Gilson Miranda Vargas

FarmSPL: Uma abordagem de reúso com base em uma LPS para aplicações web no domínio de Pecuária de Precisão

Campo Grande - MS

7 de março de 2017

Gilson Miranda Vargas

FarmSPL: Uma abordagem de reúso com base em uma LPS para aplicações web no domínio de Pecuária de Precisão

Dissertação apresentada ao Programa de Mestrado Stricto Sensu em Computação Aplicada, mantido pela Universidade Federal de Mato Grosso do Sul, como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada. (Área de Concentração: Tecnologias Computacionais para Agricultura e Pecuária. Linha de Pesquisa: Engenharia de Software.)

Universidade Federal de Mato Grosso do Sul – UFMS

Faculdade de Computação

Programa de Pós-Graduação em Ciência da Computação

Orientador: Profª. Drª. Debora Maria Barroso Paiva

Coorientador: Profº Msc. Camilo Carromeu

Campo Grande - MS

7 de março de 2017

Gilson Miranda Vargas

FarmSPL: Uma abordagem de reúso com base em uma LPS para aplicações web no domínio de Pecuária de Precisão

Dissertação apresentada ao Programa de Mestrado Stricto Sensu em Computação Aplicada, mantido pela Universidade Federal de Mato Grosso do Sul, como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada. (Área de Concentração: Tecnologias Computacionais para Agricultura e Pecuária. Linha de Pesquisa: Engenharia de Software.)

Trabalho aprovado. Campo Grande - MS, 10 de Fevereiro de 2017:

Profª. Drª. Debora Maria Barroso Paiva
Orientador

Profº Msc. Camilo Carromeu
Coorientador

Profa. Dra. Maria Istela Cagnin Machado
Convidado 1

Prof. Dr. Rogério Eduardo Garcia
Convidado 2

Campo Grande - MS
7 de março de 2017

Gilson Miranda Vargas
FarmSPL: Uma abordagem de reúso com base em uma LPS para aplicações web no domínio de Pecuária de Precisão/ Gilson Miranda Vargas. – Campo Grande - MS, 7 de março de 2017-
107p. : il. (algumas color.) ; 30 cm.

Orientador: Profª. Drª. Debora Maria Barroso Paiva
Coorientador: Profº Msc. Camilo Carromeu

Dissertação (Mestrado) – Universidade Federal de Mato Grosso do Sul – UFMS
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação, 7 de março de 2017.

1. Linhas de Produto de Software. 2. Repósitorio. 2. Pecuária de Precisão. I. Profª. Drª. Debora Maria Barroso Paiva. II. Universidade Federal de Mato Grosso do Sul. III. Mestrado em Computação Aplicada IV. FarmSPL: Uma abordagem de reúso com base em uma LPS para aplicações web no domínio de Pecuária de Precisão

A Deus, minha família, amigos,
professores e pesquisadores
da FACOM/UFMS e Embrapa Gado de Corte

Agradecimentos

Agradeço a Deus pela saúde, força de vontade e por todas as oportunidades que me proporciona, independente das condições ou dificuldades, Ele sempre nos oferece de alguma forma o caminho das realizações.

A minha orientadora Profª. Drª. Debora Maria Barroso Paiva, por ter paciência nos momentos mais difíceis, pela transmissão de conhecimento e ideias. Suas contribuições fornecidas por diálogos e experiências proporcionaram aprendizado e me ensinaram que a organização e perseverança superam quaisquer dificuldades.

Ao meu coorientador Profº Msc. Camilo Carromeu, que andou de lado a lado em todo processo, fornecendo sabedoria para ensinar e compreender situações das etapas do trabalho. Uma pessoa sensata e inteligente principalmente no que se compromete a lecionar.

A Profa. Dra. Fabiana Villa Alves, por me oferecer uma visão muito boa sobre o estudo de caso realizado neste trabalho, suas ideias me guiaram e possibilitaram o desenrolar do projeto.

Minha família especialmente minha mãe por estar sempre ao meu lado. Minha namorada Carolina, por ter vivenciado comigo de forma participativa e entusiasmada em todo o tempo, arrancando algum sorriso meu quando não haviam muitos motivos pra sorrir, me ajudando em momentos que eu estava triste, soube realmente demonstrar o que significa a palavra parceria.

Aos colegas da FACOM, especialmente do Grupo de Pesquisas em Engenharia de Software, Vanessa Aparecida de Moraes Weber e Olavo José Luiz Junior, por todas as ideias e trabalhos realizados juntos. Em especial também a Profa. Dra. Maria Istela Cagnin Machado, o Prof. Dr. Luciano Gonda e o Prof. Dr. Rogério Eduardo Garcia, que forneceram muito apoio na correção da dissertação e transmitiram muitas ideias que foram importantes no decorrer e no desfecho do trabalho.

A João Felipe Resende Nacer, Jairo Ricardes Rodrigues Filho e Luiz Fernando Delboni Lomba, por me ajudarem com amostragem de dados de seus trabalhos, proporcionaram uma grande ajuda.

A todos os profissionais da Embrapa Gado de Corte que tive o prazer de trabalhar e conhecer, Sérgio Raposo de Medeiros, Rodrigo da Costa Gomes, Quintino Izidio dos Santos Neto e Pedro Paulo Pires, me auxiliaram e contribuíram.

Aos colegas do programa de mestrado, sobretudo a Fabrício Yukio, Tatiane Queiroz, Delacyr Almeida Monteiro Ferreira e Leandro Feuser, companheiros de conquista.

“A felicidade não se resume na ausência de problemas,
mas sim na sua capacidade de lidar com eles.”
(Albert Einstein)

Sumário

Lista de ilustrações

Lista de tabelas

Lista de códigos-fontes

Lista de abreviaturas e siglas

EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
FACOM	Faculdade de Computação
FPS	Família de Produtos de Software
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LEDES	Laboratório de Engenharia de Software
LPS	Linhas de Produto de Software
MAPA	Ministério da Agricultura, Pecuária e Abastecimento
PIB	Produto Interno Bruto
URI	Uniform Resource Identifier
RFID	Radio-Frequency IDentification
ReST	Representational State Transfer
SPL	Software Product Line
RSSF	Redes de Sensores Sem Fio
UFMS	Universidade Federal de Mato Grosso do Sul
XML	Extensible Markup Language

Resumo

VARGAS, G. M. **FarmSPL: Uma abordagem de reúso com base em uma LPS para aplicações web no domínio de Pecuária de Precisão**. 2017. 107f. Dissertação (Mestrado em Computação Aplicada) - Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, 2017.

É notável o crescimento do agronegócio no Brasil, responsável por boa parte do PIB nacional, sendo a pecuária, área representativa desta fatia. A pecuária é de fato uma das áreas que mais crescem nacionalmente pois está aliada a processos de análise e melhoria que levam à Pecuária de Precisão. A Pecuária de Precisão é a área de estudo de soluções computacionais para resolver problemas da pecuária através de respostas previstas, impostas e contabilizadas o mais acuradamente possível. Diante dessa perspectiva, visou-se fomentar e padronizar o desenvolvimento de projetos de software para melhor planejamento de ações por meio de uma abordagem de reúso com a adoção de uma Linha de Produto de Software (LPS) para aplicações web no domínio da Pecuária de Precisão. Essa abordagem possibilitou o uso de uma estrutura comum, com componentes e serviços com as mesmas características que podem ser reutilizados em novos projetos ou produtos. Os requisitos e as informações importantes para o desenvolvimento deste trabalho foram fornecidas pelos sistemas homologados e em homologação da Embrapa Gado de Corte e o acesso a esses serviços foi provido por uma interface de comunicação unificada, baseada em ReST e HTTP. Com essa concentração de ativos quaisquer softwares que tenham comunicação podem se beneficiar com o seu reúso. O Titan *Framework* foi utilizado como repositório e repositório evoluído para o domínio da Pecuária de Precisão, promovendo a criação de softwares maneira mais simples, pela possibilidade de reúso de tecnologia. Outros benefícios são: a) economia, devido ao menor custo e reúso da estrutura; b) agilidade, por usar tecnologia com protocolo de estado; e c) eficiência, por ser portável para desktops, tablets, smartphones, dentre outros; bem como poder oferecer outros níveis de acesso.

Palavras-chave: Linhas de Produto de Software, Repósitorio, Pecuária de Precisão

Abstract

VARGAS, G. M. **A reuse approach-based on SPL for web applications to Precision Livestock domain**. 2017. 107f. Master dissertation (Master student Program in Applied Computing) - Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, 2017.

The growth of agribusiness in Brazil is responsible for a large part of the national GDP, with livestock being a representative area. Cattle ranching is in fact one of the fastest growing areas in the country because it is associated with processes of analysis and improvement that lead to Precision Livestock. Precision Livestock is the area of study of computational solutions to solve livestock problems through predicted answers, imposed and accounted for as accurate as possible. In this perspective, the aim was to promote and standardize the development of software projects to better plan actions through a reuse approach based on a Software Produce Line (SPL) for web applications in the Precision Livestock domain. This approach allowed the use of a common structure, with components and services with the same characteristics that can be reused in new projects or products. The requirements and important information for the development of this work were provided by the approved systems and approval of Embrapa Gado de Corte and access to these services was provided by a unified communication interface based on ReST and HTTP. With this concentration of assets any softwares that have communication can benefit from your reuse. The Titan Framework was used as a repository and was evolved into the Precision Livestock domain, promoting the creation of software in a simpler way, due to the possibility of reusing technology. Other benefits are: a) economy, due to lower cost and reuse of the structure; B) agility, for using technology with state protocol; and c) efficiency, being portable for desktops, tablets, smartphones, among others; As well as being able to offer other levels of access.

Keywords: Software Product Line, Repository, Precision Livestock

1 Introdução

1.1 Contextualização

O cenário do agronegócio é notável, estimado pela Confederação de Agricultura e Pecuária do Brasil (CNA), em parceria com o Centro de Estudos Avançados em Economia Aplicada (Cepea), da Esalq/USP, o Produto Interno Bruto (PIB) do agronegócio brasileiro apresentou crescimento de 0,30% em outubro, acumulando alta de 4,28% de janeiro a outubro de 2016. No mesmo período, a área da pecuária se destacou com as exportações brasileiras de carne bovina que apresentaram a soma de 914,54 mil toneladas, 4,5% acima da quantidade do mesmo período do ano anterior (CNA, 2017).

A Embrapa Gado de Corte, uma unidade de pesquisa que atende áreas de sanidade e nutrição do rebanho, melhoramento, reprodução e manejo animal, considera que, para atender desafios científicos e tecnológicos e para tornar a cadeia produtiva da pecuária de corte cada vez mais competitiva, moderna, eficiente e sustentável, é preciso gerar soluções tecnológicas inovadoras em benefício da sociedade brasileira (EMBRAPA, 2015). Essas soluções oferecem melhor gerenciamento do rebanho bovino ao produtor, através da utilização de tecnologias para auxílio na administração e tomada de decisão.

A tomada de decisão, de acordo com LACHTERMACHER (2009), pode ser definida como um processo de identificação de um problema ou de uma oportunidade e a seleção de uma linha de ação para resolvê-lo. Uma oportunidade ocorre quando circunstâncias oferecem a chance de um indivíduo ou de uma organização ultrapassar ou alterar seus objetivos ou metas. O processo para resolução de algum problema ou criação de solução está ligado a avaliação e escolhas certas, com a viabilidade de análise sobre diferentes formas e dados tangíveis. Provavelmente, um dos maiores desafios intelectuais da ciência e tecnologia está em como tomar decisões certas, dada uma situação específica (VARGAS; IPMA-B, 2010).

Para auxiliar nessas escolhas, ferramentas como softwares de apoio à decisão fornecem suporte ao gerenciamento e softwares baseados em reúso tornam-se fator importante para o alcance dos objetivos com maior qualidade, de forma mais rápida e econômica. A reutilização de software é o processo de criação de sistemas de software a partir de software existente, em vez de criar sistemas de software a partir do início (KRUEGER, 1992).

Em um ambiente atual mais dinâmico, a indústria do software está amadurecendo de acordo com a grande evolução da tecnologia da informação. Com isto, o conceito de reúso cresce e muitas técnicas têm sido desenvolvidas para apoiar a reutilização de software. Esse crescimento é devido aos fatores: sistemas no mesmo domínio de aplicação

são semelhantes e têm potencial para reutilização; a reutilização é possível em diferentes níveis para completar aplicações; e as normas para componentes reutilizáveis facilitam a reutilização (SOMMERVILLE, 2011).

O reúso de software já acontece normalmente em projetos de software. Quando as pessoas envolvidas tomam conhecimento de projetos ou códigos semelhantes ao que é exigido, elas os buscam, o modificam para atender certa demanda e incorporam a seus sistemas (SOMMERVILLE, 2011).

Empresas tem adotado práticas, métodos e técnicas de engenharia de LPS, para construção de software sob medida com menor custo e diminuição no tempo da entrega. Seguindo essa visão e tendência, surgiu a oportunidade proposta neste trabalho, de apoiar o desenvolvimento de software da Embrapa Gado de Corte, possibilitando o reúso de ativos software, para favorecer o desenvolvimento de novos projetos para o domínio de Pecuária de Precisão.

1.2 Motivação e Justificativa

A Embrapa Gado de Corte em parceria com a Faculdade de Computação (FACOM) da Universidade Federal de Mato Grosso do Sul (UFMS) realizam pesquisas para o desenvolvimento de soluções em tecnologia para atender as demandas da Pecuária de Precisão. Essas soluções são projetos em sistemas de informação que utilizam Redes de Sensores Sem Fio (RSSF) e Identificação por radiofrequência (RFID) para rastreabilidade animal por meio de identificação única, permitindo assim a adaptação dos sistemas a ambientes bucólicos e de difícil acesso, e a comunicação via transferência de dados para ambientes remotos. A comunicação se baseia no uso de dados fornecidos por sensores, que disponibilizam informações de comportamento animal por posições georreferenciadas; controle sanitário e manejo reprodutivo; dados fisiológicos por sistema de coleta de registros de peso; conforto térmico, coletado por informações ambientais de sensores de Integração Lavoura-Pecuária-Floresta (ILPF); também dados contextuais, proporcionados por sensores em cochos de alimentação.

Os sistemas desenvolvidos e em desenvolvimento na Embrapa Gado de Corte favoreceram a ideia de desenvolvimento de uma plataforma denominada "e-*Cattle*" de CARROMEU (2016), que busca a integração das tecnologias de software que são mantidas por dados de sensores. Essa plataforma descrita na Seção 2.4, fornecerá uma arquitetura com integração de sete camadas, com a comunicação desde a camada de sensores onde são coletados os dados à campo até a camada de automação que permite a tomada de decisão viabilizada pelos dados dos sensores.

Com esses dados disponíveis por esses sensores, surgiu a perspectiva de novos projetos para o domínio da Pecuária de Precisão. Com isso a ideia deste projeto, que

fornece apoio ao desenvolvimento de software, com qualidade, produtividade e baixo custo. Esse cenário foi possível de ser alcançado ao estabelecer o reúso de software e o processo que capacita a prática de reúso é o de LPS, pela construção de software sob medida, ao compartilhar um conjunto de características comuns que podem ser gerenciadas para atender determinado mercado ou cliente (NORTHROP, 2002). A abordagem utilizada neste projeto faz uso de repositório para concentração dos ativos que impulsionam o desenvolvimento de software. Esses ativos são componentes e serviços reusáveis que fornecem apoio para novos projetos ou produtos no contexto de Pecuária de Precisão.

A arquitetura citada na Seção 2.8 que foi utilizada também motivou o desenvolvimento deste trabalho, pois forneceu um atalho por ser uma arquitetura já testada e já ter uma evolução, a LPS para web no domínio e-Gov (CARROMEU et al., 2010). Isso proporcionou o desenvolvimento deste projeto com a evolução dessa arquitetura para o domínio de Pecuária de Precisão.

Há grande motivação deste projeto foi possibilitar o uso de uma estrutura, onde todos os componentes e/ou serviços puderam ser armazenados e podem reutilizados para fornecer benefícios no desenvolvimento de novos projetos de software para auxiliar o cotidiano do pecuarista.

Outra motivação para o desenvolvimento deste trabalho refere-se a contribuição científica, pelo fato de não ter sido encontrado algo semelhante para o domínio de Pecuária de Precisão.

1.3 Objetivo do Trabalho

O objetivo geral do trabalho é desenvolver uma abordagem de reúso de ativos de software denominada “FarmSPL”, com base na evolução de uma LPS para aplicações web já existente, para criação de novos ativos de software e membros para família de produtos de software para o domínio da Pecuária de Precisão.

Os objetivos específicos deste trabalhos são:

- Identificar no mercado tecnologias similares à abordagem proposta para observar os pontos positivos e negativos;
- Prover dados dos sensores dos sistemas de Pecuária de Precisão por meio de um serviço que servirá como iniciativa para camada de serviços da plataforma e-*Cattle* de CARROMEU (2016);
- Promover a sincronia de dados entre a camada de serviços de dados de Pecuária de Precisão e as aplicações de monitoramento desenvolvidas utilizando os ativos de software;

- Proporcionar o reúso de software por intermédio do desenvolvimento de componentes específicos para consulta dos dados de Pecuária de Precisão, categorizados como contextuais, comportamentais, fisiológicos e de microclima;
- Efetuar as validações ao realizar um estudo de caso utilizando os ativos de software previamente desenvolvidos.

1.4 Organização do Texto

A escrita desta dissertação está organizada em mais cinco capítulos. No Capítulo 2 são apresentados os principais conceitos e definições sobre LPS, Repositório, ReST.

Em seguida é apresentado o Capítulo 3 contendo os principais resultados da condução das buscas e um mapeamento sistemático com o levantamento de trabalhos relacionados ao tema.

Logo após no Capítulo 4 é relatada a Engenharia da LPS para identificar as comunalidades e as variabilidades e especificar a Arquitetura da LPS e sua evolução. Também é apresentado o desenvolvimento das funcionalidades levantadas na Seção 4.2.1 e o componentes da FarmSPL.

No capítulo 5, está descrito a Engenharia de Aplicação com um estudo de caso realizado para validar o trabalho e por fim, no capítulo 6, são discutidas as contribuições, limitações e trabalhos futuros.

2 Embasamento Teórico

2.1 Considerações Iniciais

Alguns conceitos foram importantes para a realização desta pesquisa e o embasamento teórico permitiu uma melhor compreensão dos fundamentos que foram utilizados neste trabalho de mestrado.

Neste capítulo é apresentada a fundamentação teórica necessária para entendimento deste trabalho, pespectivas sobre o crescimento da produção bovina; pecuária de precisão, reúso, definições de LPS, repositório, abordagens de desenvolvimento de LPS e desenvolvimento de ativos e artefatos, Titan Framework e sua reutilização e definições de arquitetura ReST.

2.2 Crescimento da Produção Bovina

O Brasil possui grande produção de carne bovina para atender a demanda nacional e internacional, prova disto, segundo o Departamento de Agricultura dos Estados Unidos (USDA, 2014), o país possui o maior rebanho para comércio no mundo. Além de ser o maior exportador de carne bovina, é o segundo maior em produção de carne. Pode-se observar, conforme demonstrado na Tabela 1 que o Brasil possui o maior crescimento no setor, ou seja cerca de 12%.

Tabela 1 – Rebanho efetivo - 2010 a 2014 (USDA, 2014)

	2010	**2011**	**2012**	**2013**	**2014** : Nov
Índia	316,400	320,800	323,700	327,100	329,775
Brasil	**185,159**	**190,925**	**197,550**	**203,273**	**208,628**
China	105,430	104,822	104,346	104,205	104,302
EUA	93,881	92,682	90,769	89,300	88,300
União Européia	89,829	87,836	87,051	87,106	87,600
Argentina	49,057	48,156	49,597	51,095	52,195
Colombia	30,845	30,971	30,910	30,576	30,150
Austrália	27,906	27,550	28,506	29,000	28,365
Russia	20,677	19,970	20,134	19,981	19,210
México	22,192	21,456	20,090	18,521	17,637
Canadá	12,670	12,155	12,245	12,305	12

Essas informações evidenciam a grande necessidade de realização de pesquisas que ofereçam ao produtor a utilização de tecnologias para auxílio na administração e tomada de decisão, para obtenção de melhores resultados no gerenciamento do rebanho bovino.

Para MACEDO (2009), a Pecuária brasileira destaca-se em cenário internacional por possuir um grande rebanho bovino e também pelo baixo índice de produtividade em comparação a outros países com tradição no setor. É enfatizado também que mesmo diante desta evolução recente da produtividade alcançada por uso de práticas modernas, a pecuária de corte bovina do Brasil ainda possui caráter extensivo, com os animais alimentando-se de pasto e, assim, sujeitos às intempéries climáticas. Essas características submetem o animal ao abate até atingir três anos, prazo superior a países de técnicas intensivas.

2.3 Pecuária de Precisão

O cenário da Pecuária é de fato um dos que mais crescem nacionalmente e, aliado a processos de análise e melhoria, a Pecuária de Precisão, que em seu conceito moderno afirma que respostas produtivas devem ser previstas, impostas e contabilizadas o mais acuradamente possível, corrobora para este crescimento (DETMANN et al., 2005).

A tecnologia ajuda em toda cadeia produtiva da carne, aumentando o lucro para o produtor através do controle de ganho de peso e rastreabilidade. Ao final deste processo espera-se obter uma carne de qualidade e com as informações da procedência (EMBRAPA, 2014).

Na Pecuária de Corte, de interesse deste trabalho, é importante ter um controle adequado dos dados do rebanho para que se possa avaliar o desempenho da atividade sob diferentes formas. Na área de pesquisa da Pecuária de Precisão vários sistemas podem ajudar o produtor a ter informações mais fiéis sobre as ações do histórico de cada animal, possibilitando a obtenção de um manejo mais eficiente (EMBRAPA, 2014).

Atualmente na Embrapa Gado de Corte, soluções em projetos de software foram e estão em desenvolvimento, para fornecer apoio ao produtor, sendo que grande maioria são mantidos por dados provenientes de sensores a campo. Esses dados fornecidos pelos sensores, foram categorizados em quatro tipos (CARROMEU, 2016):

1. **Comportamentais**: Sensores que geram dados relacionados ao comportamento de cada animal, tal como sua frequência sob sombra ou sol, sua trajetória na área de pastejo, se está ruminando ou se está deitado ou de pé. Esse tipo de informação pode ser gerada, por exemplo, por acelerômetros, sensores foto-sensíveis ou GPS. A característica básica desses sensores é que sempre estão associados a um animal, sendo normalmente instalados em seu corpo por meio de coleiras ou arreios.

2. **Microclima**: Dados relacionados ao ambiente de determinada área, tal como a temperatura de bulbo seco e de bulbo úmido[1], umidade ou incidência de radiação solar. Os sensores que obtém dados desses tipos normalmente são instalados em áreas abertas (por meio, por exemplo, de mini estações meteorológicas) ou em árvores (como redes de sensores sem fio em eucaliptos de áreas de integração lavoura-pecuária-floresta). Assim, uma característica básica deste tipo de dado é estar associado a uma coordenada geográfica.

3. **Fisiológicos**: Tratam-se de dados fisiológicos de cada animal ao longo do tempo, tal como o peso, temperatura corporal, frequência cardíaca ou frequência respiratória. Assim como os dados comportamentais, os fisiológicos são normalmente obtidos por sensores instalados no corpo do animal. Entretanto, podem ser obtidos por meio de sensores a campo que associam o valor aferido ao indivíduo. Assim, esse tipo de sensor sempre está associado a um animal identificado.

4. **Contextuais**: Sensores que aferem alterações em insumos da propriedade, tal como a quantidade de suplemento em cochos, a situação de porteiras (abertas ou fechadas), o consumo de água ou o tráfego de animais em determinado ponto de passagem. Esse tipo de sensor está normalmente associado a um insumo da infraestrutura da propriedade e, portanto, ao seu identificador e coordenada geográfica.

Esses diferentes tipos de dados contemplam um conjunto de informações para fomentar projetos de software para o domínio de Pecuária de Precisão. Isso permite ao pecuarista obter informações valiosas com avaliações diárias de acompanhamento do animal em relação às diversas análises por exemplo: a exibição de peso de cada animal representada graficamente, a rota de cada animal representada em um mapa, entre outros.

A heterogeneidade desses dados fornecidos por sensores, favoreceram a ideia do desenvolvimento da plataforma e-*Cattle* (CARROMEU, 2016).

2.4 Plataforma e-*Cattle*

Por meio da parceria entre Facom/UFMS e Embrapa Gado de Corte, está sendo desenvolvida uma plataforma denominada "*e-Cattle*" de CARROMEU (2016), que busca a integração das tecnologias de software que são mantidas por dados de sensores que os coletam de forma autônoma e não uniforme. Na Figura 1 é apresentada a arquitetura da plataforma e-*Cattle*, que busca integrar diversas camadas, que são nomeadas de "a" a "g", e estão definidas como:

[1] Temperatura de bulbo seco é medida com um termômetro comum, já a de bulbo úmido é indicada pelo termômetro cujo bulbo está mergulhado numa mecha úmida. Essas medidas são usadas para o cálculo da Umidade Relativa ou do Ponto de Orvalho (SCHNEIDER, 2008).

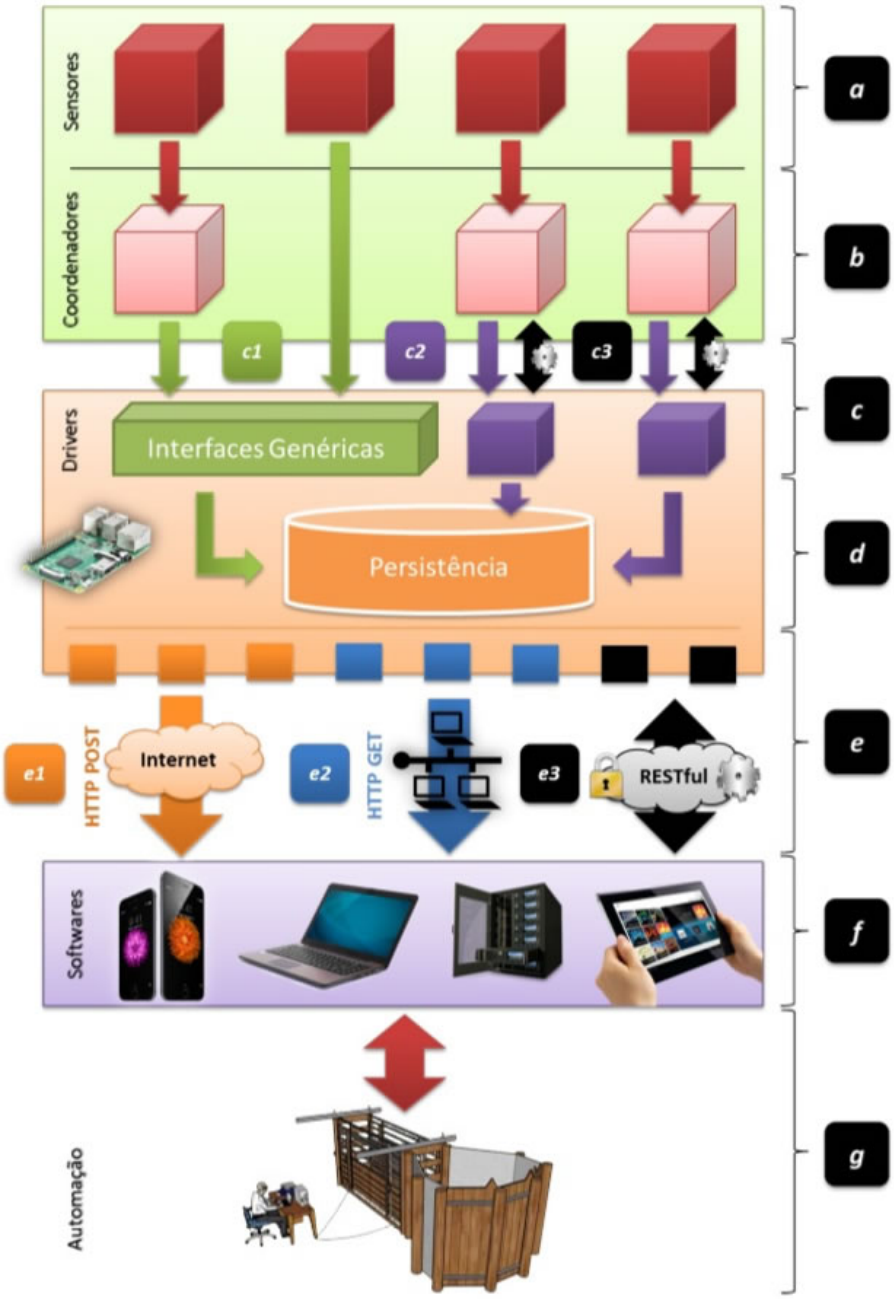


Figura 1 – Arquitetura e-*Cattle* (CARROMEU, 2016)

a **Camada de sensores**: a primeira e segunda camada da arquitetura referem-se aos componentes do produto de coleta de dados á campo (artefatos de hardware e software embarcado), composto por sensores e interfaces de comunicação para aferir variáveis: ambientais, fisiológicas, comportamentais e do que pode ser medido (como níveis de insumo), para após enviar o dado coletado a camada de drivers;

b **Camada de coordenadores**: denominados dispositivos responsáveis pelo envio do dado coletado para o barramento de *drivers*. São dispositivos auxiliares ou rotinas implementadas por meio de software embarcado nos próprios sensores;

c **Camada de *drivers***: é composta por *webservices* no padrão RestFul e aceita requisições do método POST. As URIs desta camada são providas em função do tipo de dados passíveis de serem recebidos (item c1). Esta camada pode ser estendida permitindo a inclusão de novos tipos (item c2). Também há a possibilidade de alteração de configuração da gestão do coordenador e do sensor (item c3);

d **Camada de persistência**: Dados recebidos da camada de *drivers* são persistidos nesta camada utilizando conceito de banco de dados "chave-valor" (conceito NoSQL, não relacional). Cadas tipo de sensor possui sua própria identidade de armazenamento. Um exemplo de cenário é o de pesagens, onde todas as pesagens são agrupadas na mesma entidade, independente de terem sido obtidas por diferentes modelo de balanças.

e **Camada de serviços**: é composta por *webservices* no padrãoo RESTful e aceita requisições por meio de métodos HTTP. Suas características são: a) pode ser configurada para enviar dados para a nuvem (item e1); b) provê URIs de acesso aos dados armazenados conforme o tipo (item e2); e c) implementa uma conexão autenticada para gestão de sensores (item e3).

f **Camada de software**: onde estão localizados os softwares de alto nível que fazem uso dos dados dos sensores. Dentre iniciativas de desenvolvimento, além deste trabalho, com a criação dos componentes para apoio de novos membros o domínio de Pecuária de Precisão com base nos tipos de dados de sensores, softwares da "e-Gov *Mobile*" SPL (SC02), discutido na Seção 3.5 também encontram-se nesta camada;

g **Camada de automação**: provê artefatos de hardware, permite que a tomada de decisão viabilizada pelos dados dos sensores que vieram do primeiro nível da arquitetura se transformem em açõe reais que afetam o manejo da propriedade.

