```python
#Importamos a biblioteca json para salvar dados usando json
import json
#Importamos a biblioteca re para procurar, validar ou substituir texto.
import re
#Importamos a biblioteca unicodedata para trabalhar com texto com caracteres especiais
import unicodedata
#Para verificarmos a idade do usuario
from datetime import datetime
anoAtual = datetime.now().year
arquivos = "usuariosDados.json"
# Classe para fazer as verificações: rg, cpf, senha, estado civil e a idade
class Verificacoes:
    # Função para verificar o RG
    def verificarRg(self, rg):
          rgMinusculo = rg.lower() # Se tiver letra todas vão estar em minusculo para facilitar a verificação
          possuiLetra = re.search(r'[a-z]', rgMinusculo) # Utilizamos a biblioteca re, para verificar se possui letra no input
        if not possuiLetra and len(rg) == 12:
            return True
        else:
            return False
    # Função para verificar o Cpf
    def verificarCpf(self, cpf):
         cpfMinusculo = cpf.lower() #  Se tiver letra todas vão estar em minusculo para facilitar a verificação
         possuiLetra = re.search(r'[a-z]', cpfMinusculo) # Utilizamos a biblioteca re, para verificar se possui letra no input
        if not possuiLetra and len(cpf) == 14:
            return True
        else:
            return False
    # Função para verificar a senha
    def verificarSenha(self, senha):
        # Aqui verificamos se tem número
        temNumero = re.search(r'\d', senha)
        # Aqui verificamos se tem letra maiúscula
        temLetraMaiuscula = re.search(r'[A-Z]', senha)
        # Aqui verificamos se tem letra minúscula
        temLetraMinuscula = re.search(r'[a-z]', senha)
```

```
          # Nesse if se as três variáveis forem verdadeiras, retornará
como verdade, se não como falsa.
        if temNumero and temLetraMinuscula and temLetraMaiuscula:
            return True
        else:
            return False
    # Função para verificar o estado civil
    def verificadorDeEstadoCivil(self, estadoCivil):
        # Deixamos tudo minúsculo
        estadoCivilMinusculo = estadoCivil.lower()
         # Essas duas próximas variáveis servem para deixar o texto sem
acentos.
        string = unicodedata.normalize('NFD', estadoCivilMinusculo)
        stringSemAcento = re.sub(r'[\u0300-\u036f]', '', string)
        # Agora faremos a verificação:
         if stringSemAcento == "solteiro" or stringSemAcento == "casado"
or  stringSemAcento  ==  "divorciado"  or  stringSemAcento  ==  "viuvo"  or
stringSemAcento == "separado judicialmente":
            return True
        else:
            return False
     # Função para verificar se a pessoa é de maior e se colocou um ano
válido
    def verificarIdade(self, anoDeNascimento):
        verificar = anoAtual - anoDeNascimento

        if verificar >= 18 and anoDeNascimento < anoAtual:
            return True
        else:
            return False
# testes unitários:
import unittest
class TesteDeFuncoes(unittest.TestCase):
    def setUp(self):
        self.teste = Verificacoes()
    def testeRg(self):
        self.assertTrue(self.teste.verificarRg("22.222.222-2"))
        self.assertFalse(self.teste.verificarRg("blaafsaf"))
        self.assertFalse(self.teste.verificarRg("333"))
    def testeCpf(self):
        self.assertTrue(self.teste.verificarCpf("423.956.779-09"))
        self.assertFalse(self.teste.verificarCpf("blaafsaf"))
        self.assertFalse(self.teste.verificarCpf("333"))
```

```
    def testeSenha(self):
        self.assertTrue(self.teste.verificarSenha("12Ab"))
        self.assertFalse(self.teste.verificarSenha("blaafsaf"))
        self.assertFalse(self.teste.verificarSenha("1243"))
    def testeEstadoCivil(self):

self.assertTrue(self.teste.verificadorDeEstadoCivil("solteiro"))

self.assertFalse(self.teste.verificadorDeEstadoCivil("blaafsaf"))
        self.assertTrue(self.teste.verificadorDeEstadoCivil("CASADO"))
    def testeIdade(self):
        self.assertFalse(self.teste.verificarIdade(anoAtual - 17))
        self.assertTrue(self.teste.verificarIdade(anoAtual - 20))
        self.assertFalse(self.teste.verificarIdade(anoAtual + 1))
#Para fazer os teste digite no terminal: python -m unittest
GestaDeDadosPessoais.py -v
# Funções para pegar, buscar, salvar, atualizar e editar os dados
#Pegar
class AlteracoesNosDados:
    def pegar_dados(self):
        #utilizamos try e expept caso ocorra erros
        try:
            with open(arquivos, "r", encoding="utf-8") as f:
                    # json.load pega arquivos em json e converte para
python
                dados = json.load(f)
                   # Para fazer essa verificação se o arquivo não esta
sendo como uma lista, utilizamos a função isinstance, que já é do
python.
