Disponibilização de Métodos de Tratamento de Dados Ambientais por meio de um Web Service

Thiago M. VENTURA, Anderson O. SANTOS, Allan G. de OLIVEIRA, Johnnes S. V. da CRUZ, Josiel M. de FIGUEIREDO, Claudia A. MARTINS
Instituto de Computação, Universidade Federal de Mato Grosso
Cuiabá, Mato Grosso, Brasil

RESUMO

Dados meteorológicos são importantes para os estudos dos fenômenos climáticos. A situação em que a coleta desse tipo de dado são realizadas podem ocasionar diversos erros na gravação dos mesmos. Existem métodos para tratar esses erros, mas a implementação em uma linguagem de programação específica pode gerar obstáculos na aplicação desses métodos em um sistema. Este trabalho teve como objetivo criar uma solução para disponibilizar métodos de tratamento de dados meteorológicos, não importando com a tecnologia utilizada. Para atingir este objetivo, um *web service* foi criado aproveitando do *framework* FICSED, que possui funcionalidades de tratamento de dados meteorológicos. Desta forma, agora é possível desenvolver sistemas com funcionalidades de detecção de *outliers* e preenchimento de lacunas, facilitando o desenvolvimento de sistemas especialistas em tratamento de dados ambientais.

Palavras-chaves: detecção de outlier, preenchimento de falhas, inteligência artificial, serviço, correção, FICSED.

1. INTRODUÇÃO

A tecnologia atual permite que cientistas desenvolvam técnicas que sejam capazes de analisar dados para a compreensão de eventos, como acontece no estudo da meteorologia. Nessa área, os dados são séries temporais, pois possuem uma dimensão temporal e representam-se por meio de uma sequência ordenada de valores na qual cada um corresponde ao valor observado da variável em um determinado instante de tempo [1].

Sensores são utilizados para obter estes dados, que por diversos fatores podem gerar falhas na obtenção. Problemas com sensores normalmente estão ligados à interferência e mal funcionamento. Esses eventos causam *outliers* ou lacunas nas séries, gerando uma análise imprecisa por parte dos pesquisadores. Um *outlier* é um dado que está incoerente em relação aos demais, já a lacuna, ou dado ausente, é o não armazenamento do dado. Por outro lado, existem técnicas que realizam a abordagem desses problemas tentando solucioná-los.

Há técnicas de detecção de *outliers*, como o Z-score, Boxplot, Redes Neurais e limites válidos, que verificam dados incoerentes. Da mesma forma, há técnicas de preenchimento de lacunas, como Regressão Linear, *Support Vector Machine* e Média, que estimam valores para inserir dados onde houve um erro de gravação. Essas técnicas devem ser implementadas em alguma linguagem de programação. Portanto, comumente, a técnica fica submetida a determinada linguagem, dificultando a integração com outros sistemas.

Para resolver esta situação, web *web services* podem ser utilizados. O objetivo deste trabalho é disponibilizar métodos de tratamento de dados meteorológicos por meio de um *web service*, no qual as técnicas que trabalham com detecção de *outliers* e preenchimento de lacunas poderão ser utilizadas em qualquer ferramenta, independente da tecnologia empregada no seu desenvolvimento.

Este trabalho está organizado da seguinte forma: na seção 2 são apresentados trabalhos relacionados ao tema abordado; na seção 3 é detalhado o *framework* FICSED, responsável por realizar o tratamento dos dados; na seção 4 são feitos breves comentários a respeito de *web services*; na seção 5 é detalhado o funcionamento da solução criada; a forma de acesso à solução é mostrada na seção 6 e, por fim, na seção 7 são realizadas as considerações finais.

2. TRABALHOS CORRELATOS

Os trabalhos envolvendo dados meteorológicos e a disponibilização de *web services* geralmente se limitam a fornecer os dados para acesso em *grid*, serviços de previsão climática ou aplicações específicas para um tipo de dado.