A plataforma *e-Cattle* abriu lacunas para diversos projetos com a responsabilidade de representar uma ou mais de uma camada, alguns sendo coordenadores, ou seja, vários projetos poderiam estar inseridos na camada a e alguns destes na b também. E assim posteriormente esses projetos serão integrados para compor essa plataforma.

A integração dos dados e as possibilidades fornecidas pelas camadas contempladas por essa plataforma beneficiam o pecuarista. Apesar dos dados serem gerados dos sensores, podem ser cruzados para auxiliar na tomada de decisão, por conta disso, a última camada é a de automação. Estando em condição de acessar os dados disponíveis em um repositório, as possibilidades de sistemas (membros da LPS) são inúmeras, e essa perspectiva se torna ainda mais favorável com a aplicação de reúso de software, ao promover o desenvolvimento de novos softwares com menos esforço e custo.

2.5 Reúso de Software

A Conferência de Engenharia de Software da OTAN de 1968 marca o surgimento da Engenharia de Software (RANDELL; NAUR, 1968). Essa conferência centrou-se na crise de software, com o objetivo de resolver o problema de construção de sistemas de software grandes e confiáveis de uma forma controlada e econômica e, desde seu início, a reutilização de software tem sido apontada como um meio para superar essa crise (KRUEGER, 1992).

O reúso de software é mais eficaz quando está previsto como parte de um programa de reúso de toda a organização. Um programa de reúso envolve a criação de ativos reusáveis e a adaptação de processos de desenvolvimento para incorporar esses ativos no novo software (SOMMERVILLE, 2011). O processo de reutilização é uma tendência que motiva o aumento da produção e viabiliza o desenvolvimento de uma família de sistemas. Segundo PARNAS; CLEMENTS; NORTHROP (1976, 2002), é vantajoso estudar propriedades comuns aos programas antes de analizar membros individuais, Isso favorece e traz muitos benefícios, como a redução no tempo de desenvolvimento e o aumento de qualidade nos resultados.

Em termos gerais, um processo de reutilização de software deve definir ao menos duas atividades essenciais: o desenvolvimento para reutilização, que consiste no desenvolvimento dos artefatos de software que serão posteriormente reutilizados, e o desenvolvimento com reutilização, que consiste no desenvolvimento de aplicações que reutilizam os artefatos previamente desenvolvidos (LUCRÉDIO, 2009). Esses pontos que são fundamentais para um processo de LPS.

2.6 Linhas de Produtos de Software (LPS)

A terminologia Linha de Produto de Software (do inglês *Software Product Lines* - SPL), segundo REINHR et al. (2007) é utilizada desde o final do século XIX, com a introdução da revolução do processo produtivo industrial, sugerindo processo de desenvolvimento sequencial de produtos com base em tarefas repetitivas e especializadas. Na comunidade européia utilizava-se o termo reúso efetivo, como abordagem de Engenharia de Família de Produtos (*Product Families Engineering* - PFE). O termo americano, foi evidenciado com a união das duas principais conferências do setor a: *Product Family Engineering*, da Europa, e *Software Product Lines*, dos EUA, que optaram pelo título: *International Conference on Software Product Lines* (LPSC) (REINHR et al., 2007).

Segundo COHEN (2002) a definição de LPS de maior uso na indústria é a de CLEMENTS; NORTHROP (2002). Esses autores definem LPS como: "um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de características comuns e gerenciadas, que satisfazem as necessidades de um segmento particular de mercado ou missão, e que são desenvolvidas a partir de um conjunto comum de ativos principais, de uma forma preestabelecida".

DURSCKI et al. (2004) detalham que o processo de uma LPS sugere dividir o produto em pequenas partes, mais compreensíveis e gerenciáveis (ativos); apresentar especificações claras e sem ambiguidades para produção / montagem (forma preestabelecida) e atender a um domínio específico (segmento particular ou missão) para que, como em uma linha de montagem da indústria tradicional, cada novo produto seja uma reutilização de partes de produtos construídos anteriormente, pertencentes à mesma família e, preferencialmente, com um mínimo de esforço adicional de construção.

Conforme NORTHROP (2002), as LPSs usam em seu conceito uma arquitetura comum e flexível, com possibilidade de customização rápida e efetiva para diferentes aplicações referentes a domínios específicos. Com isto conseguem vantagens econômicas sobre o uso do que há de características comuns e variáveis. O processo ilustrado na Figura 2, demonstra que uma LPS: a) possui um conjunto de produtos que são construídos para um domínio de aplicação ou uma estratégia de mercado; b) compartilham uma arquitetura; e c) utilizam componentes e serviços.

Os ativos do núcleo ou ativos principais são aqueles artefatos reutilizáveis e recursos que formam a base para a linha de produtos de software. Os ativos principais incluem, mas não estão limitados a, arquitetura, componentes de software reutilizáveis, modelos de domínio, documento de requisitos, documentação e especificações, modelos de desempenho, agendas, orçamentos, planos de teste, casos de teste, planos de trabalho e descrições de processo. A arquitetura é fundamental para o armazenamento dos ativos principais (NORTHROP et al., 2007).

Figura 2 – Linhas de Produtos de Software (CLEMENTS; NORTHROP, 2002)

Os ativos tornam-se vantajosos com a possibilidade de reúso para resolver um problema em particular, uma vez que raramente especialistas propõem alguma "nova solução". Eles apoiam-se em experiência própria ou de outros profissionais (CARROMEU et al., 2010). Ao iniciar um projeto de negócio há princípios a serem considerados, com a objetivos e conformidades e o porquê de aquele projeto ter início. Ao considerar a possibilidade de reúso, devem ser demonstrados em seus objetivos os benefícios que esperam alcançar através do projeto ou linha de produto. Uma LPS, no que diz respeito a produtividade e desempenho, promove em seu elenco alguns benefícios tangíveis, mostrados na Tabela 2.

Tabela 2 – Benefícios obtidos com uma LPS (COHEN, 2003)

Fatores	**Benefícios para organização usando abordagem de Linhas de Produto**
Lucratividade	O repositório de ativos permite que a organização produza produtos voltados para um segmento específico de mercado. O benefício é observado no aumento da participação de mercado e na lucratividade da organização.
Qualidade	A redução no número de defeitos reportados é comum em sistemas desenvolvidos em LPS. A qualidade também pode ser medida em termos de redução do tempo para reparo.
Desempenho	A utilização de ativos aumenta o desempenho em relação ao desenvolvimento tradicional, especialmente com o aumento da maturidade da LPS, o que faz com que os ativos estejam cada vez mais otimizados.

Tempo de Integração	O tempo de integração no desenvolvimento incremental é facilitado.
Volume de código	A quantidade de objetos projetados para subsistemas utilizando ativos é menor do que os estimados para sistemas individuais, resultando em redução semelhante no tamanho real de código-fonte.
Produtividade	A equipe de desenvolvimento, o custo e o cronograma são reduzidos.

A essência de uma LPS é a reutilização, que é conquistada através dos seus ativos reutilizáveis. Um ativo pode ser composto por um ou mais artefatos.

Definido pelo *Reusable Asset Specification* (RAS) da OMG (2005), os artefatos são quaisquer produtos de trabalho do ciclo de desenvolvimento de software, como documentos de requisitos, modelos, arquivos de código-fonte, descritores de implantação, casos de teste ou *scripts* e assim por diante. Em geral, o termo "artefato" está associado a um arquivo, e é um produto de trabalho que pode ser criado, armazenado e manipulado por produtores ou consumidores de bens e por ferramentas.

A Figura 3 ilustra a definição geral de ativos reutilizáveis. Demonstra que os ativos reutilizáveis fornecem uma solução para um problema para um determinado contexto. O ativo pode ter um ponto de variabilidade, que é um local no ativo que pode ter um valor fornecido ou personalizado pelo consumidor de ativos. O ativo possui regras de uso que são as instruções que descrevem como o recurso deve ser usado.

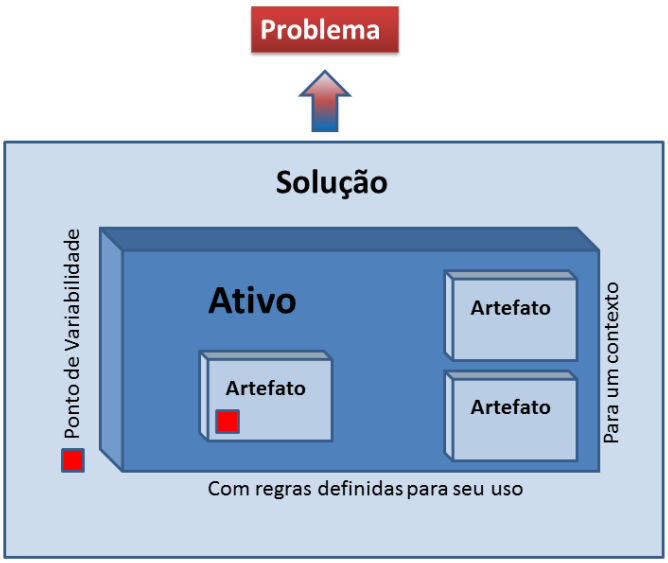


Figura 3 – Definição de Ativo Reutilizável (OMG, 2005)

2.6.1 Variabilidade

Segundo POHL; BÖCKLE; LINDEN (2005) em um domínio específico de negócio existe as características comuns (comunalidades) e as que podem ser alteradas (variabilidades).

A variabilidade é a capacidade de alterar ou personalizar um sistema e é determinada pela característica de produto que pode ser adaptada ao uso. Uma **variabilidade** é composta por um **ponto de variação** e suas **variantes** (POHL; BÖCKLE; LINDEN, 2005).

POHL; BÖCKLE; LINDEN (2005) definem ponto de variação como uma representação de uma variabilidade dentro de artefatos de um domínio com suas informações contextuais e variante como uma representação de um objeto de variabilidade dentro de artefatos de domínio.

Conforme apresentado na Figura 4 um ponto de variação é uma representação local onde a variação pode ocorrer (por exemplo, cor do veículo) e as variantes são as possíveis soluções que existem para o ponto de variação, ou seja, conjunto de possíveis instâncias de variação que um ponto de variabilidade poderá originar (por exemplo, amarelo, verde, azul e vermelho) (POHL; BÖCKLE; LINDEN, 2005).

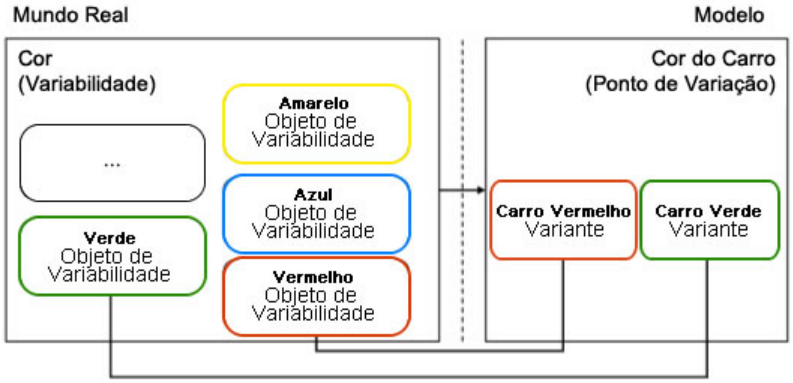


Figura 4 – Relação entre variabilidade e ponto de variação (em um modelo do mundo real) - Adaptada de POHL; BÖCKLE; LINDEN (2005)

Para representar as variabilidades e as características comuns dos sistemas em uma linha de produtos é utilizado o modelo de *features* (KANG et al., 1990).

2.6.2 Modelo de *Features*

Segundo CZARNECKI; EISENECKER (2000), *feature* pode ser vista como uma funcionalidade ou propriedade da LPS que permite distinguir membros de uma família e capturar comunalidades e variabilidades em famílias de sistemas e linhas de produtos. O modelo de *features* é elaborado durante a análise de domínio e uma das mais utilizadas

abordagens de gestão de variabilidades é a *Feature-Oriented Domain Analysis* (FODA) (KANG et al., 1990).

A representação de um modelo de *features* ou modelo de características normalmente tem uma estrutura de uma árvore, dada por conexões e folhas que são chamadas de recursos ou *features*, quanto mais detalhe o modelo tiver, mais produtos podem surgir (CZARNECKI; EISENECKER, 2000).

No exemplo do modelo de *features* da Figura 5, são descritas as comunalidades e as variabilidades de um carro simples. A partir dele é possível se obter 12 configurações distintas para um carro (duas transmissões diferentes, três tipos de motor e um engate opcional, ou seja, 2 x 3 x 2 = 12). No desenho da árvore há uma raiz ou retângulo inicial que indica o conceito que é modelado, os outros abaixo indicam que possuem dependência ao elemento que está acima. Todos os elementos são diferentes e descrevem restrições sobre como as *features* poderão ser combinadas, por exemplo: um carro não tem obrigatoriamente um engate e o motor pode ser a gasolina, elétrico ou ambos.

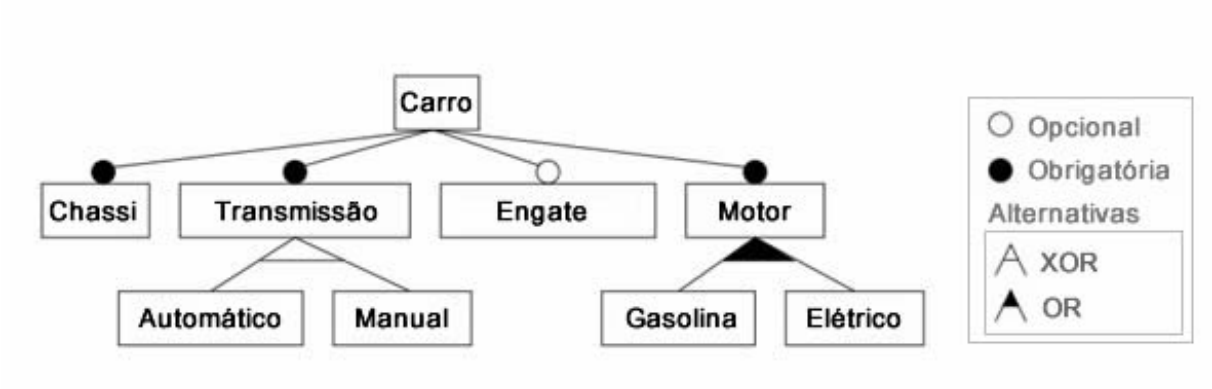


Figura 5 – Modelo de *features* de um carro - Adaptada de CZARNECKI; EISENECKER (2000)

Quando uma *feature* é selecionada as outras acima conectadas também devem ser selecionadas. Estas conexões são representadas pelos seguintes tipos:

- ***Mandatory*** (obrigatórias): a funcionalidade deve estar em todos os produtos;
- ***Optional*** (opcionais): uma funcionalidade que pode ou não estar em um produto, depende de sua seleção pelo engenheiro da aplicação para que seja incluída no produto final;
- ***Alternatives*** (XOR | OR):
  - XOR : exatamente uma das funcionalidades deve estar no produto, ou seja, representam funcionalidades mutuamente exclusivas; ou

– OR: pelo menos uma das funcionalidades deve estar no produto, ou seja, representam funcionalidades mutuamente inclusivas;

Uma LPS, apesar de trazer muitos benefícios, mas também apresenta dificuldades, como alto custo no início do desenvolvimento da linha de produtos. Porém, após a definição da LPS, os benefícios superam os custos (NORTHROP, 2002). Outra dificuldade inicial é a escolha da abordagem mais indicada para desenvolver a LPS e algumas serão citadas a seguir.

2.7 Abordagens de Desenvolvimento de LPS

Dentre várias abordagens existentes, neste projeto serão citadas duas, que são de maior interesse: a proposta por NORTHROP (2002), denominada *Product Line Practice* (PLP) uma proposta bastante citada na literatura; e a proposta por GOMAA (2005), *Product Line UML-Based Software Engineering* (PLUS), uma abordagem baseada em Unified Modeling Language (UML) que fornece um processo facilitado para a engenharia de LPS, além de já ter sido utilizado para aplicar um processo LPS em CARROMEU (2007), que serviu como base, fornecendo informações do uso da arquitetura para este trabalho.

2.7.1 *Product Line Practice* (PLP)

Na abordagem PLP existem três atividades que são essenciais à constituição de uma Linha de Produto: desenvolvimento de ativos ou engenharia de domínio, desenvolvimento de produtos ou engenharia de aplicação e o gerenciamento da linha de produto. Estas atividades são representadas na Figura 6 e descritas a seguir.

Na atividade de **Desenvolvimento de Ativos** do Núcleo (do inglês *Core Assets Development*), ou Engenharia de Domínio, os artefatos que compõem os ativos da linha e a arquitetura comum são desenvolvidos: componentes, *frameworks*, serviços, dentre outros. Para isso são realizadas as atividades de análise de domínio, projeto de domínio e implementação de domínio.

Na atividade de **Desenvolvimento de Produtos** (Engenharia de Aplicação), os produtos são construídos seguindo o plano de produção. Nessa atividade também é aplicada a derivação (seleção e configuração dos ativos do núcleo), em que pode existir interação com a atividade anterior (Desenvolvimento de ativos), pois requisitos do domínio podem não ter sido especificados a priori.

Na atividade de **Gerenciamento**, visa-se a garantia de que todas as atividades sejam executadas de maneira coordenada e planejada. Também tem a possibilidade de

Figura 6 – Atividades essenciais de PLP, adaptado de CLEMENTS; NORTHROP (2002)

estabelecer como ocorrerá a evolução da linha de produtos, ou seja, como as outras duas fases devem interagir.

2.7.2 *Product Line UML-Based Software Engineering* (PLUS)

O processo utilizado pelo PLUS é evolucionário e possui duas atividades principais, uma com a determinação de várias versões da linha de produto, identificada pela Engenharia de domínio, outra atividade, a Engenharia de Aplicação, onde se gera um produto (aplicação), com as *features* desejadas pelos usuários e cliente. Esse processo é demonstrado na Figura 7.

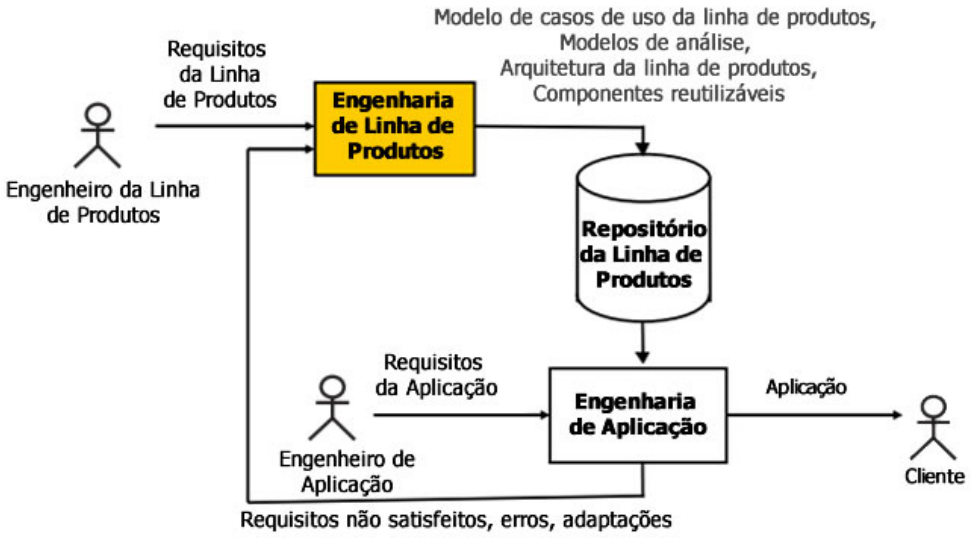


Figura 7 – *Evolutionary Software Product Line Engineering Process* - Adaptada de GOMAA (2005)

Para a engenharia de domínio e aplicação, existe um processo evolucionário deno-

minado ESPLEP (*Evolutionary Software Product Line Engineering Process* - ESPLEP). As fases do processo ESPLEP são apresentadas na Figura 8 e divide-se em:

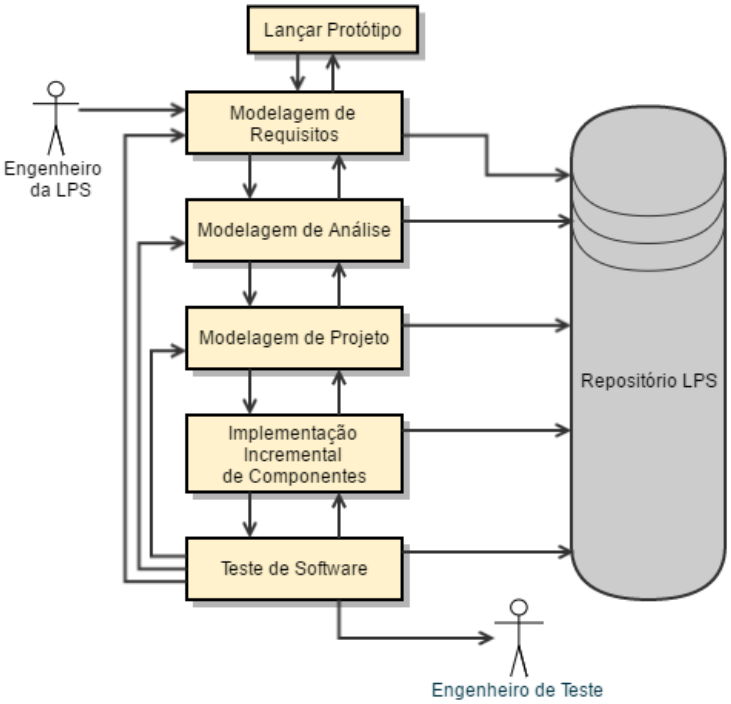


Figura 8 – Processo Evolucionário de Linha de Produto de Software - Adaptada de GOMAA (2005)

1. **Modelagem de Requisitos**: onde é definido o escopo da LPS, documentação de requisitos, modelo de casos de uso e modelo de *features*;

2. **Modelagem da Análise**: modelo estático (relações estruturais) e modelo dinâmico (desenvolvido a partir dos casos de uso, mostra os objetos que participam e como eles interagem);

3. **Modelagem do Projeto**: é responsável por definir a arquitetura de referência da LPS;

4. **Implementação Incremental de Componentes**: um subconjunto dos requisitos do domínio da LPS é selecionado para ser implementado em cada iteração. Inicia-se com a seleção de um caso de uso do núcleo e a implementação de casos de uso opcionais e alternativos, de acordo com a sequência definida no modelo dinâmico. Cada implementação consiste no detalhamento da arquitetura, codificação, e teste de unidades de componente;e

5. **Teste de Software**: esta fase consiste na realização dos testes de integração de cada componente reutilizável.

2.8 Titan *Framework*

Para uma LPS a arquitetura para estruturar e satisfazer a Engenharia de Domínio é realizada a partir de um repositório, que é essencial para sistematizar um processo de reutilização. A existência de ferramentas automatizadas e adoção de um repositório encurta a distância nesse processo.

O Titan é um *Framework*[2] que foi criado e desenvolvido pelo grupo de pesquisa LEDES/FACOM/UFMS para ser utilizado na para gestão de conteúdos em websites, portais e *intranets*, integrando ferramentas necessárias para criar, gerir (editar e inserir) conteúdos em tempo real, sem a necessidade de programação, estruturando e facilitando a criação, administração, distribuição, publicação e disponibilidade da informação. Segundo CARROMEU (2007), o Titan *Framework* foi construído com uso de tecnologias de software livre.

Segundo CARROMEU et al. (2010), a arquitetura do Titan *framework* é formada por um núcleo (*core*) e pelo repositório de componentes, conforme ilustrado na Figura 9. O núcleo é a implementação de todas as similaridades e variabilidades da LPS e, neste sentido, tem como característica ser imutável, indiferente da configuração da instância no domínio.

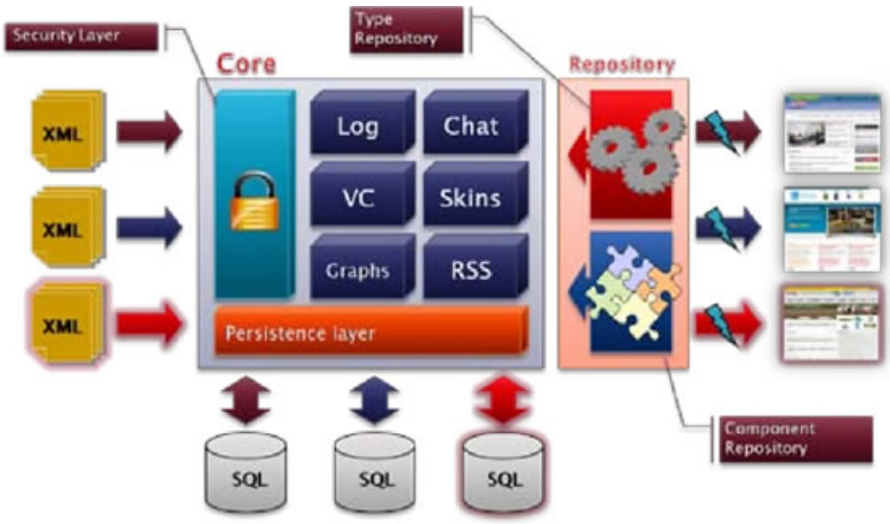


Figura 9 – Arquitetura do Titan Framework (CARROMEU et al., 2010)

O Titan *Framework* implementa o repositório LPS no formato de um repositório de componentes que permite o reúso de tipos de dados, funcionalidades e *templates* de código. Estes componentes são as variabilidades da LPS que podem ser parametrizadas através da linguagem de marcação de entrada *eXtensible Markup Language* (XML) (CARROMEU et al., 2010).

[2] JOHNSON; WIRFS-BROCK (1991) definem como um esqueleto de classes, objetos e relacionamentos agrupados para construir uma aplicação específica.

As funcionalidades do Titan são organizadas por meio de um modelo de navegação baseado em “seções” e “ações”. Uma seção é uma funcionalidade da aplicação, tal como a gestão de notícias de um site. As ações são as atividades atômicas que os usuários podem realizar naquela funcionalidade, tal como criar, editar e visualizar notícias (CARROMEU, 2014). Essas ações são controladas por políticas de segurança.

O Titan *Framework* possui segurança baseada em processos de decisão de controle de acesso, uma política de controle de acesso baseada em função (do inglês Role *Based Access Control - RBAC*). As decisões de controle de acesso são muitas vezes determinadas pelas funções que os usuários individuais assumem como parte de uma organização. Isso inclui a especificação de deveres, responsabilidades e qualificações (FERRAIOLO; CUGINI; KUHN, 1995).

Dentre algumas fortes justificativas que fizeram escolher o Titan *Framework*, além da maturidade adquirida em projetos desenvolvidos na FACOM/UFMS, possui características de: - possuir uma arquitetura única, com foco total em reúso; - possuir um repositório de artefatos parametrizáveis (com componentes, tipos de dados, *templates* de código, elementos de *layout*, ferramentas e *drives*). Desta forma, boa parte da programação, tal como a definição de modelos de dados, pode ser feita por meio de linguagem de marcação, no caso XML, o que auxilia muito a manutenção corretiva e evolutiva de funcionalidades da aplicação.

Neste projeto de mestrado, foram reutilizados alguns artefatos do Titan *Framework*, citados na Seção 4.2.3.3, além do uso de seu repositório e instanciação de seu componente *global.generic* e evolução para criação dos ativos para o domínio de Pecuária de Precisão.

A Embrapa Gado de Corte possui soluções específicas para atender demandas da Pecuária de Precisão. Essas soluções são relacionadas a sensores que fornecem as informações em forma de dados brutos, sem tratamento. Para que esses dados brutos se transformassem em alto nível para serem utilizados como informações para os ativos desenvolvidos, precisaram ser tratados, armazenados e recuperados por meio de *webservices*.

Web Services são sistemas de software projetados para suportar a interação interoperável máquina-a-máquina através de uma rede (BOOTH et al., 2004). Essas interações acontecem por troca de mensagens transportadas via requisições HTTP com uma serialização XML. Para a transmissão dessas mensagens, existem duas estruturas de empacotamento que são mais utilizadas, SOAP e ReST, que foi utilizada para neste trabalho.

2.9 ReST

Representation State Transfer (ReST) é um estilo arquitetural híbrido derivado de vários estilos arquitetônicos baseados em rede combinado com restrições adicionais que

definem uma interface de conector uniforme (FIELDING, 2000).

As restrições adicionadas ao estilo híbrido são da arquitetura cliente-servidor. São separadas as responsabilidades da interface das responsabilidades de armazenamento de dados. Com isso melhora também a portabilidade da interface do usuário em várias plataformas e melhora a escalabilidade simplificando os componentes do servidor, permitindo que cliente e servidor se desenvolvam de forma independente e que a arquitetura de ambos suporte os requisitos de aplicação de múltiplos domínios organizacionais na escala da internet (FIELDING, 2000).

Portanto, não há armazenamento pelo lado do servidor, o estado da sessão é mantido inteiramente no cliente, por isso o protocolo deriva de "Sem estado". A Figura 10 mostra requisições sem estado do cliente ao servidor.

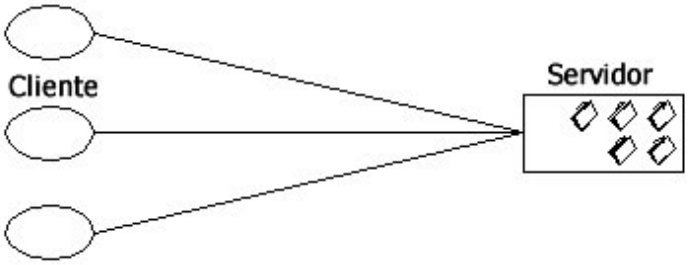


Figura 10 – Arquitetura Cliente-servidor Sem Estado(FIELDING, 2000)

A arquitetura ReST usa um identificador de recurso para identificar um recurso específico relacionado em uma interação entre os componentes. Um exemplo de um identificador de recursos é o URI (*Unified Resource Identifier*). A representação de um recurso não precisa ser igual ao próprio recurso. Por exemplo, o recurso associado com o estado de "reserva de um restaurante" terá diferentes representações dependendo de quando a representação for recuperada. Uma representação geralmente é recuperada executando uma requisição por HTTP utilizando o método "GET" em um URI. Normalmente a resposta para essa requisição é devolvida serializada em forma de dados em formato, *JavaScript Object Notation* (JSON) ou XML.

