Construção de um Web Service REST para aplicativos de gestão na pecuária de corte

Eduardo Klein Pandolfo[1], Sandro da Silva Camargo[1][0000−0001−8871−3950], Henry Gomes de Carvalho[2], and Vinícius do Nascimento Lampert[1,2]

[1] Universidade Federal do Pampa, Bagé RS 96413-172, Brasil
ekpandolfo@gmail.com,sandrocamargo@unipampa.edu.br
http://www.unipampa.edu.br/
[2] Embrapa Pecuária Sul, Empresa Brasileira de Pesquisa Agropecuária, Bagé RS 96401-970, Brasil
{henrycarvalho,viniciuslampert}@embrapa.br
https://www.embrapa.br/pecuaria-sul

Abstract. A tecnologia está em constante evolução e está disseminada em diversas áreas, auxiliando pessoas em suas atividades diárias, principalmente no sentido de tornar o cotidiano mais fácil, o trabalho mais produtivo e as decisões mais efetivas. Para um produtor rural, as tecnologias podem ser exploradas como vantagem competitiva em suas áreas de negócio, tais como produção e comercialização. Dentre estas tecnologias, os smartphones têm ocupado um papel de destaque, devido à sua disseminação entre a população, tendo um grande potencial para auxiliar o pequenos, médios e grandes produtores. Neste contexto, este trabalho tem como objetivo a criação de uma solução para permitir o uso de dispositivos móveis como recurso para a gestão de propriedades rurais. A metodologia envolveu a criação de um *Web Service* para criar uma comunicação padronizada entre um banco de dados remoto e um conjunto de aplicativos de gestão pecuária produzidos pela Empresa Brasileira de Pesquisa Agropecuária. Para o desenvolvimento da solução, utilizou-se o padrão REST (*Representational State Transfer*) como modelo de arquitetura do *Web Service*.

Keywords: Interoperabilidade · Tecnologia da Informação e Comunicação · Gestão Rural · Empresa Brasileira de Pesquisa Agropecuária · Dispositivos Móveis.

1 Introdução

Atualmente a evolução das Tecnologias da Informação e Comunicação (TIC) pode ser evidenciada pela popularização de smartphones. Esses aparelhos oferecem funcionalidades que vão além da possibilidade de fazer ligações, contando com outros recursos como: captura de áudio e imagens, downloads e instalação de aplicativos e acesso à internet. Um estudo realizado pelo IBGE (Instituto Brasileiro de Geografia e Estatística), em 2017, concluiu que o telefone móvel

estava presente em 93,2% dos domicílios e 78,2% dos brasileiros (com mais de 10 anos) possuíam smartphones para uso próprio [7].

No contexto rural, a utilização de TICs permite equilibrar algumas desvantagens econômicas, reduzindo as barreiras de tempo e de distância dos principais mercados e possibilitando uma reestruturação das organizações por interligar pessoas, processos e empresas [8]. Alguns autores [6] indicam que a velocidade de acesso a informações é um fator significativo na competitividade dos negócios e que um atraso na adoção de novas TICs podem trazer desvantagens competitivas. Já Hofer e colaboradores [5] concluíram que a maioria dos pequenos e médios produtores da região oeste do Paraná gerenciava os processos produtivos e atividades informalmente através de anotações escritas em papel. Brum e colaboradores [1, 2] complementam que anotações escritas dificultam a execução de processos analíticos destas informações, devido ao trabalho adicional de inserção dos dados em computador e a potencial inconsistência devido ao método de coleta.

Em pesquisa feita pela Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA) [4], verificou-se que os dois principais pontos de prioridade para o produtor pecuário na gestão e organização da propriedade rural são os custos de produção e a capacitação de recursos humanos e suporte técnico. Neste contexto, são desenvolvidos alguns projetos como o *Mybeef* [3], que abrange principalmente a região Sul do Brasil e visa a disseminação de conteúdo sobre a cadeia produtiva pecuária para pesquisadores e produtores rurais. Atualmente, o Projeto MyBeef está priorizando o desenvolvimento de aplicativos em ambientes móveis tendo em vista que a grande maioria das propriedades rurais tem restrição de acesso à internet tornando limitado o uso de sistemas web. Um outro fator importante é a existência diversos sistemas de informação com finalidades e linguagens de programação diferentes, que ficam isolados por não terem compatibilidade de comunicação entre si. Nesse contexto, uma solução implementada fundamentou-se em um padrão de comunicação através de *Web Services*. O desenvolvimento de um *Web Service* pode estabelecer uma conexão de dados entre sistemas que contêm finalidades e tecnologias diferentes, possibilitando assim que sistemas tenham comunicação direta entre si sem perder as funcionalidades dos sistemas em desenvolvimento no Projeto *MyBeef*. [10].

