Título temporário

Matheus J. Silva[1], **Cayo V. Menezes**[1], **Cleyton V. C. de Magalhães**[1]

[1]Bacharelado em Sistemas de Informação – Universidade Federal Rural de Pernambuco (UFRPE)
Endereço: Rua Dom Manuel de Medeiros, s/n - Dois Irmãos - Recife - PE - CEP: 52171-900

matheus.jsilva@ufrpe.br, cayo.menezes@ufrpe.br, cleyton.vanut@ufrpe.br

Abstract. *Abstract — RURALNECT is a project developed in Python aimed at supporting the UFRPE community, especially undergraduate students, in their academic development through tools that provide video lectures, a forum, question lists, and bibliographic content. In addition, it seeks to reduce the informational gap caused by fragmented data by offering centralized resources, such as an institutional information area. Although it does not present a markedly innovative character, the application integrates, in a unified environment, functionalities that enhance organization, knowledge access, and the overall academic experience.*

Resumo. *A RURALNECT é um projeto desenvolvido em Python cujo objetivo é auxiliar a comunidade da UFRPE, especialmente os discentes, em seu desenvolvimento acadêmico, por meio de ferramentas que disponibilizam videoaulas, fórum, listas de questões e conteúdos bibliográficos. Além disso, busca reduzir o déficit informacional ocasionado pela fragmentação de dados, ao oferecer recursos centralizados, como uma área destinada a informações institucionais. Embora não apresente caráter marcadamente inovador, a aplicação reúne, em um ambiente unificado, funcionalidades que favorecem a organização, o acesso ao conhecimento e a melhoria da experiência acadêmica.*

1. INTRODUÇÃO

Cotidianamente, visualizamos uma grande quantidade de informações, as quais são indispensáveis e essenciais tanto para a sociedade, no que diz respeito aos avanços científicos e, consequentemente, o desenvolvimento de novas tecnologias, como para o aprimoramento pessoal e profissional dos indivíduos [Del Claro 2009]. E, atualmente, por meio do uso de dispositivos eletrônicos como Telefones Celulares, Computadores e Tablets, além da conexão de internet, atrelados à Tecnologia da Informação (TI), permitiu que a disseminação de conhecimentos e informações sejam mais eficazes, abrangendo a maior parte da população mundial [Nunes et al. 2023].

No entanto, apesar de possibilitar o acesso aos conteúdos disponibilizados na internet, há o que pode ser chamada de ”fragmentação da informação” que é, resumidamente, a dispersão das informações nos ambientes virtuais, o que dificulta ou torna complexa a procura por algo específico. Naturalmente, o mesmo pode ser aplicado nas instituições de ensino, independente do nível de instrução, visto que estas organizações, principalmente as Universidades, noticiam muitos eventos, sem mencionar os eventos organizados por estudantes [Saplacan et al. 2020].

Ademais, embora as redes sociais sejam frequentemente utilizadas na comunicação, seja para uso pessoal ou profissional, ela não é muito eficaz em noticiar

e ”fixar” informações para grandes públicos, pois, diferente de fóruns de discussão que possibilitam a criação de tópicos sobre determinados assuntos, os quais ficam ”permanentemente” fixados e podem ser procurados a qualquer momento por meio de palavras chaves ou busca manual, as informações presente nas redes sociais é facilmente perdida devido à frequente adição de novos conteúdos. [Rodriguez et al. 2014]

Desse modo, o projeto RURALNECT, sistema desenvolvido para a Universidade Federal Rural de Pernambuco (UFRPE), tem a iniciativa de implementar uma aplicação voltada para a área da educação e informação, fornecendo informações padrões básicas para os discentes, auxiliando principalmente os ingressantes, e também informações relativas aos cursos, bem como uma área de perguntas e respostas (fórum), disponibilização de videoaulas relacionadas ao curso do discente, exercícios baseados em assuntos escolhidos, dentre outras ferramentas.

2. TRABALHOS RELACIONADOS

O presente estudo fundamenta-se diretamente no trabalho de [Alves and Cordeiro 2025], cuja contribuição se mostrou essencial para a definição dos pilares conceituais e metodológicos adotados nesta pesquisa. O referido trabalho apresentou um modelo inicial destinado à organização, tratamento e integração de dados em ambientes computacionais colaborativos, estabelecendo diretrizes que têm orientado investigações subsequentes na área de sistemas informacionais.

Entre as principais contribuições de [Alves and Cordeiro 2025], destaca-se a concepção do Mutare, uma ferramenta voltada ao gerenciamento consciente de hábitos e ao incentivo à adoção de práticas saudáveis e produtivas. Inspirado na psicologia do hábito e nas formulações de Charles Duhigg em O Poder do Hábito, o Mutare oferece suporte ao desenvolvimento e acompanhamento de novos hábitos, à correção de comportamentos prejudiciais e à adoção de práticas sustentáveis e cidadãs. Para isso, incorpora mecanismos de recompensa, métricas de desempenho e a presença de um mascote interativo, configurando uma solução tecnológica orientada ao bem-estar e ao autocuidado.

De forma complementar, observa-se o projeto RURALNECT, desenvolvido para a Universidade Federal Rural de Pernambuco (UFRPE). O sistema tem como objetivo fornecer suporte informacional e educacional aos discentes, especialmente aos ingressantes, disponibilizando dados institucionais essenciais, informações específicas sobre cursos, área de perguntas e respostas, videoaulas e exercícios organizados por temas. A proposta visa otimizar a integração acadêmica, ampliar o acesso à informação e fortalecer o processo de aprendizagem por meio de ferramentas digitais estruturadas.

O estudo de [Alves and Cordeiro 2025] também evidenciou lacunas relevantes na literatura, particularmente no que se refere à integração entre ferramentas colaborativas e mecanismos de versionamento de software. A presente pesquisa se apoia nesse diagnóstico, propondo avanços metodológicos e tecnológicos que dialogam tanto com o modelo de gerenciamento de hábitos do Mutare quanto com os princípios de suporte educacional consolidados no RURALNECT.

Assim, o trabalho precursor [Alves and Cordeiro 2025] constitui o alicerce teórico e metodológico desta investigação, orientando a formulação das hipóteses, a definição dos procedimentos analíticos e o desenvolvimento das soluções apresentadas. Sua relevância

se manifesta não apenas como referência conceitual, mas como base prática para a consolidação dos resultados aqui discutidos.

3. METODOLOGIA

3.1. IDENTIFICAÇÃO DOS REQUISITOS

O levantamento dos requisitos funcionais e, consequentemente, seu desenvolvimento foi feito com base em aplicações que executam atividades semelhantes à iniciativa deste projeto.

Portanto, inspirando-se no Sistema Integrado de Gestão de Atividades (SIGAA), para que possa ser feita a identificação dos usuários, que inclui unicamente integrantes da Universidade Federal Rural de Pernambuco (UFRPE), foram definidas as funcionalidades de cadastro e login, as quais são responsáveis, respectivamente, pela criação da conta do usuário e o meio para que ele possa acessar o sistema.

Além disso, apesar de não ser um requisito funcional, é interessante mencionar que, usando como base aplicações que dispõem de um sistema de cadastro e login (C.R.U.D), foi adicionada uma opção de recuperação de senha, possibilitando que o usuário, mesmo esquecendo a senha, possa recuperar sua conta e obter novamente o acesso.