Para este trabalho foi utilizado o formato de notação JSON, por ser um formato leve baseado em Javascript (menos *tags* que o XML) e de rápida comunicação. O resultado normalmente é representado por uma coleção de pares nome:valor, como mostrado no Código de fonte 2.1.

```
{
 "data":[
          {
            "id":982000088459895,
            "raca":"Nelore",
            "peso":"189.00",
            "dataHora":"23/10/2013 14:13:56"
          },
          {
            "id":982000088459896,
            "raca":"Hereford",
            "peso":"202.00",
            "dataHora":"23/10/2013 16:10:56"
          }
        ]
}
```

Código-Fonte 2.1 – Exemplo de notação em JSON

2.10 Considerações Finais

Através dos estudos citados objetivou-se conhecer melhor a agropecuária brasileira e as necessidades da Pecuária de Precisão e assim foi escolhido a melhor forma de atender as demandas por meio dos estudos revistos. Dentre esses estudos, o entendimento de terminologias e conceitos de LPS, foram fundamentais para iniciar a elaboração desta pesquisa, pois o trabalho consiste em prover uma abordagem que forneça suporte a uma fase de desenvolvimento para uma LPS.

O próximo capítulo é apresentado um mapeamento sistemático para investigar estudos de ferramentas que oferecem apoio a alguma das fases de uma LPS, e foram importantes para observação de estudos primários para apoio a este trabalho.

3 Ferramentas de apoio de LPS: Mapeamento Sistemático

3.1 Considerações Iniciais

Para que seja feita uma boa seleção de textos científicos sobre algum tema de pesquisa é preciso ter um bom método de busca e análise. Com base nessa boa prática, foi sugerida a realização de um mapeamento sistemático.

Segundo KITCHENHAM; CHARTERS (2007), um mapeamento sistemático permite a identificação de grupos e faltas de evidências para direcionar o foco para futuras revisões sistemáticas e para identificar áreas para a condução de novos estudos primários.

O mapeamento sistemático realizado foi baseado no processo embasado em KITCHENHAM (2004), que divide o mapeamento em três fases:

- **Planejamento**: fase onde são definidas as necessidades da pesquisa e o protocolo do mapeamento sistemático da pesquisa;
- **Condução**: fase de identificação dos estudos primários e os mesmos são avaliados de acordo com os critérios de inclusão e exclusão estabelecidos;
- **Extração dos Dados** e **Análise dos Resultados**: nesta fase é realizado o levantamento das informações após a seleção dos estudos primários e a respostas para as questões de pesquisa.

3.2 Planejamento

3.2.1 Objetivos da pesquisa

Em um mapeamento sistemático é realizada a investigação do estado da arte. A pesquisa é baseada nas palavras-chaves relevantes, escolhidas de acordo com a problemática da pesquisa.

O presente capítulo detalha um mapeamento sistemático que foi realizado no contexto de Linhas de Produto de Software para o domínio de Pecuária de Precisão. Após a pesquisa realizada e a base quantitativa elencada, foram estabelecidas a abrangência e as proximidades dos resultados no contexto Linhas de Produto de Software para o domínio de pecuária de precisão, para assim considerar uma base qualitativa.

3.2.2 Questões Relevantes da Pesquisa

KITCHENHAM; DYBA; JORGENSEN (2004) comentam que uma das razões para formular a questão é precisamente ajudar pesquisadores e profissionais a encontrar todos os estudos relevantes. A partir dessa premissa e da definição dos objetivos, busca-se encontrar a maior quantidade possível de resultados para compreender e unir evidências sobre estudos existentes sobre Linhas de Produto de Software para o domínio de Pecuária de Precisão. Então para tal demanda foi definida a questão de pesquisa (**QP**):

QP1: **Quais trabalhos apresentam: ferramenta, abordagem, processo ou técnica de LPS no domínio de Pecuária de Precisão**?

(a) Quais os detalhes de implementação que a solução apresenta?

(b) Quais fases de LPS a solução apresenta?

(c) Qual apoio computacional a solução apresenta?

3.2.3 Construção das *Strings* de Busca

A construção das *strings* de busca foi definida de acordo com a especificidade do trabalho: Linha de Produto de Software e Pecuária de Precisão. Foram definidas *strings* genéricas de busca as quais foram adaptadas de acordo com a base de busca pesquisada.

Tabela 3 – Termos relevantes para construção de *string* padrão, sinônimos.

Linhas de Produto de Software		**Pecuária**
Linhas de Produto	**Software**	**Pecuária de Precisão**
SPL, "SOFTWARE PRODUCT LINE", "PRODUCT LINE", "PRODUCT FAMILY", "PRODUCT FAMILIES", PRODUCT-LINE, PRODUCT-FAMILY, PRODUCT-FAMILIES, REPOSITORY	SOFTWARE, SYSTEM	FARM, CATTLE, "PRECISION LIVESTOCK", "PRECISION AGRICULTURE"

A construção das *strings* de busca, apresentada na Tabela 3, se apoiou nos conceitos chave dos termos relacionados às três áreas investigadas no contexto de:

1. **Linhas de Produto de Software**: linha/família de produto de software e repositório;

2. ***Software***: software e sistemas;

3. **Pecuária de Precisão**: fazenda, gado e pecuária de precisão.

Para construção das *strings* foram adicionadas as *strings* "PRODUCT-LINE", "PRODUCT-FAMILY" e "PRODUCT-FAMILIES", pois quando utilizadas as palavras hifenizadas os resultados eram maiores que sem o uso delas, como ocorreu por exemplo na base de pesquisa IEEE (*IEEE Xplore Digital Library*).

Do cruzamento destas palavras-chave, o resultado gerado é a *string* de busca apresentado na Tabela 4:

Tabela 4 – String padrão de busca em inglês do mapeamento sistemático

Cruzamento das *Strings* de termos relacionados a Linhas de Produto de Software, *Software* e Pecuária de Precisão
(SPL OR "PRODUCT LINE" OR "PRODUCT FAMILY" OR "PRODUCT FAMILIES" OR PRODUCT-LINE, PRODUCT-FAMILY OR PRODUCT-FAMILIES OR REPOSITORY) AND (SOFTWARE OR SYSTEM OR FARM) AND (FARM OR CATTLE OR "PRECISION LIVESTOCK" OR "PRECISION AGRICULTURE")

3.2.4 Seleção de Bases de Pesquisa

Para a seleção das bases de pesquisa foram elencadas as principais delas, utilizadas em revisões sistemáticas na área de Engenharia de Software que são: ACM (ACM *Digital Library*), IEEE (IEEE *Xplore Digital Library*), *Web of Science* e *Science Direct / Elsevier* (DYBÅ; DINGSØYR; HANSSEN, 2007). As bases *Scopus* e *Scielo* que possuem grande acervo e importância para buscas em Engenharia de Software foram adicionadas à pesquisa. Foram levadas também em consideração bases que possuem conteúdo atualizado, qualidade e facilidade na exportação dos resultados. As bases selecionadas estão representadas na Tabela 5.

Tabela 5 – Bases de buscas internacionacionais

Bases Pesquisadas	
ACM Digital Library	http://dl.acm.org/
IEEE Xplore Digital Library	http://ieeexplore.ieee.org/
ScienceDirect / Elsevier	http://www.sciencedirect.com/
Scopus	http://www.scopus.com/

Web of Science	http://apps.webofknowledge.com/
Scielo	http://www.scielo.org/

3.2.5 Critérios de Inclusão e Exclusão

Refinar o resultado da busca dos estudos primários é importante para se conseguir uma amostragem adequada. Para tal demanda foi criada a tabela de Critérios de Inclusão (**CI**) como modo de incluir os estudos primários relevantes no contexto do mapeamento e os Critérios de Exclusão (**CE**), com objetivo de excluir os estudos primários irrelevantes. A Tabela 6 apresenta os Critérios de Inclusão (**CI**) e a Tabela 7 apresenta os Critérios de Exclusão (**CE**):

Tabela 6 – Critérios de Inclusão (CI) de busca do mapeamento sistemático

Critérios de Inclusão (CI)	
CI1	Estudos que apresentam ferramenta, abordagem, processo ou técnica de LPS no domínio de Pecuária de Precisão.
CI2	Estudos que apresentam fase(s) de ciclo de desenvolvimento de linhas de produto de software.

Tabela 7 – Critérios de Exclusão (CE) de busca do mapeamento sistemático

Critérios de Exclusão (CE)	
CE1	Estudo publicado fora do período de 2006 a 2016.
CE2	Estudo que não contemple satisfatoriamente os critérios de inclusão.
CE3	Estudo que não esteja em inglês ou em português.
CE4	Estudo incompleto, indisponível ou similar a outros já elencados.
CE5	Estudo não publicado em anais de eventos, em revistas, como relatórios técnicos, capítulos de livros, teses ou dissertações.

3.3 Condução da Busca

A condução do mapeamento sistemático seguiu após a definição do protocolo de busca, sendo dividida em duas etapas identificação dos estudos e seleção dos estudos.

Na primeira etapa, munido das *strings* definidas a priori e utilizando a sintaxe para cada base de pesquisa de acordo com sua especificidade, foi realizado o processo de busca.

A identificação das strings para esta etapa está representada no Apêndice A, onde se pode visualizar cada base por uma sigla e ID.

Para a execução das buscas foram consideradas as configurações de pesquisa em algumas bases de dados utilizando: título, palavras-chave e *abstract*, e em outras de acordo com a quantidade de registros de retorno foram considerados também os outros filtros, definidos nos critérios de exclusão, como: período de 2006 a 2016 pelo critério de exclusão **CE2** e idioma em português ou inglês pelo **CE3**.

Os resultados das buscas foram catalogadas no software de apoio computacional *Mendeley Desktop* (MENDELEY, 2014) para que com isto houvesse uma melhor organização para consulta dos trabalhos e filtragem dos estudos primários.

Depois de concluídas as buscas foram retornados 161 estudos, como mostra a sumarização dos resultados na Tabela 8.

Tabela 8 – Sumário dos resultados por pesquisa

Sigla	**Bases de Busca**	**Resultados**
ACM	ACM *Digital Library*	6
IEEE	IEEE *Xplore Digital Library*	11
SCD	ScienceDirect / Elsevier	20
SCO	Scopus	115
WS	Web of Science	9
SCI	Scielo	0
TOTAL		**161**

Utilizando as *strings* definidas para Linha de produto de software, software e pecuária de precisão, a consulta na base de busca **ACM *Digital Libary*** retornou **6** registros. A **IEEE *Xplore Digital Library*** retornou **11** trabalhos. A base de busca **Science Direct/Elsevier** retornou um total de **20** registros. Para este resultado foi considerado na pesquisa as opções de Título, Palavras-Chave e Resumo. A ***Scopus***, foi base de dados que mais retornou estudos foi utilizada a pesquisa por Título, Palavras-Chave e Resumo, foram encontrados **115** trabalhos.

A busca realizada na base de busca ***Web of Science*** obteve o resultado de **9** registros. Por último, a busca feita na base de dados ***Scielo*** não retornou registros.

Após a realização da etapa de identificação dos dados, o resultado serviu como base para a segunda fase do mapeamento sistemático, a de seleção dos estudos, que consistiu na aplicação dos critérios de inclusão e critérios de exclusão. Todos os **161** estudos resultantes foram avaliados por Título, Resumo e Palavras-chave, data de publicação e idioma, e com isto foram excluídos 155 registros, 49 pelo CE1, 100 pelo CE2, 4 pelo CE3 e 3 pelo CE4, conforme sumarizado na Tabela 9.

Tabela 9 – Sumário da seleção depois de aplicados critérios de exclusão

		Critérios de Exclusão					
Sigla	**Resultados**	**CE1**	**CE2**	**CE3**	**CE4**	**CE5**	**Total**
ACM	6	-2	-3	-	-	-	1
IEEE	11	-2	-7	-	-	-	2
SCD	20	-6	-14	-	-	-	0
SCO	115	-37	-74	-	-2	-	2
WS	9	-2	-2	-4	-1	-	0
SCI	0	-	-	-	-	-	0
Total	161	-49	-98	-4	-3	-	5

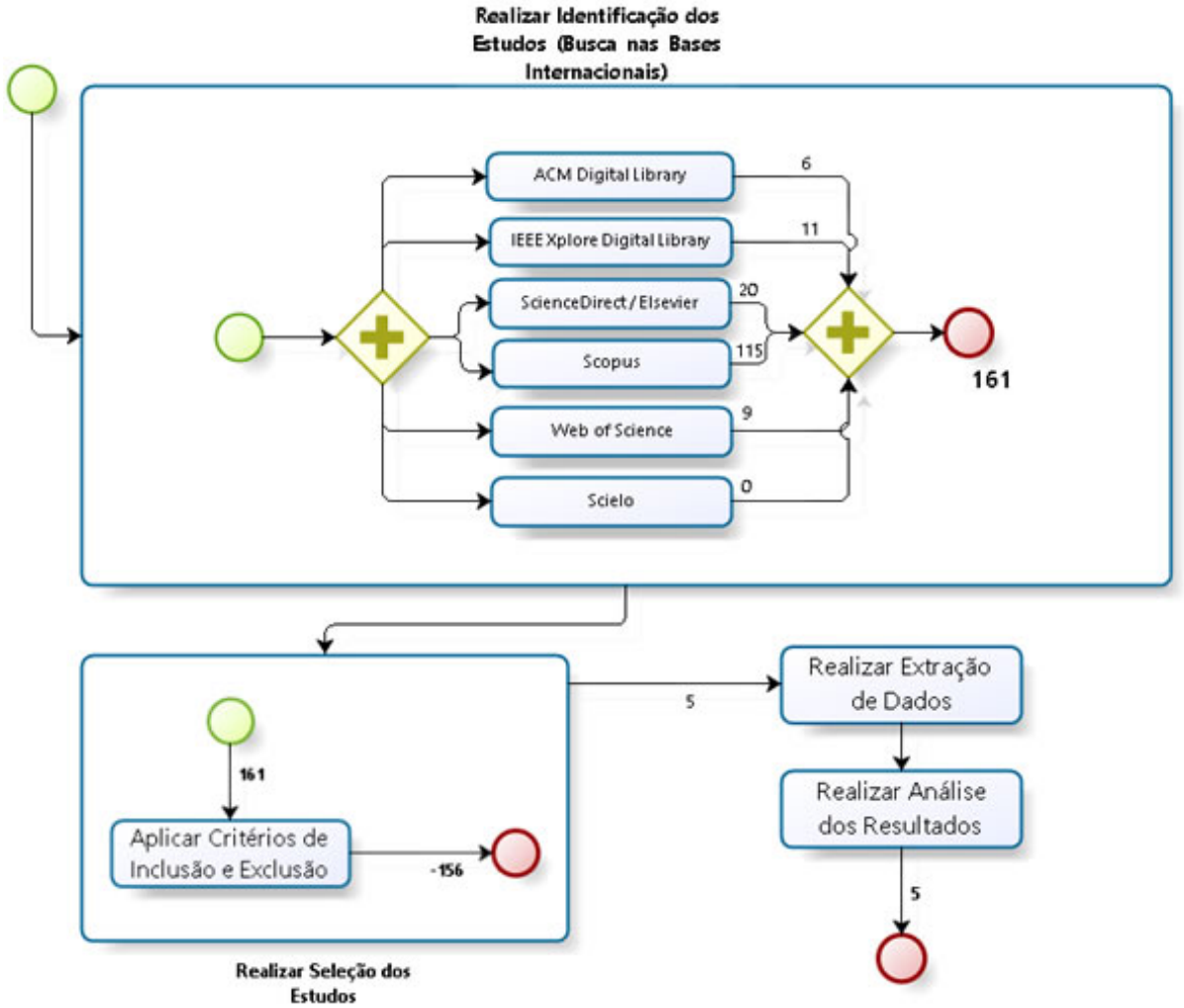


Figura 11 – Processo de Busca: Condução da Busca, Extração dos Dados e Análise dos Resultados

A Figura 11 ilustra o processo de busca, onde estão incluídas as fases 2 e 3 deste mapeamento, sendo a fase 2, da Condução da Busca identificada pelos processos:

- **Realizar Identificação dos Estudos**: Realização das buscas nas bases internacionais; e
- **Realizar Seleção dos Estudos**: fase de aplicação dos Critérios de Inclusão e Exclusão.

A grande maioria dos estudos resultantes, no caso cinco, contemplam o critério de inclusão 2 (CI2), ou seja, descrevem alguma fase de ciclo de desenvolvimento de manutenção de linhas de produto de software. Após esses resultados, sumarizados no Apêndice A, foi satisfeita a necessidade de conteúdo para a execução da fase de Extração de Dados e Análise de Resultados.

3.4 Extração dos Dados

A extração de dados consiste no levantamento de informações obtidas dos estudos após a seleção de dados. Esta seleção consiste nos estudos resultantes apresentados na Tabela 10.

Tabela 10 – Resultado da busca em bases de pesquisas

ID	Título	CI
ACM Digital Library		
ACM1	*MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications*	CI2
ACM2	*Context Variability Modeling for Runtime Configuration of Service-based Dynamic Software Product Lines*	CI2
IEEE Xplore Digital Library		
IEEE1	*Variability Modeling for Service Oriented Product Line Architectures*	CI2
Scopus		
SCO1	*The Feature-Architecture Mapping (FArM) method for feature-oriented development of software product lines*	CI2
SCO2	*The evolution from a web SPL of the e-gov domain to the mobile paradigm*	CI2

Depois de selecionados os estudos, foram aplicadas as perguntas do formulário para extração de dados, apresentado na Tabela 11. Esse formulário foi elaborado para auxiliar a responder as questões de pesquisa do mapeamento sistemático por fornecer dados do Título, Autor(es), Ano, Publicação, entre outros.

Tabela 11 – Formulário para Extração de Dados

Formulário de Extração	
Título	
Autor(es)	
Ano	
Publicação	
Tipo de Solução (Abordagem, Ferramenta, Framework, etc...)	
Descrição resumida da solução	
Fases de LPS da solução	( ) Desenvolvimento de Ativos ou Engenharia de domínio ( ) Desenvolvimento do Produto ou Engenharia de aplicação
Apoio Computacional:	( ) Apoio ao Desenvolvimento ( ) Manutenção ( ) Evolução ( ) Derivação/Configuração
Benefícios da Solução?	
Limitações da Solução?	
Possui Validação?	( ) Sim ( ) Não

3.5 Análise dos Resultados

A análise dos resultados foi feita através dos resultados obtidos após a extração dos dados dos estudos selecionados. Dos cinco trabalhos selecionados, todos se encaixam no CI2, apresentam uma ou mais fases de ciclo de desenvolvimento e/ou manutenção de linhas de produto de software, oferecem algum suporte para desenvolvimento para engenharia de domínio ou engenharia de aplicação.

Embora todos atendam o CI2, nenhum trabalho se encaixa aos anseios da pesquisa em sua completude, não atendem ao CI1, pois não se encaixam no modelo proposto, o de ferramenta, abordagem, processo ou técnica de LPS ligada ao domínio de pecuária de precisão.

A seguir são apresentados os estudos que se encaixam no CI2 e dão apoio ao desenvolvimento deste projeto.

ACM1

O artigo "***MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications***", de MARINHO et al. (2013), apresenta uma abordagem para o desenvolvimento de software móvel sensível ao contexto

usando Linha de Produto de Software (LPS) como paradigma. A LPS para o domínio de aplicativos móveis foi dividida em três ciclos:

1. No primeiro ciclo são identificadas as comunalidades e as variabilidades, gera-se a base para o modelo de *features*, modelo de classes e quaisquer componentes reutilizáveis. O software UbiFEX-*Notation* foi usado na fase de projeto e apoia a fase de gerência de domínio, ao identificar as comunalidades e variabilidades;
2. Este ciclo compreende a Engenharia de Domínio, com a elicitação de requisitos de uma família de aplicações para um domínio específico. Um Guia de Visita para dispositivo móvel(***Mobile Visit Guide***) foi usado para especificar o modelo de *features* para o domínio. Foram executadas duas ações: i) identificação dos requisitos específicos deste domínio, e ii) seleção de *features* relevantes do para gerar a base do primeiro ciclo. E para identificar as características do domínio, foram consideradas três pesquisas;
3. O último ciclo, o que corresponde à Engenharia de Aplicação LPS, um produto para um guia de visita é derivado da LPS.

A abordagem do artigo apresenta duas coberturas de LPS, engenharia de domínio e de aplicação. Na engenharia de domínio, foi utilizado um software de notação para gerar o modelos de *features*, utiliza um repositório de contexto, e na engenharia de aplicação foi derivado um produto específico para visitas.

ACM2

MURGUZUR et al. (2014) propõe no artigo intitulado "***Context Variability Modeling for Runtime Configuration of Service-based Dynamic Software Product Lines***" uma modelagem de variabilidade de contexto, que é demonstrada sua aplicabilidade e usabilidade através de um caso de uso de parques eólicos e apresentando os blocos de construção fundamentais de uma estrutura para permitir a variabilidade de contexto em Linhas de Produtos de Software Dinâmico Baseados em Serviço.

A estratégia utilizada se baseia na separação de dois modelos de *features* (modelos de variabilidade) - um para as propriedades do sistema da família de produtos (isto é, não relacionadas ao contexto e variabilidade do processo) e o outro modelo, um sistema principal de modelo de *features*, que classifica e representa apenas as propriedades do contexto.

A nível de engenharia de domínio, são discutidos os principais requisitos para suportar a variabilidade de contexto, onde foram abordados os problemas de modelagem de variabilidade de contexto de parques eólicos e foi sugerido uma variabilidade dinâmica

como forma de antecipar a evolução de mudanças quando dados ou novas *features* de contexto exigirem uma modificação da variabilidade estrutural.

Há uma especificação da abordagem e utilização do *toolkit* **LateVa** que suporta variabilidade em tempo de execução e requisitos de serviços baseados em processos. Acredita-se que a integração natural da variabilidade baseada em processos com *features* de contexto pode simplificar o gerenciamento da adição/remoção ou alteração das propriedades de contexto.

IEEE1

O artigo "***Variability Modeling for Service Oriented Product Line Architectures***", de ABU-MATAR; GOMAA (2011), enfatiza a necessidade que os sistemas SOA tem de gerenciar a variabilidade para atender às diferentes necessidades dos clientes. Como as LPS's promovem o desenvolvimento reutilizável para famílias de produtos, a abordagem discutida usa conceitos LPS para modelar sistemas SOA como famílias de serviços. Utiliza uma modelagem de variabilidade de múltiplas visões que aborda as considerações de variabilidade com base em um *framework* de metamodelagem de variabilidade SOA.

Apresenta diferenças entre SOA e LPS, cita que o processo de reutilização em LPS é feito em todas as fases do ciclo de vida de desenvolvimento usando todos os tipos de ativos, e em SOA, geralmente apenas os serviços são reutilizados.

A abordagem utilizou um modelo *multiple-view* para modelagem de variabilidade de serviço e então após, foram definidas as relações entre os elementos de modelagem de serviços. Finalmente, foi desenvolvido um modelo de variabilidade de serviço que relaciona *features* do modelo de *features* para serviços do modelo de serviço. Para criar o primeiro membro da família foi apresentada uma seleção para uma aplicação básica de e-commerce, e também apresentou um modelo do processo e o modelo de *features* para esta aplicação.

SC01

O artigo de SOCHOS; RIEBISCH; PHILIPPOW (2006), intitulado "***The Feature-Architecture Mapping (FArM) method for feature-oriented development of software product lines***", apresenta um método de mapeamento de *features* para arquitetura, utiliza um modelo de *features*, construído inicialmente e produzido por um método de análise de domínio, (KANG et al., 1990)). Para derivar componentes arquitetônicos intimamente relacionado com *features*. O objetivo da FArM é fornecer um forte mapeamento entre as *features* do modelo de *features* e da arquitetura da linha de produto.

O FArM se baseia em obtenção de requisitos não funcionais, para simplificar a análise de domínio. Para o caso desses requisitos, algumas resoluções são feitas através de resolução direta, integração em funcionalidades funcionais existentes e adição de novos recursos funcionais. É citado como exemplo uma arquitetura de Autenticação HTTP que refere-se à autenticação necessária por vários sites. Para reduzir o tráfego de rede, os arquitetos adicionam um recurso de autenticação para armazenar os nomes de usuário e senhas necessários e passá-los automaticamente para sites que precisam ser autenticados.

SC02

O artigo intitulado "***The evolution from a web SPL of the e-gov domain to the mobile paradigm***" de CARROMEU; PAIVA; CAGNIN (2015), apresenta uma evolução de uma LPS para web no domínio e-Gov (LPS Web para e-Gov) para uma LPS *mobile* no domínio e-Gov (LPS *Mobile* para e-Gov), tendo em mente a necessidade de atender à demanda do mercado.

A evolução da LPS no domínio e-Gov foi motivada pelo crescente uso de dispositivos móveis e a disseminação de redes sem fio, pelo domínio e-Gov ser uma das principais áreas de aplicação.

O artigo discute os principais resultados obtidos através de uma instanciação da LPS *mobile*.

Utiliza uma técnica de reúso de Linhas de Produtos de Software (LPS) automatizada através do uso de *frameworks* e geradores de aplicação. O processo de LPS criado considera um projeto baseado em componentes e é conduzido pela abordagem PLUS (*Product Line* UML-*Based Software Egineering*), que possui dois ciclos de vida conforme descrito na Seção 2.7.2 deste trabalho. A arquitetura da LPS utiliza o **Titan *Framework*** também descrita neste trabalho na Seção 2.8.

A sincronização dos dados para aplicações móveis são fornecidos pelas aplicações Web. Para isto foi necessário a implementação de um *Service Bus* (barramento de serviço), relacionados à comunicação e segurança, e desacoplados dos componentes do repositório da LPS.

No primeiro ciclo de evolução foram apresentados: modelo de *features*, o barramento de serviços com os serviços básicos representado pela *feature AuthenticationLayer* com o *Embrapa-Auth*, para garantia de que qualquer instância LPS conte com qualquer combinação dos níveis de autenticação deste protocolo apresentado na Figura 12. Os componentes do repositório foram adaptados para interagir com a nova camada de comunicação (*Communication feature*). As classes que implementam tipos de dados do Titan Framework adquiriram a capacidade de representar e entender valores no formato JSON.

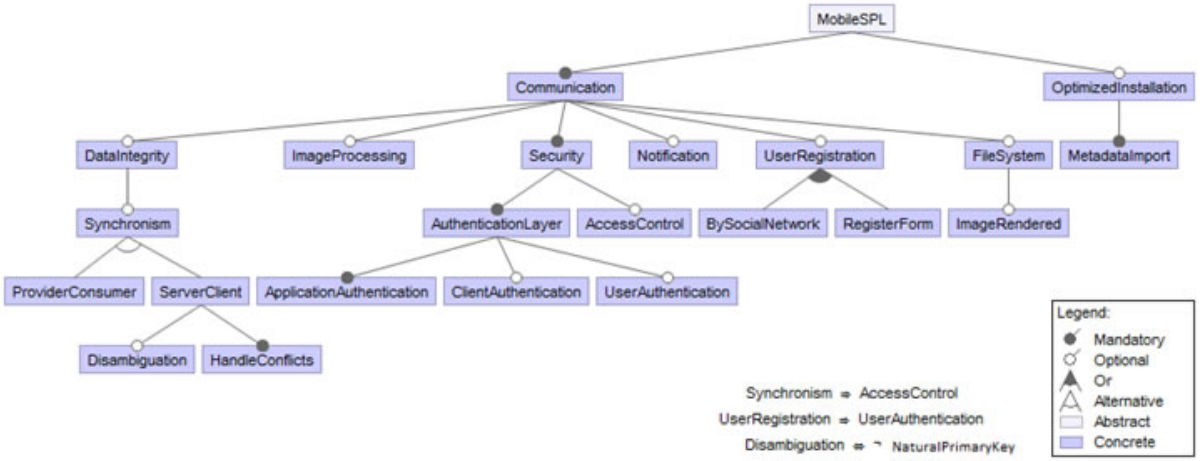


Figura 12 – Modelo de *features* da evolução da LPS para Web no domínio e-Gov para LPS *mobile* no domínio e-Gov (CARROMEU; PAIVA; CAGNIN, 2015)

Para apresentar os resultados da evolução, foi desenvolvida uma aplicação Web, com o suporte do LPS Web para e-Gov ilustrada na Figura 13 e, em seguida, uma aplicação *mobile* com suporte da LPS *mobile* para e-Gov, conforme Figura 14, cujo objetivo é fornecer uma lista de embaixadas brasileiras para viajantes estrangeiros.

Figura 13 – Aplicação Web de embaixadas brasileiras para viajantes estrangeiros (CARROMEU; PAIVA; CAGNIN, 2015)

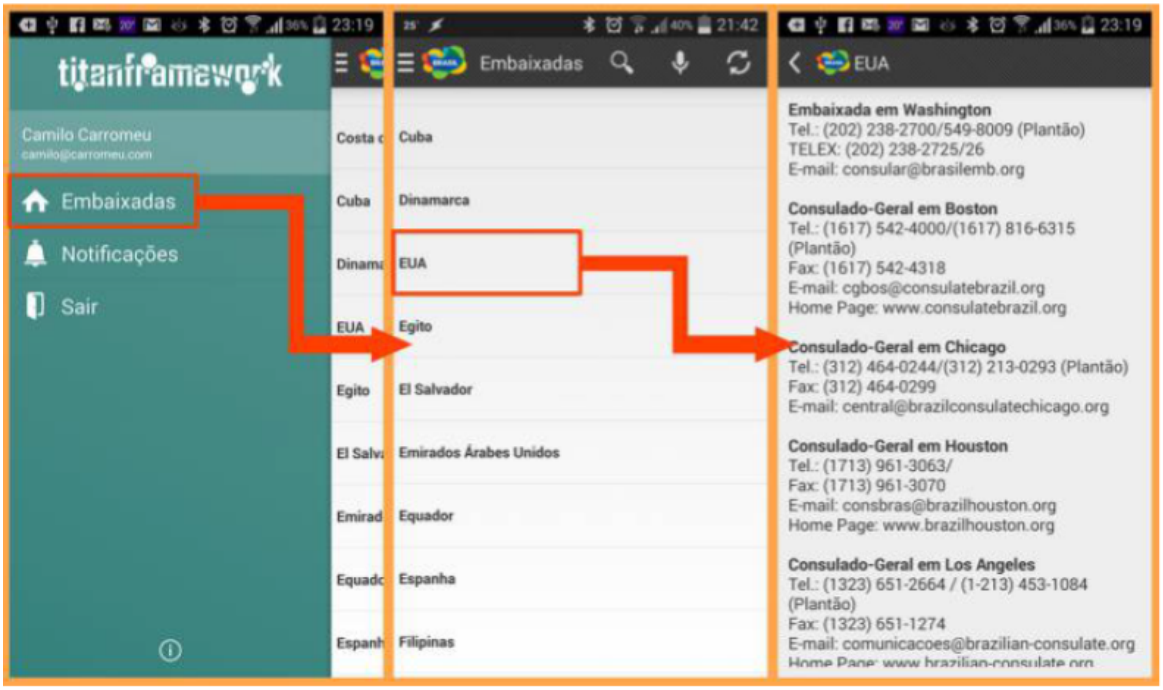


Figura 14 – Aplicação *Mobile* de embaixadas brasileiras para viajantes estrangeiros (CARROMEU; PAIVA; CAGNIN, 2015)

3.5.1 Discussão

Os resultados obtidos pelo mapeamento sistemático permitiu a identificação de soluções na literatura para apoio ao desenvolvimento deste projeto. A partir da análise dos estudos primários é possível realizar uma discussão. A Tabela 12 apresenta uma sumarização com os estudos analisados e sua características.

