Especificação de Uma Arquitetura Orientada à Serviços Web REST Para Criação de Um Ambiente Virtual de Aprendizagem

Eduardo Silva Vieira[1], **Wesley R. Oliveira**[1], **Mário Meireles Teixeira**[1,2]

[1]Laboratório de Sistemas Avançados da Web – LAWS

[2]Departamento de Informática
Universidade Federal do Maranhão (UFMA) – São Luís, MA – Brasil

edusvieirap@gmail.com, wesley231096@hotmail.com, mario@deinf.ufma.br

Abstract. *Although technological advances have brought new problems for society, it is undeniable that their development has made it possible to break through barriers and challenges previously unimagined. In education, it allowed communication between students and teachers to move beyond classrooms, changing the established pattern. However, with the growth of connected users, it is necessary to develop systems with more efficient architectures that allow the interoperability, scalability and elasticity of the applications according to the demand. In this sense, as a proposal to improve and accelerate the construction of educational tools, this work aims at specifying the methods and main tools for developing a virtual learning environment, called Classroom, mentioning the effectiveness, viability and difficulties encountered in constructing this a system based on a REST-oriented architecture, an integral part of GReAT, a serious game authoring tool.*

Resumo. *Embora o avanço tecnológico tenha trazido novos problemas para a sociedade, é inegável que o seu desenvolvimento tornou possível o rompimento de barreiras e desafios antes nunca imaginados. Na educação, ele permitiu que a comunicação entre estudantes e professores fosse para além das salas de aula, mudando o padrão estabelecido. No entanto, com o crescimento de usuários conectados, faz-se necessário o desenvolvimento de sistemas com arquiteturas mais eficientes que permitam a interoperabilidade, a escalabilidade e a elasticidade das aplicações de acordo com a demanda. Neste sentido, como proposta para aprimorar e agilizar a construção de ferramentas educacionais, este trabalho visa a especificação dos métodos e principais ferramentas para desenvolvimento de um ambiente virtual de aprendizagem, denominado Classroom, mencionando a eficácia, a viabilidade e as dificuldades encontradas ao construir este sistema baseado em uma arquitetura orientada a serviços REST, parte integrante da GReAT, uma ferramenta de autoria de jogos sérios.*

1. Introdução

Embora o avanço tecnológico tenha trazido novos problemas para a sociedade, é inegável que o seu desenvolvimento tornou possível o rompimento de barreiras e desafios antes nunca imaginados. Os sistemas computacionais modernos mudaram a maneira

como nos relacionamos, trabalhamos e estudamos. Na educação, eles permitiram que a comunicação entre estudantes e professores fossem para além das salas de aula, mudando o padrão estabelecido. No entanto, com o crescimento no número de usuários conectados e a evolução dos dispositivos físicos, faz-se necessário o uso de novos modelos arquiteturais para modelar aplicações mais eficientes.

Nesse sentido, para promover a interoperabilidade, a escalabilidade e a elasticidade, várias aplicações modernas estão sendo modeladas por meio de uma arquitetura orientada à serviços. Esta prega o princípio de que cada funcionalidade da aplicação deve ser disponibilizado como um serviço, o qual é consumido por clientes por meio de uma interface comum usando métodos e padrões estabelecidos na rede [Norbert Bieberstein et al 2006].

Como mencionado por [M. F. Ribeiro and R. E. Francisco 2016] estas aplicações dependem da adequada comunicação entre diversos sistemas oferecidos por múltiplas organizações e alocados em servidores geograficamente dispersos. Dessa forma, o sucesso destas operações exige elevados níveis de interoperabilidade onde, por questões de segurança, escalabilidade e interfuncionalidade, a adoção de tecnologias eficientes para comunicação entre sistemas distribuídos constitui pilar fundamental.

Neste contexto, os serviços web REST apresentam-se como um estilo arquitetural para permitir que aplicações possam ser desenvolvidas independentes de arquiteturas, tecnologias ou linguagens de programação. Esta interoperabilidade é decorrente do uso de padrões estabelecidos na rede e de uma linguagem de comunicação comum entre os sistemas envolvidos, como JSON.

