CÂMPUS

Bagé

ARQUITETURA DE SOFTWARE

RODRIGO R SILVA

DEFINIÇÃO ARQUITETURA SOFTWARE

As apropriações de nomenclaturas do mundo real são comuns no universo de tecnologia, e não poderia ser diferente com o termo arquitetura. No mundo físico, profissionais de arquitetura constroem projetos e desenhos de casas, prédios, entre outras estruturas, baseando-se em um contexto que envolve diversas variáveis e situações.

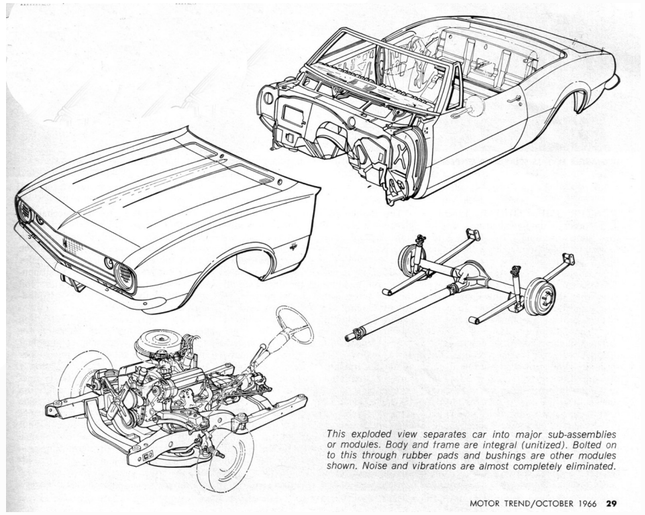
DEFINIÇÃO ARQUITETURA SOFTWARE

- **Contexto: arquitetura de uma casa**
  - Na construção civil, a arquitetura de uma casa define aspectos como:
    - a disposição dos cômodos (quartos, sala, cozinha, etc.)
    - os materiais usados (concreto, madeira, vidro)
    - o design estrutural e estético a funcionalidade (ventilação, iluminação, acessibilidade)

Há vários diagramas:
- planta baixa (cômodos)
- projeto de parte elétrica,
- projeto de parte hidráulica
- projeto de iluminação
- etc.

DEFINIÇÃO ARQUITETURA SOFTWARE

- **Contexto: arquitetura de um carro**
  - Na engenharia mecânica, a arquitetura de um carro define:
    - o chassi e sua resistência
    - o posicionamento do motor (dianteiro, central ou traseiro)
    - a aerodinâmica e o design externo
    - o sistema eletrônico (sensores, computador de bordo, etc.)



Há vários diagramas:
- projeto estrutural (chassi)
- projeto do sistema elétrico
- projeto do sistema de freios
- projeto do motor e transmissão
- etc.

DEFINIÇÃO ARQUITETURA SOFTWARE

A definição de arquitetura de software, de acordo com a ISO/IEC/IEEE 42010:2022, é a seguinte:

"Arquitetura de software é a estrutura fundamental ou o esqueleto de um sistema de software, que define seus componentes, suas relações e seus princípios de projeto e evolução."

Se refere à organização de um sistema. Ela é responsável por definir os componentes que farão parte de um projeto, suas características, funções e a forma como devem interagir entre si e com outros softwares.

DEFINIÇÃO ARQUITETURA SOFTWARE

- **Contexto: arquitetura de um software**
  - Na engenharia de software, a arquitetura de um software define aspectos como:
    - a comunicação entre os componentes do sistema (ex.: navegador web x aplicação servidor x banco de dados)
    - a estrutura dos módulos, pacotes e camadas
    - os padrões de desenvolvimento usados (MVC, camadas, microserviços, monolítico, etc).
    - a escalabilidade e segurança do sistema

Há vários diagramas:
- para projeto estrutural:
  - diagrama de classes, de componentes, etc.
  - **diagramas de arquitetura**
- para projeto comportamental:
  - diagrama de sequência, casos de uso, etc.

DEFINIÇÃO ARQUITETURA SOFTWARE

- **diagramas de arquitetura de software:**
  - projeto conceitual (arquitetura de referência)
  - diagrama de componentes
  - diagrama de implantação
  - diagrama de pacotes
  - diagrama de sequência (de sistema)

UML

DEFINIÇÃO ARQUITETURA SOFTWARE

- A arquitetura de software de um sistema é
  - a **definição dos componentes** do software, e de suas **propriedades externas** (ex.: interface ou capacidade de processamento), e
  - a definição dos **relacionamentos entre os componentes** (ou com componentes externos ao sistema)

Usamos diferentes diagramas para representar visualmente a arquitetura de um software

- O termo também se refere à **documentação** da arquitetura de software do sistema

DEFINIÇÃO ARQUITETURA SOFTWARE

- **Exemplo 1:** Arquitetura de software de um sistema que embala diferentes tipos de objetos usando um braço robótico.

