Universidade Federal de Uberlândia

Pedro Henrique Fernandes de Oliveira

Desenvolvimento de uma Plataforma Gamificada para Estudos com Extração Automatizada de Questões do ENEM e Integração com Inteligência Artificial

Uberlândia

2025

Pedro Henrique Fernandes de Oliveira

Desenvolvimento de uma Plataforma Gamificada para Estudos com Extração Automatizada de Questões do ENEM e Integração com Inteligência Artificial

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Fabiano Azevedo Dorça

Uberlândia

2025

Pedro Henrique Fernandes de Oliveira

Desenvolvimento de uma Plataforma Gamificada para Estudos com Extração Automatizada de Questões do ENEM e Integração com Inteligência Artificial

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Data de Aprovação: 23/09/2025

Banca Examinadora

Prof. Dr. Fabiano Azevedo Dorça (orientador)
(Universidade Federal de Uberlândia)

Profª. Drª. Maria Adriana Vidigal de Lima
(Universidade Federal de Uberlândia)

Prof. Dr. Rafael Dias Araújo
(Universidade Federal de Uberlândia)

Dedicatória

Dedico este trabalho aos meus pais, Alessandra e Eduardo, que me apoiaram ao longo do caminho.

Agradecimentos

Agradeço aos meus pais por sempre me apoiarem.

Agradeço ao meu orientador Fabiano Azevedo Dorça pelo aprendizado e sobretudo pelo incentivo na conclusão do curso.

Por fim, agradeço a todos amigos e professores que tive o prazer de conhecer ao longo da graduação.

"If you thought that science was certain – well, that is just an error on your part."

Richard Feynman

Resumo

Este trabalho apresenta o desenvolvimento de uma plataforma web educacional gamificada, destinada ao estudo para o Exame Nacional do Ensino Médio (ENEM). A plataforma automatiza a extração de questões, alternativas e gabaritos a partir do PDF do caderno de prova do ENEM, utilizando técnicas de processamento de texto com expressões regulares (regex) em Python. O sistema é composto por um backend desenvolvido com Django e Django Rest Framework, que disponibiliza APIs RESTful para gerenciamento de usuários, questões e a lógica de jogo, incluindo autenticação segura via JWT. A persistência de dados é realizada em um banco PostgreSQL. O frontend, construído com React, oferece uma interface interativa onde os usuários avançam por um sistema de dez fases, consomem energia para acessar questões e acumulam cristais para solicitar dicas contextuais geradas pela integração com a API do ChatGPT. Elementos gamificados como ranking global e conquistas foram implementados para aumentar o engajamento. Os resultados demonstraram eficácia da extração via regex para o formato específico do ENEM, com todas as 95 questões e 475 alternativas sendo corretamente categorizadas. Contudo, a abordagem mostrou-se dependente da estrutura do PDF e inviável para a extração robusta de imagens, apontando para a necessidade de soluções baseadas em IA (Inteligência Artificial) em trabalhos futuros. A plataforma funcional serve como uma base sólida para evoluções em personalização do aprendizado e sistemas tutores inteligentes, destacando o potencial da interseção entre gamificação, automação e inteligência artificial na educação.

Palavras-chave: gamificação; educação; extração de dados; Django; React; ChatGPT.

Abstract

This work presents the development of an educational gamified web platform, designed for studying for the Brazilian National High School Exam (ENEM). The platform automates the extraction of questions, alternatives and answer keys from the PDF the ENEM, using text processing techniques with regular expressions (regex) in Python. The system consists of a backend developed with Django and Django REST Framework, which provides a RESTful API for managing users, questions, and the game logic, including secure JWT authentication. Data persistence is handled by a PostgreSQL database. The frontend, built with React.js, offers an interactive interface where users progress through a system of ten phases, consume energy to access questions, and accumulate crystals to request contextual hints generated by integration with the ChatGPT API. Gamification elements such as a global ranking and achievements were implemented to increase engagement. The results demonstrated the effectiveness of regex extraction for the specific format of the ENEM, with all 95 questions and 475 alternatives being correctly categorized. However, the approach proved to be dependent on the PDF's structure and unfeasible for the robust extraction of images, pointing to the need for AI-based solutions in future work. The functional platform serves as a solid foundation for evolution in learning personalization and intelligent tutoring systems, highlighting the potential of the intersection between gamification, automation, and artificial intelligence in education.

Keywords: gamification; education; ENEM; data extraction; Django; React; ChatGPT.

Lista de ilustrações

Lista de abreviaturas e siglas

ENEM	*Exame Nacional do Ensino Médio*
IA	*Inteligência Artificial*
API	*Application Programming Interface*
JWT	*JSON Web Token*
JSON	*JavaScript Object Notation*
RFC	*Request for Comments*
PDF	*Portable Document Format*
OCR	*Optical Character Recognition*
PBL	*Points, Badges, Leaderboards*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
ORM	*Object Relational Mapper*
SQL	*Structured Query Language*
INEP	*Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira*

Sumário

1 Introdução

Pesquisas contemporâneas na área de tecnologia educacional indicam que a intersecção entre tecnologia e educação tem redefinido paradigmas tradicionais de aprendizagem, propiciando ambientes dinâmicos e interativos que cativam e engajam discentes (SALEN; ZIMMERMAN, 2004; PRENSKY, 2001). Neste contexto, a gamificação emerge como uma estratégia pedagógica poderosa, baseando-se nos princípios de design de jogos para transformar atividades educacionais em experiências imersivas e motivadoras. Tal qual descrito no livro *Reality is Broken* (MCGONIGAL, 2011), jogos possuem o potencial de mobilizar competências cognitivas e emocionais, estimulando a resiliência e a curiosidade – elementos fundamentais para o aprendizado efetivo.

Partindo dessa perspectiva, que destaca o poder motivacional dos elementos de jogos, o presente trabalho tem como objetivo o desenvolvimento de uma plataforma educacional gamificada. A proposta concretiza-se em um sistema com estrutura de fases, inspirado em aplicativos como o Duolingo (https://pt.duolingo.com/), e incorpora três funcionalidades principais: a extração automatizada de questões, alternativas e gabaritos a partir do PDF da prova do ENEM (Exame Nacional do Ensino Médio); a integração com inteligência artificial (IA) para fornecer dicas e esclarecer dúvidas aos usuários durante a resolução das questões e a implementação de um ambiente gamificado com sistema de fases progressivas.

Desta forma, este trabalho apresenta contribuições práticas e teóricas, servindo tanto como uma ferramenta potencialmente útil para estudantes quanto como um protótipo e base de código para pesquisas e desenvolvimentos futuros na intersecção entre educação, automação, gamificação e inteligência artificial.

1.1 Motivação

A motivação deste trabalho é o de inspirar desenvolvimentos de plataformas de estudos acessíveis e gamificadas, desenvolvendo uma de código-aberto que sirva como protótipo funcional e base para futuras evoluções na intersecção entre educação, tecnologia e inteligência artificial. O trabalho se justifica também pela automação da

extração de questões, na intenção de povoar bancos de dados que sirvam bem as aplicações educacionais. Por fim, a integração com a API do ChatGPT para geração de dicas contextualizadas introduz uma camada de suporte inteligente e personalizado, incentivando outros trabalhos educacionais a inserir maior adaptabilidade com a utilização da inteligência artificial.

1.2 Objetivo Geral

Desenvolver plataforma web gamificada para estudos, capaz de extrair automaticamente questões de um PDF do caderno de prova do ENEM, disponibilizar um sistema de fases, conquistas, ranking, login e cadastro de usuários e integração com inteligência artificial.

1.3 Objetivos Secundários

- Implementar um módulo que extraia de maneira automatizada, identificando e categorizando corretamente o número, o enunciado, as alternativas e as imagens atreladas as questões.
- Desenvolver um sistema de autenticação seguro utilizando *JSON Web Tokens* (JWT), *login* via e-mail e senha.
- Projetar uma arquitetura *backend* para gerenciamento de questões, usuários e progresso.
- Implementar um *frontend* com interface de fases progressivas.
- Integrar a API do *ChatGPT* para geração de dicas contextualizadas para cada questão.
- Implementar sistema de energia, cristais, conquistas e ranking.

1.4 Organização do Documento

Este capítulo apresentou a introdução do trabalho, contextualizando o tema, delineando os objetivos gerais e específicos. Os capítulos subsequentes estão organizados da seguinte forma:

O Capítulo 2 - Revisão Bibliográfica fundamenta teoricamente o projeto, apresentando os conceitos-chave que sustentam o desenvolvimento, como expressões regulares, autenticação JWT, ORM, APIs REST e gamificação. Além disso, analisa trabalhos correlatos nas áreas de extração de dados de PDF e gamificação na educação, posicionando a contribuição deste trabalho no estado da arte.

O Capítulo 3 - Abordagem Proposta detalha a implementação prática da solução. É dividido em três seções principais: a explicação do processo de extração de dados do PDF e modelagem do banco de dados; a descrição da arquitetura do *backend* e de todas as APIs RESTful desenvolvidas; e a apresentação do *frontend* e de suas telas, ilustrando a jornada do usuário na plataforma gamificada.

