PROGRAMA DE DESENVOLVIMENTO DE NOVOS TALENTOS – IMPLY
3ª edição

DESAFIOS II – PHP OO

Prazo de entrega: até às 8h de 02/02
Entregar pela Sala Virtual

Apresentação dos Desafios: 02/02

1 – Utilizando o arquivo GenerateCsv.php disponível na sala virtual, vocês devem gerar um arquivo .csv (com cabeçalhos e uma linha de espaço entre eles e os dados) unindo os dados gerados pela execução do arquivo .php em um único arquivo csv utilizando programação orientada a objetos.

Esses conteúdos estarão organizados em dois arquivos, products.csv (com ID do produto, nome e preço) e orders.csv (com ID do pedido , ID do produto, data da venda e quantidade vendida). O arquivo final deverá apresentar as seguintes colunas:

ID do produto, preço unitário, data da última venda, quantidade total vendida e valor total vendido

Após gerado, o arquivo deverá ser enviado por email.

Observações importantes:

- O mesmo produto pode aparecer em mais de um pedido, logo, suas datas de venda devem ser comparadas e somente a maior deve ser mantida e suas quantidades vendidas devem ser somadas
- O composer (pesquisar sobre o que é e como funciona) pode facilitar e muito a questão de enviar por email e até a geração dos arquivos, embora seja aconselhado gerar os arquivos utilizando funções nativas do PHP.
- A organização do projeto será avaliada.

2 - Utilizando Orientação a Objeto e banco de dados Postgresql.

- Criar um banco de dados com nome de sua preferência, e depois criar a tabela “funcionarios” conforme SQL informado abaixo.

```
CREATE TABLE public.funcionarios (
    id serial4 NOT NULL,
    nome varchar(100) NULL,
    genero varchar(10) NULL,
    idade int4 NULL,
    salario numeric(10, 2) NULL,
    CONSTRAINT funcionarios_pkey PRIMARY KEY (id)
);
```

- Criar uma classe de Conexão com o banco de dados e reutilizá-la dentro da classe Funcionário.

- Criar uma classe “Funcionario” onde deve possuir os atributos conforme os campos criados no banco da tabela “funcionarios”.
    - Também possuir funções que realizar o CRUD(Create, Read, Update e Delete) do funcionário no banco.
    - Nas functions de leitura ao banco possuir uma de listar todos os funcionários do banco e outra listando e carregando o objeto de somente um funcionário de acordo com ID informado.
    - Realizar validações nos atributos antes de chamar o banco em algumas funções.
    - Criar uma função que aumente o salário do funcionário passando um percentual.

        Ex: salario = 1000
        percentual aplicado = 10%
        após aumento = 1100

- Criar um arquivo index.php para rodar o código com as classes nessas ordens
    - Realizar 4 cadastros de funcionários no banco, através dos objetos de Funcionários.
    - Desses 4 funcionários alterar seus nomes e aumentar seus salários, salvar alterações no banco em seguida.
    - Realizar a listagem dos funcionários do banco e exibir no terminal.
    - Salvar o ID de um desses funcionários em uma variável, realizar o unset() do objeto dele. Instanciar um novo objeto de funcionário vazio e carregar o objeto com os dados do banco desse ID salvo na variável exibindo no terminal seus dados dos atributos.
    - Realizar a exclusão no banco de um desses funcionários e listar os funcionários do banco no terminal.

Obs:

- Só deve ser alterado os dados da tabela “funcionarios” utilizando php e as functions dentro da classe Funcionário.
- A criação da database e tabela “funcionarios” pode ser feito utilizando uma ferramenta de banco de dados.Ex: Dbeaver, PgAdmin, etc…

3 - Utilizando orientação a objetos, vamos abordar um conceito da programação chamado de refatoração. Para isso, utilizando como base o gabarito dos desafios 2 e 3 da primeira lista de desafios, deve ser feitas as seguintes melhorias:

- Deve ser criado um objeto que represente a requisição feita para a API. Ele deve conter todas as informações necessárias para que qualquer requisição que chegue à API possa ser processada.
- Criar uma classe de controlador. Essa classe deve conter um método para cada desafio e vai fazer o echo da resposta do processamento.
- Criar um objeto responsável por fazer as requisições à API utilizando cURL.
- Criar um objeto Roteador, que recebe a rota desejada pela requisição e faz o redirecionamento adequado para o método do controlador.
- O funcionamento da API deve permanecer **EXATAMENTE** o mesmo, dessa forma, todos os requisitos da primeira lista se aplicam neste desafio novamente.