Documento de Apresentação de Arquitetura de Software

Esqueleto da aplicação - Registro de nomes

Curso Ebac – TI do zero ao Pro

Maio - 2023

Bruna Dalmagro

Índice Analítico

1. Introdução

Esse documento tem como finalidade apresentar os requisitos de desenvolvimento para o projeto de registro de nomes na EBAC. O cliente nos apresentou uma dor ao registrar os colaboradores e alunos que fazem parte da empresa.

O Escopo foi definido para a criação de um menu principal com três opções, sendo elas, `Registro`, `Consulta` e `Deleta`. A Função Incluir recebe os dados do usuário e armazena em um banco de dados não relacional. A consulta recebe um valor de referência, definido como CPF e busca no banco de dados as informações, apresentando ao usuário. E a função `Deleta` exclui os dados cadastrados a partir do CPF apresentado pelo usuário. A linguagem escolhida para o desenvolvimento é o C++.

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

1.2 Escopo

O Documento de Arquitetura de Software é um guia essencial para o desenvolvimento e manutenção de sistemas de software. Ele descreve a estrutura, componentes e comportamentos do sistema, fornecendo uma referência clara para os desenvolvedores e equipes de projeto.

1.3 Definições, Acrônimos e Abreviações

Nesta seção é possível encontrar informações sobre a estrutura de código para melhor compreendê-lo e termos apresentados durante a apresentação:

“BD”: Abreviação de "Banco de Dados". Neste contexto, refere-se ao arquivo de texto "BD.txt" onde os registros são armazenados;

“CPF”: A sigla para "Cadastro de Pessoa Física", que é um número único atribuído a cada cidadão no Brasil. No código, o CPF é utilizado como identificador único para os registros;

“FILE”: É uma estrutura de dados usada em C para representar um arquivo. É usada para manipular operações de leitura e gravação em arquivos;

“fopen()”: Uma função da biblioteca <stdio.h> que é usada para abrir um arquivo em modo de leitura, gravação ou apêndice;

“stdio.h”: É um arquivo de cabeçalho da linguagem de programação C. Ele faz parte da biblioteca padrão da linguagem C e contém declarações e definições de funções e constantes relacionadas à entrada e saída de dados.

“fprintf()”: Uma função da biblioteca <stdio.h> que é usada para escrever dados formatados em um arquivo;

“fclose()”: Uma função da biblioteca <stdio.h> que é usada para fechar um arquivo após a conclusão das operações de leitura ou gravação;

“strcpy()": Uma função da biblioteca <string.h> que é usada para copiar uma string de origem para uma string de destino;

“string.h": fornece várias funções para manipular strings, como copiar, concatenar, comparar, buscar caracteres, entre outras operações, faz parte da biblioteca padrão da linguagem C;

“fgets()”: Uma função da biblioteca <stdio.h> que é usada para ler uma linha de um arquivo;

“setlocale()”: Uma função da biblioteca <locale.h> que é usada para definir a localização para a configuração do idioma e formatação;

“remove()”: Uma função da biblioteca <stdio.h> que é usada para excluir um arquivo;

“rename()”: Uma função da biblioteca <stdio.h> que é usada para renomear um arquivo;

“printf()”: Uma função da biblioteca <stdio.h> que é usada para imprimir dados formatados na saída padrão (no console);

“scanf()”: Uma função da biblioteca <stdio.h> que é usada para ler dados de entrada formatados a partir do console;

“system()”: Uma função da biblioteca <stdlib.h> que é usada para executar um comando do sistema operacional;

“console”: É uma interface de linha de comando que permite que o usuário interaja com o sistema operacional ou com um programa.

Ele é usado para exibir informações e receber entrada do usuário. O console pode ser acessado por meio de um terminal ou prompt de comando.

Essas definições ajudarão a entender o código e as operações realizadas no Documento de Arquitetura de Software fornecido.

1.4 Visão Geral

O projeto consiste em um programa que implementa um sistema de cadastro, consulta e exclusão de nomes em um arquivo de banco de dados. Abaixo está a descrição da representação arquitetural do projeto:

Função ***registro()***: É responsável pelo registro de nomes no banco de dados. Ela solicita ao usuário que digite um CPF, nome, sobrenome e cargo. Em seguida copia o valor do CPF, nome, sobrenome e cargo para o arquivo de banco de dados usando a função ***fprintf***. O arquivo é aberto em modo de adição ("a") para adicionar os registros no final. Depois de cada campo, é inserida uma vírgula (",") para separar os valores.