Em [2], por exemplo, é apresentada a ferramenta openmeteo.org que, baseado em *web services*, possibilita que usuários de dados meteorológicos e de hidrologia façam o *upload* e compartilhamento dos seus dados. Em trabalho semelhante, [3] apresentam um *RESTfull web service* para gerenciamento de dados diários de meteorologia, nesse caso, o objetivo do trabalho apresentado é que os usuários possam acessar os dados direto de suas planilhas ou aplicações. Outro projeto na mesma linha é o *SoDa project* [4], que disponibiliza séries temporais de dados de radiação solar e em [5] é apresentado um serviço de controle de qualidade para o mesmo tipo de dado. Em [6] um modelo de *web service* para acesso aos dados de meteorologia também é apresentado.

Para modelagem de dados, em [7] é apresentado o DotOpModel, um projeto que disponibiliza, por meio de *web services*, serviços de modelagem de dados para hidrologia. Em [8] é apresentado um sistema, com *web services*, que disponibiliza serviços de avisos e previsões climáticas.

Assim sendo, é possível perceber que a maioria das soluções existentes tratam apenas de compartilhamento e modelagem de dados, não em relação ao pré-processamento dos mesmo. Este trabalho faz uso de duas tecnologias para compor a solução criada. A primeira é um *framework* que possui as técnicas de

tratamento de dados meteorológicos. A segunda tecnologia são os próprios *web services*, permitindo a integração entre sistemas através da Internet.

3. FRAMEWORK PARA TRATAMENTO DE DADOS

O *Framework to Improve and Correct Several types of Environmental Data* (FICSED) foi desenvolvido utilizando a linguagem de programação Java e é voltada principalmente para tratamento de dados meteorológicos, fazendo uso de técnicas de estatísticas e de inteligência artificial. Seu uso é voltado para projetos que utilizam a tecnologia Java, sejam eles *desktop* ou para web [9].

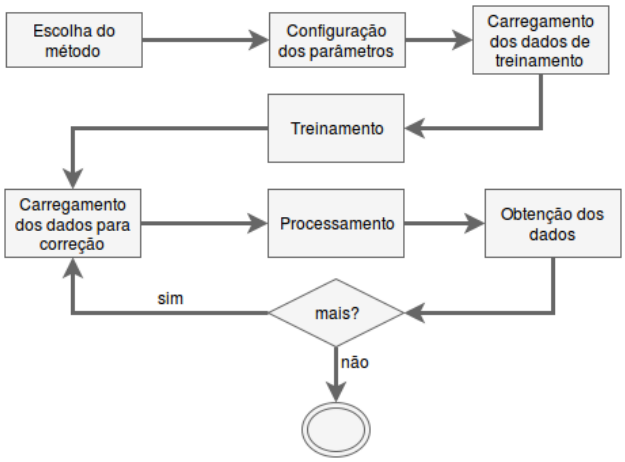

Figura 1: Ciclo de vida do framework FICSED

Atualmente, o FICSED possui cinco métodos de preenchimento de lacunas: Regressão linear, Regressão linear múltipla, SVM, Mannga e Média. Ainda contempla cinco métodos para detecção de *outliers*: Z-score, Boxplot, Mannga para *outliers*, ODHiMM e Limites.

A Figura 1 representa o ciclo de vida do *framework* FICSED, mostrando o seu processo de utilização no qual primeiramente deve-se escolher um dos métodos citados anteriormente. Após este passo, é necessário a realização da configuração dos parâmetros do método escolhido. Depois da etapa de configuração, deve ser carregado os dados de treinamento e realizar o treinamento. E, finalmente, é possível realizar o carregamento dos dados para a correção, realizar o processamento e obter os dados corrigidos. Para múltiplas correções só é necessário realizar as etapas a partir do carregamento dos dados de correção, já que o método já foi treinado.

Preenchimento de lacunas
Os métodos de preenchimento de lacunas consistem em estimar valores com base nos dados históricos, modelando o comportamento do conjunto.

Regressão Linear Simples: O método de Regressão Linear Simples (RLS) faz correlação de uma variável para a desenvolver um modelo que é capaz de fazer previsões e assim estimar valores. A Eq. (1) apresenta a RLS.

$$Y_i = \alpha + \beta X_i \quad (1)$$

Onde Yi é a variável dependente, α é o coeficiente linear, β é o coeficiente angular e Xi a variável independente.

