Revista da JOPIC
Jornada de Pesquisa e Iniciação Científica
UNIFESO Teresópolis - RJ
Revista da Jopic, v.1, n.1, 2016

ConsoleBox – Uma ferramenta para verificação e validação de APIs RESTful

ConsoleBox – A tool for verification and validation of RESTful APIs

Bruno Costa[1], Igor de Castro[2], Rodrigo Imenes[2]

[1]Docente do Centro Universitário Serra dos Órgãos, Doutorando em Informática pela Universidade Federal do Rio de Janeiro; [2]Estudante do Centro Universitário Serra dos Órgãos - curso de Ciência da Computação

Resumo

Pesquisas recentes evidenciam a necessidade de uma ferramenta de software que auxilie desenvolvedores na especificação, validação e verificação de APIs que seguem o estilo REST (APIs RESTful). Este projeto propõe tal ferramenta, denominada ConsoleBox, que auxilia desenvolvedores a verificarem se as restrições do estilo REST estão sendo atendidas apropriadamente, além de indicar estratégias visando a qualidade das APIs.

Abstract

Recent studies highlight the need for a software tool that helps developers in the specification, validation and verification of APIs that follow the REST style (RESTful APIs). This project proposes such tool, called ConsoleBox, which helps developers to verify whether the REST constraints are being properly addressed, and indicate strategies for the quality of APIs.

INTRODUÇÃO

A Arquitetura de Software é uma subárea recente da Engenharia de Software, formalmente definida no início dos anos 90. Mesmo com o seu desenvolvimento recente é considerada como o principal artefato que irá ditar a qualidade do sistema como um todo (BASS; CLEMENTS; KAZMAN, 2012), desde sua especificação e desenvolvimento até a implantação e manutenção. Na especificação da arquitetura de um sistema devem-se levar em conta os requisitos funcionais (funções ou tarefas que o sistema deverá executar) e os atributos de qualidade, também chamados de requisitos não funcionais. O documento de referência ISO/IEC 25010:2011 (ISO/IEC, 2011) apresenta cinco atributos de qualidade e suas características:

1. Confiabilidade: capacidade do software de manter o nível de desempenho especificado, quando usado sob condições específicas;
2. Usabilidade: capacidade do software ser entendido, aprendido e atraente ao usuário, quando usado sob condições específicas;
3. Desempenho: capacidade do software de fornecer a velocidade de resposta adequada, relacionado à quantidade de recursos usados, sob condições estabelecidas;
4. Manutenibilidade: capacidade do software de ser modificado. As modificações podem incluir correções melhorias ou adaptações do software a mudanças no ambiente ou nos requisitos;
5. Portabilidade: capacidade de um produto de software ser transferido de um ambiente para outro;

No decorrer dos projetos de software diversos elementos foram especificados e agrupados sistematicamente de forma a desenvolver

sistemas que se adequassem aos requisitos funcionais e atributos de qualidade. Na arquitetura de software, estes agrupamentos são chamados de estilos arquiteturais. Um estilo arquitetural é uma especialização de elementos e conectores alinhados a um conjunto de restrições de como eles devem ser utilizados (CLEMENTS et al., 2010) e, assim, delineando os atributos de qualidade que serão impactados. Atualmente estilos arquiteturais são amplamente utilizados em projetos de software, uma vez que apresentam uma solução bem fundamentada para problemas recorrentes. Em um projeto, é comum a utilização de vários estilos com propósitos distintos, por exemplo, o estilo em camadas, que favorece o desacoplamento e manutenibilidade e o estilo de serviços que fornecem mecanismos de interoperabilidade entre sistemas.

Um dos estilos arquiteturais mais utilizados em sistemas distribuídos é o estilo REST (FIELDING; TAYLOR, 2002). Esse estilo é amplamente utilizado na especificação e implementação de interfaces de comunicação, denominadas APIs, entre sistemas e dispositivos móveis. Baseado no protocolo HTTP, o estilo REST foi proposto originalmente na tese de doutorado de Roy Fielding (2000). Fielding definiu formalmente um conjunto de restrições que, aplicadas corretamente, diversos atributos de qualidade são positivamente impactados. APIs que implementam corretamente as restrições REST são denominadas APIs RESTful (RICHARDSON; RUBY, 2007).