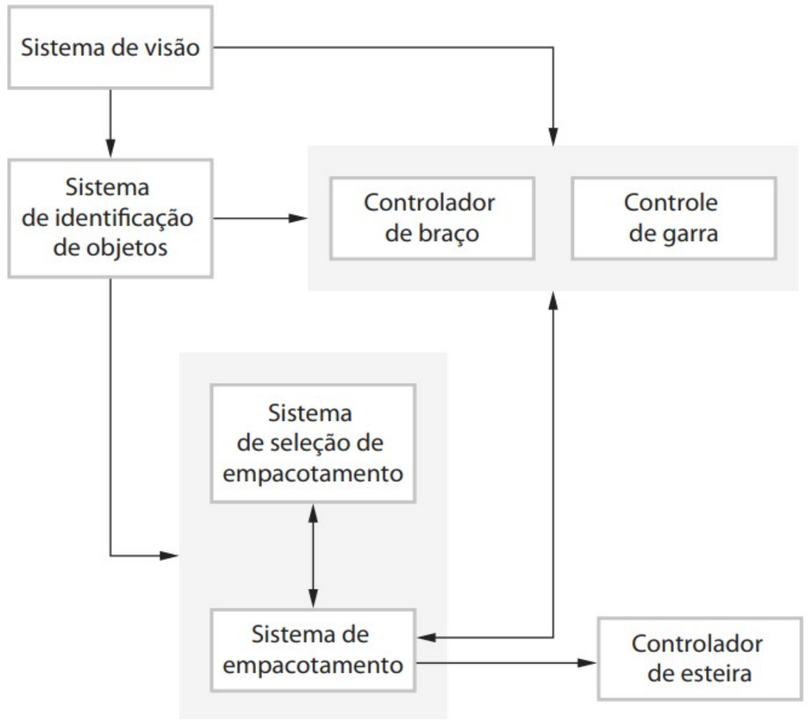
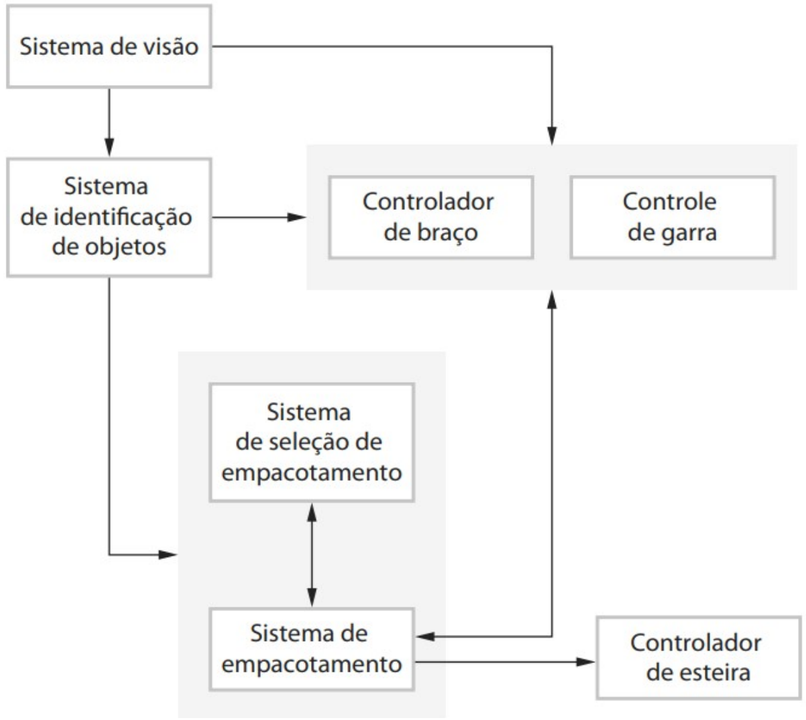


