

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE EDUCAÇÃO SUPERIOR DO ALTO VALE DO ITAJAÍ – CEAVI
ENGENHARIA DE SOFTWARE

AFONSO UÉSLEI BÖING

APLICATIVO COM GAMIFICAÇÃO PARA APOIO NO APRENDIZADO DA
ORIENTAÇÃO A OBJETOS

IBIRAMA

2023

AFONSO UÉSLEI BÖING

**APLICATIVO COM GAMIFICAÇÃO PARA APOIO NO APRENDIZADO DA
ORIENTAÇÃO A OBJETOS**

Trabalho de conclusão de curso apresentado como requisito parcial para obtenção do grau de bacharel em Engenharia de Software do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), da Universidade do Estado de Santa Catarina (UDESC).

Orientadora: Prof^a. Dr^a. Marília Guterres Ferreira

IBIRAMA

2023

AFONSO UÉSLEI BÖING

**APLICATIVO COM GAMIFICAÇÃO PARA APOIO NO APRENDIZADO DA
ORIENTAÇÃO A OBJETOS**

Trabalho de conclusão apresentado ao curso de Engenharia de Software do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), da universidade do estado de santa catarina (UDESC), como requisito parcial para a obtenção do grau de bacharel em Engenharia de Software.

Banca Examinadora:

Orientadora:_____

Profa. Dra. Marília Guterres Ferreira
CEAVI/UDESC

Membro 1:_____

Prof. M.Sc. Carlos Alberto Barth
CEAVI/UDESC

Membro 2:_____

Prof. M.Sc. Pedro Sidnei Zanchett
CEAVI/UDESC

IBIRAMA, 29/06/2023

AGRADECIMENTOS

Agradeço à minha orientadora, Marília Guterres Ferreira, pela orientação, ensinamentos, paciência, disponibilidade, pela confiança em meu trabalho, e principalmente pela motivação durante todo o decorrer do projeto, tornando-o realidade.

Agradeço imensamente aos meus pais/familiares, pela paciência, incentivo, apoio e compreensão durante todo o processo de realização deste trabalho. Sem vocês, eu não teria chegado até aqui.

Agradeço a todos os meus amigos, que a faculdade trouxe de forma direta ou indireta, vocês tornaram o caminho dessa fase, e desse trabalho, mais leve, ajudando a superar os momentos de estresse, ansiedade, e cobrança durante todo o percurso. Vocês fizeram e fazem a diferença todos os dias.

Por último e não menos importante, gostaria de agradecer a mim mesmo por nunca ter desistido, por ter persistido mesmo quando tudo parecia impossível, tendo confiado na minha capacidade e habilidade de superar grandes desafios. Eu reconheço e agradeço todo o meu trabalho árduo que coloquei em alcançar meus objetivos e espero continuar fazendo o mesmo no futuro, e a todos que me auxiliaram até aqui o meu 'Muito Obrigado'.

“Sorte é o resultado do esforço e da determinação de estar preparado quando a oportunidade surge (Oprah Winfrey).”

RESUMO

O paradigma de programação orientada a objetos busca trazer elementos do mundo real para a área de programação, representando um mundo povoado por objetos que se comunicam e trocam informações entre si. Entretanto, esse paradigma possui uma complexidade elevada acarretando a altos índices de reprovação e evasão nas aulas deste assunto devido ao desinteresse do aluno pelo conteúdo. Os métodos tradicionais de ensino, como palestras e exercícios escritos, podem não ser os com maior eficácia para o envolvimento do aluno com o conteúdo. A gamificação é uma abordagem com potencial a ser utilizada para motivar e melhorar o aprendizado do aluno e estimular a praticar os conceitos de programação orientada a objetos. A proposta deste trabalho é apoiar o ensino de programação orientada a objetos com o auxílio de recursos de gamificação. Para tal, será desenvolvida uma aplicação móvel que permitirá aos alunos adquirir e aplicar os seus conhecimentos de orientação a objetos. O uso de um sistema de gamificação incentiva a participação do aluno e redução das taxas de evasão e reprovação em disciplinas relacionadas a esse conhecimento. O sistema resultante foi validado com 30 participantes através do formulário *System Usability Scale (SUS)* obtendo média de 83,58 em 100, conferindo o conceito final A.

Palavras-chave: Programação orientada a objetos, gamificação, ensino de computação, aplicativo mobile.

ABSTRACT

The object-oriented programming paradigm seeks to bring elements from the real world to the programming area, representing a world populated by objects that communicate and exchange information with each other. However, this paradigm has a high complexity leading to high rates of failure and evasion in classes on this subject due to the student's lack of interest in the content. Traditional teaching methods, such as lectures and written exercises, may not be the most effective for engaging students with content. Gamification is an approach with the potential to be used to motivate and improve student learning and encourage them to practice object-oriented programming concepts. The purpose of this work is to support the teaching of object-oriented programming with the aid of gamification resources. To this end, a mobile application will be developed that will allow students to acquire and apply their object-oriented knowledge. The use of a game system encourages student participation and reduces dropout and failure rates in subjects related to this knowledge.

Keywords: Object-oriented programming, gamification, computing education, mobile application.

LISTA DE ILUSTRAÇÕES

Figura 1 - Metodologia de desenvolvimento.....	18
Figura 2 - Uso da mecânica, dinâmica, estética e de elementos do jogo na gamificação.....	21
Figura 3 - Exemplo de objeto	22
Figura 4 - Exemplo de classe.....	24
Figura 5 - Tela inicial do Duolingo	38
Figura 6 - Tela de Pergunta do Duolingo	38
Figura 7 - Tela de estudantes no Duolingo	39
Figura 8 - Tela de ranking do Duolingo	39
Figura 9 - Tela de seleção de conteúdos disponíveis no Duolingo	40
Figura 10 - Tela inicial Grasshopper	41
Figura 11 - Tela de pergunta Grasshopper	41
Figura 12 - Tela de pergunta do LearnU	42
Figura 13 - Tela de desafio diário do LearnU	43
Figura 14 - Tela de seleção de conteúdo do LearnU	43
Figura 15 - Diagrama de Casos de uso.....	49
Figura 16 - Diagrama de atividades	76
Figura 17 - Diagrama de componentes	77
Figura 18 - Diagrama de sequência do cadastro de formulário.....	78
Figura 19 - Diagrama lógico de banco de dados.....	79
Figura 20 - Sprints no Trello	80
Figura 21 - Teste service.....	81
Figura 22 - Classe service.....	82
Figura 23 - Cadastro de professor.....	83
Figura 24 - Cadastro de perguntas.....	83
Figura 25 - Questão a ser respondida.....	84
Figura 26 - Pergunta respondida corretamente.....	85
Figura 27 - Resultado final formulário	85
Figura 28 - Estrutura de pastas back-end	87
Figura 29 - Controller	88
Figura 30 - Validadores de request	89
Figura 31 - Service	90

Figura 32 - Rotas.....	91
Figura 33 - DTOs.....	92
Figura 34 - Estrutura de pastas front-end.....	92
Figura 35 - Pasta src front-end.....	93
Figura 36 - NPS, taxa de aceitação, adjetivos e nota associados ao resultado do SUS.	104

LISTA DE TABELAS

Tabela 1 - Exemplos de atributos do objeto apresentado na Figura 3	22
Tabela 2 - Exemplos de métodos do objeto apresentado na Figura 3.	23
Tabela 3 - Comparação com trabalhos correlatos.....	44
Tabela 4 - Requisitos funcionais	45
Tabela 5 - Requisitos gamificação	46
Tabela 6 - Requisitos não funcionais	46
Tabela 7 - Regras de negócio	47
Tabela 8 - Protótipos de Tela	72
Tabela 9 – Testes realizados	94
Tabela 10 - Aplicação final	95
Tabela 11 - Resultados do questionário.....	103

LISTA DE ABREVIATURAS E SIGLAS

OO	Orientação a Objetos
POO	Programação Orientada a Objetos
TCC	Trabalho de Conclusão de Curso
Udesc	Universidade do Estado de Santa Catarina

SUMÁRIO

1	INTRODUÇÃO	15
1.1	OBJETIVOS	16
1.1.1	Objetivo Geral	16
1.1.2	Objetivos Específicos	16
1.2	JUSTIFICATIVA	16
1.3	METODOLOGIA.....	17
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	APLICATIVOS DE ENSINO	19
2.2	GAMIFICAÇÃO	20
2.3	PROGRAMAÇÃO ORIENTADA A OBJETOS	21
2.3.1	Objetos	21
2.3.2	Atributos	22
2.3.3	Métodos.....	23
2.3.4	Classes.....	23
2.3.5	Abstração.....	24
2.3.6	Encapsulamento.....	24
2.3.7	Herança	25
2.3.8	Polimorfismo.....	25
2.4	DESENVOLVIMENTO ÁGIL	25
2.5	TECNOLOGIAS.....	27
2.5.1	Node.JS	27
2.5.2	Express	28
2.5.3	Prisma	29
2.5.4	Javascript.....	31
2.5.5	Typescript	32
2.5.6	PostgreSQL.....	33
2.5.7	React.....	34
2.5.8	WebApp.....	35
2.6	TRABALHOS CORRELATOS	36
2.6.1	Duolingo: Inglês e muito mais!	37
2.6.2	Grasshopper	40
2.6.3	LearnU: Aprende a programar.....	42

2.6.4	Comparação com os trabalhos correlatos	44
3	DESENVOLVIMENTO COM ENGENHARIA DE SOFTWARE	45
3.1	ENGENHARIA DE REQUISITOS	45
3.1.1	Requisitos funcionais	45
3.1.2	Requisitos da gamificação	46
3.1.3	Requisitos não funcionais	46
3.1.4	Regras de negócio	47
3.1.5	Casos de uso	48
3.1.6	Casos de uso 2.0	49
3.1.7	User Stories	55
3.1.7	Prototipação.....	71
3.2	PROJETO	76
3.2.1	Diagrama de atividades	76
3.2.2	Diagrama de componentes.....	76
3.2.3	Diagramas de sequência	77
3.2.4	Diagrama lógico de banco de dados	78
3.3	GESTÃO ÁGIL DO DESENVOLVIMENTO	79
3.3.1	Sprint 1	80
3.3.2	Sprint 2	82
3.3.3	Sprint 3	84
3.3.4	Sprint 4	86
3.4	PROGRAMAÇÃO	86
3.4.1	Back-end	86
3.4.1.1	<i>Controllers</i>	<i>87</i>
3.4.1.2	<i>Validadores de request.....</i>	<i>88</i>
3.4.1.3	<i>Services.....</i>	<i>89</i>
3.4.1.4	<i>Rotas</i>	<i>90</i>
3.4.1.5	<i>DTOs</i>	<i>91</i>
3.4.2	Front-end.....	92
3.5	TESTES	93
4	RESULTADOS E DISCUSSÕES	95
4.1	APLICAÇÃO MOBILE GAMIFICADA	95
4.2	RESULTADOS DA VALIDAÇÃO COM FORMULÁRIO SUS	103
4.3	DISCUSSÕES	104

5	CONSIDERAÇÕES FINAIS	106
5.1	TRABALHOS FUTUROS	107
6	REFERÊNCIAS.....	108

1 INTRODUÇÃO

O paradigma da programação orientada a objetos, de maneira sucinta, é a ideia de programas que simulam a vida real, ou seja, um mundo povoado por objetos, que se comunicam entre si (BARANAUSKAS, 1993). Sendo assim linguagens baseadas nos conceitos do paradigma de orientação a objetos, buscam trazer elementos do mundo real para a área da programação.

Atualmente percebe-se uma elevação na quantidade de reprovações e na evasão dos alunos no ensino superior em cursos de computação em disciplinas voltadas a Programação Orientada a Objetos (POO) (COSTA, MOREIRA, *et al.*, 2017). Alguns estudos conduzidos a planos curriculares de engenharia de software concluíram que muitos dos estudantes desistiram no primeiro ou no segundo semestre (PEREIRA, 2017). São diversos os fatores que influenciam no número das taxas de evasão, como por exemplo, a alta complexidade do POO, bem como o fato de os estudantes não acharem os seus conteúdos interessantes.

As metodologias de ensino tradicionais e rudimentares na aprendizagem da programação, como quadros negros, palestras orais, livros e exercícios escritos podem apresentar como problema os baixos níveis de participação na utilização de ferramentas disponibilizadas pelos professores, como também interesse nos materiais de referência disponibilizados (PEREIRA, 2017). Por consequência, pode-se aferir que as metodologias de ensino tradicional podem ser ineficazes, tornando o assunto desinteressante aos acadêmicos. Empenhando-se para confrontar esta tendência de opinião que prevalece entre estudantes, os professores seguem aprimorando-se e introduzindo novas metodologias de ensino, mesmo assim verifica-se que existe um grande problema ao nível da motivação e empenho demonstrado pelos estudantes (PEREIRA, 2017).

A aplicação da gamificação, poderia auxiliar num melhor aproveitamento no contato com este novo paradigma, auxiliando tanto no aprendizado dos conceitos principais, quanto nos pilares da orientação a objetos, de uma forma que o acadêmico possa praticar os conhecimentos adquiridos, não necessitando somente do desenvolvimento de sistemas para isto. Os resultados de estudos

sobre a aplicação da gamificação na educação, apontam em sua maioria, que a gamificação pode ser utilizada para motivar, aprimorar habilidades e maximizar o aprendizado dos alunos (AGUIAR, 2015).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O presente trabalho tem como principal objetivo desenvolver recursos de gamificação através de um aplicativo voltado para o ensino do paradigma de programação orientada a objetos (POO), com finalidade de auxiliar os estudantes a adquirir e aplicar conhecimentos voltados ao paradigma supracitado.

1.1.2 Objetivos Específicos

- Desenvolver de aplicativo para aprendizado de orientação a objetos.
- Executar as práticas da Engenharia de Software no processo de desenvolvimento do aplicativo.
- Incluir recursos de um sistema de gamificação.

1.2 JUSTIFICATIVA

Para cursos voltados para desenvolvimento de software, o aprendizado do paradigma da orientação a objetos é muito importante, principalmente nas fases iniciais, para nutrir o interesse e instigar o estudante a buscar conhecimento da programação orientada a objetos e, conseqüentemente buscando diminuir as taxas de reprovação e evasão, principalmente em fases iniciais (buscando reduzir as taxas de evasão e reprovação nas matérias relacionadas a estes conhecimentos). Neste trabalho, propõe-se o desenvolvimento de um aplicativo para dispositivos móveis, visto que os estudantes possuem seus celulares em sala de aula, e podem utilizá-los como

forma de aprendizado que pode ser utilizado em diversos ambientes, e traz praticidade para uso no tempo livre.

1.3 METODOLOGIA

Como proposto anteriormente, o presente trabalho objetiva utilizar práticas da Engenharia de Software durante todo o processo do desenvolvimento do software. A Engenharia de Software tem como objetivo auxiliar no desenvolvimento de software, mais que a própria programação individual, utilizando técnicas que ajudam na especificação, projeto e evolução de programas (SOMMERVILLE, 2011).

Na etapa inicial do desenvolvimento será realizada a prototipação, com a finalidade de auxiliar na escolha de tecnologias e estratégias a serem utilizadas no desenvolvimento do software. A prototipação estabelece uma versão do sistema ou constitui uma fração dele, desenvolvida de forma célere com o intuito de verificar as necessidades do cliente e a praticabilidade de certas decisões do projeto (SOMMERVILLE, 2011). Com a conclusão da etapa da prototipação, será possível adquirir uma melhor percepção da factibilidade do desenvolvimento do projeto.

A etapa seguinte é constituída pela execução do processo de análise do sistema. Nesta etapa é realizado o levantamento de requisitos, criação de casos de uso aplicando o Use Case 2.0 com suas respectivas *user stories*, visto que o Use Case 2.0 tem uma conexão direta com o desenvolvimento ágil, focando-se principalmente nos requisitos do usuário (JACOBSON; SPENCE; BITTNER, 2011). Contudo vale ressaltar que serão desenvolvidos diagramas condizentes para prover uma compreensão mais adequada sobre o funcionamento da aplicação.

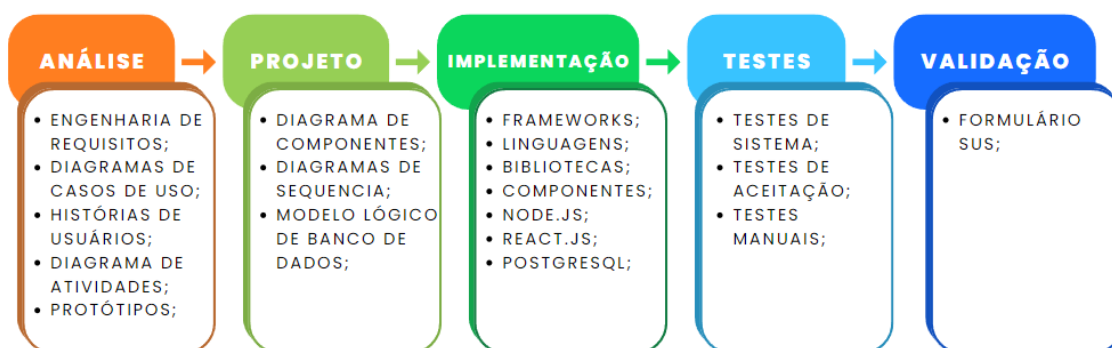
A etapa principal, constituísse pela criação do *BackLog* do sistema com foco nas *user stories* para realizar o desenvolvimento de forma iterativa e incremental, visando validar todas as funcionalidades desenvolvidas, garantindo a integridade e o correto funcionamento do sistema em sua totalidade. Assim o desenvolvimento da aplicação usa da metodologia ágil, usufruindo-se de sua

versatilidade e facilidade para adequação como às mudanças no processo de desenvolvimento de um software (FAGUNDES, 2005).

A próxima etapa, objetiva-se estruturar os casos de teste com o intuito de validar as funcionalidades do sistema, utilizando-se também de testes manuais ao decorrer de todo o projeto, assim garantindo que somente seja realizados novos desenvolvimentos, caso tudo esteja corretamente testado e validado.

Abaixo na Figura 1, temos a representação visual da metodologia de desenvolvimento que será utilizada neste projeto.

Figura 1 - Metodologia de desenvolvimento



Fonte: Adaptado de (PRESSMAN e MAXIM, 2021)

Para a etapa de validação de resultados estabeleceu-se contato com alunos e egressos de cursos da área de desenvolvimento de software, que atuam atualmente na área, onde utilizaram a aplicação como estudantes, utilizando e realizando o processo gamificado da aplicação. Após realizarem o processo, responderam ao questionário de satisfação e usabilidade SUS (System Usability Scale), que é em uma ferramenta padronizada para avaliar a usabilidade de sistemas, produtos ou serviços, consistindo em 10 itens, nos quais os usuários expressam sua concordância ou discordância em uma escala de cinco pontos, fornecendo uma avaliação quantitativa da usabilidade percebida (BROOKE, 1996), após a extração de resultados das respostas do formulário, foi realizada a análise dos dados para a validação do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo objetiva-se em explicar os conceitos e tecnologias a serem utilizados ao longo do desenvolvimento do projeto, conforme apresentado a seguir, sendo compostos por assuntos como tema, paradigma de programação, ferramentas e técnicas, tecnologias e bibliotecas utilizadas.

2.1 APLICATIVOS DE ENSINO

Atualmente, o acesso à dispositivos móveis ocorre de forma ampla em todas as partes do mundo, decorrente a isso, nota-se modificações na maneira em que o conhecimento é adquirido e compartilhado, desta forma, trazendo novos formatos para o aprendizado, baseando-se na mobilidade dos dispositivos e dos alunos, possibilitando o acesso ao conhecimento em quaisquer hora e lugar (MELO e CARVALHO, 2014). O uso de tecnologias móveis como ferramentas no processo de ensino, mostra-se eficaz na construção e desenvolvimento de conhecimentos em diversas áreas e níveis de ensino (MELO e CARVALHO, 2014).

O ambiente acadêmico necessita priorizar a busca pela concepção de espaços de aprendizagem e experimentação, utilizando tecnologias que permitam aos alunos serem livres para criar, questionar e aprender entre si. Nesse sentido, é cada vez mais frequente o uso dos aplicativos de ensino no espaço acadêmico, uma vez que despertam uma necessidade convincente para saber, perguntar, examinar, assimilar e dominar certas habilidades e áreas de conteúdo (AGUIAR, 2015).

Porém o aprendizado utilizando dispositivos móveis também enfrenta sérios desafios, pois eles, em especial os smartphones, são vistos por educadores e gestores como um fator de distração em sala de aula, porém quando utilizados de forma correta podem ajudar na obtenção de conhecimento (PEREIRA, 2017). Entre os privilégios oferecidos pelos dispositivos móveis para o ensino e a aprendizagem, pode-se destacar a difusão ao acesso a conteúdo pedagógicos, a possibilidade de desenvolvimento de comunidades de aprendizagem ativa, interativa e colaborativa. A participação em comunidades

de aprendizagem proporciona a troca de conhecimentos entre os participantes; essa interconexão entre diferentes pessoas pode potencializar a construção de conhecimento dentro e fora da sala aula (MELO e CARVALHO, 2014).

Grande parte dos aplicativos para o ensino superior funcionam como uma forma de apoio com objetivo de obter maior precisão na ação. Certamente, isso é necessário. Do ponto de vista pedagógico, o aplicativo serve como um apoio, ajudando na aprendizagem formal (MELO e CARVALHO, 2014).

2.2 GAMIFICAÇÃO

O termo “gamificação” é utilizado para descrever o processo de adaptação de elementos da mecânica, estética e dinâmica de jogos eletrônicos nas áreas de pesquisa e educação. Em uma visão geral, a gamificação trata-se da utilização de mecânicas de games para criar um espaço de aprendizagem composto de desafio, prazer e entretenimento, com objetivo de potencializar o desenvolvimento de habilidades cognitivas (FIGUEIREDO, PAZ e JUNQUEIRA, 2015).

