Ferramenta para o auxílio da aprendizagem no paradigma de Orientação a Objetos

Amanda Dias Oliveira Durães

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

Monte Carmelo - MG
2021

Amanda Dias Oliveira Durães

Ferramenta para o auxílio da aprendizagem no paradigma de Orientação a Objetos

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

Área de concentração: Ciência da Computação

Orientadora: Alessandra Aparecida Paulino

Monte Carmelo - MG
2021

Este trabalho é dedicado às crianças adultas que, quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

A minha mãe por sempre acreditar em mim, me apoiar e pelo seu amor incondicional. A Deus por me proporcionar saúde, paciência e motivação para seguir os meus sonhos. A minha orientadora Alessandra Paulino, por me instruir, motivar e pela paciência ao longo do trabalho. Aos meus amigos e professores, especialmente ao Cesar Marçal e ao Rafael Dias Araújo, que nunca deixaram eu desistir desta longa caminhada e confiaram no meu potencial e sucesso. Aos meus irmãos e a minha avó que sempre estiveram me apoiando nesta trajetória. E por fim a banca examinadora por aceitar fazer parte desse momento tão importante na minha vida, a realização de um sonho.

"Só se pode alcançar um grande êxito quando nos mantemos fiéis a nós mesmos."
(Friedrich Nietzsche)

Resumo

Nos últimos anos têm-se notado um aumento do número de evasões e reprovações na graduação em disciplinas de programação, incluindo disciplinas relacionadas ao paradigma de Orientação a Objetos (OO), então também tem aumentado a procura por soluções para a redução destes índices. Este trabalho tem como objetivo o desenvolvimento de um Sistema Web, com o intuito de auxiliar alunos e professores no processo ensino-aprendizagem deste paradigma. Para isso foram utilizados *Hypertext Preprocessor* (PHP) juntamente com o padrão de projeto arquitetural *Model-View-Controller* (MVC), e elementos de gamificação. Do ponto de vista do aluno, o sistema desenvolvido deve auxiliá-lo a testar o seu aprendizado com *feedback* imediato, dar apoio ao ensino do tópico de OO por meio da disponibilização organizada de materiais e vídeos complementares, além de fornecer uma ótima oportunidade de o mesmo verificar suas dificuldades e facilidades ao longo da aprendizagem, podendo verificar o desempenho da turma e ser incentivado por meio da competição. Do ponto de vista do professor, o sistema desenvolvido deve facilitar a verificação de quais assuntos precisam ser reforçados, sendo uma forma de avaliar de forma fácil e rápida o aprendizado dos alunos para que possam ter um melhor aproveitamento ao longo da disciplina, diminuindo assim as taxas de evasão e reprovação.

Palavras-chave: Programação orientada a objetos, Java, Gamificação, Ensino-aprendizagem.

Lista de ilustrações

Lista de tabelas

Lista de siglas

CRUD *Create, Read, Update, Delete*

DAO *Data access object*

MVC *Model-View-Controller*

OO Orientação a Objetos

POO Programação Orientada a Objetos

PHP *Hypertext Preprocessor*

Sumário

Capítulo 1

Introdução

Nos últimos anos têm-se notado um aumento no número de reprovações e na evasão dos alunos no ensino superior em cursos de computação na disciplina de Programação Orientada a Objetos (POO) (COSTA et al., 2017). Um dos fatores que contribuem com esse aumento é o alto grau de complexidade do paradigma da Orientação a Objetos (OO). Nesta disciplina, o aluno deve reconhecer que a OO é uma simulação do mundo real, um mundo povoado por objetos, onde esses objetos devem existir e trocar mensagens entre si (BARANAUSKAS, 1993). Além disso, existem também as dificuldades dos alunos na aprendizagem de programação em geral nas universidades, visto que este tópico é bastante diferente dos conteúdos que os alunos têm contato no ensino fundamental e médio.

Assim, pesquisadores e professores vêm estudando meios pra minimizar essas taxas, tendo em vista que o aluno deverá mudar o processo de seu pensamento com a aprendizagem deste novo paradigma, pensando em uma nova forma de apresentar soluções a estes novos problemas (BARANAUSKAS, 1993), e, com a ajuda de dispositivos móveis, lousa digital, jogos digitais e ambientes virtuais, o mesmo deverá aprender com uma certa facilidade os conceitos e práticas na POO (MACHADO et al., 2016).

A utilização de ferramentas, como por exemplo a gamificação, pode ajudar o aluno a ter um melhor aproveitamento no contato com este novo paradigma, auxiliando tanto na aprendizagem dos conceitos e pilares da OO quanto a praticar estes conceitos por meio do desenvolvimento de programas. A gamificação é uma técnica para o encorajamento dos comportamentos desejados pelos educadores, neste caso relacionado ao ensino do paradigma de Orientação a Objetos, que aumenta a motivação destes alunos na aprendizagem e o engajamento na disciplina (BRAZIL; BARUQUE, 2015).

Este trabalho propõe uma solução para a melhoria na aprendizagem do paradigma de Orientação a Objetos, e consequentemente a minimização dos índices de retenção e evasão já mencionados, através do desenvolvimento e da disponibilização de um sistema Web com o uso da gamificação com o intuito de auxiliar o aluno, tendo este a oportunidade de sanar diversas dúvidas e dificuldades que podem surgir ao longo da disciplina.

O sistema Web visa contribuir com o aprendizado dos alunos por meio da melhor

compreensão dos conceitos e pilares da orientação a objetos, além de ajudar professores e tutores no processo de ensino através da disponibilidade de recursos de monitoramento destes alunos que fornecem informações mais precisas e específicas, como, por exemplo, em quais conceitos cada aluno está tendo dificuldades e se o aluno está conseguindo acompanhar o conteúdo ministrado pelo professor até o momento. Além disso, o sistema Web deve servir como um material de apoio através da funcionalidade de exibir vídeos explicativos sobre a teoria e desenvolvimento de POO, e da disponibilização de materiais complementares organizados, incluindo listas de boas e más práticas em POO e Java, e sugestões de livros que servirão de embasamento ao longo do aprendizado da disciplina.

1.1 Objetivos

O objetivo deste trabalho é desenvolver um sistema web com o uso de alguns elementos de gamificação que visa auxiliar estudantes a aplicar e compreender os conceitos do paradigma de Orientação a Objetos, reduzindo assim os índices de reprovação e evasão nas disciplinas relacionadas a estes conhecimentos.

Os objetivos específicos do desenvolvimento e disponibilização da ferramenta estão listados abaixo:

- ❏ Apoiar o professor na abordagem de conceitos mais específicos do Paradigma de Orientação a Objetos, por meio de um monitoramento mais simples de como está a aprendizagem de seus alunos;
- ❏ Ajudar o aluno a ter curiosidade e motivação no aprendizado deste novo paradigma;
- ❏ Ajudar o aluno a identificar quais são as suas dificuldades ao longo do aprendizado de POO ;
- ❏ Ser um material complementar organizado no aprendizado de POO , o qual os alunos poderão consultar ao longo do aprendizado da disciplina.

1.2 Resultados Esperados

O principal resultado esperado é o desenvolvimento de um sistema atraente e de fácil utilização que incentive a aprendizagem por meio da disponibilização de conteúdos organizados e relevantes à aprendizagem da Programação Orientada a Objetos e por meio da utilização de elementos da gamificação.

A partir do desenvolvimento e da utilização deste sistema na disciplina de Programação Orientada a Objetos, espera-se uma melhor compreensão do paradigma de Orientação a Objetos pelos alunos, um melhor aproveitamento e monitoramento do professor na sala

de aula, além da diminuição das taxas de reprovação e evasão de alunos nas disciplinas relacionadas, visto que a ferramenta promoverá aulas e atividades mais dinâmicas e interessantes tanto para os alunos como para os professores, tornando o processo de aprendizagem da Programação Orientada a Objetos mais divertido e atraente.

1.3 Organização da Monografia

Este trabalho está organizado em cinco capítulos, sendo este o Capítulo 1, onde foram introduzidos o tema, a motivação, os problemas, a justificativa, os objetivos gerais e específicos e os resultados esperados do trabalho. O Capítulo 2 apresenta conceitos relacionados ao paradigma de Programação Orientada a Objetos, gamificação, dificuldades no ensino e aprendizagem deste paradigma, vantagens e benefícios da utilização do paradigma orientado a objetos, exemplos e aplicações do paradigma de orientação a objetos, além de mencionar e discutir trabalhos relacionados ao tema proposto. O Capítulo 3 apresenta a modelagem do sistema, bem como as linguagens e recursos utilizados para o desenvolvimento e funcionamento do sistema. O Capítulo 4 mostra o Sistema Web desenvolvido por meio da apresentação das suas responsabilidades e funcionalidades. Por fim, o Capítulo 5 apresenta as considerações finais e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados os principais tópicos que serão necessários para o desenvolvimento do trabalho, iniciando-se pelos conceitos e pilares da Programação Orientada a Objetos, seguido do uso da gamificação no processo de ensino-aprendizagem, uma discussão sobre as dificuldades enfrentadas pelos alunos na aprendizagem do paradigma de Orientação a Objetos e, ao final, serão apresentado alguns trabalhos relacionados ao tema.

2.1 Programação Orientada a Objetos

Os conteúdos abordados no paradigma de Orientação a Objetos, presentes nas fichas disciplinares dos cursos da área de Exatas, são organizados decorrentes do aprendizado dos alunos em linguagens de lógica estruturada, ou seja, os conteúdos de OO são organizados de forma a serem apresentados como um avanço dessa solução computacional (estruturada). Essa transição é complexa, pois requer do aluno um conjunto de habilidades e competências sobre estruturas de programação, evidenciando diferenças, vantagens e facilidades (BELTRAME, 2018), como por exemplo sobre o que são objetos, atributos, métodos, classes, herança, polimorfismo, encapsulamento e abstração. A seguir, conceitos e pilares do paradigma orientado a objetos são brevemente introduzidos.

2.1.1 Conceitos e pilares

- **Objeto:** Termo utilizado para representar um determinado elemento do mundo real. Somente são analisados os objetos que têm relevância para a solução de um problema proposto. Pode-se definir objetos como instâncias de classes, portanto se determina qual informação o objeto terá e como será manipulado (FARINELLI, 2007). A Figura 1 apresenta um exemplo de objeto.

