Aprendizado de novos idiomas por meio da leitura dinâmica por meio do uso do programa Orbitext

Felipe Augusto Neves Acioli[1]Paulo Henrique da Fonseca[1], Cleyton V.C. de Magalhães[1]

[1]Bacharelado em Sistemas de Informação – Universidade Federal Rural de Pernambuco (UFRPE)
Caixa Postal 52171-900– Recife – PE – Brasil

{paulo.henriqueb,felipe.acioli,cleyton.vanut}@ufrpe.br

Abstract. *This paper presents Orbitext, a Python software developed to assist in learning foreign languages (English, French, and Spanish) through reading. In a scenario where mastering foreign languages is essential, the tool offers an interactive console interface providing texts with translations, encouraging constant practice. The system's key differentiator lies in the implementation of gamification elements, such as an experience (XP) system based on reading time, evolution levels, and user rankings. Furthermore, the software promotes social interaction through comments and paragraph likes. This work describes the solution's architecture and how its features aim to increase student engagement and linguistic proficiency.*

Resumo. *Este artigo apresenta o Orbitext, um software desenvolvido em Python para auxiliar no aprendizado de idiomas (Inglês, Francês e Espanhol) por meio da leitura. Em um cenário onde o domínio de línguas estrangeiras é fundamental, a ferramenta oferece uma interface interativa em console que disponibiliza textos com tradução, incentivando a prática constante. O diferencial do sistema reside na implementação de elementos de gamificação, como sistema de experiência (XP) baseado no tempo de leitura, níveis de evolução e rankings de usuários. Além disso, o software promove a interação social através de comentários e curtidas em parágrafos. O trabalho descreve a arquitetura da solução e como suas funcionalidades visam aumentar o engajamento e a proficiência linguística dos estudantes.*

1. Introdução

No mundo moderno e globalizado em que a humanidade se encontra, o aprendizado de um novo idioma mostra-se fundamental tanto para o desenvolvimento profissional quanto para o crescimento pessoal e social dos indivíduos. Nesse contexto, Putra (2020) ressalta que aprender inglês é essencial, pois se trata da língua internacional comum, utilizada para a comunicação entre pessoas de diferentes países, inclusive aquelas cuja língua materna não é o inglês.

De acordo com Putra (2020), estima-se que existam mais de 1,5 bilhões de falantes da língua inglesa no mundo. Isso demonstra que o domínio do inglês abre amplas

possibilidades de comunicação global e de ampliação das redes de contato. Além disso, o conhecimento desse idioma traz benefícios concretos à vida profissional: Farah (2013) aponta que cerca de 80% das entrevistas de emprego são conduzidas em inglês e que o domínio da língua pode aumentar a remuneração em até 11%.

Esses dados evidenciam a importância do inglês como ferramenta essencial para o crescimento profissional e para a conquista de melhores oportunidades, permitindo uma vida mais próspera por meio da comunicação com milhões de pessoas ao redor do mundo. Portanto, é inegável que o inglês é o idioma estrangeiro mais relevante e amplamente conhecido, sendo indispensável que todos busquem aprendê-lo.
Além disso, é evidente que o aprendizado de novos idiomas, além do inglês, é fundamental, especialmente no contexto globalizado em que o mundo se encontra atualmente. Nesse sentido, Valle e Villa (2006) destacam a influência do governo espanhol na promoção e desenvolvimento da língua espanhola no Brasil, buscando consolidá-la como um dos principais idiomas globais. Assim, o domínio do espanhol contribui significativamente para a ampliação da interação internacional, sobretudo para os brasileiros, já que facilita a comunicação com os povos de sete países vizinhos do Brasil que têm o espanhol como língua oficial.

Somado a isso, é claro que o domínio de um terceiro idioma é essencial para a adaptação ao mundo globalizado contemporâneo. A língua francesa, por exemplo, tem se mostrado de grande importância desde o início da globalização, marcada pelas Grandes Navegações, alcançando seu auge durante o absolutismo europeu, entre os séculos XVI e XIX, quando foi adotada por diversas cortes como língua franca. Mesmo após mais de dois séculos do fim desse período e com o inglês assumindo o papel de idioma global, o francês continua sendo uma das línguas mais relevantes do planeta.