O *Classroom*, um ambiente virtual de aprendizagem, foi criado e modelado por meio de tecnologias modernas e em uma arquitetura orientadas a serviços. Dessa forma, pretendemos construir uma aplicação que seja eficaz no seu objetivo, o qual é permitir que professores e estudantes possam se relacionar e ao mesmo tempo expansiva e fácil de manter.

Este artigo visa especificar os métodos e as principais ferramentas para desenvolver um ambiente virtual de aprendizagem, denominado *Classroom*, mencionando a eficácia, a viabilidade e as dificuldades encontradas ao construir este sistema baseado em serviços web REST para ser incorporado à ferramenta GREaT.

2. Fundamentação Teórica

2.1. GREaT

A GREaT, ou *Game Ready Authoring Tool*, é um ambiente de autoria de jogos sérios pelo usuário final, a qual destina-se a professores e instrutores sem habilidade em programação de computadores que desejam realizar autoria de jogos educacionais [Alana Oliveira et al 2010]. Ela é formado por 4 (quatro) módulos: *Authoring*, *Catalog*, *Game Center* e *Classroom*.

O *Authoring* é um ambiente web para criar jogos com base em templates pré-configurados. Já o *Catalog* é um catálogo de jogos para o usuário indexado por nome e tipo de conteúdo. Por outro lado, o *Game Center* permite acompanhar o progresso, últimos placares e compartilhar resultados com amigos. Por fim, o *Classroom* é um ambiente virtual de aprendizagem.

Em uma versão inicial a GREaT foi desenvolvida segundo uma abordagem cliente servidor tradicional, baseada no padrão arquitetural MVC (*Model View Controller*) com tecnologia PHP e MySQL. Embora esta abordagem atenda ao propósito de desenvolvimento de soluções de informatização baseados na Web e seja ainda largamente utilizada, revelou-se limitante, no que se refere na inclusão de novos módulos na ferramenta, desenvolvidos em diferentes linguagens e até por terceiros. Isso se deve principalmente a utilização do padrão MVC ser ainda assim preso a uma solução tecnológica específica. Ademais uma ferramenta desse tipo é utilizada por múltiplos usuários concomitantes, o que acarreta problemas de desempenho e até de disponibilidade, como foi possível experimentar em algumas situações reais. Por esses motivos, vem sendo feita pela autora uma reengenharia da ferramenta GREaT obedecendo ao paradigma orientado a serviços.

2.2. REST - Representational State Transfer

Web Service é um conceito que continua crescendo nos últimos anos, sobretudo por causa de novas abordagens como o modelo REST. Usando uma linguagem intermediária universal como o XML ou JSON, as aplicações modeladas podem ser programadas em diferentes linguagens de programação e tecnologias, aumentando a interoperabilidade entre sistemas e a comunicação entre aplicações diferentes. Dessa forma, é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis entre si [Deitel H. and Deitel P. 2010].

Esta abordagem permite que os recursos de um sistema estejam disponíveis via rede de maneira uniforme. Isso significa que uma aplicação pode chamar outra para realizar tarefas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Em outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo serviço.

O REST ou *Representational State Transfer*, em português Transferência de Estado Representacional, é um estilo de arquitetura criado por Roy Fielding, um dos criados do protocolo HTTP, que define um conjunto de restrições e propriedades que orientam os desenvolvedores para uma maneira correta de se modelar aplicações na rede. Nesta arquitetura, cada recurso é identificado por um URI (*Uniform Resource Identifier*), ou Identificador de Recurso Uniforme, o qual será utilizado para identificar unicamente o serviço oferecido.

Como dito anteriormente, para as aplicações se comunicarem, necessita-se de uma linguagem de comunicação intermediária universal, como XML ou JSON. O REST faz uso do JSON, um acrônimo de JavaScript Object Notation, especificado por Douglas Crockford em 2000, é um formato compacto, de padrão aberto, independente, para troca de dados simples e rápida entre sistemas. Ademais utiliza texto legível por humanos, no formato atributo-valor, substituindo o uso do XML.

Portanto, aplicações que obedecem ao estilo arquitetural REST fornecem interoperabilidade entre sistemas de computadores na rede. Os *web services* compatíveis com REST permitem que os sistemas solicitantes acessem e manipulem os recursos da aplicação por meio dos métodos padrões do HTTP, como *GET*, *POST*, *PUT* e *DELETE*. Por exemplo, o serviço /classroom/students/ acessado pelo método *GET* retorna para o solicitante uma lista de todos os estudantes, por outro lado, o serviço /classroom/students/20

acessado pelo método *DELETE* remove o registro do estudante cuja a matrícula é igual a 20.

