CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Aquiles Alves Barros Junior

DESENVOLVIMENTO DE UMA PLATAFORMA LEARNING MANAGEMENT SYSTEM BASEADA NO PADRÃO SCORM

Palmas – TO

2020

Aquiles Alves Barros Junior

DESENVOLVIMENTO DE UMA PLATAFORMA LEARNING MANAGEMENT SYSTEM BASEADA NO PADRÃO SCORM

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Fabiano Fagundes.

Palmas – TO

2020

Aquiles Alves Barros Junior

DESENVOLVIMENTO DE UMA PLATAFORMA LEARNING MANAGEMENT SYSTEM BASEADA NO PADRÃO SCORM

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Fabiano Fagundes.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Fabiano Fagundes

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Profa. M.e. Madianita Bogo Marioti

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Jackson Gomes de Souza

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2020

RESUMO

BARROS, Aquiles Alves Junior. **Desenvolvimento de uma plataforma Learning Management System baseada no padrão SCORM**. Orientador: Fabiano Fagundes. 2020. 19 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2020.

Aulas à distância são uma das formas de se conquistar um curso superior mesmo morando em outra cidade, porém para que isso ocorra de maneira adequada de forma que o aprendizado do aluno não seja prejudicado é interessante que seja oferecida uma plataforma web que permita melhor interação do aluno, dando a ele os materiais compartilhados de forma fácil, rápida e segura. Essas plataformas são chamadas de *Learning Management System* (LMS) ou Sistema de Gestão da Aprendizagem em português, e oferecem aos professores a possibilidade de disponibilizar os materiais das aulas e ao aluno a possibilidade de acessá-los. Foi desenvolvido um padrão para o desenvolvimento destas plataformas, o SCORM (*Sharable Content Object Reference Model* - Modelo de Referência de Objeto de Conteúdo Compartilhável) que oferece ao desenvolvedor um modelo de desenvolvimento e de regras para padronizar a criação da plataforma. A proposta deste trabalho foi o desenvolvimento de uma plataforma LMS para compartilhamento de conteúdo usando o padrão SCORM para a sua padronização buscando oferecer uma boa interação com o usuário. Para isso foram utilizados Angular, JavaScript, HTML, CSS e Bootstrap para desenvolver o Front-end, Python e Flask para desenvolver o Back-end e MySQL para o banco de dados.

Palavras-chave: SCORM, LMS, Compartilhamento de arquivos.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API ..... Application Programming Interface
EaD ..... Educação a distância
LMS ..... Learning Management System
REST ..... Representational State Transfer
SGA ..... Sistemas de Gestão da Aprendizagem
SCORM ..... Sharable Content Object Reference Model
WWW ..... World Wide Web

SUMÁRIO

1 INTRODUÇÃO

A educação à distância tem uma longa história, existindo pelo menos desde o final do século XVIII, tendo um maior desenvolvimento a partir de meados do século XIX, quando foi criado o primeiro curso por correspondência, por Sir Isaac Pitman, *Correspondence Colleges* no Reino Unido (ANDRADE, 2000). Esse tipo de ensino vem evoluindo com o passar dos anos e com as novas tecnologias que foram surgindo.

O retorno aos bancos da escola nem sempre será possível ou concretizável, mas a interação com ambientes de aprendizagem online em geral alunos de faculdades em sua maioria já possuem computadores e internet é possível. Com o ambiente virtual é possível alcançar grande grupos de pessoas, no interior de uma grande empresa, ou simplesmente unidas por um interesse comum, para proporcionar-lhes formação profissional. Para auxílio ao gerenciamento destes ambientes foram criados *Learning Management System* - LMS (Sistemas de Gestão da Aprendizagem) (VIDAL, 2002).

Esses sistemas passaram a ser usados para o ensino a distância, mas era muito complicado quando era necessário compartilhar seus conteúdos, principalmente quando era necessário migrar de uma plataforma para outra, pois, muitas vezes, os conteúdos guardados em uma não eram compatíveis com a outra. Para resolver tal problema, nos meados dos anos 2000 foi desenvolvido o modelo SCORM - *Sharable Content Object Reference Model* (Modelo de Referência de Objeto de Conteúdo Compartilhável) (SCORM, 2010?).

O SCORM foi criado com o propósito de trazer um padrão para o desenvolvimento das plataformas LMS. O SCORM é formado por um conjunto de padrão usado para dizer ao desenvolvedor como fazer a implementação das plataformas e oferecem o compartilhamento de conteúdo.

Com a grande evolução da tecnologia nos últimos anos, o ensino *online* foi tornando-se menos complicado, em comparação com os períodos iniciais do ensino à distância. Com o tempo essa metodologia de estudo vem evoluindo, começando com o envio de materiais direto ao aluno via email, depois colocando tudo em um único lugar para os alunos acessarem, porém, a realização de atividades tem suas barreiras com esse tipo de abordagem.

Então, chegam as plataformas de ensino à distância, que trazem a possibilidade de os professores poderem, em um único lugar, dispor tanto seus materiais quanto suas atividades. Essas plataformas costumam seguir o modelo das plataformas *Learning Management System*, que são desenvolvidas voltadas para o ensino a distância. Porém, como elas são desenvolvidas

por diferentes empresas, o reaproveitamento de conteúdo de uma plataforma para a outra é muito difícil, pois elas dificilmente seguem o mesmo padrão.

Para solucionar tal problema se faz necessário que esse desenvolvimento siga um padrão, visto que é interessante que diferentes plataformas possam compartilhar seus conteúdos. Surge então o SCORM, que traz uma padronização no desenvolvimento das plataformas *Learning Management System*, tornando possível garantir o compartilhamento e o reaproveitamento de conteúdo das plataformas.

Como objetivo, o presente trabalho buscou a criação de uma plataforma de ensino a distância *Learning Management System* que possibilita o compartilhamento de conteúdo utilizando o padrão SCORM. Os objetivos específicos para assim chegar ao objetivo principal foram: fazer um levantamento sobre os modelos de plataformas *Learning Management System* a fim de entender sua arquitetura base; estudar o padrão SCORM e como é feita sua aplicação nas plataformas *Learning Management System*; desenvolver uma plataforma *Learning Management System*; e aplicar o padrão SCORM no desenvolvimento da plataforma.

As próximas seções irão apresentar mais detalhadamente o *Learning Management System* e, também, o SCORM. Após isso será apresentado o processo de desenvolvimento seguido durante a implementação da plataforma, mostrando os resultados obtidos após a implementação.

2 REFERENCIAL TEÓRICO

2.1 SHARABLE CONTENT OBJECT REFERENCE MODEL - SCORM

O SCORM é um conjunto de padrões técnicos para produtos de *software* de *eLearning*. O SCORM demostra aos programadores como escrever seu código para que ele possa "funcionar bem" com outro software de *eLearning*. É o padrão de fato da indústria para a interoperabilidade do *eLearning* (SCORM, 2010?). Especificamente, o SCORM governa como o conteúdo de aprendizagem *on-line* e os LMSs se comunicam. O SCORM não fala sobre design instrucional ou qualquer outra preocupação pedagógica, sendo puramente um padrão técnico (SCORM, 2010?).

Um dos grandes diferenciais para a utilização do SCORM no desenvolvimento de conteúdo para educação à distância é seu foco na reusabilidade, acessibilidade, interoperabilidade e durabilidade (DUTRA, 2006). O SCORM tem como um de seus objetivos propiciar a independência de plataforma na qual os objetos serão utilizados, assim como facilitar a migração de cursos entre diferentes LMS que sejam compatíveis com esse modelo (DUTRA, 2006).

2.1.1 Evolução

O SCORM passou por muitas mudanças desde o seu lançamento em 2000, apresentando várias versões (SCORM, 2010?):

- SCORM 1.0: lançado em janeiro de 2000, foi um esboço da estrutura do SCORM. Este documento não contém uma especificação totalmente implementável, mas contém uma prévia do trabalho que está por vir. O SCORM 1.0 continha os elementos principais que se tornariam a base do SCORM;
- SCORM 2004: lançado em janeiro de 2004, inclui versões muito maduras dos pacotes de conteúdo, tempo de execução e livros de metadados. As partes do SCORM 2004 derivadas do SCORM 1.2 são MUITO maduras e MUITO estáveis;
- A API Experience (xAPI): lançada em 26 de abril de 2013, a API Experience, também conhecida como API Tin Can ou xAPI, é o mais novo padrão de *eLearning* e resolve muitos problemas inerentes às versões mais antigas do SCORM, oferecendo elementos relacionados a aprendizado móvel, aprendizado baseado em equipe, funcionalidade entre domínios, sequenciamento, remoção da necessidade de um navegador da web e simulações / jogos sérios.