Na área da educação não é novidade a utilização dos jogos, como os de tabuleiro, jogos de carta, palavras cruzadas, entre outros como forma de corroborar com os processos de ensino e aprendizagem dos estudantes (TAXA, REAL, *et al.*, 2018). A incorporação da utilização da gamificação na educação acaba provocando uma revisão dos processos e paradigmas educacionais, em especial, as situações que podem ser gamificadas pelos docentes universitários com o intuito de intensificar o aprendizado dos estudantes em fase de formação acadêmica e profissional (TAXA, REAL, *et al.*, 2018).

A gamificação, é utilizada na educação com o intuito de aumentar o prazer e o engajamento dos estudantes no decorrer do processo de ensino-aprendizagem, mantendo o interesse e a inspiração para continuar aprendendo (COSTA, MOREIRA, *et al.*, 2017). A utilização da gamificação, mostrou um impacto positivo no ambiente didático, contribuindo para a mudança de comportamento dos alunos, como por exemplo o aumento da presença em sala de aula, maior frequência na realização de atividades e dedicação nos estudos em casa (COSTA, MOREIRA, *et al.*, 2017).

Existem diversas técnicas para a criação da gamificação, como principais podemos citar o uso da mecânica (regras), dinâmica (comportamentos), estética (emoções) e de elementos do jogo conforme figura 2 (ZICBERMANN e CUNNINGBAM, 2011).

Figura 2 - Uso da mecânica, dinâmica, estética e de elementos do jogo na gamificação

<i>Dinâmica de jogos</i>	<i>Mecânica de jogos</i>	<i>Estética de jogos</i>	<i>Elementos de Jogos</i>
Restrições	Regras	Motivação, Competências	Habilidades, Regras
Progressão de níveis	Recompensa	Motivação	Pontos, medalhas, níveis
Relações sociais	Compartilhamento, Equipe	Engajamento, Motivação, Cooperação, Colaboração, Altruísmo	Formação de equipe, Presentes
Posição	Status, Feedback	Competição, Motivação	Ranking
Conquista	Desafios, Missões, Vitória, Aleatoriedade	Competição, Motivação	Desbloqueio de conteúdo, Desafios do jogo, Mapas
Personalização	Configurações	Engajamento	Avatar, Bens virtuais
Reforço	<i>Re-play</i>	Competências	Loops
Narrativa	Escolhas	Engajamento, Motivação	Diálogos, enredo

(ZICHERMANN; CUNNINGHAM, 2011)

2.3 PROGRAMAÇÃO ORIENTADA A OBJETOS

A Programação Orientada a Objetos é uma forma de representar com precisão em sistemas computacionais as possíveis situações que ocorrem no mundo real, compreendendo-se que o mundo é totalmente composto por diferentes objetos que interagem entre si (FARINELLI, 2007). Portanto a Programação Orientada a Objetos busca trazer elementos do mundo real para dentro do processo de desenvolvimento de um software, consistindo em considerar os sistemas computacionais como uma coleção de objetos que interagem entre si (FARINELLI, 2007).

Uma das grandes diferenças da programação orientada a objetos a outros paradigmas de programação, é o conceito de herança, o que permite facilmente estender definições existentes. Outro ponto que se deve salientar a importância é o polimorfismo, que permite escolher funcionalidades de um programa de forma dinâmica, durante a execução (RICARTE, 2001).

2.3.1 Objetos

Objeto é a expressão utilizada para representar um determinado elemento existente no mundo real, sendo uma entidade do mundo real que recebe uma representação para o ambiente de estudo. Objetos são instâncias de classes, que definem quais informações um objeto contém e como ele pode manipular elas, sendo uma entidade capaz de reter um estado, e que disponibiliza uma série de operações ou para examinar ou para afetar o estado (RICARTE, 2001). A Figura 3 apresenta um exemplo de objeto.

Figura 3 - Exemplo de objeto



Fonte: <https://br.freepik.com/> (2022)

2.3.2 Atributos

No mundo real os objetos possuem propriedades com valores. Estes valores são o que definem o estado do objeto, essas propriedades na programação orientada a objetos recebe o nome de atributo (FARINELLI, 2007).

Pode-se dizer que atributos são “variáveis” que armazenam os diferentes valores das características dos objetos, sendo estado do objeto o conjunto de valores de seus atributos em determinado instante, e o comportamento de um objeto é como ele as mudanças de estado e trocas de mensagens com outros objetos (FARINELLI, 2007). A Tabela 1 exhibe exemplos de atributos do objeto mostrado na Figura 3.

Tabela 1 - Exemplos de atributos do objeto apresentado na Figura 3

Atributo	Valor
----------	-------

Nascimento	05/10/2018
Cor do pelo	Marrom
Raça	Golden Retriever
Nome	Cacau
Temperamento	Calma
Porte	Médio

Fonte: Elaborado pelo autor (2022)

2.3.3 Métodos

Métodos são funções ou procedimentos que realizam as ações do próprio objeto, portanto, métodos são as possíveis ações que podem ser realizadas pelo próprio objeto (DURÃES, 2021). Os métodos são as formas que o objeto se manifesta, e é através deles que o objeto interage com os outros objetos (FARINELLI, 2007). A Tabela 2 exibe exemplos de atributos do objeto mostrado na Figura 3.

Tabela 2 - Exemplos de métodos do objeto apresentado na Figura 3.

Métodos
Latir ()
Deitar ()
Sentar ()
Comer ()
Dormir ()

Fonte: Elaborado pelo autor (2022)

2.3.4 Classes

Classes retratam um conjunto de objetos que possuem as mesmas características e comportamentos, portando o objeto é uma instância de uma classe, sendo assim criamos nossos objetos baseados nas características destas classes (FARINELLI, 2007).

O enfoque da programação orientada a objetos é dado na criação das classes, e não necessariamente dos objetos, como o próprio nome sugere

(FARINELLI, 2007). A programação orientada a objetos considera que objetos do mundo real compartilham de características em comum e podem ser agrupados de acordo com elas, uma classe é um gabarito para muitos objetos, descrevendo como estes estão estruturados (RICARTE, 2001). Na Figura 4 podemos ver um exemplo de classe.

Figura 4 - Exemplo de classe



Fonte: <https://br.freepik.com/> (2022)

2.3.5 Abstração

A palavra abstração, no dicionário de língua portuguesa, define-se como um ato de separar mentalmente um ou vários elementos de uma totalidade complexa. O recurso da abstração como forma de entender e dividir problemas complexos em problemas menores, até encontrar a solução do problema como um todo (FARINELLI, 2007).

O termo “abstração” em desenvolvimento de software, de modo resumido significa que somente deve ser representado aquilo que será utilizado. Por esse princípio da abstração, em nossos objetos somente terão características pertinentes ao problema em questão (FARINELLI, 2007).

2.3.6 Encapsulamento

O encapsulamento tem como objetivo ocultar como algo é feito, mostrando apenas o resultado esperado, sendo informações não essenciais ou relevantes ocultados (DURÃES, 2021). Sendo o princípio do encapsulamento o de qual cada componente de um programa deve agregar toda a informação relevante para a sua manipulação como uma unidade, aliando-se ao conceito de manter o mínimo necessário para a sua operação disponível, o encapsulamento acaba se tornando um dos poderosos mecanismos de programação orientada a objetos (RICARTE, 2001).

2.3.7 Herança

A herança consiste em um mecanismo de características comuns a diversas classes sejam escritas em uma classe base. A partir de uma classe base, poderá haver classes derivadas, que possuem as mesmas características (estrutura e métodos) da classe base, tendo adicionado a elas o que for definido especificamente para cada uma (RICARTE, 2001).

2.3.8 Polimorfismo

O Polimorfismo é a capacidade de uma variável se referir a si mesma em tempo de execução para objetos de diferentes classes. Também podemos defini-lo como o nome dado a capacidade de objetos diferentes de reagir à mesma mensagem. A mesma mensagem pode apresentar execuções diferentes, específicas para cada objeto. Com este recurso, os usuários podem enviar mensagens genéricas e detalhes sobre a implementação exata sobre os objetos receptores (FARINELLI, 2007).

2.4 DESENVOLVIMENTO ÁGIL

A metodologia ágil é uma abordagem utilizada em projetos de desenvolvimento de software que tem como objetivo aumentar a eficiência e a qualidade do trabalho realizado. Essa metodologia se baseia em ciclos de desenvolvimento iterativos e incrementais, permitindo que a equipe trabalhe de

forma mais colaborativa e flexível. Diferentemente das metodologias tradicionais, a metodologia ágil se concentra na entrega de valor ao cliente e na adaptação contínua ao longo do processo de desenvolvimento (COCKBURN, 2007).

Essa abordagem de desenvolvimento de software surgiu no início dos anos 2000 com a publicação do Manifesto Ágil por um grupo de especialistas em desenvolvimento de software (BECK, 2001). Segundo Cockburn (2007), a metodologia ágil se baseia em uma abordagem centrada no cliente, onde os requisitos do cliente são levados em consideração ao longo de todo o processo de desenvolvimento. Isso permite que o produto final seja mais adequado às necessidades do cliente e que o feedback do cliente seja utilizado para aprimorar o produto ao longo do tempo.

Outra característica importante da metodologia ágil é a sua abordagem iterativa e incremental, que se baseia em ciclos de desenvolvimento curtos, geralmente com duração de duas a quatro semanas, chamados de iterações (HIGHSMITH, 2002). Cada iteração resulta na entrega de um incremento funcional do produto, que é avaliado pelo cliente e utilizado para orientar o desenvolvimento das próximas iterações.

Além disso, a metodologia ágil se destaca pela sua ênfase na colaboração entre os membros da equipe (SCHWABER, 2002). Essa abordagem colaborativa permite que a equipe se adapte mais rapidamente às mudanças e que o conhecimento e as habilidades de cada membro sejam utilizados de forma mais eficiente.

A metodologia ágil apresenta diversas vantagens em relação às metodologias tradicionais de desenvolvimento de software. Beck (2001) destaca que a metodologia ágil permite que o produto final seja entregue de forma mais rápida e eficiente, o que reduz o tempo e o custo do desenvolvimento. Além disso, a abordagem iterativa e incremental permite que os problemas sejam identificados e corrigidos de forma mais rápida, reduzindo o risco de falhas.

Outra vantagem da metodologia ágil é a sua flexibilidade. Martin (2002) destaca que a metodologia ágil permite que o processo de desenvolvimento seja adaptado de acordo com as necessidades e os objetivos do projeto. Isso significa que a metodologia ágil pode ser utilizada em uma ampla variedade de projetos, desde pequenos projetos até projetos complexos e de grande escala.

2.5 TECNOLOGIAS

2.5.1 Node.JS

O Node.js é uma plataforma de desenvolvimento de aplicativos web que permite aos desenvolvedores criar aplicativos escaláveis e de alta performance usando JavaScript tanto no lado do cliente quanto no lado do servidor. Uma das suas principais vantagens é a sua arquitetura orientada a eventos, que permite que um único thread de execução manipule várias conexões simultâneas. Essa característica aumenta significativamente a eficiência do aplicativo e é amplamente reconhecida pela comunidade de desenvolvedores. De acordo com Bango (2015), o Node.js é uma plataforma incrivelmente poderosa para o desenvolvimento de aplicativos web.

O Node.js possui também um vasto ecossistema de módulos disponíveis no NPM (Node Package Manager), que permitem aos desenvolvedores instalar e gerenciar facilmente dependências, bibliotecas e módulos em seus projetos. Essa é uma vantagem importante, já que o NPM é um gerenciador de pacotes confiável e amplamente utilizado. Segundo Zakas (2016), o Node.js é uma plataforma poderosa que é capaz de muitas coisas incríveis, em grande parte devido ao vasto ecossistema de módulos disponíveis no NPM.

Outra característica importante do Node.js é a sua compatibilidade com uma variedade de bancos de dados, incluindo MongoDB, MySQL e PostgreSQL, entre outros. Essa característica permite que os desenvolvedores escolham o banco de dados que melhor atenda às necessidades do aplicativo e integrem facilmente com o Node.js. Conforme afirmado por Brown (2014), o Node.js é uma plataforma altamente flexível e escalável que pode ser facilmente integrada com vários bancos de dados, tornando-o uma escolha popular para o desenvolvimento de aplicativos web.

Além disso, o Node.js possui uma ampla comunidade de desenvolvedores ativos que contribuem para o desenvolvimento contínuo da plataforma e criam bibliotecas e módulos adicionais para uso em projetos. Maffett (2017) destaca que a comunidade Node.js é uma das comunidades de desenvolvedores mais

ativas e apaixonadas que ele já encontrou, com membros dispostos a ajudar com problemas de código e oferecer sugestões para melhorias.

Em resumo, o Node.js é uma plataforma de desenvolvimento de aplicativos web poderosa e flexível que oferece uma ampla gama de recursos e vantagens para os desenvolvedores. Sua arquitetura orientada a eventos, vasto ecossistema de módulos disponíveis no NPM, compatibilidade com vários bancos de dados e comunidade de desenvolvedores ativos tornam o Node.js uma escolha popular para o desenvolvimento de aplicativos web escaláveis e de alto desempenho.

2.5.2 Express

O Express.js é uma biblioteca desenvolvida para o Node.js que simplifica a criação de aplicativos web e APIs. Ela é minimalista e flexível, oferecendo uma ampla gama de recursos e vantagens para os desenvolvedores. De acordo com Freeman (2018), o Express.js é uma das ferramentas mais poderosas e versáteis disponíveis para o desenvolvimento de aplicativos web.

Uma das principais vantagens do Express.js é a sua capacidade de gerenciar rotas e endpoints de forma eficiente e organizada, usando middlewares para processar solicitações do cliente antes de enviá-las para a rota correspondente. Isso permite que os desenvolvedores adicionem funcionalidades, como autenticação e validação, de forma modular e escalável. Segundo Sabin-Wilson (2018) o uso de middlewares é um recurso poderoso do Express.js para adicionar funcionalidades de forma fácil de gerenciar.

O Express.js também oferece uma ampla gama de recursos e ferramentas para ajudar os desenvolvedores a criar APIs RESTful poderosas e escaláveis. Isso inclui recursos como análise de solicitação, manipulação de arquivos, armazenamento em cache, autenticação e gerenciamento de sessão. De acordo com Grigorik (2013), o Express.js é uma escolha popular para o desenvolvimento de APIs RESTful devido à sua facilidade de uso e recursos poderosos.

Outra vantagem do Express.js é a sua compatibilidade com uma variedade de bancos de dados, como MongoDB, MySQL e PostgreSQL. Isso

permite que os desenvolvedores escolham o banco de dados que melhor atenda às necessidades do aplicativo e integrem facilmente com o Express.js. Como destacado por Brown B. (2014), o Express.js é altamente flexível e escalável, e pode ser facilmente integrado com vários bancos de dados.

Por fim, o Express.js é apoiado por uma comunidade de desenvolvedores ativa e engajada que criam bibliotecas e módulos adicionais para uso em projetos. Essas bibliotecas e módulos podem ajudar os desenvolvedores a adicionar recursos adicionais ao seu aplicativo e resolver problemas comuns. Conforme afirmado por Freeman (2018), a comunidade do Express.js é uma das mais ativas e apaixonadas da comunidade Node.js.

Em resumo, o Express.js é uma biblioteca poderosa e flexível que oferece uma ampla gama de recursos e vantagens para os desenvolvedores de aplicativos web e APIs. Sua capacidade de gerenciar rotas e endpoints de forma eficiente, recursos para criar APIs RESTful poderosas, compatibilidade com bancos de dados e comunidade ativa de desenvolvedores são algumas das vantagens que tornam o Express.js uma escolha popular para o desenvolvimento de aplicativos web.

2.5.3 Prisma

O Prisma.js é uma poderosa ferramenta de acesso a banco de dados para desenvolvedores que buscam uma forma moderna e eficiente de interagir com seus bancos de dados em aplicações web. Desenvolvido pela Prisma Labs, Prisma.js é uma biblioteca de código aberto que fornece uma camada de abstração para gerar consultas e mutações de banco de dados em aplicativos Node.js, facilitando a interação com o banco de dados. banco de dados mais fácil e eficiente (PRISMA LABS, 2023)

Uma das principais vantagens do Prisma.js é sua sintaxe declarativa, que permite aos desenvolvedores descrever a estrutura de um banco de dados usando uma linguagem de modelagem visual. Com isso, os modelos de dados podem ser definidos de forma clara e simples, facilitando a manutenção e evolução do esquema do banco de dados ao longo do tempo (PRISMA LABS, 2023).

Outra vantagem do Prisma.js é sua compatibilidade com vários bancos de dados populares, como MySQL, PostgreSQL e SQLite. Isso permite aos desenvolvedores escolher o banco de dados mais adequado para suas necessidades e utilizar o Prisma.js como uma camada de abstração para se comunicar com o banco de dados escolhido, o que proporciona maior flexibilidade na escolha de tecnologias de banco de dados (PRISMA LABS, 2023).

Além disso, o Prisma.js oferece recursos avançados para otimização de desempenho, como a geração de consultas SQL eficientes e a capacidade de criar consultas aninhadas e carregar dados relacionados de forma eficaz. Isso resulta em consultas de banco de dados mais rápidas e eficientes, tornando o desempenho da aplicação mais otimizado (PRISMA LABS, 2023).

Outra funcionalidade interessante do Prisma.js é a geração automática de APIs GraphQL. Com o Prisma.js, é possível criar APIs GraphQL de forma rápida e fácil, aproveitando a definição de modelos de dados e suas relações no schema do Prisma. Isso permite a exposição de APIs GraphQL poderosas e flexíveis, que podem ser utilizadas para consultar e manipular os dados no banco de dados de forma eficiente (PRISMA LABS, 2023).

A comunidade em torno do Prisma.js é ativa e em constante crescimento, com uma ampla documentação e uma comunidade engajada de desenvolvedores que compartilham conhecimentos e contribuem para a evolução da ferramenta. Isso proporciona um ambiente de desenvolvimento colaborativo e suporte para resolver dúvidas e problemas que possam surgir durante o uso do Prisma.js (PRISMA LABS, 2023).

Em resumo, o Prisma.js é uma ferramenta poderosa e versátil para acesso a banco de dados em aplicações web, oferecendo uma sintaxe declarativa, compatibilidade com diversos bancos de dados, recursos avançados de otimização de desempenho e a geração automática de APIs GraphQL. Com sua crescente comunidade e documentação abrangente, o Prisma.js é uma escolha sólida para desenvolvedores que desejam uma solução moderna e eficiente para interagir com bancos de dados em aplicações Node.js (PRISMA LABS, 2023).

2.5.4 Javascript

O Javascript é uma linguagem de programação fundamental para o desenvolvimento de aplicativos web, jogos e diversos outros tipos de software. A sua importância e relevância no cenário da tecnologia da informação é incontestável, tendo em vista a sua ampla utilização por empresas como Google, Facebook, Netflix e Amazon (FLANAGAN, 2011).

Dentre as características mais relevantes do JavaScript, destaca-se a sua capacidade de ser executado em praticamente qualquer navegador web moderno, possibilitando que os desenvolvedores criem aplicativos web complexos que possam ser executados em diversos dispositivos com acesso à internet. Essa capacidade é uma das razões pelas quais o JavaScript se tornou tão popular (DUCKETT, 2014).

Além disso, o JavaScript tem a habilidade de interagir com o DOM (Modelo de Objetos do Documento), permitindo que os desenvolvedores criem páginas web dinâmicas e interativas. Essa característica é uma técnica poderosa que permite criar experiências de usuário ricas e envolventes (MCPEAK, 2015).

O JavaScript também oferece uma ampla gama de recursos e bibliotecas que ajudam os desenvolvedores a criar aplicativos web complexos e escaláveis, como o React.js e o Angular.js. Essas bibliotecas são ferramentas poderosas que podem reduzir o tempo de desenvolvimento e aumentar a qualidade do código (HOSKINS, 2019).

Outra característica relevante do JavaScript é a sua capacidade de ser usado tanto no lado do cliente quanto no lado do servidor. Com a introdução do Node.js, os desenvolvedores podem utilizar o JavaScript para criar aplicativos web do lado do servidor com facilidade. Essa capacidade pode aumentar a velocidade de desenvolvimento e simplificar a manutenção do código (WILSON, 2018).

Por fim, é importante destacar que o JavaScript é uma linguagem de programação em constante evolução, com novos recursos e funcionalidades adicionados regularmente. A introdução do ES6, por exemplo, adicionou novos recursos que simplificam o código e tornam a linguagem mais fácil de usar (FLANAGAN, 2017).

Em resumo, o JavaScript é uma linguagem de programação poderosa e versátil que apresenta diversas características e recursos importantes para o desenvolvimento de aplicativos web e mobile, jogos e outros tipos de software. A sua capacidade de ser executado em diversos dispositivos, interagir com o DOM, oferecer uma ampla gama de recursos e bibliotecas, ser usado tanto no lado do cliente quanto no lado do servidor e estar em constante evolução são algumas das características que tornam o JavaScript uma linguagem de programação relevante e importante para o cenário da tecnologia da informação.

2.5.5 Typescript

Segundo Gonçalves & Oliveira (2020), o TypeScript é uma linguagem de programação de código aberto que surgiu em 2012 como uma alternativa ao JavaScript. Ele foi criado pela Microsoft e, desde então, tem se tornado cada vez mais popular entre os desenvolvedores web. O objetivo desta seção é apresentar as características e benefícios do TypeScript, a fim de contribuir para o desenvolvimento de aplicações web mais eficientes e escaláveis.

