Autenticação com Hash

Ary Felipe Farah e Silva

```
# funcoes.py
import hashlib
import time


def title(x):
    print('-' * 30)
    print(f'{x:^30}')
    print('-' * 30)

def strMax(msg, limite=int):
    while True:
        dado = str(input(msg))
        if len(dado) > limite:
            print(f"Limite de caracteres: {limite}.")
        else:
            break

    return dado

def novoUsuario():
    nome = strMax("Nome: ", 40)
    email = strMax("E-mail: ", 80)
    senha = strMax("Senha: ", 4)

    hash_MD5 = criptografar(senha)

    usuario = {'nome': nome,
               'email': email,
```

```
                'senha': hash_MD5
                }

    return usuario

def cadastro(usuario):
    with open('usuarios.txt', 'a') as arquivo:
        arquivo.write(f"{usuario['nome']}, {usuario['email']}
    print("Usuário cadastrado com sucesso!")
    input('<enter para continuar>')

def criptografar(senha):
    hash_MD5 = hashlib.md5()
    hash_MD5.update(senha.encode('utf-8'))
    return hash_MD5.hexdigest()

def login(email, senha):
    with open('usuarios.txt', 'r') as arquivo:
        for linha in arquivo:
            dados = linha.strip().split(', ')
            if dados[1] == email and dados[2] == criptografar
                return True
            else:
                return False
```

```
# cadastro.py

from funcoes import *

while True:
    title("Cadastrando Usuários")
    print("[1] Cadastro\n"
          "[2] Login\n"
          "[0] Sair")

    choice = int(input("O que deseja fazer? "))
    if choice < 0 or choice > 2:
```

```
        print('Digite uma opção válida!')

    elif choice == 0:
        break

    elif choice == 1:
        usuario = novoUsuario()
        cadastro(usuario)

    elif choice == 2:
        erros = 0
        while True:
            email = strMax("E-mail: ", 80)
            senha = strMax("Senha: ", 4)
            if login(email, senha):
                print("Login bem-sucedido!")
                break
            else:
                print("E-mail ou Senha incorretos")
                erros += 1
                if erros >= 3:
                    print("Muitas tentativas incorretas! Agua
                    time.sleep(10)
                    erros = 0
        input('<enter para continuar>')
```

```
# usuarios.txt

Ary, ary@gmail.com, 81dc9bdb52d04dc20036dbd8313ed055
2, 2, c81e728d9d4c2f636f067f89cc14862c
icaro, icaro@gmail.com, a39da23fd6d9da5da2abe407f2e8fa7e
Vinicius, vini@gmail.com, e45823afe1e5120cec11fc4c379a0c67
Mariana, mari@gmail.com, d93591bdf7860e1e4ee2fca799911215
Adriano, ascv@gmail.com, c96dd568316deb9d8c7dec73b4c27cbb
jaozin, joao@gmail.com, e202c4cf8d73876aa1f34a6d3eae7b21
Admin, admin@gmail.com, e6f779072c66ba0fe1ced99998251d36
Novo, novo@gmail.com, 674f3c2c1a8a6f90461e8a66fb5550ba
```

```
Nao Sei, naosei@gmail.com, e2fc714c4727ee9395f324cd2e7f331f
a, a, 0cc175b9c0f1b6a831c399e269772661
```

```
# forcaBruta.py

from funcoes import *

def ataqueBF(arquivo):
    print("-" * 50)
    with open(arquivo, 'r') as file:
        for linha in file:
            dados = linha.strip().split(', ')
            email = dados[1]
            hash = dados[2]
            print(f"Tentando ataque de força bruta para o usu
            with open('senhas.txt', 'r') as senha_file:
                for senha in senha_file:
                    senha = senha.strip()
                    if criptografar(senha) == hash:
                        print(f"Senha encontrada para o usuár
                        break
                else:
                    print(f"Não foi possível encontrar a senh
            print("-" * 50)

ataqueBF('usuarios.txt')
```

```
# senhas.txt

Ary, ary@gmail.com, 81dc9bdb52d04dc20036dbd8313ed055
2, 2, c81e728d9d4c2f636f067f89cc14862c
icaro, icaro@gmail.com, a39da23fd6d9da5da2abe407f2e8fa7e
Vinicius, vini@gmail.com, e45823afe1e5120cec11fc4c379a0c67
Mariana, mari@gmail.com, d93591bdf7860e1e4ee2fca799911215
Adriano, ascv@gmail.com, c96dd568316deb9d8c7dec73b4c27cbb
jaozin, joao@gmail.com, e202c4cf8d73876aa1f34a6d3eae7b21
Admin, admin@gmail.com, e6f779072c66ba0fe1ced99998251d36
```

```
Novo, novo@gmail.com, 674f3c2c1a8a6f90461e8a66fb5550ba
Nao Sei, naosei@gmail.com, e2fc714c4727ee9395f324cd2e7f331f
a, a, 0cc175b9c0f1b6a831c399e269772661
```