O Capítulo 4 - Conclusão encerra o trabalho, sintetizando os resultados alcançados, reconhecendo as limitações encontradas e propondo direcionamentos futuros para a evolução da plataforma e do método de extração.

2 Revisão Bibliográfica

Neste capítulo, são apresentados os conceitos relevantes para a implementação deste trabalho, assim como os trabalhos e aplicações correlacionados. Os conceitos básicos para entendimento de cada parte do sistema são introduzidos aqui, e posteriormente em outros capítulos as explicações mais detalhadas do funcionamento do sistema em si são exploradas.

2.1 Extração de Questões e Alternativas com Expressões Regulares

A extração automatizada de dados estruturados a partir de documentos não estruturados, como arquivos PDF, representa um desafio significativo no processamento de linguagem natural e na engenharia de software. Conforme destacado em *Mastering Regular Expressions* (FRIEDL, 2006), expressões regulares (*regex*) são ferramentas fundamentais para reconhecer padrões textuais complexos, permitindo a identificação precisa de elementos semânticos em fluxos de dados brutos. No contexto deste trabalho, a extração das questões, alternativas e gabaritos do ENEM 2023 se baseou intensamente nessa técnica, aliada à biblioteca PyPDF2 para a leitura do conteúdo textual dos PDFs. A escolha do *regex* em detrimento a outras ferramentas se deu mais pela experiência profissional prévia do pesquisador com expressões regulares.

As expressões regulares são padrões utilizados para realizar buscas e manipulações em cadeias de caracteres. Elas permitem identificar trechos de texto que correspondem a regras específicas, como números, palavras, formatos de data, endereços de e-mail, entre outros. No contexto deste trabalho, foi utilizada para reconhecer e extrair corretamente o número da questão, as alternativas, a quantidade de questões, a matéria ou assunto de um conjunto de questões.

Na linguagem *Python*, usada na implementação do comando de extração de questões e alternativas do ENEM, o módulo re fornece um conjunto de funções para trabalhar com expressões regulares. Entre as mais comuns, se destacam:

• *re.findall*(padrão, *string*): retorna todas as ocorrências de um padrão dentro de uma *string*;

• *re.split*(padrão, *string*): divide uma *string* em partes, utilizando o padrão como delimitador;

• *re.sub*(padrão, *sub*, *string*): substitui trechos do texto (*string*) que correspondem ao padrão por um valor de substituição (*sub*).

Um exemplo prático da aplicação de *regex* neste projeto pode ser visto nas seguintes linhas de código na Figura 1:

Na primeira linha, o padrão procura pela palavra "QUESTÃO" seguida de um espaço e de um ou mais dígitos. O uso do parênteses em (\d+) cria um grupo de captura, ou seja, apenas o número da questão é retornado. Assim se o texto contiver "QUESTÃO 15" e "QUESTÃO 23" por exemplo, os números 15 e 23 serão extraídos para a lista *question_ numbers*. Na segunda linha, *questions* será uma lista com os textos entre o padrão estabeleci do, sendo composto então pelo enunciado e alternativas. Para melhor entendimento, se o texto for composto por "QUESTÃO 15 será QUESTÃO 23 talvez", a lista *questions* seria com posta por ['', ' será ', ' talvez ']. Esse uso de expressões regulares mostra como elas podem ser aplicadas para organizar dados textuais não estruturados, como provas digitalizadas, PDFs, os transformando em estruturas úteis para processamento posterior.

Figura 1 - Exemplo do uso de expressão regular

```
question_numbers = re.findall(r'QUESTÃO (\d+)', text)
questions = re.split(r'QUESTÃO \d+', text)
```

Fonte: Elaboração Própria.

2.2 Autenticação e Autorização com JWT

A segurança de aplicações web modernas, especialmente aquelas que manipulam dados de usuários, depende fundamentalmente de mecanismos robustos para verificar a identidade do usuário (autenticação) e para definir o que um usuário autenticado tem permissão de fazer (autorização).

Conforme definido formalmente em *RFC 7519* (JONES; BRADLEY; SAKIMURA, 2015), o JWT (*JSON Web Token*) emergiu como a solução predominante para

implementar autenticação do tipo *stateless* (sem estado, o que significa que o *backend* não guarda nenhum estado para o JWT, já que ele em si já é suficiente) em APIs (Interface de Programação de Aplicações)REST (Transferência de Estado Representacional), conceitos que serão abordados posteriormente.

Tecnicamente, um JWT é um objeto JSON compacto e autocontido, codificado como uma string segura. Seguindo a especificação, sua estrutura é dividida em três partes, codificadas em *Base64Url* e separadas por pontos:

• Cabeçalho (*header*): Normalmente consiste em duas partes: o tipo de token, que é JWT, e o algoritmo de assinatura utilizado, como HMAC, SHA256 ou RSA. Exemplo decodificado: {. "*alg*": "HS256", "*typ*": "JWT" }

• Payload: Contém as declarações (claims). São declarações sobre uma entidade, normalmente o usuário e metadados adicionais. Exemplo decodificado: { "*userid*": "12345", "*name*": "John", "*admin*": true, }

• Assinatura: Ela é usada para verificar se o remetente do JWT é quem diz ser e para garantir que a mensagem não foi alterada ao longo do caminho. Ela é criada assinando o cabeçalho codificado, a payload codificada e um segredo.

No ecossistema Django deste projeto, a biblioteca *djangorestframework-simplejwt* foi a escolha para implementar o padrão *RFC 7519*. Ela foi configurada como a classe padrão de autenticação no arquivo *settings.py*, instruindo o *Django REST Framework* a utilizar *tokens* JWT como o mecanismo central de segurança para a API, que serve como *backbone* (espinha dorsal) de comunicação entre o *frontend* e o *backend,* visto na Figura 2.

Figura 2 - Configuração *SimpleJWT*

```
# backend/tcc/settings.py

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ),
}
```

Fonte: Elaboração Própria.

O SimpleJWT opera fornecendo *endpoints* pré-construídos e seguros para a geração e renovação de *tokens*, conforme espeficificado pelo padrão. Esses *endpoints* foram mapeados nas rotas principais do projeto (Figuras 3 e 4)

Figura 3 - Principais URLs

```python
# backend/tcc/urls.py

urlpatterns = [
  path('admin/', admin.site.urls),
  path('api/auth/', include('accounts.urls')),
  path('api/auth/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
  path('api/auth/registration/', include('dj_rest_auth.registration.urls')),
  path('api/questions/', include('questions.urls')),
  path('api/game/', include('game.urls')),
]
```

Fonte: Elaboração Própria.

Figura 4 - URL de *Login*

```python
# backend/accounts/urls.py

from django.urls import path

from .views import MyTokenObtainPairView


urlpatterns = [
  path('login/', MyTokenObtainPairView.as_view(), name='token_obtain_pair'),
]
```

Fonte: Elaboração Própria.

2.3 Mapeamento Objeto-Relacional (ORM) e Django REST Framework

O Mapeamento Objeto-Relacional (ORM) é uma técnica de programação que permite consultar e manipular dados de um banco de dados relacional utilizando diretamente o paradigma orientado a objetos de uma linguagem. Em vez de escrever SQL diretamente, o desenvolvedor interage com classes e objetos, e o framework ORM se encarrega de traduzir essas operações em instruções SQL otimizadas e específicas para o banco de dados em uso, no caso PostgreSQL.

Conforme introduzido em *Patterns of Enterprise Application Architecture* (FOWLER, 2002), o padrão *Data Mapper* se trata de uma camada de isolamento que media a comunicação entre a lógica de negócio da aplicação e a camada de persistência em um banco de dados 13 relacional. A essência deste padrão é manter uma separação de responsabilidades estrita: os objetos de domínio podem ser desenvolvidos e se se comportarem de forma independente, sem qualquer conhecimento de como são armazenados, enquanto o *Data Mapper* é o único componente responsável por conhecer os detalhes do banco de dados e por realizar a tradução bidirecional entre registros em tabelas e objetos em memória.

Logo, os ORMs modernos, como o do *Django*, podem ser considerados uma materialização e evolução prática desse padrão conceitual. O *Django* ORM aplica os benefícios teóricos do padrão *Data Mapper*, como o aumento da produtividade, a facilidade de manutenção e a segurança inerente ao evitar a escrita manual de SQL, funcionando como o coração da camada de persistência deste projeto.

2.4 Interfaces de Programação de Aplicações (APIs) e o Estilo Arquitetural REST

Uma Interface de Programação de Aplicações (API – *Application Programming Interface*) é, em essência, um conjunto de regras, protocolos e ferramentas que define como componentes de software devem interagir. Ela atua como um contrato entre provedor de um serviço e seu consumidor, especificando como e quais requisições podem ser feitas, como as fazer, os formatos de dados e convenções a serem seguidas. No contexto de aplicações web, as APIs permitem que *frontends* (clientes) solicitem e manipulem dados em *backends* (servidores) de forma padronizada e independente da tecnologia subjacente de cada parte.

