Israel Henrique Silva de Lima

CMPaaS: Edição de Mapas Conceituais em Ambientes Virtuais Moodle

Vitória

2016

Israel Henrique Silva de Lima

CMPaaS: Edição de Mapas Conceituais em Ambientes Virtuais Moodle

Monografia apresentada ao Curso de Engenharia de Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Informática

Orientador: Prof. M.Sc. Wagner de Andrade Perin

Coorientador: Prof. Ph.D. Davidson Cury

Vitória

2016

Este trabalho é dedicado à todos que me apoiaram durante todo o meu período acadêmico.

Agradecimentos

Agradeço primeiramente a Deus por todas as graças alcançadas na vida e durante todo o curso.

Aos meus pais, pela educação, apoio emocional, financeiro e tempo dedicado a minha vida.

Em especial a Jalany, pelo carinho, paciência e apoio durante todo o tempo e principalmente o desenvolvimento desta monografia.

Ao meu orientador Wagner, por toda a ajuda oferecida no elaboração deste trabalho.

Ao meu amigo Josmar, pelas longas conversas, orientações e broncas.

Aos meus amigos Piero e Jefferson, pelos momentos de alegria e diversão.

E a todos aqueles que contribuíram direta ou indiretamente nos diversos aspectos de minha formação acadêmica e pessoal.

"Normal people believe that if it ain't broke, don't fix it.
Engineers believe that if it ain't broke, it doesn't have enough features yet.
(Scott Adams)

Resumo

Este trabalho apresenta a concepção e o desenvolvimento de um *plugin* de edição de mapas conceituais para a plataforma Moodle de modo a favorecer a integração entre este e a plataforma de serviços do projeto CMPaaS, que oferece uma diversidade de serviços para processamento de mapas conceituais. O objetivo é favorecer ao CMPaaS a integração à uma base de usuários consistente que podem ser potenciais fontes de dados para as ferramentas e pesquisas desenvolvidas no contexto do projeto CMPaaS. São detalhados a arquitetura, os processos e as tecnologias utilizadas na construção do *plugin*. Uma prova de conceitos é apresentada.

Palavras-chave: Mapas Conceituais. Moodle. CMPaaS.

Abstract

This work presents the design and development of a concept map editor in the form of a Moodle plugin. The created maps are saved by using a persistence service from CMPaaS, a platform that offers several concept map processing services. The processes and technologies used in the construction of the plugin are further detailed. We also explore the importance of making it available through Moodle (which is central to the infrastructure of a considerable number of Distance Education programs). The possibility of integrating Moodle to other platforms and services is also discussed.

Keywords: Concept Maps. Moodle. CMPaaS.

Lista de ilustrações

Sumário

1 Introdução

Mapas conceituais são uma ferramenta gráfica muito utilizada na representação de conhecimento acerca de um dado domínio de problema. São fáceis de utilizar e possuem ampla aplicação na educação, podendo ser usados para apresentar o conteúdo de uma aula, de um livro, capítulo ou utilizado como método de avaliação.

Por sua simplicidade no processo de elaboração, mapas podem ser elaborados com lápis e papel, porém é possível alcançar novas aplicações quando da utilização de recursos computacionais, tais como a utilização de recursos dotados de inteligência artificial para análise de mapas (PERIN et al., 2013) e mineração de dados (YOO; CHO, 2012).

Pensando nisso, pesquisadores da Universidade Federal do Espírito Santo especificaram e desenvolveram uma plataforma denominada CMPaaS. Esta plataforma possui a finalidade de fornecer um ambiente favorável à disposição e integração de coleções de ferramentas voltadas para a gestão e uso inteligente de mapas conceituais que são frutos de pesquisas desenvolvidas por diversos estudantes deste grupo, além de pessoas externas ao projeto.

O CMPaaS concentra diversas ferramentas de mapas conceituais desenvolvidas em projetos do laboratório de educação da UFES, e muitas delas necessitam levantar dados para comprovação de hipóteses. Um exemplo de ferramenta é o iMap, que foi desenvolvido por Perin (2014) em sua dissertação de mestrado.

O iMap é uma ferramenta utiliza inteligência computacional para fornecer uma camada de abstração entre um mapa conceitual e um avaliador. Assim é possível a um avaliador examinar um mapa conceitual através de perguntas e respostas em linguagem natural, sem haver a necessidade de um exame visual dos conceitos e relacionamentos presentes no mapa. No entanto a real capacidade desta ferramenta não pode ser efetivamente comprovada devido a escassez de utilizadores e de dados de entrada.

Nesta linha, este trabalho acredita que uma forma efetiva de ampliar as fontes de dados para a plataforma do CMPaaS é favorecer a integração com sistemas que já possuam base de usuários consolidadas. Assim, propõe-se a criação de uma ferramenta que permita o uso dos mapas nas plataformas utilizadas no ensino a distância, já que estes são nichos grandes e, a julgar pelos avanços na utilização de recursos computacionais na educação, tendem a se ampliar nos próximos anos (RIBEIRO et al., 2012).

Assim, este projeto visa o desenvolvimento de um plugin para construção de mapas conceituais dentro do gerenciador de cursos Modular Objected-Oriented Dynamic Learning Environment (Moodle) que permita o compartilhamento dos mapas produzidos com a

plataforma CMPaaS.

Para alcançar esses objetivos, foi realizada uma revisão bibliográfica aprofundada sobre mapas conceituais e a sua importância para a educação; também um estudo do funcionamento e arquitetura do Moodle para entender como podem ser desenvolvidos plugins para adicionar novos recursos à plataforma; foi feito um estudo sobre soluções de integração de sistemas para entender como seria possível a realização da interoperabilidade entre o Moodle e o CMPaaS. Por fim, o trabalho apresenta uma arquitetura conceitual e uma prova de conceitos para explorar as funcionalidades do plugin após a sua implementação.

1.1 Objetivo

O CMPaaS é uma plataforma baseada em serviços web que agrega diversas ferramentas de manipulação de mapas conceituais. Ferramentas estas que carecem de uma base de usuários consolidada para geração de conteúdo.

A motivação para esse projeto de graduação é portanto, buscar uma forma de trazer para o CMPaaS uma base de usuários que possa alimenta-lo com conteúdo a ser consumido pelas ferramentas que integram a plataforma.

Sendo assim este trabalho propõe a adoção de uma solução de integração de sistemas que leve a interoperabilidade entre o CMPaaS e uma plataforma que possua uma grande base de usuários.

1.2 Objetivo Específico

Com o objetivo geral deste trabalho definido, para alcança-lo foram estipulados os seguintes objetivos específicos:

- Estudar o que são mapas conceituais e suas aplicações na educação.
- Estudar a arquitetura do Moodle e seus *plugins*.
- Estudar técnicas e arquiteturas de integração de sistemas.
- Analisar a arquitetura da plataforma CMPaaS a fim de entender como realizar a integração dela com outras plataformas.
- Desenvolver um novo *plugin* para o Moodle segundo a arquitetura da plataforma.

1.3 Resultados Esperados

O resultado esperado deste trabalho é demonstrar a possibilidade de integração dos serviços providos pelo CMPaaS com outras plataformas. E com isto aumentar o uso das ferramentas presentes na plataforma e o conteúdo existente na mesma.

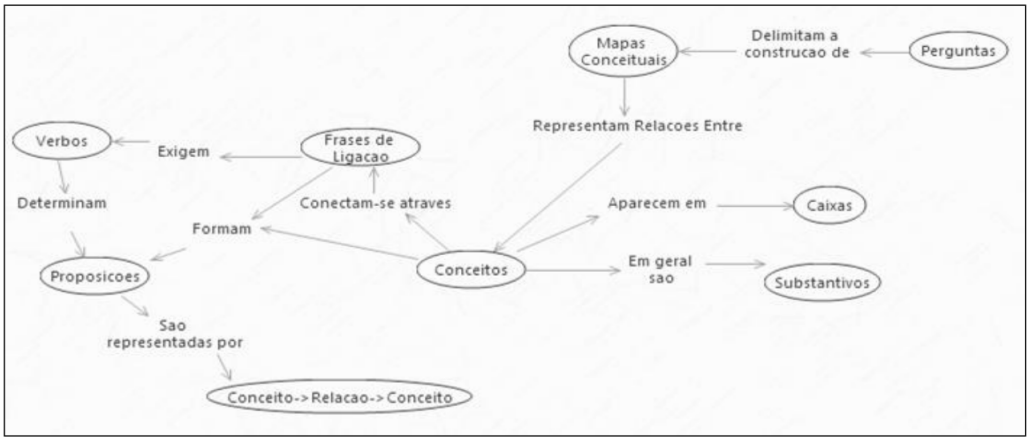
2 Mapas conceituais e educação

2.1 Origem e definição de mapas conceituais

O mapas conceituais foram criados na década de 70 durante um estudo de Novak que buscava entender como crianças aprendiam conceitos científicos (NOVAK, 2005). Durante seu estudo, Novak observou uma evolução nas proposições utilizadas pelas crianças, porém teve dificuldade de entender como esta mudança cognitiva ocorreu. Para compreender tal mudança, Novak desenvolveu um método baseado nas três ideias de Ausubel sobre a teoria da aprendizagem (AUSUBEL, 1963). A primeira ideia é que novos significados são criados a partir de conceitos e proposições previamente conhecidos pelo aprendiz. A segunda, é que a estrutura cognitiva do indivíduo é organizada hierarquicamente, com conceitos mais gerais ocupando posições superiores, acima dos conceitos mais específicos e complexos. Já a terceira ideia é que durante o processo de aprendizagem os conceitos vão se tornando mais específicos e precisos.

Assim foi criada essa ferramenta gráfica, que tem como objetivo fazer uma representação do conhecimento. Um mapa conceitual é constituído por interligações entre dois ou mais conceitos através de uma palavra de forma a gerar um significado. Ele é construído ligando-se um conceito a outro, através de um arco que contém uma palavra descritiva, que é usada para criar uma relação significativa entre os conceitos. Além disso, os conceitos mais gerais estão dispostos no topo do mapa e os mais específicos na parte inferior dele. A Figura 1 mostra um exemplo de mapa conceitual.

Figura 1 – Um exemplo de mapa conceitual



Fonte: Perin (2014, p. 28)

2.2 Mapas conceituais e educação

Os mapas conceituais podem ter diversas aplicações na educação. Podem por exemplo ser utilizados para representar o conhecimento adquirido em uma aula, ou então o conteúdo de um capítulo de um livro. Por se tratar de uma representação que é alimentada com novos conceitos conforme eles são adquiridos, pode ser utilizado pelo professor para acompanhar a aprendizagem de um estudante durante todo o período de um curso. Ademais, como os mapas conseguem representar a estrutura cognitiva de um indivíduo, eles podem ser utilizados também como método de avaliação de aprendizagem (PERIN, 2014).

