UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Curso de Graduação em Sistemas de Informação
Henrique de Lima Barroso

SISTEMA COLABORATIVO PARA CRIAÇÃO E GERENCIAMENTO DE METADADOS DE OBJETOS DE APRENDIZAGEM

Diamantina
2022

Henrique de Lima Barroso

SISTEMA COLABORATIVO PARA CRIAÇÃO E GERENCIAMENTO DE METADADOS DE OBJETOS DE APRENDIZAGEM

Trabalho de conclusão de curso apresentado ao curso de Sistemas de Informação como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM.

Orientador: Alessandro Vivas Andrade

Diamantina
2022

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Henrique de Lima Barroso

SISTEMA COLABORATIVO PARA CRIAÇÃO E GERENCIAMENTO DE METADADOS DE OBJETOS DE APRENDIZAGEM

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Data de aprovação: 25/02//2022

Profa. Dra. Luciana Pereira de Assis

Faculdade Ciências Exatas - UFVJM

Profa. Dra. Claudia Beatriz Berti

Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Luciana Pereira de Assis**, **servidor (a)**, em 25/02/2022, às 16:32, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Claudia Beatriz Berti**, **servidor (a)**, em 25/02/2022, às 16:37, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **servidor (a)**, em 25/02/2022, às 16:37, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0621005** e o código CRC **365C4539**.

Eu dedico esse trabalho aos meu pais Serafim Alves Barroso e Destarlinge Pereira Lima Barroso, por sempre me apoiarem nas minhas escolhas.

AGRADECIMENTOS

A Deus, pela minha saúde e por ter me capacitado para vencer todos os desafios que me foram propostos neste curso.

Aos meu pais e as minhas irmãs que sempre me incentivaram a seguir pela área da tecnologia da informação.

A minha namorada pelo amor e por ter estado ao meu lado, compreendendo a minha dedicação aos projetos da universidade.

A todos os professores da UFVJM em especial ao orientador Alessandro Vivas que deu valiosas contribuições para que eu pudesse escrever este trabalho.

A todos meus amigos de Diamantina, especificamente aos do meu período e da minha república que conviveram este período comigo, pela amizade incondicional.

A DTI Digital e a Squad Pagar que entendeu que eu precisava de apoio neste trabalho e não mediu esforços para me ajudar.

RESUMO

Com o avanço dos recursos tecnológicos e do ensino à distância, organizar a significante quantidade de informações e materiais didáticos tem se mostrado um grande desafio. Assim, sistemas voltados para gerenciar as informações de Objetos de Aprendizagem devem apresentar padronização e características que auxiliem na estruturação do conteúdo, de forma que as informações se apresentem organizadas e consigam responder às diversas finalidades educativas. Dessa forma, o objetivo deste trabalho foi criar um modelo de sistema colaborativo, de uso livre, que organiza os conteúdos da plataforma Youtube de forma a gerar um banco de dados com metadados de Objetos de Aprendizagem, orientados pelo padrão IEEE LOM. Para a fundamentação teórica, foi realizada uma revisão bibliográfica da literatura, selecionando materiais consonantes com o tema proposto. Para a criação do sistema, foram adotadas duas abordagens: uma com um *backend* desenvolvido para atender a interface gráfica e outra com a construção de uma API pública. Utilizou-se de processamento de linguagem natural para obter palavras-chave dos comentários dos vídeos do Youtube. O sistema desenvolvido conseguiu atender aos objetivos propostos, organizando os conteúdos de forma clara e objetiva. Constatou-se o bom funcionamento do Python e do micro-framework Flask em conjunto com o padrão de arquitetura MVC e a API pública do Youtube.

Palavras-chave: Objetos de Aprendizagem. Metadados. IEEE LOM. Python. API Youtube.

ABSTRACT

With the progress of technological resources and distant learning, to organize the significant amount of information and teaching materials has shown to be a great challenge. Therefore, systems guided towards managing the information of Learning Objects should present standardization and characteristics that help structuring the content in a way it appears organized and is able to answer the diverse educational goals. Thereby, the purpose of this work is to create a collaborative system template of free use, that organizes the contents from Youtube platform in order to generate a databank with Learning Objects metadata, oriented by IEEE LOM model. For theoretical validity, a bibliographic review of the literature was made, selecting materials in accordance to the subject matter. To create the system two approaches were used: one with a developed backend to answer to the graphic interface and another one with the development of a public API. Natural language processing was used to obtain the keywords on Youtube videos comments. The system developed managed to meet the purposes propounded, organizing the contents in a clear and objective manner. The well functioning of Python and the microframework Flask was noted in conjunction with the MVC architecture pattern and Youtube public API.

Keywords: Learning Objects. Metadata. IEEE LOM. Python. API Youtube.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Com o avanço dos recursos tecnológicos e com a ampliação dos materiais de ensino na Internet, a estruturação deles tem se mostrado um grande desafio. Estes materiais precisam estar bem descritos de acordo com suas características para que possam apresentar um conteúdo significativo, estruturado, organizado e representativo com relação às diferentes finalidades educativas, apoiando assim a educação.

Uma vez que existe uma demanda por estruturar e organizar tais materiais didáticos, o conceito de Objeto de Aprendizagem é importante para atendê-la. Um Objeto de Aprendizagem (OA) é um produto digital utilizado para apoiar a educação. A conceituação mais clássica de OAs é de autoria de Wiley (2002), que define tais objetos como "qualquer recurso de aprendizagem digital que possa ser utilizado em vários contextos instrucionais para dar suporte à aprendizagem". Pode-se então, estabelecer o OA como recurso para o processo de ensino e aprendizagem. Imagens, livros digitais ou videoaulas são materiais determinados como OAs.

Segundo Fontana *et al.* (2019, p. 47), estes objetos têm origem nas "percepções, construções, ressignificações e reelaborações humanas". Partindo do pressuposto que o OA tem origem no ser humano ainda é importante saber quais são seus principais aspectos.

Existem seis características fundamentais dos Objetos de Aprendizagem, que são: capacidade de ser utilizado diversas vezes, em diferentes ambientes de aprendizagem (reusabilidade); adaptação à vários ambientes de ensino (adaptabilidade); divisão de seu conteúdo em partes (modularidade); facilidade de acesso via Internet (acessibilidade); capacidade de ser usado de forma contínua, superando as mudanças tecnológicas (durabilidade) e; possibilidade de ser operado por meio de diferentes hardwares e sistemas operacionais (interoperabilidade).

A categorização de OAs assegura as suas características fundamentais, mas para isso é importante adotar padrões para fazer essa categorização. Podem ser utilizados metadados que, obedecendo à um padrão, facilitam o processo de descobrir, analisar e selecionar os Objetos de Aprendizagem. Os metadados, como a etimologia da palavra pode sugerir, significa "dados sobre dados" e, quando utilizados para descrever OAs, exibem informações que permitam o conhecimento acerca de suas características (MACHION, 2007).

Dentre os padrões de metadados, está o *Learning Object Metadata* (LOM) concebido pelo *Institute of Electrical and Electronics Engineers* (IEEE) e por isso definido como IEEE LOM, este padrão é constituído por mais de 50 campos de dados, distribuídos em nove categorias, que, segundo os autores Resende *et al.* (2014, p. 450), "descreve OA's segundo um modelo universalmente aceito, o qual facilita a interoperabilidade entre os diferentes repositórios que o empregam". Este padrão permite a utilização de extensões como, por exemplo, o *Customized Learning Experience Online* (CLEO), assim possibilitando que seja feita a inclusão de termos representativos, corrigindo problemas que o padrão pode apresentar.

Considerando a importância de se estabelecer uma estruturação correta dos Objetos de Aprendizagem, em qualquer sistema que se proponha a organizá-los por categorias, o objetivo geral deste trabalho é desenvolver um sistema colaborativo de uso livre para criação e gestão de

metadados de Objeto de Aprendizagem, a partir de dados de vídeos do Youtube. E os objetivos específicos são:

- Possibilitar a recuperação de dados de vídeos selecionados da plataforma Youtube, utilizando a API disponibilizada pela organização;
- Estabelecer uma metodologia de autenticação oferecida pelo Flask;
- Estruturar os dados dos vídeos em OAs no formato JSON de acordo com os campos de um Objeto de Aprendizagem no padrão IEE LOM com a extensão CLEO;
- Disponibilizar os dados de OAs para visualização, edição e exclusão;
- Utilizar o processamento de linguagem natural para analisar os últimos comentários do vídeo selecionado e oferecer as palavras-chave a partir dos comentários deste;
- Apresentar informações dos registros advindos das interações dos usuários no sistema;
- Possibilitar que outros sistemas façam a utilização das informações geradas, a partir de uma API pública.

Para que um sistema desse tipo apresente um bom funcionamento e seja confiável, a escolha da linguagem de programação é fundamental. Para a criação do sistema apresentado neste trabalho foi escolhida a linguagem Python. O Python é uma linguagem de programação de alto nível que foi concebida para ser acessível e, ao mesmo tempo, produtiva, com característica modular que facilita o tempo de desenvolvimento de projetos, assim correspondendo aos objetivos de desenvolvimento do sistema deste trabalho.

Os primeiros três capítulos foram dedicados a apresentar os conceitos e conhecimentos sobre temas pertinentes e necessários para o entendimento teórico dos elementos utilizados no sistema. Assim, foram abordados os conceitos sobre: Objetos de Aprendizagem, IEEE LOM, CLEO, Repositórios, Python, Bibliotecas, Flask, Banco de dados, MongoDB, API e Processamento de Linguagem Natural.

O quarto capítulo foi dedicado a apresentar o sistema, perpassando por todas as etapas de seu desenvolvimento. O quinto capítulo apresentou os resultados obtidos a partir do desenvolvimento do sistema e, por último, o sexto capítulo apresentou as considerações finais, discorrendo sobre os principais resultados advindos do desenvolvimento do sistema, bem como as sugestões de projetos futuros.

2 OBJETOS DE APRENDIZAGEM

Os conteúdos digitais com finalidade de ensino que vem sendo disponibilizados por meio da Internet, se enquadram numa categoria que pode ser chamda como Objetos de Aprendizagem (OAs). Alguns exemplos são: imagens, livros digitais, videoaulas e matérias publicadas em sites.

Assim, é possível dizer que os OAs fazem parte de uma tecnologia que é utilizada para construir materiais didáticos na Web, nela criando diversos componentes, os objetos, que tem a possibilidade de reutilização em diferentes contextos educacionais.

Segundo os autores Tarouco *et al.* (2014), os OAs são importantes ferramentas educacionais, que facilitam a aprendizagem e a instrução. As possibilidades de utilização destes, ampliam a disseminação do conhecimento e, a depender da forma como são criados, o desenvolvimento do pensamento crítico dos alunos. Entende-se que os Objetos de Aprendizagem, no contexto de ensino atual, são importantes para potencializar as possibilidades de aquisição de conhecimento. Segundo a autora Machion (2007), a proposta original do termo era descrever materiais de aprendizagem que eram criados de forma independente, de modo que permitisse a tais objetos que estes se combinassem com outros.

Além do conceito receber diferentes definições, o próprio termo "Objetos de Aprendizagem" pode variar, Machion (2007) esclarece que outras denominações são utilizadas para os materiais didáticos disponíveis em formato eletrônico. Dentre tais termos, é possível citar: objeto de conhecimento, componente instrucional, documento pedagógico, componente de software educacional, material de aprendizagem online e recurso. Já os autores South e Monson (2000) utilizam um termo ligeiramente diferente, "Objetos de Mídia", cujo uso deve obedecer a um propósito educacional.

Apesar das diferentes conceituações e termos utilizados para denominar os Objetos de Aprendizagem, é notável que estes precisam responder às diferentes finalidades educacionais de uma forma significativa, alinhada à intenção do estudante. Assim sendo, é possível encontrar características que são comuns a muitas delas, como referem Gomes *et al.* (2005): devem ser projetados para que sejam úteis sem que precisem de atualizações de *hardware* ou *software*, sempre seguindo os padrões de metadados (estes descrevem propriedades do objeto); preferencialmente, devem ser criados de forma que sua utilização independa de plataformas, navegadores de Internet ou *software*, além de seu uso para ambientes Web e; podem apresentar qualquer mídia ou formato, com reutilização ampliada a diferentes contextos.

É importante também que um OA tenha: um objetivo didático único, para que possa ser usado em vários contextos, coerência, para que sua finalidade consiga ser facilmente capturada com o uso de poucos termos e portabilidade entre plataformas, para que seu acesso não esteja inerente a uma única plataforma. Outra característica importante é a construção em formato modular, que permite que as unidades didáticas sejam combinadas de diferentes formas, fazendo que os OAs consigam atender à diversas finalidades educativas.

A modularidade beneficia alunos e professores, pois os módulos podem ser personalizados de forma que correspondam a necessidades mais específicas de cada aluno, considerando suas particularidades de aprendizagem, conhecimento prévio e velocidade de aquisição de conhecimento. Porém o autor Carvalho (2017), destacou outras características como sendo as fundamentais: interoperabilidade, acessibilidade e durabilidade.

Com relação à interoperabilidade, esta representa a capacidade que um OA possui de poder ser utilizado em diferentes ambientes daqueles que o conceberam. A acessibilidade representa a possibilidade de utilizar OAs de forma presencial ou remota. Por último, a durabilidade representa a viabilidade de utilização sem que haja a necessidade de reprojetar ou recriar o OA, independente das transformações tecnológicas ao longo do tempo.