De acordo com Scarpino (2018), o TypeScript se diferencia do JavaScript por ser uma linguagem de programação tipada, ou seja, os tipos de dados são definidos explicitamente. Isso proporciona um código mais robusto e escalável, reduzindo o tempo gasto com depuração e manutenção. Além disso, a tipagem estática ajuda os desenvolvedores a entender melhor o comportamento das variáveis e funções no código, permitindo a detecção de erros antes mesmo da compilação.

Outra característica importante do TypeScript é a sua capacidade de ser transpilado para JavaScript, como destacado por (SOUSA, 2020). Isso significa que o código escrito em TypeScript pode ser executado em qualquer ambiente que suporte JavaScript. Isso é especialmente útil para equipes que trabalham em projetos maiores e precisam garantir a compatibilidade do código em diferentes plataformas.

Segundo Marques (2019), além das características já mencionadas, o TypeScript oferece uma série de benefícios para os desenvolvedores web. Um deles é o aumento da produtividade, já que a tipagem estática ajuda a evitar

erros comuns de sintaxe e tipos de dados, permitindo que os desenvolvedores se concentrem em tarefas mais complexas.

Outro benefício é a melhoria da manutenção do código, como destacado por Oliveira (2021). A tipagem estática e a capacidade de transpilação para JavaScript tornam o código mais legível e fácil de entender, mesmo para equipes que não possuem experiência com TypeScript.

2.5.6 PostgreSQL

O PostgreSQL é um dos sistemas gerenciadores de bancos de dados (SGBD) mais populares do mundo. Ele é um software livre e de código aberto que oferece um grande número de recursos e funcionalidades, tais como suporte a transações, replicação de dados, integridade referencial, etc. Segundo Sadykov & Kantarbayeva (2020), o PostgreSQL tem sido amplamente utilizado em todo o mundo e é particularmente popular entre as empresas que buscam uma solução de banco de dados confiável e escalável.

O PostgreSQL é um SGBD robusto e altamente escalável, capaz de lidar com grandes quantidades de dados e usuários simultaneamente. Ele possui recursos avançados de segurança, tais como autenticação de usuários e criptografia de dados em repouso, o que garante a confidencialidade e integridade dos dados armazenados. Além disso, o PostgreSQL suporta diferentes tipos de dados, como texto, números, imagens, entre outros, permitindo a criação de bancos de dados complexos e com múltiplas aplicações. Conforme observado por Oliveira (2021), o PostgreSQL também oferece suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo que as operações realizadas no banco de dados sejam consistentes e confiáveis.

De acordo com Oliveira (2021), o PostgreSQL suporta o uso de índices, o que torna as consultas mais eficientes e melhora o desempenho do sistema como um todo. Além disso, o PostgreSQL tem a capacidade de lidar com grandes volumes de dados e usuários simultaneamente. Como destacado por Sadykov & Kantarbayeva (2020), essa característica é particularmente

importante para empresas que precisam gerenciar grandes quantidades de dados em tempo real.

Outra característica importante do PostgreSQL é sua capacidade de replicar dados. Com o uso de ferramentas como o *Streaming Replication*, é possível manter várias cópias do banco de dados sincronizadas em diferentes servidores, garantindo alta disponibilidade e tolerância a falhas. Segundo Nascimento & Santana (2017), a replicação de dados também pode ser utilizada para balanceamento de carga e aumento da capacidade de processamento do sistema.

2.5.7 React

React é uma biblioteca JavaScript para a construção de interfaces de usuário modernas desenvolvidas e mantidas pelo Facebook. Lançado em 2013, o React se tornou a ferramenta de desenvolvimento de aplicativos da Web mais popular e amplamente utilizada para sites e aplicativos. Muitos aplicativos populares em todo o mundo (FACEBOOK, 2023).

Um dos principais recursos do React é seu modelo de programação baseado em componentes. Os componentes do React são blocos de construção reutilizáveis que representam partes da interface do usuário. Eles podem ser combinados para formar interfaces complexas e interativas. A vantagem dos componentes é que eles são independentes uns dos outros e podem ser facilmente atualizados ou substituídos sem afetar o restante do aplicativo (REACT, 2023).

Outra característica importante do React é o conceito de "DOM virtual". O DOM virtual é uma representação na memória da estrutura da interface do usuário que o React gerencia internamente para atualizar de forma otimizada o DOM real, que é a representação visual da página da web. Ao usar o DOM virtual, o React pode minimizar as atualizações reais do DOM, resultando em melhor desempenho e eficiência na renderização de componentes (REACT, 2023).

Uma das maiores vantagens do React é sua crescente comunidade e ecossistema. Existem muitos plug-ins e bibliotecas disponíveis para o React,

como Redux para gerenciamento de estado, React Router para roteamento, Material-UI para componentes de interface do usuário prontos para uso e muito mais. Além disso, a comunidade React é ativa e engajada com uma riqueza de recursos de aprendizagem, documentação e fóruns de suporte ao desenvolvedor (REDUX, 2023).

Outro ponto forte do React é sua capacidade de trabalhar bem com outras tecnologias e estruturas. Ele pode ser facilmente integrado a outras bibliotecas ou estruturas como Angular ou Vue para criar interfaces de usuário mais complexas e ricas em recursos. Isso permite que os desenvolvedores usem o React em combinação com outras ferramentas e tecnologias existentes em seus aplicativos sem precisar reescrever todo o código (ROUTER, 2023).

Além disso, o React é altamente escalável e adequado para desenvolvimento de aplicativos em larga escala. Sua arquitetura baseada em componentes, modular e otimizada para desempenho permite que os aplicativos React manipulem grandes quantidades de dados e interações do usuário sem comprometer a velocidade ou a capacidade de resposta (MATERIAL-UI, 2023).

2.5.8 WebApp

WebApp é um termo que se refere a uma aplicação web, ou seja, um software que é acessado e executado em um navegador da web. As WebApps têm se tornado cada vez mais populares devido à sua acessibilidade e capacidade de funcionar em diferentes dispositivos, como computadores, tablets e smartphones (MOZILLA DEVELOPER NETWORK, 2023).

Uma das principais vantagens de uma WebApp é a sua capacidade de ser acessada em diferentes plataformas e sistemas operacionais, sem a necessidade de instalação de software adicional. Isso permite aos usuários acessarem a aplicação em qualquer dispositivo com um navegador da web e uma conexão à Internet, proporcionando maior flexibilidade e conveniência (MOZILLA DEVELOPER NETWORK, 2023).

Outra característica importante das WebApps é a sua capacidade de ser atualizada e aprimorada de forma centralizada. Ao contrário de aplicativos tradicionais que precisam ser atualizados em cada dispositivo individualmente,

as WebApps podem ser atualizadas no servidor, o que facilita a implementação de correções de bugs, melhorias de desempenho e novos recursos de forma mais eficiente (MOZILLA DEVELOPER NETWORK, 2023).

Além disso, as WebApps são desenvolvidas usando tecnologias web padrões, como HTML, CSS e JavaScript, o que as torna altamente compatíveis com os navegadores modernos. Isso significa que as WebApps podem ser executadas em diferentes navegadores da web sem a necessidade de ajustes específicos para cada navegador, proporcionando uma experiência consistente para os usuários (MOZILLA DEVELOPER NETWORK, 2023).

Outra vantagem das WebApps é a sua capacidade de serem facilmente compartilhadas com outros usuários. Por serem acessadas via navegador da web, as WebApps podem ser compartilhadas por meio de URLs, links ou até mesmo incorporadas em outras páginas da web, o que facilita a disseminação e compartilhamento das aplicações (MOZILLA DEVELOPER NETWORK, 2023).

Porém, é importante destacar que as WebApps também possuem algumas limitações, como a dependência de uma conexão à Internet para funcionar corretamente e a impossibilidade de acessar recursos do dispositivo, como câmera, sensores de GPS, entre outros, sem a permissão do usuário. No entanto, essas limitações são mitigadas por meio de tecnologias web modernas, como *Progressive Web Apps* (PWAs) e Web APIs, que permitem às WebApps oferecerem experiências mais ricas e avançadas aos usuários (MOZILLA DEVELOPER NETWORK, 2023).

Em resumo, as WebApps são aplicações web acessadas por meio de navegadores da web, oferecendo vantagens como acessibilidade em diferentes dispositivos, facilidade de atualização, compatibilidade com navegadores modernos e facilidade de compartilhamento. Embora tenham algumas limitações, as WebApps são uma opção viável para o desenvolvimento de aplicações multiplataforma e amplamente utilizadas em diversos cenários.

2.6 TRABALHOS CORRELATOS

Nesta seção, abordaremos os trabalhos relacionados a esta pesquisa, e que inspiraram a ser desenvolvido, abordaremos eles com fins de complementação e comparação.

2.6.1 Duolingo: Inglês e muito mais!

O Duolingo é um aplicativo móvel focado no ensinamento de línguas para pessoas de todas as faixas etárias, segundo Duolingo (2022) “O jeito grátis, divertido e eficaz de aprender um idioma!”.

O aplicativo possui como principal funcionalidade o uso da gamificação como forma de ensinar idiomas, utilizando da combinação do melhor da inteligência artificial e da ciência da linguagem, com lições feitas sob medida para ajudar a aprender no nível e ritmo certos (DUOLINGO, 2022). O uso da gamificação facilita a criação do hábito de aprender idiomas, reforçar a obtenção do hábito usam de desafios divertidos e lembretes por notificação.

O Duolingo ainda possui uma versão para escolas, com uma visão mais gerencial para professores permitindo a eles diferentes configurações, como o conteúdo que irá ser disponibilizado, as questões que podem ser exibidas, e ainda permite verifica o progresso dos alunos de forma detalhada (DUOLINGO, 2022).

Nas figuras abaixo pode-se ver um pouco sobre as funções disponíveis no aplicativo, na Figura 5 - Tela inicial do Duolingo: é exibida a tela inicial do Duolingo, onde tem-se a separação dos diferentes conteúdos disponíveis para o aluno e os diferentes níveis. A Figura 6 - Tela de Pergunta do Duolingo: representa como são exibidas as perguntas e respostas aos alunos. A Figura 7 - Tela de estudantes no Duolingo: exibe a tela de alunos, onde o professor poderá verificar o desempenho de cada aluno detalhadamente. A Figura 8 - Tela de ranking do Duolingo: mostra a visão dos alunos do ranqueamento atual. Figura 9 - Tela de seleção de conteúdos disponíveis no Duolingo: tela para configuração dos conteúdos aos quais os alunos terão acesso.

Figura 5 - Tela inicial do Duolingo



Fonte: <https://pt.duolingo.com/> (2022)

Figura 6 - Tela de Pergunta do Duolingo



Fonte: <https://pt.duolingo.com/> (2022)

Figura 7 - Tela de estudantes no Duolingo

French 2 Period 4

STUDENTS ASSIGN REPORTS SETTINGS

12 students

NAME	XP EARNED	TIME SPENT	LATEST ASSIGNMENT
Bea	1000 XP	23h 30m	2/4 lessons
Eddy	2900 XP	23h 2m	3/4 lessons
Junior	2340 XP	22h 10m	0/4 lessons
Lin	1400 XP	10h 30m	4/4 lessons
Oscar	1200 XP	12h 20m	2/4 lessons
Vikram	950 XP	22h 10m	0/4 lessons

Bea
bea@duolingo.com

Progress since joining class

5200 XP 120 hrs Time

Past assignments

- 12 of 20 Completed
- 4 of 20 Late
- 4 of 20 Missed

Fonte: <https://schools.duolingo.com/> (2022)

Figura 8 - Tela de ranking do Duolingo

Divisão Ouro
Os 10 primeiros avançam pra próxima divisão.
3 dias

1	Violeta Oliveira	105 XP
2	camymn	84 XP
3	Anyuska krisia Moura...	53 XP
4	tamara	44 XP
5	fede	30 XP
6	Alvia	28 XP
7	Anna	1 dia



Fonte: <https://pt.duolingo.com/> (2022)

Figura 9 - Tela de seleção de conteúdos disponíveis no Duolingo






French 2 Period 4

STUDENTS **ASSIGN** REPORTS SETTINGS

Browse and assign content

SKILL	LESSON	WORDS
 Basics 1	1	un, homme, garçon, chat, et
	2	une, femme, suis, fille, je
	3	chien, cheval, c'est, tu, es
	4	croissant, orange, mange, manges, pizza
 Greetings	1	bonjour, ça va, comment, bien, oui
	2	bonsoir, merci, salut, toi, très
	3	à bientôt, au revoir, enchanté, bonne journée, bienvenue
	4	bonne nuit, beaucoup, bonne soirée, à demain

Course overview

-  **7230** words
-  **8** units
-  **324** skills
-  **200** stories
-  **Novice low-Advanced mid** ACTFL levels

Fonte: <https://schools.duolingo.com/> (2022)

2.6.2 Grasshopper

O Grasshopper é um aplicativo de programação para iniciantes, com o propósito de ensinar com lições divertidas e rápidas no seu smartphone que ensinam você a programar em JavaScript (Grasshopper, 2022).

O Grasshopper utiliza de níveis cada vez mais desafiadores enquanto desenvolve suas habilidades, usa de recursos visuais e de gamificação como uma forma de prender a atenção do usuário, e motivá-lo a continuar estudando o assunto (Grasshopper, 2022).

Na Figura 10 – Tela inicial Grasshopper: pode-se ver a organização em diferentes fases e conteúdo. Na Figura 11 – Tela de pergunta Grasshopper: mostra um exemplo de como é exibida a pergunta ao usuário.

Figura 10 - Tela inicial Grasshopper



Fonte: https://grasshopper.app/pt_br/ (2022)

Figura 11 - Tela de pergunta Grasshopper



Fonte: https://grasshopper.app/pt_br/ (2022)

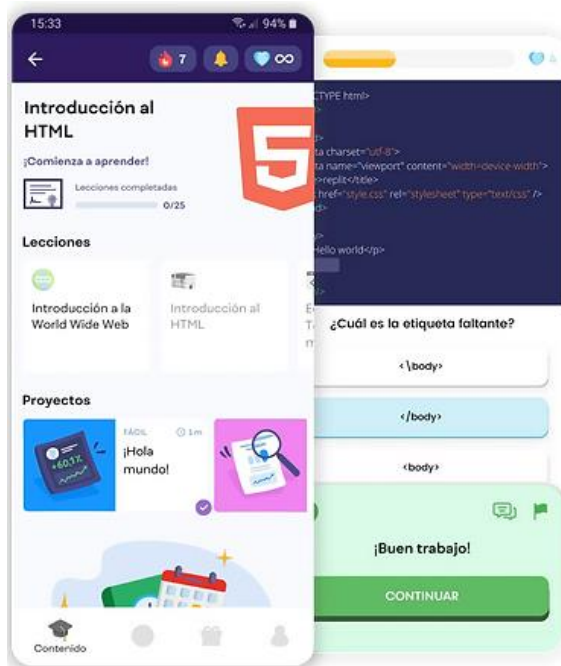
2.6.3 LearnU: Aprende a programar

O LearnU é um aplicativo para aprendizado de programação, possuindo vários recursos de gamificação, além de possuir conteúdos por extenso para leitura e em formato de vídeo. O LearnU possui como principais pontos fortes, o aprendizado com uso de lições e projetos pequenos e iterativos, possui uma comunidade para auxílio do aprendizado, além de focar de focar nas habilidades interpessoais dos usuários (LearnU, 2022).

O aplicativo possui múltiplas metodologias de ensino para melhorar efetivamente as habilidades e os conhecimentos do usuário, assim ajudando de diferentes formas, pois cada um possui um ritmo e capacidade diferentes, assim abrangendo um grupo maior de usuários satisfeitos (LearnU, 2022).

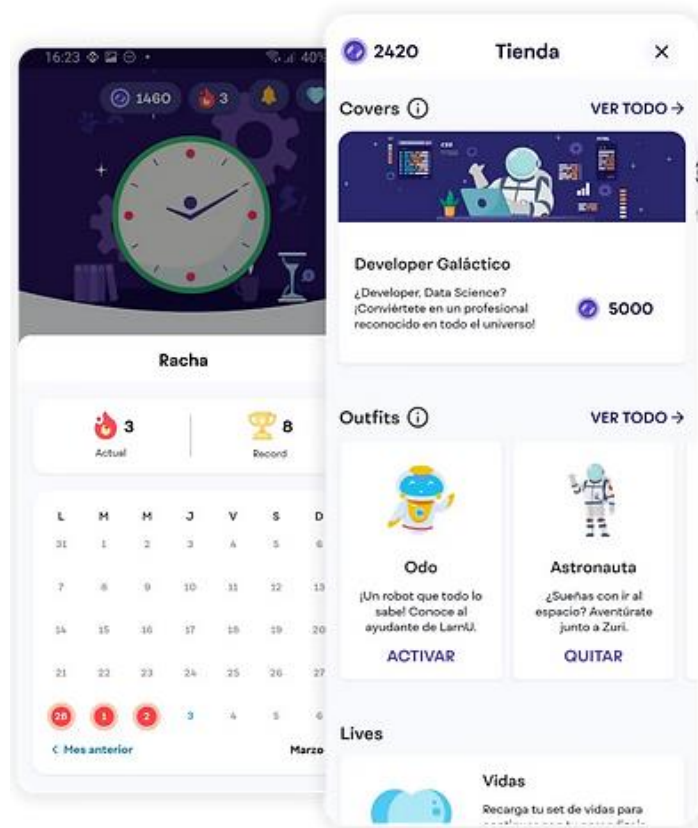
A Figura 12 – Tela de pergunta do LearnU: temos a tela de curso e ao lado direito temos a tela de perguntas, a qual os usuários respondem. A Figura 13 – Tela de desafio diário LearnU: é a tela de desafio diário, utilizada como forma de motivar os usuários diariamente. A Figura 14 – Tela de seleção de conteúdo do LearnU: tela do aplicativo que permite a seleção dos diferentes tipos de conteúdo do curso.

Figura 12 - Tela de pergunta do LearnU



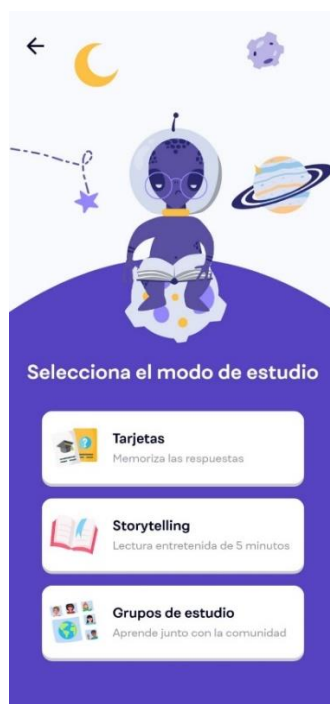
Fonte: <https://www.larnu.com/> (2022)

Figura 13 - Tela de desafio diário do LearnU



Fonte: <https://www.larnu.com/> (2022)

Figura 14 - Tela de seleção de conteúdo do LearnU



Fonte: <https://www.larnu.com/> (2022)

2.6.4 Comparação com os trabalhos correlatos

Entre todos os trabalhos correlatos citados anteriormente, o com maior semelhança ao desenvolvimento proposto neste trabalho é o Duolingo, mesmo que o mesmo não seja para o aprendizado de programação, os recursos do aplicativo possuem foco além do aprendizado do aluno, propõe o acompanhamento do professor, enquanto os outros trabalhos correlatos têm como principais semelhanças o uso de recursos de gamificação e o aprendizado a programação.

A Tabela 3, contém as principais funcionalidades do produto desenvolvido neste trabalho, e os trabalhos correlatos listados citados anteriormente.

Tabela 3 - Comparação com trabalhos correlatos.

	Este Trabalho	Duolingo	Grasshopper	LearnU
Possui recursos de gamificação?				
Possui visão de desempenho dos alunos para o professor?				
Possui sistema de rankeamento?				
Possui gerenciamento de turmas pelo professor?				
Possui visão de maiores dúvidas da turma?				

Fonte: Elaborado pelo autor (2022)

3 DESENVOLVIMENTO COM ENGENHARIA DE SOFTWARE

Na presente seção, será apresentado detalhadamente todos os itens da metodologia da engenharia de software aplicada para desenvolver o produto, em conformidade com a metodologia anteriormente apresentada na seção 1.3.

3.1 ENGENHARIA DE REQUISITOS

3.1.1 Requisitos funcionais

Os requisitos funcionais definem as principais funcionalidades e os problemas que o software em si busca resolver, utilizando com base os trabalhos correlatos, citados anteriormente, que possuem resultados comprovados, como o Duolingo, usando lições feitas sob medida para ajudar a aprender no nível e ritmo certos (DUOLINGO, 2022). A ferramenta permite ao professor construir formulários com flexibilidade, para que sejam alinhados conforme as aulas e conteúdos por ele tratados em sala. Com isso, a ferramenta poderá ser utilizada no ensino de outras disciplinas, mantendo os recursos de gamificação.

Tabela 4 - Requisitos funcionais

Número	Descrição	Regras de negócios
RF01	O aplicativo deve manter professores.	RN01
RF02	O aplicativo deve manter alunos.	RN01
RF03	O aplicativo deve manter turmas.	
RF04	O aplicativo deve realizar login.	
RF05	O aplicativo deve manter alunos na turma.	
RF06	O aplicativo deve permitir ao aluno selecionar palavras relacionadas às suas dúvidas.	RN06
RF07	O aplicativo deve gerar formulário a partir das palavras mais selecionadas pelos alunos.	RN07, RN08
RF08	O aplicativo deve permitir o professor ver as maiores dúvidas por nível.	

Fonte: Elaborado pelo autor (2022)

3.1.2 Requisitos da gamificação

Abaixo apresentaremos os requisitos da gamificação para o desenvolvimento do aplicativo.

