

Mapeamento de casos de uso em classes de PHP

Metodologia padronizada para desenvolvimento de projetos Web em PHP

<http://www.ManuelLemos.net/>

Manuel Lemos

mlemos@acm.org

1º PHPDF Road Show

Brasília, 19 de Maio de 2007



Licença Creative Commons

Atribuição-Não a obras Derivadas 2.5



Você pode:

- copiar, distribuir, exibir e executar a obra
- fazer uso comercial da obra

Sob as seguintes condições:



Atribuição. Você deve dar crédito ao autor original, da forma especificada por **Manuel Lemos**.



Vedada a Criação de Obras Derivadas. Você não pode alterar, transformar ou criar outra obra com base nesta.

- Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.
- Qualquer uma destas condições podem ser renunciadas, desde que Você obtenha permissão de **Manuel Lemos**.

Qualquer direito de uso legítimo (ou "fair use") concedido por lei, ou qualquer outro direito protegido pela legislação local, não são em hipótese alguma afetados pelo disposto acima.

Este é um sumário para leigos da [Licença Jurídica \(na íntegra\)](#).

Creative Commons License Deed

Attribution-No Derivs 2.5



You are free:

- to **Share** -- to copy, distribute, display, and perform the work
- to make commercial use of the work

Under the following conditions:



Attribution. You must attribute the work in the manner specified by **Manuel Lemos**.



No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from **Manuel Lemos**.

Your fair use and other rights are in no way affected by the above.
This is a human-readable summary of the [Legal Code \(the full license\)](#).

Metodologia de desenvolvimento de projetos

RUP - Rational Unified Process

Desenvolvimento iterativo em 3 fases:

1. Análise

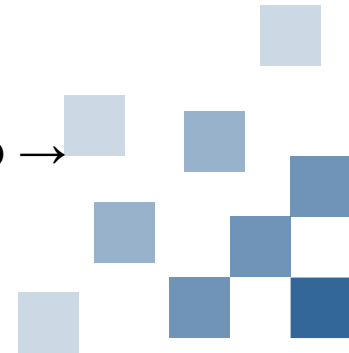
Requisitos, riscos → especificação

2. Planejamento

Sistemas, sub-sistemas, casos de uso → diagramas UML

3. Execução

Implementação, teste, correção → documentação para usuário →



Intervenientes no projeto

Stakeholders

- **Solicitador**

Solicita o projeto, define os objetivos, aprova o orçamento e a implementação: cliente ou gerente de outra área

- **Patrocinador**

Garante os recursos para progresso do projeto: diretor de tecnologia

- **Coordenador**

Avalia os requisitos, planeja a implementação, faz o orçamento, supervisiona a implementação: gerente de desenvolvimento

- **Analistas**

Implementam, testam e documentam o projeto →



Análise do projeto

- **Levantamento de requisitos**

Que funcionalidades? Metas? Prazos?

- **Análise dos riscos**

O que pode não falhar? Quais os parâmetros de sucesso?

- **Documento de especificação**

Resumo dos requisitos, riscos, prioridade de implementação

- **Intervenientes**

Solicitador, patrocinador, coordenador →



Planejamento do projeto

- **Definição de escopo**
O que faz ou não faz parte do escopo do projeto
- **Divisão do projeto**
Que sistemas e sub-sistemas devem ser implementados
- **Planejamento das iterações**
Divisão do projeto em fases e definição dos casos de uso implementados em cada fase
- **Especificação de casos de uso**
Que casos de uso deve implementar cada sistema
- **Documentação do planejamento**
Documentos e diagramas UML: casos de uso, de sequência, etc..
- **Intervenientes**
Coordenador, analistas →



Implementação do projeto

- **Modelagem**
Modelagem de componentes, classes de objetos, estruturas de dados
- **Codificação**
Escrita do código que implementa os componentes modelados
- **Homologação**
Teste e verificação das funcionalidades de acordo com a especificação
- **Documentação para o usuário**
Documentação de como usar as funcionalidades implementadas
- **Intervenientes**
Analistas, coordenador, solicitador →



Do planejamento para execução

- **Dividir para conquistar**

Sistemas, sub-sistemas, casos de uso

- **Implementação de casos de uso**

Desenvolvimento de componentes para:

