UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

KODA GABRIEL FARIA MELO
Orientador: Prof. Dr. Saul Emanuel Delabrida Silva

UMA ARQUITETURA DE INTEGRAÇÃO DE LEARNING MANAGEMENT SYSTEM E LEARNING RECORD SYSTEM PARA JOGOS SÉRIOS

Ouro Preto, MG
2022

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

KODA GABRIEL FARIA MELO

UMA ARQUITETURA DE INTEGRAÇÃO DE LEARNING MANAGEMENT SYSTEM E LEARNING RECORD SYSTEM PARA JOGOS SÉRIOS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Saul Emanuel Delabrida Silva

Ouro Preto, MG
2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M528a Melo, Koda Gabriel Faria.
Uma arquitetura de integração de Learning Management System e Learning Record System para jogos sérios. [manuscrito] / Koda Gabriel Faria Melo. - 2022.
36 f.: il.: color..

Orientador: Prof. Dr. Saul Emanuel Delabrida Silva.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Jogos sérios. 2. Arquitetura. 3. Jogos educativos. I. Silva, Saul Emanuel Delabrida. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Koda Gabriel Faria Melo

UMA ARQUITETURA DE INTEGRAÇÃO DE LEARNING MANAGEMENT SYSTEM E LEARNING RECORD SYSTEM PARA JOGOS SÉRIOS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 14 de Junho de 2022.

Membros da banca

Saul Emanuel Delabrida Silva (Orientador) - Doutor - Universidade Federal de Ouro Preto
Rodrigo Geraldo Ribeiro (Examinador) - Doutor - Universidade Federal de Ouro Preto
Andrea Gomes Campos Bianchi (Examinadora) - Doutora - Universidade Federal de Ouro Preto

Saul Emanuel Delabrida Silva, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 14/06/2022.

Documento assinado eletronicamente por **Saul Emanuel Delabrida Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 21/06/2022, às 16:45, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0343223** e o código CRC **198A0952**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

“Nunca é tarde demais para mudar o que você é;

levei muito tempo para descobrir isso.”

(MORGENSTERN, 2019)

Resumo

A utilização de jogos sérios tem se mostrado efetiva no auxílio do ensino de diversos tópicos a estudantes de várias idades. Entretanto, ainda há poucos estudos que descrevam o comportamento dos estudantes durante esses jogos e que se utilizem dessas informações para apontar possíveis melhorias. Esse trabalho tem como objetivo a criação de uma arquitetura que realize a integração de uma plataforma de ensino com uma plataforma de armazenamento de dados sobre aprendizado. Foi implementado um Learning Management System que permite a inserção de aulas e cursos, sendo possível que essas aulas definidas como aulas para jogos. O conteúdo dessas aulas é enviado aos jogos, permitindo um grau de personalização por parte dos educadores. Foram desenvolvidos diversos plugins, que funcionam em conjunto com o LearnPress, e provêm os recursos necessários para funcionamento do LMS e da API que envia os dados para os jogos. Esses jogos, por suas vez, geram informações de uso para análise futura pelos educadores. Foi desenvolvido um Helper, que utiliza a xAPI como padrão e a biblioteca TinCan.NET para salvar as informações de uso em um LRS. Como trabalho futuro, uma plataforma de visualização e análise desses dados pode ser desenvolvida, para aumentar ainda mais as aplicações da arquitetura.

Palavras-chave: LMS. LRS. Jogos sérios. Arquitetura.

Abstract

The use of serious games has been shown to be effective in helping teachers to show different topics to students of different ages. However, there are still very few studies that describe the behavior of students during these games and fewer that use this information to point out possible improvements. This work aims to create an architecture that integrates a teaching platform with a learning data storage platform. A Learning Management System was implemented, and allows the upload of classes and courses, making it possible for these classes to be defined as classes for games. The content of these classes is sent to the games, allowing a degree of customization by the educators. Several plugins were developed, which work together with LearnPress, and provide the necessary resources for the functioning of the LMS and the API that sends data to the games. These games generate usage information for future analysis by educators. A Helper was developed, which uses the xAPI standards and the TinCan.NET library to save usage information in an LRS. As a future work, a platform for visualization and analysis of this data can be developed, to further increase the usability and applications of the architecture.

Keywords: LMS, LRS, Serious Games, Architecture.

Lista de Ilustrações

Figura 3.1 – Visualização da proposta da arquitetura 8
Figura 3.2 – Exemplo de JSON aceito pela xAPI 9
Figura 3.3 – Exemplo de um verbo na xAPI Registry 10
Figura 3.4 – Exemplo de visualização de uma informação de teste salva na ScormCloud . 11
Figura 3.5 – Interface do Moodle após instalação limpa 12
Figura 3.6 – Exemplo de site configurável com o plugin LearnPress. 12
Figura 3.7 – Visualização da arquitetura com os componentes selecionados. 15
Figura 4.1 – Lista de aulas no LMS e a possibilidade de criar uma nova aula 17
Figura 4.2 – Trecho de código do plugin que permite marcar aula para jogo 18
Figura 4.3 – Seletor de imagens para a lição com algumas imagens adicionadas 19
Figura 4.4 – Seletor de vídeos para a lição 19
Figura 4.5 – Seletor de questionário para a lição 20
Figura 4.6 – Print do aplicativo Postman, exibindo a chamada para a API com o resultado das aulas armazenadas no banco 21
Figura 4.7 – Exibição da lição online indicando que a aula deve ser finalizada em plataforma externa 22
Figura 4.8 – Arquivo de normalização da criação dos verbos 23
Figura 4.9 – Arquivo de normalização da criação das atividades 24
Figura 4.10–Arquivo que visa padronizar o envio das informações para o xAPI 24
Figura 4.11–Arquivo dos verbos a serem utilizados, realizado com auxílio o *Helper* . . . 26
Figura 4.12–Arquivo das atividades a serem utilizadas, realizado com o auxílio do *Helper* 26
Figura 4.13–Print das informações salvas na xAPI 27
Figura 4.14–Exibição da lição online indicando que a aula foi concluída 27
Figura 4.15–Imagem de uma aula dentro do jogo sério desenvolvido por Arilton Aguilar . 27
Figura A.1 – Parte I do tutorial de instalação do LMS 33
Figura A.2 – Parte II do tutorial de instalação do LMS 34
Figura B.1 – Parte I do tutorial de uso do Helper do xAPI 35
Figura B.2 – Parte II do tutorial de uso do Helper do xAPI 36

Lista de Abreviaturas e Siglas

API	Application Programming Interface
xAPI	Experience API
TLA	Training and Learning Architecture
ADL	Advanced Distributed Learning
LRS	Learning Records Store
LMS	Learning Management System
JSON	Javascript Object Notation
LAP	Learning Analytics Plataform

Sumário

1 Introdução 1
1.1 Justificativa 1
1.2 Objetivos 2
1.3 Organização do Trabalho 2
2 Revisão Bibliográfica 3
2.1 Fundamentação Teórica 3
2.1.1 Jogos Sérios 3
2.1.2 Learning Analytics 3
2.1.3 API 3
2.1.4 xAPI 4
2.1.5 Learning Records Store — LRS 4
2.1.6 Learning Management System — LMS 4
2.1.7 Learning Analytics 4
2.1.8 JSON 4
2.2 Trabalhos Relacionados 5
2.3 Lições aprendidas 6
3 Desenvolvimento 7
3.1 Levantamento de Requisitos para implementação da arquitetura 7
3.1.1 Requisitos Funcionais 7
3.1.2 Requisitos não funcionais 8
3.2 Seleção, customização e implementação dos componentes 9
3.2.1 LRS API 9
3.2.1.1 xAPI Registry 10
3.2.1.2 Integração da xAPI com jogos sérios 10
3.2.2 LRS e LRS BD 11
3.2.3 LMS e LMS BD 11
3.2.3.1 Plugins implementados e/ou customizados 13
3.2.3.1.1 Video metabox aoc 13
3.2.3.2 Image metabox 13
3.2.3.3 LearnPress associate quiz 14
3.2.3.4 LearnPress custom lessons 14
3.2.3.5 Customização do LearnPress 14
3.3 LAP 14
3.4 LMS API 14
3.5 Arquitetura proposta 15
3.5.1 Educadores 16

3.5.2 Estudantes 16
4 Resultados 17
4.1 Cadastro de aulas 17
4.2 Cadastro de aulas para jogo 17
4.3 Inserção de slides no jogo 18
4.4 Inserção de vídeos no jogo 18
4.5 Um questionário poderá ser associado a uma aula para jogo 19
4.6 Poderá ser criada uma sequência de aulas, gerando um curso 20
4.7 Alunos podem se cadastrar na plataforma e em cursos e acessar e concluir esses cursos 20
4.8 Conteúdo de aulas para jogo precisa estar disponível via API com todos os dados associados 21
4.9 Deve ser possível marcar uma aula para jogo como finalizada via API 22
4.10 Deve ser exibido na progressão do curso no LMS quando uma aula é para jogo . 22
4.11 Deve ser possível registrar via API informações de uso da aula para jogo no LRS 23
4.12 Tutoriais 23
4.13 Integração em um jogo sério 24
4.13.1 Definição do JSON a ser salvo 25
4.14 Realizando logs de uma aula teste 26
5 Considerações Finais 28
5.0.1 Trabalhos futuros 28

Referências 30

Apêndices 32
APÊNDICE A Tutorial de instalação do LMS e plugins 33
APÊNDICE B Tutorial de uso do xAPI Helper 35

1 Introdução

Existem inúmeras perspectivas sobre o que define um jogo sério, variando entre academia e indústria (LAAMARTI; EID; SADDIK, 2014). A definição mais comum é a de "jogos que não tem o entretenimento, prazer ou diversão como seu objetivo principal" (MICHAEL; CHEN, 2005). Por exemplo, jogos educacionais são incluídos e muitas vezes recebem o maior foco, mas outros propósitos como aprimoramento de habilidades, reabilitação e treino militar também aparecem. Observa-se na literatura que os estudos feitos com jogos sérios são bem heterogêneos, e já existem revisões que visam trazer alguma organização e estrutura para essa área, que é relativamente nova (BOYLE et al., 2016). Ainda segundo Boyle et al. (2016), as áreas ditas ciências "duras" (como matemática, física, química, etc) tendem a ter metodologias de avaliações mais bem definidas e estruturadas, enquanto áreas definidas como ciências "moles" (como ciências sociais, história e psicologia) geralmente optam por pesquisas e avaliações mais subjetivas.

Segundo Egenfeldt-Nielsen (2006), algumas barreiras que dificultam a avaliação são as atividades feitas em um ambiente específico, habilidades com jogos que variam entre os estudantes e alguns problemas práticos como preparação dos professores e custos de instalação. Para auxiliar nesses estudos, é importante conhecer os estudantes e como cada um deles executa as ações disponíveis nos jogos. Informações como tempo de uso, erros durante a progressão, se falhou ou atingiu o objetivo, etc, poderiam ser salvas para estudos futuros. Esse tipo de informação é comumente armazenada em Learning Record Systems, bancos já preparados para receber dados de aprendizado coletados. Outra ferramenta muito frequentemente utilizada é a de Learning Management System, que são plataformas de gestão de cursos, aulas e atividades avaliativas.