Considerando as suas muitas características e ampla definição, os OAs podem apresentar tamanhos e formas diferentes, por isso ele precisa ser estruturado em três partes. Os autores Tarouco *et al.* (2014) as define como: objetivos – se referem aos objetivos pedagógicos que vão direcionar a utilização do objeto; conteúdo instrucional – a apresentação do conteúdo didático responsável pelo alcance dos objetivos pelo aluno e; prática – agente de retorno para que o aluno receba *feedback* sobre ter alcançado ou não os objetivos do OA.

Em conformidade com as características citadas, para que um Objeto de Aprendizagem seja útil ao ensino, precisa responder a determinados requisitos de relevância e qualidade. A qualidade de um Objeto de Aprendizagem está relacionada à sua adequação para os diferentes tipos de objetivos de aprendizagem, e pode ser avaliada, segundo os autores Romero *et al.* (2021), por meio de um instrumento denominado *Learning Object Review Instrument* (LORI), o qual utiliza nove variáveis para tal avaliação. O Quadro 1 apresenta as variáveis utilizadas neste instrumento.

Quadro 1 – *Learning Object Review Instrument* (LORI)

1. Qualidade do conteúdo
2. Adequação dos objetivos de aprendizagem
3. Adaptabilidade
4. Motivação
5. Apresentação
6. Usabilidade e interação
7. Acessibilidade
8. Reusabilidade
9. Conformidade com os padrões

Fonte: O próprio autor (2022)

Uma vez que existe essa preocupação com relação a qualidade exibida pelos Objetos de Aprendizagem que serão distribuídos na Web, existem entidades internacionais que atuam de maneira colaborativa, elaborando recomendações para a criação de conteúdo para a Web, padronizando e especificando a forma de construir, armazenar e distribuir os OAs e suas informações.

O autor Macedo (2010) cita algumas destas entidades: Electrical and Electronics Engineer's Learning Technology Standards Comitee (IEEE LTSC); DublinCore, Instructional Management Systems - Global Learning Consortium (IMS GLC), Advanced Distributed Learning – Sharable Content Object Reference Model (ADL SCORM), e World Wide Web Consortium – Web Content Accessibility Guidelines (W3C WCAG).

2.1 Repositórios de Objetos de Aprendizagem

Por conta do interesse nos OAs, atualmente existem diversos Repositórios de Objetos de Aprendizagem (ROAs) espalhados pela Web. Os repositórios são sistemas que armazenam Objetos de Aprendizagem no formato digital e/ou seus metadados, de forma que estes possam ser mantidos, gerenciados e acessados apropriadamente.

Uma das grandes vantagens dos ROAs é a facilidade que estes oferecem para a recuperação dos OAs e sua posterior incorporação à outras aplicações ou para uso dos docentes e alunos, nas diferentes situações de ensino e aprendizagem (JUNQUEIRA; LOSCIO, 2014). O Quadro 2 apresenta exemplos de alguns ROAs.

Quadro 2 – Repositórios de Objetos de Aprendizagem

Nome	**Endereço Web**
Laboratório Virtual de Matemática	<http://www.projetos.unijui.edu.br/ matematica/>
Repositório de Objetos de Aprendizagem Univates	< https://www.univates.br/roau >
Portal do Professor	<http://portaldoprofessor.mec.gov.br/>
Banco Internacional de Objetos Educacionais	<http://objetoseducacionais.mec.gov.br/>
Escola Digital	<https://escoladigital.org.br/>

Fonte: O próprio autor (2022)

Os Repositórios de Objetos de Aprendizagem exibem três propriedades básicas, que são: tipo, técnicas e qualidade. O tipo do repositório pode ser geral ou de conteúdo (também conhecido como temático), ele envolve aspectos como público-alvo, localização geográfica, área de conhecimento e idioma utilizado nos recursos pedagógicos. As técnicas descrevem os serviços que são fornecidos e os padrões de metadados utilizados enquanto, a qualidade está relacionada à segurança deste repositório, autenticação exigida e mecanismos de avaliação do conteúdo (SILVA; SOUZA, 2017).

Os autores Silva *et al.* (2010) referem aos tipos de ROAs com outra abordagem, estabelecendo que os repositórios podem ser centralizados em dois tipos: 1 – armazenam os OAs com seus respectivos metadados e; 2 – armazenam somente os metadados. No caso desta última, os OAs podem estar armazenados em outro local, podendo ser localizados através das informações dos metadados e da utilização de ferramentas específicas. Os autores Tarouco *et al.* (2014) propuseram ainda um terceiro tipo, os repositórios ditos híbridos, que apresentariam características inerentes às duas outros classes.

Uma vez que o número e diversidade de funções que os repositórios oferecem é elevado, os autores Sampson e Zervas (2013) propuseram o agrupamento dos recursos em três dimensões, as quais estão apresentadas no Quadro 3.

Quadro 3 – Dimensões propostas por Sampson e Zervas (2013) para funcionalidades dos ROAs

Dimensão de Objetos de Aprendizagem	Funcionalidades relacionadas a interação e manipulação dos OAs pelos usuários, localmente ou remotamente (ex.: pesquisar, comentar e visualizar).
Dimensão de metadados que descrevem OAs	Operações relacionadas a interação e manipulação dos metadados pelos usuários (ex.: armazenar e baixar).
Dimensão de serviços de valor agregado	Funções relacionadas a experiência do usuário quando na utilização das outras duas dimensões (ex.: criar contas, feed de notificações e fóruns).

Fonte: Adaptado de Silva e Souza (2017)

Os ROAs, ainda, podem ser colaborativos, apresentando um controle flexível de seus conteúdos e da autoria dos documentos, geralmente tendo como representantes aqueles que são voltados para o público geral, ou podem ser mais controlados e direcionados para um público específico (COSTA *et al.*, 2017).

2.2 *Learning Object Metadata*

Os Objetos de Aprendizagem possuem um papel fundamental quando se trata da recuperação e interoperabilidade de materiais didáticos, entretanto a categoria pode ser composta por uma infinidade de tipos de materiais, que podem variar desde pequenos parágrafos contendo textos explicativos até cursos completos. Por serem tão abrangentes, suas propriedades precisam refletir algumas informações de utilização e combinação.

Quando essas informações são descritas, o Objeto de Aprendizagem consegue atingir os objetivos didáticos para os quais foi criado. Para tanto, é necessário que essas informações sejam especificadas de forma coesa com as características dos OAs e sejam descritas utilizando

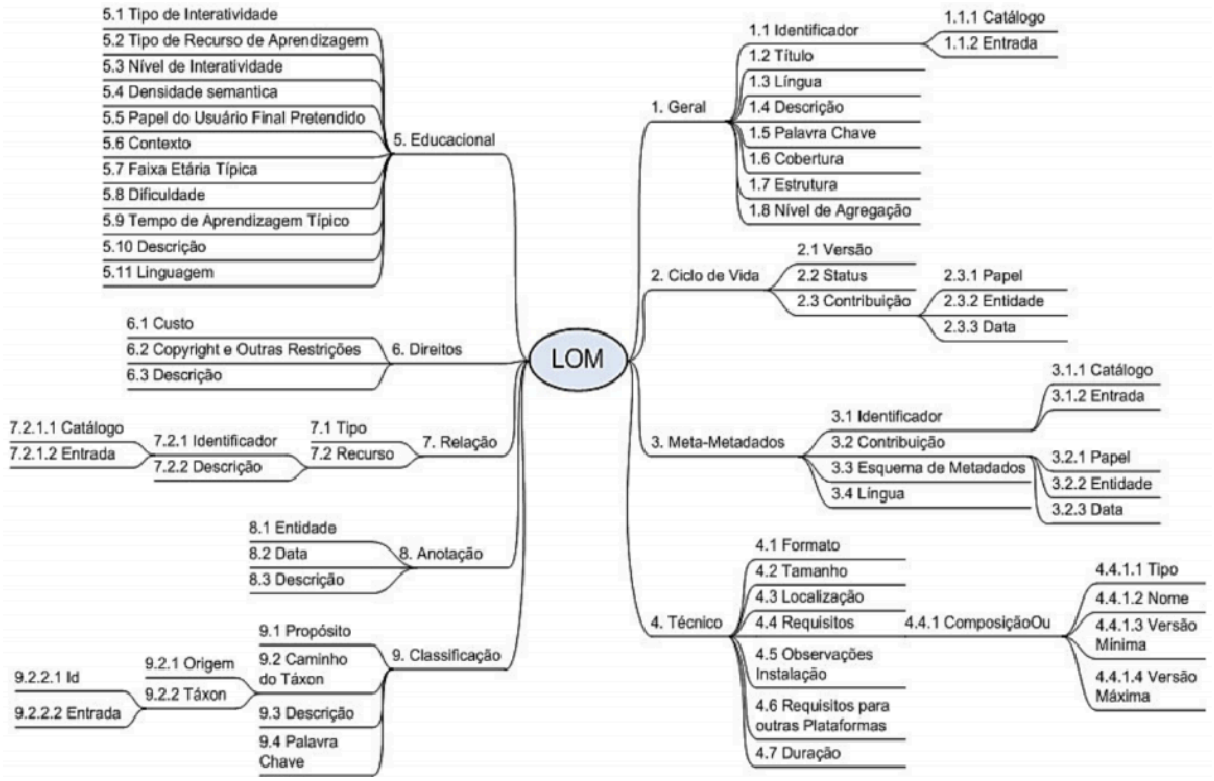
os metadados que, obedecendo à um padrão, facilitam o processo de descoberta destes objetos, bem como sua análise, seleção e inserção em repositórios. Essa padronização, feita através dos metadados, é indispensável para manter as características básicas de um OA, como já mencionado anteriormente, pois os metadados direcionam a utilização e condições de uso de um Objetos de Aprendizagem.

O padrão IEEE *Learning Object Metadata* (LOM) apresenta a importante função de descrever OAs e outros recursos digitais. Com relação ao padrão IEEE LOM, o IEEE (2020) refere que:

> O objetivo deste padrão é facilitar a pesquisa, avaliação, aquisição e uso de Objetos de Aprendizagem, por exemplo, por alunos, instrutores ou processos de software automatizados. O objetivo é também facilitar o compartilhamento e a troca de Objetos de Aprendizagem, permitindo o desenvolvimento de catálogos e inventários, levando em consideração a diversidade de contextos culturais e linguais em que os Objetos de Aprendizagem e seus metadados serão explorados. Ao especificar um esquema de dados conceituais comum, esse padrão ajuda a garantir que as ligações de Metadados de Objetos de Aprendizagem (LOM) provavelmente terão um alto grau de interoperabilidade semântica. Como resultado, as transformações entre as ligações serão diretas.

Este padrão é composto por mais de 50 campos distribuídos em nove categorias, "Geral, Ciclo de Vida, Meta-Metadado, Técnico, Educacional, Direitos, Relação, Anotação e Classificação que obedecem à uma hierarquia (MENDES *et al.*, 2017). Estas categorias, segundo Carvalho (2017, p. 30) "descrevem a semântica dos metadados de um OA". Desta forma, estas categorias são importantes para relacionar as informações que estabelecem a relação semântica entre os dados disponíveis, atribuindo o significado ao significante da forma correta. Os elementos que constituem o IEEE LOM podem ser observados na Figura 1.

Figura 1 – Categorias de metadados do IEEE LOM



Fonte: Machion (2007)

A Figura 1 mostra que algumas categorias, ainda, podem ser subdivididas, assim conferindo maior nível de detalhes às descrições, com relação aos elementos agregados e simples, estes últimos possuindo valores. (MACHION, 2007).

Cada categoria do padrão LOM integra um conjunto de campos importantes para a identificação dos Objetos de Aprendizagem, e os valores que cada campo assume são usados na recuperação dos objetos desejados (JUNIOR, 2018). O quadro 4 apresenta as categorias do padrão IEEE LOM, com suas respectivas descrições.

Quadro 4 – Descrição das categorias do padrão IEEE LOM

Categoria	Descrição
1. Geral	Agrega informações gerais que descrevem o OA, tais como identificador, título e palavra-chave.
2. Ciclo de Vida	Descreve o histórico do OA, de sua criação até seu estado atual, usando atributos como versão e contribuição.
3. Meta-metadados	Reúne atributos sobre os metadados, como contribuição, data de criação e língua em que são descritos.
4. Técnico	Características técnicas do OA, como formato, tamanho, duração e observações de instalação.
5. Educacional	Descreve características pedagógicas do OA, como tipo de interatividade, tipo do recurso de aprendizagem, densidade semântica, faixa etária, grau de dificuldade e tempo de aprendizagem.
6. Direitos	Agrega os direitos de propriedade intelectual e as condições de uso do OA.
7. Relação	Define os relacionamentos do OA com outros OAs, como "é parte de", "é baseado em" e "é requerido por".
8. Anotação	Provê comentários de pessoas que já utilizaram o OA.
9. Classificação	Classifica o OAs dentro de uma taxonomia (categorização).

Fonte: Adaptado de Junior (2018)

.

Ainda, segundo o autor (AGUIAR *et al.*, 2015), este padrão é altamente aceito no meio acadêmico, por conta de sua organização que permite a inserção de novas categorias e de outros elementos em categorias que já existem. O padrão IEEE LOM define a estrutura de instâncias de metadados para OAs a partir de dados conceituais, sendo o esquema de metadados

mais utilizado para descrever Objetos de Aprendizagem, conforme afirmam os autores Resende *et al.* (2014). Adicionalmente, os padrões de metadados são fundamentais para a reutilização dos conteúdos, definição de regras para a navegação condicional e/ou adaptativas.