3. Metodologia

Para implementação do protótipo inicialmente foram feito um estudo de tecnologias que melhores atenderam aos propósitos do projeto e como estas se relacionam com a arquitetura REST, escolhida para modelar a aplicação. Com base nisso, conseguimos realizar um estudo de eficácia e viabilidade com as principais dificuldades encontradas.

3.1. Linguagens de Programação

A linguagem escolhida foi o Python, em sua versão 3.7.0. Esta escolha se baseia no fato do Python ser uma linguagem de programação de alto nível com uma curva de aprendizado pequena comparada a outras linguagens e fácil integração com as demais tecnologias usadas no projeto.

3.2. Framework

Para desenvolvimento do projeto usamos o Flask, um *micro framework* web escrito em Python e baseado na biblioteca WSGI Werkzeug e na biblioteca de Jinja2. Ele está licenciado sob os termos da Licença BSD, o que permite a criação e distribuição do *Classroom* livremente. Ademais o Flask é uns dos principais *frameworks* em Python. Ele possui uma rica biblioteca e mostrou um bom desempenho para o escopo da aplicação, tornando-o a nossa escolha. Dentre as funcionalidades providas pelo Flask se destacam o sistema de rotas com os métodos HTTP, parâmetros e condições, redirecionamento e paradas, renderização de modelo personalizada, tratamentos de erros e depuração, além da facilidade de configuração.

3.3. Banco de Dados

Para modelagem dos dados no sistema, utilizamos o MongoDB. Ele é uma banco de dados de código aberto, de alta performance, sem esquemas e orientado a documentos. Este SGBD foi utilizado no projeto devido a flexibilidade necessária para construção e manipulação dos dados no banco de dados.

4. Resultados

Como mencionado este trabalho tem como objetivo especificar a construção de um ambiente virtual de aprendizagem em uma arquitetura REST com o intuito de permitir que este módulo possa ser integrado na GREaT, uma ferramenta de autoria de jogos sérios, a qual está passando por um processo de reengenharia.

Inicialmente foi feito um estudo sobre os princípios e as técnicas relacionados à construção de sistemas baseados em REST, os quais possibilitam o desenvolvimento de uma aplicação que seja acessada por dispositivos com diferentes arquiteturas por meio de uma interface comum. Depois de realizar a especificação do sistema, a tarefa de definir os recursos essenciais foi relativamente simples, uma vez que já conhecíamos a maioria das funcionalidades. Neste ponto, identificamos os recursos essenciais do *Classroom* para cada entidade do sistema.

Logo em seguida, nós partirmos para a construção do sistema. Como mencionado o REST não é um método, mas um conjunto de princípios e técnicas que orientam o desenvolvedor a programar suas aplicações baseadas em serviços. Dessa forma, encontramos problemas em definir uma forma ideal para construir nossa aplicação. É importante ressaltar a importância do Flask nesta etapa, o seu sistema de rotas ajudou muito a definir um padrão de modelagem aos serviços disponíveis.

Outro ponto importante é a segurança em sistemas orientados a serviços. O REST, assim como outros padrões, não possui uma camada de segurança própria. Portanto cada desenvolvedor é responsável por criar mecanismos que impeçam que terceiros acessem dados confidenciais sem permissão. Neste projeto utilizamos a biblioteca PyCrypto, a qual usa o algoritmo *Advanced Encryption Standard* (AES) ou Padrão Avançado de Criptografia, para criptografar e decriptografar os dados do sistema.

Na tabela 1 visualizamos os recursos disponíveis para cada entidade, o método HTTP usado (*GET*, *POST*, *PUT* ou *DELETE*), a URI para identificar o recurso e uma pequena descrição do serviço.