Por intermédio desses resultados foi possível encontrar benefícios e limitações. Os benefícios observados nos trabalhos encontrados é que o a maioria oferece soluções ou parte delas para apoio tanto no desenvolvimento quanto na evolução de LPS.

Algumas características são comuns entre as soluções, todas contemplam o **CI2**, ou seja, possuem fases de Engenharia de domínio e Engenharia de aplicação. Também foi observado que todas oferecem algum suporte computacional ao desenvolvimento e possuem algum tipo de validação.

Tabela 12 – Estudos analisados relevantes a pesquisa

			Fases		Apoio Computacional			
ID	**Ano**	**Tipo da Solução**	**Engenharia de Domínio**	**Engenharia de Aplicação**	**Desenvolvimento**	**Manutenção**	**Evolução**	**Validação**
ACM1	2015	Abordagem, Ferramenta	•	•	•		•	•
ACM2	2014	Abordagem	•	•	•			•
IEEE1	2011	Abordagem	•	•	•			•
SCO1	2006	Método	•	•	•		•	•
SCO2	2015	Abordagem, Framework	•	•	•	•	•	•

As soluções **ACM1** (MURGUZUR et al., 2014), **SCO1** (SOCHOS; RIEBISCH; PHILIPPOW, 2006) e **SCO2** (CARROMEU; PAIVA; CAGNIN, 2015) oferecem apoio computacional a evolução da LPS, destacando a evolução da LPS.

Destaca-se a solução de CARROMEU; PAIVA; CAGNIN (2015) que é uma evolução de LPS Web para arquitetura *mobile*, e também fornece apoio computacional a manutenção por sua arquitetura ser baseada no *framework* Titan, que possibilita a configuração de regras de negócios e parâmetros de novas aplicação por meio da edições da linguagem de marcação.

3.6 Considerações Finais

Através da aplicação do mapeamento sistemático, dos 161 trabalhos identificados incialmente, não foi encontrado trabalho que conseguisse responder a QP1 em sua particularidade, ou seja, que provesse ferramenta, abordagem, processo ou técnica de LPS no domínio de Pecuária de Precisão, valorizando cientificamente este projeto. Quanto as subquestões da QP1, que podem ser respondidas pelo CI2, foram selecionados e analisados 5 estudos, que ajudaram em um entendimento superior no funcionamento de ciclos de vida de LPS, assim como metamodelos e modelagem de *features*.

A avaliação dos resultados foi feita de acordo com o tipo de suporte oferecido, qualidade da ferramenta e se oferece algum apoio ao desenvolvimento deste projeto.

A solução utilizada por CARROMEU; PAIVA; CAGNIN (2015) é a que possui maior completude e ajudou no desenvolvimento deste projeto, que se baseou na evolução da LPS Web para e-Gov para LPS Web para domínio de Pecuária de Precisão.

4 Aplicação do processo de LPS no domínio de Pecuária da Precisão

4.1 Considerações Iniciais

Neste capítulo é apresentada uma contextualização do projeto, a aplicação do processo da LPS, especialmente a identificação de comunalidades e variabilidades, e a arquitetura de referência obtida da LPS.

4.1.1 Visão Geral

Os diferentes tipos de dados dos sistemas de informação que utilizam RSSF da Embrapa Gado de Corte contemplam um conjunto de informações para fomentar projetos de software para o domínio de Pecuária de Precisão. Conforme foi descrito na Seção 2.4, esses dados serão integrados pela camada de serviços da plataforma "*e-Cattle*" de CARROMEU (2016). Este projeto teve o apoio dessa plataforma para ser desenvolvido e surgiu com a oportunidade de oferecer um processo de desenvolvimento de softwares para uma camada específica dessa plataforma, a camada de software **f** mostrada na figura 1. É fornecido por este trabalho a aplicação de um processo de LPS, por uma abordagem de reúso por meio de desenvolvimento de software baseados em ativos de software, mas especificamente componentes e/ou serviços de consulta de dados.

Os componentes fornecidos foram relacionados aos tipos de dados e cada um dos quatro produzidos serve como base para tomada de decisão ao desenvolver um software para o contexto de Pecuária de Precisão. Isso permite ao desenvolvedor mais agilidade por já ter algo previamente desenvolvido que pode ser reutilizado e menor custo na produção de um novo membro da LPS.

Para que este trabalho fosse desenvolvido, foi necessário fazer inicialmente uma elicitação e análise de requisitos, para melhor compreensão dos dados desses sensores e para identificar com qual trabalho cada um deles se relaciona.

4.2 Engenharia da LPS

Conforme representado na Figura 8 e descrito na Seção 2.7.2, a Engenharia da Linha de Produtos no processo *Evolutionary Software Product Line Engineering Process* (ESPLEP) é dividida em cinco fases: Modelagem de Requisitos, Modelagem da Análise, Modelagem do Projeto, Implementação Incremental de Componentes e Teste de Software, sendo que em cada fase modelos são produzidos e armazenados no Repositório da LPS.

4.2.1 Modelagem de Requisitos

A modelagem de requisitos é iniciada através da pesquisa sobre o domínio, incluindo entrevistas com o cliente/especialista do domínio para observação das funcionalidades para definição dos requisitos da LPS. A atividade de modelagem de requisitos gera um artefato de saída que é o documento de requisitos da LPS com os requisitos obrigatórios, opcionais e alternativos. Este documento possui uma visão geral do que foi desenvolvido, os requisitos funcionais[1] e os requisitos não funcionais[2], e as variabilidades dos componentes e sistema.

Como o objetivo deste projeto foi construir componentes para reutilização para projetar softwares para Pecuária de Precisão, também foi necessário realizar a validação inicial para cada um deles. Essa validação foi concretizada através da construção de funcionalidades de monitoramento de dados que são fornecidos pelos sensores que a plataforma *e-Cattle* oferecerá.

Para elaborar a modelagem dos requisitos foram levantadas informações sobre quais os sensores utilizados nos projetos da Embrapa Gado de Corte e quais os tipos de dados que fornecem. Com isso foi possível analisar os dados para atender as necessidades e geração dos componentes.

Para essa demanda, foi criado um formulário genérico para coletar as informações de onde seriam fornecidos esses dados para logo após conseguir amostras reais coletadas em campo. Um exemplo de coleta desses dados pode ser visualizada na Tabela 21 no Apêndice B.

Depois que esses dados foram levantados, foi criada uma planilha com os detalhes de todos os sensores utilizados nos sistemas desenvolvidos e em desenvolvimento na Embrapa Gado de Corte, para posteriormente obter uma amostra de dados de quatro projetos, cada um com dados de tipos diferentes, que foram tratados e armazenados em um base de dados, o banco de dados PostgreSQL (versão 9.5)[3]. Uma imagem dessa planilha é ilustrada na Figura 39, no Apêndice F.

Essa amostragem de dados permitiu que fossem criadas validações para os componentes e foram sugeridas para cada tipo de componente duas funcionalidades listadas na Visão Geral da Linha. Perceba em seguida que essas funcionalidades são Requisitos Funcionais, porém são apenas para título de validação dos componentes, que podem ser utilizados para inúmeras abstrações. Por exemplo: o componente para dados fisiológicos pode ser reutilizado para monitorar peso, frequência cardíaca, temperatura corporal, entre

[1] São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações (SOMMERVILLE, 2011)

[2] São restrições aos serviços ou funções oferecidas pelo sistema (SOMMERVILLE, 2011)

[3] Gerenciador de Banco de Dados Relacional de código aberto (do inglês *open-source*): <https://www.postgresql.org/>

outros, o que limita é o dado que é fornecido. Dentre essas funcionalidades a de monitorar índice de temperatura e umidade é mostrada detalhadamente como estudo de caso posteriormente no Capítulo 5.

Visão Geral da Linha (VGL)

1. Os componentes produzidos devem fornecer ao desenvolvedor a possibilidade de obter instâncias da LPS no domínio de Pecuária de Precisão. As categorias que os componentes atenderão são para dados comportamentais, contextuais, fisiológicos e de microclima;

2. Os dados dos sensores devem ser concentrados e fornecidos já em alto nível via *webservice*, como um barramento de serviços;

3. O **sistema** deve fornecer serviço de agendamento de atualização que possibilite a sincronia de dados utilizados pelos componentes e o fornecedor os dados, no caso o barramento;

4. O **sistema** pode utilizar os dados **comportamentais** para monitorar os **rastreamentos** dos animais através da movimentação e coordenada geográfica;

5. O **sistema** pode utilizar os dados **comportamentais** para monitorar os horários que o animal está **pastejando no sol**;

6. O **sistema** pode utilizar os dados **contextuais** para monitorar a **visitas ao cocho** dos animais contabilizando a quantidade de visitas diária;

7. O **sistema** pode utilizar os dados **contextuais** para monitorar a **variação de temperatura local do cocho**;

8. O **sistema** pode utilizar os dados **fisiológicos** para monitorar o **peso** dos animais exibindo a variação do peso dos animais durante o dia;

9. O **sistema** pode utilizar os dados **fisiológicos** para monitorar o conforto térmico dos animais através do **índice de temperatura animal** exibindo a variação durante o dia;

10. O **sistema** pode utilizar os dados de **microclima** para monitorar a **temperatura bulbo úmido** ao longo do dia;

11. O **sistema** pode utilizar os dados de **microclima** para monitorar o índice que mede **temperatura do ar** e exibir a sua variação do durante o dia;

Requisitos Funcionais (RF)

1. O **sistema** deve permitir ao **administrador do sistema** as ações de inclusão, alteração e remoção de agendamento de atualização (*Scheduler*) para os tipos de dados comportamentais, contextuais, fisiológicos e de microclima, com possibilidade de intervalos por mês, semana, dia, hora e minuto;

2. O **sistema** deverá executar os processos de agendamentos de atualização dos dados utilizados pelos componentes de acordo com os intervalos cadastrados;

3. O **sistema** poderá permitir ao **administrador do sistema** visualizar quais animais possuem registros de **rastreamentos**;

4. O **sistema** poderá permitir ao **administrador do sistema** visualizar quais animais possuem registros de **pastejo no sol**;

5. O **sistema** poderá permitir ao **administrador do sistema** visualizar quais animais possuem registros de **visitas ao cocho** de alimentação e quais cochos foram visitados;

6. O **sistema** poderá permitir ao **administrador do sistema** visualizar quais animais possuem registros capturados de **temperatura local do cocho** de alimentação e a variação de temperatura;

7. O **sistema** poderá permitir ao **administrador do sistema** visualizar quais animais possuem registros de **pesagens**;

8. O **sistema** poderá permitir ao **administrador do sistema** visualizar registros de **índice de temperatura e umidade** e a variação conforto térmico durante o dia;

9. O **sistema** poderá permitir ao **administrador do sistema** visualizar registros de capturados de **temperatura de bulbo úmido** e a variação durante o dia;

10. O **sistema** poderá permitir ao **administrador do sistema** visualizar registros capturados de **temperatura do ar** e a variação durante o dia;

11. O **sistema** deve permitir o **administrador do sistema** pesquisar os dados por Data Inicial e Data Final antes de serem exibidos pelos componentes;

Requisitos Não Funcionais (RNF)

1. **Confiabilidade** - O **sistema** deve ter a capacidade de atualizar as tabelas que utilizam os componentes e manter as aplicações que utilizam os componentes com dados atuais;

2. **Portabilidade** - O **componentes** devem ser projetados de maneira que sua renderização se comporte de maneira adaptável para navegadores em *desktops*, *smartphones* e *tablets*.

Variabilidades (VA)

1. **Processa Dados**: o resultado da pesquisa é executado pelo componente escolhido de acordo com o tipo dado do registro, não mutuamente exclusivo:
   - **Comportamentais** - nomeado como **FarmSPL.*behaviorals*** o componente para dados comportamentais, deve exibir o resultado da pesquisa em um mapa com a opção de filtrar por dia no intervalo previamente selecionado;
   - **Contextuais** - nomeado como **FarmSPL.*contextuals***, o componente para dados contextuais deve exibir o resultado da pesquisa em um calendário mostrando a quantidade de atividades executadas por dia;
   - **Fisiológicos** - nomeado como ***FarmSPL.fisiologicals***, o componente para dados fisiológicos deve exibir o resultado da pesquisa graficamente com opção de filtrar por dia no intervalo previamente selecionado;
   - **Microclima** - nomeado como **FarmSPL.*microlimates***, o componente para dados de microclima deve exibir o resultado da pesquisa graficamente com opção de filtrar por dia no intervalo previamente selecionado;
2. **Barra de Informações** do Animal - na resposta da pesquisa do **componente** poderá ser exibida uma barra contendo as informações: data selecionada, quantidade de registros e identificação do animal pesquisado;

Na Tabela 13, é mostrado um resumo dos principais requisitos da FarmSPL, diferenciando-os do núcleo das variabilidades por meio da coluna **Categoria de Reúso**.

Tabela 13 – Identificação dos casos de uso

Código	Descrição	Categoria de Reúso
RF01	O sistema deve permitir a opção de inclusão, alteração e remoção de agendamento (Gerenciar Agendamentos de atualização dos dados utilizados pelos componentes) de atualização dos componentes	Núcleo
RF02	O serviço deve oferecer um processo de execução dos agendamentos de atualização dos dados utilizados pelos componentes de acordo com os intervalos cadastrados (Executar Agendamentos)	Núcleo

RF03	O sistema pode exibir uma listagem com quais animais possuem registros de **pesagens** (Monitorar Pesagens)	Opcional
RF04	O sistema pode exibir uma listagem com registros de **conforto térmico** através do Índice de Temperatura e Umidade (ITU) (Monitorar ITU)	Opcional
RF05	O sistema pode exibir uma listagem com quais animais possuem registros de **rastreamentos** (Monitorar Rastreamentos)	Opcional
RF06	O sistema pode exibir uma listagem com quais animais possuem registros de **Pastejo no Sol** (Monitorar Pastejo no sol)	Opcional
RF07	O sistema pode exibir uma listagem com quais animais possuem registros de **visitas ao cocho** de alimentação (Gerenciar Visitas ao Cocho)	Opcional
RF08	O sistema pode exibir uma listagem com quais animais foram ao cocho e registraram a **temperatura local do cocho** de alimentação (Monitorar Temperatura local do cocho)	Opcional
RF09	O sistema pode exibir uma listagem com registros capturados de **temperatura de bulbo úmido** (Monitorar Temperatura de Bulbo Úmido)	Opcional
RF10	O sistema pode exibir uma listagem com registros capturados de **temperatura ar** (Monitorar Temperatura do Ar)	Opcional
VA01	Deve ser processado o resultado da busca considerando os componentes para dados: Comportamentais, Contextuais, Fisiológicos ou de Microclima	Alternativo
VA02	Mostrar barra de informações é opção que pode exibir uma barra contendo as informações de data selecionada, quantidade de registros e identificação do animal	Opcional

4.2.1.1 Modelo de Casos de Uso

A partir do documento de requisitos é elaborado o modelos de caso de uso, que serve como uma especificação do sistema que descreve o seu comportamento. Nesse documento são anotadas informações adicionais, de forma mais simples, com um caso de uso identificando os atores envolvidos em sua interação (SOMMERVILLE, 2011).

Inicialmente foi elaborada uma tabela de identificação dos casos de uso, para definir suas categorias de reúso com base em GOMAA (2005). Os casos foram relacionados e os requisitos aos quais são representados diante de suas categorias de reúso estão especificados na Tabela 14.

Tabela 14 – Identificação dos casos de uso

Caso de uso	**Ator**	**Descrição**	**Categoria de Reúso**	**Requisitos Relacionados**
Gerenciar Agendamentos	Administrador do Sistema	Consultar, editar e remover agendamentos de atualização das tabelas que utilizam os componentes	Núcleo	RF01
Executar Agendamentos	Sistema	Executar os processos de agendamentos de atualização dos dados dos componentes de acordo com os intervalos cadastrados em Gerenciar Agendamentos	Núcleo	RF01
Monitorar Pesagens	Administrador do Sistema	Consultar quais animais possuem registros de **pesagens**	Opcional	RF03, RF01, VA01
Monitorar Índice de Temperatura e Umidade (ITU)	Administrador do Sistema	Consultar registros de **ITU**	Opcional	RF04, RF01, VA01
Monitorar Rastreamentos	Administrador do Sistema	Consultar quais animais possuem registros de **rastreamentos**	Opcional	RF05, RF01, VA01
Monitorar Pastejo no Sol	Administrador do Sistema	Consultar quais animais possuem registros de **pastejo no sol**	Opcional	RF06, RF01, VA01

Monitorar Visitas ao Cocho	Administrador do Sistema	Consultar quais animais possuem registros de **visitas ao cocho** de alimentação e quais cochos foram visitados	Opcional	RF07, RF01, VA01
Monitorar Temperatura local do Cocho	Administrador do Sistema	Consultar quais animais foram ao cocho e registraram a **temperatura local do cocho** de alimentação e quais cochos foram visitados	Opcional	RF08, RF01, VA01
Monitorar Temperatura de Bulbo Úmido	Administrador do Sistema	Consultar registros capturados de **temperatura de bulbo úmido**	Opcional	RF09, RF01, VA01
Monitorar Temperatura de Ar	Administrador do Sistema	Consultar registros capturados de **temperatura de ar**	Opcional	RF10, RF01, VA01
Mostrar Barra de Informações	Administrador do Sistema	Exibir uma barra com informações: data selecionada, quantidade de registros e identificação do animal	Opcional	VA02

Após serem identificados os casos de uso é necessário identificar os Atores do sistema. O sistema foi projetado somente para integração inicial dos componentes, não sendo utilizados níveis de acesso, apesar da arquitetura a utilizada fornecer. Foram identificados apenas dois usuários, representados devidamente com suas responsabilidades a seguir:

1. **Administrador do Sistema**: Gerenciar Agendamentos, Monitorar Pesagens, Monitorar ITU - Índice de Temperatura Animal, Monitorar Rastreamentos, Monitorar Pastejo no Sol, Monitorar Visitas ao cocho,

Monitorar Temperatura Local do Cocho, Monitorar Temperatura de Bulbo Úmido, Monitorar Temperatura do ar, Mostrar Barra de Informações;

2. **Sistema**: Executar Agendamentos;

O diagrama de casos de uso do ator Administrador do sistema está ilustrado na Figura 15 e do ator Sistema na Figura 16. Esses diagramas foram elaborados a partir das informações levantadas. Cada estereótipo representa uma categoria de reúso, conforme abordagem de GOMAA (2005):

- ***kernel***: correspondem aos casos de uso que fazem parte do núcleo do sistema; estão presentes em todos os membros da LPS, representado no modelo pelo estereótipo *«kernel»*;
- ***optional***: correspondem aos casos de uso que podem ou não serem usados nos membros da linha, representado no modelo pelo estereótipo *«optional»*; e
- ***alternative***: correspondem a escolhas alternativas que podem ser feitas, possuem grupos de opções para a escolha, de um deles em um certo membro da linha, representado no modelo pelo estereótipo *«alternative»*.

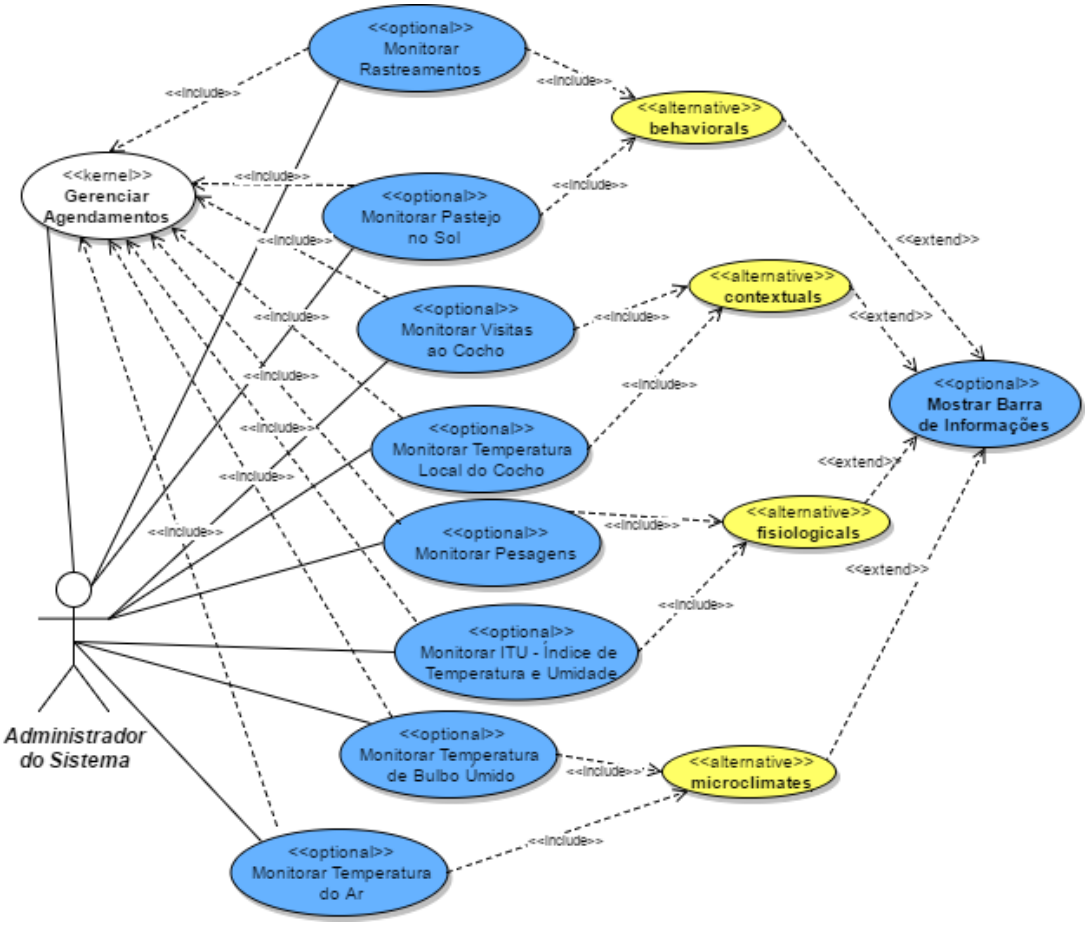


Figura 15 – Diagrama de casos de uso - ator: Administrador do Sistema

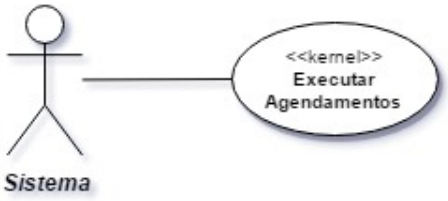


Figura 16 – Diagrama de casos de uso - ator: Sistema

4.2.1.2 Modelo de *Features*

O modelo de *features* é muito importante para o desenvolvimento de uma LPS pois através dele são identificadas as *features* que são membros da LPS. As *features* são representadas como: *common*, *alternative* e *optional*. Conforme apresentado na Seção 2.6.2,

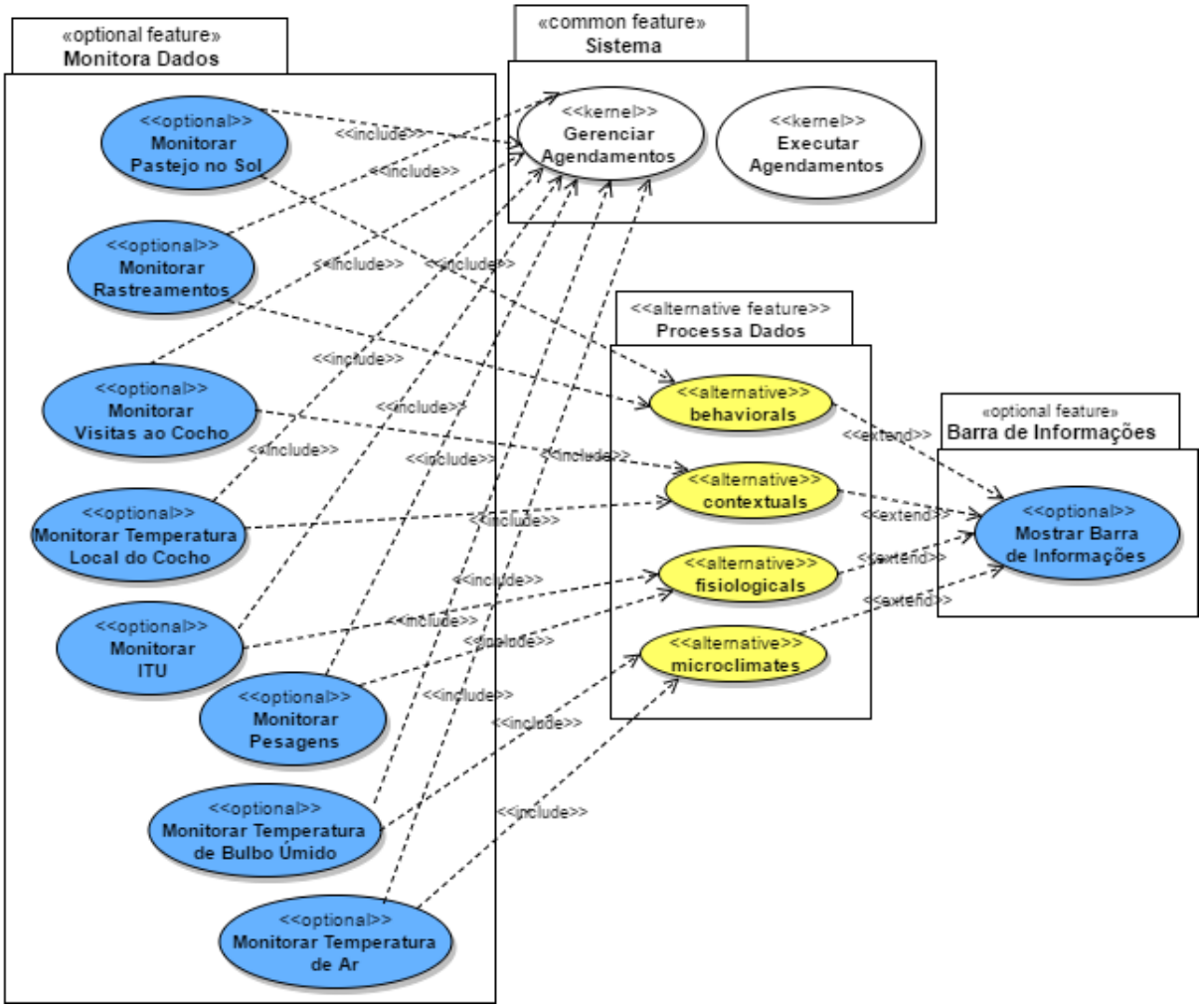


Figura 17 – Pacotes de casos de uso - Relação entre casos de uso e as *features*

O primeiro diagrama apresentado na Figura 17, proposto por GOMAA (2005), modela as *features* do sistema com as *features* dos pacotes de casos uso. Esse tipo de modelo identifica quais *features* e casos de uso estão contidas nele, sendo as *features* menores consideradas como pontos de variação em casos de uso.

Os casos de uso que possuem categoria de reúso "núcleo" de estereótipo *«kernel»* são colocados em um mesmo pacote, que foi denominado **Sistema** (ou o software a ser gerado), em seguida, para cada variabilidade possível da LPS são mostradas as *features*. As variabilidades de estereótipo *«alternative feature»* e mesmo caso de uso, foram nomeadas como **Processa Dados** pois nela estão os pontos de variabilidade, onde são processados

os dados que os componentes podem utilizar. O caso de uso com estereótipo *«optional feature»* nomeado **Barra de Informações** pode não ser incluso, por isto foi mantido como opcional.

Os casos de uso Monitorar Pesagens, Monitorar ITU, Monitorar Rastreamentos, Monitorar Pastejo no Sol, Monitorar Visitas ao Cocho, Monitorar Temperatura Local do Cocho, Monitorar Temperatura de Bulbo Úmido e Monitorar Temperatura do Ar, também são indicados com o estereótipo *«optional»* e o pacote de nome **Monitora Dados** identificado com estereótipo *«optional feature»*. Cada elemento especificado no pacote **Monitora Dados** é uma funcionalidade que pode ou não ser criada, apesar que tenhamos partido dessas funcionalidades, sabe-se que pode ter outras sendo que cada uma poderá utilizar um devido componente ou um ponto de variação. Para o funcionamento de cada uma dessas funcionalidades necessita-se de um agendamento para atualização periódica, por isso a foi mantido como *«kernel»* o caso de uso Gerenciar Agendamentos.

Na Figura 18 é apresentado o modelo hierárquico das *features*, conforme a proposta de GOMAA (2005), representando as *features* que devem estar presentes em todos os sistemas alvo. Na Figura 19 o modelo hierárquivo entre as *features* de variabilidades, também conforme a proposta de GOMAA (2005), já com a *feature* **Monitora Dados** representada por *«optional feature»*. Esse *feature group* representa que suas *features* devem ser mantidas como mutuamente inclusivas, ou seja, se alguma dessas *features* forem incluídas, automaticamente a que está abaixo (uma *feature* do **Processa Dados**) será incluída. Este modelo explica de forma desagrupada as *features* de estereótipo *«alternative feature»* definidas no início do processo, que são os componentes que definem cada ponto de variação da LPS.