A Web é considerada como uma grande plataforma para a implementação de sistemas baseados no estilo arquitetural REST (WILDE; PAUTASSO, 2011). Diariamente milhares de APIs são criadas para os mais diversos fins. Além disso, diversas iniciativas têm proposto soluções para a especificação de APIs RESTful. Porém, especificar e implementar APIs RESTful (APIs que implementam corretamente as restrições REST), não é uma tarefa fácil. Pesquisas recentes (MALESHKOVA; PEDRINACI; DOMINGUE, 2010; RENZEL; SCHLEBUSCH; KLAMMA, 2012) demonstram que a maioria APIs existentes não implementam corretamente as restrições impostas pelo estilo e, assim, os atributos de qualidade são negligenciados ou negativamente impactos. Resultando em APIs com baixa qualidade. Os autores deste projeto desconhecem uma solução que, além de auxiliar a concepção de APIs RESTful, forneça mecanismos para auxiliar a verificação e validação APIs após a implementação, de forma que garanta a realização de requisitos de atributos de qualidade.

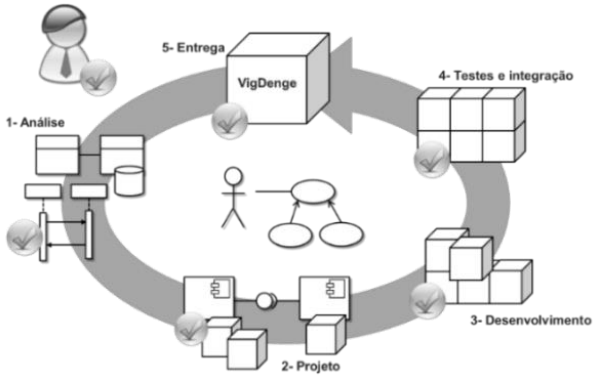
O objetivo deste projeto é desenvolver uma ferramenta web, denominada ConsoleBox, que auxilie arquitetos e desenvolvedores a projetarem, verificarem e validarem APIs RESTful, levando em consideração as restrições do estilo REST e os impactos em atributos de qualidade.

METODOLOGIA

O desenvolvimento do projeto foi realizado em quatro grandes etapas, a saber: (i) pesquisa bibliográfica com o objetivo coletar o referencial teórico acerca de abordagens e ferramentas de apoio ao desenvolvimento de APIs RESTful e delimitar o estado da arte na pesquisa sobre especificação de APIs RESTful; (ii) testes de utilização das APIs da Marvel e Transparência Brasil, de forma a aprofundar o conhecimento técnico sobre APIs RESTful; (iii) desenvolvimento da ferramenta ConsoleBox, e; (iv) avaliação da ferramenta através da verificação de diferentes APIs.

Em termos de atividades de desenvolvimento, optou-se por um processo de desenvolvimento ágil, iterativo e incremental, centralizado em casos de uso onde, para cada iteração, são executadas cinco fases básicas (validadas continuamente): análise, projeto, desenvolvimento, teste e integração e, entrega. A Figura 1 ilustra as atividades do processo.

Figura 1: Processo de Desenvolvimento



Legenda: 1. Análise - compreensão os requisitos de domínio e definir o escopo da iteração; 2. Projeto - definição os componentes e especificação da arquitetura; 3.Desenvolvimento – implementação dos componentes; 4.Testes e Integração - objetiva assegurar a conformidade com os requisitos de domínio modelados nos casos de uso, verificar os componentes, integrá-los, executar testes integrados e testes de integração; 5.Entrega – liberação do componente para utilização.

A avaliação final do projeto foi realizada através da verificação das APIs Marvel e Transparência Brasil. Esta verificação foi feita utilizando aleatoriamente algumas URIs das APIs utilizando a ferramenta ConsoleBox.

O objetivo da ferramenta ConsoleBox é realizar verificações em APIs REST de forma a averiguar se ela atende às restituições RESTful e oferecer informações para melhoria das APIs. A versão atual da ConsoleBox leva em consideração apenas a verificação da restrição Interface Uniforme, além da verificação acerca do versionamento do recurso. Trabalhos futuros compreendem a verificação das demais restrições do estilo.