Regressão Linear Múltipla: O método de Regressão Linear Múltipla (RLM) faz correlações entre variáveis para a desenvolver um modelo que é capaz de fazer previsões e assim estimar valores. A Eq. (2) apresenta RLM.

$$P_x = a_0 + \sum_{i=1}^{n} a_i P_i \quad (2)$$

Onde P_x é a variável dependente (a falha a ser preenchida), a_0 e a_i são coeficientes do modelo linear, *n* é o número de variáveis independentes e P_i são as variáveis independentes (atributos).

Support Vector Machine (SVM): O método SVM consistem em reconhecer padrões em um conjunto de dados e categoriza-los entre duas categorias por meio de uma linha de separação (hiperplano), onde busca-se maximizar a distâncias entre os pontos e a linha.

O SVM foi utilizado por meio da biblioteca LIBSVM, e detalhes do seu funcionamento pode ser obtido em [10].

Mannga: O método Mannga usa de conceitos de Rede Neural Artificial (RNA) e Algoritmos Genéticos (AG) para realizar o preenchimento de lacunas por meio de cálculos levando em consideração valores mensurados por outros sensores no momento que ocorreu a falha. O AG tem a responsabilidade de definir qual a melhor RNA para realizar as estimativas. Já a RNA, definida previamente pelo AG, tem a responsabilidade de calcular as estimativas para preencher as lacunas. O complexo funcionamento do método Mannga é descrito em [11].

Média: O método Média é o mais simples. Ele consiste em calcular a média do conjunto. Essa média pode levar em consideração dados anteriores, posteriores ou ambos. A Eq. (3) apresenta o cálculo da média.

$$x_i = \frac{\sum_{k=i-\frac{n}{2}}^{i-1} x_k + \sum_{k=i+1}^{i+\frac{n}{2}} x_k}{n} \quad (3)$$

Detecção de Outliers
Os métodos de detecção de *outliers* consistem em buscar por dados que estão fora dos padrões do conjunto de dados, sendo provavelmente erros de gravação ou fenômenos raros.

Z-score: No Z-score a detecção de *outliers* é feita comparando e dando uma pontuação ao valor da própria variável em relação ao conjunto. A Eq. (4) trata de calcular a pontuação no método Z-score.

$$z_i = \frac{x_i - \bar{x}}{s} \quad (4)$$

Para calcular a pontuação é necessário calcular a média e o desvio padrão da variável analisada. Assim, o resultado em z_i determina se o dado x_i é um outlier.

Boxplot: O método Boxplot consiste em estimar valores que determinam um limite inferior e superior de um conjunto de dados. Assim, a detecção de *outliers* fica em verificar se o dado está entre esse intervalo.

Mannga para *outliers*: O Mannga para detecção de *outliers* consiste no mesmo principio para o de preenchimento de lacunas. Porém, na detecção de *outliers*, é utilizado a Eq. (5) para calcular a distância entre os dados. Quanto maior for a distância, maior a probabilidade desse dados ser um *outlier*.

$$d = \frac{|r - e|}{r} \tag{5}$$

Onde *d* é a distância, *r* é o valor do dado que se deseja analisar e *e* é o valor estimado pelo Mannga. O conceito do método é descrito em [12].

ODHiMM: O método *Outlier Detection with Hidden Markov Model* (ODHiMM) utiliza a técnica *Hidden Markov Model* para a detecção de *outliers.* São criados 3 modelos: um representando os dados normais; outro com *outliers* superiores e o terceiro com *outliers* inferiores. Depois, cada dado é analisado para verificar em qual modelo ele se assemelha mais. Caso não seja o modelo de dados normais, ele é considerado um *outlier*. Mais detalhes pode ser visto em [13].

Limite: Um dos métodos mais simples para detecção de *outliers*. Trata-se de estabelecer um valor máximo e mínimo válido para o dado que está sendo tratado. Caso haja algum valor maior que o máximo ou menor que o mínimo, será considerado um *outlier*.

É possível observar que o FICSED possui diversos métodos para tratamento de dados ambientais. A desvantagem é que apenas sistemas desenvolvidos em Java podem usufruir dessas funcionalidades, uma vez que o *framework* está desenvolvido nesta linguagem. Para resolver esse problema, *web services* podem ser utilizados.