O SCORM passou por muitas versões e assim como outras ferramentas que recebem atualizações cada uma de suas versões veio com o objetivo de melhorar a versão anterior.

2.1.2 Implementação do SCORM

Para a versão do SCORM 2004, a ADL publicou as especificações em quatro livros: Visão Geral (*The SCORM Overview*), Modelo de Agregação de Conteúdo (*The SCORM Content Aggregation Model*), Ambiente de Execução (*The SCORM Runtime Environment*) e Sequenciamento e Navegação (*The SCORM Sequencing & Navigation*) (DUTRA, 2006).

No Modelo de Agregação de Conteúdo são definidos a Estrutura de Conteúdos, Empacotamento de Conteúdo, o Dicionário de Metadados e o Sequenciamento e navegação (DUTRA, 2006):

- Estrutura de Conteúdos: definição de terminologia comum usada em todo o CAM livro (DODDS, 2004).
- Empacotamento de Conteúdo: descrições e requisitos para agregação e empacotamento conteúdo de aprendizagem (DODDS, 2004).
- Dicionário de Metadados: descrições e requisitos para descrever componentes SCORM (DODDS, 2004).
- Sequenciamento e navegação: descrições e requisitos para definir informações de sequenciamento e navegação (DODDS, 2004).

O Empacotamento de Conteúdo, em conjunto com a Estrutura de Conteúdos e o Dicionário de Metadados, refere-se ao agrupamento, à organização e à identificação de todos os objetos de aprendizagem necessários para disponibilizar unidades de aprendizagem em diferentes LMS (DUTRA, 2006).

O SCORM também especifica os métodos para conduzir as comunicações entre o curso e o LMS. Isso é descrito como o Ambiente de Execução do SCORM, que inclui comunicações sobre a situação do curso, ou seja, quais materiais estão sendo apresentados para o estudante, assim como informações sobre o progresso do aluno durante o curso. A padronização dessas comunicações minimiza os problemas associados com a migração de cursos entre LMS diferentes, uma vez que tradicionalmente cada ambiente utiliza sua própria forma de rastreamento e gravação do progresso do aluno durante um curso (DUTRA, 2006).

2.2 LEARNING MANAGEMENT SYSTEM

Os Sistemas de Gestão da Aprendizagem (SGA) ou *Learning Management System* (LMS) são ferramentas imprescindíveis ao processo de ensino/aprendizagem, pois permitem acompanhar a construção do conhecimento individual dos alunos por meio do registro da

discussão, reflexão e colaboração. Estes sistemas abrangem funcionalidades de armazenamento, distribuição e gerenciamento de conteúdos de aprendizado, de forma interativa e gradativa. Deste modo, é possível registrar e apresentar as atividades do aluno, bem como seu desempenho, além da emissão de relatórios, propiciando o aperfeiçoamento do processo de ensino-aprendizagem (GOMES, 2009).

Em outras palavras, estes ambientes têm como objetivo primordial apoiar o processo de aprendizado, quer seja à distância, denominado de *e-learning*, quer seja semipresencial, denominado *Blending Learning* ou *B-Learning* (MACDONALD, 2006). Por consequência, eles proporcionam e facilitam diversas formas de interação, como as que ocorrem entre os próprios alunos, entre os alunos e o conteúdo, e entre os alunos e o professor. Seguem algumas das características básicas deste tipo de sistema (GOMES, 2009):

- recursos interativos;
- controle das atividades e monitoração de todas as interações e acessos dos alunos;
- compatibilidade com as especificações existentes de conteúdos, característica fundamental na transferência de conteúdos entre plataformas;
- gestão de conteúdo, que permite aos instrutores criarem cursos, organizando as informações de maneira que os usuários encontram facilmente o que precisam;
- sistema Colaborativo de Aprendizagem, o que permite pessoas com interesses comuns trabalharem em grupo, integrando-se e compartilhando conhecimentos;
- customização do conteúdo e adequação às necessidades do aprendiz.

No SCORM, o LMS é responsável pelo controle da distribuição dos objetos de aprendizagem aos estudantes obedecendo ao que foi estabelecido na Agregação de Conteúdos e no Sequenciamento e Navegação. O LMS tem a função de determinar o que e quando deve ser entregue e rastrear o progresso do estudante durante o curso (DUTRA, 2006).

O SCORM preocupa-se basicamente com o conteúdo, em como ele é organizado e sequenciado, em como será mostrado e como rastrear as ações do aluno no que se refere à interação do conteúdo (DUTRA, 2006).

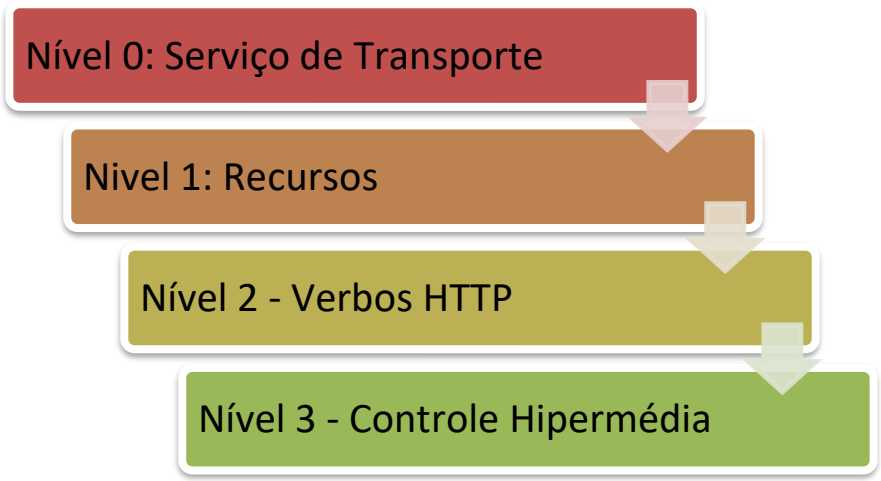
2.3 APPLICATION PROGRAMMING INTERFACE - API

Para se falar de API primeiro é preciso entender o *Representational State Transfer* – REST e o REST*ful*. O REST é um estilo de arquitetura da *World Wide Web* (WWW) desenvolvido com o intuito de servir aplicações Web. Para ser chamada de REST a aplicação precisa separar as arquiteturas do cliente-servidor. O cliente fica responsável pela interface

enquanto o servidor é responsável pelas resposta ao cliente por meio de requisições (MARQUES, 2018).

O REST*ful*, por sua vez, é um serviço Web que segue os princípios do REST. Para que uma *Application Programming Interface* – API seja considerada REST*ful* ela deve seguir as regras da arquitetura REST e do Modelo de Maturidade Richardson (MARQUES, 2018), como apresenta a figura 1.

Figura 1 – Modelo de Maturidade de Richardson



Observando a figura 1 verificam-se os seguintes níveis:

- Nível 0 – Serviço de Transporte: o nível responsável pelo transporte de informações do servidor.
- Nível 1 – Recursos: na API são criados *endpoints* responsáveis por receber cada requisição separadamente.
- Nível 2 – Verbos HTTP: nesse nível são implementados os verbos HTTP para execução das diferentes requisições, um exemplo desses verbos são é o CRUD (*Create, Read, Update, Delete* – inserção, leitura, atualização, remoção).
- Nível 3 – Controle Hipermídia: criação das URLs por onde o cliente fara a interação com a aplicação.

Ao se implementar os 4 níveis a API se torna REST*full*, tornado possível sua integração em um sistema onde é ela pode ser conectada diretamente com o banco de dados e fazer todo o seu gerenciamento.

2.4 TRABALHOS RELACIONADOS

Na busca por trabalhos relacionados a plataformas LMS desenvolvidas usando o padrão SCORM dois trabalhos chamaram a atenção:

Do pesquisador (VIEIRA, 2007): Desenvolvimento de Objeto de Aprendizagem, baseado em Especificações de Normatização SCORM, para o Caso de Suporte à Aprendizagem de Funções.

Do pesquisador (SILVA, 2008): Projeto e desenvolvimento de um Sistema Multi-agentes para Objetos Inteligentes de Aprendizagem baseado no padrão SCORM.

O primeiro trabalho tem como objetivo: “o desenvolvimento de um Objeto de Aprendizagem direcionado à área da matemática usando recursos de multimídia como figuras, animações e sistemas interativos para a compreensão do assunto” (VIEIRA, 2007).