Essa relevância é perceptível nas relações internacionais, já que instituições como a ONU e a UNESCO reconhecem o francês como uma de suas línguas oficiais. Além disso, diversos concursos e exames diplomáticos, como o CACD, consideram o francês uma de suas línguas-base. No campo cultural, o idioma mantém grande prestígio: conteúdos artísticos de origem francesa continuam a dominar as artes e a moda mundial, com expressões notáveis como as obras expostas no Museu do Louvre, os prestigiados desfiles de moda parisienses e diversos filmes premiados e reconhecidos internacionalmente, como A Noite Americana (La Nuit Américaine, 1973) e O Fabuloso Destino de Amélie Poulain (Le fabuleux destin d'Amélie Poulain, 2001).

A partir dos fatores já expostos, o projeto Orbitext tem como principal objetivo preparar seus usuários para o mundo globalizado por meio do aprendizado de novos idiomas, utilizando a leitura como ferramenta central de desenvolvimento linguístico. Conforme destacam Koda e Yamashita (2018), a prática da leitura contribui de forma significativa para a aquisição de habilidades cognitivas e comunicativas, tanto básicas quanto complexas. Entre elas, incluem-se a capacidade de compreender a ideia principal de um texto, distinguir informações essenciais de detalhes secundários e interpretar adequadamente informações provenientes de textos extensos ou de múltiplas fontes. Dessa forma, a proposta do Orbitext visa não apenas à aprendizagem de novos idiomas, mas também ao aprimoramento da competência da leitura e interpretação, indispensável

à inserção efetiva em um contexto globalizado.

Esse projeto será feito por meio da possibilidade de escolha entre os idiomas já citados e sua respectiva leitura em português, podendo escolher essa tradução de forma posterior ou simultânea. Somado a isso, o Orbitext tem como seu diferencial um estudo coletivo entre os alunos, no qual possibilita conferir os textos mais marcados, e um sistema de níveis entre os seus usuários, assim permitindo um desenvolvimento dinâmico e o preparo de novos alunos para o complexo mundo que a globalização criou, formando novos caminhos que podem se apresentar como um passo inicial tanto para o desenvolvimento profissional quanto o pessoal.

2. Trabalho Relacionado

O Mutare (ALVES; CORDEIRO, 2025) é uma ferramenta utilizada para realizar o gerenciamento de hábitos, levando mais produtividade e qualidade de vida para os usuários. Baseado na psicologia do hábito, tendo como referência o livro "O Poder do Hábito" do autor Charles Duhigg.

3. Metodologia

O método seguido do trabalho foi composto pelas seguintes fases.

3.1. Identificação de Requisitos

Baseando-se em todo o contexto e ambiente acadêmico, o processo de definição dos requisitos surgiu pensando no usuário sendo o estudante, e em suas necessidades de aprendizado de uma língua estrangeira. Além dos requisitos funcionais (leitura de texto, tradução e escolha de idioma), foram priorizados os seguintes requisitos não funcionais, evidenciados nos arquivos do projeto:

Persistência de Dados: Garantir o armazenamento seguro e estruturado dos dados dos usuários, textos e interações, optando pelo formato JSON para facilidade de manipulação e flexibilidade.

Segurança: Implementar regras estritas para a criação de senhas e garantir a unicidade de *e-mails* no cadastro, protegendo o acesso à conta (conforme verificação.py e usuario.py).

Interatividade: Desenvolver módulos que permitam a interação assíncrona dos usuários por meio de comentários e curtir comentários e parágrafos.

Usabilidade (*CLI*): Garantir a clareza e o fluxo lógico de navegação na interface de linha de comando (CLI) construída.

Competitividade: Foi feito um sistema de rank para que os usuários fiquem incentivados a constantemente usar o sistema

Adaptabilidade: O usuário pode escrever e publicar novos textos que outros usuários do

programa possam interagir e ler, permitindo um bom fluxo de novas informações.