Função ***consulta()***: É responsável pela consulta no banco de dados. Ela solicita ao usuário que digite um CPF para verificar. O arquivo de banco de dados é aberto em modo de leitura ("r"). Em seguida, a função percorre cada linha do arquivo

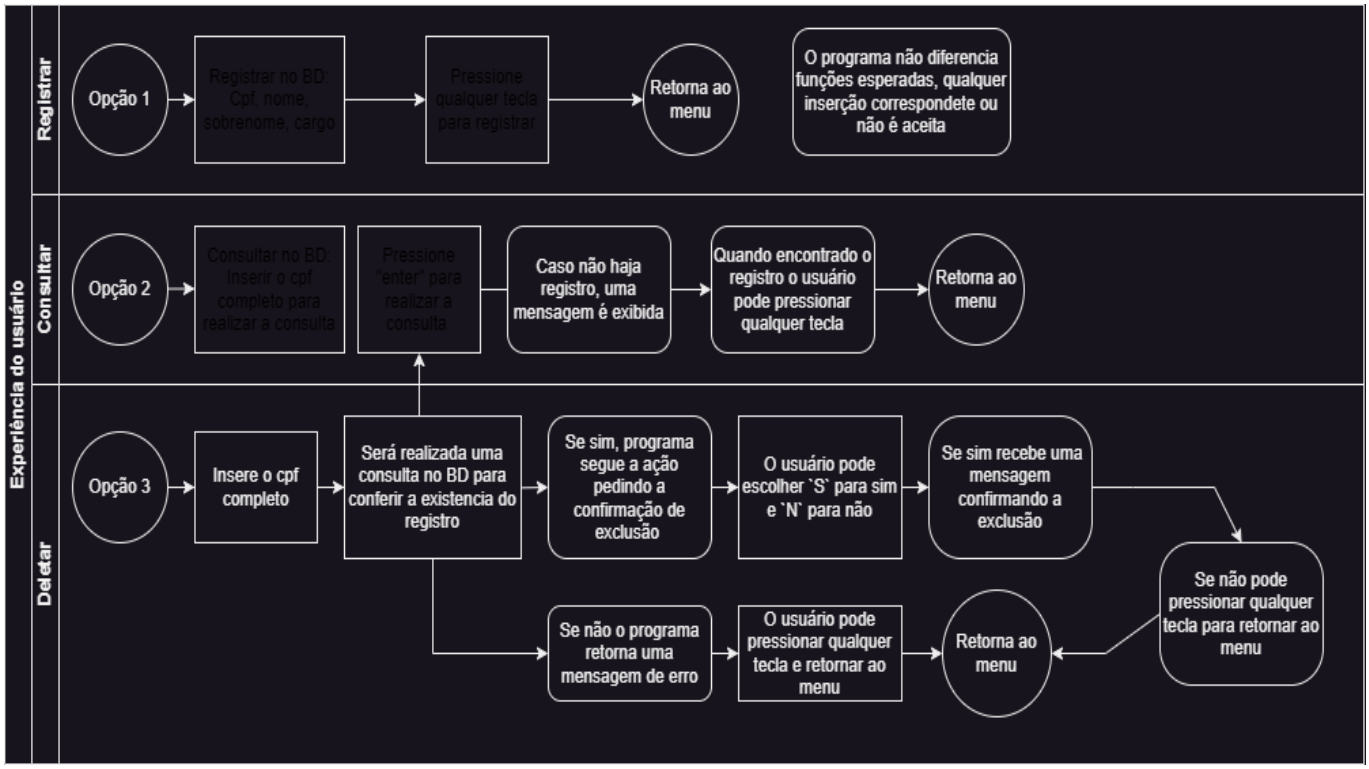
usando a função ***fgets***. Se a linha contiver o CPF digitado, ela imprime as informações do usuário na linha usando a função ***printf***. A função ***setlocale*** é usada para definir a linguagem como português, assim espera-se que o programa se comporte conforme o esperado em caso de acentuações.

Função ***deleta()***: Essa função é responsável pela exclusão de um registro no banco de dados. Ela solicita ao usuário que digite o CPF do registro a ser excluído. O arquivo de banco de dados é aberto em modo de leitura ("r"). A função lê cada linha do arquivo e verifica se o CPF corresponde ao digitado pelo usuário. Se houver correspondência, o registro é excluído do arquivo. A função ***remove*** é usada para remover o arquivo original e a função ***rename*** é usada para renomear um arquivo temporário ("temp.txt") como o arquivo original. Se o usuário cancelar a exclusão, o arquivo temporário é removido.