4. WEB SERVICES

A forma na qual as pessoas interagem com os meios computacionais vem evoluindo consideravelmente. E, para conseguir manter essa experiência de usuário avançado, os sistemas devem ser capazes de se comunicar, além dos computadores tradicionais, com *Smartphones*, *Smart* TVs, *Tablets*, entre outros.

Adota-se o uso de *web services* para facilitar essa comunicação. Seria dispendioso ter que reescrever as regras de negócio de uma aplicação *desktop* para um aplicativo *mobile,* assim como a manutenção seria muito mais trabalhosa no caso de surgir a necessidade de alteração de alguma regra. Na Figura 2 está destacado o modelo computacional utilizando *web service*.

No modelo por serviços, a lógica de negócio é concentrada totalmente no *web service*. Com isso, caso seja necessário alterar alguma rotina, basta ir no código do serviço e atualizá-lo. Os clientes desse serviço não terão acesso direto ao banco de dados, e no caso de requerimento de algum recurso que já está em *cache*, o serviço meramente retornará esse dado, sem haver a necessidade de abrir uma consulta ao banco, o que diminui a latência da resposta.

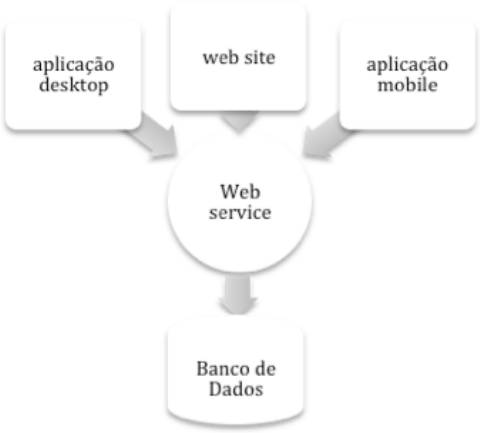

Figura 2: Modelo geral de um web service

Segundo a W3C, consórcio de empresas de tecnologia tem como o objetivo criar padrões comuns para conteúdo da Web, *web service* é um sistema de *software* responsável por proporcionar a interação entre duas máquinas através de uma rede [14].

Um *web service* envia e recebe informações através de protocolos de comunicação em rede (HTTP, IMAP, SMTP, entre outros). Existem diversas especificações de *web services*, sendo um dos mais conhecidos o REST, um estilo híbrido derivado de vários estilos arquitetônicos baseados em rede [15].

Surgido no início dos anos 2000, o REST (*Representational State Transfer*) é fruto da tese de Ph.D do cientista da computação Roy Fielding. Seu objetivo era a formalização de um conjunto de práticas que deveriam ser observados ao trabalhar com os padrões HTTP e URI para aproveitar melhor os recursos oferecidos por tais padrões. Diferentemente de outras arquiteturas, como por exemplo o SOAP, o REST trabalha exclusivamente com o protocolo HTTP.

Assim, criando *web services* que disponibilizam as funcionalidades do FICSED, é possível criar facilmente sistemas capazes de detectar *outliers* e preencher lacunas, mesmo que os desenvolvedores não dominem a linguagem de programação Java, técnicas estatísticas ou de inteligência artificial.

5. TRATAMENTO DE DADOS POR WEBSERVICE

Seria muito útil permitir que todo o potencial do *framework* FICSED pudesse ser utilizado em qualquer projeto, independente da linguagem de programação ou arquitetura empregada. Com o RESTful, implementação das especificações propostas pelo REST, isso pôde ser alcançado. Assim, o desenvolvedor que precisar utilizar os métodos de preenchimento de lacunas e detecção de *outliers* em seu algoritmo deve simplesmente conhecer alguns dos métodos de requisição HTTP, como utilizá-los na linguagem de programação escolhida e os recursos do *web service*.

No protocolo HTTP existem métodos padronizados de requisições e respostas, a RFC 7231 [16], documento que designa padrões de semântica e contexto para requisições e

respostas, estabelece oito métodos que podem ser utilizados para a criação de um *web service*. O RESTful faz uso dos 4 principais métodos, sendo eles:

- GET (obter dados);
- POST (inserir novo dado);
- PUT (atualizar dado já existente);
- DELETE (remover dados cadastrado).