1.1 Justificativa

Jogos sérios demonstram um efeito positivo no aprendizado de cerca de 72% dos estudantes de áreas e níveis de escolaridade diversas (BACKLUND; HENDRIX, 2013). O uso de uma API (Application Programming Interface) que colete e armazene logs de jogos educativos, integrada a um sistema de gerenciamento do conteúdo oferecido nesses jogos e que forneça avaliações e testes possibilitaria conhecer melhor o perfil do estudante e como ele aprende. Além disso, uma integração que una conteúdos tradicionais como vídeos e atividades, normalmente já utilizados em salas de aula, com jogos sérios poderia possibilitar a comparação dos resultados obtidos em cada uma das plataformas, além de facilitar modificações nas abordagens e melhorias da metodologia de ensino. Ao possibilitar que qualquer aplicação desenvolvida possa utilizar-se dessa arquitetura para armazenar informações relevantes para seu estudo, análises comparativas entre jogos e conteúdos com propósitos distintos seriam facilitadas. A construção dessa ferramenta que permite essa integração dos jogos com outras plataformas e que aumente a dinamicidade do seu

conteúdo também pode auxiliar na popularização dessa tecnologia para estudantes de diversos níveis e áreas. É importante notar, entretanto, que jogos não são substitutos de outras formas de ensino. Como toda ferramenta pedagógica, precisam ser planejados e integrados apenas quando são relevantes para o objetivo final de aprendizado (NGUYEN, 2021).

1.2 Objetivos

O objetivo principal é a integração de um LMS com um LRS e um jogo sério, focados na educação e em realidade virtual. A integração fornecerá conteúdo cadastrado pelo professor para os jogos sérios, e os jogos realizarão logs de uso dos estudantes. Dessa forma, será desenvolvida uma aplicação que permita a integração de um Learning Record System com um Learning Management System.

Objetivos específicos são:

- Modelar arquitetura que conecta o LMS com o LRS e o jogo, delimitando componentes e funcionalidades necessárias
- Selecionar componentes LMS e LRS que irão compor cada módulos
- Realizar melhorias necessárias em cada componente dos módulos
- Implementar funcionalidades e componentes não contemplados pelos LMS e LRS selecionados
- Integrar os componentes, formalizando a arquitetura

1.3 Organização do Trabalho

No Capítulo 1, é apresentada a introdução do trabalho, bem como a justificativa e os objetivos gerais e específicos. No Capítulo 2, é feita a revisão bibliográfica, contendo a fundamentação teórica e os trabalhos relacionados. Logo após, no Capítulo 3 é descrito o funcionamento da arquitetura proposta e quais foram os componentes selecionados para integrá-la, bem como o que seria necessário desenvolver. O capítulo 4 apresenta os resultados obtidos com o desenvolvimento, e um teste realizado. O Capítulo 5 relaciona as conclusões obtidas e uma listagem dos trabalhos futuros.

2 Revisão Bibliográfica

2.1 Fundamentação Teórica

2.1.1 Jogos Sérios

Existem várias definições de jogos e também vários conceitos chaves que são comuns entre a maioria delas. Salen, Tekinbaş e Zimmerman (2004) definem jogos como “um sistema no qual os jogadores engajam em um conflito artificial, definido por regras, que tem um resultado quantificável”. Laamarti, Eid e Saddik (2014) definem ainda videogames como “um tipo especial de jogos onde se joga com um computador seguindo um conjunto de regras com o objetivo geral de entretenimento, recreação, ou o ganho um prêmio ou recompensa”. Conceitos como regras, conflitos, competição, resultado, feedback, cooperação também são citados ao definir jogos (SALEN; TEKINBAŞ; ZIMMERMAN, 2004).

Jogos sérios são uma categoria específica de jogos. A definição mais comum é a de "jogos que não tem o entretenimento, prazer ou diversão como seu objetivo principal"(MICHAEL; CHEN, 2005). Dessa forma, jogos educacionais são incluídos, mas também jogos com outros propósitos como: treinamento, reabilitação, marketing e desenvolvimento social (BACKLUND; HENDRIX, 2013). A definição tomada para esse estudo é a de que jogos sérios "são atividades desenvolvidas para entreter usuários em um ambiente no qual eles também podem aprender e ser educados e treinados em tarefas e áreas bem definidas"(CHECA; BUSTILLO, 2020).

2.1.2 Learning Analytics

Learning Analytics podem ser definidas como a medição, coleta, análise e a visualização de dados sobre aprendizes e o contexto onde estão inseridos. Tem como propósito o entendimento e a otimização do aprendizado e do ambiente onde o apredizado ocorre. (LANG et al., 2017)

2.1.3 API

Uma API é uma interface lógica feita para comunicação entre computadores ou componentes de software que esconde os detalhes internos que são necessários para implementá-la. Ela oferece uma abstração de alto nível e promove a reutilização de código ao possibilitar que várias aplicações compartilhem as mesmas funcionalidades. Termos comuns de engenharia de software como desenvolvimento modular, reuso de código, componentização, arquitetura orientada à serviços, entre outros, indicam a necessidade de um bom desenvolvimento de APIs (REDDY, 2011).

2.1.4 xAPI

A Experience API (ou xAPI) é parte da Arquitetura de treinamento e aprendizado (TLA) que o projeto de aprendizado distribuído avançado (ADL) desenvolveu. Originalmente denominada Tin Can API, a xAPI é uma especificação código aberto para aplicações digitais de ensino. A especificação foi feita para auxiliar na coleta de dados e informações sobre o aprendizado e a experiência de usuários dessas aplicações, seja de forma online ou de forma local (LIM, 2015). A xAPI provê um grupo de normas para armazenar e recuperar afirmações sobre as atividades de um usuário. A forma padrão dessas afirmações é a de ⟨ator, verbo, objeto⟩(SECRETAN; WILD; GUEST, 2019). Por exemplo: ⟨João, leu, o documento sobre imigração⟩. Os dados são estruturados utilizando-se do formato JSON.

2.1.5 Learning Records Store — LRS

Um *Learning Records Store* (ou LRS) foi definido na especificação da xAPI como um servidor que é responsável por receber, armazenar e prover acesso a Learning Records. Um LRS é desenvolvido para armazenar e recuperar afirmações xAPI, armazenar estados da xAPI e salvar vários outros metadados relacionados (ADL, 2021).

2.1.6 Learning Management System — LMS

Um *Learning Management System* (ou LMS) é um portal online que conecta professores e estudantes. Ele provê meios para atividades e materiais de aula serem compartilhados de forma simples e rápida. Também permite discussões em fóruns e outras atividades e interações extraclasse (ADZHARUDDIN; LING et al., 2013).

2.1.7 Learning Analytics

Learning analytics é o nome dado ao uso de *analytics* para descobrir informações sobre o processo de aprendizado e representar e entender as experiências e comportamentos dos usuários. Esses estudos ocupam um espaço vital para compreender detalhes de como o aprendizado se dá, auxiliando na identificação e validação de processos, resultados e atividades. Além disso, cria evidências que podem servir de apoio para práticas a partir de avaliações do progresso dos usuários, motivação, atitudes e satisfação (MANGAROSKA; GIANNAKOS, 2018).

2.1.8 JSON

Javascript Object Notation (JSON) é um formato que foi desenhado para ser utilizado em trocas de dados, de forma que fosse ao mesmo tempo legível por seres humanos e fácil de converter e utilizar em computadores (NURSEITOV et al., 2009). É independente de qualquer linguagem, mas usa convenções conhecidas por diversas linguagens. Nos últimos anos, o formato JSON

ganhou muita popularidade entre desenvolvedores web e se tornou o principal formato de troca de informações pela web (PEZOA et al., 2016).

2.2 Trabalhos Relacionados

Obikwelu e Read (2012) fizeram uma proposta de modelo de framework construtivista para auxiliar no aprendizado de crianças. É discutida a diferença de propostas de aprendizado construtivistas e cognitivas, e benefícios do modelo construtivista, que reconhece que cada pessoa aprende baseado em suas próprias experiências e percepções, através de exploração e contexto social. Algumas limitações de outros modelos existentes são citadas, como a falta de uma etapa de "desapego", onde a criança passa a se tornar mestre naquela determinada tarefa ou atividade, e as pessoas autoras citam como trabalhos futuros a possibilidade de testar esse framework e encontrar possíveis melhorias.

Já Vidakis et al. (2019) desenvolveram a Experiencie API (ou xAPI) para log de jogos e utilização desses logs em serviços de *analytics*. Utilizando-se de JSONs pré-definidos, logs são gerados a partir de interações do usuário, indicando acessos a funcionalidades, resultados corretos ou incorretos em questionários, textos lidos, etc. Esses dados são salvos e podem ser visualizados em uma dashboard com alguns indicativos visuais dos resultados gerais. Entretanto, a xAPI foi feita em C# na arquitetura mono, e somente funciona com aplicativos desenvolvidos nessa linguagem e arquitetura. Além disso, foi apontada a falta de uma documentação acessível para desenvolvedores futuros e a falta de uma padronização dos dados de saída.

Serrano-Laguna et al. (2017) desenvolveram a xAPI Profile, que é a implementação de um modelo base para se utilizar *Learning Analytics* em jogos sérios. A implementação foi feita em conjunto com a xAPI, por ser considerada um modelo promissor na área. A xAPI Profile é utilizado em um jogo demo e explora os aspectos práticos da implementação da normalização dos dados obtidos em jogos sérios. O trabalho foi considerado um primeiro passo na área de *analytics* em jogos sérios, em especial por cobrir apenas os casos mais comuns e genéricos do aprendizado. Um possível trabalho futuro seria estender essas possibilidades de métricas para nuances de mecânica e outros detalhes que também influenciam no aprendizado. Outro problema identificado foi que, ao se utilizar da xAPI em uma situação que exigia muitos logs, havia um aumento grande do uso de banda, que poderia gerar problemas. Propostas de melhorias envolviam a remoção de alguns dados considerados padrão, deixando a inserção deles a cargo do *back-end*, ou a codificação do formado JSON em um formato mais eficiente para ser utilizado apenas durante a transmissão dos dados, mantendo a codificação do lado do jogo e a decodificação a cargo do *back-end*.

Alonso-Fernandez et al. (2019) realizaram uma revisão sistemática da literatura para definir como técnicas de Ciência de Dados têm sido utilizadas em dados de Game Analytics e de Learning Analytics. Foi identificado que a maioria dos estudos se interessa na avaliação

de estudantes e também em seus comportamentos, sugerindo um interesse em medir qual o impacto de um jogo no usuário e como isso se relaciona aos comportamentos do usuário dentro do jogo. No quesito de análise, técnicas de visualização, supervisionadas e não-supervisionadas foram utilizadas. Dentre os modelos, os mais comuns eram os modelos lineares, de correlação ou clusterização. Os principais *stakeholders* são *designers*, desenvolvedores e pesquisadores, seguidos por professores e educadores em geral. A maioria dos jogos ensinavam tópicos relacionados a ciência, principalmente matemática. Os dados coletados envolvem principalmente tempo para conclusão das atividades, ações, interações e pontuações. Também reportaram que não encontraram nenhum *data set* aberto de *analytics* de jogos ou de *analytics* de aprendizado em jogos sérios, e apontam esse caminho como algo a ser pesquisado e coletado dentro da área futuramente. Suas conclusões apontavam que as informações coletadas poderiam ajudar a prever o impacto de aprendizado ao se utilizar jogos sérios, e também o quanto é importante conhecer o perfil dos estudantes que irão utilizar a aplicação, utilizando-se de técnicas de criação de perfil de usuário ou de clusterização.

2.3 Lições aprendidas

Jogos podem ser utilizados como diversão, mas também promover o ensino e desenvolver habilidades. Jogos que possuem foco em algo além da diversão são chamados de jogos sérios. Nota-se uma importância na coleta de dados em jogos sérios, e uma busca pela construção de uma padronização dos logs gerados para *analytics*, de forma a facilitar análises futuras e também a construção de data sets que possam ser analisados em conjunto. É importante que as ferramentas de log sejam suficientemente adaptativas para permitir a inserção de atividades padrão, mas também de informações e dados específicos de cada aplicação. Entretanto, um padrão constante ainda precisa estar bem definido para que não se prejudique o resultado final da análise.