Após a elaboração do Menu Inicial, o qual contém unicamente os requisitos anteriores, foram estabelecidas 6 requisitos, sendo eles:

Table 1. Requisitos Funcionais do Sistema Agrupados por Release

Feature	Requisito Funcional
1ª RELEASE	Menu Inicial
1ª RELEASE	Cadastro (discente e docentes)
1ª RELEASE	Login
1ª RELEASE	Menu Principal
1ª RELEASE	Área de informações (1)
1ª RELEASE	Área com links de vídeoaulas (2)
2ª RELEASE	Sistema de cadastramento de perguntas e respostas (3)
2ª RELEASE	Área com conteúdo bibliográfico (4)
2ª RELEASE	Área com links gerais (Redes sociais, site oficial, etc) (5)
2ª RELEASE	Sistema de lista de exercícios (6)

1. A área com informações dos cursos (1) foi inspirada no site oficial da UFRPE, que disponibiliza informações relativas aos cursos, e redireciona os usuários para as informações desejadas dos seus respectivos cursos;
2. A área com Links de videoaulas (2) foi desenvolvida com base no Brainly, aplicação voltada a área da educação, que exibe livros com base em uma pesquisa interna. Neste caso, ao invés de exibir livros, serão exibidas videoaulas com base no curso do usuário;
3. O sistema de cadastramento de perguntas e respostas (3), ou, para simplificar, Fórum é uma ferramente que permite ao usuário que sejam feitas perguntas e que, consequentemente, sejam respondidas;

4. A área de bibliografia (4) exibe livros, apostilas, ou quaisquer conteúdo bibliográfico com base nas escolhas do usuário dentro da aplicação, redirecionando-o para o conteúdo;
5. A área com links gerais (5) visa permitir que o usuário acesse as redes sociais, sites oficiais e/ou sites independentes, criados por estudantes, garantindo que ele obtenha acesso a todos os meios de comunicação da universidade existentes;
6. O sistema de lista de exercícios foi baseado em aplicações como Qconcursos e Passei Direto, que disponibilizam uma série de exercícios filtrados por assunto, disciplina, etc.(6)

3.2. DEFINIÇÃO DE FLUXOS ALTERNATIVOS E DE ERRO E SEUS TRATAMENTOS

Neste seção, vamos abordar os principais e mais recorrentes tratamentos de erros que podem ocorrer durante a execução do programa.

```
try:
    opcao = int(input('\nInsira a opção desejada: '))
    condicionais_menu(opcao)
except ValueError:
    Util.erro_txt('O valor inserido não é um número inteiro, tente novamente!')
    Util.pausa(5)
```

Figure 1. código de erro de valor.

Em primeiro lugar, por ser um programa que opera, de momento, apenas no terminal, o sistema funciona unicamente por meio das entradas feitas pelo teclado. E, como a navegação é baseada em dígitos inteiros para acessar as funcionalidades, caso o usuário insira um valor que não está presente ou que sejam um caractere alfabético, o sistema exibe uma mensagem de erro.

Então, o código inicialmente executa o **try** e, se o valor for diferente de um número inteiro, é feita uma exceção que, ao invés de finalizar o programa exibindo o erro padrão, é simplemente exibida a mensagem de erro mantendo a continuidade do sistema.

Observe a estrutura do código na Figure 1.

```
else:
    Util.erro_txt('A opção inserida é inválida!')
    Util.pausa(3)
    return
```

Figure 2. mensagem de erro para números inválidos.

Na Figure 2, quando o usuário insere um valor numérico que não está previsto no código, é exibida uma mensagem falando que o número inserido não é válido.

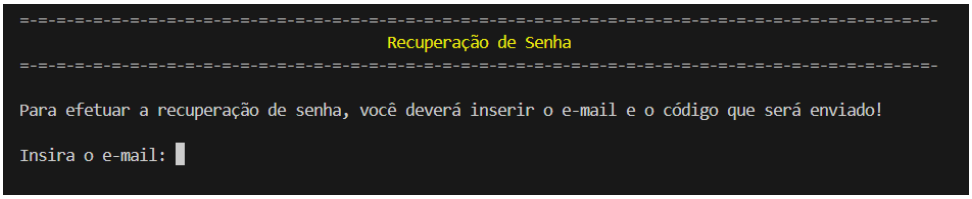

Figure 3. fluxo alternativo do login.

Na área de login, foi adicionado um fluxo alternativo que possibilita aos usuários a capacidade de recuperar a senha, caso tenham esquecido. O código para o envio do e-mail foi disponibilizado por um criador de conteúdo que atua no ramo da tecnologia[Programação 2022].

```
try:
    email_obj.esqueci_minhasenha(usuarios, user, email_user)

except socket.gaierror:
    Util.erro_txt('E-mail não enviado: Usuário sem internet')
    Util.continuar()
    return menu_inicial()
```

Figure 4. mensagem de erro para números inválidos.

Neste caso, presente na Figure 4, é feito o tratamento de erro em caso de envio de e-mail sem interrnet.

Caso o usuário solicitasse o código para redefinição de senha sem a conexão com a internet, o sistema era finalizado e exibia a mensagem padrão de erro. E, para solucionar o problema, foi importada a biblioteca socket para fazer esta validação, que se encontra no **Except**, permitindo que apareça a mensagem de erro e o sistema continue normalmente.

3.3. TECNOLOGIAS USADAS

Para o desenvolvimento deste projeto, que momentaneamente opera apenas no terminal, foram utilizadas as seguintes tecnologias:

Table 2. Tecnologias Utilizadas no Desenvolvimento do Projeto

Categoria	Tecnologias
Linguagem de Programação	Python
Persistência de Dados	Listas e Dicionários (Python)
IDE / Ferramentas	Visual Studio Code, GitHub (Repositório), GitHub Desktop

Para o processo de codificação do projeto, foi utilizada a linguagem de alto nível Python, a qual dispõe de uma sintaxe relativamente fácil, bem como suporta múltiplos paradigmas de programação, incluindo a Programação Orientada a Objetos (POO). A Orientação a Objetos foi incorporada quase que completamente ao programa, permitindo a modularização e reutilização dos códigos em outros arquivos. Ademais, o Python

também disponibiliza a interface gráfica Tkinter, modulo presente na instalação padrão da linguagem, que pode eventualmente ser implementada no código para conferir interações mais dinâmicas e intuitivas.

Para o funcionamento do código, foram utilizadas diversas bibliotecas consideradas essenciais para a implementação de determinadas funcionalidades, além de contribuírem para uma interação mais clara e organizada pelo terminal. As principais bibliotecas empregadas estão empregadas no Table 3

Table 3. Bibliotecas e sua Funcionalidade no Sistema

Biblioteca	Funcionalidade
os	Módulo para limpeza do terminal (interface de linha de comando)
time	Inclusão de atraso temporal (delay) antes da exibição de resultados
smtplib	Estabelecimento de conexão com o servidor de e-mail (SMTP)
email.message	Construção e formatação das mensagens de correio eletrônico
secrets	Geração de códigos seguros para procedimentos de recuperação de senha
webbrowser	Redirecionamento do usuário para recursos externos (videoaulas, links, apostilas, etc.)
colorama	Aplicação de cores e estilos na saída do terminal
emoji	Suporte e inclusão de caracteres emoji na interface
maskpass	módulo para mascaramento de senha
socket	utilizado unicamente para fazer o tratamento de erro do envio do e-mail, com ausência de internet

Dando prosseguimento à Table 2, sobre as tecnologias utilizadas no projeto, foram utilizadas listas e dicionários para fazer a manipulação das informações dos usuários. Além disso, para executar a codificação, foi utilizado o Visual Studio Code (VSCODE), que é o principal e mais utilizado editor de texto, pois possui uma interface amigável e permite a instalação de extensões que facilitam o processo de codificação, além de possibilitar a integração com diferentes linguagens.

Em associação com o Visual Studio Code, tem-se a utilização do GitHub que é uma plataforma de hospedagem de código-fonte online, em outras palavras, é um repositório remoto que detém a função de armazenar o código-fonte conforme for alterado. O GitHub utiliza o Git para fazer o controle de versionamento, ou seja, a alteração de um arquivo ou um conjunto deles, modificando-os para uma versão mais recente ou retornar a uma versão anterior.

Apesar de programadores experientes terem preferência por utilizar os comandos do Git pelo terminal, utilizou-se o GitHub Desktop, uma interface gráfica que simplifica

as funções do Git.