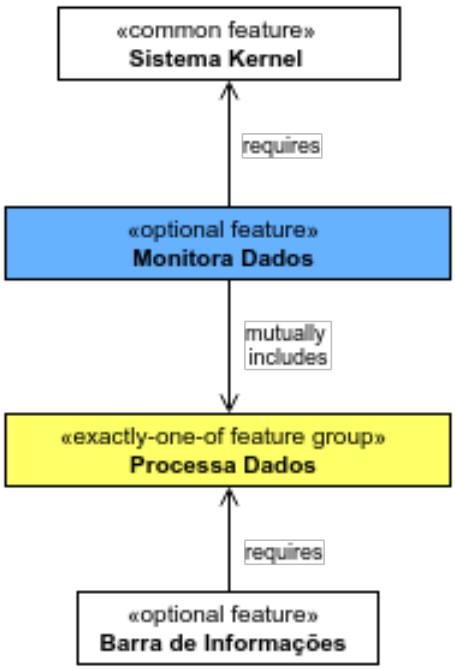


Figura 18 – Modelo Hierárquico de *features*

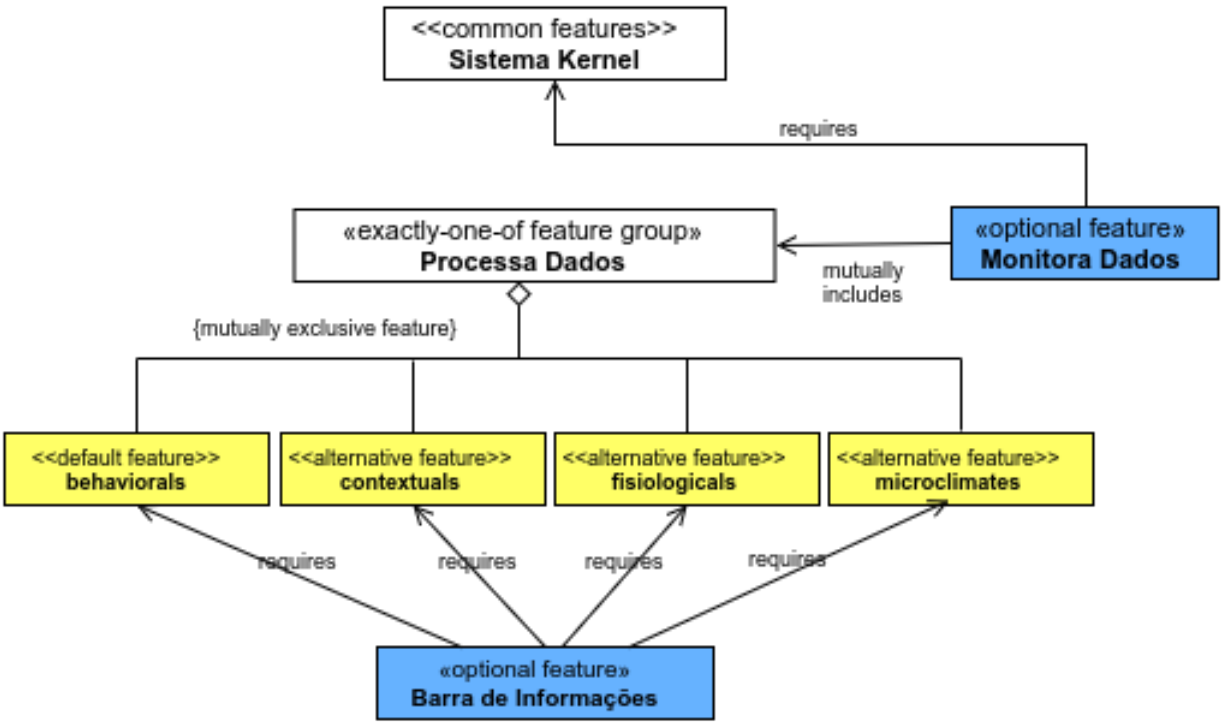


Figura 19 – Modelo Hierárquico entre o pacote ***kernel*** e as *features* de variabilidade

4.2.2 Atividades de Análise

Segundo o processo de GOMAA (2005), na modelagem de análise, o modelo estático define as relações entre as classes, representando-as com seus relacionamentos em um diagrama de classes. Esse modelo representa categoriza as classes usando esteriótipos UML e cada estereótipo define um novo bloco de construção que é derivado de um elemento de modelagem UML existente adaptado a partir do domínio do problema (GOMAA, 2005).

Os membros da LPS com conceito de núcleo possuem o estereótipo *«kernel»*, os que estão de forma opcional são definidos como *«optional»* e os que possuem versões alternativas para diferentes membros da linha são denominados com conceito de variantes e possuem o estereótipo *«variant»* (GOMAA, 2005). O modelo estático da LPS é ilustrado na Figura 20, com conceitos de núcleo, opcionais e conceitos alternativos.

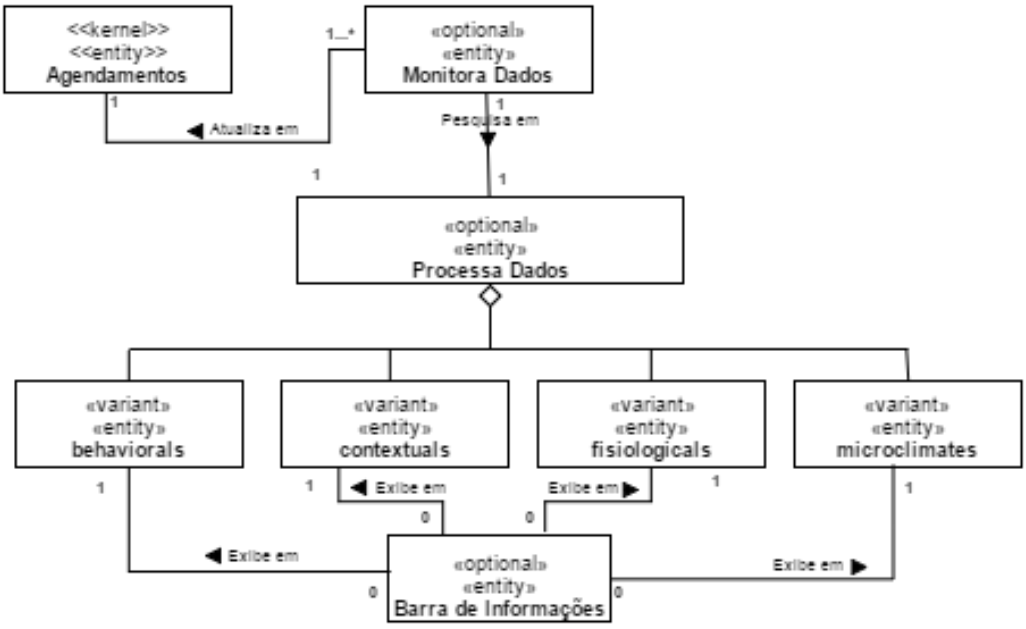


Figura 20 – Modelo Estático de classes de entidade da LPS proposta

A partir do modelo estático são extraídas as *features* de nível mais alto, que são correlacionadas a alguma funcionalidade ou caso de uso que será provido pela arquitetura que será reutilizada. O modelo de comunicação integrado é uma síntese de todos os diagramas de comunicação de casos de uso relevantes mostrando todos os objetos e suas interações, esse modelo integrado é representado como um modelo genérico de comunicação UML GOMAA.

Baseado na representação de GOMAA (2005), as entidades mais relevantes estão demostradas na Figura 21. Nesse modelo estão contidos dois grupos: o primeiro, que possui as *features* de nível mais alto, foi descrito como **Apoio a SPL**; no outro grupo denominado **FarmSPL**, estão contidas as variabilidades da LPS, também especificado

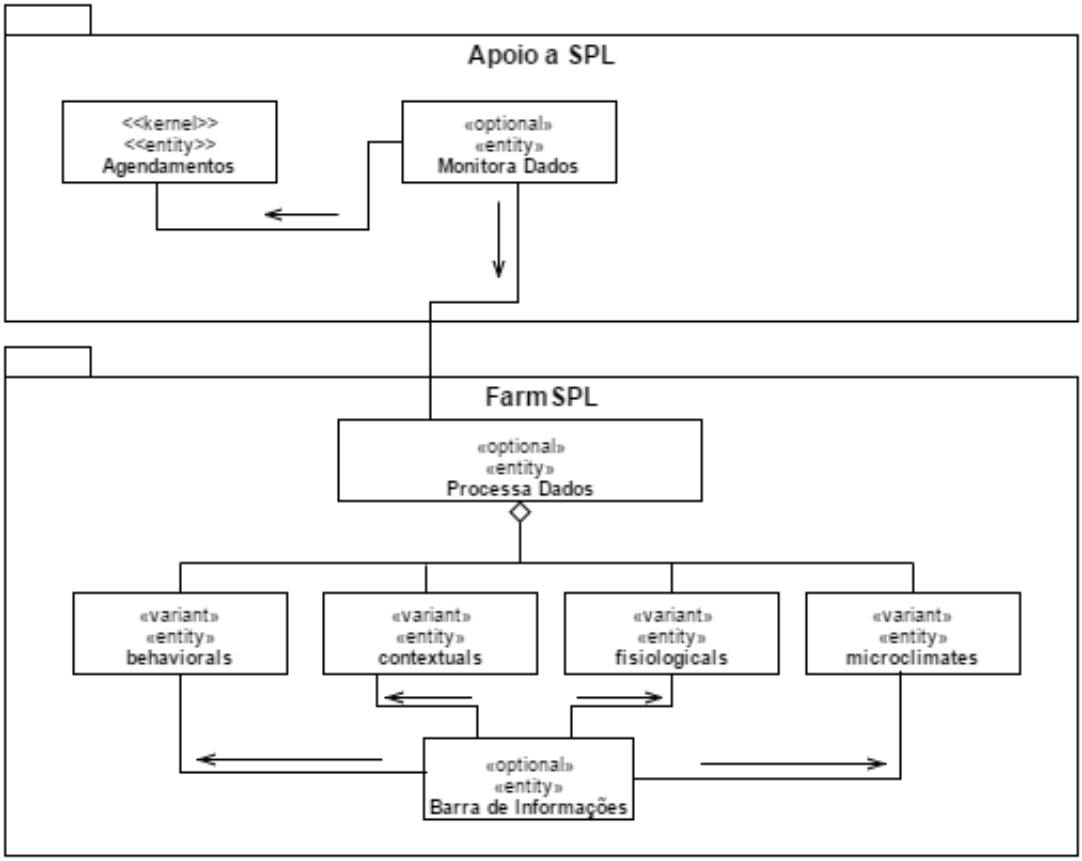


Figura 21 – Modelo de Comunicação entre as *features*

no pacote **Processa Dados** do diagrama de pacotes dos casos de uso, representado na Figura 17. A entidade **Processa Dados** é constituído pelos componentes:

- **FarmSPL.*behaviourals***: para processamento de dados comportamentais;
- **FarmSPL.*contextuals***: para processamento de dados contextuais;
- **FarmSPL.*fisiologicals***: para processamento de dados fisiológicos;
- **FarmSPL.*microclimates***: para processamento de dados de microclima;

O caso de uso Gerenciar Agendamentos está identificado como **Agendamentos** e está relacionado as funcionalidades de Monitoramento de dados para realizar a atualização dos dados. A entidade **Monitora Dados** é responsável pelos casos de uso: Monitorar Pesagens, Monitorar ITU, Monitorar Rastreamentos, Monitorar Pastejo no Sol, Monitorar Visitas ao Cocho, Monitorar Temperatura Local do Cocho, Monitorar Temperatura de Bulbo Úmido e Monitorar Temperatura do Ar.

4.2.3 Atividades de *Design* e Implementação

Durante a engenharia da aplicação é desenvolvida uma aplicação. Essa aplicação é um membro da LPS, que é desenvolvido com o uso dos artefatos desenvolvidos durante a engenharia da LPS. A modelagem de de projeto é identificada a arquitetura de referência, um ambiente para geração de aplicações. Na sequência é apresentado o **Titan *Framework***, descrito na Seção 2.8, que servirá como arquitetura de referência para o projeto.

4.2.3.1 Arquitetura

A arquitetura da linha de produto baseada em componentes é concrerizada após ser feito o modelo de comunicação, onde os componentes são agrupados em subfuncionalidades de acordo com seus objetivos. Para desenvolver os componentes do repositório e sistematizar um processo de reutilização, é fundamental a existência de ferramentas automatizadas, e o **Titan *Framework*** provê um repositório com núcleo de ativos e artefatos para abstração de seus padrões para especificação de Engenharia da LPS e evolução para uso para domínios específicos. Outro detalhe é que o Titan já foi estentido e adaptado em CARROMEU (2007), e evoluído a arquitetura em um projeto denominado "Arquitetura baseada em componentes para o desenvolvimento de sistemas Web e-Gov" (CARROMEU et al., 2010). Também possui evolução no uso de arquitetura *mobile*, trabalho selecionado no mapeamento sistemático, denominado "***The evolution from a web SPL of the e-gov domain to the mobile paradigm***" (CARROMEU; PAIVA; CAGNIN, 2015). Levando essa perspectiva em consideração, percebe-se que o Titan *Framework* é uma arquitetura flexível que possibilita evolução para o domínio de Pecuária de precisão.

4.2.3.2 Titan *Framework* como Repositório

A arquitetura do Titan *Framework* forneceu apoio para o desenvolvimento das entidades agrupadas como **Apoio a SPL**, exibido na Figura 21, no modelo de comunicação. Ela possui um repositório global com artefatos nativos importantes para qualquer instância de aplicações com padrões análise e projeto como CRUDs (acrônimo de Create, Retrieve, Update and Delete).

As sessões e ações utilizadas no Titan *Framework* deram apoio para a geração das funcionalidades de *features* definidas no caso de uso e das oito definidas são apresentadas nesse capítulo seis, que são: Monitorar Pesagens, Monitorar Rastreamentos, Monitorar Visitas ao Cocho e Monitorar Temperatura de Bulbo Úmido. Também é apresentado o caso de uso Gerenciar Agendamentos.

O Titan renderiza essas seções e ações em tempo de execução por meio da parametrização, utilizando XML, de código de script (*server side*) PHP, onde fica organizado por meio de "**componentes**" e "**motores**"(do inglês *engines*). Cada seção precisa estar

relacionada a um componente e cada ação dessa seção precisa estar relacionada a uma *engine* deste componente, conforme ilustrado e na Figura 22.

Figura 22 – Esquema geral da renderização de seções e ações por meio de componentes e engines (CARROMEU, 2014)

4.2.3.3 FarmSPL - Evolução do componente **global.generic** do Titan *Framework*

O componente denominado "**global.generic**" recebe um arquivo "**config.inc.xml**" que possui a definição de todas as ações da seção que está sendo gerada, mais detalhes de seu uso podem ser encontrados no *cookbook* do Titan framework (CARROMEU, 2014). Para configurar cada seção para uso do componente são fornecidas as opções de configurações pelos artefatos: "**config.inc.xml**", "**all.inc.xml**" e "**list.inc.xml**". Esse componente forneceu os artefatos necessários para as funcionalidades levantadas para monitoramento de Pecuária de Precisão propostas neste trabalho, havendo uma evolução para atender as demandas do projeto e atender as variabilidades. Para cada funcionalidade a ser criada deverão ser incluídos mais dois arquivos, que são listados abaixo.

- **farmsplview.xml**: metadados com as configurações para montar a os critérios de busca da requisição;
- **farmsplfile.xml**: metadados com as configurações para geração do arquivo JSON após a requisição da busca;

A arquitetura dos componentes FarmSPL é uma evolução do componente ***global.generic***. Cada seção, para executar algum dos componente, deve ser configurada com os artefatos "**config.inc.xml**", "**list.inc.xml**", além dos dois artefatos citados previamente, "**farmsplfile.xml**" e "**farmsplview.xml**".

Após configurado o ambiente de desenvolvimento para utilizar o Titan *Framework*, conforme CARROMEU (2014), foi criada a primeira instância[4], que trata-se da funcionalidade de **Rastreamentos**. Essa funcionalidade utiliza dados comportamentais e através do *framework* fornece uma listagem com os animais que possuem rastreamentos cadastrados para que possa ser selecionado algum animal e na pesquisa sejam verificados seus rastreamentos durante os dias selecionados.

O banco de dados foi criado conforme a sequência de configuração explicada por CARROMEU (2014), as tabelas e relações foram criadas conforme a última atualização ou versão do *framework*. As tabelas mandatórias do Titan são prefixadas pelo caractere *underline*, todas elas ficam no *schema* denominado "titan", que é o schema principal da aplicação.

Com o banco de dados já em funcionamento, pode-se fazer as alterações necessárias para que ele comporte as seções da instância FarmSPL. Para os testes e estudo de caso deste trabalho, foi criado um *schema* chamado "FarmSPL". A primiera funcionalidade criada foi a de Monitorar Rastreamentos e para ela foi criada a funcionalidade ***geotracking*** e uma tabela com o mesmo nome no *schema* FarmSPL. Essa tabela armazena os dados que serão exibidos na seção. O nome escolhido para a seção de **Rastreamentos** foi o de ***geotracking***, porém não há obrigatoriedade de nome, fica a critério do desenvolvedor criar algum padrão.

A tabela de rastreamentos armazena dados comportamentais e possui uma coluna chamada "_user", chave estrangeira para a tabela mandatória de usuários da aplicação (referencia o último usuário que criou ou alterou a tupla), uma outra coluna chamada "_create" com a data de criação e "_update" com a última alteração, conforme ilustrado e na Figura 23.

Após ser criada a tabela, foi feita a configuração da seção ***geotracking*** com os arquivos "config.inc.xml", "list.inc.xml", "farmsplview.xml" e "farmsplfile.xml", a estrutura de diretórios fica conforme a ilustrada na Figura 24.

Ao carregar o seu núcleo o Titan *Framework* lê o arquivo "**configure/business.xml**", onde estão declaradas todas as seções da instância (CARROMEU, 2014). A derivação do componente **global.generic** foi feita e ele foi evoluído dentro da pasta local, com o nome **farmSPL**. Essa derivação é apontada no caminho da seção dentro do arquivo **business.xml** com o caminho: "**local/component/farmSPL**", conforme o Código-Fonte

[4] No Titan, a instanciação é a concretização, por meio de um conjunto de arquivos que referenciam o core, de uma nova aplicação Web (CARROMEU, 2014)

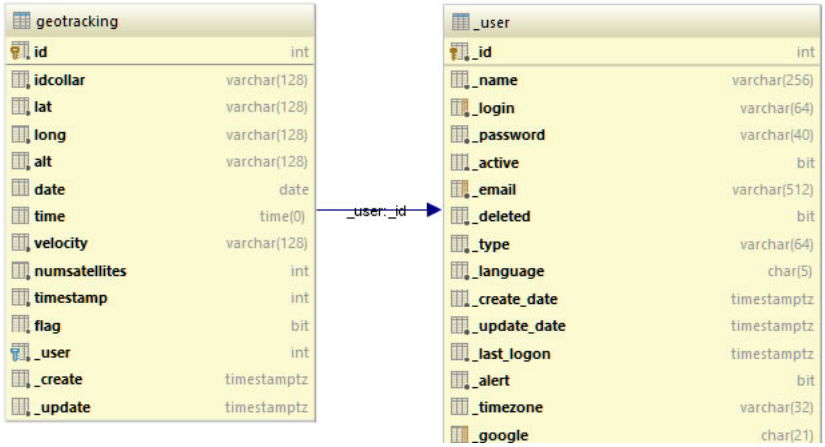


Figura 23 – Tabela do banco de dados *geotracking* para dados de rastreamento e relação com tabela mandatória do TITAN

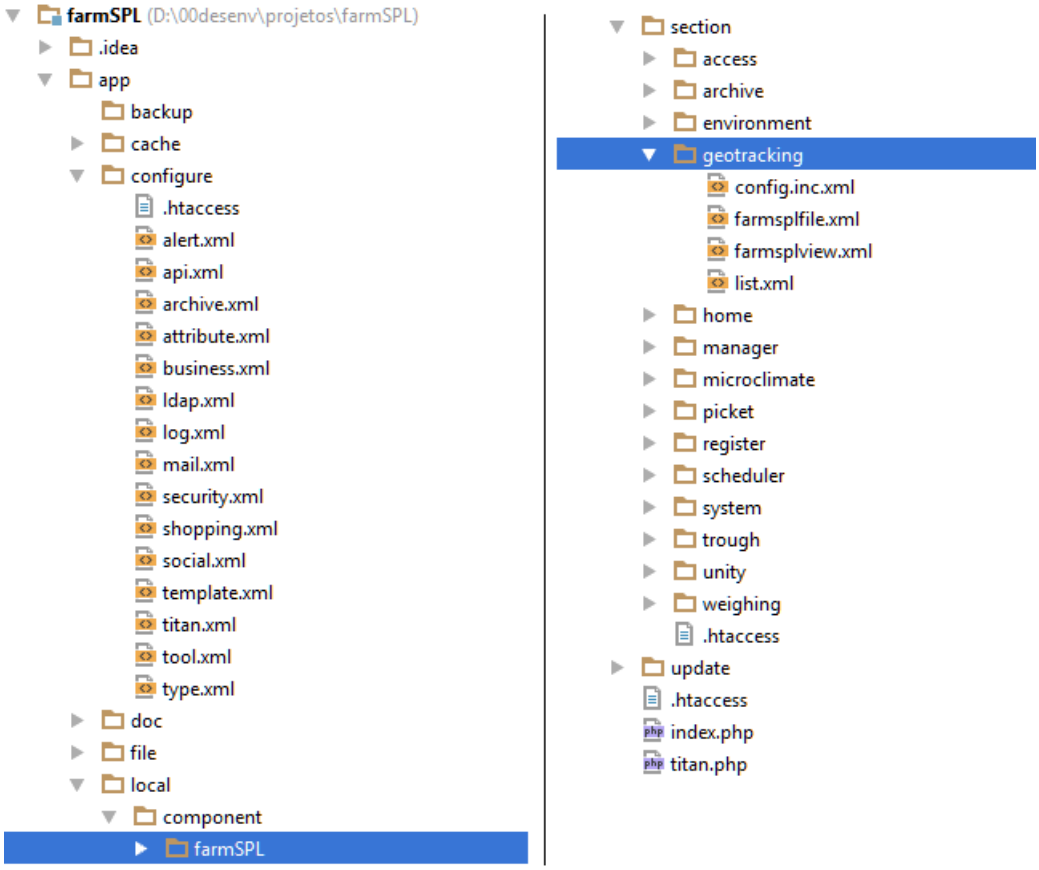


Figura 24 – Diretórios e arquivos de uma instância do Titan com o componente FarmSPL

4.1.

Este procedimento é suficiente para criar a hierarquia de acesso no menu, depois é preciso mapear os arquivos da aplicação, um deles é o **list.xml**, exibido no Código-Fonte C.1 e o artefato **config.inc.xml**, exibido no Código-Fonte C.2.

O artefato **list.xml** oferece opção de configuração de lista com uma tabela e/ou

```
1  <!-- Menu Pai -->
2  <section
3          label="FarmSPL | pt_BR: FarmSPL"
4          name="farmSPL" />
5
6  <!-- SubMenu Rastreamentos -->
7  <section
8          label="Geotracking | pt_BR: Rastreamentos"
9          name="geotracking"
10         component="local/component/farmSPL/"
11         father="farmSPL" />
```

Código-Fonte 4.1 – Mapeamento da seção de **Rastreamentos - business.xml**

uma busca, definidos com dois blocos de metadados:

- ***view***: mapeamento apontando para os campos de formulário (<*field*>) que serão exibidos da consulta na lista. O atributo "*table*" aponta para a referência da consulta no banco. Para essa consulta foi criada uma visão com o nome **farmspl.vwgeotracking_qtde**, que utiliza a tabela ***geotracking*** para fazer consultas de quais animais que possuem registros de rastreamentos a quantidade;
- ***search***: mapeamento de formulário para pesquisa;

O artefato **config.inc.xml** assim como o **list.inc.xml** é padrão do Titan *Framework* e ficam localizados na mesma pasta da seção. Conforme pode ser verificado no Código-Fonte C.2, as ações são definidas após a linha 23 com as seguintes ações (*actions*):

- **name**="**list**": corresponde a ação para listagem padrão do *framework*;
- **name**="**farmsplview**": aponta ação para a pesquisa, com as configurações no metadados **farmsplview.xml**;
- **name**="**farmsplfile**": aponta ação para a geração do metadado JSON a ser consultado como resultado para o componente relacionado, de acordo com as configurações no metadados **farmsplfile.xml**;

Também é possível notar no Código Fonte C.2 que existem *tags* ***directives***, que são opções que o *framework* oferece para possibilitar a codificação de forma mais rápida de estruturas de controle parametrizáveis, com parâmetros que podem ser recuperados por consultas na aplicação. Essas *directives* foram utilizadas para enviar parâmetros com os componentes.

As variáveis de pesquisa são ***directives*** que relacionam o nome do campo da pequisa ao nome da coluna no banco de dados. Além dessas variáveis, outros quatro parâmetros foram predefinidos:

- **__NAME__** - informa qual componente será usado, o padrão é o componente FarmSPL.*fisiologicals*;
- **__RESOURCE__** - indica a visão ou tabela no banco de dados que será utilizada para a consulta;
- **__MEASURE__** - quais campos vão sere utilizados para visualização no componente, no exemplo tem-se: animal, lat e long. São as medidas que serão utilizadas para gerar o rastreamento de cada animal;
- **__TITLE__**: título a ser apresentado na renderização da pesquisa; e
- **__DESCRIPTION__**: Descrição a ser apresentado na renderização da pesquisa;
- **__TYPECHART__**: Tipo do gráfico a ser renderizado, não foi citado no artefato **config.inc.xml** mostrado no Código-Fonte C.2 pois a configuração acima, para o componente FarmSPL.*behaviorals*, utiliza representação em mapa, em outros quando não citado o padrão é valor “bars”, ou gráfico em barras;

A renderização da seção de **Rastreamentos** é mostrada na Figura 25 com a listagem dos animais que possuem registros rastreamentos e a opção de visualização de mais detalhes representada pelo ícone da lupa para realizar a pesquisa.

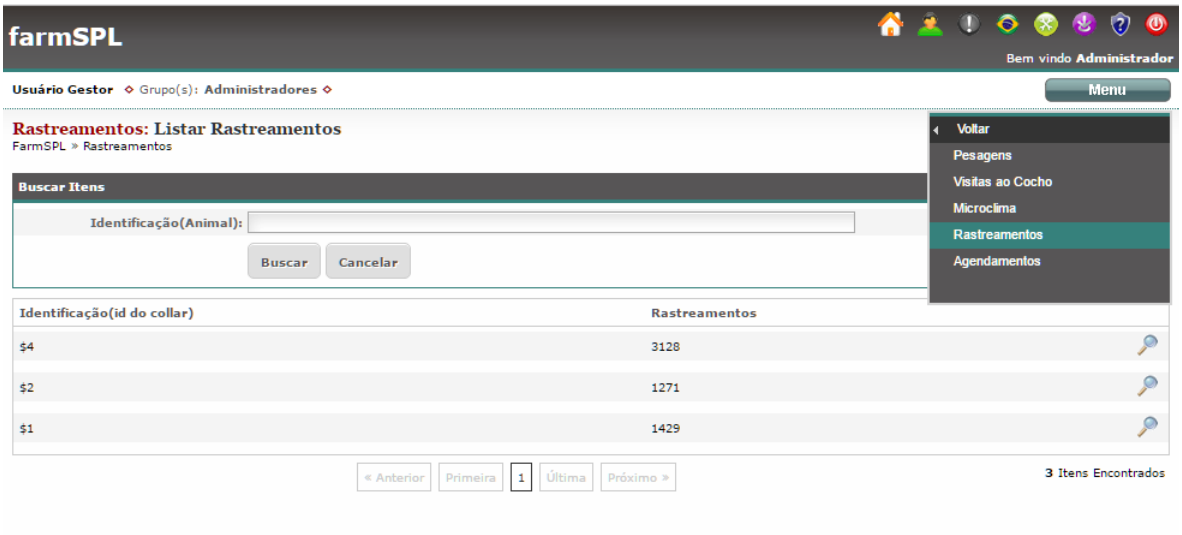


Figura 25 – Listagem de animais que possuem registros de **Rastreamentos**

A pesquisa oferece uma opção de busca dos registros do animal selecionado, pela data inicial e data final. Para utilizá-la o usuário deve clicar na opção representada pelo ícone da lupa, à direita do registro na lista. A configuração dela é feita de acordo com o metadado **farmsplview.xml**, exibido no Código-Fonte C.3 e sua requisição é orientada pelo **farmsplfile.xml**, exibido no Código-Fonte C.4, que fornece um mapeamento para geração do resultado da consulta, posteriormente utilizada no componente. Esses dois artefatos utilizam as configurações de mapeamento padrões do *framework*.

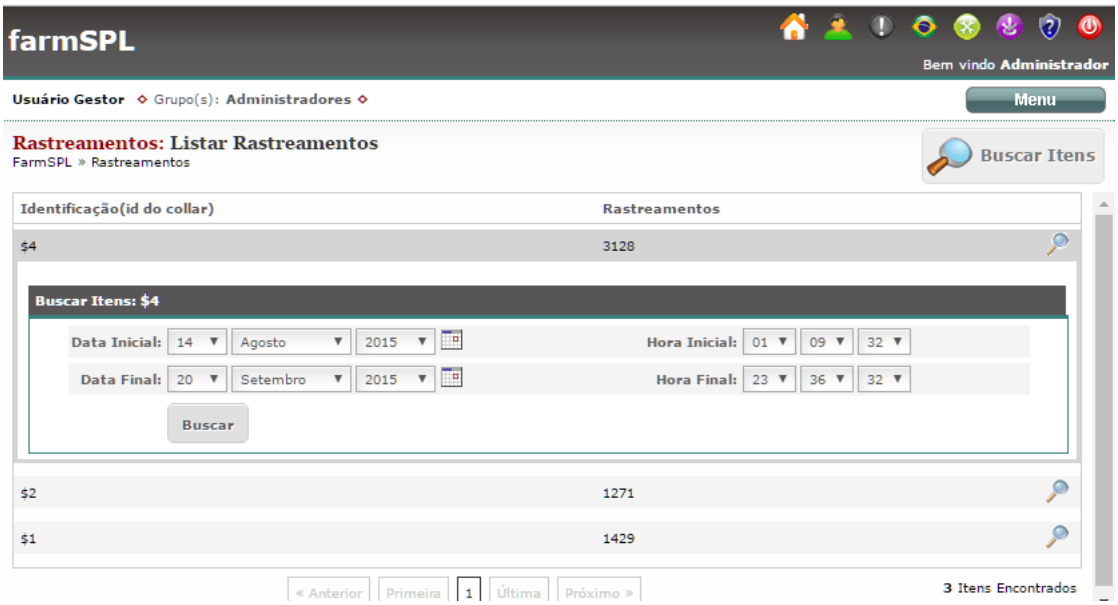


Figura 26 – Pesquisa aberta após clicar em visualizar na lista de animais com **Rastreamentos**

A renderização da pesquisa é ilustrada na Figura 26, e seu título utiliza o *label* **Buscar Itens**. Esta busca é executada dentro do componente correlacinado, nesse caso o

farmspl.*behaviorals*, pré-definido na ***directive*** "_NAME_", conforme o mapeamento feito no **config.inc.xml** da seção.

A execução da pesquisa é feita após o ator clicar no botão **Buscar**, com a renderização no componente exibindo o rastreio do animal de acordo com o dia escolhido, ilustrando onde foi percorrido e o horário para cada ponto selecionado. Esse processo pode ser visualizado na renderização do componente **farmspl.*behaviourals*** ilustrado na Figura 27.

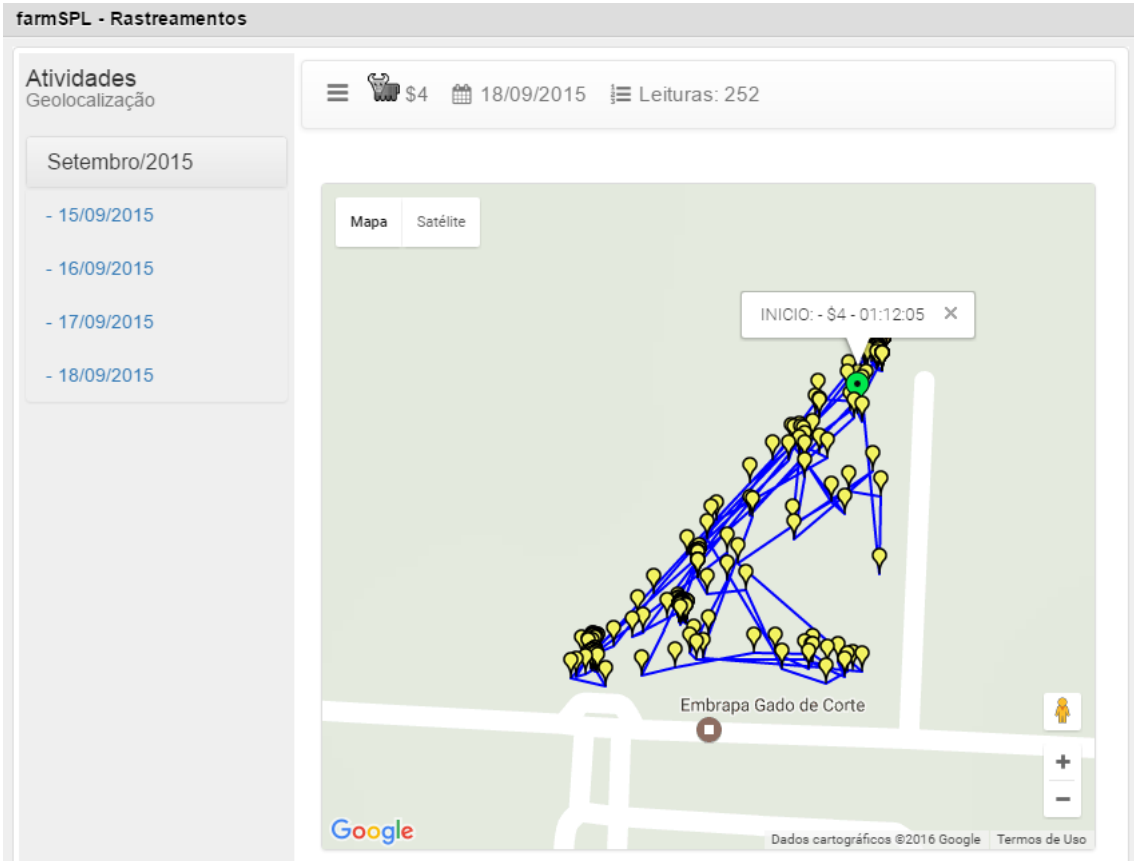


Figura 27 – Renderização do componente **farmspl.*behaviorals*** - **Rastreamentos**

4.2.3.4 Reutilização dos componentes da FarmSPL

Depois de ter sido feita a primeira funcionalidade, a de **Rastreamentos**, as outras *features* definidas no caso de uso também puderam ser configuradas, pois puderam reutilizar os artefatos. O processo é o mesmo, o que modifica somente é a escolha do componente.

Para o processo de reutilização foram seguidos os mesmos passos, com os artefatos mapeados de acordo com a configuração de cada um para sua especificidade, conforme os seguintes passos:

1. Criar a tabela no banco de dados para a seção e como o mesmo nome;

2. Criar a pasta com o **nome da seção** dentro da pasta **section/**;

3. Mapear o artefato **list.inc.xml** (conforme o Código-Fonte C.1), **config.inc.xml** (conforme o Código-Fonte C.2), **farmsplview.xml** (conforme o Código-Fonte C.3) e **farmsplfile.xml** (conforme o Código-Fonte C.4) dentro da pasta da seção;

4. Criar uma *tag* apontando o caminho do componente no artefato **business.xml** dentro da pasta **configure/**, (conforme o Código-Fonte 4.1).

Para o banco de dados foi criada uma tabela relacionada para cada tipo de componente. Essas tabelas foram relacionadas com a tabela mandatória **"_user"** do *framework*. A Figura 28 exibe as tabelas relacionadas. Cada seção possui o mesmo nome da tabela e cada uma faz uso de uma categoria de dados. Os dados são renderizados no componente previamente definido que podem possuir ou não a **Barra de Informações**.

As seções criadas para utilização dos componentes criados foram organizadas da seguinte forma:

geotracking*:** Utiliza o componente **FarmSPL.*behaviorals e dados comportamentais, que exibe os registros de rastreamento do animal através de um mapa e data escolhida. Um exemplo desta renderização pode ser vista na Figura 27, com uma consulta para o animal com identificação **$4** no dia **18/09/2015**, com o retorno de **252** leituras.

trough*:** utiliza o componente **FarmSPL.*contextuals e dados contextuais, que exibe a quantidade de visitas ao cocho de alimentação de determinado animal para cada dia através de um calendário. A Figura 29 mostra uma consulta para o animal que possui a identificação número **21110720930577**, filtrando registros do dia **19/07/2015** no cocho de alimentação **A01**, com retorno de **5** visitas.

weighing*:** utiliza o componente **FarmSPL.*fisiologicals e dados fisiológicos, que exibe um gráfico com a medida do peso do animal selecionado pela hora do registro de peso do animal. A Figura 30 mostra a consulta do dia **30/01/2014** para o animal com identificação número **982000088460021**.

wetbulb*:** utiliza o componente **FarmSPL.*microclimates e dados de microclima, exibe um gráfico com dados coletados de índice temperatura de bulbo úmido. No exemplo, ilustrado na figura 31, a consulta mostra a variação no dia **28/10/2014** com **488** leituras;

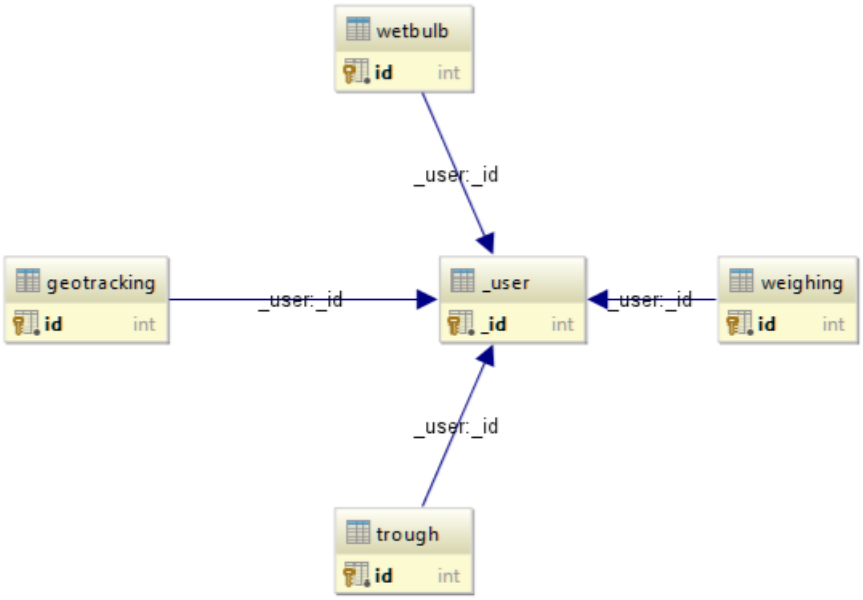


Figura 28 – Relacionamento entre as tabelas utilizadas para as funcionalidades e a tabela mandatória **__user** do TITAN *Framework*

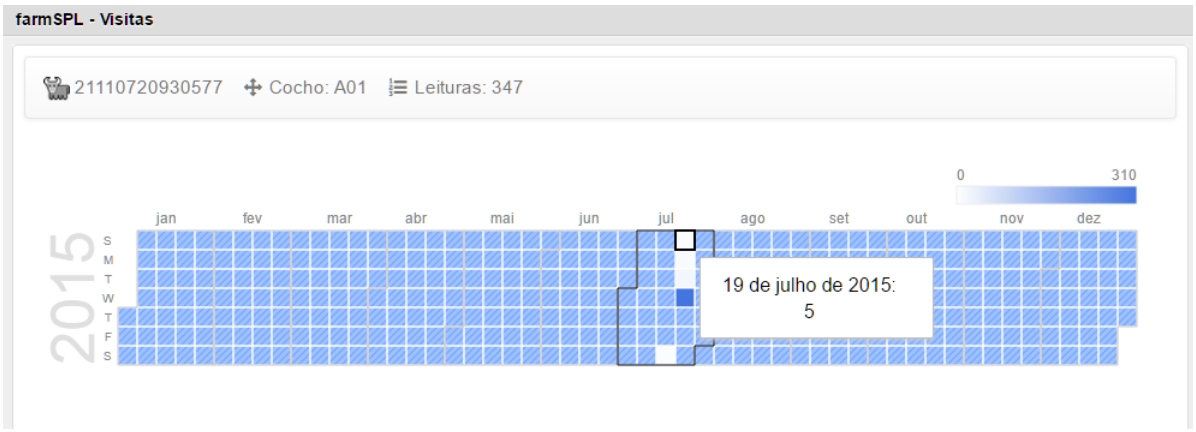


Figura 29 – Renderização do componente **farmspl.*contextuals* - Visitas ao Cocho**

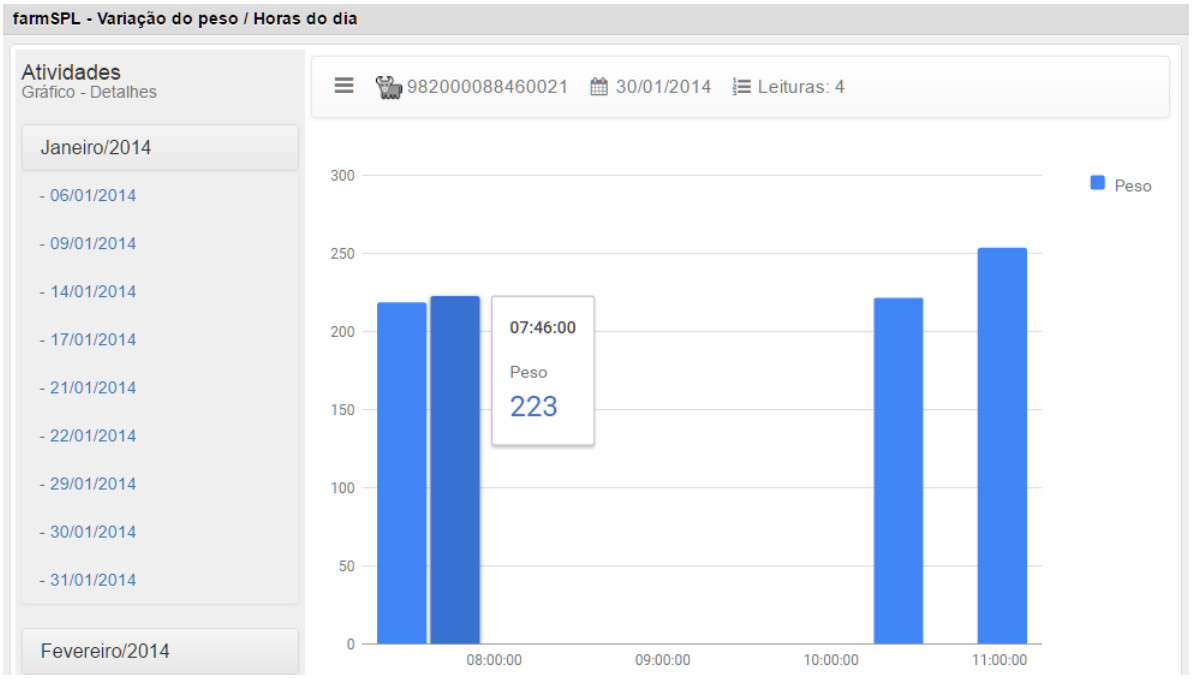


Figura 30 – Renderização do componente **farmspl.*fisiologicals* - Pesagens**

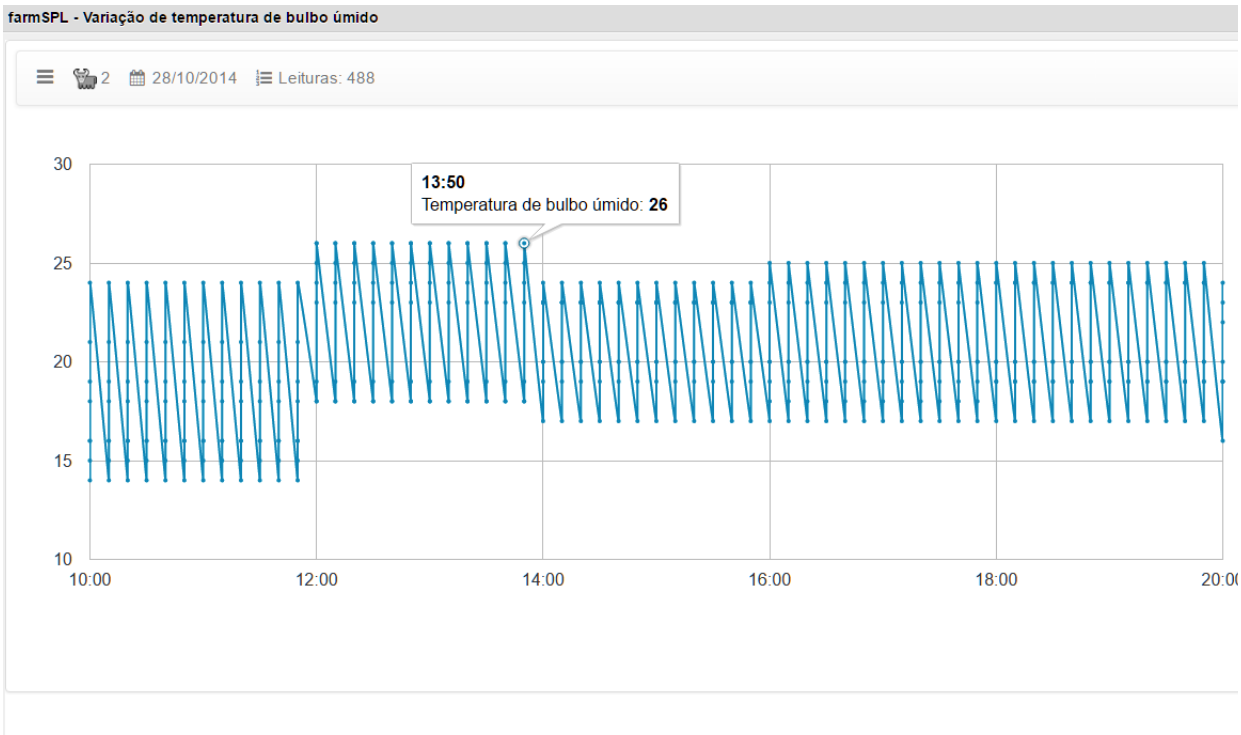


Figura 31 – Renderização do componente **farmspl.*microclimates* - Variação de temperatura de bulbo úmido**

O repositório com a documentação técnica oficial da FarmSPL com os passos de reutilização e uma instância criada com os componentes pode ser acessado no endereço:

<https://git.cnpgc.embrapa.br/farmspl/docs>

4.2.3.5 Sincronização dos dados

Apoiado no reúso da arquitetura do *framework* escolhida, foi necessário criar a estrutura definida na **Visão Geral da Linha** para o documento de requisitos, especificado na Seção 4.2.1, que serviu como uma sugestão para o item **e**, a camada definida na plataforma ***e-Cattle***, ilustrada na Figura 1. Foi implementado um serviço de *webservice* para fornecer os dados já em alto nível para sincronia/recuperação para as funcionalidades definidas neste projeto.

A camada de serviços proposta em (CARROMEU, 2016) é composta por *webservices* no padrão RESTful e aceita requisições por meio de métodos **POST** e **GET** . Para este projeto será necessário somente a requição do tipo **GET** com fornecimento de resultados por meio de arquivo JSON, exemplificado na Seção 2.9.

Como foi mencionado na Seção 4.2.1, os dados foram inseridos no mesmo banco de dados utilizado no Titan *Framework*, o PostgreSQL versão 9.5 e linguagem de programação *server-side* PHP. Após os dados da amostragem terem sidos tratados e inseridos no banco de dados, foi criado o exemplo para geração de webservice, com os quatro tipos de dados: comportamentais, contextuais, fisiológicos e de microclima.

As requisições de método GET são definidos por URIs como identificadores com a seguinte semântica: **esquema://domínio:porta/caminho/serviço/recurso?querystring**

- **esquema**: ou protocolo, usado HTTP.
- **domínio**: ou máquina, ou servidor que disponibiliza o recurso designado.
- **porta**: ponto lógico onde pode-se fazer a conexão com o servidor (opcional), não foi utilizado;
- **caminho**: especifica o local (geralmente em um sistema de arquivos) onde encontra-se recurso dentro do servidor;
- ***query string***: conjunto de parâmetros a serem enviados ao servidor, usado para(neste projeto) filtrar um recurso (opcional).

Exemplo de Requisição

<http://www.gilsonvargas.com.br/teste/ws/behaviorals/lat-long?per-page=3&page=0&flag=1>

Esse exemplo faz uma requisição com método GET e um recurso denominado ***behaviorals***, que representa dados comportamentais, seguido por **lat-long** que corresponde uma solicitação de dados de **lat**: latitude e **long**: longitude. Os recursos solicitados devem estar separados por hífen. As ***querystrings*** são concatenadas após o "**?**" e são definidas como:

- **per-page**: Referência utilizada para paginar o resultado da pesquisa, nessa requisição foram solicitados três itens, porém é ilimitado;
- **page**: De onde inicia a paginação do resultado da pesquisa, no exemplo, na página 0, para esse exemplo, o resultado será os primeiros três da lista;
- **flag**: valor 1 para registros válidos;
- **datetime-start**: retornar registros a partir da data e hora solicitada;
- **datetime-end**: retornar registros até da data e hora solicitada;

O resultado é apresentado no Código-Fonte 4.2, que representa um retorno com três conjunto de valores, com uma consulta que resulta em três registros. Esse *webservice* serviu para atender o caso de uso Executar Agendamentos(RF08), responsável por guardar os dados de consulta no banco de dados da aplicação local. Com esse serviço em funcionamento, foi possível criar a estrutura de atualização das tabelas da aplicação local que utilizam os dados desse serviço.

```
{
data: [
        {
            "id": 46,
            "idcollar": "$4",
            "lat": -20.479118,
            "long": -54.635620,
            "date": "2015-09-15",
            "time": "05:34:26"
        },
        {
            "id": 47,
            "idcollar": "$4",
            "lat": -20.479118,
            "long": -54.635620,
            "date": "2015-09-15",
            "time": "05:34:26"
        },
        {
            "id": 281,
            "idcollar": "$4",
            "lat": -20.439751,
            "long": -54.725273,
            "date": "2015-09-15",
            "time": "19:15:25"
        }
    ]
}
```

Código-Fonte 4.2 – Exemplo requisição com método GET com resultado de notação em JSON para a funcionalidade de **Rastreamentos**

Agendamento de tarefas

O Titan **Framework** possui um serviço de agendamento de tarefas. Com ele pode-se criar agendamento de execução de *scripts* para determinadas tarefas e serviu como apoio para o caso de uso Executar Agendamentos(RF02). Esse serviço, denominado ***Scheduler Jobs***, permite que se crie execuções por determinado intervalo de tempo, possibilitando a execução de *scripts* no intervalo de: minuto, hora, dia, semana e mês. Cada execução desse *script* aciona o processo que é mostrado na Figura 32. Esse processo denominado **Executar Agendamentos** possui um subprocesso que faz a iteração por agendamento e está descrito no Apêndice D.

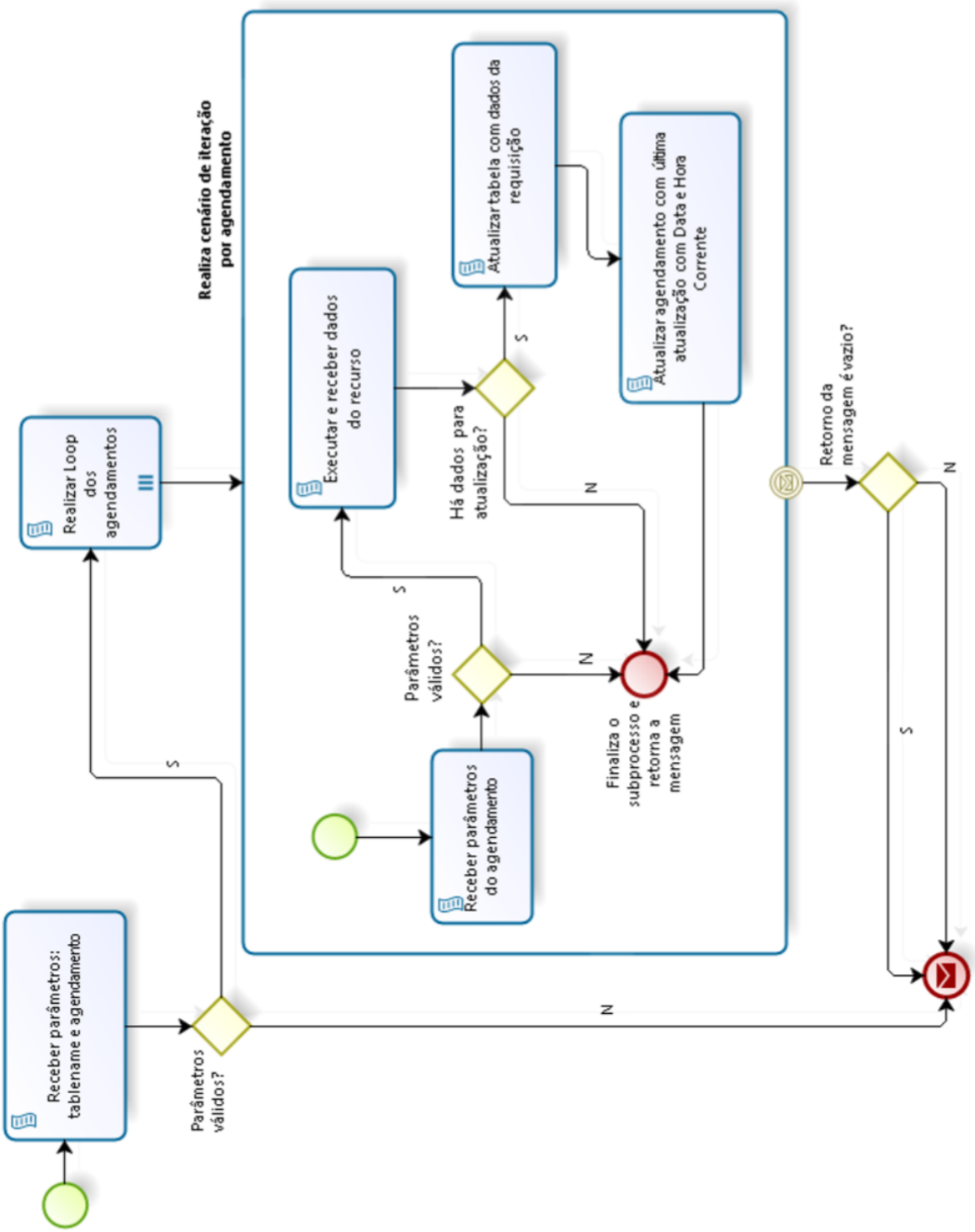


Figura 32 – Processo **Executar Agendamentos** - Realiza as atualizações de agendamentos de atualização de tabelas que utilizam a FarmSPL

O processo **Executar agendamentos** contempla o caso de uso para atualização dos dados das tabelas que utilizam os componentes. A validação dos parâmetros do agendamento foi definido com parâmetros de intervalo e o agendamento é o nome da tabela que será atualizada. Esse processo é executado pelos ***Scheduler Jobs***, um exemplo de execução pode ser visto no Código-Fonte 4.3. A função criada no banco de dados, no *schema* **FarmSPL**, nomeada como **update_tables_farmspl**, faz o processo **Executar agendamentos** por completo de acordo com os dois parâmetros enviados:

1. ***tablename***: nome da tabela/seção que precisa ser/verificar se há atualização;

2. **agendamento**: *hourly*(por hora), *daily*(por dia), *weekly* (por semana), *mounthly*(por mês). Indica a periodicidade da consulta atual, no exemplo *daily*.

```php
<?php
  /*Cria uma conexao Singleton com Titan*/
  $db = Database::singleton ();

  /*String SQL com a execucao da funcao do processo Executar
    Agendamentos no banco*/
  $sql = "select * from
    farmspl.update_tables_farmspl('".$section->getName ()."',
    'daily');"

  $sth = $db->prepare ($sql);
  $sth->execute ();
  $obj = $sth->fetch (PDO::FETCH_OBJ);

  if (!$obj){
      throw new Exception (__ ("does not exist scheduler to daily
      DB!"));
    }

  /*Imprime o retorno a ser gravado no log*/
  echo $obj->update_tables_farmspl. "";

  $cache = Instance::singleton ()->getCachePath ();
  $thumb = $cache . "scheduler/" . str_pad ("daily", 7, "0",
  STR_PAD_LEFT) ."_". $size;
  $cache = Instance::singleton ()->getCachePath ();

?>
```

Código-Fonte 4.3 – Agendamento utilizando ***Scheduler Jobs*** para execução por hora - daily.php

Este exemplo é um agendamento que é executado uma vez por dia, e para que isso ocorra, basta nomeá-lo dentro da pasta **_job/** com o nome **daily.php** e criar o agendamento no *Scheduler Job* do sistema operacional, no exemplo do agendamento, o código 4.4, então serviço executará diariamente às 3 horas da manhã. No caso do Debian, edite o arquivo "/etc/crontab" inserindo a linha abaixo:

Código-Fonte 4.4 – Agendamento do *Scheduler Jobs* para atualização diária

```
$ 0 3 * * * root /usr/local/bin/wget --no-check-
certificate "https://www.minha-instancia.com/titan.php?target=
schedule&hash=31dcbaa78184424c7dd59f83daf429d8&job=daily" -O
/dev/null -o /dev/null
```

Criado os serviços de atualização das tabelas, foi preciso criar um cadastro para automatizar esses agendamentos, tratado no caso de uso (RF01) de GERENCIAR AGENDAMENTOS. Nesse cadastro foi preciso relacionar a seção/tabela com a periodicidade que deveria ser executada, por exemplo: Seção de **Visitas ao cocho** precisa ser atualizada diariamente. Foi feito um cadastro com uma relação da tabela ***trough*** e atualização **por dia**, além do endereço do recurso que irá atualizar, no caso do serviço do *webservice* exemplo criado. A Figura 33 ilustra esse exemplo.

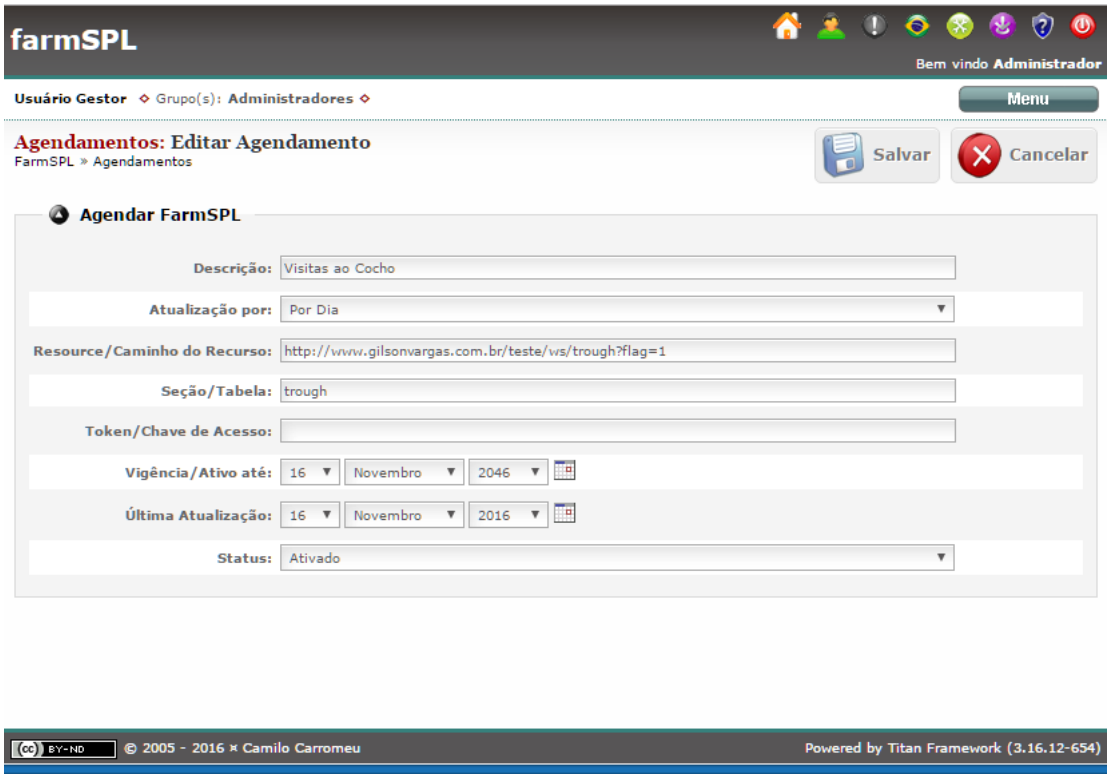


Figura 33 – Exemplo de cadastro de agendamento de execução do serviço em **Gerenciar Agendamentos**

Para esse cadastro foi preciso fazer o procedimento comum de um CRUD fornecido

pelo Titan *Framework*, e então a seção e a tabela no banco de dados foram criadas com o nome de ***scheduler***.

Também é necessário criar sempre uma função SQL no banco de dados para que esse processo ocorra. Essa tarefa existe pois não tem ainda uma definição de como será o funcionamento da camada que fornecerá os dados, então como foi sugerido, cabe ao desenvolvedor criar sempre a função no banco conforme a função apresentada no Código-Fonte E.1.

Sempre que for replicada essa função, deve-se usar o padrão para o nome: **"update_table_geotracking" + "nome da tabela a ser atualizada"**. Para a funcionalidade de Rastreamentos ficou definido o nome **"update_table_geotracking"**. O trecho que deve ser alterado são as colunas que serão atualizadas na tabela relacionada a seção. Esse processo pode ser automatizado futuramente para não haver a necessidade de criar essa função para cada atualização.