Embora o padrão IEEE LOM tenha sido desenvolvido especialmente para descrever objetivos de aprendizagem, este pode apresentar falhas. Assim, algumas extensões deste padrão foram disponibilizadas, as quais determinam outros termos para campos específicos do padrão (CARVALHO, 2017).

2.3 *Customized Learning Experience Online*

Pensando nas falhas decorrentes do padrão IEEE LOM, criou-se o *Customized Learning Experience Online* (CLEO). Ele proporciona maior flexibilidade e abre novas possibilidades no processo de recomendação, permitindo fornecer conteúdo personalizado para uma grande quantidade de diferentes perfis de alunos. Com relação as principais contribuições da extensão CLEO, é possível citar: vocabulários adicionais, vocabulários alternativos, de acordo com o tipo de recurso de aprendizagem e seu propósito e; novos componentes para a categoria educacional.

A extensão cria uma estrutura capaz de descrever objetos específicos dentro do padrão IEEE LOM, o que reforça os requisitos de adaptabilidade dos OAs. Sendo assim, o Quadro 5 mostra os campos que foram adicionados.

Quadro 5 – Categorias CLEO

Campos adicionados pela extensão CLEO	
Subnível de agregação	Tipo de recurso de aprendizagem
Intervalo de tempo de aprendizado	Domínio cognitivo
Estratégia cognitiva	Propósito

Fonte: Adaptado de Taliesin (2003)

.

Entre os motivos da utilização destes campos estão: o maior detalhamento do relacionamento dos OAs com outros objetos, a identificação do tipo de recurso de aprendizado (pode ser vídeo, imagem, texto entre outros), identificação do tempo médio gasto para aprender através de um determinado OA e a identificação da estratégia cognitiva utilizada no OA. A descrição de todos os campos mostrados na figura 5 podem ser conferidas na seção 6, onde são mostrados os campos do padrão IEEE LOM e a extensão CLEO.

3 TECNOLOGIAS PARA DESENVOLVIMENTO WEB

Para o desenvolvimento deste trabalho foram analisadas tecnologias e ferramentas relevantes para a criação de sistemas Web. Este capítulo tem o objetivo de descrever estas tecnologias e ferramentas bem como apresentar e justificar a utilização delas.

3.1 Python

Python é uma linguagem de programação considerada de alto nível, ou seja, uma *Very High Level Language* (VHLL), que foi concebida por Guido Van Rossum, um holandês, cujo objetivo primeiro era a acessibilidade no que se refere a programação, tornando a "programação de computadores para todos". Foi este ideal que norteou algumas das premissas que se refletem na linguagem: liberdade (código aberto e gratuita), disponibilidade (roda em vários sistemas operacionais, como Windows, Linux e Mac OS X) e sintaxe clara, o que torna o Python uma das linguagens que demonstra maior produtividade (rapidez para aprender e a desenvolver com a linguagem).

É uma linguagem Orientada a Objetos (OO), que segue um paradigma o qual utiliza coleções cooperativas de objetos para a construção de programas, motivo que o torna um facilitador no desenvolvimento de projetos, especialmente quando estes tomam grandes proporções. Embora a OO seja encarada como um grande paradigma, mesmo dentre os especialistas da área, o Python permite ao usuário que a programação seja feita de forma estruturada.

Por conta destas características, empresas e universidades de renome começaram a utilizar o Python com relativo sucesso, dentre estes é possível citar: Philips, Industrial Light and Magic, Nasa, Aliança Espacial Universal (Estados Unidos), Nokia, Disney, Google e Yahoo (LABAKI, 2016).

Na área de *Frequently Asked Question* (FAQ) do Portal Brasileiro de Python, é possível conferir a descrição completa das vantagens de usar a linguagem, porém as características determinantes para sua escolha para o projeto desenvolvido neste trabalho foram: sintaxe intuitiva, muita documentação e materiais didáticos a respeito disponíveis e ter bibliotecas abertas.

É possível compreender o que tornou o Python a segunda linguagem de programação mais utilizada no mundo Branco (2021), já que esta linguagem é de fácil aprendizado, ensinamento, utilização, entendimento e obtenção. O Quadro 6 apresenta os resultados de uma pesquisa realizada pela empresa de análise de dados SlashData no terceiro trimestre de 2021, abrangendo mais de 160 países.

Quadro 6 – As 6 linguagens de programação mais utilizadas no mundo, segundo a SlashData

Linguagem	**Estimativa de desenvolvedores utilizando**
Javascript	16,4 milhões
Python	11,3 milhões
Java	9,6 milhões
C/C++	7,5 milhões
PHP	7,3 milhões
C#	7,1 milhões

Fonte: Adaptado de Branco (2021)

.

Sua utilização abrange a implementação de serviços como mecanismos de pesquisa, armazenamento e ferramentas em nuvem, mídias sociais, dentre outros. O Python também é bastante utilizado por aplicativos de uso rotineiro. Conforme o Python Institute (2017), alguns exemplos de utilização desta linguagem incluem:

- Desenvolvimento para Web e Internet (*frameworks* Django e Pyramid, *micro-frameworks* Flask e Bottle);
- Computação Científica e numérica (SciPy e IpyThon);
- Educação;
- Desenvolvimento de Softwares (controle de compilação, gerenciamento e teste – Scons, Builbot, Apache Gump, Roundup, Trac);
- Aplicativos de negócios (sistemas ERP e e-commerce);
- Jogos, sites e serviços.

Para o desenvolvimento em Python, existem ferramentas para auxiliar o programador, como o editor de código que possui funcionalidades como realce de sintaxe, complementação e refatoração de códigos, entre outras. Outra ferramenta muito importante é o *Integrated Development Environment* (IDE), que além de ajudar a codificar traz mais ferramentas que dão suporte ao desenvolvedor como análise de código, depurador, relatórios, controle de versão, suporte em testes de *software* e uma das mais importantes, o compilador, que traduz o código fonte desenvolvido em linguagem de máquina.

Os IDEs oferecem muitos benefícios aos programadores, como redução do tempo para montar o ambiente de desenvolvimento, maior velocidade para as tarefas e padronização dos processos que envolvem a criação de software. Alguns IDEs possuem código aberto, en-

quanto outros são comercializados, podendo ser aplicativos independentes ou fazer parte de um pacote de software. O Python disponibiliza de uma IDE nativa denominada Integrated Development and Learning Environment (IDLE), através da qual é possível executar linhas de código que permitem testar o programa e até rodá-lo completamente, gerando imediatamente o conhecimento sobre os objetos que foram definidos, e fazer a interpretação referente a estes objetos e outros recursos do Python (LABAKI, 2016). Além da IDLE, existem outras IDEs para Python, gratuitas, por exemplo: PyCharm Community, Komodo-Edit, DrPython, IPython e PythonCard.

Embora programar em Python seja considerado relativamente simples, uma desvantagem de utilizar as linguagens de alto nível é que o processo de conversão do código fonte em linguagem de máquina, realizado antes da execução do programa por meio dos interpretadores ou compiladores, gasta um tempo extra (em comparação com linguagens de baixo nível) para serem processadas.

Sobre o compilar e o interpretador, a diferença entre eles é que enquanto o interpretador realiza a leitura do código fonte de um programa e o executa, realizando todas as ações chamdas no programa, o compilador realiza a leitura do programa e o traduz completamente de código-fonte para objeto/executável, para que só então este seja executado. Uma vez compilado, o programa poderá ser executado novamente sem a necessidade de utilizar um compilador. Python é uma linguagem interpretada. Mas diferente de outras linguagens, antes desse processo de interpretação acontecer, o Python é compilado para *bytecode* (linguagem de máquina) e só depois é interpretado linha a linha. A Figura 2 faz a representação esquemática dos passos executados por interpretadores e compiladores.

Figura 2 – Processos de interpretação e compilação

Fonte: O próprio autor (2022)

3.2 Bibliotecas

Com as bibliotecas muitas das funções já existentes podem ser utilizadas com a escrita de algumas poucas linhas de códigos, ao utilizar as bibliotecas, é possível ter acesso aos programas que já foram desenvolvidos e testados, minimizando os erros e direcionando a atenção para a resolução do que é necessário. O autor Costa (2006) afirma que recorrer às bibliotecas Python permite aos programadores importar módulos e os utilizar em seus códigos, dispensando a necessidade de reescrever comandos que são usualmente utilizados, os quais podem ser chamados/importados no início de um *script*.

O Python apresenta uma biblioteca padrão, a qual é distribuída juntamente com ele, segundo informa o site oficial da Associação Python Brasil (2021). Esta biblioteca é muito extensa, oferece diversos recursos (os quais podem ser conferidos em detalhes no referido site) e apresenta módulos embutidos, que são escritos em linguagem C e conferem acesso às funcionalidades do sistema, como à E/S de arquivos que, caso contrário, não seriam acessíveis para programadores Python, e módulos escritos em Python que oferecem soluções padrão para vários dos problemas que normalmente acontecem em programação. Além de sua biblioteca padrão, existem bibliotecas desenvolvidas por terceiros. Dentre estas bibliotecas, algumas podem ser citadas considerando sua relevância para o desenvolvimento deste trabalho, como Copy, JSON, BSON, Werkzeug, Datetime, Wtforms e Jinja2.

A biblioteca JSON auxiliam na manipulação dos formatos de dados JSON, capturando os dados recebidos e os convertendo em matrizes ou outros formatos que os servidores consigam utilizar. As bibliotecas BSON funcionam da mesma maneira que as bibliotecas JSON, facilitando a manipulação dos dados BSON. A Werkzeug é uma biblioteca Python que disponibiliza diversas ferramentas para implementar aplicativos WSGI (do inglês - Web *Server Gateway Interface*). O WSGI refere-se a especificações de interface entre servidores Web/*frameworks*/aplicações Web para a linguagem Python.

A Datetime é uma biblioteca nativa do Python, com funcionalidades importantes que lidam com variáveis como dias, meses, anos e horas, os quais podem ser transformados em um objeto *datetime*. A Wtforms, por sua vez, traz uma forma de gerenciamento e validação de formulários, utilizada para o desenvolvimento Web em Python. Suporta a validação de dados, proteção de dados pessoais, suporta desenvolvimento em diferentes idiomas, dentre outras funcionalidades.

Por fim, a biblioteca Jinja2 é muito utilizada para a construção de *templates* (modelos) para diversos formatos de saída, tendo origem um arquivo, utilizado como *template* principal. Apresenta uma Application Programming Interface (API) completa e muitas diretrizes sintáticas, o que permite que seja feita a pronta inserção de conteúdos no arquivo *template* (IBM, 2021).

3.3 Flask

O Flask é um *micro-framework* da Web escrito em Python, desenvolvido por Armin Ronacher e lançado no ano 2010. *Frameworks* convencionais são modelos de dados com

diversas ferramentas criadas em determinada linguagem de programação com o objetivo de auxiliar o desenvolvedor. Os *micro-frameworks* tem estruturas mais enxutas e possibilita que o desenvolvedor inclua no projeto apenas o básico.

Para o desenvolvimento de pequenas aplicações, cujos requisitos sejam mais simples, como sites mais básicos, o Flask é adequado. Ele apresenta um núcleo, que é simples, porém extensível, assim permitindo ao projeto apresentar apenas o essencial para sua execução, mas recrutar novos pacotes para incrementar suas funcionalidades, conforme a necessidade (ASLAM *et al.*, 2015).

O Flask foi projetado para facilitar os primeiros passos de desenvolvimento de uma aplicação Web, implementando os recursos mínimos para tornar os projetos viáveis e permitindo ao desenvolvedor adicionar os recursos. Assim, apresenta a capacidade de ser escalado rapidamente para aplicativos mais complexos, oferecendo ao desenvolvedor flexibilidade total para adicionar os requisitos conforme a necessidade (GHIMIRE, 2020).

Para que uma aplicação Flask funcione, ela precisa estar integrado a um servidor Web. Este, recebe e responde às requisições feitas através do protocolo HTTP (do inglês - *HyperText Transfer Protocol*), que é um protocolo de comunicação entre o cliente (geralmente o navegador Web) e o servidor.

Antes de prosseguir com as utilidades do Flask, é importante esclarecer alguns conceitos: *backend*, *frontend* e *endpoint*. Enquanto o *backend* está relacionado a parte lógica das aplicações, fazendo procedimentos e operações envolvendo a aplicação, o servidor e o banco de dados, o *frontend* está relacionado a estrutura, conteúdo visível e *design* de uma página, sendo responsável em mostrar os dados. Com isso, o usuário consegue interagir apenas com o *frontend*. O *endpoint*, por sua vez, é o identificador, utilizado para determinar qual será a unidade lógica do código, responsável por tratar determinada solicitação.

O Flask pode ser utilizado tanto para *backend* puro quanto para *frontend*, se necessário, quando usado para *backend* fornece o depurador interativo, objeto de solicitação completa, sistema de roteamento para *endpoint*, controles de *cache* (informações que o navegador armazena para recarregar a página mais rapidamente nas próximas vezes), datas, *cookies* (informações que a aplicação armazena no navegador), entre outros.