Para o desenvolvimento do trabalho, o pesquisador realizou uma pesquisa sobre o Objeto de Aprendizagem (OA) definindo do que se tratam esses tipos de sistema e o que seria necessário para o seu desenvolvimento, percebendo que para alcançar o objetivo seria preciso a padronização nos parâmetros de desenvolvimento observando, ainda, que a adoção de padrões possibilitaria a uniformização do produto. Em sua pesquisa também observou que o SCORM é desenvolvido de modo a incorporar diversas especificações de objetos de ensino, sendo assim permitindo sua compatibilidade com diversas plataformas LMS.

Para o desenvolvimento do trabalho, ele utilizou a linguagem de programação Java, pois permite a inserção de gráficos interativos além de outros tipos de animação. Para a criação das animações foi utilizada a ferramenta Macromedia Flash MX na versão de testes, já que ela permite que o usuário crie figuras e anime-as.

A comunicação do objeto de aprendizagem com um LMS foi realizada com o uso de uma API. No trabalho, também, foi utilizado o programa Dreamweaver MX na sua versão de teste para a montagem das páginas Web.

O segundo trabalho teve como objetivo: “o desenvolvimento de um Sistema de Multi-agendas baseado no padrão SCORM, utilizando a engenharia de sistemas Multi-agentes (*Multiagent System Engineering* – MaSE)” (SILVA, 2008).

Para realizar o desenvolvimento, o pesquisador buscou mais sobre Objetos Inteligentes de Aprendizagem, vendo que uma das melhores maneiras de implementá-lo seria usando o LMS. Percebeu ainda que seria preciso uma padronização no desenvolvimento o pesquisador buscou também o SCORM para seu trabalho.

Para realizar a comunicação entre o objeto de aprendizagem e o LMS, o pesquisador desenvolveu um adaptador da API SCORM. A API comum provê uma forma de padronização para que os Objetos de Conteúdos Compartilhados se comuniquem com o LMS. Já um adaptador de API é uma biblioteca de funcionalidades que podem ser manipuladas

externamente. Com isso, o LMS precisa controlar este adaptador de forma que permita que o Objetos de Conteúdos Compartilhados tenha acesso as informações.

No desenvolvimento foi usado o paradigma cliente-servidor. No lado do servidor foi usada a plataforma JADE que gerencia os agentes, ontologias e regras de negócio. Para o armazenamento foi usado o sistema gerenciador de banco de dados PostgreSQL. Já do lado do cliente entra o objeto de aprendizagem, o qual pode conter qualquer tipo de mídia que possa ser apresentada através de uma página HMTL.

Para realizar a comunicação entre as duas estruturas foi utilizada a tecnologia RMI provida pela linguagem Java, permitindo que independente do LMS utilizado, seja possível se conectar ao Sistema Multiagentes, desde que implemente a API do modelo SCORM.

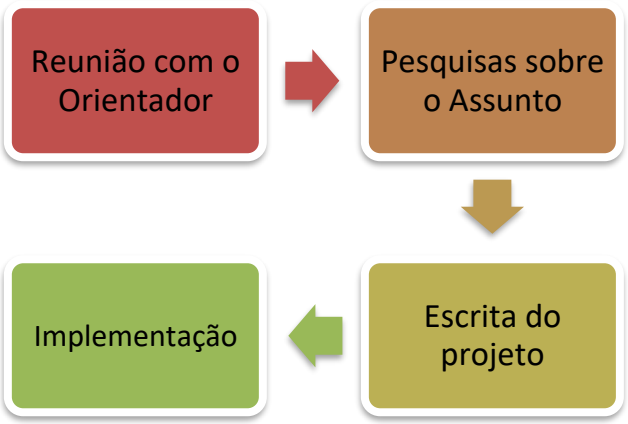
3 METODOLOGIA

Para o desenvolvimento da plataforma foram utilizados os modelos do LMS e padrão SCORM e, para isso, é preciso criar um ambiente de desenvolvimento onde os mesmos possam ser aplicados. Para tanto serão trabalhados três elementos o *Front-end*, *Back-end* e o Banco de Dados. Essa seção apresenta a metodologia utilizada na criação destes três elementos dentro da plataforma.

3.1 METODOLOGIA DE DESENVOLVIMENTO

A figura 2 apresenta as etapas seguidas para o desenvolvimento do trabalho.

Figura 2 – Sequência do trabalho



Na primeira etapa foram realizadas reuniões com o orientador para escolha do assunto do projeto, decidindo então pela criação de uma plataforma de estudo baseada no modelo das plataformas *LMS*, aplicando o padrão SCORM para padronizar a plataforma e possibilitar o compartilhamento de conteúdo.

Na segunda etapa foi realizada uma pesquisa sobre as plataformas *LMS* e o padrão SCORM. As plataformas *LMS* são focadas em desenvolver um ambiente de melhor interação entre usuário e o sistema, já SCORM tem como objetivo torna os conteúdos do sistema reaproveitáveis desta forma é possível migrar os conteúdos para outras plataformas que também tenham o padrão SCORM em sua implementação.

Na terceira etapa, após as pesquisas, teve início o processo de escrita do projeto e a definição das ferramentas que foram usadas para o desenvolvimento da plataforma, além da definição do conteúdo da plataforma. Para a definição das ferramentas para o desenvolvimento foi escolhido usar ferramentas que já se tem um maior conhecimento.

Na quarta etapa foi feita a implementação da plataforma, nessa etapa começou o desenvolvimento dos três elementos:

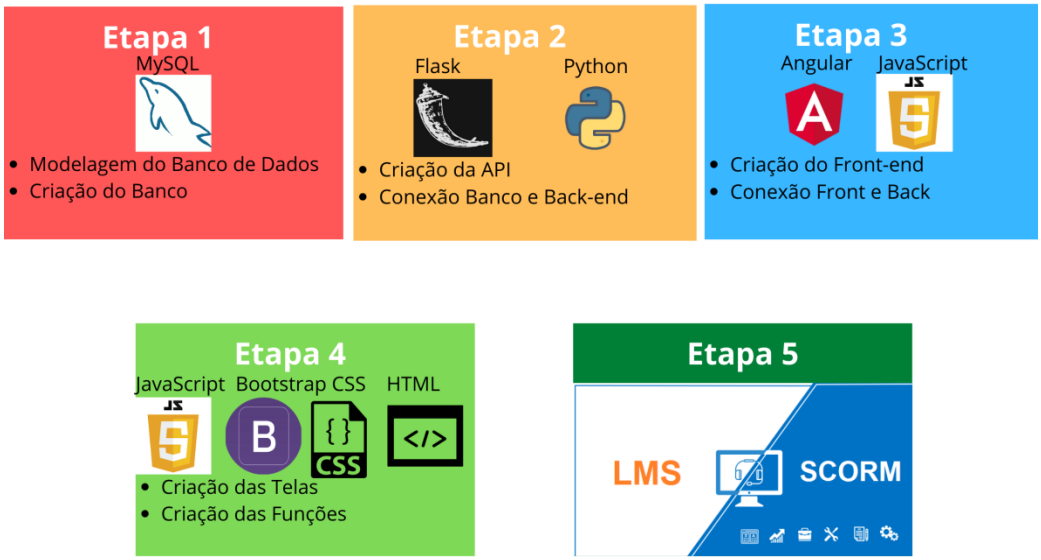
- *Front-end*: criado para gerenciar a interação do usuário com a plataforma.
- *Back-end*: desenvolvido para realizar a ligação entre o *Front-End* e o Banco de Dados, buscando e levando dados para ambas as partes.
- Banco de Dados: feito para armazenar os dados da plataforma.

Todo o desenvolvimento seguiu o modelo das plataformas *LMS* e aplicando o padrão SCORM para possibilitar o compartilhamento de conteúdo.

3.2 IMPLEMENTAÇÃO

A figura 3 apresente a sequência que foram seguidas na implementação da plataforma.

Figura 3 – Sequência da Implementação



A primeira etapa da implementação se iniciou com a modelagem do Banco de Dados como apresentado na figura 4, na qual foram definidos os dados que serão guardados nele e seus tipos. Em seguida, foram realizados a criado o Banco de Dados usando a ferramenta MySQL Workbench.

Na segunda etapa começou a implementação do *Back-end*, em que foi criada a API com a ajuda do *framework* Flask permitindo fazer requisições diretas ao banco de dados. A API conterá todas as URLs de acesso que o *Front-end* utilizará para pedir dados do banco e as suas requisições que serão enviadas ao banco de dados.

O Flask é feito em Python, logo, esta é a linguagem de programação utilizada para manipular a criação das rotas URLs e das requisições ao banco de dados. Para o contato direto com o banco são usados códigos do próprio banco de dados, ou seja, do próprio MySQL.