3.2. Definição de Fluxos Alternativos e de erro e seus tratamentos

A definição dos fluxos foi feita pensando em um fluxo principal a ser seguido, erros e caminhos alternativos que o usuário pode acabar realizando. Os fluxos alternativos mais importantes são por exemplo, o de recuperação de senha, que quando o usuário acaba esquecendo a senha, ele coloca um email válido, e um código de recuperação de senha é enviado para ele. Esse fluxo é de suma importância, tendo em vista que um usuário acaba perdendo sua conta, ele vai perder todas suas anotações particulares e marcações de textos. Um fluxo de erro muito significante, é o de senha fraca, caso o usuário escolha uma senha considerada fraca ou seja, menos que 8 caracteres ou mais que 12, não possuem pelo menos 1 letra maiuscula, 1 caractere especial e 1 número. Então o sistema vai pedir para que ele crie uma senha considerada forte, visando a segurança de sua conta. Esses foram exemplos de fluxos importantes no cadastro. Alguns exemplos de fluxos importantes na escolha de idiomas é o fluxo alternativo. Caso o usuário escolha um número errado e seja redirecionado a um texto que não queria ler, existe a opção de voltar, dentro do menu, essa opção ajuda a manter o menu mais flexível, mais correto, permitindo que o usuário tenha a opção de ler um texto e depois volte para a parte principal.

3.3. O que foi usado no código e onde foi feito?

A linguagem de programação adotada para o desenvolvimento do projeto foi Python (versão 3.x), escolhida pela sua robustez na manipulação de strings e vasta disponibilidade de bibliotecas padrão. A arquitetura do sistema foi modularizada para garantir a separação de responsabilidades, fazendo uso de bibliotecas nativas e externas para atender aos requisitos funcionais.

No que tange à segurança e autenticação, utilizou-se a biblioteca externa maskpass, implementada nos módulos de login e cadastro para ocultar a entrada de caracteres no terminal, garantindo a confidencialidade das senhas durante a digitação. Complementarmente, a biblioteca nativa secrets foi empregada no módulo de recuperação de senha (recuperação_senha.py) para a geração de tokens hexadecimais criptograficamente seguros, substituindo o uso de geradores pseudoaleatórios comuns.

Para a comunicação e notificações, o sistema integra as bibliotecas smtplib e email.message. Estas foram utilizadas para estabelecer conexão com servidores SMTP e estruturar o envio automatizado de e-mails contendo os códigos de verificação para redefinição de credenciais.

O processamento e validação de dados foi realizado através das bibliotecas re (Regular Expressions) e unicodedata. A primeira foi aplicada no módulo verificação.py para validar padrões complexos de endereços de e-mail e sanitizar strings, enquanto a segunda foi essencial no módulo util.py para a normalização de textos (remoção de

acentos e diacríticos), facilitando a comparação entre termos nos idiomas estrangeiros e a busca no acervo.

A persistência dos dados foi implementada sem o uso de Sistemas Gerenciadores de Banco de Dados (SGBD) convencionais, optando-se pela biblioteca json. O sistema armazena e recupera informações de usuários, textos e interações (curtidas e comentários) em arquivos de texto estruturados (.json), manipulados principalmente pelos módulos dados.py e usuario.py.

Por fim, o controle de fluxo temporal e a lógica de gamificação utilizaram a biblioteca time, permitindo o cálculo do tempo de permanência em leitura para a atribuição justa de pontos de experiência (XP). O ambiente de desenvolvimento integrado (IDE) empregado foi o VSCode, e o versionamento de código bem como o controle de releases foram gerenciados através da plataforma GitHub.

3.4 Code Review

Durante o período de desenvolvimento inicial, o projeto Orbitext passou por uma fase de *Code Review* conduzida pela equipe de desenvolvimento, com o objetivo de garantir a qualidade do código, aderência a padrões de nomenclatura e otimização estrutural. Os critérios de avaliação utilizados foram:

1. Variáveis e Nomenclatura;
2. Operadores e Operações;
3. Estruturas de Decisão;
4. Estruturas de Repetição;
5. Indentações.

A seguir, detalhamos os achados e as refatorações realizadas com base no *review*:

3.4.1 Foi encontrado nomes confusos na nomenclatura de algumas variáveis. Por exemplo, o local onde era armazenado onde era armazenado os textos com suas listas foram considerados confusos, já que vaiaram tanto no nome do país original onde os textos eram falados quanto no nome da língua do texto tanto em portugues quanto na língua inglesa, por causa disso, foi adotado um método padrão para colocar o nome das nomenclaturas nos quais a lista de texto eram armazenadas.

Além disso, outras variáveis eram enumeradas de maneira confusa, com algumas enumerações sendo feitas com base na ordem de criação delas, não na ordem em que elas aparecem no código. Além disso, foi notado que algumas partes do codigo não precisavam ser enumeradas, por exemplo a parte de tentativas, que se encontra em usuario.py, na classe cadastrar_usuario , já que o valor sempre é resetado para 0 antestes dele ser modificado novamente.