Para o desenvolvimento em servidor local (máquina onde o aplicativo está sendo executado e testado), o Flask oferece o Web Server Gateway Interface (WSGI). Ele é uma camada localizada entre a aplicação e o servidor, seu objetivo é receber uma requisição e chamar o *endpoint* correspondente passando os dados dessa requisição. Além de disponibilizar esta camada, é possível citar outras propriedades do Flask:

- Simplicidade, uma vez que apresenta apenas o essencial para o desenvolvimento das aplicações, seus projetos são mais simples quando comparados aos *frameworks* convencionais, pois possui número de arquivos reduzido e a arquitetura mais simplificada;
- Rapidez com relação ao desenvolvimento: o desenvolvedor pode focar no necessário para o projeto, sem precisar fazer configurações que, na maioria das vezes, não são usadas;

- Projetos pequenos: considerando sua estrutura mais simples, projetos em Flask costumam ser mais leves e menores, quando comparados à maiores *frameworks*;
- Aplicações mais robustas: possibilita a elaboração de aplicações robustas, apesar de sua simplicidade, pois é altamente personalizável, assim permitindo a concepção de uma arquitetura mais clara;
- Suportes integrados para teste de unidades;
- Utiliza modelos de *frontend*;
- Suporte para *cookies* seguros.

Embora no PyPI (repositório de *softwares* para a linguagem Python) seja possível encontrar grande número de extensões para o Flask, também é possível criar extensões que atendam à necessidades específicas. Além de oferecer a vantagem de poder usar muitas extensões, o Flask também tem um processo de instalação simplificado com o gerenciador de pacotes oficial do Python pip (GHIMIRE, 2020).

Entre as principais bibliotecas para se trabalhar com o Flask estão o Jinja2 e o Werkzeug. O Jinja2 é uma biblioteca para se trabalhar com *frontend*. Ela tem um mecanismo de modelo completo e, dentre seus recursos, pode-se citar: execução em área restrita, prevenção de ataques Cross-Site-Scripting (XSS) (tipo de ataque que injeta códigos maliciosos em aplicações Web), herança de *templates*, fácil apuração e sintaxe configurável. Em contrapartida, o Werkzeug fornece um servidor Web com finalidades de desenvolvimento, tendo seu início como uma coleção de diversos utilitários para aplicativos WSGI (GHIMIRE, 2020; THE PALLETS PROJECT, 2021).

3.4 Banco de dados

Bancos de dados são as estruturas de organização e armazenamento de informações para finalidade de conferência futura e/ou segurança. Existem vários tipos diferentes de bancos de dados disponíveis atualmente, cada qual apresentado sua particularidade. Entretanto, podem ser divididos, principalmente, em duas categorias principais: bancos de dados relacionais e bancos de dados não relacionais. Os modelos de banco de dados relacionais utilizam a linguagem *Structured Query Language* (SQL) e apresentam uma estrutura de dados em formato de tabela, enquanto os modelos de banco de dados não relacionais, que utilizam linguagem NoSQL, possuem estrutura de dados horizontal (COSTA, 2006).

Segundo a Associação Python Brasil (2021), o Pyhton não possui nenhum tipo de acesso nativo aos bancos de dados SQL, quando considerada sua biblioteca padrão, mas somente ao *Berkeley Database Engine* (BDB). Entretanto, ele define uma API padrão, a qual os *drivers* de acesso a estes bancos de dados precisam seguir, o que torna os procedimentos relacionados a qualquer banco de dados parecidos (MENEZES, 2015). Os *drivers* de acesso possibilitam a interação entre uma aplicação e um banco de dados. Ele é responsável por implementar um protocolo para que as consultas e resultados sejam transferidos entre o cliente e o banco de dados.

Nem todos os sistemas precisam de banco de dados, somente aqueles que necessitam recuperar e armazenar as informações da aplicação. Persistir os dados é essencial para criar um repositório. Sendo assim, para alcançar os objetivos deste trabalho, escolheu-se dentre os banco de dados Oracle, MySQL, Microsoft SQL Server, PostgreSQL, MongoDB, IBM Db2, Redis, Elasticsearch, SQLite e Microsoft Access, que são os bancos de dados mais utilizados e populares segundo a pesquisa da empresa Austrian IT Consulting, de 2021 Boghi (2021), o MongoDB, por se tratar de um banco de dados gratuito, de código aberto e eficiente.

3.4.1 MongoDB

Em 2007, uma empresa chamada 10gen começou a trabalhar em uma plataforma, a qual consistiria, futuramente, em um servidor de banco de dados. Assim nasceu o MongoDB, que foi lançado oficialmente em 2009, cuja principal característica é apresentar um alto desempenho no processamento de grandes quantidades de dados. Ele se destina ao uso de aplicativos da Web e da internet, assim, desde o início, sua estrutura de dados foi otimizada para tais aplicações.

Segundo os autores Boicea *et al.* (2012), este banco de dados foi a escolha para projetos importantes de grandes empresas, como MTV Networks, Craiglist, Disney Interactive Media Group, Sourceforge, Wordnik, Yabblr, SecondMarket, The Guardian, Forbes, The New York Times, bit.ly, GitHub, FourSquare e Disqus, PiCloud.

O MongodB, é um banco de dados do tipo NoSQL e por isso, os dados são armazenados em documentos. Cada documento consiste em um conjunto ordenado de propriedades que correspondem à um nome e um valor. Estes valores podem ser tipos de dados mais simples ou outros documentos. Outra característica dos documentos é a ausência de uma estrutura fixa, sendo preenchidos com os dados necessários conforme a necessidade (MOGLICH, 2011). O conjunto de documentos presentes em um mesmo banco de dados formam as coleções.

Os dados são salvos em formato Binary JavaScript Object Notation (BSON) e JavaScript Object Notation (JSON). O BSON é caracterizado pela eficiência e economia de espaço, além de poder ser codificado e decodificado com alto desempenho, devido à sua representação de tipos, baseada em C. Independentemente do *driver* utilizado, cada documento é mapeado internamente para uma construção BSON.

O JSON, por sua vez, tem objetivo de facilitar a troca de dados e informações independente da linguem de programação, similar ao *Extensible Markup Language* (XML), porém enquanto XML utiliza uma sintaxe baseada em *tags* personalizadas, o JSON adota uma notação que consiste em chave e valor. É importante conceituar também que XML se tratar de uma linguagem de marcação e o JSON é uma metalinguagem. Os tipos de dados que podem estar presentes no JSON são texto, número, booleano e nulo. A Figura 3 demonstra a sintaxe do JSON.

Figura 3 – Exemplo de JSON

```
{
    "texto": "Henrique",
    "numero": 26,
    "booleano": true,
    "nulo": null
}
```

Fonte: O próprio autor (2022)

Comparado a bancos de dados relacionais, o MongoDB, utiliza documentos podem ser expandidos de forma dinâmica, ou seja, respondendo prontamente a necessidade, assim facilitando operações de leitura e escrita. Os documentos são a contrapartida das colunas, enquanto as coleções são a contrapartida das tabelas (MOGLICH, 2011).

Dessa forma, enquanto o MongoDB lida com documentos, os bancos de dados relacionais lidam com registros, que são representados adotando uma abordagem bidimensional, em que as tabelas apresentam a dimensão de linhas e de colunas. As coleções, por sua vez, podem conter documentos com estruturas completamente diferentes.

Outro ponto importante, que diferencia o MongoDB de bancos de dados relacionais, é que os documentos são autossuficientes, ou seja, integram todos os dados que são necessários, ao invés do conceito utilizado pelos bancos de dados relacionais, de não repetição e chaves estrangeiras (JUNIOR, 2017).

3.5 API

Application Programming Interface (API) é um padrão para integração de softwares por meio de mensagens de requisição e reposta. Geralmente usam mensagens no formato JSON ou XML e um protocolo de comunicação. Dessa forma, os desenvolvedores de APIs disponibilizam uma interface que pode ser utilizada por terceiros, cuja acessibilidade depende do tipo de API, se pública, privada ou entre parceiros.

As APIs públicas permitem que pequenas funcionalidades, dentro da plataforma principal, sejam acessadas. Uma API pública muito conhecida é a disponibilizada pelo Google Maps que, ao fornece-la, possibilitou que outros sites e aplicativos pudessem criar os seus serviços baseados nos dados que são recebidos do Google Maps. Por exemplo, um aplicativo de transporte urbano que pode localizar pontos e traçar rotas utilizando as funcionalidades providas pelo Goole Maps.

As APIs privadas têm um aspecto diferente, elas são usadas exclusivamente pelas empresas para as quais elas foram desenvolvidas, uma vez que conferem acesso a dados e sistemas internos da organização. Os objetivos principais de uma API privada, segundo Brandão (2020) são: integrar sistemas internos, construir novos sistemas, elevar a produtividade, integrar setores diferentes e otimizar a comunicação interna. Sobre, as APIs entre parceiros, elas

são similares as privadas, porém com um nível de acessibilidade um pouco maior, uma vez que integra sistemas de empresas parceiras.

Conforme afirma o autor Lopes (2018), as APIs surgiram de uma necessidade de permitir que aplicações interagissem com outros sistemas. Uma vez que as funções de um API só são acessíveis por meio de programação, quem configura este processo são os desenvolvedores, que integram as funcionalidades oferecidas por essas APIs em outras aplicações. Com isso, as APIs permitem a troca de informações entre aplicativos sem a intervenção dos usuários.

Dentre os tipos de APIs existentes está a Web API. Ela tem o objetivo de fazer a integração entre um servidor e um cliente. Para desenvolver Web APIs existe uma arquitetura muito relevante entre os desenvolvedores chamada Representational State Transfer (REST). Essa arquitetura adota o protocolo HTTP na versão 1.1 para controlar a comunicação e a transferência de dados entre o cliente e o servidor. O HTTP aceita alguns tipos de métodos de requisição (estes apontam a ação a ser executada), por exemplo: *get*, *post*, *put* e *delete*. Através desses métodos foi possível padronizar as APIs, esse é o objetivo da REST.

A API do Youtube, relacionada ao projeto desenvolvido neste trabalho, é considerada uma API pública, que possibilita aos desenvolvedores integrarem funções presentes nesta plataforma aos seus sistemas. De acordo com o site oficial do YOUTUBE (2019), são possíveis de serem recuperados, utilizando a API, diversos recursos, como: *activities*, *channelBanners*, *channels*, *guideCategories*, *playlistItems*, *playlists*, *search*, *subscriptions*, *thumbnails*, *videoCategories* e *videos*.

Para o desenvolvimento deste trabalho foram realizadas requisições que tinham o objetivo de obter: uma lista de dados de vídeos de acordo com um determinado termo, um único vídeo de acordo com seu identificador e os comentários de um determinado vídeo. Mais descrições de como esse processo foi realizado estão na seção 5, porém para exemplificar, uma requisição foi feita com o objetivo de obter um determinado vídeo no Youtube chamado “[Cap.13 Comandos para Analisar Desempenho] - Comando sar – Completo”. Os dados passados como parâmetro foram: o identificador do vídeo e uma propriedade chamada *snippet*, ela quer dizer que se pretende obter informações básicas do vídeo. A API do Youtube retornou o código de status HTTP 200 (significa sucesso) e um objeto. A Figura 4 mostra o JSON devolvido pelo API (nesta figura se omitiu alguns objetos com informações menos relevantes).

Figura 4 – JSON retornado pela API do Youtube


```
{
  "kind": "youtube#videoListResponse",
  "etag": "fhzlSyilPrH3gdxwO-V-RSQ5qWo",
  "items": [
    {
      "kind": "youtube#video",
      "etag": "dUfV3-q8FNOQ3-V3Axb-VUK776c",
      "id": "Tv_8pxNsXq8",
      "snippet": {
        "publishedAt": "2021-03-11T11:15:15Z",
        "channelId": "UCTsiWtFDpiZAgd6-k_F7Qdg",
        "title": "[Cap.13 Comandos para Analisar Desempenho] -
        Comando sar - Completo",
        "description": "Esta série de vídeos tem como objetivo
        apresentar os conceitos básicos sobre o uso do
        Terminal do Linux. Vamos utilizar como referência o
        Livro Linux: Comandos Básicos e Avançados disponível
        gratuitamente no site
        http://www.decom.ufvjm.edu.br/vivas.\n\nNeste vídeo
        apresento várias maneiras de utilizar o comando sar.",
        "thumbnails": {},
        "channelTitle": "Alessandro Vivas",
        "tags": [],
        "categoryId": "27",
        "liveBroadcastContent": "none",
        "localized": {},
        "defaultAudioLanguage": "pt-BR"
      }
    }
  ],
  "pageInfo": {
    "totalResults": 1,
    "resultsPerPage": 1
  }
}
```


Fonte: O próprio autor (2022)

4 PROCESSAMENTO DE LINGUAGEM NATURAL

O Processamento de Linguagem Natural (PLN) faz parte da Inteligência Artificial (tecnologia que desenvolve ferramentas capazes de resolver problemas com base em aprendizado de acordo com os dados fornecidos ou coletados), constituindo uma vertente que auxilia os computadores no entendimento, interpretação e manipulação da linguagem humana. Sua base é multidisciplinar, englobando conhecimentos de disciplinas como Linguística Computacional e Ciência da Computação, dentre outras, e tem como objetivo fundamental preencher o espaço que existe entre a comunicação humana e seu entendimento pelos computadores (SAS, 2021).

Como os autores Barbosa *et al.* (2017) referem, a linguagem considerada natural é toda linguagem utilizada para a comunicação do dia a dia por seres humanos, as quais contrastam com as línguas artificiais, como as linguagens de programação e notações matemáticas. O PLN extrai significados e representações mais completas de textos livres escritos nas linguagens naturais.

A PLN tem o objetivo de estabelecer comunicação entre o ser humano e o computador. Ela capacita computadores para que façam leitura de textos, interpretação de falas, identifiquem intenções e sentimentos. Além disso, ela possibilita que essas e outras tarefas sejam escaladas, o que torna a sua aplicação importante para analisar grandes quantidades de dados não estruturados de forma eficiente e completa.