Figura 1 – Exemplo de um objeto.
Fonte: Retirado do site br.freepik.com. [1]

- **Atributos:** São os valores que definem o estado do objeto, ou seja, variáveis ou campos que armazenam diferentes características do objeto (FARINELLI, 2007). A Tabela 1 demonstra exemplos de atributos do objeto mostrado na Figura 1.

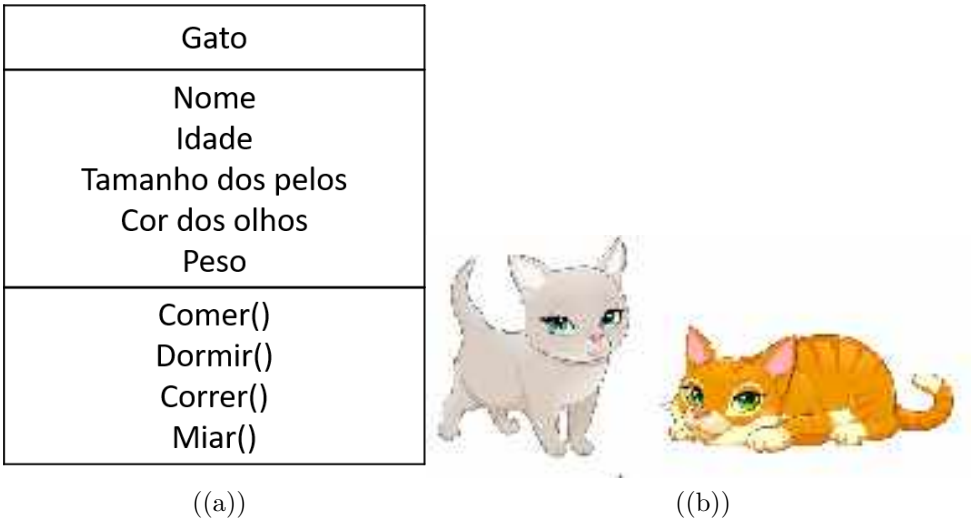
Tabela 1 – Exemplos de atributos do objeto mostrado na Figura 1.

Gato	
Nome:	Tufo
Idade:	2 anos
Tamanho dos pelos:	Curtos
Cor dos olhos:	Amarelo
Peso:	5 kg

- **Métodos:** São procedimentos ou funções que realizam as ações do próprio objeto, ou seja, os métodos são as ações que o objeto poderá realizar. Os métodos definem o que é possível fazer com cada objeto (ou instância da classe) (FARINELLI, 2007; RICARTE, 2001). Pode se citar como exemplos de métodos para o objeto (gato) mostrado na Figura 1: comer(), dormir(), correr(), miar() etc., que representam ações que cada uma das instâncias pode executar.

- **Classe:** Representação de um conjunto de objetos que possui características e comportamentos comuns; logo, um objeto é a instanciação de uma determinada classe. A partir da representação de uma classe, a mesma descreverá quais as propriedades, métodos e atributos que o objeto constituirá, além de mencionar as funcionalidades que podem ser aplicadas ao objeto (FARINELLI, 2007; RICARTE, 2001). Na Figura 2 são mostrados dois objetos de uma mesma classe (gatos), e cada um dos objetos pode ser representado pelo modelo definido pela classe, mas com estados diferentes para os atributos.

[1] <https://br.freepik.com/vetores-premium/gato-preto-e-branco-dos-desenhos-animados_6879950.htm/>



((a)) ((b))

Figura 2 – Exemplo de uma classe (modelo) para gatos com atributos e métodos, e duas instâncias da classe Gato (objetos).
Fonte: Figuras dos gatos retiradas do site br.freepik.com[2], atributos de autoria própria.

Em resumo, temos que: (i) objetos são, em geral, representações de elementos do mundo real; (ii) as características que descrevem um objeto são chamadas de atributos; (iii) as ações que um objeto pode executar são chamadas de métodos; e (iv) a principal diferença entre objeto e classe é que a classe é um modelo e todos os objetos daquela classe possuem atributos e métodos comuns (FARINELLI, 2007).

O paradigma de orientação a objetos é norteado por alguns pilares: a abstração, o encapsulamento, a herança e o polimorfismo, que são apresentados a seguir.

- **Abstração:** Consiste na ideia de não se preocupar com características não relevantes, devendo concentrar somente nos aspectos essenciais do objeto (LEITE et al., 2016).

- **Encapsulamento:** Sua finalidade é esconder a forma de como algo é feito, mostrando apenas o resultado esperado. No encapsulamento, informações não essenciais ou relevantes são ocultadas. Por exemplo, quando vamos ao médico, se um remédio é receitado, em geral o médico não descreverá todos os ingredientes que o compõem, mas apenas descreverá o nome do remédio e a forma de utilização; neste caso, é desejado apenas o resultado final (o remédio) e não o interesse ou necessidade de conhecer quais as substâncias das quais este é formado. Na Figura 3 é representado um exemplo de ocultamento da informação: ao olhar para as operações como os botões do controle, que são externos, para mudarmos as informações na televisão, não temos acesso aos dados no controle (que são internos), e nem ao processo que ocorre para que as informações sejam alteradas; ou seja, as informações internas

[2] <https://br.freepik.com/vetores-gratis/grupo-de-gatos-bonitos-com-cores-diferentes-do-vetor-dos-desenhos-anima
1066826.htm/>

do controle e os processos são ocultos aos seus usuários – os dados e processos são encapsulados (LEITE et al., 2016).

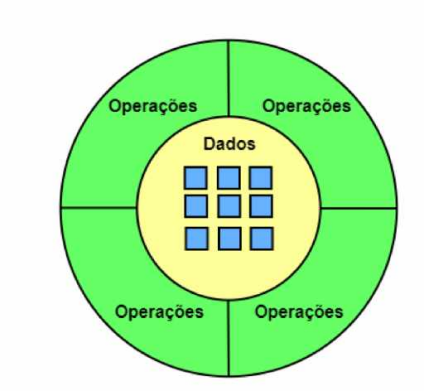


Figura 3 – Exemplo de encapsulamento: ocultamento de uma informação.

Fonte: (LEITE et al., 2016).

- **Herança:** Herança é o relacionamento entre classes onde uma subclasse (denominada classe filha) é uma extensão de outra classe (denominada classe mãe ou superclasse). Com isso, uma subclasse consegue reaproveitar métodos e atributos da superclasse. Uma classe pode herdar características de outras classes, como métodos e atributos, e aquelas que recebem suas características são as herdeiras. A partir de uma superclasse (a que repassará suas características), suas subclasses receberão métodos e atributos, além de ser possível adicionar características próprias. A título de exemplo, suponhamos que duas classes, automóvel e moto, herdam características da superclasse veículo, a qual possui o atributo quantidade de passageiros que faz sentido para ambas as subclasses; a classe automóvel poderia ter um atributo tamanho do porta-malas, que não faria sentido na classe moto, ainda mantendo o atributo quantidade de passageiros da superclasse (FARINELLI, 2007; RICARTE, 2001). Na Figura 4 temos um exemplo de herança mostrando a superclasse veículo e as subclasses carro, moto e bicicleta, onde todas as subclasses possuem a mesma característica que a superclasse, mas também pode possuir características próprias.

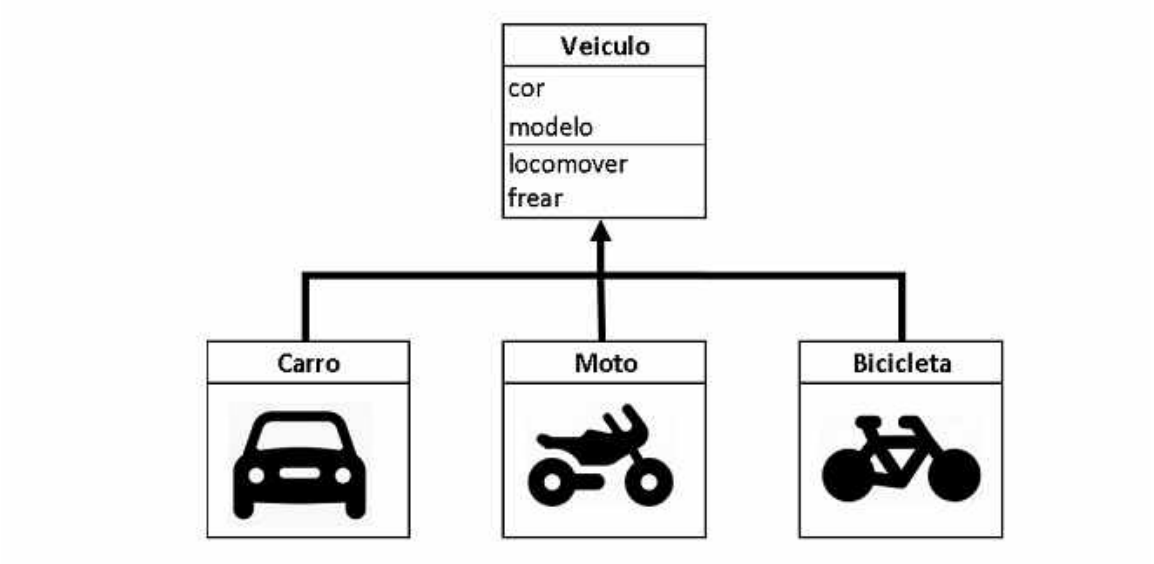


Figura 4 – Exemplo de herança.
Fonte: <https://testeavelocidade.com.br/heranca-superclasse-e-subclasse-java/x>.

- **Polimorfismo:** Polimorfismo é o princípio onde duas ou mais classes podem invocar o mesmo método, mas que diferem em seus comportamentos (RICARTE, 2001). Por exemplo, objetos das classes aves e anfíbios podem invocar o mesmo método de locomoção, mas as formas como os objetos de cada classe se locomovem são diferentes; assim, temos o mesmo método nas duas classes, porém seus comportamentos se diferem. Na Figura 5 é demonstrado um exemplo de polimorfismo: tanto o computador quanto a filmadora possuem a mesma função que é gravar, mas o comportamento desta função se difere no modo como a gravação é feita.