→ Implementar regras de negócio

→ Gerar a interface com o usuário

→ Acessar sistemas externos

→ Acessar e armazenar dados persistentes →



Casos de uso

Situações que o sistema irá processar – as telas

- **Descrição da situação e atores envolvidos**
- **Pré-condições**
- **Pós-condições**
- **Ativação**
- **Fluxo normal de eventos**
- **Fluxos alternativos de eventos**
- **Situações excepcionais previstas**

[Exemplo de descrição de casos de uso](#)



Implementação de casos de uso de uma aplicação Web

- **Script de aplicação Web**

Recebe o pedido HTTP e retorna a resposta

- **Componente de regras de negócio**

Implementa as situações previstas na especificação do caso de uso →



Script de aplicação Web

- Carrega os componentes necessários
- Invoca o componente de regras de negócio
- Processa erros não previstos no caso de uso

Falhas de acesso a banco de dados ou a sistemas externos, etc..

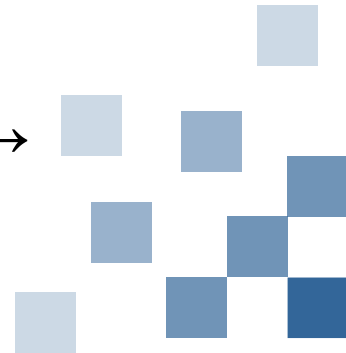
[Exemplo de script Web](#)



Componente de regras de negócio

Implementação do caso de uso

- Verifica as pré-condições
- Implementa os fluxos de eventos normal e alternativos
- Assegura a satisfação das pós-condições
- Processa situações excepcionais previstas
- Implementado por uma classe de caso de uso →



Classe de caso de uso

Estrutura da classe

- **Variáveis públicas para configuração e retorno de erros**
- **Classes, variáveis e funções privadas auxiliares**
- **Funções públicas separando processamento dos resultados**
 - Inicialização
 - Processamento
 - Geração de resultados →



Inicialização do caso de uso

- **Ocorre antes da ação propriamente dita**
- **Inicialização de variáveis e objetos auxiliares**
- **Acesso a dados persistentes necessários**
- **Verificação das pré-condições**
- **Pode falhar em caso de erro não previsto →**



Processamento do caso de uso

- Implementa as ações associadas aos fluxos de eventos
- Nada faz se as pré-condições não forem satisfeitas
- Trata de situações excepcionais previstas
- Assegura as pós-condições
- Prepara resultados a apresentar usando variáveis privadas
- Não exhibe resultados a menos que isso faça parte do processamento, exceto em casos de *streaming* ou **AJAX**
- Pode falhar em caso de erro não previsto →



Geração de resultados do caso de uso

- **Agrega resultados obtidos durante o processamento**
- **Gera a saída do script Web**
- **Nunca falha →**



Esqueleto de uma classe de caso de uso

```
class caso_de_uso_esqueleto
{
    /* Variáveis públicas */
    var $erro;

    /* Variáveis privadas */
    var $pre_condicoes_verificadas = false;
    var $saida = "";

    /* Funções privadas aqui */

    /* Funções públicas */
    Function Inicializar()
    {
        $this->pre_condicoes_verificadas =
            $verificar_pre_condicoes;
        if($se_ocorreu_alguns_erro)
        {
            $this->erro = "mensagem de erro";
            return false;
        }
        return true;
    }
}
```

```
Function Processar()
{
    if($this->pre_condicoes_verificadas
        == false)
        return true;

    /* Executar tarefas de processamento */
    $this->saida = $dados_da_saida_preparada;

    if($se_ocorreu_alguns_erro)
    {
        $this->erro = "mensagem de erro";
        return false;
    }
    return true;
}

Function GerarResultados()
{
    echo $this->saida;
}
};
```

Exemplo de classe de caso de uso

Tratamento de erros não previstos

Erros durante a inicialização ou processamento

- Implementação com uma classe de caso de uso especial
- Invocação no final do script Web em caso de erro
- Registro de ocorrências (geração de arquivo de log)
- Notificação dos responsáveis pelo sistema (email, pager, messenger)
- Mostrar mensagem amigável sem expor detalhes dos erros →