3 Desenvolvimento

Atualmente, existem várias soluções para resolver individualmente problemas relacionados ao ensino. Para criação e manutenção de aulas e cursos online, por exemplo, temos a disposição diversos LMS. Para produzir e armazenar logs de jogos sérios também existem propostas. Entretanto, ainda não há uma ferramenta disponível que crie uma ponte entre esses dois componentes e mescle os resultados obtidos em cada uma dessas plataformas. Outra carência observada é a de jogos educativos que possuam conteúdo dinâmico, que pode ser alterado com facilidade pelo professor, e um método que assegure esse conteúdo dinâmico.

A proposta de uma arquitetura que ligue esses componentes se encontra na Figura 3.1. Nessa arquitetura haveria, além de um LRS e um LMS, um terceiro componente, chamado de Learning Analytics Plataform (LAP). Essa plataforma, junto das APIs necessárias para alimentá-la, promoveria a união dos conteúdos de um LRS com os conteúdos de um LMS, gerando uma visão mais completa de como cada estudante lida com as plataformas e como isso reflete nas atividades que eles executam e no testes ou provas que realizam, por exemplo. O desenvolvimento do LAP está fora do escopo desse trabalho, mas é um passo futuro que se tornará possível após a implementação da versão inicial da arquitetura. Outro componente importante da arquitetura são os jogos sérios, e seu desenvolvimentos será abordado em outros trabalhos.

Existem ferramentas que resolvem individualmente algumas dessas situações, como a de salvar os dados de aprendizado, ou prover uma plataforma de ensino, mas ainda não há destaque para alguma arquitetura que promova a integração dessas propostas. Dessa forma, o primeiro passo foi definir os requisitos necessários para atender essa arquitetura, e, posteriormente, selecionar quais ferramentas individuais disponíveis seriam selecionadas para integrar a solução completa.

3.1 Levantamento de Requisitos para implementação da arquitetura

Como primeiro passo, foi feita a descrição dos requisitos necessários para que a primeira versão da arquitetura fosse concluída. Essa composição precisa conter elementos o suficiente para caracterizar a proposta, e possibilitar melhorias futuras que aumentem o leque de funcionalidades e possibilidades de uso.

3.1.1 Requisitos Funcionais

Os requisitos funcionais definidos para o trabalho são listados a seguir:

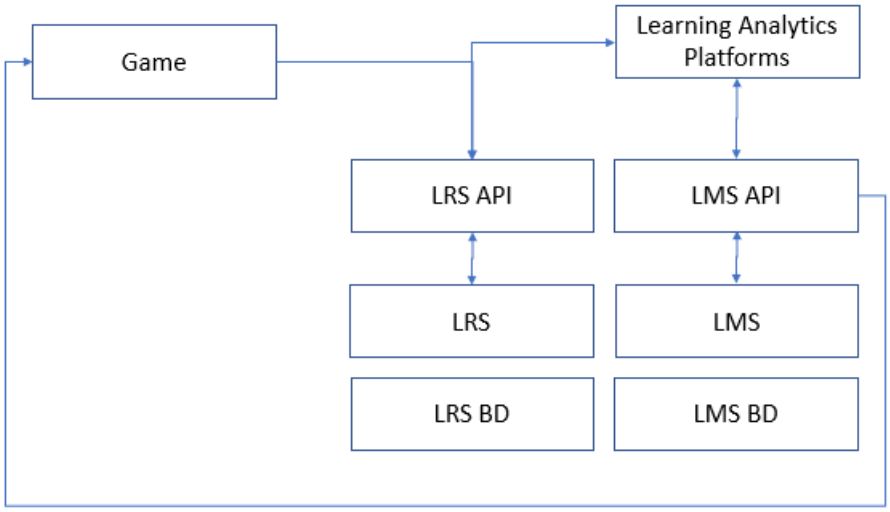

Figura 3.1 – Visualização da proposta da arquitetura

- Cadastro de aulas.
- Cadastro de aulas para jogo.
- Inserção de slides no jogo.
- Inserção de vídeos no jogo.
- Um questionário poderá ser associado a uma aula para jogo.
- Poderá ser criada uma sequência de aulas, gerando um curso.
- Alunos podem se cadastrar na plataforma e em cursos e acessar e concluir esses cursos.
- Conteúdo de aulas para jogo precisa estar disponível via API com todos os dados associados.
- Deve ser possível marcar uma aula para jogo como finalizada via API.
- Deve ser exibido na progressão do curso no LMS quando uma aula é para jogo.
- Deve ser possível registrar via API informações sobre a aula para jogo no LRS.

3.1.2 Requisitos não funcionais

- As ferramentas selecionadas para compor a arquitetura precisaram ser gratuitas e código aberto.
- Os critérios de seleção devem considerar a possibilidade de personalização das ferramentas para atingir as necessidades da aplicação.

3.2 Seleção, customização e implementação dos componentes

Dados os requisitos funcionais listados, e a proposta da arquitetura, foi feita uma busca dos componentes já disponíveis que poderiam atender as necessidades do projeto. Além disso, foram levantadas as adaptações necessárias para cada módulo e o que precisaria ser implementado, pois as opções disponíveis não existem ou não atendem a proposta.

3.2.1 LRS API

Para analisar como a forma de interagir com um jogo sério afeta o aprendizado dos estudantes, é importante que logs de uso sejam gerados. Esses dados, no entanto, são geralmente gerados em plataformas múltiplas, que possuem seu próprio vocabulário que pode ser incompatível e prejudicar análises futuras. Por exemplo, uma plataforma de vídeo pode gerar dados relacionados a assistir vídeo, pausar, fazer anotações, etc. Por outro lado, um leitor de documentos pode ter sua própria proposta, como abrir arquivo, grifar alguma parte, tempo de leitura, etc. A diferença na forma de salvar esses dados pode gerar um ruído que dificulte ou até inviabilize análises futuras (BAKHARIA et al., 2016).

Foi feita uma busca na literatura de APIs que permitissem o log de Learning Analytics, e a xAPI foi selecionada por representar o estado da arte dentro dessa área. Ela foi publicada com a proposta de unificar a forma de se salvar informações sobre o aprendizado, facilitando análises futuras e organizando os dados. Ela propõe o uso de um padrão para o JSON salvo, no formato "ator-verbo-objeto". Na Figura 3.2 observa-se um exemplo de JSON aceito pela xAPI. Ele carrega um ator, seu nome e um identificador; um verbo, que pode ser definido em um banco próprio do projeto ou utilizado de uma lista proposta pela API, e um objeto, que deve ser definido e documentado por cada projeto.

Figura 3.2 – Exemplo de JSON aceito pela xAPI
Fonte: https://xapi.com/statements-101/

3.2.1.1 xAPI Registry

Para realizar o log de uma informação no padrão da xAPI, é necessário o formato "ator-verbo-objeto". Portanto, é necessário definir o melhor verbo e objeto que definem cada interação dos estudantes.

A xAPI provê um registro de verbos, objetos (chamados atividades) e outros itens que podem auxiliar no log das informações. O *xAPI Registry* tem o objetivo de manter essa coleção de registros utilizáveis sem que seja necessário criar uma lista própria para cada projeto desenvolvido. Ele foi criado e mantido pela comunidade, dessa forma, auxiliando também na padronização de projetos e sendo uma referência segura.

No *xAPI Registry*, cada verbo ou atividade contém um identificador, um link associada e uma definição, como observado na Figura A.1. O identificador e a url são utilizados no registro dos dados, tornando possível encontrar o significado de forma rápida.

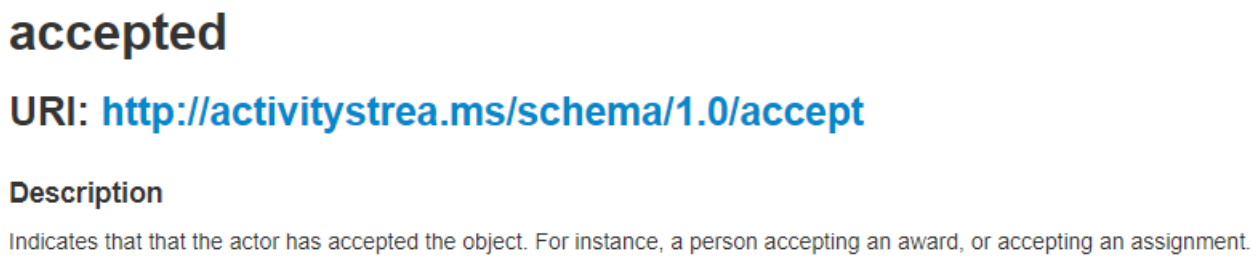

Figura 3.3 – Exemplo de um verbo na xAPI Registry
Fonte: https://registry.tincanapi.com/

Portanto, o xAPI Registry foi escolhido como fonte das definições de verbos e atividades utilizadas.

3.2.1.2 Integração da xAPI com jogos sérios

Existem clientes da xAPI disponíveis para diversas linguagens, que padronizam o envio das informações e facilitam a conexão com um LRS. O cliente escolhido foi uma biblioteca chamada TinCan.NET, que pode ser integrada ao Unity e utilizada em aplicações desenvolvidas em C#.

Embora seja uma facilitadora, a biblioteca ainda precisa de ajustes que exigem tempo e estudo. Para facilitar usos futuros, foi desenvolvida uma camada auxiliar que facilita a implementação e uso da xAPI por parte de futuros desenvolvedores. Ela é composta de *Helpers*, que facilitam a criação de *Statements* e envio dessas informações para o BD LRS escolhido. Além disso, padroniza a declaração e armazenamento de verbos e atividades a serem utilizados por uma aplicação.

3.2.2 LRS e LRS BD

Para salvar os dados obtidos com a xAPI, existem várias soluções de LRS disponíveis atualmente. Aplicações robustas e complexas podem exigir uma instalação dedicada em nuvens como Amazon™, Google Cloud™ e similares, ou até soluções já pré-configuradas, como a ScormCloud™. Entretanto, para soluções menores ou no contexto de desenvolvimento e lapidação dessa arquitetura, uma hospedagem de simples instalação e gratuita é mais interessante. A própria ScormCloud, anteriormente citada, possui uma modalidade gratuita que dá suporte a até 100MB de dados e 10 registros simultâneos. Um mesmo estudante fazendo várias atividades ligadas a um único sistema de aprendizado conta como apenas um registro. Dessa forma, a opção gratuita do ScormCloud é suficiente para os testes necessários e a construção de uma versão inicial da arquitetura.

O ScormCloud já é preparado para receber afirmações cadastradas pela xAPI. Além disso, possui uma interface pré-configurada para visualização dos dados, conforme visto na Figura 3.4. Ele faz uso MySQL como banco de dados, e verifica antes de realizar as inserções se os dados enviados se adéquam a proposta da xAPI.

Ele é facilmente integrado a um cliente da xAPI, bastando criar um banco de dados na conta do ScormCloud e então utilizar as informações desse banco para realizar a conexão com o cliente. Ao criar a conexão deve-se inserir a URL identificadora do banco e uma combinação de chave e segredo de chave.

Figura 3.4 – Exemplo de visualização de uma informação de teste salva na ScormCloud

3.2.3 LMS e LMS BD

O último módulo a ser selecionada para a composição da arquitetura é um LMS. Os dois LMSs mais famosos disponíveis atualmente são o Moodle, como pode ser observado na Figura 3.5, plataforma de ensino código livre e gratuita, e o LearnPress, um plugin código livre e gratuito

para Wordpress que acrescenta funcionalidades de ensino e cursos, como pode ser observado na Figura 3.6. No quesito precificação, a principal diferença é que o LearnPress possui inúmeras funcionalidades extras pagas, que facilitam algumas atividades.