A confiabilidade da avaliação da aprendizagem tradicional, como textos dissertativos e questões de múltipla escolha, tem sido questionada por pesquisadores e educadores. Nestes tipos de teste apenas os acertos são contabilizados e os erros são descartados, implicando em importantes informações para a avaliação serem desprezadas. As provas tradicionais não proporcionam a possibilidade do estudante apresentar como construiu o seu aprendizado. Elas conseguem avaliar somente a aprendizagem mecânica, não mostrando como que o aprendiz alterou suas estruturas cognitivas. Os mapas conceituais tem se destacado como alternativa a avaliação tradicional, pois conseguem demonstrar com facilidade as modificações cognitivas que ocorrem durante o processo de aprendizagem do estudante (DUTRA; FAGUNDES; CAÑAS, 2002).

Neste contexto, o *plugin* que será desenvolvido neste trabalho tem como finalidade permitir que estudantes de cursos gerenciados pelo Moodle respondam tarefas avaliativas com a elaboração de mapas conceituais. Assim poderá ampliar muito o uso de mapas como método de avaliação, já que o Moodle é uma plataforma utilizada por diversas instituições.

2.3 Mapas conceituais e tecnologia da informação

A construção de mapas conceituais é muito simples, uma caneta e um pedaço de papel são ferramentas suficientes para a confecção desse grafo. Porém a tarefa de revisá-lo, armazená-lo, e editá-lo a longo prazo pode ser muito cansativa e complexa. A introdução do uso de computadores na confecção de mapas pode facilitar a tarefa (NOVAK; CAÑAS, 2006).

Os primeiros programas de computadores criados com este objetivo se limitavam a mostrar apenas os mapas na tela, sem oferecer nenhum recurso adicional a ferramenta. Com a popularização do uso de computadores e da Internet houve um aumento na oferta de aplicações que tem como objetivo a construção de mapas gráficos. Porém há uma grande confusão entre usuários e desenvolvedores sobre a diferença entre mapas conceituais, organogramas e mapas mentais, devido a semelhança entre os diagramas (PERIN, 2014). Um mapa mental, por exemplo, é muito semelhante a um mapa conceitual

pois apresenta ligações entre ideias, mas ao contrario de um mapa mental não possui uma palavra descritiva ligando essas ideias e criando organicidade entre as ideias. Portanto uma aplicação de criação de mapas mentais não permite a construção de um mapa conceitual.

Na seção 2.3 de sua dissertação de mestrado, Perin (2014) realizou uma pesquisa sobre o estado da prática em mapas conceituais. Nela foi feito um levantamento sobre softwares que permitem a criação de mapas conceituais e quais são as funcionalidades oferecidas por eles. Nesta pesquisa foram avaliadas 16 ferramentas de edição de mapas, das quais apenas o CMapTools foi desenvolvido com o objetivo de oferecer suporte específico para mapas conceituais.

Percebendo esta carência, um grupo de pesquisadores do laboratório de Informática na Educação da Universidade Federal do Espírito Santo (Ufes) propôs uma plataforma de serviços para mapas conceituais, conhecida por CMPaaS (PERIN; CURY, 2016) cujo objetivo é fornecer serviços avançados para facilitar os processos e as aplicações dos mapas conceituais, tanto no ensino quanto no mercado. Neste trabalho, interessa-nos a criação de uma ferramenta integrada ao CMPaaS que proporcione a aplicação dos mapas conceituais na Educação a Distância.

2.3.1 CMPaaS

Na seção 2.2 vimos como os mapas conceituais são importantes para educação, seja para auxiliar o estudante a aprender novos conceitos, ou como método de avaliação, além de algumas outras aplicações educacionais.

Apesar de haver este grande interesse acadêmico em ferramentas computacionais para criação e manipulação de mapas, a fragmentação das pesquisas nesta área impede que o desenvolvimento destas aplicações ocorra de forma acelerada. Assim, a integração de ferramentas já existentes facilitaria o progresso de novas pesquisas.

Se houvesse uma infraestrutura de apoio a criação e manipulação de mapas conceituais o desenvolvimento de novas ferramentas seria facilitado. Um pesquisador poderia criar uma nova solução mais facilmente se não houvesse a necessidade dele se preocupar com toda uma estrutura de gestão de mapas. Por exemplo, alguém que queira desenvolver uma aplicação que avalie mapas conceituais poderia utilizar o conteúdo existente previamente em uma solução computacional de criação e armazenamento de mapas.

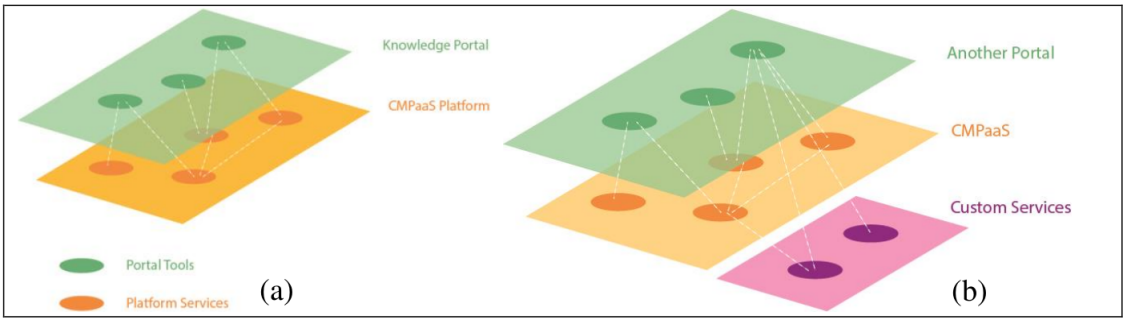
Um outro problema apontado por Perin (2014) é a dificuldade que a sociedade tem para acessar os resultados das pesquisas.

> Consideramos importante a criação de um mecanismo de acesso eficiente aos resultados das pesquisas científicas, para que a comunidade possa contribuir para a evolução delas. O que propomos, portanto, é o lançamento de bases para uma convivência mais estreita entre o mundo acadêmico e a sociedade em geral (PERIN, 2014).

Assim foi proposto por Perin (2014) a criação de uma plataforma denominada CMPaaS, cujo objetivo é possibilitar que a comunidade em geral acesse o resultado de pesquisas acadêmicas e prover uma infraestrutura que permita que esta comunidade crie e estenda suas funcionalidades.

Os serviços do CMPaaS são oferecidos através da Internet, podendo ser acessado de qualquer plataforma. Mas é necessário que exista um portal ou site associado a plataforma CMPaaS para que as ferramentas possam ser acessadas através de um navegador de Internet. Este portal foi nomeado como "Portal do Conhecimento"(PERIN, 2014).

Figura 2 – (a) Integração do Portal do Conhecimento com o CMPaaS. (b) Integração do CMPaaS com serviços externos



Fonte: Perin (2014, p. 81)

A Figura 2 ilustra a arquitetura do CMPaaS e como as aplicações fornecidas pelo portal utilizam os serviços da plataforma, muitas vezes consumindo mais de um dos serviços oferecidos. Além disso, é ilustrado também como um portal externo pode aproveitar os serviços disponíveis no CMPaaS, como é o caso do plugin que será apresentado posteriormente neste trabalho. Trata-se de um editor de mapas conceituais que armazena os mapas na plataforma. Assim são utilizados dois serviços fornecidos pelo CMPaaS: o de armazenagem de mapas e o de autenticação.

Atualmente a plataforma CMPaaS oferece os seguintes serviços:

- Criação, edição e formatação de mapas conceituais.
- Serviço de cadastramento e autenticação de usuários.
- Persistência e repositório de mapas.
- Controle de versão de mapas conceituais.
- Importação e exportação de mapas para o CmapTools
- Serviço de inferência para mapas, que permite que o usuário elabore perguntas sobre os mapas em linguagem natural.

- Criação de ontologias rasas a partir de um mapa.
- Geração automática de mapas a partir de textos.
- Serviço de mesclagem de mapas conceituais.
- Validação estrutural de mapas.

Como um dos objetivos do CMPaaS é oferecer um banco de dados de mapas conceituais para ser utilizado como base para pesquisas e criação de novas ferramentas, é desejável que seu serviço de repositório de mapas seja constantemente utilizado e incrementado com novos conteúdos. Pensando nisto este trabalho visa realizar a integração do Moodle com o CMPaaS, abrindo a possibilidade deste utilizar a ampla base de usuários do Moodle para alimentar o seu repositório de mapas.

2.4 Moodle e Educação à Distância (EaD)

Segundo CHAVES (1999) EaD é um ensino que ocorre quando há uma separação física entre mestre e aprendiz, sendo utilizadas tecnologias de transmissão de voz, dados e imagens para a comunicação entre ambos. A EaD também é definida no Decreto nº 5.622 de 19 de dezembro de 2005 (BRASIL, 2005).

> Art. 1o Para os fins deste Decreto, caracteriza-se a educação a distância como modalidade educacional na qual a mediação didático-pedagógica nos processos de ensino e aprendizagem ocorre com a utilização de meios e tecnologias de informação e comunicação, com estudantes e professores desenvolvendo atividades educativas em lugares ou tempos diversos (BRASIL, 2005).

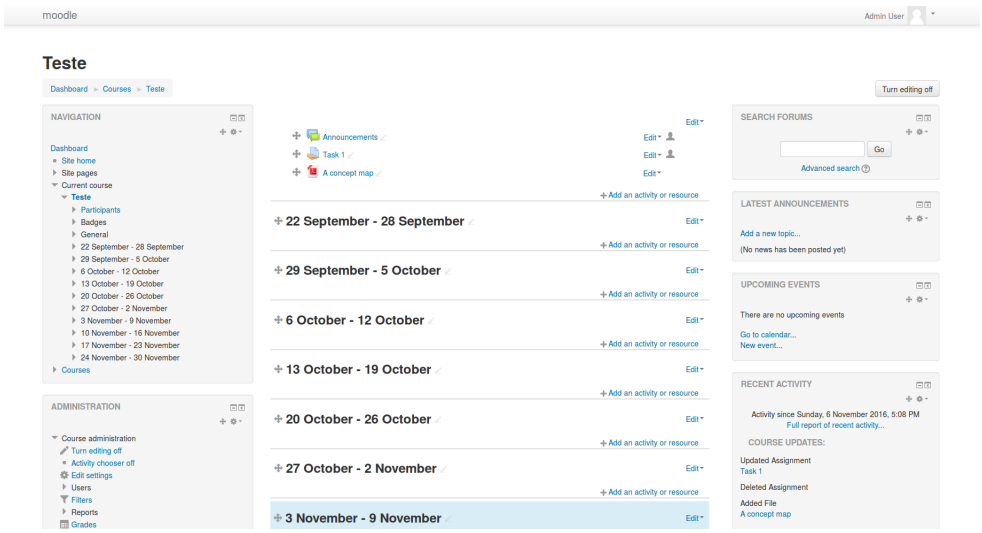
Em síntese é possível definir a educação a distância como um método de ensino no qual estudante e professor ficam separados fisicamente e a interação e feita através de tecnologias de comunicação (que podem ser dos mais diversos) de maneira a contornar esta separação.