                if not isinstance(dados, list):
                    return []
                return dados
            # usamos expect para identificar o erro caso ele ocorra.
        except FileNotFoundError:
            # Caso não tenha um arquivo, vamos criar um
            with open(arquivos, "w", encoding="utf-8") as f:
                #Aqui cria uma lista vazia
                json.dump([], f)
            return []
        except Exception as e:
            print(f"Erro ao ler arquivo: {e}")
            return []
    # Função para verificar se o usuário existe
```

```
    def buscar_dados(self, dados, email):
        dados = self.pegar_dados()
        for usuario in dados:
            if usuario.get("email") == email:
                return usuario
        return None
    # Salvar
    def salvar_dados(self, usuario):
        dados = self.pegar_dados()
        # coloca o usuário na variável de dados.
        dados.append(usuario)
        # salva na variável arquivos
        with open(arquivos, "w", encoding="utf-8") as f:
               # json.dump pega arquivos em python e converte em json, usamos ensure_ascii=False para os dados serem salvos como estão, indent=4 formata o json com indentação de 4 espaços
            json.dump(dados, f, ensure_ascii=False, indent=4)
    # Atualizar
              def    atualizar_dados(self,    antigoEmail,    novoEmail, novaNacionalidade, novoEstadoCivil):
        #pegamos  os dados
        dados = self.pegar_dados()

        #Agora vamos atualizar os dados
        for i, user in enumerate(dados):
            if user.get("email") == antigoEmail:
                dados[i].update({
            "email": novoEmail,
            "nacionalidade": novaNacionalidade,
            "estado civil": novoEstadoCivil
        })
                break
        #Vamos salvar eles no arquivo json
        with open(arquivos, "w", encoding="utf-8") as f:
            json.dump(dados, f, ensure_ascii=False, indent=4)
    # Editar
    def editar_dados(self, email):
        #Pegamos os dados que vão ser editados
        antigoEmail = email
        novoEmail = input("Novo e-mail: ")
        novaNacionalidade = input("Nova nacionalidade: ")
          novoEstadoCivil = input("Novo estado civil (solteiro, casado, divorciado, viúvo ou separado judicialmente): ")
```

```
        verificador = Verificacoes()
        # Verificação do novo estado civil
        if not verificador.verificadorDeEstadoCivil(novoEstadoCivil):
            print("Estado civil incorreto.")
            return
        #Vamos atualizar os dados, usando a função atualizar
         self.atualizar_dados(antigoEmail, novoEmail, novaNacionalidade,
novoEstadoCivil)
# Função da entrada de dados
def entrar():
    print("Login: ")
    nome = input("Nome de usuário: ")
    email = input("Email: ")
    senha = input("Senha: ")
    alteracoes = AlteracoesNosDados()
    dados = alteracoes.pegar_dados()
      #  para evitar erros, aqui verificamos se os dados são uma lista,
se forem vai causar erros no json.
    if not isinstance(dados, list):
        print("Erro: Dados corrompidos. Por favor, recadastre-se.")
        return
    usuario = alteracoes.buscar_dados(dados, email)
    # mensagem caso o usuário não seja encontrado
    if usuario is None:
        print("Usuário não encontrado.")
        return
     # para evitar erros, aqui verificamos se o usuário esta salvo como
um dicionário
    if not isinstance(usuario, dict):
        print("Erro: Dados do usuário corrompidos.")
        return
    if nome == usuario["nome"] and senha == usuario["senha"]:
        print("Dados do usuário:")
        print(f"Nome: {usuario['nome']}")
        print(f"Email: {usuario['email']}")
        print(f"Rg: {usuario['rg']}")
        print(f"Cpf: {usuario['cpf']}")
        print(f"Nacionalidade: {usuario['nacionalidade']}")
        print(f"Estado civil: {usuario['estado civil']}")
        res = input("Você deseja editar os seus dados (s/n): ").lower()

        if res == "s":
            alteracoes.editar_dados(email)
```

```
    else:
        print("Dados não encontrados.")
# Função do cadastro de dados
def cadastrar():
    print("Cadastro: ")
    print("-------------------------------------")
    nome = input("Nome de usuário: ")
    email = input("Email: ")
     senha = input("Senha (precisa ter uma letra maiúscula, uma letra minúscula e um número pelo menos): ")
    confirmaSenha = input("Confirmar senha: ")
    rg = input("Rg (exemplo: 22.222.222-2): ")
    cpf = input("Cpf: (exemplo: 222.222.222-22): ")
    nacionalidade = input("Nacionalidade: ")
     estadoCivil = input("Estado civil (solteiro, casado, divorciado, viúvo ou separado judicialmente): ")
    anoDeNascimento = int(input("Seu ano de nascimento: "))
    alteracoes = AlteracoesNosDados()
    verificador = Verificacoes()
    # Verificação de senha, cpf e rg
    if not verificador.verificarSenha(senha):
        print("senha não é forte o suficiente")
        return
    if not verificador.verificarCpf(cpf) :
        print("CPF escrito errado")
        return
    if not verificador.verificarRg(rg) :
        print("RG escrito errado")
        return
    if not verificador.verificarIdade(anoDeNascimento):
         print("Verifique o ano de nascimento, pode estar incorreto ou ser menor de idade.")
        return
    # Verificação se as senhas são iguais
    if senha != confirmaSenha:
        print("Senhas diferentes.")
        return
    if not verificador.verificadorDeEstadoCivil(estadoCivil):
        print("Estado civil incorreto.")
        return
    usuario = {
        "nome": nome,
        "email": email,
```

```
        "senha": senha,
        "rg": rg,
        "cpf": cpf,
        "nacionalidade": nacionalidade,
        "estado civil": estadoCivil,
        "Data de nascimento": anoDeNascimento
    }
    alteracoes.salvar_dados(usuario)
    print("Cadastrado.")
print("Plataforma de Gestão de Dados Pessoais")
print("--------------------------------------")
while True:
     resposta = input("Digite 1, se deseja entrar em sua conta. Digite
2, se deseja se cadastrar. Digite 3, se deseja sair: ")
    if resposta == "1":
        entrar()
    elif resposta == "2":
        cadastrar()
    elif resposta == "3":
        break
```