Uma API Web para suporte à análise de aprendizagem em jogos digitais educacionais para o contexto da disciplina de Química

Yan Gustavo Pegyn Silva

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Monte Carmelo - MG
2023

Yan Gustavo Pegyn Silva

Uma API Web para suporte à análise de aprendizagem em jogos digitais educacionais para o contexto da disciplina de Química

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Rafael Dias Araújo

Monte Carmelo - MG
2023

Dedico este trabalho à minha mãe, Joana Dark de Fátima e Silva, e ao meu pai, Wilson da Silva, que me deram todo o apoio que eu precisei para concluir esta etapa da minha vida acadêmica. Agradeço pelo amor, pela confiança e pelo incentivo que me fizeram persistir e superar os desafios. Sem eles, este trabalho não seria possível.

Agradecimentos

Quero agradecer a todos os professores que me ensinaram e inspiraram durante o meu curso. Com o conhecimento, o empenho e a tolerância deles, eu adquiri as competências necessárias para realizar este trabalho. Agradeço de forma especial ao meu orientador, que me guiou de forma atenta e me forneceu valiosas orientações e avaliações. Ele foi essencial para o desenvolvimento e a qualidade deste trabalho. Também agradeço aos meus amigos e colegas que me apoiaram e me deram ânimo nos momentos difíceis. Por fim, agradeço à minha família, que sempre me deu suporte, confiança e incentivo. Este trabalho é dedicado a todos vocês.

"Mais importante que as riquezas naturais são as riquezas artificiais da educação e tecnologia." (Neil deGrasse Tyson)

Resumo

Jogos digitais que ensinam a Tabela Periódica (TP) de Química podem auxiliar no processo de ensino-aprendizagem em vários níveis e contextos. No entanto, há um desafio de monitoramento contínuo decorrente da necessidade de entender a evolução e as possíveis dificuldades de cada estudante, além de identificar até que ponto o jogo proposto está sendo efetivo ou não nesse processo. Para isso, é importante coletar as interações realizadas dentro do jogo a fim de desenvolver mecanismos para análise de comportamento do estudante, e assim, permitir um melhor acompanhamento do processo de absorção do conhecimento. Além disso, tais mecanismos podem contribuir para a identificação de possíveis melhorias na experiência do usuário. Nesse sentido, este trabalho propõe uma interface de programação de aplicações (API) para integrar os dados de um jogo digital educativo para o ensino de química com plataformas de gerenciamento que apresentem indicadores de desempenho e aprendizagem dos estudantes. A API visa facilitar a análise e o monitoramento da eficácia e da evolução dos alunos que usam o jogo como ferramenta de aprendizado, bem como identificar possíveis melhorias na experiência do usuário. O trabalho também sugere a criação de uma API de lado cliente, que possa se comunicar com diferentes ferramentas de desenvolvimento e enviar dados mais específicos e detalhados sobre o jogo e a interação do jogador.

Palavras-chave: Game Learning Analytics, Análise de Aprendizagem em Jogos, Interface de Programação de Aplicação, Jogos Sérios, Jogos Digitais Educativos.

Lista de ilustrações

Lista de siglas

API Interface de Programação de Aplicação - *Application Programming Interface*

CSV Valores Separados Por Virgula - *Comma-separated values*

DER Diagrama Entidade-Relacionamento - *Entity-Relationship Diagram*

DTO Objeto de Transferência de Dados - *Data Transfer Object*

GLA Análise de aprendizagem de jogos - *Game Learning Analytics*

HTTP Protocolo de Transferência de Hipertexto - *Hypertext Transfer Protocol*

HATEOAS Hipermídia como o motor do estado de aplicação - *Hypermedia as the Engine of Application State*

JWT Token JSON da Web - *JSON Web Token*

JSON Notação de objeto JavaScript - *JavaScript Object Notation*

KPI Indicadores Chave de Performance - *Key performance indicators*

MVC Controlador de Visualização de Modelos - *Model-View-Controller*

PoC Prova de Conceito - *Proof of Concept*

POJO Objeto Java Antigo e Simples - *Plain Old Java Object*

POPO Objeto PHP Antigo e Simples - *Plain Old PHP Object*

REST Transferência de estado representacional - *Representational State Transfer*

SGBD Sistema Gerenciador de Banco de Dados - *Database Management System*

SQL Linguagem de consulta estruturada - *Structured Query Language*

TCC Trabalho de Conclusão de Curso - *Undergraduate Final Project*

TP Tabela Periódica - *Periodic table*

URL Localizador Uniforme de Recursos - *Uniform Resource Locator*

Sumário

Capítulo 1

Introdução

Com a popularização dos meios de acesso digitais, tais como *smartphones* e computadores portáteis, os jogos digitais têm acompanhado essa evolução, e passaram a apresentar aspectos que podem favorecer a integração do estudante no mundo digital, assim favorecendo a otimização dos recursos disponíveis e aumentando a variedade das formas de acesso ao conhecimento, de forma dinâmica, moderna e prazerosa (LIMA; MOITA, 2011).

Diante desse cenário, diferentes metodologias de ensino podem ser exploradas para aproveitar o potencial dos jogos digitais como formas de aprendizagem. Uma delas é o uso de jogos sérios, que são jogos cujo a função primária é para uso em um domínio específico não somente a de entretenimento, como os jogos digitais educacionais e os jogos de treinamento, mas que podem e devem ser divertidos e estimulantes (CASSIMIRO, 2020). Jogos sérios podem ser utilizados para apoiar o ensino de diversos conteúdos e habilidades, em diferentes níveis e contextos (BALDISSERA, 2021b). Para avaliar o impacto dos jogos sérios na aprendizagem dos estudantes, é necessário aplicar técnicas de Análise de aprendizagem de jogos (GLA), como a Análise de Desempenho, que permite medir o progresso e a proficiência dos estudantes em relação aos objetivos educacionais do jogo (DINIZ, 2020).

Dessa forma, a aplicação de técnicas para extração de conhecimento sobre as interações de um jogador se torna altamente relevante, visto que os jogos já analisam as ações efetuadas pelo usuário, assim possibilitando efetuar reações que impactam na jogabilidade e elaboração de placares de pontuação. Contudo, no meio educacional, essas métricas mesmo sendo útil para exibir a proficiência do usuário no jogo, não extraem um detalhamento suficiente sobre todos os tópicos de *Game Design*, sendo eles dados sobre estética, mecânica, historia e tecnologia, a fim de exibir informações que permitem identificar pontos de melhorias (FREIRE et al., 2016a; SCHELL, 2008).

1.1 Problema

Existem muitas ferramentas que dão suporte ao processo de ensino-aprendizagem em vários níveis e contextos. No entanto, há um desafio de monitoramento contínuo decorrente da necessidade de entender a evolução e as possíveis dificuldades de cada estudante, além de identificar até que ponto a ferramenta proposta está sendo efetiva ou não nesse processo.

Especificamente no contexto do uso de jogos digitais para o ensino da Tabela Periódica (TP) de Química, é importante coletar as interações realizadas dentro do jogo a fim de desenvolver mecanismos para análise de comportamento do estudante, e assim, permitir um melhor acompanhamento do processo de aprendizagem e identificar as lacunas de aprendizado em determinados elementos químicos. Além disso, tais mecanismos podem contribuir para a identificação de possíveis melhorias na experiência do usuário.

1.2 Justificativa

A aplicação de ferramentas digitais para auxílio da retenção de conhecimentos muitas vezes não tem o impacto desejado. Os jogos digitais educativos, categorizados como Jogos Sérios, são um exemplo dessas ferramentas, onde em sua maioria não tem seu potencial máximo extraído por conta da ausência de funcionalidades integradas que permitem o acompanhamento da aprendizagem. Tendo esse problema em vista, uma possível solução seria fornecer um ecossistema que permita a coleta de informações sobre a interação dentro do jogo, com isso favorecendo análises que possibilitem que o jogo tenha um *feedback* aos usuários e desenvolvedores.

Os jogos digitais podem ser usados como formas de aprendizagem, pois oferecem diversão e desafio aos estudantes. No entanto, para motivar o uso de jogos digitais no meio educacional, é necessário desenvolver uma maior compreensão do impacto causado por eles no processo de aprendizagem.

1.3 Questão de Pesquisa

Este trabalho se pautou em estudar e responder a seguinte questão de pesquisa: *Como desenvolver mecanismos para análise de desempenho de estudantes que utilizam um jogo digital sério no aprendizado, e como esses mecanismos podem contribuir para o acompanhamento do processo de aprendizagem e a melhoria da experiência do usuário?*

1.4 Objetivos

O **objetivo geral** consistiu em desenvolver uma interface de programação de aplicações (API) para permitir integrar os dados de interações em jogos digitais educacionais com uma plataforma Web de gerenciamento com indicadores de desempenho e aprendizagem dos estudantes. Para isso, os seguintes **objetivos específicos** foram delineados:

- ❏ Efetuar um estudo da literatura para identificação das técnicas utilizadas no contexto de Análise de aprendizagem de jogos (GLA);
- ❏ Realizar um levantamento de requisitos, sejam eles funcionais ou não, a fim de realizar a escolha de tecnologias e métodos;
- ❏ Estudar as tecnologias e métodos que permitem implementar os requisitos levantados;
- ❏ Estudar e planejar formas de captura de dados;
- ❏ Projetar e implementar uma API Web para coleta das interações e análise de desempenho;
- ❏ Trabalhar em conjunto com outro estudante de Trabalho de Conclusão de Curso (TCC) para integração da API com uma aplicação *front-end*.

1.5 Organização da Monografia

O Capítulo 1 apresenta um breve contexto sobre o tema, a problemática, a questão de pesquisa e os objetivos deste trabalho.

O Capítulo 2 aborda o embasamento teórico e conceitual do presente trabalho, mostrando o referencial que sustenta todo o conteúdo discriminado nos capítulos seguintes, além de exemplificar trabalhos relacionados já existentes na área.

O Capítulo 3 explica como o trabalho foi desenvolvido e quais foram as ferramentas utilizadas para isso. Além disso, apresenta os detalhes da proposta criada para solucionar o problema estudado.

O Capítulo 4 apresenta os resultados preliminares obtidos com a implementação da proposta, que incluem: a funcionalidade e a acessibilidade da proposta, garantidas pela validação e documentação dos *endpoints*; a qualidade e a confiabilidade do código, verificadas pelos testes unitários; o nível de conhecimento, habilidade ou competência adquirido pelos jogadores após jogarem o jogo, avaliado pela análise de desempenho, e comparado com os objetivos de aprendizagem do jogo; e os pontos de melhoria para o

design e a implementação do jogo, identificados pela análise de *logs* que correlacionam as interações dos estudantes com o ambiente do jogo.

O Capítulo 5 retoma os objetivos gerais do trabalho, resume as principais contribuições do estudo para o campo de conhecimento, sugere possíveis aplicações práticas dos resultados e indica possíveis linhas de pesquisa futuras sobre o tema.

Capítulo 2

Fundamentação Teórica

Neste capítulo, serão apresentados os conceitos e as teorias que embasam o trabalho, bem como os trabalhos relacionados na literatura. O objetivo é apresentar o referencial teórico que sustenta o trabalho e a solução proposta, bem como comparar e diferenciar o trabalho dos demais existentes na área.

2.1 Análise de aprendizagem de jogos (GLA)

Durante o processo de criação de um jogo educativo, as metas educacionais de análise de aprendizagem (*Learning Analytics*) e as tecnologias de análise de jogos (*Game Analytics*) devem ser combinadas. E quando essa combinação ocorre, ela pode ser chamada de GLA, a qual, por sua vez, contribui para a uma melhor aplicação de jogos educativos.

Monitorar o que se ocorre ao longo do percurso do jogo, enquanto o utilizador está jogando, é uma tarefa essencial, visto que se torna possível relacionar o jogo com o processo de aprendizagem, e assim, se possibilita comprovar a eficácia do jogo por meio de provas e não mais apenas por uma abordagem teórica (FREIRE et al., 2016a).