Para organizar e padronizar a construção de APIs web, o arquiteto de software Roy Fielding definiu, em sua tese de doutorado *Architectural Styles and the Design of Networkbased Software Architectures* (FIELDING, 2000), o estilo arquitetural REST (*Representional State Transfer*, ou Transferência de Estado Representacional): um modelo arquitetural projetado para sistemas de hipermídia distribuídos em larga escala, cujos princípios fundamentais incluem:

- Cliente-Servidor: Separação que permite que o cliente (*frontend*) e o servidor (*backend*) evoluam independentemente.

- *Stateless* (Sem estado): Cada requisição do cliente para o servidor deve conter toda a informação necessária para ser entendida e processada. O servidor não armazena nenhum estado de sessão do cliente entre requisições, o que aumenta a escalabilidade e a confiabilidade.

- Interface uniforme: define uma forma padronizada de comunicação entre componentes, baseada em URIs (Identificador Uniforme de Recurso) como /api/questions/123, em XML ou JSON (no caso deste projeto) como meio de representação de dados e uso dos métodos HTTP semânticos: GET, POST, PUT, PATCH, DELETE.

2.5 Fundamentos de Gamificação na Educação

A gamificação é uma estratégia que transcende o âmbito do entretenimento e tem sido sistematicamente incorporada em contextos educacionais para aumentar o engajamento e a motivação dos discentes. Tratada de forma abrangente em The *Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education* (KAPP, 2012), a gamificação na educação pode ser definida como o uso de elementos e mecânicas característicos de jogos em ambientes de aprendizagem não lúdicos, com o objetivo de influenciar comportamentos, enriquecer a participação e potencializar a aquisição de conhecimentos e competências. Alguns elementos apresentados por Kapp (2012) envolvem progressão, objetivos, recompensas, *feedback*, engajamento.

2.6 Trabalhos Correlatos

Sobre a extração de dados, no caso questões e alternativas de um arquivo PDF, um trabalho é particularmente relevante, intitulado *Automated Data Extraction From PDF Documents: Application to Large Sets of Educational Tests* (WIECHORK; CHARÃO, 2021). Nele, os autores desenvolveram uma metodologia abrangente para extração automatizada de dados 15 de provas educacionais em larga escala, utilizando um conjunto de 343 testes do ENADE (Exame Nacional de Desempenho dos Estudantes) com 11.196 questões objetivas e discursivas. A abordagem empregou múltiplas ferramentas especializadas: *Aletheia, CyberPDF, ExamClipper, Excalibur, PDFMiner, Tabula*, comparando os resultados obtidos pelas extrações. A principal diferença entre a abordagem acima e a implementada neste projeto reside na estratégia de

processamento e escopo de aplicação. Enquanto o trabalho correlato empregou múltiplas ferramentas especializadas para lidar com o caso geral de provas em PDF, o presente trabalho adotou uma estratégia mais focalizada, utilizando Expressões Regulares (*Regex*) e extraindo também o gabarito. Além disso, enquanto a pesquisa de Wiechork e Charão (2021) focou na extração e disponibilização dos dados brutos, o presente projeto avança na integração direta do processo de extração com uma plataforma de estudos gamificada, criando um ciclo completo desde a aquisição dos dados até sua aplicação em um ambiente interativo de aprendizagem.

Já sobre a questão da gamificação na educação, o artigo de revisão sistemática *Gamification of e-learning in higher education: a systematic literature review* (KHALDI; BOUZIDI; NADER, 2023) fornece um panorama abrangente do estado da arte da gamificação no ensino superior, destacando tendências, elementos mais utilizados e lacunas na literatura. Conforme identificado no artigo, os elementos de gamificação mais empregados em ambientes de aprendizagem do ensino superior são os componentes PBL (*Points, Badges, Leaderboards*), ou pontos, emblemas e tabelas de classificação. Este projeto se alinha diretamente com essa tendência, utilizando sistema de fases, ranking de classificação e página de conquistas, além de cristais utilizados para auxiliar a resolução da questão através de conexão com IA. Khaldi, Bouzidi e Nader (2023) também indicam a tendência emergente em direção à gamificação adaptativa, onde a experiência é personalizada de acordo com o perfil do aluno, o que não é implementado neste trabalho.

De forma mais alinhada ao escopo de desenvolvimento de uma plataforma gamificada completa, o trabalho intitulado *Gamificação e Inovação na Aprendizagem de Programadores: uma proposta de plataforma gamificada para o ensino de programação* (COSTA; COSTA; BOTTENTUIT JUNIOR, 2024) apresenta paralelos diretos com este projeto. Os autores não apenas propuseram mas também implementaram uma plataforma baseada em trilhas de aprendizagem com missões sequenciais, sistema de recompensas (*badges* e pontos) e elementos narrativos, com o objetivo de aumentar o engajamento e a motivação no aprendizado de programação. O estudo ressaltou, como desafio comum, a complexidade de se projetar mecânicas de jogo que estejam em excelente sintonia com os objetivos pedagógicos. Este trabalho serve como um correlato importante por demonstrar na prática a aplicação de conceitos de gamificação em um ambiente de aprendizado estruturado em fases, reforçando a

viabilidade e os benefícios da abordagem aqui adotada para o contexto de preparação para o ENEM.

Com os fundamentos teóricos consolidados, o próximo passo se dá na direção da materialização destes conceitos, através de arquiteturas, tecnologias e mecanismos.

2.7 Tecnologias Utilizadas

Para a extração de dados do PDF, foi utilizada a linguagem *Python* e a biblioteca PyPDF2 junto ao módulo *re* para expressões regulares. Além disso o banco de dados para persistência dos dados é o PostgreSQL, sendo a conexão com o mesmo e comandos de criação de tabelas e persistência responsabilidade do *Django* e *Django* ORM. O *backend* foi implementado com *Django* e *Django Rest Framewor*k, já o *frontend* com *React*.

3 Desenvolvimento

Neste capítulo, é explicado de forma mais detalhada a abordagem proposta, desde a extração de questões, alternativas e gabaritos, banco de dados, *backend*, APIs e *frontend*.

3.1 Extração do PDF do ENEM e Banco de Dados

Antes de mais nada, a primeira ação para início deste projeto foi a de baixar um PDF de um caderno de prova do ENEM 2023 como exemplo, além do gabarito atrelado a ela. O ano de 2023 foi escolhido devido ao começo da implementação do sistema de extração se dar neste ano. Foram baixados do site oficial do INEP (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira) os arquivos:

- https://download.inep.gov.br/enem/provas_e_gabaritos/2023_PV_impresso_D1_CD1.pdf (Prova)
- https://download.inep.gov.br/enem/provas_e_gabaritos/2023_GB_impresso_D1_CD1.pdf (Gabarito)

Como o *framework Django* oferece um sistema robusto para a criação de comandos personalizados, estendendo a funcionalidade padrão do *manage.py*, o código *python* para extração foi colocado em *tcc/backend/questions/management/commands/import_enem.py* (dentro da aplicação *questions*), pois para que funcionem devem seguir o padrão *[nome_app]/management/commands/*. O nome do arquivo *Python* (sem a extensão .py) se torna o nome do comando a ser invocado, no caso deste criado, sua invocação é feita assim: *python manage.py import_enem.*

É essencial, antes de detalhar o funcionamento, compreender a estrutura de dados que este comando irá preencher, possuindo três entidades, Test, Question e Alternative:

1. *Test* (Prova): Representa o nome da prova ou simulado a ser extraído, no caso "Enem 2023"
2. *Question* (Questão): Representa uma pergunta individual da prova, seu enunciado, matéria (assunto) e também imagem se possuir.

3. *Alternative* (Alternativa): Representa uma das opções de resposta de uma questão (A, B, C, D ou E). Cada alternativa é vinculada à sua questão pai e possui um campo *booleano is_correct* que indica se é a resposta correta.

Como apontado na subseção 2.3, este trabalho utiliza o ORM do *Django* como camada de abstração para interagir com o banco de dados PostgreSQL. Nas Figuras 5 e 6, respectivamente, vemos os códigos do *tcc/backend/questions/models.py* e trecho do *tcc/backend/tcc/settings.py*:

Figura 5 - Modelos *Test*, *Question* e *Alternative*

```python
from django.db import models


class Test(models.Model):
  name = models.TextField(unique=True)

class Question(models.Model):
  test = models.ForeignKey(Test, on_delete=models.CASCADE)
  question_number = models.IntegerField()
  subject = models.TextField()
  question_text = models.TextField()
  image = models.ImageField(upload_to="questions/", null=True, blank=True)

class Alternative(models.Model):
  question = models.ForeignKey(Question, on_delete=models.CASCADE)
  alternative_text = models.TextField()
  alternative_number = models.CharField(max_length=1)
  is_correct = models.BooleanField(default=False)
```

Fonte: Elaboração Própria.