Embora os padrões permitam que o método PUT seja utilizado para a solução proposta, optou-se unicamente pelo método POST, dado que suas especificações se aproximam mais do contexto desejado. Assim, o cliente do serviço enviará e receberá dados, não havendo a necessidade de atualizar ou remover algum dado já disposto o que exclui os verbos PUT e DELETE.

Quando necessário obter alguma versão, o cliente deve informar algumas informações triviais que, por questões de segurança, utilizar os recursos do verbo GET não seria indicado.

A RFC 3986 [17] define URI (*Uniform Resource Identifier*) como uma maneira simples e extensível para identificar um recurso. A princípio, o serviço proposto fornece quatro recursos para o cliente. A Figura 3 apresenta os recursos.

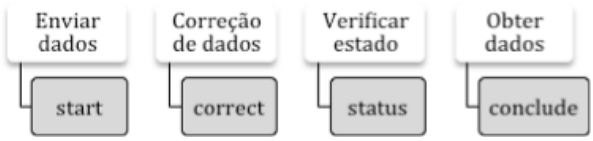

Figura 3: Recursos e ações do serviço

Enviar dados (start)
Inicialmente, o cliente efetua uma requisição POST contendo um arquivo no formato CSV (*Comma Separated Values*) que contém os dados a serem trabalhados. Recebendo esse arquivo, o servidor armazena-o e gera as informações que são importantes para os demais passos. Essas informações estão em formato JSON (*JavaScript Object Notation*), contendo os identificadores de sessão e de usuário, além da versão do arquivo armazenado e data de remoção do mesmo. No caso de falha ele retorna as possíveis causas. A Figura 4 apresenta o diagrama de sequência do processo de envio dos dados.

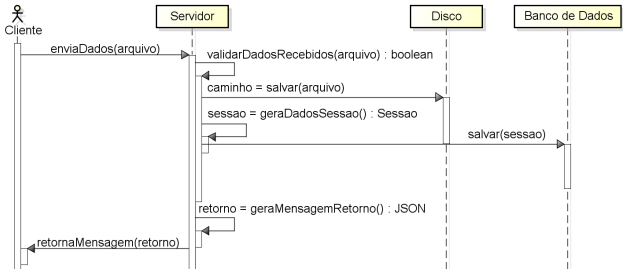

Figura 4: Sequência da requisição de envio

Correção de dados (correct)
Para realizar o processo de correção de dados, o cliente deve efetuar outra requisição *POST*, enviando um JSON contendo as informações de sessão, de usuário, a versão do arquivo e o método que deseja utilizar. No lado do servidor haverá o recebimento e validação desses dados. Caso as informações sejam coerentes é realizado uma cópia da versão do arquivo e iniciado o processo de correção. Após o início o cliente deve receber um JSON com as informações atualizadas dessa sessão. Detalhes deste processo podem ser vistos na Figura 5.

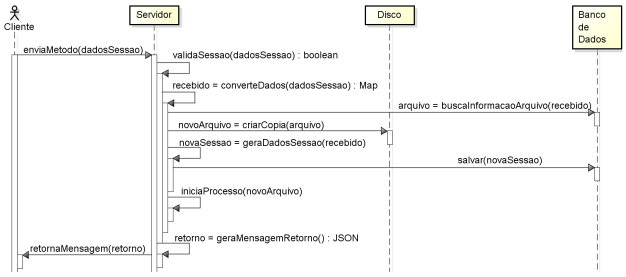

Figura 5: Sequência da requisição de correção

Durante a execução, o cliente pode requisitar diversas vezes as operações de preenchimento de lacunas e/ou detecção de *outliers*. Cada requisição gera um novo arquivo com os dados estimados, mantendo a cópia original.

O cliente deve levar em consideração que cada requisição nesse recurso cria uma nova cópia do arquivo, ou seja, se o cliente utilizou o método Mannga para preenchimento de lacunas e gostaria de logo após executar o método z-score para detectar *outliers*, ele deve aguardar o primeiro processo terminar para só então invocar o segundo. Se ele invocar o procedimento enquanto o outro ainda está em execução, o segundo processo será realizado, contudo, os dados utilizados serão da cópia do arquivo gerado no começo do processo anterior, não mantendo nenhuma relação entre os dados já processados com o primeiro método chamado.