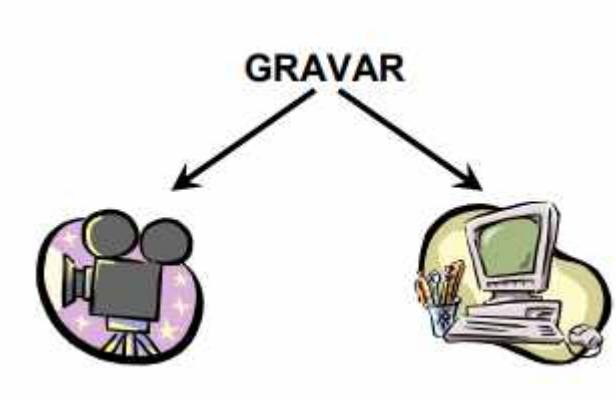


Figura 5 – Exemplo de Polimorfismo: a função gravar apresenta comportamento diferente na filmadora e no computador.
Fonte: (FARINELLI, 2007).

2.1.2 Exemplos de implementações na linguagem Java

A programação Orientada a Objetos se baseia na composição e interação de objetos, portanto o funcionamento de um sistema orientado a objetos acontece por meio do

relacionamento e troca de mensagens entre esses objetos.

Diversas linguagens baseiam-se no modelo orientado a objetos, como por exemplo: Smalltalk, Python, Ruby, C++, Object Pascal, Java, C#, Oberon, Ada, Eiffel, Simula, .Net. Um exemplo de implementação escrita na linguagem Java que cria uma classe Cliente com alguns atributos e método relevantes é apresentado no Código fonte 2.1, no qual os atributos da classe Cliente são `nome` e `cpf`, e ambos podem ser acessados apenas de dentro da classe; `mostraDados(String, String)` é um método definido nessa classe.

```
1  package exemplo;
2
3  public class Cliente {
4
5      private String nome;
6      private String cpf;
7
8      private void mostraDados(String cpf, String nome){
9          System.out.println("Nome do cliente: " + nome +"\n");
10         System.out.println("CPF do cliente: " + cpf +"\n");
11     }
12 }
```

Código fonte 2.1 – Exemplo de um código fonte em Java que implementa uma classe Cliente com alguns atributos e método relevantes.

No Código fonte 2.2 é apresentada uma implementação da classe Conta que se utiliza da classe Cliente, com os atributos `nrDaConta`, `descricao`, `saldo`, `limite` e `cliente`, todos definidos para serem acessados apenas de dentro da classe, ou seja, apenas pelos métodos definidos dentro da classe, que são `saque(double)` e `deposita(double)`.

```
1  package exemplo;
2
3  public class Conta {
4      private String nrDaConta;
5      private String descricao;
6      private double saldo;
7      private double limite;
8
9      private Cliente cliente = new Cliente();
10
11     public boolean saque (double valor){
12         if (valor<=(saldo+limite)){
13             saldo-=valor;
14             return true;
15         } else{
```

```
16              return false;
17          }
18      }
19
20      public boolean deposita (double valor){
21          if (valor<=(saldo+limite)){
22              saldo+=valor;
23              return true;
24          } else{
25              return false;
26          }
27      }
28  }
```

Código fonte 2.2 – Exemplo de um código fonte em Java que implementa uma classe Conta com alguns atributos e métodos relevantes (extraído de (JUNGTHON; GOULART, 2009)).

2.2 Vantagens e benefícios da utilização do paradigma orientado a objetos

O grande diferencial do paradigma de Orientação a Objetos em relação aos outros paradigmas está relacionado ao uso do conceito de herança e polimorfismo, onde a herança permite a extensão de classes e o polimorfismo permite que funcionalidades de um programa utilizem uma forma dinâmica, durante a sua execução (FARINELLI, 2007). A seguir são apresentadas algumas das vantagens da utilização deste paradigma:

Unificação entre dados e processos: O mesmo modelo orientado a objetos, consegue ter a conformidade entre dados e processos, diferentemente de outros paradigmas, onde procuram primeiro definir os procedimentos onde deverão ser automatizados e posteriormente identificar os dados que o sistema necessita (FARINELLI, 2007).

Reutilização e aumento da produtividade: A Orientação a Objetos permite a reutilização de objetos, ou seja, os objetos utilizados em sistemas anteriores poderão ser utilizados em novos sistemas sem modificar suas estruturas internas, ou serem modificados com o intuito de acomodar esses novos códigos. Com isso, ao implementar um novo sistema o desenvolvedor poderá buscar objetos implementados em outros sistemas e utilizá-los sem alterar a estrutura do mesmo; caso esses objetos já existam, estes poderão ser incluídos no novo sistema sem a necessidade de custo e criação. Outra opção quanto a reutilização é transformar objetos semelhantes, ou seja, objetos já implementados que possuem uma certa semelhança podem sofrer pequenas modificações e assim serem úteis ao novo sistema (FARINELLI, 2007).

2.3 Dificuldades no aprendizado do paradigma de Orientação a Objetos

As linguagens de programação estão presentes no dia a dia de estudantes nos cursos relacionados ao aprendizado de computação. Contudo, como é efetuada a apresentação dos alunos a este novo paradigma orientado a objetos é de suma importância para o seu aprendizado. É preciso saber motivar os estudantes, fazer com que se interessem e tenham o hábito de gostar de aprender e estudar, pois, desta forma, terão incentivo suficiente para superar suas dificuldades, que na maioria dos casos envolvem matemática, raciocínio lógico e capacidade de abstração (BARROS; FREIBERGER, 2008).

A atividade de programação no quesito de mudança de paradigma é algo bastante complexo, pois envolve muito mais do que conhecer entidades sintáticas e a semântica do novo paradigma a ser estudado; deve-se mudar a forma de pensar ao se resolver um determinado problema, ajustando-se ao novo pensar do paradigma proposto (BARANAUSKAS, 1993).

Em outras palavras, existem grandes dificuldades no paradigma de orientação a objetos pois, ao desenvolver um programa para resolver um problema, o estudante, além de precisar saber sobre a lógica de programação, deverá também aplicar um grau de abstração diferenciado para que possa utilizar corretamente os conceitos de objeto, classe, encapsulamento, herança etc. (BARROS; FREIBERGER, 2008).

De acordo com (COSTA et al., 2017), outras dificuldades também são apontadas pelos alunos como causadoras dos altos índices de reprovação e evasão, sendo elas: falta de eficácia na metodologia de ensino devido ao baixo número de exemplos teóricos e práticos, baixa atenção dada pelos alunos ao conteúdo e falta de infraestrutura em algumas universidades.

2.4 Gamificação como ferramenta de apoio ao aprendizado

O termo gamificação é oriundo da palavra inglesa *gamification* e consiste em usar características e princípios de jogos, facilitando o aprendizado e a resolução de problemas e favorecendo oportunidades para a motivação e engajamento de estudantes (AGUIAR, 2015). Quando aplicado à educação, o termo gamificação tem como objetivo utilizar mecanismos de jogos na concepção educativa, com o intuito de tornar os conteúdos das disciplinas mais atrativos ao estudante.

A gamificação é vista como uma forma de melhorar o potencial e a qualidade do aprendizado dos alunos, através de um maior empenho dos mesmos nas atividades propostas (BRAZIL; BARUQUE, 2015).

Uma forma de motivar o interesse dos alunos no aprendizado com o uso da gamificação, é a utilização de recompensas a cada acerto do aluno, essas recompensas podem incluir níveis a serem atingidos a cada acerto de uma questão, ranking entre os melhores, prêmios, entre outras formas de bonificar estes alunos (COSTA et al., 2017).

2.5 Trabalhos relacionados

Como visto na Seção 2.3, as dificuldades no processo de ensino-aprendizagem de programação, e em particular do paradigma de orientação a objetos, são diversas. A preocupação com este processo tem crescido, e muitos trabalhos têm sido apresentados que se utilizam do conceito de gamificação para apoio em disciplinas de programação. Nesta seção discutiremos alguns desses trabalhos.

Brazil e Baruque (2015) abordam a aplicação do uso da gamificação no curso de Desenvolvimento de Jogos Digitais do Instituto Federal de Educação, Ciência e Tecnologia do Rio de Janeiro (IFRJ), evidenciando que a aplicação da gamificação ao ensino das disciplinas foi favorável e bastante significativa para a maioria dos alunos da disciplina (BRAZIL; BARUQUE, 2015). O trabalho baseou-se na aplicação de elementos específicos de gamificação para a melhora na aprendizagem: títulos, conquistas e músicas, e a exposição dos alunos a um questionário online que apresentava perguntas referentes ao grau de satisfação dos mesmos. De acordo com os resultados obtidos por meio das respostas aos questionários, concluiu-se que os desafios e conquistas são os requisitos mais favoráveis e que têm a maior influência no aprendizado dos alunos.

Além do mencionado acima, outro trabalho em destaque foi o de Aguiar (2015), que constituiu na realização de atividades referentes a aprendizagem de conteúdos sobre a programação orientada a objetos, especialmente a herança, no contexto da gamificação. Esta abordagem foi empregada no curso técnico em Informática da Escola Técnica Redentorista (Campina Grande, Paraíba).

A experiência consistiu na realização de uma atividade que visava a verificação do aprendizado de conteúdos abordados em sala de aula: a turma foi dividida em cinco grupos de seis integrantes cada, onde cada grupo recebeu um problema a ser solucionado utilizando-se o computador e a IDE Eclipse. Foi observado que o uso da metodologia de gamificação fez com que os alunos se mostrassem mais participativos e entusiasmados em comparação com as aulas tradicionais.

A partir da conclusão destas atividades, notou-se em geral uma mudança na parte dos alunos em relação a disciplina de POO: os mesmos se tornaram mais participativos e entusiasmados no curso de computação, extraindo resultados positivos para o processo de ensino aprendizagem.

Uma abordagem que também apresentou melhoras no processo de ensino-aprendizagem do paradigma de Orientação a Objetos foi desenvolvido e aplicado na Universidade Fede-

ral do Ceará - Campus Russas, e utilizou-se da metodologia da sala de aula invertida e do uso de elementos de gamificação para a melhoria no aprendizado (COSTA et al., 2017). A mesma objetivava motivar e facilitar o aprendizado na disciplina de POO, diminuindo os altos índices de reprovação, melhorando as habilidades e as competências dos estudantes.