Os princípios básicos do GLA de acordo com (FREIRE et al., 2016a) são:

1. Instrumentação (*Instrumentation*): Instrumentar o jogo para que o mesmo exporte informações sobre as interações do jogador periodicamente;

2. Coleta e armazenamento (*Collection and Storage*): Desenvolver um sistema servidor capaz de receber, classificar e armazenar as informações providenciadas pela instrumentação;

3. Análises em tempo real (*Collection and Storage*): Mostrar com o menor atraso possível informações a respeito da atividade de um jogador, para que seja possível o acompanhamento por um instrutor;

4. Análise em lotes (*Aggregated (batched) analysis*): Criação de agregações baseadas em análises complexas com a finalidade de comparar sessões de jogos;

5. Indicador-chave de desempenho (*Indicadores Chave de Performance (KPI)*): Assimilar fatores que levaram à melhores pontuações (notas), conclusão do objetivo ou uma melhora na eficácia educacional;

6. Painel de análise (*Analytics Dashboard*): Permitir o acompanhamento dos KPIs de maneira visualizável e simplificada.

Durante o processo de coleta de informações para aplicar técnicas de GLA em jogos sérios, algumas informações agregam um certo valor para compreender a interação dos jogadores com os jogos e avaliar a sua aprendizagem. Essas informações podem ser classificadas em diferentes categorias, tais como:

- ❏ O número de jogadores e suas características demográficas (idade, gênero, localização etc.);
- ❏ O tempo de jogo e a frequência de sessões;
- ❏ Os níveis jogados e os resultados obtidos (pontuação, tempo, vidas restantes etc.);
- ❏ Os itens coletados e os inimigos derrotados;
- ❏ As ações realizadas pelo jogador (pulos, movimentos, ataques etc.);
- ❏ As preferências e o feedback do jogador (avaliações, comentários, sugestões etc.);
- ❏ Informações relacionadas ao ambiente (pontos de informação, obstáculos, pontos de decisão etc.).

Essas informações podem fornecer insights valiosos sobre o comportamento, o desempenho e a aprendizagem dos jogadores, bem como sobre o design e a usabilidade dos jogos. A seguir, descrevemos as principais técnicas de GLA e os seus benefícios e desafios para o desenvolvimento e a pesquisa de jogos sérios.

O GLA pode usar diferentes técnicas de análise de dados para extrair conhecimento dos dados do jogo. Algumas das principais técnicas são:

- ❏ Análise de padrões de comportamento: consiste em identificar e classificar os diferentes tipos de comportamento dos jogadores, como exploratório, estratégico, colaborativo etc., e relacioná-los com os objetivos de aprendizagem do jogo (ALONSO-FERNÁNDEZ et al., 2019);
- ❏ Análise de trajetórias: consiste em analisar os caminhos percorridos pelos jogadores dentro do jogo, como as escolhas feitas, os desafios superados, os erros cometidos

etc., e verificar como eles afetam a experiência e o resultado do jogo (KANG; LIU; QU, 2017);

- ❏ Análise de engajamento: consiste em medir o nível de interesse, motivação e satisfação dos jogadores com o jogo, usando indicadores como tempo de jogo, frequência de uso, progresso alcançado, *feedback* recebido etc., e verificar como eles influenciam na aprendizagem (FREIRE et al., 2016b);

- ❏ Análise de desempenho: consiste em avaliar o nível de conhecimento, habilidade ou competência adquirido pelos jogadores após jogarem o jogo, usando métodos como testes pré e pós-jogo, questionários, entrevistas etc., e comparar os resultados com os objetivos de aprendizagem do jogo (FREIRE et al., 2016b).

2.2 Interface de Programação de Aplicação (API)

Traduzido como Interface de Programação de Aplicação, API é um conjunto de métodos e padrões estabelecidos e disponibilizados por um software para integração ou extensão de outros sistemas de software (BIEHL, 2016), para que aplicativos que não pretendem ter contato com a implementação desse sistema tenha acesso a essas funcionalidades por meio de um ou mais serviços. Já no contexto de desenvolvimento Web, API é um conjunto padronizado de mensagens de requisições e respostas de Protocolo de Transferência de Hipertexto (HTTP) (BIEHL, 2016).

As APIs que estão em conformidade com o estilo arquitetural Transferência de estado representacional (REST), denominados API REST, têm o potencial de implementar integração de software de maneira simplificada. Conforme (FIELDING, 2000), o REST não é um padrão imposto, mas sim é um conjunto de padrões e recomendações que auxiliam na padronização da interface.

A padronização proporcionada pelo REST é feita por meio de restrições, definidas em (FIELDING, 2000; PLANSKY, 2014) sendo elas:

1. Client-Server: Implica que o serviço cliente deve ser desenvolvido separadamente do serviço servidor, possibilitando assim a evolução concorrente dos dois serviços, de maneira que suas responsabilidades sejam voltadas a seu objetivo.

2. Stateless: O servidor não deve guardar informações sobre requisições anteriores, sendo assim, a cada nova requisição o cliente deverá conter todas as informações para que o servidor consiga processá-la de forma independente.

3. Cacheable: O servidor deve rotular se os recursos de uma requisição são ou não armazenáveis em cache, para que não seja gasto processamento desnecessário.

4. Uniform Interface: Implica na utilização de quatro restrições de interface: identificação de recursos; manipulação de recursos por meio de representações; mensagens auto descritivas; e, Hipermídia como o motor do estado de aplicação (HATEOAS).

5. Layered System: A aplicação deve ser composta por camadas, garantindo uma maior flexibilidade, já que o cliente não se comunica diretamente com a aplicação.

6. Code-On-Demand: Esta por sua vez, está mais para uma recomendação, visto que sua utilização não é algo obrigatório no padrão da arquitetura REST. Define que seja disponibilizada alguma forma de execução de código no lado cliente, a fim de possibilitar a extensão da lógica do servidor a um cliente em específico.

2.3 Conceitos e padrões de projeto para o desenvolvimento web

Neste item da fundamentação teórica, serão apresentados alguns conceitos e padrões de projeto que são relevantes para o desenvolvimento de sistemas e aplicações web, usando a linguagem PHP e o *Framework* Slim. Os conceitos e padrões de projeto aqui citados são soluções abstratas e reutilizáveis para problemas comuns que ocorrem no desenvolvimento de software, visando facilitar e otimizar o trabalho dos programadores, bem como garantir a qualidade e a confiabilidade dos produtos finais. Esses conceitos e padrões de projeto serão utilizados neste trabalho para desenvolver uma Interface de Programação de Aplicação (API) para gerenciar os dados obtidos de jogos sérios, seguindo as boas práticas e as recomendações da área. A seguir, serão explicados os principais conceitos e padrões de projeto utilizados no presente trabalho.

2.3.1 Objeto PHP Antigo e Simples (POPO)

O termo POPO é utilizado para representar o conceito de Objeto Java Antigo e Simples (POJO) para objetos escritos na linguagem PHP ao invés de Python.

Já o termo POJO foi cunhado por Martin Fowler, Rebecca Parsons e Josh MacKenzie em 2000 (FOWLER, 2003), para enfatizar as vantagens de codificar a lógica de negócios em objetos Java regulares. A ideia dos termos se mantém a mesma, visto que, onde um Objeto Java Antigo e Simples (POJO) é um objeto simples que não está vinculado a nenhuma restrição especial, como implementar interfaces ou estender classes específicas.

Sendo assim, uma grande vantagem da utilização de *Plain Old Objects* é que eles aumentam a legibilidade e a reutilização do código do projeto (Codez Up, 2020).

2.3.2 Objeto de Transferência de Dados (DTO)

Um objeto que transporta dados entre processos é chamado de Objeto de Transferência de Dados (DTO). Esse padrão, proposto por Martin Fowler em seu livro (FOWLER, 2012), visa diminuir o número de chamadas de métodos necessárias para a comunicação. Assim, ele melhora o desempenho da rede nessas operações remotas. Além disso, o padrão encapsula a lógica da serialização (processo que converte a estrutura e os dados do objeto em um formato específico para armazenamento e transferência). Isso facilita as alterações na forma de serialização, se necessário. Ele também isola os modelos de domínio da camada de apresentação, dando mais flexibilidade para ambos evoluírem separadamente (WILLIANS, 2022).

2.3.3 Controlador de Visualização de Modelos (MVC)

Baseado na ideia de separar a responsabilidade de métodos ligados à apresentação de dados, e métodos que interagem diretamente com os dados, o padrão de arquitetura de software Controlador de Visualização de Modelos (MVC) foi proposto. O padrão incentiva o desenvolvimento de sistemas modulares, facilitando a manutenção do mesmo (HOPKINS, 2013).

De acordo com (SOUZA, 2011), o padrão MVC pode ser dividido em três camadas:

1. *Model*: é a camada que lida com os dados da aplicação, realizando as operações de leitura e escrita no banco de dados e as validações necessárias. Ela representa o domínio do problema e as regras de negócio.

2. *View*: é a camada que mostra os dados ao usuário, usando elementos visuais como telas, formulários, botões, etc. Ela representa a interface de usuário e a forma como os dados são apresentados.

3. *Controller*: é a camada que faz a ligação entre o *model* e a *view*, recebendo as solicitações do usuário e decidindo qual *model* usar e qual *view* mostrar. Ela representa a lógica de controle e o fluxo da aplicação.

Essa separação de camadas visa reduzir o acoplamento entre os componentes do sistema, aumentar a coesão das classes, facilitar a manutenção e o reuso do código e promover uma melhor organização do projeto.

Durante as etapas de desenvolvimento do presente trabalho, o padrão MVC foi seguido parcialmente, tendo em vista que classes relacionadas a camada *View* não são contidas dentro do projeto em si, sendo delegadas a um outro sistema.

2.3.4 Notação de objeto JavaScript (JSON)

Notação de objeto JavaScript (JSON) é um formato de arquivo e de troca de dados que usa texto legível para armazenar e transmitir objetos de dados compostos por pares de atributo-valor e *arrays* (ou outros valores serializáveis). JSON é um formato independente de linguagem, derivado dos literais de objeto do *JavaScript*, mas que pode ser gerado e analisado por diversas linguagens de programação. JSON é um padrão aberto, especificado pela ECMA-404 (Ecma International, 2017), RFC 8259 (BRAY, 2017) e ISO/IEC 21778:2017 (International Organization for Standardization, 2017). JSON foi proposto por Douglas Crockford em 2001, como uma alternativa aos protocolos de comunicação em tempo real entre navegadores e servidores (CROCKFORD, 2006).

2.3.5 Token JSON da Web (JWT)

De acordo com (JONES; BRADLEY; SAKIMURA, 2015), JWT é técnica definida na RFC 7519 que define uma forma compacta e autocontida de transmitir informações entre partes por meio de um objeto JSON. Essas informações podem ser verificadas por meio de uma assinatura digital, tornando-as confiáveis.

Ele consiste em três partes separadas por pontos (.): o cabeçalho, a carga útil e a assinatura. O cabeçalho contém o tipo do *token* e o algoritmo de assinatura usados. A carga útil contém as afirmações, que são declarações sobre uma entidade (tipicamente, dados do usuário) e dados adicionais. A assinatura é calculada usando o cabeçalho e a carga útil, juntamente com um segredo ou uma chave privada, e garante a integridade e a autenticidade das informações (AUTH0, 2021).

2.4 Jogos Sérios: Conceito e Aplicações

Jogos sérios são aqueles que têm um objetivo primário diferente de puramente entreter, ou seja, que visam transmitir algum tipo de conhecimento, habilidade, valor ou atitude aos jogadores. Esses jogos podem ser utilizados em diversos contextos, como educação, saúde, treinamento, propaganda, política, arte, entre outros (SAVI; ULBRICHT, 2008).

Os jogos sérios podem contribuir para o processo de ensino-aprendizagem de diversas formas, eles podem aumentar o engajamento e a motivação dos estudantes, proporcionar uma experiência imersiva e interativa, simular situações reais ou fictícias que estimulem a criatividade e o raciocínio e facilitar a avaliação e o acompanhamento do desempenho dos estudantes (BALDISSERA, 2021a; DARIN, 2020).

Dentre os principais segmentos dos jogos sérios, estão contidos os jogos digitais educacionais, os quais são destinados a promover o aprendizado e desenvolvimento de habilidades na educação básica e no ensino superior (BALDISSERA, 2021a).

No entanto, para que os jogos sérios sejam eficazes, eles devem ser analisados, projetados, desenvolvidos e avaliados de forma sistemática e holística, considerando seu ciclo de vida de desenvolvimento e a abrangência dos requisitos de suas partes e inter-relações. Essas fases envolvem diversos desafios e oportunidades, que estão distribuídos em vários artigos e livros disponíveis na literatura.