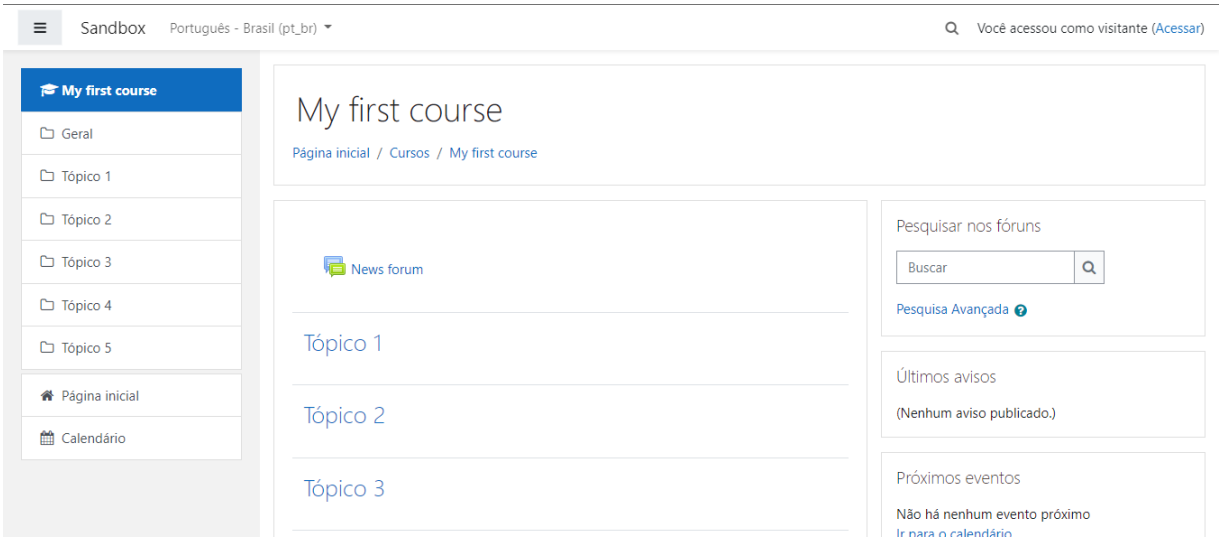

Figura 3.5 – Interface do Moodle após instalação limpa

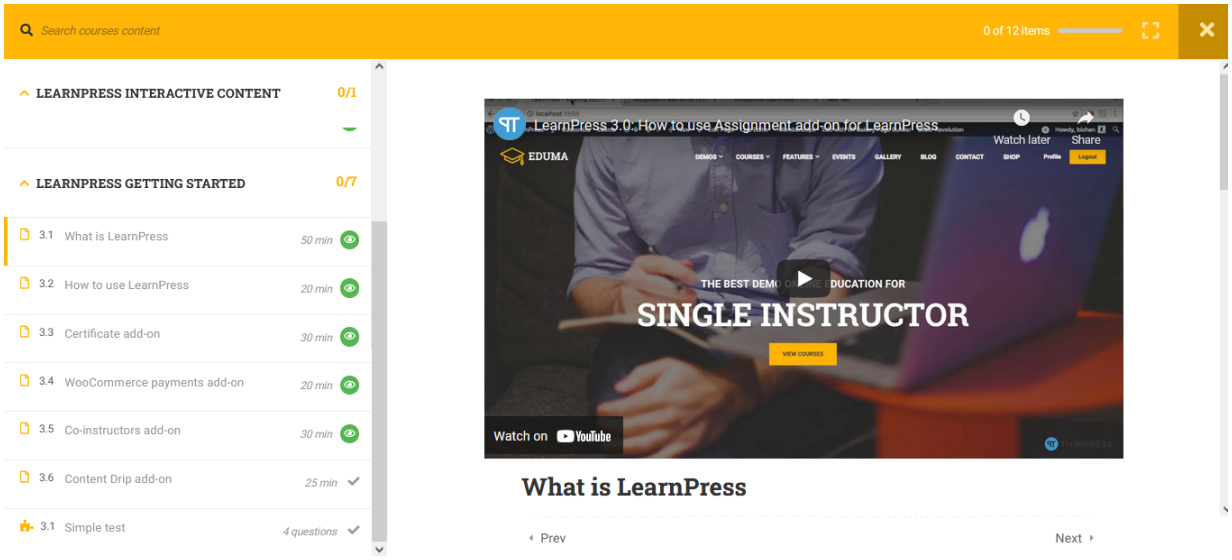

Figura 3.6 – Exemplo de site configurável com o plugin LearnPress.
Fonte: Fonte: https://thimpress.com/learnpress-plugin/

No que diz respeito as funcionalidades, alguns critérios baseados nos requisitos foram eleitos para comparar às duas plataformas e facilitar a escolha por uma delas, sendo eles:

1. O professor precisa poder cadastrar um curso com aulas, atividades, provas, etc.
2. Deve ser possível marcar uma aula como finalizada via API.
3. Deve ser possível criar um espaço para que o professor possa inserir arquivos de personalização dessas atividades/jogos externos e eles precisam ser recuperáveis via API.

No item 1, ambas atendem as expectativas. Apesar de o LearnPress ter muitas funcionalidades voltadas para cursos pagos, ele suporta bem o cadastro de cursos gratuitos e a inscrição de estudantes. O Moodle é uma plataforma conceituada e amplamente utilizada nas instituições de ensino superior com esse propósito.

No item 2, ambas as plataformas exigiriam adaptações. O banco de dados de ambas pode ser acessado e um módulo poderia ser construído que possibilitasse essa funcionalidade. No quesito facilidade de customização, o Wordpress é um sistema bem documentado e possui comunidade ampla com muito conteúdo voltado para auxiliar customizações de sua API. No Moodle, entretanto, boa parte do conteúdo auxilia apenas na criação de customizações dentro de um padrão, e não conta com uma API que já permita chamadas externas.

No item 3, ambas as plataformas exigiriam adaptações. A inserção de arquivos já é uma possibilidade em ambas, mas não de uma forma simples de ser visualizada e organizada. Além disso, a recuperação via API precisaria ser implementada. Novamente, o Wordpress/LearnPress conta com mais suporte a customizações especiais, fora de implementações prévias.

Dessa forma, a plataforma selecionada foi a do LearnPress, por facilitar e suportar modificações que atenderiam as necessidades da arquitetura.

3.2.3.1 Plugins implementados e/ou customizados

O LearnPress, apesar de ser a solução escolhida, não atende todos os requisitos da aplicação. Dessa forma, foram desenvolvidos plugins adicionais, que funcionam em conjunto com o LearnPress.

3.2.3.1.1 Video metabox aoc

O plugin Video Metabox AOC é um plugin open source que possibilita que um vídeo seja adicionado em um post do wordpress. Dessa forma, ele atendia parcialmente as necessidades da arquitetura, e foi selecionado para ser modificado. Foi estendida a funcionalidade, possibilitando que uma lição agora possua zero, um ou mais vídeos associados. Os vídeos são armazenados e a url de acesso é salva no banco para uso posterior.

3.2.3.2 Image metabox

Não foi encontrado plugin gratuito que possibilitasse o upload de zero, uma ou mais imagens em uma lição, e salvar esse conteúdo corretamente no banco, possibilitando recuperação futura. Dessa forma, foi desenvolvido um plugin que abrange essa funcionalidade e atenderia, por exemplo, uma apresentação de slides. As imagens são armazenadas e a url de acesso é salva no banco para uso posterior.

3.2.3.3 LearnPress associate quiz

O LearnPress possui nativamente ferramentas para criar um banco de questões, e, a partir dele, criar questionários. Entretanto, não havia uma forma de atrelar um questionário a uma aula, combinando às duas coisas. Dessa forma, foi desenvolvido um plugin que permite associar um questionário a uma lição. É salvo no banco um identificador do questionário junto do identificador da lição, possibilitando o uso posterior.

3.2.3.4 LearnPress custom lessons

Esse plugin tornou possível marcar uma lição como aula para jogo, e foi desenvolvido para esse trabalho. Dessa forma, a aula ficará marcada como possível de ser recuperada via API e utilizada por jogos sérios.

3.2.3.5 Customização do LearnPress

Uma única customização feita diretamente nas funcionaldades do plugin do LearnPress foi necessária, realizada seguindo as recomendações dos desenvolvedores, para não bloquear atualizações importantes do plugin. A customização foi feita para impedir que o estudante finalize aulas feitas para serem visualizadas pelos jogos sem acessar de fato o jogo disponibilizado.

3.3 LAP

O LAP será composto principalmente por um visualizador de resultados que agregue as informações obtidas com a xAPI e salvas no LRS com os dados de uso do LMS e possíveis métricas obtidas em testes, provas, etc. Esse visualizador deverá inteiramente construído, mas está fora do escopo desse trabalho. Suas principais funcionalidades serão a organização em gráficos, tabelas e similares dos resultados e visualização das informações de forma dinâmica.

3.4 LMS API

A LMS API também foi construída, e é a partir dela que possíveis jogos sérios ou materiais externos se conseguem recuperar informações cadastradas no LMS para personalizar a experiência de usuário. Outra funcionalidade possibilitará que jogos atualizem no LMS as suas informações de uso, por exemplo, para marcar tarefas como concluídas.

Essas funcionalidades foram implementadas no plugin de LearnPress custom lessons compõem a API de acesso às lições customizadas para jogos. Atualmente, existem três rotas definidas que abarcam os requisitos necessários. A primeira valida e-mails, identificando se esse e-mail está cadastrado no banco do LMS ou não. A segunda fornece todo o conteúdo de todas as aulas externas disponíveis no momento naquele LMS, possibilitando que jogos externos

selecionem a aula a ser exibida. Ela retorna junto do conteúdo básico de uma lição (título e descrição), todo o conteúdo associado a ela, como imagens, vídeos e o questionário, com suas perguntas e respostas definidas. A última rota disponível marca uma aula como concluída pelo usuário, utilizada para finalizar as aulas feitas através do jogo.

3.5 Arquitetura proposta

A organização final da arquitetura proposta pode ser observada na Figura 3.7, e sua estrutura pode ser dividida em três módulos: A (composto pela xAPI, ScormCloud e SQL BD), B (composto pela API, LearnPress e Wordpress SQL BD) e Learning Analytics Platform.

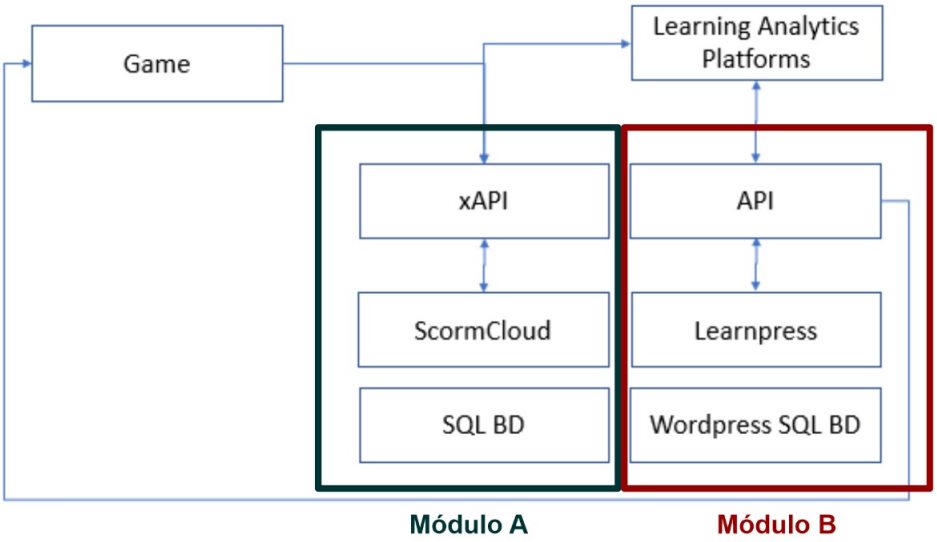

Figura 3.7 – Visualização da arquitetura com os componentes selecionados.