A Interface Uniforme é uma restrição composta de regras para comunicação entre a API e o sistema que interage com ela. A primeira regra é a identificação do recurso, onde cada um deve ter um identificador específico de acesso, como "http://www.unifeso.br/professores", para os recursos representando professores do UNIFESO ou "http://www.unifeso.br/aluno/rodrigo", identificando o recurso que apresenta informações sobre o aluno "Rodrigo". Este identificador é denominado URI. A segunda regra está relacionada à definição da representação do recurso (como HTML, JSON ou XML). A terceira regra impõe que APIs RESTful devem seguir os métodos HTTP (GET, PUT, POST, DELETE) para interagir com recursos, além de que a URI não deve conter verbos. Por fim, os recursos devem ser auto relacionados entre si, de forma que os sistemas que utilizem as APIs consigam navegar entre os diversos recursos associados, esta restrição é denominada HATEOAS (Hypermedia as the Engine of Application State).

Arquitetura

O desenvolvimento da ferramenta foi dividido em duas partes denominadas Backend e Frontend. O Frontend é composto pelos componentes responsáveis pela interface do sistema com o usuário. O Backend caracteriza-se pelos componentes executados no lado do servidor.

Backend

O desenvolvimento do Backend foi realizado utilizando a linguagem Python em sua versão 2.7. Para aprimorar a arquitetura e manter o projeto flexível, foi utilizada um framework ORM (Object Relational Mapper), que que faz uma relação dos objetos da aplicação (segundo o paradigma de Orientação a Objetos) com as tabelas de um banco de dados relacional. Frameowrks ORM promovem uma manutenção eficiente do software.

Para utilização do ORM em Python, foi escolhido o framework Web Django, que possui diversas características próprias para servidores Web que utilizam Python. O Django conta também com o Django Rest Framework, que agiliza a implementação de servidores REST e já realiza a maioria dos métodos necessários a um servidor.

Frontend

O segundo componente da arquitetura é o Frontend. No desenvolvimento do Frontend,

foi utilizado o framework AngularJS, com o objetivo de controlar as principais rotinas da interface. Esse framework foi utilizado visto a linguagem HTML permitir uma integração eficiente com códigos Javascript dinâmicos. Além disso, o AlgularJS utiliza o padrão arquitetural MVC (Model-View-Controller), tornando a aplicação mais coesa e menos acoplada entre os módulos.

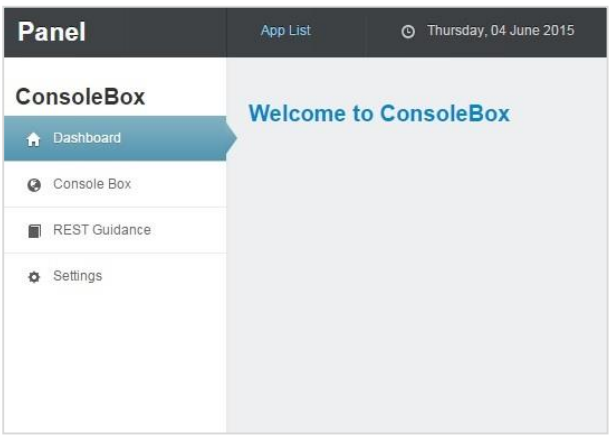
Também foram adicionados outros recursos para desenvolvimentos futuros na ferramenta, como a instalação do Bower. O Bower é um gerenciador de frameworks e bibliotecas onde, através da linha de comando, é possível obter um novo pacote ou simplesmente atualizar uma biblioteca já instalada. Isso agiliza o desenvolvimento, porque diminuir o tempo necessário para instalação de um novo recurso.

Outra importante biblioteca utilizada foi o jQuery, que oferece diversas funcionalidades já implementadas em Javascript, como manipulação do HTML, controle de eventos e animações.

ConsoleBox: Uma Ferramenta para Verificação e Validação de APIs RESTful

Nessa seção será demonstrado o funcionamento básico da ferramenta, iniciando pela tela inicial, como apresentada na Figura 2. Nela é possível observar um menu à esquerda que contém os dois principais módulos da ferramenta: Console Box e REST Guidance.

Figura 2: Tela inicial da ConsoleBox



Guia para avaliação arquitetural de APIs RESTful

O guia para avaliação arquitetural possui o conteúdo do artigo “Evaluating a Representational State Transfer (REST) Architecture - What is the impact of REST in my architecture?” (COSTA, 2014). Este artigo apresenta diversas diretrizes para o desenvolvimento de APIs RESTful. Do seu conteúdo foram selecionadas alguns tópicos, disponibilizados em um módulo da ferramenta ConsoleBox intitulado REST Guidance. O objetivo é oferecer aos desenvolvedores as diretrizes do artigo em um formato digital, promovendo usabilidade para os usuários.