Para cada uma das tabelas foram criados os quatro comandos básicos do banco de dados, o chamado CRUD, que em inglês seriam *Create*, *Read*, *Update* e *Delete*, traduzidos para o

português criação, consulta, atualização e remoção de dados do banco. Após a implementação da API a mesma foi conectada ao Banco de Dados.

Na terceira etapa iniciou-se o desenvolvimento do *Front-end*, criando os componentes necessários para realizar o gerenciamento da plataforma, os componentes foram usados para realizar os tratamentos nas páginas onde a verá interação com o usuário.

Ao mesmo tempo é criado o *service* onde estará as configurações de conexão com o *Back-end* e também o tratamento dos dados que serão inseridos pelo usuário ou que viram do Banco de Dados. O Angular permite também o gerenciamento das URLs criando assim as páginas para realizar a interação com o usuário.

Na quarta etapa foi realizado o desenvolvimento das páginas e funcionalidades da plataforma. Nos componentes será usado o HTML para o desenvolvimento da página *Web*, o JavaScript com os arquivos e configurações do Angular será usado para implementa o gerenciamentos das funções dos componentes.

Na quinta etapa, que foi realizada em paralelo à quarta, foram feitas as modelagens das páginas, seguindo o modelo LMS para deixar a plataforma no formato de ensino à distância. Nessa etapa também foram aplicados os padrões SCORM, para permitir o compartilhamento e reaproveitamento de conteúdos.

4 RESULTADOS E DISCUSSÃO

O objetivo deste trabalho foi o desenvolvimento de uma plataforma *LMS* que possibilite o compartilhamento de conteúdo usando a padronização do SCORM. Essa seção apresenta os resultados obtidos com o desenvolvimento do trabalho.

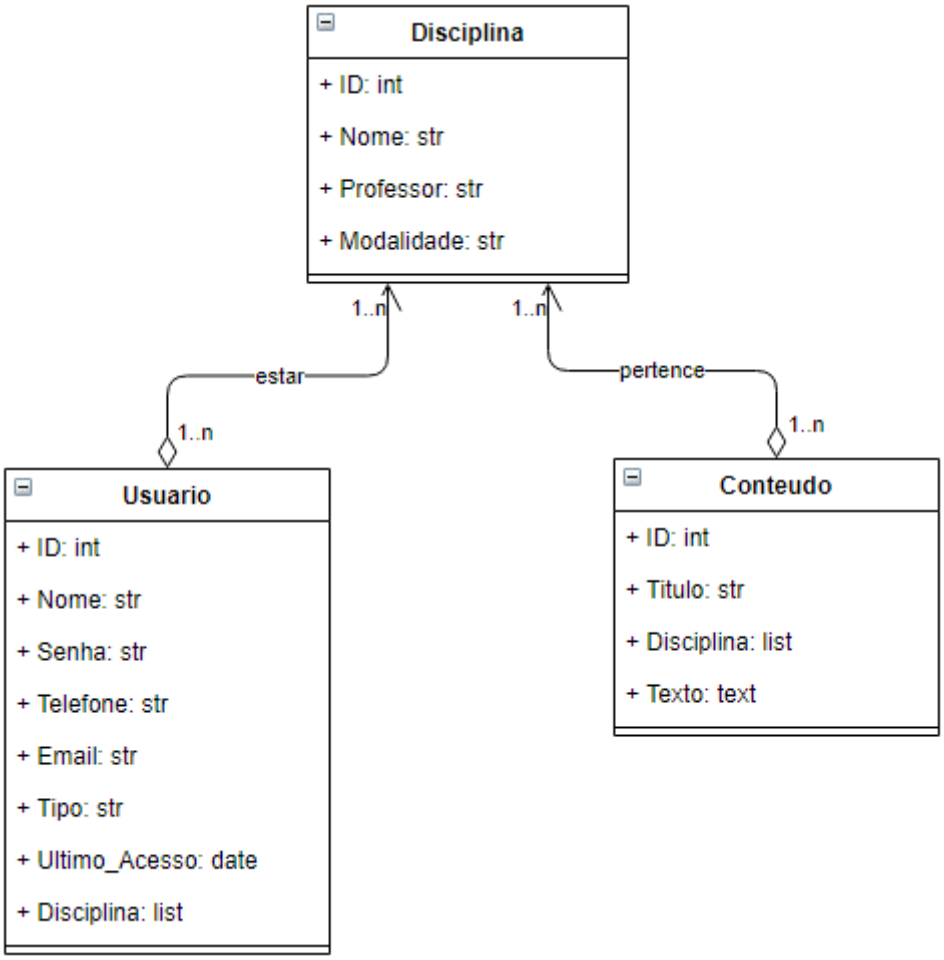
O sistema se baseia na arquitetura Cliente-Servidor. O servidor tem o papel de gerenciar e armazenar os dados do sistema sendo o Banco de Dados o que representa essa parte na plataforma. O cliente é aquele que utiliza as informações armazenadas no servidor, no projeto quem desempenha este trabalho é o *Front-end*. A ligação entre o Cliente-Servidor e possível de várias formas no presente trabalho foi pela criação de uma API para realizar este papel.

4.1 BANCO DE DADOS

Um banco de dados é formado pela coleção de dados, que são fatos que podem ser gravados e possuem um significado. Um bom exemplo são nomes, números telefônicos, endereços, *e-mail* e outros tipos de dados que possam ser aproveitados. Esses dados podem ter sido escritos em uma agenda de telefones ou armazenados em um computador, seja em um aplicativo ou em forma de anotações. Essas informações são uma coleção de dados com um significado implícito, consequentemente, um banco de dados (ELMASRI, 2005).

No presente trabalho o Banco de Dados realiza a função de armazenar: as informações básicas dos alunos, como nome, senha, tipo de usuário, além de informações do relacionamento dos aluno com a plataforma, como último *login* na plataforma, tempo que ficou conectado, dentre outros; os dados dos professores e das disciplinas; e os conteúdos postados pelos professores que, quando solicitado pelos usuários, serão passados através da API para o *Front-end*, que o apresentará ao usuário.

Figura 4 – Tabelas do Banco de Dados



Cada um dos campos das tabelas do Banco foi pensado de forma a facilitar a manipulação no *Front-end*. No total o banco possui três tabelas: uma para disciplina onde serão guardadas as matérias dos alunos, a tabela de conteúdos onde ficarão os textos postados pelos professores e a tabela usuário com os dados dos alunos e professores.

4.2 BACK-END

O *Back-end* fará a ligação entre o *Front-end* e o Banco de dados. Para o presente trabalho foi criada uma API para fazer esse papel. A vantagem da criação de uma API é que ela pode ser moldada mais facilmente às necessidades da plataforma. A API foi criada usando o *framework* Flask, que possibilita a comunicação direta com o banco de dados. O Flask é escrito em linguagem de programação Python, usando o Python e códigos de banco de dados é possível criar as funções da API. Uma das vantagens do Flask é que, quando é solicitado algum dado, ele retorna-o em formato JSON (*JavaScript Object Notation*), o que possibilita trabalhar de forma mais fácil com o Angular.

No *Front-end* é usado o módulo *HttpClientModule* para gerenciar a conexão entre o Angular e a API. Utilizando esse modulo são escritas as requisições que são enviadas diretamente paras as URLs implementadas no *Back-end*. A figura 5 demostra uma das URLs presente na API.

Figura 5 – Select nas informações do Aluno

```
@app.route('/selectaluno', methods=['GET'])
def select():
    try:
        alunos = {}
        lista = []
        cursor = db.cursor()
        cursor.execute("SELECT * FROM dados.Aluno")
        results = cursor.fetchall()
        for linha in results:
            aux = {'id': linha[0], 'nome': linha[1], 'senha': linha[2], 'telefone': linha[3], 'email': linha[4],
                   'diciplina': linha[5], 'tipo': linha[6], 'UltimoAcesso': linha[7]}
            lista.append(aux)
            alunos.update(alunos=lista)
        return alunos

    except Exception as e:
        print(e)

    finally:
        cursor.close()
```

A URL para a requisição de dados é um pouco mais simples que a de inserção pois não precisa carregar nenhum dado adicional. Através da função implementada no *Front-end* é possível solicitar as informações dos alunos cadastrados. Na sequência, a API, usando comados do próprio banco, solicita os dados de todos os alunos e retorna para o Front-end.

Quando é preciso realizar um cadastramento de dados o *Front-end* passa os dados para a API e ela, então, faz essa solicitação ao banco como apresentado pela figura 6.