Atualmente a educação a distância está relacionada ao uso de um Ambiente Virtual de Aprendizagem (AVA), que nada mais é que um sistema computacional cuja finalidade é prover suporte a atividades mediadas pelas tecnologias de informação e comunicação (ALMEIDA, 2010). É um ambiente que tem como objetivo integrar mídias e recursos, além de apresentar informações de forma organizada e permitir interação entre estudantes, tutores e professores (FRANCISCATO et al., 2008). Um exemplo de um AVA são os cursos que podem ser criados e gerenciados no Moodle.

O Moodle é um sistema gerenciador de cursos que foi criado por Martin Dougiamas em 1999. Ele é uma aplicação *open-source*, o que significa que ele é livre para ser instalado, utilizado, modificado e distribuído (DOUGIAMAS; TAYLOR, 2003).

O Moodle trabalha por padrão com cinco perfis de usuário: administrador, criador de cursos, professor, aluno e visitante. O administrador é o responsável técnico, é ele quem realiza a instalação e configuração do ambiente, além de manter ele funcionando corretamente. Já o criador de cursos tem como responsabilidade a criação e configuração dos cursos disponíveis na plataforma. Por sua vez, o professor tem como função o acompanhamento dos estudantes e a inserção de recursos e tarefas nos cursos. O aluno é quem realiza o curso, é ele quem vai utilizar os recursos e tarefas disponíveis no AVA. Por fim, o visitante é um usuário que só tem acesso as informações disponíveis na tela inicial do sistema. A Figura 3 mostra a interface de um curso no Moodle.

Figura 3 – Interface de um curso no Moodle



Fonte: Elaborada pelo autor

Ele foi escolhido para ser utilizado neste trabalho pois é amplamente usado por instituições em todo o mundo, possuindo mais de 73 mil sites registrados em 232 países (MOODLE, 2016b), e possui uma grande comunidade que contribui para correção de erros e criação de novas ferramentas. Além disso ele é utilizado para gerenciar os AVAs da Ufes, e é muito utilizado também em diversas instituições no país, sendo o Brasil o terceiro maior utilizador da plataforma, conforme pode ser visto na Tabela 1.

O Moodle é composto por módulos instaláveis, configuráveis e estendíveis. Assim, é possível desenvolver *plugins* e componentes que adicionam novas funcionalidades a plataforma, como é o caso do que está sendo desenvolvido neste trabalho.

O que propomos neste projeto é, portanto, a criação de um novo *plugin* para o Moodle que permita ao usuário construir, armazenar e recuperar seus mapas dentro desta plataforma. Além disto, propomos uma arquitetura para integração do Moodle com o CMPaaS de modo a permitir que mapas produzidos dentro da Plataforma Moodle sejam sincronizados com a Nuvem oferecida pelo projeto CMPaaS.

Tabela 1 – Os 10 países que mais utilizam o Moodle.

País	Sites Registrados
Estados Unidos	10.131
Espanha	7.067
Brasil	4.401
Reino Unido	3.486
México	3.464
Alemanha	2.444
Itália	2.414
Austrália	2.324
Colômbia	2.264
Rússia	1.993

Fonte: Moodle (2016b)

3 Integração de Sistemas

Nos últimos anos muitas organizações têm utilizado aplicações para dar suporte aos seus negócios e por esse motivo sistemas de informação têm se tornado um de seus pilares de funcionamento (MARTINS, 2006). O aumento do uso da tecnologia de informação juntamente com a busca pelo controle e flexibilização de informações fez crescer a demanda por soluções que viabilizassem a partilha de informações e de funcionalidades de aplicações (EDWARDS; NEWING, 2000). Aplicações estas, que em grande parte dos casos, foram desenvolvidas de forma customizada e sem a capacidade de integração com outros sistemas. Neste contexto, as organizações seguiram diversas abordagens para a integração dos seus sistemas de informação, criando então soluções sem nenhuma norma técnica, devido a inexistência das mesmas.

O crescimento da necessidade de soluções de integração de sistemas levou ao desenvolvimento de normas técnicas e protocolos direcionados a interoperabilidade de sistemas, surgindo então um cenário desafiador onde as instituições precisam escolher entre diversas normas e técnicas de integração de sistemas, muitas delas incompatíveis entre si (MARTINS, 2006).

Dentre estas tecnologias se destacam Simple Object Access Protocol (SOAP) e Representational State Transfer (REST), que obtiveram ampla aceitação no mercado pois conseguem se aproveitar do protocolo Hypertext Transfer Protocol (HTTP). Com o REST tendo destaque por ser um modelo de arquitetura simples e robusto, que é capaz de atenter requisitos de integração complexos e com pouco acoplamento (LUSA; KRAEMER, 2016).

3.1 HTTP

O HTTP é o protocolo base de comunicação de aplicações Web e vem sendo utilizado na World-Wide Web desde 1990. Na sua versão inicial ele se tratava de um protocolo simples, utilizado para transmitir apenas dados brutos, mas foi aprimorado de forma a permitir a transmissão de mensagens contendo metadados sobre a informação transmitida, e também modificadores para semânticas de requisições e respostas. Assim, hoje ele é o protocolo padrão dos navegadores Web, sendo utilizado por diversas aplicações na Internet (FIELDING et al., 1999) .

No HTTP a comunicação é realizada através de requisições e respostas entre clientes e servidores. Um cliente, que pode ser um navegador web, requisita um recurso a um servidor enviando uma mensagem contendo um cabeçalho. Esta mensagem deve ser enviada para uma URI e o servidor então responde com um recurso ou com um outro cabeçalho.

A Figura 4 é uma captura de tela que mostra o cabeçalho de uma requisição http. Nela é possível observar a URI, no caso /api/users/, e o metódo da requisição, que no exemplo é GET. Também é possível observar o campo authorization, que é uma chave de autorização usada na autenticação do usuário que realizou a requisição. O servidor responde a esta requisição como um outro cabeçalho e também um conteúdo, que no caso é uma lista de usuários.

Figura 4 – Cabeçalho de uma requisição HTTP

```
GET /api/users/ HTTP/1.1
Host: localhost:8000
Connection: keep-alive
Accept: */*
Origin: null
authorization: Bearer x00KyutlEFNTBQh4yzKgE8ndBuVVCS
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate, br
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
```

Fonte: Elaborado pelo autor

Figura 5 – Cabeçalho de uma resposta HTTP

```
HTTP/1.0 200 OK
Date: Fri, 06 Oct 2017 18:11:32 GMT
Server: WSGIServer/0.2 CPython/3.5.2
Content-Type: application/json
Allow: GET, OPTIONS
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: *
Vary: Accept
```

Fonte: Elaborado pelo autor

Um cabeçalho de resposta HTTP pode ser visto na Figura 5. Este cabeçalho possui um código que serve para identificar se a requisição foi concluida com sucesso. No cabeçalho deste exemplo, o código de resposta 200 indica que a requisição foi concluida com sucesso. Já um código 401 indicaria que URI acessada exige autenticação do usuário. O campo *date* informa a data e hora em que a mensagem foi enviada. Já o campo *server* contém o nome do servidor que respondeu a requisição. O cabeçalho informa em *Content-type* o tipo de conteúdo da mensamge e em *Allow* os métodos HTTP que são aceitos.

O HTTP é um protocolo stateless, ou seja, sem estado. Isto significa que ele não guarda informações entre requisiçoes diferentes. Caso seja necessário armazenar dados é preciso utilizar alguma outra tecnologia em conjunto com o HTTP, como cookies ou variáveis de sessão.

O protocolo possui métodos para identificar o tipo de requisição que está sendo realizada. Estes métodos são GET, POST, DELETE, PUT, HEAD, OPTIONS, TRACE e CONNECT.

Neste trabalho os métodos utilizados são os que explicarei abaixo:

- GET: utilizado para requisitar informações sobre um recurso.
- POST: tem como funcionalidade a criação de um novo recurso ou o processamento de informações.
- PUT: é utilizado para atualizar um recurso já existente ou para criar um novo.
- DELETE: apaga um recurso.

3.2 REST

REST é um estilo de arquitetura que representa a abstração da interações de aplicações web. É um conjunto de restrições de arquitetura que busca minimizar o tempo de resposta na comunicação entre aplicações e ao mesmo tempo aumentar a independência na implementação de componentes. O REST foi criado por Roy T. Fielding (2000) em sua tese de doutorado e faz uso de tecnologias amplamente utilizadas, como o protocolo HTTP, para orientar a criação de aplicações baseadas nos fundamentos da Web. A arquitetura REST descreve interfaces que utilizam o HTTP para transmissão de dados sem a utilização de mensagens adicionais como ocorre com SOAP (MORO; DORNELES; REBONATTO, 2009).

Apesar de não ser obrigatório o uso do protocolo HTTP, ele é o único protocolo conhecido que é totalmente compatível com a tecnologia. Sendo o REST orientado às boas práticas de uso do HTTP, como o uso correto de seus métodos, a criação de URL's e códigos de resposta padronizados e o uso adequado de cabeçalhos. Como o HTTP é um dos protocolos mais utilizados na web a adoção do REST pela comunidade de desenvolvedores foi grande e rápida.

Ao se navegar na web todo documento acessível é chamado de recurso e isto não é diferente no REST. Este estilo de arquitetura é orientado a recursos, que são os conjuntos de dados que estarão disponível em uma aplicação web que implementa o REST. Cada recurso necessita de uma identificação única ou URI, que irá nomea-lo e fornecer um caminho para acessa-lo.

A interação com as URIs é realizada através dos métodos HTTP. Esta interação é guiada por uma regra na qual as URIs são substantivos e os métodos são verbos. E portanto os metodos sao aqueles que realizam alteraçoes nos recusrsos identificados pelas

URIs. O método GET, por exemplo, é utilizado para se obter dados de um recurso, já o método DELETE é tem como funcionalidade apagar um recurso.

O REST também padroniza também o uso dos códigos de respostas do HTTP. Estes códigos são divididos em cinco grupos, e cada grupo possui um significado. Códigos 1xx são informacionais, códigos 2xx indicam sucesso, codigos 3xx informam um redirecionamento, 4xx indicam erros do cliente e 5xx erros de servidor.