Figura 3: Guia para Avaliação Arquitetural REST

A Figura 3 apresenta a tela inicial do Guia de Avaliação REST. Para melhor utilização, ao selecionar este módulo o usuário é redirecionado para outra página, onde não estão presentes os conteúdos do painel à esquerda da página principal da. No Guia existem os módulos do Guidance, Cheat Sheet e Proof of Concept. Todos eles serão explicados a seguir.

O módulo Guidance possui tópicos importantes sobre restrições REST, exemplos de cenários gerais de atributos de qualidade para REST, e questões de design que afetam atributos de qualidade. No módulo Cheat Sheet, são apresentadas as questões de design e quais atributos de qualidade estão ligados a ela. No

módulo Proof of Concept, são exibidas as informações da prova de conceitos realizada no artigo mencionado anteriormente (COSTA, 2014).

Módulo ConsoleBox

O segundo módulo da ferramental é a ConsoleBox em si. Para utilizá-la é necessário o cadastro de uma nova aplicação. O conceito de "aplicação" na ConsoleBox caracteriza um registro de testes em uma determinada API (Figura 4).

Após o cadastro da aplicação, basta acessá-la e a ferramenta irá direcionar o usuário para a interface de testes de API, onde haverá no menu lateral o item ConsoleBox, e estará na tela principal do site.

Figura 4: Tela de aplicativos da ConsoleBox

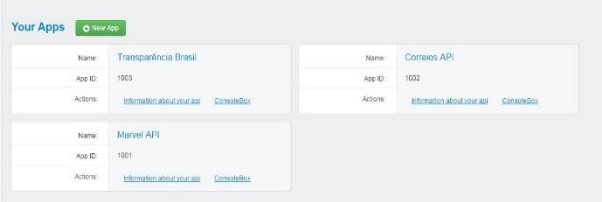
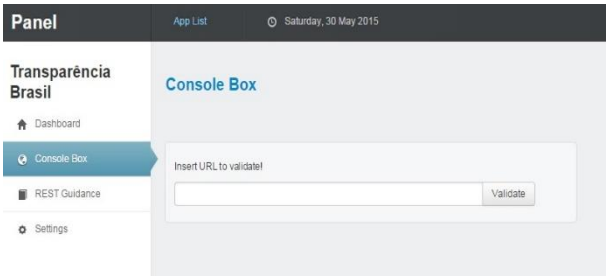


Figura 5: Tela de Verificação da ConsoleBox



Nessa tela, basta inserir a URL e a ConsoleBox irá verificar atributos de qualidade na própria URL.

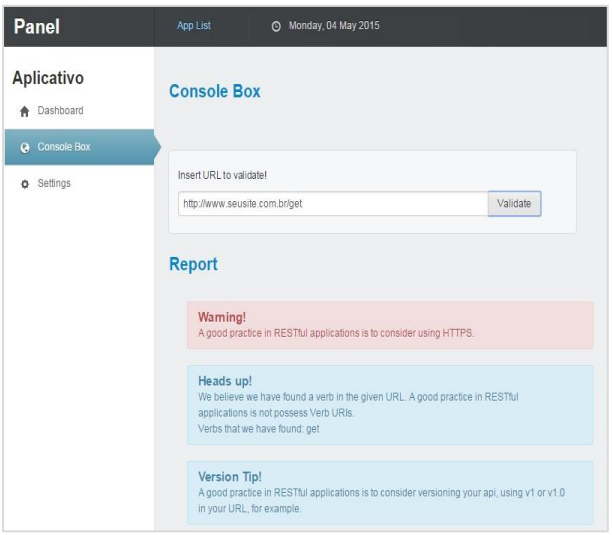
Verificação da restrição Interface Uniforme na Consolebox: versionamento e inexistência de verbos na URI

As verificações das URIs realizadas na ConsoleBox são feitas através de expressões regulares. Cada expressão regular testa uma condição de verificação e retorna um erro específico em caso de encontrar problemas.

Para verificar se uma URL está utilizando versionamento ou não, é utilizada uma expressão regular '(.*)/v\d+(\.\d)*/?'. Essa expressão é capaz de verificar se existe na URL um versionamento no formato "/v1", "/v1.1" ou até mesmo "/v1.1.1.1". Com isso, essa expressão consegue detectar se existe realmente uma versão declarada na URL ou não. Outra expressão regular é a '^https', que verifica se a URL se utiliza de HTTPS ou não.