Segundo os autores Barbosa *et al.* (2017), o PLN faz uso, geralmente, de conceitos linguísticos como classes de palavras, as chamadas *Part-Of-Speech*, além de outras estruturas gramaticais. As técnicas utilizadas para interpretar a linguagem humana são diversificadas, incluindo desde métodos estatísticos, *machine learning* até abordagens algorítmicas que se baseiam em regras específicas. Tal diversidade de métodos se justifica pela divergência existente entre os tipos de informações, sejam em texto ou em voz, bem como suas aplicações práticas (SAS, 2021).

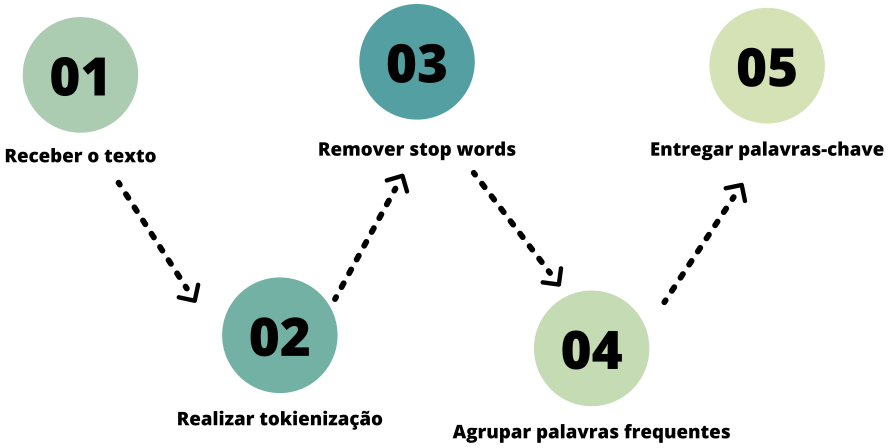
Dentre as tarefas básicas do PLN, é possível citar: tokienização e análise sintática, lematização/stemização, detecção de idiomas, rotulagem de componentes do discurso e reconhecimento de funções semânticas. Em termos práticos, as tarefas do PLN realizam uma segmentação da linguagem, inferindo relação entre as partes e explorando o funcionamento destas juntas para criar significado. Os níveis mais complexos do PLN incluem:

- Categorização de conteúdos: resumo do documento com base em linguística, incluindo pesquisa e indexação, alertas com relação ao conteúdo e identificação de duplicações;
- Descoberta e modelagem de tópicos: detecta precisamente os temas em coleções de textos, aplicando *advanced analytics* para otimização e *forecasting*;
- Extração contextual: extração automática de informações estruturadas, a partir de fontes textuais;
- Análise de sentimento: identificação do estado de espírito ou das opiniões implícitas em grandes quantidades de texto, o que inclui o sentimento médio;

- Conversão entre fala-texto e texto-fala: transforma comandos de voz na forma textual escrita, e vice-versa;
- Sumarização: cria sinopse de textos grandes de forma automática;
- Tradução de máquina: traduz a fala ou o texto escrito de um idioma para outro.

Neste trabalho se utilizou a PLN para se obter as palavras-chave de textos utilizando as tarefas de: *Part-Of-Speech* e *tokienização* juntamente com o conjunto de palavras chamado *stop words*. A próxima seção vai explicar detalhadamente cada uma destas tarefas. A Figura 5 mostra o fluxo de extração das palavras-chave que foi implementado.

Figura 5 – Fluxo para extração de palavras-chave



Fonte: O próprio autor (2022)

4.1 Aplicação do PLN em Python com NLTK e Spacy

A *Natural Language Tollkit* (NLTK) é uma biblioteca que foi criada no ano de 2001 pelo Departamento de Ciência da Computação e Informação da Universidade da Pensilvânia, sendo utilizada para construir programas Python, os quais trabalham com dados de linguagem humana, para aplicação em PLN. O NLTK define uma estrutura, a qual pode ser utilizada para a construção de programas de PLN em Python; fornece classes básicas para a representação de dados que sejam relevantes para o processamento da linguagem natural; interfaces padrão para a execução de tarefas como tokienização, análise sintática, *Parts-Of-Speech* e classificação textual (BARBOSA *et al.*, 2017).

A biblioteca Spacy, por sua vez, segundo o autor Ceccon (2020), foi desenvolvida para ser utilizada em ambientes de produção, e possibilita a criação de aplicativos que processam e "compreendem" grandes quantidades de texto. Seu uso inclui a extração de informações, entendimento da linguagem natural e pré-processamento de textos.

As suas tarefas empregam modelos linguísticos estatísticos, além de processamento independente, e incluem: tokienização, *Part-of-Spech*, determinação de dependência, lematização, detecção de sentenças, reconhecimento de entidades nomeadas, similaridade e classificação de textos.

4.1.1 Part-of-Speech

A tarefa de *Part-Of-Speech* (POS) está relacionada a determinação de um *token*, de forma a classificá-lo como verbo, advérbio, adjetivo ou outras classes gramaticais. A tarefa de *Part-Of-Speech*, quando realizada pela Spacy, possibilita a resolução de conflitos, por exemplo: decidir, dentro de um contexto, se a palavra "casa" é um substantivo (como em "ali está a minha casa") ou um verbo (como em "ele casa amanhã") (CECCON, 2020).

Segundo o autor Thiele (2015), a função principal dos POS é serem capazes de observar e catalogar as palavras de um texto de acordo com as funções morfossintáticas. Dentro do contexto dos POS, existe uma função de processamento e identificação de determinado grupo de palavras, as quais devem ser reunidas em tipos pré-definidos. Tal agrupamento acontece obedecendo a razões sintáticas, morfológicas ou morfossintáticas.

4.1.2 Lematização

O processo de lematização, faz a redução das flexões das palavras ao seu lema, ou seja, à sua raiz. Com isso, acontece uma simplificação, ou generalização dos resultados de algumas das tarefas de PLN que serão realizadas posteriormente.

4.1.3 Stop Words

As *Stop Words* são as palavras que um mecanismo de pesquisa foi programado para ignorar, tanto na indexação de entradas de pesquisa quanto na sua recuperação, como resultado de uma consulta de pesquisa. O objetivo principal das *Stop Words* é evitar que espaços em bancos de dados sejam ocupados por palavras desnecessárias, assim otimizando também o tempo de processamento (DUTTA, 2021). No "*Blog of an enthusiast*", é apresentada uma exata definição de *Stop Words*:

> Palavras que não aparecem no índice de uma base de dados específica porque são insignificantes (ou seja, artigos, preposições) ou tão comuns que os resultados seriam superiores ao que o sistema pode suportar (como no caso da IUCAT onde termos como Estados Unidos ou Departamento são palavras irrelevantes na pesquisa de palavras-chave.) As palavras irrelevantes variam de sistema para sistema. Além disso, alguns sistemas simplesmente ignorarão as palavras irrelevantes, em que o uso de palavras irrelevantes em outros sistemas resultará na recuperação de zero acertos.

É importante remover as *Stop Words*, porque elas não acrescentam nenhuma informação valiosa para as operações ou modelagem PLN, reduzindo o tamanho do texto e aumentando a robustez e o desempenho do modelo de PLN. Na biblioteca Spacy, é possível personalizar ou remover as palavras que fazem parte do grupo.

4.1.4 Tokienização

A tokienização é o processo que segmenta o texto em *tokens*, que constituem a forma mais elementar que ainda carrega algum significado. Este processo deve considerar muitos detalhes como, por exemplo, se uma pontuação determinada indica a separação de *tokens* ou não. Os modelos que estão disponíveis na biblioteca Spacy permitem que a tokienização ocorra de

forma especializada, o que gera resultados de alta qualidade para as tarefas seguintes (CECCON, 2020).

4.2 Serviços de PLN

Atualmente, existem diversos serviços pagos que utilizam a PLN, dentre eles estão: Google Cloud, IBM Watson, Amazon Comprehend e Microsoft Azure. O Google Cloud, segundo o site oficial do Google, propõe aos clientes gerar *insights* a partir de textos não estruturados, utilizando, para tanto, o *machine learning* do Google, e aplicar o PLN a aplicativos utilizando a API Natural Language (GOOGLE CLOUD, 2020).

A IBM Watson, por sua vez, oferece aos usuários serviços como análise de textos para extrair metadados de conteúdos como conceitos, entidades, categorias, palavras-chave, emoções, sentimentos, funções semânticas e relacionamentos, utilizando a compreensão da linguagem natural. O *Chief Executive Officer* (CEO) da empresa, Greg Wodlf, refere a ferramenta como uma grande potência, quando se trata de processamento de linguagem natural (IBM, 2015).

A Amazon (2021), conforme informações disponíveis no site oficial, utiliza, para o processamento de linguagem natural, o *machine learning*, para gerar *insights* e relações nos textos. Ainda, as tarefas de PLN deste serviço incluem análise de sentimentos, modelagem de tópicos, reconhecimento de entidades, dentre outros.

A Microsoft Azure constitui outro serviço pago para o processamento de linguagem natural, utilizando análise textual, tradução, e serviços de reconhecimento vocal para facilitar a criação de aplicativos, os quais são suporte à idiomas naturais. Tal serviço, baseado em nuvem, oferece um processamento de linguagem natural avançado, analisando os textos brutos para determinar sentimentos, extrair frases-chave, reconhecer entidades nomeadas e detectar idiomas (MICROSOFT, 2021).

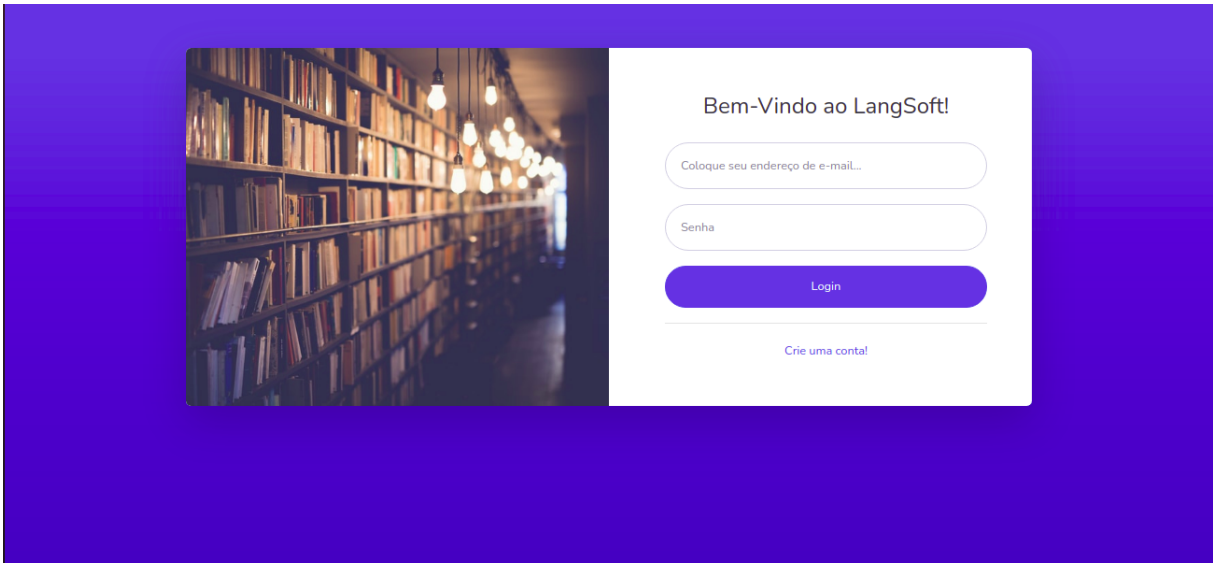
5 SISTEMA PARA GERENCIAMENTO DE METADADOS DE OA'S

O sistema foi desenvolvido com um único *backend*, porém utilizou-se duas abordagens diferentes para implementá-lo. Uma abordagem desenvolvida para atender a interface gráfica, criada com Jinja2, e outra abordagem que visou desacoplar o sistema com a construção da API pública. Ambas funcionam em conjunto e essa seção descreverá tanto o funcionamento do sistema retratando a experiência do usuário que vai utilizar a interface gráfica criada, quanto os Os detalhes da implementação das duas abordagens que serão discutidos na subseção de arquitetura e desenvolvimento.

5.1 Gestão da conta do usuário

O sistema foi construído para ser utilizado por várias pessoas com interesse nos OAs, porém, para que a pessoa consiga interagir com a aplicação, é necessário que se registrem. Para isso, o utilizador pode criar uma conta. Os requisitos são: inserir o nome, o e-mail e uma senha. Com a conta criada, o usuário é capaz de entrar no sistema sempre que desejar inserindo seu e-mail e senha. Ele também poderá editar suas informações ou excluir sua conta, se assim desejar, na área do perfil do usuário. A Figura 6 mostra a tela para autenticação.

Figura 6 – Tela para autenticação do usuário



Fonte: O próprio autor (2022)

5.2 Gestão de um metadado

Assim que o usuário entra no sistema ele é direcionado para a página principal, onde são listados, por ordem de inclusão, todos os metadados de Objetos de Aprendizagem cadastrados. Para consultar um metadado, o utilizador deverá selecionar o objeto escolhido, sendo assim, é aberta uma página com todas as informações do mesmo. Ele também pode editar ou excluir aquele objeto.