Neste sentido, o objetivo deste trabalho é desenvolver uma solução *Web Service* para permitir o intercâmbio de dados entre sistemas heterogêneos para aplicações desenvolvidas pela Embrapa Pecuária Sul. Assim, este trabalho descreve a modelagem, o desenvolvimento e a documentação de um *Web Service* para um modelo lógico de banco de dados, no contexto do do projeto da plataforma *MyBeef* intitulado "Desenvolvimento de Sistemas de Apoio à Decisão e de Métodos de Coleta, Análise de Dados e Monitoramento da Pecuária na Região Sul do Brasil" [3].

[3] https://www.embrapa.br/busca-de-projetos/-/projeto/210797/desenvolvimento-de-sistemas-de-apoio-a-decisao-e-de-metodos-de-coleta-analise-de-dados-e-monitoramento-da-pecuaria-na-regiao-sul-do-brasil

Este artigo está organizado da seguinte forma: na seção 2 é exposta a metodologia proposta para o desenvolvimento do trabalho. Na seção 3 é apresentado o processo de desenvolvimento do *Web Service*. E, por último, na seção 4, são apresentadas as conclusões e trabalhos futuros.

2 Material e Métodos

A metodologia para o desenvolvimento foi dividida em etapas, que são apresentadas na Fig. 1.

Fig. 1. Etapas para execução do projeto.

A seguir, são descritas cada uma das etapas executadas neste trabalho [9]:

1. Delimitação do Problema: Antes de começar o desenvolvimento da solução, foi preciso delimitar seu escopo. O trabalho necessitava disponibilizar uma solução tecnológica para acesso a banco de dados armazenado em um servidor web por uma aplicação móvel. Esta solução é responsável por receber solicitações de um aplicativo, executá-las e, por último, devolver uma resposta. Resumindo, a Interface de Programação da Aplicação (API) intermedia o intercâmbio de dados entre um aplicativo em um dispositivo móvel e um banco de dados remoto.
2. Levantamento do Referencial Teórico: Foi realizada uma revisão sistemática de literatura com as seguintes palavras-chaves para as pesquisas de referenciais: interfaces de programação de aplicativos, *REST API*, pecuária, banco de dados e *Web Services*. Como fontes primárias da pesquisa, foram utilizados os anais do Simpósio da Ciência do Agronegócio (CIENAGRO), do Simpósio de Pecuária de Corte (SIMPEC) e do Congresso Brasileiro de Agroinformática

(CBIAgro), além da *Scientific Electronic Library Online* (SciELO) e publicações da própria Embrapa. Outras fontes foram utilizadas para fornecer conhecimento de ferramentas, linguagens, padrões de programação e protocolos que estão envolvidos na parte de desenvolvimento da API.

3. Análise do Banco de Dados: Foi atualizado o modelo de banco de dados utilizado pelo projeto *MyBeef*, a partir de onde o estudo foi feito. Foram identificadas mais de vinte tabelas divididas em dois esquemas de banco de dados Postgres [4]: Um público (denominado *public*) que contém tabelas como: usuários, fazendas, *logs* de ações de usuários, categorias de animais, cidades, estados e países. O segundo esquema é sobre a Agenda Campeira, que é um aplicativo que está desenvolvido pela Embrapa e que utilizará a API produzida nesse trabalho para realizar o acesso remoto ao banco de dados.
4. Definição de Requisitos: A definição dos requisitos ocorreu através de reuniões a equipe de desenvolvimento do *MyBeef* e com os técnicos da Embrapa. Como resultado das entrevistas, criou-se um documento de requisitos da *Web* API, com o objetivo de identificar e registrar exigências que necessitam ser cumpridas [9]. O documento teve o objetivo de nortear o processo de documentação e desenvolvimento durante a execução prática deste trabalho.
5. Modelagem dos Recursos: Nesta etapa realizou-se a modelagem dos recursos do *Web Service*. Foi utilizada a ferramenta Swagger[5], que possibilitou obter uma documentação visual dos recursos, seus métodos e as tabelas manipuladas por cada recurso. Esta ferramenta foi escolhida por ser *open-source* e por estar padronizada pela Embrapa para o desenvolvimento de *Web Services* REST. Com o documento devidamente analisado, partiu-se para o desenvolvimento da API. Como resultado desta etapa, foram modelados recursos para manipulação de doze tabelas e recursos auxiliares para autenticação e sincronização.
6. Desenvolvimento da API: A API foi programada com a linguagem orientada a objetos Java [6]. O desenvolvimento contemplou a utilização de ferramentas e *frameworks* complementares, tais como o Maven [7] para gerenciamento do projeto Java e o Hibernate [8] para mapeamento e persistência dos dados, além do uso de padrões para transferências de objetos Java (DTO). Assim como a escolha do Swagger, a linguagem de programação e as ferramentas complementares também foram escolhidas por já estarem padronizadas pela Embrapa, o que facilitou também o processo de manutenção da API pela própria empresa. Como resultado dessa etapa, os recursos modelados na etapa anterior foram implementados.
7. Testes e Validação: Nesta etapa foram feitos os testes dos recursos desenvolvidos para validação do *Web Service*. Para isso, foi necessário realizar o *upload* do *Web Service* em um servidor web para deixá-lo disponível e