Foram criadas estratégias como o uso de pontuação a cada resposta correta, ranking entre os alunos, níveis de dificuldade, recompensas, entre outras estratégias, para motivar a aprendizagem do aluno e aumentar a frequência de utilização da ferramenta virtual, bem como o comprometimento e o empenho do aluno. Um efeito positivo se deu no aumento significativo no acesso pelos alunos ao material da disciplina em períodos diversos das vésperas de avaliações, ou seja, os alunos deixaram de acessar o conteúdo da matéria apenas na véspera das avaliações.

Outras melhorias utilizando a nova metodologia foram observadas: melhor experiência expositiva nas aulas, pois os professores puderam torná-las mais interessantes e dialogadas, e um maior acompanhamento pelo professor do desempenho dos alunos, inclusive tornando os dois principais agentes do processo de ensino-aprendizagem muito mais próximos.

Foi demonstrado que a utilização da metodologia da sala de aula invertida trouxe um melhor desempenho no ensino, enquanto a gamificação despertou nos alunos um maior interesse no aprendizado, melhorando a participação dos mesmos no processo.

Após este estudo, foi observado um aumento no índice de aprovação bem como na média dos alunos na disciplina durante os semestres que foram utilizadas as estratégias descritas anteriormente. Neste método, o rápido entendimento do professor a respeito do desempenho do aluno garante que o mesmo possa averiguar quais alunos estão tendo dificuldades no aprendizado ainda com tempo para agir na correção.

Na pesquisa feita com os estudantes, foi mencionado um certo grau de insatisfação, que pode demonstrar a falta de contentamento causado pelo fato do aluno ter que sair da sua zona de conforto e enfrentar um novo ambiente de aprendizado.

Capítulo 3

Desenvolvimento

Este capítulo apresenta algumas etapas da modelagem do sistema, abordando os requisitos funcionais do sistema, os casos de uso e os padrões de projeto e tecnologias utilizados para a criação do sistema.

3.1 Modelagem do sistema

Nesta seção serão apresentados os requisitos funcionais do sistema, que são necessários para o funcionamento do sistema e o diagrama de casos de uso.

3.1.1 Diagrama de Casos de Uso

Diagramas de casos de uso tem como finalidade descrever os requisitos que deverão ser atendidos pelo sistema, demonstrando os seus atores e qual a funcionalidade de cada um no sistema.

A Figura 6 ilustra o diagrama de casos de uso do sistema, o qual contém dois atores, o professor e o aluno, que utilizarão o sistema. O professor tem como papel cadastrar-se e realizar login, cadastrar turmas e consultar o rendimento dos alunos cadastrados na sua turma. Já o papel do aluno é se cadastrar no sistema, realizar login, se cadastrar na turma com o código recebido pelo seu professor, realizar as atividades, visualizar o rendimento da turma de acordo com as atividades concluídas, editar dados de seu perfil, como senha, nome e instituição que estuda.

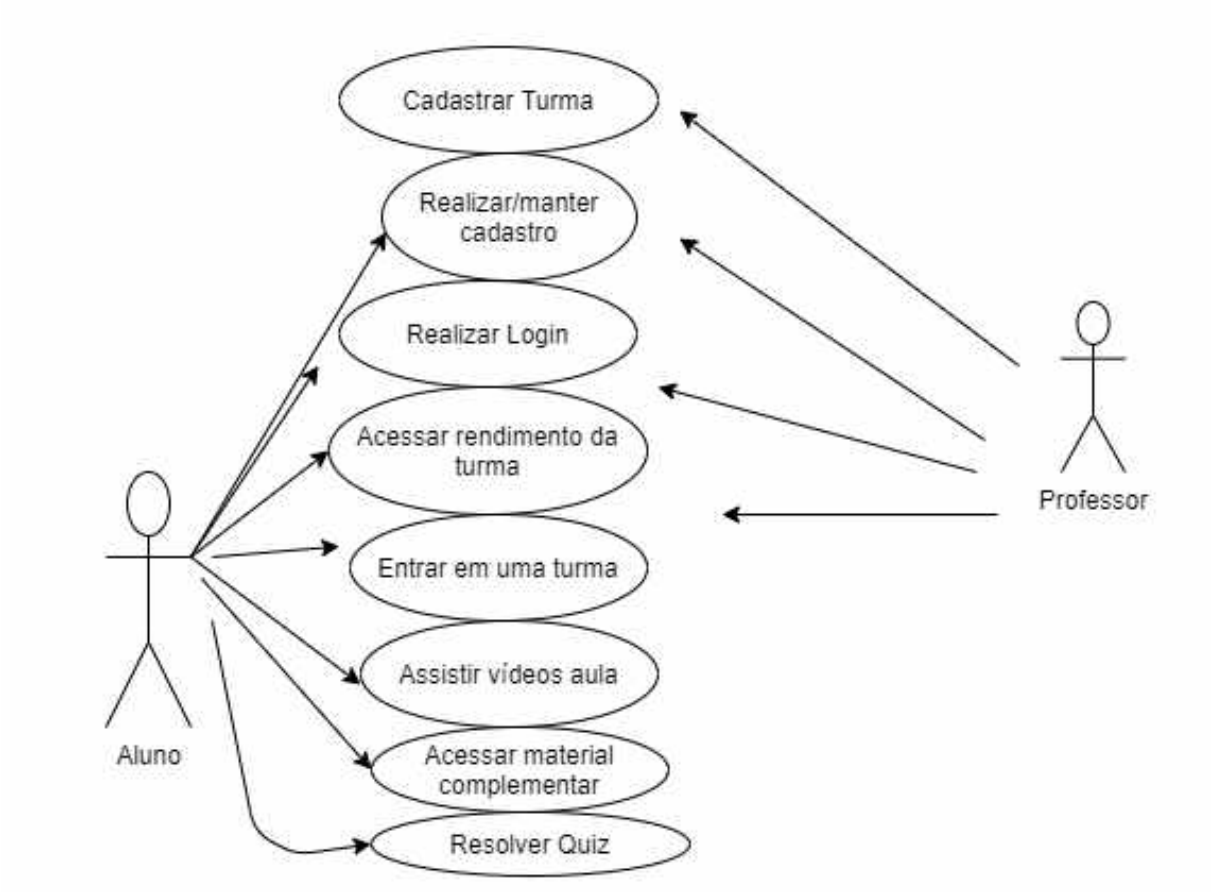


Figura 6 – Diagrama de casos de uso do sistema.

Fonte: Autoria própria.

3.1.2 Análise dos Requisitos

Os requisitos funcionais têm como finalidade mostrar os itens essenciais para o funcionamento do sistema, ou seja, as principais funcionalidades que o sistema disponibilizará. A Tabela 2 mostra todos os requisitos funcionais utilizados no sistema.Que descreve um pouco sobre como será o desenvolvimento do sistema, demonstrando os itens essenciais em seu funcionamento.

Tabela 2 – Requisitos funcionais

Número	Descrição
RF01	O usuário deverá realizar login para acessar o conteúdo do Sistema Web. Antes de realizar o login, será necessário efetuar um cadastro com seus dados, após o qual o usuário estará apto para acessar as funcionalidades do Sistema.
RF02	O sistema não aceitará mais de um cadastro com o mesmo e-mail.
RF03	Para a realização das atividades é necessário o cadastro do aluno em uma turma.
RF04	Disponibilizar uma página com materiais complementares, caso o aluno fique com alguma dúvida sobre um determinado assunto.
RF05	Disponibilizar uma tela de vídeos, na qual o aluno poderá rever algum aprendizado que não foi bem compreendido.
RF06	Disponibilizar uma tela de cadastro de turma, para que o professor possa cadastrar uma turma e os alunos possam acessar sua turma.

Fonte: Autoria própria.

3.2 Banco de dados

O banco de dados foi criado para o funcionamento e melhor integração do sistema, e envolve o controle das tabelas para as entidades turmas, professores, alunos, questões que fazem parte das atividades que o aluno pode desenvolver, entre outras. A Figura 7 mostra as interações entre as tabelas do banco de dados, no qual ao todo foram criadas sete tabelas: (i) a tabela `Usuário` armazena as informações tanto de alunos como de professores no banco de dados, e a unicidade do cadastro é garantida por meio do e-mail – o sistema só pode ser acessado por usuários cadastrados; (ii) a tabela `Turma` armazena as informações de turmas cadastradas pelo professor – o código e a senha da turma serão utilizados para que os alunos se cadastrem em turmas específicas do professor; (iii) a tabela `Atividade` armazena as atividades feitas pelos alunos relacionando-as com as turmas; (iv) a tabela `Questão` contém as perguntas que o aluno responderá no quiz; (v) a tabela `Alternativa` contém as alternativas de uma determinada questão; (vi) a tabela `Usuario_has_atividade` salva a maior nota do usuário para posterior visualização do rendimento da turma; (vii) a tabela `Usuario_has_turma` relaciona alunos com uma determinada turma criada.

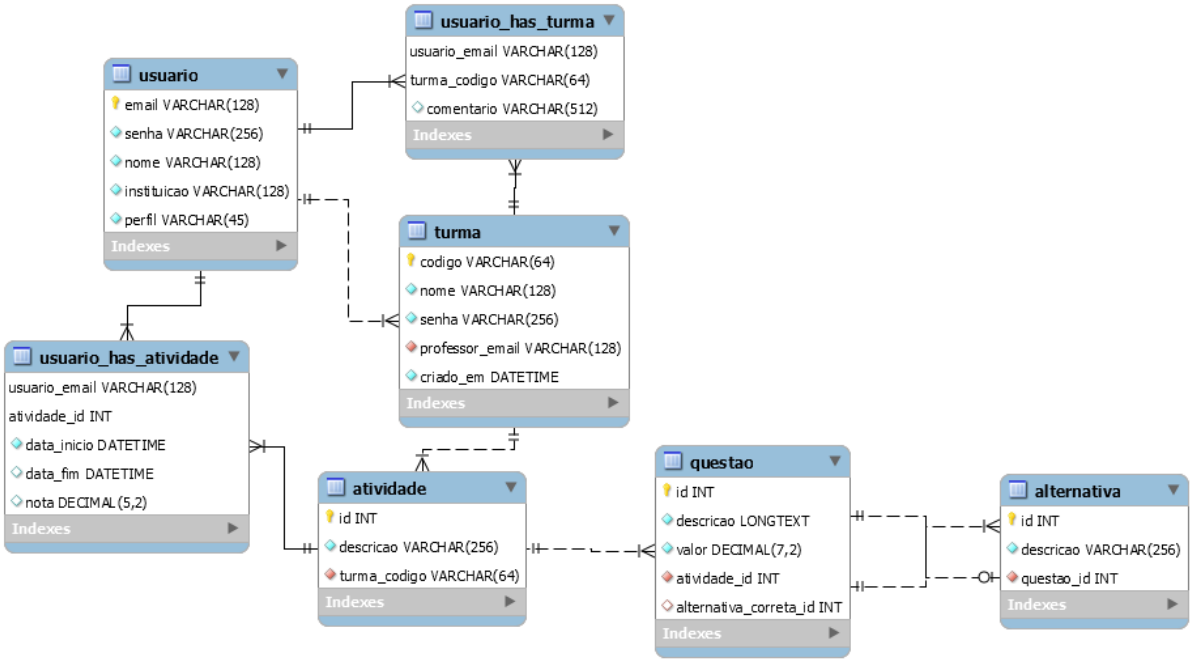


Figura 7 – Diagrama Entidade Relacionamento (DER) do Banco de Dados.

