Guia de Implementação

SUMÁRIO

1. INTRODUÇÃO

O Guia de Desenvolvimento do projeto QuaEPI tem por finalidade padronizar a codificação do sistema, referencia como devem ser declaradas variáveis, métodos, etc.

2. ORGANIZAÇÃO E ESTILO DO CÓDIGO

Devido a linguagem utilizada, que é PHP, não há restrição para tamanho de classe e métodos, sendo utilizada a quantidade de linhas necessárias, somente se fazendo necessária a separação de métodos com uma quebra de linha. Além disso, as linhas não devem possuir mais de 80 caracteres sem quebra de linha.

3. COMENTÁRIOS

Os comentários explicativos estarão presentes nas classes controladoras e de visão, pois, devido ao uso de um Mapeador Objeto-Relacional, as classes modelo conterão comentários específicos para a montagem da entidade na base de dados. Os comentários para classes e métodos estão descritos a seguir:

Classes controladoras e de visão

```
/**
 * <Descrição da funcionalidade da Classe>
*/
```

```
/**
 * <Descrição completa das funcionalidades do método>
 * @return <tipo do retorno do método: String, int, etc>
 * @param <parâmetro1> <Descrição do parâmetro>
 * @param <parâmetro2> <Descrição do parâmetro>
*/
```

4. NOMEAÇÃO

Arquivo PHP

- NomeClasse.php

Scripts SQL

- NOMESCRIPT.sql

Arquivo Javascript

- NomeArquivo.js

5. DECLARAÇÃO

1. Padrão de início e fim de arquivo PHP:

   <?php

```
...

?>
```

2. Padrão de indentação de declaração de métodos:

```
metodoTal($argumento) {
        ...
}
```

3. Padrão de indentação de declaração de classes controladoras:

```
class ClasseController extends CI_Controller {
        ...
}
```

4. Padrão de indentação de declaração de classes de modelo:

```
class Classe {
        ...
}
```

5. Padrão de declaração de variáveis em classes:

```
private $nomeVariavel;
```

6. TESTE UNITÁRIO

- Devido ao tempo hábil, foram realizados diversos ciclos de testes, conforme Roteiro de Testes, com entradas pré-definidas e resultados esperados.

7. DIRETRIZES GERAIS

No desenvolvimento do projeto, serão utilizadas duas ferramentas, sendo: CodeIgniter (framework para PHP) e Doctrine, que é um Mapeador Objeto-Relacional. Devido a este fato, as classes controladoras estendem o CI_Controller, que é uma classe padrão do CodeIgniter, que facilitará o desenvolvimento devido a métodos específicos.

Para não atrapalhar a funcionalidade do Doctrine, não será utilizada a classe nativa chamada de CI_Model, que também possui métodos próprios.