[4] https://www.postgresql.org/
[5] http://swagger.io
[6] http://www.java.com
[7] https://maven.apache.org/
[8] https://hibernate.org/

conseguir utilizar ferramentas que simulam chamadas *Hypertext Transfer Protocol* (HTTP) para o endereço dos recursos. Assim, foi possível realizar os testes dos recursos sem a necessidade da criação de um aplicativo ou de um ambiente para testes mais robusto. No modelo de APIs proposto por REST, as solicitações devem ser independentes e, por isso, os testes foram feitos por recurso. A validação foi realizada através da comparação entre os resultados desejados pelos recursos modelados e obtidos através da solução implementada.

8. Análise dos Resultados e Conclusões: Etapa de finalização do trabalho foi dividida duas partes: uma de avaliações das tecnologias e outra de conclusões apresentando possíveis desdobramentos do trabalho.

3 Resultados e Discussões

O funcionamento básico do *Web Service* pode ser visto na Fig. 2, mostrando em etapas como acontece a comunicação do aplicativo e o banco de dados. No exemplo, é representada uma requisição de leitura pelo aplicativo, onde o *Web Service* é responsável por executar uma clausula *Structured Query Language* (SQL) na tabela de usuários a partir do identificador enviado como parâmetro no endereço, e responder ao aplicativo o resultado dessa leitura.

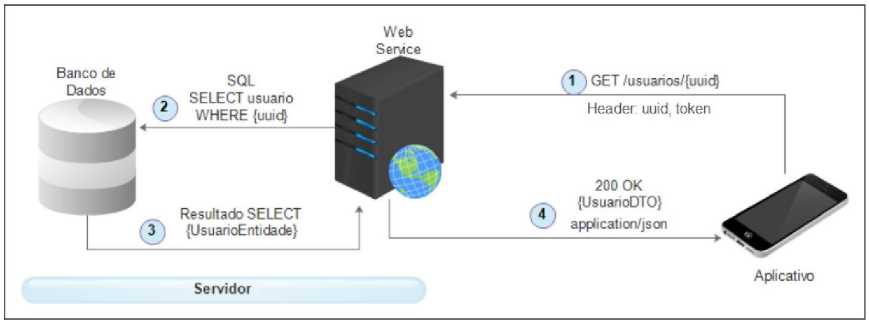

Fig. 2. Funcionamento básico do *Web Service*.

O funcionamento básico pode ser descrito da seguinte forma:

1. O aplicativo solicita, via HTTP, uma requisição ao endereço *"/usuarios/{uuid}"* enviando o identificador deste usuário como parâmetro.
2. Ao receber a solicitação, o *Web Service* é responsável por reconhecer e validar a chamada. Após isso, ela utiliza o identificador enviado para executar uma leitura no banco de dados no servidor. Se a chamada contiver inconsistências, seja no corpo de requisição ou nos parâmetros enviados, a resposta contendo os erros é enviada nesta etapa para o aplicativo, não sendo propagada a requisição para a instância do banco de dados.

3. O banco de dados retorna para o *Web Service* o resultado da consulta, que é armazenado em uma classe denominada *"UsuarioEntity"*.
4. O *Web Service* cria outra classe, denominada *"UsuarioDTO"*, a partir da classe da etapa anterior e a envia ao aplicativo como resposta da solicitação HTTP da etapa 1. O formato dos dados utilizado na resposta é *JavaScript Object Notation* (JSON).