Fonte: Autoria própria.

3.3 Sistema Web

Para o desenvolvimento do sistema, foram utilizados: a linguagem de marcação de hipertexto(HTML), folha de estilo em cascata (CSS), Java Script (JS), PHP e padrões de projeto (Singleton, Model-View-Controller e DAO).

Capítulo 4

Resultados

Neste capítulo são apresentados os resultados obtidos com a implementação do Sistema Web, detalhando a hierarquia do trabalho, incluindo a reprodução das telas e alguns códigos fonte principais, além de informações adicionais de elementos do sistema e sua hospedagem e disponibilização.

4.1 Estruturação e descrição do sistema

Na Figura 8 temos a representação das pastas do Sistema Web.

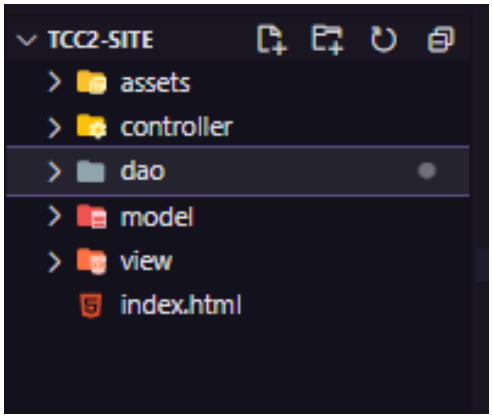


Figura 8 – Representação dos padrões de projeto utilizados na construção do sistema.

Fonte: Autoria própria.

O primeiro arquivo do sistema a ser analisado é a página inicial (`index.html`), reproduzida na Figura 9, o usuário pode escolher entre se cadastrar ou realizar o login. Caso o usuário esteja com dúvidas sobre como utilizar o sistema, este pode acessar a ajuda por meio do botão com o ponto de interrogação que contém uma breve explicação do funcionamento do Sistema.

Figura 9 – Reprodução da tela inicial do sistema.

Fonte: Autoria própria.

4.1.1 Camada *view*

A pasta *view* (visão) é responsável por toda a parte visual do sistema, constituída por arquivos PHP, e está representada na Figura 10.

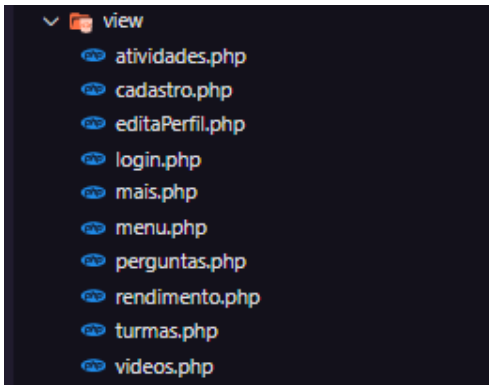


Figura 10 – Representação da pasta *view* do sistema.

Fonte: Autoria própria.

A tela de cadastro (arquivo `cadastro.php`) reproduzida na Figura 11, é responsável por cadastrar novos usuários ao sistema; caso o usuário já esteja cadastrado, basta clicar no botão que já é cadastrado e será redirecionado para a tela de Login.

Uma função Ajax é responsável por verificar todas as informações registradas pelo usuário, assim que o usuário finaliza o seu preenchimento. Esta função, que está representada na Figura 12, fará algumas verificações e o cadastro só será aceito caso a senha

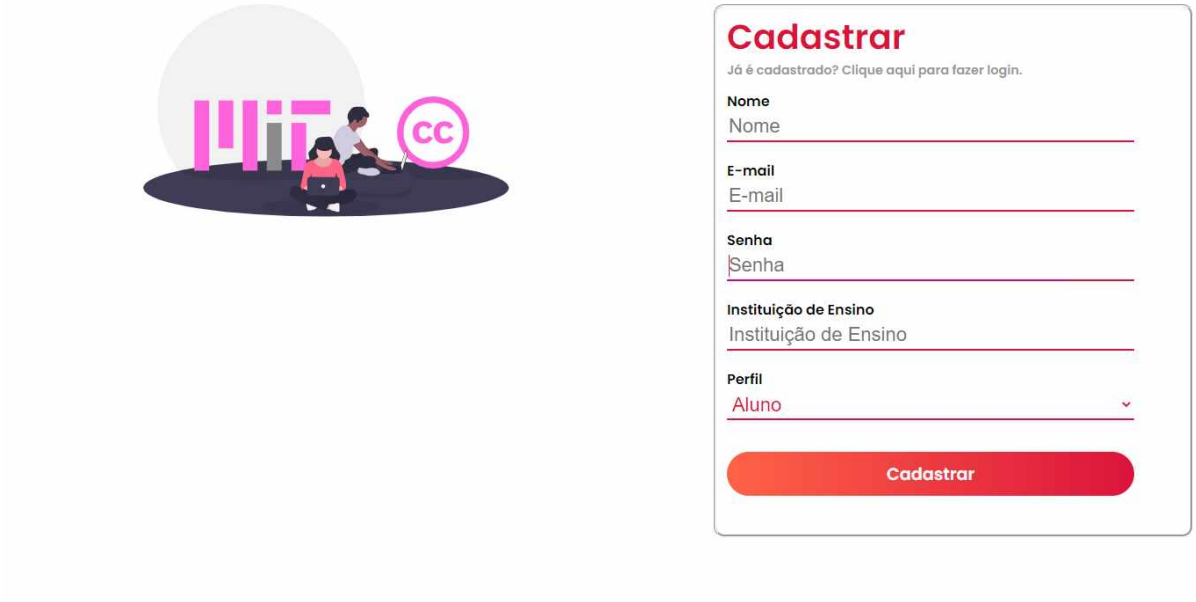


Figura 11 – Reprodução da tela cadastro.

Fonte: Autoria própria.

tenha cinco ou mais caracteres e o e-mail não tenha sido previamente cadastrado (não são aceitos cadastros com e-mails duplicados). Nos casos em que todo o cadastro esteja preenchido corretamente, o usuário será redirecionado para a página turmas.

```
$.ajax({
    type: "POST",
    url: "../controller/UsuarioController.php",
    data: {
        save: 1,
        nome,
        email,
        senha,
        instituicao,
        perfil
    },
    success: function(data) {
        var value=JSON.parse(data)
        if (value.error)
        {
            alert("E-mail já cadastrado!")
            return
        }
        alert(
            data
                ? "Cadastro efetuado com sucesso!"
                : "Erro ao realizar cadastro!"
        )

        location.href="./login.php"
    },
    error: function (data) {
        alert("Erro inesperado, tente novamente!")
        console.log(data)
    }
})
})
})
```

Figura 12 – Representação da função Ajax de cadastro.

Fonte: Autoria própria.

Na tela para editar o perfil (arquivo `editaPerfil.php`) reproduzida na Figura 13, o usuário pode alterar as suas informações como nome, senha e Instituição. Ao acessar esta opção, todos os seus dados estarão listados na página, bastando ao usuário alterá-los da forma que achar necessário.

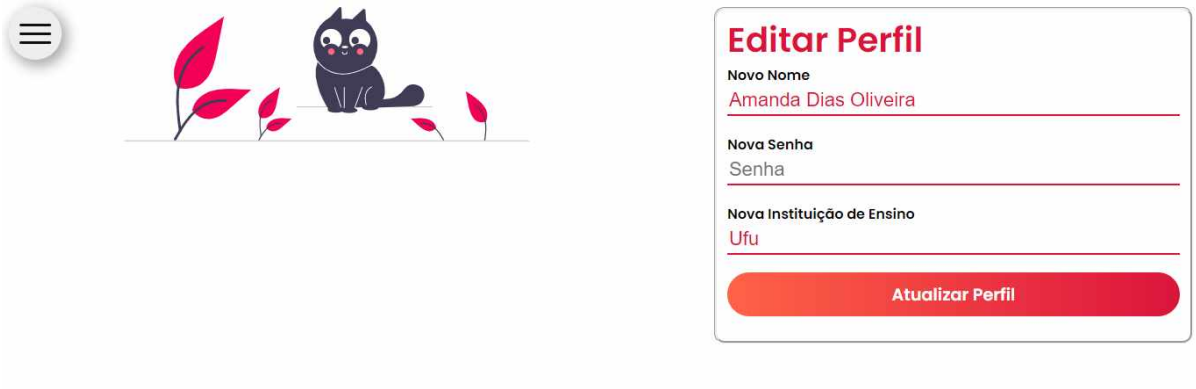


Figura 13 – Reprodução da tela Edita Perfil.

Fonte: Autoria própria.

Assim que o usuário efetuar o cadastramento de todos os dados e estes estiverem corretos, o usuário poderá efetuar o login com seu nome e senha através da tela de login, reproduzida na Figura 14.