Quando o usuário seleciona a opção “novo” no programa, ele é direcionado para a página onde poderá escolher um vídeo do Youtube para adicionar como Objeto de Aprendizagem. O utilizador digita um termo de sua preferência e escolhe, dentre a lista de vídeos gerada, qual deseja adicionar ao sistema. Quando escolhido, o sistema converte as informações daquele vídeo para o formato IEEE LOM com as extensões CLEO e o salva para que seja utilizado como objeto de aprendizado. A Figura 7 retrata um Objeto de Aprendizagem salvo no sistema.

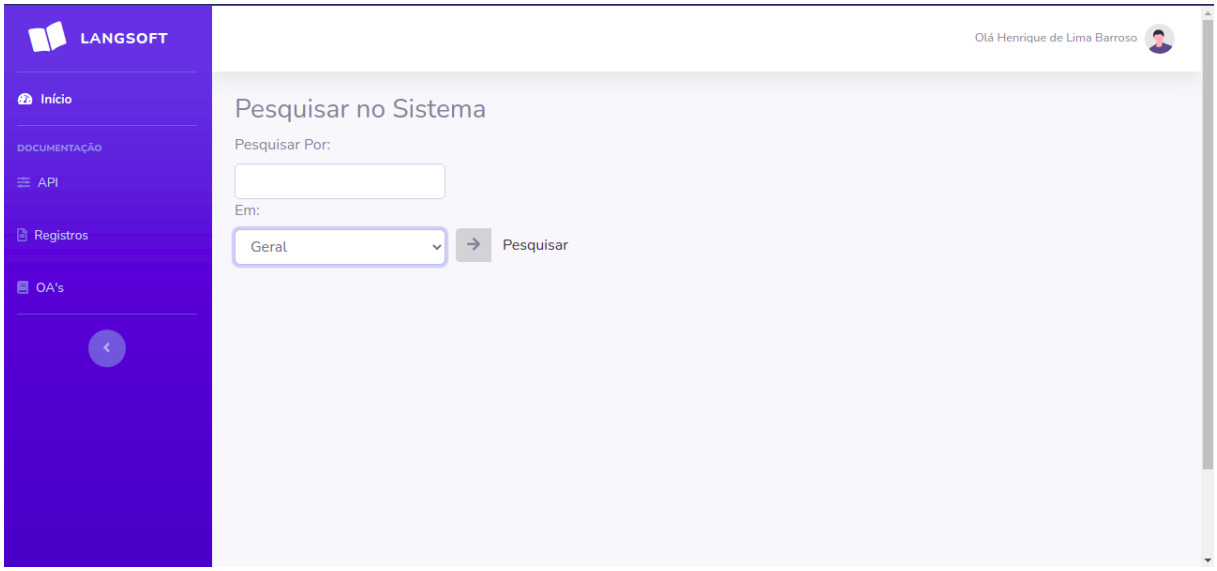
Figura 7 – Tela de visualização das informações de um OA

Fonte: O próprio autor (2022)

5.3 Pesquisa

Um dos objetivos da padronização de OAs é facilitar a aprendizagem e produtividade dos alunos e professores, para isso é fundamental que estes consigam encontrar com facilidade um OA a partir de uma filtragem. Para fazer uma filtragem no sistema é necessário escolher a opção de "pesquisa" na tela inicial. Depois, escolher um termo e uma categoria. Por exemplo, se a pessoa procura por conteúdo em um idioma específico, ela pode digitar o idioma escolhido e procurar na categoria geral, que é a categoria que carrega a informação do idioma do OA. A Figura 8 mostra a tela de pesquisa.

Figura 8 – Tela de pesquisa

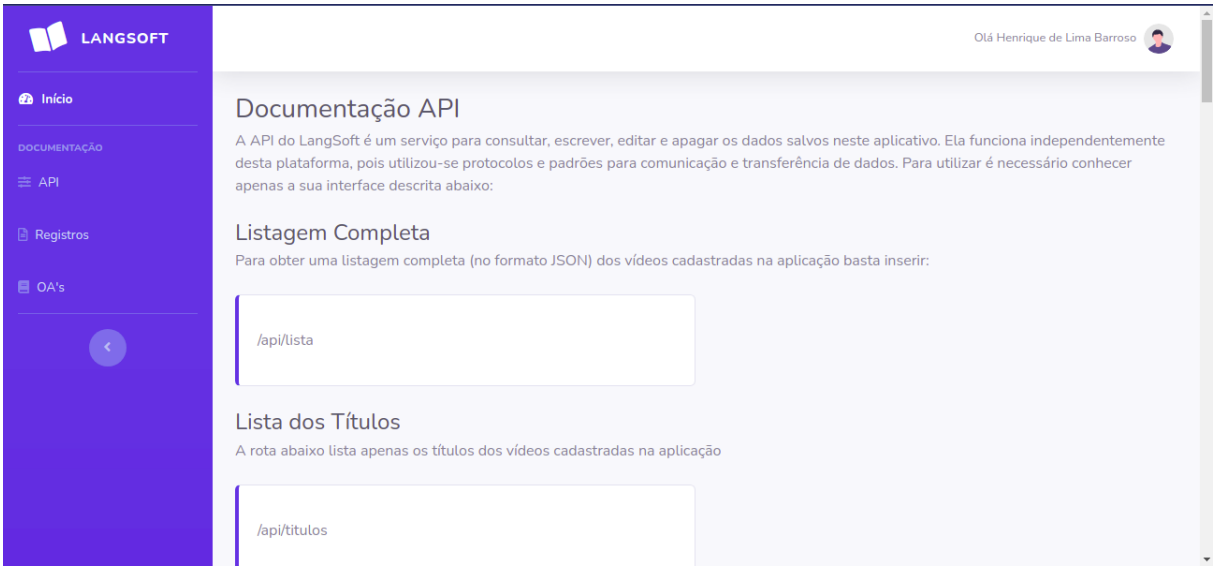


Fonte: O próprio autor (2022)

5.4 Documentações

O sistema possui três páginas de documentação. A documentação da API pública, que mostra aos desenvolvedores quais as rotas disponíveis e as instruções para utilizá-las. A documentação de registros, onde são mostradas as alterações feitas no sistema como a edição ou exclusão de um OA e a documentação sobre o formato LOM com a extensão CLEO para o usuário se orientar quando for utilizar um Objeto de Aprendizagem. As documentações podem ser acessadas através da barra lateral. A Figura 9 mostra a tela de documentação da API.

Figura 9 – Tela de documentação



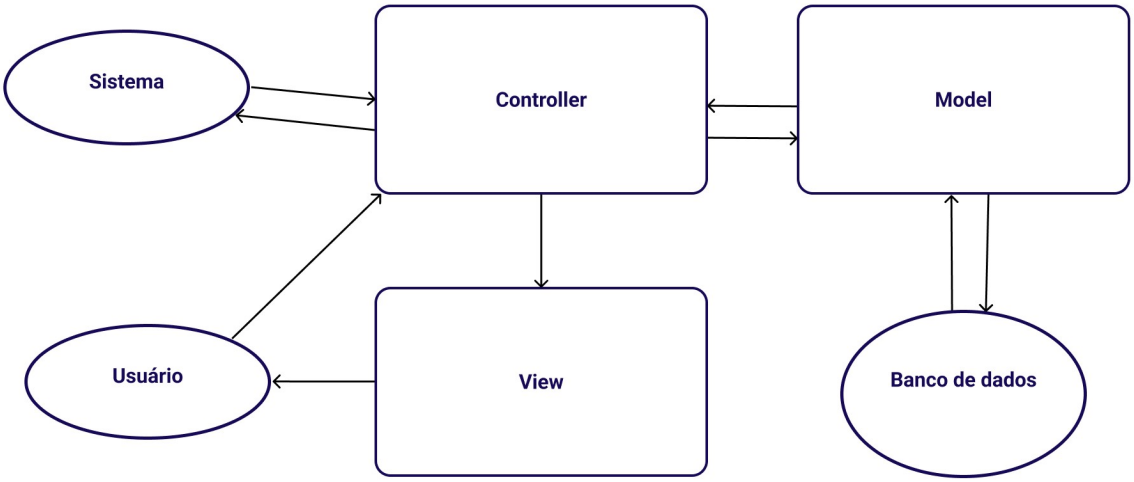
Fonte: O próprio autor (2022)

5.5 Arquitetura e Desenvolvimento

O sistema foi desenvolvido apenas com ferramentas de código aberto para que os recursos utilizados pudessem ser personalizados e controlados para incentivar a colaboração e transparência. Dessa forma, foi utilizado o Ubuntu, versão 20.04, como sistema operacional, Visual Studio Code, versão 1.64, para editar o código fonte e o Git, versão 2.35, para controlar as versões do sistema. A linguagem de programação escolhida foi o Python, versão 3.1 e o Flask, versão 1.1, foi o micro-framework adotado.

Este sistema foi construído para ser utilizado de forma colaborativa, então, os usuários vão ter acesso aos dados de todos os OAs adicionados, facilitando a recuperação e utilização destes. Para disponibilizar essas informações, uma interface gráfica foi criada, mas também foi desenvolvida uma API pública para integração de outros sistemas posteriormente. A API reforça a interoperabilidade, possibilitando que os OAs sejam utilizados em diferentes sistemas, assim favorecendo a portabilidade entre aplicativos e plataformas. Isso significa que o sistema é capaz de responder requisições de outros aplicativos e que os usuários destes, poderão usufruir das funcionalidades criadas neste trabalho. A Figura 10 representa o sistema detalhando as suas camadas.

Figura 10 – Fluxo de funcionamento do sistema



Fonte: O próprio autor (2022)

Utilizou-se o padrão de arquitetura de software: Model View Controller (MVC) e se baseia no fluxo mostrado pela figura 10. A camada *model* contém o modelo de negócio da aplicação e gerencia o comportamento dos dados de OAs, registros e contas. Todos os novos objetos do sistema são criados a partir dessa camada, onde os valores são preenchidos conforme as informações enviadas da *controller*. A *model* também guarda as constantes do sistema e as funcionalidades para processamento de linguagem natural, manipulação de formulários e acesso a API do Youtube. A *controller* é responsável por controlar as requisições que neste caso serão feitas através da interface gráfica do próprio sistema ou de aplicações terceiras, feitas para a API pública que também integra essa camada. Nela, se tem diferentes *endpoints* e diferentes funções para cada uma dessas requisições. Por fim, a camada *view* é responsável por mostrar as informações de forma estilizada para o usuário. Ela concentra todo o código de *frontend* desse sistema e seu funcionamento consiste em receber dados da *controller*, organizá-los e enviá-los para o cliente, o intuito é que o usuário consiga manipular esses dados de forma intuitiva. A API pública não utiliza essa camada pois não tem *frontend*.

5.5.1 Interface gráfica

Para representar os elementos e funcionalidades propostas para os utilizadores, com uma boa interface do usuário, utilizou-se o SB Admin em conjunto com a biblioteca Jinja2. SB Admin é um tema de código aberto, baseado em um *framework* para *frontend* chamado Bootstrap 4. Ele foi criado pela empresa Start Bootstrap. O tema usa a licença criada pelo Instituto de Tecnologia de Massachusetts, MIT, o que significa que pode ser utilizado para qualquer finalidade, inclusive comercial.

O código da interface foi concentrado na camada *view*. Foram criadas telas para a maior parte das funcionalidades. É possível criar e gerenciar contas de usuário e OAs, porém a funcionalidade de obter as palavras-chave dos últimos 100 comentários de um vídeo, ainda está disponível apenas na API pública.

5.5.2 Banco de dados

Para persistir os dados utilizou-se o MongoDB na versão 3.6. Este banco armazena e disponibiliza os OAs de forma individual na Web para serem acessados simultaneamente por um grande número de usuários. Além dos OAs, as informações das contas dos usuários e registros também são armazenados neste banco. A principal vantagem considerada ao escolher o MongoDB é a velocidade, já que é necessário viabilizar a utilização do OAs sem que haja a necessidade de recriá-los, por muitos usuários.

5.5.3 Autenticação

O sistema usa um controle de acesso para permitir que apenas usuários com uma conta criada possam acessar e fazer alterações na aplicação. Para conceder estes privilégios para um grupo restrito, utilizou-se a biblioteca Flask-Login. Com ela é possível gerenciar sessões de usuário no sistema e automatizar tarefas comuns de autenticação com *login* e *logout*. O Flask-Login também: armazena a identificação do usuário ativo na sessão de forma segura, restringe páginas a usuários cadastrados e integra as funcionalidades de autenticação a outras do sistema.

Para registrar um usuário no sistema, o Flask-Login disponibiliza a função *login_user*. Ela recebe como parâmetro um objeto que retrata o usuário a ser registrado, neste sistema as propriedades utilizadas para definir um usuário foram: nome, e-mail e senha. Para que uma conta de usuário seja criada, o sistema verifica no banco de dados se não existe um usuário com mesmo e-mail informado e se as senhas digitadas em dois campos são iguais. Se sim, esse objeto é passado como parâmetro para a função *login_user*. Depois de logado o usuário é redirecionado para página inicial. O trecho de código da Figura 11 representa o comportamento citado.

Figura 11 – Função para registrar usuário

```
if usuario:
    is_pass_ok = check_password_hash(usuario['password'], password)
    if is_pass_ok:
        user = User(usuario['name'], usuario['email'],
        usuario['password'])
        login_user(user)
        return redirect(url_for("index"))
    else:
        error = 1
else:
    error = 1
```

Fonte: O próprio autor (2022)

Para logar um usuário no sistema, inicialmente é verificado se o e-mail informado existe no banco de dados e se a senha digitada é a mesma salva com este e-mail. Se as informa-

ções estiverem coerentes é chamada a função *login_user*. Caso as verificações não satisfaçam as condições citadas, o usuário receberá uma mensagem de erro em tela e não conseguirá se registrar nem realizar *login*. Para fazer o *logout* do usuário, utiliza-se a função *logout_user*, também da biblioteca Flask_Login, não é necessário passar nenhum parâmetro pois só existe um usuário logado por vez.