3.1 Implementação

Para obter maior controle sobre os dados recebidos pelo aplicativo o *Web Service* utilizou um padrão chamado DTO (Objeto de Transferência de Dados), cujo objetivo é mapear, organizar e validar os corpos das mensagens recebidas e enviadas para o aplicativo. Sendo assim, a implementação deste padrão facilitou que as informações recebidas pelo *Web Service* fossem validadas antes que fossem encaminhadas ao banco de dados.

Para que as classes pudessem ser reconhecidas pela aplicação, deveriam possuir as notações referentes aos formatos utilizados nas mensagens. Nas classes DTO, as notações se referem ao JSON e as classes de entidades contém referências das tabelas do banco. As classes, que representam as entidades do banco, tem em sua implementação as notações ORM que realizam o mapeamento das variáveis da linguagem Java para os campos do SGBD PostgreSQL.

A implementação do padrão DTO trouxe uma confiança maior no formato dos dados enviados pelo aplicativo devido a camada de validação feita pelo *Web Service*, entretanto, no contexto do projeto, o padrão obrigou que fosse construída uma classe auxiliar para cada DTO, que faz a transformação para uma classe que representa uma entidade e de uma entidade para um DTO.

A implementação do gerenciamento dos dados e toda organização das pastas principais do projeto foram divididas em: Modelos (*Models*), Recursos (*Resources*) e Serviços (*Services*).

1. Modelos: A pasta modelos contém três subdivisões, onde cada uma contém um formato de classe necessária para a troca de dados com o aplicativo e o banco de dados. A primeira, denominada DTO, contém as classes que reconhecem o formato de mídia JSON e, recebem e enviam dados diretamente com o aplicativo. A segunda, denominada *Entity*, contém todas as classes que serão manipuladas com operações CRUD no banco de dados. A última, *Mapper*, contém as classes que contém os métodos que geram uma classe DTO a partir de uma entidade e vice-versa.
2. Serviços: São as classes que implementam uma *Entity Manager* para controle da lógica das operações CRUD. Essas classes tem a função de validar as informações das classes DTO e realizar de forma persistente as manipulações de dados no banco de acordo com cada chamada realizada pelas classes de recursos. Por organização, existe uma interface que expõe os métodos e outra classe que as implementa. Para cada recurso modelado foi criada uma classe *Service*.

3. Recursos: São as classes que gerenciam um serviço, elas utilizam a notação *@Inject* para criar uma instância de uma classe Serviço e os utilizam para criar as respostas de cada método. Nela, estão contidas também as notações referentes aos métodos, endereços e formato de dados. Cada recurso modelado contém uma classe nesse formato.

3.2 Autenticação e Validação

Pelo fato de cada requisição REST ser independente, existe a necessidade de enviar informações que identifiquem o usuário utilizador do aplicativo no cabeçalho da requisição HTTP. Após a análise das alternativas de autenticação [9], foi implementada a autenticação de usuários através de *tokens*. Sempre que um novo usuário for criado no banco de dados ou houver uma chamada no recurso de usuários para autenticação será retornado um *token* de acesso para o aplicativo. Nestas chamadas, o *token* é criado e salvo no banco de dados na tabela de usuários e cada requisição feita pelo aplicativo deve conter o mesmo *token* e o identificador do usuário no cabeçalho HTTP da requisição. Sendo assim, a partir da primeira chamada para criação de um usuário, só será enviada a senha para o *Web Service* novamente quando um usuário efetuar um login no aplicativo. Em cada requisição recebida pelo *Web Service*, exceto na criação e autenticação de usuários, os dados do identificador e *token* do usuário são lidos da requisição e um procedimento de verificação é realizado.

Um ponto importante para validação das requisições é que um usuário só poderá editar os dados de uma fazenda na qual ele está vinculado. Então, para cada requisição de qualquer recurso das tabelas do esquema da *agenda_campeira*, deverá ser avaliado se o usuário tem autorização para realizar qualquer manipulação desses dados. Para isso, em cada classe que não contém diretamente o identificador da fazenda, foi necessária a criação de uma função que encontrasse uma fazenda a partir dos identificadores únicos dessas tabelas, para posteriormente realizar o teste e validar a associação.