3.2.1 Princípios da arquitetura REST

Em sua tese de doutorado Fielding (2000) estabeleceu príncipios que devem ser repeitados na implementação da arquitetura REST. E com isto as seguintes características devem ser respeitadas:

- **Arquitetura cliente-servidor:** é um caracterísica comum em serviços Web. O cliente inicia a comunicação requisitando um serviço ou recurso a um servidor. O servidor escuta e processa a requisição feita pelo cliente e então o envia uma resposta. Esta resposta pode ser o serviço ou recurso que o cliente requisitou, ou então pode ser uma rejeição à requisição realizada pelo cliente.
- **Stateless:** a comunicação entre cliente e o servidor é sem estado. Isto significa que a interação entre ambos é feita sem que seja armazenado qualquer tipo de estado no servidor. Ou seja, cada requisição é independente uma da outra e deve conter toda informação necessária para a comunicação.
- ***Cacheable*:** as respostas enviadas pelo servidor podem ser armazenadas em um cache no cliente. Assim, quando uma requisição semelhante é realizada o dado é acessado no cache do cliente, não sendo necessário receber uma nova resposta do servidor.
- **Interface Uniforme:** a interação entre o cliente e o servidor é realizada com a utilização de uma interface padronizada. Esta inferface deve identificar os recursos e manipula-los através de representações. Neste sentido o HTTP é o protocolo mais indicado para ser utilizado com o REST, pois possui uma interface uniforme com métodos bem definidos para realizar operações básicas.
- **Multicamada:** a arquitetura REST preve a divisão das funcionalidades de um sistema em camadas. Cada uma delas tendo a possibilidade de se comunicar com a camada adjacente.
- ***Code On Demand*:** esta característica que permite obter um código do servidor que será executa diretamente no lado do cliente. Um exemplo de uso é um código JavaScript que é baixado do servidor e executado por um navegador Web no cliente.

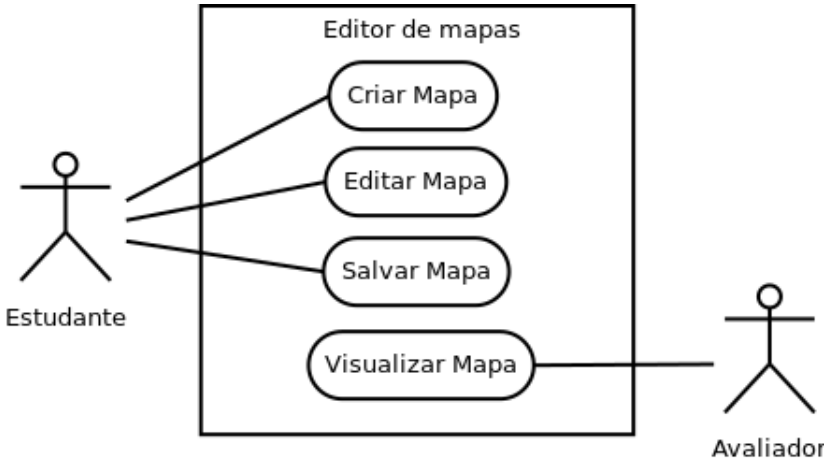
4 Projeto

4.1 Arquitetura

O *plugin* proposto consite de três partes: configurações, sumário de envio e formulário de envio. Nas configurações é onde será definido o comportamento do *plugin*. O sumário de envio será visto pelo estudante e avaliadores na tela inicial da tarefam, e contém um resumo da tarefa. Já o formulário de envio, que é onde o estudante responde a tarefa.

Para o desenvolvimento do *plugin* foi elaborado um diagrama de casos de uso, que pode ser visto na Figura 6.

Figura 6 – Casos de Uso



Fonte: Elaborado pelo autor

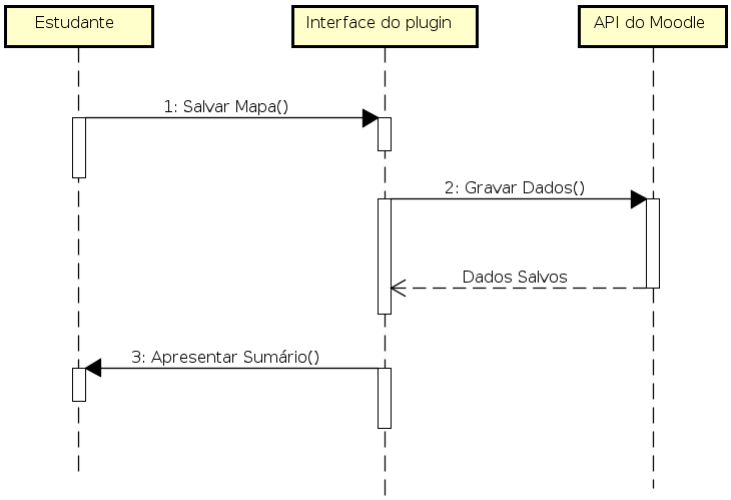
O *plugin* oferece as seguintes funcionalidades:

1. Criar Mapa: permite a criação de um novo mapa conceitual. O novo mapa é também criado nos repositórios do CMPaaS
2. Salvar Mapa: persiste as alterações do mapa criado nos repositórios do CMPaaS.
3. edição de um mapa previamente salvo no repositório do CMPaaS. Esta funcionalidade é representada pelo caso de uso Editar Mapa.
4. visualização de um mapa previamente salvo no repositório do CMPaaS. Esta funcionalidade é representada pelo caso de uso Visualizar Mapa.

Os atores que irão utilizar o *plugin* são o Estudante, que irá criar e editar mapas conceituais que serão submetidos a avaliação, e um Avaliador, que irá visualizar os mapas criados pelos estudantes e avaliá-lo. Os atores e casos de uso são representados na Figura 6.

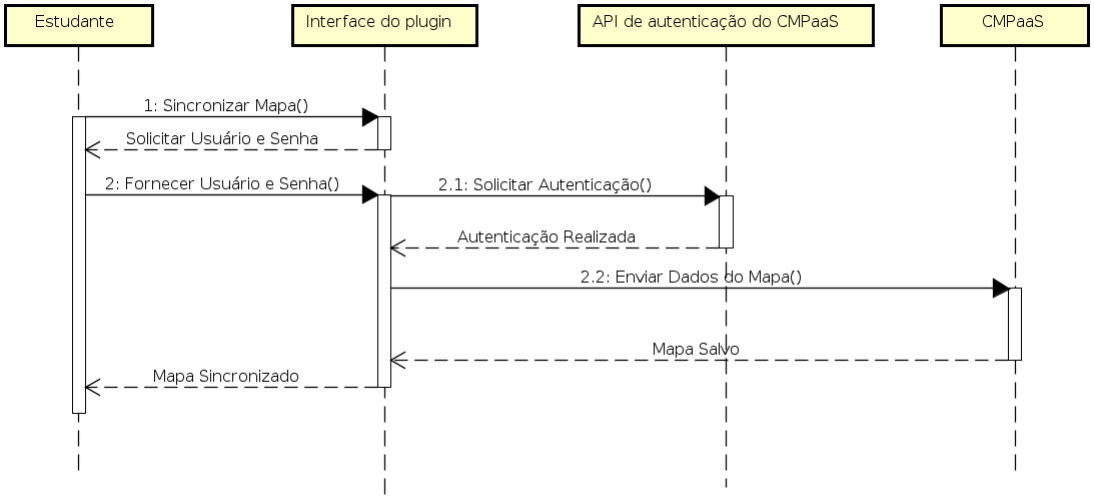
O caso de uso Salvar Mapa é apresentado na Figura 7. Quando o estudante solicita que o mapa seja salvo a interface do *plugin* chama a rotina da API do Moodle responsável por enviar os dados do formulário para serem armazenados. Após esta rotina ser executada e o mapa ser armazenado um sumário do envio é apresentado ao estudante.

Figura 7 – Diagrama de Sequencia do Caso de Uso Salvar Mapa



Fonte: Elaborado pelo autor

Figura 8 – Diagrama de Sequencia do Caso de Uso Sincronizar Mapa



Fonte: Elaborado pelo autor

O caso de uso Sincronizar Mapa é apresentado na Figura 8. É este caso de uso que realiza a integração do *plugin* com a plataforma CMPaaS. Quando o estudante executa a ação de sincronização do mapa a interface do *plugin* chama uma rotina para que seja feita autenticação no portal. Depois que o estudante é autenticado no CMPaaS os dados do mapa são enviados para serem armazenados na plataforma.

4.2 Tecnologias utilizadas

4.2.1 JavaScript

O JavaScript é uma linguagem orientada a objeto criada em 1995 por Brendan Eich que foi idealizada para permitir que pessoas que não são programadoras pudessem estender as funcionalidades de sites de Internet (RICHARDS et al., 2010). Ela é uma linguagem interpretada, e apesar de ter sido criada para desenvolvimento web, ela é uma linguagem de proposito geral, e pode ser usada para o desenvolvimento de qualquer tipo de aplicação (FLANAGAN, 2006).

Ela foi escolhida para ser utilizada neste projeto pois o editor de mapas disponível no CMPaaS foi desenvolvido com esta tecnologia e, além disso, trata-se de uma linguagem compatível com todos os navegadores modernos.

4.2.2 Json

O *JavaScript Object Notation* (Json) é uma linguagem em formato de texto cuja função é a serialização de dados estruturados. A serialização é o processo de transformar objetos em um fluxo de *bytes* para ser armazenado em um banco de dados ou disco. Ele pode ser utilizado para representar tipos primitivos, como cadeia de caracteres e números, ou tipos estruturados, como objetos e vetores (CROCKFORD, 2006). Ele é utilizado para intercambio de dados entre os serviços da plataforma CMPaaS.

4.2.3 PHP

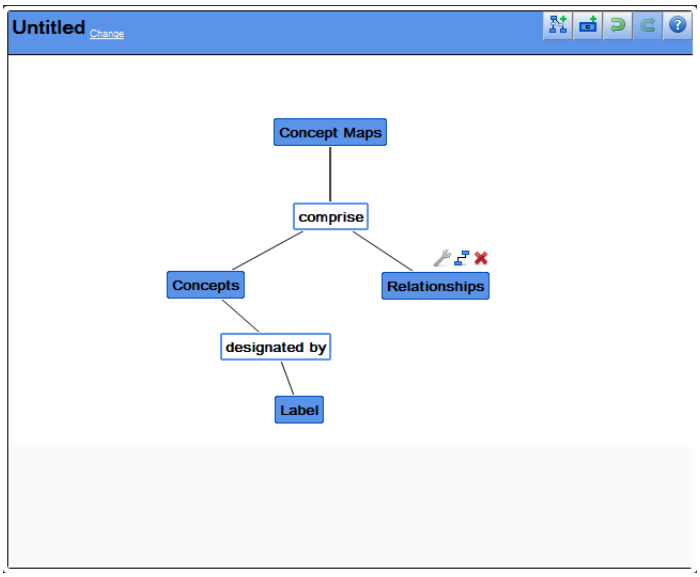
PHP é um acrônimo recursivo para *PHP: Hypertext Preprocessor*, originalmente ele foi criado como uma linguagem de script estruturada cuja finalidade era o desenvolvimento de aplicações com a funcionalidade de geração de HTML dinâmico. Com o passar do tempo a linguagem evoluiu e passou a oferecer recursos para o desenvolvimento orientado a objeto (MINETTO, 2007).

A plataforma Moodle e seus *plugins* são desenvolvidos em PHP, por isto esta linguagem foi utilizada neste projeto.