3.4.2 Foi encontrado erros na coloração que não foram vistos no teste exploratório, em alguns casos a mensagem que aparece colorida como resultado de alguma operação no voltavam ao normal quando a mensagem deixa de aparecer, o que resolutivo em todos

texto posterior ficar de uma cor inapropriada. para resolver esse problema, só foi necessário tirar uma barra quando uma mensagem que utilizava o colorama aparecia.

3.5. Teste Exploratório

Título do bug: 001 - **validação nome**

Passos para Reproduzir:

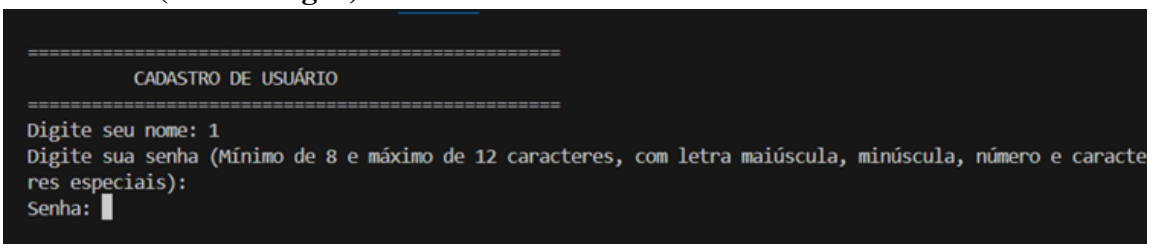
1 - ir para o cadastro

2 - inserir um nome iniciando com valores numéricos

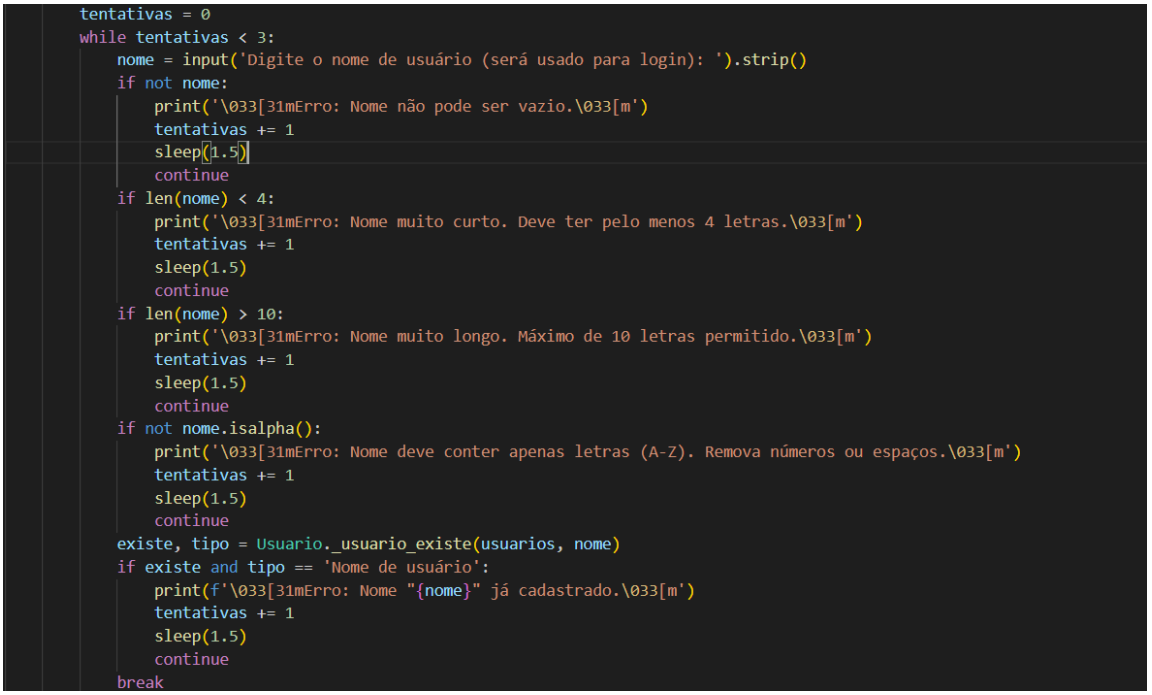
Comportamento atual: O sistema permite o cadastro, mesmo que o nome contenha apenas números.