Figura 6 – Insert de conteúdos

```
@app.route('/insertconteudo/<titu>/<dici>/<tex>', methods=['POST'])
def insert(titu,dici,tex):
    try:
        titulo = titu
        diciplina = dici
        texto = tex
        cursor = db.cursor()
        query = "INSERT INTO dados.Conteudo (Titulo,Diciplina,Texto) VALUES (%s,%s,%s)"
        data = (titulo, diciplina, texto)
        cursor.execute(query, data)
        db.commit()
        return 'Inserido com Sucesso'

    except Exception as e:
        print(e)

    finally:
        cursor.close()
```

Na URL de inserção de dados é preciso passar todos os dados que serão inseridos no banco de dados. A função apresentada recebe as informações e realiza a requisição para o banco informando em qual tabela deve ser inserido bem como quais elementos serão inseridos.

4.3 FRONT-END

O *Front-end* é a parte gráfica do desenvolvimento, ou seja, a interação direta com o usuário. Nele, são feitas as configurações que vão gerar a interação entre a plataforma e o usuário.

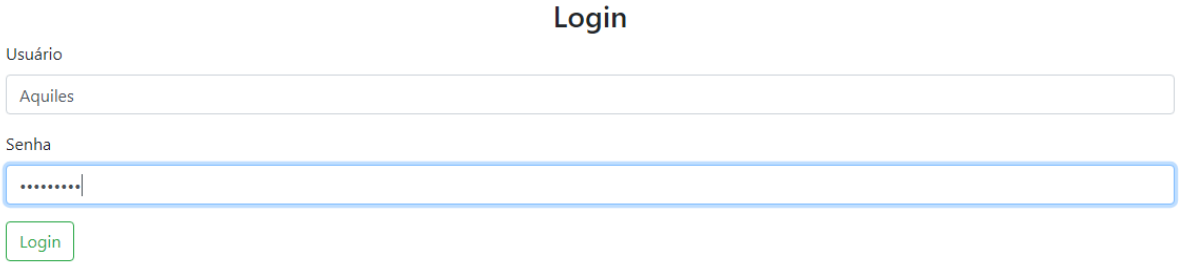
Para criar o *Front-end* foi definido o *framework* Angular por ele ser capaz de oferecer o controle dos códigos separadamente, além do controle com as URLs necessária para criar a navegação do usuário. O Angular foi usado para criar a configuração inicial do projeto. Também, foi utilizada a linguagem de programação JavaScript para, junto ao Angular, fazer a configuração das funcionalidades da plataforma.

No Angular também foram criados os *Components* onde foi implementado as funções de cada uma das telas da plataforma e também um *service* para realizar as requisições com o *Back-end*.

Foi utilizado o *framework* Bootstrap para dar uma melhor interação do usuário com a plataforma; o CSS, que foi usado para melhorar a aparência do site; o Bootstrap, que junto ao CSS, ofereceu a visualização da plataforma, facilitando a navegação do usuário; e o HTML, que foi usado para criar o corpo das páginas da plataforma.

Usando o modulo *RouterModule* foram criadas as URLs de acesso. Quando um usuário entrar na plataforma a primeira URL é a de *login* como apresentado pela figura 7.

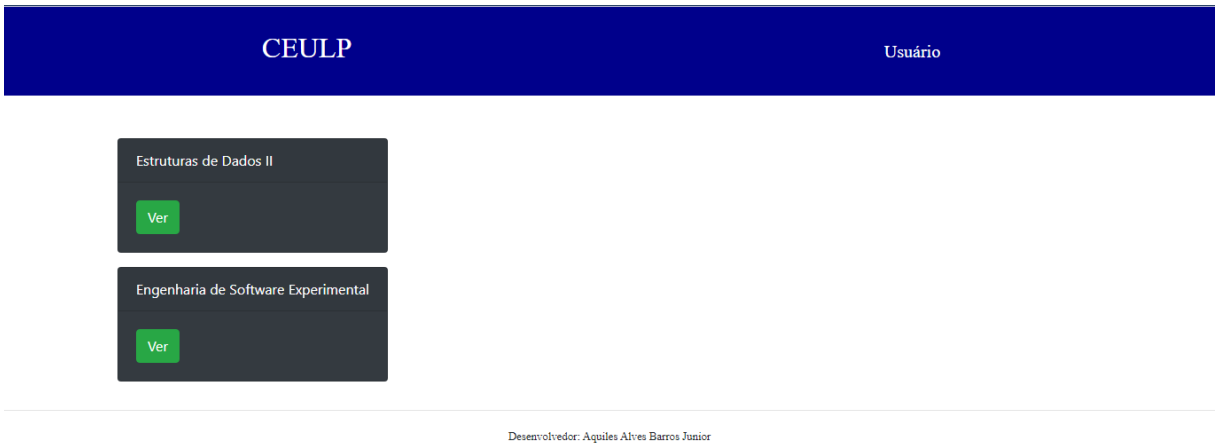
Figura 7 – Tela de login



Nessa primeira tela é possibilitado ao usuário realizar seu *login* permitindo, assim, que o mesmo acesse todas as funcionalidades da plataforma. Ao preencher os campos e selecionar a opção *login* para executar a função que realiza o acesso do usuário na plataforma, presente no *Component*. Essa função e responsável por passar os dados preenchidos pelo usuário e a lista de usuários para o *service*. Essa lista de usuários é trazida usando o modulo *HttpClientModule*.

Como o *Front-end* precisa se comunicar com o *Bank-end*, para assim levar e trazer conteúdo do banco de dados, o Angular utiliza o *HttpClientModule* que permite solicitar conteúdos do Back-end e enviá-los para o *Front-end*. Usando esse módulo é solicitado todos os usuários armazenados no banco de dados. Após receber os usuários é feita a comparação se os dados inseridos correspondem ao de algum usuário armazenado, se sim e realizado o *login*. Após realizar seu *login* o usuário é redirecionado para a tela inicial da plataforma como demostrado pela figura 8.

Figura 8 – Tela Inicial da Plataforma



Nessa tela o usuário poderá ver suas matéria cadastradas, além de poder utilizar a NavBar para navegar pelas páginas da plataforma podendo acessar seu perfil e, também, a página inicial. Quando o usuário escolhe uma disciplina, selecionando a opção ver, uma nova tela será exibida e serão apresentados os conteúdos dela, como demostrados pelas figuras 9 e 10.

Figura 9 – Conteúdos visão do Aluno

Figura 10 – Conteúdos visão do Professor

Quando o usuário seleciona o botão “Ver” em uma das disciplinas, mostrado na figura 8, é usado um recurso do módulo *RouterModule* que permite passar na URL o nome da disciplina correspondente. Usando esse recurso e o módulo *HttpClientModule* usado para trazer a lista de conteúdos cadastrados, é realizada uma verificação de que conteúdos estar cadastrado na disciplina escolhida pelo usuário. Feito essa verificação são apresentados os conteúdos da matéria.

Dependendo do tipo de usuário eles possuem algumas permissões diferentes, ambos possuem acesso a todas as telas da plataforma, mas somente os professores podem cadastrar novos conteúdos. O Aluno poderá ver os conteúdos de suas matérias cadastradas, já o professor, além de poder acessar a esses conteúdos, poderá realizar o cadastro de novos, como mostra a figura 11.

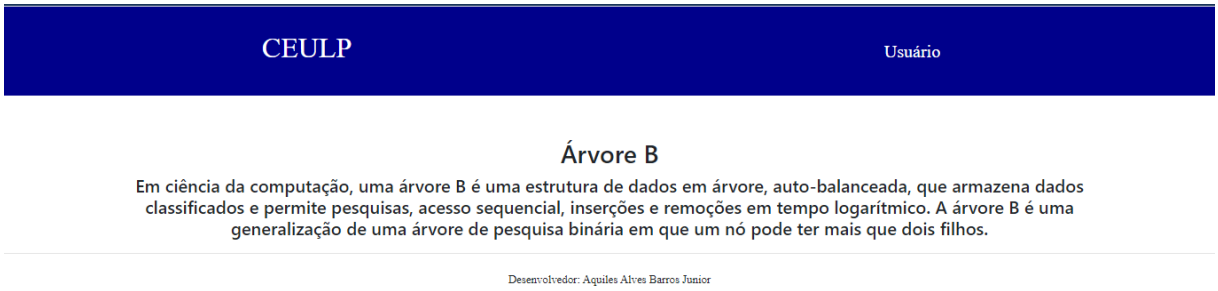
Figura 11 – Tela de cadastro de conteúdos

O professor ao escolher a opção de “Cadastrar Conteúdo”, apresentada na figura 10, é redirecionado para a tela apresentada na figura 11. Ao inserir o título e o texto do novo conteúdo que deseja cadastrar na plataforma e selecionar o botão “Cadastrar” será acionada uma função implementada no *Component* responsável por essa tela, usando o recurso do *RouterModule* é passado a disciplina para o Component, a função então chama a função de cadastro de

conteúdos implementado no service ele então utiliza o *HttpClientModule* para realizar uma requisição de inserção de dados para o *Back-end* passando os dados do conteúdo.