Uma revisão da literatura sobre o desenvolvimento de jogos sérios e áreas afins foi realizada por (ROCHA; BITTENCOURT; ISOTANI, 2015), que identificaram sete principais desafios:

- ❏ Balancear e integrar os requisitos do jogo, da simulação e do conteúdo instrucional;
- ❏ Definir o público-alvo, as competências a serem desenvolvidas e os objetivos de aprendizagem;
- ❏ Escolher as estratégias pedagógicas adequadas ao domínio, ao conteúdo e aos objetivos;
- ❏ Projetar cenários realistas, motivadores e desafiadores;
- ❏ Implementar o jogo usando ferramentas adequadas e acessíveis;
- ❏ Avaliar o jogo em termos de usabilidade, jogabilidade, aprendizagem e impacto;
- ❏ Disseminar o jogo para os potenciais usuários e pessoas interessadas.

Para cada um desses desafios, os autores apresentam uma discussão sobre as dificuldades, as lacunas e as oportunidades de pesquisa na área. Eles também sugerem algumas recomendações práticas para os desenvolvedores de jogos sérios.

2.5 Trabalhos Relacionados

A área de GLA tem recebido muita atenção nos últimos anos, com novos trabalhos que propõem modelos, métodos e ferramentas para avaliar o aprendizado dos jogadores em jogos sérios. Nesta seção, será apresentado uma revisão bibliográfica dos principais trabalhos relacionados ao nosso tema, onde seus objetivos giram em torno de explorar diferentes abordagens e ferramentas que podem auxiliar os desenvolvedores e educadores a coletar e analisar dados sobre o uso e o impacto dos jogos sérios.

Uma dessas ferramentas é o GLBoard, o qual é um sistema elaborado com foco em auxiliar na captura e análise de dados em jogos educacionais, proposto por (SILVA et al., 2022). Esse sistema é um exemplo da aplicação do conceito de Análise de aprendizagem de jogos (GLA) tal como o presente trabalho, ou seja, aplica técnicas de análise de dados para entender e melhorar os processos de ensino e aprendizagem em jogos educacionais. O GLBoard permite que os desenvolvedores e educadores definam

indicadores de desempenho e comportamento dos jogadores, visualizem os dados coletados em gráficos e tabelas, e exportem os dados para outras ferramentas de análise.

Além de ferramentas, também há estudos que se baseiam a explorar diferentes metodologias de aplicação de técnicas de GLA, tal como a pesquisa (MELO et al., 2020), onde os autores apresentam uma estratégia para avaliação de *Level Design* em jogos educacionais. *Level Design* é um aspecto importante do desenvolvimento de jogos que envolve a criação de níveis ou áreas do mundo do jogo pelos quais o jogador deve navegar. É um processo que envolve arte, design e programação e é crítico para o processo de design do jogo (KARLSSON; BRUSK; ENGSTRöM, 2022). A estratégia proposta consiste em definir métricas de GLA relacionadas ao *Level Design*, visando métricas como tempo médio, taxa de sucesso e número de tentativas, por cada nível do jogo. Tais métricas são calculadas a partir de registros coletados durante as partidas e podem indicar se os níveis estão equilibrados em termos de dificuldade, interesse e motivação.

Aumentando o escopo de abordagens, podemos identificar trabalhos que utilizam de metodologias diferentes do GLA para avaliar a aprendizagem dos estudantes em jogos sérios. Uma delas é o Pro-AvaliaJS, um protocolo de planejamento e execução das avaliações de reação e aprendizagem. O protocolo consiste em definir etapas e atividades para orientar a equipe ao longo de todo o processo de avaliação e testes de jogos sérios (OLIVEIRA et al., 2022). O protocolo foi proposto baseado no modelo conceitual AvaliaJS, que é um modelo para planejar a avaliação do desempenho humano em jogos sérios (OLIVEIRA; ROCHA, 2021).

Os trabalhos anteriormente citados, mostram algumas formas possíveis de avaliar a eficácia da utilização de jogos digitais para o ensino. Possibilitando assim a aplicação dessas técnicas com jogos sérios que tenham sua ênfase no ensino de Química. No entanto, ainda há lacunas na literatura sobre o desenvolvimento de jogos que sejam capazes de integrar diferentes conceitos químicos em uma narrativa lúdica e envolvente, assim como existe uma ausência de conteúdos que explorem quais dados são ou não necessários para desenvolver um ecossistema de software para apoiar no ensino de química. Essas lacunas trouxeram novos desafios ao desenvolvimento da presente pesquisa.

Capítulo 3

Desenvolvimento

Neste capítulo, serão descritos os procedimentos e as ferramentas empregadas para o desenvolvimento do trabalho e os detalhes da proposta criada.

3.1 Método

Este trabalho utilizou métodos experimentais e exploratórios para responder à pergunta de pesquisa. Um protótipo funcional da API foi elaborado e o comportamento das interfaces foi avaliado através de testes automatizados. A metodologia ágil de desenvolvimento foi utilizada (PRESSMAN; MAXIM, 2016), com um planejamento incremental e iterativo. A execução aconteceu em pequenas etapas, chamadas iterações, cada uma resultando em um software potencialmente funcional.

Para o processo de avaliação, foram utilizados a documentação e validação dos *endpoints*, testes unitários e um teste preliminar com estudantes. Esses elementos foram escolhidos com base em um conjunto de parâmetros que incluiu a precisão, a eficiência e a usabilidade da API, dados coletados das interações dos estudantes, as requisições e as respostas da API e trabalhos comparáveis da área que abordaram problemas similares ou utilizaram tecnologias relacionadas.

3.2 Prova de conceito

Durante o desenvolvimento deste trabalho, foi elaborado uma Prova de Conceito (PoC) da API, com o objetivo de experimentar o funcionamento da teoria aqui descrita de maneira prática, a qual após seu desenvolvimento, foi apresentada a pessoas que estavam a realizar pesquisas relacionadas e foi disponibilizado uma versão hospedada da API aos que se propuseram a utiliza-la em seus projetos.

No início do desenvolvimento da PoC, foram pesquisadas as tecnologias disponíveis no mercado que atenderiam aos requisitos do sistema. Após análise, foi decidido utilizar as seguintes tecnologias:

1. PHP (*Hypertext Preprocessor*): PHP é uma linguagem de programação de *script open-source* amplamente utilizada para desenvolvimento web (The PHP Group, 2018). De acordo com a pesquisa realizada pela W3Techs, PHP é utilizado por cerca de 79% dos sites pesquisados como sua linguagem de programação do lado do servidor (W3Techs, 2021). É uma das linguagens mais populares para a criação de sites dinâmicos e aplicações web. Para implementar uma prova de conceito de uma API que visa auxiliar no manuseio de informações de química correlacionadas a um jogo, PHP é uma escolha ideal devido à sua facilidade de integração com outras tecnologias web, como bancos de dados e HTML (TATROE; LERDORF, 2006). Além disso, PHP possui uma grande comunidade ativa que oferece suporte e recursos para desenvolvedores, facilitando a criação de soluções robustas e escaláveis.

2. Slim Framework: é um *micro-framework open-source* para PHP, voltado para o desenvolvimento de aplicações web simples e rápidas, desenvolvido por Josh Lockhart (LOCKHART, 2013). Ele foi projetado para fornecer um conjunto básico de recursos, incluindo rotas, middlewares e manipulação de requisições e respostas.

3. PSR-7 é um padrão de desenvolvimento de software proposto pela PHP Framework Interoperability Group (PHP-FIG) (PHP Framework Interop Group, 2014), que define interfaces comuns para trabalhar com mensagens HTTP. Ele especifica as interfaces para requisições e respostas HTTP, bem como para mensagens de corpo, permitindo que os desenvolvedores criem aplicações PHP de forma consistente e interoperável.

4. PHPUnit: O PHPUnit é uma ferramenta de teste unitário para aplicações PHP. De acordo com (REHKOPF, 2021), A utilização da automação de testes é um assunto de grande importância quando se trata de garantir a qualidade do software. Portanto, é recomendável considerar a aplicação desta prática em todos os projetos de software.

A API Web foi elaborada conforme os conceitos e padrões de projeto abordados na seção 2.3. No momento da redação deste trabalho, a API conta com 44 *endpoints* que são responsáveis por lidar com dados de usuários, questionários, elementos da Tabela Periódica e registro de toda a interação realizada com o jogo. As estruturas relacionadas com a Tabela Periódica e os registros de interação foram modeladas de forma não relacional, utilizando o formato JSON, enquanto a estrutura gerencial da plataforma foi modelada de forma relacional, utilizando o Sistema Gerenciador de Banco de Dados (SGBD) MySQL.

A API tem como objetivo dois públicos alvos, consumidores, como painéis de gerenciamento, e provedores de informação como jogos. O diagrama da Figura 1 ilustra essa arquitetura, onde, o bloco “Web Server” corresponde aos sistemas consumidores, o

"Game Server (Web Backend)" aos provedores e o Bloco "Chemical API (Backend)" à API.

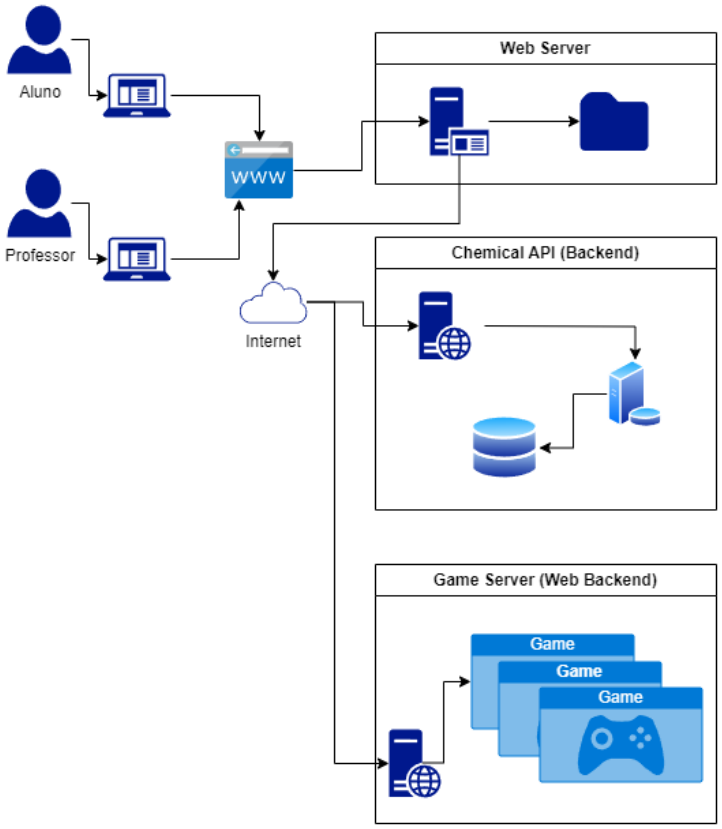

Figura 1 – Visão geral da arquitetura proposta.

Vale ressaltar que o diagrama ilustra os principais cenários de utilização, ou seja, estudante e professor, desconsiderando participantes indiretos, como administradores da API ou sistemas periféricos aos consumidores ou aos produtores.

Durante o processo de experimentos, as funcionalidades veiculadas a público de consumidores de informações, foram disponibilizadas a um outro estudante que estava desenvolvendo uma ferramenta Web com o intuito de auxiliar o acompanhamento do ensino digital, em seu Trabalho de Conclusão de Curso (TCC), com o intuito de validar se a API cumpre com seu objetivo. Já, relacionado às funcionalidades de produtores, foi utilizado de um jogo sério desenvolvido por um terceiro estudante de TCC que tinha como alvo o ensino de elementos químicos categorizados como Gases Nobres.

Para o cenário de consumidor, a API disponibiliza funções distintas para três perfis de usuário, sendo eles: estudantes, professores e administradores. Já para o cenário de provedor, o único perfil disponível atualmente é o de estudante.