3.3 Logs

Como requisito do *Web Service*, ele precisa registrar no banco de dados um histórico de cada requisição válida que efetuou alguma alteração nos dados. Portanto, sempre que houver uma requisição válida de um usuário autenticado, antes de enviar a resposta ao aplicativo os dados dessa requisição devem ser salvos no banco. No desenvolvimento desta funcionalidade, não conseguiu-se através da ferramenta *Hibernate* mapear diretamente uma *string* com o endereço de IP do usuário para o formato *inet* presente no PostgreSQL. Para resolver tal problema, criou-se uma clausula SQL exclusiva para esta tabela, implementada na *EntityManager* para que todas classes que utilizem herança tenham acesso. O procedimento de inserir um *log* no banco de dados utilizou a função *CAST* de SQL para realizar a tipagem correta.

3.4 Distribuição do aplicativo

O *Wildfly Swarm*[9] oferece uma abordagem para empacotar e executar aplicativos Java EE (*Java Enterprise Edition*). Este é o *framework* utilizado para controlar toda a execução do *Web Service*. Com o *Wildfly Swarm* é possível declarar as especificações de Java EE que serão necessárias para a execução. Assim, ele fornece suporte a desenvolvimento de microsserviços em conjunto com um servidor de aplicação para execução de testes.

Como o *Swarm* será o responsável pela execução da aplicação, foi necessário incluir algumas configurações para que a ferramenta tivesse acesso ao banco de dados da Embrapa. Propriedades como nome de usuário, senha, endereço do banco e as configurações estão contidas no arquivo .POM (*Project Object Model*). Por padrão, para execução o *Swarm* concede acesso à aplicação na porta 8080 do servidor web onde se encontra o banco de dados.

3.5 Testes

Os testes foram realizados com o objetivo de verificar se a modelagem está de acordo com os recursos disponibilizados pelo *Web Service*. Para isso, criou-se coleções de chamadas organizadas pelos recursos modelados, onde em cada coleção foram geradas requisições para os testes dos recursos implementados. Todas as chamadas para o *Web Service* necessitam de dados para autenticação de usuário, com exceção das requisições para criação e autenticação de um usuário. Em cada chamada constam os campos com o identificador do usuário e o seu *token*, necessários para identificar o usuário e o validar antes de executar cada manipulação no banco de dados. Para execução dos testes foi utilizada a ferramenta Postman[10], que é uma aplicação gratuita para realizar requisições HTTP através de uma interface compartilhada, que facilita os testes de serviços de APIs em equipes.

4 Conclusões

Neste trabalho, um *Web Service* foi criado para realizar a comunicação de um aplicativo móvel produzido pela Embrapa Pecuária Sul com o banco de dados web da empresa, possibilitando assim, o armazenamento e leitura de dados lançados no aplicativo por produtores rurais pecuários. Dentre as funcionalidades viabilizadas pelo desenvolvimento da solução proposta neste trabalho, podem ser citadas:

1. O uso de um banco de dados centralizado, integrado com uma aplicação *mobile*, permitiu à Embrapa o armazenamento de dados de produtores rurais, tais como o perfil do produtor e características gerais do seu sistema produtivo.

[9] http://developers.redhat.com/jboss/wildfly
[10] https://www.getpostman.com/

2. Através de autenticação, o aplicativo pode restringir acesso e recuperar dados de fazendas de um usuário através dos recursos de sincronização entre o banco da Embrapa e a aplicação *mobile*. Também, através da validação feita pelo *Web Service*, usuários associados a uma mesma fazenda poderão realizar os lançamentos de registros no aplicativo de forma colaborativa.
3. Mudanças no banco de dados da Embrapa, dependendo da complexidade, puderam ser implementadas somente no *Web Service*, reduzindo a necessidade de atualizações do aplicativo.

A utilização de REST foi adequada para o problema apresentado pela empresa. A utilização do protocolo HTTP para indicar quais manipulações devem ser realizadas no banco de dados foi mapeada e modelada. Através das ferramentas apresentadas no trabalho, foi possível desenvolver uma solução que cumpre sua função de conectar duas aplicações. REST e JSON foram boas escolhas para realizar uma comunicação do aplicativo com o *Web Service* por serem eficientes em tempo de leitura. Os testes apresentados nesse trabalho foram realizados utilizando ferramentas que simulam chamadas HTTP. Através da aplicação Postman, foi possível armazenar respostas do *Web Service* para que servissem de exemplos para possíveis erros da utilização do *Web Service* pelos desenvolvedores do aplicativo.

A principal dificuldade encontrada durante a execução deste trabalho foi a de manter todos os documentos (modelagem, projeto e testes) atualizados durante o desenvolvimento do *Web Service*. O banco de dados teve várias mudanças no decorrer do trabalho e para cada mudança, foi necessário atualizar a modelagem, realizar as mudanças no código e realizar os testes novamente para adequação dos documentos.