Tabela 5 - Requisitos gamificação

Número	Descrição	Regras de negócios
RFG01	O aplicativo deve manter níveis.	
RFG02	Os níveis do jogo serão dispostos em formato de trilha.	
RFG03	As perguntas podem ter diferentes pontuações.	RN03
RFG04	O aplicativo deve permitir ao professor manter perguntas.	RN05
RFG05	Os alunos poderão jogar a gamificação.	RN02, RN12
RFG06	O aplicativo deve possuir ranqueamento dos alunos.	RN04
RFG07	Os alunos poderão ganhar medalhas.	RN10
RFG08	Os professores poderão ganhar medalhas.	RN11

Fonte: Elaborado pelo autor (2022)

3.1.3 Requisitos não funcionais

Os requisitos não funcionais determinam a parte de desenvolvimento do software como performance, confiabilidade, manutenibilidade, portabilidade, regras, entre outros pontos (CYSNEIROS, 2001).

Tabela 6 - Requisitos não funcionais

Número	Descrição
RNF01	O aplicativo deve ser desenvolvido com React.js.
RNF02	O aplicativo validará e-mail e senha no login.
RNF03	O aplicativo deve ser desenvolvido com Node.js
RNF04	O aplicativo deverá usar PostgreSQL

Fonte: Elaborado pelo autor (2022)

3.1.4 Regras de negócio

Abaixo apresentaremos as regras de negócio identificadas para o desenvolvimento do aplicativo.

Tabela 7 - Regras de negócio

Número	Descrição
RN01	Não poderá ser cadastrado alunos ou professores com o mesmo e-mail.
RN02	O aplicativo deve somente permitir acesso ao próximo nível após o anterior ser concluído.
RN03	As questões terão pesos de 1 à 10.
RN04	O ranqueamento dos alunos será baseado no somatório, das questões corretas respondidas por eles.
RN05	A quantidade de perguntas de cada nível será definida pelo professor.
RN06	As palavras chaves deverão ser cadastradas pelo professor nas perguntas.
RN07	O aplicativo deve permitir ao professor selecionar a quantidade máxima de perguntas.
RN08	O aplicativo deve permitir gerar questionário a partir das palavras mais selecionadas.
RN09	O aplicativo deve permitir gerar questionário padrão.
RN10	Os alunos ganharão medalhas conforme a quantidade de formulários respondidos, sendo: <ul style="list-style-type: none"> • Bronze: 5 formulários; • Prata: 10 formulários; • Ouro: 15 formulários; • Mestre: 20 formulários;
RN11	Os professores ganharão medalhas conforme a quantidade de formulários criados, sendo: <ul style="list-style-type: none"> • Bronze: 5 formulários;

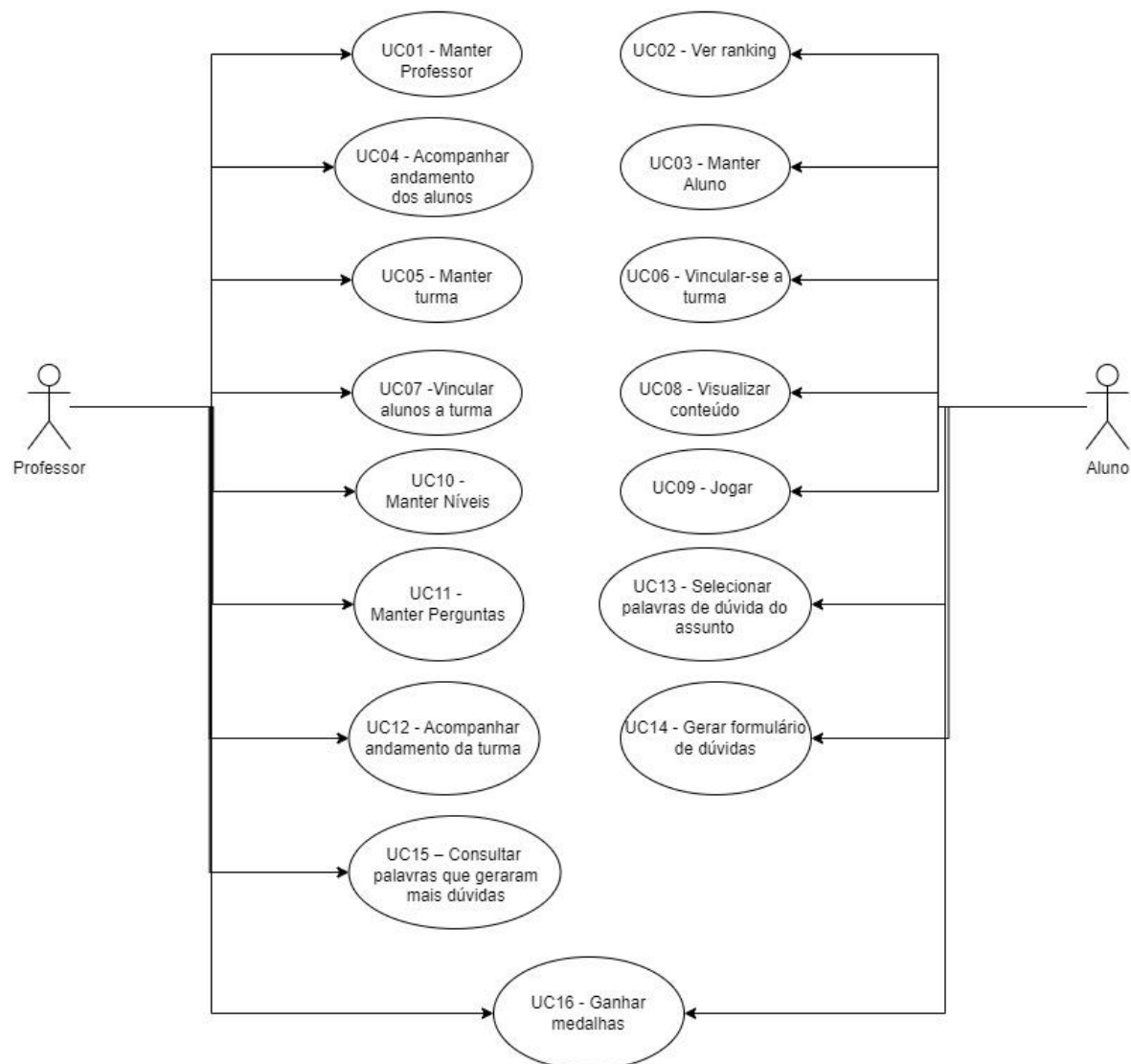
	<ul style="list-style-type: none"> • Prata: 10 formulários; • Ouro: 15 formulários; • Mestre: 20 formulários;
RN12	<p>O aluno será notificado sobre alterações na sua posição ao logar no aplicativo, sendo:</p> <ul style="list-style-type: none"> • Sucesso: caso se mantenha entre os 3 melhores; • Aviso: caso mantenha-se na posição abaixo dos 3 melhores; • Erro: caso caia posições, abaixo dos 3 melhores;

Fonte: Elaborado pelo autor (2022)

3.1.5 Casos de uso

Abaixo são apresentados os diagramas de caso de uso do projeto (Figura 15), visando detalhar os requisitos, e proporcionando um maior entendimento das suas respectivas funcionalidades.

Figura 15 - Diagrama de Casos de uso



Fonte: Elaborado pelo autor (2022)

3.1.6 Casos de uso 2.0

Abaixo apresentaremos os casos de uso 2.0 identificados para o desenvolvimento do aplicativo.

Caso de Uso	UC01 - Manter Professor
Descrição	Este caso de uso descreve o cadastro de professores
Requisitos	RF01
Atores	Professor

Pré-condições	-
Pós-condições	O cadastro será realizado e salvo em banco de dados, podendo o mesmo logar no sistema.
Fluxo Base	US01
Fluxos Alternativos	US02; US03; US04
Fluxos de Exceção	US29

Caso de Uso	UC02 - Ver ranking
Descrição	Este caso de uso descreve a visualização do ranking do aluno
Requisitos	RFG06
Atores	Alunos
Pré-condições	Possuir alunos cadastrados.
Pós-condições	-
Fluxo Base	US05
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC03 – Manter Aluno
Descrição	Este caso de uso descreve o cadastro do aluno
Requisitos	RF02
Atores	Aluno
Pré-condições	-
Pós-condições	O cadastro será realizado e salvo em banco de dados, podendo o mesmo logar no sistema.
Fluxo Base	US06
Fluxos Alternativos	US07; US08; US09
Fluxos de Exceção	US30

Caso de Uso	UC04 - Acompanhar andamento dos alunos.
Descrição	Este caso de uso descreve a etapa do professor ver o desempenho dos alunos.

Requisitos	RF05
Atores	Professor
Pré-condições	Possuir turma, e alunos cadastrados.
Pós-condições	-
Fluxo Base	US10
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC05 – Manter turma
Descrição	Este caso de uso se refere ao cadastro de turmas
Requisitos	RF03
Atores	Professor
Pré-condições	-
Pós-condições	A turma é salva no banco de dados, e o professor conseguirá ver ela na tela das suas turmas.
Fluxo Base	US11
Fluxos Alternativos	US12; US13; US14
Fluxos de Exceção	-

Caso de Uso	UC06 - Vincular-se a turma
Descrição	Esse caso de uso descreve a etapa do aluno se vincular a uma turma.
Requisitos	RF06
Atores	Aluno
Pré-condições	O aluno estar logado, e possuir turmas cadastradas.
Pós-condições	O aluno pode ver os conteúdos relacionados a turma.
Fluxo Base	US15
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC07 - Vincular alunos a turma
--------------------	--------------------------------

Descrição	Este caso de uso descreve a etapa de vincular um aluno a uma turma.
Requisitos	RF06
Atores	Aluno, Professor
Pré-condições	-
Pós-condições	Após vincular os alunos a uma turma, o professor poderá visualizar na tela de alunos da turma os alunos vinculados.
Fluxo Base	US16
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC08 - Visualizar conteúdo
Descrição	Este caso de uso descreve a etapa da visualização dos conteúdos.
Requisitos	RFG05
Atores	Alunos
Pré-condições	-
Pós-condições	Poderá visualizar em sua tela inicial os diferentes níveis para serem jogados.
Fluxo Base	UC17
Fluxos Alternativos	-
Fluxos de Exceção	US28

Caso de Uso	UC09 - Jogar
Descrição	Esse caso de uso descreve a etapa do aluno jogar a gamificação.
Requisitos	RFG05
Atores	Alunos
Pré-condições	O aluno precisa estar vinculado a uma turma.
Pós-condições	O aluno após responder as questões do nível ganha acesso ao próximo.

Fluxo Base	US18
Fluxos Alternativos	-
Fluxos de Exceção	US31

Caso de Uso	UC10 - Manter Níveis
Descrição	Esse caso de uso descreve a etapa de manter níveis
Requisitos	RFG01
Atores	Professor
Pré-condições	O professor deve ter uma turma cadastrada.
Pós-condições	Os alunos dessa turma terão acesso ao nível cadastrado.
Fluxo Base	US19
Fluxos Alternativos	US20; US21; US22
Fluxos de Exceção	-

Caso de Uso	UC11 - Manter Perguntas
Descrição	Esse caso de uso descreve o cadastro de perguntas.
Requisitos	RFG04
Atores	Professor
Pré-condições	O professor deve ter uma turma cadastrada.
Pós-condições	Os alunos terão acesso as perguntas cadastradas.
Fluxo Base	US23
Fluxos Alternativos	US24; US25; US26
Fluxos de Exceção	-

Caso de Uso	UC12 – Acompanhar andamento da turma
Descrição	Esse caso de uso descreve a geração dos gráficos de acompanhamento da turma.
Requisitos	RF05
Atores	Professor
Pré-condições	Ter uma turma cadastrada; Ter alunos cadastrados na turma.

Pós-condições	-
Fluxo Base	US27
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC13 - Selecionar palavras-chave do assunto
Descrição	Esse caso de uso descreve a seleção de palavras chaves pelo professor sobre o assunto.
Requisitos	RF08
Atores	Professor
Pré-condições	Ter uma turma cadastrada; Ter alunos cadastrados na turma; Ter assuntos cadastrados; Ter palavras chaves informadas;
Pós-condições	-
Fluxo Base	US32
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC14 – Gerar formulário por palavras selecionadas
Descrição	Esse caso de uso descreve a geração de formulários de dúvida pelo aluno.
Requisitos	RF07
Atores	Aluno
Pré-condições	Ter uma turma cadastrada; Ter alunos cadastrados na turma; Ter assuntos cadastrados; Ter palavras chaves informadas;
Pós-condições	-
Fluxo Base	US33
Fluxos Alternativos	US34
Fluxos de Exceção	-

Caso de Uso	UC15 – Consultar palavras que geraram mais dúvidas
Descrição	Esse caso de uso descreve a verificação das maiores dúvidas da turma no nível.
Requisitos	RF09
Atores	Professor
Pré-condições	Ter uma turma cadastrada; Ter alunos cadastrados na turma; Ter assuntos cadastrados; Ter palavras chaves informadas; Ter formulários de dúvidas gerados.
Pós-condições	-
Fluxo Base	US35
Fluxos Alternativos	-
Fluxos de Exceção	-

Caso de Uso	UC16 – Ganhar medalhas
Descrição	Esse caso de uso descreve como os alunos e professores ganham medalhas
Requisitos	RFG07, RFG08
Atores	Professor, Alunos
Pré-condições	Ter uma turma cadastrada; Ter alunos cadastrados na turma; Ter assuntos cadastrados; Ter palavras chaves informadas; Ter formulários de dúvidas gerados.
Pós-condições	-
Fluxo Base	US36
Fluxos Alternativos	US37
Fluxos de Exceção	-

3.1.7 User Stories

Abaixo apresentaremos as user stories identificadas para o desenvolvimento do aplicativo.

User Story	US01 – Cadastrar Professor
Use Case	UC01 - Manter Professor
Como	Professor
Eu quero	Realizar o meu cadastro.
Para	Que eu possa me cadastrar e utilizar o sistema.
Script	<ol style="list-style-type: none"> 1. Professor seleciona a opção para criar conta. 2. Professor seleciona a opção de que é um professor. 3. Professor informa os dados do cadastro. 4. Sistema valida se já possui alguém com o mesmo e-mail cadastrado.
Critério de aceitação	- Apresenta uma mensagem de sucesso e retorna para a tela de login, caso não exista alguém com o mesmo e-mail cadastrado.

User Story	US02 – Alterar Professor
Use Case	UC01 - Manter Professor
Como	Professor
Eu quero	Alterar ou corrigir informações do meu cadastro
Para	Que caso me cadastre com informações incorretas ou deseje alterar as informações cadastradas seja possível.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de visualizar o seu usuário. 3. Professor seleciona a opção de editar informações. 4. Professor altera as informações desejadas. 5. Sistema valida se existe algum cadastro com o e-mail informado.
Critério de aceitação	- Apresenta uma mensagem de erro, caso exista alguém com o mesmo e-mail cadastrado.

	- Apresenta uma mensagem de sucesso e retorna para a tela do usuário, caso não exista alguém com o mesmo e-mail cadastrado.
--	---

User Story	US03 – Consultar Professor
Use Case	UC01 - Manter Professor
Como	Professor
Eu quero	Ver meus dados cadastrados.
Para	Verificar se estão todos corretos ou se necessitam de alguma alteração.
Script	1. Professor realiza login no sistema. 2. Professor seleciona a opção de visualizar o seu usuário.
Critério de aceitação	- Apresenta as informações de nome e e-mail em tela.

User Story	US04 – Excluir Professor
Use Case	UC01 - Manter Professor
Como	Professor
Eu quero	Excluir meu cadastro.
Para	Excluir minha conta caso não deseje fazer mais parte do aplicativo.
Script	1. Professor realiza login no sistema. 2. Professor seleciona a opção de visualizar o seu usuário. 3. Professor seleciona a opção de excluir conta. 4. Professor seleciona a confirmar.
Critério de aceitação	- Apresenta uma mensagem de sucesso em tela, e retorna para a tela de login. - Caso tentar logar novamente o usuário não conseguirá e o sistema emitirá um aviso de credenciais incorretas.

User Story	US05 – Ver ranqueamento dos alunos
Use Case	UC02 - Ver ranking

Como	Aluno
Eu quero	Ver o ranking da minha classe.
Para	Poder verificar e comparar meu desempenho com o restante dos alunos da minha turma.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de ranking. 3. O Sistema exibe o ranking atual da turma.
Critério de aceitação	- O Sistema irá exibir em tela o ranking da classe.

User Story	US06 – Cadastrar Aluno
Use Case	UC03 – Manter Aluno
Como	Aluno
Eu quero	Realizar meu cadastro no sistema.
Para	Poder logar e utilizar o sistema.
Script	<ol style="list-style-type: none"> 1. Aluno seleciona a opção para criar conta. 2. Aluno seleciona a opção de que é um aluno. 3. Aluno informa os dados do cadastro. 4. Sistema valida se já possui alguém com o mesmo e-mail cadastrado.
Critério de aceitação	- Apresenta uma mensagem de sucesso e retorna para a tela de login, caso não exista alguém com o mesmo e-mail cadastrado.

User Story	US07 – Alterar Aluno
Use Case	UC03 – Manter Aluno
Como	Aluno
Eu quero	Alterar ou corrigir informações do meu cadastro
Para	Que caso me cadastre com informações incorretas ou deseje alterar as informações cadastradas seja possível.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de visualizar o seu usuário.

	3. Aluno seleciona a opção de editar informações. 4. Aluno altera as informações desejadas. 5. Sistema valida se existe algum cadastro com o e-mail informado.
Critério de aceitação	- Apresenta uma mensagem de sucesso e retorna para a tela do usuário, caso não exista alguém com o mesmo e-mail cadastrado.

User Story	US08 – Consultar Aluno
Use Case	UC03 – Manter Aluno
Como	Aluno
Eu quero	Ver meus dados cadastrados.
Para	Verificar se estão todos corretos ou se necessitam de alguma alteração.
Script	1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de visualizar o seu usuário.
Critério de aceitação	- Apresenta as informações de nome e e-mail em tela.

User Story	US09 – Excluir Aluno
Use Case	UC03 – Manter Aluno
Como	Aluno
Eu quero	Excluir meu cadastro.
Para	Excluir minha conta caso não deseje fazer mais parte do aplicativo.
Script	1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de visualizar o seu usuário. 3. Aluno seleciona a opção de excluir conta. 4. Aluno seleciona a confirmar.
Critério de aceitação	- Apresenta uma mensagem de sucesso em tela, e retorna para a tela de login.

	- Caso tentar logar novamente o usuário não conseguirá e o sistema emitirá um aviso de credenciais incorretas.
--	--

User Story	US10 – Acompanhar desempenho dos alunos.
Use Case	UC04 - Acompanhar andamento dos alunos.
Como	Professor
Eu quero	Ver o desempenho individual dos alunos da turma.
Para	Poder acompanhar suas dificuldades, e poder tomar planos de ação para agir.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona o aluno. 5. Professor seleciona a opção para visualizar desempenho.
Critério de aceitação	- Em tela serão exibidas as informações de como está o andamento do aluno no curso.

User Story	US11 – Cadastrar turma.
Use Case	UC05 – Manter turma
Como	Professor
Eu quero	Cadastrar uma turma
Para	Poder adicionar os alunos, o conteúdo e questões da matéria.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a opção de adicionar turmas. 4. Professor preenche os campos e confirma.
Critério de aceitação	- Sistema retornará para a tela de turmas, com a turma recém adicionada na listagem.

User Story	US12 – Alterar turma
Use Case	UC05 – Manter turma
Como	Professor

Eu quero	Alterar informações da turma.
Para	Que caso me cadastre com informações incorretas ou deseje alterar as informações cadastradas seja possível.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de alterar. 5. Professor informa os dados desejados e confirma.
Critério de aceitação	- Sistema retornará para a tela da turma, com as novas informações cadastradas já em tela.

User Story	US13 – Consultar turma
Use Case	UC05 – Manter turma
Como	Professor
Eu quero	Visualizar os dados de determinada turma.
Para	Conferir se os dados foram corretamente preenchidos.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor poderá visualizar as informações da turma.
Critério de aceitação	- As informações exibidas na tela são equivalentes as informadas no cadastro.

User Story	US14 – Excluir turma
Use Case	UC05 – Manter turma
Como	Professor
Eu quero	Excluir uma turma.
Para	Caso tenha cadastrado por engano poder exclui-la
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção para excluir turma.

	5. Professor confirma a exclusão.
Critério de aceitação	- Retornará para a tela de turmas, e a turma excluída não estará mais na listagem.

User Story	US15 – Vincular-se a turma
Use Case	UC06 - Vincular-se a turma
Como	Aluno
Eu quero	Poder me vincular a turma.
Para	Poder acessar os conteúdos e jogar.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de visualizar suas turmas. 3. Aluno seleciona a opção de vincular-se a uma turma. 4. Aluno seleciona a turma desejada e confirma.
Critério de aceitação	- O sistema retorna para a tela de turmas do aluno, com a turma recém vinculada.

User Story	US16 – Vincular alunos a turma.
Use Case	UC07 - Vincular alunos a turma
Como	Professor
Eu quero	Poder vincular alunos a uma turma.
Para	Poder adicionar os alunos a turma.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de adicionar alunos. 5. Professor seleciona e confirma a adição do aluno.
Critério de aceitação	- Sistema retorna para a turma, com os alunos adicionados na listagem de alunos.

User Story	US17 – Visualizar conteúdo do nível
Use Case	UC08 - Visualizar conteúdo
Como	Aluno

Eu quero	Ver o conteúdo do nível a ser jogado.
Para	Estudar antes de fazer a gamificação.
Script	1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de jogar. 3. Aluno seleciona a opção de conteúdo no nível desejado.
Critério de aceitação	- O sistema fará o download de um pdf com o conteúdo da matéria.

User Story	US18 – Jogar
Use Case	UC09 - Jogar
Como	Aluno
Eu quero	Jogar
Para	Aprender e avançar no conteúdo da matéria.
Script	1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de jogar. 3. Aluno seleciona o nível desejado, e clica para jogar.
Critério de aceitação	- O sistema emite em tela a primeira questão do nível atual.