Comportamento esperado: O sistema bloquear a tentativa imediatamente, exibindo uma mensagem de erro.

Evidências (Prints ou gifs)



Resultado após o texto exploratório para o código: Pelo fato do nome ter poucos caracteres, podendo ser formado por apenas um dígito, e o fato dele poder ser formado por apenas números, foi considerado inapropriado para o nosso código é feito mudança imediatas. Para isso, foi retirada a possibilidade de ter um nome vazio, ter letras no nome e imposto um limite de tamanho para o nome em se.



```
    tentativas = 0
    while tentativas < 3:
        nome = input('Digite o nome de usuário (será usado para login): ').strip()
        if not nome:
            print('\033[31mErro: Nome não pode ser vazio.\033[m')
            tentativas += 1
            sleep(1.5)
            continue
        if len(nome) < 4:
            print('\033[31mErro: Nome muito curto. Deve ter pelo menos 4 letras.\033[m')
            tentativas += 1
            sleep(1.5)
            continue
        if len(nome) > 10:
            print('\033[31mErro: Nome muito longo. Máximo de 10 letras permitido.\033[m')
            tentativas += 1
            sleep(1.5)
            continue
        if not nome.isalpha():
            print('\033[31mErro: Nome deve conter apenas letras (A-Z). Remova números ou espaços.\033[m')
            tentativas += 1
            sleep(1.5)
            continue
        existe, tipo = Usuario._usuario_existe(usuarios, nome)
        if existe and tipo == 'Nome de usuário':
            print(f'\033[31mErro: Nome "{nome}" já cadastrado.\033[m')
            tentativas += 1
            sleep(1.5)
            continue
        break
```

Título do bug: 002 - validação e-mail Passos para

Reproduzir (exemplo):