A estrutura da xAPI, ligada ao ScormCloud, proverá um meio de salvar e recuperar informações sobre o uso de estudantes em plataformas externas.Essas tarefas estão englobadas no módulo A. Dessa forma, por exemplo, um jogo sério se utilizará de um cliente xAPI para registrar informações de aprendizado que poderão posteriormente ser utilizadas pelos educadores para avaliar a efetividade de cada modelo de atividade ou aprendizado proposto, comparando se ao assistir um vídeo, por exemplo, os alunos demonstraram maior absorção de conteúdo do que lendo uma sequência de textos.

Ao possibilitar que jogos sérios registrem informações sobre o aprendizado, o educador ganha um leque maior de ferramentas de apoio ao ensino que podem ser utilizadas em conjunto com avaliações padrão e metodologias mais comuns. O LearnPress, nessa arquitetura, supriria o papel de um Learning Manager System, criando um ambiente virtual de sala de aula que permite disponibilizar textos, provas, atividades e conteúdos que os estudantes de um curso

podem executar. Os resultados e logs de cada uma dessas tarefas também é salvo, tornando possível compará-los com os dados obtidos em ferramentas externas acopladas.

Outra responsabilidade do módulo B, será salvar os conteúdos personalizados que ferramentas externas podem precisar. Por exemplo, um mesmo jogo sério poderia ser personalizado a cada módulo de um curso, enviando informações diferentes conforme o estudante avança no conteúdo. Esses dados personalizados serão cadastrados pelo educador e poderão ser obtidos via API.

O último módulo da arquitetura envolve a criação de um Learning Analytics Platform. Essa estrutura receberá as informações salvas de ferramentas externas pela xAPI e as informações salvas pelo LearnPress e proverá meios para facilitar a análise desses dados. Uma padronização será utilizada no modelo de dados, garantindo que as informações possam ser analisadas em conjunto mesmo vindo de fontes diferentes. A proposta de ferramentas de análise de dados como essa é auxiliar a tomada de decisão, permitindo comparar resultados obtidos com diferentes metodologias de ensino e materiais de apoio. Esse módulo não é escopo desse trabalho, e trata-se de um trabalho futuro.

Os dois principais usuários da plataforma serão os educadores e os estudantes. Eles terão acessos a funcionalidades distintas.

3.5.1 Educadores

Os educadores poderão se utilizar da plataforma em três frentes. A primeira virá no formato padrão do LMS, com as funcionalidades de cadastro de conteúdo, aulas e atividades. Aliado a ele, será possível cadastrar também conteúdo para jogos sérios, por exemplo, e incluir informações como imagens, vídeos e perguntas. Além disso, o jogo produzirá logs de uso e estatísticas, que serão devidamente armazenados e recuperáveis. Esse conteúdo gerado poderá ser analisado e visualizado no LAP através de gráficos, comparativos e etc. A padronização proposta pela arquitetura tornará possível analisar dados mistos sem que haja incongruência entre as categorias de dados.

3.5.2 Estudantes

Os estudantes terão acesso às atividades propostas no LMS e também poderão acessar os jogos disponibilizados pelos educadores. Os conteúdos estarão entrelaçados, possibilitando um aprendizado mais dinâmico.

4 Resultados

Para o teste das funcionalidades desenvolvidas, foi necessária uma instalação do wordpress com o plugin LearnPress hospedada nos servidores da UFOP, e foi utilizado um jogo sério de demonstração desenvolvido pelo aluno Arilton Aguilar em seu trabalho de monografia (AGUILAR, 2022). Também foi criada uma conta no ScormCloud na opção gratuita. As próximas seções desse capítulo detalham a implementação dos requisitos descritos na Seção 3.1.

4.1 Cadastro de aulas

Esse requisito foi atendido com a instalação e configuração do plugin LearnPress, que permite a criação de aulas pelos professores. Pode-se observar na Figura 4.1 que as diferentes aulas ficam listadas para os professores, e existe a possibilidade de criar novas lições.

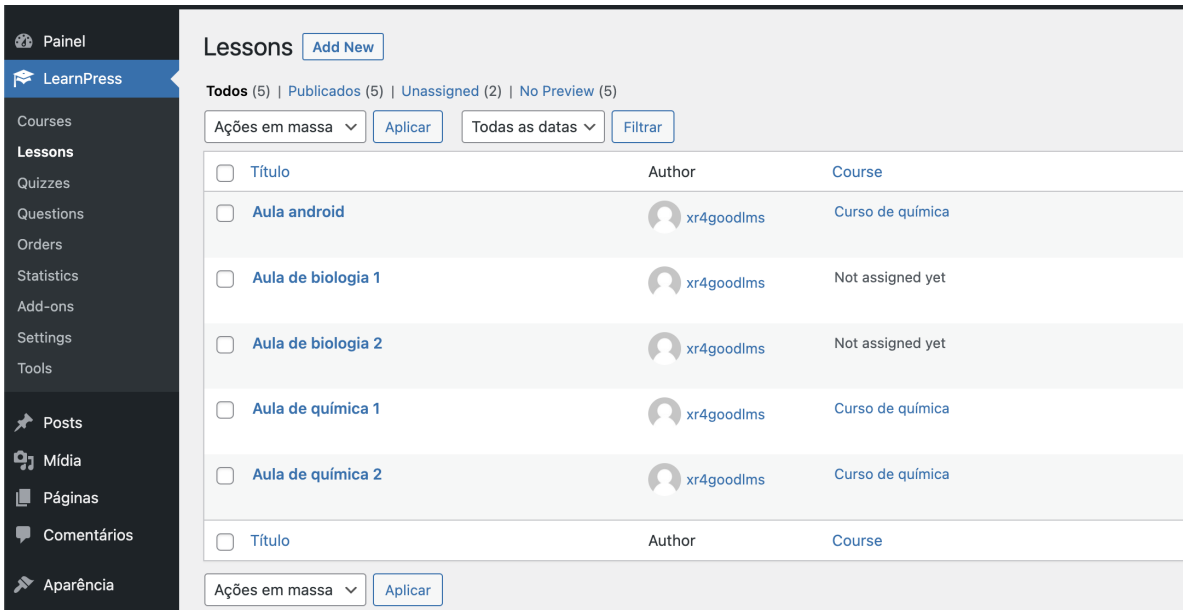

Figura 4.1 – Lista de aulas no LMS e a possibilidade de criar uma nova aula

4.2 Cadastro de aulas para jogo

Para atender o requisito de aulas customizadas para jogo, foi necessário customização. Nenhum dos LMS possuia essa possibilidade, e o LearnPress, junto do Wordpress, possibilitava essa customização e melhoria. O plugin LearnPress custom lessons foi desenvolvido, e conta com a possibilidade de marcar uma aula como "aula externa", definindo assim que a aula marcada tem seu conteúdo projetado para aulas externas ao LMS, em jogos sérios ou outras plataformas. Dessa forma, ele auxilia no filtro de aulas convencionais, que podem ser realizadas do próprio navegador

pelo usuário, das aulas para jogo, que apenas podem ser finalizadas pelo jogo. Observa-se na Figura 4.2 um trecho de código do plugin desenvolvido onde é criado um seletor de aula externa.

```
public function admin_meta_box( $meta_boxes ) {

        if ( version_compare( LEARNPRESS_VERSION, '4.0.0-beta-0', '>=' ) ) {
                $meta_boxes['_lp_custom_lesson_is_for_game'] = new LP_Meta_Box_Checkbox_Field(
                        esc_html__( 'Aulas externas', 'learnpress-custom-lessons' ),
                        esc_html__( 'É uma aula externa.', 'learnpress-custom-lessons' ),
                        'no'
                );
```

Figura 4.2 – Trecho de código do plugin que permite marcar aula para jogo

4.3 Inserção de slides no jogo

Uma funcionalidade desejada dos jogos externos é a exibição de slides customizados pelo professor. Dessa forma, aulas diferentes exibem slides diferentes, possibilitando um aprendizado mais dinâmico. No Learnpress, o envio de imagens para uma lição apenas é possível de ser feito no texto de descrição, misturando texto e imagens. Com esse recurso, a inserção de um conjunto de slides se utilizando apenas dos recursos nativos seria prejudicada, limitando as opções. Foi desenvolvido o plugin Image metabox, que possibilita que o professor faça upload de uma ou mais imagens para uma lição, separadamente do conteúdo textual. Essas imagens ficam salvas no banco, associadas a lição, e podem ser acessadas e recuperadas. Na Figura 4.3 pode-se observar o estado do seletor de imagens após a adição de vários slides a uma aula. Com esse plugin finalizado, cada aula pode possuir um conjunto de slides associados, que pode ser inserido no jogo e terá métricas de visualização associadas.

4.4 Inserção de vídeos no jogo

Outra funcionalidade desejada de jogos externos é a exibição de vídeos personalizados, que podem possuir conteúdo complementar das aulas. A inserção de vídeos também apenas é possível no LearnPress se inserida no texto de descrição, e, fazendo dessa forma, a recuperação fica prejudicada e a exibição no jogo fica limitada ao corpo da descrição. Para sanar essa flexibilização de quantidade e uso dos vídeos, o plugin Video metabox AOC foi customizado para atender essa necessidade. Esse plugin é código aberto, e possibilitava o upload de um vídeo. Com a customização, ele agora aceita o upload de múltiplos vídeos, que são associados a uma lição e podem ser acessados e recuperados. Observa-se na Figura 4.4 o seletor com um vídeo selecionado, e com a possibilidade de adicionar mais. Com esse plugin, a URL de exibição dos vídeos pode ser facilmente recuperados via API e exibida no jogo conforme for necessário.

Figura 4.3 – Seletor de imagens para a lição com algumas imagens adicionadas

Fonte das imagens: Setor de citologia do Laboratório de Análises Clínicas da Universidade Federal de Ouro Preto

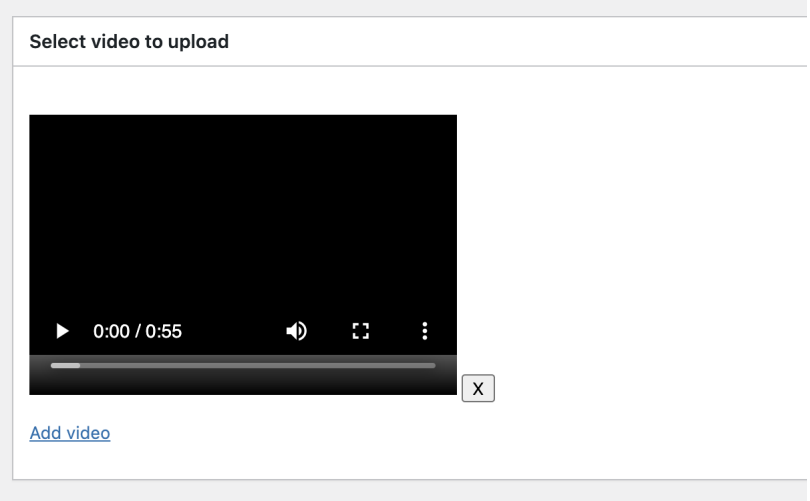

Figura 4.4 – Seletor de vídeos para a lição

4.5 Um questionário poderá ser associado a uma aula para jogo

Um elemento comum de utilizado durante o processo de aprendizado é o questionário. O LearnPress possui a funcionalidade de cadastrar questões, formando um banco de questões, e a possibilidade de criar questionários a partir dessas questões. Entretanto, esses questionários não

podiam ser associados a uma lição, podendo apenas ser tarefas sequenciais dentro de um curso. O plugin LearnPress associate quiz foi desenvolvido e permite associar um dos questionários existentes a uma lição. O identificador do questionário é salvo no banco, junto do identificador da lição, e pode ser posteriormente recuperado para uso. Pode-se observar na Figura 4.5 um exemplo de questionário selecionado para uma aula. Com o plugin, o jogo agora pode utilizar esse questionário associado para testar os conhecimentos obtidos durante uma lição e obter métricas.

