Exercício Prático: Cadastro de Usuários e Validações

Parte I: Implementação da Classe `CadastroUsuario`

Neste exercício, você deverá implementar uma classe chamada `**CadastroUsuario**` que será responsável por armazenar e gerenciar as informações de um usuário, realizando validações e persistência de dados em um arquivo.

Requisitos da Classe `CadastroUsuario`:
1. **Atributos**:
A classe deverá ter um construtor (`__init__`) que receba os seguintes parâmetros ao ser instanciada:
- `nome`: Nome completo do usuário (string).
- `cpf`: Cadastro de Pessoa Física (string, no formato "XXX.XXX.XXX-XX").
- `nascimento`: Data de nascimento do usuário (string, no formato "DD/MM/AAAA").
- `endereco`: Endereço residencial do usuário (string).
- `telefone`: Número de telefone (string, no formato "(XX) XXXX-XXXX" ou "(XX) XXXXX-XXXX").
- `senha`: Senha de acesso do usuário (string).

2. Métodos:
- `mostrar_info()`: Imprime todas as informações do usuário (nome, CPF, data de nascimento, endereço, telefone), exceto a senha.
- `gravar_info()`: Grava as informações do usuário em um arquivo de texto. O nome do arquivo deve ser `cadastro_<cpf>.txt`, e deve conter todos os dados do usuário (exceto a senha). Antes de gravar as informações, o método deve validar que todos os atributos estão corretos, lançando exceções em caso de erro.

- Métodos de Validação:
- `validar_nome()`: Verifica se o nome é válido (não vazio e possui pelo menos duas palavras).
- `validar_cpf()`: Verifica se o CPF está no formato correto "XXX.XXX.XXX-XX" e é válido (basta garantir que o formato esteja correto, sem validar os dígitos verificadores).
- `validar_nascimento()`: Verifica se a data de nascimento está no formato "DD/MM/AAAA" e se a data faz sentido (por exemplo, uma data futura seria inválida).
- `validar_endereco()`: Verifica se o endereço não está vazio.
- `validar_telefone()`: Verifica se o telefone está no formato correto, podendo aceitar tanto telefones fixos quanto celulares.
- `validar_senha()`: Verifica se a senha tem pelo menos 8 caracteres, incluindo números e letras.
- `validar_gravacao()`: Valida todos os atributos antes de gravar os dados no arquivo, utilizando os métodos de validação mencionados acima. Se houver um erro em qualquer validação, deve-se lançar uma exceção ou exibir uma mensagem de erro.

Tarefas:
- Implemente a classe `**CadastroUsuario**` com o construtor e todos os métodos descritos.
- Garanta que as informações inseridas pelo usuário sejam devidamente validadas antes de serem exibidas ou gravadas.
- Teste a gravação das informações em um arquivo de texto com o nome adequado.

Parte II: Testes com `pytest`

Agora que você implementou a classe `CadastroUsuario`, é hora de garantir que tudo funciona corretamente, escrevendo testes automatizados utilizando o framework `pytest`.

Requisitos do Arquivo `test_cadastro_usuarios.py`:
- Escreva um arquivo de teste chamado `test_cadastro_usuarios.py` que teste todos os métodos da classe `CadastroUsuario`, incluindo os métodos de validação.

Métodos de Teste:

1. Teste de Instanciação:
   - Verifique se a classe é instanciada corretamente com todos os atributos fornecidos.

2. Teste de Validação:
   - Teste a validação de cada atributo individualmente:
     - Verifique se o nome é válido ou inválido.
     - Teste CPFs no formato correto e incorreto.
     - Teste datas de nascimento válidas e inválidas.
     - Teste números de telefone nos formatos aceitos e inválidos.
     - Verifique senhas com e sem os requisitos mínimos.

3. Teste de Exibição:
   - Verifique se o método `mostrar_info()` exibe corretamente os dados do usuário.

4. Teste de Gravação:
   - Verifique se o método `gravar_info()` cria o arquivo corretamente e grava as informações no formato esperado.
   - Teste o comportamento em caso de falha na validação.

Tarefas:
- Escreva todos os testes necessários no arquivo `test_cadastro_usuarios.py`.
- Execute seus testes utilizando o `pytest` e garanta que todos passem sem erros.

Entrega:
- Parte I: Envie o arquivo `**cadastro_usuarios.py**` com a implementação da classe `**CadastroUsuario**`.
- Parte II: Envie o arquivo `**test_cadastro_usuarios.py**` com os testes para os métodos da classe `**CadastroUsuario**`.