08

Faça o que eu fiz na aula

Vamos fazer a validação do e-mail, para isso, vamos criar uma função:

```
private function validaEmail(string $email)
{

}
```

Dentro dessa função, vamos usar a função `filter_var` do PHP para validar se a string passada é um e-mail:

```
return filter_var($email, FILTER_VALIDATE_EMAIL);
```

Agora podemos chamar esta função no construtor da classe `Contato`, fazendo uma condicional para erro e sucesso da validação.

```
if ($this->validaEmail($email) !== false)) {
    $this->setEmail($email);
} else {
    $this->setEmail("Email inválido");
}
```

Para definirmos esta função `setEmail` que utilizamos, vamos criar dentro da classe `Contato`:

```
private function setEmail(string $email)
{
    $this->email = $email;
}
```

E vamos exibir agora o e-mail no arquivo `cadastro.php`:

```
<li class="list-group-item">Email: <?php echo $contato->getEmail(); ?></li class="list-group-item">
```

E se recarregarmos a página, o e-mail agora será exibido na tela de cadastro!

Para exibirmos o endereço, precisamos adicionar ele e o cep no objeto:

```
$contato = new Contato($_POST['email'], $_POST['endereco'], $_POST['cep']);
```

E na classe `Contato`, vamos adicionar as propriedades e os dois argumentos no construtor:

```
private $endereco;
private $cep;
```

```
public function __construct(string $email, string $endereco, string $cep)
```

Agora para fazer a exibição, vamos criar aqui uma função:

```
public function getEnderecoCep(): string
{

}
```

E dentro desta função, vamos utilizar a função `implode` para juntar o endereço com o cep, unidos pelo caractere " - ", só que para isso, temos que primeiro transformar o endereço e o cep em um array:

```
$enderecoCep = [$this->endereco, $this->cep];
return implode(" - ", $enderecoCep);
```

Agora podemos chamar o método no HTML:

```
<li class="list-group-item">Endereço: <?php echo $contato->getEnderecoCep(); ?></li class="list-grou
```

E se recarregarmos a página, o endereço será exibido junto com o CEP!

Ainda falta implementarmos a senha! para impedir que a pessoa cadastre uma senha muito fácil de ser adivinhada, é uma boa prática inserir um número mínimo de caracteres para ela.

Então, para implementarmos a senha, usaremos algo bem parecido com o que temos feito, vamos adicionar a variável `$_POST['senha']` ao construtor da classe Usuario

```
$usuario = new Usuario($_POST['nome'], $_POST['senha']);
```

Agora no construtor da classe, vamos adicionar este argumento.

E vamos criar um método para fazermos a validação o tamanho da senha

```
    private function validaSenha(string $senha): void
    {

    }
```

Mas como podemos verificar o tamanho da senha? existe um método de string que chama strlen, podemos utilizá-la para pegar o tamanho de uma string e verificar se ela tem mais do que 6 caracteres:

Também utilizaremos a função trim antes para remover espaços do começo e do final da string:

```
    private function validaSenha(string $senha): void
    {
        $tamanhoSenha = strlen(trim($senha));
```

```
    if ($tamanhoSenha >= 6) {
        $this->setSenha($senha);
    } else {
        $this->setSenha(“Senha inválida”);
    }
    }
```

E vamos criar a função `setSenha` que utilizamos:

```
private function setSenha(string $senha)
{
    $this->senha = $senha;
}
```