Implementação de sistemas

LES0750
Desenvolvimento
de Sistemas de Informação

Conteúdo

- Considerações iniciais
- Metodologia Garrett
- Projeto de Componentes
- Visões de Componentes
- Projeto orientado a objetos
- Classes
- Exemplos
- Ferramentas de desenvolvimento
- Referências bibliográficas

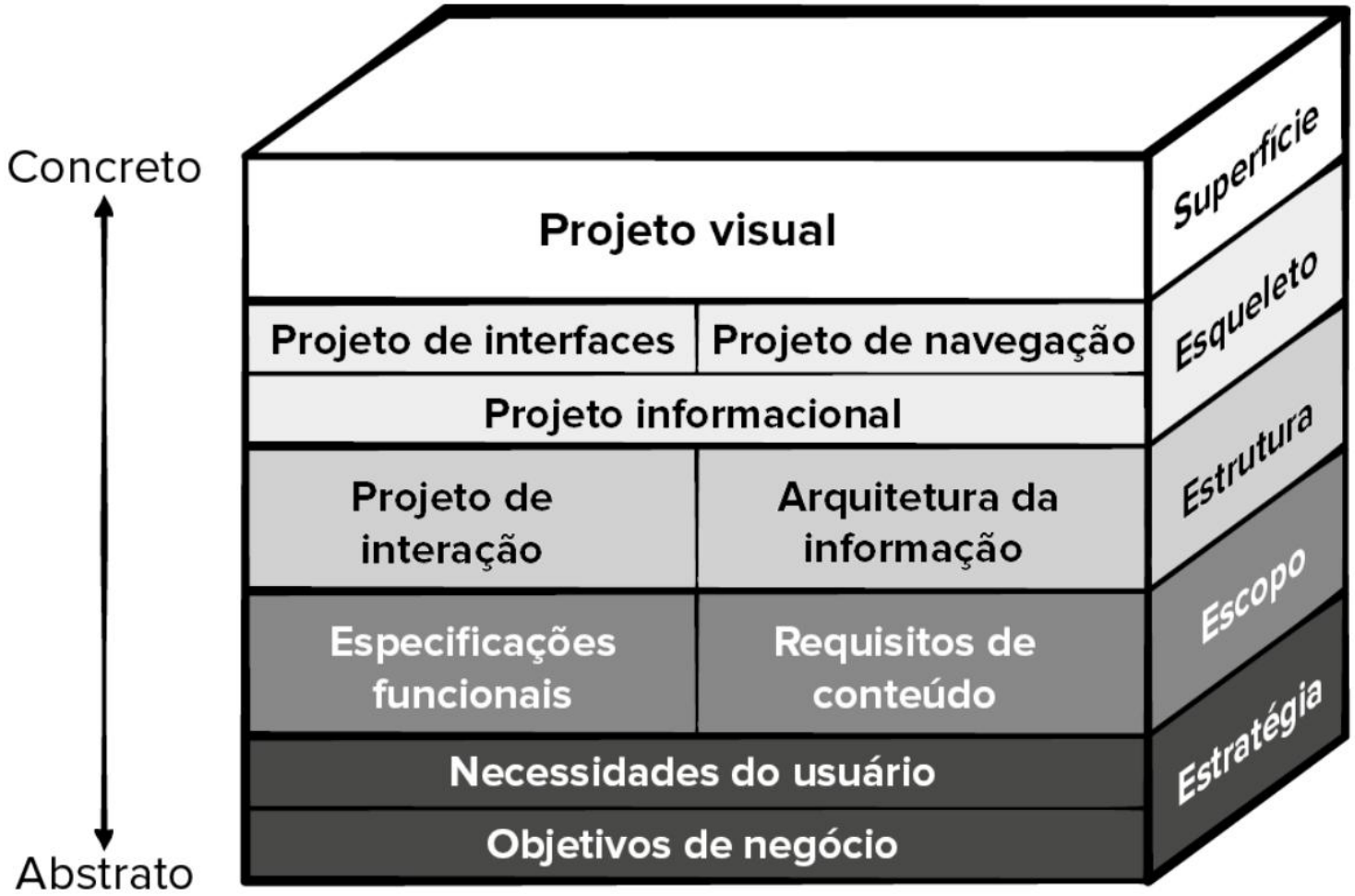
Considerações iniciais

Considerações iniciais

- A implementação de software é o estágio no processo de engenharia de software no qual um sistema de software executável é desenvolvido;
- A implementação é o processo de realização do projeto em um programa.

Metodologia Garrett

Metodologia Garrett



Pressman & Maxim, 2021

Mais informações: https://jera.com.br/blog/7021/design-ux/metodologia-de-garrett

Projeto de Componentes

Projeto de Componentes

- O projeto de componentes serve de ponte entre o projeto de arquitetura e a codificação;
- Componente é um bloco construtivo modular para software de computador;
- Decomposição de um sistema complexo em módulos autônomos e interligáveis;
- Cada componente possui uma função específica e delimitada.

Benefícios de uma abordagem modular

Modularidade e coesão:

- **Componentes bem definidos**: cada componente possui uma função específica e delimitada;
- **Maior independência**: os componentes interagem através de interfaces bem definidas;
- **Reutilização facilitada**: componentes testados e validados podem ser reutilizados em outros projetos.

Benefícios de uma abordagem modular

Flexibilidade e manutenabilidade:

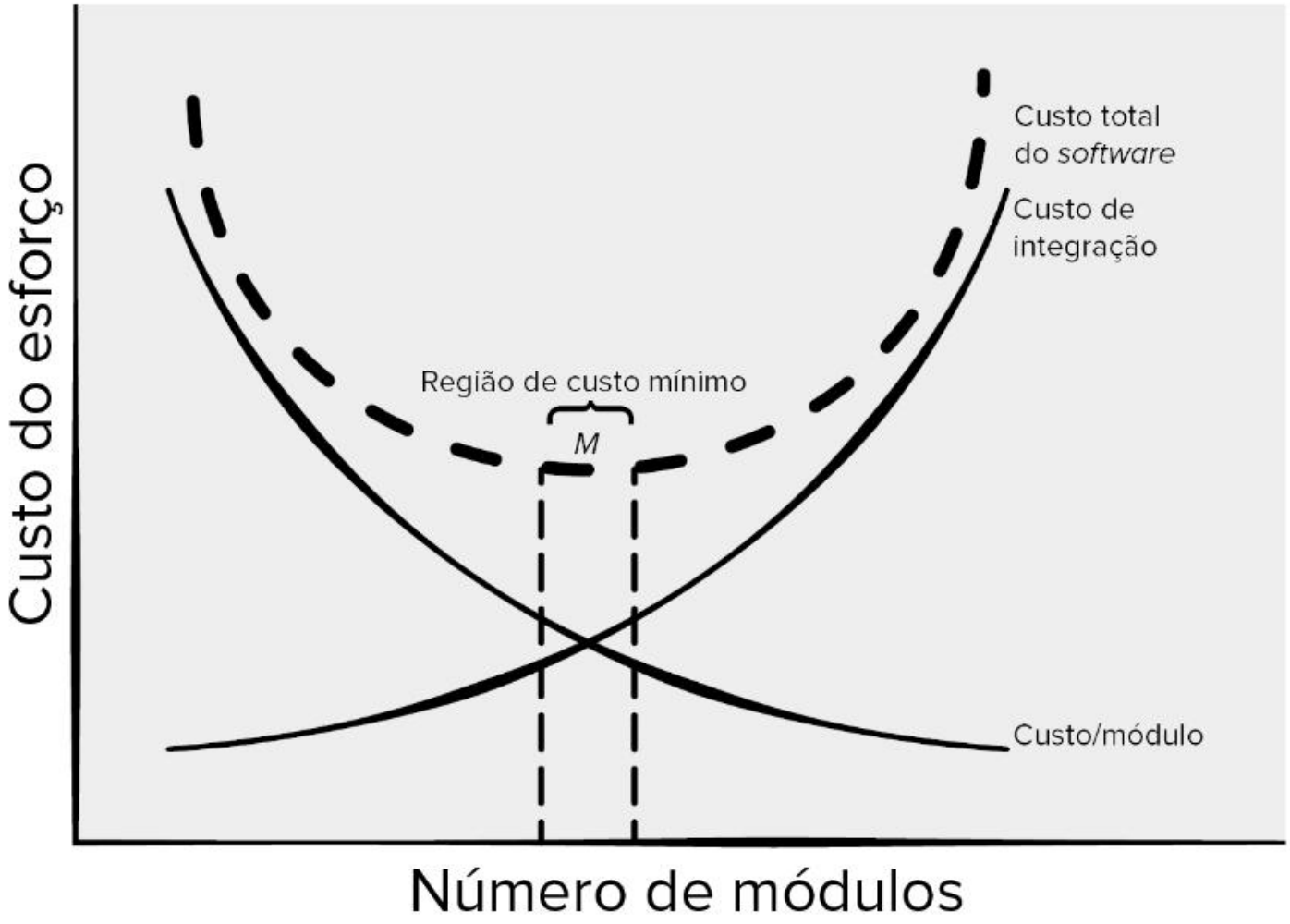
- **Maior adaptabilidade**: a arquitetura modular permite modificações e atualizações em componentes específicos sem afetar todo o sistema;
- **Manutenção simplificada**: a localização e o reparo de falhas são facilitados, pois os componentes são isolados e encapsulados;
- **Escalabilidade aprimorada**: O sistema pode ser facilmente expandido adicionando novos componentes ou aprimorando os existentes.

Benefícios de uma abordagem modular

Melhoria na produtividade e qualidade:

- **Desenvolvimento mais rápido**: a divisão em componentes permite a realização de tarefas em paralelo;
- **Testes mais eficientes e confiáveis**: componentes menores e isolados são mais fáceis de serem testados e depurados;
- **Maior qualidade do código**: o foco num determinado componente permite aos desenvolvedores escreverem código mais limpo, robusto e sustentável.

Modularidade e custo do software



Pressman & Maxim, 2021

Exemplos de componentes

Exemplo: geral

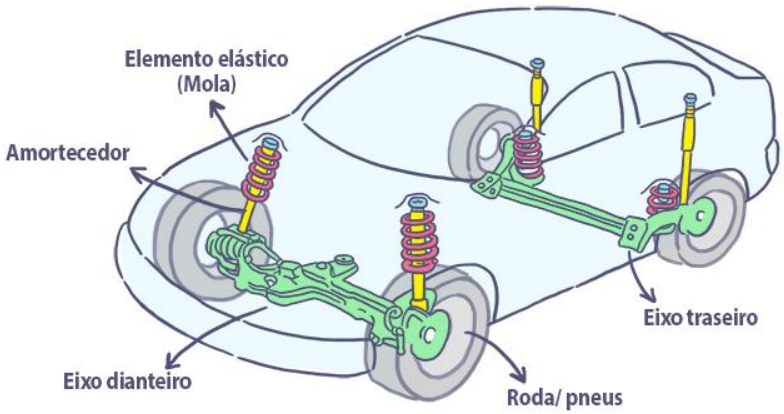
- Autenticação e autorização;
- Comunicação;
- Gerenciamento de erros e logs;
- Acesso a banco de dados;
- Visualização de dados;
- Interface Gráfica (GUI - Graphical User Interface).

Exemplo: Banco online

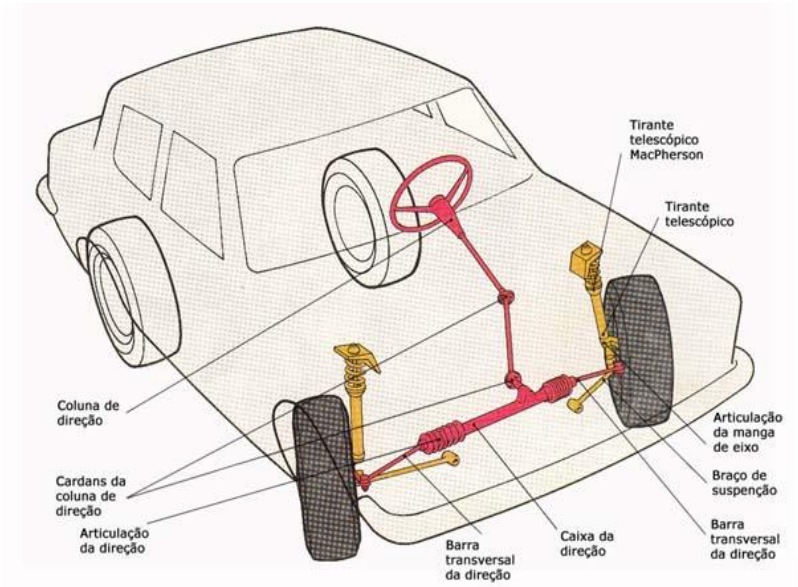
- Autenticação;
- Gerenciamento de contas;
- Atendimento ao cliente;
- Detecção de fraude.

Exemplos de componentes (sistemas)

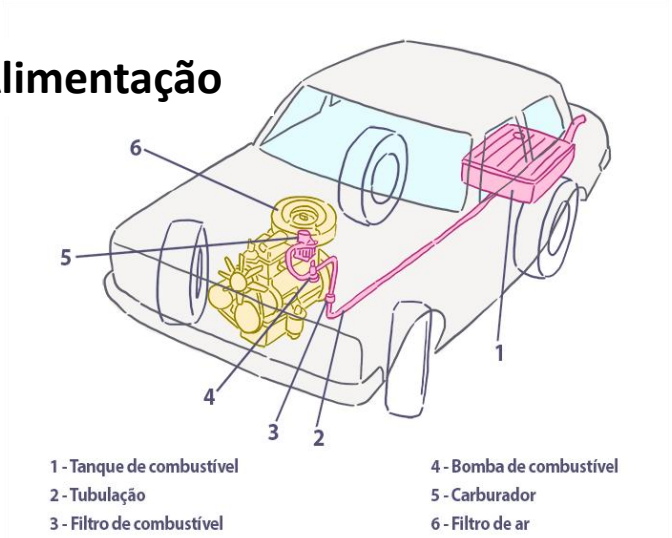
Suspensão



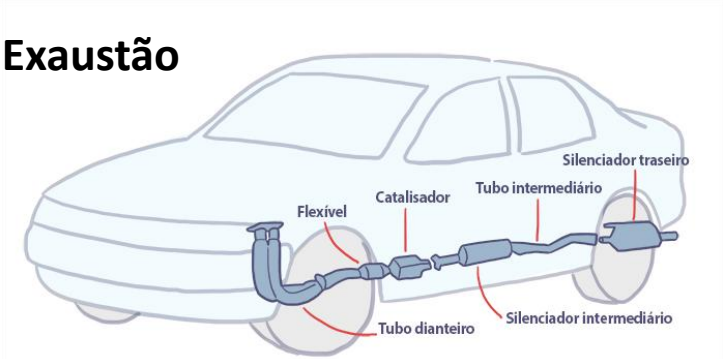
Direção



Alimentação



Exaustão



Visões de Componentes

Visões de Componentes

- **Tradicional**: componentes como unidades funcionais autônomas (gerenciamento de estoque em um armazém);
- **Relacionada a processos**: componentes como parte de um fluxo de trabalho (gerenciamento de pedidos num restaurante);
- **Orientada a objetos**: componentes como entidades encapsuladas (atributos, comportamentos e relacionamentos) (gerenciamento de clientes em uma loja virtual).

Projeto orientado a objetos

Projeto orientado a objetos

- Metodologia de desenvolvimento de software que se baseia na criação de objetos, entidades que encapsulam dados (atributos) e comportamentos (métodos).

Projeto orientado a objetos

Exemplo 1: **Sistema de um banco**

Objeto Conta:

- **Atributos**: número da conta, saldo, tipo de conta;
- **Métodos**: depositar, sacar, transferir e consultar saldo.

Projeto orientado a objetos

Exemplo 1: **Sistema de um banco**

Objeto Cliente:

- **Atributos**: nome, CPF, endereço, telefone e conta bancária;
- **Métodos**: consultar dados cadastrais, alterar dados e atualizar conta.

Projeto orientado a objetos

Exemplo 1: **Sistema de um banco**

Objeto Transação:

- **Atributos**: data, valor, tipo de transação e conta bancária;
- **Métodos**: registrar transação, consultar histórico e gerar extrato.

Projeto orientado a objetos

Exemplo 2: **Rede Social**

Objeto Usuário:

- Atributos:
- Métodos:

Objeto Publicação:

- Atributos:
- Métodos:

Objeto Comentário:

- Atributos:
- Métodos:

Projeto orientado a objetos

Exemplo 3 : **Loja Virtual**

Objeto Produto:

- Atributos:
- Métodos:

Objeto Carrinho de compras:

- Atributos:
- Métodos:

Objeto Pedido:

- Atributos:
- Métodos:

Classes

Classes

- Define as características (atributos) e comportamentos (métodos) de todos os objetos;
- Servem de modelo para a criação de objetos;
- Plano de construção dos objetos;
- É como um molde que determina como algo deve ser estruturado e como deve agir.

Classes

Exemplo para a classe Cliente

Atributos: nome, cpf, email, senha

Métodos (ações): verificarSenha, consultarCadastro, alterarDados

Exemplo para a classe ContaBancaria

Atributos: numeroConta, saldoConta, tipoConta e titularConta

Métodos (ações): depositar, sacar, transferir e consultarSaldo

Exemplo

Sistema e-Commerce

Componentes

Gerenciador de Produtos:

- **Funcionalidades**: cadastrar, editar, remover e pesquisar produtos;
- **Interfaces**: para cadastro, edição, remoção e pesquisa de produtos.

Gerenciador de Pedidos:

- **Funcionalidades**:criar, visualizar, atualizar e cancelar pedidos;
- **Interfaces**: para criação, visualização, atualização e cancelamento de pedidos.

Componentes

Gerenciador de Pagamentos:

- **Funcionalidades**: processar pagamentos de pedidos;
- **Interfaces**: para processamento de pagamentos com cartão de crédito, PIX, boleto bancário e PayPal.

Gerenciador de Usuários:

- **Funcionalidades**: cadastrar, editar, remover e autenticar usuários;
- **Interfaces**: para cadastro, edição de usuários, remoção e autenticação de usuários.

Objetos

Produto:

- **Atributos**: ID, nome, descrição, preço, estoque, categoria e imagens;
- **Métodos**: obter detalhes, adicionar ao carrinho, remover do carrinho e comprar.

Pedido:

- **Atributos**: ID, data, cliente, itens, status e forma de pagamento;
- **Métodos**: obter detalhes, adicionar item, remover item, calcular total, confirmar e cancelar.

Objetos

Pagamento:

- **Atributos**: ID do pagamento, tipo de pagamento, status do pagamento e valor do pagamento;
- **Métodos**: processar pagamento, verificar status, enviar confirmação de pagamento.

Usuário:

- **Atributos**: ID, nome, email, senha, endereço e telefone;
- **Métodos**: cadastrar, editar, remover, autenticar e acessar perfil.

Classes

Classe Produto:

- **Atributos**: ID, nome, descricao, preço, estoque, categoria e imagens
- **Métodos**:
  - obterDetalhes(): Retorna os detalhes do produto
  - adicionarAoCarrinho(): Adiciona o produto ao carrinho de compras
  - removerDoCarrinho(): Remove o produto do carrinho de compras
  - comprar(): Realiza a compra do produto

Classes

Classe Pedido:

- **Atributos**: ID, data, cliente, itens, status e formaPagamento
- **Métodos**:
  - obterDetalhes(): retorna os detalhes do pedido
  - adicionarItem(): adiciona um item ao pedido
  - removerItem(): remove um item do pedido
  - calcularTotal(): calcula o total do pedido
  - confirmar(): confirma o pedido
  - cancelar(): cancela o pedido

Classes

Classe Pagamento:

- **Atributos**: idPagamento, idPedido, tipoPagamento, statusPagamento, valorPagamento, dataHoraPagamento, dadosPagamento
- **Métodos**:
  - processarPagamento(): aciona o processo de pagamento
  - verificarStatusPagamento(): verifica o status da transação
  - enviarConfirmacaoPagamento(): envia uma confirmação de pagamento para o cliente
  - cancelarPagamento(): cancela o pagamento
  - registrarPagamento(): registra a transação

Classes

Classe Usuário:

- **Atributos**: ID, nome, email, senha, endereco e telefone
- **Métodos**:
  - cadastrar(): cadastra um novo usuário.
  - editar(): edita os dados de um usuário.
  - remover(): remove um usuário.
  - autenticar(): autentica um usuário e retorna um token de acesso.
  - acessarPerfil(): acessa o perfil do usuário.

Código

Python

```python
class Cliente:
    def __init__(self, nome, cpf, email, senha):
        self.nome = nome
        self.cpf = cpf
        self.email = email
        self.senha = senha

    def verificar_senha(self, senha_tentativa):
        return self.senha == senha_tentativa


class ContaBancaria:
    def __init__(self, cliente, numero_conta, saldo):
        self.cliente = cliente
        self.numero_conta = numero_conta
        self.saldo = saldo

    def consultar_saldo(self, senha):
        if self.cliente.verificar_senha(senha):
            print(f"Saldo da conta {self.numero_conta}: R$ {self.saldo:
        else:
            print("Senha incorreta. Acesso negado.")
```

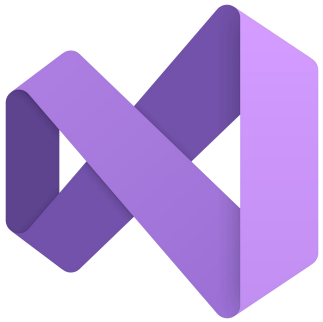
Ferramentas de desenvolvimento

Ferramentas de desenvolvimento

- Compilador: sistema de edição orientado a sintaxe, que permita a criação, edição e compilação de códigos;
- Sistema de depuração de linguagem (identificação de bugs);
- Ferramentas de edição gráfica (UML - Linguagem de Modelagem Unificada) para determinar a arquitetura do software;
- Ferramentas de teste;
- Ferramentas de apoio a projetos.

Ferramentas de desenvolvimento

- IDE – Integrated Development Environment;
- Ambiente de desenvolvimento integrado;
- Fornece apoio para o desenvolvimento de um software, sistema e aplicativo;
- Pode suportar várias linguagens de programação.



Visual Studio

https://visualstudio.microsoft.com/pt-br/

Ferramentas de desenvolvimento

Open Source

- **Código aberto**: desenvolvido e suportado por comunidades de forma colaborativa

Exemplos

- LINUX: sistema operacional
- MySQL, PostgreSQL: banco de dados
- Python, Java, JavaScript, C++, TypeScript, PHP, C, C Sharp: linguagem de programação

Referências bibliográficas

- Pressman, Roger, S. e Bruce R. Maxim. Engenharia de software (9th edição). Grupo A, 2021.