Figura 6 - Configuração do Banco de Dados

```python
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.postgresql',
    'NAME': os.getenv('POSTGRES_DB'),
    'USER': os.getenv('POSTGRES_USER'),
    'PASSWORD': os.getenv('POSTGRES_PASSWORD'),
    'HOST': os.getenv('POSTGRES_HOST'),
    'PORT': os.getenv('POSTGRES_PORT', 5432),
  }
}
```

Fonte: Elaboração Própria.

O primeiro código (Figura 5) mostra como o ORM define as tabelas no banco através de classes *python* chamadas Modelos. O segundo (Figura 6) revela a configuração de conexão com o banco (PostgreSQL), utilizando variáveis de ambiente carregadas de um arquivo *.env*. Este arquivo serve para armazenar configurações sensíveis e específicas de cada ambiente (como desenvolvimento, teste ou produção) – tais como credenciais de banco de dados, chaves de API e outros parâmetros – as mantendo separadas do código-fonte.

O uso do *regex*, já elucidado na subseção 2.1, é parte fundamental do funcionamento do código, que segue o fluxo lógico abaixo:

1. Lê o PDF com as questões (*QUESTIONS_PDF*) e extrai todo o texto (extract_text).

2. Analisa esse texto para obter *question_data* – lista de dicionários com número, assunto, enunciado e alternativas de cada questão.

3. Lê o PDF de gabarito (*ANSWERS_PDF*) e extrai pares (número da questão, alternativa correta) através da função *extract_answers*.

4. Cria um Test no banco "Enem 2023", um *Question* para cada questão extraída, cinco alternativas (*Alternative*) por questão, sendo a correta possuindo o campo *is_correct* = *True*.

A classe ou modelo da questão (Question) pode possuir uma imagem. Bibliotecas do Python como PyMuPDF (Fitz) e pdfplumber fizeram parte da tentativa de extração de imagens, porém os resultados foram em maioria não satisfatórios, extraindo muitas vezes imagens incompletas ou incompatíveis com as apresentadas em PDF, como nos exemplos na Figura 7 e na Figura 8.

Figura 7 - Imagem extraída utilizando *Fitz/Python*

Fonte: Elaboração Própria.

Figura 8 - Imagem extraída com a ferramenta de captura do *Windows*

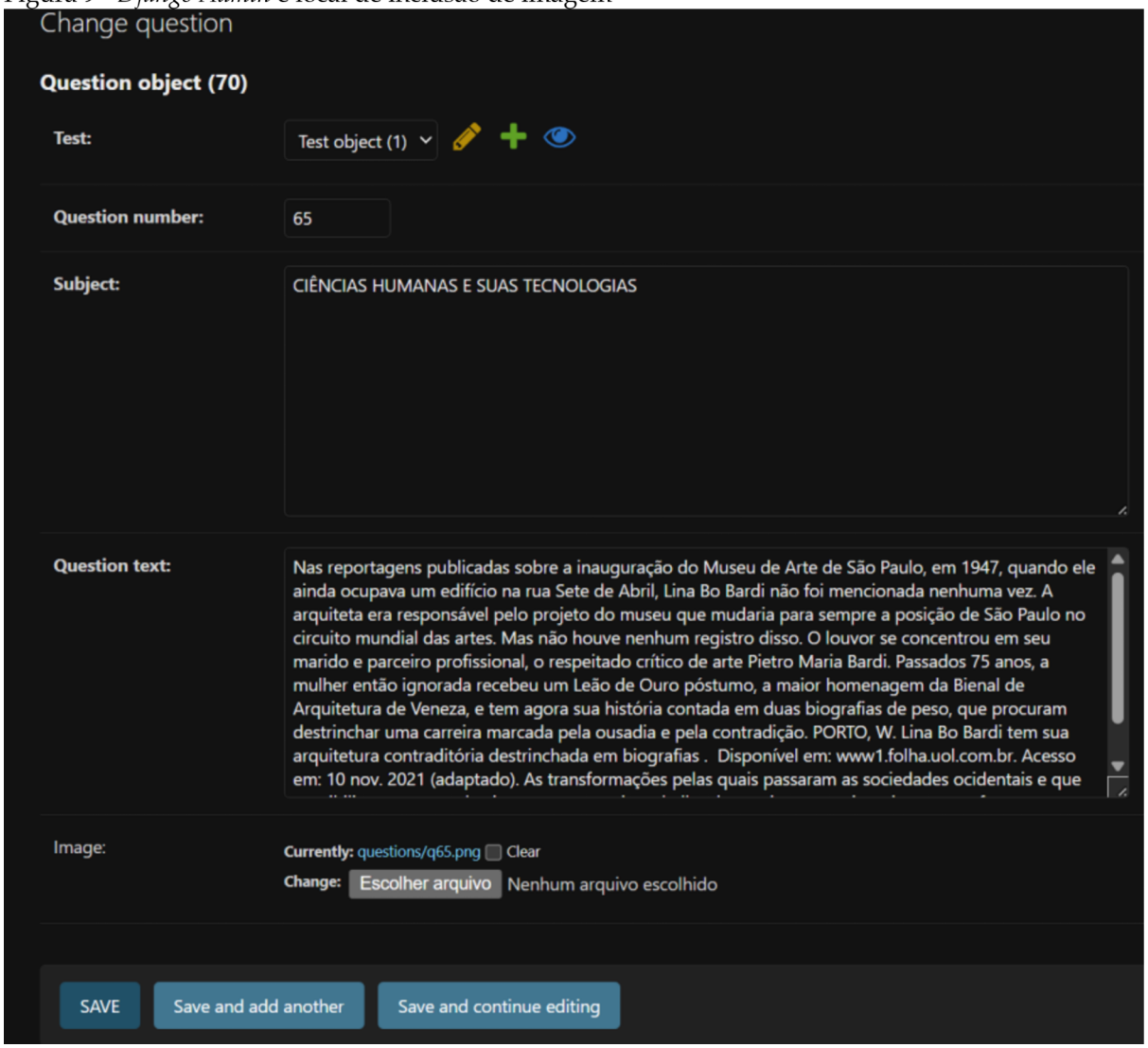
Fonte: https://download.inep.gov.br/enem/provas_e_gabaritos/2023_PV_impresso_D1_CD1.pdf

Portanto, para efeito de extração de imagens da prova do Enem 2023 neste trabalho, foi decidido executar manualmente utilizando a captura de tela do *Windows*, colocando-as na pasta *tcc/backend/media/questions*, e as persistindo no banco de dados através do *Django Admin*.

O *Django Admin* é uma interface administrativa que o próprio *Django* gera automaticamente a partir dos seus modelos, é frequentemente usada para inserir ou ajustar dados manualmente, o que foi feito com as imagens. Neste projeto, é acessado pela URL (Localizador Uniforme de Recursos) *http://localhost:8000/admin/* e a imagem no banco de dados nada mais é que uma *string* com o caminho relativo da imagem,

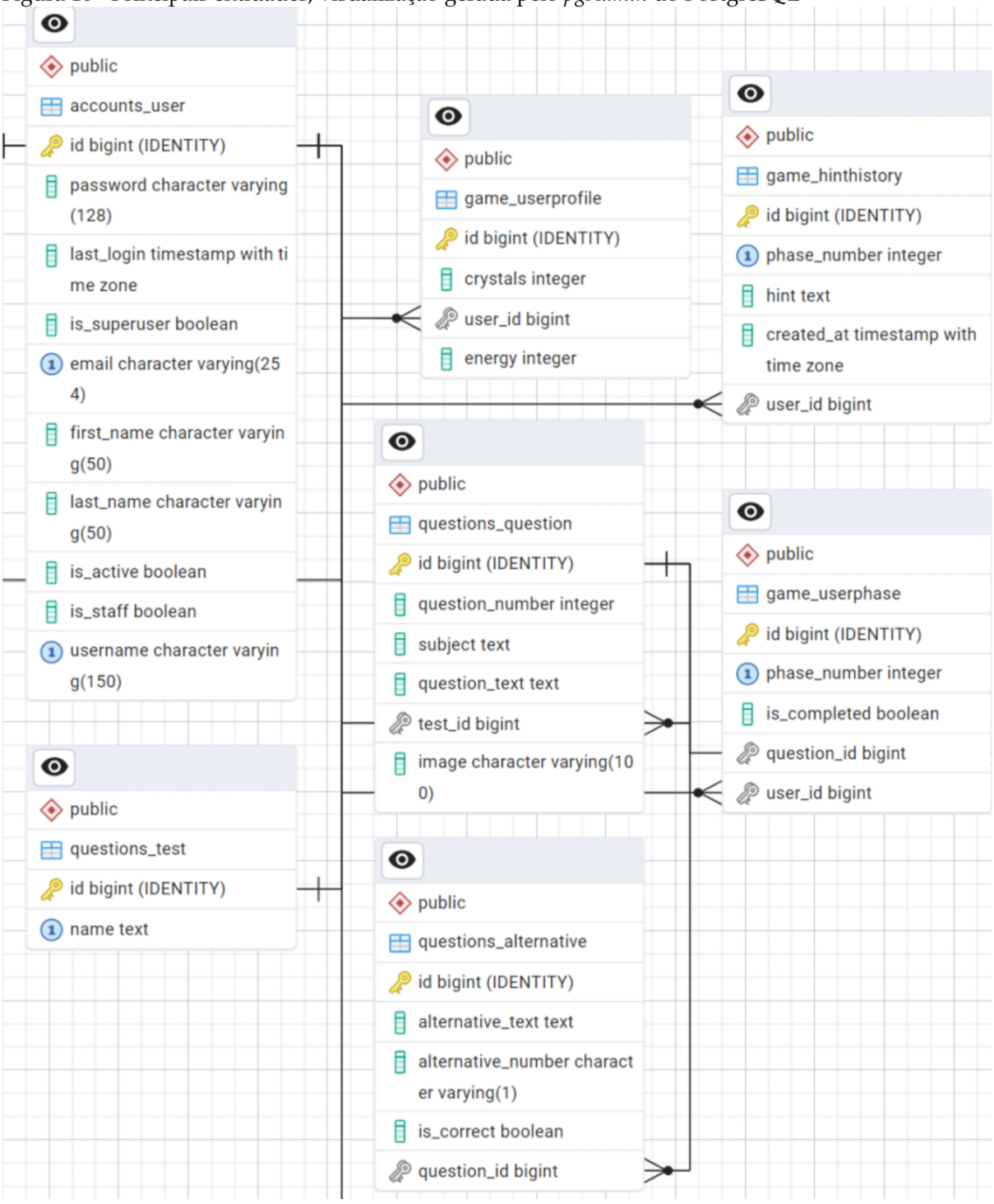
como exemplo "*question/q11.png*". Se observa na Figura 9 uma das telas do *Django Admin*, especificamente a de edição de questões.