Depois de logado, os dados do usuário são salvos pelo Flask-Login. Além dos valores de nome, e-mail e senha, o objeto com as informações do usuário possui funcionalidades que são definidas na classe User. Essas funcionalidades são pré definidas pela Flask-Login e são fundamentais para o seu bom funcionamento. Nessa classe existe o método *get_id* que retorna o identificador do usuário que neste caso foi definido para ser o e-mail e o método *get* que é um método estático, ele retorna um usuário do banco de dados de acordo com o identificador passado como parâmetro. A implementação deles é mostrada na Figura 12.

Figura 12 – Métodos da classe User

```
1  def get_id(self):
2      query = db.filtrar('users', {"email": self.email})
3      user_bd = query[0]
4      return str(user_bd['email'])
5
6  @staticmethod
7  def get(user_id):
8      query = db.filtrar('users', {"email": user_id})
9      if query:
10         user_bd = query[0]
11         user = User(user_bd['name'], user_bd['email'],
12         user_bd['password'])
13     else:
14         user = None
15     return user
```

Fonte: O próprio autor (2022)

Para garantir que somente usuários autenticados acessem rotas restritas, a biblioteca disponibiliza o *decorator* chamado *login_required*. Ele restringe uma funcionalidade apenas a usuários registrados, caso o usuário não esteja autenticado, a função vai retornar um status HTTP com código 401. Isso significa que o cliente tentou acessar um recurso ao qual ele não está autorizado. Neste sistema toda resposta com este código, redireciona o usuário para a página de *login*. Na Figura 13 é possível ver a aplicação do *login_required* em umas das rotas do sistema.

Figura 13 – Aplicação do *login_required*

```
1  @app.route("/")
2  @login_required
3  def index():
4      videos = db.list('objetoAprendizagem')
5      return render_template('index.html', videos=videos)
```

Fonte: O próprio autor (2022)

5.5.4 API do Youtube

A API do Youtube foi utilizada para requisitar as informações de determinados vídeos para que sejam convertidas em metadados de OAs. Para isso, inicialmente é feita uma busca para que o usuário encontre o vídeo que procura. Para fazer essa busca, o usuário digita um termo de sua preferência e então a API retorna uma lista com informações de, no máximo, 20 vídeos. Essas informações são organizadas e mostradas no sistema para que o usuário escolha um vídeo. Depois de selecionado um vídeo, uma nova requisição é realizada, dessa vez para trazer as informações do vídeo escolhido que vão ser a base dos metadados do Objeto de Aprendizagem.

Antes de usar as funcionalidades que a API provê, foi necessário acessar o Console de APIs do Google e solicitar uma chave de API, pois o Google só envia solicitações de API para clientes autenticados. Para fazer as requisições, o sistema tem uma classe chamada Youtube, onde foi configurada a versão da API do Youtube que o sistema irá usar, o nome do serviço e a chave de API. A partir disso, criou-se os métodos necessários para requisitar os dados de vídeos.

As funções que acessam a API pública do Youtube implementadas nessa classe, utilizam operações do tipo *list*, essas operações recuperam uma lista vazia ou com recursos. Dessas funções, duas são chamadas durante o fluxo de manipulação de metadados de OAs. Uma delas é a função “buscarListaVideos” que recebe como parâmetro um termo (palavra ou frase) e retorna, de acordo com esse termo, uma lista de objetos com informações de vídeos do Youtube. A outra função, “retornarVideo”, recebe um identificador de um vídeo e retorna informações básicas dele, como título, descrição e os comentários feitos. Na figura 14 pode-se conferir a função ”buscarListaVideos”.

Figura 14 – Função “buscarListaVideos”

```
def buscarListaVideos(self, termo):
    listaVideos = None
    listaVideosApi = self.apiYoutube.search().list(
        q=termo,
        part="id, snippet",
        maxResults=20,
        type='video',
    ).execute()
    listaVideos = []
    for infoVideo in listaVideosApi.get("items", []):
        listaVideos.append({"id": infoVideo['id']['videoId'],
        "titulo": infoVideo['snippet']['title'],
        "img": infoVideo['snippet']['thumbnails']['default']['url']})
    return listaVideos
```

Fonte: O próprio autor (2022)

Nessa função se utilizou a operações “search” da API do Youtube. Por padrão ela retorna um conjunto que identifica recursos de vídeos, canais e *playlists*, mas com a utilização do parâmetro “type”, especificou-se que os únicos recursos requeridos devem ser do tipo: vídeo. Outros parâmetros usados foram: “maxResults”, “q” e “part”. “MaxResults” define o número máximo de itens trazidos por requisição, “q” determina o termo de consulta a ser pesquisado e o parâmetro “part” especifica uma lista de recursos a serem requisitados, os nomes id e snippet podem ser inseridos no valor do parâmetro, sendo que “id” é o identificador do vídeo e “snippet” são informações básicas como o título, a descrição e imagens. No final da função, as informações buscadas são organizadas de forma relevante para o propósito definido.

5.5.5 Modelagem das informações de um Objeto de Aprendizagem

A partir da seleção do vídeo e da disponibilização das suas informações pela API do Youtube, o sistema modela o Objeto de Aprendizagem no formato IEEE LOM com as extensões CLEO. Alguns campos do OA são preenchidos automaticamente com as informações do vídeo, por exemplo: identificador, título, descrição, imagem e data de publicação, enquanto os outros campos são criados, mas deixados vazios para que os usuários possam preencher conforme a necessidade. Vários objetos que retratam OAs podem ser criados a partir de um mesmo vídeo, eles são criados de forma independente.

O procedimento de criação de metadados de um OA se inicia a partir da requisição feita através da API pública ou da interface gráfica do sistema. A função desenvolvida para atender a interface recebe o identificador do vídeo como parâmetro, busca informações dele utilizando a API do Youtube, cria um objeto de metadados e preenche alguns campos com as informações trazidas, salva esse objeto no banco de dados, registra a atividade nos históricos

do sistema e devolve uma mensagem de sucesso ao usuário com o título do vídeo. Caso ocorra algum erro neste procedimento, o sistema poderá também enviar uma mensagem de erro. O código da figura 15 retrata essa operação.

Figura 15 – Função para adicionar vídeo

```
@app.route("/adicionar/<videoId>", methods=['POST', 'GET'])
@login_required
def adicionarVideo(videoId):
    youtube = Youtube()
    video = None
    try:
        video = youtube.retornarVideo(videoId)
    except:
        flash("Erro ao buscar informações sobre vídeo na api")
    learningObject = LearningObject(video)
    db.inserir("objetoAprendizagem", learningObject)
    reg = Registro()
    reg.registrarVideoAdicionado(videoId)
    return render_template('adicionado.html',
    tituloVideo=learningObject.geral['titulo'])
```

Fonte: O próprio autor (2022)

Existem funções exclusivas para gerenciar metadados através da API pública e da interface gráfica, uma vez que elas precisam retornar diferentes propriedades, porém independente de onde venha a requisição, o padrão de metadados criado deve ser o mesmo. Por isso, existe uma única classe na camada *model* em que os metadados são criados, chamada LearningObject.

5.5.6 Registros

Como esse sistema é colaborativo, é importante que se tenha controle das ações dos usuários. Os registros foram desenvolvido com este objetivo, sem bibliotecas externas. Com ele é possível analisar as atividades dos utilizadores criando um registro para cada modificação que o usuário faz na aplicação, onde é detalhado o que foi feito, por quem e quando. Todas as funções que fazem alteração no banco de dados do sistema instanciam a classe de registros situada na camada *model* e chamam um método adequado para gravar essas informações no histórico. Essa documentação é disponibilizada para todos os utilizadores no próprio sistema, podendo ser acessada pelo menu lateral.

5.5.7 Geração de Palavras-Chave

‘‘Palavras-chave’’ é um campo que compõe os metadados de OAs criados por este sistema. Pensando nisso e na importância das palavras-chave para sumarizar um Objeto de Aprendizagem, foi criada essa funcionalidade com a utilização do processamento de linguagem natural. Como a descrição dos vídeos feitas pelos autores são normalmente curtas, objetivas

e em muitas das vezes contém propagandas e assuntos não relacionados ao vídeo em questão, optou-se por criar a funcionalidade de gerar as palavras-chave a partir dos comentários dos vídeos.

A biblioteca usada para gerar as palavras-chave foi a Spacy porque tem suporte para a língua portuguesa. A primeira operação a ser realizada tem o objetivo de obter uma lista contendo todas as palavras escritas nos últimos 100 comentários vídeo. A tarefa de Part-of-Speech foi utilizada nessa fase em que se exclui *tokens* indesejados, como: pontuação, símbolos, espaços e números. Se remove também palavras formadas por uma única letra que se repete, por exemplo, uma utilizada nos comentários é a palavra informal para representar risos, onde se repete a letra "k" várias vezes. A imagem 16 mostra a função que realiza a tarefa de Part-of-Speech.

Figura 16 – Operação Part-of-Speech

```
1  def obterPalavras(self, comentarios):
2      palavras = []
3      comentariosPln = self.obterDocumentoPln(comentarios)
4      for token in comentariosPln:
5          if token.pos_ not in ['PUNCT', 'SIMB', 'X',
6          'EOL', 'SPACE', 'PART',
7          'NUM'] and not self.identificarStringUnicaLetra(token.text):
8              palavras.append(token.text)
9      return palavras
```

Fonte: O próprio autor (2022)

Na sequência, utilizou-se as *stop words* oferecidas pela Spacy para remover palavras indesejadas e ainda, foi necessário estender este conjunto com palavras informais muito utilizadas nos comentários, exemplo "pra" e "pro". Por fim, foi feito um procedimento que conta quantas vezes cada palavra aparece, aquelas que se repetem com maior frequência são agrupadas e devolvidas ao usuário. A função da imagem 17 demonstra esse procedimento.

Figura 17 – Método para gerar palavras-chave

```
def obterPalavrasChave(self, comentarios, quantidade):
    palavras = self.obterPalavras(comentarios)
    palavrasSemStopWords = self.removerStopWords(palavras)
    contadorPalavras = []
    documento = " ".join(palavra for palavra in palavrasSemStopWords)
    for palavra in palavrasSemStopWords:
        palavraIncluida = False
        for objeto in contadorPalavras:
            if objeto['palavra'] == palavra:
                palavraIncluida = True
                break
        if not palavraIncluida:
            contadorPalavras.append(
            {"palavra": palavra,
            "ocorrencias": documento.count(' ' + palavra + ' ')})
    contadorPalavras.sort(key=lambda objeto: objeto['ocorrencias'])
    palavrasChave = []
    for i in range(quantidade):
        if contadorPalavras:
            if contadorPalavras[-1]['ocorrencias'] > 1:
                palavrasChave.append(
                contadorPalavras.pop()['palavra'])
        else:
            break
    return(palavrasChave)
```

Fonte: O próprio autor (2022)

Essa funcionalidade está disponível na API pública da aplicação. O campo "palavras-chave" dos OAs não é preenchido automaticamente com as palavras-chave geradas, porque o objetivo dessa ferramenta é apenas facilitar o processo de criação. O ideal é que elas possam ser selecionadas pelo utilizador levando em consideração também outros aspectos do OA, uma vez que as palavras-chave são geradas apenas a partir dos comentários do vídeo.

Como o sistema foi desenvolvido apenas com ferramentas livres de custos, optou-se por não se utilizar os serviços de processamento de linguagem natural já testados e aprovados existentes no mercado fornecidos pelas empresas Google, AWS, Microsoft e IBM.

5.5.8 Qualidade do código

Todos métodos e variáveis foram criados com nomes significativos, respeitou-se as boas práticas da orientação a objetos e implementou-se um padrão de arquitetura de software preparando o sistema para receber evoluções e melhorias.

6 CONSIDERAÇÕES FINAIS

Após o desenvolvimento do sistema e alguns dias de testes, foi possível constatar a organização dos estudos trazida por ele, por meio da criação de metadados de OAs no formato IEEE com as extensões CLEO, tornando mais objetivos os aprendizados adquiridos através de vídeos do Youtube, visto que a plataforma disponibiliza um grande volume de conteúdo dos mais diversos assuntos.

Isso foi possível, no tempo delimitado, porque recorreu-se a utilização de uma linguagem de programação de alto nível e de código aberto, o que trouxe facilidade para se encontrar documentações e tutoriais, poupando tempo na solução de falhas. Foi constatado o bom funcionamento do Python e do micro-framework Flask em conjunto com o padrão de arquitetura MVC e a API pública disponibilizada pelo Youtube.

Obteve-se deste modo o serviço em conformidade com os objetivos propostos, podendo ser utilizado gratuitamente mediante a execução do serviço em um servidor local ou através da implementação em ambiente de produção para ser disponibilizado para diversos usuários via *Web*.

Para continuação deste projeto, sugere-se a criação de seções específicas por disciplina no sistema, isso vai relacionar e combinar os OAs automaticamente, elaborando módulos instrucionais. Outra evolução sugerida para este serviço é modificá-lo para que tenha uma API única que atenda tanto a interface gráfica deste sistema quanto à aplicações terceiras.

