Introdução

Empresa Modelo

A empresa fictícia que estamos modelando é uma loja online chamada "Tech Store". A Tech Store vende produtos eletrônicos, como smartphones, laptops, tablets e acessórios. A empresa possui diferentes tipos de usuários que interagem com o sistema, incluindo gerentes, funcionários, estagiários e clientes.

Tipos de Usuário

- **Gerente**: Tem todas as permissões, incluindo gerenciamento de usuários e produtos/serviços.
- **Funcionário**: Pode gerenciar produtos/serviços, mas não tem acesso ao gerenciamento de usuários.
- **Estagiário**: Tem permissões limitadas para visualizar produtos/serviços.
- **Cliente**: Pode visualizar produtos/serviços, mas não tem permissões de gerenciamento.

Produtos e Serviços

A Tech Store oferece uma variedade de produtos eletrônicos, cada um com atributos como nome, preço e quantidade em estoque.

Implementação

Usuários

Estrutura de Dados

Os usuários são armazenados em uma lista de listas, onde cada sublista contém informações sobre um usuário: nome de usuário, senha e papel.

```
usuarios = [
    ["admin", "admin123", "gerente"],
    ["funcionario", "func123", "funcionario"]
]
```

Estrutura do Arquivo

Os usuários são registrados em um arquivo CSV chamado `usuarios.csv`. Cada linha do arquivo contém o nome de usuário, senha e papel.

Funcionalidades CRUD

Create (Criar): Adiciona um novo usuário à lista e salva no arquivo CSV.

```
def criar_usuario(username, password, role):
    usuarios.append([username, password, role])
    salvar_arquivo('usuarios.csv', usuarios)
```

-

Read (Ler): Lê todos os usuários da lista.

```
def ler_usuarios():
    return usuarios
```

Update (Atualizar): Atualiza as informações de um usuário existente.

```
def atualizar_usuario(username, new_password, new_role):
    for user in usuarios:
        if user[0] == username:
            user[1] = new_password
            user[2] = new_role
    salvar_arquivo('usuarios.csv', usuarios)
```

Delete (Deletar): Remove um usuário da lista.

```
def deletar_usuario(username):
    global usuarios
    usuarios = [user for user in usuarios if user[0] != username]
    salvar_arquivo('usuarios.csv', usuarios)
```

Produtos/Serviços

Estrutura de Dados

Os produtos/serviços são armazenados em uma lista de listas, onde cada sublista contém informações sobre um produto: nome, preço e quantidade.

```
produtos_servicos = [
    ["Produto1", 10.0, 100],
    ["Produto2", 20.0, 200]
]
```

Estrutura do Arquivo

Os produtos/serviços são registrados em um arquivo CSV chamado `produtos_servicos.csv`. Cada linha do arquivo contém o nome, preço e quantidade de um produto.

Exemplo de `produtos_servicos.csv`:

```
Produto1,10.0,100
Produto2,20.0,200
```

Funcionalidades CRUD

Create (Criar): Adiciona um novo produto/serviço à lista e salva no arquivo CSV.

```
def criar_produto(nome, preco, quantidade):
    produtos_servicos.append([nome, preco, quantidade])
    salvar_arquivo('produtos_servicos.csv', produtos_servicos)
```

Read (Ler): Lê todos os produtos/serviços da lista.

```
def ler_produtos():
    return produtos_servicos
```

Update (Atualizar): Atualiza as informações de um produto/serviço existente.

```
def atualizar_produto(nome, novo_preco, nova_quantidade):
    for produto in produtos_servicos:
        if produto[0] == nome:
            produto[1] = novo_preco
            produto[2] = nova_quantidade
    salvar_arquivo('produtos_servicos.csv', produtos_servicos)
```

Delete (Deletar): Remove um produto/serviço da lista.

```
def deletar_produto(nome):
    global produtos_servicos
    produtos_servicos = [produto for produto in produtos_servicos if
produto[0] != nome]
    salvar_arquivo('produtos_servicos.csv', produtos_servicos)
```

Buscar Produto/Serviço: Busca um produto/serviço específico pelo nome.

```
def buscar_produto(nome):
    for produto in produtos_servicos:
        if produto[0] == nome:
            return produto
    return None
```

-

Listar Produtos/Serviços por Nome: Lista todos os produtos/serviços ordenados por nome.

```
def listar_produtos_ordenados_por_nome():
    return sorted(produtos_servicos, key=lambda x: x[0])
```

-

Listar Produtos/Serviços por Preço: Lista todos os produtos/serviços ordenados por preço.

```
def listar_produtos_ordenados_por_preco():
    return sorted(produtos_servicos, key=lambda x: x[1])
```

