

Softworks:

PHP OO: Módulo 1

- mais de 80% da internet é PHP
- arquitetura CLI e WEB
- print_r, var_dump para debugar
- atribuição de endereço: \$b = &\$a; (por referência)
- o objeto é sempre passado por referência

Ex: \$pessoa1 = meu_stdClass;

\$pessoa1 -> nome = 'Pedro'; }
\$pessoa2 = \$pessoa1; } pessoa1 e pessoa2 serão juntas
\$pessoa2 -> nome = 'joão'; } (pessoa2 referencia pessoa1)

- se eu colocar, por exemplo, float na frente de parâmetro de função, ele converte (ou chamar função com variável tipada (float))

Ex: function inc (float \$peso, float \$altura) : int

↳ força retorno int

- ativar tipagem estrita: declare (strict_types = 1);

- tipos onuláveis: → ex: string \$label = null ↪

Ex: function link (?string \$label) : string {

 \$label = \$label ?? 'Clique aqui';
 return "\$label";

parâmetro obrigatório
único

}

print link(null); → ex: print link(); ↪

- união de tipos: git

- argumentos nomeados: git

- parse_ini_file lê arquivo e retorna array

- parâmetro object aceita qualquer objeto de uma classe (\$conf -> attrib)

- superglobais: \$_SERVER, \$_GET, \$_POST, \$_REQUEST

- constante global: define ('NUM', 5)
- constante mágica --FILE--, --LINE--, --DIR--
- \$ teste = \$ valor ++ ≠ \$ teste = ++\$ valor;
- var_dump(\$valor); var_dump(\$teste);
- ==, != (comparação e comparação com tipagem)
- foreach (\$lista as \$fruta)
 - print \$fruta;
- match expressions: git
- incluir arquivo: include, require, include_once, require_once 'php'
- "" string e '' string literal (não printa conteúdo variável)
- funções str: strtoupper, strtolower, strlen, substr, str_replace
 - * protected não precisa declarar nem usar variáveis classe pai

Módulo 2 * static mantém seu valor mesmo após chamadas de métodos diferentes

- classe, objeto, método, atributo
- construtor é onde o atributo vai inicialmente (sem chamar método)

Agregação: TODO - PARTE (foco) : a parte pode viver sem o todo (costa)

Composição: TODO - PARTE (frente) : a parte depende do todo para existir (coroa)

* static: acesso método sem instanciar (uso self :: em vez de this →)

- protected permite acesso atributo da classe pai (precisa ser classe filha)
- interface é uma lista de métodos obrigatórios

Padrões de Projetos:

- Singleton:
 - construtor privado
 - só pode instanciar dentro da própria classe
 - cria um único objeto e sempre retorna ele.
 - não permite mudar ou gravar outra coisa depois de instanciado

→ Facade:

- oferece interface única para um conjunto de interfaces do subsistema

- Adapter (também conhecido como Wrapper)
- oferece novos métodos à uma classe já existente.
- Object Wrapper: encapsula adaptado por composição
- Class Wrapper adota interface por herança

Módulo 3:

- instalar php-pgsql, create table, setor user, setor senha

Problemas programa N1

- precisa criar muitos arquivos
- HTML misturado com PHP
- muitas conexões ao BD
- difícil manutenção e segurança

N2:

- usando action p/diminuir número de arquivos
- mesmos problemas N1

Problemas N3:

- str_replace é rudimentar, melhor usar templates (HTML separado)

N4:

- usando funções p/exibir comandos SQL e facilitar manutenção
- técnica rudimentar, melhor usar classes

N5:

- se usar try-catch, sempre que der erro, ele irá direto p/catch (obj BD)
- trocando funções do BD por Classes
- uso de métodos estáticos para manter valores