REFERÊNCIAS

AGUIAR, J. *et al.* Análise comparativa de abordagens de associação entre os estilos de aprendizagem de Felder-Silverman e os metadados do padrão IEEE LOM. In: **IV Workshop de Desafios da Computação Aplicada à Educação**. 2015. 10 p. Disponível em: /<https://sol.sbc.org.br/index.php/desafie/article/view/10038>. Acesso em: 06 jan. 2022.

AMAZON. **Amazon Comprehend**. 2021. Disponível em: /<https://aws.amazon.com/pt/comprehend>. Acesso em: 06 dez. 2021.

ASLAM, F. A. *et al.* **Efficient way of web development using python and flask**. 2015. International Journal Of Advanced Research In Computer Science, [S.L.], v. 6, n. 2, p. 54-57, mar. 2015. Disponível em: /<https://core.ac.uk/download/pdf/55305148.pdf>. Acesso em: 24 out. 2021.

ASSOCIAçãO PYTHON BRASIL. **Ferramentas de desenvolvimento**. 2021. Disponível em: /<https://python.org.br/>. Acesso em: 13 fev. 2022.

BARBOSA, J. L. N. *et al.* **Introdução ao Processamento de linguagem natural usando Python**. 2017. III Escola Regional de Informática do Piauí, [S.L.], v. 1, n. 1, p. 336-360, jun. 2017.

BOGHI, C. **Ranking de Sistemas de Bancos de Dados mais usados em 2020/2021**. 2021. Disponível em: /<https://www.redetv.uol.com.br/colunistas/desvendando-a-tecnologia/ranking-de-sistemas-de-bancos-de-dados-mais-usados-em-2020-2021>. Acesso em: 15 fev. 2022.

BOICEA, A. *et al.* **MongoDB vs Oracle: database comparison**. 2012. Ieee, [S.L.], v. 1, n. 1, p. 330-335, jan. 2012. Disponível em: /<https://ieeexplore.ieee.org/abstract/document/6354766>. Acesso em: 01 dez. 2021.

BRANCO, D. C. **Qual é a linguagem de programação mais usada no mundo atualmente?** 2021. Disponível em: /<https://canaltech.com.br/mercado/qual-e-a-linguagem-de-programacao-mais-usada-no-mundo-atualmente-200857/>. Acesso em: 13 jan. 2022.

BRANDãO, B. **Você sabe o que é API de integração? Entenda de uma vez por todas!** 2020. Disponível em: /<https://maplink.global/blog/o-que-e-api/#:~:text=Trello-,O%20que%20%C3%A9%20API%20Privada,acessadas%20pelos%20seus%20pr%C3%B3prios%20desenvolvedores>. Acesso em: 16 fev. 2022.

CARVALHO, V. C. de. **OntAES**: uma ontologia para sistemas adaptativos educacionais baseada em objetos de aprendizagem e estilos de aprendizagem. 2017. 73 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, 2017. Disponível em: /<http://repositorio.ufu.br/handle/123456789/21440>. Acesso em: 20 nov. 2021.

CECCON, D. **Spacy**: processamento de linguagem natural avançado em python. processamento de linguagem natural avançado em Python. 2020. Disponível em: /<https://iaexpert.academy/2020/02/03/spacy-processamento-de-linguagem-natural-avancado-em-python/>. Acesso em: 20 nov. 2021.

COSTA, M. J. M. *et al.* Bibliotecas e Repositórios de Objetos de Aprendizagem: potencialidades para o processo de aprendizagem. **Revista Tecnologias na Educação**, [S.L.], v. 22, n. 22, p. 1-16, out. 2017. Disponível em: /<http://tecedu.pro.br/wp-content/uploads/2017/10/Art16-vol.22-Edi%C3%A7%C3%A3o-Tem%C3%A1tica-VI-Outubro-2017.pdf>. Acesso em: 06 jan. 2022.

COSTA, R. L. de C. **SQL**: guia prático. 2006. Rio de Janeiro: Brasport, 2006. 231 p.

DUTTA, A. J. **Removing stop words with NLTK in Python**. 2021. Disponível em: /<https: //www.geeksforgeeks.org/removing-stop-words-nltk-python>. Acesso em: 01 nov. 2021.

FONTANA, M. V. *et al*. **Objetos de Aprendizagem de autoria coletiva: uma concepção possível na ead**. 2019. Disponível em: /<https://www.seer.ufrgs.br/InfEducTeoriaPratica/article/view/86624>. Acesso em: 08 fev. 2022.

GHIMIRE, D. **Comparative study on Python web frameworks**: flask and django. 2020. 40 f. Tese (Doutorado) - Curso de Engenharia de Mídia, Metropolia University Of Applied Sciences, [S.L.], 2020. Disponível em: /<https://www.theseus.fi/bitstream/handle/10024/339796/Ghimire_Devndra.pdf?sequence=2>. Acesso em: 30 out. 2021.

GOMES, S. R. *et al.* Objetos de Aprendizagem funcionais e as limitações dos metadados atuais. In: **XVI Simpósio Brasileiro de Informática na Educação**. 2005. 10 p. Disponível em: /<http: //br-ie.org/pub/index.php/sbie/article/view/406/392>. Acesso em: 04 jan. 2022.

GOOGLE CLOUD. **Natural Language AI**. 2020. Disponível em: /<https://cloud.google.com/natural-language>. Acesso em: 06 dez. 2021.

IBM. **Entendendo a linguagem natural do Watson**. 2015. Disponível em: /<https: //www.ibm.com/br-pt/cloud/watson-natural-language-understanding>. Acesso em: 06 dez. 2021.

IBM. **Modelos Jinja2**. 2021. Disponível em: /<https://www.ibm.com/docs/pt-br/qsip/7.4?topic=1-jinja2-templates>. Acesso em: 15 fev. 2022.

IEEE. **1484.12.1-2020 - Padrão IEEE para metadados de objetos de aprendizagem**. 2020. Disponível em: /<https://ieeexplore.ieee.org/document/9262118>. Acesso em: 20 nov. 2021.

JUNIOR, C. F. B. **Reúso de conteúdo da Web na Recomendação Personalizada de Objetos de Aprendizagem**: uma abordagem baseada em um algoritmo genético, tecnologias da web semântica e uma ontologia. 2018. 98 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Uberlândia, Uberlândia, 2018. Disponível em: /<http: //repositorio.ufu.br/bitstream/123456789/23294/1/ReusoConteudoWeb.pdf>. Acesso em: 15 nov. 2021.

JUNIOR, L. F. D. **MongoDB para iniciantes em NoSQL**. 2017. Disponível em: /<https: //imasters.com.br/banco-de-dados/mongodb-para-iniciantes-em-nosql>. Acesso em: 15 fev. 2022.

JUNQUEIRA, R. de P.; LOSCIO, B. F. Repositórios de Objetos de Aprendiagem: uma análise comparativa com ênfase no reuso de conteúdos. In: **III Congresso Brasileiro de Informática na Educação**. 2014. Disponível em: /<https://br-ie.org/pub/index.php/sbie/article/view/3039/2550>. Acesso em: 04 jan. 2022.

LABAKI, J. **Introdução a Phyton**: módulo a. Módulo A. 2016. Disponível em: /<https: //dcc.ufrj.br/~fabiom/python/pythonbasico.pdf>. Acesso em: 29 nov. 2021.

LOPES, P. J. O. **Criação e evolução de uma API pública**. 2018. 153 f. Dissertação (Doutorado) - Curso de Engenharia da Informática, Instituto Superior de Engenharia do Porto, Porto, 2018. Disponível em: /<https://wiki.python.org.br/BancosDeDadosSql>. Acesso em: 01 nov. 2021.

MACEDO, C. M. S. de. **Diretrizes para a criação de Objetos de Aprendizagem acessíveis**. 2010. 271 f. Tese (Doutorado) - Curso de Engenharia e Gestão de Conhecimento, Universidade Federal de Santa Catarina, Florianópolis, 2010. Disponível em: /<https://repositorio.ufsc.br/bitstream/handle/123456789/94396/288186.pdf?sequence=1&isAllowed=y>. Acesso em: 05 jan. 2022.

MACHION, A. C. G. **Uso de ontologias e mapas conceituais na descoberta e análise de objetos de aprendizagem**: um estudo de caso em eletrostática. 2007. 140 f. Tese (Doutorado) - Curso de Ciências da Computação, Universidade de São Paulo, São Paulo, 2007. Disponível em: /<https://www.teses.usp.br/teses/disponiveis/45/45134/tde-06042009-122508/pt-br.php>. Acesso em: 15 nov. 2021.

MENDES, M. M. *et al.* Agrupamento e recomendação de Objetos de Aprendizagem no padrão IEEE-LOM considerando estilos de aprendizagem. In: **VI Congresso Brasileiro de Informática na Educação**. 2017. 10 p. Disponível em: /<http://br-ie.org/pub/index.php/sbie/article/view/7650/5446>. Acesso em: 05 jan. 2022.

MENEZES, N. **Acesso a banco de dados SQL**. 2015. Disponível em: /<https://wiki.python.org.br/BancosDeDadosSql>. Acesso em: 25 nov. 2021.

MICROSOFT. **Explorar o processamento de idioma natural**. 2021. Disponível em: /<https://aws.amazon.com/pt/comprehend>. Acesso em: 07 dez. 2021.

MOGLICH, A. I. **Einführung in MongoDB**. 2011. Javaspektrum, [S.L.], v. 1, n. 1, p. 1-4, jan. 2011. Disponível em: /<https://www.sigs-datacom.de/uploads/tx_dmjournals/alvermann_JS_01_11.pdf>. Acesso em: 01 dez. 2021.

PYTHON INSTITUTE. **What is Python?** 2017. Disponível em: /<https://pythoninstitute.org/what-is-python/>. Acesso em: 20 nov. 2021.

RESENDE, D. T. *et al.* Em direção à recuperação automática de Objetos de Aprendizagem em repositórios através da associação dos estilos de aprendizagem de estudantes com metadados no padrão IEEE-LOM. In: **III Congresso Brasileiro de Informática na Educação**. 2014. 10 p. Disponível em: /<http://br-ie.org/pub/index.php/wcbie/article/view/3265/2805>. Acesso em: 04 jan. 2022.

ROMERO, I. J. *et al.* Evaluation of a virtual learning object with augmented reality technology for teaching of the computer parts**. Iop Conference Series:**Materials Science and Engineering. 2021. Disponível em: /<https://iopscience.iop.org/article/10.1088/1757-899X/1154/1/012019/meta>. Acesso em: 08 fev. 2022.

SAMPSON, D.; ZERVAS, P. Learning Object Repositories as Knowledge Management Systems. In: **Knowledge Management E-Learning,**University of Hong Kong, Hong Kong. 2013. V. 5, n. 2, p. 117-136. Disponível em: /<http://www.kmel-journal.org/ojs/index.php/online-publication/article/view/199>. Acesso em: 11 fev. 2022.

SAS. **Processamento de linguagem natural**: o que é e qual a sua importância? 2021. Disponível em: /<https://www.sas.com/pt_br/insights/analytics/processamento-de-linguagem-natural.html>. Acesso em: 02 dez. 2021.

SILVA, E. L. *et al.* Os objetos educacionais, os metadados e os repositórios na sociedade da informação. **Ciência da Informação,**[S.L.], v. 39, n. 3, p. 93-104,

dez. 2010. FapUNIFESP (SciELO). 2010. Disponível em: /<https://www.scielo.br/j/ci/a/y3TDqgmMh3xJB8GcNVphRhw/?lang=pt>. Acesso em: 11 fev. 2022.

SILVA, J. W. F.; SOUZA, C. T. **. Repositórios de Objetos de Aprendizagem: características; classificações; limitações e tendências. In:** VI Congresso Brasileiro de Informática na Educação. 2017. Disponível em: /<https://www.br-ie.org/pub/index.php/sbie/article/view/7535>. Acesso em: 11 fev. 2022.

SOUTH, J. B.; MONSON, D. W. **A university-wide system for creating, capturing, and delivering learning objects**. 2000. Disponível em: /<https://docsbay.net/a-university-wide-system-for-creating-capturing>. Acesso em: 06 jan. 2022.

TALIESIN, B. **CLEO Extensions to the IEEE Learning Object Metadata**. 2003. Disponível em: /<https://www.oasis-open.org/committees/download.php/20490/CLEO_LOM_Ext_v1d1a.pdf>. Acesso em: 17 fev. 2022.

TAROUCO, L. M. R. *et al.* **Objetos de aprendizagem**: teoria e prática. 2014. Porto Alegre: Evangraf. 506 p. Disponível em: /<https://www.lume.ufrgs.br/bitstream/handle/10183/102993/000937201.pdf>. Acesso em: 05 jan. 2022.

THE PALLETS PROJECT. **Werkzeug**. 2021. Disponível em: /<https://palletsprojects.com/p/werkzeug/>. Acesso em: 01 dez. 2021.

THIELE, P. F. O. **Desambiguação de anotações morfossintáticas feitas por MTMDD**. 2015. 2015. 58 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre. Disponível em: /<http://tede2.pucrs.br/tede2/bitstream/tede/6341/2/475518\%20-\%20Texto\%20Completo.pdf>. Acesso em: 25 set. 2021.

WILEY, D. A. **The instructional use of learning objects**. 2002. Disponível em: /<http://www.reusability.org/read/>. Acesso em: 06 jan. 2022.

YOUTUBE. **API Reference**. 2019. Disponível em: /<https://developers.google.com/youtube/v3/docs/?hl=pt-br>. Acesso em: 05 dez. 2021.


UFVJM