Figura 9 - *Django Admin* e local de inclusão de imagem



Fonte: Elaboração Própria.

Figura 10 - Principais entidades, visualização gerada pelo *pgAdmin* do PostgreSQL



Fonte: Elaboração Própria.

Na figura 10 podemos visualizar entidades importantes, envolvendo *questions* (questões, testes e alternativas) e game (controle de cristais e energia, quais fases o usuário já completou, dicas).

3.2 Backend e suas APIs

Nesta subseção serão mostradas todas as APIs e suas razões de existir, envolvendo autenticação (*/api/auth/*), controle de questões (*/api/questions/*) e lógicas das fases, rankings e conquistas (*/api/game/*), além de detalhar os dois tipos de usuários.

- POST */api/auth/login/*: Realiza o *login* do usuário, entra com e-mail e senha e retorna *tokens* JWT, *access* e *refresh*.
- POST */api/auth/refresh/*: Permite renovar o *token* de acesso sem precisar fazer *login* novamente.
- POST */api/auth/registration/*: *Endpoint* de registro de novos usuários, com email e senha.
- GET */api/questions/*: Lista todas as questões.
- GET */api/questions/{id}*:Retorna os detalhes de uma questão específica pelo seu *id*.
- POST */api/game/initialize*: Sorteia 10 fases iniciais para o usuário com base nas questões disponíveis.
- GET */api/game/phases*: Lista todas as fases associadas ao usuário autenticado.
- GET */api/game/phases/{phase_number}/*: Retorna os detalhes de uma fase específica associada ao usuário autenticado, o acesso é bloqueado caso a fase anterior não tenha sido concluída.
- POST */api/game/enter/{phase_number}/*: Permite que o usuário entre em uma fase, só é possível entrar em uma fase se a anterior já estiver sido concluída, além disso a entrada na fase exige um ponto de energia do usuário.

• POST */api/game/answer/{phase_number}*: Recebe a alternativa escolhida pelo usuário para a questão da fase, em caso de acerto, ele recebe 10 cristais.

• POST */api/game/hint/{phase_number}*: Gera uma dica personalizada para a questão da fase usando a API do ChatGPT, custa 30 cristais do usuário. Caso já exista uma dica salva para a fase, a mesma é retornada (sem gastar cristais novamente).

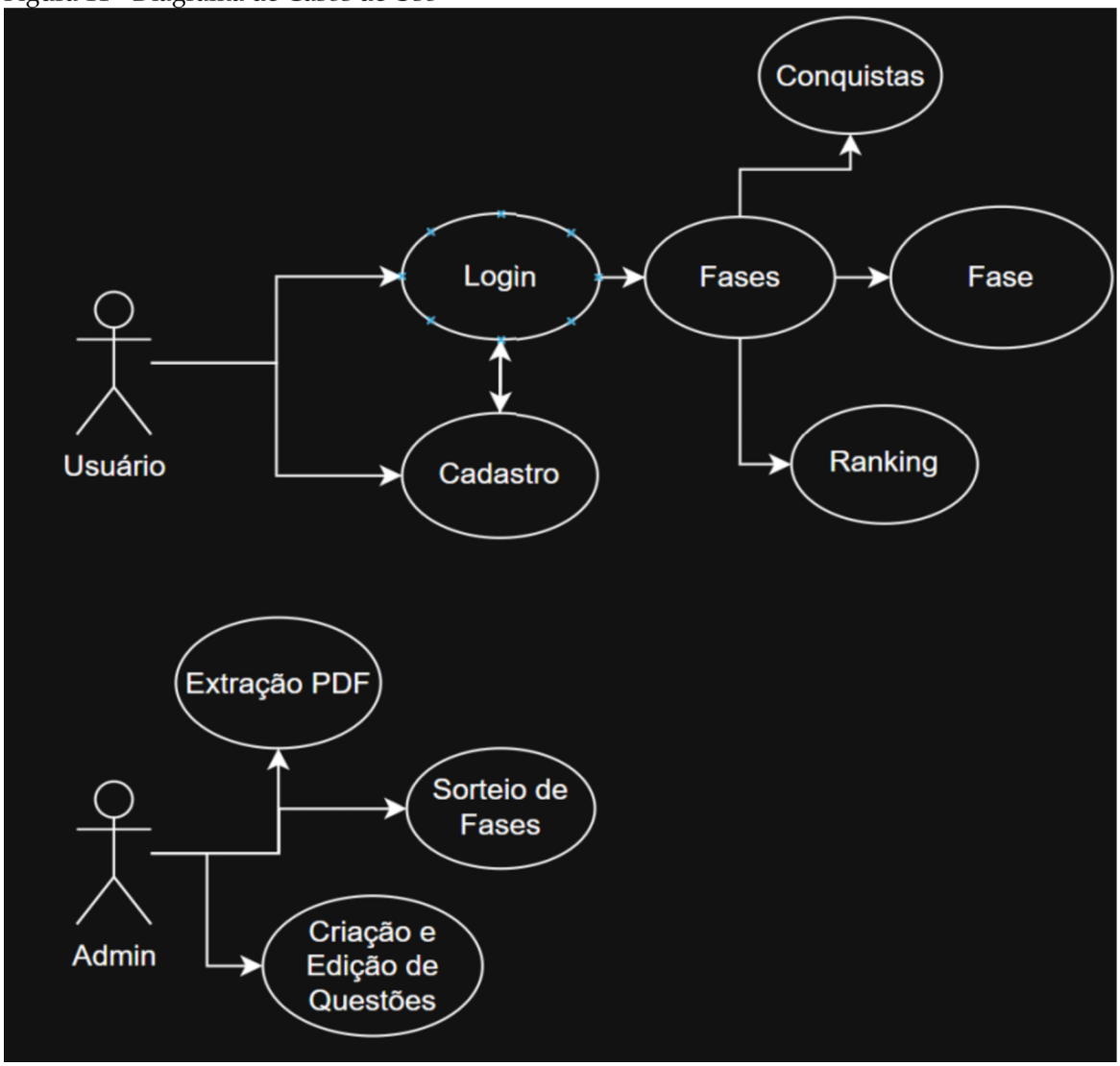
- GET */api/game/profile*: Retorna as informações do perfil do usuário autenticado, quantidade de energia e de cristais, soma um à energia a cada um dia.

• GET */api/game/ranking*: Retorna as informações sobre o ranking global e sobre o ranking do usuário autenticado.

• GET */api/game/achievements*: Retorna as conquistas adquiridas pelo usuário autenticado. O sistema conta com dois tipos de usuários, cada um com seu papel e funções distintas:

O sistema conta com dois tipos de usuários, cada um com seu papel e funções distintas apresentados pelo Diagrama de Caso de Usos mostrado na Figura 11.

Figura 11 - Diagrama de Casos de Uso



Fonte: Elaboração Própria.

O usuário comum, que acessa o sistema pela interface principal, utilizando a tela de cadastro para criar sua conta e a tela de login para se autenticar. Ele pode:

- Acessar as fases do jogo, respondendo às questões sorteadas para ele.
- Gastar energia ao entrar na fase e cristais para solicitar dicas.
- Acompanhar seu desempenho no ranking em relação a outros jogadores.