4.3 Trabalhos correlatos

Existe um *plugin* semelhante ao desenvolvido neste trabalho no repositório do Moodle. Este *plugin* possibilita que estudantes respondam questionários com a elaboração de um mapa conceitual. As principais funcionalidades dele são a possibilidade de criar um número ilimitado de conceitos e relacionamentos, compatibilidade com padrões de internacionalização e persistência de mapas com a utilização de XML. Infelizmente a última versão dele foi lançada em fevereiro de 2015 e não é compatível com as versões mais atuais do Moodle. A Figura 9 mostra um mapa criado neste *plugin*.

Figura 9 – Um plugin de elaboração de mapas conceituais



Fonte: Moodle (2016a)

5 Desenvolvimento

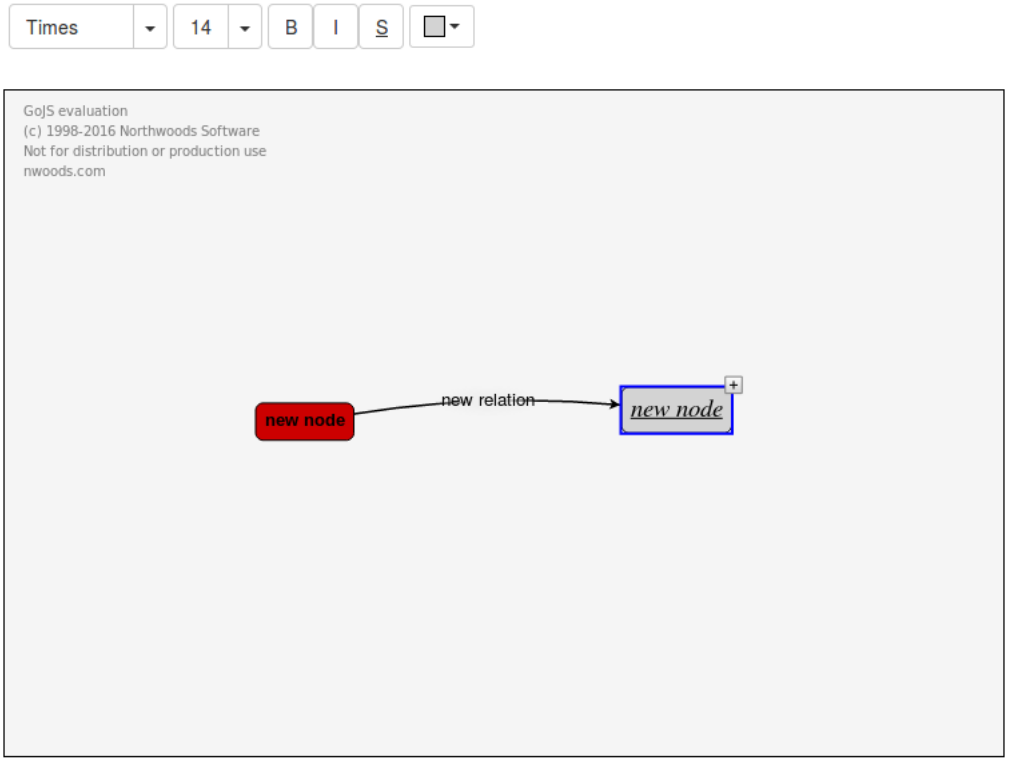
5.1 Serviço de Edição de Mapas

O serviço de edição de mapas foi implementado com a utilização do editor de mapas conceituais disponível no CMPaaS. Ele foi desenvolvido utilizando GoJS, que é uma biblioteca em JavaScript para criação de diagramas para navegadores *Web*.

Este editor salva os mapas no formato Json, possibilitando que outras aplicações utilizem os dados que ele produz. Foi concebido desta forma para acompanhar a proposta do CMPaaS de permitir a interoperabilidade dos serviços oferecidos.

A primeira parte do trabalho foi realizar um aprimoramento deste editor, adicionando funções para a formatação dos mapas criados. Não existia por exemplo, a possibilidade de alterar a fonte da letra dos conceitos ou a cor de fundo dos nós. Uma barra de ferramenta foi adicionada ao editor original para que funções de formatação estivessem disponíveis.

Figura 10 – Editor com barra de formatação



Fonte: Elaborada pelo autor

5.1.1 Implementação das funções de formatação

O editor de mapas é dividido em dois componentes, um Model e um Diagram. O Model é o que contém os dados do mapa que está sendo criado, nele os nós e links são descritos por vetores de objetos JavaScript. Já o Diagram é usado para visualizar os dados contidos no modelo.

Na Figura 11 temos um trecho de código que gera um diagrama simples em GoJS. Nele é criado um Model que possui um vetor com três objetos JavaScript, nomeados Alpha, Beta e Gamma. Este vetor de objetos da classe Node representa três nós que serão apresentados no diagrama. O Model nomeado myModel é então adicionado ao Diagram chamado myDiagram. Este trecho de código gera o diagrama ilustrado na Figura 12.

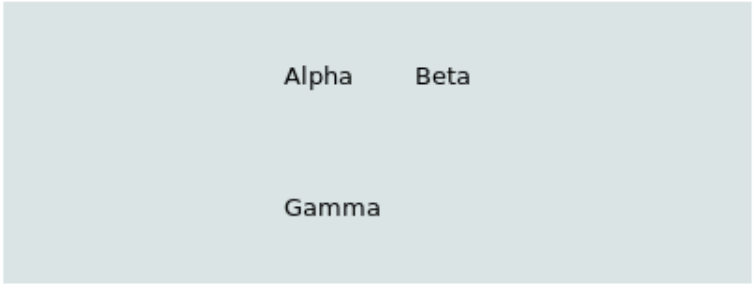
Figura 11 – Componentes do GoJS

```
myDiagram =
    $$(go.Diagram, "myDiagram",
      { initialContentAlignment: go.Spot.Center,
        "toolManager.mouseWheelBehavior": go.ToolManager.WheelZoom,
        "undoManager.isEnabled": true,
        "clickCreatingTool.archetypeNodeData": { text: "new node" }
      });

    var myModel = $(go.Model);
    myModel.nodeDataArray = [
      { key: "Alpha" },
      { key: "Beta" },
      { key: "Gamma" }
    ];
    myDiagram.model = myModel;
```

Fonte: Elaborada pelo autor

Figura 12 – Um diagrama gerado em GoJS



Fonte: Elaborada pelo autor

Cada nó do mapa criado pelo editor é representado por um objeto da classe Node, que por sua vez, é composto por blocos que determinam a sua aparência. Os blocos que o editor utiliza são o Shape e o TextBlock. A classe Shape é utilizada para mostrar uma forma geométrica colorida. Já a classe TextBlock tem como função mostrar um texto. Ambas as classes possuem diversas propriedades que servem para determinar a sua aparência e o seu comportamento no diagrama.

Os trechos de código abaixo são usados para criar um Shape e um TextBlock. A forma geométrica desenhada é um retângulo com largura de 40 pixels, altura de 60 pixels, com margem de 4 pixels e preenchimento na cor vermelha. Já o TextBlock criado é um texto "a Text Block" de cor vermelha.

Figura 13 – Código dos elementos Shape e TextBlock

```
$(go.Shape, "Rectangle", { width: 40, height: 60, margin: 4, fill: "red" })

$(go.TextBlock, { text: "a Text Block", stroke: "red" })
```

Fonte: Elaborada pelo autor

As novas funcionalidades de formatação adicionadas ao editor modificam as propriedades dos objetos das classes Shape e TextBlock, alterando assim sua aparência. Foram criados seis botões de formatação com as funções de alterar o tipo de fonte, aumentar o tamanho da fonte, alterar o estilo do texto para negrito, itálico e sublinhado, e para alterar a cor de preenchimento dos nós. Os botões de formatação de fonte alteram a propriedade *font* da TextBlock, que deve ser uma *string Cascading Style Sheets* (CSS). Já o botão que altera a cor do nó modifica a propriedade *fill* da Shape.

As funcionalidades de formatação foram implementadas em JavaScript no arquivo toolbar.js. Elas foram desenvolvidas de forma semelhante, utilizando eventos disparados pelos botões da barra de ferramenta.

Para realizar alteração de estilo da fonte e de cor do nó foram implementadas funções que modificam os atributos dos nós. Para cada botão da barra de ferramenta foi criada uma rotina que é executada quando o evento de clique é acionado.

A Figura 14 mostra a rotina que altera a cor de preenchimento de um nó. Inicialmente é criada uma espera para o evento que será disparado pelo botão, no caso o evento *input*. Quando ele ocorre é chamada a função que altera a propriedade *fill* do objeto Shape de todos os nós selecionados, mudando assim a cor dos mesmos. As funcionalidades de estilo da fonte são implementadas com a mesma rotina, o que muda é a propriedade alterada, que passa a ser a *font* do objeto TextBlock de todos os nós selecionados.

Figura 14 – Rotina que altera a cor de preenchimento de um nó

```
colorButton = document.getElementById("color");
colorButton.addEventListener("input", function() {
  myDiagram.startTransaction("change color");
  var it = myDiagram.selection.iterator;
  while (it.next()) {
    var node = it.value;
    var shape = node.findObject("SHAPE");
    if (shape !== null) {
      shape.fill = colorButton.value;
    }
  }
  myDiagram.commitTransaction("change color");
});
```

Fonte: Elaborada pelo autor

5.2 Implementação no Moodle

Para implementar o serviço de edição de mapas no Moodle foi utilizado um *plugin* já existente na plataforma como modelo. Conforme visto no Capítulo 4 o *plugin* escolhido para a implementação da funcionalidade de edição de mapas foi o de envio de tarefa. Todos os *plugins* deste tipo possuem uma estrutura de arquivo padrão, que será detalhada abaixo.

- version.php: este arquivo contém informações sobre a versão do *plugin*, é utilizado para que o Moodle instale e atualize o *plugins* corretamente.
- settings.php: este arquivo permite que se adicione opções personalizadas para a página configuração do *plugins*.
- lang/en/submission_nomedoplugin.php: este é o arquivo de linguagem, ele é usado para internacionalização do *plugin*.
- db/access.php: e utilizada para adicionar capacidades adicionais ao *plugin*. Este arquivo é opcional, não sendo necessário se o *plugin* não tiver capacidades adicionais.
- db/upgrade.php: este arquivo define a rotina de atualização do *plugin*.
- db/install.xml: este arquivo define as tabelas de banco de dados que o *plugin* vai utilizar.
- db/install.php: contem o código de instalação do *plugin*.
- db/locallib.php: é o arquivo mais importante, é ele que define todas as funcionalidades do *plugin*.

5.2.1 Desenvolvimento do plugin

O *plugin* utilizado como base para o desenvolvimento deste trabalho foi o de envio de texto online, nele o estudante tem a possibilidade responder uma tarefa por meio de um texto escrito diretamente na plataforma. Este projeto foi desenvolvido aplicando engenharia reversa a este *plugin*, buscando entender a sua funcionalidade e assim descobrir como modificá-lo para transformá-lo em um editor de mapas.