User Story	US19 – Cadastrar nível
Use Case	UC10 - Manter Níveis
Como	Professor
Eu quero	Poder cadastrar níveis
Para	Os visualizarem o conteúdo de forma organizada para jogar.
Script	1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona a opção de cadastrar. 6. Professor informa os dados necessários e confirma.
Critério de aceitação	- O sistema retorna para a página de níveis, com o nível recém cadastrado na lista.

User Story	US20 – Alterar nível
Use Case	UC10 - Manter Níveis
Como	Professor
Eu quero	Alterar as informações do nível
Para	Que caso me cadastre com informações incorretas ou deseje alterar as informações cadastradas seja possível.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível a editar. 6. Professor informa os dados desejados e confirma.
Critério de aceitação	- O sistema retorna para a página de níveis, com o as informações do nível alteradas.

User Story	US21 – Consultar nível
Use Case	UC10 - Manter Níveis
Como	Professor
Eu quero	Ver o nível
Para	Verificar se as informações estão corretas ou necessitam ajustes.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível e ver as informações
Critério de aceitação	- Em tela são exibidas as informações do nível.

User Story	US22 – Excluir nível
Use Case	UC10 - Manter Níveis
Como	Professor

Eu quero	Excluir um nível
Para	Remover o nível do jogo
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível e selecionar a opção de remover. 6. Professor confirma a remoção.
Critério de aceitação	- Sistema retorna para a tela de nível, e o nível removido não se encontra lá.

User Story	US23 – Cadastrar pergunta
Use Case	UC11 - Manter Perguntas
Como	Professor
Eu quero	Cadastrar perguntas no nível
Para	Os alunos poderem responder as questões do conteúdo
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível. 6. Professor seleciona a opção de adicionar questão. 7. Professor Informa a pergunta e as resposta e confirma.
Critério de aceitação	- Sistema retorna para tela de nível com a pergunta já adicionada.

User Story	US24 – Alterar pergunta
Use Case	UC11 - Manter Perguntas
Como	Professor
Eu quero	Alterar perguntas no nível
Para	Corrigir alguma informação incorreta da questão.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema.

	2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível. 6. Professor seleciona a questão incorreta. 7. Professor informa os dados corretamente, e confirma.
Critério de aceitação	- Sistema retorna para tela de nível com a pergunta já adicionada.

User Story	US25 - Consultar pergunta
Use Case	UC11 - Manter Perguntas
Como	Professor
Eu quero	Consultar e verificar perguntas
Para	Verificar se as informações estão corretas.
Script	1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível. 6. Professor seleciona a questão desejada.
Critério de aceitação	- Sistema exibirá em tela a questão desejada, com as informações da pergunta e da resposta.

User Story	US26 – Excluir pergunta
Use Case	UC11 - Manter Perguntas
Como	Professor
Eu quero	Excluir perguntas
Para	Remover perguntas adicionadas erroneamente, ou atualizar o conteúdo
Script	1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada.

	4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível. 6. Professor seleciona a pergunta deseja. 7. Professor seleciona a opção de excluir, e confirma.
Critério de aceitação	- O Sistema retorna para o nível, e a pergunta removida não estará mais disponível.

User Story	US27 – Consultar desempenho da turma
Use Case	UC12 – Acompanhar andamento da turma
Como	Professor
Eu quero	Acompanhar o desempenho da turma
Para	Verificar as matérias e assuntos a serem melhorados.
Script	1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção para ver desempenho.
Critério de aceitação	- Em tela serão gerados gráficos do desempenho atual da turma detalhadamente.

User Story	US28 – Visualizar nível sem conteúdo cadastrado
Use Case	UC08 - Visualizar conteúdo
Como	Aluno
Eu quero	Ver o conteúdo do nível a ser jogado.
Para	Estudar antes de fazer a gamificação.
Script	1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de jogar. 3. Aluno seleciona a opção de conteúdo no nível desejado.
Critério de aceitação	- O nível não possui conteúdo cadastrado então emite uma mensagem de aviso

User Story	US29 – Cadastrar Professor Existente
Use Case	UC01 - Manter Professor

Como	Professor
Eu quero	Realizar o meu cadastro.
Para	Que eu possa me cadastrar e utilizar o sistema.
Script	<ol style="list-style-type: none"> 1. Professor seleciona a opção para criar conta. 2. Professor seleciona a opção de que é um professor. 3. Professor informa os dados do cadastro. 4. Sistema valida se já possui alguém com o mesmo e-mail cadastrado.
Critério de aceitação	- Apresenta uma mensagem de erro, caso exista alguém com o mesmo e-mail cadastrado.

User Story	US30 – Cadastrar Aluno Existente
Use Case	UC03 – Manter Aluno
Como	Aluno
Eu quero	Realizar meu cadastro no sistema.
Para	Poder logar e utilizar o sistema.
Script	<ol style="list-style-type: none"> 1. Aluno seleciona a opção para criar conta. 2. Aluno seleciona a opção de que é um aluno. 3. Aluno informa os dados do cadastro. 4. Sistema valida se já possui alguém com o mesmo e-mail cadastrado.
Critério de aceitação	- Apresenta uma mensagem de erro, caso exista alguém com o mesmo e-mail cadastrado.

User Story	US31 – Jogar nível indisponível
Use Case	UC09 - Jogar
Como	Aluno
Eu quero	Jogar
Para	Aprender e avançar no conteúdo da matéria.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de jogar. 3. Aluno seleciona o nível desejado, e clica para jogar.

Critério de aceitação	- Se o nível estiver indisponível emite uma mensagem de aviso que o nível está indisponível
------------------------------	---

User Story	US32 - Selecionar palavras-chave do assunto
Use Case	UC13 - Selecionar palavras-chave do assunto
Como	Professor
Eu quero	Selecionar palavras-chave do assunto
Para	Os alunos poderem responder questionários com perguntas relacionadas às suas dúvidas.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível. 6. Professor seleciona a opção de adicionar questão. 7. Professor Informa as palavras chaves.
Critério de aceitação	- Sistema retorna para tela de nível com a pergunta já adicionada.

User Story	US33 - Selecionar palavras de dúvida do assunto
Use Case	UC14 - Selecionar palavras de dúvida do assunto
Como	Aluno
Eu quero	Responder questionários sobre as maiores dúvidas sobre o assunto.
Para	Poder entender melhor o conteúdo.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de jogar. 3. Aluno seleciona o nível desejado, e seleciona as palavras de dúvida.
Critério de aceitação	- O Aluno retorna para o nível e recebe uma mensagem de confirmação;

User Story	US34 – Responder questionários de maiores dúvidas.
Use Case	UC14 - Selecionar palavras de dúvida do assunto
Como	Aluno
Eu quero	Responder questionários sobre as maiores dúvidas sobre o assunto.
Para	Poder entender melhor o conteúdo.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno seleciona a opção de jogar. 3. Aluno seleciona o nível desejado. 4. Seleciona a opção de responder questionário sobre dúvidas.
Critério de aceitação	- O Aluno pode responder as perguntas normalmente;

User Story	US35 – Ver maiores dúvidas da turma no nível.
Use Case	UC15 – Ver maiores dúvidas do nível
Como	Professor
Eu quero	Acompanhar os assuntos de mais dúvidas da turma
Para	Poder reforçar os conteúdos que os alunos possuem mais dúvidas.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor seleciona a opção de turmas. 3. Professor seleciona a turma desejada. 4. Professor seleciona a opção de níveis. 5. Professor seleciona o nível. 6. Professor seleciona a opção de palavras chaves.
Critério de aceitação	- O professor pode ver as palavras chaves cadastradas, e pode ver as palavras chaves com mais dúvidas;

User Story	US36 – Ganhar medalhas respondendo formulários
Use Case	UC16 – Ganhar medalhas
Como	Aluno
Eu quero	Ganhar medalhas respondendo questionários

Para	Estimular o aluno a utilizar e responder os questionários do sistema.
Script	<ol style="list-style-type: none"> 1. Aluno realiza login no sistema. 2. Aluno entra na tela de conquistas. 3. Aluno verifica quantos questionários faltam para ganhar a próxima medalha. 4. Aluno responde a quantidade faltante de questionários, para ganhar a nova medalha.
Critério de aceitação	- O aluno acessa a tela de conquistas e verifica que a nova medalha está desbloqueada;

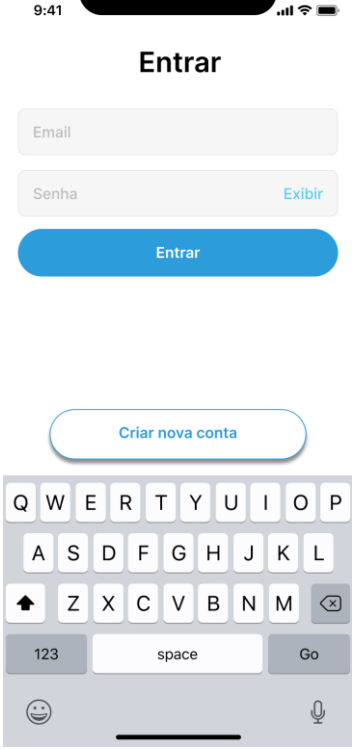
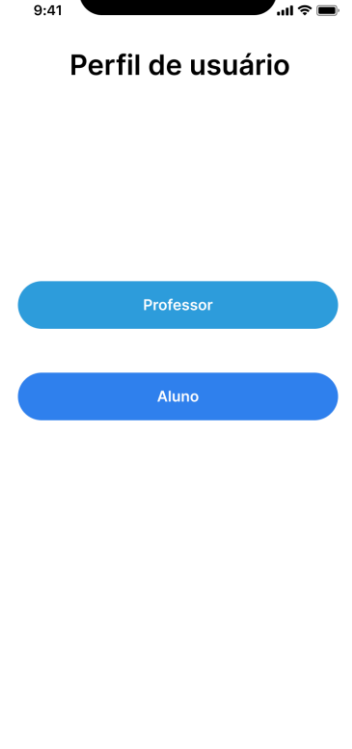

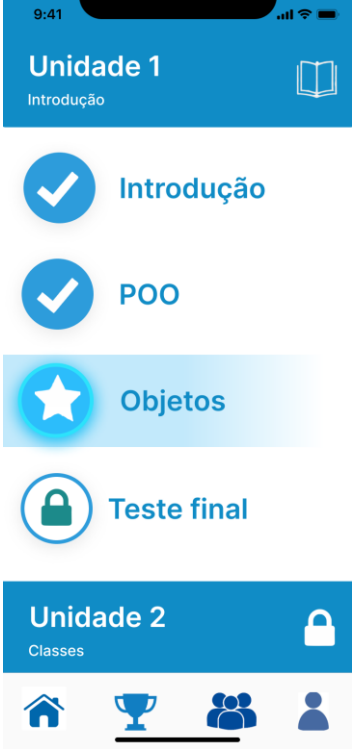
User Story	US37 – Ganhar medalhas cadastrando formulários
Use Case	UC16 – Ganhar medalhas
Como	Professor
Eu quero	Ganhar medalhas cadastrando questionários
Para	Estimular o professor a utilizar e cadastrar os questionários no sistema.
Script	<ol style="list-style-type: none"> 1. Professor realiza login no sistema. 2. Professor entra na tela de conquistas. 3. Professor verifica quantos questionários faltam para ganhar a próxima medalha. 4. Professor cadastra a quantidade faltante de questionários, para ganhar a nova medalha.
Critério de aceitação	- O Professor acessa a tela de conquistas e verifica que a nova medalha está desbloqueada;

3.1.8 Prototipação

Para a criação dos protótipos de tela, os quais serão utilizados para nortear o desenvolvimento da aplicação, foi utilizada a ferramenta Figma, que possibilita a criação de componentes reutilizáveis em diferentes telas.

Na Tabela 8 - Protótipos de Tela, contém os protótipos desenvolvidos, junto a um breve comentário de sua principal funcionalidade.

Tabela 8 - Protótipos de Tela

 <p>1 – Tela de Login</p>	 <p>2 – Tela de seleção de tipo de perfil</p>
 <p>3 – Tela de Cadastro de usuário</p>	 <p>4 – Tela inicial aluno</p>







Fonte: Elaborado pelo autor (2022)

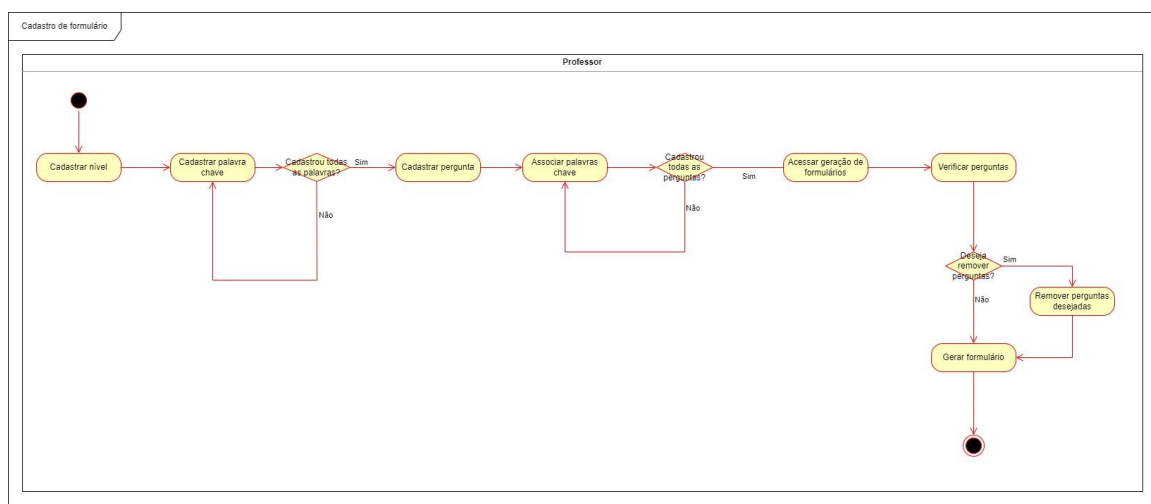
3.2 PROJETO

Nesta seção, serão apresentados os detalhes sobre a fase de projeto deste trabalho, incluindo os diagramas UML que estão relacionados ao funcionamento do sistema, além de fornecer informações mais detalhadas sobre o desenvolvimento do projeto e sua estrutura interna.

3.2.1 Diagrama de atividades

No diagrama de atividades a seguir apresentado na figura 16, pode-se ver o fluxo necessário para realizar o cadastro de formulários na aplicação, desde o cadastro do nível até a etapa de geração de formulário, sendo o momento em que o mesmo é disponibilizado para os alunos.

Figura 16 - Diagrama de atividades



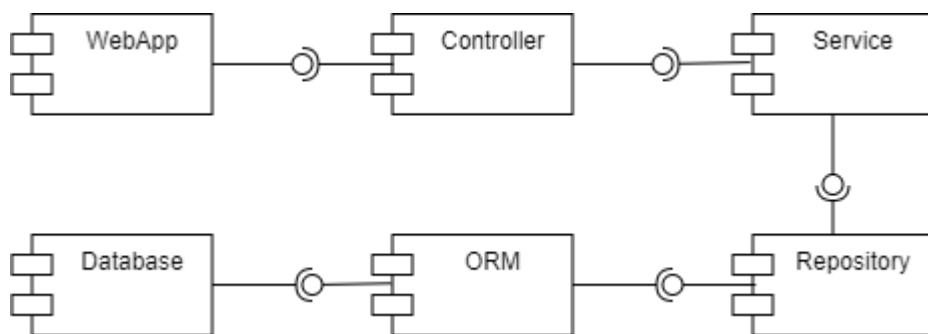
Fonte: Elaborado pelo autor (2023)

3.2.2 Diagrama de componentes

Abaixo apresentaremos o diagrama de componentes elaborado para o desenvolvimento do aplicativo. Esse diagrama é uma representação visual dos componentes do sistema e suas interações, o que ajuda a compreender melhor como o aplicativo funciona e como seus diferentes elementos estão relacionados entre si. A seguir, apresentamos o diagrama de componentes para o nosso aplicativo:

- WebApp: Representa o front-end, responsável da parte visual e funcional do sistema;
- Controller: Responsável por receber as requisições do front-end;
- Service: Responsável pelos tratamentos dos dados e aplicações de regras de negócio;
- Repository: Responsável por ir ao banco de dados buscar os dados requisitados;
- ORM: Responsável pelo correto mapeamento dos dados ao requisitar o Banco de Dados;
- DataBase: Representando o banco de dados do sistema.

Figura 17 - Diagrama de componentes

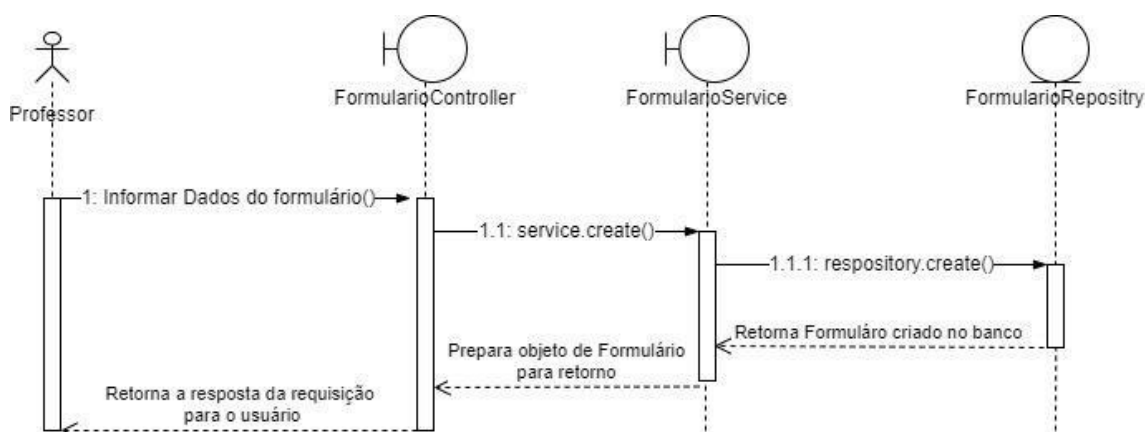


Fonte: Elaborado pelo autor (2022)

3.2.3 Diagramas de sequência

Abaixo está apresentado o diagrama de sequência, representando o fluxo de cadastro de formulários no sistema.

Figura 18 - Diagrama de sequência do cadastro de formulário



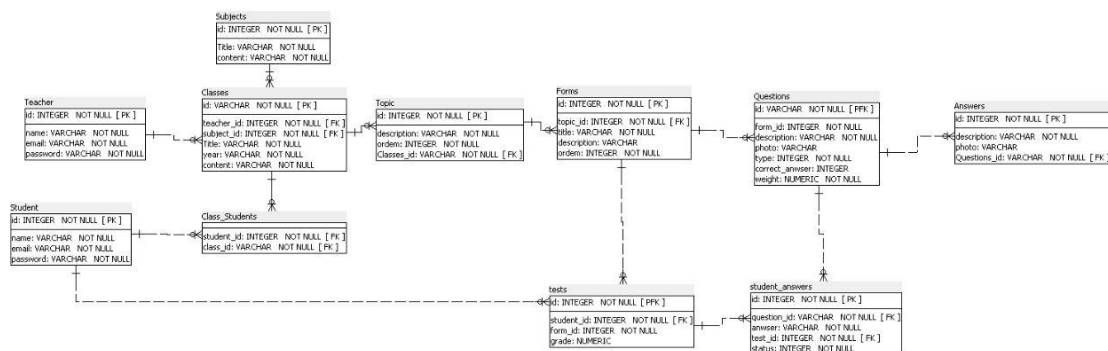
Fonte: Elaborado pelo autor (2022)

Na figura 18, temos o diagrama de sequência representando o fluxo de cadastro de formulário realizado pelo professor, sendo um fluxo de trabalho de menor complexidade, onde o professor cadastra as informações do formulário, o sistema abstrai a informação e faz as conversões desses dados para criação do formulário, retornando novamente para o professor as informações do formulário recém-criado.

3.2.4 Diagrama lógico de banco de dados

Abaixo está representado o diagrama lógico de banco de dados da aplicação, que ilustra os relacionamentos entre as tabelas do banco de dados. O diagrama apresenta a relação entre as questões e os formulários, os formulários e os níveis, bem como a relação entre os professores e alunos com as turmas. Este diagrama é fundamental para o desenvolvimento do sistema, uma vez que exibe de forma clara e organizada todos os relacionamentos entre as entidades e tabelas envolvidas no aplicativo. Com isso, os desenvolvedores podem ter uma visão completa da estrutura do banco de dados e garantir a integridade e eficiência do sistema.