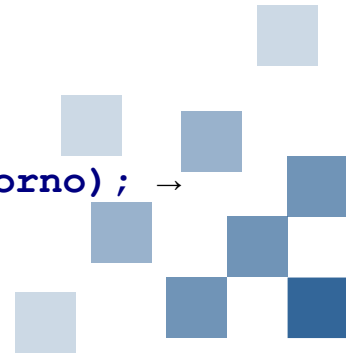


Pré-condições complexas

- **Pré-condições satisfeitas por outros casos de uso**
Por exemplo: usuário tem de estar cadastrado e identificado
- **Verificação de pré-condições na função de inicialização**
- **Invocação de classe de caso de uso separada**
- **Redirecionamento ou link para os respectivos scripts Web**
- **Parâmetro de retorno aponta para script Web atual**
- **Exemplo:**

```
$retorno='options.php';
```

```
Header('Location: http://www.site.com/login.php?volta='.$retorno); →
```



Desenvolvimento e implantação

- **Repositório com todos arquivos do projeto**

Controle de versões de arquivos de projeto com CVS, SubVersion, etc..

- **Ambientes de desenvolvimento, teste e produção**

Mesmo projeto, diferentes instalações, configurações ajustadas

- **Ambiente de administração**

Atualização e controle dos ambientes de teste e produção →



Arquivos de configuração

- **Definidos através de classes de PHP**
Carregam com `require()`, beneficiam de *cache* de compilação de PHP
- **Valores iniciais para ambiente de produção**
- **Valores críticos ficam indefinidos por segurança**
Senhas de acesso a banco de dados e outros
- **Script local altera valores iniciais a partir da função de inicialização**
Definição de valores críticos e alterações para ambiente de desenvolvimento

[Exemplo de classe de configuração de opções](#)

[Exemplo de script de opções locais](#)



Opções de configuração importantes

- `application_path`

Caminho da aplicação, inicializada pelos *scripts Web*

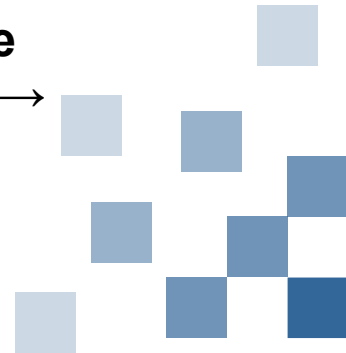
- `database_connection`

Conexão ao banco de dados, todos detalhes num URL, por exemplo:

`mysql://usuario:senha@servidor/banco`

- `debug`

Geração de informação para depuração, ligada em ambiente de desenvolvimento e teste, desligada em ambiente de produção →



Controle de versões de arquivos dos projetos

O que é? Para que serve?

- Manter o histórico de todas as alterações feitas aos arquivos do projeto
- Saber quem alterou que arquivos, quando e porquê
- Poder voltar a versões passadas do projeto
- Sincronização de versões entre os ambientes de desenvolvimento, teste, e produção dum projeto
- Programas de controle de versões mais conhecidos:
CVS, SubVersion, SourceSafe →



Regras de uso do repositório CVS

- Todos arquivos adicionados devem ter descrições
- Arquivos binários devem ser marcados como tal
- Todas as alterações devem ser descritas
- Arquivos de código devem respeitar o estilo original

Não reformate o código dos outros sem permissão

- Modificações nos arquivos devem incluir apenas alterações funcionais

Não altere código que não corrige nenhuma funcionalidade →



Estrutura de diretórios

- /caminho/da/aplicação/ - diretório de instalação**
- ↳ **usecases/ - classes de implementação de casos de uso**
- ↳ **web/ - scripts das páginas acessíveis pela Web**
 - ↳ **graphics/ - arquivos de imagens estáticas usadas no site**
 - ↳ **css/ - arquivos CSS usados no site**
- ↳ **templates/ - arquivos e classes definindo aspectos da apresentação**
- ↳ **components/ - classes prontas para fins diversos**
- ↳ **configuration/ - arquivos de configuração**
- ↳ **locale/ - arquivos de texto e configuração para uso em diferentes países**
- ↳ **setup/ - scripts de instalação**
- ↳ **backend/ - scripts de manutenção**
- ↳ **logs/ - arquivos de registro de ocorrências →**



Bibliografia recomendada

- **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML**
José Carlos Cordeiro
ISBN: 8574522112
- **Applying Use Cases: A Practical Guide**
Geri Schneider, Jason P. Winters, Ivar Jacobson
ISBN: 0201309815
- **UML Distilled: A Brief Guide to the Standard Object Modeling Language**
Martin Fowler, Kendall Scott
ISBN: 020165783X
- **Metastorage example application implemented with Use Case mapping**
Manuel Lemos
<http://www.meta-language.net/metastorage-example.html> →



Fim

Obrigado pela atenção

Questões?