1 - ir para o cadastro

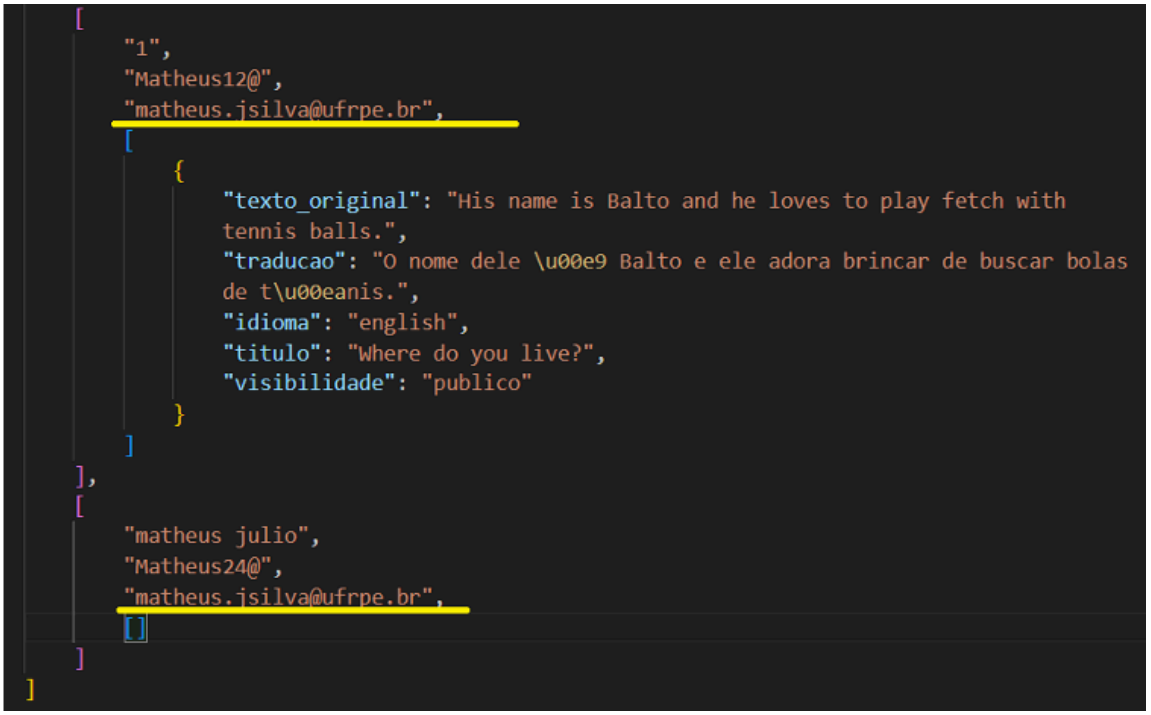
2 - inserir um e-mail

3 - repetir a etapa 1 e adicionar o mesmo e-mail

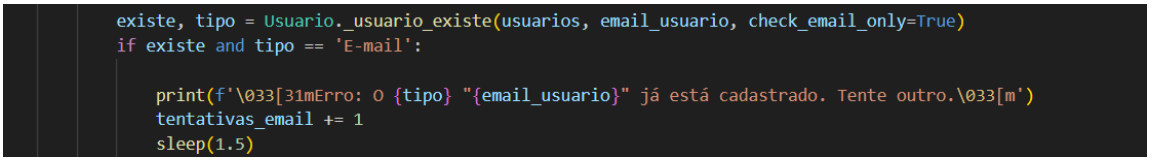
Comportamento atual: O sistema permite que o mesmo e-mail pode ser cadastrado ilimitadamente.

Comportamento esperado: O sistema interromper o cadastro caso o usuário tente adicionar um e-mail já cadastrado.

Evidências (Prints ou gifs)



Resultado após o texto exploratório para o código: Esse erro foi considerado inaceitável para o desenvolvimento do código, pois, além de atrapalhar a parte de login, com um usuário podendo se confundir com suas várias contas de e-mail, ele também atrapalha a parte de recuperação de senha, já que o usuário só recuperar com o código de senha apenas uma conta com o e-mail registrado. Por causa disso, foi feita a modificação do código para que um mesmo e-mail não possa se registrar duas vezes no mesmo código.



Esse é o print do código corrigido do nosso trabalho, no qual impedimos que um usuário coloque um e-mail que já existe no cadastro.

4. Resultado

Primeiramente foi a parte do menu cadastro, que foi feito simultaneamente com a parte de senha e a parte de verificação. A parte mais difícil do main(menu cadastro) foi colocar os ‘handle’ e suas funções para melhorar as mensagens e o entendimento do usuário, além de filtrar, parcialmente, alguns erros.

```python
def handle_recuperacao():
    limpar_tela()
    print("\n--- INICIANDO RECUPERAÇÃO DE SENHA (INTERAJA ABAIXO) ---")

    try:
        recuperação_senha.Recuperar_Senha(usuario.Usuario.usuarios, main_console).menu_pri
    except Exception as e:
        print(f"\033[31mERRO durante a recuperação: {e}\033[m")
        sleep(2)

def handle_remover_usuario():
    if usuario.Usuario.remover_usuario(usuario.Usuario.usuarios):
        usuario.Usuario.usuarios = usuario.Usuario.carregar_usuarios()
```

A parte de usuario, em geral, foi mais complicado a implementação de dicionários json dentro dela, além de ter que reabrir dicionários json que já tinham sido fechados anteriormente(usuario.json) para colocar novos dados, além de ter que criar um formato para abrir e interpretar esses novos parágrafos em json. Essas dificuldades ficaram mais presentes, principalmente, com a manipulação de parágrafos que estão no arquivo usuario.py. Houve também uma grande dificuldade na implementação e manipulação de dicionários e tuplas, mas elas ficam mais visíveis na parte de menu_leitura.

```python
    @staticmethod
    def salvar_paragrafos_publicos(novo_paragrafo, remover=False):
        paragrafos = Usuario.carregar_paragrafos_publicos()
        if remover:
            paragrafos = [p for p in paragrafos if p.get('id') != novo_paragrafo.get('id')]
        else:
            if novo_paragrafo.get('visibilidade') == 'publico' or novo_paragrafo.get('visibilidade') == 'privado':
                paragrafos.append(novo_paragrafo)
        with open(ARQUIVO_PUBLICOS, 'w', encoding='utf-8') as f:
            json.dump(paragrafos, f, indent=4, ensure_ascii=False)
```

```python
@staticmethod
def mostrar_meus_paragrafos(para_remover=False):
    util.limpar_tela()
    if not Usuario.usuario_logado:
        print('\033[31mVocê precisa estar logado para ver seus textos.\033[m')
        sleep(1.5)
        return
    nome = Usuario.usuario_logado[0]
    ids = Usuario.usuario_logado[3] if len(Usuario.usuario_logado) > 3 else []

    itens = []
    for tid in ids:
        texto = dados.DataManager.buscar_texto_por_id(tid)
        if texto:
            itens.append({'id': tid, 'titulo': texto.get('Titulo'), 'tipo': 'personalizado', 'autor': texto.get('Autor')})
            continue
        p = next((pp for pp in Usuario.carregar_paragrafos_publicos() if pp.get('id') == tid), None)
        if p:
            itens.append({'id': tid, 'titulo': p.get('titulo'), 'tipo': 'paragrafo', 'autor': p.get('autor'), 'visibilidade': p

    if not itens:
        print('\n\033[33mNenhum texto/parágrafo salvo por você.\033[m')
        input('Pressione ENTER para voltar...')
        return

    while True:
        util.limpar_tela()
        print('=' * 50)
        print(f"       MEUS TEXTOS ({nome})")
        print('=' * 50)
        for i, it in enumerate(itens):
            tipo = 'Texto' if it['tipo'] == 'personalizado' else 'Parágrafo'
            ref = ''
            vis = ''
            snippet = ''
```

