Aula 1

Desenvolvimento *Web – Back End*

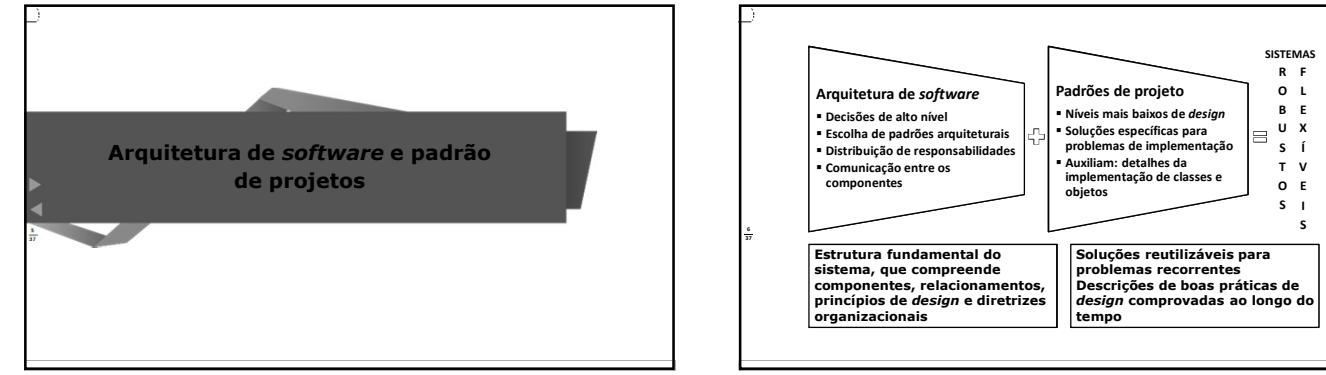
Profª Luciane Yanase Hirabara Kanashiro

Conversa Inicial

- Nesta aula veremos um pouco sobre arquitetura de 3 camadas e conheceremos um pouco sobre a plataforma e a arquitetura Java

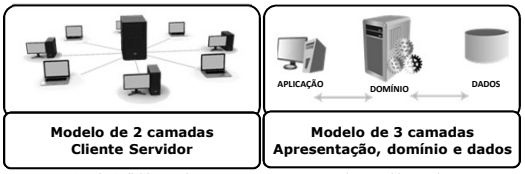
- Relação entre a arquitetura de *software* e padrões de projetos
- Arquitetura Java EE
- Tecnologia Java EE
- Eclipse Jakarta EE
- Revisitar algumas classes do Java

Arquitetura de *software* e padrão de projetos



Arquitetura multicamadas

Uma arquitetura multicamadas é qualquer arquitetura que tenha mais de uma camada



Modelo de 2 camadas Cliente Servidor	Modelo de 3 camadas Apresentação, domínio e dados

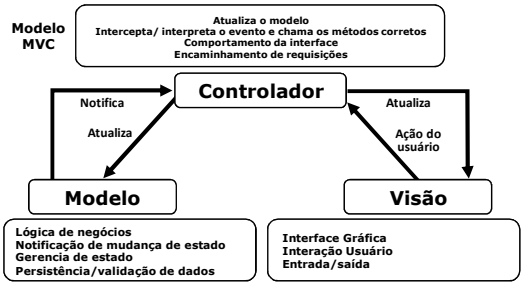
Centurion Studio/Shutterstock

TechnoVectors/Shutterstock

Camada física x camada lógica

Tier (Camada)	*Layer* (nível)
▪ Execução em infraestrutura separada ▪ Implementação concreta do sistema ▪ Como os componentes são distribuídos fisicamente ▪ Infraestrutura real que suporta a execução da aplicação	▪ Representação da estrutura e organização do sistema com base em uma perspectiva funcional e conceitual ▪ Relação com: • Lógica de negócios • Serviços • Funcionalidades da aplicação

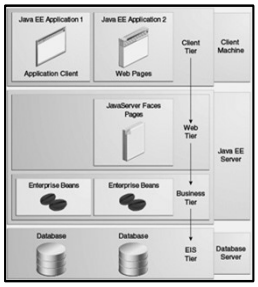
Arquiteturais	Estruturais	Comportamentais
▪ Organização global ▪ Estrutura fundamental ▪ Divisão de responsabilidades ▪ Comunicação entre componentes ▪ Visão geral da arquitetura	Organização interna de classes e objetos dentro de um sistema Como as partes individuais se relacionam/ compõem para formar uma estrutura funcional	▪ Padrões de interação e responsabilidades entre objetos ▪ Como os objetos colaboram e interagem para distribuir responsabilidades
MVC	*Composite*	*Strategy* e *Observer*



Arquitetura Java EE

Java EE

- **4 camadas lógicas: camada do cliente, web, de negócios, de sistema de informação corporativa**
- **Considerada aplicação de 3 camadas: máquinas clientes, servidor Java EE e BD ou máquinas legadas no *back end***

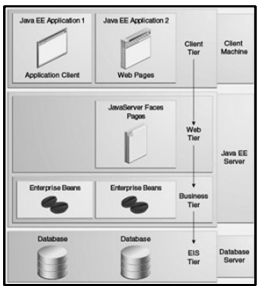


13/37

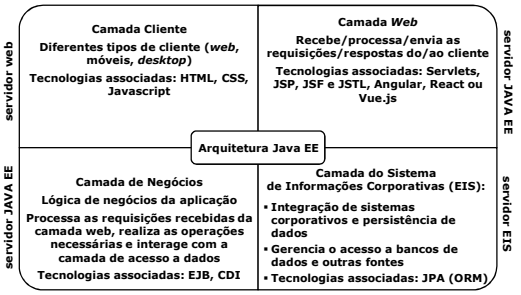
Java EE

- **Aplicação Java EE deve conter ao menos três camadas lógicas: apresentação, domínio e fonte de dados**

14/37



15/37



16/37

Tecnologias Java EE

17/37

Tecnologias Java EE

- **Especificações**
- **APIs**
- **Tecnologias**

Componentes e serviços

18/37

*** API: Interfaces de Programação de Aplicações**

Componentes

- Clientes aplicativos e mini aplicativos: desenvolvido em Java e executado no lado do cliente
- Enterprise JavaBeans (EJB): representam a lógica de negócios
- Três tipos principais: EJB de Sessão, EJB de Entidade e EJB de Mensagem

Componentes

- *Servlets* ("servidorzinhos"):
  - Classes java
  - Estendem a funcionalidade de servidores *web*
  - Criação de aplicativos *web* dinâmicos
- JSP: incorporação de código Java em páginas HTML
- Componentes da *web* executados no servidor

Componentes

- *Managed Beans* (CDI)
  - Componentes gerenciados pelo CDI (*Contexts and Dependency Injection*)
  - Armazenam e gerenciam estados

POJO

```
public class Pessoa {
     private String nome;
     private int idade;
//Construtores
   public Pessoa(){
        //Construtor padrão
    }
    public Pessoa(String nome, int idade){
         this.nome = nome;
         this.idade = idade;
    }
// Métodos de Acesso (Getters e Setters)
   public String getNome(){
         return nome;
    }
   public void setNome(String nome){
         this.nome = nome;
    }
    public int getIdade(){
         return idade;
    }
    public void setIdade(int idade){
         this.idade = idade;}
}
```


Serviços e bibliotecas

- Java Persistence API (JPA):
  - API para ORM
  - Facilita a persistência de dados
  - Fornece um *framework* ORM
- Java Message Service (JMS):
  - API para comunicação assíncrona
  - Envio e recebimento de mensagens

Serviços e bibliotecas

- JavaServer Faces (JSF):
  - *Framework* para desenvolvimento de interfaces *web* baseadas em componentes
  - Simplifica a construção de aplicações *web*
  - Conjunto de componentes visuais reutilizáveis
  - Modelo de programação baseado em eventos

Eclipse Jakarta EE

- Eclipse Jakarta EE é uma plataforma de desenvolvimento para construção de aplicações empresariais em Java

Java EE até sua evolução para Jakarta EE

1999
J2EE

O Java EE surgiu como extensão do Java SE

2006
Java EE

a partir da versão 5.0
melhorias, novas tecnologias e atualizações

2017
Jakarta EE

Transferência de tecnologias para organização de código aberto
Eclipse Foundation

2022
Mudanças Visíveis ao desenvolvedor

Considerações práticas para o programador Java

- Migração gradual
- Governança aberta e código aberto
- Mudanças visíveis em 2022
- Necessidade de atualização

Revisitando o Java básico

Classe Object

- Toda classe é subclasse de Object
- toString()
  - Retorna uma *string* representando o objeto
- equals(Object obj)
  - Compara dois Objects
- hashCode()
  - Retorna um código *hash* para o objeto

Classe System

- Propriedades do sistema, ambiente de execução e manipulação de fluxos E/S
- Métodos e variáveis estáticos
  - Fluxos de entrada: ler dados
  - Fluxos de saída: impressão na tela
- Exemplo:
  - System.out.print("Olá Mundo!");

Classe Scanner

- Ler dados: entrada do teclado, arquivos ou *strings*
- Métodos para analisar e converter *tokens* (sequências de caracteres) em tipos primitivos
- *Scanner* de texto simples: analisa tipos primitivos e Strings
- Exemplo:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

Classe String

- Representa cadeias de caracteres
- Literais de string são implementados como instâncias desta classe → Ex: "abc"
- Strings são constantes
- Exemplo: `String str = "abc";`

Equivalentes

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Classe Math

- Métodos estáticos
- Métodos:
  - Operações numéricas básicas
  - Funções exponenciais elementares
  - Logaritmos
  - Raízes quadradas e trigonométricas
- Exemplo:

```
double d = 4;
double i = Math.sqrt(d);
```

Classe Wrapper

- Manipulação de valores dos tipos da linguagem como se fossem objetos
- Frequentemente: necessário representar valor de tipo primitivo como objeto

```
String itxt = "123";
int i = Integer.parseInt(itxt);
System.out.println(i + "\n ");
```

Tipo Primitivo	Classe Wrapper
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Collection Frameworks

- Coleção: objeto que representa um grupo de objetos
- Estrutura de coleções: arquitetura unificada para representar e manipular coleções
- Manipulação independente dos detalhes de implementação
- Classes e interfaces: listas, conjuntos e mapas

Tabela contendo as interfaces e as classes de implementação

Interface	Hash Table	Resizible Array	Balanced Tree	Linked List	Hash Table + Linked List
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Deque		ArayDeque		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

Interfaces

- **Collection: interface base. Inclui métodos comuns: add, remove, size**
- **List: coleção ordenada, elementos acessados por índice**
- **Set: não permite elementos duplicados**
- **Queue: FIFO (First-In-First-Out)**
- **Deque: utilizado como uma fila (FIFO) ou uma pilha (LIFO). Queue + operações adicionais**
- **Map: associação de chaves a valores únicos**