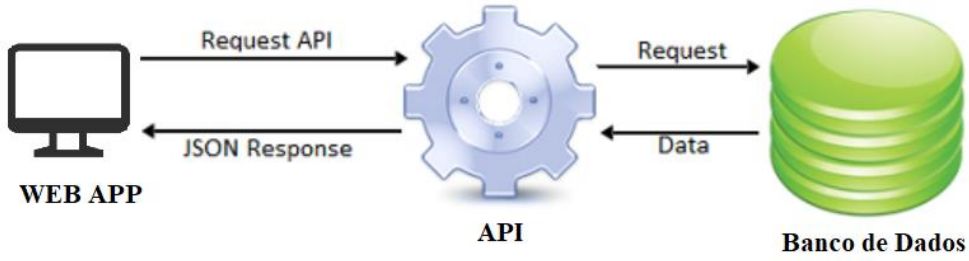
Quando o usuário clicar para ver um conteúdo como demostrado pela figura 9 e 10, é utilizado o RouterModule passado o id correspondente ao conteúdo e o mesmo e apresentado como demostra a figura 12.

Figura 12 – Tela de apresentação de conteúdo visão Aluno



A figura 13 apresenta o funcionamento da plataforma em relação à comunicação entre *Front-End*, *Back-End* e o banco de dados para troca de dados após o usuário estar conectado.

Figura 13 – Comunicação da plataforma



Após o acesso do usuário na plataforma, sempre que é solicitado algum conteúdo que esteja armazenado no banco de dados, o Módulo do Angular *HttpClientModule* manda uma requisição para a API através de uma de suas URLs de acesso. A API, após receber a requisição, analisa qual o pedido do *Front-end* e, então, realiza a execução da função. Ao se executar a função, a API envia o comando necessário para o Banco de dados.

O banco recebe o comando e o executa, depois, retorna os dados para a API ou uma resposta se a operação foi realizada com sucesso. A API, da mesma forma, retorna um JSON para o Angular quando tem dados ou somente uma resposta de operação realizada ou não.

As plataformas *LMS* têm como objetivo a criação de um ambiente de fácil interação com o usuário. Para isso busca-se um *Front-end* onde o usuário possa encontrar aquilo que quer com maior facilidade.

A plataforma será utilizada pelos professores para compartilharem conteúdos para os alunos, com a intenção de que esses conteúdos possam ser reaproveitados futuramente, mantendo todos os conteúdos com a padronização do SCORM, o que facilitará a migração futura para outra plataforma em que também se aplica este padrão.

4.4 APLICAÇÃO DO SCORM

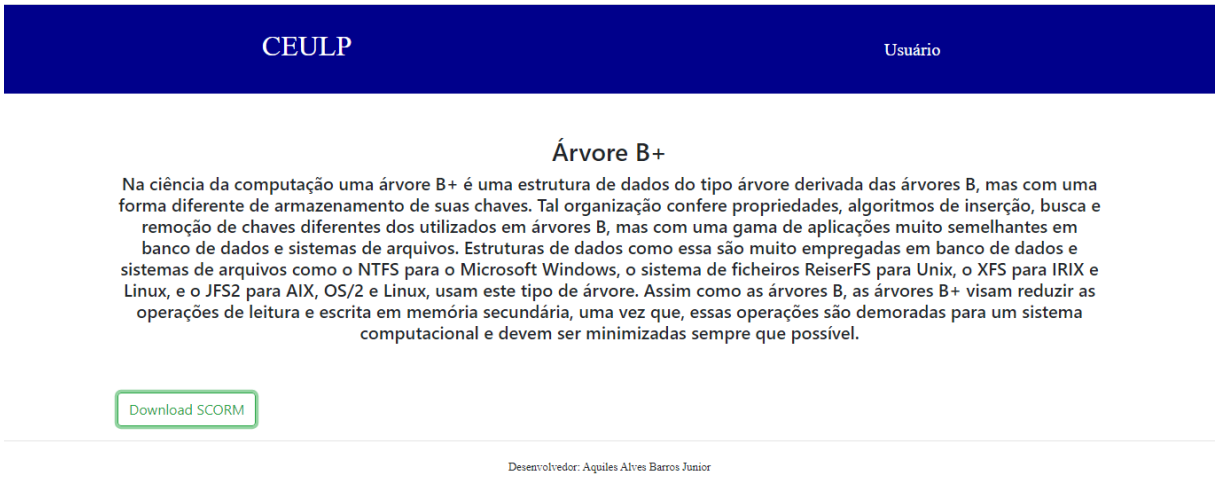
Para utilizar o SCORM em uma plataforma existem dois conceitos fundamentais: a leitura e a criação de arquivos SCORM. Para isso, é preciso interpretar cada uma das partes da arquitetura SCORM buscando permitir o compartilhamento de conteúdo.

Para este compartilhamento de conteúdo é preciso construir os arquivos que correspondem a cada conteúdo em formato SCORM e também uma página HTML, pois quando o conteúdo é escrito para ser compartilhado via SCORM ele precisa poder ser apresentado separadamente. Esse arquivo SCORM é, na verdade, um conjunto de documentos contendo todas as informações do conteúdo que será usado pelo administrador do sistema para realizar o compartilhamento dos conteúdos de uma plataforma para outra que tenha nela a implementação capaz de receber arquivos SCORM.

Como o SCORM gerar esse conjunto de documentos foi decidido então que para ser realizador o *download* e não fazer o usuário baixar várias arquivos todos os documentos criados são compactados e o usuário realiza um único *download* onde todos os documentos estão juntos.

A figura 14 apresenta a de apresentação de conteúdos na visão do professor.

Figura 14 – Tela de apresentação de conteúdo visão Professor



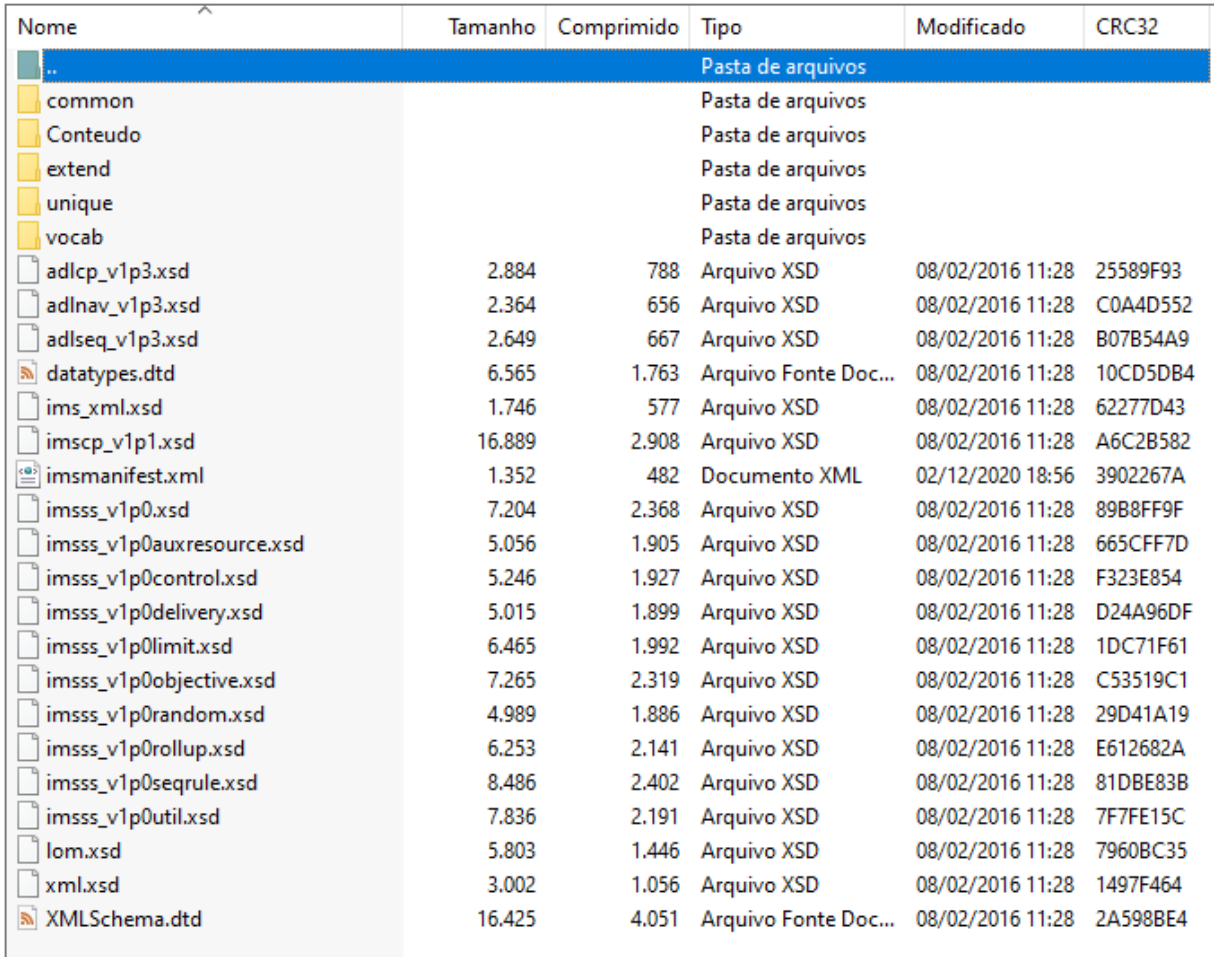
Nessa tela para o professor foi dada a permissão de realizar o *download* do conteúdo no padrão SCORM. Ao clicar no botão de ‘Download SCORM’, o *component* responsável realiza uma requisição ao *Back-end* pedindo o *download* do conteúdo. Essa requisição leva para o