Acontecimentos

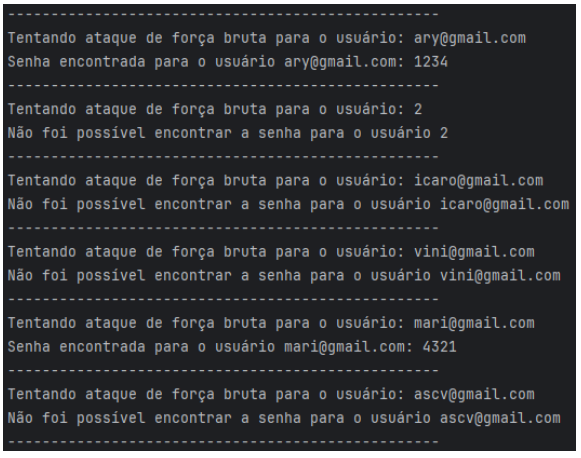
Para desenvolver o código do 'cadastro.py' e as funções de cadastro, eu já tinha o conhecimento prévio, só precisei relembrar alguns conceitos.

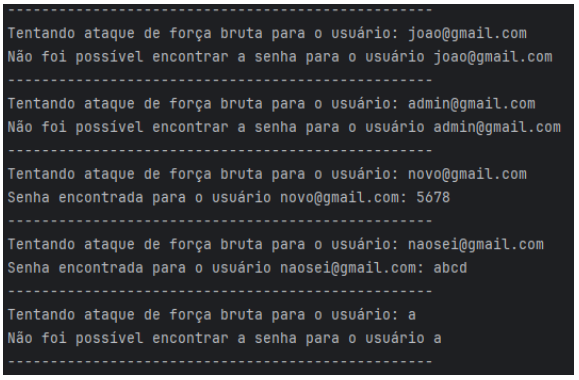
Com relação a criptografia de hash MD5, pesquisei e encontrei no site Awari juntamente com a explicação. Tentei achar no Github, mas não obtive sucesso.

Para fazer a parte do login e do brute force, pedi ajuda ao Chat GPT para concluir os códigos e gerar uma lista com 30 senhas mais utiliazdas de até 4 caracteres.

Brute Force

Com a brute force, o tempo sem a solução foi básicamente instantânea para rodar a lista e identificar se as senhas correspondiam ou não aos e-mails presentes





Solução

A solução que eu encontrei foi de adicionar a função `sleep(10)` , que seria chamada após 3 tentativas incorretas de login e manteria as tentativas bloqueadas durante 10 segundos.

Referências

Awari → https://awari.com.br/aprenda-a-programar-em-python-e-domine-a-arte-do-hash/#:~:text=Em Python%2C o hash é,determinado por sua estrutura interna.

Chat GPT → https://chat.openai.com

GitHub → https://github.com