CURSO EBAC - TI DO ZERO

Registro de Nomes
Documento de Arquitetura de Software

Versão <1.0>

```
--> Cartório da EBAC <--

Escolha a opção desejada no menu a seguir!

        Seção: Usuarios

        1 > Registrar
        2 > Consultar
        3 > Deletar
        4 > Sair

Digite uma opção:
```

Prod. juiceluqi

1. **Introdução**

Objetivo:

Este documento descreve a arquitetura de um sistema de cartório digital, com funcionalidades de registro, consulta e exclusão de usuários. O sistema interage com arquivos de texto para armazenar dados e mantém os 5 últimos registros.

Escopo:

O sistema permite que um operador registre, consulte ou delete informações de usuários, baseando-se nos dados fornecidos pelo CPF. Ele armazena as informações de forma simples em arquivos e mantém um arquivo de registros recentes. Não envolve banco de dados, sendo puramente baseado em arquivos de texto.

Público-alvo:

- Desenvolvedores do sistema
- Administradores do sistema
- Equipe de suporte técnico

---

2. **Visão Geral do Sistema**

O sistema é composto pelas seguintes funcionalidades principais:

- Registro de Usuários: Permite o cadastro de um novo usuário com CPF, nome, sobrenome e cargo.
- Consulta de Usuários: Permite consultar informações de um usuário com base no CPF. Também exibe os últimos 5 registros e oferece a opção de cadastrar um novo usuário caso não seja encontrado.
- Deleção de Usuários: Permite excluir um usuário com base no CPF fornecido, removendo também o registro recente associado a ele.

Os dados dos usuários são armazenados em dois arquivos de texto:

- *registrados.txt*: Armazena o CPF, nome, sobrenome e cargo de todos os usuários registrados.
- *recentes.txt*: Armazena os 5 últimos registros de usuários.

3. **Componentes Principais**

3.1. **Funções**

- Função *confirmaAcao*: Solicita confirmação do usuário para prosseguir com ações como registro, consulta e deleção.
- Função *registro*: Registra um novo usuário no arquivo *registrados.txt* e atualiza os 5 registros mais recentes no arquivo *recentes.txt*.
- Função *consulta*: Permite consultar um usuário pelo CPF, exibindo informações do usuário e dos 5 registros mais recentes.
- Função *confirmaDeletacao*: Solicita confirmação para excluir um usuário do sistema.
- Função *deletar*: Remove um usuário do arquivo registrados.txt e atualiza o arquivo *recentes.txt*.

3.2. **Arquivos de Dados**

- *registrados.txt*: Armazena os dados completos de todos os usuários registrados.
- *recentes.txt*: Armazena os 5 registros mais recentes, de acordo com o CPF inserido.

---

4. **Diagramas de Arquitetura**

O sistema tem uma arquitetura simples baseada em leitura e escrita de arquivos. Não possui interações complexas entre múltiplos componentes, sendo um modelo sequencial.

- **Diagrama de Fluxo de Dados**:

  1. O usuário interage com o sistema através de um menu.
  2. Dependendo da opção escolhida (Registrar, Consultar, Deletar), o sistema solicita informações e executa ações de leitura e gravação nos arquivos.
  3. Após a execução da ação, o sistema retorna ao menu principal.

---

5. **Tecnologias e Ferramentas**

- Linguagem de Programação: **C**
- Bibliotecas Usadas:
  - *stdio.h*: Comunicação com o usuário.
  - *stdlib.h*: Funções de alocação e controle de sistema.
  - *locale.h*: Suporte a caracteres especiais e regionalização.
  - *string.h*: Manipulação de **strings**.
  - *wchar.h*: Manipulação de caracteres wide (não utilizado diretamente no código).
- Armazenamento de Dados: *Arquivos de texto simples* (.txt).

6. **Decisões Arquiteturais**

- **Uso de Arquivos de Texto**: Optamos por utilizar arquivos de texto simples para armazenamento de dados, mantendo o sistema simples e com baixo overhead. Isso limita a escalabilidade do sistema, mas é adequado para o escopo atual do projeto.

- **Armazenamento de Registros Recentes**: O sistema mantém os 5 últimos registros, proporcionando uma forma rápida de acessar os dados mais recentes.

---

7. **Riscos e Desafios**

- **Risco de Perda de Dado**s: Como o sistema utiliza arquivos de texto simples, há o risco de perda de dados em caso de falha de escrita ou corrupção dos arquivos. Para mitigar, seria interessante introduzir backups regulares dos arquivos.

- **Escalabilidade**: O uso de arquivos de texto não é ideal para um grande número de registros, o que pode tornar o sistema lento à medida que cresce. Para escalabilidade futura, seria importante migrar para um banco de dados.

---

8. **Plano de Implementação**

- **Desenvolvimento**: O sistema será desenvolvido inicialmente como uma aplicação de linha de comando simples, com interações diretas com o usuário. O código será estruturado em funções modulares para cada operação (registro, consulta, deleção).

- **Testes**: Serão realizados testes manuais para validar as funcionalidades, como o registro de dados, consulta de informações e exclusão de usuários.

---

9. **Manutenção e Evolução**

- **Manutenção**: O sistema deverá ser monitorado para garantir que os arquivos de dados não se corrompam. A inclusão de logs e mensagens de erro mais detalhadas poderia ajudar na manutenção.

- **Evolução**: Futuras melhorias podem incluir:

    - Implementação de persistência de dados em banco de dados.
    - Melhorias na interface do usuário, como um sistema gráfico (GUI).
    - Funcionalidades adicionais, como atualização de dados de usuários ou exportação de registros.
    - Tratativa de erros e Monitoria de usuários.
    - Backup de Dados e armazenamento em nuvem.