Para verificação de verbos na URL, foi utilizada uma biblioteca da linguagem Python conhecida como NLTK . Essa biblioteca possui um grande repertório de palavras que são caracterizadas como verbos. Todos os verbos testados durante a implementação apresentaram resultado positivo. No caso da verificação se a URL era válida ou não, foi utilizado um método próprio do framework Django. Abaixo segue um exemplo do retorno das verificações no site.

Figura 6: Exemplos de mensagens de erros e avisos da ConsoleBox



Verificação de HATEOAS na ConsoleBox

Através desta mesma tela é possível verificar o nível de recursos interligados em APIs RESTful. Na versão atual da ConsoleBox este recurso está em fase final de desenvolvimento.

RESULTADOS

Nesta seção será descrita a avaliação da ConsoleBox em algumas APIs populares com o objetivo de verificar se a ferramenta atinge seus objetivos.
Os testes de validação da ConsoleBox foram executados em 4 APIs diferentes, com base em URIs disponíveis em sua documentação. Foram selecionados aleatoriamente 10 URIs utilizadas no teste. Os resultados dos testes é apresentado a seguir.

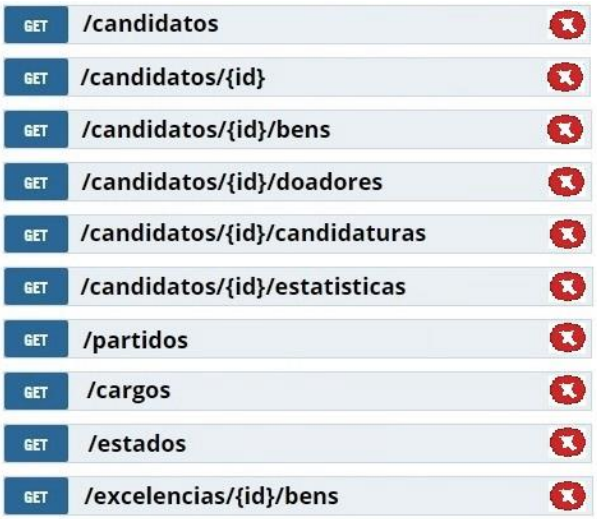
Os testes serão divididos pelo nome da API e uma tabela que representa as 10 URIs testadas com um ícone ao lado direito, indicando se passou em todas as verificações da ConsoleBox. Se for um check verde, representa que passou em todos e, se for um "X" vermelho, indica que foi reprovado em alguma validação, e será então explicado logo abaixo da imagem a falha gerada.

Figura 7: Resultados da API Marvel

Todas as URIs foram validadas com sucesso. O versionamento existe (v1) e a API seguiu as regras da restrição Interface Uniforme.

Resultados da Transparência Brasil API:

Figura 8: Resultados da Transparência Brasil API

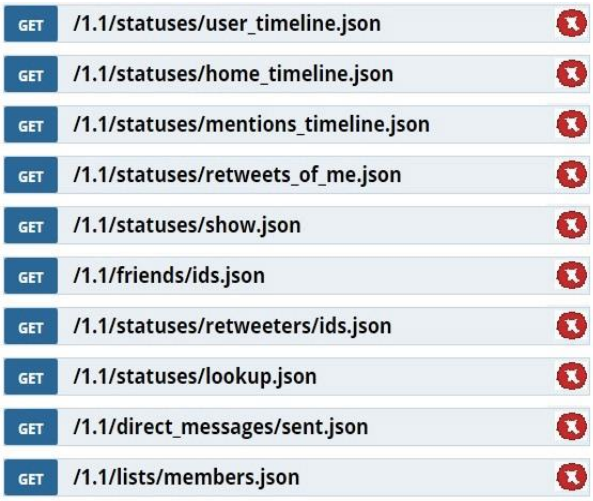


Os resultados da API Transparência Brasil marcados como inválidos pois a API não possui versionamento.

Figura 9: Resultados da Paypal API

Sobre os resultados falhos, a única mensagem de erro exibida era por possuir um verbo, geralmente a última palavra das URIs.

Figura 10:Resultados da Twitter API



A condição de falha apareceu pois a expressão regular esperava o versionamento com a caractere "v" e não somente o número, por isso foi demonstrada somente a mensagem de versionamento.