Tabela 1. Recursos do Classroom

Objeto	Método	URI	Descrição
Turma	GET	/classroom/classes/<class_id>/	Obter uma turma específica.
Turma	GET	/classroom/classes/<class_id>/users/	Obter usuários de uma turma específica.
Turma	GET	/classroom/classes/<class_id>/notices/	Obter avisos de uma turma específica.
Turma	GET	/classroom/classes/<class_id>/tasks/	Obter tarefas de uma turma específica.
Turma	GET	/classroom/users/<user_id>/classes/	Obter turmas criadas por um usuário específico.
Turma	POST	/classroom/classes/	Criar uma nova turma.
Turma	DELETE	/classroom/classes/<class_id>/	Remover uma turma específica.
Turma	PUT	/classroom/classes/<class_id>/	Atualizar uma turma específica.
Turma	GET	/classroom/classes/<class_id>/invites/	Obter convites de uma turma específica.
Convite	POST	/classroom/invites/	Criar um novo convite.
Convite	PUT	/classroom/invites/<invite_id>/	Atualizar um convite específico.
Convite	DELETE	/classroom/invites/<invite_id>/	Remover um convite específico.
Usuário	GET	/classroom/users/<user_id>/	Obter um usuário específico.
Usuário	POST	/classroom/login/	Autenticar um usuário específico.
Usuário	POST	/classroom/signup/	Criar um novo usuário.
Usuário	DELETE	/classroom/users/<user_id>/	Remover um usuário específico.
Usuário	PUT	/classroom/users/<user_id>/	Atualizar um usuário específico.
Aviso	GET	/classroom/notices/<notice_id>/	Obter um aviso especifico.
Aviso	POST	/classroom/notices/	Criar um novo aviso.
Aviso	DELETE	/classroom/notices/<notice_id>/	Remover um aviso específico.
Aviso	PUT	/classroom/notices/<notice_id>/	Atualizar um aviso específico.
Tarefa	GET	/classroom/tasks/<task_id>/	Obter uma tarefa específica.
Tarefa	POST	/classroom/tasks/	Criar uma nova tarefa.
Tarefa	PUT	/classroom/tasks/<task_id>/	Atualizar uma tarefa específica.
Tarefa	DELETE	/classroom/tasks/<task_id>/	Remover uma tarefa específica.

5. Conclusão

Este projeto propôs e cumpriu o desenvolvimento de um ambiente virtual de aprendizagem modelado por meio da arquitetura de serviços REST, produzindo um resultado relevante no estudo dos benefícios e malefícios da aplicação de sistemas orientados a serviços para a pesquisa que vem sendo realizada no laboratório avançado de sistemas web (LAWS, em inglês) na Universidade Federal do Maranhão na área de ferramentas para autoria de jogos educacionais.

O uso de REST proporcionou uma fácil e simples integração com a GREaT, uma ferramenta de autoria de jogos sérios, por meio de uma arquitetura baseada em serviços

que permite que os módulos que a formam possam ser programadas em diferentes linguagens de programação e tecnologias, aumentando a interoperabilidade entre sistemas e a comunicação entre aplicações diferentes.

Com o desenvolvimento do Classroom comprovamos que o uso de uma arquitetura orientada a serviços promove muitos benefícios como a interoperabilidade entre sistemas, fácil integração e desenvolvimento mais simples, uma vez que visualizamos nossas aplicações como uma união de módulos e podemos trabalhar cada um individualmente. No entanto, percebemos dois problemas principais neste tipo de arquitetura como a dificuldade de se estabelecer um padrão de fato para se desenvolver e a ausência de uma camada de segurança nas aplicações desenvolvidas com REST.

6. Referencias

Ribeiro, M. F. and R. E.. *Web Services REST Conceitos, Análise e Implementação. Educação, Tecnologia e Cultura - E.T.C.*, [S.l.], n. 14, jun. 2016.

Oliveira, Alana, Mário Meireles Teixeira, Carlos Salles, (2010). *Um Ambiente de Autoria de Jogos Sérios pelo Usuário Final Aplicados a Educação*.

DEITEL, Paul; DEITEL, Harvey. *Java: Como Programar*. 8. ed. São Paulo: Pearson, 2010. 1144 p. Tradução de: Edson Furmankiewicz.

BIEBERSTEIN, Norbert; BOSE, Sanjay; FIAMMANTE, Marc; JONES, Keith; SHAH, Rawn. *Service-Oriented Architecture (SOA) Compass: Business Value, Planning , and Enterprise Roadmap (paperback)*. Published Oct 25, 2005. IBM Press.