Back-end a chave id do conteúdo, o *Back-end* então faz uma requisição ao banco pelo conteúdo correspondente ao id recebido.

Com o conteúdo, então são escritos o HTML e o *Manifest*, o padrão SCORM já possui alguns arquivos que são base para compor os arquivos que serão feitos *download*. No *Manifest* e preciso informar qual versão do SCORM está sendo usada, qual arquivo é referente ao conteúdo e, também, o título do conteúdo. Após as escritas desses arquivos eles são então compactados e retornados para o *Front-end*, onde é realizado o *download*.

A figura 15 apresenta os arquivos baixados que o *Back-end* retorna para a requisição do *Front-end*.

Figura 15 – Arquivos compactados para download



Nome	Tamanho	Comprimido	Tipo	Modificado	CRC32
..			Pasta de arquivos		
common			Pasta de arquivos		
Conteudo			Pasta de arquivos		
extend			Pasta de arquivos		
unique			Pasta de arquivos		
vocab			Pasta de arquivos		
adlcp_v1p3.xsd	2.884	788	Arquivo XSD	08/02/2016 11:28	25589F93
adlnav_v1p3.xsd	2.364	656	Arquivo XSD	08/02/2016 11:28	C0A4D552
adlseq_v1p3.xsd	2.649	667	Arquivo XSD	08/02/2016 11:28	B07B54A9
datatypes.dtd	6.565	1.763	Arquivo Fonte Doc...	08/02/2016 11:28	10CD5DB4
ims_xml.xsd	1.746	577	Arquivo XSD	08/02/2016 11:28	62277D43
imscp_v1p1.xsd	16.889	2.908	Arquivo XSD	08/02/2016 11:28	A6C2B582
imsmanifest.xml	1.352	482	Documento XML	02/12/2020 18:56	3902267A
imsss_v1p0.xsd	7.204	2.368	Arquivo XSD	08/02/2016 11:28	89B8FF9F
imsss_v1p0auxresource.xsd	5.056	1.905	Arquivo XSD	08/02/2016 11:28	665CFF7D
imsss_v1p0control.xsd	5.246	1.927	Arquivo XSD	08/02/2016 11:28	F323E854
imsss_v1p0delivery.xsd	5.015	1.899	Arquivo XSD	08/02/2016 11:28	D24A96DF
imsss_v1p0limit.xsd	6.465	1.992	Arquivo XSD	08/02/2016 11:28	1DC71F61
imsss_v1p0objective.xsd	7.265	2.319	Arquivo XSD	08/02/2016 11:28	C53519C1
imsss_v1p0random.xsd	4.989	1.886	Arquivo XSD	08/02/2016 11:28	29D41A19
imsss_v1p0rollup.xsd	6.253	2.141	Arquivo XSD	08/02/2016 11:28	E612682A
imsss_v1p0seqrule.xsd	8.486	2.402	Arquivo XSD	08/02/2016 11:28	81DBE83B
imsss_v1p0util.xsd	7.836	2.191	Arquivo XSD	08/02/2016 11:28	7F7FE15C
lom.xsd	5.803	1.446	Arquivo XSD	08/02/2016 11:28	7960BC35
xml.xsd	3.002	1.056	Arquivo XSD	08/02/2016 11:28	1497F464
XMLSchema.dtd	16.425	4.051	Arquivo Fonte Doc...	08/02/2016 11:28	2A598BE4

Dentre estes arquivos a pasta ‘Conteudo’ é onde está sendo armazenada a página HTML do conteúdo, o arquivo ‘imsmanifest.xml’ é o arquivo onde são escritos os campos citados na figura 15. A figura 16 apresenta o conteúdo do arquivo ‘imsmanifest.xml’.

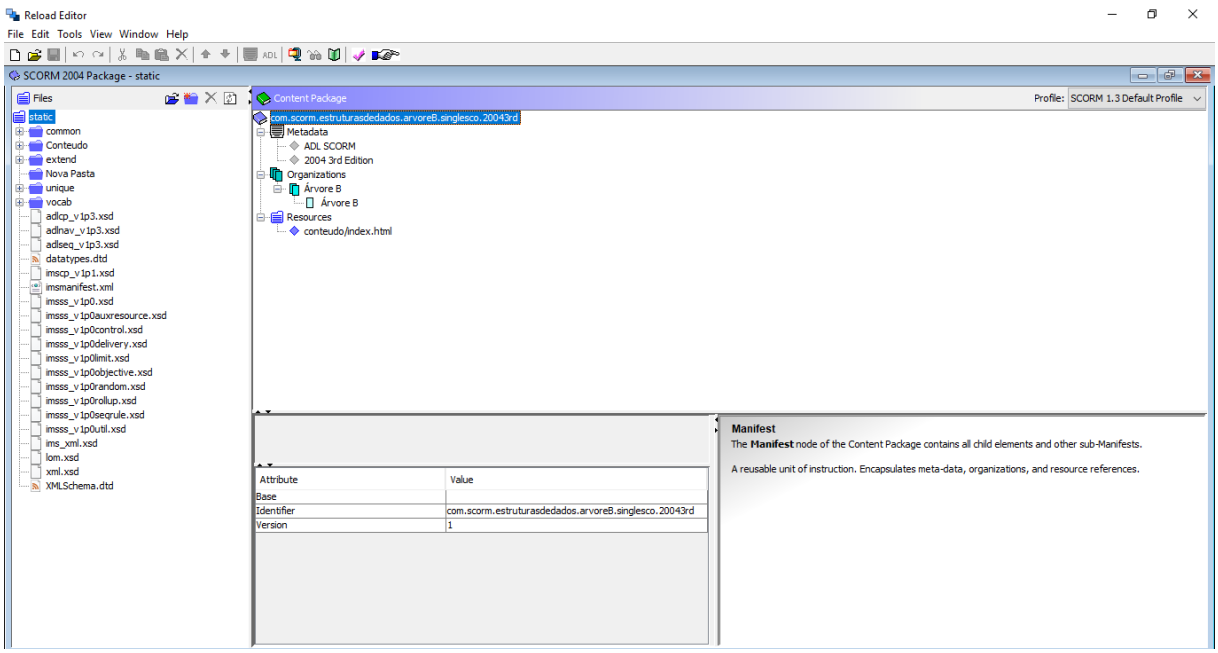
Figura 16 – Arquivo imsmanifest.xml

```
1  <?xml version="1.0" standalone="no" ?>
2  <manifest identifier="com.scorm.manifesttemplates.scorm2004.4thEd.nometadata" version="1"
3  xmlns = "http://www.imsglobal.org/xsd/imscp_v1p1"
4  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5  xmlns:adlcp = "http://www.adlnet.org/xsd/adlcp_v1p3"
6  xmlns:adlseq = "http://www.adlnet.org/xsd/adlseq_v1p3"
7  xmlns:adlnav = "http://www.adlnet.org/xsd/adlnav_v1p3"
8  xmlns:imsss = "http://www.imsglobal.org/xsd/imsss"
9  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
10 xsi:schemaLocation = "http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd
11 http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd
12 http://www.adlnet.org/xsd/adlseq_v1p3 adlseq_v1p3.xsd
13 http://www.adlnet.org/xsd/adlnav_v1p3 adlnav_v1p3.xsd
14 http://www.imsglobal.org/xsd/imsss imsss_v1p0.xsd" >
15 <metadata>
16  <schema>ADL SCORM</schema>
17  <schemaversion>2004 4th Edition</schemaversion>
18 </metadata>
19 <organizations default="Estruturas de Dados II">
20  <organization identifier="Estruturas de Dados II>
21  <title>Árvore B</title>
22 <item identifier="item_1" identifierref="resource_1">
23  <title>Árvore B</title>
24 </item>
25 </organization>
26 </organizations>
27 <resources>
28 <resource identifier="resource_1" type="webcontent" adlcp:scormType="sco" href="Conteudo/index.html">
29  <file href="Conteudo/index.html" />
30 </resource>
31 </resources>
32 </manifest>
33
```