A função ***main()*** é a principal função do programa. Ela exibe um menu com três opções: registrar nomes, consultar nomes e deletar nomes. O usuário pode escolher uma opção digitando o número correspondente. Com base na escolha do usuário, a função ***switch*** chama a função apropriada: ***registro()***, ***consulta()*** ou ***deleta()***. Após cada opção, o programa pausa e espera que o usuário pressione qualquer tecla antes de prosseguir. O programa continua executando em um loop infinito até que o usuário escolha sair.

2.0 Representação Arquitetural

Representação diagramada da experiência do cliente.



3.0 Metas e Restrições da Arquitetura

Os requisitos e objetivos do sistema incluem funcionalidades básicas de registro, consulta e exclusão de dados armazenados.

Funcionalidade: O sistema deve permitir o registro de informações (CPF, nome, sobrenome, cargo), consulta e exclusão de registros.

Privacidade: O sistema deve cumprir as regulamentações de privacidade e proteção de dados aplicáveis, garantindo que as informações pessoais dos usuários sejam tratadas de forma adequada e respeitando sua privacidade.

Portabilidade: O sistema deve ser desenvolvido de forma a ser executado em diferentes plataformas e ambientes. Isso pode

envolver a escolha de tecnologias multiplataforma e a adesão a padrões abertos.

Distribuição e reutilização: O sistema pode ser distribuído como um produto (quando estiver pronto) para uso em diferentes organizações ou contextos. Além disso, pode-se buscar a reutilização de componentes ou bibliotecas existentes para agilizar o desenvolvimento e melhorar a eficiência.

Restrições especiais

Código-fonte legado: Pode haver código-fonte legado existente que precisa ser integrado ou migrado para o novo sistema. Isso pode impor restrições ao design e à implementação.

4.0 Visão de Casos de Uso

Esses são alguns dos principais casos de uso ou cenários identificados com base no código fornecido. É possível adicionar outros casos de uso dependendo dos requisitos, melhorias e funcionalidades desejadas para o sistema.

Registrar nomes:

- O usuário seleciona a opção de registro de nomes no menu.
- O programa solicita ao usuário que insira o CPF a ser cadastrado.
- O usuário insere o CPF.

- O programa solicita ao usuário que insira o nome a ser cadastrado.
- O usuário insere o nome.
- O programa solicita ao usuário que insira o sobrenome a ser cadastrado.
- O usuário insere o sobrenome.
- O programa solicita ao usuário que insira o cargo desejado.
- O usuário insere o cargo.
- O programa salva as informações no arquivo "BD.txt".
- O programa exibe uma mensagem para continuar.
- O programa retorna ao menu principal.

Consultar nomes:

- O usuário seleciona a opção de consulta de nomes no menu.
- O programa solicita ao usuário que insira o CPF completo para verificar.
- O usuário insere o CPF.
- O programa abre o arquivo "BD.txt".
- O programa percorre o arquivo em busca do registro correspondente ao CPF.
- O programa exibe as informações do usuário (CPF, nome, sobrenome e cargo) se o registro for encontrado.
- O programa exibe uma mensagem de erro se o registro não for encontrado.
- O programa retorna ao menu principal.

Deletar nomes:

- O usuário seleciona a opção de exclusão de nomes no menu.
- O programa solicita ao usuário que insira o CPF do registro a ser excluído.
- O usuário insere o CPF.
- O programa abre o arquivo "BD.txt".
- O programa percorre o arquivo em busca do registro correspondente ao CPF.
- O programa cria um arquivo temporário ("temp.txt") para armazenar os registros não excluídos.
- O programa exclui o registro correspondente ao CPF do arquivo temporário.
- O programa remove o arquivo original ("BD.txt").
- O programa renomeia o arquivo temporário ("temp.txt") para o nome original ("BD.txt").
- O programa exibe uma mensagem de sucesso ou erro.
- O programa retorna ao menu principal.

4.1 Realizações de Casos de Uso

Vamos ilustrar o funcionamento do software apresentando alguns cenários de casos de uso e explicar como os diversos elementos do modelo de design contribuem para a funcionalidade do código fornecido até agora.