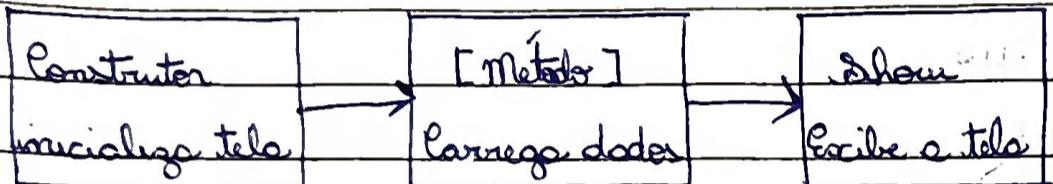
/ /

N6:

- criação de função estática getConnection()
- criação cfg BD (.ini e apache 2.conf)
- não permitir SQL injection (usando prepare())

N7

- refatorar código para usar classes
- centralizar tudo no index, onde é passado na URL classe e método



= index.php ? class=Form & method=edit & id=2

Módulo 4:

- throw e try catch (Exception)
 - simplexml_load_file('países.xml'); objeto com as variações (tag)
 - processando tags filhos: \$xml → geografia → clima;
 - simplexml mais simples que domxml
 - do p/ colocar dados no tag atribuindo normalmente, p/ gravar no arquivo: file_put_contents('países.xml', \$xml->asXML());
 - se é uma lista com tags de mesmo valor, vira: um vetor
 - processando atributo da tag <estado nome="...">
- ```

foreach ($xml->estados->estado as $estado) {
 $estado['nome'];
 $estado['capital'];
 foreach ($estado->atributes() or $key => $value) // percorre
 print "$key: $value";
}

```

- leitura domxml: \$base->getElementsByTagName; \$names->item(0)->nodeValue
- escrita domxml: \$t

