 4–6, 2006.

BANERJEE, S. B. Quem sustenta o desenvolvimento de quem? O desenvolvimento sustentável e a reinvenção da natureza. In: *Contra-Discurso do Desenvolvimento Sustentável*. 2. ed. Belém: Associação de Universidades Amazônicas, 2006. p. 77–128. ISBN 85-86037-13-31.

BANZI, M. *Primeiros Passos com o Arduino*. São Paulo: O'Reilly Novatec, 2012. 152 p. ISBN 978-85-7522-290-4.

CHEN, C. et al. An imaging system for monitoring the in-and-out activity of honey bees. *Computers and Electronics in Agriculture*, Elsevier B.V., v. 89, p. 100–109, nov 2012. ISSN 0168-1699.

COSTA, R. A. G. da; CUGNASCA, C. E. Use of data warehouse to manage data from wireless sensors networks that monitor pollinators. *Proceedings - IEEE International Conference on Mobile Data Management*, p. 402–406, 2010. ISSN 1551-6245.

DECOURTYE, A. et al. Honeybee tracking with microchips: a new methodology to measure the effects of pesticides. *Ecotoxicology (London, England)*, v. 20, n. 2, p. 429–437, mar 2011. ISSN 1573-3017.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of Database Systems*. 4. ed. Boston: Pearson Education, Inc., 2003. 1029 p. ISBN 0-321-12226-7.

GARNER, S. R. WEKA: The Waikato Environment for Knowledge Analysis. *Proceedings of the New Zealand computer science*, p. 57–64, 1995.

GMBH, L. *NanosG20 Technical Reference*. 2012. 1–40 p.

GOLDSCHMIDT, R.; PASSOS, E. Etapas do Processo de KDD. In: *Data mining: um guia Prático*. Rio de Janeiro: Elsevier, 2005. p. 23–58. ISBN 85-352-1877-7.

HELENE, O. *Métodos dos Mínimos Quadrados*. São Paulo: Livraria da Física, 2006. ISBN 85-88325-54-3.

HENEIN, M.; LANGWORTHY, G.; ERSKINE, J. *Vanishing of the Bees*. 2009. Disponível em: <http://www.vanishingbees.com>.

HENRY, M. et al. A common pesticide decreases foraging success and survival in honey bees. *Science (New York, N.Y.)*, v. 336, n. 6079, p. 348–350, apr 2012. ISSN 1095-9203.

IIBA. *Um guia para o Corpo de Conhecimento de Análise de Negócios^{TM} (Guia BABOK®)*. 2. ed. Toronto: International Institute of Business Analysis, 2011. 258 p. ISBN 978-0-9811292-42.

KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit, The Definitive Guide to Dimensional Modeling*. Indianapolis: Wiley, 2013. 600 p. ISBN 978-1-118-53080-1.

LANE, P. et al. *Oracle® Database Data Warehousing Guide 11g Release 2 (11.2) E25554-02*. Redwood City, 2013. v. 2, n. July.

MCGUIRE, M. et al. A user-centered design for a spatial data warehouse for data exploration in environmental research. *Ecological Informatics*, v. 3, n. 4-5, p. 273–285, 2008. ISSN 1574-9541.

MESSAGE, D.; TEIXEIRA, É. W.; JONG, D. D. Situação da Sanidade das Abelhas no Brasil. In: *Polinizadores no Brasil: Contribuição e Perspectivas para a Biodiversidade, Uso Sustentável, Conservação e Serviços Ambientais*. São Paulo: Editora da Universidade de São Paulo, 2012. p. 237–356. ISBN 978-85-314-1344-5.

MOODY, D. L.; KORTINK, M. a. R. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000)*, v. 2000, p. 5–16, 2000.

NUNAMAKER, J. F.; CHEN, M.; PURDIN, T. D. M. Systems Development in Information Systems Research. *Journal of Management Information Systems*, v. 7, p. 89–106, 1991. ISSN 0742-1222.

POTTS, S. G. et al. Global pollinator declines: Trends, impacts and drivers. *Trends in Ecology and Evolution*, Elsevier Ltd, v. 25, n. 6, p. 345–353, 2010. ISSN 0169-5347.

POTTS, S. G. et al. Declines of managed honey bees and beekeepers in Europe. v. 49, n. 1, p. 15–22, 2009. ISSN 2078-6913.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*. New York: McGraw-Hill, 2001. 860 p. ISBN 0-07-365578-3.

PRINCE, D. *RU-824 UHF RFID USB Desktop Reader*. 2012. 2 p.

RATNIEKS, F. L. W.; CARRECK, N. L. Clarity on honey bee collapse? *Science (New York, N.Y.)*, v. 327, n. 5962, p. 152–3, jan 2010. ISSN 1095-9203.

SARAIVA, A. M.; CANHOS, D. A. L. Sistemas de Informação e Ferramentas Computacionais para Pesquisa, Educação e Disseminação do Conhecimento sobre Polinizadores. In: *Polinizadores no Brasil: Contribuição e Perspectivas para a Biodiversidade, Uso Sustentável, Conservação e Serviços Ambientais*. São Paulo: Editora da Universidade de São Paulo, 2012. p. 385–432. ISBN 978-85-314-1344-5.

SCHNEIDER, C. W. et al. RFID tracking of sublethal effects of two neonicotinoid insecticides on the foraging behavior of Apis mellifera. *PloS one*, v. 7, n. 1, p. 9, jan 2012. ISSN 1932-6203.

SHARMA, A. K. *Text Book Of Correlations And Regression*. New Delhi: Discovery Publishing House, 2005. ISBN 81-7141-935-6.

SIEGEL, T. *Queen of the Sun: What Are the Bees Telling Us?* 2010. Disponível em: <http://www.queenofthesun.com>.

SOMERVILL, B. A. *Africanized Honey Bee*. Ann Arbor: Cherry Lake Publishing, 2008. 32 p.

SOUZA, P. de. *Projeto Swarm Sensing*. Hobart, 2013.

STREIT, S. et al. Automatic life-long monitoring of individual insect behaviour now possible. *Zoology (Jena, Germany)*, v. 106, n. 3, p. 169–171, 2003. ISSN 0944-2006.

SUMMERFIELD, M. *Rapid Gui Programming with Python and Qt: The Definite Guid to PyQt Programming*. Massachusetts: Prentice Hall, 2008. NP p. ISBN 978-0-13-235418-9.

SWAROOP, C. *A Byte of Python*. [S.l.: s.n.], 2003. 110 p. ISBN 978-957-729-646-7.

TEOREY, T. et al. *Projeto e Modelagem de Banco de Dados*. 2. ed. São Paulo: Elsevier, 2014. 328 p. ISBN 978-85-352-6445-6.

VEIGA, J. E. da. *Desenvolvimento sustentável: o desafio do século XXI*. Rio de Janeiro: Garamond, 2010. 220 p. ISBN 85-7617-051-5.

WANG, H. C.; GUO, J. L. Constructing a water quality 2.0 OLAP system in Taiwan. *Journal of Cleaner Production*, Elsevier Ltd, v. 40, p. 40–45, 2013. ISSN 0959-6526.

WANT, R. An introduction to RFID technology. *IEEE Pervasive Computing*, v. 5, p. 25–33, 2006. ISSN 1536-1268.

WATSON, H. J. Recent Developments in Data Warehousing. *Communications of the Association for Information Systems*, v. 8, p. 1–25, 2002. ISSN 1529-3181.

WIESE, H. *Apicultura*. 2. ed. Guaíba: Agrolivros, 2005. 378 p.

Glossário

crow's foot

é uma notação utilizada para indicar os relacionamentos entre as entidades na construção de Diagramas de Entidades e Relacionamentos, também referenciada como "pé de galinha".

CSV

(*Comma Separated Values*) é um padrão utilizado em arquivos no formato texto cujos valores encontram-se separados por vírgulas, muito utilizado no intercâmbio de dados entre sistemas de informação.

Data Mart

é um conjunto de dados que pode representar parte de um *Data Warehouse*, apresentando um tipo de estrutura que disponibiliza os dados de forma mais organizada e estruturada para consultas e geração de gráficos e relatórios.

Data Warehouse

funciona como um grande armazém de dados, geralmente composto por alguns *Data Marts* e organizado de forma a facilitar extrações de dados históricos.

ETL

(*Extraction, Transformation and Loading*) é o processo responsável pela extração, tratamento e carga dos dados em um *Data Mart* ou em um *Data Warehouse*.

front end

é a parte do *software* visível ao usuário, a interface própriamente dita, representada pelas telas de um sistema, por exemplo.

metadados

são dados que trazem informações sobre outros dados, *e.g.*, os relatórios de campo que indicam os códigos das etiquetas eletrônicas utilizadas em um determinado dia.

random forest

é um algoritmo de aprendizado de máquina utilizado para classificação de dados.

requisito

representa uma funcionalidade do sistema imposta pela necessidade do usuário, como a necessidade de armazenar um determinado dado, *e.g.*, a data de implantação das etiquetas eletrônicas nas abelhas.

RFID

(*Radio Frequency Identification*) é o processo de identificação por rádio frequência, que utiliza ondas de rádio na transmissão de dados.

SQL

(Structured Query Language) é uma linguagem de programação não procedural baseada na álgebra relacional, voltada para bancos de dados relacionais. É por meio desta linguagem que as estruturas utilizadas para armazenar os dados, chamadas de tabelas, são criadas e consultadas.

Anexos

ANEXO A – Metadados

Os metadados, que são informações sobre dados, tem origem no trabalho de campo semanal, realizado há pouco mais de um ano no município de Santa Bárbara do Pará, na Cooperativa de Produtores de Cacau e Mel de Santa Bárbara. Após cada visita ao campo de pesquisa, um relatório é gerado com as atividades desempenhadas no dia. Esses relatórios apresentam relevância *sine qua non* ao processo de levantamento de requisitos funcionais do protótipo, sendo a principal fonte de informação para tal atividade. Encontram-se em inglês pelo fato do projeto ser parte de uma cooperação internacional.

Date of activity (YYYY-MM-DD):	2014-05-28
Time In:	2pm
Time Out:	5.30pm
Participants:	PdS, Gustavo Pessin, Leon Cardoso, Bruno Ferreira, Paulo, Luciana and Genilson (Apiarists)

Rationale:

- Install the system
- Tag first bees
- Clear the area a bit further

Site Observations:

- Bees were relatively active (6)
- A few (and painful) ants came along
- We have a strong rain before arrival, and two short rains during the experiment

Infrastructure:

- Power supply:
  - very unstable
  - no-break was not able to hold the supply at first
  - a transformer (220V-110V) was needed to provide a more stable supply to the no-break
  - A very long cable was needed (100m long for a 80m distance) to bring the power supply to the hive
  - Further work will be needed to ensure the power will be good enough for the experiment
- Hive support:
  - The hive was initially installed over a tripot that proved to be inadequate. The hive was place over a table.
- Ants
  - Ants are aggressive to the the bees. In a single day (actually the day before) another hive at the same area was completely destroyed by ants.
  - burnt oil was placed on the legs of the table.
  - A piece of tissue with oil was placed in the electrical table to prevent bees to reach the hive
- Additional cleaning will be needed as a TV crew will be coming on Saturday and Tuesday/Wednesday.

Hive:

- In the morning Paulo and Gerson (local apiarists) replaced the original box and installed a second one. The original box was replaced by the one we installed the reader's housing and the second one was to increase the space as this hive was a bit over-populated.

- We spotted a few drones.
- 50 sensors were installed. Six were lost. There is a risk of the original sensor fffff (installed in a straw) be tagged to a bee. In this case we need to check if these fffff readings occur when we are not there.
- The glue was not ideal. Probably affected by intense moisture. We need to find a quicker super glue.
- Installing sensors on bees:
    - the frames were suspended by a few wires. As a consequence of this, there was little support to press the "queen marker ring" against the frame. The apiarist (Paulo) found a way to hold the bees with the hands (with gloves) while another person (Paulo, the scientist) could attach the sensors. The process was very fast then. We believe this is a better process we could try in Australia as well.
    - We lost five tags from 1-20, and one from 21-50.

Feeders:

Data:

Oldest Bee recorded: days hours.

Health, Safety, and Environment:

- The heat and moisture were high and wearing the cloths was close to unbearable. Hydration is essential.

Who completed this form? PdS and Gustavo Pessin

Attachments:

Figure 1: Hot, wet and wild...

Swarm Sensing

Date of activity (YYYY-MM-DD):	2014-05-29
Time In:	2.30pm
Time Out:	4pm
Participants:	PdS. Bruno Ferreira, Leon Cardoso, GP, Paulo Said e Luciane (Apiarists)

Rationale:

- install temperature, moisture and light sensors
- tag 50 bees (51-100)
- download data

Site Observations:

- no ants around the hive (but on the ground)

Infrastructure:

- The power supply was cut once to install the temperature/moisture/light sensor. It was working well before our departure
- The computer clock is not working properly (set time). As a consequence the data needs to be corrected.
- Sensors:
  - temperatures, moisture, sun light (solar panel) - DHTxx sensors
  - The sensors are supported by an Arduino Uno

Hive:

- 50 sensors we installed (51-100). Two sensors were lost and one bee was tagged with two sensors (she could fly without visible issues). We spent 40min to install the sensors
- Bee activity (4 to 5)
- All cables were isolated to avoid ants to attack the hive

Data:

Oldest Bee recorded: days hours.

Health, Safety, and Environment:

- extremely hot and humid. Hydration is really needed.
- An ant stung Gustavo Pessin.

Who completed this form? PdS and GP

Attachments:

Date of activity (YYYY-MM-DD):	2014-06-04
Time In:	10am
Time Out:	12.30pm
Participants:	PdS, GP, Alexandre Castilho, Paulo Said, Gerson de Moraes, Cares Belchior (MMA), Flavia Viana (IBAMA), Christiano Borges (Comms Vale), 14 journalists

Rationale:

- install 20 sensors
- record video and photograph the activities with press
- inspect site
- download data

Site Observations:

- site as clean and walls painted

Infrastructure:

- no changes

Hive:

- sensors from 221-240 installed (one sensor lost, felt from bee)
- No issues with data
- Bees were very active (5-7)
- We opened the hive for the press. A couple of bees with sensors were spotted.
- The number of drones is impressive.

Data:

- lots of data collected
- Temperature, humidity and light incidence were recorded.

Oldest Bee recorded: days hours.

Health, Safety, and Environment:

- We performed a debriefing of safety to all journalists in the association center on their arrival.
- The shuttle bus brought water (cold).
- Nobody was sting.

Who completed this form? PdS, GP

Attachments:

Brazilian Press at the site... >:o)

Interview with Globo TV

Attaching sensors with the hands.

Date of activity (YYYY-MM-DD):	2014-08-05
Time In:	10:30am
Time Out:	11:40pm
Participants:	GP, Paulo Said (apiarist), Gerson de Moraes (apiarist). Some visitors from Emater (Empresa de Assistência Técnica e Extensão Rural).

Rationale:

- install sensors 401 – 420
- Inspect Site
- Download data

Site Observations:

- Bees were calm.
- No spider or other animal in the beehive this time.

Infrastructure:

Same as last week, beehive and surrounding environment were very clear. The apiarists are cleaning the area more than usual. They are preparing the area to receive visitors. They want to open the area for visitors during the CONBRAPI - Congresso Brasileiro de Apicultura e Meliponicultura (in November).

Gerson de Moraes wanted to open the beehive because he was thinking that there were a lower number of bees. He said that have occurred a swarming (enxameação) some weeks before. A new queen was found and she is currently putting new eggs. He said that the swarming is a normal behavior of the bees and it is (probably) not related to the experiment. Paulo Said agreed with Gerson de Moraes about the swarming.

- Hardware that collects temperature, light and humidity was working fine.