Cenário 1: **Registrar nomes**

1. O usuário inicia o programa.
2. O menu principal é exibido.
3. O usuário seleciona a opção de registrar nomes.

4. A função “registro()” é chamada.
5. O programa solicita ao usuário que insira o CPF a ser cadastrado.
6. O usuário insere o CPF.
7. O programa solicita ao usuário que insira o nome a ser cadastrado.
8. O usuário insere um nome composto utilizando dois espaços variável separadas pela função “***scanf*** (“%s, variável”)”.
9. O programa solicita ao usuário que insira o sobrenome a ser cadastrado.
10. O usuário insere o sobrenome que vai automaticamente para a variável “cargo”.
11. As informações são armazenadas no arquivo "***BD.txt***" por meio da manipulação do arquivo com as funções “***fopen, fprintf e fclose***”.
12. O programa exibe uma mensagem de sucesso.
13. O programa retorna ao menu principal.

Nesse cenário, os elementos do modelo de design que contribuem para a funcionalidade são:

"Registrar nomes" que é a funcionalidade selecionada pelo usuário no menu.

Arquivo "**BD.txt**": Armazena os registros dos nomes cadastrados.

Comandos “***fopen, fprintf e fclose”***: São utilizados para abrir o arquivo, escrever as informações e fechar o arquivo após o registro.

Mensagens de feedback: O programa exibe mensagens de sucesso ou erro para informar ao usuário sobre o resultado da operação, mas não é capaz de entender possíveis erros no processo de armazenamento das informações em seus respectivos.

Neste caso não é possível a utilização de nomes compostos, pois automaticamente serão separados pela “,” enviando o segundo nome para a variável sobrenome, impedindo o registro do subsequente “Sobrenome” ou “Cargo”.

5.0 Visão Lógica e Geral

O código fornecido aos usuários é um programa simples de registro, consulta e exclusão de nomes em um arquivo de texto. Não segue uma arquitetura formal de design de software. No entanto, identificam-se algumas partes significativas do código com base em sua estrutura e funcionalidade.

Classes significativas (do ponto de vista da arquitetura):

- N/A - Não possui uma estrutura orientada a objetos e não define classes explicitamente.

Funções significativas:

int registro():

- Responsabilidades: Capturar informações do usuário e escrevê-las no arquivo de texto.
- Relacionamentos: Essa função interage com o arquivo de texto fornecido.

- Operações e atributos de grande importância: As informações inseridas pelo usuário são armazenadas em variáveis locais e escritas no arquivo usando a função "***fprintf()***".

int consulta():

- Responsabilidades: Capturar o CPF do usuário, ler o arquivo de texto, procurar registros correspondentes e exibir as informações encontradas.
- Relacionamentos: Essa função interage com o arquivo de texto fornecido.
- Operações e atributos de grande importância: O CPF fornecido pelo usuário é comparado com os registros lidos do arquivo. O conteúdo do arquivo é exibido usando a função "***printf()"***.

void deleta():

- Responsabilidades: Capturar o CPF do usuário, ler o arquivo de texto, procurar o registro correspondente e excluí-lo do arquivo.
- Relacionamentos: Essa função interage com o arquivo de texto fornecido.
- Operações e atributos de grande importância: O CPF fornecido pelo usuário é comparado com os registros lidos do arquivo. O registro correspondente é removido usando as funções "***remove()*** e ***rename()***".

Observações adicionais:

- **Função main():** que é o ponto de entrada do programa. Ela exibe um menu para o usuário escolher entre as

diferentes opções (registro, consulta, exclusão) e chama as funções correspondentes com base na escolha do usuário.

- Bibliotecas padrão em C (**stdio.h, stdlib.h, locale.h** e **string.h**) para fornecer funcionalidades como entrada/saída, alocação de memória, configuração regional e manipulação de **strings**.

É importante ressaltar que o código fornecido ainda não segue as práticas de design e estruturação recomendadas para um projeto de software de médio e grande porte.

6.0 Visão de Processos

- Processos leves:

Função registro: Responsável por registrar um nome no sistema. Esta função realiza a comunicação com o usuário através da entrada de dados e interage com o arquivo de texto para armazenar as informações.

Função consulta: Realiza a consulta de nomes no sistema. Também interage com o arquivo de texto para ler as informações e exibi-las na tela.

Função deleta: Responsável pela exclusão de nomes do sistema. Assim como as outras funções, interage com o arquivo de texto para realizar a exclusão dos registros.

- Processos pesados:

Não há processos pesados neste código. Todas as funções são relativamente simples e executadas de forma sequencial.

- Grupos de processos que se comunicam ou interagem:

Função “registro”, função “consulta”, função “deleta”.