3.4. CODE REVIEW

Durante o período ”inicial” de desenvolvimento, o projeto foi testado por uma equipe de desenvolvedores, com o objetivo de identificar possíveis falhas e avaliar como o código está estruturado. Os desenvolvedores, Paulo Henrique e Felipe Augusto, responsáveis por liderar o projeto OrbiText, ferramenta voltada para o aprendizado de língua estrangeira, revisaram o código com os seguintes critérios:

1. Variáveis;
2. Operadores e Operações;
3. Estruturas de decisão;
4. Estruturas de repetição;
5. indentação.

Os revisores de código apontaram que as varíaveis (1) seguem o padrão snake_case, uma convenção de nomenclatura onde palavras (variáveis) devem ser escritas em minúsculo e, em caso de separação de espaço entre as palavras, deve ser adicionado o traço inferior, ou underline (_). Também identificaram algumas variáveis que, apesar de seguirem a convenção anteriormente citada, possuem nomes que dificultam sua compreensão, como ”opcao_geinfo”, que é uma variável para o usuário digitar o tipo de informação geral desejado. Esta variável foi renomeada para ”opcao_info_geral”, para torná-la mais clara, além de outros identificadores descritos em um arquivo à parte. Para finalizar esta parte, é importante ressaltar que não foi identificado nenhum problema relacionado à incialização das variáveis.

Os operadores e operações (2) estão, de acordo com eles, sendo utilizados corretamente, porém há repetitividade excessiva em alguns trechos de códigos, que podem ser consertados definindo blocos de código para serem reutilizados. Este problema foi, em parte, sanado nas versões mais recentes do programa, reduzindo consideravelmente o tamanho do código.

As estruturas de decisão (3) estão sendo utilizadas corretamente, porém a extensão do código dificulta a legibilidade, além de tornar o código relativamente mais pesado. Para a área de videoaulas, requisito definido na tabela 1 na subseção 3.1, o encadeamento das estruturas de decisão estavam, apesar de corretos e aninhados, extremamente repetitivos. Nas versões atuais do código, o problema foi, em sua maior parte, corrigido.

As estruturas de repetição (4), por sua vez, estão aplicadas corretamente na estrutura do código, mas uma sugestão dos revisores é a implementação de um while em certos trechos, para evitar que o usuário retorne para um ponto distante do qual ele está.

Por fim, as indentações (5) estão aplicadas corretamente, seguindo a convenção de quatro espaços para cada nível de recuo ao iniciar um bloco de código.

3.5. TESTE EXPLORATÓRIO

Nesta subseção, os desenvolvedores, Paulo Henrique e Felipe Augusto, responsáveis pelo projeto OrbiText, realizaram testes exploratórios com o intuito de verificar possíveis problemas no sistema. Para efetuar esse teste, eles devem seguir o caminho principal, porém verificando se os componentes do sistema apresentam alguma falha ou bug não previsto.

Ademais, há a presença de sugestões de aprimoramento para a versão atual. O layout é dividido em:

1. nome da falha/bug;
2. passos para reproduzir;
3. comportamento atual;
4. comportamento esperado;
5. evidência (imagem).

3.5.1. BUG 001 — Curso inválido

Nome do bug: Curso inválido

Passos para reproduzir:

1. No menu inicial, selecionar a opção ”Cadastro”.
2. Inserir o nome completo.
3. Inserir iniciais de curso inválidas.
4. O sistema permite concluir o cadastro.

Comportamento atual: Mesmo inserindo uma sigla de curso inexistente, o sistema aceita e finaliza o cadastro normalmente.

Comportamento esperado: O sistema deveria impedir o cadastro e exibir uma mensagem de erro solicitando uma sigla válida.

Evidência:

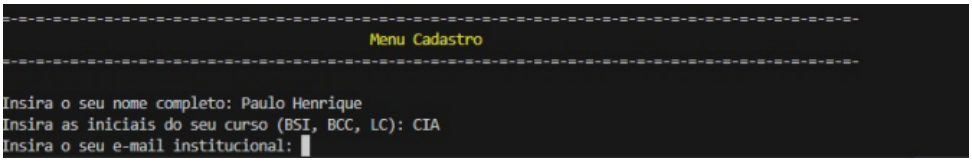

Figure 5. Processo intermediário — adicionando curso inválido

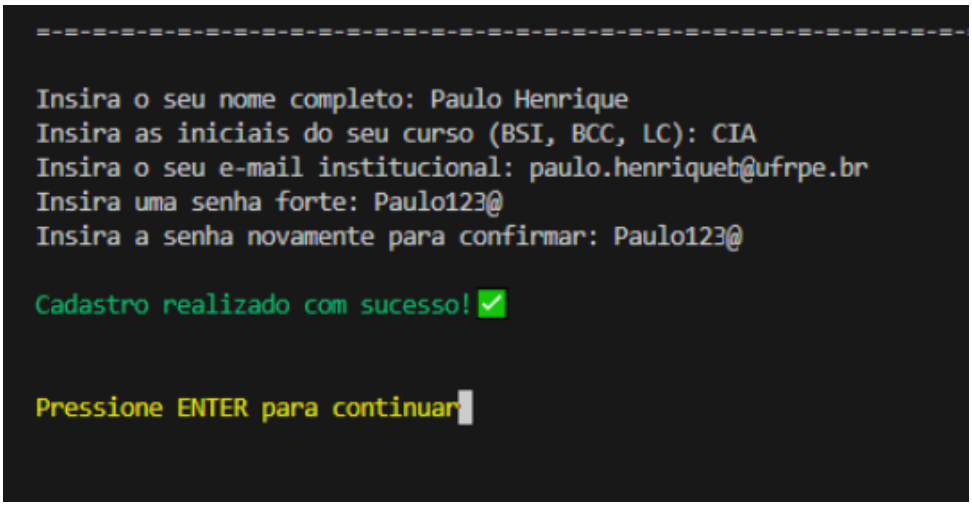

Figure 6. Processo final — Curso inválido sendo aceito e finalização do cadastro

3.5.2. BUG 002 — Fórum sem opção de retornar

Nome do bug: Fórum sem opção de retornar

Passos para reproduzir:

1. Efetuar o login.
2. No menu principal, acessar o fórum.
3. Inserir o valor ”0” para retornar ao menu principal.

Comportamento atual: Quando o usuário seleciona ”0”, ao invés de retornar ao menu principal, ele fica repetindo a página de fórum sem regressar ao menu.

Comportamento esperado: Quando o usuário selecionar ”0”, ele deve retornar ao menu principal.

Evidência:

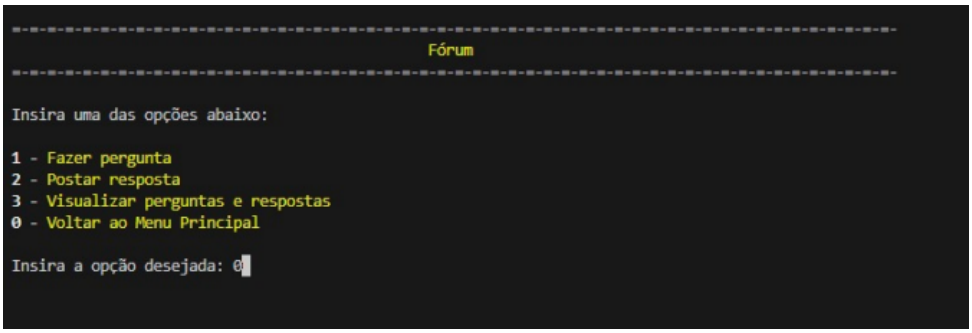

Figure 7. Inserindo o valor ”0” para retornar ao menu principal

3.5.3. BUG 003 — Código de recuperação de senha repetido

Nome do bug: Código de recuperação de senha repetido

Passos para reproduzir:

1. Realizar o cadastro.
2. Após o cadastro, acessar a área de login.
3. Selecionar a opção de recuperação de senha.
4. Inserir o e-mail cadastrado e aguardar o envio do código.
5. Repetir o processo e comparar os códigos.

Comportamento atual: O sistema está enviando o mesmo código de recuperação em vez de gerar um código aleatório. Isso é arriscado para o usuário, pois qualquer pessoa pode alterar a senha de uma conta utilizando o código fixo.

Comportamento esperado: O sistema deve enviar um código aleatório para cada solicitação, não um valor fixo.

Evidência:

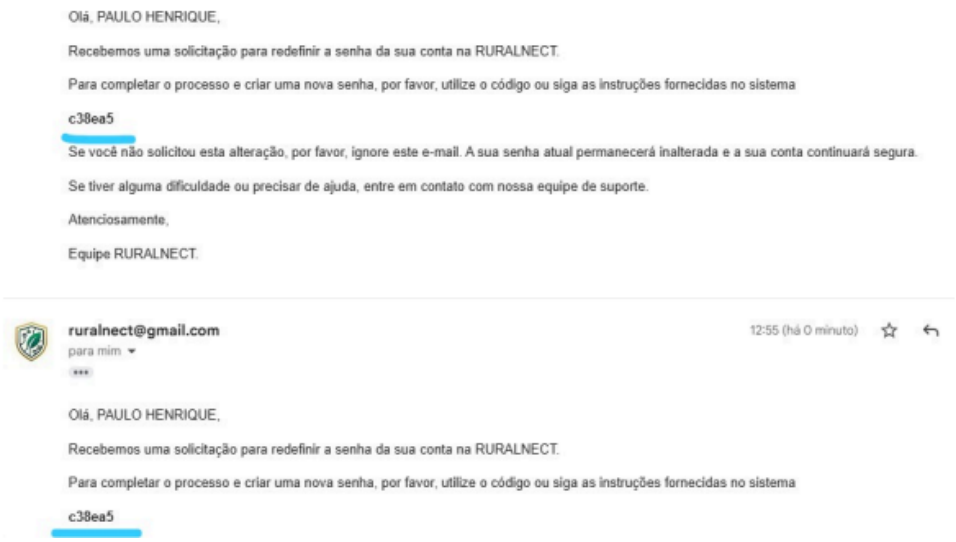

Figure 8. Solicitações diferentes com códigos iguais

3.5.4. BUG 004 — Aparição do login após exceder as tentativas

Nome do bug: Aparição do login após exceder as tentativas

Passos para reproduzir:

1. Realizar o cadastro.
2. Após o cadastro, acessar a área de login.
3. Selecionar a opção para efetuar login.
4. adicionar incorretamente os dados de entrada 3 vezes

Comportamento atual: Após o usuário exceder o limite de tentativas, aparece novamente os espaços para inserir os dados de entrada, ao invés da opção de recuperação de senha.

Comportamento esperado: Após o usuário exceder o limite de tentativas, deve aparecer a opção de recuperação de senha.

Evidência:

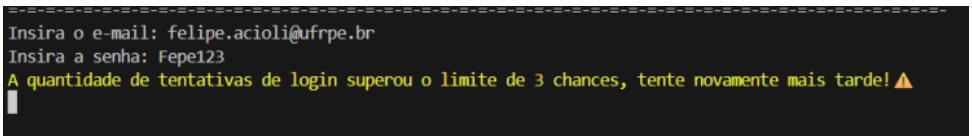

Figure 9. Solicitações diferentes com códigos iguais

3.5.5. BUG 005 — Retorno ao menu inicial após erro no menu principal

Nome do bug: Retorno ao menu inicial após erro no menu principal

Passos para reproduzir:

1. Realizar o cadastro.
2. Após o cadastro, acessar a área de login.
3. Selecionar a opção para efetuar login.
4. após efetuar o login, pressionar enter ou inserir um valor alfabético

Comportamento atual: Quando o usuário pressiona enter, sem inserir um valor, ou adiciona um valor alfabético, ele retorna ao menu inicial ao invés de permanecer no menu principal.

Comportamento esperado: Quando o usuário pressiona enter, sem inserir um valor, ou adiciona um valor alfabético, deve aparecer uma mensagem de erro e permanecer no menu principal.

Evidência:

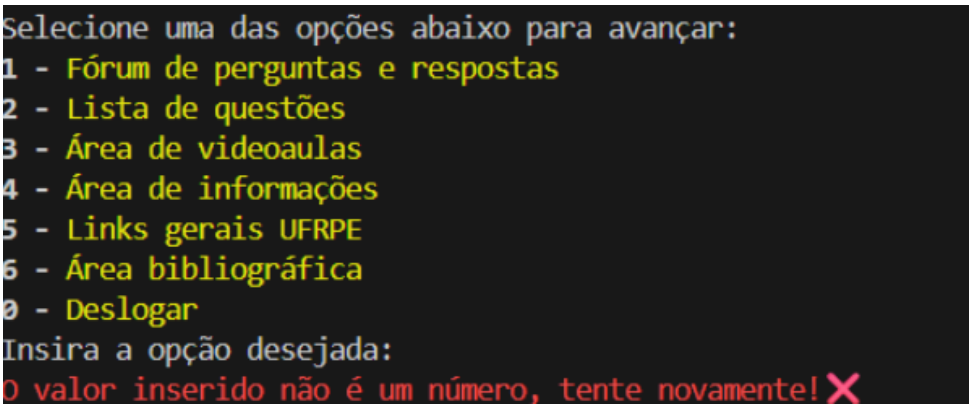

Figure 10. Processo intermediário — usuário erra no menu principal

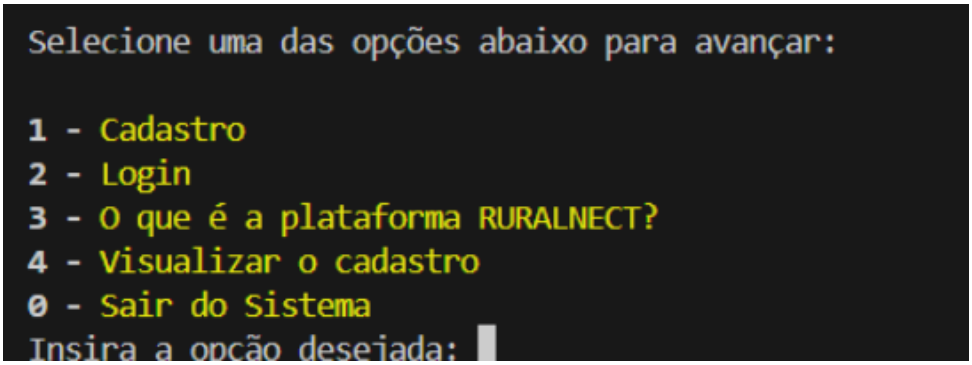

Figure 11. Processo final — retorno do menu principal ao inicial após o erro

3.5.6. BUG 006 — Login automático mesmo com erro na conta

Nome do bug: Login automático mesmo com erro na conta

Passos para reproduzir:

1. Acessar a área de login (sem necessidade de cadastro).
2. pressionar enter ou inserir um valor alfabético.

Comportamento atual: O sistema efetua a parte de login mesmo com o erro e vai para o menu principal.

Comportamento esperado: O usuário retornar ao menu inicial.

Evidência:

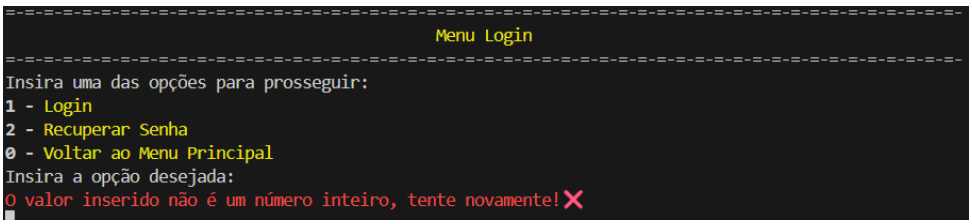

Figure 12. Processo intermediário — usuário erra na tela de login.

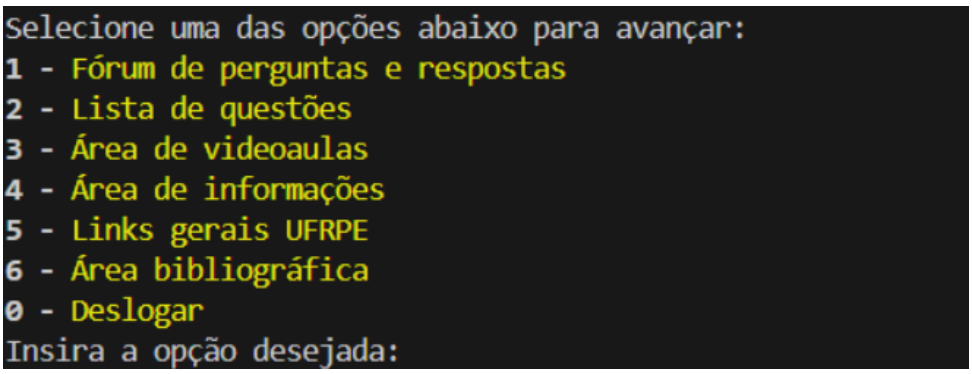

Figure 13. Processo final — O usuário acessa o menu principal após errar na tela de login.

4. RESULTADO

Resultado: RURALNECT

Requisitos: A lista de requisitos está descrita na Table 1, presente na subseção 3.1. Nesta seção serão apresentados todos os requisitos com imagens do seu funcionamento e trechos de código com suas respectivas explicações.

4.1. 1ª RELEASE

4.1.1. Menu Inicial

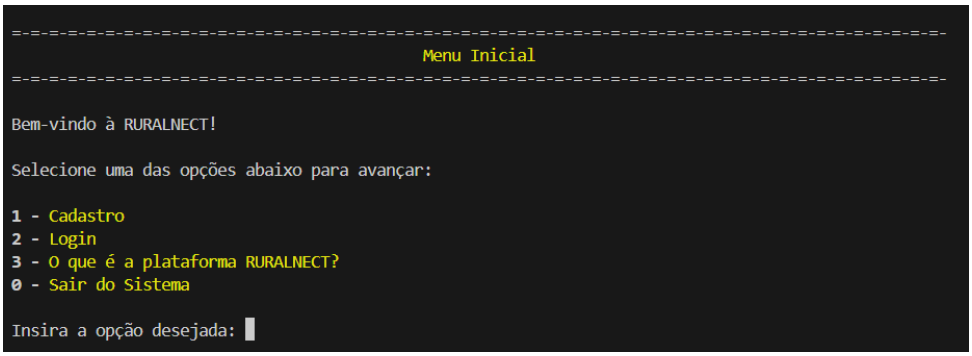

Figure 14. Tela de exibição do Menu Inicial.

Nesta funcionalidade inicial, a primeira tela apresentada ao usuário é o Menu Inicial, que disponibiliza um conjunto limitado de opções para quem ainda não realizou o login. Nesse estágio, o usuário pode navegar entre quatro alternativas: realizar o cadastro, efetuar o login, acessar a área destinada à apresentação do propósito da aplicação ou encerrar o sistema.

```python
def menu_inicial():
    while True:
        Util.limpar_tela()
        Util.cabecalho('Menu Inicial')
        print('\nBem-vindo à RURALNECT!')
        print('\nSelecione uma das opções abaixo para avançar:\n')
        Util.txt_opcao('1', 'Cadastro')
        Util.txt_opcao('2', 'Login')
        Util.txt_opcao('3', 'O que é a plataforma RURALNECT?')
        Util.txt_opcao('0', 'Sair do Sistema')


        try:
            opcao = int(input('\nInsira a opção desejada: '))
            condicionais_menu(opcao)
        except ValueError:
            Util.erro_txt('O valor inserido não é um número inteiro, tente novamente!')
            Util.pausa(5)
```

Figure 15. Trecho de código do Menu Inicial.

Nesta imagem, apresenta-se como o código do Menu Inicial está formatado. Há também a utilização de um código à parte que está estruturado com Orientação a Objetos. Nas seguintes funcionalidades estarão presentes os métodos ”limpar_tela()” ”cabecalho()” e ”txt_opcao()”, os quais limpam a tela, exibem o cabeçalho e formatam a exibição das opções, respectivamente.

Dando prosseguimento à explicação, o usuário insere o valor corresponde à opção para acessar determinada funcionalidade. (talvez adicionar a parte de erro)

4.1.2. Cadastro

Figure 16. Trecho de código do Menu Inicial.

Após o usuário inserir o valor ”1” para ser redirecionado ao cadastro, ele deve inserir todos os dados solicitados: nome, curso (apenas os disponibilizados), e-mail e senha. Somente após preencher os espaços corretamente que poderá efetuar o Login.

```python
class Auth:
    def __init__(self):
        self.usuarios = []
        self.usuario_logado = {'nome': '', 'curso': '', 'email': ''}
```

Figure 17. Trecho das listas e dicionários que armazenam o usuário.

```python
email = str(input('\nInsira o seu e-mail institucional: ').strip().lower())
if not email.endswith('@ufrpe.br'):
    Util.limpar_tela()
    Util.erro_txt('Adicione um e-mail institucional válido!')
    Util.pausa(5)
    return
for usuario in self.usuarios:
    if email == usuario['email']:
        Util.erro_txt('Este e-mail já está cadastrado!')
        Util.pausa(5)
        return
senha = maskpass.askpass('\nInsira uma senha forte: ').strip()
senha_tam = len(senha)
if senha_tam < 6 or senha_tam > 20:
    Util.erro_txt('A senha não possui a quantidade mínima de 6 caracteres ou excedeu a quantidade máxima de 20!')
    Util.pausa(5)
    return
if not any(chr.isnumeric() for chr in senha):
    Util.erro_txt('Sua senha não possui pelo menos um número')
    Util.pausa(5)
    return
if not any(chr.isupper() for chr in senha):
    Util.erro_txt('Sua senha não possui pelo menos uma letra maiúscula')
    Util.pausa(5)
    return
elif ' ' in senha:
    Util.erro_txt('A sua senha possui espaços, remova-os!')
    Util.pausa(5)
    return
senha_confirm = maskpass.askpass('\nInsira a senha novamente para confirmar: ').strip()
if senha_confirm != senha:
    Util.erro_txt('Erro: A senha inserida não corresponde à adicionada anteriormente!')
```

Figure 18. Trecho do código de cadastro exibindo e-mail e senha.

Nas imagens acima, apresenta-se a classe Auth (authentication) onde as funcionalidades de cadastro e login estão localizadas. O atributo de instância **self.usuarios** armazena os usuários cadastrados durante a execução do sistema e o atributo de instância **self.usuario_logado** armazena os dados relativos ao usuário atual que está logando e as utiliza no Menu Principal.

Além disso, foi adicionada a mascaração da senha que transforma os caracteres em asterisco (*), aumentando a segurança ao impedir que a senha possa ser visualizada por outras pessoas. Para fins informativos, foi utilizado o módulo ”maskpass” para fazer esta ocultação.

4.1.3. Login

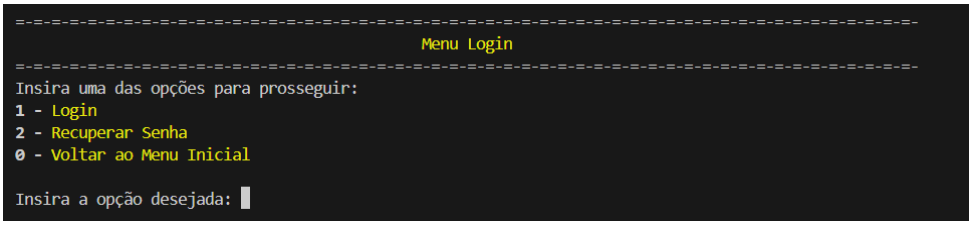

Figure 19. Menu Login funcionando.

Nesta funcionalidade, após o usuário realizar o cadastro e for acessar o sistema, ele deve selecionar uma das opções acima, sendo elas: Login, recuperação de senha e voltar ao menu inicial. Caso o usuário escolha a opção de login, ele deve, durante 3 tentativas de Login, inserir o e-mail e senha corretamente para acessar o menu principal. E, se o usuário selecionar a opção de acessar a recuperação de senha, ele deverá inserir seu e-mail para receber um código, e, após inserir o código corretamente, deverá adicionar uma nova senha.

```python
if opcao_login == 1:

    rest_senha = 0
    cont = 3

    while rest_senha != '1':
        Util.limpar_tela()
        Util.cabecalho('Menu Login')
        if cont < 1:
                Util.txt_aviso('A quantidade de tentativas de login superou o limite de 3 chances, tente novamente mais tarde!')
                Util.pausa(5)
                Util.limpar_tela()
                rest_senha = int(input('Deseja ir para a área de recuperação de senha? (1 - S/0 - N): ').strip().lower())
                if rest_senha == 1:
                    rec.recuperar_senha(self.usuarios, menu_inicial)
                    return
                elif rest_senha == 0:
                    print('Voltando ao menu inicial...')
                    Util.pausa(3)
                    return menu_inicial()
```

Figure 20. Código do login (opção 1).

Neste trecho do código, após o usuário inserir o valor “1” para realizar o login, é feita uma primeira verificação, representada por $cont < 1$, a qual verifica se a quantidade de tentativas de login chegou a zero. Caso o número de tentativas realmente tenha se esgotado, o sistema exibe uma mensagem de aviso e, em seguida, pergunta ao usuário se

ele deseja recuperar a senha. Se a variável **rest_senha** for igual a "1", isto é, se o usuário digitar "1" quando a mensagem de recuperação for exibida, o sistema o redireciona para a área destinada ao processo de recuperação de senha (**rec.recuperar_senha**()).

```
usuario_email = str(input('Insira o e-mail: ').strip().lower())
senha_user = maskpass.askpass('Insira a senha: ').strip()
for usuario in self.usuarios:
    if usuario_email == usuario['email'] and senha_user == usuario['senha']:
        print('Bem-vindo, {}'.format(usuario['nome']))
        self.usuario_logado['nome'] = usuario['nome']
        self.usuario_logado['curso'] = usuario['curso']
        self.usuario_logado['email'] = usuario['email']
        return
    elif usuario_email == usuario['email'] and senha_user != usuario['senha'] or usuario_email != usuario['email'] and
    senha_user == usuario['senha'] or usuario_email != usuario['email'] and senha_user != usuario['senha']:
        Util.erro_txt('Login não-sucedido, tente novamente!')
        cont = cont - 1
        print('chances restantes {}'.format(cont))
```

Figure 21. Código do login (opção 1).

Nesse trecho do código, as variáveis **usuario_email** e **senha_user** recebem, respectivamente, o e-mail e a senha informados pelo usuário para realizar o login. Em seguida, é utilizada uma estrutura de repetição **for**, responsável por percorrer cada dicionário armazenado na lista **self.usuarios**. Dentro do bloco dessa estrutura, é realizada a verificação dos dados inseridos pelo usuário, comparando o e-mail e a senha digitados com aqueles presentes em cada dicionário da lista. Caso ambos os valores coincidam, a variável **self.usuario_logado** é preenchida com três informações do usuário correspondente, permitindo então o redirecionamento para o Menu Principal.

4.1.4. O que é a RURALNECT?

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                                O que é a plataforma RURALNECT?
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
A RURALNECT é uma plataforma criada para integrar e facilitar o acesso às informações da Universidad-

e Federal Rural de Pernambuco (UFRPE), além de promover a colaboração e o aprendizado entre os estud-

antes. Por meio do sistema, é possível encontrar fóruns de perguntas e respostas, videoaulas, listas-

de questões e outros recursos voltados ao desenvolvimento acadêmico e à troca de conhecimento. O pro-

jeto também visa tornar a experiência universitária mais interativa e envolvente, com futuras funcio-

nalidades como gamificação e agendamento de cabines de estudo (sala 33, terceiro andar), incentivand-

o o engajamento da comunidade. Em essência, a RURALNECT busca unir informação e educação, servindo c-

omo um ponto de conexão entre a universidade e seus estudantes.

Pressione ENTER para continuar
```

Figure 22. texto sobre o objetivo da RURALNECT.

Esta funcionalidade cumpre o objetivo de exibir quais são os focos desta aplicação, apontando o público-alvo e ferramentas disponibilizadas.

```python
def info_rural(texto_recortado):
    Util.limpar_tela()
    Util.cabecalho('O que é a plataforma RURALNECT?')
    contador = 0
    texto_junto = ' '.join(texto_recortado)
    for letra in texto_junto:
        print(letra, end='')
        contador = contador + 1
        if contador % 100 == 0:
            print('-\n')
    Util.continuar()
```

Figure 23. Código de formatação do texto.

O código acima assume a função de formatar o texto para que ele fique alinhado com o cabeçalho, além de garantir que o texto não fique tão disperso, dificultando a leitura.

4.1.5. Menu Principal

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                                    Menu Principal
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
usuário: MATHEUS JULIO
curso: Bacharelado em Sistemas de Informação

Selecione uma das opções abaixo para avançar:
1 - Fórum de perguntas e respostas
2 - Lista de questões
3 - Área de videoaulas
4 - Área de informações
5 - Links gerais UFRPE
6 - Área bibliográfica
7 - configurações
0 - Deslogar

Insira a opção desejada:
```

Figure 24. Tela do Menu Principal

Esta funcionalidade, assim como o Menu Inicial, disponibiliza uma quantidade limitada de opções para o usuário. Nesse requisito, o nome e o curso do usuário são exibidos logo abaixo do cabeçalho.

A apresentação do código não se faz necessária, pois sua estrutura é semelhante à utilizada no Menu Inicial.

4.1.6. Área de informações

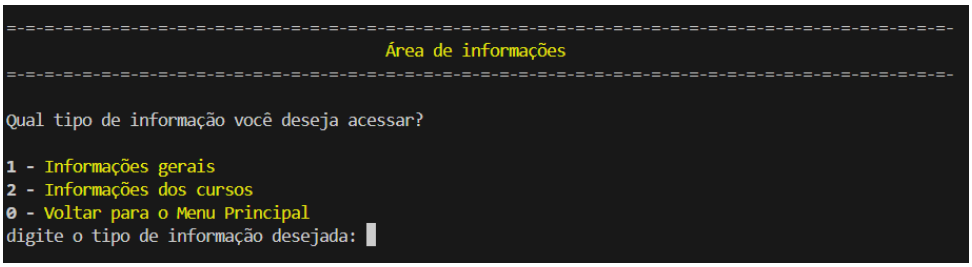

Figure 25. Tela da área de informações.

Este requisito disponibiliza dois tipos de informações: informações gerais da UFRPE e informações específicas sobre os cursos (como ementas, site institucional, disciplinas, entre outros). Caso o usuário selecione a opção referente às informações do curso, o sistema identifica automaticamente o curso associado ao usuário e exibe apenas os dados correspondentes a ele. Vale destacar que, nesta versão do sistema, somente os cursos da área de tecnologia estão disponíveis.

```
elif opcao == 2:
    curso_info = usuario_logado['curso']
    if curso_info == 'Bacharelado em Sistemas de Informação':
        #COLOCAR COMO ESTÁ EM INFORMAÇÕES GERAIS
        print('Opções:')
        Util.txt_opcao('1', 'informações básicas do curso')
        Util.txt_opcao('2', 'matriz curricular')
        Util.txt_opcao('3', 'ementas e programas')
        Util.txt_opcao('4', 'projeto pedagógico')
        Util.txt_opcao('5', 'repositório de disciplinas')
        Util.txt_opcao('6', 'atividades complementares')

        opcao_infocurso = int(input('Insira a opção desejada: '))
        Util.limpar_tela()
        if opcao_infocurso == 1:
            print('Curso: Bacharelado em Sistemas de Informação (BSI) - SEDE RECIFE\n\nDuração Média: 10 semestres (5 anos)\n\nTurno: Manhã
            (presencial - verificar edital)\n\nFoco: Desenvolvimento e Gestão de Sistemas de Informação, Banco de Dados e Redes.\n\nContato
            (Coordenação): coordenacao.bsi@ufrpe.br / (81) 3320-6491')
            Util.continuar()
            Util.limpar_tela()
            return
        elif opcao_infocurso == 2:
            print('redirecionando para a matriz curricular...')
            Util.pausa(3)
            Util.redirecionador('https://sites.google.com/view/bsi-ufrpe/contato/matriz-curricular?authuser=0')
            return
```

Figure 26. Verificação do curso na informações sobre o curso.

Na funcionalidade ilustrada na imagem, após o usuário selecionar a segunda opção, referente às informações do curso, a variável **curso_info** recebe o valor associado à chave **"curso"** presente na variável **usuario_logado**. Em seguida, esse valor é comparado com a string "Bacharelado em Sistemas de Informação". É importante destacar que não há necessidade de criar uma variável intermediária para armazenar o valor do curso, pois a comparação pode ser realizada diretamente utilizando **usuario_logado['curso']**.

Ademais, na imagem há a presença do método **Util.redirecionador()**, o qual redireciona o usuário para um link específico em seu navegador, auxiliando o usuário na utilização do sistema ao permitir que acesse o conteúdo com mais facilidade.

4.1.7. Área de videoaulas

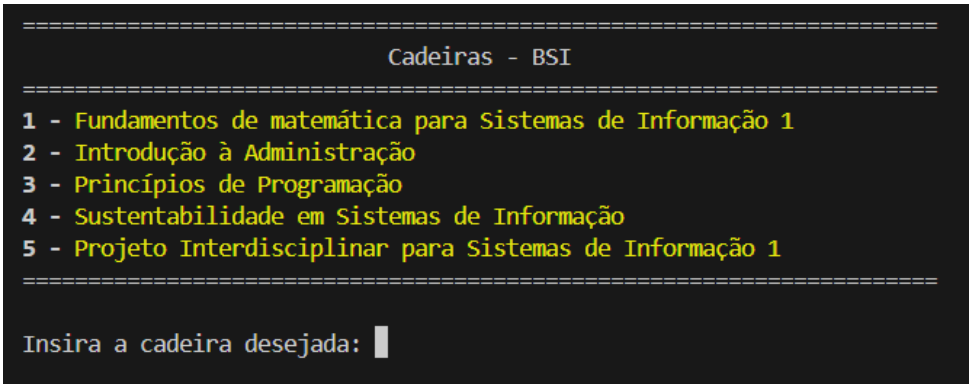

Figure 27. videoaulas de Sistemas de Informação.

No Menu Principal, ao selecionar a área de videoaulas, o usuário é automaticamente redirecionado para o conjunto de videoaulas correspondente ao seu curso. O funcionamento desse requisito é semelhante ao utilizado na área de informações: o sistema identifica o curso associado ao usuário e exibe apenas conteúdos relacionados a ele.

4.2. 2ª RELEASE

4.2.1. Área com conteúdo bibliográfico

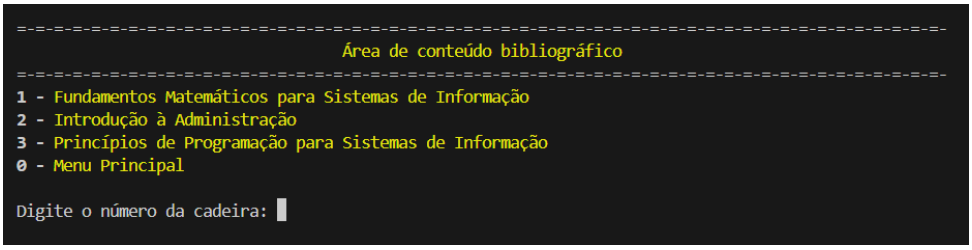

Figure 28. Tela da área de bibliografia de Sistemas de Informação.

Nesta funcionalidade também é feita a leitura do curso do usuário e, em seguida, exibidos as cadeiras que dispõem dos conteúdos bibliográficos, além de redirecionar para o material selecionado.

```
for i, item in enumerate(conteudinho):
    Util.txt_opcao(i + 1, item["cadeira"])
```

Figure 29. enumeração das cadeiras.

```
Util.limpar_tela()
if 1 <= opcao <= len(conteudinho):
    cadeira = conteudinho[opcao - 1]
    Util.cabecalho(cadeira["cadeira"])
    for u, conteudo in enumerate(cadeira['conteudos']):
        print('{}. {}'.format(u + 1, conteudo))

    redirecionador = str(input(Fore.LIGHTYELLOW_EX + '\nDeseja ser redirecionado para alguma conteúdo (s/n)?: ' + Fore.RESET).strip().
    lower())

    if redirecionador == 's':
        num_conteudo = int(input('Insira o número do conteúdo desejado: '))
        Util.redirecionador(cadeira['conteudos'][num_conteudo - 1])

    elif redirecionador == 'n':
        Util.txt_aviso('Voltando para o Menu Principal')
        Util.pausa(3)
        return

    else:
        Util.erro_txt('A opção escolhida é inexistente!')
        Util.pausa(3)
```

Figure 30. enumeração dos conteúdos.

Na Figure 29, foi utilizada a estrutura de repetição **For** que apresenta três variáveis: **i**, **item**, **conteudinho**.

A variável **i**, cumpre a função de enumerar as cadeiras, enquanto a varíavel **item** corresponde individualmente a cada dicionário presente na variável **listinha**, a qual é um nome genérico que irá receber uma lista.

Agora, na Figure 30, quando o usuário selecionar uma opção, será verificado se o valor digitado está entre 1 e o comprimento da lista de dicionários, **len(conteudinho)**. Após isso, uma variável chamada **cadeira** recebe o dicionário do conteúdo selecionado e o processo apresentado inicialmente é repetido na estrutura For.

Por fim, caso o usuário selecione um conteúdo, será questionado se deseja ser redirecionado.

4.2.2. Área de Links Gerais

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
                                           Links Gerais
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Insira uma das opções de links abaixo:

1 - Redes sociais
2 - Sites Oficiais
3 - Sites independentes (feitos por estudantes)
0 - Voltar ao Menu Principal

Insira a opção desejada:
```

Figure 31. Tela de menu da área de links gerais.

Esta funcionalidade se destina a agrupar todos os links de sites oficiais e não-oficiais, redes sociais e ferramentas do aluno. Na imagem, há a disponibilização de três opções: Redes sociais, Sites oficiais e Sites independentes, além da opção de retorno.

Sua estrutura de código é semelhante às apresentadas anteriormente, se diferenciando apenas na presença de três listas distintas que correspondem a cada opção.

4.2.3. Área de lista de exercícios

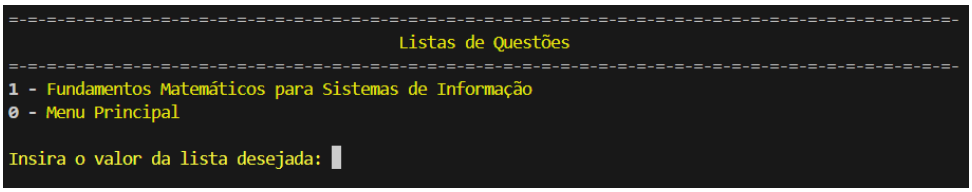

Figure 32. Tela de menu da área de questões.

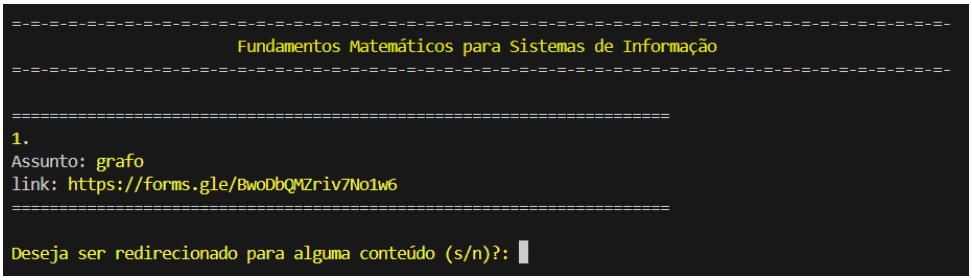

Figure 33. Tela da escolha da questão.

Na Figure 33, quando o usuário acessa o sistema de lista de questões, são exibidas as cadeiras com questões disponíveis, além da opção de retorno.

Agora, na Figure 34, são exibidas as listas de questões disponíveis, as quais são separadas por assunto. De momento, a lista funciona por meio de redirecionamento para um formulário com as questões, mas eventualmente podem ser adicionadas as questões diretamente no código.

Com relação ao código, sua estrutura é semelhante com as áreas desta segunda release.

4.2.4. Fórum

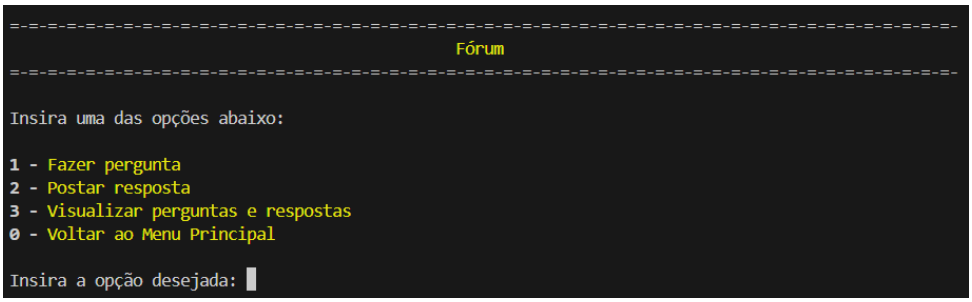

Figure 34. Tela de menu do fórum.

Na imagem acima, o usuário, ao acessar o fórum, tem quatro opções disponíveis: fazer uma pergunta, responder a uma pergunta e visualizar as perguntas e respostas. Vale destacar que se trata de um fórum geral, portanto não há separação por cursos.

Caso o usuário selecione ”Fazer pergunta”, ele deverá inserir um texto contendo entre 10 e 500 caracteres.

Se ele optar por ”Responder”, também precisará inserir uma resposta dentro do mesmo intervalo de 10 a 500 caracteres.

Por fim, caso deseje apenas navegar, o usuário pode visualizar todas as perguntas e respostas cadastradas.

```
Util.limpar_tela()
if opcao == 1:
    Util.cabecalho('Área de perguntas')

    perguntar = str(input('Deseja fazer uma pergunta? (s/n)\n').strip().lower())
    if perguntar == 's':
        Util.limpar_tela()
        Util.cabecalho('Área de perguntas')

        pergunta = str(input(Fore.LIGHTCYAN_EX + 'Pergunta (deve conter, no mínimo, 10 caracteres e, no máximo, 500): ' + Fore.RESET).
        strip().capitalize())

        if not pergunta:
            Util.erro_txt('O campo está vazio, preencha-o!')
            Util.pausa(3)
            return

        if len(pergunta) < 10 or len(pergunta) > 500:
            Util.erro_txt('Sua pergunta ou não atinge a quantidade mínima de caracteres ou ultrapassa o limite!')
            Util.pausa(3)
            return

        if not any(char.isalpha() for char in pergunta):
            Util.erro_txt('Sua pergunta precisa conter caracteres alfabéticos!')
            Util.pausa(3)
            return

        self.perguntas.append({'pergunta': pergunta, 'respostas': []})

        Util.txt_certo('Pergunta cadastrada com sucesso!')
        Util.continuar()
```

Figure 35. código de cadastramento de perguntas.

Na Figure 35, a variável **perguntar** indica se o usuário deseja prosseguir com a ação de realizar uma pergunta. Caso a resposta seja afirmativa, o sistema verifica inicialmente se o campo de entrada está vazio (**if not pergunta**). Em seguida, é analisado o comprimento da *string*, de modo a invalidar perguntas que não alcancem o mínimo de caracteres exigido ou que ultrapassem o limite permitido.

Além disso, por meio de uma estrutura de repetição **for**, o sistema verifica se há caracteres alfabéticos na entrada, invalidando quaisquer perguntas que não contenham esse tipo de caractere.

Por fim, é adicionada uma estrutura à variável **self.perguntas**, responsável por armazenar todas as perguntas cadastradas. Cada elemento inserido nessa lista é composto pela própria pergunta e por outra lista destinada a armazenar suas respectivas respostas.

```
for i, item in enumerate(self.perguntas):
    print(Fore.LIGHTYELLOW_EX + '{}. {}'.format(i + 1, item['pergunta']))
    for resp in item['respostas']:
        if not item['respostas']:
            Util.erro_txt('Esta pergunta ainda não foi respondida!')
        else:
            print('     - {}'.format(resp))
```

Figure 36. código de exibição das perguntas e respostas.

```
self.perguntas[esc_pergunta - 1]['respostas'].append(resposta)
```

Figure 37. código de cadastramento de respostas.

Na Figure 36, foi utilizada a estrutura de repetição **For** que apresenta três variáveis: **i**, **item**, **self.perguntas**.

A variável **i**, cumpre a função de enumerar as cadeiras, enquanto a variável **item** corresponde individualmente a cada dicionário presente na variável **self.perguntas**, lista que abriga as perguntas.

Após a exibição da pergunta, outra estrutura de repetição é utilizada, desta vez para ”imprimir” todas as respostas relacionadas à pergunta, e, caso não tenha nenhuma resposta, será mostrada uma mensagem informando que ”a pergunta ainda não foi respondida”.

Finalizando, na Figure 37, a resposta é adicionada à lista da sua determinada pergunta.

Não se faz necessária a apresentação do código da opção de visualização de perguntas e respostas, pois é a mesma estrutura utilizada na Figure 36.

5. CONCLUSÃO

Em síntese, o objetivo deste projeto é o desenvolvimento de um sistema que possibilite aos estudantes, e, futuramente, também aos docentes, a capacidade de aprimoramento dos seus estudos e o ampliamento das informações compartilhadas na UFRPE, sanando a necessidade da comunidade por informações que se encontram fragmentadas ou que demandam um certo esforço para que possam ser achadas.

A concretização e disponibilização de ferramentas essenciais, como fórum, área de videoaulas, área de informações gerais e áreas de informações específicas de cada curso, as quais foram elaboradas tendo em vista as implementações de sistemas similares, que atendem à necessidade do corpo acadêmico e integram toda a comunidade.

Por fim, destaca-se que os requisitos implementados atendem ao escopo definido e estão funcionando adequadamente, sem apresentar falhas que comprometam o uso do sistema.

6. TRABALHOS FUTUROS

Para este projeto, RURALNECT, ainda podem ser implementadas melhorias que auxiliam os usuários tanto na segurança como usabilidade. Algumas propostas de melhorias são:

1. **Sistema de moderação por docentes ou monitores**: Adicionar moderação no fórum para evitar publicações com palavras de baixo calão ou conteúdos sensíveis.
2. **Lista de questões**: Criar listas de exercícios diretamente no sistema, eliminando a necessidade de acesso via navegador e possibilitando gamificação (XP, níveis, rankings, etc.).
3. **Sistema de avaliação**: Implementar um sistema de avaliação para os conteúdos e para o fórum, seja por meio de notas (1 a 10) ou estrelas.

4. **Fóruns específicos por curso**: Adicionar seções de fórum vinculadas ao curso do usuário, permitindo comunicação direcionada entre integrantes da mesma formação.
5. **Banco de dados**: Implementar um banco de dados para armazenar informações dos usuários e do fórum, garantindo maior organização, persistência e segurança.

7. APÊNDICE

- Repositório do projeto: `https://github.com/MatheusJS12/RURALNECT_projeto.git`
- Desenvolvedor Matheus Júlio: `https://github.com/MatheusJS12`
- Desenvolvedor Cayo Victor: `https://github.com/CayoMenezes`

Figure 38. Logomarca da RURALNECT.

References

[Alves and Cordeiro 2025] Alves, P. A. and Cordeiro, L. (2025). mutare-project: Repositório para MUTARE – PISI1. Acesso em 26 nov. 2025.

[Del Claro 2009] Del Claro, F. (2009). O avanço tecnológico no mundo econômico. *Revista FAE, Vitrine da Conjuntura*, 2(8):1–4.

[Nunes et al. 2023] Nunes, A. P., Pascoal, M. H., de Menezes Souto, M. C. C., Abood, E. M., Pantuza, A. C. M., Cardoso, J. C. P., Gouvea, G. A. T. B., and Vaz, C. S. (2023). O uso de telas e tecnologias pela população infanto-juvenil: revisão bibliográfica sobre o impacto no desenvolvimento global de crianças e adolescentes. *Brazilian Journal of Health Review*, 6(5):19926–19939.

[Programação 2022] Programação, H. (2022). Como enviar e-mail pelo Gmail com Python. Acesso em 20 set. 2025.

[Rodriguez et al. 2014] Rodriguez, M. G., Gummadi, K., and Schoelkopf, B. (2014). Quantifying information overload in social media and its impact on social contagions. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 170–179.

[Saplacan et al. 2020] Saplacan, D., Herstad, J., and Pajalic, Z. (2020). Use of digital learning environments: A study about fragmented information awareness. *Interaction Design and Architecture(s)*, 43:86–109.