- The beehive computer was working fine. We did not need to restart it to get the network to connect.

Hives:

To install the sensors, Paulo Said got the bees from the beehive “door”. We have employed a mosquito net to hold some of the bees for a little long. Two bees were not able to move the wings; they were killed and the sensors installed in other two.

One sensor was lost.

Data:

Oldest Bee recorded: days hours.

Health, Safety, and Environment:

- Nobody stung

Who completed this form? GP, PdS

Attachments:

Date of activity (YYYY-MM-DD):	2015-07-16
Time In:	09:19
Time Out:	10:58
Participants:	GP, HA

Rationale:

Deploy of one hundred tags.

Site Observations:

This was our last tagging day in the site. Next month we will collect the data and disassemble the experiment.

HA gathered the activity data.

HA tagged the bees. GP held the bees to be tagged.

Hives:

We have installed 100 sensors, from 1421 to 1520. Two sensors were lost.

The Logs for Santa Barbara do Para (Brazil) was updated with last tagging data.

Bee Activity:

7

Data:

The activity files were got.

Health, Safety, and Environment:

Nobody was stung.

Who completed this form? HA and GP

ANEXO B – Código-fonte

O código-fonte do protótipo compreende scripts em linguagem *Python* e *SQL*. O Anexo B.1 apresenta o módulo principal do protótipo. Os Anexos B.2 e B.3 são os scripts de geração dos *data marts*. Os Anexos B.4 e B.5 são referentes à matriz de covariância.

B.1 EiruSoft.py

```
#!/usr/bin/env python
# coding=utf-8

from subprocess import Popen, PIPE
import sys
import os
import sqlite3
import csv
import time
from datetime import datetime
from PyQt4 import QtCore
from PyQt4 import QtGui
from PyQt4.QtCore import *
from PyQt4.QtGui import *
import random
import matplotlib.pyplot as plt
import sqlite3
import datetime
import numpy as np
from matplotlib.image import BboxImage
from matplotlib.transforms import Bbox, TransformedBbox
from scipy.stats.stats import pearsonr
import sys
from PyQt4.QtGui import QWidget, QPushButton, QMainWindow, QMdiArea,
    QVBoxLayout, QApplication
from PyQt4.QtCore import Qt
from pylab import *
from matplotlib.backends.backend_qt4agg import (
    FigureCanvasQTAgg as FigureCanvas,
    NavigationToolbar2QTAgg as NavigationToolbar)
from matplotlib.backends.backend_wx import NavigationToolbar2Wx
```

```
path = "/Users/helder/Documents/Mestrado/Abelhas/EiruSoft/"

database = path + "data/EiruSoftDB.sqlite"

title = "EiruSoft"

class LacrimaeRa( QtGui.QMainWindow ) :

   def __init__( self ) :
      super( LacrimaeRa, self ).__init__()
      self.initUI()

   def initUI( self ) :

      self.statusBar()

      importsDataAction = QtGui.QAction( QtGui.QIcon( path +
         "images/microssensor30.png" ), "Imports Data", self )
      importsDataAction.triggered.connect( self.callImportsData )
      importsDataAction.setStatusTip( "Imports Data" )

      createsDataMartAction = QtGui.QAction( QtGui.QIcon( path +
         "images/db30.png" ), "Creates Data Mart", self )
      createsDataMartAction.triggered.connect( self.callCreatesDataMart )
      createsDataMartAction.setStatusTip( "Creates Data Mart" )

      generatesGraphicsAction = QtGui.QAction( QtGui.QIcon( path +
         "images/chartbar30.png" ), "Generates Graphics", self )
      generatesGraphicsAction.triggered.connect( self.callGeneratesGraphics )
      generatesGraphicsAction.setStatusTip( "Generates Graphics" )

      aboutAction = QtGui.QAction( QtGui.QIcon( path + "images/info30.png" ),
         "About the System", self )
      aboutAction.triggered.connect( self.callAboutTheSystem )
      aboutAction.setStatusTip( "About the System" )

      exitAction = QtGui.QAction( QtGui.QIcon( path + "images/exit30.png" ),
         "Exit", self )
      exitAction.triggered.connect( self.callExit )
      exitAction.setStatusTip( "Exit" )
```

```
        toolbar = self.addToolBar( "Menu" )
        toolbar.addAction( importsDataAction )
        toolbar.addAction( createsDataMartAction )
        toolbar.addAction( generatesGraphicsAction )
        toolbar.addAction( aboutAction )
        toolbar.addAction( exitAction )
        toolbar.setToolButtonStyle( QtCore.Qt.ToolButtonTextUnderIcon )

        self.setGeometry( 230, 120, 800, 600 )
        self.setWindowTitle( title )
        self.show()

    def callImportsData( self ) :
        impDat = ImportsData()
        self.setCentralWidget( impDat )
        self.setWindowTitle( title + " [Imports Data]" )

    def callCreatesDataMart( self ) :
        creDat = CreatesDataMart()
        self.setCentralWidget( creDat )
        self.setWindowTitle( title + " [Creates Data Mart]" )

    def callGeneratesGraphics( self ) :
        genGra = GeneratesGraphics()
        self.setCentralWidget( genGra )
        self.setWindowTitle( title + " [Generates Graphics]" )

    def callAboutTheSystem( self ) :
        aboSys = AboutTheSystem()
        self.setCentralWidget( aboSys )
        self.setWindowTitle( title + " [About the System]" )

    def callExit( self ) :
        sys.exit()

    def closeWidget( self ) :
        self.setCentralWidget( None )
        self.setWindowTitle( title )



class AboutTheSystem( QtGui.QWidget ) :
```

```
def __init__( self ) :

    QtGui.QWidget.__init__( self )

    self.vale = QtGui.QLabel( self )
    self.vale.setPixmap( QPixmap( path + "images/vale100.png" ) )
    self.vale.resize( 257, 100 )
    self.vale.move( 170, 80 )

    self.csiro = QtGui.QLabel( self )
    self.csiro.setPixmap( QPixmap( path + "images/csiro150.png" ) )
    self.csiro.resize( 150, 150 )
    self.csiro.move( 480, 50 )

    self.text = QtGui.QTextEdit( self )
    self.text.resize( 550, 200 )
    self.text.move( 125, 230 )
    self.text.append( "EiruSoft" )
    self.text.append( "" )
    self.text.append( "This software is part of Helder Arruda master's
        thesis at Vale Institute of Technology under the supervision of Prof.
        Dr. rer. nat. Paulo de Souza and was developed thanks to the
        international cooperation between VALE and CSIRO.")
    self.text.append( "" )
    self.text.append( "Brazil, 2015" )
    self.text.setReadOnly( True )

    self.closeButton = QtGui.QPushButton( "Close", self )
    self.closeButton.setIcon( QIcon( path + "images/close30.png" ) )
    self.closeButton.setIconSize( QSize( 30, 30 ) )
    self.closeButton.resize( 140, 53 )
    self.closeButton.clicked.connect( self.close )
    self.closeButton.move( 654, 461 )

def close( self ) :

    window = self.parent()
    window.closeWidget()
```

```
class CreatesDataMart( QtGui.QWidget ) :

   def __init__( self ) :

      QtGui.QWidget.__init__( self )
      self.setGeometry( 30, 70, 500, 400 )

      self.text = QtGui.QTextEdit( self )
      self.text.resize( 780, 445 )
      self.text.move( 10, 10 )
      self.text.setReadOnly( True )

      self.execButton = QtGui.QPushButton( "Creates", self )
      self.execButton.setIcon( QIcon( path + "images/database30.png" ) )
      self.execButton.setIconSize( QSize( 30, 30 ) )
      self.execButton.resize( 140, 53 )
      self.execButton.move( 5, 461 )
      self.execButton.clicked.connect( self.execute )

      self.closeButton = QtGui.QPushButton( "Close", self )
      self.closeButton.setIcon( QIcon( path + "images/close30.png" ) )
      self.closeButton.setIconSize( QSize( 30, 30 ) )
      self.closeButton.resize( 140, 53 )
      self.closeButton.clicked.connect( self.close )
      self.closeButton.move( 654, 461 )

   def execute( self ) :

      window = self.parent()
      window.statusBar().showMessage( "Creating Data Mart..." )

      p = Popen( [ path + "data/DMWeatherHour.sh" ], stdin = PIPE, stdout =
          PIPE, stderr = PIPE )
      output, err = p.communicate( b"x" )
      rc = p.returncode
      self.text.append( output )

      dmapis = Popen( [ path + "data/DMDriveWeather.sh" ], stdin = PIPE,
          stdout = PIPE, stderr = PIPE )
      dmapisoutput, dmapiserr = dmapis.communicate( b"y" )
      dmapisrc = dmapis.returncode
      self.text.append( dmapisoutput )
```

```
        window.statusBar().showMessage( "" )

    def close( self ) :

        window = self.parent()
        window.closeWidget()


class GeneratesGraphics( QtGui.QWidget ) :

    def __init__( self ) :

        QtGui.QWidget.__init__( self )
        self.setGeometry( 30, 70, 500, 400 )

        vBox = QVBoxLayout()

        self.botao = QtGui.QPushButton( "Graphic", self )
        self.botao.setIcon( QIcon( path + "images/chart15.png" ) )
        self.botao.setIconSize( QSize( 15, 15 ) )
        self.botao.resize( 100, 35 )
        self.botao.move( 670, 11 )
        self.botao.clicked.connect( self.teste )

        self.edit = QtGui.QLineEdit( "2014-06-01", self )
        self.edit.resize( 100, 23 )
        self.edit.move( 310, 15 )

        self.label = QtGui.QLabel( "Correlation of Daily Movement and Weather",
            self )
        self.label.resize( 278, 23 )
        self.label.move( 11, 15 )

        self.botaoSair = QtGui.QPushButton( "Close", self )
        self.botaoSair.setIcon( QIcon( path + "images/close30.png" ) )
        self.botaoSair.setIconSize( QSize( 30, 30 ) )
        self.botaoSair.resize( 140, 53 )
        self.botaoSair.clicked.connect( self.sair )
        self.botaoSair.move( 654, 461 )

    def sair( self ) :
```

```
        window = self.parent()
        window.closeWidget()

    def fecha( self ) :
        window = self.parent()
        window.callGeneratesGraphics()

    def teste( self ) :

        self.edit.clearFocus()

        date = str( self.edit.text() )

        fig = plt.figure( facecolor = "white" )
        ax = fig.add_subplot( 111 )

        canvas = FigureCanvas( fig )
        canvas.setParent( self )
        canvas.setFocusPolicy( Qt.StrongFocus )

        self.button = QPushButton( "Close", self )
        self.button.setIcon( QIcon( path + "images/close30.png" ) )

        vBox = QVBoxLayout()
        vBox.addWidget( canvas )
        self.navigation_toolbar = NavigationToolbar( canvas, self )
        self.navigation_toolbar.addWidget( self.button )
        self.connect( self.button, SIGNAL( 'clicked()' ), self.fecha )
        vBox.addWidget( self.navigation_toolbar )

        self.setLayout( vBox )

        ax.tick_params( labelsize = 10 )

        plt.subplots_adjust( right = 0.78 )

        def plotImage(xData, yData, im):
            for x, y in zip(xData, yData):
              if y > 0 :
                   bb = Bbox.from_bounds(x,y,1,5)
                   bb2 = TransformedBbox(bb,ax.transData)
```

```
            bbox_image = BboxImage(bb2,
                                   norm = None,
                                   origin=None,
                                   clip_on=False)

            bbox_image.set_data(im)
            ax.add_artist(bbox_image)

db = sqlite3.connect( database )
cursor = db.cursor()

cursor.execute('''

   select

      date,

      time,

      (
         temp2
         -
         ( select min( temp2 ) from dmweatherhour where date( date ) =
            date( w.date ) )
      ) / (
         ( select max( temp2 ) from dmweatherhour where date( date ) =
            date( w.date ) )
         -
         ( select min( temp2 ) from dmweatherhour where date( date ) =
            date( w.date ) )
      ) temp2,

      (
         humi
         -
         ( select min( humi ) from dmweatherhour where date( date ) =
            date( w.date ) )
      ) / (
         ( select max( humi ) from dmweatherhour where date( date ) =
            date( w.date ) )
         -
```

```
            ( select min( humi ) from dmweatherhour where date( date ) =
                date( w.date ) )
        ) humi,

        (
            inso
            -
            ( select min( inso ) from dmweatherhour where date( date ) =
                date( w.date ) )
        ) / (
            ( select max( inso ) from dmweatherhour where date( date ) =
                date( w.date ) )
            -
            ( select min( inso ) from dmweatherhour where date( date ) =
                date( w.date ) )
        ) inso,

        (
            cast( movi as real )
            -
            ( select min( cast( movi as real ) ) from dmweatherhour where
                date( date ) = date( w.date ) )
        ) / (
            ( select max( cast( movi as real ) ) from dmweatherhour where
                date( date ) = date( w.date ) )
            -
            ( select min( cast( movi as real ) ) from dmweatherhour where
                date( date ) = date( w.date ) )
        ) movi

    from dmweatherhour w
    where date( date ) = :date
    order by date, time;

''', [date] )

all_rows = cursor.fetchall()

time = []
temp2 = []
humi = []
inso = []
```

```
movi = []

for row in all_rows :
   time.append( int( row[ 1 ] ) )
   temp2.append( row[ 2 ] )
   humi.append( row[ 3 ] )
   inso.append( row[ 4 ] )
   movi.append( row[ 5 ] )

   if row[ 5 ] > 0 :
      ax.annotate( "{0:.2f}".format( row[ 5 ] ), ( int( row[ 1 ] ) +
          0.35, float( row[ 5 ] ) ), fontsize = 10 )

plt.plot( time, temp2, linewidth = 2, label = 'Temperature (C)', color =
    'red' )
plt.plot( time, humi, linewidth = 2, label = 'Humidity (%)', color =
    'blue' )
plt.plot( time, inso, linewidth = 2, label = 'Solar Radiation (V)',
    color = 'orange' )
plt.plot( time, movi, 'ro', linewidth = 2, label = "Bees's Activity",
    color = 'green' )

plt.legend( loc = 'center left', bbox_to_anchor = ( 1, 0.8 ), fancybox =
    True, shadow = True, prop = { 'size' : 10 } )

plt.xlim( 0, 23 )

plt.xticks( range( 0, 24 ) )
plt.yticks( np.arange( 0, 1.1, 0.1 ) )

plt.suptitle( "Correlation of Daily Movement and Weather", fontsize = 13
    )
plt.title( "Date: " + date, fontsize = 13 )
plt.xlabel( "Time (hours)", fontsize = 10 )

ax.text( 23.30, 0.53, u"ρ", style = "italic", color = "black", fontsize
    = 17 )
ax.text( 23.80, 0.50, "Activ.", style = "italic", color = "green",
    fontsize = 13 )
ax.text( 25.50, 0.50, ",", style = "italic", color = "black", fontsize =
    13 )
```

```
ax.text( 25.70, 0.50, "Temp.", style = "italic", color = "red", fontsize
    = 13 )
ax.text( 27.40, 0.51, "=", style = "italic", fontsize = 15 )
ax.text( 30.40, 0.51, "{:.3f}".format( pearsonr( movi, temp2 )[ 0 ] ),
    fontsize = 15, ha = "right" )

ax.text( 23.30, 0.43, u"ρ", style = "italic", color = "black", fontsize
    = 17 )
ax.text( 23.80, 0.40, "Activ.", style = "italic", color = "green",
    fontsize = 13 )
ax.text( 25.50, 0.40, ",", style = "italic", color = "black", fontsize =
    13 )
ax.text( 25.70, 0.40, "Humi.", style = "italic", color = "blue",
    fontsize = 13 )
ax.text( 27.40, 0.41, "=", style = "italic", fontsize = 15 )
ax.text( 30.40, 0.41, "{:.3f}".format( pearsonr( movi, humi )[ 0 ] ),
    fontsize = 15, ha = "right" )

ax.text( 23.30, 0.33, u"ρ", style = "italic", color = "black", fontsize
    = 17 )
ax.text( 23.80, 0.30, "Activ.", style = "italic", color = "green",
    fontsize = 13 )
ax.text( 25.50, 0.30, ",", style = "italic", color = "black", fontsize =
    13 )
ax.text( 25.70, 0.30, "Radi.", style = "italic", color = "orange",
    fontsize = 13 )
ax.text( 27.40, 0.31, "=", style = "italic", fontsize = 15 )
ax.text( 30.40, 0.31, "{:.3f}".format( pearsonr( movi, inso )[ 0 ] ),
    fontsize = 15, ha = "right" )

ax.text( 23.30, 0.23, u"ρ", style = "italic", color = "black", fontsize
    = 17 )
ax.text( 23.80, 0.20, "Temp.", style = "italic", color = "red", fontsize
    = 13 )
ax.text( 25.50, 0.20, ",", style = "italic", color = "black", fontsize =
    13 )
ax.text( 25.70, 0.20, "Humi.", style = "italic", color = "blue",
    fontsize = 13 )
ax.text( 27.40, 0.21, "=", style = "italic", fontsize = 15 )
ax.text( 30.40, 0.21, "{:.3f}".format( pearsonr( temp2, humi )[ 0 ] ),
    fontsize = 15, ha = "right" )
```

```
        ax.text( 23.30, 0.13, u"ρ", style = "italic", color = "black", fontsize
            = 17 )
        ax.text( 23.80, 0.10, "Temp.", style = "italic", color = "red", fontsize
            = 13 )
        ax.text( 25.50, 0.10, ",", style = "italic", color = "black", fontsize =
            13 )
        ax.text( 25.70, 0.10, "Radi.", style = "italic", color = "orange",
            fontsize = 13 )
        ax.text( 27.40, 0.11, "=", style = "italic", fontsize = 15 )
        ax.text( 30.40, 0.11, "{:.3f}".format( pearsonr( temp2, inso )[ 0 ] ),
            fontsize = 15, ha = "right" )

        ax.text( 23.30, 0.03, u"ρ", style = "italic", color = "black", fontsize
            = 17 )
        ax.text( 23.80, 0.00, "Humi.", style = "italic", color = "blue",
            fontsize = 13 )
        ax.text( 25.50, 0.00, ",", style = "italic", color = "black", fontsize =
            13 )
        ax.text( 25.70, 0.00, "Radi.", style = "italic", color = "orange",
            fontsize = 13 )
        ax.text( 27.40, 0.01, "=", style = "italic", fontsize = 15 )
        ax.text( 30.40, 0.01, "{:.3f}".format( pearsonr( humi, inso )[ 0 ] ),
            fontsize = 15, ha = "right" )

        plt.grid()

class ImportsData( QtGui.QWidget ) :

    directory = path + "files"

    def __init__( self ) :

        QtGui.QWidget.__init__( self )
        self.setGeometry( 30, 70, 500, 400 )

        self.tabela = QtGui.QTableWidget( self )
        self.tabela.move( 10, 10 )
        self.tabela.resize( 780, 242 )

        self.botaoArquivo = QtGui.QPushButton( "File", self )
        self.botaoArquivo.setIcon( QIcon( path + "images/csv30.png" ) )
        self.botaoArquivo.setIconSize( QSize( 30, 30 ) )
```

```
self.botaoArquivo.clicked.connect( self.abreArquivo )
self.botaoArquivo.resize( 140, 53 )
self.botaoArquivo.move( 5, 461 )

self.pbar = QtGui.QProgressBar( self )
self.pbar.move( 10, 433 )
self.pbar.resize( 780, 20 )

self.testando = QtGui.QLabel( self )
self.testando.setPixmap( QPixmap( path +
    "images/animated_gif_bees_06_01.png" ) )
self.atual = "um"
self.testando.resize( 61, 54 )
self.testando.move( 370, 460 )
self.timer = QtCore.QTimer()
self.timer.timeout.connect( self.tempo )

self.botaoSair = QtGui.QPushButton( "Close", self )
self.botaoSair.setIcon( QIcon( path + "images/close30.png" ) )
self.botaoSair.setIconSize( QSize( 30, 30 ) )
self.botaoSair.resize( 140, 53 )
self.botaoSair.clicked.connect( self.sair )
self.botaoSair.move( 654, 461 )

self.checkMonitor = QtGui.QCheckBox( "Monitor progress (can be very
    slow)", self )
self.checkMonitor.move( 10, 255 )

self.labelLast = QtGui.QLabel( "Last imported files:", self )
self.labelLast.move( 10, 300 )

self.text = QtGui.QTextEdit( self )
self.text.resize( 780, 100 )
self.text.move( 10, 320 )
self.text.setReadOnly( True )

# Verifies if the database file already exists. If not, the system
    creates it.
self.verifyDatabase()
```

```
def sair( self ) :

   window = self.parent()
   window.closeWidget()




def tempo( self ) :

   if self.atual == "um" :
      self.testando.setPixmap( QPixmap( path +
          "images/animated_gif_bees_06_02.png" ) )
      self.atual = "dois"

   elif self.atual == "dois" :
      self.testando.setPixmap( QPixmap( path +
          "images/animated_gif_bees_06_03.png" ) )
      self.atual = "tres"

   elif self.atual == "tres" :
      self.testando.setPixmap( QPixmap( path +
          "images/animated_gif_bees_06_04.png" ) )
      self.atual = "quatro"

   elif self.atual == "quatro" :
      self.testando.setPixmap( QPixmap( path +
          "images/animated_gif_bees_06_05.png" ) )
      self.atual = "cinco"

   elif self.atual == "cinco" :
      self.testando.setPixmap( QPixmap( path +
          "images/animated_gif_bees_06_06.png" ) )
      self.atual = "seis"

   elif self.atual == "seis" :
      self.testando.setPixmap( QPixmap( path +
          "images/animated_gif_bees_06_07.png" ) )
      self.atual = "sete"
```

```
        elif self.atual == "sete" :
            self.testando.setPixmap( QPixmap( path +
                "images/animated_gif_bees_06_08.png" ) )
            self.atual = "oito"

        elif self.atual == "oito" :
            self.testando.setPixmap( QPixmap( path +
                "images/animated_gif_bees_06_01.png" ) )
            self.atual = "um"




    def abreArquivo( self ) :

        fname = QtGui.QFileDialog.getOpenFileName( self, "Selecionar arquivo",
            self.directory )

        # If the length of the opened file is equal to 18, the system agrees
            that it is a Behavior file.
        if len( os.path.basename( str( fname ) ) ) == 18 :

            t0 = time.time()

            self.tabela.setColumnCount( 2 )
            self.tabela.setHorizontalHeaderLabels( [ "Time Stamp", "Bee RFID" ] )
            self.tabela.setColumnWidth( 0, 170 )
            self.tabela.setColumnWidth( 1, 160 )

            self.tabela.setRowCount( 0 )

            self.botaoArquivo.setEnabled( False )
            self.checkMonitor.setEnabled( False )

            data = os.path.basename( str( fname ) )[:10].replace( ".", "-" )

            self.timer.start( 100 )

            with open(fname) as f:
                barra = sum(1 for _ in f)
```

```
self.pbar.setMaximum( barra )

f = open( fname, "r" )

db = sqlite3.connect( database )

cursor = db.cursor()

with f :
   reader = csv.reader( f )
   for x, row in enumerate( reader ) :

      window = self.parent()
      window.statusBar().showMessage( "Importing file \"" +
          os.path.basename( str( fname ) ) + "\"..." )

      self.pbar.setValue( x + 1 )
      QtGui.qApp.processEvents()
      time.sleep( 0.001 )

      cursor.execute('''
         insert into behavior( timestamp, rfid, idspot ) values( ?, ?,
             ? )
      ''', (
         data + " " + row[ 0 ].decode( "utf-8" ),
         row[ 1 ].decode( "utf-8" )[4:8].lstrip( "0" ),
         row[ 1 ].decode( "utf-8" )[0:3].lower()
      ) )
      db.commit()

      if self.checkMonitor.isChecked() :

         self.tabela.insertRow( x )
         self.tabela.setItem( x, 0, QtGui.QTableWidgetItem( data + " "
             + row[ 0 ].decode( "utf-8" ) ) )
         self.tabela.setItem( x, 1, QtGui.QTableWidgetItem( row[ 1
             ].decode( "utf-8" )[4:8].lstrip( "0" ) ) )
         self.tabela.scrollToItem( self.tabela.item( x, 0 ) )

self.tabela.scrollToItem( self.tabela.item( x, 0 ) )
f.close()
window.statusBar().showMessage( "" )
```

```
        db.close()

        self.timer.stop()

        self.botaoArquivo.setEnabled( True )
        self.checkMonitor.setEnabled( True )

        print( os.path.basename( str( fname ) ) + " : " + str( round(
            time.time() - t0, 2 ) ) + " seconds" )
        self.text.append( os.path.basename( str( fname ) ) + " : " + str(
            round( time.time() - t0, 2 ) ) + " seconds" )

    # If the length of the opened file is equal to 20, the system agrees
        that it is a Weather file.
    elif len( os.path.basename( str( fname ) ) ) == 20 :

        t0 = time.time()

        self.tabela.setColumnCount( 5 )
        self.tabela.setHorizontalHeaderLabels( [ "Time Stamp", "Temp. 1",
            "Temp. 2", "Humidity", "Insolation" ] )
        self.tabela.setColumnWidth( 0, 170 )
        self.tabela.setColumnWidth( 1, 70 )
        self.tabela.setColumnWidth( 2, 70 )
        self.tabela.setColumnWidth( 3, 70 )
        self.tabela.setColumnWidth( 4, 70 )

        self.tabela.setRowCount( 0 )

        self.botaoArquivo.setEnabled( False )
        self.checkMonitor.setEnabled( False )

        self.timer.start( 100 )

        with open(fname) as f:
          barra = sum(1 for _ in f)
        self.pbar.setMaximum( barra )

        f = open( fname, "r" )

        db = sqlite3.connect( database )
```

```
cursor = db.cursor()

with f :
   reader = csv.reader( f )
   for x, row in enumerate( reader ) :

      window = self.parent()
      window.statusBar().showMessage( "Importing file \"" +
          os.path.basename( str( fname ) ) + "\"..." )

      self.pbar.setValue( x + 1 )
      QtGui.qApp.processEvents()
      time.sleep( 0.001 )

      if row[ 0 ].decode( "utf-8" ) <> "dmy" :

         cursor.execute('''
            insert into weather( timestamp, temperature1, temperature2,
               humidity, insolation, idweatherstation ) values( ?, ?,
               ?, ?, ?, ? )
         ''', (
            str( datetime.datetime.strptime( row[ 0 ].decode( "utf-8" )
               + " " + row[ 1 ].decode( "utf-8" ), "%d/%m/%y %H:%M:%S"
               ) ),
            row[ 2 ].decode( "utf-8" ),
            row[ 3 ].decode( "utf-8" ),
            row[ 4 ].decode( "utf-8" ),
            row[ 5 ].decode( "utf-8" ),
            "w00"
         ) )
         db.commit()

         if self.checkMonitor.isChecked() :

            self.tabela.insertRow( x - 1 )
            self.tabela.setItem( x - 1, 0, QtGui.QTableWidgetItem(
               str( datetime.datetime.strptime( row[ 0 ].decode(
                  "utf-8" ) + " " + row[ 1 ].decode( "utf-8" ),
                  "%d/%m/%y %H:%M:%S" ) )
            ) )
```

```
                self.tabela.setItem( x - 1, 1, QtGui.QTableWidgetItem( row[
                    2 ].decode( "utf-8" ) ) )
                self.tabela.setItem( x - 1, 2, QtGui.QTableWidgetItem( row[
                    3 ].decode( "utf-8" ) ) )
                self.tabela.setItem( x - 1, 3, QtGui.QTableWidgetItem( row[
                    4 ].decode( "utf-8" ) ) )
                self.tabela.setItem( x - 1, 4, QtGui.QTableWidgetItem( row[
                    5 ].decode( "utf-8" ) ) )
                self.tabela.scrollToItem( self.tabela.item( x - 1, 0 ) )

        self.tabela.scrollToItem( self.tabela.item( x - 1, 0 ) )
        f.close()
        window.statusBar().showMessage( "" )

        db.close()

        self.timer.stop()

        self.botaoArquivo.setEnabled( True )
        self.checkMonitor.setEnabled( True )

        print( os.path.basename( str( fname ) ) + " : " + str( round(
            time.time() - t0, 2 ) ) + " seconds" )
        self.text.append( os.path.basename( str( fname ) ) + " : " + str(
            round( time.time() - t0, 2 ) ) + " seconds" )




def verifyDatabase( self ) :

    if not os.path.isfile( database ) :

        p = Popen( [ path + "data/CreatesDatabase.sh" ], stdin = PIPE, stdout
            = PIPE, stderr = PIPE )
        output, err = p.communicate( b"x" )
        rc = p.returncode
        print( output )
        print( err )
        print( str( rc ) )
```

```
def main() :
   app = QtGui.QApplication( sys.argv )
   lr = LacrimaeRa()
   #lr.initUI()
   sys.exit( app.exec_() )

if __name__ == '__main__' :
   main()
```

B.2 DMDriveWeather.sh

```
#!/bin/bash

# This SQL sentence is responsible to aggregate both behavior and weather data
    in a data mart format. Temperature, humidity and insolation are obtained
    from a arithmetic mean among the same minute (0-59 seconds).

echo "Emptying table DMDriveWeather..."

/usr/bin/sqlite3
    /Users/helder/Documents/Mestrado/Abelhas/EiruSoft/data/EiruSoftDB.sqlite "

delete from dmdriveweather

"

echo "...OK!"



echo "Feeding table DMDriveWeather..."

/usr/bin/sqlite3
    /Users/helder/Documents/Mestrado/Abelhas/EiruSoft/data/EiruSoftDB.sqlite "

insert into dmdriveweather( timestamp, rfid, temp1, temp2, humi, inso )
select
```

```
   b.timestamp,
   b.rfid,
   (
      select round( sum( wi.temperature1 ) / count( * ), 2 )
      from weather wi
      where wi.timestamp between datetime( b.timestamp, strftime( '-%S
         seconds', b.timestamp ) )
      and datetime( b.timestamp, strftime( '-%S seconds', b.timestamp ),
         strftime( '+59 seconds', b.timestamp ) )
   ) temp1,
   (
      select round( sum( wi.temperature2 ) / count( * ), 2 )
      from weather wi
      where wi.timestamp between datetime( b.timestamp, strftime( '-%S
         seconds', b.timestamp ) )
      and datetime( b.timestamp, strftime( '-%S seconds', b.timestamp ),
         strftime( '+59 seconds', b.timestamp ) )
   ) temp2,
   (
      select round( sum( wi.humidity ) / count( * ), 2 )
      from weather wi
      where wi.timestamp between datetime( b.timestamp, strftime( '-%S
         seconds', b.timestamp ) )
      and datetime( b.timestamp, strftime( '-%S seconds', b.timestamp ),
         strftime( '+59 seconds', b.timestamp ) )
   ) humidity,
   (
      select round( sum( wi.insolation ) / count( * ), 2 )
      from weather wi
      where wi.timestamp between datetime( b.timestamp, strftime( '-%S
         seconds', b.timestamp ) )
      and datetime( b.timestamp, strftime( '-%S seconds', b.timestamp ),
         strftime( '+59 seconds', b.timestamp ) )
   ) insolation
from
   behavior b
      left outer join
   weather w
      on w.timestamp = b.timestamp
where
   b.rfid <> 'FFFF'
order by
```

```
   b.timestamp

"

echo "...OK!"
```

B.3 DMWeatherHour.sh

```
#!/bin/bash

echo "Emptying table DMWeatherHour..."

/usr/bin/sqlite3
    /Users/helder/Documents/Mestrado/Abelhas/EiruSoft/data/EiruSoftDB.sqlite "

delete from dmweatherhour

"

echo "...OK!"



echo "Feeding table DMWeatherHour..."

/usr/bin/sqlite3
    /Users/helder/Documents/Mestrado/Abelhas/EiruSoft/data/EiruSoftDB.sqlite "

insert into dmweatherhour( date, time, temp1, temp2, humi, inso, movi )
select
   date( timestamp ) date,
   strftime( '%H', timestamp ) time,
   round( sum( temperature1 ) / count( * ), 2 ) temp1,
   round( sum( temperature2 ) / count( * ), 2 ) temp2,
   round( sum( humidity ) / count( * ), 2 ) humi,
   round( sum( insolation ) / count( * ), 2 ) inso,
   (
      select count( * )
      from behavior b
      where b.rfid <> 'FFFF' and
```

```
      datetime( b.timestamp ) between
         datetime( date( w.timestamp ) || ' ' || strftime( '%H', w.timestamp )
            || ':00:00' ) and
         datetime( date( w.timestamp ) || ' ' || strftime( '%H', w.timestamp )
            || ':59:59' )
   ) movi
from
   weather w
group by
   date( timestamp ),
   strftime( '%H', timestamp )
order by
   timestamp;

"

echo "...OK!"
```

B.4 CovarianceMatrix.py

```
# coding=utf-8

import math
import random
import matplotlib.pyplot as plt
import sqlite3
import datetime
import numpy as np
from matplotlib.image import BboxImage
from matplotlib.transforms import Bbox, TransformedBbox
from scipy.stats.stats import pearsonr
from sklearn.covariance import ledoit_wolf, GraphLassoCV
from numpy import corrcoef, sum, log, arange
from sklearn.preprocessing import normalize
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import PolyCollection
from matplotlib.colors import colorConverter
```

```
def corrcoefr2(x, y=None, rowvar=1, bias=0, ddof=None):
    c = np.cov(x, y, rowvar, bias, ddof)
    try:
        d = np.diag(c)
    except ValueError:
        return c / c
    r = c / np.sqrt(np.multiply.outer(d, d))
    r2 = np.array( [ [ y**2 for y in x ] for x in r ] )
    return r2


def writeCorr( pr ) :

    return ""



def colorCorr( pr ) :

   if pr > 0.5 :
      return "white"
   else :
      return "black"

database =
    "/Users/helder/Documents/Mestrado/Abelhas/EiruSoft/data/EiruSoftDB.sqlite"

date1 = "2014-06-27"
date2 = "2014-06-27"
time1 = "10"
time2 = "13"

fig = plt.figure( facecolor = "white", dpi = 110 )
ax = fig.add_subplot( 111 )
ax.tick_params( labelsize = 9 )
plt.subplots_adjust( right = 0.78 )

def plotImage(xData, yData, im):
    for x, y in zip(xData, yData):
      if y > 0 :
          bb = Bbox.from_bounds(x,y,1,5)
          bb2 = TransformedBbox(bb,ax.transData)
```

```
            bbox_image = BboxImage(bb2,
                                   norm = None,
                                   origin=None,
                                   clip_on=False)

            bbox_image.set_data(im)
            ax.add_artist(bbox_image)

db = sqlite3.connect( database )
cursor = db.cursor()

cursor.execute("""

   select date, time, temp2, humi, inso, movi
   from dmweatherhour
   where date between :date1 and :date2
   and time between :time1 and :time2
   order by date, cast( time as integer )

""", [ date1, date2, time1, time2 ] )

all_rows = cursor.fetchall()

print( all_rows )

date = []
time = []
temp2 = []
humi = []
inso = []
movi = []

for row in all_rows :

   date.append( row[ 0 ] )
   time.append( int( row[ 1 ] ) )

   temp2.append( float( row[ 2 ] ) )
   humi.append( float( row[ 3 ] ) )
   inso.append( float( row[ 4 ] ) )
   movi.append( int( row[ 5 ] ) )
```

```
plt.suptitle( u"Covariance Matrix (R²)" )
plt.title( "Between " + date1 + " and " + date2 + " in the range off " + time1
    + "h and " + str( int( time2 ) + 1 ) + "h.", fontsize = 10 )

ax.set_xticklabels( [ '', 'Temperature', '', 'Humidity', '', 'Solar
    Radiation', '', "Bees's Activity" ] )
ax.set_yticklabels( [ '', 'Temperature', '', 'Humidity', '', 'Solar
    Radiation', '', "Bees's Activity" ] )

mv2 = np.array( [ temp2, humi, inso, movi ] )
i = ax.imshow( corrcoefr2( mv2 ), interpolation = "nearest", cmap = "Greys",
    vmin = 0, vmax = 1 )

print( "r²" )
print( 55 * "-" )
mr2 = corrcoefr2( mv2 )
print( mr2 )
print( "" )

fig.colorbar( i, fraction = 0.045, pad = 0.055 )

ax.annotate( writeCorr( corrcoefr2( mv2 )[0][1] ) + "\n" + "{0:.3f}".format(
    corrcoefr2( mv2 )[0][1] ), ( 0 - 0.15, 1 + 0.05 ), fontsize = 10, color =
    colorCorr( corrcoefr2( mv2 )[0][1] ) )
ax.annotate( writeCorr( corrcoefr2( mv2 )[0][2] ) + "\n" + "{0:.3f}".format(
    corrcoefr2( mv2 )[0][2] ), ( 0 - 0.15, 2 + 0.05 ), fontsize = 10, color =
    colorCorr( corrcoefr2( mv2 )[0][2] ) )
ax.annotate( writeCorr( corrcoefr2( mv2 )[0][3] ) + "\n" + "{0:.3f}".format(
    corrcoefr2( mv2 )[0][3] ), ( 0 - 0.15, 3 + 0.05 ), fontsize = 10, color =
    colorCorr( corrcoefr2( mv2 )[0][3] ) )
ax.annotate( writeCorr( corrcoefr2( mv2 )[1][3] ) + "\n" + "{0:.3f}".format(
    corrcoefr2( mv2 )[1][3] ), ( 1 - 0.15, 3 + 0.05 ), fontsize = 10, color =
    colorCorr( corrcoefr2( mv2 )[1][3] ) )
ax.annotate( writeCorr( corrcoefr2( mv2 )[1][2] ) + "\n" + "{0:.3f}".format(
    corrcoefr2( mv2 )[1][2] ), ( 1 - 0.15, 2 + 0.05 ), fontsize = 10, color =
    colorCorr( corrcoefr2( mv2 )[1][2] ) )
ax.annotate( writeCorr( corrcoefr2( mv2 )[2][3] ) + "\n" + "{0:.3f}".format(
    corrcoefr2( mv2 )[2][3] ), ( 2 - 0.15, 3 + 0.05 ), fontsize = 10, color =
    colorCorr( corrcoefr2( mv2 )[2][3] ) )

plt.show()
```

B.5 CovarianceMatrixSearch.py

```
# coding=utf-8

import math
import random
import matplotlib.pyplot as plt
import sqlite3
import datetime
import numpy as np
from matplotlib.image import BboxImage
from matplotlib.transforms import Bbox, TransformedBbox
from scipy.stats.stats import pearsonr
from sklearn.covariance import ledoit_wolf, GraphLassoCV
from numpy import corrcoef, sum, log, arange
from datetime import timedelta, date
import matplotlib.dates as md
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import PolyCollection
from matplotlib.colors import colorConverter



database =
    "/Users/helder/Documents/Mestrado/Abelhas/EiruSoft/data/EiruSoftDB.sqlite"
tempr2 = "temp-r2.sqlite"

time1 = "10"
time2 = "13"



def daterange( start_date, end_date ):
    for n in range( int ( ( end_date - start_date).days ) ) :
        yield start_date + timedelta( n )

def corrcoefr2(x, y=None, rowvar=1, bias=0, ddof=None):
    c = np.cov(x, y, rowvar, bias, ddof)
    try:
        d = np.diag(c)
    except ValueError: # scalar covariance
        # nan if incorrect value (nan, inf, 0), 1 otherwise
```

```
        return c / c
    # return c / np.sqrt(np.multiply.outer(d, d))
    r = c / np.sqrt(np.multiply.outer(d, d))
    r2 = np.array( [ [ y**2 for y in x ] for x in r ] )
    return r2

def writeCorr( pr ) :
    return ""

def colorCorr( pr ) :
   if pr > 0.5 :
      return "white"
   else :
      return "black"

def plotImage(xData, yData, im):
    for x, y in zip(xData, yData):
      if y > 0 :
           bb = Bbox.from_bounds(x,y,1,5)
           bb2 = TransformedBbox(bb,ax.transData)
           bbox_image = BboxImage(bb2,
                              norm = None,
                              origin=None,
                              clip_on=False)

           bbox_image.set_data(im)
           ax.add_artist(bbox_image)

def createsDatabase() :

   db = sqlite3.connect( tempr2 )

   cursor = db.cursor()

   cursor.execute( "drop table if exists r2" )

   cursor.execute( '''

      create table r2 (
         date1   text,
         date2   text,
         time1   text,
```

```
        time2   text,
        acti_temp  real,
        acti_humi  real,
        acti_inso  real,
        temp_humi  real,
        temp_inso  real,
        humi_inso  real
      )

   ''' )

   db.close()

def insertRow( date1, date2, time1, time2, acti_temp, acti_humi, acti_inso,
    temp_humi, temp_inso, humi_inso ) :

   db = sqlite3.connect( tempr2 )

   cursor = db.cursor()

   cursor.execute( '''

      insert into r2 ( date1, date2, time1, time2, acti_temp, acti_humi,
         acti_inso, temp_humi, temp_inso, humi_inso )
      values( ?, ?, ?, ?, ?, ?, ?, ?, ?, ? )

   ''', (

      date1, date2, time1, time2, acti_temp, acti_humi, acti_inso, temp_humi,
         temp_inso, humi_inso

   ) )

   db.commit()

   db.close()

def query() :

   db = sqlite3.connect( tempr2 )

   cursor = db.cursor()
```

```
   cursor.execute( '''

      select *
      from (
         select 1 'cod','acti_temp' r2, max( acti_temp ) maxs, date1 date from
            r2
         union
         select 2, 'acti_humi', max( acti_humi ), date1 from r2
         union
         select 3, 'acti_inso', max( acti_inso ), date1 from r2
         union
         select 4, 'temp_humi', max( temp_humi ), date1 from r2
         union
         select 5, 'temp_inso', max( temp_inso ), date1 from r2
         union
         select 6, 'humi_inso', max( humi_inso ), date1 from r2
      ) x
      order by cod

   ''' )

   all_rows = cursor.fetchall()

   print( "" )

   for row in all_rows :
      cod = row[ 1 ]
      r2 = str( row[ 2 ] ).ljust( 5, "0" )
      maxs = row[ 3 ]
      x = cod + u" (max r²) : " + r2 + " (" + maxs + ")"
      print( x.encode( "utf-8" ) )

   db.close()

   print( "" )

def queryGraphic() :

   db = sqlite3.connect( tempr2 )

   cursor = db.cursor()
```

```
cursor.execute( '''

   select *
   from r2

''' )

all_rows = cursor.fetchall()

date = []
acti_temp = []
acti_humi = []
acti_inso = []
temp_humi = []
temp_inso = []
humi_inso = []

for row in all_rows :
   date.append( row[ 0 ] )
   acti_temp.append( row[ 4 ] )
   acti_humi.append( row[ 5 ] )
   acti_inso.append( row[ 6 ] )
   temp_humi.append( row[ 7 ] )
   temp_inso.append( row[ 8 ] )
   humi_inso.append( row[ 9 ] )

fig = plt.figure( facecolor = "white", dpi = 110 )
ax = fig.add_subplot( 111 )
ax.tick_params( labelsize = 9 )

datex = md.datestr2num( date )
plt.plot( datex, acti_temp, label = "acti_temp" )
plt.plot( datex, acti_humi, label = "acti_humi" )
plt.plot( datex, acti_inso, label = "acti_inso" )
plt.plot( datex, temp_humi, label = "temp_humi" )
plt.plot( datex, temp_inso, label = "temp_inso" )
plt.plot( datex, humi_inso, label = "humi_inso" )
ax.xaxis_date()
fig.autofmt_xdate()

plt.subplots_adjust( right = 0.83 )
```

```
    plt.xlim( [ min( datex ), max( datex ) ] )
    plt.xticks( range( int( min( datex ) ), int( max( datex ) ) + 1, 1 ),
        fontsize = 7 )
    plt.yticks( np.arange( 0, 1.1, 0.1 ), fontsize = 7 )

    plt.legend( loc = 'center left', bbox_to_anchor = ( 1, 0.8 ), fancybox =
        True, shadow = True, prop = { 'size' : 10 } )

    plt.suptitle( u"Distribution of the Coefficient of Determination (R^2)",
        fontsize = 13 )
    plt.xlabel( "Days", fontsize = 10 )
    plt.ylabel( u"R^2", fontsize = 10 )

    plt.grid()

    plt.show()

    db.close()



createsDatabase()

start_date = date( 2014, 6, 1 )
end_date = date( 2014, 7, 1 )
for single_date in daterange( start_date, end_date ) :
    print single_date.strftime( "%Y-%m-%d" )
    date1 = single_date.strftime( "%Y-%m-%d" )
    date2 = single_date.strftime( "%Y-%m-%d" )

    db = sqlite3.connect( database )
    cursor = db.cursor()

    cursor.execute("""

        select date, time, temp2, humi, inso, movi
        from dmweatherhour
        where date between :date1 and :date2
        and time between :time1 and :time2
        order by date, cast( time as integer )
```

```
    """, [ date1, date2, time1, time2 ] )

    all_rows = cursor.fetchall()

    temp2 = []
    humi = []
    inso = []
    movi = []

    for row in all_rows :

        temp2.append( float( row[ 2 ] ) )
        humi.append( float( row[ 3 ] ) )
        inso.append( float( row[ 4 ] ) )
        movi.append( int( row[ 5 ] ) )

    mv2 = np.array( [ temp2, humi, inso, movi ] )

    mr2 = corrcoefr2( mv2 )

    print( "[date1] = " + date1 )
    print( "[date2] = " + date2 )
    print( "[time1] = " + time1 )
    print( "[time2] = " + time2 )
    print( "[acti_temp] = " + str( round( mr2[ 0 ][ 3 ], 3 ) ) )
    print( "[acti_humi] = " + str( round( mr2[ 1 ][ 3 ], 3 ) ) )
    print( "[acti_inso] = " + str( round( mr2[ 2 ][ 3 ], 3 ) ) )
    print( "[temp_humi] = " + str( round( mr2[ 0 ][ 1 ], 3 ) ) )
    print( "[temp_inso] = " + str( round( mr2[ 0 ][ 2 ], 3 ) ) )
    print( "[humi_inso] = " + str( round( mr2[ 1 ][ 2 ], 3 ) ) )

    print( 80 * "-" )

    insertRow( date1, date2, time1, time2, round( mr2[ 0 ][ 3 ], 3 ), round(
        mr2[ 1 ][ 3 ], 3 ), round( mr2[ 2 ][ 3 ], 3 ), round( mr2[ 0 ][ 1 ], 3
        ), round( mr2[ 0 ][ 2 ], 3 ), round( mr2[ 1 ][ 2 ], 3 ) )

query()

queryGraphic()
```