O *plugin* de envio de texto online pode ser dividido em duas partes, uma é o editor de textos onde o estudante realiza a sua tarefa e a outra e o sumário da atividade onde tanto o estudante quanto o avaliador tem acesso ao conteúdo da tarefa enviada. O *plugin* desenvolvido neste trabalho mantém esta estrutura, alterando o editor de textos por um de mapas conceituais e alterando o sumário de forma que o conteúdo apresentado nele passa a ser um Json do mapa ao invés de um texto.

As alterações feitas no *plugin* de envio de texto online se concentraram no arquivo locallib.php, que é o que determina o seu comportamento e funcionalidades. Dentro deste arquivo, as mudanças ocorreram nas funções get_form_elements(), que constrói o formulário de envio de tarefa, e no arquivo save(), que realiza a submissão do conteúdo criado pelo estudante.

A primeira parte do trabalho foi substituir o formulário de envio de texto pelo editor de mapas conceituais. Esta modificação foi realizada na função get_form_elements(). O código que realizava a inserção do editor de texto foi substituído por um outro, que insere um iframe contendo o editor de mapas do CMPaaS.

Assim, o *plugin* já apresentava o editor de mapas, porém ainda era necessário salvar o mapa criado no banco de dados do Moodle. Para realizar isto, Json do editor contido no iframe precisava ser salvo em algum campo de formulário. Como solução, foi criado uma entrada de dados, oculta na página da tarefa, responsável por receber o Json do mapa criado no editor.

Quando o estudante aciona o botão de submissão da tarefa, o Json respectivo ao mapa que criado é salvo em um campo de formulário e então armazenado no banco de dados do Moodle.

Para finalizar, foi necessário criar uma forma de o editor de mapas carregar o mapa salvo pelo estudante, para caso ele tivesse interesse em editar um mapa já armazenado. A solução para isto foi realizar o caminho inverso que foi feito ao salvar o mapa criado. Ao carregar o formulário de edição de mapas o Json armazenado no banco de dados e carregado em um campo oculto e um código JavaScript se encarrega se obter o conteúdo dele e carregar no mapa.

A primeira parte do desenvolvimento do *plugin* foi concluída, o estudante conseguia criar um mapa no editor, salvar o seu conteúdo na plataforma e editar novamente caso

fosse necessário. Ainda havia a necessidade de permitir que o avaliador tivesse acesso ao conteúdo gerado pelo aluno. Além disso, falta a integração com o CMPaaS e utilização de seus recursos.

5.3 Integração com o CMPaaS

A integração com o CMPaaS é funcionalidade mais importante do *plugin*, visto que o objetivo do trabalho é aumentar a base de usuários e o conteúdo do CMPaaS.

O *plugin* utiliza a API Rest do CMPaaS para guardar mapas no repositório da plataforma e também para recuperar mapas que foram previamente armazenados nela pelo usuário.

5.3.1 Lista de mapas do usuário

Para obter a lista de mapas do usuário o *plugin* envia um request de método GET com URI /api/users/. A API então responde com um JSON com os dados do usuário e sua lista de mapas, conforme visto na Figura 15. Este JSON contém a URI do usuário, seu *username* e também a lista de URIs dos mapas que o usuário possui na plataforma.

Figura 15 – Json com dados do usuário


```
{
    "url": "http://localhost:8000/api/users/1/",
    "username": "israel",
    "maps": [
        "http://localhost:8000/api/maps/7/",
        "http://localhost:8000/api/maps/8/"
    ]
}
```


Fonte: Elaborada pelo autor

A partir deste JSON o plugin elabora novas requests que serão enviadas para a API do CMPaaS, para que sejam obtidas as informações dos mapas do usuário. Para cada URI de mapa que o usuário possui no repositório é feita uma nova request GET para URI com formato /api/maps/id de forma a ser obtida uma resposta contendo um JSON com metadados de cada mapa. Este JSON de resposta é representada na Figura 16.

Os metadados do maps contidos neste JSON são a sua URI, o autor, o título, a questão e a sua descrição. Com estas informações é criada a interface que mostra a listagem de mapas do usuário, conforme Figura 17. Nesta interface o usuário pode visualizar as versões de um mapa listado ou apagar este mapa.

Figura 16 – Json com os metadados de uma mapa


```
{
    "url": "http://localhost:8000/api/maps/7/",
    "author": "http://localhost:8000/api/users/1/",
    "title": "Mapa 1",
    "question": "Question1",
    "description": "Descrição 1"
}
```


Fonte: Elaborada pelo autor

Figura 17 – Lista de mapas de um usuário

Fonte: Elaborada pelo autor

5.3.2 Lista de versões de um mapa

O repositório de mapas do CMPaaS possui a funcionalidade de controle de versão, assim sempre que um mapa é editado uma nova versão é criada e a antiga é mantida. O *plugin* utiliza esta funcionalidade de versionamento, permitindo que o usuário visualize as versões de um mapa e as manipule. Esta interface pode ser vista na Figura 18.

Figura 18 – Versões de um mapa

Fonte: Elaborada pelo autor

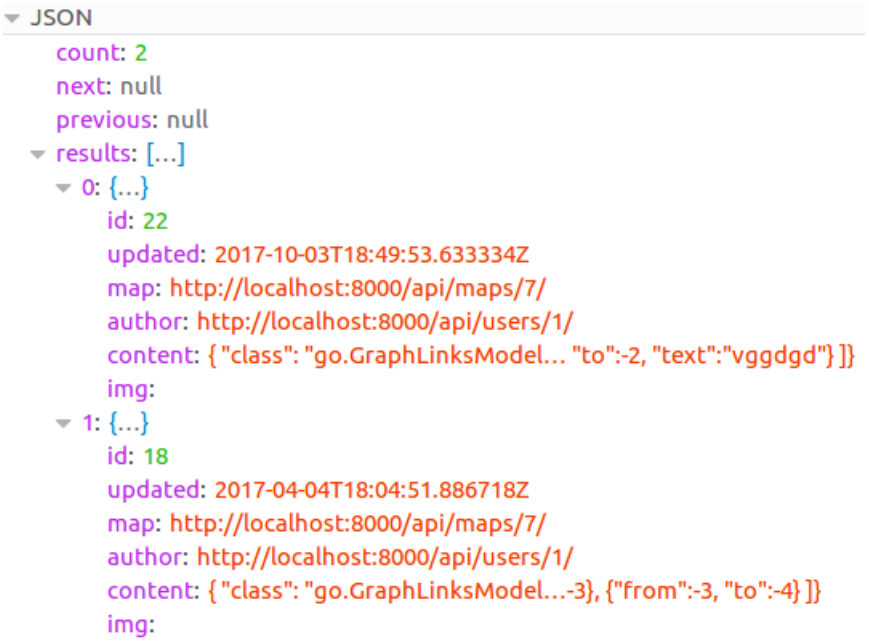
O conteúdo desta interface é obtido através de uma request GET enviada para a URI /api/maps/id/version. A API responde a esta request com um JSON contendo as versões do mapa solicitado, além dos metadados e conteúdo do mapa conceitual de cada versão. A resposta obtida pode ser vista na Figura 19.

Este Json contém a lista de versões de um mapa e o conteúdo de cada uma. Os dados de cada versão são seu identificador, a data e horário de criação, a URI do mapa a qual ela pertence, seu autor e o conteúdo da versão, que é a representação do diagrama do mapa conceitual criado pelo usuário. Este conteúdo da versão é o Json que será utilizado pelo editor de mapas para desenhar o diagrama criado pelo usuário.

5.3.3 O editor de mapas

Quando o usuário aciona o botão de editar uma versão de um mapa a função loadMapVersionContent() é chamada. Esta função armazena no localStorage do browser a

Figura 19 – Json com as versões de um mapa



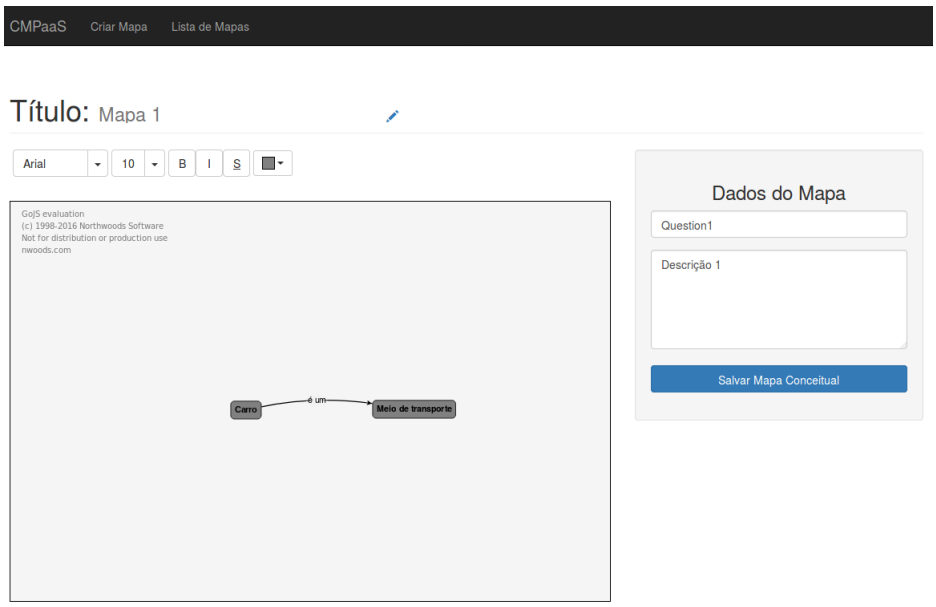
Fonte: Elaborada pelo autor

URI do mapa e o Json do conteúdo que representa o diagrama do mapa conceitual. Após os dados serem salvos no localStorage a função carrega a interface do editor de mapas. A URI do mapa então é utilizada em uma request GET, ilustrado na Figura 16, para carregar metadados na interface do editor e o diagrama é finalmente carregado e plotado para o usuário editar. O resultado final pode ser visto na Figura 20.

5.3.4 Exclusão de mapas

O *plugin* também possui a funcionalidade de exclusão de um mapa e as suas versões. As exclusões são realizadas através do envio de requests do tipo DELETE para a API do CMPaaS. Uma versão é apagada com o envio de uma request DELETE para a URI api/maps/id/versions/id/. Já para excluir um mapa é necessário enviar a mesma request para a URI api/maps/id/versions/id/.

Figura 20 – Editor de mapas



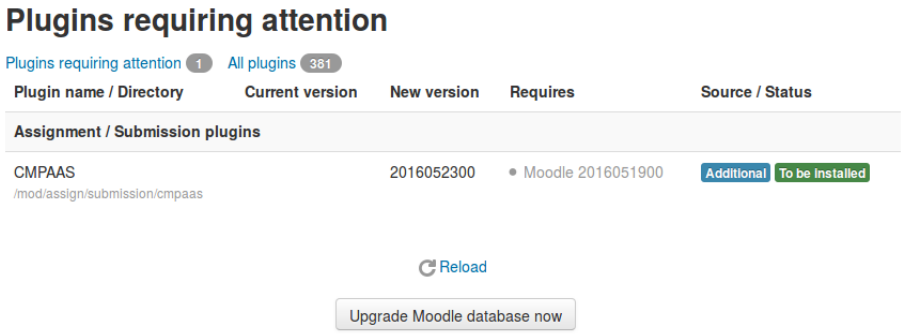
Fonte: Elaborada pelo autor

6 Instalação e Testes

6.1 Instalação

A pasta contendo os arquivos do *plugin* deve ser copiada para o diretório do Moodle mod/assign/submission/. Após copiar os arquivos para este local é necessário acessar o Moodle com um usuário com perfil de administrador. Assim que efetuar login irá aparecer uma mensagem de instalação de *plugin*, conforme a Figura 21.