Figura 19 - Diagrama lógico de banco de dados

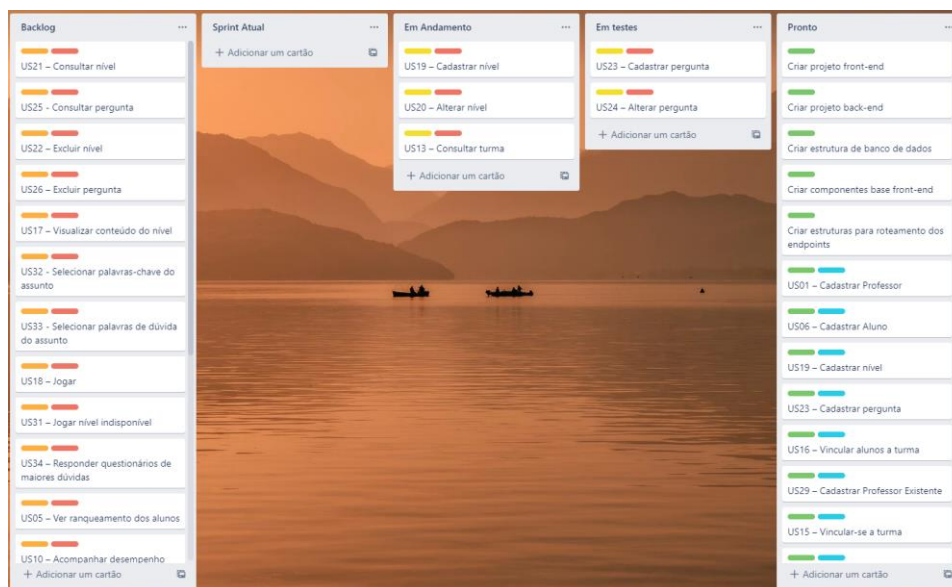


Fonte: Elaborado pelo autor (2022)

3.3 GESTÃO ÁGIL DO DESENVOLVIMENTO

Para o desenvolvimento da aplicação foi utilizado o Trello para a gestão das sprints, o projeto foi organizado em quatro sprints iniciais, após a construção do backlog, havendo mais duas sprints para correção e alinhamento de funcionalidades posteriormente, visando principalmente o desenvolvimento do MVP (Minimum Viable Product). O backlog do projeto foi elaborado utilizando como base as User Storys que já haviam sido escritas anteriormente, havendo a decomposição de cada história em tarefas de front-end e back-end, para um melhor planejamento do desenvolvimento. Na figura 20, pode-se ver o quadro de sprints em andamento.

Figura 20 - Sprints no Trello



Fonte: <https://trello.com/> (2022)

A separação das task sprints ocorreu da seguinte maneira, na primeira sprint temos a criação dos projetos de front-end e back-end, e as definições iniciais deles, e o desenvolvimento das principais funcionalidades do back-end. Na segunda sprint foi visado finalizar o back-end e desenvolver os cadastros necessários do front-end e começar o desenvolvimento da gamificação, na terceira sprint visa finalizar as funcionalidades da gamificação, exibição de ranqueamento, e finalizar alguns detalhes dos cadastros, enquanto na quarta sprint tem-se como foco a realização dos testes e validação de tudo que foi desenvolvido.

3.3.1 Sprint 1

Na Sprint 1 foi planejado para a iniciação dos projetos do back-end e front-end, e o desenvolvimento foi focado na estrutura do back-end, e grande parte dos CRUD's básicos no back-end, por se tratar de estruturas similares, foram desenvolvidos juntos. A parte de maior consumo de tempo foi separar e estruturar o back-end em três diferentes módulos sendo:

- Usuários: Agrupando alunos e professores;
- Universidade: Agrupa tópicos, classes e matérias;

- Formulários: Agrupa testes, respostas, questões e palavras chaves.

Cada divisão dos módulos possui um controller, um service, uma rota e um arquivo de validação, também havendo arquivos para tratamentos e autenticação dos diferentes grupos de usuário.

Nas imagens a seguir temos algumas classes do back-end que foram criadas nessa sprint.

Figura 21 - Teste service

```

You, há 3 dias | 1 author (You)
interface IRequestTest {
  form_id: number
  student_id: number
  answers: IRequestAnswer[]
}

You, há 3 dias | 1 author (You)
interface IRequestAnswer {
  question_id: number
  description: string | number | number[]
}

You, há 3 dias | 1 author (You)
@Injectable()
export default class TestsService extends Service {
  constructor(private questionsService: QuestionsService) {
    super();
  }

  client = client.test;

  public async findById(id: number) {
    const test = await this.client.findFirst({
      where: { id },
    });
    return test;
  }

  public async findAll(data: object = {}) {
    const tests = await super.findAll(data);
    return tests;
  }

  public handleGradeOfMultipleAlternative({
    answer,
    question,
  }): {
    answer: StudentAnswers & {
      question: Question | null
    }
    question: Question
  } {
    const answerArray = answer.description.split(', ');

```

Fonte: *print screen* de trecho de código da aplicação no VSCode (2023)

Figura 22 - Classe service

```

import { Class } from '@prisma/client';
import { injectable } from 'tsyringe';

import Service from '@shared/core/Service';
import client from '@shared/infra/prisma/client';

import ITeacherDTO from '...../users/dtos/ITeacherDTO';
import ClassRequest from '...../infra/http/requests/ClassRequest';

You, há 3 dias | I author (You)
interface IRequest {
  title: string;
  content: string;
  year: number;
  semester: number;
  teacher_id: number;
  subject_id: number;
  students: number[];
}

You, há 3 dias | I author (You)
@injectable()
export default class ClassesService extends Service {

  client = client.class;

  public async findById(id: number) {
    const classe = await this.client.findFirst({
      where: { id },

      include: {
        subject: true,
        teacher: {
          select: {
            id: true,
            email: true,
            name: true,
          },
        },
        classes_students: {
          include: {
            student: {
              select: {
                id: true,
                email: true,
                name: true,
              },
            },
          },
        },
      },
    });
  }
}

```

Fonte: *print screen* de trecho de código da aplicação no VSCode (2023)

3.3.2 Sprint 2

Na Sprint 2 foi planejado para a finalização do back-end e o início do front-end, sendo criada toda a estrutura básica e a arquitetura do front-end, sendo desenvolvidas as interfaces do TypeScript que estão relacionadas aos modelos do back-end. Nessa sprint também foram desenvolvidos os componentes básicos que serão reutilizados no sistema como menus, botões, tabelas, campos entre outros, além dos gerenciadores de autenticação, toasts e formulários.

Nas imagens a seguir pode-se ver os cadastros de professor e o cadastro de pergunta que foram desenvolvidos nessa sprint.

Figura 23 - Cadastro de professor

Cadastrar
Faça o seu cadastro para poder desfrutar da plataforma!

Nome

Email

Senha

Confirme a sua Senha

☐ Professor

CADASTRAR

[← VOLTAR PARA LOGON](#)

Fonte: *print screen* da aplicação (2023)

Figura 24 - Cadastro de perguntas

P.O.O

Atributos

Dados

Descrição

Palavras-chave

Peso
1,00

Ordem

NOVA RESPOSTA

☐

☐

☐

CADASTRAR **VOLTAR**

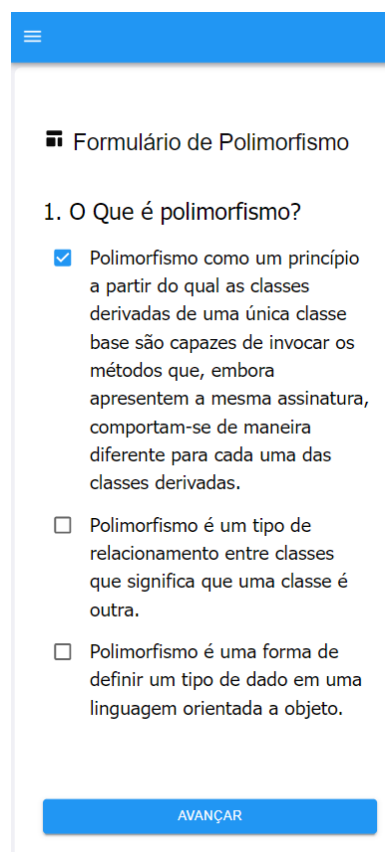
Fonte: *print screen* da aplicação (2023)

3.3.3 Sprint 3

A Sprint 3 tinha como objetivo finalizar os CRUDs básicos do sistema e a construção da parte de gamificação do aplicativo. Sendo entregue nessa sprint a parte de geração de formulários para a utilização dos alunos, criação da gamificação em si. Sendo a dificuldade da sprint a correção dos formulários corretamente, para a geração correta da nota.

Nas imagens a seguir temos algumas telas da gamificação, com pergunta, resposta e resultado.

Figura 25 - Questão a ser respondida



The screenshot shows a mobile application interface with a blue header bar containing a hamburger menu icon. Below the header, the title 'Formulário de Polimorfismo' is displayed. The question '1. O Que é polimorfismo?' is followed by three multiple-choice options. The first option is selected, indicated by a blue checkmark. At the bottom of the form, there is a blue button labeled 'AVANÇAR'.

☰

■ Formulário de Polimorfismo

1. O Que é polimorfismo?

- ☒ Polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas.
- ☐ Polimorfismo é um tipo de relacionamento entre classes que significa que uma classe é outra.
- ☐ Polimorfismo é uma forma de definir um tipo de dado em uma linguagem orientada a objeto.

AVANÇAR

Fonte: *print screen* da aplicação (2023)

Figura 26 - Pergunta respondida corretamente

☰

Formulário de Polimorfismo

1. O Que é polimorfismo?

☒ Polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas.

☐ Polimorfismo é um tipo de relacionamento entre classes que significa que uma classe é outra.

Parabéns, a resposta está correta!

Polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas.

ENVIAR

Fonte: *print screen* da aplicação (2023)

Figura 27 - Resultado final formulário

☰

Parabéns, você foi muito bem!

5.00
/5

FINALIZAR

Fonte: *print screen* da aplicação (2023)

3.3.4 Sprint 4

Na Sprint 4 tem-se como objetivo revisar, testar e validar o sistema por completo, como forma de garantir que as funcionalidades que foram pensadas e elaboradas durante todo o período de desenvolvimento, sendo repassados todos os cenários de testes como forma de garantia de qualidade. Nessa Sprint foram realizadas correções pelo sistema, como telas que redirecionavam novamente para o login, validações incorretas e alguns layouts inconsistentes, na seção de testes serão detalhados mais amplamente os resultados obtidos nos testes.

3.4 PROGRAMAÇÃO

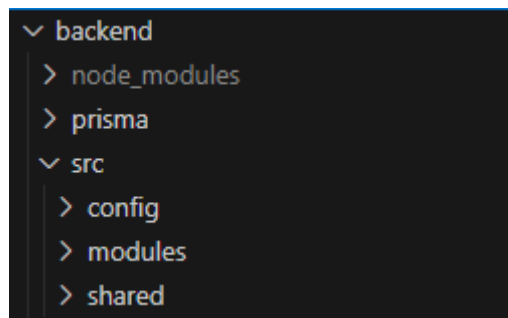
Nessa sessão apresenta-se sobre como o sistema foi estruturado e desenvolvido, separando-se em duas partes principais o back-end e front-end, que serão apresentados nas subseções a seguir.

3.4.1 Back-end

O back-end foi planejado e arquitetado com base no padrão de projeto MVC (Model View Controller), teve sua estrutura de pastas principais em três pastas sendo `node_modules`, `prisma` e `src`, onde cada pasta possui suas próprias particularidades. A `node_modules`, possui as dependências instaladas no projeto, a pasta `prisma` que possui as migrations e os schema do banco de dados, e a pasta `src` possui os arquivos do projeto em si.

A organização da pasta `src` ocorreu da seguinte forma, a pasta `config` é a pasta responsável por armazenar as configurações, a pasta `modules` possui os módulos onde cada um deles possui uma interação única no sistema, enquanto a pasta `shared`, possui arquivos compartilhados entre todos os módulos, na Figura 28 - Estrutura de pastas back-end, pode ser visto a estrutura supracitada.

Figura 28 - Estrutura de pastas back-end



Fonte: *print screen* da estrutura de pastas no VSCode (2023)

Para o desenvolvimento do projeto a estrutura foi planejada havendo a divisão em módulos, cada um possuindo um controller, um service, uma rota e um arquivo de validação, também havendo arquivos para tratamentos e autenticação dos diferentes grupos de usuário e um DTOs.

3.4.1.1 Controllers

Os controllers possuem em todos os seus métodos recebem uma request, instanciam os services através dos containers obtidos através do Tsyringe e chamam seus métodos correspondentes no service e retornam uma response com os resultados obtidos por ele. Na Figura 29 – Controller, pode-se ver o exemplo de uma classe Controller e os métodos padrões que as mesmas possuem e que todos os métodos recebem uma requisição e retornam um response.

Figura 29 - Controller

```

TS SubjectsController.ts X
backend > src > modules > university > infra > http > controllers > TS SubjectsController.ts > ...
4  import Controller from '@shared/core/Controller';
5
6  import SubjectsService from '@modules/university/services/SubjectsService';
7
8  import SubjectRequest from '../requests/SubjectRequest';
9
10 export default class SubjectsController extends Controller {
11
12   public async create(req: Request, res: Response): Promise<Response> {
13     const service = container.resolve(SubjectsService);
14     const result = await service.create(req.body);
15
16     return res.status(200).json(result);
17   }
18
19   public async find(req: Request, res: Response): Promise<Response> {
20     const service = container.resolve(SubjectsService);
21     const result = await service.findAll();
22
23     return res.status(200).json(result);
24   }
25
26   public async findOne(req: Request, res: Response): Promise<Response> {
27     const id = super.getIdParam(req);
28
29     const service = container.resolve(SubjectsService);
30     const result = await service.findOneFullData(id);
31
32     return res.status(200).json(result);
33   }
34
35   public async findByStudent(req: Request, res: Response): Promise<Response> {
36     const student_id = Number(req.params.student_id);
37
38     SubjectRequest.isTokenOwner(student_id, req);
39
40     const service = container.resolve(SubjectsService);
41     const result = await service.findByStudent(student_id);
42
43     return res.status(200).json(result);
44   }
45
46   public async update(req: Request, res: Response): Promise<Response> {
47     const id = super.getIdParam(req);
48
49     const service = container.resolve(SubjectsService);
50     const result = await service.update(id, req.body);
51
52     return res.status(200).json(result);
53   }
54
55   public async delete(req: Request, res: Response): Promise<Response> {
56     const id = super.getIdParam(req);
57     SubjectRequest.isTokenOwner(id, req);
58
59     const service = container.resolve(SubjectsService);
60     const result = await service.delete(id);
61
62     return res.json(result);
63   }
64 }
65

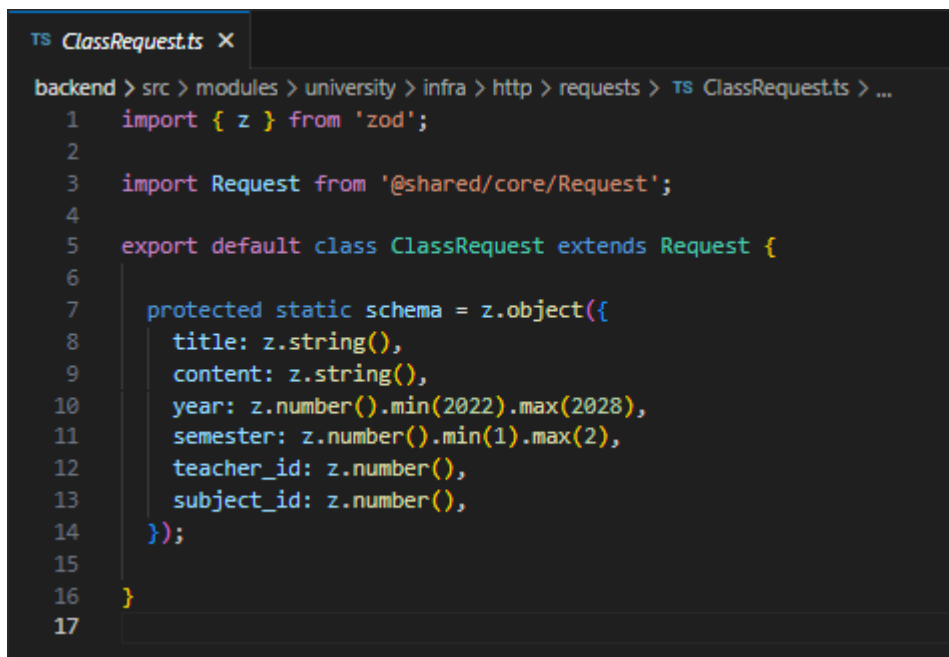
```

Fonte: Elaborado pelo autor (2023)

3.4.1.2 Validadores de request

Os validadores de request, são classes possuem como principal objetivo, validar os valores recebidos através da biblioteca Zod, através de um objeto que possui fácil legibilidade, assegurando assim que os dados que estão sendo recebidos possuem o formato esperado. Na Figura – 30 Validadores de request, pode-se ver um exemplo de classe de validação de requisição.

Figura 30 - Validadores de request



```
TS ClassRequest.ts X
backend > src > modules > university > infra > http > requests > TS ClassRequest.ts > ...
1  import { z } from 'zod';
2
3  import Request from '@shared/core/Request';
4
5  export default class ClassRequest extends Request {
6
7      protected static schema = z.object({
8          title: z.string(),
9          content: z.string(),
10         year: z.number().min(2022).max(2028),
11         semester: z.number().min(1).max(2),
12         teacher_id: z.number(),
13         subject_id: z.number(),
14     });
15
16 }
17
```

Fonte: Elaborado pelo autor (2023)

3.4.1.3 Services

As classes services possuem como principal responsabilidade, lidar com as regras de negócio da aplicação, nesse processo, podem fazer novas requisições, requisitar informações ao banco de dados, remover e aplicar máscaras, chamar os validadores de request, fazer a manipulação e tratamento dos dados. Na Figura 31 – Service, pode-se ver o exemplo de uma classe service, e como a mesma é estruturada.

Figura 31 - Service

```

18 SubjectsService.ts
backland > src > modules > university > services > 18 SubjectsService.ts > 18 SubjectsService
1 import { Subject } from '@prisma/client';
2 import { inject, injectable } from 'tsyringe';
3
4 import Service from '@shared/core/Service';
5 import client from '@shared/infra/prisma/client';
6
7 import SubjectRequest from '../infra/http/requests/SubjectRequest';
8
9 interface IRequest {
10   id?: number;
11   title: string;
12   content: string;
13 }
14
15 @injectable()
16 export default class SubjectsService extends Service {}
17
18 client = client.subject;
19
20 public async findById(id: number) {
21   const subject = await this.client.findFirst({ where: { id } });
22
23   return subject;
24 }
25
26 public async findAll(data: object = {}): Promise<Subject[]> {
27   const subjects = await super.findAll(data);
28
29   return subjects;
30 }
31
32 public async findByStudent(student_id: number) {
33   const subjects = await this.client.findMany({
34     include: {
35       classes: true,
36     },
37     where: {
38       classes: {
39         every: {
40           classes_students: {
41             every: {
42               student_id,
43             },
44           },
45         },
46       },
47     },
48   });
49
50   return subjects;
51 }
52
53 public async create(data: IRequest) {
54   data = super.removeMask(data) as IRequest;
55
56   let { title, content } = data;
57
58   await SubjectRequest.create({ title, content });
59
60   const subject = await this.client.create({
61     data: {
62       title,
63       content,
64     },
65   });
66
67   return subject;
68 }
69
70 public async update(id: number, data: IRequest) {
71   data = super.removeMask(data) as IRequest;
72
73   let { title, content } = data;
74
75   await SubjectRequest.update({ id, title, content });
76
77   const subject = await this.client.update({
78     data: {
79       title,
80       content,
81     },
82     where: { id },
83   });
84
85   return subject;
86 }
87

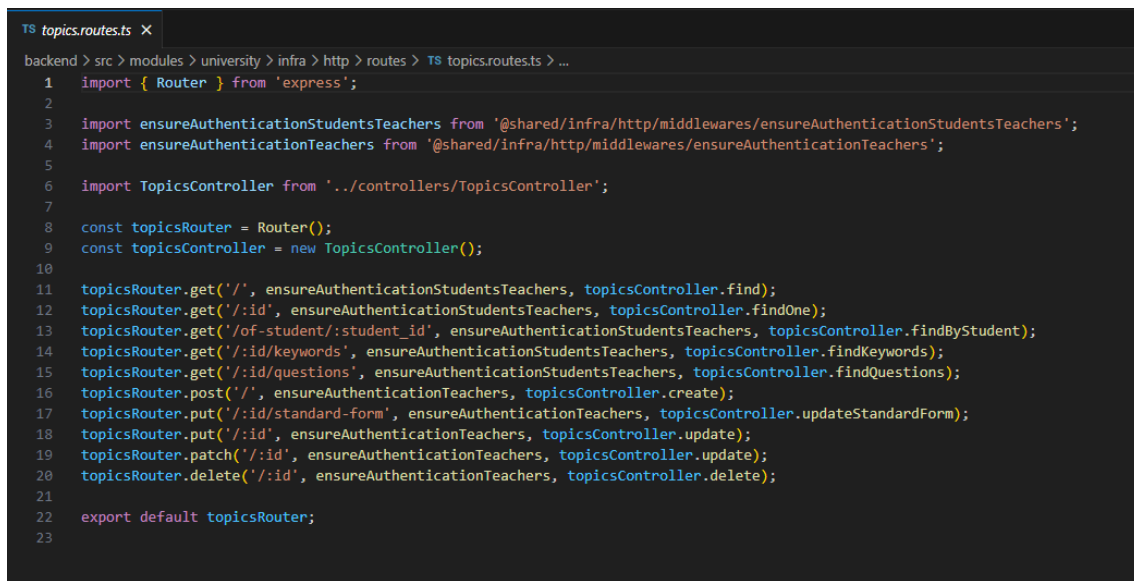
```

Fonte: Elaborado pelo autor (2023)

3.4.1.4 Rotas

O arquivo de rotas define as rotas relacionadas ao módulo em questão, as rotas são definidas com uma URL, com um método HTTP, um método controller que realiza o tratamento dessa rota, e os middlewares que validarão a rota e os níveis de acesso da mesma. Na Figura 32 – Rotas, é apresentado o arquivo de rotas que está relacionada a parte de níveis e formulários do sistema.