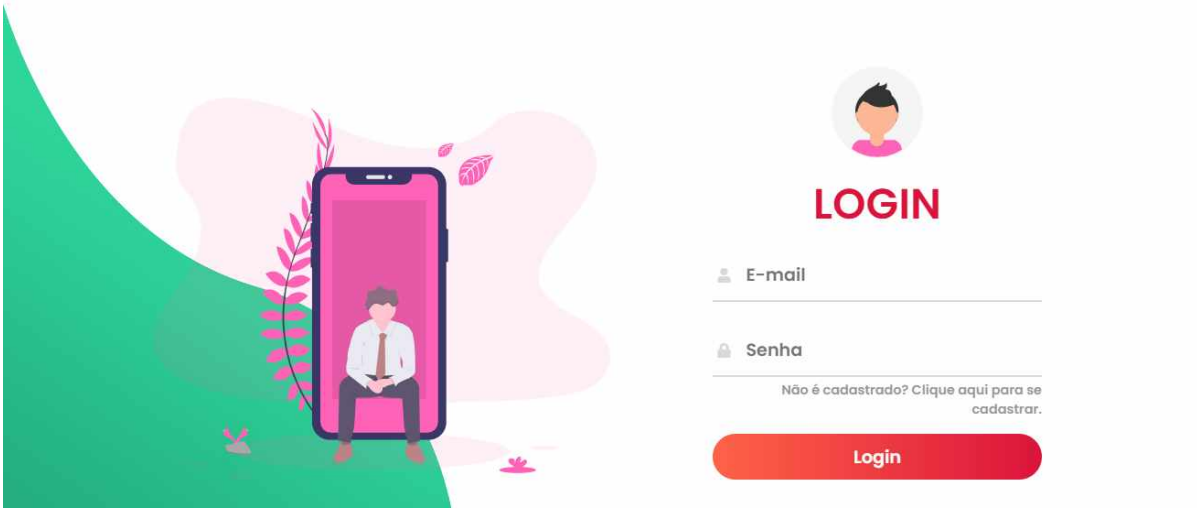


Figura 14 – Reprodução da tela login.

Fonte: Autoria própria.

Na função Ajax referente ao login será verificado se as informações correspondem às presentes no banco de dados. Pode-se verificar, através da Figura 15 que apresenta a função de login, que se as informações estiverem corretas, o usuário será redirecionado de acordo com o seu perfil, para a página de atividades se for aluno, e para a página de turmas se for professor.

```
$.ajax({
  type: "POST",
  url: "../controller/UsuarioController.php",
  data: {
    login: 1,
    email,
    senha,
  },
  error: function(jqXHR, textStatus, errorThrown) {
    console.log({
      jqXHR,
      textStatus,
      errorThrown
    })

  },
  success: function(data) {
    var response = JSON.parse(data);

    if (response.res) {
      const user = response.user;

      localStorage.setItem("user_email", user.email);
      localStorage.setItem("user_name", user.nome);
      localStorage.setItem("user_perfil", user.perfil);

      if (user.perfil === "aluno") {
        location.href = "atividades.php";
      } else {
        location.href = "turmas.php";

      }

      console.log(user);
    } else {
      alert("Usuário ou senha inválidos!");
    }
  }
});
```

Figura 15 – Representação da função de login.

Fonte: Autoria própria.

A página `mais.php` é responsável por apresentar informações sobre o sistema, o contato com o desenvolvedor do sistema e, caso o usuário queira enviar alguma dúvida ou sugestão sobre o sistema, basta preencher o formulário e enviar a sua informação (Figura 16).

Figura 16 – Reprodução da página mais.

Fonte: Autoria própria.

A funcionalidade `menu.php` é responsável por mostrar as opções que o usuário pode acessar no sistema, e existem dois menus separados, um para o aluno e outro para o professor. A diferença entre eles é que no menu de professor o mesmo pode criar uma turma e adicionar alunos, já na turma do aluno ele se cadastra em uma turma criada pelo professor. Toda essa tratativa é feita utilizando o e-mail inserido na Sessão. O modelo escolhido no sistema de menu foi o menu-hambúrguer. A função responsável por verificar o perfil do usuário está representada na Figura 17.

```php
<?php
include_once "menu.php";
include_once "../model/Usuario.php";
session_start();


if(!isset($_SESSION["user"])){
    header("Location: ../index.php");
}

        $usuario = $_SESSION["user_perfil"];


if($usuario == 'professor'){
    navProfessor();
}else{
    navAluno();
}
?>
```

Figura 17 – Representação da função de Sessão utilizada para verificar o perfil do usuário, e redirecioná-lo para o menu correspondente.

Fonte: Autoria própria.

A tela `turmas.php` é usada tanto pelo professor quanto pelo aluno. A funcionalidade do professor é utilizada para a criação de uma turma e listagem das turmas criadas pelo professor. No *card* é mencionado o nome da Turma, onde ele poderá ir para a tela de Atividades da Turma ou listar o rendimento dos alunos cadastrados na turma. O botão nova turma para cadastro de uma nova turma solicita que o professor escolha um código e senha, e estes deverão ser repassados aos alunos para o cadastro na turma. Estes detalhes podem ser observados na Figura 18.

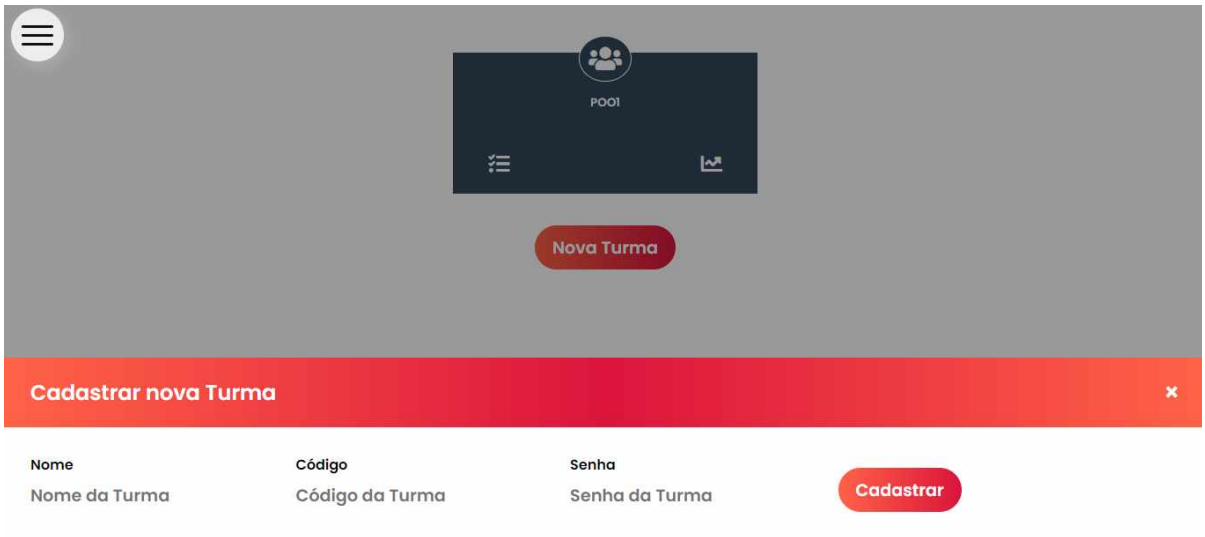


Figura 18 – Reprodução da tela Turma no perfil de professor.

Fonte: Autoria própria.

Na tela turmas de aluno terá o *card* que mostrará o nome da turma que ele está cadastrado, opções de listar as atividades da turma, mostrar o seu rendimento e sair daquela turma. O botão nova turma serve para ele se vincular a uma nova turma, e o código e a senha devem ser passados pelo seu professor. As possibilidades para o aluno podem ser observadas na Figura 19.

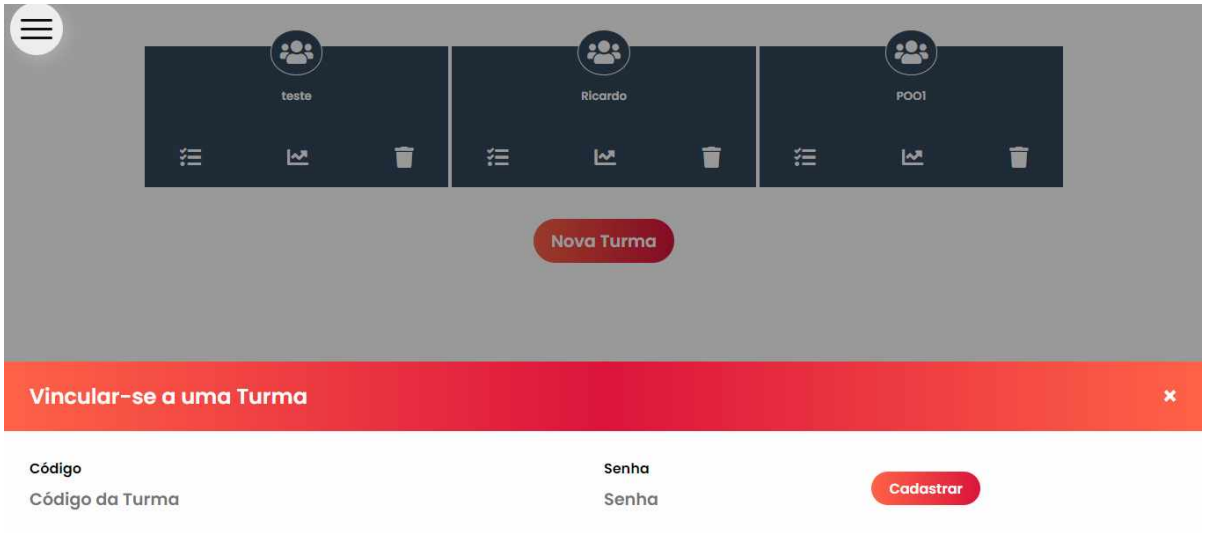


Figura 19 – Reprodução da tela Turma no perfil de aluno.

Fonte: Autoria própria.

Através da tela de atividades (arquivo `atividades.php`) reproduzida na Figura 20 o usuário pode escolher o tema sobre o qual ele deseja realizar o quiz, que é um teste de múltipla escolha com tempo determinado para responder cada questão. Todas as perguntas estão cadastradas na tabela `Atividades`, onde o nome será a descrição da tarefa. Atualmente estão disponíveis sete atividades de assuntos diversos dentro do paradigma orientado a objetos: Classes e Objetos, Métodos, *Arrays* e *ArrayLists*, Mais sobre Classes e Objetos, Herança, Polimorfismo e Tratamento de exceções.