Apesar de adequada a solução ao problema apresentado, alguns pontos do trabalho podem ser melhorados e outras tecnologias ou modelos de programação podem ser acrescentados. Sobre continuação e aprimoramento da API desenvolvida nesse trabalho, podem ser citadas as seguintes possibilidades:

1. Estudo da implementação de DAOs (*Data Access Objects*) para substituir o padrão DTO em algumas classes. Um DAO é um objeto que pode ser mapeado diretamente via *frameworks* para a comunicação entre o banco de dados e outra aplicação. Por mais que seja requisito da API a utilização de DTOs, em algumas classes sua utilização é obsoleta, porque todos os dados da entidade são enviados para a resposta sem nenhuma restrição. Logo, nesses casos, não há necessidade de ter uma classe DTO e executar os procedimentos necessários para utilizá-la.
2. Implementação de recursos adicionais para criação de usuários a partir de dados de autenticação da Google e do Facebook;
3. Aprimoramento do sistema de autenticação, utilizando alguma forma de serviço secundário para desvalidar *tokens* através de um limite de tempo ou por número de requisições.
4. Após a finalização e distribuição do aplicativo que utilizará o *Web Service*, pode-se realizar análises das informações armazenadas no banco de dados da Embrapa.

5. Definir um protocolo de manutenção do *Web Service*.

References

1. Brum, L.M.L.: Aplicação de Técnicas de Business Intelligence em Sistemas de Apoio à Tomada de Decisão de Produtores Rurais. Master's thesis, UNIPAMPA - Universidade Federal do Pampa, Bagé (2019), http://dspace.unipampa.edu.br/handle/riu/3946
2. Brum, L.M.L., Lampert, V.N., Camargo, S.S.: Business intelligence and data warehouse in agrarian sector: A bibliometric study. journal of agricultural science **11**(2), 353–368 (2019). https://doi.org/10.5539/jas.v11n2p353, http://www.ccsenet.org/journal/index.php/jas/article/view/0/38109
3. EMBRAPA: Desenvolvimento de sistemas de apoio à decisão e de métodos de coleta, análise de dados e monitoramento da pecuária na região Sul do Brasil. (2015), https://www.embrapa.br/busca-de-projetos/-/projeto/210797/desenvolvimento-de-sistemas-de-apoio-a-decisao-e-de-metodos-de-coleta-analise
4. EMBRAPA: Gestão de custos é a principal preocupação do pecuarista brasileiro. Brasília (2018), https://www.embrapa.br/busca-de-noticias/- /noticia/36645433/gestao-de-custos-e-a-principal-preocupacao-do-pecuaristabrasileiro
5. Hofer, E., Pacheco, V., Souza, A., Protil, R.: A relevância do controle contábil para o desenvolvimento do agronegócio em pequenas e médias propriedades rurais. Revista Contabilidade e Controladoria **3**(1), 27–42 (2011). https://doi.org/10.5380/rcc.v3i1.21490, https://revistas.ufpr.br/rcc/article/view/21490
6. Hollifield, C., Donnermeyer, J.: Creating demand: influencing information technology diffusion in rural communities. Government Information Quarterly **20**(2), 135–150 (2003). https://doi.org/10.1016/S0740-624X(03)00035-2, https://www.sciencedirect.com/science/article/pii/S0740624X03000352
7. IBGE: Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal : 2016. IBGE - Instituto Brasileiro de Geografia e Estatística (2018), https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhesid=2101543
8. Machado, C.F., Nantes, F.D.: Adoção da tecnologia da informação em organizações rurais: o caso da pecuária de corte. Gest. Prod. **18**(3), 555– 570 (2011), http://www.scielo.br/pdf/gp/v18n3/09.pdf
9. Pandolfo, E.K.: Desenvolvimento de um Web Service REST para um protótipo de aplicativo no contexto pecuário. (2019), monografia (Bacharel em Engenharia de Computação), UNIPAMPA (Universidade Federal do Pampa), Bagé, Brazil
10. Silva, T.W.B., Morais, D.C., Andrade, H.G.R., Lima, A.M.N., Melcher, E.U.K., Brito, A.V.: Environment for integration of distributed heterogeneous computing systems. Journal of Internet Services and Applications **9**(4), 135–150 (2018). https://doi.org/10.1186/s13174-017-0072-1, https://link.springer.com/article/10.1186/s13174-017-0072-1