Na parte de munu_leitura.py foi apresentado as maiores dificuldades, que se concentraram em salvar os textos e interpretar os textos já salvos, todo devidamente encaixado em sua tuplas e dicionários e com suas nomeações por por idiomas, além do tratamento do fluxo de erro, o que foi a parte mais complicada de trabalhar no código.

```python
    num_mostrar = 0
    while True:

        limpar_tela()
        print('=' * 50)
        print(f"      LENDO: {texto['Titulo'].upper()} (Parágrafo {indice_atual + 1} de {num_paragrafos})")
        print('=' * 50)

        entrada = input(prompt).strip()

        if not entrada:
            num_mostrar = 1
            break

        try:
            num_mostrar = int(entrada)
            if num_mostrar > 0 and num_mostrar <= restantes:
                break
            else:
                print(f"\033[31mNúmero inválido. Digite um número entre 1 e {restantes}.\033[m")
                sleep(1.5)
        except ValueError:
            print("\033[31mEntrada inválida. Digite um número ou apenas ENTER.\033[m")
            sleep(1.5)


    paragrafos_exibidos = exibir_paragrafos_organizado(paragrafos, indice_atual, num_mostrar, idioma_key)
```

```
        while True:
            modo_salvamento = input("Opção: ").strip()

            if modo_salvamento == '0':
                break

            if modo_salvamento in ['1', '2']:
                tipo_visibilidade = 'publico' if modo_salvamento == '1' else 'privado'


                salvo_id = dados.DataManager.salvar_paragrafo_publico(
                    titulo_paragrafo,
                    idioma_key,
                    paragrafo_obj.get('Lingua'),
                    paragrafo_obj.get('portugues'),
                    tipo_visibilidade,
                    paragrafo_numero=num_paragrafo_escolhido,
                    texto_id=texto.get('id', dados.DataManager.gerar_novo_id())
                )
                if salvo_id and usuario.Usuario.usuario_logado:
                    if not usuario.Usuario.usuarios:
                        usuario.Usuario.usuarios = usuario.Usuario.carregar_usuarios()
                    usuario.Usuario.usuario_logado[3].append(salvo_id)
                    usuario.Usuario.salvar_usuarios(usuario.Usuario.usuarios)
                print(f'\033[32mParágrafo {num_paragrafo_escolhido} salvo como {tipo_visibilidade} com sucesso!\033[m')
                sleep(1.5)
                break
            else:
                print("\033[31mOpção de visibilidade inválida. Digite 1, 2 ou 0.\033[m")
                sleep(1)

    else:
        print(f"\033[31mNúmero de parágrafo inválido. Escolha entre {indice_atual + 1} e {indice_atual + paragrafos_exi
        sleep(1.5)
except ValueError:
```