Figura 32 - Rotas



```

TS topics.routes.ts X
backend > src > modules > university > infra > http > routes > TS topics.routes.ts > ...
1  import { Router } from 'express';
2
3  import ensureAuthenticationStudentsTeachers from '@shared/infra/http/middlewares/ensureAuthenticationStudentsTeachers';
4  import ensureAuthenticationTeachers from '@shared/infra/http/middlewares/ensureAuthenticationTeachers';
5
6  import TopicsController from '../controllers/TopicsController';
7
8  const topicsRouter = Router();
9  const topicsController = new TopicsController();
10
11 topicsRouter.get('/', ensureAuthenticationStudentsTeachers, topicsController.find);
12 topicsRouter.get('/:id', ensureAuthenticationStudentsTeachers, topicsController.findOne);
13 topicsRouter.get('/of-student/:student_id', ensureAuthenticationStudentsTeachers, topicsController.findByStudent);
14 topicsRouter.get('/:id/keywords', ensureAuthenticationStudentsTeachers, topicsController.findKeywords);
15 topicsRouter.get('/:id/questions', ensureAuthenticationStudentsTeachers, topicsController.findQuestions);
16 topicsRouter.post('/', ensureAuthenticationTeachers, topicsController.create);
17 topicsRouter.put('/:id/standard-form', ensureAuthenticationTeachers, topicsController.updateStandardForm);
18 topicsRouter.put('/:id', ensureAuthenticationTeachers, topicsController.update);
19 topicsRouter.patch('/:id', ensureAuthenticationTeachers, topicsController.update);
20 topicsRouter.delete('/:id', ensureAuthenticationTeachers, topicsController.delete);
21
22 export default topicsRouter;
23

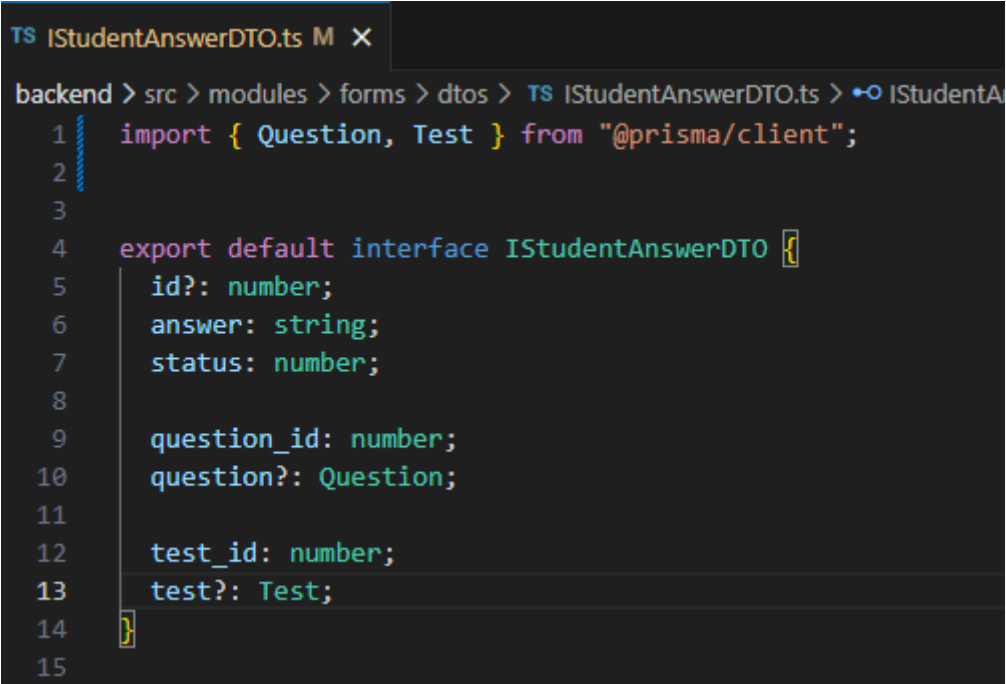
```

Fonte: *print screen* de trecho de código da aplicação no VSCode (2023)

3.4.1.5 DTOs

Os DTOs (Data Transfer Object), são responsáveis por facilitar a comunicação e transferência de dados entre os componentes do sistema, atuando como estruturas de dados simples e neutras, contendo apenas os campos necessários para transmitir as informações relevantes de um objeto ou entidade. Na Figura 33 – DTOs temos o exemplo de uma classe de DTO e a estrutura da mesma.

Figura 33 - DTOs



```

TS IStudentAnswerDTO.ts M X
backend > src > modules > forms > dtos > TS IStudentAnswerDTO.ts > IStudentA
1  import { Question, Test } from "@prisma/client";
2
3
4  export default interface IStudentAnswerDTO {
5      id?: number;
6      answer: string;
7      status: number;
8
9      question_id: number;
10     question?: Question;
11
12     test_id: number;
13     test?: Test;
14 }
15

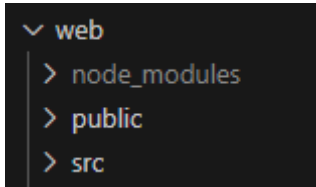
```

Fonte: *print screen* de trecho de código da aplicação no VSCode (2023)

3.4.2 Front-end

Em relação a estrutura do front-end, a separação de pasta acontece da seguinte maneira, a pasta raiz possui o `node_modules`, que possui as dependências instaladas no projeto, a pasta `public` que possui os arquivos compartilhados por todo o projeto, e a pasta `src` possui os arquivos do projeto em si.

Figura 34 - Estrutura de pastas front-end



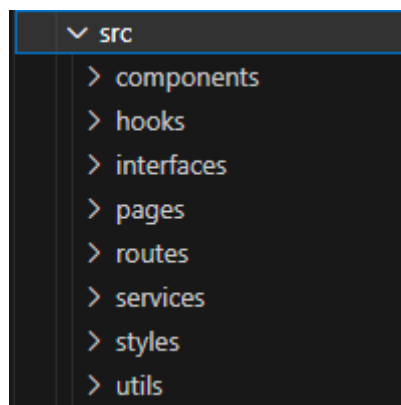
```

web
├── node_modules
├── public
└── src

```

Fonte: *print screen* da estrutura de pastas no VSCode (2023)

Figura 35 - Pasta src front-end



Fonte: *print screen* da estrutura de pastas no VSCode (2023)

A figura 35, demonstra a estrutura da pasta src, que segue o seguinte padrão:

- Components: Possui os componentes do sistema como campos, botões, menus, entre outros componentes;
- Hooks: Possui a partes de autenticação, controle de formulários, controle de emissão dos toasts etc;
- Interfaces: Possui as interfaces comuns do sistema, e possui interfaces relacionadas aos modelos do back-end;
- Pages: Possui as páginas do sistema;
- Routes: Possui, manipula e valida as rotas do sistema;
- Services: Realiza a comunicação com a api do sistema;
- Styles: Possui a parte de CSS e estilos compartilhados;
- Utils: Funções básicas de uso geral.

3.5 TESTES

Nesta seção serão apresentados os testes realizados, a fim de assegurar e validar os fluxos e comportamentos esperados do sistema assim como seus atores. Realizou-se o atrelamento dos testes funcionais com as User Stories da Seção 3.1.7 – User Stories, conforme temos na tabela 9 – Testes realizados.

Tabela 9 – Testes realizados

Id	Teste	User Stories	Resultado
1	Cadastrar-se no sistema como aluno.	US06	✓
2	Cadastrar-se no sistema como professor.	US01	✓
3	Cadastrar perguntas no sistema.	US23	✓
4	Responder o formulário.	US18	✓
5	Responder formulários de dúvidas.	US34	✓
6	Ver ranqueamento dos alunos.	US05	✓
7	Cadastrar níveis	US19	✓
8	Logar no sistema como aluno.	US06	✓
9	Logar no sistema como professor.	US01	✓

Fonte: Elaborado pelo autor (2023)

4 RESULTADOS E DISCUSSÕES


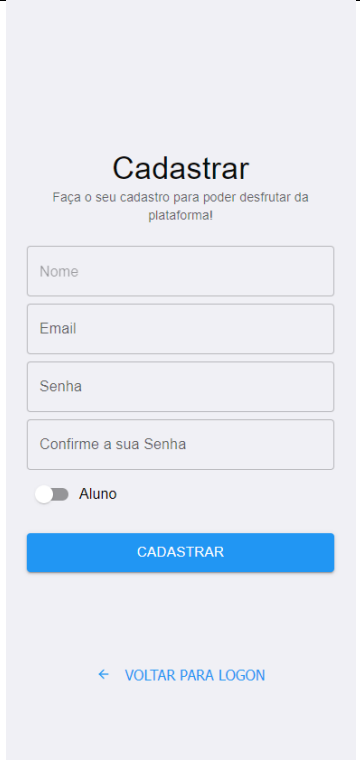
Nesta seção é apresentado os resultados construídos ao decorrer do projeto, trazendo a aplicação final construída, os resultados obtidos, discussões e trabalhos futuros.

4.1 APLICAÇÃO MOBILE GAMIFICADA

Ao longo deste projeto buscou-se realizar o processo de desenvolvimento de uma aplicação mobile para a ajuda no aprendizado de orientação de objetos. A aplicação final buscou seguir e cumprir os seus requisitos planejados, regras de negócio, casos de uso e user stories.

Como resultado final obtido desse desenvolvimento, tem-se a aplicação demonstrada na Tabela 10 – Aplicação final, contendo imagens de todas as partes da aplicação como logins, cadastros de usuários, resolução de formulários, rankings e exibição de feedbacks, etc.

Tabela 10 - Aplicação final

 <p>1 – Tela de Login</p>	 <p>2 – Tela de cadastro de usuário</p>
--	---

TCC

Dados

Nome
professor

Email
professor@hotmail.com

Senha atual

Senha

Confirme a sua senha



EDITAR VOLTAR

3 – Tela de edição de perfil

TCC

CADASTRAR VOLTAR

Matérias

Título	Conteúdo	Ações
Programação 1	Orientação a Objetos	 

Mostrar 10 1-1 de 1 < >

☐ Concentrado

4 – Tela de consulta de matérias

TCC

Dados

Título

Conteúdo





CADASTRAR VOLTAR

5 – Tela de cadastro de matérias

TCC

CADASTRAR VOLTAR

Turmas

Título	Período	Professor	
Programação 1	2023/1	professor	 
Programação 2	2023/1	professor	 

Mostrar 10 1-2 de 2 < >

☐ Concentrado

6 -Tela de Consulta de Turma

TCC

Dados

Título

Conteúdo

Ano

Semestre

Matéria

Professor

Estudantes

CADASTRAR VOLTAR

7 – Tela de cadastro de turma

TCC

CADASTRAR NÍVEL VOLTAR

Níveis

Descrição	Classe	
Introdução a orientação a objetos	Programação 1	PALAVRAS CHAVES
Aula 1	Programação 1	PALAVRAS CHAVES
Aula 2	Programação 1	PALAVRAS CHAVES
Aula 3	Programação 1	PALAVRAS CHAVES
Aula 4	Programação 1	PALAVRAS CHAVES

Mostrar 10 1-5 de 5 < >

☐ Concentrado

8 – Tela de consulta de nível

TCC

Dados

Descrição

Classe

Ordem

CADASTRAR VOLTAR

9 – Tela de cadastro de nível

TCC

Palavras-chave

Descrição	Quantidade dúvidas ↑	Ações
Classe	4	
Decomposição	2	
Herança	3	
Polimorfismo	2	
Objetos	2	
Atributos	1	
Métodos	1	
Interface	1	

Mostrar 10 1-8 de 8 < >

☐ Concentrado

10 – Tela de palavras chaves

Introdução a orientação a objetos

Dados

Descrição

CADASTRAR VOLTAR

11 – Cadastro de palavras chaves

Introdução a orientação a objetos

CADASTRAR VOLTAR

Questões

Descrição	Peso	Palavra-chaves
Considere que um Analista de TI sabe que uma class...	3	Classe, Herança, Decomposição
Os quatro pilares do paradigma de Orientação a Obj...	6	Polimorfismo, Herança, Classe
Quanto aos conceitos do paradigma da orientação a ...	8	Interface, Objetos, Decomposição

12 – Tela de consulta de questões

Dados

Descrição

Palavras-chave

Peso 1,00

Ordem

NOVA RESPOSTA

☐

☐

☐

CADASTRAR VOLTAR

13 – Tela de cadastro de questões

Formulário Ativo!

Título Formulário de Introdução a orientaçê

Description Formulário referente ao tópico de Int

Considere que um Analista de TI sabe que uma classe Pessoa F...

Os quatro pilares do paradigma de Orientação a Objetos são:

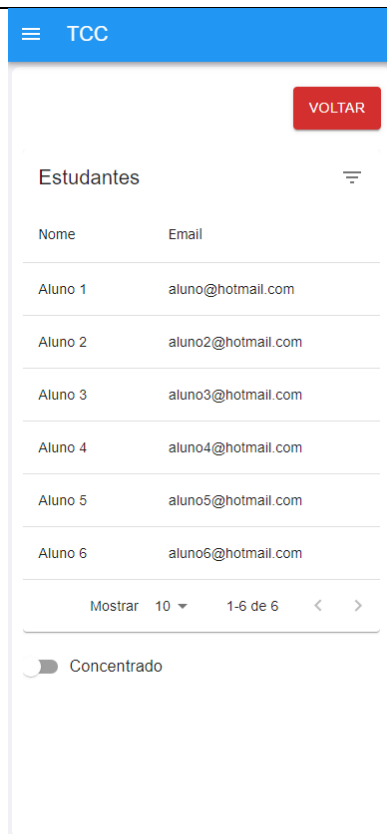
Quanto aos conceitos do paradigma da orientação a objetos, é...

Polimorfismo é um conceito usado em programação orientada a ...

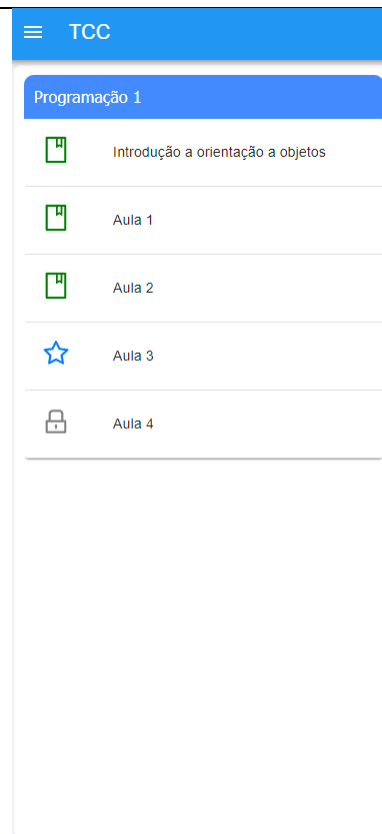
Considere as seguintes afirmações sobre alguns fundamentos d...

GERAR VOLTAR

14 – Tela de geração de formulário



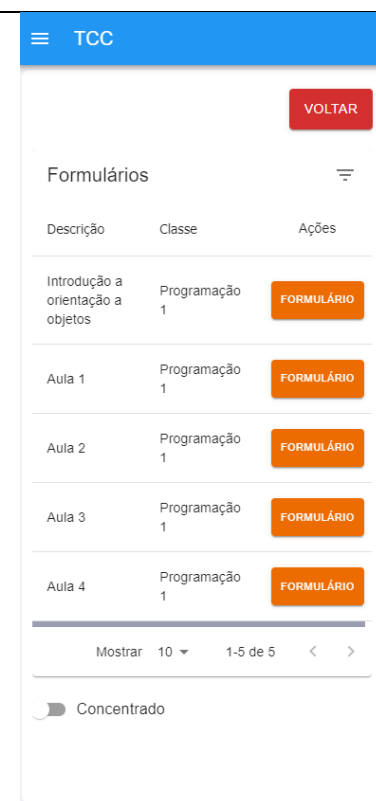
15 – Tela de listagem de estudantes



16 – Tela inicial estudante



17 – Tela de ranking dos alunos



18 – Tela de seleção de geração de formulário de dúvidas

TCC

Dados

Título

Description

Quantidade de questões

Palavras-chave

GERAR VOLTAR

19 – Tela de geração de formulário de dúvidas

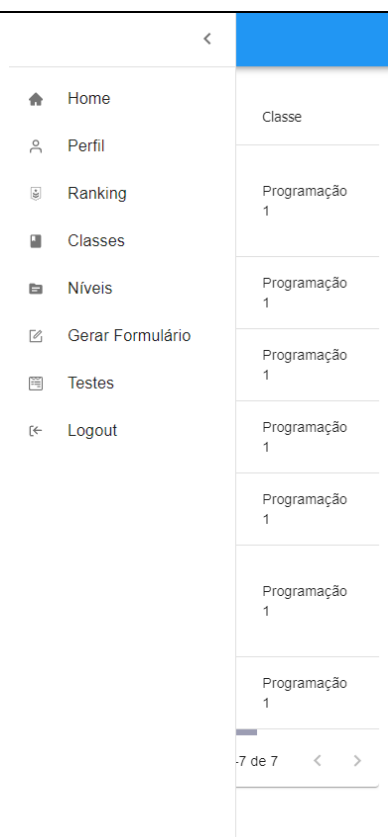
TCC

Formulário	Nível	Classe
Formulário de Introdução a orientação a objetos	Introdução a orientação a objetos	Programação 1
Formulário de Aula 1	Aula 1	Programação 1
Formulário de Aula 2	Aula 2	Programação 1
Formulário de Aula 3	Aula 3	Programação 1
Formulário de Aula 4	Aula 4	Programação 1
Dúvidas Introdução	Introdução a orientação a objetos	Programação 1
Duvidas da aula 1	Aula 1	Programação 1

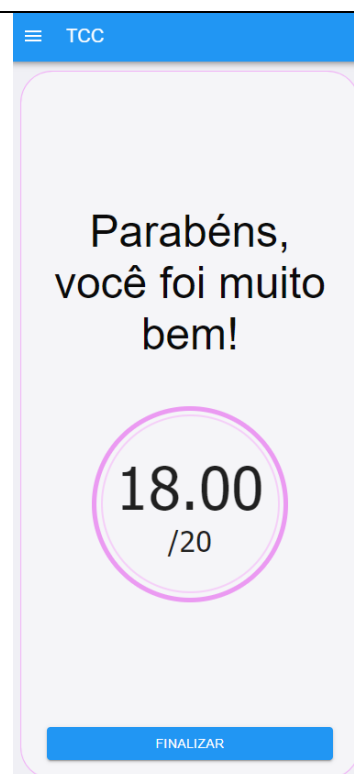
Mostrar 10 1-7 de 7

Concentrado

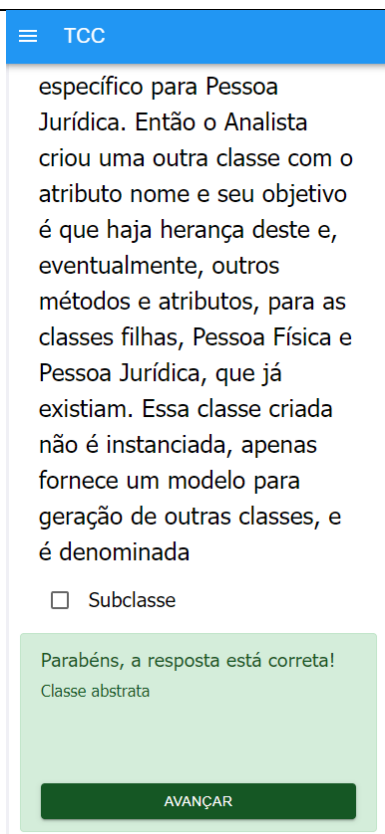
20 – Tela de seleção de formulário de dúvida



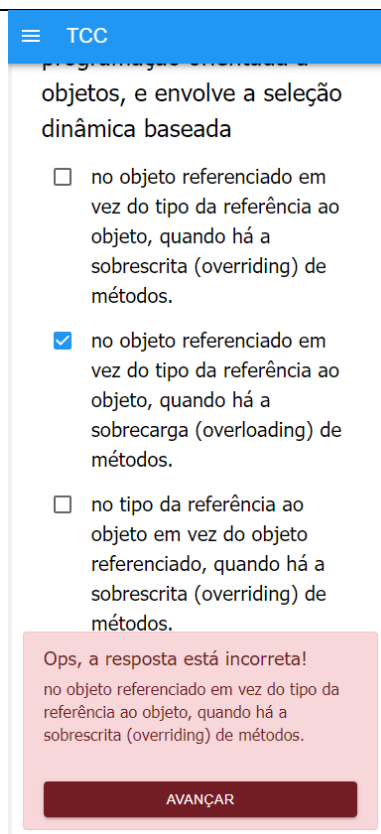
21 – Menu de ações do usuário



22 – Tela de feedback da resolução do formulário



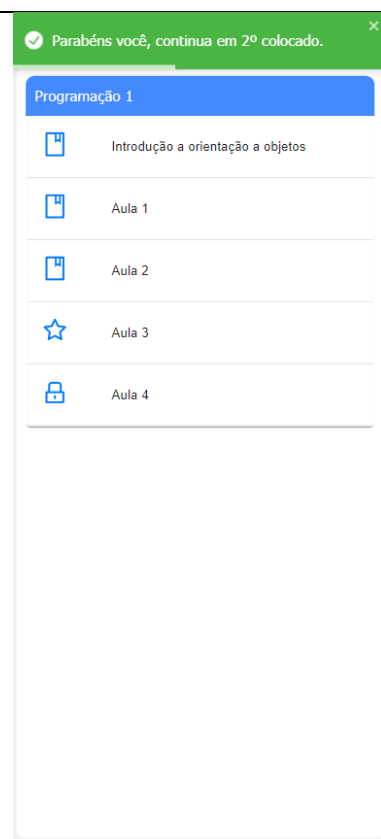
23 – Tela de resposta correta



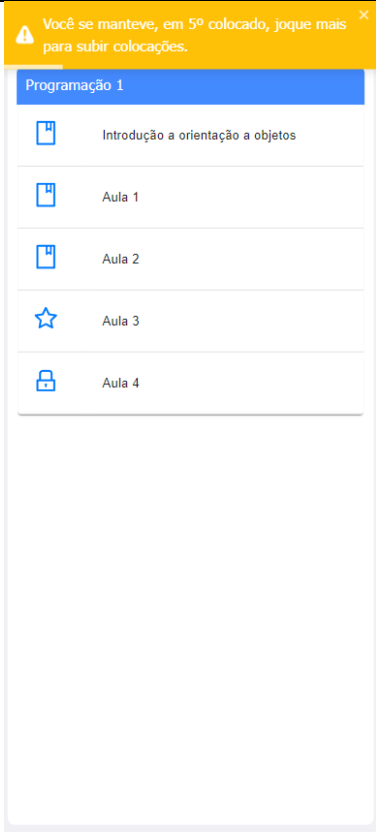
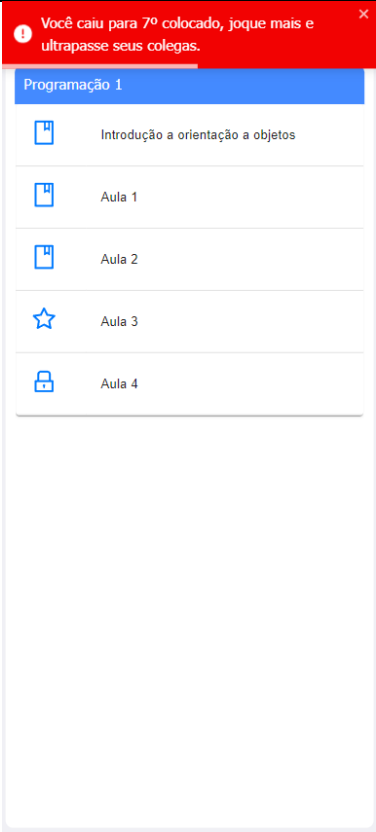
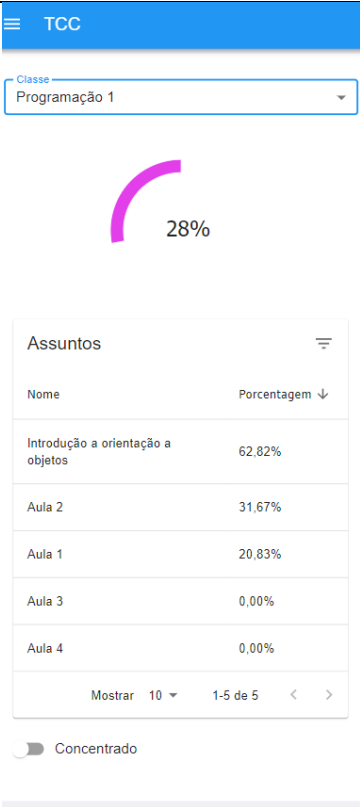
24 – Tela de resposta incorreta



25 – Tela de medalhas



26 – Notificação entre os 3 melhores

 <p>27 – Notificação manteve posição</p>	 <p>28 – Notificação caiu posição.</p>
 <p>29 – Dashboard turma</p>	

4.2 RESULTADOS DA VALIDAÇÃO COM FORMULÁRIO SUS

Para realizar a avaliação da usabilidade da ferramenta foi utilizado o método *System Usability Scale* (SUS), verificando três diferentes aspectos da interface: efetividade, eficiência e satisfação. O método possui um questionário com 10 afirmações, o qual foi enviado aos usuários, apresentando opções de resposta em escala linear de 1 (Discordo totalmente) a 5 (Concordo totalmente).