• Adquirir conquistas conforme seu progresso, como a conquista Penta (ao acertar 5 questões) ou Competitivo (ao alcançar top 10 no ranking).

Já o usuário administrador é aquele cadastrado utilizando o comando do *framework Django, python manage.py createsuperuser*, e acessa o sistema pelo painel administrativo. Ele pode:

• Criar e editar questões do banco de dados.

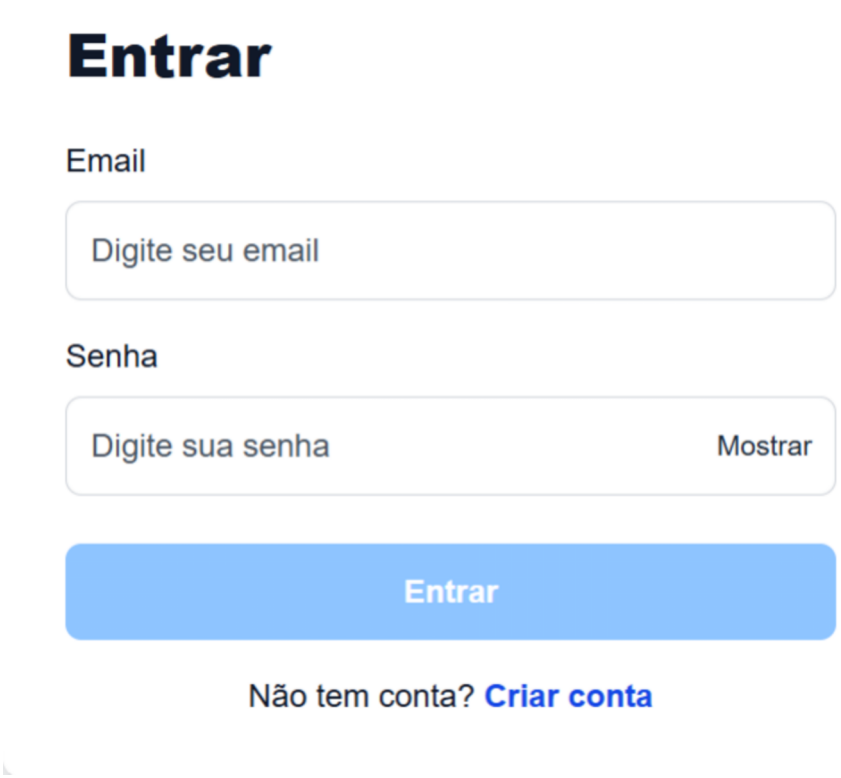
• Gerenciar usuários e perfis por meio da interface administrativa.

• Executar comandos personalizados do sistema, como extração de questões de PDFs e resorteio de fases.

3.3 Frontend

Nesta subseção as telas do sistema são apresentadas. A página inicial é a de *Login* (Figura 12), aonde um usuário já cadastrado poderá efetuar o login e entrar no sistema, ela conta com opção de mostrar a senha e botão de Criar conta. Uma mensagem de erro “Credenciais Inválidas” é mostrada no canto superior direito em caso de falha na autenticação. Também avisa da falta de @ se tentar entrar com um e-mail sem este caractere.

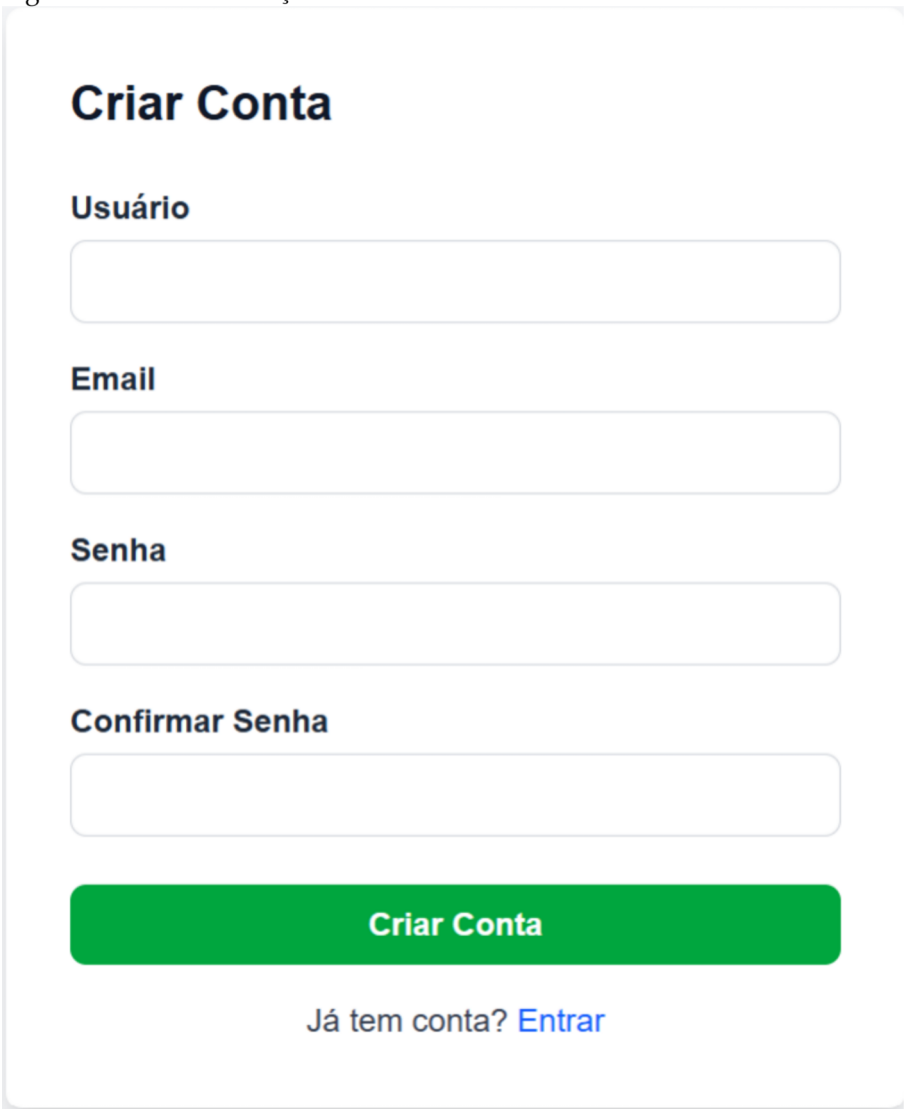
Figura 12 - Tela de Login (http://localhost:3000/login)



Fonte: Elaboração Própria.

Na Figura 13 podemos ver a página de criação da conta, aonde o usuário deve entrar com nome, e-mail e a senha duas vezes. Se as senhas não forem iguais, gerará um erro ao criar a conta. Se clicado, o texto “Entrar” volta para a página de *Login*.

Figura 13 - Tela de Criação de Conta



Fonte: Elaboração Própria.

Se o *Login* for efetuado com sucesso, o usuário se depara com a tela das fases (Figura 14 e Figura 15), aonde pode clicar na próxima fase disponível, checar a quantidade de cristais e de energia que possui, ou sair (*deslogar* da aplicação). As fases em verde mostram fases já concluídas, a fase em azul é a próxima fase a ser concluída e as em cinza são fases bloqueadas.

Figura 14 - Tela das Fases (parte 1)

Fonte: Elaboração Própria.

Figura 15 - Tela das Fases (parte 2)