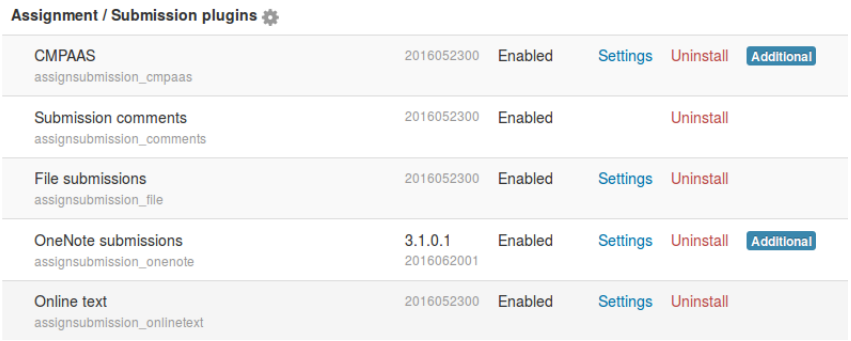
Figura 21 – Configurando a tarefa para utilizar o *plugin*



Fonte: Elaborada pelo autor

Ao clicar no botão de atualização do banco de dados do Moodle o *plugin* será instalado e poderá ser visto na lista de *plugins*, conforme Figura 22.

Figura 22 – Lista de *plugins* instalados

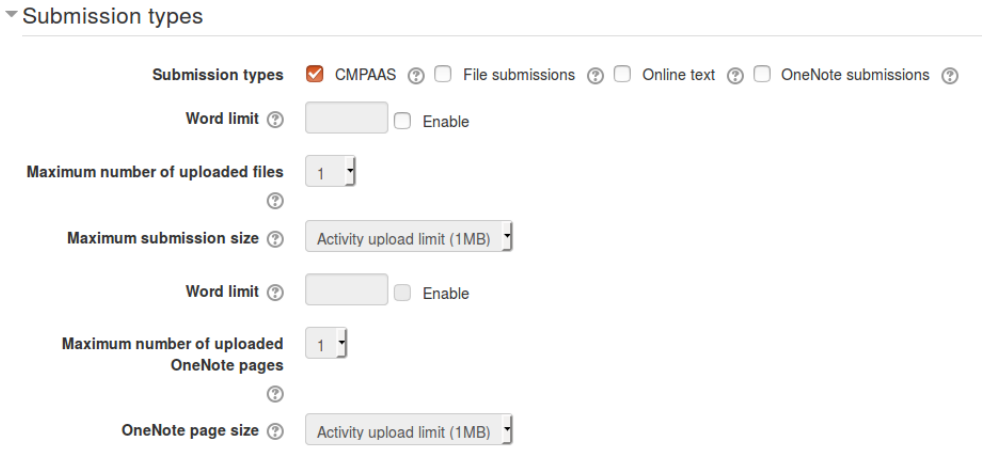


Fonte: Elaborada pelo autor

Depois do *plugin* ser instalado é necessário realizar a configuração da tarefa para utilizá-lo. Para que a tarefa utilize o *plugin* para submissão de dados é necessário selecioná-lo

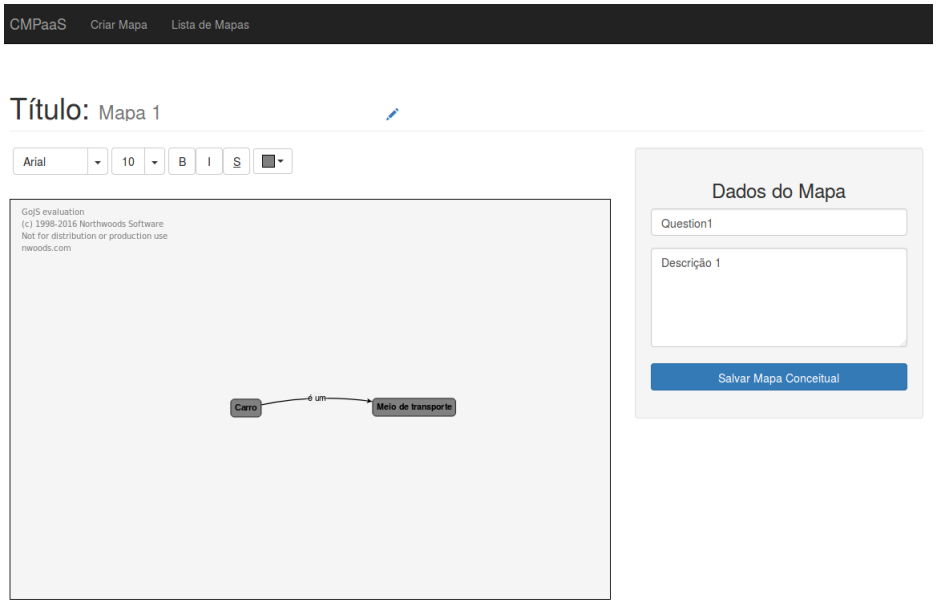
na tela de configuração da atividade, conforme ilustrado na Figura 23.

Figura 23 – Configurando a tarefa para utilizar o plugin



Fonte: Elaborada pelo autor

Figura 24 – Editor de mapas



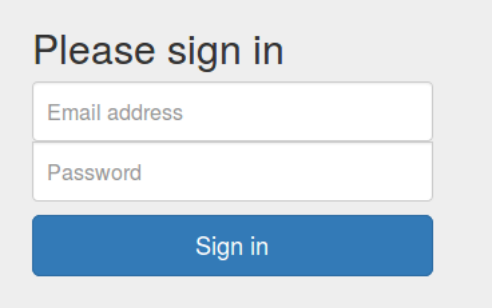
Fonte: Elaborada pelo autor

6.2 Como usar

A utilização do *plugin* é bem simples. Ao acessar uma tarefa o estudante deve clicar no botão de envio de tarefa e ao fazer isto será mostrada uma tela de login, onde o usuário deverar realizar autenticação no CMPaaS. Esta tela é mostrada na Figura 25. Após o login

ser realizado o usuário é redirecionado para a interface que contém a lista de mapas que ele possui armazenado no CMPaaS. Esta lista foi apresentada anteriormenta na Figura 17.

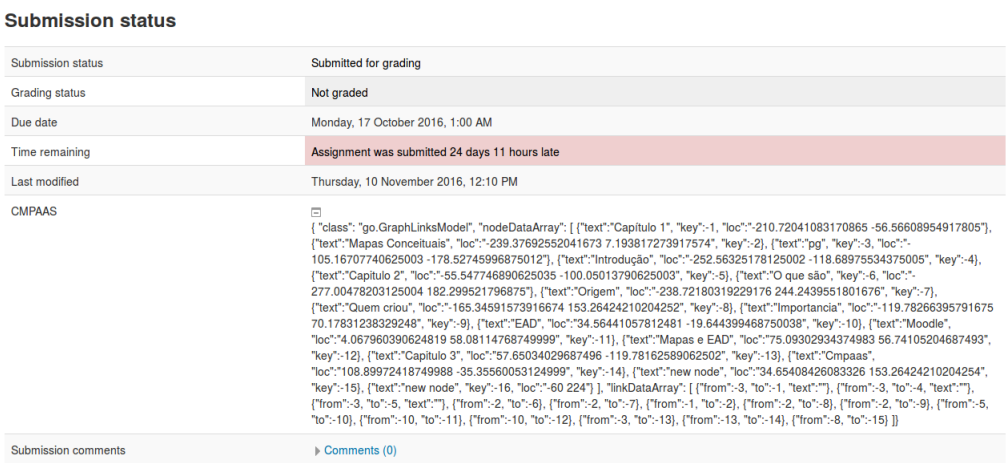
Figura 25 – Interface de login



Fonte: Elaborada pelo autor

O estudante então pode criar um novo mapa ou então editar mapas já existentes. Após criar ou editar o mapa o aluno deve clicar no botão "Salvar alterações" e o Json será salvo no banco de dados da plataforma e aparecerá no sumário, conforme Figura 26.

Figura 26 – Sumário da tarefa



Submission status

Submission status	Submitted for grading
Grading status	Not graded
Due date	Monday, 17 October 2016, 1:00 AM
Time remaining	Assignment was submitted 24 days 11 hours late
Last modified	Thursday, 10 November 2016, 12:10 PM
CMPAAS	{ "class": "go.GraphLinksModel", "nodeDataArray": [{"text":"Capítulo 1", "key":-1, "loc":"-210.72041083170865 -56.56608954917805"}, {"text":"Mapas Conceituais", "loc":"-239.37692552041673 7.193817273917574", "key":-2}, {"text":"pg", "key":-3, "loc":"-105.16707740625003 -178.52745996875012"}, {"text":"Introdução", "loc":"-252.56325178125002 -118.68975534375005", "key":-4}, {"text":"Capítulo 2", "loc":"-55.547746890625035 -100.05013790625003", "key":-5}, {"text":"O que são", "key":-6, "loc":"-277.00478203125004 182.2995217968875"}, {"text":"Origem", "loc":"-238.72180319229176 244.2439551801676", "key":-7}, {"text":"Quem criou", "loc":"-165.34591573916674 153.26424210204252", "key":-8}, {"text":"Importancia", "loc":"-119.78266395791675 70.17831238329248", "key":-9}, {"text":"EAD", "loc":"34.56441057812481 -19.644399468750038", "key":-10}, {"text":"Moodle", "loc":"4.067960390624819 58.08114768749999", "key":-11}, {"text":"Mapas e EAD", "loc":"75.09302934374983 56.74105204687493", "key":-12}, {"text":"Capítulo 3", "loc":"57.65034029687496 -119.78162589062502", "key":-13}, {"text":"Cmpaas", "loc":"108.89972418749988 -35.35560053124999", "key":-14}, {"text":"new node", "loc":"34.65408426083326 153.26424210204254", "key":-15}, {"text":"new node", "key":-16, "loc":"-60 224"}], "linkDataArray": [{"from":-3, "to":-1, "text":""}, {"from":-3, "to":-4, "text":""}, {"from":-3, "to":-5, "text":""}, {"from":-2, "to":-6}, {"from":-2, "to":-7}, {"from":-1, "to":-2}, {"from":-2, "to":-8}, {"from":-2, "to":-9}, {"from":-5, "to":-10}, {"from":-10, "to":-11}, {"from":-10, "to":-12}, {"from":-3, "to":-13}, {"from":-13, "to":-14}, {"from":-8, "to":-15}]}
Submission comments	Comments (0)

Fonte: Elaborada pelo autor

6.3 Provas de Conceitos

Nesta seção será apresentada a utilização de todas as funcionalidades oferecidas pelo *plugin*. Elas podem ser dividas em dois grupos: recursos para elaboração e edição dos mapas e persistência dos diagramas.