As funcionalidades já providenciadas pela API nesse momento de prova de conceito foram agrupadas em:

1. **Funcionalidades globais**: Onde as funcionalidades aqui disponibilizadas não necessitam de um usuário autenticado para exibi-las. Como listagem de escolas que o software está disponível, listagem das fases (de jogos distintos ou do mesmo) que estão provendo informações à API, além de funcionalidades de autenticação;

2. **Funcionalidades de Administrador**: Conjunto de funcionalidades restritas à usuários que tenham acesso de administrador do ecossistema, podendo realizar ações como gerenciar o cadastro de Escolas, Fases e Professores;

3. **Funcionalidades de Professor**: A API disponibiliza todas funcionalidades necessárias para que um professor consiga gerenciar atividades de suas turmas no ecossistema, assim como acompanhar estatísticas em diferentes níveis de detalhes;

4. **Funcionalidades de Estudantes**: Se é providenciado funcionalidades para viabilizar o uso do ecossistema por parte do estudante, como, acesso a turmas, listagem das fases disponíveis para jogar, e estatísticas básicas sobre o seu desempenho;

5. **Listagem global de Elementos Químicos**: Disponibiliza a busca de diferentes dados relacionados aos Elementos Químicos.

A avaliação sobre a PoC foi efetuada de maneira informal ao longo de reuniões, de forma que sugestões e correções fossem incluídas na lista de melhorias do projeto. Após a implementação, ao questionar os desenvolvedores do projeto sobre as vantagens de ter utilizado a API em suas soluções, foram obtidos os resultados abaixo.

Do ponto de vista do consumidor de informação os principais resultados obtidos da utilização da API foram:

1. Remoção da necessidade de se criar uma arquitetura de *backend* para reter as informações;

2. Praticidade trazendo a listagem dos elementos químicos e suas propriedades;

3. O tratamento prévio dos dados simplifica o processo de exposição das informações;

4. Fácil integração.

Do ponto de vista de provedor, os principais resultados obtidos foram:

1. Permitiu a extração de dados do questionário de forma simplificada, dispensando o pre-processamento dos dados para embaralhar ou identificar a resposta correta, uma vez que essas informações já são fornecidas pela API;

2. Possibilitou a geração de questões personalizadas para cada grupo de usuário, evitando a criação de um *backend* próprio do jogo para armazenar as perguntas.

Portanto, ao desenvolver o ecossistema de software composto pela API, pelos jogos digitais educacionais e pelos demais componentes, é possível obter as seguintes vantagens:

1. Reduzir o tempo e o custo de desenvolvimento dos jogos digitais educacionais, aproveitando os recursos e as funcionalidades da API;
2. Aumentar a qualidade e a confiabilidade dos jogos digitais educacionais, garantindo que eles sigam os padrões e as boas práticas de desenvolvimento de software;
3. Estimular a colaboração e a inovação entre os desenvolvedores de jogos digitais educacionais, criando um ecossistema de software que favoreça o compartilhamento de conhecimentos e experiências.

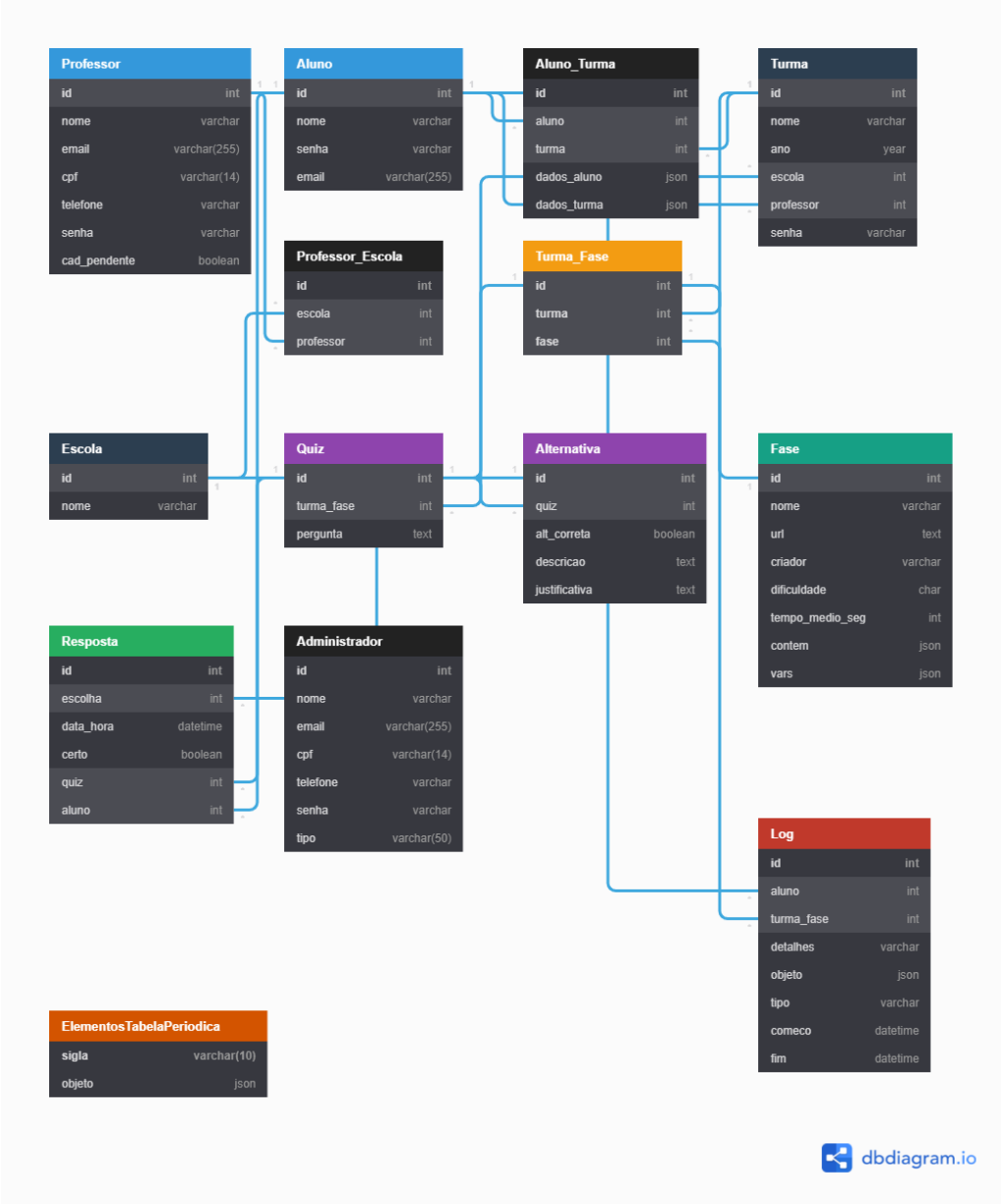

Figura 2 – Modelagem da estrutura do banco de dados.

Para modelagem do banco foi utilizado o software *dbdiagram.io*[1], que por meio de uma sintaxe simplificada, é possível criar tanto o Diagrama Entidade-Relacionamento (DER) quanto o código Linguagem de consulta estruturada (SQL) a ser utilizado no sistema. A modelagem do utilizada na PoC está disponível na Figura 2.

Para a elaboração da documentação dos *Endpoints*, foi utilizado o *Software Insomnia*[2], que é um aplicativo de código aberto e atua como cliente de API REST, onde se é possível realizar a documentação, a depuração e os testes, vide o exemplo da Figura 3. Além disso, o *Insomnia*[2] conta com funcionalidades que auxiliam na autenticação, geração de código e gerenciamento de *Cookies* e variáveis de ambiente.

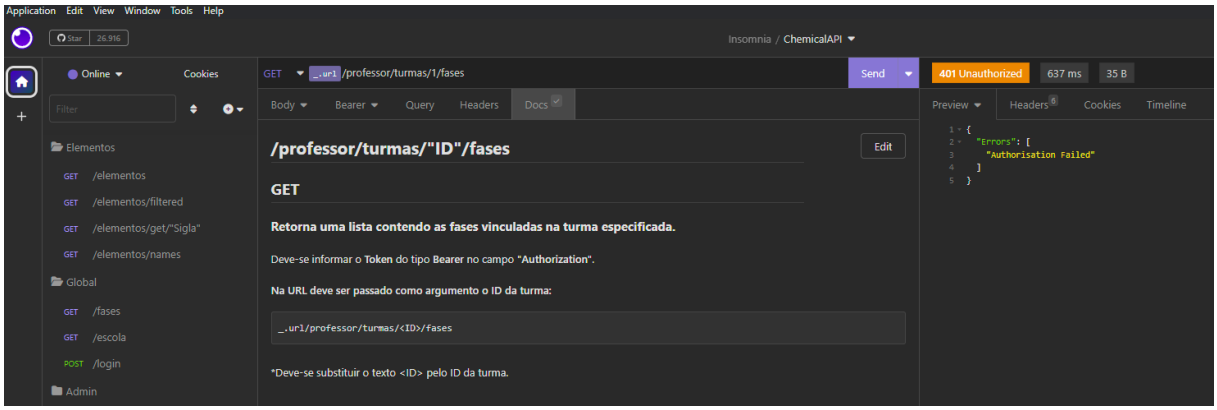

Figura 3 – Visão da documentação no Insomnia

A estrutura de nomes dos *endpoints* segue o formato exemplificado na Figura 4, onde o primeiro subnível da rota da *URL* representa o 'Tipo do Usuário' que é capaz de executar aquela ação, caso não contenha um usuário, a rota deve ser de uso compartilhado entre diferentes tipos de usuário, seguido pelo 'Escopo da ação' que determina onde a ação será executada, como visto no exemplo o escopo 'turmas'. Sendo assim, podemos dizer que um professor no executa uma ação no escopo de uma turma, esse escopo pode ser genérico ou especificado caso necessário utilizando um identificador (ID), assim como no exemplo que é determinada a turma a ser usada, ao especificar o escopo é possível acessar recursos pertencentes aquele escopo, como no exemplo as fases de uma turma.

[1] https://dbdiagram.io/
[2] https://insomnia.rest/

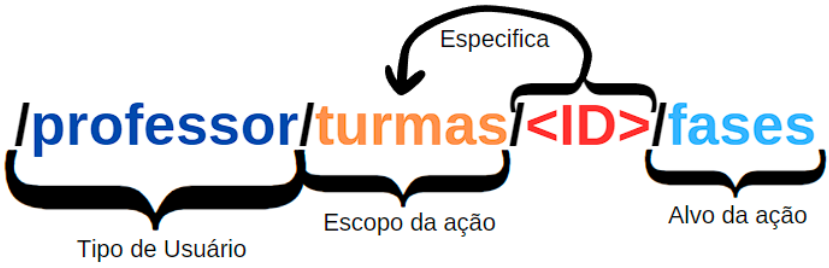

Figura 4 – Estrutura de rotas da API.

Ainda seguindo o exemplo da Figura 4, ao efetuar uma requisição do tipo *GET*, deve ser passado um *Token* de autenticação referente a um professor, caso contrário, a solicitação será rejeitada. Tendo em vista o *endpoint* mostrado na figura, chamado com o parâmetro ID=1, o retorno esperado é a lista de fases contidas na turma especificada, conforme o exemplo da Figura 5.


```
[
  {
    "ID": "1",
    "Fase": {
      "id": 1,
      "nome": "Bal\u00f5es",
      "url": "http:\/\/quimicotgames.com\/games\/level1\/",
      "criador": "Fulano da Dairel",
      "dificuldade": "D",
      "tempo_medio_seg": "600",
      "contem": {
        "elementos": [ "Ar", "He", "Ne", "Rn" ]
      },
      "vars": null
    }
  },
```


Figura 5 – Exemplo de mensagem de retorno da rota /professor/turmas/1/fases

Na Figura 5, são evidenciados os campos retornados pela API, onde:

- ❏ *'ID'*: Identificador da correlação entre a fase e a turma;
- ❏ 'Fase': um objeto que representa uma fase do jogo. Onde:
  - *'id'*: um identificador único para a fase;
  - 'nome': o nome da fase;
  - *'url'*: o endereço web onde a fase pode ser acessada;

- 'criador': o nome do autor da fase;
- 'dificuldade': o nível de dificuldade da fase;
- 'tempo_medio_seg': tempo médio em segundos que um jogador leva para completar a fase;
- 'contem': um objeto que especifica os conteúdos abordados pela fase como por exemplo os elementos químicos;
- *'vars'*: um atributo opcional que pode conter variáveis adicionais para customização da fase.

Capítulo 4

Resultados

Neste capítulo, serão discutidos os resultados alcançados com o desenvolvimento da proposta.

4.1 Validação e Documentação dos *Endpoints*

Ao utilizar o *Insomnia*[1] como cliente de API REST, é possível enviar requisições HTTP e receber as respostas correspondentes. Com isso, viabilizando a validação dos *endpoints*, utilizando-se de suas funcionalidades para:

- ❏ Enviar requisições HTTP para os *endpoints* da API, podendo definir o método, o cabeçalho, o corpo e os parâmetros da requisição.
- ❏ Receber as respostas HTTP dos *endpoints* da API, podendo visualizar o status, o cabeçalho, o corpo e os dados da resposta.
- ❏ Verificar se os *endpoints* da API estão funcionando corretamente e se retornam os dados esperados pelo aplicativo móvel.
- ❏ Identificar possíveis problemas ou inconsistências nos *endpoints* da API, como erros, falhas ou dados inválidos.
- ❏ Testar diferentes cenários e casos de uso para os *endpoints* da API, podendo alterar os valores das requisições e das respostas.

Além disso, foi utilizado também o *plugin Insomnia Documenter*[2], que torna possível o acesso da documentação via *Web*, sem a necessidade de instalação do software *Insomnia*[1]. A Figura 6 mostra um exemplo da página inicial da documentação gerada.

[1] https://insomnia.rest/
[2] https://github.com/insodoc/insomnia-documenter

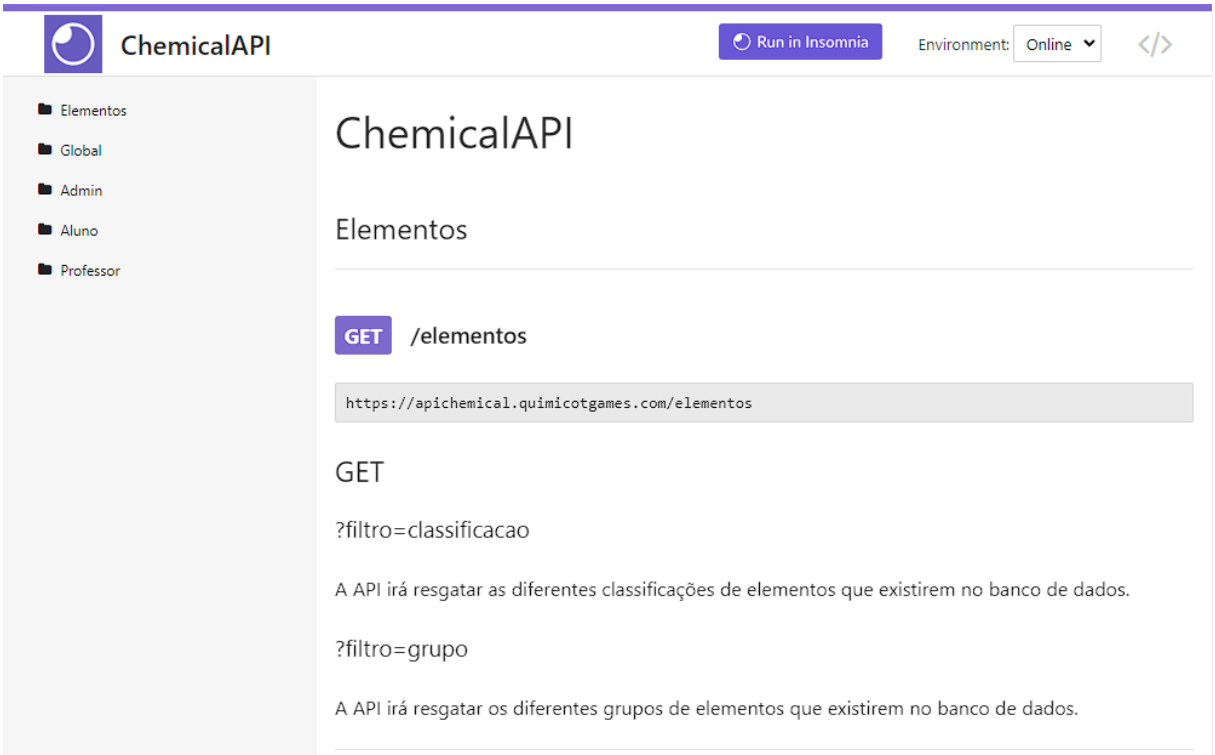

Figura 6 – Visão da documentação *Web*.

O conteúdo contido na documentação da página *Web* gerada pelo *plugin*, pode ser encontrada no Apêndice A. *ChemicalAPI* é a API resultante do processo de Prova de Conceito (PoC) abordado na seção 3.2. Essa documentação descreve como utilizar os *endpoints*, explicando parâmetros necessários para efetuar as requisições bem como o formato e dados da mensagem de resposta.

4.2 Testes Unitários

Para mitigar os riscos associados às alterações frequentes na PoC e garantir a qualidade de suas funcionalidades, foram implementados testes funcionais seguindo o padrão AAA (*Arrange*, *Act*, *Assert*). Esse padrão divide o código de teste em três etapas distintas: preparar o ambiente de teste, executar a ação a ser testada e verificar se o resultado obtido corresponde ao resultado esperado (OLIVEIRA, 2019).

A fim de automatizar esses testes, foi utilizado o *PHPUnit*, conforme mencionado na página 25. Ao adotar essa ferramenta em vez de depender exclusivamente do software *Insomnia*[1], o processo de teste se tornou mais ágil e confiável. Isso ocorre porque os testes automatizados evitam erros e a omissão de testes necessários que podem ocorrer durante a execução manual, que é mais suscetível a falhas humanas.

[1] https://insomnia.rest/

A Figura 7, representa a etapa de preparação do ambiente de teste (*Arrange*), onde o trecho de código executa a preparação do banco de dados para execução dos testes.

```php
/**
 * @depends test_clearDB
 */
public function test_createDB(): void
{
    $sql = file_get_contents(__DIR__ . "/../../Docs/IC.sql");
    $p_sql = Conexao::getInstance()->prepare($sql);
    $this->assertEquals($p_sql->execute(), 1);
}
```

Figura 7 – Padrão AAA, exemplo de *Arrange*.

Já a Figura 8, mostra a execução do teste (*Act*), onde é feita uma requisição HTTP e capturado sua resposta. Assim como é executado a verificação dos resultados (*Assert*), onde é conferido se a resposta da execução cumpre os requisitos para ser considerada funcional.

```php
/**
 * @depends Aluno_Test::test_login
 * @depends Aluno_Test::test_ingresso_turma
 */
public function test_listar_turmas($token): void
{
    // Act
    $client = new GuzzleHttp\Client();
    $response = $client->request('GET', 'http://' . $_ENV['URL_API'] . '/aluno/turmas', [
        'headers' => [
            'Authorization' => "Bearer {$token}"
        ]
    ]);

    // Assert
    $this->assertEquals('application/json', $response->getHeader('content-type')[0]);
    $this->assertEquals(200, $response->getStatusCode());
    $json_body = json_decode($response->getBody()->getContents(), true);
    $this->assertIsArray($json_body);
    $must_be_keys = [
        "Aluno_Turma_id", "Turma_id", "Turma_nome",
        "Turma_ano", "Escola_id", "Escola_nome",
        "Professor_id", "Professor_nome", "Professor_email"
    ];
    foreach ($must_be_keys as $key) {
        $this->assertArrayHasKey($key, $json_body[0]);
    }
}
```

Figura 8 – Padrão AAA, exemplo de *Act* e *Assert*.

4.3 Implantação

Para implantar a API desenvolvida, foi necessário o cumprir alguns pré-requisitos para o seu funcionamento, tais como: processamento de recursos *PHP*, *HTML*, *CSS* e *JavaScript*, habilitação do módulo '*mod_rewrite.c*' para o redirecionamento de rotas para o *Slim Framework*, e disponibilização do sistema gerenciador de banco de dados *MySQL*. O módulo '*mod_rewrite.c*' é um módulo do servidor Apache que permite reescrever as URLs solicitadas de forma dinâmica, baseado em um conjunto de regras definidas pelo usuário. O '*mod_rewrite.c*' usa uma sintaxe baseada em expressões regulares para verificar e transformar as URLs, podendo redirecionar um Localizador Uniforme de Recursos (URL) para outro, ou invocar um *proxy* interno (Apache Software Foundation, 2021).

Foram realizadas as seguintes etapas para a implantação da API em um servidor na Internet:

- ❏ Efetuar a assinatura em um serviço de hospedagem de sites, escolhendo-se um plano adequado ao projeto.

- O serviço de hospedagem selecionado utilizava do painel de controle *cPanel*, o qual permite gerenciar diversos aspectos da hospedagem, como domínios, *e-mails*, arquivos e bancos de dados.

  Pelo *cPanel*, foram feitas as seguintes etapas de configuração:

  - Foi vinculado o domínio 'quimicotgames.com' a hospedagem;
  - Criação da estrutura de pastas dentro do diretório padrão '*public_html*', onde ficam os arquivos do site. Também foi criado um diretório chamado '*APIs*' para armazenar as pastas referentes às API's, e dentro dele um diretório chamado '*ChemicalAPI*' contendo todo o código da API desenvolvida por meio da PoC;
  - Para evitar exposição de código, foi utilizado do arquivo oculto '*.htaccess*', o qual é usado para a configuração do site, para efetuar o bloqueio da navegação pelos diretórios da pasta '*APIs*' usando um arquivo , um arquivo de configuração que permite alterar o comportamento do servidor *Apache*;
  - Foi criado um subdomínio chamado '*apichemical*', viabilizando o acesso das funcionalidades da API por um caminho simplificado e mais amigável;
  - Foi necessário configurar outro arquivo '*.htaccess*', vide Figura 9, para redirecionar toda requisição encaminhada para o subdomínio para o arquivo '*index.php*' da API, que é o arquivo responsável por tratar as chamadas e retornar os dados.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /
    RewriteRule ^index\.php$ - [L]
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule . /index.php [L]
    RewriteRule .* - [env=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

Figura 9 – Redirecionador *.htaccess*

- Por meio do painel *phpMyAdmin*, que é uma ferramenta Web para gerenciar bancos de dados *MySQL*, foi criada uma instância de banco de dados para armazenar os dados da API. Nessa instância, foram inseridos os dados referentes aos elementos químicos da tabela periódica.

Conforme visto, foi necessário realizar alguns ajustes na configuração do serviço de hospedagem para realizar a implantação da API e permitir o seu correto funcionamento. Foi necessário configurar o ambiente, criar a estrutura de pastas, reescrever as URLs, criar o subdomínio e provisionar a instância do banco de dados.

Com isso, a API ficou disponível para ser acessada e testada por meio do endereço 'https://apichemical.quimicotgames.com'.

4.4 Resultados preliminares

Para captura de informação e análise preliminar de uso da API, foi solicitado um teste voluntário por cinco alunos do ensino médio da Escola Estadual Messias Pedreiro, em Uberlândia, onde os mesmos utilizaram o jogo analisado durante o prazo de uma semana. Ao final desse período, dois tipos de avaliações foram feitos, uma análise de desempenho e uma análise de registros (*logs*).

4.4.1 Análise de Desempenho

De maneira automatizada, a API teve como um de seus objetivos elaborar uma análise de desempenho por meio da utilização de questionários de pós-avaliação para as fases integradas ao ecossistema. O projeto contemplou análises em quatro níveis, sendo eles, nível de turma, de fase, de questionário, ou de um aluno.

Sobre os resultados dessas análises, podemos obter as seguintes informações:

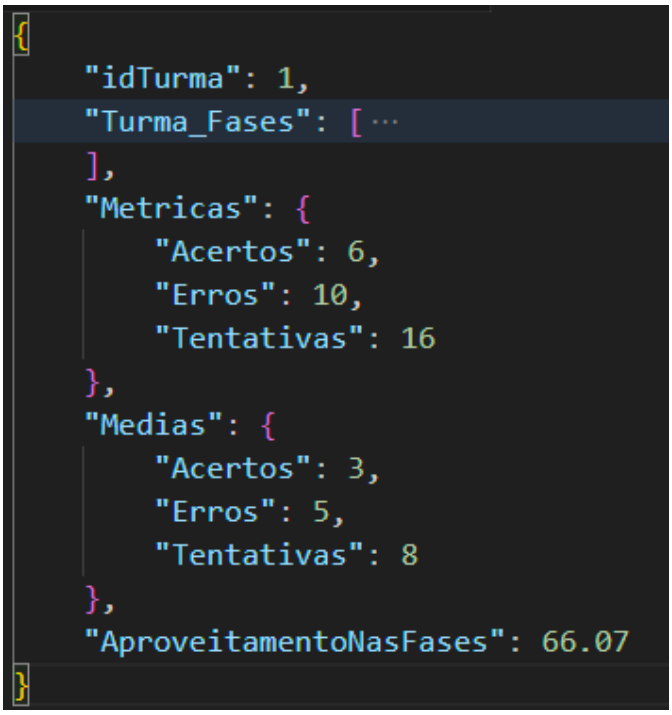

Figura 10 – Resposta da análise de nível turma.

Turma:

- A turma teve um total de 6 acertos e 10 erros em 16 tentativas, com uma média de 3 acertos, 5 erros e 8 tentativas por fase.

- ❏ A turma teve um aproveitamento nas fases de 66.07%, que é o percentual resultante da média dos aproveitamentos de cada fase.

Já mediante a cada fase individualmente, os dados extraídos foram:

Fase 1:

- ❏ A turma teve 5 acertos e 9 erros em 14 tentativas, com uma média de 1 acerto, 2 erros e 2 tentativas por questão.
- ❏ A turma jogou 6 dos 11 questionários disponíveis.
- ❏ A turma teve um aproveitamento de 57.14%, que é o percentual resultante da média dos aproveitamentos de cada questão, conforme a Equação 1.

Fase 2:

- ❏ A amostragem dessa fase foi baixa, onde apenas 2 alunos acessaram.
- ❏ A turma teve 1 acerto e 1 erro em 2 tentativas, com uma média de 1 acerto, 1 erro e 1 tentativa por questão.
- ❏ A turma jogou 2 dos 10 questionários disponíveis.
- ❏ A turma teve um aproveitamento de 75%, que é o percentual resultante da média dos aproveitamentos de cada questão, conforme a Equação 1.

Para se obter os aproveitamentos acima citados, foram utilizadas as seguintes equações:

$$Ap.\ da\ Fase = \frac{\sum_{n=1}^{qtd.Questoes} Ap.\ da\ questao\ n}{qtd.\ Questoes} \tag{1}$$

$$Ap.\ da\ questao = \frac{\sum_{n=1}^{qtd.Alunos} Aproveitamento\ do\ aluno\ n}{qtd.\ Alunos} \tag{2}$$

$$Aproveitamento\ do\ aluno = \frac{1}{1 + penalidade} \tag{3}$$

4.4.2 Análise de *logs*

A fins de explorar possíveis evoluções para o ecossistema, foi feito uma análise classificativa de maneira manual, utilizando da linguagem de programação *Python* para auxiliar na iteração dos dados.

Dentre os dados analisados, a estrutura continha os seguintes atributos:

- ❏ "aluno": identificador do aluno que esteve envolvido na interação;
- ❏ "turma_fase": identificador da fase que o aluno jogou e sua respectiva turma;

- ❏ “detalhes”: descritivo sobre a interação que o aluno teve com o jogo naquele determinado momento;
- ❏ “objeto”: JSON contendo dados complementares ao registro, como quantidade de moedas obtidas e obstáculo que ocasionou a derrota.
- ❏ “tipo”: tipo correspondente de impacto que a interação provocou, como derrota, vitória, fim de jogo, ou conclusão de uma etapa do jogo.
- ❏ “comeco”: contendo o momento de inicio da partida;
- ❏ “fim”: contendo o momento de encerramento da partida.

A análise dos *logs* revelou alguns aspectos relevantes sobre o comportamento e o desempenho dos jogadores, assim como pontos de melhoria a serem implementados no jogo. A Figura 11 mostra uma visão geral estruturada dos resultados das análises preliminares de *logs* realizada.

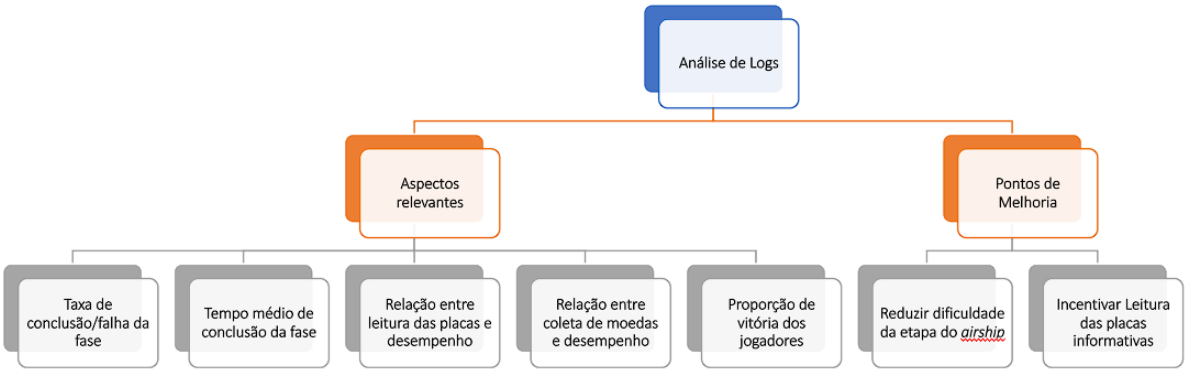

Figura 11 – Resultados - Análise de *logs*.

Sobre os aspectos relevantes, podemos destacar:

1. A taxa de conclusão/falha da fase, que indica o nível de dificuldade e engajamento do jogo, o qual, para o jogo analisado, o resultado foi equivalente ao da análise de desempenho (subseção 4.4.1).
2. O tempo médio de conclusão da fase, que indica o tempo de exposição e aprendizagem do conteúdo, o qual se mostrou muito baixo, seja para vitórias ou derrotas;
3. A relação entre a leitura das placas informativas e o desempenho do jogador, que indica o grau de atenção e interesse pelo conteúdo, essa relação por sua vez mostrou que os conteúdos escritos no jogo não refletiam o impacto desejado, não promovendo alterações no desempenho do jogador;

4. A relação entre a coleta de moedas e o desempenho do jogador, que indica o grau de motivação e recompensa pelo jogo, a partir dessa relação, foi possível identificar que jogadores que coletavam mais moedas tinham as maiores taxas de vitórias do jogo;

5. A proporção de vitória dos jogadores, que indica o nível de desafio e satisfação pelo jogo, no qual o resultado foi equivalente ao da análise de desempenho (subseção 4.4.1).

A análise também permitiu identificar pontos de melhoria para o design e a implementação do jogo, como por exemplo:

1. Reduzir a dificuldade da etapa do *airship*, trecho da fase do jogo onde o jogador é desafiado a desviar de obstáculos enquanto mantém seu personagem em um dirigível, a qual apresentou a maior taxa de falha dos jogadores;

2. Aumentar a duração do jogo, que apresentou um tempo médio de conclusão muito curto;

3. Incentivar a leitura das placas informativas, que não apresentou influência no desempenho dos jogadores.

Essas contribuições podem auxiliar os desenvolvedores de jogos educativos a criar experiências mais eficazes e atrativas para os alunos, bem como os professores a avaliar o impacto dos jogos no processo de ensino-aprendizagem. Além disso, a API proposta na PoC, pode ser utilizada para analisar outros tipos de jogos e contextos educacionais, desde que sejam respeitadas as especificações de comunicação.

4.4.3 Desafios encontrados

Um dos desafios encontrados neste trabalho foi a impossibilidade de padronizar a captura de alguns dados relevantes dos jogos sem a criação de uma API *Client-Side*, que se integre com um ou mais *Frameworks* ou *Engines* de jogos. Esses dados incluem, por exemplo, um mapa de etapas dos jogos, ações determinísticas e mais detalhes sobre a interação do jogador com os elementos do jogo. Esses dados poderiam enriquecer a análise do impacto do jogo nos alunos e fornecer mais informações para os desenvolvedores e professores.

Uma possível solução para esse problema seria desenvolver uma API de lado cliente que pudesse se comunicar com a API de lado servidor e enviar os dados dos jogos de forma padronizada e segura. Essa API de lado cliente poderia ser implementada em diferentes linguagens e plataformas, de acordo com os *Frameworks* ou *Engines* de jogos mais utilizados na criação de jogos educativos. Dessa forma, seria possível capturar

dados mais específicos e personalizados dos jogos, sem depender apenas dos *logs* gerados manualmente pelos desenvolvedores.

Capítulo 5

Conclusão

Neste capítulo, são apresentadas as principais conclusões obtidas a partir da realização deste trabalho, que teve como objetivo desenvolver mecanismos para análise de desempenho de estudantes que utilizam um jogo digital sério no aprendizado, e verificar como esses mecanismos podem contribuir para o acompanhamento do processo de aprendizagem e a melhoria da experiência do usuário. Além disso, são discutidas as limitações do estudo, as contribuições acadêmicas e as sugestões para trabalhos futuros.

5.1 Principais Contribuições

Mediante os resultados apresentados no Capítulo 4, as principais contribuições deste trabalho, são:

- A proposta e o desenvolvimento de uma Interface de Programação de Aplicação (API) que permite a integração de jogos educativos a um ecossistema de aprendizagem, facilitando a comunicação e a troca de dados entre os jogos e o sistema educacional. Essa API pode ser uma contribuição significativa para a melhoria da educação digital e do ensino de Química através de jogos sérios, bem como para outros contextos educacionais que envolvam jogos digitais;

- A validação e a documentação dos *endpoints* da API usando o software *Insomnia* e o *plugin Insomnia Documenter*, garantindo a funcionalidade e a acessibilidade da API. Essa validação também auxiliou na implementação de uma ferramenta Web desenvolvida por outro estudante, que tem o objetivo de auxiliar o acompanhamento do ensino digital;

- A implementação e a automação de testes unitários seguindo o padrão AAA (*Arrange, Act, Assert*) e usando o *PHPUnit*, garantindo a qualidade e a confiabilidade da API. Esses testes seguiram um método reconhecido e usaram uma ferramenta adequada para evitar erros e inconsistências no código da API;

- ❑ A realização e a análise de um teste preliminar e voluntário com estudantes do ensino médio, que usaram por uma semana um jogo sério que tinha como objetivo o ensino de elementos químicos categorizados como Gases Nobres. Esse teste avaliou o impacto do jogo no processo de ensino-aprendizagem, medindo o nível de conhecimento, desempenho, comportamento e engajamento dos estudantes com o jogo;

- ❑ A identificação de aspectos relevantes e pontos de melhoria para o design e a implementação do jogo, baseados nos *logs* das interações dos estudantes com o jogo. Essas sugestões podem auxiliar os desenvolvedores de jogos educativos a criar experiências mais eficazes e atrativas para os estudantes, bem como os professores a avaliar o impacto dos jogos no processo de ensino-aprendizagem.

5.2 Trabalhos Futuros

Com as investigações e análises levantadas, podemos citar novos objetivos futuros que possam originar novos trabalhos com o intuito de evoluir a extração de informações sobre o jogo, que atualmente é feita pela API de lado servidor. Primeiramente, é necessário executar mais testes e estudos de casos para coletar mais dados para que seja possível propor novos indicadores de análise de aprendizagem e comportamento nos jogos.

Além disso, sugere-se a criação de uma API de lado cliente, que possa se integrar a diferentes ferramentas de desenvolvimento, como Unity, Unreal Engine, Game Maker etc. Essa API de lado cliente seria responsável por enviar dados mais específicos sobre o jogo para a API de lado servidor, como posição em relação a um mapa de etapas dos jogos, ações determinísticas e mais detalhes sobre a interação do jogador com os elementos do jogo. Dessa forma, a extração de informações seria mais precisa e completa, permitindo uma melhor análise e compreensão do jogo.

5.3 Contribuições em Produção Bibliográfica

Os resultados deste trabalho ainda não foram publicados em nenhum veículo científico. Entretanto, houve uma publicação relacionada e decorrente do final da pesquisa de Iniciação Científica e início deste Trabalho de Conclusão de Curso:

DAIREL, J. G., TUPINAMBÁ, R. C., SILVA, Y. G. P., ARAÚJO, R. D. Em direção a um ecossistema de software para apoio ao ensino de química por meio de jogos digitais. In: **Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital**. Porto Alegre, RS, Brasil: SBC, 2021. p. 689–692. (DAIREL et al., 2021)

Referências

ALONSO-FERNÁNDEZ, C. et al. Applications of data science to game learning analytics data: A systematic literature review. **Computers & Education**, v. 141, p. 103612, 2019. ISSN 0360-1315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360131519301654>. Citado na página 17.

Apache Software Foundation. **mod_rewrite - Apache HTTP Server Version 2.4**. 2021. Disponível em: <https://httpd.apache.org/docs/current/mod/mod_rewrite.html>. Citado na página 35.

AUTH0. **Introduction to JSON Web Tokens**. 2021. Acessado em: 12/06/2023. Disponível em: <https://jwt.io/introduction/>. Citado na página 21.

BALDISSERA, O. **O que é serious game, estratégia poderosa de gamificação**. 2021. <https://posdigital.pucpr.br/blog/serious-game>. Citado na página 21.

_____. **O que é serious game, uma das estratégias mais poderosas de gamificação**. 2021. <https://posdigital.pucpr.br/blog/serious-game>. Accessed: 2023-06-20. Citado na página 12.

BIEHL, M. **RESTful API Design**. 1. ed. [S.l.]: CreateSpace Independent Publishing Platform, 2016. 294 p. (API-University Series). Citado na página 18.

BRAY, T. **The JavaScript Object Notation (JSON) Data Interchange Format**. RFC Editor, 2017. RFC 8259. (Request for Comments, 8259). Disponível em: <https://www.rfc-editor.org/info/rfc8259>. Citado na página 21.

CASSIMIRO, W. **Diferenças entre jogos, jogos sérios e gamification**. 2020. <https://espresso3.com.br/diferencas-entre-jogos-jogos-serios-e-gamification/>. Accessed: 2023-06-20. Citado na página 12.

Codez Up. **What is POJO in Java with Explanation and Example**. 2020. <https://codezup.com/what-is-pojo-in-java-with-explanation-and-example/>. Accessed: 2022-12-30. Citado na página 19.

CROCKFORD, D. **The application/json Media Type for JavaScript Object Notation (JSON)**. RFC Editor, 2006. RFC 4627. (Request for Comments, 4627). Disponível em: <https://www.rfc-editor.org/info/rfc4627>. Citado na página 21.

DAIREL, J. G. et al. Em direção a um ecossistema de software para apoio ao ensino de química por meio de jogos digitais. In: **Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital**. Porto Alegre, RS, Brasil: SBC, 2021. p. 689–692. Citado na página 43.

DARIN, T. **Jogos sérios: desenvolver habilidades cognitivas pode ser divertido!** 2020. <https://horizontes.sbc.org.br/index.php/2020/11/jogos-serios-desenvolver-habilidades-cognitivas-pode-ser-divertido/>. Citado na página 21.

DINIZ, Y. **Entenda o que é Learning Analytics e para que serve**. 2020. <https://educacao.imaginie.com.br/learning-analytics/>. Accessed: 2023-06-20. Citado na página 12.

Ecma International. **The JSON Data Interchange Syntax**. 2nd. ed. [S.l.], 2017. Disponível em: <https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>. Citado na página 21.

FIELDING, R. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Doutorado) — UNIVERSITY OF CALIFORNIA, IRVINE, 01 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado na página 18.

FOWLER, M. **POJO**. 2003. <https://www.martinfowler.com/bliki/POJO.html>. Accessed: 2023-06-12. Citado na página 19.

_____. **Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch**. Pearson Education, 2012. (Addison-Wesley Signature Series (Fowler)). ISBN 9780133065213. Disponível em: <https://books.google.com.br/books?id=vqTfNFDzzdIC>. Citado na página 20.

FREIRE, M. et al. Game learning analytics: Learning analytics for serious games. In: _____. [S.l.: s.n.], 2016. p. 1–29. ISBN 978-3-319-17727-4. Citado 2 vezes nas páginas 12 e 16.

_____. Game learning analytics: Learning analytics for serious games. In: _____. [s.n.], 2016. p. 1–29. ISBN 978-3-319-17727-4. Disponível em: <https://www.researchgate.net/publication/305371728_Game_Learning_Analytics_Learning_Analytics_for_Serious_Games>. Citado na página 18.

HOPKINS, C. **PHP master**. SitePoint, 2013. Disponível em: <https://www.sitepoint.com/the-mvc-pattern-and-php-1/>. Citado na página 20.

International Organization for Standardization. **Information technology – The JSON data interchange syntax**. [S.l.], 2017. Disponível em: <https://www.iso.org/standard/71616.html>. Citado na página 21.

JONES, M. B.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. RFC Editor, 2015. RFC 7519. (Request for Comments, 7519). Disponível em: <https://www.rfc-editor.org/info/rfc7519>. Citado na página 21.

KANG, J.; LIU, M.; QU, W. Using gameplay data to examine learning behavior patterns in a serious game. **Computers in Human Behavior**, Elsevier, v. 72, p. 757–770, 2017. ISSN 0747-5632. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0747563216306975>. Citado na página 18.

KARLSSON, T.; BRUSK, J.; ENGSTRöM, H. Level design processes and challenges: A cross section of game development. **Games and Culture**, 11 2022. Disponível em: <https://doi.org/10.1177/15554120221139229>. Citado na página 23.

LIMA, É.; MOITA, F. A tecnologia e o ensino de química: jogos digitais como interface metodológica. In: _____. [S.l.: s.n.], 2011. p. 131–154. ISBN 9788578791247. Citado na página 12.

LOCKHART, J. **Slim Framework**. 2013. Disponível em: <https://www.slimframework.com/>. Citado na página 25.

MELO, D. et al. Uma estratégia de game learning analytics para avaliar level design em um jogo educacional. In: **Anais do XXXI Simpósio Brasileiro de Informática na Educação**. Porto Alegre, RS, Brasil: SBC, 2020. p. 622–631. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/12818>. Citado na página 23.

OLIVEIRA, M. **Teste unitário e o padrão AAA (*Arrange, Act, Assert*)**. 2019. Disponível em: <https://medium.com/@alamonunes/teste-unitário-e-o-padr~ao-aaa-arrange-act-assert-cb81d587368a>. Citado na página 33.

OLIVEIRA, R. de et al. Pro-avaliajs: Protocolo para planejamento e execução da avaliação da reação e aprendizagem de jogos sérios. In: . [s.n.], 2022. Disponível em: <https://www.researchgate.net/publication/365654378_Pro-AvaliaJS_Protocolo_para_planejamento_e_execucao_da_avaliacao_da_reacao_e_aprendizagem_de_jogos_serios>. Citado na página 23.

OLIVEIRA, R. de; ROCHA, R. Avaliajs: Planejamento da avaliação do desempenho de alunos em jogos sérios. In: . [s.n.], 2021. Disponível em: <https://www.researchgate.net/publication/356587494_AvaliaJS_Planejamento_da_Avaliacao_do_Desempenho_de_Alunos_em_Jogos_Serios>. Citado na página 23.

PHP Framework Interop Group. **PSR-7: HTTP message interfaces**. 2014. Disponível em: <https://www.php-fig.org/psr/psr-7/>. Citado na página 25.

PLANSKY, R. **Definição, restrições e benefícios do modelo de arquitetura REST**. 2014. [Online; accessed 20-October-2021]. Disponível em: <https://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest>. Citado na página 18.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. [S.l.]: AMGH, Bookman, 2016. 968 p. ISBN 9788580555332. Citado na página 24.

REHKOPF, M. **Testes de software automatizados**. 2021. <https://www.atlassian.com/br/continuous-delivery/software-testing/automated-testing>. Acessado em 9 de maio de 2023. Citado na página 25.

ROCHA, R. V. d.; BITTENCOURT, I. I.; ISOTANI, S. Análise, projeto, desenvolvimento e avaliação de jogos sérios e afins: uma revisão de desafios e oportunidades. In: SBC. **Anais do XXVI Simpósio Brasileiro de Informática na Educação**. 2015. p. 692–701. Disponível em: <https://doi.org/10.5753/cbie.sbie.2015.692>. Citado na página 22.

SAVI, R.; ULBRICHT, V. R. Jogos digitais: aprendizagem divertida. **Revista Novas Tecnologias na Educaç ao**, v. 6, n. 1, 2008. Disponível em: <https://doi.org/10.22456/1679-1916.14405>. Citado na página 21.

SCHELL, J. **The Art of Game Design: A book of lenses**. Taylor & Francis, 2008. ISBN 978-0123694966. Disponível em: <https://books.google.com.br/books?id=LP5xOYMjQKQC>. Citado na página 12.

SILVA, D. et al. Glboard: um sistema para auxiliar na captura e análise de dados em jogos educacionais. In: **Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital**. Porto Alegre, RS, Brasil: SBC, 2022. p. 959–968. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/23733>. Citado na página 22.

SOUZA, M. B. de. **Padrão MVC (Model-View-Controller) tutorial**. 2011. Disponível em: <https://www.devmedia.com.br/padrao-mvc-java-magazine/21995>. Citado na página 20.

TATROE, K.; LERDORF, R. **Programming PHP**. O'Reilly Media, 2006. Disponível em: <https://books.google.com/books/about/Programming_PHP.html?id=h-E1lVko-skC>. Citado na página 25.

The PHP Group. **PHP: Hypertext Preprocessor**. The PHP Group, 2018. Disponível em: <http://www.php.net/>. Citado na página 25.

W3Techs. **Usage of server-side programming languages for websites**. W3Techs, 2021. Disponível em: <https://w3techs.com/technologies/overview/programming_language/all>. Citado na página 25.

WILLIANS, W. **O que é DTO?** School Of Net, 2022. Disponível em: <https://fullcycle.com.br/o-que-e-dto>. Citado na página 20.

Apêndices

APÊNDICE A

ChemicalAPI

A.1 Elementos

A.1.1 GET /elementos

Exemplo de URL:

https://apichemical.quimicotgames.com/elementos

A API resgatará as todos os elementos que existirem no banco de dados.

A.1.2 GET /elementos/filtered

Exemplo de URL:

https://apichemical.quimicotgames.com/elementos/filtered

- '?filtro=classificacao': A API resgatará as diferentes classificações de elementos que existirem no banco de dados;
- '?filtro=grupo': A API resgatará os diferentes grupos de elementos que existirem no banco de dados.

A.1.3 GET /elementos/get/<Sigla>

Exemplo de URL:

https://apichemical.quimicotgames.com/elementos/get/<Sigla>

Retorna o elemento da Sigla informada. A API resgatará um objeto que contenha a Sigla assim como a *Path*.

Na URL deve ser passado como argumento a Sigla do Elemento Químico.
*Deve-se substituir o texto <Sigla> pela Sigla do Elemento Químico.

A.1.4 GET /elementos/names

Exemplo de URL:

```
https://apichemical.quimicotgames.com/elementos/names
```

Retorna o elemento da Sigla informada. A API resgatará uma lista contendo a "sigla" e o "nome" dos elementos da TP.

A.2 Global

A.2.1 GET /

Exemplo de URL:

```
https://apichemical.quimicotgames.com
```

Acessa a documentação da API.

A.2.2 GET /fases

Exemplo de URL:

```
https://apichemical.quimicotgames.com/fases
```

Retorna uma lista contendo as fases cadastradas no sistema.

A.2.3 GET /escola

Exemplo de URL:

```
https://apichemical.quimicotgames.com/escola
```

Retorna uma lista contendo as escolas cadastradas no sistema.

A.2.4 POST /login

Exemplo de URL:

```
https://apichemical.quimicotgames.com/login
```

Executa a ação de *Login* para um Aluno ou Professor.
Para tal ação deve-se passar os seguintes atributos na `body` em formato JSON:

```
{
    "email": "required|string|email",
    "senha": "required|string"
}
```

Também é possível renovar o *Login* utilizando um *Token* que ainda não espirou. Para isso basta enviar o *Token* no cabeçalho *"Authorization"* seguindo o padrão do Tipo *"Bearer"* e não enviar conteúdo na `body`.

A.3 Administrador

A.3.1 POST /admin/login

Exemplo de URL:

https://apichemical.quimicotgames.com/admin/login

Executa a ação de *Login* para um Administrador.
Para tal ação deve-se passar os seguintes atributos na `body` em formato JSON:

```
{
    "email": "required|string|email",
    "senha": "required|string"
}
```

A.3.2 POST /admin/logsCsv

Exemplo de URL:

https://apichemical.quimicotgames.com/admin/logsCsv

Resgata os *logs* registrados no banco de dados em formato de Valores Separados Por Virgula (CSV).

A.3.3 POST /admin/fase

Exemplo de URL:

https://apichemical.quimicotgames.com/admin/fase

Adiciona uma nova fase no sistema.
Deve-se seguir o seguinte padrão de atributos:

```
{
    "nome": "required|string|max:255",
    "url": "required|url|max:~65535",
    "criador": "required|string|max:255",
    "dificuldade": "required|char|[F, M, D]",
    "tempo_medio_seg": "optional|int|min:1|max:~2147483648",
    "contem": "optional|json",
    "vars": "optional|json"
}
```

Observações:
No atributo `dificuldade` as letas F, M e D representam respectivamente Fácil, Médio e Difícil.
Atributos com especificados como `optional` caso não queira colocar um valor a eles, é necessário não enviar a sua chave na `body`. Ou seja, caso seja passado um valor vazio ou `null` o campo não será ignorado.
O atributo `contem` tem como objetivo salvar uma lista de itens que a fase aborda, tais como o nome dos elementos químicos.
O atributo `vars` tem como objetivo salvar valores que são necessários para que a fase funcione corretamente.

A.3.4 POST /admin/escola

Exemplo de URL:

https://apichemical.quimicotgames.com/admin/escola

Adiciona uma nova escola no sistema.
Deve-se seguir o seguinte padrão de atributos:

```
{
    "nome": "required|unique|string|max:255"
}
```

A.3.5 GET /admin/professor

Exemplo de URL:

https://apichemical.quimicotgames.com/admin/professor

Retorna uma lista contendo os professores cadastrados no sistema.

A.3.6 POST /admin/acessoProfessor/<ID>

Exemplo de URL:

https://apichemical.quimicotgames.com/admin/acessoProfessor/<ID>

Aceita ou Revoga o cadastro de um professor
Na URL deve ser passado como argumento o 'ID' do professor.
Deve-se seguir o seguinte padrão de atributos:

```
{
    "metodo": "required|string|[revogar, aceitar]"
}
```

Observações:
O atributo `metodo` tem como finalidade descrever a ação a ser tomada, sendo elas `revogar` ou `aceitar` o acesso de um professor.

A.4 Aluno

A.4.1 [GET|POST] /aluno

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno

- **GET**: Busca os Dados do Aluno
  Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
  A API resgatará os dados de quem o *Token* pertence.

- **POST**: Efetua o cadastro de um aluno.
  Deve-se seguir o seguinte padrão de atributos em formato JSON:

```
{
    "nome": "required|string|max:255",
    "email": "required|unique|email|max:255",
    "senha": "required|string|min:8|max:255"
}
```

A.4.2 POST /aluno/login

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno/login

Executa a ação de Login para um Aluno.
Para tal ação deve-se passar os seguintes atributos na `body` em formato JSON:

```
{
    "email": "required|string|email",
    "senha": "required|string"
}
```

Também é possível renovar o Login utilizando um *Token* que ainda não espirou. Para isso basta enviar o *Token* no cabeçalho *"Authorization"* seguindo o padrão do Tipo *"Bearer"* e não enviar conteúdo na `body`.

A.4.3 GET /aluno/turmas

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno/turmas

Busca as Turmas que o Aluno participa
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
A API resgatará os dados de quem o *Token* pertence.

A.4.4 POST /aluno/turmas/ingressar

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno/turmas/ingressar

Faz o ingresso do aluno na turma.
Deve-se informar o *Token* do tipo *"Bearer"* no campo *"Authorization"*.
Deve-se seguir o seguinte padrão de atributos:

```
{
    "turma": "required|int",
    "senha": "required|string|min:8|max:255"
}
```

Observações:
O atributo `turma` se refere ao 'ID' da turma.
O atributo `senha` se refere à senha de acesso da turma.

A.4.5 GET /aluno/turmas/fases

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno/turmas/fases?idTurma=<ID>

A API resgatará as diferentes fases disponíveis para a turma e seus 'ID's de correlação.

A.4.6 [GET|POST] /aluno/turmas/fases/<ID>/quiz

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno/turmas/fases/<ID>/quiz

- **GET**: A API resgatará um *quiz* referente a fase da turma.
  Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.

- **POST**: Registra a resposta do Quiz selecionado.
  Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
  Deve conter os seguintes parâmetros no formato JSON:

```
{
    "escolha": "required|int",
    "data_hora": "required|string|'Y-m-d H:i:s'",
    "quiz": "required|int"
}
```

Observações:
Deverá ser indicado o 'ID' da correlação "Turma_Fase".
O parâmetro `quiz` deve ser o 'ID' válido de um "quiz" cadastrado para aquela fase, o qual a resposta será gravada.

A.4.7 GET /aluno/resultados/qtdJogadas

Exemplo de URL:

https://apichemical.qui(...)/aluno/resultados/qtdJogadas ?turma=<ID>

Retorna a quantidade de vezes que o aluno jogou cada fase (Turma_Fase).
A API resgatará um objeto que contenha o "Turma_Fase" e a quantidade de vezes jogada.
Na URL deve ser passado como argumento o 'ID' da Turma do Aluno.

A.4.8 POST /aluno/log

Exemplo de URL:

https://apichemical.quimicotgames.com/aluno/log

Registra algum evento ocorrido durante o jogo.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Deve conter os seguintes parâmetros no formato JSON:

```
{
    "turma_fase": "required|int",
    "detalhes: "required|string",
    "tipo": "required|string",
    "comeco": "required|string|‘Y-m-d H:i:s’",
    "fim": "required|string|’Y-m-d H:i:s’",
    "objeto": "string|JSON"
}
```

A.5 Professor

A.5.1 [GET|POST] /professor

Exemplo de URL:

https://apichemical.quimicotgames.com/professor

- **GET**: Busca os Dados do Professor
  Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
  A API resgatará os dados de quem o *Token* pertence.

- **POST**: Efetua o cadastro de um Professor.
  Deve-se seguir o seguinte padrão de atributos em formato JSON:

```
{
    "nome": "required|string|max:255",
    "email": "required|unique|email|max:255",
    "cpf": "required|unique|cpf",
    "senha": "required|string|min:8|max:255",
    "telefone": "optional|string|max:15|telefone",
    "escolas": "optional|int array"
}
```

Observações:

O atributo `cpf` deve ser um Cadastro de Pessoa Física válido, podendo conter ou não a pontuação (Observação: O sistema salva sem nenhuma pontuação).

O atributo `telefone` deve ser um telefone válido, seguindo o padrão brasileiro (DDD + Número) sem pontuação.

Atributos com especificados como `optional` caso não queira colocar um valor a eles, é necessário não enviar a sua chave na `body`. Ou seja, caso seja passado um valor vazio ou `null` o campo não será ignorado.

O atributo `escolas` deve conter um *array* de 'ID's, onde cada 'ID' irá indicar uma escola em que o professor atua. O professor deverá aguardar que um administrador aceite sua solicitação para cada escola.

A.5.2 POST /professor/login

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/login

Executa a ação de Login para um Professor.
Para tal ação deve-se passar os seguintes atributos na 'body em formato JSON:

```
{
    "email": "required|string|email",
    "senha": "required|string"
}
```

Também é possível renovar o Login utilizando um *Token* que ainda não espirou. Para isso basta enviar o *Token* no cabeçalho *"Authorization"* seguindo o padrão do Tipo "Bearer"e não enviar conteúdo na body.

A.5.3 [GET|POST] /professor/turmas

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas

- **GET**: Retorna uma lista contendo todas as turmas do professor.
  Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
- **POST**: Cadastra uma nova turma para o professor.
  Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
  Deve-se seguir o seguinte padrão de atributos em formato JSON:

```
{
    "nome": "required|string|max:255",
    "escola": "required|int",
    "senha": "required|string|min:8|max:255",
    "ano": "int|Year"
}
```

Observações:
O atributo `senha` se refere à senha de acesso da turma.

A.5.4 GET /professor/turmas/alunos

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/alunos

Lista todos alunos do professor.

Observações:
É possível pegar os alunos de uma turma especifica utilizando o seguinte parâmetro na *Query*: "?turma=<ID>". Onde deve-se substituir o "<ID>" pelo 'ID' da turma a se recuperar os alunos.

A.5.5 POST /professor/turmas/quiz

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/quiz

Cadastra uma nova turma para o professor.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Deve-se seguir o seguinte padrão de atributos em formato JSON:

```
{
    "pergunta": "required|string|max:~65535",
    "turma_fase": "required|int",
    "alternativas": "required|array",
    "alt_correta": "required|int"
}
```

Observações:
O atributo `alternativas` deve ser um *array* contendo todas as alternativas de um 'quiz', cada uma dessas seguindo o seguinte formato:

```
{
    "descricao": "required|string|max:~65535",
    "justificativa": "optional|string|max:~65535"
}
```

O atributo `alt_correta` deve ser um inteiro referenciando o índice da alternativa correta no *array* do atributo `alternativas`.

A.5.6 POST /professor/turmas/quiz/<ID>/deletar

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/quiz/<ID>/deletar

Deleta o 'Quiz' especificado.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' do 'Quiz'.

A.5.7 POST /professor/turmas/quiz/<ID>/atualizar

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/quiz/<ID>/atualizar

Atualiza a pergunta do 'Quiz' especificado.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' do 'Quiz'.
Deve-se seguir o seguinte padrão de atributos no padrão JSON:

```
{
    "pergunta": "required|string|max:~65535"
}
```

A.5.8 POST (...)/quiz/<ID>/atualizarAlternativa/<ALT>

Exemplo de URL:

https://apichemical.qui(...)/professor/turmas/quiz/<ID>/atualizar/<ALT>

Atualiza a alternativa do 'Quiz' especificado.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Deve-se seguir o seguinte padrão de atributos no padrão JSON:

```
{
    "alternativas": "required|array"
}
```

Observações:
Na URL deve ser passado como argumento o 'ID' do 'Quiz' e 'ID' da Alternativa.
O atributo `alternativas` deve ser um *array* contendo todas as alternativas de um 'quiz', cada uma dessas seguindo o seguinte formato:

```
{
    "descricao": "required|string|max:~65535",
    "justificativa": "optional|string|max:~65535"
}
```

A.5.9 GET /professor/turmas/<ID>/fases

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/<ID>/fases

Retorna uma lista contendo as fases vinculadas na turma especificada.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' da turma.

A.5.10 POST /professor/turmas/<ID>/vincularFase

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/<ID>/vincularFase

Vincula uma Fase para uma Turma.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' da turma.
Deve-se seguir o seguinte padrão de atributos em formato JSON:

```
{
    "fase": "required|int"
}
```

Observações:
O atributo `fase` deve ser o 'ID' da fase que se deseja vincular (Adicionar) na turma.

A.5.11 POST /professor/turmas/<ID>/desvincularFase

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/<ID>/desvincularFase

Vincula uma Fase para uma Turma.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' da turma.
Deve-se seguir o seguinte padrão de atributos em formato JSON:

```
{
    "fase": "required|int"
}
```

Observações:
O atributo `fase` deve ser o 'ID' da fase que se deseja desvincular (Remover) da turma.

A.5.12 GET /professor/turmas/<ID>/fases

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/turmas/<ID>/fases

Ver fases vinculadas a uma turma, ou não vinculadas com "?vinculadas=false"
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Observações:
Na URL deve ser passado como argumento o 'ID' da turma.

A.5.13 GET /professor/turmas/<ID>/fases/<FASE>/quizes

Exemplo de URL:

https://apichemical.qui(...)/professor/turmas/<ID>/fases/<FASE>/quizes

Ver 'quizes' vinculados a uma fase da turma
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Observações:
Na URL deve ser passado como argumento o 'ID' da turma e o 'ID' da fase.

A.5.14 GET /professor/analises/quiz/<ID>

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/analises/quiz/<ID>

Retorna um objeto contendo os resultados das análises sobre o 'Quiz' especificado.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' do 'Quiz'.

A.5.15 GET /professor/analises/turmaFase/<ID>

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/analises/turmaFase/<ID>

Retorna um objeto contendo os resultados das análises sobre o 'Turma_Fase' especificado.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' do 'Turma_Fase'.

A.5.16 GET /professor/analises/aluno/<ID>

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/analises/aluno/<ID>

Retorna um objeto contendo os resultados das análises sobre o Aluno especificado.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' do Aluno.

A.5.17 GET /professor/analises/turma/<ID>

Exemplo de URL:

https://apichemical.quimicotgames.com/professor/analises/turma/<ID>

Retorna um objeto contendo os resultados das análises sobre a turma especificada.
Deve-se informar o *Token* do tipo *Bearer* no campo *"Authorization"*.
Na URL deve ser passado como argumento o 'ID' da Turma.