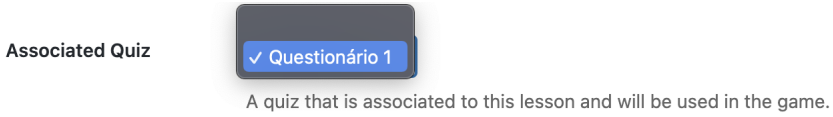

Figura 4.5 – Seletor de questionário para a lição

4.6 Poderá ser criada uma sequência de aulas, gerando um curso

A possibilidade de criar um curso formado pela sequência de aulas aproxima a arquitetura do uso dia a dia dos professores e estudantes,e já existe no plugin LearnPress. Ela permite integrar lições comuns, questionários e lições para jogo, marcadas como aulas externas. Dessa forma, uma sequência pode ser composta de aulas que podem ser realizadas no navegador da preferência do estudante, questionários que podem ser realizados pelo navegador, e aulas que só podem ser finalizadas pelo jogo externo. Essas aulas feitas pelo jogo podem conter texto, vídeo, slides e um questionário.

4.7 Alunos podem se cadastrar na plataforma e em cursos e acessar e concluir esses cursos

Para que as aulas possam ser acessadas por estudantes, e para que as informações de respostas de questionários e o log de uso do jogo possa ser identificado, é importante que os estudantes tenham uma conta que os identifique. Essa funcionalidade já existe no plugin LearPress, permitindo o cadastro de usuários na plataforma, e, posteriormente, permite que eles se cadastrem ou sejam cadastrados nos cursos existentes. Além disso, foi desenvolvida uma rota de API que se utiliza desses cadastros no banco, e valida se um e-mail fornecido via URL existe nos registros.

4.8 Conteúdo de aulas para jogo precisa estar disponível via API com todos os dados associados

Para que o jogo sério possa ser personalizado, seria necessário que uma API disponibilizasse o conteúdo salvo no banco de dados. Informações como título e descrição da aula, slides associados, vídeos fornecidos e questionário, precisam ser recuperáveis. Foi construída uma API no plugin LearnPress custom lessons, que possibilita recuperar as lições para jogo utilizando uma rota específica. É feita uma busca no banco, filtrando as aulas externas. Para que o resultado seja completo, um objeto é construído com todo o conteúdo associado a cada aula, como questionários, com suas perspectivas perguntas e respostas, url de vídeos, imagens e textos. Na Figura 4.6, observa-se uma chamada para a rota de listagem, feita pelo aplicativo Postman, que mostra parcialmente os resultados retornados. Esse resultado pode ser utilizado pelo jogo para exibir os conteúdos customizados corretamente, sem precisar de configurações adicionais no jogo por parte do professor.

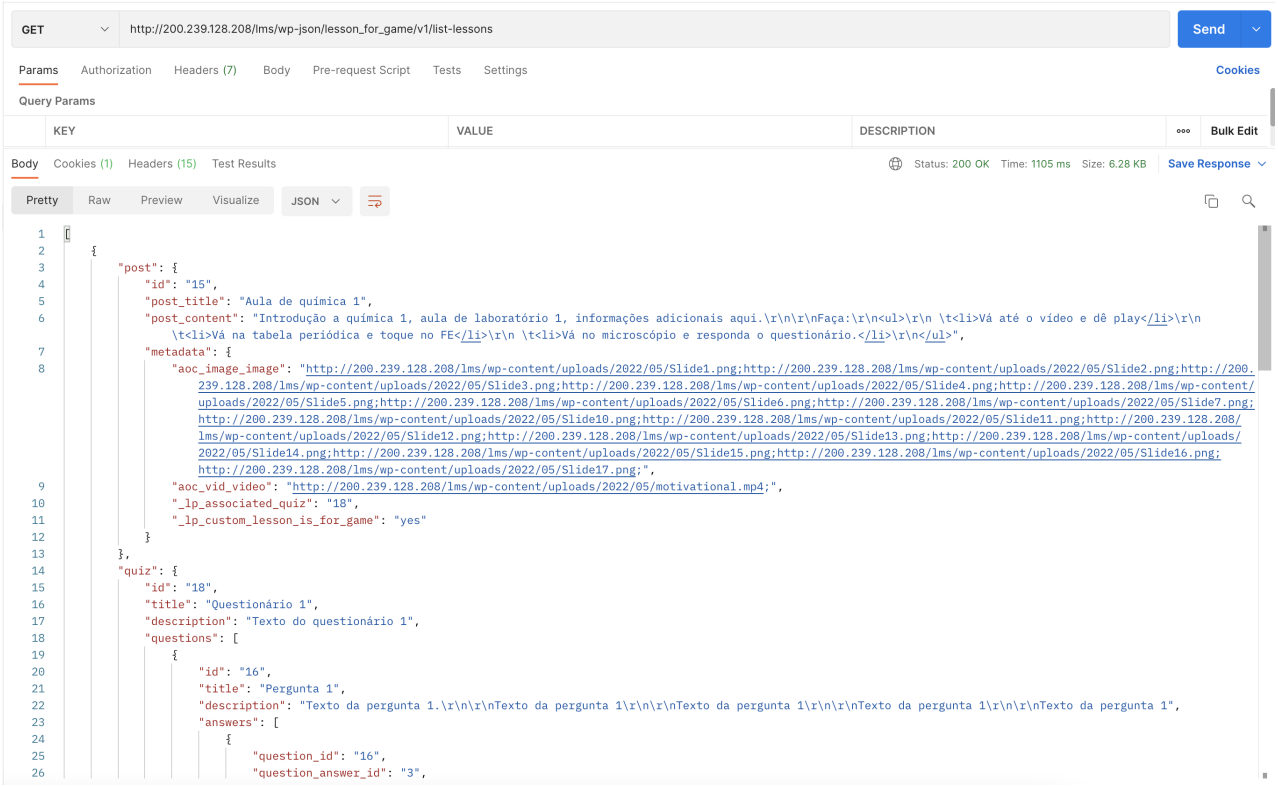

Figura 4.6 – Print do aplicativo Postman, exibindo a chamada para a API com o resultado das aulas armazenadas no banco

4.9 Deve ser possível marcar uma aula para jogo como finalizada via API

As aulas configuradas para jogo não podem ser finalizadas pelo navegador. Essa decisão tem objetivo de levar o estudante a de fato passar pelo jogo, realizando as atividades e cumprindo os objetivos propostos. Dessa forma, a única possibilidade de finalizar tarefas definidas como "para jogo"é pelo jogo. Para marcar a conclusão das atividades, foi desenvolvida uma rota de API no plugin LearnPress custom lessons que possibilita que, dado um e-mail e um identificador de aula, essa aula seja marcada como concluída no banco de dados do LMS. Dessa forma, no próximo acesso do aluno, ela já constará marcada como finalizada.

4.10 Deve ser exibido na progressão do curso no LMS quando uma aula é para jogo

Quando um aluno se cadastra em um curso, ele pode visualizar as aulas pelo navegador. Em cada aula, é possível marcá-la como concluída, e questionários são concluídos após o estudante responder às perguntas e garantir acertos. Para as aulas que são para jogos, foi necessário impossibilitar que o aluno marcasse as aulas como concluídas pelo navegador. Como a remoção do botão de concluir, também surgiu a necessidade de indicar que a aula deve ser concluída utilizando outra plataforma. Para isso, foi desenvolvida uma modificação para o plugin LearnPress que atende esse requisito, e uma aula é exibida no curso como aula externa. Na Figura 4.7 pode-se observar a indicação de que a aula não pode ser finalizada apenas pelo LMS.

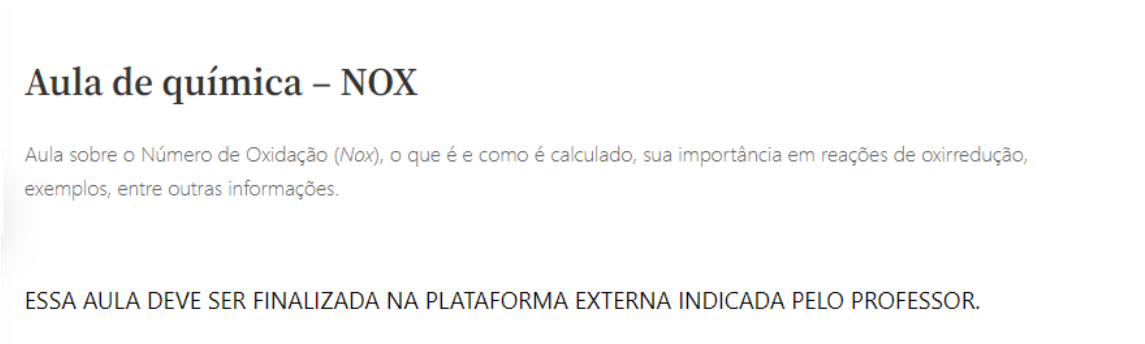

Figura 4.7 – Exibição da lição online indicando que a aula deve ser finalizada em plataforma externa

4.11 Deve ser possível registrar via API informações de uso da aula para jogo no LRS

Para obter informações sobre tempo de uso, acertos e erros, e quais componentes da aula para jogo foram visualizados pelos estudantes, e quais não, é importante salvar as interações que o estudante realiza no jogo. A xAPI propôs um padrão de dados para serem salvos, que facilita comparações futuras. Existe uma biblioteca que é cliente da xAPI na linguagem C# e permite que sejam feitos logs de informações na plataforma Unity, muito comumente utilizada para desenvolvimento de jogos. Foi desenvolvido um *Helper*, também em C#, para facilitar ainda mais o uso dessa biblioteca, padronizando o formato de envios dos logs e a documentação dos verbos e atividades utilizados.

Na Figura 4.8 observa-se o auxiliar para definir verbos com mais facilidade, e na Figura 4.9 o equivalente para as atividades. Ele pode ser expandido para incluir traduções dos itens e também particularidades de cada um. Na Figura 4.10 observa-se um trecho do auxiliar para salvar as definições da xAPI, salvando o ator responsável, verbo, atividade e resultado associado. Há também um feedback para o desenvolvedor indicando se o conteúdo foi salvo ou se houve alguma falha.

O Helper, combinado com a TinCan.NET, possibilita que as informações de uso dos estudantes sejam salvas no LRS e, posteriormente, elas poderão ser analisadas pelos educadores, ou recuperadas e listadas no LAP. Essas métricas de uso, acertos e erros, e listagens de quais textos, vídeos e slides foram vistos e quais não pode auxiliar na tomada de decisão futura dos educadores, e possibilitar melhorias no conteúdo apresentado para os alunos.

C# **Verb.cs** 217 bytes

```
1  public struct Verb
2  {
3      public string url;
4      public string descriptionEnUS;
5
6      public Verb(string url, string descriptionEnUS)
7      {
8          this.url = url;
9          this.descriptionEnUS = descriptionEnUS;
10     }
11 }
```

Figura 4.8 – Arquivo de normalização da criação dos verbos

4.12 Tutoriais

Esse desenvolvimento tem objetivo de ser apenas uma primeira versão dessa arquitetura, deixando portas abertas para melhorias futuras e expansão de funcionalidades. Para facilitar esses

C# **Activity.cs** 226 bytes

```
1  public struct Activity
2  {
3      public string url;
4      public string descriptionEnUS;
5
6      public Activity(string url, string descriptionEnUS)
7      {
8          this.url = url;
9          this.descriptionEnUS = descriptionEnUS;
10     }
11 }
```

Figura 4.9 – Arquivo de normalização da criação das atividades

ReusableStatementSender 2.78 KiB

```
27     public IEnumerator SaveStatement(Agent actor,
28                                      TinCan.Verb verb,
29                                      TinCan.Activity activity,
30                                      Result result) {
31
32         //Build out full Statement details
33         Statement statement = new Statement();
34         statement.actor = actor;
35         statement.verb = verb;
36         statement.target = activity;
37         statement.result = result;
38
39         //Send the statement
40         StatementLRSResponse lrsResponse = lrs.SaveStatement(statement);
41         if (lrsResponse.success) //Success
42         {
43             Debug.Log("Save statement: " + lrsResponse.content.id);
44         }
45         else //Failure
46         {
47             Debug.Log("Statement Failed: " + lrsResponse.errMsg);
48         }
49         yield break;
50     }
51
```

Figura 4.10 – Arquivo que visa padronizar o envio das informações para o xAPI

usos futuros e auxiliar na documentação do projeto, alguns tutoriais foram desenvolvidos. Eles passam instruções básicas de instalação dos plugins desenvolvidos, do uso das APIs disponibilizadas, de instalação e uso do TinCan.NET e do Helper em C#. Esses tutoriais estão disponíveis no Apêndice A.

4.13 Integração em um jogo sério

Foi definido um caso de aula protótipo para teste das funcionalidades, desenvolvido pelo aluno Arilton Aguilar (AGUILAR, 2022). Dessa forma, o primeiro passo para o teste foi definir

quais atividades gerariam logs por meio da xAPI, e como seria cada um desses logs.

A aula protótipo simula uma aula de biologia, onde estudantes podem aprender como limpar e manusear um microscópio. A primeira tarefa do estudante é ler um conjunto de *slides* que contém as informações sobre limpeza e manutenção do microscópio. Foi determinado que seria importante salvar quando e por quem cada um desses *slides* foi lido, e se o estudante concluiu a leitura dos *slides* ou não.

A segunda tarefa do estudante é responder um questionário a respeito do conteúdo aprendido nos slides. Nessa parte, definiu-se que seriam salvos os erros e acertos de cada estudante em cada pergunta, e também seria salvo quando o estudante finalizou o teste, e a porcentagem de acerto do teste.

Junto de toda informação salva é armazenado o nome e e-mail do estudante, além da hora em que a atividade foi realizada. Com o horário salvo, métricas como tempo gasto em determinada atividade ou para responder uma pergunta podem ser obtidos a partir dos dados.

4.13.1 Definição do JSON a ser salvo

Na aula protótipo definida, duas categorias de logs de informações seriam necessários. Para o primeiro caso, o objetivo era salvar a leitura de *slides*, quais páginas foram lidas e quais ainda não, e se o estudante havia finalizado a leitura de todas as páginas.

Para a construção do objeto a ser salvo, foi definido o verbo *viewed*, visualizou, e a atividade *slide*. A xAPI também aceita um objeto de resultado, e nele foi indicado a porcentagem de slides visualizados até o momento. Também foi indicado se o usuário já completou a visualização.

Quando o usuário finaliza a visualização dos slides pela primeira vez, também é salvo um objeto com o verbo *completed*, completou, e a atividade *slide deck*, grupo de slides. No resultado, é indicado que o usuário visualizou 100% dos slides, completando a atividade.

Para a segunda categoria de atividade, o objetivo era indicar os acertos e erros do estudante ao completar um quiz sobre os slides lidos. Para tal, a cada tentativa foi salvo um registro com o verbo *answered* (respondeu) e a atividade *assessment* (prova). Como resultado, era indicado se o usuário acertou ou errou a questão. Além de aceitar verbos e atividades no formato proposto, a xAPI também aceita extensões desses itens. As extensões aceitam um objeto JSON, e servem de complemento com as informações já registradas. Como complemento da atividade foi registrada qual alternativa foi selecionada pelo estudante, e o conteúdo dessa resposta. Isso pode facilitar ao educador identificar as possíveis melhorias de ensino, e também facilita análises posteriores.

Quando o estudante completa o teste, também é salvo um registro com o verbo *completed*, completou, e a atividade *assessment*, prova. A porcentagem de acerto é indicada como resultado.

Pode-se observar na Figura 4.11 o formato final da declaração dos verbos a serem utilizados, realizado com o auxílio do *Helper*. Na Figura 4.12, observa-se a lista de Atividades para

essa aplicação, também codificada utilizando-se o *Helper*.

C# LogVerb.cs 378 bytes

```
public class LogVerb
{

    public static Verb Viewed   { get { return new Verb("http://id.tincanapi.com/verb/viewed", "viewed"); } }
    public static Verb Completed   { get { return new Verb("http://activitystrea.ms/schema/1.0/complete", "completed"); } }
    public static Verb Answered   { get { return new Verb("http://adlnet.gov/expapi/verbs/answered", "answered"); } }
}
```

Figura 4.11 – Arquivo dos verbos a serem utilizados, realizado com auxílio o *Helper*

C# LogActivity.cs 551 bytes

```
public class LogActivity
{

    public static Activity Slide   { get { return new Activity("http://id.tincanapi.com/activitytype/slide", "slide"); } }
    public static Activity SlideDeck   { get { return new Activity("http://id.tincanapi.com/activitytype/slide-deck", "slide-deck"); } }
    public static Activity Question { get { return new Activity("http://activitystrea.ms/schema/1.0/question", "question"); }}
    public static Activity Assessment { get { return new Activity("http://adlnet.gov/expapi/activities/assessment", "assessment"); }}
}
```

Figura 4.12 – Arquivo das atividades a serem utilizadas, realizado com o auxílio do *Helper*

4.14 Realizando logs de uma aula teste

Após todas as integrações necessárias, os logs foram feitos corretamente no ScormCloud. Informações sobre as atividades definidas foram salvas, e métricas poderiam ser obtidas a partir das mesmas. O estudante conseguiu finalizar a aula do jogo corretamente, e continuar o curso online no LMS. Observa-se na Figura 4.13 um exemplo de informação salvo no ScormCloud via xAPI. Na Figura 4.14 observa-se a aula após finalizada pelo aluno via jogo. Na Figura 4.15, observa-se uma imagem de dentro do jogo desenvolvido por Arilton Aguilar (AGUILAR, 2022), utilizando da arquitetura proposta para alimentar o conteúdo e salvar registros.

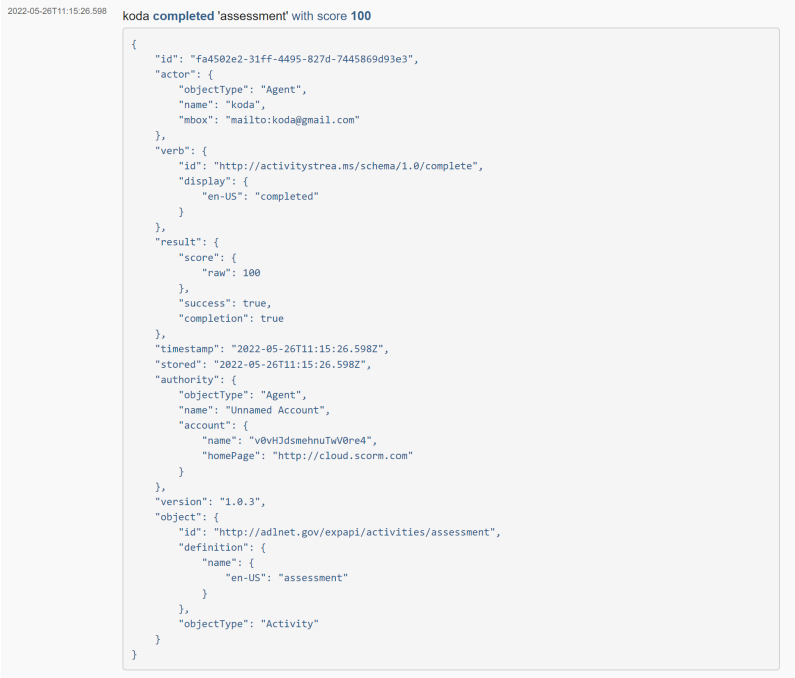

Figura 4.13 – Print das informações salvas na xAPI

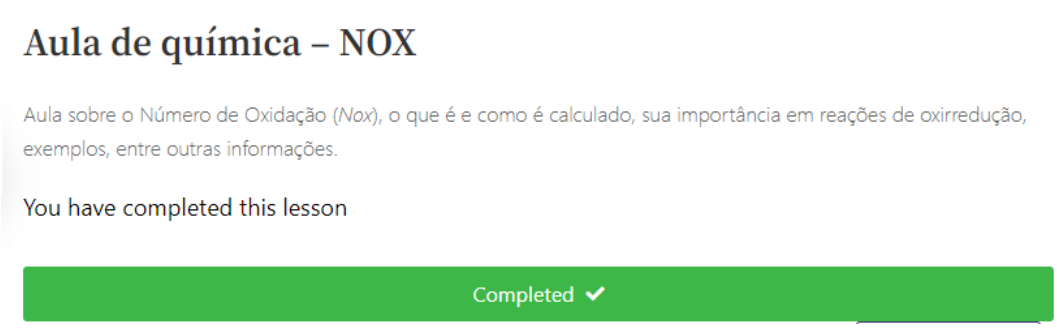

Figura 4.14 – Exibição da lição online indicando que a aula foi concluída

Figura 4.15 – Imagem de uma aula dentro do jogo sério desenvolvido por Arilton Aguilar

5 Considerações Finais

A utilização de jogos sérios como ferramenta de apoio ao ensino tem crescido nos últimos anos, e demonstra efeito positivo no aprendizado de boa parte dos estudantes. Entretanto, ainda há lacunas na compreensão de que fatores influenciam positiva ou negativamente no aprendizado e de como possíveis melhorias podem ser aplicadas para maximizar os resultados. Além disso, não há uma ferramenta prática que possibilite a análise cruzada de informações obtidas em testes clássicos, como provas e atividades, com outras ferramentas de apoio, como jogos sérios. Dessa forma, esse trabalho implementou uma arquitetura que possibilite esse tipo de análise, criando uma integração entre LMS e LRS.

Inicialmente, foram definidos os componentes a serem utilizados para o LRS, a xAPI e o LMS. Além disso, também foram definidos os requisitos para compor a API de integração que será desenvolvida e quais são as funcionalidades esperadas do LAP que possibilitará as análises dos dados.

Com base na arquitetura idealizada, foi realizada a implantação e modificação do LearnPress, o LMS selecionado. Para realizar a instalação, foi utilizada a nuvem da UFOP para hospedagem.

Após a instalação do LMS, foi necessário configurar o ScormCloud e conectá-lo a um cliente externo da xAPI. O cliente escolhido foi o TinCan.NET, por conversar com projetos feitos em Unity, e, junto dele, foi desenvolvido um *Helper* auxiliar, para facilitar a comunicação com o ScormCloud e o envio de informações no formato da xAPI.

O próximo passo foi a implementação de uma API que acrescentou funcionalidades ao LearnPress. Ela permite o envio de conteúdos personalizados para jogos externos, e também atualiza informações de aulas realizadas por estudantes.

O sistema de visualização de métricas foi definido como um trabalho futuro e ficou fora do escopo do desenvolvimento. Seu objetivo é receber dados do ScormCloud e da API do LMS para gerar gráficos, tabelas e outras informações relevantes para o educador que auxiliem a tomada de decisão.

Essa arquitetura possibilita um primeiro passo, criando possibilidades de expansão de usos e funcionalidades.

5.0.1 Trabalhos futuros

Como trabalhos futuros desse projeto, a implementação do Learning Analytics Plataform pode ser efetuada, facilitando a análise dos dados. Além disso, pode haver um incremento que permita mais formatos de mídias associadas às lições, e a criação de categorias de lição, com

formatos de mídia pré-associados. Outro trabalho futuro relevante é o teste formal da arquitetura, para encontrar pontos de melhoria e validar sua efetividade e funcionalidades.

Referências

ADL. *What is an LRS? Learn more about Learning Record Stores*. 2021. Disponível em: <https://xapi.com/learning-record-store/>.

ADZHARUDDIN, N. A.; LING, L. H. et al. Learning management system (lms) among university students: Does it work. *International Journal of e-Education, e-Business, e-Management and e-Learning*, IACSIT Press, v. 3, n. 3, p. 248–252, 2013.

AGUILAR, A. Jogos sérios em realidade virtual para ensino: Caso de estudo aulas laboratoriais em ciências biológicas. 2022.

ALONSO-FERNANDEZ, C.; CALVO-MORATA, A.; FREIRE, M.; MARTINEZ-ORTIZ, I.; FERNÁNDEZ-MANJÓN, B. Applications of data science to game learning analytics data: A systematic literature review. *Computers & Education*, Elsevier, v. 141, p. 103612, 2019.

BACKLUND, P.; HENDRIX, M. Educational games-are they worth the effort? a literature survey of the effectiveness of serious games. In: IEEE. *2013 5th international conference on games and virtual worlds for serious applications (VS-GAMES)*. [S.l.], 2013. p. 1–8.

BAKHARIA, A.; KITTO, K.; PARDO, A.; GAŠEVIĆ, D.; DAWSON, S. Recipe for success: lessons learnt from using xapi within the connected learning analytics toolkit. In: *Proceedings of the sixth international conference on learning analytics & knowledge*. [S.l.: s.n.], 2016. p. 378–382.

BOYLE, E. A.; HAINEY, T.; CONNOLLY, T. M.; GRAY, G.; EARP, J.; OTT, M.; LIM, T.; NINAUS, M.; RIBEIRO, C.; PEREIRA, J. An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. *Computers & Education*, Elsevier, v. 94, p. 178–192, 2016.

CHECA, D.; BUSTILLO, A. A review of immersive virtual reality serious games to enhance learning and training. *Multimedia Tools and Applications*, Springer, v. 79, n. 9, p. 5501–5527, 2020.

EGENFELDT-NIELSEN, S. Overview of research on the educational use of video games. *Nordic Journal of Digital Literacy*, Universitetsforlaget, v. 1, n. 03, p. 184–214, 2006.

LAAMARTI, F.; EID, M.; SADDIK, A. E. An overview of serious games. *International Journal of Computer Games Technology*, Hindawi, v. 2014, 2014.

LANG, C.; SIEMENS, G.; WISE, A.; GASEVIC, D. *Handbook of learning analytics*. [S.l.]: SOLAR, Society for Learning Analytics and Research New York, NY, USA, 2017.

LIM, K. C. Case studies of xapi applications to e-learning. In: *The twelfth international conference on eLearning for knowledge-based society*. [S.l.: s.n.], 2015. p. 3–1.

MANGAROSKA, K.; GIANNAKOS, M. Learning analytics for learning design: A systematic literature review of analytics-driven design to enhance learning. *IEEE Transactions on Learning Technologies*, IEEE, v. 12, n. 4, p. 516–534, 2018.

MICHAEL, D. R.; CHEN, S. L. *Serious games: Games that educate, train, and inform*. [S.l.]: Muska & Lipman/Premier-Trade, 2005.

MORGENSTERN, E. *O mar sem estrelas*. [S.l.]: Morro Branco, 2019.

NGUYEN, H. P. *How to use gameplay to enhance classroom learning*. George Lucas Educational Foundation, 2021. Disponível em: <https://www.edutopia.org/article/how-use-gameplay-enhance-classroom-learning>.

NURSEITOV, N.; PAULSON, M.; REYNOLDS, R.; IZURIETA, C. Comparison of json and xml data interchange formats: a case study. *Caine*, v. 9, p. 157–162, 2009.

OBIKWELU, C.; READ, J. C. The serious game constructivist framework for children's learning. *Procedia Computer Science*, Elsevier, v. 15, p. 32–37, 2012.

PEZOA, F.; REUTTER, J. L.; SUAREZ, F.; UGARTE, M.; VRGOČ, D. Foundations of json schema. In: *Proceedings of the 25th International Conference on World Wide Web*. [S.l.: s.n.], 2016. p. 263–273.

REDDY, M. *API Design for C++*. [S.l.]: Elsevier, 2011.

SALEN, K.; TEKINBAŞ, K. S.; ZIMMERMAN, E. *Rules of play: Game design fundamentals*. [S.l.]: MIT press, 2004.

SECRETAN, J.; WILD, F.; GUEST, W. Learning analytics in augmented reality: Blueprint for an ar/xapi framework. In: IEEE. *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*. [S.l.], 2019. p. 1–6.

SERRANO-LAGUNA, Á.; MARTÍNEZ-ORTIZ, I.; HAAG, J.; REGAN, D.; JOHNSON, A.; FERNÁNDEZ-MANJÓN, B. Applying standards to systematize learning analytics in serious games. *Computer Standards & Interfaces*, Elsevier, v. 50, p. 116–123, 2017.

VIDAKIS, N.; BARIANOS, A. K.; TRAMPAS, A. M.; PAPADAKIS, S.; KALOGIANNAKIS, M.; VASSILAKIS, K. Generating education in-game data: The case of an ancient theatre serious game. In: *CSEDU (1)*. [S.l.: s.n.], 2019. p. 36–43.

Apêndices

APÊNDICE A – Tutorial de instalação do LMS e plugins

Pode-se observar na Figura A.1 e na Figura A.2 o tutorial de instalação do LMS e dos plugins criados.

Pré-requisitos

Wordpress

É necessária uma instalação do wordpress, seja local ou em uma hospedagem. Para uso local, utilize soluções como wampserver ou xampp.

Você pode baixar a última versão do wordpress aqui.

Plugins necessários

Instale e ative os plugins abaixo no wordpress. Caso tenha dúvidas sobre como instalar um plugin, leia esse artigo. Você pode instalar os plugins externos diretamente pelo wordpress, com base nos nomes. Os links abaixo estão apenas como guia.

1. Learnpress - Necessário fazer o quick setup após ativar.

Plugins do projeto

Alguns plugins adicionais foram desenvolvidos para garantir todas as funcionalidades do projeto. Você pode encontrá-los nesse git. Cada pasta precisa estar em formato .zip, e você deve baixá-las em seu computador antes de continuar.

1. Image metabox
2. Learnpress associate quiz
3. Learnpress custom lessons
4. Video metabox aoc

Baixe todos os .zip e faça a instalação e ativação de todos os plugins via wordpress. Todos esses plugins podem ser utilizados ao se criar Lições (Painel de administrador > Learnpress> Lessons). Eles possibilitam a customização das aulas com imagens, questionário, ativar uma aula como lição externa e também associar vídeos.

Customização do Learnpress

Esse último passo é diferente dos anteriores, e garante que os estudantes não consigam completar uma aula externa pelo site, sem jogar o jogo.

Para isso, navegue até a pasta onde está a instalação do wordpress (seja localmente ou em uma hospedagem). Siga o caminho

```
wp-content > plugins > learnpress > templates
```

Selecione a pasta `template` e copie/baixe a pasta para seu computador.

Figura A.1 – Parte I do tutorial de instalação do LMS

Depois, configura o tema selecionado para a sua instalação no wordpress. Você pode conferir essa informação na aba de administrador do site seguindo o caminho `Aparência > Temas`. O tema selecionado no momento estará descrito como "Ativo".

Com o nome do tema em mente, navegue novamente nas pastas da instalação no caminho

`wp-content > themes > nome_do_tema_selecionado`

Cole ou faça upload da pasta que você baixou anteriormente para a pasta do tema. Ao finalizar, renomeie a pasta TEMPLATES para LEARNPRESS.

Por último, abra o arquivo `content.php` encontrado no caminho:

`wp-content > themes > nome_do_tema_selecionado > learnpress > content-lesson`.

Insira o seguinte código no final do arquivo:

```
<?php
    global $id_for_the_lesson;
    $id_for_the_lesson = $lesson->get_id();
?>
```

Prontinho! Seu projeto está pronto para uso :).

A documentação de cada plugin em específico pode ser encontrada navegando nas pastas acima.

A documentação da modificação feita no learnpress pode ser encontrada junto do plugin `Learnpress custom lesson`.

Figura A.2 – Parte II do tutorial de instalação do LMS

APÊNDICE B – Tutorial de uso do xAPI Helper

Pode-se observar na Figura B.1 e na Figura B.2 o tutorial de uso do Helper da xAPI.

xAPI Helpers Doc v1

Pré-requisitos

1. Criar ou ter um projeto em Unity (C#)
2. Baixar e importar a biblioteca TinCan.net que pode ser encontrada aqui.
3. Criar uma conta no ScormCloud (ou configurar LRS à sua escolha).

Como usar

1. Incluir os scripts abaixo no seu projeto
2. Criar classes com as Activities e Verbs relativos ao projeto
3. Incluir o script `ReusableStatementSender` em algum objeto na sua cena no Unity
4. Configurar as variáveis `remoteLRSURL`, `remoteKey` e `remoteSecret` com os dados do seu LRS no ScormCloud (ou similar).
5. Criar funcionalidades / classes que utilizem o `ReusableStatementSender` para configurar e enviar statements

Informações adicionais

Para entender mais sobre a xAPI:

- O que é a xAPI
- O que é um LRS
- Básico sobre Statements da xAPI
- Statement interativo com explicações e exemplo
- Clientes xAPI

xAPI Registry

Para salvar informações utilizando a xAPI é necessário que haja um registro do que significa cada uma das informações passadas (verbos, atividades, extensões, contextos, etc). Muitas dessas coisas são específicas do contexto da aplicação, mas muitas são comuns e poderiam ser reutilizadas. Para atender esse cenário, a xAPI possui um registro de atividades, verbos e extensões comuns que pode ser utilizado. Para mais informações, leia aqui.

Você pode acessar o xAPI Registry aqui. Ele contém tipos de verbos, atividades, extensões e etc.

Figura B.1 – Parte I do tutorial de uso do Helper do xAPI

Exemplo rápido de como criar dados

Ator / usuário

```
Agent actor = sender.getActor(email);
```

É importante que a variável `email` seja um e-mail válido, ou o *statement* não será salvo.

Verbo

```
TinCan.Verb verb = sender.getVerb(url, description);
```

Activity

```
TinCan.Activity activity = sender.getActivity(url, description);
```

Activity com extensão

```
TinCan.Activity activity = sender.getActivity(url, description, extensionString);
```

A extensão é uma informação adicional que será passada junto da atividade e deve ser armazenada em uma string respeitando o formato JSON.

Resultado

```
Result result = sender.getResult(isCompleted,  isCorrect,  score);
```

Figura B.2 – Parte II do tutorial de uso do Helper do xAPI