Para realizar o quiz, basta o usuário clicar no botão Iniciar Quiz, ler as instruções e responder ao quiz. Ao finalizar, será apresentada a tela de pontuação, e um feedback de como está o aprendizado do aluno neste tema.

Figura 20 – Reprodução da tela de Atividades.

Fonte: Autoria própria.

Para a inserção das atividades, foi utilizada uma chamada Ajax, representada na Figura 21, e assim que a página Atividades é carregada, ela busca no banco de dados todas as atividades inseridas e lista estas atividades com o seu ID e sua descrição. A busca pelas perguntas se dá pela turma cadastrada no sistema.

```
<script>
  $(document).ready(function () {
    $.ajax({
      url: "../controller/AtividadeController.php",
      type: "GET",
      data: {
        getByTurmaCodigo: 123,
      },
      success: function (data) {
        const atividades = JSON.parse(data)
        atividades.forEach(({ id, descricao }, i) => {
          $(".events ul").append(`
            <li>
              <div class="time">
                  <h2>  <br> <span>${i + 1}</span></h2>
              </div>
              <div class="details">
                <h3>${descricao}</h3>
                <a href="./perguntas.html?ativ=${id}">Iniciar Quiz</a>
              </div>
            </li>
          `)
        });
      }
    })
  });
</script>
```

Figura 21 – Representação da função Ajax, utilizada para retornar as atividades cadastradas no banco de dados.

Fonte: Autoria própria.

A tela em `perguntas.php`, que é disponibilizada assim que o usuário clica para iniciar o quiz, contém algumas regras de funcionamento que são mostradas ao usuário antes do efetivo início do quiz (Figura 22).

Figura 22 – Reprodução da tela inicial de Perguntas contendo as instruções e regras do quiz.

Fonte: Autoria própria.

O início efetivo do quiz acontece com a listagem das perguntas e alternativas, e o usuário conta com um tempo de 80 segundos para responder cada questão; se este tempo acabar, o sistema mostra qual a alternativa correta e passa para a próxima questão. Todas as questões são inicializadas aleatoriamente e, ao final de cada quiz, a nota do aluno e seu aproveitamento são disponibilizados. Um exemplo de reprodução da tela de perguntas está apresentado na Figura 23.

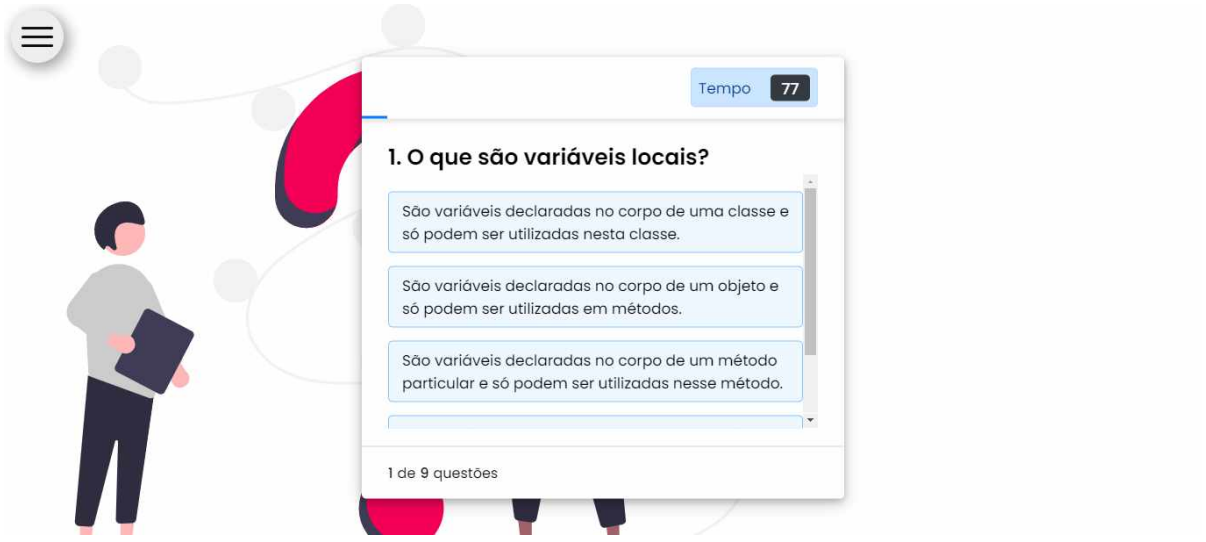


Figura 23 – Reprodução da tela de perguntas após o início efetivo do quiz.

Fonte: Autoria própria.

A tela `rendimento.php` é responsável por listar as notas dos alunos que responderam o quiz. Terá informações sobre o id do aluno, seguido com o número da atividade que

respondeu e suas notas são listadas, e tanto aluno quanto professor conseguem acessar esta página. As notas são atualizadas sempre com a maior das notas, e os nomes listados são de todos os alunos cadastrados naquela turma. Nesta tela o aluno pode comparar o seu rendimento com os outros alunos da turma, incentivando a competição.

A tela `vídeos.php` é responsável por disponibilizar vídeos relacionados a programação orientada a objetos e Java. O início desta tela está reproduzido na Figura 24. Estes vídeos explicativos englobam conteúdo desde a instalação da ferramenta necessária para a programação, passando por explicações da teoria e exemplos práticos, explicando como funciona um código, e detalhando a programação orientada a objetos. No total foram disponibilizados dezoito vídeos, sendo nove vídeos sobre Java e seu desenvolvimento e nove vídeos sobre a teoria de POO.

Todos os vídeos disponibilizados estão disponíveis no canal do Gustavo Guanabara na playlist sobre POO, <https://www.youtube.com/watch?v=KlIL63MeyMY&list=PLHz_AreHm4dkqe2aR0tQK74m8SFe-aGsY>.

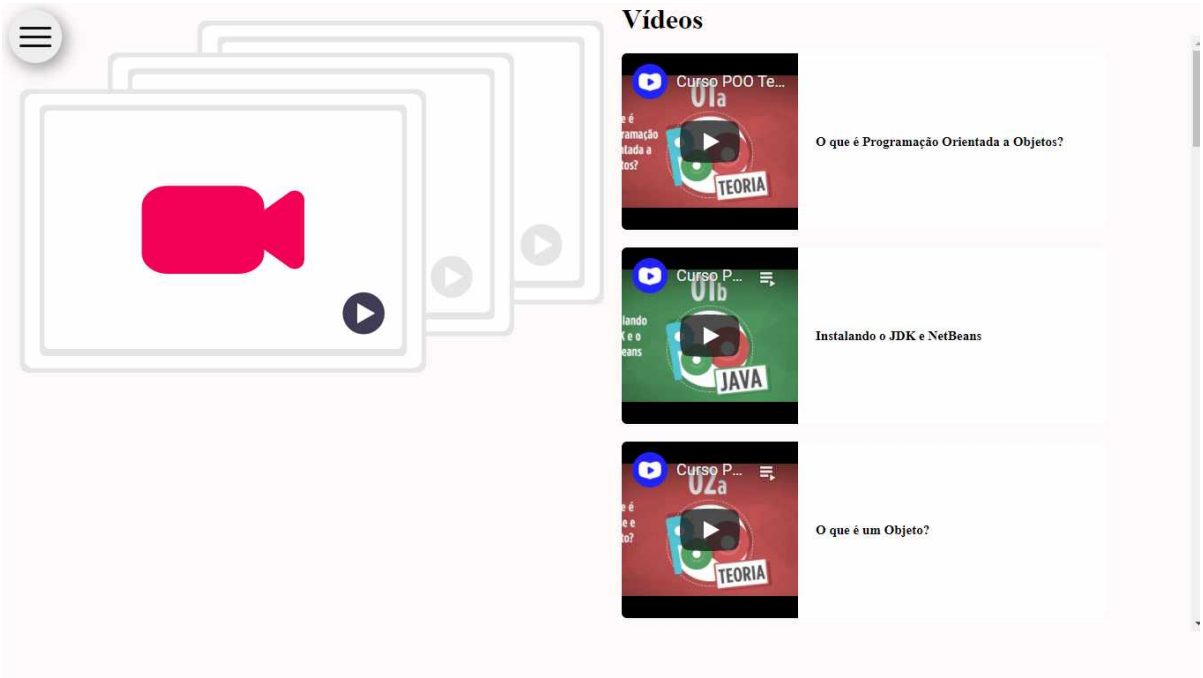


Figura 24 – Reprodução da tela Vídeo.

Fonte: Autoria própria.

4.1.2 Camada *model*

A camada *model* (modelo) é responsável pela representação das tabelas do banco de dados e por definir como as entidades do banco estão sendo mapeadas. A implementação da camada *Data access object* (DAO) trabalha juntamente com o *model* no objetivo de definir como as tabelas do banco de dados serão manipuladas. A pasta *model* está representada na Figura 25.

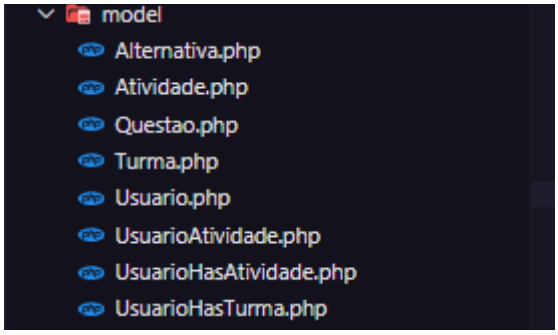


Figura 25 – Representação da pasta *model*.

Fonte: Autoria própria.

4.1.3 Camada *controller*

A camada *controller* (controladora) é responsável por processar as requisições feitas pelos usuários, sendo a principal encarregada por controlar todo o fluxo do programa. Na camada *controller* é criado um controlador para cada uma das principais entidades do banco, onde é necessário mapear quais tipos de requisições HTTP ele atende. O controlador da entidade usuário, o qual pode receber requisições do tipo GET e POST. Cada uma dessas requisições realiza manipulações diferentes na base de dados, com requisições do tipo GET buscando informações, e requisições do tipo POST criando novas entradas nas tabelas.