DISCUSSÃO

Com a validação aplicada, foi possível perceber que algumas APIs não realizam o versionamento, e a API do Paypal possui verbos nas URIs. A API da Marvel se destacou por passar em todas as verificações da ConsoleBox, não possuindo nenhum alerta. Dessa forma, foi possível demonstrar que a ConsoleBox desenvolvida é capaz de realizar uma verificação eficiente da restrição Interface Uniforme em APIs REST.

A ferramenta ConsoleBox foi implementada e disponibilizada através de uma página na Internet. Através dela é possível realizar avaliações arquiteturais em novos e antigos sistemas com arquitetura baseada em REST, sendo assim uma ferramenta que é capaz de auxiliar e realizar verificações nesse tipo de arquitetura. Desta forma, o trabalho atingiu aos três primeiros objetivos específicos propostas, auxiliando arquitetos na verificação da APIs RESTful, principalmente no que tange à verificação de características nas URIs que garantem a realização de restrições REST, como a Interface Uniforme.

A ConsoleBox, atualmente, ainda conta com algumas limitações a nível de verificação. O primeiro ponto importante é a base de verbos utilizada, que precisa ser complementada e também possuir recursos para conseguir trabalhar com outras línguas.

CONCLUSÃO

Com a implementação da ConsoleBox tornou-se viável uma ferramenta que avalie arquiteturas RESful. As principais contribuições deste projeto foram: (i) o guia para verificação arquitetural no formato digital e (ii) a ferramenta para verificação automática de URIs em APIs de forma a validarem se foram atendidas a restrição REST Interface Uniforme.

Diante desse quadro, é possível que trabalhos sejam feitos para ampliar as verificações já existentes, de forma a atender a outras restrições do estilo REST, além da Interface Uniforme. Também é possível ampliar o conteúdo do site, criando, por exemplo, um relatório com informações de cada aplicativo que tenha usado a ConsoleBox, de forma a permitir ao usuário verificar a evolução de sua API. Outra melhoria a ser realizada seria criar uma forma onde o usuário cadastraria todas as rotas de sua aplicação, e então o sistema se encarregaria de executá-las sempre que requerido em uma única vez.

O projeto já foi referenciado em dois artigos científicos, um deles publicado em uma das mais importantes conferências mundiais de arquitetura de software (COSTA et al, 2015); o outro publicado em uma das principais revistas científicas de computação em nível mundial (COSTA et al, 2016).

Além disso, o projeto promoveu o trabalho de conclusão de curso de dois alunos do curso de graduação em Ciência da Computação do UNIFESO (Rodrigo Imenes e Igor de Castro).

Finalmente, dois alunos do projeto Jovens Talentos da FAPERJ participaram do projeto realizando atividades de criação de um manual de desenvolvimento para alunos do ensino médio que não conhecem a área de desenvolvimento de software possam também

iniciar seus estudos utilizando a ferramenta ConsoleBox

REFERÊNCIAS

2. BASS, L.; CLEMENTS, P.; KAZMAN, R. Software Architecture in Practice. [s.l.] Addison-Wesley, 2012.

3. ISO/IEC. ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.

4. COSTA, B; PIRES, P. F. ; DELICATO, F. C. ; OQUENDO, F . Towards a View-Based Process for Designing and Documenting RESTful Service Architectures. In: the 2015 European Conference, 2015, Dubrovnik. Proceedings of the 2015 European Conference on Software Architecture Workshops - ECSAW 15, 2015. p. 1.

5. COSTA, BRUNO; PIRES, P. F. ; DELICATO, F. C. ; Merson, P. Evaluating REST architectures-Approach, tooling and guidelines. The Journal of Systems and Software, v. 112, p. 156-180, 2016.

6. CLEMENTS, P. et al. Documenting Software Architectures: Views and Beyond. [s.l.] Pearson Education, 2010. p. 608

7. FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern Web architecture. ACM Transactions on Internet Technology, v. 2, n. 2, p. 115–150, 1 maio 2002.

8. RICHARDSON, L.; RUBY, S. Restful web services. 1 maio 2007.

9. WILDE, E.; PAUTASSO, C. REST: From Research to Practice. Springer Science & Business Media, 2011. p. 526

10. MALESHKOVA, M.; PEDRINACI, C.; DOMINGUE, J. Investigating Web APIs on the World Wide Web2010 Eighth IEEE European Conference on Web Services. Anais...IEEE, dez. 2010

Contato:

Nome: Bruno Carlos da Cunha Costa
e-mail: brunocosta.dsn@gmail.com

: