

Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências de Bauru
Departamento de Computação
Bacharelado em Ciências da Computação

Antônio Eugênio Domingues Silva

**Sistema de Ensino Remoto de Educação Física Para Crianças Através da
Gamificação**

Bauru
2021

Antônio Eugênio Domingues Silva

Sistema de Ensino Remoto de Educação Física Para Crianças Através da Gamificação

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação, como parte dos requisitos necessários à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Kleber Rocha de Oliveira

Bauru
2021

S586s

Silva, Antônio Eugênio Domingues

Sistema de ensino remoto de educação física para crianças através da gamificação / Antônio Eugênio Domingues Silva. -- Bauru, 2022
59 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru

Orientador: Prof. Dr. Kleber Rocha de Oliveira

1. Ensino. 2. Desenvolvimento de software. 3. Engenharia de software. 4. Educação física. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Antônio Eugênio Domingues Silva

Sistema de Ensino Remoto de Educação Física Para Crianças Através da Gamificação

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Kleber Rocha de Oliveira

Orientador

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Engenharia e Ciências
Coordenadoria de Curso de Engenharia de Energia

Prof^a. Dr^a. Simone das Graças Domingues Prado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências
Departamento de Ciência da Computação

Prof^a. Dr^a. Andrea Carla Gonçalves Vianna

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 7 de Março de 2022

À minha família que sempre me apoiaram e me impeliram para frente.

Agradecimentos

Agradeço primeiramente minha família por sempre me apoiarem e por possibilitar essa oportunidade e a liberdade de aprofundar nos meus interesses. Sempre incentivaram e discursavam sobre a importância da educação, e do valor do trabalho e do respeito.

Agradeço também aos meus amigos que vieram e também aos que já foram, pois além de contribuírem com o meu desenvolvimento como ser humano, ofereceram apoio e perspectivas diferentes em momentos difíceis e decisivos além de alegria compartilhada em momentos felizes.

Agradeço a todos os professores do curso de Ciência da Computação, que sempre estiveram dispostos e engajados em ensinar e educar. Agradeço a todos os funcionários que possibilitaram isso e também à UNESP Bauru por proporcionar experiências incríveis e lições importantes.

Por fim, agradeço meu orientador pela paciência e perspectivas que enriqueceram o trabalho.

“Nenhum homem é mais infeliz do aquele que nunca enfrentou a adversidade. Pois ele não está autorizado a provar a si mesmo.”

Sêneca

Resumo

Com o crescente uso do ensino à distância amplificado pelo isolamento social resultante da pandemia global de COVID-19, surge novos desafios para a educação e para a vida, ainda mais quando se refere a crianças em desenvolvimento. Problemas emocionais, cognitivos e sociais afetaram as crianças. E além disso, na situação de distanciamento social, o ensino remoto sofre com falta de engajamento e retenção do conhecimento pelos alunos. Este projeto relata o desenvolvimento de um sistema de ensino de educação física remotamente para crianças utilizando ferramentas, tecnologias e técnicas de desenvolvimento e gamificação modernas para enfrentar esses problemas.

Palavras-chave: Ensino a distância. Gamificação. Engenharia de Software. Banco de dados. Sistema web.

Abstract

The rising use of remote learning amplified by social isolation caused by the global pandemic of COVID-19 has created new challenges to education and to life, especially when referring to children in development. Emotional, cognitive and social problems stemming from social isolation has affected all people, but has been worse in children. Moreover, remote learning suffers from the lack of motivation and knowledge retention from the students. So being, this project consists of using modern software development and gamification tools and techniques to create a remote learning application to teach physical education to children in hopes of countering the problems presented.

Keywords: Distance learning. Gamification. Software engineering. Database. Web system.

Lista de figuras

Figura 1 – Camadas da engenharia de software	19
Figura 2 – Alguns elementos da gamificação	21
Figura 3 – Exemplo de fluxo de uma aplicação, onde o SGBD gerencia o banco de dados	23
Figura 4 – Exemplo de armazenamento de um documento de um repositório de pedidos	24
Figura 5 – Exemplo de diagrama entidade-relacionamento	25
Figura 6 – Exemplo de fluxo do gitflow utilizando <i>branches</i>	27
Figura 7 – Uma <i>story</i> do Pivotal Tracker	29
Figura 8 – Diagrama de casos de uso levantados	37
Figura 9 – Diagrama entidade relacionamento do banco de dados	38

Lista de códigos

Código 1 – Configuração do Docker Compose	40
Código 2 – Configuração do Nginx	42
Código 3 – Model de conteúdo do SailsJS	43
Código 4 – Código do controller de login	45
Código 5 – Código do service de autenticação	47

Lista de imagens

Imagem 1 – Tela de login da aplicação	48
Imagem 2 – Tela de recordes da aplicação	48
Imagem 3 – Tela de ranking do usuário	49
Imagem 4 – Tela para acesso aos conteúdos da aplicação	50
Imagem 5 – Tela para responder o questionário	50
Imagem 6 – Tela do professor no painel administrativo	51
Imagem 7 – Tela do diretor no painel administrativo	51
Imagem 8 – Tela do administrador no painel administrativo	52
Imagem 9 – Tela de criação/edição de conteúdo	52
Imagem 10 – Tela de criação/edição de questionário	53
Imagem 11 – Tela de criação/edição de pontuação dos alunos	53
Imagem 12 – Tela de visualização do conteúdo	54

Lista de tabelas

Tabela 1 – Casos de uso levantados	35
--	----

Lista de abreviaturas e siglas

API	Application Programming Interface
BSON	Binary JSON
COVID-19	Corona Vírus
DVCS	Distributed Version Control System
ER	Entidade Relacionamento (entity-relationship)
HTML	HyperText Markup Language
HTTP	Hyper Text Transport Protocol
IBM	International Business Machines
ID	Identificação Digital
IP	Internet Protocol
JSON	JavaScript Object Notation
MVC	Modew-View-Controller
ORM	Object-relational mapping
REST	Representational State Transfer
RH	Recursos Humanos
SGBD	Sistema de Gerenciamento de Banco de Dados
SO	Sistema Operacional
SQL	Structured Query Language
UC	User Case – Caso de Uso.
VCS	Version Control System
XML	EXtensible Markup Language
YAML	Yet Another Markup Language

Sumário

1	INTRODUÇÃO	16
1.1	Objetivos	17
1.1.1	Objetivo Geral	17
1.1.2	Objetivos Específicos	18
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	Engenharia de software	19
2.1.1	Engenharia de requisitos	19
2.2	Gamificação	20
2.3	Arquitetura de software	21
2.3.1	Arquitetura MVC	21
2.3.2	<i>Framework</i>	22
2.4	Banco de dados	22
2.4.1	Banco de dados não relacional	23
2.4.2	Projeto de banco de dados	24
2.5	Mapeamento Objeto-Relacional	25
2.6	Sistema de controle de versão	26
2.7	Gitflow	26
3	FERRAMENTAS	28
3.1	Git	28
3.1.1	Gitflow	28
3.2	Pivotal Tracker	28
3.3	Docker	30
3.3.1	Docker Compose	30
3.4	Nginx	30
3.5	MongoDB	31
3.6	Angular	31
3.7	SailsJS	31
4	DESENVOLVIMENTO	33
4.1	Levantamento de requisitos	33
4.2	Análise e modelagem dos requisitos	34
4.3	Construção da representação do banco de dados	37
4.4	Processo de desenvolvimento	38
4.5	Configuração da infraestrutura da API	38
4.6	Desenvolvimento da API	43

4.7	Desenvolvimento do front-end	46
5	CONCLUSÃO	55
6	TRABALHOS FUTUROS	56
	Referências	57

1 INTRODUÇÃO

A educação é um dos componentes fundamentais no desenvolvimento de uma sociedade justa e igualitária e a formação de crianças e jovens é pilar fundamental neste contexto.

O mundo enfrenta atualmente uma crise sem precedentes dada a pandemia de COVID-19 em escala global, com enormes prejuízos para a educação uma vez que o agente patógeno causador da pandemia se dissemina através do contato físico e por meio de superfícies, exigindo distanciamento e aumento de atividades online (WHO, 2022). E, para completar Silva, Pereira, Oliveira, Surdi e Araújo (2020),

Com a pandemia, a volta à normalidade não apresenta soluções fáceis, a vida social, educacional e econômica estão extremamente afetadas. O mundo presencia uma nova forma de comportamento social. Com a Pedagogia Pandêmica, as formas de se relacionar, se movimentar, de consumir, as estratégias de trabalhos e, sobretudo, o trabalho docente foram impactados.

Nos últimos anos, a ampliação do uso de ferramentas tecnológicas que permitem interação não presencial tem sido a solução encontrada para o suporte a estudos e atividades pedagógicas através do ensino remoto (SILVA, 2020). Devido ao contexto da pandemia global essas ferramentas ganharam destaque, uma vez que facilita o estudo independente do lugar e a qualquer tempo (BRITO, 2010).

Segundo Litto (2006), as questões mais importantes para um bom rendimento no ensino a distância está intimamente vinculado à qualidade e o que é atribuído como conteúdo mediado pelas metodologias de ensino e tecnologias envolvidas, incluindo as possibilidades de interações, que são propiciadas entre os grupos para que atinjam o aprendizado compartilhado. E continuando com Brito (2010), “Entretanto, apesar de sua importância, o nível de engajamento dos alunos na realização das atividades assíncronas é, ainda, muito limitado. A maioria desses ambientes não oferece atividades assíncronas, que auxiliem na interação e na realização de atividades em grupo.” Percebe-se que apesar do destaque e crescente evolução, é necessário investir a atenção devida para maximizar o impacto e efetividade da ferramenta e notamos que ainda há muito onde melhorar o ensino remoto, especialmente ao refletirmos sobre um dos grupos mais afetados pela pandemia e afastamento social: as crianças.

Observe-se uma porcentagem significativa de mudança de comportamento para pior em crianças durante a pandemia (PEDROSA; DIETZ, 2020), e continuando a cerca das mudanças notadas, segundo Pedrosa e Dietz (2020)

Na presente pesquisa, alguns comportamentos não cotidianos na vida das crianças foram relatados pelos pais/responsáveis, entre eles, a presença de agitação, desânimo e inquietação; irritabilidade; estresse; menos interesses nas atividades; foco excessivo no uso de aparelhos eletrônicos e incômodo nas atividades de vida diária.

Os efeitos do isolamento social sem dúvidas afetam toda a sociedade, mas ao considerar a capacidade cognitiva e emocional dos adultos em lidar com situações caóticas

e se adaptar a essas, é possível afirmar com toda certeza que crianças são o grupo possivelmente mais afetado. É necessário dar atenção a esses fatores comportamentais negativos das crianças, visto que algumas mudanças podem causar prejuízos a saúde (PEDROSA; DIETZ, 2020). De qualquer forma, uma maneira efetiva de enfrentar esse estresse de um novo padrão de vida é a atividade física. A atividade física diminui os níveis de estresse, além de contribuir consideravelmente nas emoções do praticante. Segundo Berbart (2018), citado por Pedrosa e Dietz (2020),

A prática de atividades físicas contribui não apenas para o pleno desenvolvimento físico, mas para absorção de competências socioemocionais dos estudantes, tais como, o senso de responsabilidade, cooperação entre colegas e familiares, autocontrole na realização de ações, capacidade de lidar com frustrações, disciplina e concentração.

Atividades físicas também auxiliam no desenvolvimento de aspectos motores, cognitivos e sociais das crianças, onde esse desenvolvimento se sobressai para crianças entre dois e sete anos de idade, relacionado a fase da aquisição dos movimentos que constituirão sua cognição motora base (PEDROSA; DIETZ, 2020).

Neste contexto, com o objetivo de combater esses problemas e aumentar o envolvimento do aluno nas aulas remotas, o objetivo desse trabalho é aplicar gamificação para construir um sistema remoto de aulas de educação física para crianças.

Ao construir uma plataforma de uso fácil, com escolha, *feedback*, desafio, interatividade, a tecnologia facilita o envolvimento dos alunos e proporciona a estes oportunidades de engajamento que permite a construção do pensamento crítico, a tornar decisões e resolver problemas em atividades assíncronas (BRITO, 2010).

A introdução da gamificação no ensino, é uma forma de aumentar a participação, a lealdade e o engajamento de usuários, amplamente utilizada em sites como Foursquare e Stack Overflow. Essa técnica vem sendo usada amplamente em contextos educacionais nos últimos anos (BORGES, 2014). Definida como a utilização de elementos de jogos fora do contexto de jogos, a gamificação pode ser utilizada para aprimorar a motivação e o engajamento, assim como servir de apoio para processos de ensino e treinamento (Kapp (Jan. 2012); Deterding, Dixon, Khaled e Nacke (Set. 2011)). Diversos trabalhos discutem os benefícios relacionados ao uso da gamificação em ambientes educacionais, dentre eles os mais citados são a melhoria no engajamento, no processo de aprendizagem ou maestria de habilidades e, também, mudanças positivas de comportamento (BORGES, 2014).

1.1 Objetivos

1.1.1 Objetivo Geral

Criar uma aplicação web de uso facilitado para crianças, professores, e diretores no contexto de ensino a distância de educação física utilizando técnicas de gamificação para prender a atenção dos usuários.

1.1.2 Objetivos Específicos

O presente trabalho tem como objetivos específicos:

- Construir a infraestrutura necessária para o desenvolvimento de um aplicativo web em Angular e uma API Sails utilizando Docker;
- Modelar banco de dados necessário para suprir os requisitos do aplicativo;
- Desenvolver API para a comunicação do aplicativo com o banco de dados;
- Utilizar técnicas e ferramentas modernas de desenvolvimento de *software* que complementem e aprofundam o conhecimento adquirido no curso, usando como referência um ambiente profissional;
- Manter uma interatividade simples e acessível para todos os usuários;
- Providenciar uma plataforma onde os alunos podem ler materiais e responder questionários, controlados pelos professores e administradores, consequentemente desenvolvendo cognição voltado a resolução de problemas;
- Instigar o conhecimento e realização de atividades físicas;

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção é destinada a relatar sobre conceitos da Ciência da Computação e como eles se relacionam com ferramentas usadas no projeto, além de abordar alguns conceitos relacionados a gamificação e educação.

2.1 Engenharia de software

Segundo Pressman e Maxim (2016), a Engenharia de Software abrange um processo, um conjunto de métodos (práticas) e ferramentas que possibilitam aos profissionais desenvolverem *software* de altíssima qualidade, conforme demonstrado na figura 1.

Figura 1 – Camadas da engenharia de software



Fonte: Pressman e Maxim (2016, pág 16)

A qualidade do produto é crucial para o sucesso de um software, e vários estudos concluem que ela está intrinsecamente ligada ao atendimento à requisitos do projeto. A camada de processo constitui o pilar para o controle e gerenciamento do trabalho de desenvolvimento. A camada de métodos oferece subsídios técnicos para as tarefas desenvolvidas durante o ciclo de desenvolvimento, e por último na camada mais alta existe as ferramentas fornecendo automação das atividades de forma organizada e repetível. Para Pressman e Maxim (2016), esses elementos são fundamentais para o sucesso do projeto de software.

2.1.1 Engenharia de requisitos

Engenharia de requisitos é o processo na engenharia de software que se inicia durante a comunicação e continua na modelagem. Segundo Pressman e Maxim (2016), o processo de engenharia de requisitos visa garantir que o sistema gerado por esses requisitos atenda adequadamente às necessidades e satisfaça as expectativas dos clientes:

A engenharia de requisitos fornece um mecanismo adequado para entender o que o cliente deseja, analisar as necessidades, avaliar a exequibilidade, negociar uma solução razoável, especificar a solução de maneira não ambígua, validar a especificação e administrar os requisitos à medida que eles são transformados num sistema em operação.

Dessa forma, através da engenharia de requisitos, é esperado que o sistema atenda ao propósito inicial e satisfaça os objetivos esperados pelo cliente. Quanto mais bem feito for essa parte do processo de desenvolvimento, maiores as chances de se obter um produto de qualidade, reduzindo re-trabalho, e apoiando um desenvolvimento linear e objetiva.

2.2 Gamificação

Aliando recursos comuns aos jogos com objetivos e aplicativos sociais com consequências reais, a gamificação é uma tendência que cada vez mais vem ganhando espaço, especialmente no contexto da pandemia global. De uso universal, pode ser aplicada em várias áreas transformando atividades que poderiam ser maçantes em experiências agradáveis.

Essa técnica proporciona um ambiente imersivo e interativo para motivar as pessoas a realizarem determinadas ações. Da mesma forma que os jogos estimulam os jogadores a acumularem pontos, essa metodologia incorpora aspectos lúdicos para que o usuário cumpra as missões proporcionando uma experiência prazerosa e desafiadora (BUSARELLO, 2014), sempre ligado a um objetivo - como por exemplo, assimilar determinados conteúdos e obter conhecimento em áreas específicas.

Isso resulta em uma aproximação e engajamento das pessoas na dinâmica, despertando interesse na atividade e retendo sua atenção nas tarefas propostas. O estilo de gamificação em um jogo estimula a busca constante por progresso e aprimoramento, e ao traduzir isso para uma aplicação na educação, pode resultar na assimilação de mais conhecimentos. Assim, a área da educação pode aproveitar essa natureza engajadora da gamificação para tornar o processo de aprendizagem mais eficiente.

De acordo com Busarello, Ulbricht e Fadel (2014), existe vários benefícios:

[...] a utilização da gamificação na educação, além de fazer com que o processo de ensino e aprendizagem aconteça de forma lúdica, apresenta várias outras vantagens: Maior interação social e maior participação dos alunos; Ambientes de ensino mais dinâmicos Desenvolvimento de criatividade, autonomia e colaboração; Promoção do diálogo; Alunos mais engajados, curiosos e motivados; Maior absorção e retenção do conteúdo; Estimulo ao protagonismo e na resolução de problemas; Melhora de resultados e desempenho.; Desenvolvimento de competências socioemocionais.

Para conseguirmos alcançar essas vantagens listadas, a gamificação voltada a educação faz uso de características como competição, feedback instantâneos, evolução e premiações, usados para instigar a motivação e o desenvolvimento cognitivo dos alunos. (VIANA, 2020). Klock, Carvalho, Rosa e Gasparini (2014), analisa alguns elementos na figura 2.

Figura 2 – Alguns elementos da gamificação

Pontuação	Sistema quantitativo de pontuação conforme tarefas que o usuário realiza
Níveis	Refere-se ao acompanhamento do progresso do usuário no sistema.
Ranking	Criação da competição por meio da visualização do progresso de outros usuários. Tem o propósito de motivar o usuário.
Conquistas	Elementos gráficos que o usuário recebe por realizar tarefas específicas.
Desafios	Tarefas específicas que o usuário deve realizar dentro do sistema.

Fonte: Klock et al. (2014)

É possível transformar estes elementos de uma forma que pontos são transformados em tarefas realizadas; níveis são vistos como mecanismo para monitorar o progresso dos alunos; *ranking* e conquistas são vistos como ferramentas para estimular a competição e motivação entre os participantes; e os desafios são usados para alcançar objetivos de aprendizagem. (VIANA, 2020)

Este projeto utiliza essas conclusões como base para desenvolver seus requisitos, interface e interatividade. Principalmente do fato que a pontuação pode ser baseada em um sistema de acompanhamento da evolução dos alunos, como Viana, Queiroz Neto, Silva e Wandermurem (2020) acrescenta:

Os próprios métodos avaliativos efetuados nos AVAs funcionam dessa maneira (quiz, fóruns), porém com a gamificação, o que era obrigação, se torna mais atraente para o aluno. As recompensas e a competição podem incentivar os alunos a buscar uma evolução acelerada do aprendizado

2.3 Arquitetura de software

Consiste em entender como os sistemas de software são desenhados e construídos, por meio de um conjunto das principais decisões de design de um sistema. O foco é analisar os componentes, como vão se comunicar, como são separadas e organizadas as responsabilidades de cada um. É crucial para o desenvolvimento de um software de qualidade.

2.3.1 Arquitetura MVC

Essa arquitetura é composta por três módulos: *model*, *view* e *controller*. Existe essa separação para facilitar o desenvolvimento e manutenção do software, uma vez que os módulos são isolados um dos outros. Os módulos se definem como:

- *Model* - Sua responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.
- *View* - Representa a visualização dos dados, é a parte que o usuário apenas vê, geralmente apresentando os dados do *model*.

- *Controller* - Aqui é feita a manipulação do *model* (dos dados) por meio de interações que podem vir de outras requisições ou do usuário através da *view*.

É um padrão muito utilizado, inclusive pelos *frameworks* usados neste projeto, por proporcionar facilidade e vários benefícios para o desenvolvedor como segurança, organização, e eficiência. Principalmente pelo fato de conseguir isolar cada responsabilidade em um lugar, o que promove um melhor entendimento macroscópico do que está sendo desenvolvido.

2.3.2 *Framework*

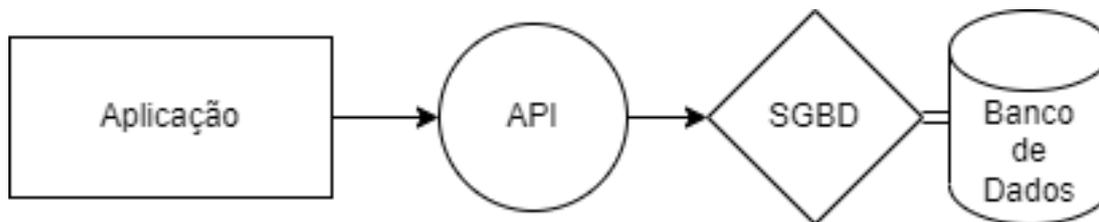
Há várias formas de definir o termo, no entanto, uma das mais fáceis de serem compreendidas é que se trata de uma coleção de classes abstratas, objetos e padrões dedicados a resolver determinados problemas em uma arquitetura flexível e extensível. O principal motivo de uso destes são a facilidade entregue à quem as utilizam. O fato do *framework* padronizar como seu trabalho será feito, além de organizar e facilitar muita coisa atrai muitos usufruintes.

A arquitetura dos *frameworks* é caracterizada por um montante de código grande e concreta, onde o usuário não tem influência, e é a base do *framework*. Em um nível acima da base, está a parte mais maleável do código, onde o usuário tem quase controle total de como é usado. Dessa forma, o *framework* dita e padroniza como você utiliza certos aspectos ou blocos, entregando funcionalidades e blocos prontos ou semi prontos para você construir em cima da forma que desejar, geralmente de uma forma muito escalável e organizada.

2.4 Banco de dados

Segundo Korth e Silberschatz (1994), um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”. Com isso, dados agrupados que tem relação e tratam do mesmo assunto pode ser considerado um banco de dados. Um exemplo disso é um sistema de controle de RH de uma empresa, uma lista telefônica, uma lista de receitas.

Neste contexto, há também um Sistema de Gerenciamento de Banco de Dados (SGBD), um software gerencial que possui ferramentas capazes de manipular as informações do banco de dados e interagir com os usuários. Alguns exemplos são: Oracle, SQL Server, PostgreSQL, MySQL, MongoDB, e Firebase. A figura 3 representa um fluxo de comunicação desde a aplicação, passando pela API, onde essa comunica com o SGBD para manipular o banco de dados.

Figura 3 – Exemplo de fluxo de uma aplicação, onde o SGBD gerencia o banco de dados

Fonte: elaborado pelo autor

Todo bom sistema de banco de dados deve apresentar um projeto, onde a organização das informações e utilização de técnicas como normalização são importantes para otimizar a *performance* e manutenções, uma vez que essas são características que frequentemente apresentam problemas em projetos de software. (RICARDO REZENDE, 2006)

2.4.1 Banco de dados não relacional

A principal aspecto que caracteriza um banco de dados não relacional é que os dados podem ser armazenados como pares de chave e valor simples como documentos JSON ou como um gráfico que consiste em bordas e vértices (MICROSOFT, 2021). Esses tipos de dados intersectam no fato de que não usam um modelo relacional, com linhas e colunas pré-definidas. O tipo de dados suportados e o modo que são consultados é normalmente mais específico em relação a bancos relacionais. Como MICROSOFT (2021) pontua,

Por exemplo, armazenamentos de dados de série temporal são otimizados para consultas em sequências de dados baseadas em tempo. No entanto, os armazenamentos de dados do grafo são otimizados para explorar relações ponderadas entre entidades.

Uma forma de descrever bancos de dados não relacionais é o termo NoSQL, que se refere a armazenamentos de dados que não usam SQL para consultas. Efetivamente, NoSQL significa “banco de dados não relacional”, mesmo que suporte consultas compatíveis com SQL. Geralmente, é diferente como esses SGBD manuseiam a execução das consultas SQL em relação a um banco de dados relacional.

Dentro desse contexto, uma das formas mais utilizadas de armazenamento é no modelo de documentos, onde é gerenciado um conjunto de campos de cadeia de caracteres nomeados e valores de dados de objeto em uma entidade, denominada documento. Como dito anteriormente, é normalmente encontrado armazenado como JSON e cada valor de campo pode ser um item escalar, como um número ou um elemento composto, como uma lista ou uma coleção de pai-filho. Os valores também podem ser XML, YAML, JSON, BSON ou até mesmo texto sem formatação. Esses campos são expostos e permite a consulta e filtro desses dados utilizando os valores nos campos (MICROSOFT, 2021).

Normalmente, os documentos contêm todos os dados da entidade e são específicos do contexto da aplicação, como por exemplo os dados de um usuário ou a lista de compras. No caso do documento, um só pode conter várias informações que seriam distribuídas em várias tabelas relacionais em um banco de dados relacional. Além disso, nesse caso o sistema não exige que todos os documentos tenham a mesma estrutura, permitindo flexibilidade ampla, uma vez que é possível armazenar dados que foram adaptados a uma nova regra de negócio, por exemplo. Na figura 4 é representado um exemplo de armazenamento de um documento.

Figura 4 – Exemplo de armazenamento de um documento de um repositório de pedidos

Key	Document
1001	<pre>{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }</pre>
1002	<pre>{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }</pre>

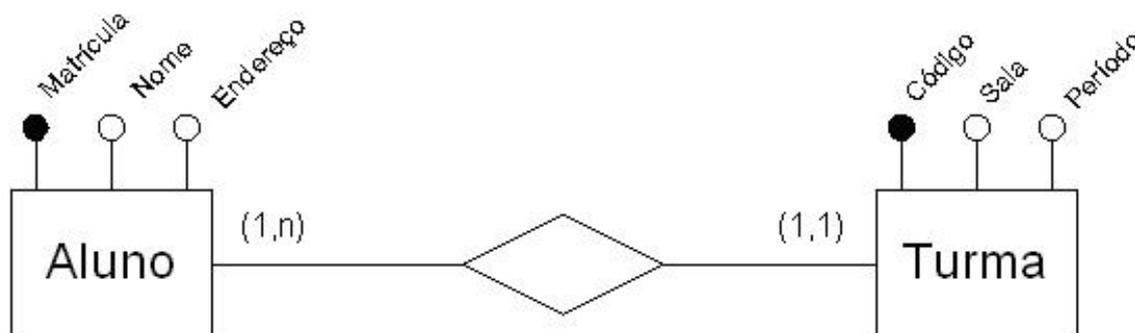
Fonte: Microsoft (2021)

2.4.2 Projeto de banco de dados

De acordo com RICARDO REZENDE (2006), "O projeto de banco de dados se dá em duas fases: Modelagem conceitual; Projeto lógico;". O Modelo Conceitual é uma representação do banco de dados sem se preocupar com qual ferramenta será realmente usado como SGBD, o que permite desenhar todo o banco de dados de uma forma universal para decisão posterior do SGBD. Para isso, é frequentemente utilizado a abordagem

entidade-relacionamento (ER), onde o modelo é representado graficamente através do diagrama entidade relacionamento (DER) (RICARDO REZENDE, 2006) como representado na figura 5.

Figura 5 – Exemplo de diagrama entidade-relacionamento



Fonte: Ricardo Rezende (2006)

Como se pode observar, é informado quais informações serão armazenadas para cada entidade. Por exemplo, para cada Aluno será armazenado sua matrícula, seu nome e endereço.

Já o modelo lógico descreve o banco de dados no nível do SGBD e é totalmente dependente de qual SGBD será utilizado, que pode ser relacional, orientado a objetos, hierárquico, etc (RICARDO REZENDE, 2006).

2.5 Mapeamento Objeto-Relacional

Essa técnica, conhecida como *Object-Relational Mapping (ORM)* é usada para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados, que pode ser relacional ou não. Ultimamente, as ferramentas que implementam o ORM tem sido muito utilizadas pela facilidade de uso e também pela camada de isolamento resultante. O banco de dados utilizado é praticamente insignificante para a maioria dos usuários, o uso da ferramenta será a mesma já que cria uma camada de separação entre o código e o banco. No caso, invés do próprio desenvolvedor fazer *queries* ou buscas no banco, ele utiliza a sintaxe do ORM no código para o fim desejado, e a própria ferramenta se encarrega de montar a *query* em SQL ou NoSQL, qualquer que seja o banco de dados suportado. Este crescimento é causado principalmente por causa do SQL, uma língua que muitos desenvolvedores não se sentem à vontade, e pela produtividade resultante do ORM (DEV MEDIA, 2011).

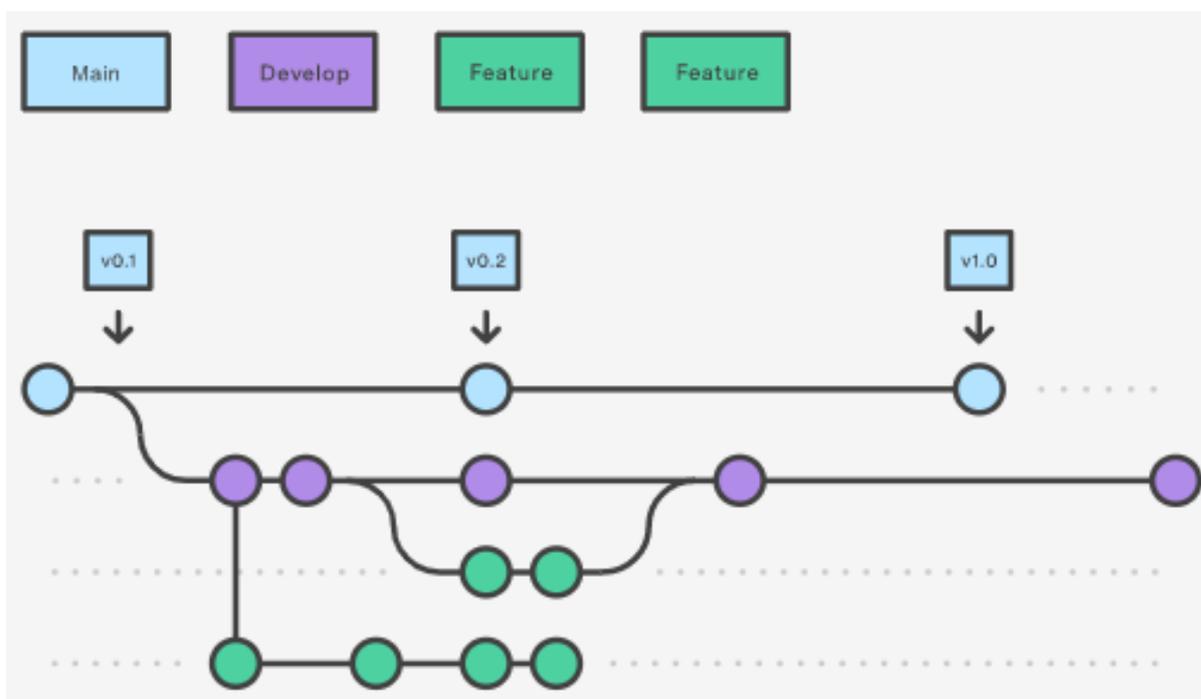
2.6 Sistema de controle de versão

Um sistema de controle de versões (VCS) é a prática de gerenciar e acompanhar mudanças no código do software. Como os ambientes de desenvolvimento tem evoluído bastante, o VCS ajuda os desenvolvedores a trabalharem mais rapidamente, uma vez que todo o trabalho fica organizado e rastreável em uma linha de tempo. Toda modificação ao código é armazenado e descrito em um tipo de banco de dados especial, para caso um erro for feito, os desenvolvedores conseguem “voltar no tempo”, comparando versões mais antigas do código com o atual para ajudar a concertar os erros enquanto minimiza os efeitos negativos para o restante da equipe (ATLASSIAN, 2022b). Além disso, permite que cada contribuinte consiga desenvolver suas próprias tarefas isoladamente e posteriormente, “comitar” as mudanças ao projeto, realmente adicionando o trabalho feito no projeto, onde os outros desenvolvedores podem atualizar o projeto local, baixando as modificações.

Dentre os tipos de VCS, há três mais usados: Local Version Control Systems (LVCS); Centralized Version Control Systems (CVCS); Distributed Version Control Systems (DVCS). As diferenças entre cada se resume basicamente em onde se concentra a fonte de verdade de todo o projeto, toda a linha de tempo das modificações introduzidas. Hoje, o tipo mais comum a ser usado é o DVCS, e a implementação mais utilizada deste é o git.

2.7 Gitflow

Gitflow é um processo de desenvolvimento que utiliza o *git*, onde é definido um fluxo de trabalho que aumenta a organização de quem está desenvolvendo, além de facilitar a manutenção, produtividade e resolução de conflitos. Este conceito é considerado melhores práticas para desenvolvimento de software moderno. Nesse processo, é definido duas *branches* principais, a *develop* e a *main*. Uma *branch* é uma ramificação do projeto em um determinado ponto, onde modificações podem ser adicionadas independente de outras, ou seja, várias pessoas podem trabalhar ao mesmo tempo num mesmo estado de projeto e posteriormente, juntar suas mudanças com uma *branch* de destino, desfazendo a ramificação (ATLASSIAN, 2022a).

Figura 6 – Exemplo de fluxo do gitflow utilizando *branches*

Fonte: Atlassian (2022)

Na figura 6, é possível observar o fluxo definido onde as *branches* de trabalho, chamadas *features*, são ramificadas a partir da *branch* que concentra todo o trabalho atual, a *develop*. Após seu desenvolvimento terminar, todas as mudanças realizadas nessa *branch* são integradas novamente para a *develop*, simbolizando o término da implementação de uma funcionalidade. Em algum momento, é necessário gerar uma versão estável do software e nesse momento a *develop* é integrada à *main* e lá é fechado uma versão estável do software, mantendo isolado o ambiente de desenvolvimento (*develop*) e de *release* (*main*) (ATLASSIAN, 2022a).

3 FERRAMENTAS

Nessa seção é relatado as ferramentas de desenvolvimento de software utilizadas na construção do sistema. Entre elas, estão ferramentas de organização de trabalho, *frameworks*, e da infraestrutura.

3.1 Git

Como dito anteriormente, o git é o DVCS mais utilizado em ambientes profissionais e foi projetado para facilitar a manutenção do código fonte em projetos com vários integrantes. Essa ferramenta possui uma variedade ampla de comandos que permitem o controle total do seu ambiente de desenvolvimento, permitindo acessar o trabalho separado dos outros integrantes, assim como ver e acompanhar o histórico das mudanças.

3.1.1 Gitflow

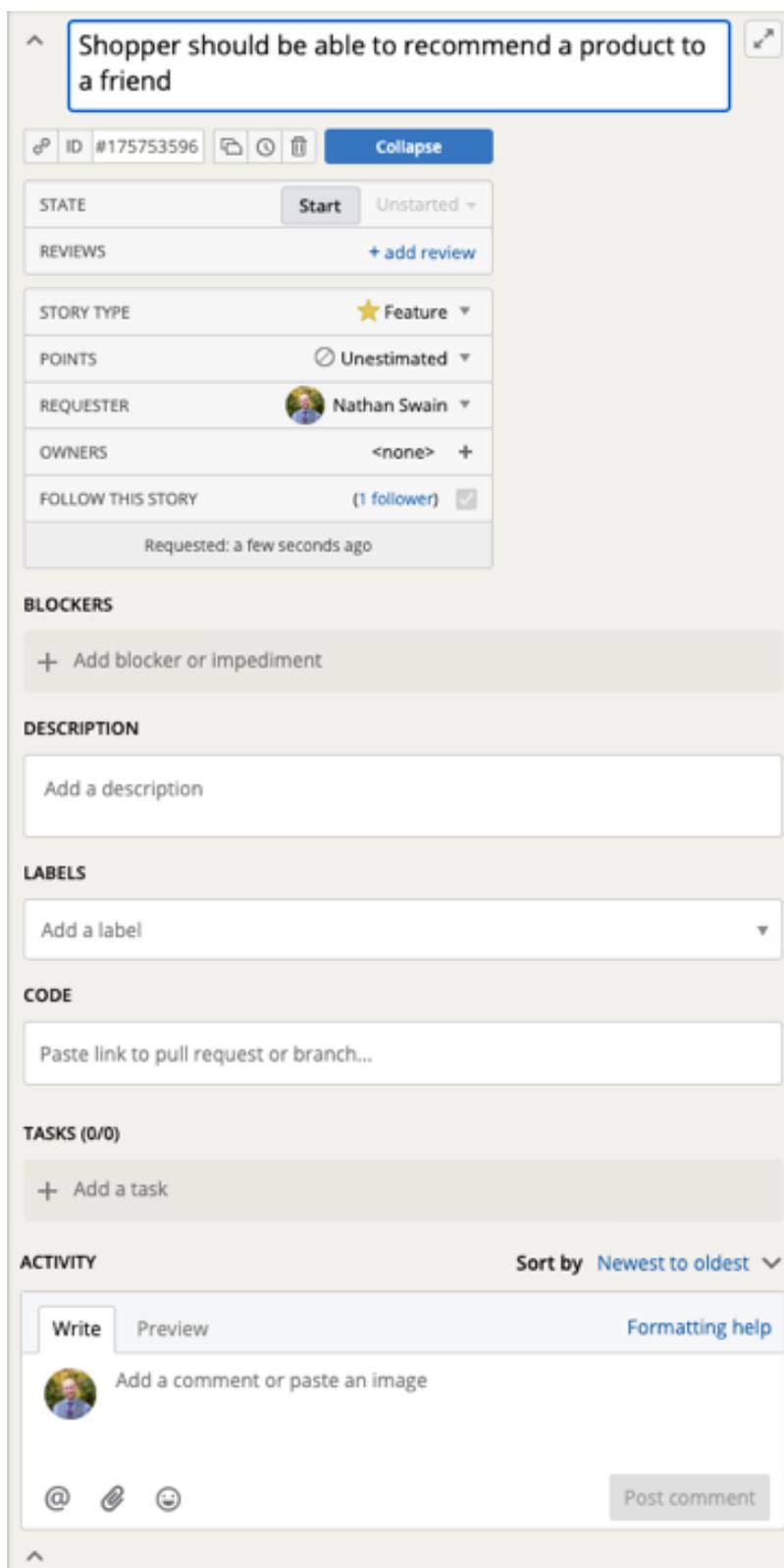
Dentro do contexto do *git*, foi utilizado o fluxo de trabalho *gitflow*, onde há duas *branches* principais, uma em que se concentra o trabalho mais atual e outra usada explicitamente para criar *releases*. Para integrar mudanças ao projeto, cada funcionalidade ou *feature* é desenvolvido “separadamente” em sua própria *branch* e posteriormente juntado à *branch develop*, efetivando a integração das mudanças ao projeto.

3.2 Pivotal Tracker

Em conjunto com o git e gitflow, o Pivotal Tracker auxilia no planejamento e organização do desenvolvimento de projetos de software, uma vez que é possível dividir e gerenciar todo o trabalho como quiser, adicionando descrições, tarefas, prioridades e discussões. Usando essa ferramenta, é possível dividir o trabalho a ser feito da forma que quiser, em vários *stories*, contendo descrição, objetivo, tarefas a ser realizadas e espaço para deixar comentários para discutir entre os envolvidos. Isso proporciona uma documentação maior sobre as tarefas, o que faz com que uma vez entregues, podem ser visualizadas e acompanhado todo o trabalho e conclusões feitas pela equipe naquela tarefa, o que traz vários benefícios ao rever alguma mudança que pode ter introduzido um *bug*.

A documentação agregada com essa ferramenta tem grande valor no desenvolvimento de software, contando com o fato de que este tende à desorganização a medida que os *deadlines* e pressão aumentam e o engajamento e produtividade dos desenvolvedores abaxem ao longo do projeto. Apesar que os benefícios são melhor aproveitados em equipes profissionais, a organização que o Pivotal Tracker entrega justifica seu uso até em trabalhos individuais, permitindo uma visualização melhor sobre todo o escopo do projeto.

Figura 7 – Uma story do Pivotal Tracker



Fonte: VMware Inc. (2022)

Na figura 7 há um exemplo de uma *story* do Pivotal Tracker evidenciando toda a customização que pode-se fazer em relação as tarefas. Ao criar uma *story*, é possível

atribuir prioridade, descrição do trabalho a ser feito, *tags*, descrever mais detalhadamente as tarefas relacionado ao trabalho feito e até escrever comentários. Além de documentar e organizar o trabalho fora do software, o Pivotal Tracker ainda te permite ter a conexão com o *software*, pelo ID do *story*. Ao iniciar a tarefa, o desenvolvedor criaria sua *branch* com o ID e tipo do *story*. Utilizando a figura 7 como exemplo, a *branch* criada pelo desenvolvedor seria a *branch* *feature/175753596*, e uma vez finalizado o trabalho, seria criado um *Merge Request* ou *Pull Request* para integrar as mudanças à *branch* principal, podendo também ser anexado ao *story* pelo campo *code*.

3.3 Docker

Essa ferramenta é destinada ao uso de desenvolvedores para facilitar o desenvolvimento e *deploy* dos projetos. O Docker utiliza de técnicas de virtualização para criar *containers*, um processo que está isolado de todos os outros processos na sua máquina, nesse contexto chamada de *host*. Os *containers* tem seu próprio sistema operacional (SO), software, executáveis e configurações e são executáveis em qualquer SO *host*, ou seja, você no *Windows* pode rodar um container que está utilizando o *Linux* e vice versa. Ao utilizar um *container*, é criado um sistema de arquivos isolados do *host* e é providenciado pela imagem do *container*. A imagem do *container* contém tudo necessário para montar e executar o *container* como configurações, softwares, *scripts*, executáveis, variáveis de ambiente e o comando padrão a ser executado ao iniciar o *container* (DOCKER INC., 2022a).

3.3.1 Docker Compose

Docker Compose é uma ferramenta para a definição e execução de aplicações Docker com vários *containers*. Com um arquivo de configuração e um só comando, é possível criar e iniciar todos os serviços necessários para seu projeto, o que reduz uma possível documentação de vários passos e páginas para iniciar o desenvolvimento no projeto para apenas algumas frases e alguns comandos (DOCKER INC., 2022b).

Com isso, é possível criar um ambiente de trabalho homogêneo para todos os desenvolvedores, o que facilita o trabalho e a integração de novos membros na equipe. Em apenas alguns comandos e configurações iniciais é possível executar tudo necessário para o funcionamento do projeto como banco de dados, servidor web, gerenciador de cache, API, e *front-end*.

3.4 Nginx

Nginx é um servidor *web* de código aberto para hospedar arquivos, serviços e pode ser usado como cache, balanceador de carga e *proxy* reversa. É um dos servidores mais

utilizado e reconhecido por empresas como o Atlassian, GitLab, Microsoft, IBM e Google, entre outros (KINSTA INC., 2022).

3.5 MongoDB

MongoDB é um SGBD não relacional moderno amplamente escalável e flexível. É uma ferramenta fácil e simples de utilizar, providenciando as capacidades requeridas para satisfazer as necessidades mais complexas em qualquer nível de escala. Os dados são armazenados em documentos JSON flexíveis, o que significa que os documentos podem variar em estrutura, armazenando campos diferentes e a estrutura de dado pode ser alterada a qualquer momento (MONGODB, INC, 2022).

O MongoDB é um banco de dados distribuído e integra nativamente alta disponibilidade, scalagem horizontal, e distribuição geográfica. Além disso, utilizando pesquisas, indexação e agregação em tempo real, providencia maneiras poderosas de analisar seus dados (MONGODB, INC, 2022).

3.6 Angular

O Angular é um *framework* moderno de desenvolvimento de aplicações web *front-end*, para serem utilizadas em navegadores, celulares e desktops. A arquitetura é **baseada** na arquitetura MVC e a linguagem utilizada é TypeScript, criada pela Microsoft com foco na experiência do desenvolvedor, onde ele completa algumas necessidades e fraquezas do Javascript. No Angular são inclusos algumas facilidades, descrito pelo GOOGLE (2022) como:

- Componentes (como tabelas, menus, botões) para criar aplicações web escaláveis
- Uma coleção de bibliotecas bem integradas que cobrem uma variedade de funcionalidades como roteamento, gerenciamento de formulários, comunicação cliente-servidor e mais
- Um conjunto de ferramentas de desenvolvimento que auxiliam a desenvolver, buildar, testar e atualizar seu código.

Esta ferramenta é muito popular principalmente pelo fato que funciona muito bem em projetos individuais e pequenos até aplicações corporativas maiores, com equipes grandes. Um dos seus diferenciais é a capacidade de escalar muito bem. (GOOGLE, 2022)

3.7 SailsJS

Essa ferramenta é um *framework* para *back-end* construída em cima da arquitetura MVC, para serem utilizadas nos servidores, onde é feito o processamento dos dados e

entregue para o *front-end* normalmente construídas com arquitetura REST. Com *frameworks* para *back-end* você consegue construir API's que processam dados e disponibiliza serviços, hospedar páginas HTML e lidar com milhares de usuários simultâneos (THE SAILS COMPANY, 2021).

Incluído no SailsJS, está uma ferramenta chamada Waterline, um ORM poderoso. Com o ORM, é possível realizar manipulações no banco de dados de uma forma unificada, independente do banco de dados utilizado, a escolha do banco de dados se torna insignificante para grande parte dos usuários. A linguagem utilizada no SailsJS é javascript, o que complementa o Angular, uma vez que as linguagens são bem similares, o trabalho do desenvolvedor de trocar de contexto é minimizada, facilitando seu desenvolvimento.

4 DESENVOLVIMENTO

Aqui será relatado todo o processo de desenvolvimento, partindo do levantamento dos requisitos até o desenvolvimento total do sistema. O processo de desenvolvimento envolve várias ferramentas e técnicas de engenharia de software para a concepção de um sistema de qualidade.

O projeto é voltado para o ensino de educação física à distância para crianças visando manter o interesse e engajamento destas no ensino e também auxiliando no desenvolvimento cognitivo e emocional. Portanto, o objetivo foi desenvolver um sistema de uso simples que consiga satisfazer essas premissas. Primeiramente houve o levantamento e modelagem dos requisitos e posteriormente a construção da representação do banco de dados e a criação de um fluxo de trabalho organizado.

4.1 Levantamento de requisitos

Ao longo do processo de fundamentação teórica deste trabalho foi colhido alguns conceitos e ideias que influenciam o levantamento de requisitos deste projeto, como as palavras da Brito (2010),

Ao proporcionar facilidade de utilização, escolha, feedback, desafio, interatividade, a tecnologia tem grande potencial para envolver os alunos. Isso também proporciona aos alunos oportunidades de engajamento que lhes permitam pensar criticamente, tomar decisões e resolver problemas em atividades assíncronas, como fórum de discussão.

Portanto, investindo o foco na satisfação dos objetivos relatados, foram levantados os seguintes requisitos:

- **Autenticação dos usuários credenciados**
- **Cadastro e acesso à conteúdos**
 - A criação e edição do conteúdo deve ser confortável para o usuário, permitindo anexar imagens e vídeos.
 - Exclusivamente os professores e diretores poderão criar e editar os conteúdos
 - Alunos poderão consumir o conteúdo pela aplicação
- **Cadastro e acesso à questionários**
 - Os questionários devem ser relacionados aos conteúdos passados pelos professores e diretores.
 - Exclusivamente os professores e diretores poderão criar e editar os questionários
 - Alunos poderão acessar e responder os questionários, acumulando pontos

- **Acompanhamento das pontuações em seus contextos específicos**
 - As pontuações serão calculadas a partir dos questionários e atividades passadas pelos professores e diretores
 - Os professores e diretores no painel administrativo terão acesso às notas das salas
 - Os alunos poderão acompanhar seu *ranking* em sua sala
- **Criação de atividades externas ao sistema e adição das pontuações**
 - A atividade, que é apenas um identificador para a pontuação será criado pelos professores e diretores com o objetivo de anexar esses à pontuação para os alunos
- **Interface simples e de fácil uso para promover o uso da aplicação**
- **Painel administrativo para os professores, diretores e administradores utilizarem as funções listadas acima**
- **Utilizar conceitos de pontos, *ranking* e conquistas da gamificação para promover o engajamento na aplicação e fortalecer o desenvolvimento do aluno**

4.2 Análise e modelagem dos requisitos

O objetivo geral dessa aplicação é apresentar uma forma melhor de utilizar o ensino à distância utilizando gamificação e simplicidade. Com isso, os requisitos relacionados aos conteúdos e questionários fazem parte da interatividade do sistema, onde os administradores específicos poderão criar conteúdos ricos através do sistema e posteriormente, criar questionários relacionados a esses conteúdos para os alunos solidificarem seu conhecimento.

Ao pensar na gamificação, como relatado na fundamentação teórica, existe três fatores principais: pontos, *ranking* e conquistas. Na aplicação os pontos gerados pelos questionários e as atividades passadas pelos professores acumularão e com isso cada aluno terá sua própria nota no sistema. Será possível acompanhar seu *ranking* em relação a sua sala e com isso poderá ganhar uma conquista no sistema, no caso visando a simplicidade, uma medalha.

O painel administrativo é importante uma vez que cada usuário terá um nível de acesso, constituído por: aluno; professor; diretor; administrador. Apenas os usuários de nível administrativo poderão acessar o painel, sendo eles: professor; diretor; administrador.

Finalmente, a simplicidade na interface tem como objetivo facilitar e manter o uso e engajamento de todos os usuários, uma vez que o engajamento no ensino remoto é

prejudicado e os professores de educação física na pandemia tiveram mais cansados e estressados devido a pressões para cumprir suas exigências (MACEDO; NEVES, 2021).

Assim, podem ser vistos na tabela 1 os casos de usos levantados dos requisitos:

Tabela 1 – Casos de uso levantados

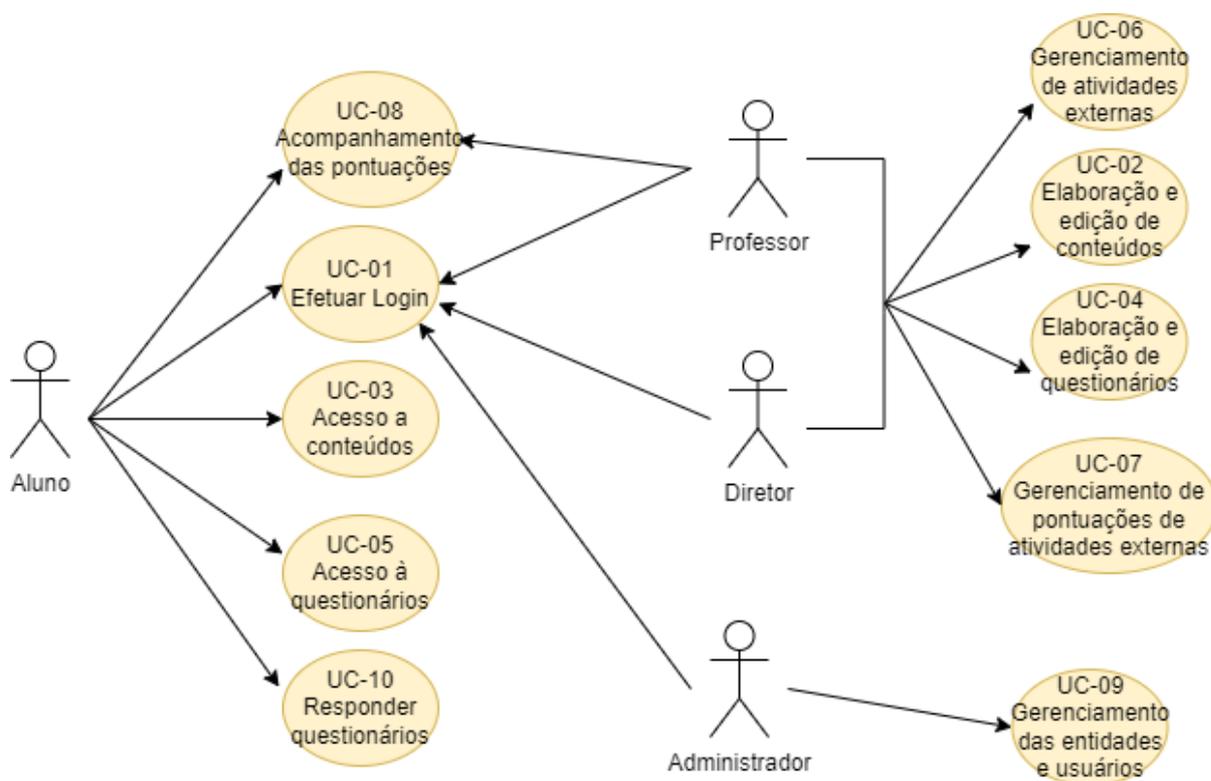
#UC	Nome UC	Descrição UC
UC-01	Efetuar Login	Autenticação de usuários (Aluno, Professor, Diretor, Administrador) cadastrados no sistema, permitindo a realização de operações nas seções específicas do sistema.
UC-02	Elaboração e edição de conteúdos	Usuários administrativos poderão escrever conteúdos e disponibilizá-los para os alunos acessarem.
UC-03	Acesso a conteúdos	Alunos poderão acessar e consumir os conteúdos cadastrados pelos administradores (professores, diretores, administradores)
UC-04	Elaboração e edição de questionários	Usuários administrativos poderão elaborar questionários, ou <i>quizes</i> relacionados ao material para a avaliação dos alunos.
UC-05	Acesso à questionários	Alunos deverão acessar os questionários relacionados aos conteúdos e responder estes.
UC-06	Gerenciamento de atividades externas	Administradores poderão adicionar atividades externas ao sistema, para adicionar pontuações para os alunos relacionadas a estas no sistema.

#UC	Nome UC	Descrição UC
UC-07	Gerenciamento de pontuações de atividades externas	Administradores deverão conseguir adicionar pontuações relacionadas as atividades externas passadas fora do sistema de forma que a pontuação do aluno na disciplina fique centralizada no sistema
UC-08	Acompanhamento das pontuações	Os usuários poderão acompanhar seu rendimento nas atividades. Os administradores terão acesso as notas dos alunos e as salas e os alunos poderão ver seu rendimento em relação a seus colegas de sala
UC-09	Gerenciamento das entidades e usuários	Alguns administradores (Diretor, Administrador) deverão cadastrar e editar colégios, professores, diretores e alunos. Somente pessoas autorizadas terão acesso ao sistema e o acesso será gerenciado pelos administradores.
UC-10	Responder questionários	Os alunos cadastrados no sistema deverão conseguir responder os questionários elaborados pelos administradores, acumulando pontuações.

Fonte: elaborado pelo autor

Utilizando esses casos de uso simplificados como referência, foi elaborado a representação da figura 8:

Figura 8 – Diagrama de casos de uso levantados

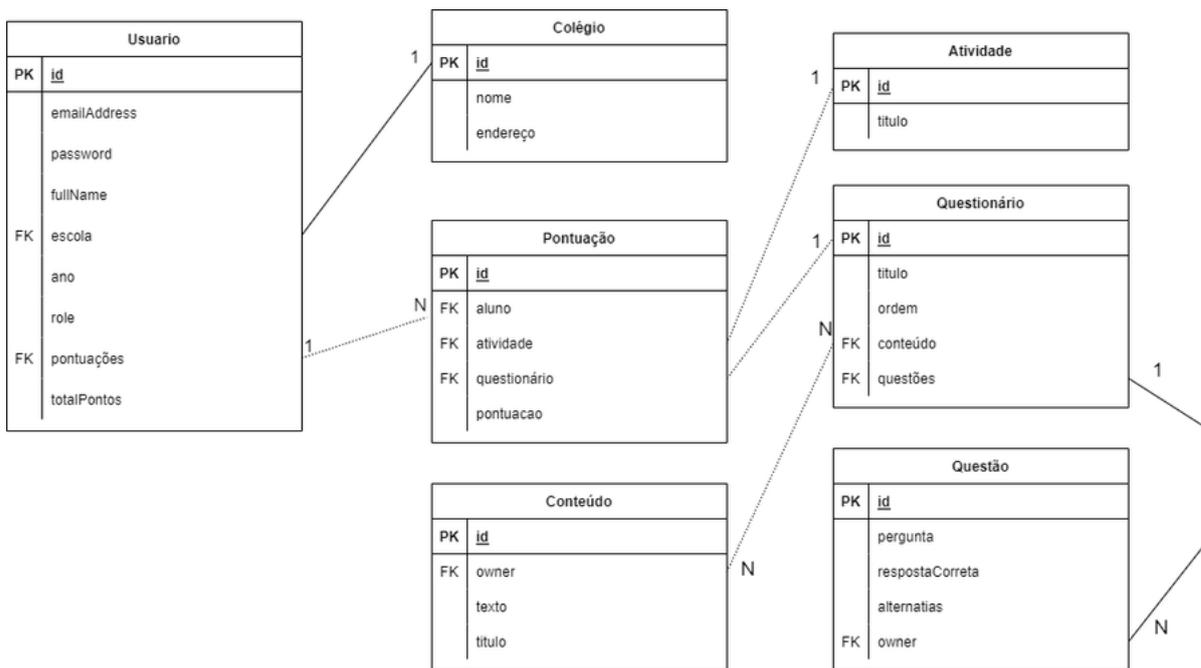


Fonte: elaborado pelo autor

4.3 Construção da representação do banco de dados

Com os requisitos definidos foi possível criar uma estrutura objetiva para o banco de dados, satisfazendo as necessidades para as funcionalidades, representado na figura 9.

Figura 9 – Diagrama entidade relacionamento do banco de dados



Fonte: elaborado pelo autor

Considerando as relações entre os objetos o melhor tipo de banco de dados para essa situação seria um banco de dados SQL, já que as chances de ter alterações na modelagem são pequenas e os dados se relacionam consideravelmente entre si. Porém, no contexto dessa aplicação, é utilizado o Waterline imbutido no SailsJS e visando uma oportunidade de explorar ferramentas com paradigmas diferentes, optou-se pelo banco de dados não relacional MongoDB, já que seu uso e configuração são simples.

4.4 Processo de desenvolvimento

O processo de desenvolvimento utilizado foi baseado no gitflow, onde existe uma *branch* principal, denominada *main* e todas as funcionalidades serão desenvolvidas em suas próprias *branches* para posteriormente ser integrado à *branch* principal. As tarefas serão quebradas anteriormente em *stories* do pivotal para facilitar o desenvolvimento, iniciando pelo desenvolvimento da API. Cada *story* do pivotal tem um ID único que é utilizado para criar as *branches* do trabalho a ser feito, assim conectando o código aos *stories* contendo informações sobre o que foi desenvolvido.

4.5 Configuração da infraestrutura da API

A infraestrutura desse projeto foca no uso da ferramenta Docker com Docker Compose pela facilidade de criar ambientes de desenvolvimento de uma forma fácil e padroni-

zada. Neste projeto, existem serviços que precisam estar ativos e comunicando entre si para funcionar, como exemplo:

- NodeJS, executando a API em SailsJS
- MongoDB, banco de dados não relacional
- Nginx, o servidor web que disponibilizará a API e o *front-end*

Dessa forma, a configuração do Docker Compose fica no arquivo `docker-compose.yml`, representado pelo código 1:

Código 1 – Configuração do Docker Compose

```
services:

  nodejs:
    container_name: ${NODE_CONTAINER_NAME}
    build:
      context: ./docker/nodejs
    ports:
      - 9229:9229
    environment:
      NODE_ENV: ${NODE_ENV}
    volumes:
      - ./:/home/node/app
    depends_on:
      - mongo
    networks:
      - backend

  mongo:
    container_name: ${MONGO_CONTAINER_NAME}
    image: mongo:5.0.1
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: admin
    ports:
      - 27017:27017
    volumes:
      - ${DATA_SAVE_PATH}/mongo:/data/db
    networks:
      - backend

  nginx:
    container_name: ${NGINX_CONTAINER_NAME}
    depends_on:
      - nodejs
    build:
      context: ./docker/nginx
    volumes:
      - ${HOST_NGINX_LOG_PATH}:/var/log/nginx
      - ${HOST_NGINX_TEMPLATES_PATH}:/etc/nginx/templates
      - ${HOST_FRONTEND_DIR}:/var/www/public
      # Para o tempo dentro dos containers ficar igual ao host
      - "/etc/timezone:/etc/timezone:ro"
      - "/etc/localtime:/etc/localtime:ro"
    environment:
      - SERVER_NAME=${SERVER_NAME}
      - NODE_CONTAINER_NAME=${NODE_CONTAINER_NAME}
    ports:
      - 80:80
      - 443:443
    networks:
      - backend
```

Onde a linguagem utilizada para descrever o ambiente é YAML. Aqui há os três serviços citados, onde o serviço do NodeJS e do Nginx usam configurações locais de Dockerfile para criar os *containers* e o serviço do MongoDB utiliza uma imagem pública disponibilizada na internet. Junto com as definições dos serviços, há definições de armazenamento, redes de comunicação entre os *containers* e variáveis que definem algumas configurações como o nome dos containers. Essas variáveis são populadas automaticamente pelo Docker através de um arquivo `.env` no projeto, frequentemente usado em ambientes profissionais para proteger as chaves e variáveis sensíveis.

Assim, ao criar o arquivo `.env`, usando o modelo no arquivo `.env.template` disponibilizado no projeto, e executando o comando do docker `compose` para iniciar os serviços, será criado todo o ambiente. Cada serviço tem suas instruções para inicialização. No caso do NodeJS, será rodado uma *script* que executa o projeto do SailsJS, conectando a API ao banco de dados. O serviço do Nginx vai executar normalmente consumindo uma configuração customizada disponibilizada no projeto que faz o mapeamento do endereços HTTP respectivos do localhost, representado no código 2:

Código2 – Configuração do Nginx

```
upstream nodejs_upstream {
    server ${NODE_CONTAINER_NAME}:1337;
    keepalive 64;
}

server {
    listen      80 default_server;
    server_name ${SERVER_NAME};

    root        /var/www/public;
    index       index.php index.htm /index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }

    location @fallback_web {
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://nodejs_upstream/;
        proxy_http_version 1.1;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_max_temp_file_size 0;
        proxy_redirect off;
        proxy_read_timeout 240s;
        proxy_cache_bypass $http_upgrade;
    }

    location = /favicon.ico { access_log off; log_not_found off; }

    error_log /var/log/nginx/${SERVER_NAME}_error.log;
    access_log /var/log/nginx/${SERVER_NAME}_access.log;
}
```

Fonte: elaborado pelo autor

No caso do desenvolvimento local, a variável SERVER_NAME seria o endereço IP

da máquina, ou apenas 127.0.0.1 conhecido como localhost. Nessa configuração é definido várias *locations* ou pontos de entrada, como o / e o /api, e ao fazer uma requisição para o endereço localhost/ ou localhost/api, a requisição será redirecionado para o recurso correto. No caso do endereço localhost/api, a requisição será redirecionada para a API e apenas localhost/ será redirecionado para o front-end, já compilado dentro do *container* do Nginx.

4.6 Desenvolvimento da API

Com a infraestrutura feita, e todos os serviços no ar, foi possível passar para o desenvolvimento da API em SailsJS. O primeiro passo foi gerar um projeto utilizando os comandos do SailsJS, o que resulta em uma estrutura de projeto bem organizado, com vários arquivos e pastas de uma forma modular. Após configurar a conexão do WaterlineORM com o MongoDB, o primeiro passo é criar os *models* que representam as tabelas no banco de dados e é por onde todas as manipulações em cima dessa entidade será feita no código.

Na pasta *models* do projeto, basta apenas criar um arquivo *javascript* e definir os nomes e tipos dos campos que sua entidade vai ter. O nome do arquivo define como será usado no código. No código 3 há o código do *model* da entidade conteúdo:

Código 3 – Model de conteúdo do SailsJS

```
/**
 * Conteudo.js
 *
 * @description :: A model definition. Represents a database
table/collection/etc.
 * @docs      :: https://sailsjs.com/docs/concepts/models-and-orm/models
 */

module.exports = {

  attributes: {
    texto: {
      type: 'string',
    },
    owner: {
      model: 'Questionario',
    },
    titulo: {
      type: 'string',
      required: true,
      unique: true,
    }
  },
};
```

Fonte: elaborado pelo autor

É possível observar que não há um campo id, essa configuração já está definida

em uma configuração global para os *models*. Aqui, é possível adicionar validações nos campos de tamanho, obrigatoriedade e definir campos que referenciam outras entidades, como é o caso do campo *owner* nessa entidade. Através desse arquivo o projeto uma vez reiniciado, através do WaterlineORM, cria a tabela e as relações necessárias caso suportado e armazena informações relacionados ao *schema* do model para posteriormente realizar as manipulações necessárias, como por exemplo, ao buscar um conteúdo, trazer também o questionário anexado.

Uma vez que os *models* das entidades foram criados, é possível introduzir dados facilmente no banco de dados para testes e desenvolvimento através da fase de *bootstrap* do SailsJS, onde é possível escrever um trecho de código para executar antes de iniciar a API. Após popular o banco de dados, o próximo passo é definir as necessidades para cada *model* e construir os *controllers*, utilizados para tratar dos dados e realizar a regra de negócio necessário em cada contexto. No código 4 é representado um exemplo de um *controller* de *login*:

Código 4 – Código do controller de login

```
module.exports = {
  friendlyName: 'Login',
  description: 'Log in using the provided email and password combination.',
  inputs: {
    emailAddress: { type: 'string', required: true },
    password: { type: 'string', required: true },
    rememberMe: { type: 'boolean' }
  },
  exits: {
    success: {
      description: 'The requesting user agent has been successfully logged
in.',
    },
    badCombo: {
      description: `The provided email and password combination does not match
any user in the database.`,
      responseType: 'unauthorized'
    }
  },
  login: async function (inputs, exits) {
    var userRecord = await Usuario.findOne({
      emailAddress: inputs.body.emailAddress.toLowerCase(),
    }).populate('escola');

    if(!userRecord) {
      throw 'badCombo';
    }

    await sails.helpers.passwords.checkPassword(inputs.body.password,
userRecord.password).intercept('incorrect', 'BadCombo').intercept('success',
'success');

    if (inputs.body.rememberMe) {
      if (this.req.isSocket) {
        sails.log.warn(
          'Received `rememberMe: true` from a virtual request, but it was
ignored\n'+
          'because a browser\'s session cookie cannot be reset over
sockets.\n'+
          'Please use a traditional HTTP request instead.'
        );
      } else {
        this.req.session.cookie.maxAge =
sails.config.custom.rememberMeCookieMaxAge;
      }
    }
    inputs.session.User = userRecord;

    return exits.json(userRecord);
  }
};
```

Fonte: elaborado pelo autor

Nesse exemplo, no campo *inputs* é definido um objeto de entrada aceito e no campo

exits os possíveis resultados desse controller que podem ser acessados de qualquer método desse *controller* com o intuito de reaproveitar respostas e entradas comuns. No caso, só tem uma função nesse arquivo, ou seja, apenas uma funcionalidade, que seria a função *login* para obviamente, autenticar o usuário.

Assim, nesse método é definido a regra de negócio para a autenticação dos usuários. Além disso, contém um exemplo de uso de um dos *models*, através do nome da entidade, *Usuario*. Neste *model* é executado o método *findOne* filtrando pelo email e populando a relação escola, que é do tipo Colégio. Portanto, ao buscar esse usuário, o campo escola virá preenchido com a entidade colégio relacionado ao usuário, graças ao WaterlineORM.

Dessa forma, o desenvolvimento da API se resume em implementar os requisitos necessários de cada entidade nos *controllers* respectivos e conectando as funções dos controllers a rotas HTTP, para serem acessadas pelo *front-end*.

4.7 Desenvolvimento do front-end

Para iniciar e facilitar o desenvolvimento foi utilizado um *template* de código aberto contendo componentes visuais prontos para uso, como botões, menus, e tabelas com estilo já configurado. Como o foco do Angular é ser modular, foi possível desenvolver tranquilamente um módulo de cada vez iniciando pelo *login* e módulo do aluno para posteriormente implementar o módulo administrativo.

Durante o desenvolvimento foi necessário implementar alguns serviços comuns aos módulos, como um serviço de notificação para dar *feedback* ao usuário em situações específicas, como por exemplo ao editar um conteúdo, ou responder um questionário. Pelo Angular ser tão modular, o uso compartilhado do mesmo serviço em módulos diferentes foi muito confortável.

Na arquitetura do Angular cada componente do *framework* tem uma utilidade e isola dentro de si um funcionamento para ser acessado externamente pelos outros componentes. Dessa forma, é possível organizar muito bem seu projeto criando serviços para autenticação, manipulação de entidades específicas e até para contextos específicos como foi o caso do serviço de notificação, que se encarrega de mostrar uma notificação na tela. Todo o código relacionado a isso se concentra dentro do serviço, e os componentes importam esse serviço e utilizam apenas passando uma configuração de como a notificação deve aparecer. Como é o caso do serviço de notificação, as requisições HTTP para o *back-end* estão todos em seus próprios serviços, como a documentação do Angular sugere. No código 5 é representado o serviço de autenticação que disponibiliza várias funções úteis para os componentes que precisarão, além do uso do pacote *HttpClient* do Angular para realizar requisições HTTP para as rotas relevantes:

Código 5 – Código do service de autenticação

```
import { BACKEND_URL } from './backend.constants';
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { LocalStorage } from '@ngx-pwa/local-storage';
import { Router } from '@angular/router';
import { concatMap, tap } from 'rxjs/operators';

@Injectable()
export class AuthService {

  constructor(
    private http: HttpClient,
    protected localStorage: LocalStorage,
    private route: Router,
  ) { }

  login(obj: any) {
    return this.http.post(`${BACKEND_URL}/login/`, {...obj}, { headers:
this.getHeaders(), withCredentials: true });
  }

  signup(obj: any) {
    return this.http.post(`${BACKEND_URL}/signup/`, {...obj}, { headers:
this.getHeaders() });
  }

  Logged(obj: any) {
    this.localStorage.setItemSubscribe('user', obj);
  }

  isLoggedIn() {
    return this.localStorage.getItem('user');
  }

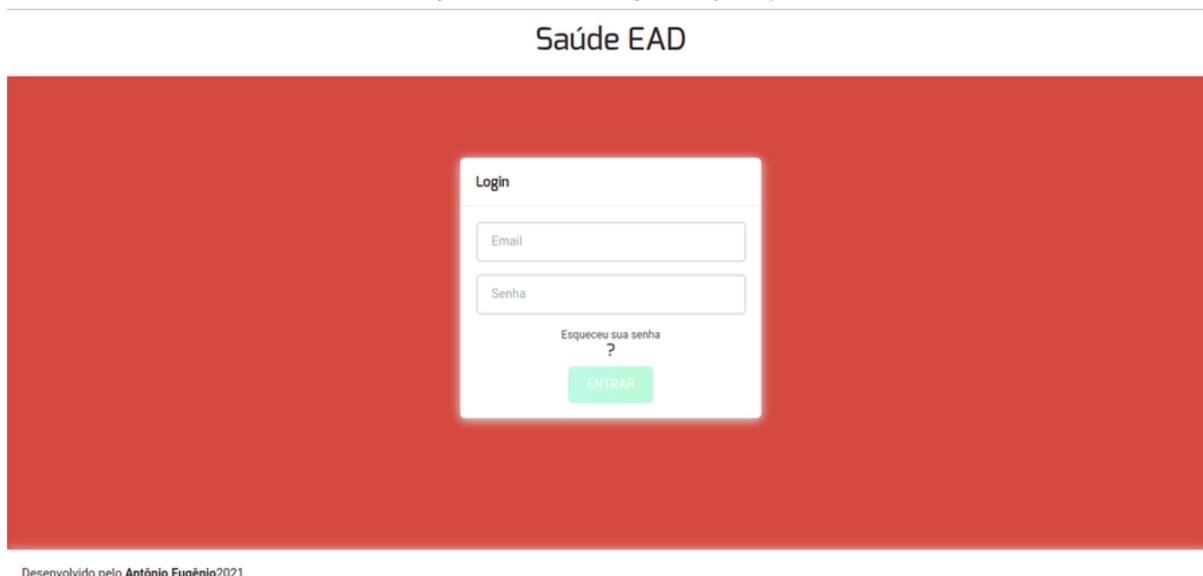
  logout() {
    this.localStorage.clear().subscribe(() => {});
    this.route.navigate(['/home/main']);
  }

  refresh() {
    return this.isLoggedIn().pipe(
      concatMap(user => {
        return this.http.get(`${BACKEND_URL}/Account/${user.id}`, { headers:
this.getHeaders() });
      }),
      tap(user => this.Logged(user)),
    );
  }

  getHeaders() {
    return new HttpHeaders({
      'Content-Type': 'application/json; charset=utf-8',
      'Accept': 'application/json',
    });
  }
}
```

Pelo sistema, nota-se uso de cores simples e botões destacados para fácil acesso e com o objetivo de simplificar a interface, além de melhorar sua interatividade, tanto no painel administrativo quanto no módulo do aluno. Na imagem 1 é possível observar uso de cores simples.

Imagem 1 – Tela de login da aplicação



Fonte: elaborado pelo autor

No módulo do aluno, existe quatro seções principais constituído por recordes, *ranking*, conteúdo, e *quiz* (questionário). Nas telas de recordes e *ranking* o aluno conseguirá acompanhar suas notas e seu colocação em relação a sua sala.

Imagem 2 – Tela de recordes da aplicação

Atividade	Pontos
1. Atividade Corrida Rasa	5p
2. Atividade Corrida Rasa Prolongada	7p
3. Atividade 3	4p
4. Atividade 4	10p
5. Atividade 5 - Sem Pontuacoes (TESTE)	9p
6. Quiz Corrida Rasa	8p
7. Quiz Corrida Prolongada	3p
Total de Pontos:	46p

Fonte: elaborado pelo autor

Imagem 3 – Tela de ranking do usuário

Ranking dos alunos do 5º ano do Colégio Getúlio Vargas	
Nome	Pontos
1. Guilherme da Silva	45p
2. Roberto da Silva	42p
3. Iodites da Silva	41p
4. Emilio da Silva	34p
5. Usuário da Silva	0p

Desenvolvido pelo Antônio Eugênio2021

Fonte: elaborado pelo autor

É possível notar a medalha da colocação do aluno em relação a sua sala nas duas telas para instigar o aspecto de conquista das técnicas citadas de gamificação. Com o intuito de entreter o aluno e acrescentar pra experiência, essa medalha tem algumas animações: a medalha mexe pra um lado e pro outro caso o aluno estiver dentro dos três melhores da sala; ao passar o mouse por cima da medalha, ela cresce de tamanho e mexe com mais intensidade.

Já nas imagem 4 o aluno conseguirá consumir o conteúdo elaborado pelo professor para responder os questionários relacionados, representado na imagem 5. Nessa tela também há uma animação ao escolher um conteúdo.

Imagem 4 – Tela para acesso aos conteúdos da aplicação



Desenvolvido pelo Antônio Eugênio2021

Fonte: elaborado pelo autor

Imagem 5 – Tela para responder o questionário



Fonte: elaborado pelo autor

Na imagem 6, ao responder um *quiz* o aluno desbloqueia a próxima, caso houver. Cada um tem um título para facilmente identificar qual conteúdo está relacionado as perguntas.

Imagem 6 – Tela do professor no painel administrativo



Fonte: elaborado pelo autor

No módulo administrativo, cada administrador tem acesso a recursos específicos de acordo com os requisitos. Durante o desenvolvimento, foi adicionado um tutorial tanto para os alunos quanto para os administradores para auxiliar o uso do sistema. Esse tutorial é configurado pelo administrador e é possível configurar separadamente o tutorial do aluno e do professor, anexando texto e vídeo.

Imagem 7 – Tela do diretor no painel administrativo

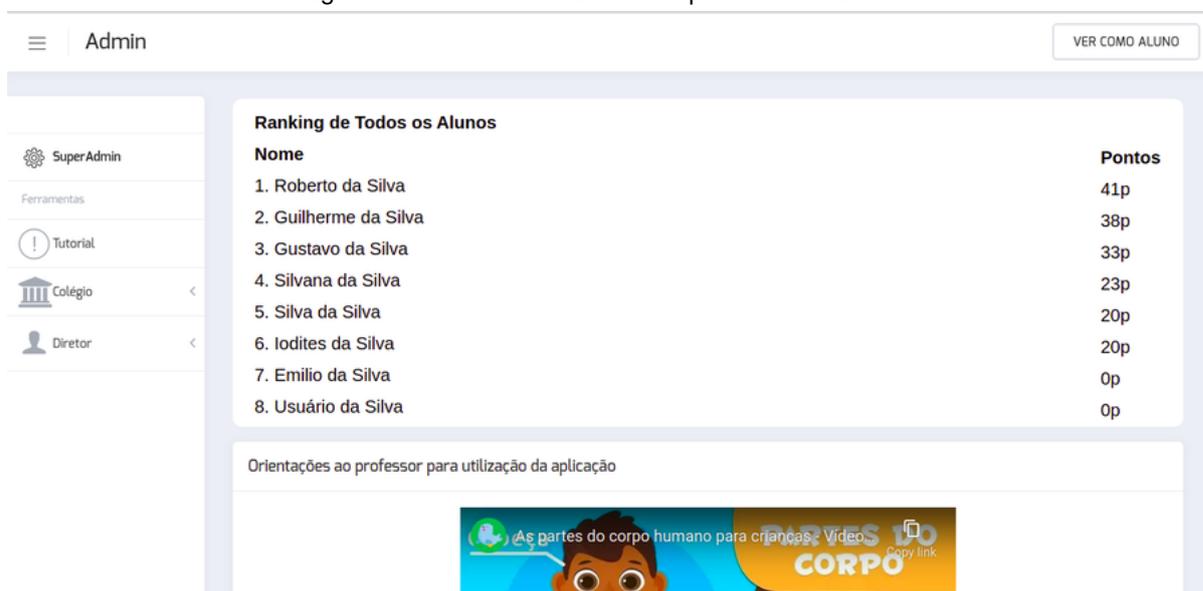


Fonte: elaborado pelo autor

Na imagem 7 representando a tela administrativa do diretor é possível observar as funcionalidades de criação e edição de professores e alunos, além de outras funcionalidades.

des para gerenciar os conteúdos, pontuações e questionários.

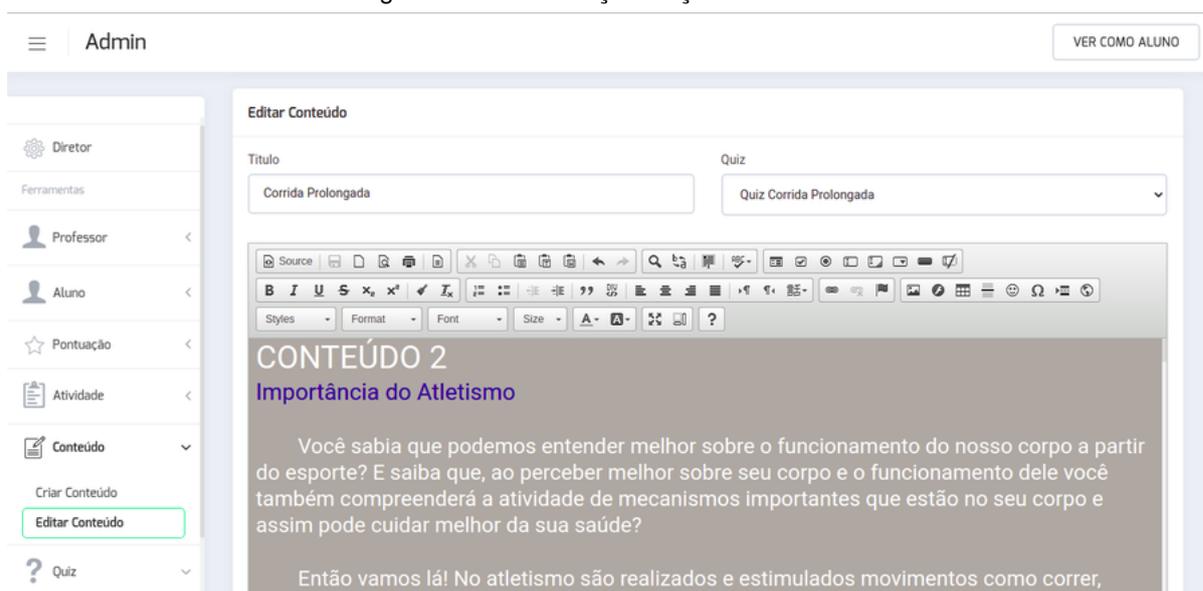
Imagem 8 – Tela do administrador no painel administrativo



Fonte: elaborado pelo autor

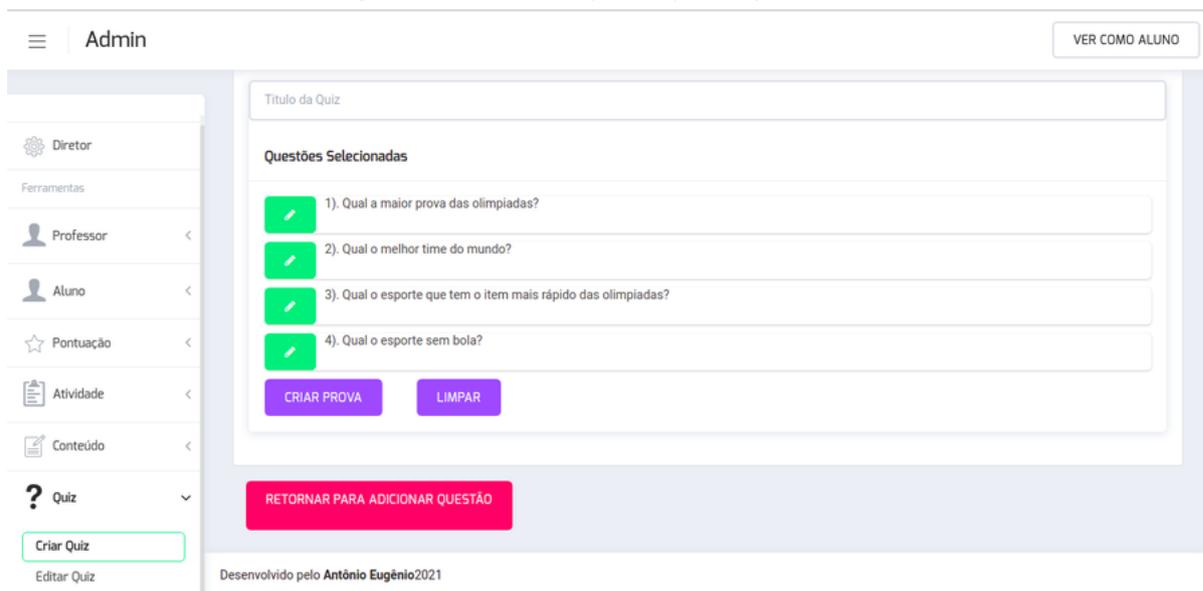
Já na imagem 8 representando a tela do administrador, é possível gerir os colégios, diretores e tutoriais do sistema. O administrador se encarrega de cadastrar os colégios e seus diretores, assim os diretores poderão cadastrar os professores, e estes podem cadastrar os alunos do colégio. Abaixo segue as imagens 9, 10, 11, e 12 que representam telas da aplicação do administrador:

Imagem 9 – Tela de criação/edição de conteúdo



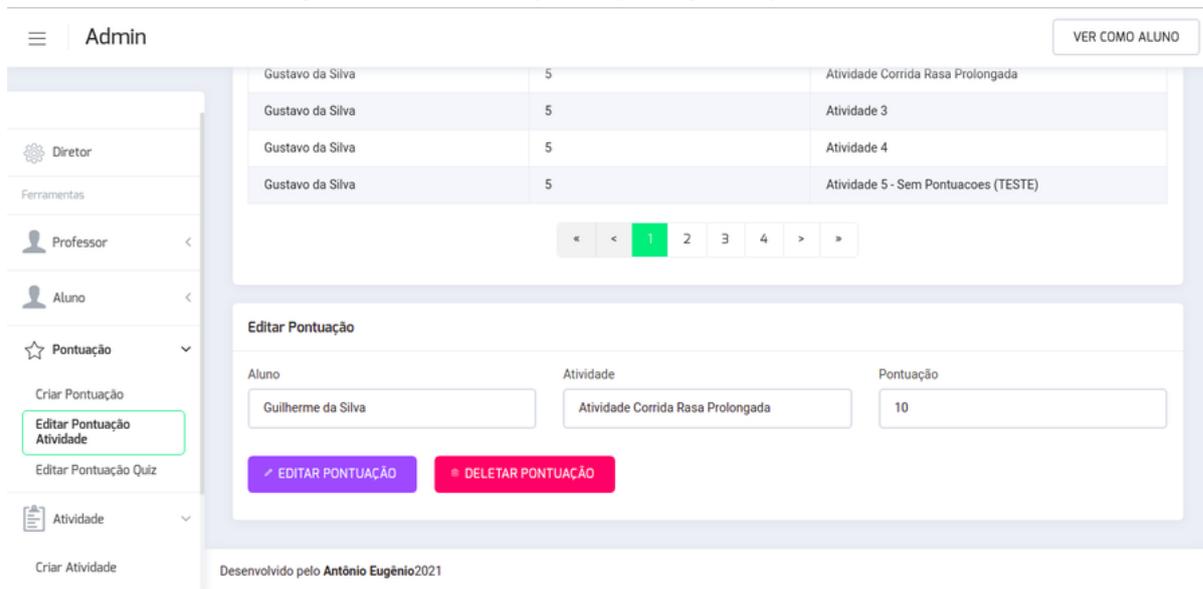
Fonte: elaborado pelo autor

Imagem 10 – Tela de criação/edição de questionário



Fonte: elaborado pelo autor

Imagem 11 – Tela de criação/edição de pontuação dos alunos



Fonte: elaborado pelo autor

Imagem 12 – Tela de visualização do conteúdo



Emílio da Silva
5º ano
Colégio Getúlio Vargas

[MEUS RECORDES](#) [RANKING](#) [CONTEÚDO](#) [QUIZ](#)

[SAIR](#)

História do Atletismo

Há muito tempo atrás, o homem pré-histórico fazia pinturas nas paredes das cavernas que demonstravam seus movimentos. Para se alimentar, o homem das cavernas precisa caçar e assim percorria longas distâncias para encontrar animais ou fugir de predadores.



Fonte: elaborado pelo autor

5 CONCLUSÃO

Com o crescente uso do ensino à distância amplificado pelo isolamento social resultante da pandemia global de COVID-19, surge novos desafios para a educação e para a vida, ainda mais quando se refere a crianças em desenvolvimento. Problemas emocionais, cognitivos e sociais afetaram todos os grupos e faixas etárias, mas o grupo mais afetado sem dúvidas são as crianças. Além disso, o ensino remoto sofre com falta de engajamento e retenção do conhecimento pelos alunos. As soluções aplicadas para tais problemas nesse projeto foram a gamificação para a falta de engajamento e o ensino e prática de educação física para as questões emocionais e cognitivas.

O desenvolvimento desse projeto foi uma oportunidade para entrar em contato não só com ferramentas e técnicas de desenvolvimento profissionais, mas também técnicas relacionados ao ensino e a educação. Foi possível notar a importância da educação física não só na saúde física, mas também mental, principalmente para crianças e a importância do engajamento no ensino. A gamificação se apresenta como uma ferramenta poderosa para a retenção da atenção e desempenho do aluno no ensino remoto, porém após considerar o conhecimento das fontes mencionados nesse trabalho, foi possível chegar a conclusão que a educação não se refere apenas à transferência de conhecimento, mas também a transferência de atenção, coletividade e humanidade, uma vez que os melhores resultados foram obtidos em cenários onde os alunos estavam engajados coletivamente com seus colegas para resolver as tarefas passadas pelo docente, mesmo que a distância. O sentimento de coletividade é um dos maiores fatores de engajamento relatados, e é indispensável que mais esforço seja investido para melhorar ambientes educacionais online, transformando tarefas em discussões abertas com os envolvidos, instigando o pensamento crítico e o desenvolvimento cognitivo.

Em relação a tecnologia e conhecimento, essa aplicação envolveu várias disciplinas e foi uma ótima oportunidade para por em prática e aprofundar o conhecimento obtido ao longo do curso de Ciência de Computação. Os desafios e problemas ao longo do processo foram importantes para a compreensão da complexidade envolvida com o processo de desenvolvimento. Além disso, o uso de ferramentas, *frameworks* e técnicas modernas como o Docker, Gitflow, Angular e NodeJS auxiliaram na contextualização de um ambiente de trabalho profissional, o que coloca em perspectiva como uma corporação desenvolveria um projeto profissional de software. A utilização de ferramentas robustas e escaláveis como o Angular, MongoDB e WaterlineORM proporcionaram novas perspectivas sobre arquitetura, escala e organização de software, por serem exemplos de arquiteturas complexas diferentes, bem projetadas e muito bem aplicadas.

Portanto, o projeto se encarrega de aliar as tecnologias modernas de desenvolvimento com técnicas modernas de ensino à distância para a melhoria de desafios atuais apresentados pela pandemia e pelo ensino à distância.

6 TRABALHOS FUTUROS

Considerando as ideias de Marriot (2008) citado por Brito (2010) sobre engajamento no ensino remoto:

Pesquisas recentes apontam que metodologias de aprendizagem colaborativa podem reduzir o sentimento de solidão. Quando um grupo de estudantes começa a colaborar e interagir em conjunto, surge um sentimento de presença, que incentiva a aprendizagem. A aprendizagem colaborativa leva os alunos a um maior nível de engajamento, aumenta sua capacidade de resolver problemas, oferece vantagens cognitivas aos alunos e tem uma influência positiva no reforço dos traços de personalidade que são benéficos para a aprendizagem autônoma ou cooperativa para a vida e para o trabalho.

Visando não só a melhora na educação mas também na saúde mental e desenvolvimento cognitivo, a próxima etapa do projeto seria adicionar um módulo de desafios colaborativos com mecanismo de comunicação e discussão entre os dicentes e docentes. Como o projeto foi feito em Angular, sua arquitetura modular permite o acréscimo de novos módulos de uma forma escalável e organizada. Além disso, o uso do MongoDB pelo fato de ser uma arquitetura não relacional permite armazenar altas quantidades de dados como no cenário de um fórum onde é armazenado muitas mensagens, vídeos, e imagens de uma forma escalável tendo o menor impacto em buscas e manipulações.

Outra forma de melhorar a aplicação seria a utilização de mais recursos da gamificação na interatividade do site e também nas tarefas realizadas em colaboração com os outros alunos, uma vez que as crianças são mais afetadas por essas técnicas.

Referências

- ATLASSIAN. **Gitflow Workflow**. 2022a. Disponível em: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=Gitflow%20is%20a%20legacy%20Git,software%20development%20and%20DevOps%20practices>. Acesso em: 07/02/2022.
- ATLASSIAN. **What is version control?** 2022b. Disponível em: <https://www.atlassian.com/git/tutorials/what-is-version-control>. Acesso em: 04/02/2022.
- BERBART, V. **Diversificar para incluir**. 2018. Disponível em: <https://www.institutounibanco.org.br>. Acesso em: 15/06/2020.
- BORGES, S. D. S.; DURELLI, V. H. S.; REIS, H. M.; ISOTANI, S. A systematic mapping on gamification applied to education. In: **ACM Symposium on Applied Computing**. [S.l.: s.n.], 2014. p. 216 – 222.
- BRITO, J. A. **Engajamento em atividades assíncronas na modalidade de ensino a distância: requisitos de interfaces colaborativas**. 2010. 130 p. Dissertação (Ciência da Computação) — Universidade Federal de Pernambuco. Disponível em: https://repositorio.ufpe.br/bitstream/123456789/2327/1/arquivo2993_1.pdf. Acesso em: 20/01/2022.
- BUSARELLO, R. I.; ULBRICHT, V. R.; FADEL, L. M. **A gamificação e a sistemática de jogo: conceitos sobre a gamificação como recurso motivacional**. [S.l.]: Pimenta Cultural, 2014.
- DETERDING, S.; DIXON, D.; KHALED, R.; NACKE, L. **From Game Design Elements to Gamefulness: Defining Gamification**. [S.l.]: Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, Set. 2011.
- DEV MEDIA. **ORM : Object Relational Mapper**. 2011. Disponível em: <https://www.devmedia.com.br/orm-object-relational-mapper/19056>. Acesso em: 12/02/2022.
- DOCKER INC. **Orientation and setup**. 2022a. Disponível em: <https://docs.docker.com/get-started/>. Acesso em: 09/02/2022.
- DOCKER INC. **Overview of Docker Compose**. 2022b. Disponível em: <https://docs.docker.com/compose/>. Acesso em: 09/02/2022.
- GOOGLE. **What is Angular?** 2022. Disponível em: <https://angular.io/guide/what-is-angular>. Acesso em: 02/02/2022.
- KAPP, K. **The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education**. San Francisco, CA: Pfeiffer, Jan. 2012. Disponível em: https://www.researchgate.net/scientific-recruitment?utm_source=researchgate&utm_medium=community-loggedout&utm_campaign=indextop. Acesso em: 28/01/22.
- KINSTA INC. **What Is Nginx? A Basic Look at What It Is and How It Works**. 2022. Disponível em: <https://kinsta.com/knowledgebase/what-is-nginx/>. Acesso em: 09/02/2022.

- KLOCK, A. C. T.; CARVALHO, M. F. de; ROSA, B. E.; GASPARINI, I. Análise das técnicas de gamificação em ambientes virtuais de aprendizagem. **RENOTE**, v. 12, n. 2, p. 1 – 10, dez. 2014. ISSN 1679-1916.
- KORTH, H. F.; SILBERSCHATZ, A. **Sistemas de Bancos de Dados**. 2a. edição revisada. ed. [S.l.]: Makron Books, 1994.
- LITTO, F. M. A Nova Ecologia do Conhecimento: Conteúdo Aberto, Aprendizagem e Desenvolvimento. **Inclusão Social**, IBICT-MCT, Brasília, v. 1, n. 2, p. 60 – 65, abr./set. 2006.
- MACEDO, L. M. M.; NEVES, L. E. de O. Práticas de Educação Física na pandemia por Covid-19. **Ensino em Perspectivas**, Fortaleza, v. 2, n. 3, p. 1 – 5, jan. 2021.
- MICROSOFT. **Dados não relacionais e NoSQL**. 2021. Disponível em: <https://docs.microsoft.com/pt-br/azure/architecture/data-guide/big-data/non-relational-data#:~:text=Um%20banco%20de%20dados%20n%C3%A3o,de%20banco%20de%20dados%20tradicionais.&text=o%20termo%20NoSQL%20refere%2Dse,n%C3%A3o%20usam%20SQL%20para%20consultas>. Acesso em: 09/02/2022.
- MONGODB, INC. **What Is MongoDB?** 2022. Disponível em: <https://www.mongodb.com/what-is-mongodb>. Acesso em: 09/02/2022.
- PEDROSA, G. F. S.; DIETZ, K. G. A prática de ensino de arte e educação física no contexto da pandemia da COVID-19. **Boletim de Conjuntura (BOCA)**, v. 6, n. 2, p. 1 – 12, Jan. 2020.
- PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software Uma Abordagem Profissional**. 8. ed. [S.l.]: AMGH, 2016. 968 p.
- RICARDO REZENDE. **Conceitos Fundamentais de Banco de Dados**. 2006. Disponível em: <https://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649>. Acesso em: 04/02/2022.
- SILVA, A. J. F. da; PEREIRA, B. K. M.; OLIVEIRA, J. A. M. de; SURDI, A. C.; ARAÚJO, A. C. de. A adesão dos alunos às atividades remotas durante a pandemia: realidades da educação física escolar. **Corpoconsciência**, v. 24, n. 2, p. 57 – 70, mai./ago. 2020. ISSN 1517-6096.
- THE SAILS COMPANY. **What is Sails?** 2021. Disponível em: <https://sailsjs.com/whats-that/>. Acesso em: 02/02/2022.
- VIANA, J. D. F.; QUEIROZ NETO, E. G.; SILVA, T. S. da; WANDERMUREM, A. V. A GAMIFICAÇÃO COMO RECURSO DIDÁTICO NO ENSINO A DISTÂNCIA. **Educação & Linguagem**, v. 7, n. 1, p. 84 – 95, Jan.-Abr. 2020. ISSN 2359-277X.
- WHO. **Coronavirus disease (COVID-19)**. 2022. Disponível em: https://www.who.int/health-topics/coronavirus#tab=tab_1. Acesso em: 18/01/2022.