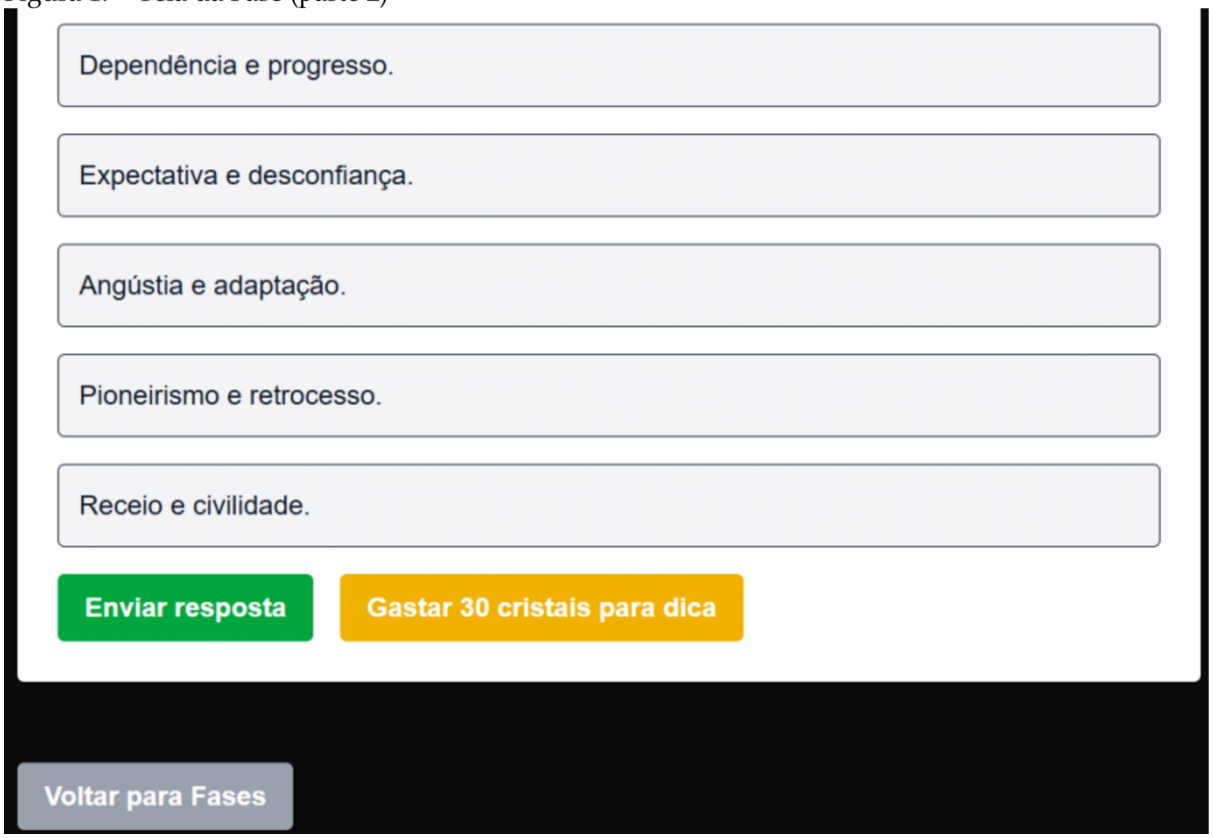
Fonte: Elaboração Própria.

Ao clicar numa fase azul (fase elegível para acesso), o usuário vai para a página de fase específica (Figura 16) gastando uma energia, mostrando o enunciado da questão, imagem se houver, quantidade de cristais, um botão para enviar resposta, outro para gastar 30 cristais para dica (conecta com a API do ChatGPT e retorna alguma dica sobre a questão) e, por fim, um botão para voltar para a página geral de fases.

Figura 16 - Tela da Fase (parte 1)

Fonte: Elaboração Própria.

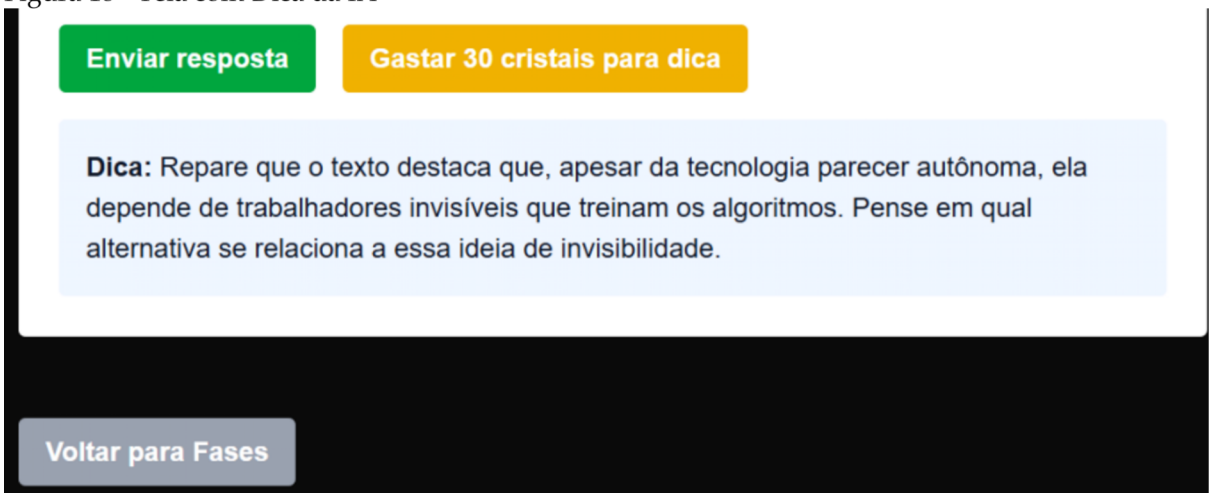
Figura 17 - Tela da Fase (parte 2)



Fonte: Elaboração Própria.

Ao clicar no botão Gastar 30 cristais para dica, é chamada a API do ChatGPT e a dica é exibida sobre a questão, logo abaixo, como explicitado na Figura 18.

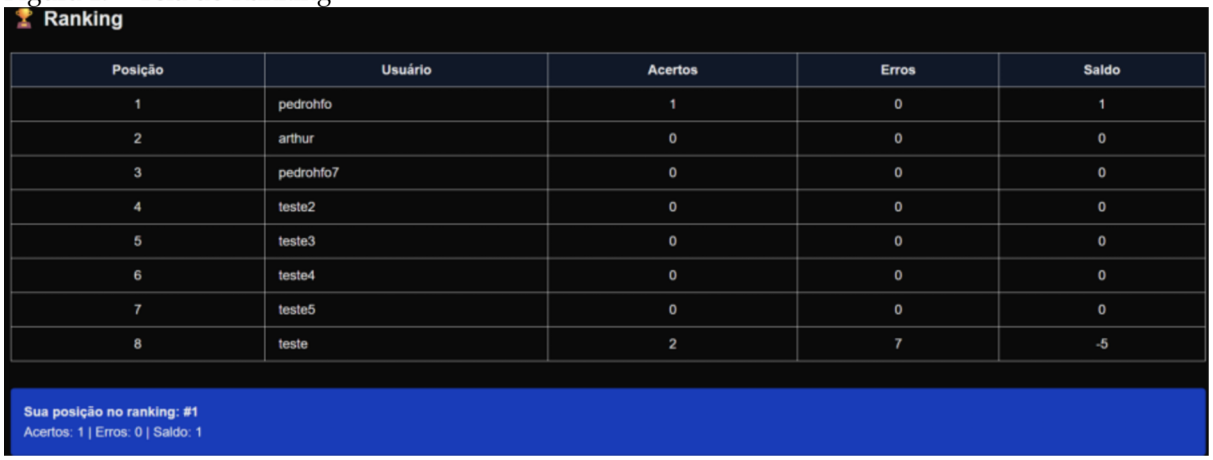
Figura 18 - Tela com Dica da IA



Fonte: Elaboração Própria.

Ao clicar no botão *Ranking*, na tela geral de Fases, o usuário se redirecionará para a tela do Ranking (Figura 19), aonde são mostrados os dez primeiros melhores ranqueados, além da posição global no do usuário autenticado no *ranking*.

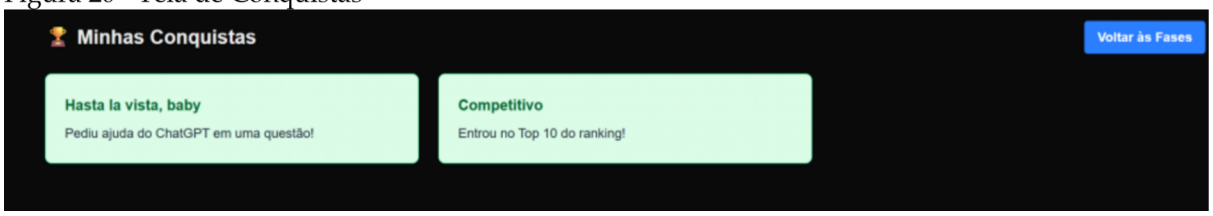
Figura 19 - Tela do Ranking



Fonte: Elaboração Própria.

A partir da tela geral de fases, se clicar em Conquistas, o usuário se depara com todas as conquistas adquiridas assim como a explicação de como as conseguiu, demonstrado na Figura 20.

Figura 20 - Tela de Conquistas



Fonte: Elaboração Própria.

3.4 Elementos Gamificados

O primeiro elemento gamificado é simples, a progressão em fases. Isto significa que para *n-ésima* fase, todas as fases com número menor que *n* tem que estarem completas.

Outro elemento é o sistema de *ranking*, ilustrado na Figura 19, que ordena os colocados pelo saldo, calculado pela quantidade de acertos menos a quantidade de erros em fases.

O sistema de cristais serve como uma espécie de moeda para gastar com dicas da inteligência artificial, gastando 30 cristais para obtenção de uma dica, e ganhando 10 cristais para cada fase concluída com sucesso.

O sistema de energia é utilizado para a entrada na fase, gastando uma energia para isto. O máximo de energia de um usuário é estabelecido em sete, e o usuário recupera uma energia a cada 24 horas.

As conquistas implementadas são três: *"Hasta la vista, baby"*, adquirida quando o usuário solicita uma dica pela primeira vez, *"Penta"*, obtida quando se conclui cinco fases e *"Competitivo"*, se o usuário checar o *Ranking* e estiver entre os dez melhores colocados.