- Modos principais de comunicação entre processos:

Transmissão de mensagens: As funções registro, consulta e deleta se comunicam com o usuário através da entrada de dados e exibição de informações na tela.

Interações com o arquivo de texto: As funções registro, consulta e deleta interagem com o arquivo de texto ("BD.txt") para armazenar, ler e excluir informações dos registros. Isso é feito através das operações de abertura, escrita e leitura do arquivo utilizando as funções “***fopen, fprintf, fgets*** e ***fclose”***.

Interrupções: Pequenos erros de identificação e armazenamento das entradas.

7.0 Visão de Implantação

Com base no código fornecido, podemos considerar uma configuração básica para implantação e execução do software. Esta configuração envolve um computador físico que executa o software.

Não há necessidade de interconexões adicionais, pois pode ser executado em uma única máquina.

8.0 Visão da Implementação

É importante lembrar que o código fornecido é uma versão simplificada e pode ser expandida e “refatorada” para atender

a requisitos e padrões de arquitetura mais complexos, dependendo das necessidades do sistema.

- Seus componentes significativos são:

Função registro

Função consulta

Função deleta

Função main

Já listados nos capítulos anteriores.

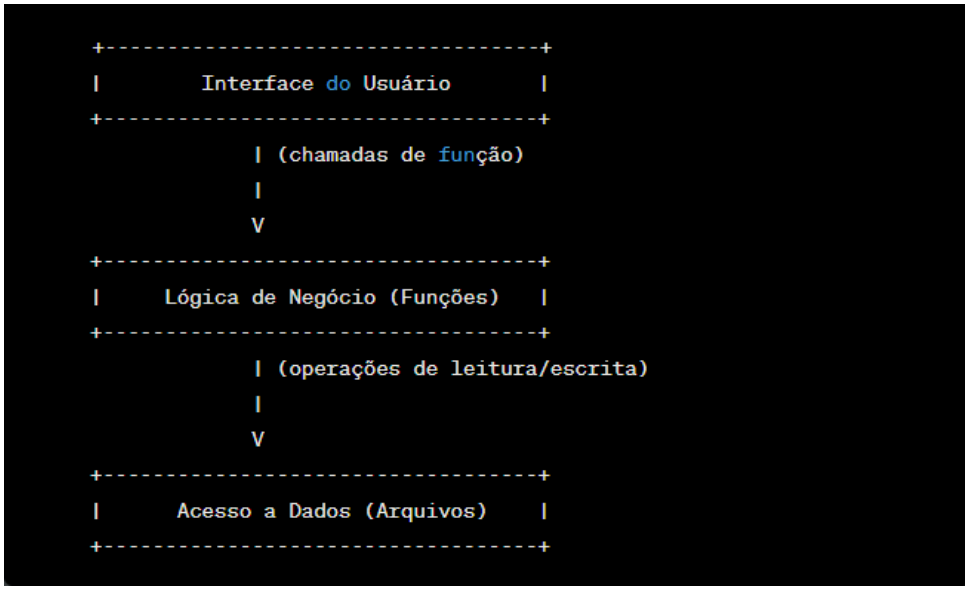
A estrutura do código é relativamente simples e não segue um padrão de arquitetura, com modelo de camadas, cliente-servidor ou MVC.

8.1 Visão Geral

Por enquanto não é possível realizar uma divisão consistente na fase atual do projeto, no entanto, essa divisão pode ajudar a organizar o código e separar as responsabilidades de cada parte do sistema.

As regras geralmente são estabelecidas com base na responsabilidade e na coesão dos componentes. Componentes relacionados à interface do usuário devem ser agrupados na camada de interface, enquanto componentes relacionados à lógica de negócio devem ser agrupados na camada de lógica de negócio. Componentes relacionados à persistência de dados devem ser agrupados na camada de acesso a dados.

A “separação” entre essas camadas pode ser estabelecida através de interfaces bem definidas, onde os componentes de uma camada se comunicam com os componentes de outra camada apenas por meio dessas interfaces. Isso facilita a manutenção e permite a substituição ou atualização de componentes sem afetar outras partes do sistema.



Este pode ser um exemplo de comunicação entre a interface do usuário chamando as funções que por sua vez interagem com a camada de acesso de dados.

8.2 Camadas

Neste código, atualmente não há uma divisão explícita em camadas. No entanto, é possível identificar algumas camadas conceituais com base nas funcionalidades.

Vamos nomear e definir essas camadas como:

- Camada de Interface do Usuário:

Responsável por interagir com o usuário, exibindo menus, solicitando entrada de dados e apresentando resultados.

Composta pelas funções dentro da função main que exibem os menus, solicitam opções e leem a entrada do usuário.

- Camada de Lógica de Negócio:

Contém a lógica e as regras da aplicação.

Composta pelas funções registro, consulta e deleta que implementam as funcionalidades principais do sistema.

- Camada de Acesso a Dados:

Responsável por interagir com o sistema de arquivos e realizar operações de leitura e escrita nos arquivos de texto.

Composta pelas operações de abertura, escrita, leitura e fechamento de arquivos presentes nas funções registro, consulta e deleta.

9 Visão de Dados

Os dados são armazenados diretamente em formato de texto. A estrutura dos registros é definida pela forma como os dados são gravados e lidos no arquivo de texto, usando vírgulas para separar os campos, o sistema mantém um arquivo chamado "BD.txt" onde os registros são armazenados.

10 Tamanho e desempenho

As principais características de dimensionamento de software que tem um impacto na arquitetura são:

1. **Volume de dados**: O tamanho e a quantidade de dados que o software precisa e o volume de dados for grande, podem ser necessárias considerações especiais de armazenamento, como a

utilização de bancos de dados escaláveis ou estratégias de particionamento de dados.

2. **Número de usuários**: O número de usuários que irão interagir com o software afeta a arquitetura em termos de capacidade de processamento, escalabilidade e desempenho. Se o software precisa suportar muitos usuários simultâneos, pode ser necessário adotar uma arquitetura capaz de distribuir a carga de trabalho e garantir uma resposta eficiente.
3. **Transações concorrentes**: Se o software precisa suportar muitas transações concorrentes, como em sistemas bancários ou de comércio eletrônico, a arquitetura deve ser projetada levando em consideração a concorrência e a consistência dos dados. Isso pode envolver o uso de bloqueios, transações isoladas e estratégias de gerenciamento de concorrência.

(A concorrência é usada para melhorar a eficiência e o desempenho dos sistemas, permitindo que múltiplas tarefas sejam executadas em paralelo. Isso pode ser especialmente útil em sistemas com vários núcleos de processamento, onde cada núcleo pode ser dedicado a executar uma tarefa específica simultaneamente)

4. **Requisitos de desempenho**: Os requisitos de desempenho, como tempos de resposta rápidos ou capacidade de processamento em tempo real, podem influenciar a arquitetura e pode ser necessário adotar técnicas de otimização, como o uso de cache, paralelização de tarefas ou distribuição de carga.
5. **Crescimento futuro**: É importante considerar o potencial de crescimento do software no futuro. A arquitetura deve ser escalável e capaz de lidar com aumentos futuros na carga de trabalho, volume de dados e número de usuários sem comprometer o desempenho e a funcionalidade do sistema

11 Qualidade

A arquitetura do software desempenha um papel fundamental na contribuição para diversos recursos e características do sistema, além da funcionalidade principal, analisando possíveis melhorias e implementando práticas de desenvolvimento adequadas.

Extensibilidade: Permite que o software seja facilmente estendido e modificado para incorporar novas funcionalidades ou atender a requisitos futuros. Interfaces bem definidas e pontos de extensão claros permitem a incorporação de novos módulos ou componentes sem perturbar o funcionamento existente.

Tempo de resposta: O software pode ter um requisito específico para fornecer respostas rápidas aos usuários, garantindo tempos de resposta aceitáveis para suas interações.

Taxa de transferência: Em sistemas que lidam com alto volume de transações, pode ser necessário garantir uma alta taxa de transferência para processar muitas transações por unidade de tempo.

Disponibilidade: Em sistemas críticos, a disponibilidade contínua do software é uma restrição importante. Isso pode exigir a adoção de arquiteturas resilientes, como replicação de servidores ou tolerância a falhas.

Eficiência de recursos: O uso eficiente dos recursos do sistema, como memória, processamento e armazenamento, pode ser uma restrição importante, especialmente em

sistemas embarcados ou em dispositivos com recursos limitados.

Confiabilidade: Uma arquitetura robusta inclui estratégias de tratamento de exceções, recuperação de erros, tolerância a falhas e mecanismos de redundância. A distribuição adequada das responsabilidades entre os componentes e a implementação de padrões de projeto como o "tratamento de erros local" ajuda a garantir que o software seja confiável e resiliente situações adversas.