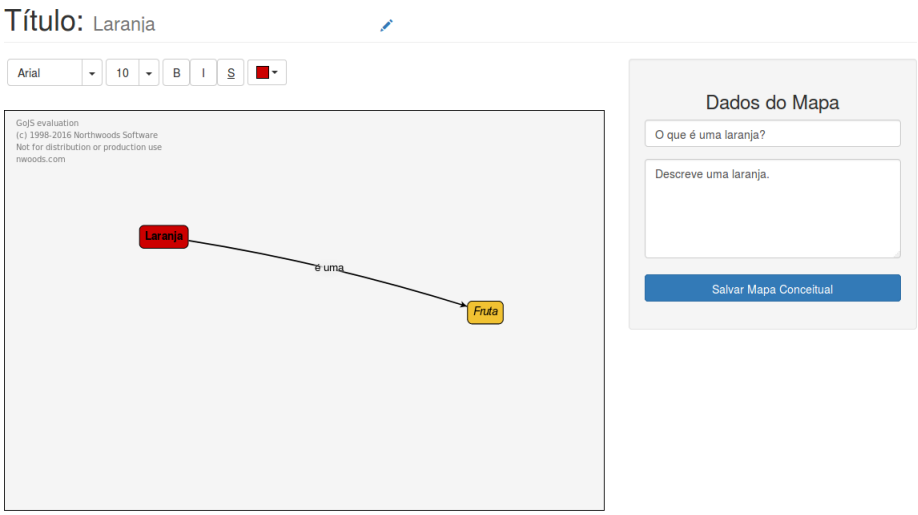
Os recursos para edição de mapas são as funções de formatação da aparência do diagrama disponíveis na barra de ferramentas. Elas permitem alterar a cor de preenchimento

dos nós e o estilo da fonte de cada conceito. A Figura 27 mostra um mapa no qual foi aplicado estas funções.

Já a persistência do diagrama pode ser dividida em duas etapas: a gravação do mapa no Moodle e a persistencia dos mapas no repositório do CMPaaS. A primeira ocorre quando o usuário clica no botão Salvar Alterações, que foi apresentado na Figura 20. Ao fazer isto, o Json so mapa que está sendo editado é salvo no banco de dados do Moodle.

Já a persistencia no CMPaaS ocorre quando o estudante utiliza o botão Salvar Mapa Conceitual, que pode ser visto na cor azul na Figura 27.

Figura 27 – Mapa que utiliza os recursos de formação



Fonte: Elaborada pelo autor

7 Considerações finais

7.1 Resultados

O objetivo deste trabalho foi a criação de uma ferramenta que permitisse a utilização de mapas conceituais em cursos gerenciados pela plataforma Moodle. Para realizar isto foi idealizado o desenvolvimento de um *plugin* de envio de tarefa que oferecesse a possibilidade do estudante responder uma atividade com a elaboração de um mapa conceitual. O *plugin* deveria prover também uma interface para um professor ou tutor visualizar o mapa criado pelo estudante e avaliar o mesmo. Adicionalmente ofereceria também uma interface de integração com o plataforma CMPaaS.

O protótipo desenvolvido neste trabalho alcançou o objetivo de oferecer uma ferramenta para o estudante responder uma tarefa com mapas conceituais. O *plugin* desenvolvido possui um editor de mapas que permite a elaboração de diagramas e modificação de sua aparência, além de realizar a persistência dos mapas criados.

A integração com o CMPaaS também foi implementada, assim o estudante tem a possibilidade de sincronizar o mapa criado no editor com o serviço de armazenamento na Nuvem oferecido pela plataforma. Já a interface para que o mapa seja visualizado para avaliação não foi implementada no protótipo, porém o *plugin* desenvolvido permite que este recurso seja adicionado em trabalhos futuros.

7.2 Conclusão

Neste trabalho foi apresentada a definição de mapas conceituais e a sua importância para a educação. Neste contexto foi discutido como eles podem ser utilizados no ensino e foi levantado quais são as ferramentas disponíveis para criação e manipulação de mapas. Neste levantamento foi identificada uma carência de aplicações computacionais que proporcionam suporte a criação de mapas conceituais e a necessidade do desenvolvimento de novas ferramentas com esta finalidade.

Foi pensado então no projeto de uma ferramenta que proporcionasse a aplicação de mapas conceituais na Educação a Distância. Escolhemos o Moodle como a plataforma onde seria realizada a implementação deste projeto. Ele foi escolhido por ser um gerenciador de cursos open-source amplamente utilizado em todo o mundo e por ser facilmente extensível através de inclusão de novos módulos.

Um *plugin* que permite a aplicação de mapas conceituais em tarefas dos cursos do Moodle foi então desenvolvido. Este *plugin* possibilita que os estudantes respondam

tarefas com a elaboração de mapas conceituais e oferece a possibilidade de sincronização do mapa na plataforma CMPaaS.

O *plugin* que foi criado neste trabalho trata-se de um protótipo, ou seja, não está totalmente funcional e carece de melhorias. No entanto serve como base para a criação de uma ferramenta que pode ser adicionada a lista de repositórios de *plugin* do Moodle, ampliando assim a possibilidade do uso de mapas conceituais no ensino.

7.3 Trabalhos futuros

O protótipo desenvolvido neste trabalho teve foco na interface para o estudante criar mapas conceituais, não possuindo o recurso de apresentar o mapa elaborado nele para um avaliador. Assim, uma proposta de trabalho futuro é a criação de uma forma de o avaliador visualizar o mapa submetido pelo estudante.

Ademais é sugerido o desenvolvimento de trabalhos que visam a melhoria da experiência do usuário que utiliza o editor de mapas. Visto que este trabalho não teve como foco a criação de uma interface otimizada.

Referências

ALMEIDA, M. E. B. de. Tecnologia e educação a distância: Abordagens e contribuições dos ambientes digitais e interativos de aprendizagem. *Revista Brasileira de Educação a Distância*, p. 6, 2010.

AUSUBEL, D. P. The psychology of meaningful verbal learning. Grune and Stratton, New York, 1963.

BRASIL. Decreto 5.622, de 19 de dezembro de 2005. Regulamenta o artigo 80 da Lei no 9.394, de 20 de dezembro de 1996, que estabelece as diretrizes e bases da educação nacional. 2005. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2005/decreto/d5622.htm>. Acesso em: 08/11/2016.

CHAVES, E. Tecnologia na educação, ensino a distância, e aprendizagem mediada pela tecnologia: conceituação básica. *Revista Educação da Faculdade de Educação da Pontifícia Universidade Católica de Campinas*, v. 3, n. 7, 1999.

CROCKFORD, D. The application/json media type for javascript object notation (json). 2006.

DOUGIAMAS, M.; TAYLOR, P. Moodle: Using Learning Communities to Create an Open Source Course Management System. *Research.Moodle.Net*, p. 1–16, 2003. Disponível em: <http://research.moodle.net/pluginfile.php/15/mod_data/content/1121/Moodle-Dougiamas-2003.pdf>.

DUTRA, Í. M.; FAGUNDES, L. d. C.; CAÑAS, A. J. Um ambiente integrado para apoiar a avaliação da aprendizagem baseado em mapas conceituais. *XIII Simpósio Brasileiro de Informática na Educação - SBIE - Unisinos*, p. 49–59, 2002.

EDWARDS, P.; NEWING, R. *Application Integration for E-business: Strategies for Integrating Key Applications, Systems and Processes*. [S.l.]: Business Intelligence, 2000.

FIELDING, R. et al. *Hypertext transfer protocol–HTTP/1.1*. [S.l.], 1999.

FIELDING, R. T. Rest: architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*, 2000.

FLANAGAN, D. *JavaScript: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2006.

FRANCISCATO, F. T. et al. Avaliação dos Ambientes Virtuais de Aprendizagem Moodle, TelEduc e Tidia - Ae: um estudo comparativo. *Novas Tecnologias na Educação*, v. 6, n. 2, p. 5–10, 2008.

LUSA, D. A.; KRAEMER, R. P. Conhecendo o modelo arquitetural REST. *Engenharia de Software Magazine*, 2016. Disponível em: <http://www.devmedia.com.br/conhecendo-o-modelo-arquitetural-rest/28052>.

MARTINS, V. M. M. *Integração de Sistemas de Informação: Perspectivas, normas e abordagens*. Tese (Doutorado), 2006.

MINETTO, E. L. Frameworks para desenvolvimento em php. *São Paulo: Novatec*, 2007.

MOODLE. *Moodle Plugins*. 2016. Disponível em: <https://moodle.org/plugins/qtype_conceptmap>. Acesso em: 09/12/2016.

MOODLE. *Moodle Statistics*. 2016. Disponível em: <https://moodle.net/stats/>. Acesso em: 16/11/2016.

MORO, T. D.; DORNELES, C.; REBONATTO, M. T. Web services ws-* versus web services rest. *Revista de Iniciação Científica*, v. 11, n. 1, 2009.

NOVAK, J. D. Results and Implications of a 12-Year Longitudinal Study of Science Concept Learning. *Research in Science Education*, v. 35, n. 1, p. 23–40, mar 2005. ISSN 0157-244X. Disponível em: <http://link.springer.com/10.1007/s11165-004-3431-4>.

NOVAK, J. D.; CAÑAS, A. J. The origins of the concept mapping tool and the continuing evolution of the tool. *Information Visualization*, v. 5, n. 3, p. 175–184, 2006. ISSN 1473-8716. Disponível em: <http://ivi.sagepub.com/lookup/doi/10.1057/palgrave.ivs.9500126>.

PERIN, W.; CURY, D. Uma Plataforma de Serviços para Mapas Conceituais. n. Cbie, p. 230, 2016. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/6703>.

PERIN, W. d. A. *iMap – UM MECANISMO DE INFERÊNCIA PARA MAPAS CONCEITUAIS*. 120 p. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2014.

PERIN, W. de A. et al. Construindo mapas conceituais utilizando a abordagem imap. *Anais do Computer on the Beach*, p. 208–217, 2013.

RIBEIRO, E. et al. Um estudo sobre o incremento da coesão e coerência (expressividade) em mapas conceituais. *Anais do XVI SBIE: Simpósio Brasileiro de Informática na Educação*, p. 233–242, 2012.

RICHARDS, G. et al. An analysis of the dynamic behavior of JavaScript programs. *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation*, p. 1, 2010. ISSN 03621340.

YOO, J. S.; CHO, M.-H. Mining concept maps to understand students' learning. In: *Educational Data Mining 2012*. [S.l.: s.n.], 2012.