Verificar estado (status)
O processo de correção pode levar desde alguns segundos até várias horas, dependendo do método e parâmetros selecionados. Portanto, seria dispendioso manter o cliente aguardando a resposta por todo esse período. Assim, é necessário um terceiro recurso, para permitir que o cliente averigue o estado do processo. Sua função é validar os dados recebidos, buscar o estado da operação e retornar os dados atualizados, conforme é mostrado na Figura 6.

Figura 6: Sequência da requisição de verificação

Obter dados (conclude)
A partir da primeira requisição, a qualquer momento pode ser obtido uma cópia de alguma das versões previamente geradas. Para obtê-la, o cliente deve produzir uma requisição POST com os dados da operação que gerou a cópia do arquivo, a versão do arquivo desejada e a identificação de usuário. No lado do servidor há a validação dessas informações, a busca dos dados e a construção do retorno. Essa requisição retorna um texto contendo os dados no formato CSV, como segue na Figura 7.

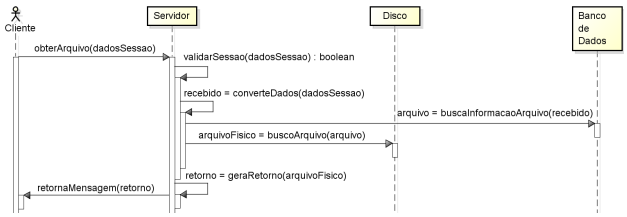

Figura 7: Sequência da requisição de envio

Obter uma versão dos dados não equivale a sua remoção. Os arquivos permanecerão salvos e poderão ser trabalhados enquanto não atingir a data de remoção informada na primeira requisição. Após esse prazo, o serviço removerá todas as informações de sessão desse cliente, para fins de economia de recursos e segurança.

6. DISPONIBILIDADE DO SERVIÇO

O *web service* criado como solução deste trabalho está disponível publicamente. Para utilizar o primeiro recurso, a respeito do envio de dados, a URL a seguir deve ser utilizada:

http://ceda.ic.ufmt.br/service/session/start

Neste recurso deve ser enviado os dados em formato CSV por meio da chave *file*, estando contido no corpo da requisição. A resposta desta requisição será em formato JSON contendo os dados da sessão.

O recurso de correção deverá conter um JSON informando os dados de *userToken*, *sessionToken* e *nameVersion*. Todas estas informações serão obtidas da requisição *start*. Além disso, deve ser enviado também o nome do método a ser utilizado por meio da propriedade *method*. A URL para iniciar a correção está a seguir:

http://ceda.ic.ufmt.br/service/session/correct

Como mencionado, o cliente poderá consultar o andamento do processo de correção por meio da requisição de verificação de estado. Para tanto, a URL abaixo deve ser utilizada:

http://ceda.ic.ufmt.br/service/session/status

Por fim, a requisição de obter os dados tratados pode ser chamada pela URL a seguir. Assim como nas outras requisições, os dados de sessão devem ser utilizados para habilitar o processo. O retorno desta requisição estará no formato *text/csv*.

http://ceda.ic.ufmt.br/service/session/conclude

7. CONSIDERAÇÕES FINAIS

O entendimento dos fenômenos climáticos auxiliam na tomada de decisão das organizações, na criação de medidas preventivas e em previsões variadas sobre o meio ambiente. Entretanto, nas medições de dados ambientais, dados ausentes ou inválidos são problemas comuns devido, por exemplo, à avaria ou desligamento de equipamentos, manutenção, calibração, limitações físicas ou fenômenos climáticos [18]. Por isso, há a importância de haver o tratamento dos dados, tornando as abordagens de detecção de *outliers* e preenchimento de lacunas imprescindível para as organizações.

Neste trabalho foi apresentado uma solução de *web service* em conjunto com o *framework* FICSED para possibilitar que os desenvolvedores executem procedimentos de correção de dados ambientais, independente da linguagem de programação utilizada.