FORONI

Seg Ter Qua Qui Sex Sáb Dom

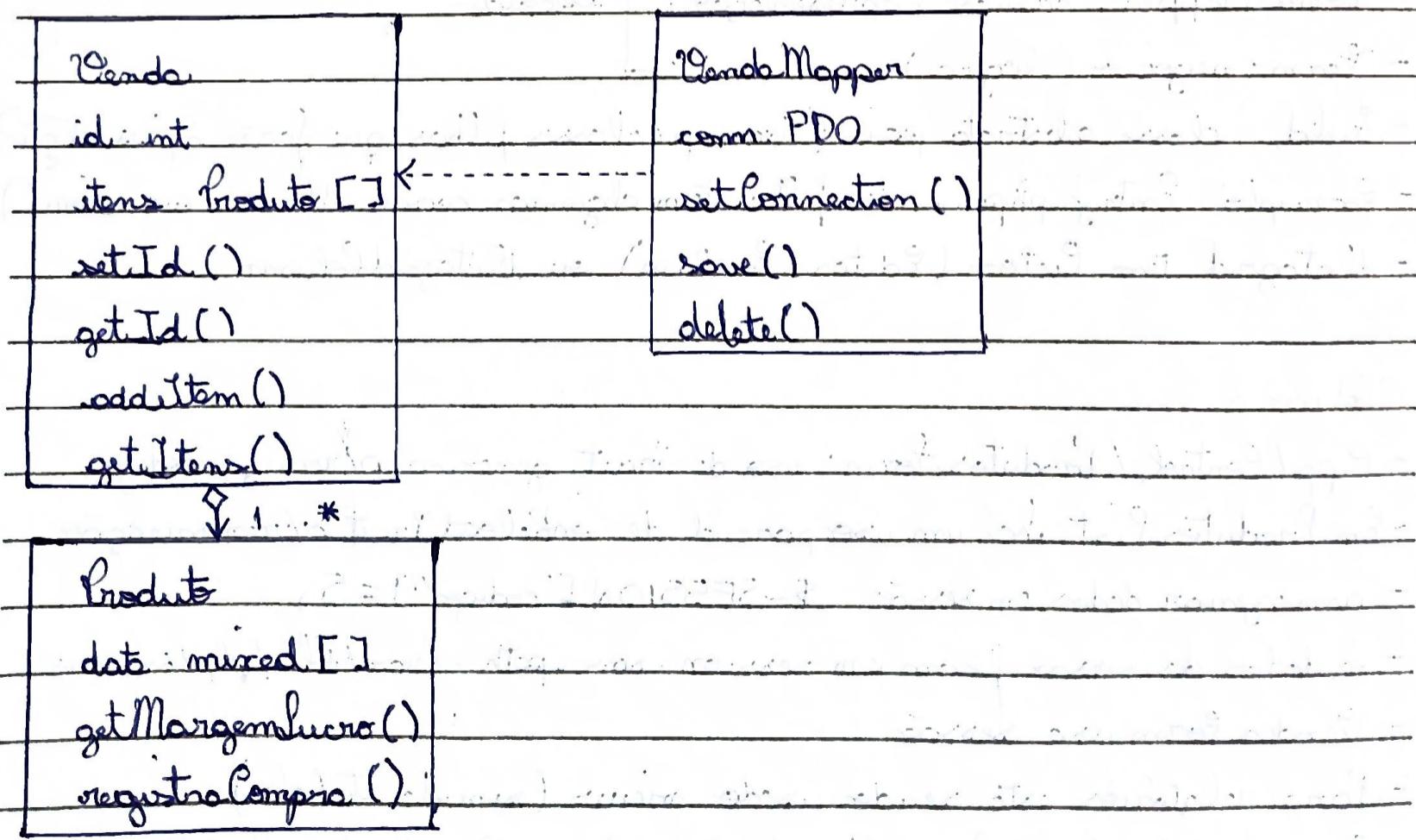
111

- SPL: pillars, filos, etc.
- Reflection: investigar classes
- Traits: copia métodos para uma classe
- Injeção de dependência: passa a classe como parâmetro para interface para garantir métodos
- PSR (PHP Standard Recommendation)
- namespace: isola classes permitindo que tenha mesmo nomes
- composer: gerenciador de dependências

## Módulo 5:

### → Padrões de Projeto de Persistência:

- Gateway: classe que reúne operações de persistência
- Active Record: mistura regras de persistência e negócio
- Late Mapper: classe de persistência que grava uma classe e todo seu tipo de relacionamento



- Classe para conexões usando Factory
- Transaction, método ACID, só grava se tudo ocorrer certo
- Log: arquivo com registro das transações
- Picture Record como classe pai
- Picture Record: entidade sabe se salvar
- Repository: entidade pura, outro objeto salva
- read\_table.sql

### Módulo 6:

- Padrão MVC para projetos API REST
- Componentes X Templates (Twig)
- componente usa classe estruturada (Widgets)
- template usa molde de .html pronto

### Módulo 7:

- FormWrapper: visual (apresentação) ] decorador
- Form: negócio (lógica) ]
- Field: classe abstrata pai (base p/ classes filhas que fazem apresentação)
- Exemplo: Entry.php: classe filha Item tem algumas coisas alteradas no show()
- Datagrid tem Action (Editor / Excluir) ou DatagridColumn

### Módulo 8:

- Cpp / Control / ProdutosForm: uso de Trait genérico p/redigir código
- Em ProdutosList há um uso parcial de onReloadTrait c/ renomeação
- armazenar dados em sessão: \$SESSION['codigo']=5;
- os dados da sessão ficam em session-save-path: /var/lib/php/session
- Grandes form usa sessão
- Para relatórios está sendo usado views (simula tabela)
- Exemplo relatório: PessoaReport.php, ContaReport.php, ProdutoReport.php
- Criando gráficos com Chart.js nos templates em Resources

- Exemplo gráfico tratamento de dados em Venda.php, VendalesChart.php
- o nome no javascript serve para mandar o vetor com os [ ] ou ""
- usando vetores indexados para os gráficos
- simulando login em index-login.php

### Módulo Extra:

- geração PDFs com fpdf
- geração RTFs com phptable
- geração XLSs com xlsdocumentgenerator
- fpdf - pede uso classe para deixar mais estruturado
- rtf - corte uso classe também
- criação blog simples
- corrigir template e colocar {{base url}} na frente das ja e redirecionamentos (img, etc), trocar index.html → index.php, usar twig para trocar posts estáticos pelo que está no bd ({{post.title}}), etc
- {{content}}