(class) Usuario

```
while True:
    limpar_tela()
    print(f"--- Adicionando Parágrafo {num_paragrafo} ---")
    print("Digite 0 para cancelar a criação do texto e voltar ao menu.")
    print('=' * 50)
    texto_original = input(f"[{idioma_original}] Parágrafo Original: ").strip()
    if texto_original == '0':
        return
    if not texto_original:
        print("\033[31mO parágrafo original não pode ser vazio.\033[m")
        sleep(1.5)
        continue
    texto_traducao = input("[Português] Tradução correspondente: ").strip()
    if texto_traducao == '0':
        return
    if not texto_traducao:
        print("\033[31mA tradução não pode ser vazia.\033[m")
        sleep(1.5)
        continue

    lista_paragrafos.append({'original': texto_original, 'traducao': texto_traducao})
    num_paragrafo += 1
    while True:
        print('\nEscolha: 1 - Adicionar outro parágrafo | 0 - Finalizar texto e salvar')
        escolha_cont = input('Opção (0/1): ').strip()
        if escolha_cont == '1':
            break
        elif escolha_cont == '0':
            break
        else:
            print("\033[31mOpção inválida. Digite 1 ou 0.\033[m")
            sleep(1)
    if escolha_cont == '0':
        break
```

Outro arquivo no qual houve dificuldade na sua codificação foi os dados.py, pois, mesmo ele sendo menor que menu_leitura.py, ele é a base do menu leitura, no qual a maior parte das fontes de json são chamadas e onde os arquivos json são armazenados em python até sua transformação em json.

```python
NOME_ARQUIVO = 'textos_idiomas.json'
ARQUIVO_PUBLICOS = 'paragrafos_publicos.json'
ARQUIVO_COMENTARIOS = 'comentarios.json'
ARQUIVO_LIKES = 'likes.json'

class DataManager:
    IDIOMAS_MAP = {
        '1': 'english',
        '2': 'french',
        '3': 'spanish',
    }

    IDIOMAS_NOMES = {
        'english': 'Inglês',
        'french': 'Francês',
        'spanish': 'Espanhol'
    }

    @staticmethod
    def carregar_textos_idiomas():
        if path.exists(NOME_ARQUIVO):
            with open(NOME_ARQUIVO, 'r', encoding='utf-8') as f:
                try:
                    return json.load(f)
                except json.JSONDecodeError:
                    return {"english": [], "french": [], "spanish": []}
        return {"english": [], "french": [], "spanish": []}

    @staticmethod
    def salvar_textos_idiomas(data):
        with open(NOME_ARQUIVO, 'w', encoding='utf-8') as f:
            json.dump(data, f, indent=4, ensure_ascii=False)

    @staticmethod
    def gerar_novo_id():
        return str(int(time() * 1000))
```

5. Conclusão e Trabalhos Futuros

Com relação ao andamento do trabalho, ele pode ser considerado aceitável, entregando os requisitos básicos para deixar o usuário imersivo, como as competições e um número razoável de texto que pode crescer de forma indefinida, além de apresentar, de maneira fácil e intuitiva, os textos que podem ser lidos e que tem uma boa dose de interação no próprio texto em si, a capacidade de salvar parágrafos e de comentar os parágrafos que foram salvos.

Com relação ao trabalho futuro, foi pensado a possibilidade de aplicar uma interface gráfica no código, para melhorar a imersão do usuário, além disso, não existe um sistema de notas para os textos, algo que também pode ser implementado em uma melhora futura. Outro aspecto de melhora seria na visualização tanto dos parágrafos

quanto dos textos e comentários, com os mais curtidos podendo ter prioridades na visualização em detrimento dos menos curtidos.

6. References

PUTRA, Erlangga; NOPEMBER, S. The importance of learning English nowadays. Jurnal Institute of Technology Sepuluh November at Surabaya, v. 1, n. 7, p. 7, 2020.

The Brazil Business. (n.d.). *The Value of a Second Language in Brazil*. Available at: https://thebrazilbusiness.com/article/the-value-of-a-second-language-in-brazil

DEL VALLE, José; VILLA, Laura. Spanish in Brazil: Language policy, business, and cultural propaganda. **Language Policy**, v. 5, n. 4, p. 371-394, 2006.

KODA, Keiko; YAMASHITA, Junko (Ed.). **Reading to learn in a foreign language: An integrated approach to foreign language instruction and assessment**. Routledge, 2018.

ALVES, Pedro; CORDEIRO, Laura. Mutare-project: Repositório para MUTARE – PISI1 – Projetos Interdisciplinares de Sistemas de Informação 1. 2025. Disponível em: https://github.com/pedroailton/mutare-project . Acesso em: 26 nov. 2025.