4.1.4 DAO

O padrão DAO é o responsável pelo acesso ao banco de dados e pelo *Create, Read, Update, Delete* (CRUD) do sistema, como mostra a Figura 26.

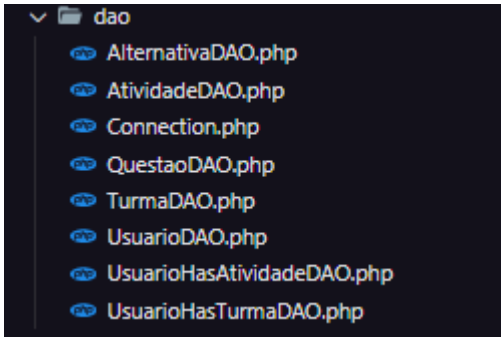


Figura 26 – Representação do DAO.

Fonte: Autoria própria.

Foram criados os métodos CRUD, com a finalidade de criar uma nova tupla na tabela de sua entidade, buscar tuplas na tabela de acordo com parâmetros estabelecidos pelo método, atualizar uma ou várias tuplas de uma tabela e excluir uma ou várias linhas de uma

tabela, respectivamente. É possível perceber uma relação sendo estabelecida entre cada um dos tipos de requisições do controlador e cada um dos métodos das classes DAO. Essa relação é dada pelas requisições do tipo GET que tem como objetivo buscar informações da base de dados, e, para isso, o controlador invoca métodos READ de classes DAO para acessar tais informações, assim como requisições POST chamam métodos CREATE, UPDATE e DELETE. A função de Create foi utilizada, por exemplo, no cadastro do usuário e cadastro de turmas. A função Read é utilizada para buscar informações armazenadas no banco de dados, sendo utilizada nas seguintes funcionalidades:

- ❏ Login: Verificação se o usuário está informando corretamente suas credenciais para acessar a plataforma.
- ❏ Cadastro de usuário: Verificação se o e-mail informado já está sendo utilizado.
- ❏ Turmas: Listar todas as turmas do usuário logado.
- ❏ Atividades: Buscar todas as questões e alternativas de uma atividade.
- ❏ Rendimento: Listar a pontuação dos alunos que responderam ao quiz.

A função UPDATE é responsável por atualizar as informações da tela de editar perfil, onde o usuário atualizará seus dados cadastrais. Além disso, ela também foi utilizada para atualizar a nota do usuário ao finalizar uma atividade.

A função DELETE é responsável por remover um aluno de uma turma na qual ele esteja cadastrado.

4.1.5 *Assets*

Assets são responsáveis por guardar arquivos de extensões css, js e bibliotecas utilizadas ao longo do desenvolvimento do programa. No trabalho foram utilizadas as fontes do Google, que são disponibilizadas pelo Google Fonts [1].

Todas as imagens utilizadas na parte visual do programa são extensão SVG e estão disponíveis publicamente [2].

4.2 Elementos de Gamificação

A gamificação está presente em dois elemento, sendo o primeiro elemento encontrado na parte na qual o aluno consegue realizar o quiz e aplicar o conhecimento adquirido ao longo do aprendizado de POO, tendo um tempo limitado para a solução da questão (80 segundos para indicar a alternativa correta). Caso ele erre a questão, ele poderá conferir

[1] <https://fonts.google.com/>
[2] <https://undraw.co/>

qual alternativa está correta e conferir o porque do erro daquela questão. Quando ele ver e entender o porque do erro, poderá novamente jogar o quiz e ver se realmente aprendeu a questão que ele errou. O quiz está disponível para o aluno jogar diversas vezes, utilizando o conceito de aprender com os seus próprios erros, e tendo sempre a possibilidade de verificar o quanto ele está aprendendo em um determinado assunto.

O segundo elemento aparece na tela de rendimentos, que mostra a ele em quais tópicos ele está com o aprendizado aguçado e em quais está mais precário juntamente com os resultados dos outros alunos da sua turma, o que incentiva a competitividade saudável entre eles. O professor também tem acesso a página rendimento, e, caso ele verifique que os alunos estão com notas muito baixas em um determinado tópico, o mesmo poderá planejar alguma forma de reforçar o aprendizado daquele tópico específico.

4.3 Disponibilização e hospedagem do sistema

O sistema desenvolvido está disponível por meio do GitHub[3], e esta disponibilização inclui todo o código fonte utilizado ao longo do trabalho.

Além da disponibilização do código fonte, o Sistema também está disponível em um servidor e pronto para utilização[4]. O sistema foi hospedado de forma gratuita pelo site da *Infinity Free*[5], porém na versão gratuita algumas funcionalidades estão indisponíveis. Como exemplo, no trabalho havia sido utilizado na camada *Controller* o PUT para atualização dos dados do usuário, e a função DELETE, para o aluno sair de uma turma; mas, como o site restringe este uso apenas para domínios pagos, os métodos foram alterados. A utilização da hospedagem do site em si é bem fácil e intuitiva, e alguns canais do *YouTube* oferecem um mini tutorial explicando como fazer a hospedagem de sites.

[3] <https://github.com/amandadduraes/tcc2-site>
[4] <http://aprendendopoo.infinityfreeapp.com/?i=1>
[5] <https://infinityfree.net/>

Capítulo 5

Conclusão

Neste trabalho foi abordado o problema das dificuldades enfrentadas por alunos ao longo do processo de aprendizagem do paradigma orientado a objetos, e que acabam gerando um aumento das taxas de evasão e reprovação nas disciplinas relacionadas à este paradigma. Este aumento tem influenciado muitos pesquisadores a buscarem soluções que melhorem o aprendizado incentivando os alunos.

Para auxiliar na redução destas taxas de evasão e reprovação, um Sistema Web foi criado e disponibilizado para alunos e professores apresentando um conteúdo organizado, materiais complementares e atividades em formato de quiz juntamente com o acompanhamento e retorno do rendimento do aluno ao longo do aprendizado deste paradigma de programação.

Alunos e professores podem se beneficiar do uso do sistema: por um lado o aluno deve conseguir manter um melhor engajamento na disciplina através dos recursos de gamificação (competição e limitações de tempo), além do acesso a um conteúdo organizado e relevante para o seu aprendizado e do *feedback* instantâneo fornecido através das atividades (quiz); por outro lado, o professor deve conseguir entender melhor os tópicos em que a turma possui mais dificuldade, conseguindo realizar um melhor planejamento sobre o conteúdo que será abordado na disciplina e possivelmente alterando este planejamento para reforçar conteúdos não compreendidos.

Assim, o sistema desenvolvido apresenta uma utilização intuitiva e com grande potencial para auxiliar alunos e professores no processo de ensino-aprendizagem dos conceitos da programação orientada a objetos, tanto incentivando os alunos a compreenderem melhor o conteúdo para obterem um melhor desempenho comparado aos seus colegas de turma quanto auxiliando o professor a identificar onde estão as dificuldades da turma para poder saná-las em tempo hábil para evitar a evasão e a reprovação.

O sistema já está funcional, mas como trabalhos futuros podemos considerar acrescentar algumas funcionalidades para que mais benefícios possam advir da utilização do sistema. Algumas opções que poderiam aumentar a utilidade do sistema incluem:

- ❏ Inclusão de novas disciplinas no sistema;
- ❏ Inclusão de funcionalidade que permite ao professor inserir perguntas de seu interesse para suas turmas;
- ❏ Criação de uma versão mobile do sistema para que os usuários possam utilizar dispositivos móveis para acesso à ferramenta.

Referências

AGUIAR, J. Experiência baseada em gamificação no ensino sobre herança em programação orientada a objetos. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. [S.l.: s.n.], 2015. v. 4, n. 1, p. 1444. Citado 2 vezes nas páginas 22 e 23.

BARANAUSKAS, M. C. C. Procedimento, função, objeto ou lógica? linguagens de programação vistas pelos seus paradigmas. **Computadores e Conhecimento: Repensando a Educação. Campinas, SP, Gráfica Central da Unicamp**, 1993. Citado 2 vezes nas páginas 12 e 22.

BARROS, E. M. D. de; FREIBERGER, E. C. O paradigma do desenvolvimento orientado a objetos a sua dificuldade de aprendizado. **Profiscientia**, n. 3, 2008. Citado na página 22.

BELTRAME, W. Projeto ágil de aplicativo como mediação da aprendizagem sobre orientação a objetos. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. [S.l.: s.n.], 2018. v. 7, n. 1, p. 669. Citado na página 15.

BRAZIL, A.; BARUQUE, L. Gamificaçaõ aplicada na graduação em jogos digitais. In: **Simpósio Brasileiro de Computadores na Educação (Simpósio Brasileiro de Informática na Educação - SBIE)**. [S.l.: s.n.], 2015. v. 26, n. 1, p. 677. Citado 3 vezes nas páginas 12, 22 e 23.

COSTA, A. F. F. et al. **Aplicação de Sala Invertida e Elementos de Gamificação para Melhoria do Ensino-Aprendizagem em Programação Orientada a Objetos**. [S.l.]: TISE, 2017. Citado 4 vezes nas páginas 12, 22, 23 e 24.

FARINELLI, F. Conceitos basicos de programação orientada a objetos. **Instituto Federal Sudeste de Minas Gerais**, 2007. Citado 6 vezes nas páginas 15, 16, 17, 18, 19 e 21.

JUNGTHON, G.; GOULART, C. M. Paradigmas de programação. **Monografia (Monografia)—Faculdade de Informática de Taquara, Rio Grande do Sul**, v. 57, 2009. Citado na página 21.

LEITE, T. et al. **Orientação a Objetos: Aprenda seus conceitos e suas aplicabilidades de forma efetiva**. [S.l.]: Editora Casa do Código, 2016. Citado 2 vezes nas páginas 17 e 18.

MACHADO, L. D. P. et al. Uma abordagem colaborativa para aprendizagem de programação orientada a objetos. In: SBC. **Anais Principais do XIII Simpósio Brasileiro de Sistemas Colaborativos**. [S.l.], 2016. p. 320–333. Citado na página 12.

RICARTE, I. L. M. Programação orientada a objetos: uma abordagem com java. **http://www. dca. fee. unicamp. br/cursos/PooJava/Aulas/poojava. pdf> Acesso em 16/08/2020**, v. 29, n. 10, p. 2014, 2001. Citado 3 vezes nas páginas 16, 18 e 19.