4.3 Tecnologias Adotadas

Além das tecnologias utlizadas pelo Titan *Framework*, foram necessárias a utilização de outras incluindo bibliotecas e um *Framework* para desenvolvimento *front end*[5] responsivo. As tecnologias utilizadas no projeto estão listadas abaixo:

- **PostgreSQL** (POSTGRESQL, 2005): O Sistema de Gerenciamento de Banco de Dados é utilizado no desenvolvimento do Titan *Framework*, além de ser software livre é um banco de dados robusto e estável;
- **PHP** (PHP, 2005): A linguagem PHP foi utilizada para implementação e codificação *server-side*. É uma linguagem livre e de código aberto, popular, de uso geral e que é especialmente adequada para o desenvolvimento web;
- **XML** (*Extensible Markup Language*) (BRAY et al., 1998): Linguagem de marcação derivada do SGML (*Standard Generalized Markup Language*), foi utilizada para descrição de tipos de dados e mapeamento das seções do Titan *Framework*;
- **JSON** (CROCKFORD, 2006): um acrônimo para *JavaScript Object Notation*, é um formato leve para troca de dados computacionais, baseado em JavaScript, utilizado para o tráfego de dados no projeto;
- **JavaScript** (CENTER, 2009): é uma linguagem de programação interpretada, atualmente a principal linguagem para programação cliente-servidor em navegadores web, é baseada em ECMAScript padronizada pela Ecma international nas especificações ECMA-262[3] e ISO/IEC 16262;

[5] Etapa final do processo de software apresentado ao usuário final

- **Google *Charts*** (GOOGLE, 2016a): ferramentas de gráficos do Google, poderosas, simples de usar e gratuitas. Possui uma rica galeria de gráficos interativos e ferramentas de dados. Foi utilizada para gerar representação gráfica nos componentes;
- **Google Maps** (GOOGLE, 2016b): é uma API de criação de mapas, rápida e leve, com base em parâmetros de URL enviados por meio de uma solicitação HTTPS padrão. Foi utilizada para visualização dos dados comportamentais na extensão do componente FarmSPL.*behaviorals*.
- **JQuery**: é uma biblioteca JavaScript com muitos recursos, rápida e de arquivo com tamanho pequeno. Melhora a comunicação e manipulação com o documento HTML, fornece assertivas para tratamento de erros, entre outros. É utilizada para comunicação com AJAX (Asynchronous JavaScript e XML) e manipulação dos componentes;
- **Bootstrap** (OTTO; THORNTON, 2013): é o *front end framework* de HTML, CSS e Javascript, mais popular para o desenvolvimento web. Fornece agilidade e confiabilidade no desenvolvimento, além de um arcabouço conceitual muito poderoso para princípios de usabilidade e adaptatividade. Foi usado para desenvolver a interface final dos componentes;
- **Bizagi *Modeler*** (MODELER, 2016): Software de modelagem de processo simples e intuitiva, é uma ferramenta de notação ágil para fluxos de negócio. Foi utilizado para modelar o processo de agendamentos de atualização das tabelas dos componentes e para mostrar os processos do mapeamento sistemático;
- ***Draw*.io** (DRAW.IO, 2016): é um software livre em linha de diagrama para fazer fluxogramas, diagramas de processo, organogramas, diagramas de UML, de ER e de rede. Foi utilizado para elaborar os fazer os modelos na Engenharia de Domínio da LPS.

4.4 Considerações Finais

Este capítulo forneceu a visão ampla do projeto desenvolvido, o processo de LPS sendo aplicado para Pecuária de Precisão. A aplicação do processo utilizou a abordagem PLUS e seus dois ciclos de vida: Engenharia de Domínio e Engenharia da Aplicação. O ciclo se iniciou com a elicitação dos requisitos para criação do Documento de Requisitos, Modelo de Casos de Uso, Modelo de *Features* e Análise do Domínio e o reúso da arquitetura fornecida pelo Titan *Framework* para oferecer apoio a criação dos componentes FarmSPL e suas derivações para fornecer ao desenvolvedor e cliente uma abordagem para gerenciar dados comportamentais, contextuais, fisiológicos e de microclima. No próximo capítulo será apresentado a a Engenharia da Aplicação com um estudo de caso explorando o ambiente.

5 Conforto Térmico - Um estudo de caso utilizando a FarmSPL

5.1 Considerações Iniciais

Neste capítulo é apresentada a Engenharia da Aplicação com a derivação de uma funcionalidade com um estudo de caso no domínio de Pecuária de Precisão, a fim de validar a reutilização de um dos componentes gerados na Engenharia de Domínio. O processo foi aplicado conforme os passos listados na Seção 4.2.3.4. A validação foi realizada por meio de uma funcionalidade que verifica o índice de conforto térmico animal, com base em trabalhos recomendados na literatura. A amostragem de dados foi obtida junto ao levantamento de requisitos descrita na Seção 4.2.1, com informações de pesquisa e análise sobre dados de microclima e condições de conforto térmico animal.

5.2 Descrição do Problema

Uma forma de mensuração possível de condições de conforto térmico animal é a comportamental, de importância tal qual a fisiológica para o estudo do bem-estar animal. Os indicadores comportamentais estão relacionados à ocorrência de reações e comportamentos anormais ou que afastem dos que ocorrem no ambiente natural (OLIVEIRA, 2013).

Para o funcinamento geral dos processos fisiológicos, a temperatura se destaca como um dos principais elementos que condicionam o conforto térmico, pois ela envolve a superfície corporal dos animais e pode influenciar a velocidade das reações no organismo, influenciando, assim, a produção animal (NAZARENO, 2008; OLIVEIRA, 2013). Na literatura e em experimentos são citadas medidas para avaliar o conforto térmico animal, por meio do Índice de Temperatura e Umidade (ITU), Índice de Temperatura de Globo e Umidade (ITGU) e da Carga Térmica Radiante (CTR).

Para este estudo de caso será apresentada a utilização do índice de temperatura e umidade, que é usado para avaliar as condições de conforto térmico e leva em consideração a temperatura de bulbo seco e temperatura de ponto de orvalho. BACCARI; JOHNSON; HAHN (1983) e KELLY; BOND (1971) expressam em termos adimensionais para bovinos pela equação abaixo:

Processo: $\mathbf{ITU = tbs + 0{,}36tpo + 41{,}2}$

Onde:

tbs = temperatura de bulbo seco (ºC);

tpo = temperatura de ponto de orvalho (ºC).

Este estudo de caso objetivou realizar de forma específica a validação de um dos componentes da FarmSPL, utilizando uma representação gráfica para observar o conforto térmico animal por meio do índice de temperatura e umidade, para demonstrar o conforto pela situação térmica do animal (bovino) em quatro situações (HAHN et al., 1985):

Conforto: <= 70

Alerta: 71 - 78

Perigo: 79 - 83

Emergência: > 83

5.3 Instância da FarmSPL

Aproveitando a estrutura já criada no capítulo anterior e o ambiente do Titan *Framework* já evoluído para o domínio, para esta representação foi realizada a configuração do uso de um dos componentes para criar mais um membro da LPS. Assim foram seguidos os mesmos passos citados na Seção 4.2.3.4 e criado uma representação gráfica diária para observar o conforto térmico animal por intermédio do índice de temperatura e umidade.

Para criar essa funcionalidade com a representação foram selecionados os dados do serviço de microclima, dos quais já são oferecidos valores calculados de ITU conforme a fórmula de HAHN et al. (1985). O componente selecionado foi o **FarmSPL.*fisiologicals***, com a representação dos dados de ITU através de um gráfico de barras. Pode-se também utilizar outros gráficos, o Google *Charts* oferece outros tipos como gráfico de pizza, linha, área, entre outros, que podem ser adaptados ao derivar os componentes. A tabela nomeada como *thermalcomfort*, exibida na Figura 34, para organizar os dados necessitou dos seguintes atributos:

- id: chave primária da tabela;
- data: data recuperada, idêntica ao que o serviço do barramento oferece;
- horario: hora recuperada, idêntica ao que o serviço do barramento oferece;
- sistema: indica qual sistema o dado pertence. Na amostragem foram inseridos dados de sistemas de ILPF e ILP;

- itu: índice de temperatura e umidade, calculado utilizando a fórmula de HAHN et al. (1985);
- _user: chave estrangeira com ligação a tabela mandatória do Titan;
- _create: data de criação do registro;
- _update: data de atualização do registro;

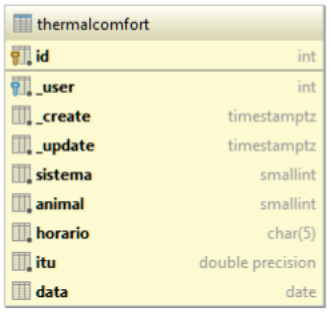


Figura 34 – Tabela do banco de dados *thermalcomfort* para dados de rastreamento e relação com tabela mandatória do Titan

```
1  /*Consulta na tabela thermalcomfort agrupada*/
2  CREATE OR REPLACE VIEW farmspl.vwthermalcomfort_itu (
3      animal, sistema, data, horario, datetime, itu)
4  AS
5  SELECT m.animal, m.sistema, m.data, m.horario,
6  concat_ws(' '::text, substr(m.data::text, 1, 10),
7  m.horario::text)::timestamp without time zone AS datetime, m.itu
8  FROM farmspl.thermalcomfort m
9  GROUP BY m.animal, m.sistema, m.data, m.horario, m.itu
10 ORDER BY m.animal, m.sistema, m.data, m.horario, m.itu;
11
12 /*Consulta por quantidade de registros por animal
13 tabela thermalcomfort agrupada*/
14 CREATE OR REPLACE VIEW farmspl.vwthermalcomfort_por_animal (
15     animal,
16     registros)
17 AS
18 SELECT vw.animal,
19     count(vw.animal) AS registros
20 FROM farmspl.vwthermalcomfort_itu vw
21 GROUP BY vw.animal;
```

Código-Fonte 5.1 – Visão ***vwthermalcomfort_itu*** e ***vwthermalcomfort_por_animal*** criadas no banco de dados

A instrução de criação de uma visão no banco de dados foi adotada pois facilita as consultas aos dados. O artefato **list.xml** foi mapeado para exibir os registros por animal e utiliza a visão ***vwthermalcomfort_por_animal*** que foi criada para exibir os dados agrupados por sistema. A visão ***vwthermalcomfort_itu*** foi criada para exibir os resultados agrupados da tabela ***thermalcomfort***. O código com a criação dessas visões esta exibido no Código-Fonte 5.1.

Depois de serem criadas a tabela e as visões, foi feita a configuração da seção ***thermalcomfort*** com os artefatos (**"config.inc.xml"**, **"list.inc.xml"**, **"farmsplview.xml"** e **"farmsplfile.xml"**) ilustrados na Figura 35.

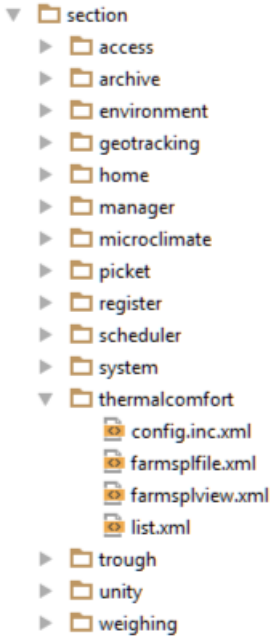


Figura 35 – Estrutura da seção ***Thermalcomfort***

Foram criadas a tabela e visões, logo após foi feito o mapeamento no arquivo **business.xml** na pasta **configure/**, exibido no Código-Fonte 5.2.

```
1 <!-- SubMenu Registros de ITU -->
2     <section
3                 label="ITU Registers | pt_BR: Registros de ITU"
4                 name="thermalcomfort"
5                 component="local/component/farmSPL/"
6                 father="farmSPL" />
```

Código-Fonte 5.2 – Mapeamento da seção de **Registros de ITU - business.xml**

Depois foi feito o mapeamento no **config.inc.xml**, exibido no Código-Fonte 5.3 e o artefato **list.xml**, exibido no Código-Fonte 5.4.

```
1
2  <?xml version="1.0" encoding="UTF-8"?>
3  <action-mapping>
4  <!-- CARACTERISTICAS PREDEFINIDAS DO COMPONENT -->
5   <directive name="_NAME_" value="farmspl.fisiologicals"/>
6   <directive name="_RESOURCE_" value="thermalcomfort"/>
7   <directive name="_MEASURE_" value="itu"/>
8   <directive name="_TITLE_" value="Variação do itu / Horas do dia"/>
9   <directive name="_DESCRIPTION_" value="ITU"/>
10 <!-- VARIAVEIS DE PESQUISA -->
11  <directive name="itemId" value="animal"/>
12  <directive name="timeStart" value="horario" />
13  <directive name="timeEnd" value="horario" />
14  <directive name="dateStart" value="data" />
15  <directive name="dateEnd" value="data" />
16  <directive name="time" value="horario" />
17  <directive name="date" value="data" />
18  <directive name="datetimeStart" value="(concat_ws(' ',
19  substr(data::text, 1,10),horario::text))::TIMESTAMP" />
20 <directive name="datetimeEnd" value="(concat_ws(' ',
21 substr(data::text, 1, 10), horario::text))::TIMESTAMP" />
22  <directive name="datetime" value= "(concat_ws(' ',
23  substr(data::text, 1, 10), horario::text))::TIMESTAMP" />
24 <directive name="sistema" value="sistema" />
25
26   <!-- listagem de Rastreamentos -->
27   <action name="list"
28     label="ITU | pt_BR: Registros de ITU - Indice de
29     Temperatura e Umidade"
30     default="true">
31          <menu function="search"/>
32   </action>
33   <!-- Aponta o action para a Pesquisa  - farmsplview.php -->
34   <action name="farmsplview" label="Buscar Registros">
35   </action>
36   <!-- Aponta o action para gerar o Metadados JSON - farmsplfile.php -->
37   <action name="farmsplfile" label="Arquivo"
38           description="Registros de ITU - Indice de
39           Temperatura e Umidade">
40   </action>
41 </action-mapping>
```

Código-Fonte 5.3 – Mapeamento das ações da seção de **Registros de ITU - config.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- lista dos itens : item e quantidade de registro por item -->
    <view table="farmspl.vwthermalcomfort_por_animal"
    primary="animal" paginate="15">
        <field type="Integer" column="animal" label="Animal" />
        <field type="String" column="registros"
        label="ITU Registers | pt_BR: Registros de ITU" />
        <icon  function="[ajax]"  action="farmsplview"
               table="farmspl.microclimate"  primary="id" column="id"
               image="view.gif"  label="View | pt_BR: Visualizar"
               default="true" />
    </view>
    <search table="farmspl.vwthermalcomfort\_por\_animal">
        <field type="String" column="animal"
        label="Identificação do Animal - ID" />
</search>
```

Código-Fonte 5.4 – Configuração de listagem da seção de **Registros de ITU - list.xml**

Essa listagem se assimila as citadas no Capítulo anterior, o que mudou nesse caso foi a pesquisa acionada quando clicada no ícone da lupa visualizar do registro. Foi acrescentado na opção de busca além da data inicial e data final a opção do tipo sistema a escolher. A configuração que foi feita no metadado **farmsplview.xml** possui uma *tag* a mais para a opção de sistema, exibido no Código-Fonte 5.5. A requisição é orientada pelo **farmsplfile.xml**, exibido no Código-Fonte 5.6, com o mapeamento para geração do resultado da consulta.

```
<?xml version="1.0" encoding="UTF-8"?>
<search table="farmspl.thermalcomfort">
    <field type="Date" id="dateStart"  column="date" required="true"
           label="Begin Date | pt_BR: Data Inicial" first-year="2000"
           last-year="[now]" show-age="true" format="YYYY/MM/DD" />
    <field type="Time" id="timeStart" column="time" required="true"
           label="Begin Time | pt_BR: Hora Inicial" format="HH:ii:ss" />
    <field type="Date" id="dateEnd"  column="date"  required="true"
           label="End Date | pt_BR: Data Final" first-year="2000"
           last-year="[now]" show-time="[now]" show-age="true"
           format="YYYY/MM/DD" />
    <field type="Time" id="timeEnd" column="time" required="true"
           label="End Time | pt_BR: Hora Final"  format="HH:ii:ss" />
    <field type="Select" id="sistema" column="sistemanome"
            label="System | pt_BR: Sistema" required="true"
            link-table="farmspl.vwmicroclimate_por_sistema"
            link-column="sistema" link-view="sistemanome" />
</search>
```

Código-Fonte 5.5 – Configurações para Pesquisa de **Registros de ITU - farmsplview.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- A ser renderizado no documento - farmsplfile.php -->

<view table="farmspl.vwthermalcomfort_itu" primary="animal">
   <field type="Integer" column="animal"/>
    <field type="Integer" column="sistema" />
    <field type="Date" column="datetime" id="datetime"
        tip="Data/Hora" show-time="true" format="DD/MM/YYYY HH:ii:ss" />
    <field type="Float" column="itu" />
</view>

<!-- Utiliza o search para complementar a pesquisa com as variaveis
          GETADAS na requisicao -->
<search table="farmspl.vwthermalcomfort_itu">
</search>
```

Código-Fonte 5.6 – Mapeamento do resultado da processamento da pesquisa para a funcionalidade de **Registros de ITU - farmsplfile.xml**

A renderização da listagem com a execução da pesquisa é ilustrada na Figura 36. Essa busca é executada no **farmspl.*fisiologicals***, pré-definido na ***directive*** "_NAME_", conforme o mapeamento feito no **config.inc.xml** da seção.

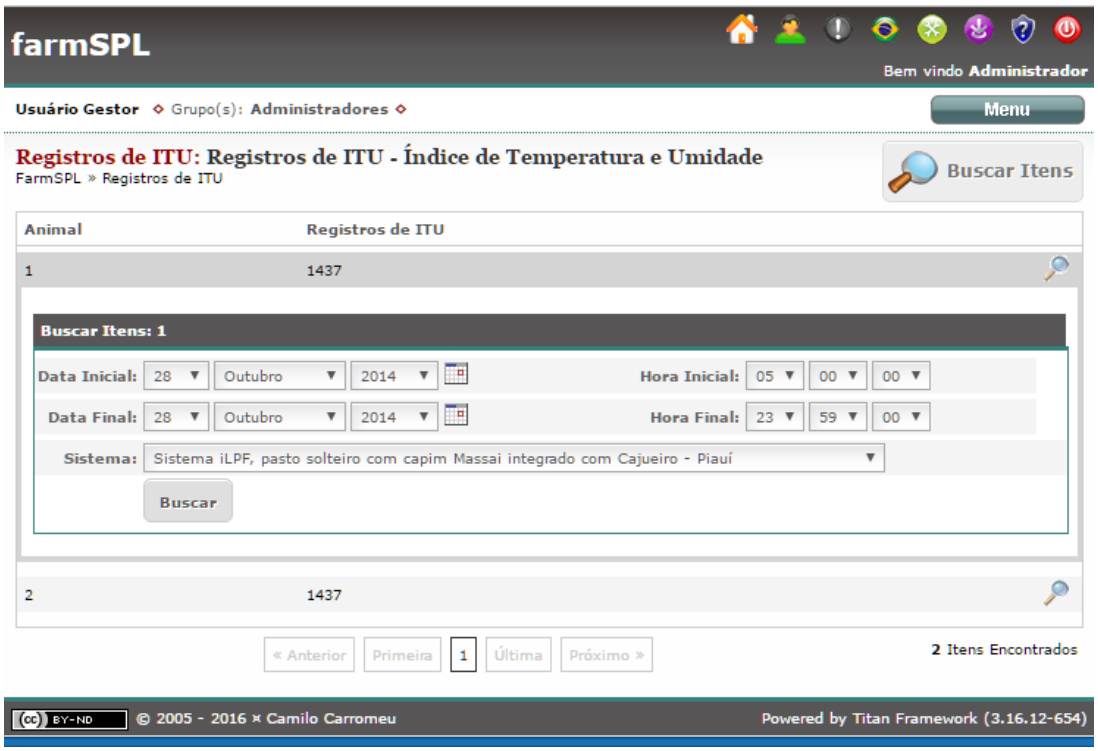


Figura 36 – Pesquisa após clicar em visualizar na lista de **Registros de ITU**

A seção foi criada e relacionada a pesquisa com os artefatos devidamente mapeados. O resultado dessa pesquisa é a geração da visualização dos registros de ITU no componente

FarmSPL.*fisiologicals* apresentada na Figura 37. Nessa visualização consta um histórico por dia do animal, demonstrando a variação do ITU e a sua situação térmica, conforme foi definido na descrição do problema, na Seção 5.2.

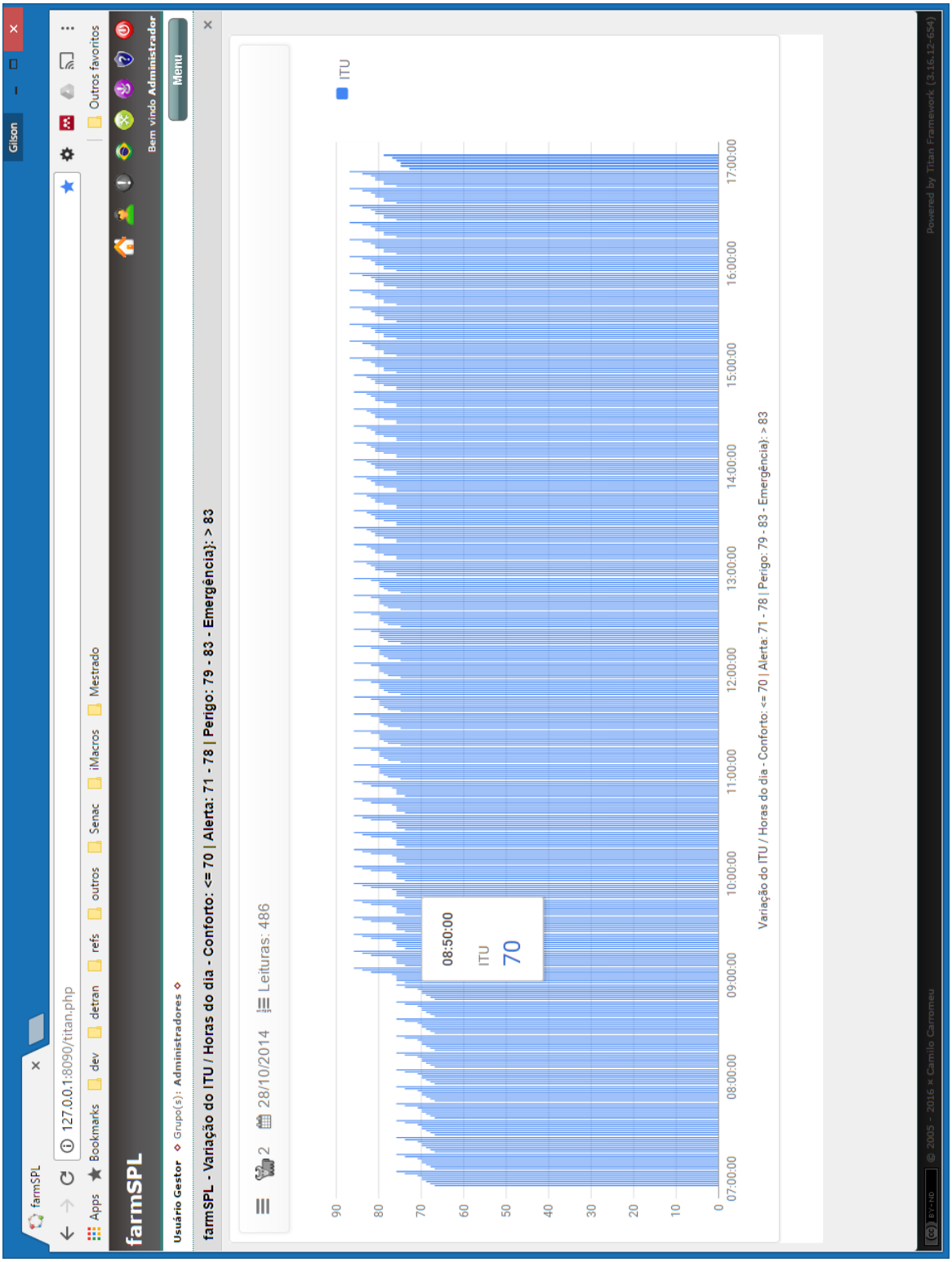
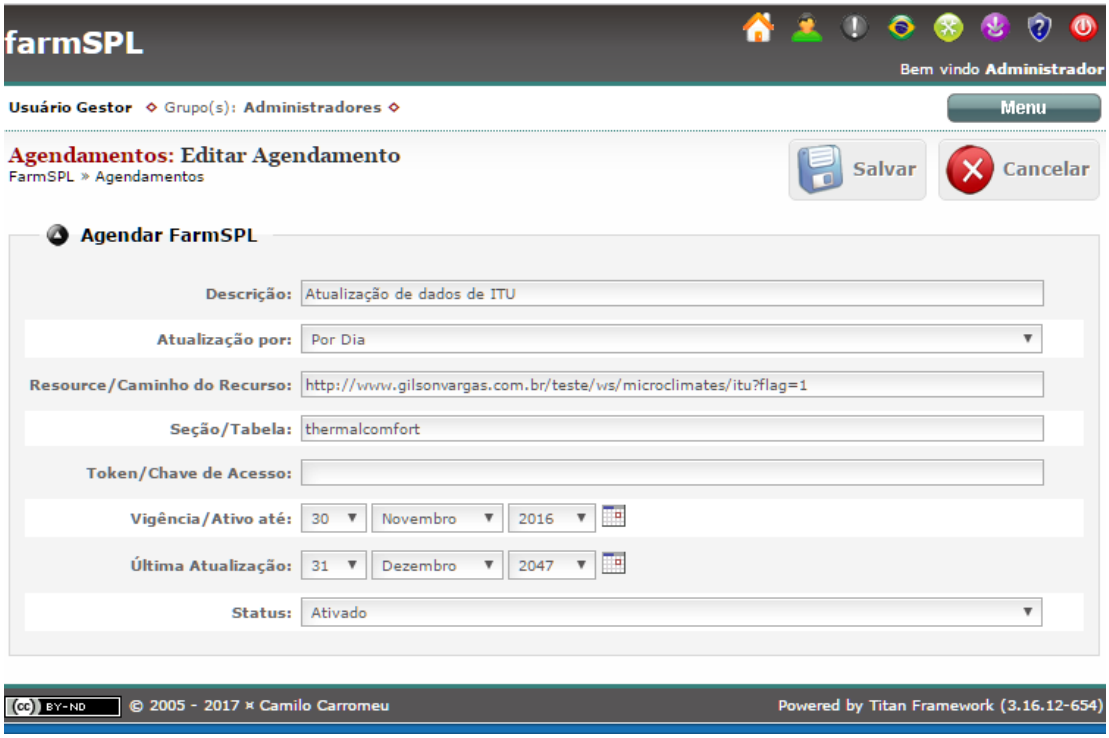


Figura 37 – Pesquisa após clicar em buscar na lista de animais com **Registros de ITU**

Após concluídas essas etapas, restou criar o agendamento de atualização para tabela ***thermalcomfort***, na funcionalidade criada previamente de GERENCIAR AGENDAMENTOS, conforme exibido na Figura 38.

Figura 38 – Exemplo de cadastro de agendamento de execução do serviço em **Gerenciar Agendamentos**



5.4 Considerações Finais

Foi abordado neste capítulo a Engenharia de Aplicação com a geração de um novo membro da LPS no domínio de Pecuária de Precisão. Essa criação se iniciou a partir da arquitetura, já definida no Capítulo 4. Então foi seguido o processo para criação de uma funcionalidade para gerar a visualização de uma representação gráfica para observar o conforto térmico animal através do índice de temperatura e umidade, para demonstrar o conforto através da situação térmica do animal, para por fim validar o componente FarmSPL.*fisiologicals*.

Para esta funcionalidade ser gerada precisaram ser criadas as visões (mostradas no Código-Fonte 5.1) no banco de dados, para retornar os dados das consultas de forma mais organizada e a tabela apresentada na Figura 34, fora esse trabalho, o restante foi de reutilização, que durou o tempo de 30 minutos entre testes e mapeamentos. Isso promove de maneira significativa a redução de esforço da produção e traz também benefícios com a economia de tempo.

6 Conclusão

6.1 Contribuições

As contribuições deste trabalho de mestrado são principalmente no contexto de reúso. O projeto foi identificado através de uma lacuna, com a ideia de melhorar o processo de desenvolvimento de aplicações para web através do processo de uma LPS e trabalhou pela principal finalidade que é a de reúso. A abordagem foi desenvolvida e junto a ela os componentes para pecuária de precisão, contribuindo para o processo de desenvolvimento de software. Brevemente essa abordagem pode ser utilizada para apoiar de desenvolvimento de novos softwares ou produtos e contribuir através dos seguintes benefícios:

- **Economia**, devido ao menor custo e pelo reúso da arquitetura já testada;
- **Agilidade**, por usar tecnologia com protocolo de estado com o tráfego de informações utilizando arquivos no formato JSON;
- **Eficiência**, por poder utilizar a renderização dos componentes em desktops, tablets, smartphones, dentre outros;
- Potencial aumento da **qualidade** do software, pois a abordagem faz o usuário seguir um processo de desenvolvimento;

6.2 Limitações

Algumas limitações foram constatadas quanto ao desenvolvimento do projeto e a abordagem proposta. Dentre elas as principais são listadas a seguir:

- Não foi desenvolvido um processo de reutilização de código automático, ou plugin para geração de código, o que facilitaria e agilizaria o reúso dos artefato;
- A abordagem não oferece um processo automatizado completo, sempre que for criado um agendamento precisa ser replicada uma função, apresentada no Código-Fonte E.1, e alterados os campos que devem ser atualizados na tabela referente. Embora ofereça o código completo, o que facilita o processo de criação da automatização;
- O fato de ter que criar um serviço de *webservice* que serviu como uma sugestão para o item **e** da Figura 1 da plataforma ***e-Cattle***, atrasou o desenvolvimento do projeto e criou essas duas limitações citadas anteriormente;

- Não foi criada toda a estrutura para integração de gráficos do Google *Charts*, porém o código de renderização dos componentes já mostra o processo de como implantar um novo tipo gráfico, através da replicação e fácil alteração.
- A renderização dos componentes está limitada a exibição de gráficos, porém o código pode ser alterado e ajustados outros tipos de acordo com que o desenvolvedor necessitar, ou simplesmente recuperar a requisição do arquivo JSON para usar para outro tipo como relatórios (*Reports*);

6.3 Trabalhos Futuros

Neste trabalho foram observadas as limitações e junto a inspiração para trabalhos futuros. Dentre elas as principais são:

- Automatizar totalmente o processo agendamento de tarefas fornecendo ao usuário uma ferramenta mais conclusiva nesse processo;
- Após o trabalho de desenvolvimento da camada **e** da Figura 1 ser finalizada, alinhar com este trabalho para automatizar o processo da plataforma;
- Criar o processo de geração automática dos ativos para melhorar o tratamento dos componentes FarmSPL;
- Aumentar as opções de geração de visualização, assim como geração de impressão e salvar documentos;
- Desenvolver uma API de de *Garbage Collector* para guardar dados antigos de cada tabela dos componentes para manter o desempenho de consultas sempre rápidas; e
- Outro importante trabalho que deve ser feito é a utilização de um identificador único universal (do inglês *Universal Unique IDentifier* ou **UUID**[1]). Esse padrão possibilita aos sistemas distribuídos identificarem unicamente informações sem coordenação central significante, e servirá para resolver problemas futuros de conflitos de colisão de identificadores (ID).

[1] Universally Unique IDentifier: <https://tools.ietf.org/html/rfc4122>

Referências

ABU-MATAR, M.; GOMAA, H. Variability Modeling for Service Oriented Product Line Architectures. In: *2011 15th International Software Product Line Conference*. IEEE, 2011. p. 110–119. ISBN 978-1-4577-1029-2. Disponível em: <http://ieeexplore.ieee.org/document/6030052/>.