Diagrama de projeto conceitual da arquitetura

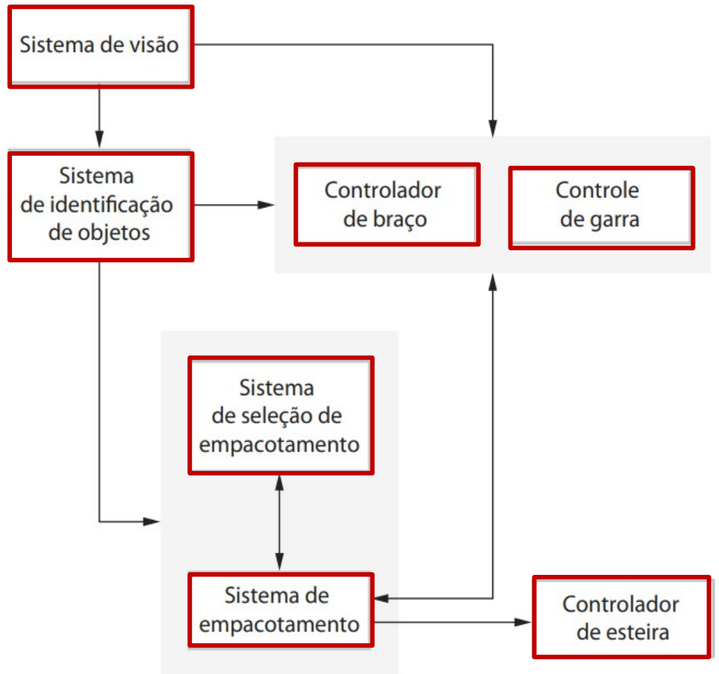
DEFINIÇÃO ARQUITETURA SOFTWARE



Tem vários componentes que **precisam ser desenvolvidos**:

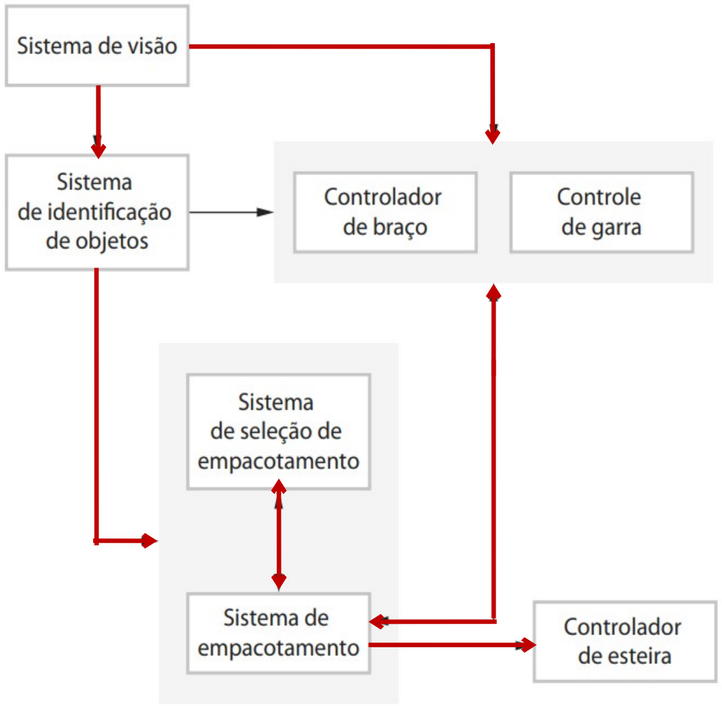
- por exemplo, o componente *Sistema de Visão,* o qual obtém a imagem do objeto e envia informações dela para o componente composto pelos componentes *Controle de braço* e *Controle de garra*, e para o componente *Sistema de Identificação de objetos;*

DEFINIÇÃO ARQUITETURA SOFTWARE



- Cada **bloco** representa um componente do sistema a ser desenvolvido (ex.: Sistema de Visão, Controle de Garra, etc.)
  - Os **componentes individuais do sistema** implementam os requisitos funcionais (e podem implementar requisitos não funcionais);

DEFINIÇÃO ARQUITETURA SOFTWARE

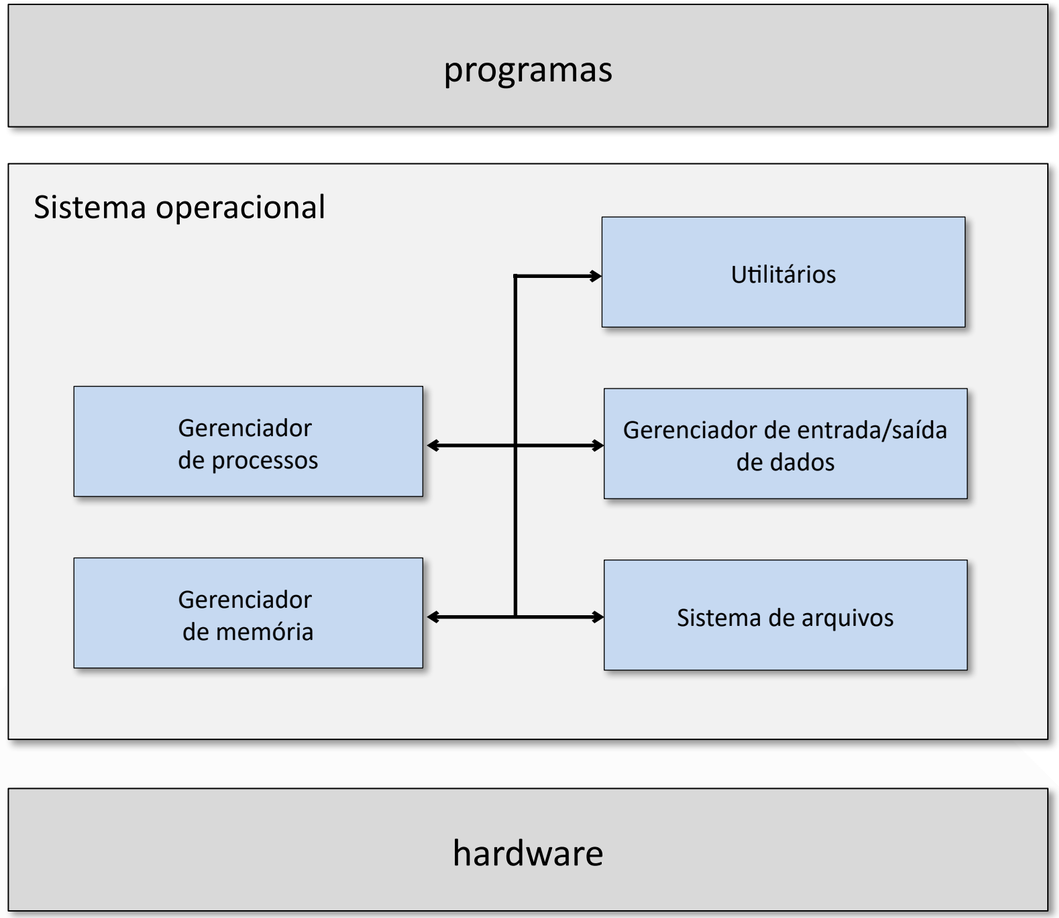


- As **setas** indicam informações enviadas ou trocadas entre componentes (ex.: informações de visão do objeto enviada para o controle de garra)
- As setas são chamadas à API (Application Program Interface)

  Ex.: um componente A chama a API do componente B para ler as propriedades de B visíveis externamente

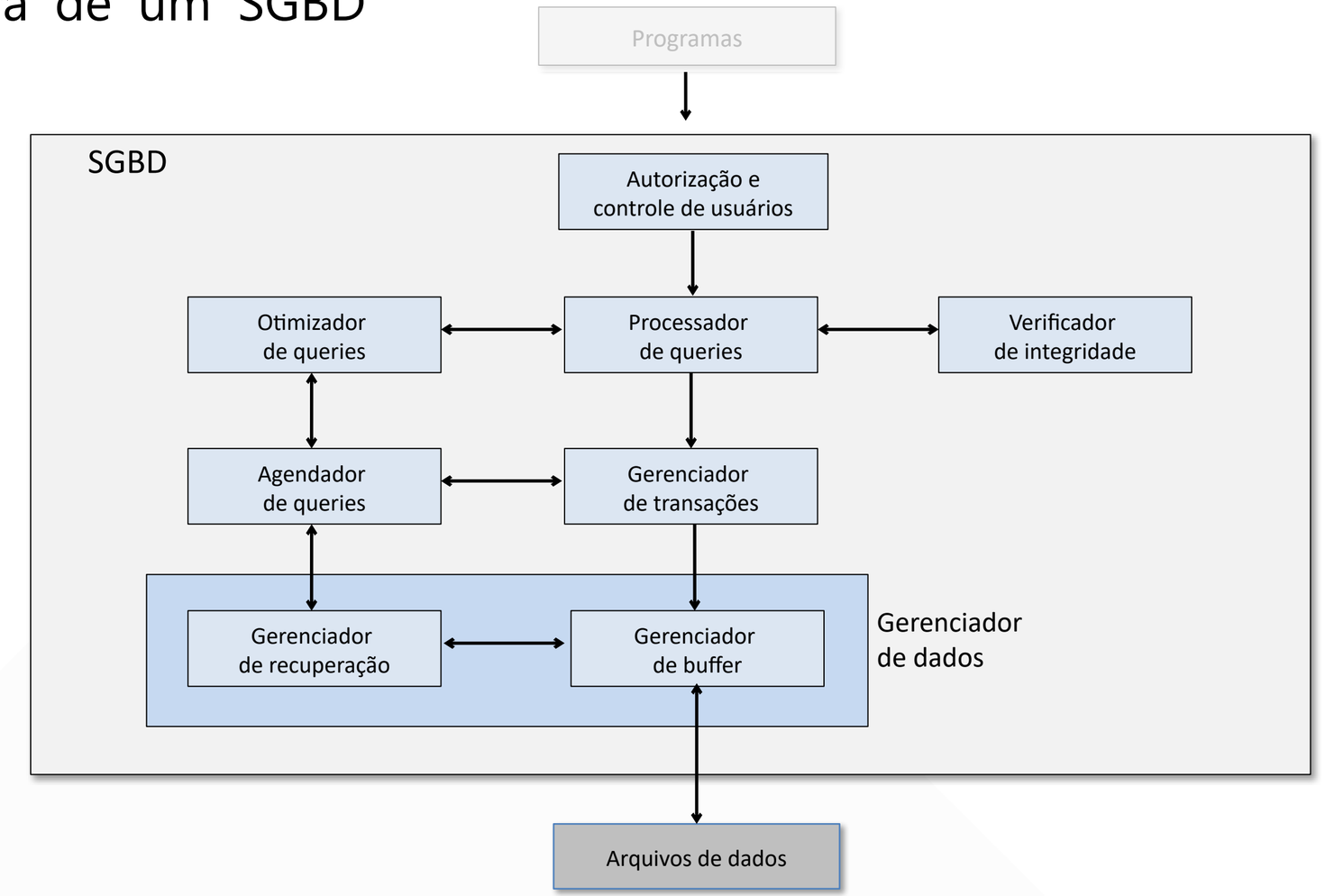
DEFINIÇÃO ARQUITETURA SOFTWARE

- **Exemplo 2:** Arquitetura de um sistema operacional.



DEFINIÇÃO ARQUITETURA SOFTWARE

- **Exemplo 3:** Arquitetura de um SGBD



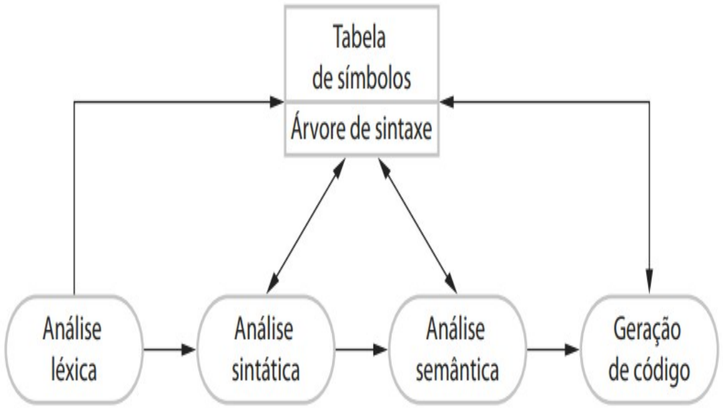
DEFINIÇÃO ARQUITETURA SOFTWARE

Os requisitos funcionais e não funcionais do sistema determinam a arquitetura do sistema → assim um mesmo sistema pode ter 2 ou + arquiteturas

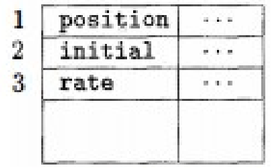
Imagine um compilador
(software que transforma código-fonte em um programa executável)

Exemplo 4-a: Arquitetura de software do compilador em filtros e dutos

Esta arquitetura usa o padrão de arquitetura de filtros e dutos - esse padrão de arquitetura é ideal quando o sistema funciona isoladamente, em um processo de *batch*, processando dados em fila (neste caso, compila um programa após o outro, em uma fila, e de forma **offline**, por exemplo, de madrugada)

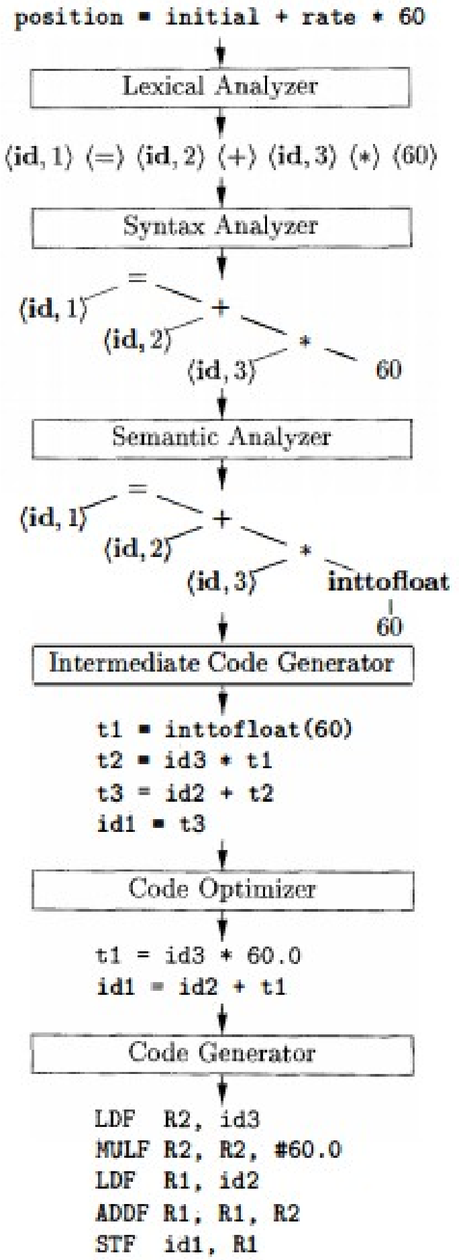


DEFINIÇÃO ARQUITETURA SOFTWARE



1	position	···
2	initial	···
3	rate	···

SYMBOL TABLE



1º analisa se a linguagem aceita cada uma das palavras do código-fonte

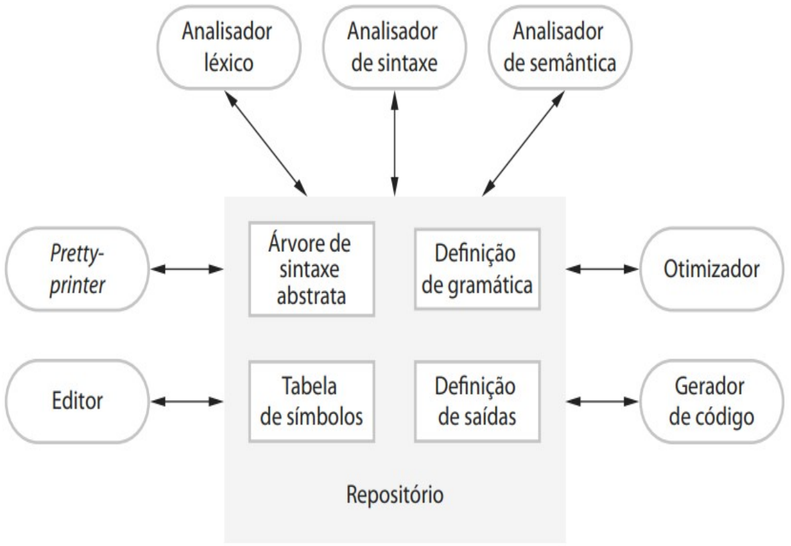
2º analisa se cada expressão do código-fonte é válida na linguagem

3º nível de análise. Por exemplo: Se A, B foram declarados, existem outras variáveis com mesmos nomes, são do tipo float.

DEFINIÇÃO ARQUITETURA SOFTWARE

Exemplo 4-b: Arquitetura de software do compilador na arquitetura de repositório

Esta arquitetura usa o padrão de arquitetura de repositório
essa arquitetura é ideal para sistemas com módulos que usam ou atualizam diferentes fontes de dados
(neste caso, os módulos são os plug-ins dentro da IDE;
e o compilador funciona **online**: vai usando as fontes de dados a medida o programador vai criando e testando o programa)



DEFINIÇÃO ARQUITETURA SOFTWARE

Os requisitos funcionais e não funcionais do sistema determinam a arquitetura do sistema → assim um mesmo sistema pode ter 2 ou + arquiteturas

a arquitetura escolhida para a implementação é aquela que atente aos requisitos funcionais e não funcionais

Por exemplo, se o **cliente** indicar nos requisitos de software que o sistema será usado online dentro de uma IDE, com diversos plug-ins, usando diferentes fontes de dados, a opção 2 é a **escolhida**, pois é aquela que atende a tais requisitos.

Por outro lado, se o **cliente** indicar nos requisitos de software que o sistema será usado uma vez ao dia, de forma offline, compilando uma sequência de 10 mil programas/dia, a opção 1 é a **escolhida**, pois é aquela que atende a tais requisitos.

DEFINIÇÃO ARQUITETURA SOFTWARE

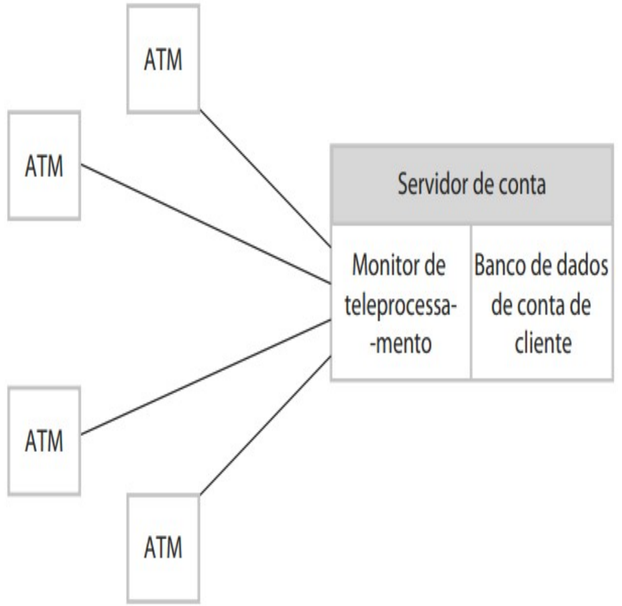
- **Exemplo 5:** Arquitetura de software de um sistema distribuído de caixas eletrônicos de bancos

  - imagine um conjunto de caixas eletrônicos (ATMs) de um banco, distribuídos em várias cidades, se comunicando por uma rede de computadores própria do banco

- Esta figura mostra uma versão simplificada da organização do sistema de ATMs (caixas eletrônicos). Observe que as ATMs não estão ligadas diretamente com o banco de dados do banco, mas sim a um monitor de teleprocessamento

  Um monitor de teleprocessamento é um sistema de middleware que organiza as comunicações com os clientes remotos e serializa as transações de clientes para o processamento no banco de dados. Isso garante que as transações sejam independentes e não interfiram entre si

  Usar transações seriais significa que o sistema pode se recuperar de defeitos sem corromper os dados do sistema (requisito não funcional de resiliência)



Há 2 Níveis de A.S

- **Arquitetura de** pequena escala = usada no projeto de programas individuais **(sistemas monolíticos = 1 só módulo)**
  - foca na decomposição do programa em componentes menores

- **Arquitetura de** grande escala = usada no projeto de **sistemas distribuídos** (vários módulos separados remotamente)

  - estes sistemas possuem outros sistemas completos internos e programas
  - os programas e sistemas podem rodar em diferentes computadores

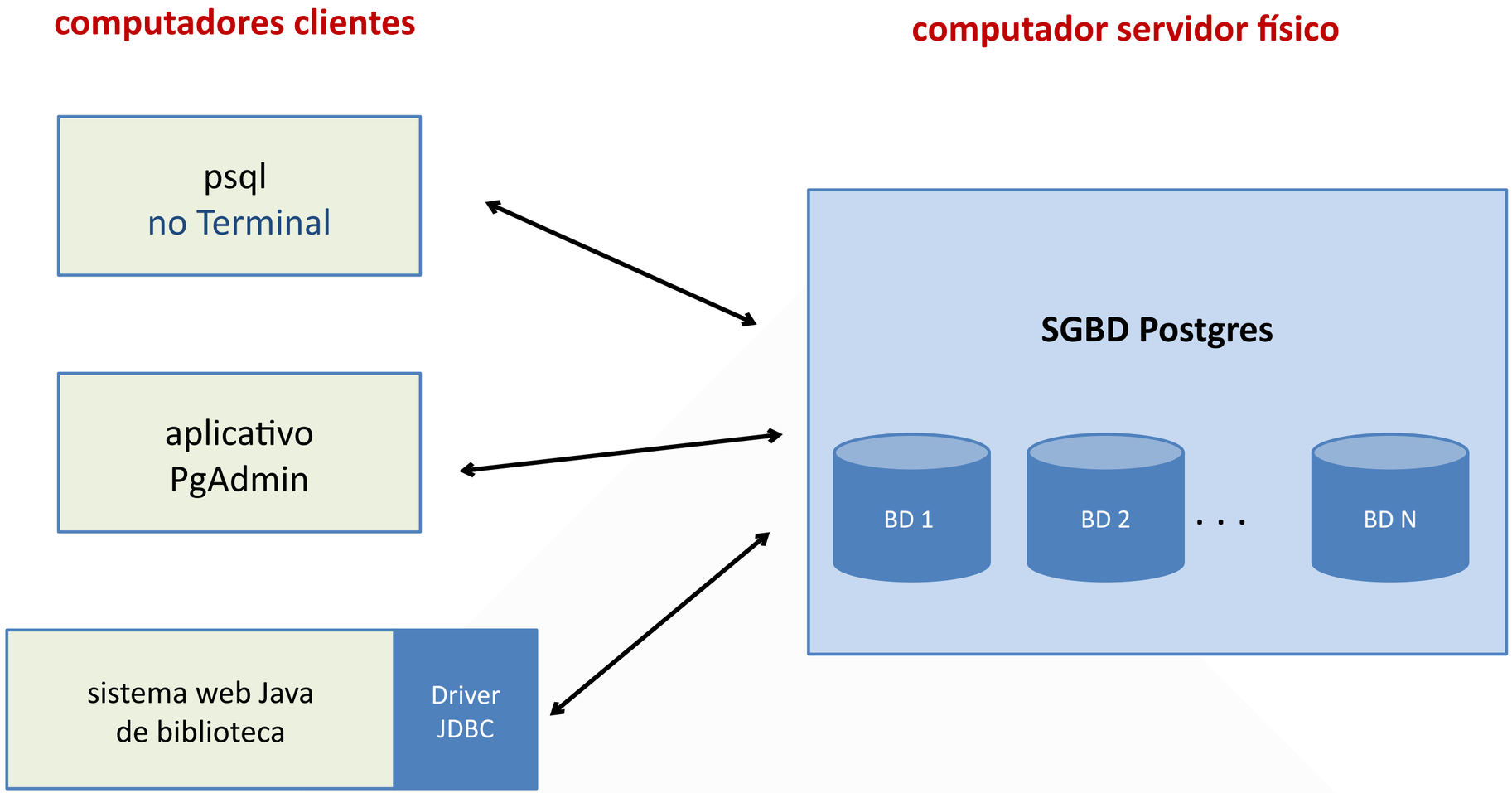
Por que a A.S é importante?

- Porque **afeta** requisitos **não** funcionais do sistema (desempenho, disponibilidade, capacidade de distribuição, manutenibilidade, segurança, etc.)
  - por exemplo, se há requisitos de disponibilidade, o tipo de arquitetura escolhido deve permitir manter a disponibilidade planejada para o sistema

- Porque ela **facilita** ou **dificulta** a implementação dos requisitos funcionais do sistema

- Apresenta a visão geral do sistema, que é útil tanto para a equipe de desenvolvimento, quanto para negociações e análise de requisitos com o cliente na etapa inicial de Especificação de Software

Arquitetura de um SGBD



Arquitetura de um SGBD

- Como visto, a arquitetura de um **SGBD** (**Sistema Gerenciador de Banco de Dados)** segue um modelo **cliente-servidor**:
  - o banco de dados é executado em um **servidor** (que pode ser um computador servidor físico ou virtual - ex.: rodando em um software de virtualização como o VirtualBox), e
  - os **clientes** acessam os dados por meio de conexões de rede, no modelo de comunicação **requisição-resposta**

- Principais componentes dessa arquitetura:
  - **Servidor de Banco de Dados**
- O SGBD, por exemplo o **PostgreSQL** ou **MySQL**, é instalado em um computador servidor. Ele fica responsável por gerenciar os dados, responder às requisições dos clientes e garantir a segurança e a integridade de dados.
- Ele **escuta** conexões em uma porta específica (por padrão, no PostgreSQL, é a **porta 5432**).
- Pode ser configurado para aceitar conexões **remotas** ou apenas **locais.**
- Geralmente, ele roda como um serviço no sistema operacional

Arquitetura de um SGBD

- Principais componentes dessa arquitetura:
  - os clientes são aplicações que se conectam ao banco para executar consultas e comandos SQL.
  - Podem ser:
    - **clientes gráficos**: por exemplo, PgAdmin
    - **clientes de linha de comando no Terminal**: por exemplo, o psql do PostgreSQL
    - **aplicações web, desktop, mobile que nós criamos**: estas aplicações usam componentes chamados de **drivers** para se conectar ao banco via componentes JDBC, ODBC ou bibliotecas, como pg-promise (Node.js)

- **Comunicação entre Cliente e Servidor**
  - esta pode ser feita por diferentes métodos:
    - **local**: se o cliente estiver no mesmo computador do servidor, ele pode se conectar via **Unix socket** (no Linux) ou **pipe nomeado** (no Windows)
    - **remoto (via rede)**: se o cliente estiver em outro computador, ele usa o protocolo TCP/IP para se conectar à porta do SGBD no servidor.
      É aqui que entram a configuração de **firewall**, **autenticação** e **segurança** (SSL/TLS)

Arquitetura de um SGBD

- **Comunicação entre Cliente e Servidor**
  - imagine que você tenha um servidor com **PostgreSQL** rodando na nuvem ou em um servidor físico
  - você pode acessar o banco de dados a partir de outro computador usando um comando de Terminal como:

    psql -h meu-servidor.com -U meu_usuario -d meu_banco

    ou configurar o **PgAdmin** para se conectar ao IP do servidor, informando a porta correta.

Conclusão

- Definição de arquitetura de software.
- Exemplos de arquitetura de software.

Bibliografia básica

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo, SP: Pearson, 2018. E-book. Disponível em: https://plataforma.bvirtual.com.br.
MORAIS, Izabelly Soares de (org.). **Engenharia de Software**. São Paulo: Pearson, 2017. E-book. Disponível em: https://plataforma.bvirtual.com.br.
GALLOTTI, Giocondo Marino Antonio (org.). **Arquitetura de Software**. São Paulo: Pearson, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br.

Bibliografia complementar

GIRIDHAR, Chetan; KINOSHITA, Lúcia Ayako. **Aprendendo padrões de projeto em Python**: Tire proveito da eficácia dos padrões de projeto (design patterns) em Python para resolver problemas do mundo real em arquitetura e design de software. São Paulo, SP: Novatec, 165 p. ISBN 9788575225233.
FREEMAN, Eric; FREEMAN, Elisabeth. **Use a cabeça:** padrões e projetos. 2.ed. Rio de Janeiro, RJ: Alta Books, 2009. 478 p. ISBN 9788576081746.