Como mostrado, esta área de pesquisa carece de ferramentas flexíveis para a tarefa de tratamento de dados. Portanto, a solução proposta pode auxiliar diversas pesquisas proporcionando uma maneira fácil e eficiente de execução de procedimentos complexos como a detecção de *outliers* e preenchimento de lacunas.

8. AGRADECIMENTOS

Os autores agradecem o apoio financeiro da Fundação de Amparo a Pesquisa do Estado de Mato Grosso (FAPEMAT) processo 223633/2015 e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

9. REFERÊNCIAS

[1] M. W. Jeffrey. Introductory Econometrics: A modern approach. Canada: South-Western Cengage Learning, 2009.

[2] S. Kozanis; A. Christofides; N. Mamassis; D. Koutsoyiannis. openmeteo.org: A Web Service for the Dissemination of Free Meteorological Data. In Advances in Meteorology, Climatology and Atmospheric Physics, organizado por Costas G. Helmis e Panagiotis T. Nastos, 203–8. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[3] T. Kim; J. Lee; W. Nam; K. Suh. Development of RESTful Web Service for Loading Data focusing on Daily Meteorological Data. Journal of The Korean Society of Agricultural Engineers. Volume 56, Issue 6, 2014, p. 93-102.

[4] B. Gschwind; L. Ménard; M. Albuisson; L. Wald. Converting a successful research project into a sustainable service: The case of the SoDa Web service, Environmental Modelling & Software, Volume 21, Issue 11, November 2006, p. 1555-1561.

[5] M. Geiger; L. Diabaté; L. Ménard; L. Wald. A web service for controlling the quality of measurements of global solar irradiation, Solar Energy, Volume 73, Issue 6, December 2002, p. 475-480.

[6] A. Woolf; K. Haines; C. Liu. A Web Service Model for Climate Data Access on the Grid. International Journal of High Performance Computing Applications 17, no 3, 2003, p.281-295.

[7] Y. Tan; X. Li; Z. Yang; M. Huang. The design and preliminary realized of DotTopModel based on WebService under B/S Architecture. International Conference on Information Science and Technology, Nanjing, 2011, p. 352-356.

[8] J. Zhou; X. Y. Sun; J. L. Du. Research and Implementation of WebGIS Based on Silverlight. Advanced Engineering Forum, Vols. 6-7, 2012, p. 952-956.

[9] Grupo CED. Ficsed. http://ceda.ic.ufmt.br/#/ficsed. 2016.

[10] C-C. Chang; C-J. Lin. LIBSVM: A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf. 2013.

[11] T. M. Ventura; A. G. Oliveira; H. O. Marques; R. S. Oliveira; C. A. Martins; J. M. Figueiredo; A. G. Bonfante. Uma abordagem computacional para preenchimento de falhas em dados micrometeorológicos. Revista Brasileira de Ciências Ambientais, v1, 2013, p. 61-70.

[12] T. M. Ventura; H. O. Marques; A. G. Oliveira; C. A. Martins; M. C. J. A. Nogueira; W. R. S. Teixeira; J. M. Figueiredo; A. G. Bonfante. Detecção de Outliers em Dados Micrometeorológicos. IX Congresso Brasileiro de Agroinformática, Cuiabá, 2013.

[13] T. M. Ventura. Criação de um Ambiente Computacional para Detecção de Outliers e Preenchimento de Falhas em Dados Meteorológicos. Tese, Universidade Federal de Mato Grosso. 2015.

[14] D. Booth; H. Haas; F. McCabe; E. Newcomer; M. Champion; C. Ferris; D. Orchard. Web services architecture. latest edition, W3C. http://www.w3.org/TR/2004/REC-xml-20040204. 2004.

[15] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures. Tese, University of California. 2000.

[16] R. T. Fielding; J. Reschke. Hypertext transfer protocol (http/1.1): Semantics and content. Doi 10.17487/rfc7231, RFC. http://www.rfc-editor.org/info/rfc7231. 2014.

[17] R. T. Fielding; L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. https://www.ietf.org/rfc/rfc3986.txt. 2005.

[18] D. Hui. Gap-filling missing data in eddy covariance measurements using multiple imputation (mi) for annual estimations. Agricultural and Forest Meteorology, v. 121, n. 1–2, 2004, p. 93–111.