Nas linhas 16 e 17 são especificados a versão do SCORM, na linha 23 o título do conteúdo, na linha 28 é dito qual arquivo e o arquivo de início do conteúdo e da linha 29 até o elemento ‘</resource>’ são especificados quais arquivos fazem parte para a apresentação do conteúdo. A função implementada e capaz de escrever os arquivos SCORM para conteúdo de textos.

Para realizar teste nos documentos baixados foi utilizado o aplicativo Reload Editor onde e possível testar arquivos SCORM a figura 17.

Figura 16 – Arquivo imsmanifest.xml



Com ele é possível verificar todas as partes do arquivo ‘imsmanifest.xml’ a *Metadata* onde foi descrito a versão dos SCORM, a *organizations* onde foi descrito o título do conteúdo e os *resouces* que é onde é apresentados todos os arquivos que compõem o conteúdo.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve objetivo o desenvolvimento de uma plataforma de ensino a distância seguindo o modelo LMS onde fosse possível o compartilhamento de conteúdos usando para isso o padrão SCORM.

Para chegar no objetivo foram realizadas inicialmente várias pesquisas para poder se entender melhor o LMS e SCORM, onde foi observado como cada um seria aplicado dentro da plataforma. Entendeu-se então que o LMS são modelos de plataformas de ensino a distância que buscam melhorar a interação do usuário com a plataforma.

O SCORM, por sua vez, é uma forma de padronizar os conteúdos tendo duas aplicações a escrita onde o conteúdo é escrito junto aos padrões é então enviado para download e a leitura onde a plataforma interpreta esses arquivos e compõem a sua apresentação os conteúdos escritos em formato SCORM.

Com os objetivos de conhecer melhor o LMS e SCORM deu-se início o desenvolvimento da plataforma Web usando o *framework* Angular seguindo os modelos das plataformas de ensino a distância LMS para, assim, possibilitar uma melhor interação com o usuário, além de ser utilizado CSS e Boostrap para oferecer à plataforma uma melhor visualização e compreensão ao usuário. O HTML foi usado para estruturar as páginas e o JavaScript para implementar as funções da plataforma.

A plataforma então seguiu o modelo Cliente-Servidor precisando assim da criação de uma API para realizar essa conexão, no Servidor foi criado todas as requisições necessárias para a busca de dados no banco de dados, e também os retornos de dados para o *Front-end.*

No Cliente foi feita todas as funções para possibilitar a interação do usuário com a plataforma e também possibilitado o controle dos professores em relação aos conteúdos que o professor poder cadastrar na plataforma e disponibilizar para os alunos.

Durante a implementação foi feito a implantação do SCORM para assim disponibilizar o compartilhamento de conteúdo dando assim a possibilidade para os professores de ser possível levar os conteúdos de uma plataforma para outra que também utilize o padrão SCORM em sua implementação.

Como trabalhos futuros, pode-se realizar melhorias e adicionamento de novas funções na plataforma possibilitando um melhor controle para os professores dando a eles a opção de inserir outros formatos de conteúdos como vídeo, slide, atividades entre outros. E também uma melhor experiência para os alunos por possibilitar responder atividades assistir aulas gravadas

tudo na mesma plataforma. Tendo ainda o desenvolvimento para que a plataforma seja capaz de ler os arquivos escritos no padrão SCORM.

REFERÊNCIAS

ANDRADE, P. Ensino à distância. **Coimbra:** Departamento de Engenharia de Informática da Universidade de Coimbra, 2000. Disponível em: http://student.dei.uc.pt/~pandrade/sf/texto.htm. Acesso em 18 Jun. 2020.

DODDS, P.; THROPP, S. E. **SCORM content aggregation model**. 1.3.1. ed. [S. l.]: Advanced Distributed Learning Initiative, 2004. Disponível em: https://web.fe.up.pt/~ee92193/documentacao/scormcam.pdf. Acesso em jun 2020.

DUTRA, Renato Luís de Souza; TAROUCO, Liane Margarida Rockenbach. Objetos de Aprendizagem: uma comparação entre SCORM e IMS Learning Design. **Centro Interdisciplinar de Novas Tecnologias da Educação Universidade Federal do Rio Grande do Sul**, 2006. Disponível em: https://www.seer.ufrgs.br/renote/article/download/13862/7783. Acesso em abr 2020.

ELMASRI, Ramez et al. **Sistemas de banco de dados**. São Paulo: Pearson Addison Wesley, 2005. Disponível em: http://tonysoftwares.com.br/attachments/article/5297/Sistema_de_banco_de_dados_Navathe.pdf. Acesso abr de 2020.

GHIRARDINI, Beatrice. **E-learning methodologies**: A guide for designing and developing e-learning courses. Roma: Food and Agriculture Organization of the United Nations, 2011. Disponível em: http://www.fao.org/3/i2516e/i2516e.pdf. Acesso abr de 2020.

GOMES, Alex Sandro; CARVALHO, Rosângela Saraiva; MELO FILHO, Ivanildo José de; ROLIM, Ana Luiza de Souza; MONTEIRO, Bruno de Sousa; OLIVEIRA, Gleibson Rodrigo Silva de. Amadeus: novo modelo de sistema de gestão de aprendizagem. **Revista Brasileira de Aprendizagem Aberta e A Distância**, [s.l.], v. 8, p. 1-17, 24 maio 2009. ABED - Associacao Brasileira de Educacao a Distancia. Disponível em: http://seer.abed.net.br/index.php/RBAAD/article/view/216. Acesso em abr 2020.

MACDONALD, Janet. **Blended learning and online tutoring**: A good practice guide. [S. l.: s. n.], 2006. Disponível em: https://www.semanticscholar.org/paper/Blended-Learning-and-Online-Tutoring%3A-A-Good-Guide-Macdonald/6a91d4c4c23f4d7c283a3d3e39786ddf855bcfb8?p2df. Acesso em abr 2020.

MARQUES, Ana Isabel Alves. **Desenvolvimento de API para aplicação cloud**. 2018. Tese de Doutorado (Mestrado em Engenharia Informática) - Escola Superior de Tecnologia e Gestão, Porto, 2018. Disponível em: https://iconline.ipleiria.pt/handle/10400.8/3263. Acesso jun de 2020.

SCORM. **solved and explained** .[S.I][2010?] Disponível em: https://SCORM.com/. Acesso em abr. 2020.

SILVA, Júlia Marques Carvalho; BAVARESCO, Natanael; SILVEIRA, Ricardo Azambuja. Projeto e desenvolvimento de um Sistema Multi-agentes para Objetos Inteligentes de Aprendizagem baseado no padrão SCORM. **Revista Brasileira de Informática na Educação**, Florianópolis – SC, v. 16, n. 01, 2008. Disponível em: https://www.br-ie.org/pub/index.php/rbie/article/view/19. Acesso em jun 2020.

VIDAL, Elisabete. **Ensino a distância vs ensino tradicional**. Universidade Fernando Pessoa, Porto, 2002. Disponível em: http://homepage.ufp.pt/lmbg/monografias/evidal_mono.pdf. Acesso em jun 2020.

VIEIRA, Carlos Eduardo Milanezi; NICOLEIT, Evanio Ramos. Desenvolvimento de Objeto de Aprendizagem, baseado em Especificações de Normatização SCORM, para o Caso de Suporte à Aprendizagem de Funções. **RENOTE-Revista Novas Tecnologias na Educação**, Santa Catarina, v. 5, n. 1, 2007. Disponível em: https://www.seer.ufrgs.br/renote/article/download/14168/8098. Acesso em jun 2020.