O sistema foi implementado pensando na ideia de cada usuário ter a possibilidade de completar as dez fases (o que pode ser facilmente incrementado para mais) por semana, e mais dez serem sorteadas novamente em outra semana. Como a extração de dados também captura a matéria ou assunto de cada questão, uma evolução interessante seria abrir a possibilidade de escolha do usuário das matérias que mais quer focar, além de *feedbacks* sobre quais matérias ele possui mais acertos e quais possui mais erros.

3.5 Integração com Inteligência Artificial

A API utilizada para geração de dicas para o usuário se dá pela plataforma *OpenAI* (https://platform.openai.com/docs/overview), e a chave da API se situa no arquivo *.env*. Como pode ser visto na Figura 21, o modelo utilizado no código é o "*gpt-4o-mini*", o máximo de *tokens* (*max_tokens*) igual a 150, e a temperatura (*temperature*) igual a 0,7. Um *token* é uma unidade de texto que pode ser uma palavra ou parte dela, ou seja, uma palavra pode refletir em dois ou mais *tokens*. A temperatura define o grau de aleatoriedade e criatividade da resposta, sendo próximo a zero mais determinística e direta, próximo ou maior que 1 acarretando em uma resposta mais criativa e menos consistente. O valor 0,7 é considerado equilibrado, é um padrão recomendado pela plataforma OpenAI.

Figura 21 - API da integração com IA

```python
class HintView(APIView):
    def post(self, request, phase_number):

            # 4. monta prompt para ChatGPT
            prompt = f"""
Você é um tutor de ENEM. Ajude o aluno a resolver a questão abaixo, sem dar a resposta final direta.
Questão: {question.question_text}

Alternativas:
{alternatives_text}

Dê uma dica curta, em português, que oriente o raciocínio sem revelar a alternativa correta explicitamente.
"""

            # 5. chama OpenAI API
            response = client.chat.completions.create(
                model="gpt-4o-mini",
                messages=[{"role": "system", "content": "Você é um tutor útil."},
                          {"role": "user", "content": prompt}],
                max_tokens=150,
                temperature=0.7,
            )

            hint_text = response.choices[0].message.content.strip()

            # 6. desconta cristais e salva histórico
            profile.crystals -= 30
            profile.save()

            HintHistory.objects.create(
                user=user,
                phase_number=phase_number,
                hint=hint_text
            )

            return Response({"hint": hint_text}, status=status.HTTP_200_OK)

        except UserPhase.DoesNotExist:
            return Response(
                {"detail": "Fase não encontrada para o usuário."},
                status=status.HTTP_404_NOT_FOUND
            )
```

Fonte: Elaboração Própria.

3.6 Disponibilização do Código no GitHub

A gestão do código-fonte e o controle de versão do projeto foram realizados integralmente através da plataforma GitHub. O repositório completo do projeto, incluindo todo o código do backend em Django, frontend em React, scripts de extração, configurações e instruções de implantação, está publicamente disponível no seguinte endereço: https://github.com/pedrohfo/tcc.

A estrutura do repositório é organizada de modo a separar claramente os componentes da aplicação, com as pastas principais *backend/* e *frontend/* na raiz do projeto. Um dos elementos mais críticos para a segurança e integridade de qualquer projeto versionado é o arquivo *.gitignore*, presente no diretório raiz. Este arquivo tem a função fundamental de instruir o *Git* a ignorar e não rastrear arquivos e diretórios específicos, evitando que sejam *commitados* acidentalmente para o repositório.

Além do *.gitignore*, o repositório é ancorado por um arquivo *README.md*. Este documento serve como o ponto de entrada e manual do projeto, contendo instruções claras para garantir a reprodutibilidade do projeto.

Diante da estrutura e funcionamento da plataforma assim apresentados, torna-se viável, então, uma análise conclusiva sobre os resultados obtidos, as limitações enfrentadas e os futuros desdobramentos possíveis para este trabalho.

4 Conclusão

Este trabalho demonstrou a viabilidade de extração automatizada de questões, alternativas e gabaritos de provas do ENEM utilizando técnicas de processamento de texto com expressões regulares (*regex*). Os resultados obtidos foram satisfatórios: todas as 95 questões e 475 alternativas foram extraídas corretamente, bem como o gabarito oficial, comprovando a eficácia do método para o formato específico do caderno de provas do ENEM 2023.

No entanto, é crucial ressaltar que a solução baseada em *regex* é altamente dependente da estrutura do PDF. Enquanto o formato do arquivo se mantiver consistente, a extração ocorrerá sem necessidade de ajustes. Por outro lado, se houver alterações significativas na organização do conteúdo (como mudanças no layout, ordem dos elementos ou formatação textual), será necessária manutenção manual e readequação dos padrões *regex*, o que pode ser custoso e tecnicamente complexo. Além disso, a técnica se mostrou inviável para provas de outras fontes ou anos com estruturas não similares, dada a alta variabilidade de formatos entre documentos PDF.

Quanto à extração de imagens, a abordagem inicial com *Python* e *Fitz* (PyMuPDF) apresentou limitações significativas: as imagens foram recuperadas de forma fragmentada, desordenada e incompleta, inviabilizando seu uso prático. Como alternativa, o uso de ferramentas de OCR (Reconhecimento Óptico de Caracteres) acopladas a modelos de IA (como *Tesseract* integrado a *frameworks* de detecção de layout) ou soluções de extração inteligente de objetos baseadas em aprendizado de máquina (como exemplo *Adobe Extract API*, *Google Document AI*) capazes de interpretar contextos visuais e extrair elementos de forma assertiva.

O trabalho avançou além da extração de dados, propondo uma plataforma gamificada desenvolvida com *Django REST Framework* (*backend*) e *React* (*frontend*), onde os usuários podem resolver as questões em um sistema organizado em dez fases, com mecanismos de energia para acesso e cristais para solicitação de dicas via integração com a API do *ChatGPT*. Essa conexão com IA não apenas enriquece a experiência do usuário mas também abre precedentes para personalização do aprendizado com base em dificuldades individuais.

Embora a plataforma apresentada seja funcionalmente simples, ela serve como base sólida para evoluções futuras, como:

• Sistemas de recomendação de questões baseadas em desempenho;

• Elementos gamificados adicionais (missões, competições em tempo real, recompensas);

• Adaptação de conteúdo conforme o perfil do usuário.

Por fim, este projeto evidencia que é possível construir soluções robustas para educação gamificada com integração de IA, pavimentando o caminho para pesquisas futuras em personalização massiva e aprendizado adaptativo. A extração automatizada de questões e sua evolução é de suma importância, criando bancos de dados estruturados e completos para alimentar os sistemas educacionais interativos.

Referências

COSTA, Marcus Vinicius Silva; COSTA, Maurício José Morais; BOTTENTUIT JUNIOR, João Batista. **Gamificação e Inovação na Aprendizagem de Programadores: uma proposta de plataforma gamificada para o ensino de programação**. Revista Interdisciplinar em Cultura e Sociedade, p. 74–94, 2024. DOI: 10.18764/2447-6498.v10n2.2024.18. Disponível em: https://periodicoseletronicos.ufma.br/index.php/ricultsociedade/article/view/25390. Acesso em: 24 set 2025.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. Universidade da Califórnia, Irvine. Tese de doutorado, 2000. ISBN: 978-0-599-87118-2. Disponível em: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm. Acesso em: 24 set 2025.

FOWLER, Martin. **Patterns of Enterprise Application Architecture**. Boston: Addison-Wesley, 2002. ISBN: 978-0321127426

FRIEDL, Jeffrey E. F. **Mastering Regular Expressions**. O'Reilly & Associates, 3ª edição, 2006. ISBN: 978-0596528126

KAPP, Karl M. **The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education**. San Francisco: Pfeiffer, 2012. ISBN: 978-1118096345

KHALDI, Amina; BOUZIDI, Rokia; NADER, Fahima. **Gamification of e-learning in higher education: a systematic literature review**. Smart Learning Environments, v. 10, nº 1, 2023. DOI: 10.1186/s40561-023-00227-z.

MCGONIGAL, Jane. **Reality is Broken: Why Games Make Us Better and How They Can Change the World**. The Penguin Press, New York, 2011. ISBN: 978-0143120612

PRENSKY, Marc. **Digital Natives, Digital Immigrants, Part 1**. *On the Horizon*, Vol. 9, nº 5, 2001, p. 1-6. DOI: 10.1108/10748120110424816.

SALEN, Katie; ZIMMERMAN, Eric. **Rules of Play: Game Design Fundamentals**. MIT Press, Cambridge (Massachusetts), 2004. ISBN: 978-0-262-24045-1.

WIECHORK, Karina; CHARÃO, Andrea Schwertner. **Automated Data Extraction from PDF Documents: Application to Large Sets of Educational Tests**. In: Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) – Volume 1, p. 359-366. SciTePress, 2021. DOI: 10.5220/0010524503590366.