BACCARI, F.; JOHNSON, H.; HAHN, G. L. Environmental heat effects on growth, plasma t3, and postheat compensatory effects on holstein calves. *Experimental Biology and Medicine*, SAGE Publications, v. 173, n. 3, p. 312–318, 1983.

BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E.; CHAMPION, M.; FERRIS, C.; ORCHARD, D. Web services architecture. 2004.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E.; YERGEAU, F. Extensible markup language (xml). *World Wide Web Consortium Recommendation REC-xml-19980210*, v. 16, p. 16, 1998. Disponível em: <http://www.w3.org/TR/1998/REC-xml-19980210>.

CARROMEU, C. *Linha de produtos de software no processo de geração de sistemas Web de apoio a gestão de fomento de projetos*. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, 2007.

CARROMEU, C. Titan framework cookbook. In: . [s.n.], 2014. v. 1. Disponível em: <http://www.titanframework.com/Cookbook.pdf>.

CARROMEU, C. *e-Cattle: Uma Plataforma de IoT para Pecuária de Precisão*. Tese (Doutorado) — Faculdade de Computação da Universidade Federal de Mato Grosso do Sul, 2016.

CARROMEU, C.; PAIVA, D. M. B.; CAGNIN, M. I. The evolution from a web SPL of the e-gov domain to the mobile paradigm. In: . [S.l.]: Springer Verlag, 2015. v. 9155, p. 217–231. ISBN 9783319214030. ISSN 16113349.

CARROMEU, C.; PAIVA, D. M. B.; CAGNIN, M. I.; RUBINSZTEJN, H. K. S.; TURINE, M. A. S.; BREITMAN, K. Component-based architecture for e-gov web systems development. In: IEEE. *Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on*. [S.l.], 2010. p. 379–385.

CENTER, M. D. Javascript. v. 5, 2009. Disponível em: <https://developer.mozilla.org/en/About_JavaScript.-{Ú}ltimavi>.

CLEMENTS, P.; NORTHROP, L. *Software product lines*. [S.l.]: Addison-Wesley„ 2002.

CNA, C. d. A. e. P. d. B. Boletim pib: Pib do agronegócio cresceu 4,28% de janeiro a outubro de 2016 / janeiro de 2017. 2017. Disponível em: <http://www.cnabrasil.org.br/boletins/boletim-pib-pib-do-agronegocio-cresceu-428-de-janeiro-outubro-de-2016-janeiro-de-2017>.

COHEN, S. *Product line state of the practice report*. [S.l.], 2002.

COHEN, S. *Predicting when product line investment pays*. [S.l.], 2003.

CROCKFORD, D. *JSON (JavaScript Object Notation)*. Technical report, RFC 4627, 2006, 2006, 2006. Disponível em: <http://www.ietf.org/rfc/rfc4627.txt>.

CZARNECKI, K.; EISENECKER, U. W. Generative programming. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, Springer, 2000.

DETMANN, E.; PAULINO, M. F.; CABRAL, L.; FILHO, S. V.; CECON, P. R.; ZERVOUDAKIS, J. T.; LANA, R.; LEÃO, M. I.; MELO, A. Simulação e validação de parâmetros da cinética digestiva em novilhos mestiços suplementados a pasto, por intermédio do sistema in vitro de produção de gases. *Revista Brasileira de Zootecnia*, SciELO Brasil, v. 34, n. 6, p. 2112–2122, 2005.

DRAW.IO. *Software livre em linha do diagrama para fazer fluxogramas, diagramasde processo, organogramas, diagramas de UML, de ER e de rede*. 2016. Disponível em: <http://www.draw.io/>.

DURSCKI, R. C.; SPINOLA, M. M.; BURNETT, R. C.; REINEHR, S. S. Linhas de produto de software: riscos e vantagens de sua implantação. *Simpósio Brasileiro de Processo de Software. S. Paulo*, 2004.

DYBÅ, T.; DINGSØYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: *ESEM*. [S.l.: s.n.], 2007. v. 7, p. 225–234.

EMBRAPA. *Gestão - Pecuária de precisão*. 2014. Disponível em: <https://www.youtube.com/watch?v=iChH-nA_Xj0>.

EMBRAPA. *IV Plano Diretor da Embrapa Gado de Corte*. 2015. <https://www.embrapa.br/documents/1355108/1528910/PDU/8e551651-0870-458d-a635-0fceaabdd37a?version=1.0>. Accessed january, 2015.

FERRAIOLO, D.; CUGINI, J.; KUHN, D. R. Role-based access control (rbac): Features and motivations. In: *Proceedings of 11th annual computer security application conference*. [S.l.: s.n.], 1995. p. 241–48.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Tese (Doutorado) — University of California, Irvine, 2000.

GOMAA, H. Designing software product lines with uml. In: *SEW Tutorial Notes*. [S.l.: s.n.], 2005. p. 160–216.

GOOGLE. *Gráficos interativos para navegadores e dispositivos móveis*. 2016. Disponível em: <https://developers.google.com/chart/>.

GOOGLE. *A solução para aplicativos do Google Maps para dispositivos desktop e móveis*. 2016. Disponível em: <https://developers.google.com/maps/documentation/javascript/>.

HAHN, G. et al. Management and housing of farm animals in hot environments. *Stress physiology in livestock. Volume II. Ungulates.*, CRC Press, Inc., p. 151–174, 1985.

JOHNSON, R.; WIRFS-BROCK, R. Object-oriented frameworks. tutorial notes. In: *Proceedings of ACM OOPSLA*. [S.l.: s.n.], 1991.

KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S. *Feature-oriented domain analysis (FODA) feasibility study*. [S.l.], 1990.

KELLY, C.; BOND, T. Bioclimatic factors and their measurement. *A guide to environmental research on animals. Washington: National Academy of Sciences*, p. 71–92, 1971.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, v. 33, n. 2004, p. 1–26, 2004.

KITCHENHAM, B. A.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. In: *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. [S.l.: s.n.], 2007.

KITCHENHAM, B. A.; DYBA, T.; JORGENSEN, M. Evidence-based software engineering. In: IEEE COMPUTER SOCIETY. *Proceedings of the 26th international conference on software engineering*. [S.l.], 2004. p. 273–281.

KRUEGER, C. W. Software reuse. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 24, n. 2, p. 131–183, jun. 1992. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/130844.130856>.

LACHTERMACHER, G. *Pesquisa Operacional na Tomada de Decisão*. 4nd. ed. [S.l.]: Editora Campus, 2009.

LUCRÉDIO, D. Uma abordagem orientada a modelos para reutilização de software. *INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO UNIVERSIDADE DE SÃO PAULO*, p. 37, 2009.

MACEDO, L. O. B. *Perfil de governança e a coordenação de alianças estratégicas do sistema agroindustrial da carne bovina brasileira*. Tese (Doutorado) — Escola Superior de Agricultura Luiz de Queiroz, 2009.

MARINHO, F. G.; ANDRADE, R. M. C.; WERNER, C.; VIANA, W.; MAIA, M. E. F.; ROCHA, L. S.; TEIXEIRA, E.; FILHO, J. a. B. F.; DANTAS, V. L. L.; LIMA, F.; AGUIAR, S. MobiLine: A Nested Software Product Line for the Domain of Mobile and Context-aware Applications. *Sci. Comput. Program.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 78, n. 12, p. 2381–2398, 2013. ISSN 0167-6423. Disponível em: <http://dx.doi.org/10.1016/j.scico.2012.04.009>.

MENDELEY. *Release Notes for Mendeley Desktop v1.12.3*. 2014. <http://www.mendeley.com/release-notes/v1_12_3/>. Accessed december, 2014.

MODELER, B. *Software de modelagem de processo simples e intuitiva*. 2016. Disponível em: <http://www.bizagi.com/>.

MURGUZUR, A.; CAPILLA, R.; TRUJILLO, S.; ORTIZ, O.; LOPEZ-HERREJON, R. E. Context Variability Modeling for Runtime Configuration of Service-based Dynamic Software Product Lines. In: *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2*. New York, NY, USA: ACM, 2014. (SPLC '14), p. 2–9. ISBN 978-1-4503-2739-8. Disponível em: <http://doi.acm.org/10.1145/2647908.2655957>.

NAZARENO, A. *Influência de diferentes sistemas de criação na produção de frangos de corte industrial com ênfase no bem-estar animal*. Tese (Doutorado) — Dissertação (Mestrado em Engenharia Agrícola). Departamento de Tecnologia Rural. Universidade Federal Rural de Pernambuco.[Links], 2008.

NORTHROP, L.; CLEMENTS, P.; BACHMANN, F.; BERGEY, J.; CHASTEK, G.; COHEN, S.; DONOHOE, P.; JONES, L.; KRUT, R.; LITTLE, R. et al. A framework for software product line practice, version 5.0. *SEI.–2007–http://www.sei.cmu.edu/productlines/index.html*, 2007. Disponível em: <http://www.sei.cmu.edu/productlines/frame_report/terminology.htm>.

NORTHROP, L. M. Sei's software product line tenets. *IEEE software*, IEEE Computer Society, v. 19, n. 4, p. 32–40, 2002.

OLIVEIRA, C. C. d. Desempenho e comportamento ingestivo diurno de novilhas nelore em sistemas integrados de produção no cerrado brasileiro. UFVJM, 2013.

OMG. Object management group, specification, business process modeling notation. *Needham, MA, USA*, v. 2, n. 2, 2005.

OTTO, M.; THORNTON, J. Bootstrap. *Twitter Bootstrap*, 2013.

PARNAS, D. L. On the design and development of program families. *IEEE Transactions on software engineering*, IEEE, n. 1, p. 1–9, 1976.

PHP, P. Hypertext preprocessor. *Internet WWW page(versão atual de 22 de Maio de 1999)*, 2005. Disponível em: <http://www.php.net>.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. van D. *Software product line engineering: foundations, principles and techniques*. [S.l.]: Springer Science & Business Media, 2005.

POSTGRESQL. Postgresql, documentation. *Acesso em*, v. 8, 2005. Disponível em: <http://www.postgresql.org>.

RANDELL, B.; NAUR, P. *Software engineering in 1968*. [S.l.], 1968. 1–10 p.

REINHR, S. d. S.; PESSÔA, M. S. d. P.; DURSCKI, R. C.; SPINOLA, M. d. M.; PALUDO, M. A.; BURNETT, R. C. R. Linhas de produto de software: tornando realidade o reúso sistematizado de software. RNTI – Revista Negócios e Tecnologia da Informação, p. 69–83, 2007.

SCHNEIDER, P. S. Termometria e psicrometria. *Departamento de Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, Porto Alegre*, 2008. Disponível em: <http://www.ufrgs.br/medterm/areas/area-i/termometria.pdf>.

SOCHOS, P.; RIEBISCH, M.; PHILIPPOW, I. The Feature-Architecture Mapping (FArM) method for feature-oriented development of software product lines. In: *Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems*. [s.n.], 2006. p. 308–316. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-34249996808&partnerID=40&md5=f5dc94ee6da233be360731de9f5bd93f>.

SOMMERVILLE, I. *Software Engineering*. 9nd. ed. [S.l.]: Addison Wesley, 2011. v. 22. 792 p. ISBN 978-0137035151.

USDA, U. S. D. of A. *Foreign Agricultural Service*. 2014. <http://apps.fas.usda.gov/psdonline/>. Accessed october, 2014.

VARGAS, R. V.; IPMA-B, P. Utilizando a programação multicritério (analytic hierarchy process-ahp) para selecionar e priorizar projetos na gestão de portfólio. In: *PMI Global Congress*. [S.l.: s.n.], 2010. p. 1–22.

Apêndices

APÊNDICE A – Condução das Buscas do Mapeamento Sistemático

Esse apêndice mostra os resultados ocorridos durante o processo de Mapeamento Sistemático. Nas tabelas abaixo são apresentadas as *strings* utilizadas para cada base e sua correlação.

Tabela 15 – Fonte de Pesquisa: Base de busca ACM *Digital Library*

***String* de Busca Adaptada**
(SPL OR "PRODUCT LINE"OR "PRODUCT FAMILY"OR "PRODUCT FAMILIES"OR "PRODUCT-LINE"OR "PRODUCT-FAMILY"OR "PRODUCT-FAMILIES"OR "REPOSITORY") AND (SOFTWARE OR SYSTEM) AND (FARM OR CATTLE OR "PRECISION LIVESTOCK"OR "PRECISION AGRICULTURE")
Aplicado Filtro:
Idioma da pesquisa=Inglês

Tabela 16 – Fonte de Pesquisa: Base de busca IEEE Xplore *Digital Library*

***String* de Busca Adaptada**
(SPL OR "PRODUCT LINE"OR "PRODUCT FAMILY"OR "PRODUCT FAMILIES"OR "PRODUCT-LINE"OR "PRODUCT-FAMILY"OR "PRODUCT-FAMILIES"OR "REPOSITORY") AND (SOFTWARE OR SYSTEM) AND (FARM OR CATTLE OR "PRECISION LIVESTOCK"OR "PRECISION AGRICULTURE")
Aplicado Filtro:
Idioma da pesquisa=Inglês

Tabela 17 – Fonte de Pesquisa: Base de busca *ScienceDirect*

***String* de Busca Adaptada**
TITLE-ABSTR-KEY(((SPL or "PRODUCT LINE"or "PRODUCT FAMILY"or "PRODUCT FAMILIES"or "PRODUCT-LINE"or "PRODUCT-FAMILY"or "PRODUCT-FAMILIES"or REPOSITORY) and (SOFTWARE or SYSTEM) and (FARM or CATTLE or "PRECISION LIVESTOCK"or "PRECISION AGRICULTURE")))
Aplicado Filtro:
Idioma da pesquisa=Inglês

Tabela 18 – Fonte de Pesquisa: Base de busca *Scopus*

***String* de Busca Adaptada**
TITLE-ABS-KEY(((SPL or "PRODUCT LINE"or "PRODUCT FAMILY"or "PRODUCT FAMILIES"or "PRODUCT-LINE"or "PRODUCT-FAMILY"or "PRODUCT-FAMILIES"or REPOSITORY) and (SOFTWARE or SYSTEM) and (FARM or CATTLE or "PRECISION LIVESTOCK"or "PRECISION AGRICULTURE"))) AND PUBYEAR > 2005
Aplicado Filtro
Tempo estipulado>2005

Tabela 19 – Fonte de Pesquisa: Base de busca *Web of Science*

***String* de Busca Adaptada**
(TI=SPL OR TI="PRODUCT LINE"OR TI="PRODUCT FAMILY"OR TI="PRODUCT FAMILIES"OR TI="PRODUCT-LINE"OR TI="PRODUCT-FAMILY"OR TI="PRODUCT-FAMILIES"OR TI=REPOSITORY) AND (TI=SOFTWARE OR TI=SYSTEM) AND (TI=FARM OR TI=CATTLE OR TI="PRECISION LIVESTOCK"OR TI="PRECISION AGRICULTURE")
Aplicado Filtro:
Idioma da pesquisa=Inglês

Tabela 20 – Fonte de Pesquisa: Base de busca *Scielo*

***String* de Busca Adaptada**
(SPL or "PRODUCT LINE"or "PRODUCT FAMILY"or "PRODUCT FAMILIES"or "PRODUCT-LINE"or "PRODUCT-FAMILY"or "PRODUCT-FAMILIES"or "REPOSITORY") and (SOFTWARE or SYSTEM) and (FARM or CATTLE or "PRECISION LIVESTOCK"or "PRECISION AGRICULTURE") Aplicado Filtro: Idioma da pesquisa=Inglês

APÊNDICE B – Formulário de Elicitação de Requisitos

Tabela 21 – Formulário de Elicitação de Requisitos

Formulário de Elicitação de Requisitos - Sensor	
Nome	**GPS**
Descrição	Fornece dados de posição, latitude e longitude, horário e data de posição, qualidade da posição, informações de hora, distância/tempo, rumo percorrido, velocidade, declinação magnética, rumo percorrido, qualidade da posição e satélites visíveis.
Variáveis/Tipo de dados	1 - **$GPGGA** (solução de posição determinada): fornece a posição representada por latitude e longitude, o horário em que a posição foi obtida e sua validade; 2 - **$GPGSV** (satélites visíveis); 3 - **$GPGSA** (DOP e satélites ativos): fornece informações sobre a qualidade da posição obtida; 4 - **$GPRMC** (Recommended minimum specific GPS/Transit data): fornece a posição representada pela latitude e longitude, informações de hora, validade, data, velocidade e declinação magnética; 5 - **$GPVTG** (rumo percorrido).
Tipos de Dados	(X) Comportamentais ( ) Contextuais ( ) Fisiológicos ( ) Microclima
Projeto(Título)	Uma Rede de Sensores sem Fio para Monitoramento Bovino.
Autores/Data	João Felipe Resende Nacer e Irineu Sotoma. Junho de 2015.
Exemplo de Dados Coletados	**IDcolar**: $1 **lat**: -20.439884 **long**: -54.725243 **alt**: 514.40002 **data**: 150915 **hora** (gps): 18:44:15 **velocidade m/s**: 0.00000 **num satelites**: 8 **timestamp**: 18441545

APÊNDICE C – Códigos de Fonte - Artefatos FarmSPL

```
1
2  <?xml version="1.0" encoding="UTF-8"?>
3  <!-- lista: item e quantidade de registro por item -->
4  <view table="farmspl.vwgeotracking_qtde" primary="idcollar"
5  paginate="15">
6      <field type="String" column="idcollar"
7      label="ID | pt_BR: Identificação(id do collar)" />
8      <field type="String" column="rastreamentos"
9      label="Geotracking | pt_BR: Rastreamentos" />
10     <icon  function="[ajax]"  action="farmsplview"
11            table="farmspl.geotracking"  primary="idcollar"
12            column="idcollar" image="view.gif"
13            label="View | pt_BR: Visualizar"
14            default="true" />
15 </view>
16 <!-- busca por id do animal -->
17 <search table="farmspl.geotracking">
18 <field type="String" column="idcollar"
19         label="ID | pt_BR:  Identificação(Animal)" />
20 </search>
```

Código-Fonte C.1 – Configuração de listagem da seção de **Rastreamentos - list.xml**

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <action-mapping>
3    <!-- CARACTERISTICAS PREDEFINIDAS DO COMPONENT -->
4    <directive name="_NAME_" value="farmspl.behaviorals" />
5    <directive name="_RESOURCE_" value="vwgeotracking" />
6    <directive name="_MEASURE_" value="animal,lng,lat" />
7    <directive name="_TITLE_" value="Rastreamentos" />
8    <directive name="_DESCRIPTION_" value="Latitude e Longitude" />
9
10   <!-- VARIAVEIS DE PESQUISA -->
11   <directive name="timeStart" value="time" />
12   <directive name="timeEnd" value="time" />
13   <directive name="dateStart" value="date" />
14   <directive name="dateEnd" value="date" />
15   <directive name="time" value="time" />
16   <directive name="date" value="date" />
```

```
17    <directive name="flag" value="flag" />
18    <directive name="itemId" value="idcollar" />
19    <directive name="datetimeStart" value='("date" + "time")' />
20    <directive name="datetimeEnd" value='("date" + "time")' />
21    <directive name="datetime" value='("date" + "time")' />
22
23    <!-- listagem de Rastreamentos -->
24    <action name="list" label="Geotracking | pt_BR: Listar Rastreamentos"
25            default="true">
26           <menu function="search"/>
27    </action>
28    <!-- Aponta o action para Pesquisa - farmsplview.php -->
29    <action name="farmsplview" label="Buscar Registros">
30    </action>
31    <!-- Aponta o action para gerar o Metadados JSON - farmsplfile.php -->
32    <action name="farmsplfile" label="Arquivo"
33            description="Rastreamentos">
34    </action>
35  </action-mapping>
```

Código-Fonte C.2 – Mapeamento das ações da seção de **Rastreamentos - config.xml**

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- Campos de Formulário a serem utilizados para item Escolhido -->
3  <search table="farmspl.vwgeotracking">
4      <field type="Date" id="dateStart"  column="date"
5             label="Begin Date | pt_BR: Data Inicial" first-year="2000"
6             last-year="[now]" show-age="true"  required="true"
7             format="YYYY/MM/DD" />
8      <field type="Time" id="timeStart" column="time"
9             label="Begin Time | pt_BR: Hora Inicial" required="true"
10            format="HH:ii:ss" />
11     <field type="Date" id="dateEnd"  column="date"
12            label="End Date | pt_BR: Data Final" first-year="2000"
13            last-year="[now]" show-time="[now]" show-age="true"
14            required="true"  format="YYYY/MM/DD" />
15     <field type="Time" id="timeEnd"  column="time"
16            label="End Time | pt_BR: Hora Final" required="true"
17            format="HH:ii:ss" />
18 </search>
```

Código-Fonte C.3 – Mapeamento da Pesquisa para a funcionalidade de **Rastreamentos - farmsplview.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A ser renderizado no documento - farmsplfile.php
lat = latitude
lng = longitude
-->

<view table="farmspl.vwgeotracking" primary="idcollar">
    <field type="String" column="idcollar" label="animal" />
    <field type="String" column="lat" label="lat" id="lat" />
    <field type="String" column="long" id="lng" label="lng" />
    <field type="Date" column="date" label="date" show-time="true"
           format="YYYY/MM/DD HH:ii:ss" id="_datetime" />
    <field type="Time" column="time" id="time" label="time"
           show-time="true" format="HH:ii:ss" />
    <field type="Integer" column="flag" label="flag"
           value="1" read-only="true" required="true" />
</view>

<!-- Utiliza o search para complementar a pesquisa com as variaveis
           GETADAS na requisicao -->
<search table="farmspl.vwgeotracking">
</search>
```

Código-Fonte C.4 – Mapeamento do resultado da processamento da busca da pesquisa para a funcionalidade de **Rastreamentos - farmsplfile.xml**

APÊNDICE D – Processo Executar Agendamentos

Processo descrito para execução de agendamentos de atualização de tabelas que utilizam a FarmSPL

1. Inicia o processo
2. Receber parâmetros: *tablename* (nome da tabela), agendamento (periodicidade: *hourly, daily, weekly, monthly*)
3. Se parâmetros forem válidos:
   - 3.1. **S**: Realizar *Loop* dos Agendamentos
     - 3.1.1. Realizar Subprocesso: **Realiza cenário de iteração por agendamento**
     - 3.1.2. Se retorno da mensagem for vazio
       - 3.1.2.1. **S**: Finaliza com mensagem com valor: "Não há agendamentos ou dados para atualização"
       - 3.1.2.2. **N**: Finaliza com mensagem com valor de retorno do subprocesso
   - 3.2. **N**: Finaliza com mensagem com valor: "Não está defina a tabela ou procedimento / function ou view para o agendamento"
4. Finaliza o processo com mensagem de retorno

Subprocesso: **Realiza Cenário de iteração por agendamento**

1. Inicia o processo
2. Receber parâmetros do agendamento
3. Se parâmetros forem válidos
   - 3.1. **N**: Preenche a mensagem com valor "Parâmetros indefinidos ou tabela não localizada no banco"
   - 3.2. **S**: Executar e receber dados do recurso
     - 3.2.1. Se houver dados para atualização
       - 3.2.1.1. **N**: Preenche a mensagem com valor ""
       - 3.2.1.2. **S**: Atualizar tabela com dados da requisição

3.2.1.2.1. Atualizar agendamento com última atualização com data e hora corrente e preenche a mensagem com valor “Atualização efetuada”

4. Finaliza o subprocesso e retorna a mensagem

APÊNDICE E – Função SQL para agendamentos da tabela *geotracking*

```
1
2  CREATE OR REPLACE FUNCTION farmspl.update_table_geotracking (
3    id varchar = NULL::character varying,
4    _table varchar = NULL::character varying,
5    resourceuri varchar = NULL::character varying,
6    lastupdate varchar = NULL::character varying
7  )
8  RETURNS text AS
9  $body$
10 DECLARE
11   tbl varchar;
12   check_query text;
13   check_exists integer;
14   retorno text;
15 BEGIN
16   retorno = '';
17     IF id IS NOT NULL AND resourceuri IS NOT NULL OR _table IS
18     NOT NULL THEN
19       --verifica se existe a tabela configurada(salva no scheduler)
20         tbl = replace(_table, 'farmspl.', '');
21         IF lastupdate IS NULL THEN
22             --primeira data
23                 lastupdate = '12-31-1969 23:59:59';
24         END IF;
25         check_query := '    SELECT 1 FROM information_schema.tables '
26                     || '    where table_name   = '
27                     || '''' || tbl || '''';
28         EXECUTE check_query INTO check_exists;
29         -- Se existir tabela definida
30         IF check_exists IS NOT NULL THEN
31             tbl = 'farmspl.' || tbl;
32             EXECUTE(' CREATE TEMP TABLE IF NOT EXISTS
33                 tbl_json(data json); '||
34                 ' COPY tbl_json ' ||
35                 ' FROM PROGRAM ''wget -q -O - "$@" "' ||
36                 resourceuri || '&datetime-start=' || lastupdate
37                 || '"'';' ||
38                 ' INSERT into ' || tbl || '
39                 (idcollar, lat, long,alt,date, time, velocity,
40                 numsatellites, timestamp, flag, _user) ' ||
```

```
41                  '       SELECT values->>''idcollar'' as idcollar,
42                  values->>''lat'' as lat, values->>''long'' as long,
43                  values->>''alt'' as alt, ' ||
44                  ' (values->>''date'')::date as date,
45                  (values->>''time'')::time as time, values->>''
46                  velocity'' as velocity, ' ||
47                  ' (values->>''numsatellites'')::integer
48                  as numsatellites, (values->>''timestamp'')::integer as
49                      timestamp, 1 ' ||
50                  '       FROM  ( ' ||
51                  '         SELECT json_array_elements(data->''data'')
52                          as values ' ||
53                  '        FROM  tbl_json ' ||
54                  '       ) a; ');
55
56              --Atualiza tabela de agendamentos
57              EXECUTE('UPDATE farmspl.scheduler SET lastupdate =
58              CURRENT_TIMESTAMP WHERE  id = ' || id ||';');
59              retorno = retorno || ' - ' || CURRENT_TIMESTAMP::varchar
60              || ' Tabela: ' || tbl || ';';
61          ELSE
62              retorno = retorno || ' - ' || CURRENT_TIMESTAMP::varchar
63              || ' Não Definida Tabela: ' || tbl || ';';
64          END IF;
65      END IF;
66    RETURN retorno;
67    EXCEPTION WHEN unique_violation THEN
68        ROLLBACK;
69
70  END;
71  $body$
72  LANGUAGE 'plpgsql'
73  VOLATILE
74  CALLED ON NULL INPUT
75  SECURITY DEFINER
76  COST 100;
```

Código-Fonte E.1 – Função que deve ser replicada e alterada para cada agendamento - Código para agendamento da funcionalidade de **Rastreamentos**

APÊNDICE F – Planilha de Sensores

Figura 39 – Sensores utilizados nos projetos da Embrapa Gado de Corte

A	B	C	D	E	F	G
Orientador	Co-orientador	Autor	Título	Descrição	Sensores	Dados
Hana Karina Salles Rubinsztejn	Pedro Paulo Pires	**Luiz Fernando Delboni Lomba e Hana Karina Salles Rubinsztejn. Julho de 2014**	Identificação do Comportamento Bovino a partir dos Dados de Aceleração do Animal e Monitoramento da Luminosidade do Ambiente	Neste trabalho são utilizados sensores para capturar dados para identificação do comportamento animal. Os sensores utilizados são: GPS, acelerômetro, giroscópio e magnetômetro. Inicialmente foi utilizado um LDR para capturar dados de luminosidade, mas o trabalho em campo não coletou os dados deste sensor com sucesso.	A- GPS; B- Acelerômetro; C - Giroscópio; e D- Magnetômetro	(A) GPS: 1 - $GPGGA (solução de posição determinada): fornece a posição representada por latitude e longitude, o horário em que a posição foi obtida e sua validade; 2 - $GPGSV (satélites visíveis); 3 - $GPGSA (DOP e satélites ativos): fornece informações sobre a qualidade da posição obtida; 4 - $GPRMC (Recommended minimum specific GPS/Transit data): fornece a posição representada pela latitude e longitude, informações de hora, validade, data, velocidade e declinação magnética.; 5 - $GPVTG (rumo percorrido); (B) Acelerômetro: 1- Medições magnéticas nos 3 eixos (X,Y,Z); 2- Configurável em ±2, ±4, ±8, ou ±12 gauss; 3- Também trabalhando no primeiro valor (±2); (C) Giroscópio: 1- Movimento de rotação nos 3 eixos (X,Y,Z); 2- Configurável nas escalas ±245, ±500, ou ±2000 °/s; 3- Trabalhando em ±245; (D) Magnetômetro: 1 - Dados de aceleração nos 3 eixos (X,Y,Z); 2 - Pode ser configurado nas escalas - ±2, ±4, ±6, ±8, ou ±16 g; 3 - Estamos trabalhando com o primeiro valor (±2);
Irineu Sotoma	Pedro Paulo Pires	**João Felipe Resende Nacer e Irineu Sotoma**	Uma rede de Sensores sem Fio para Monitoramento Bovino	A rede de sensores irá receber as posições de GPS dos bois em um raspberry PI + Xbee.	GPS	1 - $GPGGA (solução de posição determinada): fornece a posição representada por latitude e longitude, o horário em que a posição foi obtida e sua validade; 2 - $GPGSV (satélites visíveis); 3 - $GPGSA (DOP e satélites ativos): fornece informações sobre a qualidade da posição obtida; 4 - $GPRMC (Recommended minimum specific GPS/Transit data): fornece a posição representada pela latitude e longitude, informações de hora, validade, data, velocidade e declinação magnética.; 5 - $GPVTG (rumo percorrido);
Debora Maria Barroso Paiva	Sergio Raposo	**Olavo José Luiz Júnior**	Estação de Baixo Custo para Monitoramento da Presença de Animais em Cochos de Alimentação	Solução para monitoramento e suporte à avaliação do comportamento de visitas ao cocho de suplemento por grupos de bovinos, de forma a auxiliar a avaliação da suplementação de bovinos a pasto e a gestão de rebanho. Utiliza transponder RFID para identificação de tags fdx(full duplex) e hdx(half duplex) para solução de monitoramento.	RFID (tag hdx e fdx)	1 - Data/hora; 2 - Tag (rfid - numerico); 3 - Temperatura (float); e 3 - Umidade (float);
Ricardo Santos	Fabiana Villa Alves			Colar de coleta de dados fisiológicos para fins de aferição de conforto térmico. O sensor é usado para medir a luminosidade e através dele será possível obter informações sobre comportamento animal, verificar se o animal está na sombra, por quanto tempo e em quais horários.	LDR(Light Dependent Resistor) - Resistor Dependente de Luz ou Fotoresistência	1 - Tempo / data (timestamp); e 2 - Luminosidade / float (porcentagem).