O questionário foi aplicado com pessoas entre 20 e 45 anos de idade que trabalham ou estudam na área de programação, tendo no final 30 respostas o questionário, conforme a Tabela 11 – Resultados do questionário.

Tabela 11 - Resultados do questionário

Afirmação	1	2	3	4	5
1 - Eu achei o sistema fácil de usar	2	2	3	7	16
2 - Eu achei que precisava de ajuda de um profissional para usar o sistema	17	10	3	-	-
3 - As diferentes funcionalidades do sistema foram bem integradas	-	1	2	9	18
4 - Eu achei o sistema muito complexo	15	10	5	-	-
5 - Eu achei que o sistema foi fácil de aprender	1	2	2	7	18
6 - Eu achei que precisava de muito esforço para usar o sistema	16	12	-	1	1
7 - As informações apresentadas no sistema foram fáceis de entender	1	1	5	6	17
8 - Eu achei que o sistema foi organizado de forma confusa	17	9	3	-	1
9 - Eu me senti confiante usando o sistema	-	1	2	6	21
10 - Eu precisaria de treinamento para ser capaz de usar todas as funcionalidades do sistema	16	7	5	2	-

Fonte: Elaborado pelo autor (2023)

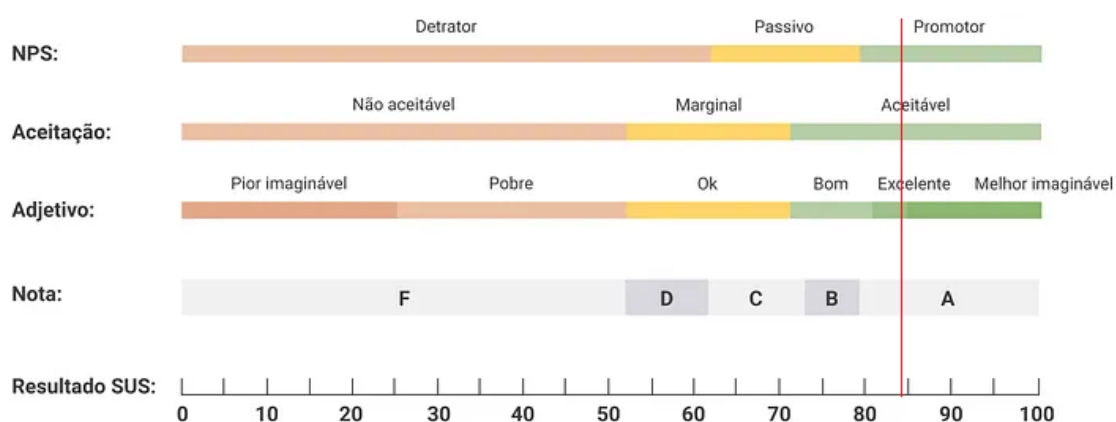
Para realizar o cálculo dos resultados, Brooke (1996) sugere que os seguintes passos sejam realizados para obter uma pontuação:

- Nas perguntas ímpares é subtraído 1 (um) da pontuação do usuário.
- Nas perguntas pares é subtraído 5 (cinco) da pontuação do usuário.
- Todos os valores são somados e multiplicados por 2,5, tendo uma resposta que varia entre 0 (zero) e 100 (cem). Tendo uma pontuação média de 83,58.

4.3 DISCUSSÕES

Para a análise dos resultados obtidos, foi utilizada a tabela de comparação onde pode-se ver a taxa de aceitação, o adjetivo e a nota de usabilidade alcançada pela aplicação.

Figura 36 - NPS, taxa de aceitação, adjetivos e nota associados ao resultado do SUS.



Fonte: <https://brasil.uxdesign.cc/> (2023)

Como pode-se verificar na tabela da Figura 36 - NPS, taxa de aceitação, adjetivos e nota associados ao resultado do SUS, a aplicação alcançou um NPS promotor, com aceitação resultando em aceitável, com adjetivo excelente e resultando em uma nota A.

Tendo em vista os pontos supracitados anteriormente, verifica-se o alto grau de satisfação e usabilidade do sistema, onde o mesmo busca trazer uma experiência simples e prática ao usuário, com recursos de gamificação como

forma de auxílio no aprendizado, e ainda permitindo ao professor verificar as dúvidas dos alunos referentes aos conteúdos, ajudando na forma como o mesmo irá prosseguir com os conteúdos lecionados.

Sobre a experiência do autor no desenvolvimento desse TCC, o presente trabalho ajudou a compreender melhor e aplicar processos da Engenharia de Software que foram apresentados e estudados ao longo de toda a graduação. O processo da aplicação da Engenharia de Software na aplicação demanda tempo e esforço, porém traz consigo a melhor assertividade e robustez do produto final desenvolvido. Algumas etapas do processo podem muitas vezes aparentar simplicidade, mas demandam tempo e conhecimento para realização, desde o levantamento dos requisitos e casos de uso do sistema, elaboração das user storie, escolhas de tecnologias que atendessem à necessidade do projeto, metodologias ágeis de desenvolvimento, testes e validação da aplicação.

5 CONSIDERAÇÕES FINAIS

Com a análise de trabalhos correlatos por meio da tabela comparativa, verificando as principais funcionalidades do software desenvolvido neste trabalho, é possível concluir que a utilização tanto dos aplicativos de ensino e da gamificação vem sendo aplicados para o aprendizado de diferentes tipos de conteúdo, podendo ser utilizado como uma forma de adquirir conhecimento, e como uma forma de fixação dele.

O presente trabalho buscou além de trazer a gamificação no processo de ensino e aprendizado dos alunos, como também trazer o professor como utilizador da ferramenta, permitindo um melhor controle e assertividade dos conteúdos a serem disponibilizados, permitindo a ele verificar as maiores dúvidas dos alunos relacionadas a cada conteúdo. A ferramenta desenvolvida buscou manter a flexibilidade da criação de formulários, permitindo ao professor construí-los conforme os conteúdos aplicados em sala de aula, tendo como objetivo inicial apoiar o ensino da POO, a ferramenta pode ser utilizada em outras disciplinas.

O desenvolvimento da aplicação ocorreu utilizando de algumas diferentes tecnologias, para a parte do back-end utilizou-se Node.js junto do framework Prisma, com banco de dados utilizando PostgreSQL, enquanto para a parte de front-end utilizou-se HTML, CSS, JavaScript e a biblioteca de React.js, uma vez que o autor possui base de conhecimentos das tecnologias diminuindo a curva de aprendizagem para o desenvolvimento. A ferramenta foi avaliada sob a ótica do formulário SUS por 30 pessoas e resultou em score final de 83,58 em 100, o que significa a aplicação atingiu níveis ótimos de usabilidade, conforme visto na figura 36.

Para o êxito do desenvolvimento da proposta deste trabalho, foi preciso a aplicação de princípios do processo da Engenharia de Software, com o objetivo, de especificar, analisar e desenvolver o projeto, além do uso de metodologias ágeis para um desenvolvimento mais fluido da aplicação, demonstrado principalmente na seção 3, deste projeto. Portanto verifica-se na prática, que apesar dos custos que a utilização da Engenharia de Software tem, os benefícios resultantes do seu uso, além de especificações com melhor detalhamento,

adicionam robustez ao processo de desenvolvimento, resultando em uma aplicação sólida, e consistente. Portanto, garantindo a qualidade da aplicação, e suprimindo os níveis de exigência esperados, conforme pode ser visto no resultado da avaliação pelo SUS.

5.1 TRABALHOS FUTUROS

Como trabalhos futuros, pode-se evoluir a aplicação para aprendizado de orientação a objetos, aprimorando e adicionando novas funcionalidades ao software. Podendo-se desenvolver a aplicação em linguagens específicas para o desenvolvimento mobile como Flutter e ReactNative, trazendo assim a facilidade para o adicionar notificações a aplicação quando algum conteúdo é adicionado, lembretes estimulando o uso, avisos que o estudante caiu de posição.

Pensando em evolução da aplicação, para resultar em uma melhor visibilidade do desempenho dos alunos para o professor, poderia haver dashboards detalhados do desempenho dos alunos a cada conteúdo, e o desempenho individual, facilitando ao professor aplicar metodologias diferentes e revisar os conteúdos com maior necessidade.

Em recursos de gamificação, poderia haver perguntas com diferentes tipos de respostas além das questões de escolha simples, recursos de tempo, podendo-se gerar os formulários de dúvida utilizando-se Data-mining para melhorar a assertividade das perguntas a serem apresentadas.

6 REFERÊNCIAS

AGUIAR, J. J. B. **Experiência baseada em Gamificação no Ensino sobre**. Universidade Federal de Campina Grande. Campina Grande, p. 10. 2015.

BANGO, R. **Learning Node.js: A Hands-On Guide to Building Web Applications in JavaScript**. [S.l.]: Addison-Wesley Professional., 2015.

BARANAUSKAS, M. C. C. **PROCEDIMENTO, FUNÇÃO, OBJETO OU LÓGICA?** Campinas, SP: Computadores e Conhecimento: Repensando a Educação, 1993.

BECK, K. E. A. **Manifesto para desenvolvimento ágil de software**. [S.l.]: [s.n.], 2001. Disponível em: <<https://agilemanifesto.org/iso/ptbr/manifesto.html>>. Acesso em: 21 abr. 2023 .

BROOKE. **SUS**: A quick and dirty usability scale. London: [s.n.], 1996.

BROWN, B. **Node.js, MongoDB, and AngularJS Web Development**. [S.l.]: Addison-Wesley Professional, 2014.

BROWN, M. **Node.js, MongoDB, and AngularJS Web Development**. [S.l.]: Addison-Wesley Professional, 2014.

COCKBURN, A. **Agile Software Development: The Cooperative Game**. 2ª. ed. Boston: Pearson, 2007.

COMPONENTS and Props. **React**. Disponível em: <<https://reactjs.org/docs/components-and-props.html>>. Acesso em: 21 abr. 2023.

COSTA, A. F. F. et al. **Aplicação de sala invertida e elementos de gamificação para melhoria do ensino-aprendizagem em programação orientada a objetos**. Santiago de Chile: TISE, v. 13, 2017.

CYSNEIROS, L. M. **Requisitos Não Funcionais: Da Elicitação ao**. PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO. [S.l.], p. 19. 2001.

DUCKETT, J. **JavaScript and JQuery: Interactive Front-End Web Development**. [S.l.]: John Wiley & Sons, 2014.

DUOLINGO. Duolingo. **Duolingo**, 2022. Disponível em: <<https://pt.duolingo.com/>>. Acesso em: 10 nov. 2022.

DURÃES, A. D. O. **Ferramenta para o auxílio da aprendizagem no paradigma de Orientação a Objetos**. Universidade Federal de Uberlândia. Monte Carmelo, p. 49. 2021.

FACEBOOK. A JavaScript library for building user interfaces. **React**, 2023. Disponível em: <<https://reactjs.org/>>. Acesso em: 20 abr. 2023.

FARINELLI, F. **CONCEITOS BÁSICOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS**. Instituto Federal Sudeste de Minas Gerais. [S.l.]. 2007.

FERNANDES, L. T. V. **React Native**: desenvolvimento de aplicações móveis multiplataforma com o framework do Facebook. Cuiabá: Universidade Federal de Mato Grosso, 2019.

FIGUEIREDO, M.; PAZ, T.; JUNQUEIRA, E. Gamificação e educação: um estado da arte das pesquisas. **Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação**, Fortaleza - CE, 2015. 1155.

FLANAGAN. **The Definitive Guide**: Activate Your Web Pages. [S.l.]: O'Reilly Media, 2011.

FLANAGAN, D. **JavaScript for Web Developers**: Working with JavaScript Libraries. [S.l.]: O'Reilly Media, 2017.

FREEMAN, A. . & R. A. **Pro Express.js**: Master Express.js: The Node.js Framework for Your Web Development. [S.l.]: Apress, 2018.

GOMES, A. C. C. et al. Uma revisão sistemática sobre o framework React Native. **XII Simpósio Brasileiro de Sistemas de Informação**, 2019. 33-42.

GOMES, A. C. C. et al. Uma revisão sistemática sobre o framework React Native. **XII Simpósio Brasileiro de Sistemas de Informação**, 17 mar. 2023. 33-42.

GONÇALVES, J. P.; OLIVEIRA, R. L. **TypeScript**: Uma alternativa mais segura ao JavaScript. [S.l.]: [s.n.], v. 3, 2020.

GRASSHOPPER. **Grasshopper**, 2022. Disponível em: <https://grasshopper.app/pt_br/>. Acesso em: 12 nov. 2022.

GRIGORIK, I. **High Performance Browser Networking**: What every web developer should know about networking and web performance. [S.l.]: O'Reilly Media, 2013.

HIGHSMITH, J. **Agile Software Development Ecosystems**. Boston: Addison-Wesley, 2002.

HOSKINS, A. **React.js Essentials**. [S.I.]: Packt Publishing, 2019.

LABS, P. Prisma.js - The Database Toolkit for Node.js. **Prima**. Acesso em: 21 abr. 2023.

LEARNU. **LearnU**, 2022. Disponível em: <<https://www.larnu.com/>>. Acesso em: 12 nov. 2022.

MAFFETT, I. **Beginning Node.js, Express & MongoDB Development**. [S.I.]: Apress, 2017.

MARQUES, R. TypeScript: uma linguagem mais produtiva para desenvolvedores JavaScript. **IBM Developer**, 2019. Disponível em: <<https://developer.ibm.com/br/languages/typescript/>>. Acesso em: 17 mar. 2023.

MARTIN, R. **Agile Software Development: Principles, Patterns, and Practices**. [S.I.]: Pearson Education, 2002.

MATERIAL-UI. Material-UI - React UI Framework. **Material-UI**, 2023. Disponível em: <<https://mui.com/>>. Acesso em: 21 abr. 2023.

MCPEAK, T. **JavaScript: Novice to Ninja**. [S.I.]: SitePoint Pty. Ltd, 2015.

MELO, R. D. S.; CARVALHO, M. J. S. APLICATIVOS EDUCACIONAIS LIVRES PARA MOBILE LEARNING. **Anais do Encontro Virtual de Documentação em Software Livre e Congresso Internacional de Linguagem e Tecnologia Online**, 1 Junho 2014.

MOZILLA DEVELOPER NETWORK. Web Apps - MDN Web Docs. **Mozilla Developer Network**, 2023. Disponível em: <<https://developer.mozilla.org/en-US/docs/Web/Apps>>. Acesso em: 21 abr. 2023.

NASCIMENTO, J. S.; SANTANA, R. A. Implementação de replicação de dados em PostgreSQL. **Revista de Tecnologia da Informação e Comunicação**, 5, 2017. 1-8.

OLIVEIRA, D. A. S. E. A. Uma análise comparativa entre os frameworks React Native e Flutter para o desenvolvimento de aplicações móveis. **Anais do XVIII Simpósio Brasileiro de Computação Ubíqua e Pervasiva**, 2019.

OLIVEIRA, P. H. P. E. A. Uma análise comparativa entre o desenvolvimento em JavaScript e TypeScript. **Anais do Congresso Nacional de Informática**, 2021.

PEREIRA, F. G. F. CONSTRUÇÃO E VALIDAÇÃO DE APLICATIVO DIGITAL PARA ENSINO DE INSTRUMENTAÇÃO CIRÚRGICA. **Cogitare Enfermagem**, 24 março 2019.

PEREIRA, R. M. S. **Gamificação como Solução para os Problemas da Aprendizagem da Programação**. Instituto Universitário de Lisboa. Lisboa, p. 73. 2017.

PRISMA LABS. Prisma.js - The Database Toolkit for Node.js. **Prisma.js**, 2023. Disponível em: <<https://www.prisma.io/>>. Acesso em: 21 abr. 2023.

REACT. Components and Props. **React**, 2023. Disponível em: <<https://reactjs.org/docs/components-and-props.html>>. Acesso em: 21 abr. 2023.

REACT. Optimizing Performance. **React**, 2023. Disponível em: <<https://reactjs.org/docs/optimizing-performance.html>>. Acesso em: 21 abr. 2023.

REACT. Components and Props. **React**, <https://reactjs.org/docs/components-and-props.html>. Acesso em: 21 abr. 2023.

REDUX. Redux - A Predictable State Container for JavaScript Apps. **Redux**, 2023. Disponível em: <<https://redux.js.org/>>. Acesso em: 21 abr. 2023.

RICARTE, I. L. M. **Programação Orientada a Objetos: Uma Abordagem com Java**. UNIVERSIDADE ESTADUAL DE CAMPINAS. Campinas, p. 118. 2001.

ROUTER, R. React Router - Declarative Routing for React.js. **React Router**, 2023. Disponível em: <<https://reactrouter.com/>>. Acesso em: 21 abr. 2023.

SABIN-WILSON, L. **WordPress All-In-One for Dummies**. [S.l.]: John Wiley & Sons, 2018.

SADYKOV, R.; KANTARBAYEVA, A. PostgreSQL como um banco de dados confiável e escalável. **ResearchGate**, 2020. Disponível em: <https://www.researchgate.net/publication/343648177_PostgreSQL_como_um_banco_de_dados_confialvel_e_escavel>. Acesso em: 17 mar. 2023.

SCARPINO, R. O que é TypeScript. **Devmedia**, 2018. Disponível em: <<https://www.devmedia.com.br/o-que-e-typescript/37457>>. Acesso em: 17 mar. 2023.

SCHWABER, K. & B. M. **Agile Software Development with Scrum**. [S.l.]: Prentice Hall, 2002.

SILVA, E. A. Integração de bibliotecas externas em aplicações móveis com o framework React Native. **Anais do Congresso Brasileiro de Informática. Sociedade Brasileira de Computação**, 2020. 234-245.

SOMMERVILLE, I. **Engenharia de Software**. 9ª. ed. São Paulo: Escola Politécnica da Universidade de São Paulo (EPUSP)., 2011.

SOUSA, M. A. M. E. A. **Análise da utilização do TypeScript em projetos open source**. [S.l.]: [s.n.], 2020.

TAXA, F. D. O. S. et al. **DA FORMAÇÃO CONTINUADA À EXTENSÃO: INTERFACES DA GAMIFICAÇÃO EM PROJETOS UNIVERSITÁRIOS**. PUC. CAMPINAS, p. 2. 2018.

WILSON, D. **Node.js 8**. [S.l.]: [s.n.], 2018.

ZAKAS, N. C. **Understanding ECMAScript 6**: The Definitive Guide for JavaScript Developers. [S.l.]: No Starch Press, 2016.

ZICBERMANN, G.; CUNNINGBAM, C. **Gamification by Design**: Implementing Games Mechanics in Web and Mobile Apps. Sebastopol: O'Reilly Media, Inc, 2011.