-

Controle de Acesso

Função de Login

A função de login valida as credenciais do usuário e retorna o papel do usuário, se as credenciais forem válidas.

```
def login(username, password):
    for user in usuarios:
        if user[0] == username and user[1] == password:
            return user[2]
    return None
```

Função Principal

A função principal gerencia o fluxo do programa, exibindo menus e chamando as funções apropriadas com base nas permissões do usuário.

```
def main():
    print("Bem-vindo ao sistema de gerenciamento de empresa!")
    username = input("Nome de usuário: ")
```

```
    password = input("Senha: ")

    role = login(username, password)
    if not role:
        print("Login falhou!")
        return

    print(f"Bem-vindo, {username}! Seu papel é: {role}")

    while True:
        print("\nMenu:")
        if role == 'gerente':
            print("1. Criar usuário")
            print("2. Ler usuários")
            print("3. Atualizar usuário")
            print("4. Deletar usuário")
        print("5. Criar produto/serviço")
        print("6. Ler produtos/serviços")
        print("7. Atualizar produto/serviço")
        print("8. Deletar produto/serviço")
        print("9. Buscar produto/serviço")
        print("10. Listar produtos/serviços por nome")
        print("11. Listar produtos/serviços por preço")
        print("0. Sair")

        opcao = int(input("Escolha uma opção: "))

        if opcao == 0:
            break
        elif opcao == 1 and role == 'gerente':
            username = input("Nome de usuário: ")
            password = input("Senha: ")
            role = input("Papel
(gerente/funcionario/estagiario/cliente): ")
            criar_usuario(username, password, role)
        elif opcao == 2 and role == 'gerente':
            for user in ler_usuarios():
                print(user)
        elif opcao == 3 and role == 'gerente':
            username = input("Nome de usuário: ")
            new_password = input("Nova senha: ")
```

```
            new_role = input("Novo papel: ")
            atualizar_usuario(username, new_password, new_role)
        elif opcao == 4 and role == 'gerente':
            username = input("Nome de usuário: ")
            deletar_usuario(username)
        elif opcao == 5:
            nome = input("Nome do produto/serviço: ")
            preco = float(input("Preço: "))
            quantidade = int(input("Quantidade: "))
            criar_produto(nome, preco, quantidade)
        elif opcao == 6:
            for produto in ler_produtos():
                print(produto)
        elif opcao == 7:
            nome = input("Nome do produto/serviço: ")
            novo_preco = float(input("Novo preço: "))
            nova_quantidade = int(input("Nova quantidade: "))
            atualizar_produto(nome, novo_preco, nova_quantidade)
        elif opcao == 8:
            nome = input("Nome do produto/serviço: ")
            deletar_produto(nome)
        elif opcao == 9:
            nome = input("Nome do produto/serviço: ")
            produto = buscar_produto(nome)
            if produto:
                print(produto)
            else:
                print("Produto/serviço não encontrado!")
        elif opcao == 10:
            for produto in listar_produtos_ordenados_por_nome():
                print(produto)
        elif opcao == 11:
            for produto em listar_produtos_ordenados_por_preco():
                print(produto)
        else:
            print("Opção inválida!")

if __name__ == "__main__":
    main()
```

Conclusão

Dificuldades Encontradas

Uma das principais dificuldades encontradas foi garantir que o sistema de permissões funcionasse corretamente, permitindo que apenas usuários autorizados realizassem certas operações. Além disso, garantir a integridade dos dados ao ler e escrever nos arquivos CSV foi um desafio, especialmente ao lidar com diferentes tipos de dados (strings, floats, inteiros).

Escolhas Bem-Sucedidas

A escolha de usar arquivos CSV para armazenamento de dados facilitou o processo de leitura e escrita dos registros, mantendo o sistema simples e eficiente. A estrutura modular do código, com funções específicas para cada operação CRUD, facilitou a manutenção e a extensibilidade do sistema.

O Que Faltou Fazer

O sistema poderia ser aprimorado com a adição de uma interface gráfica para melhorar a experiência do usuário. Além disso, a segurança das senhas poderia ser melhorada utilizando técnicas de hash. Outra melhoria seria a implementação de testes automatizados para garantir a qualidade do código e evitar erros futuros.

O Que Faria Diferente

Se tivesse mais tempo, eu implementaria um sistema de gerenciamento de sessões para melhorar a segurança. Adicionaria também recursos de auditoria para registrar ações dos usuários, ajudando a monitorar o uso do sistema e identificar possíveis problemas de segurança.