



**FEUP**

FACULDADE DE ENGENHARIA DA  
UNIVERSIDADE DO PORTO

SIEM 2016/2017

---

## Relatório

### Trabalho Prático 3

---

Luís Filipe Fernandes Costa MELO

201206020

Lidia Miguel Barros CERQUEIRA

201205960

*Professor:*

André Monteiro de Oliveira RESTIVO

José António Rodrigues Pereira de FARIA

Janeiro 2017



# 1 Arquitetura de *Software*

Para ambos os trabalhos, foi adotada uma arquitetura baseada em camadas, frequentemente designada por *Three-tier architecture*. As camadas de software consideradas foram:

- **Presentation Layer:** encarregue da parte da apresentação da informação e design
- **Business Layer:** essencialmente responsável por toda a parte da lógica do negócio
- **Database Layer:** constituída pelas chamadas à base de dados

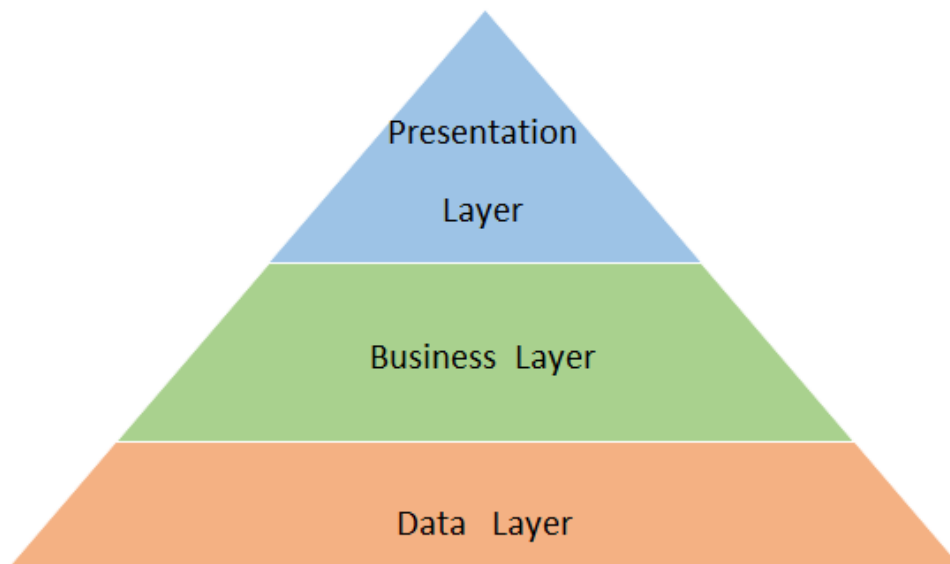


Figura 1: Arquitetura de Software

A figura que se segue ilustra a organização das diretorias do projeto. Uma organização cuidada dos ficheiros pelas diferentes diretorias é importante pois torna mais intuitivo e rápido o acesso aos mesmos. Adicionalmente, é também indicada a distribuição das diretorias pelas diferentes *software layers*:

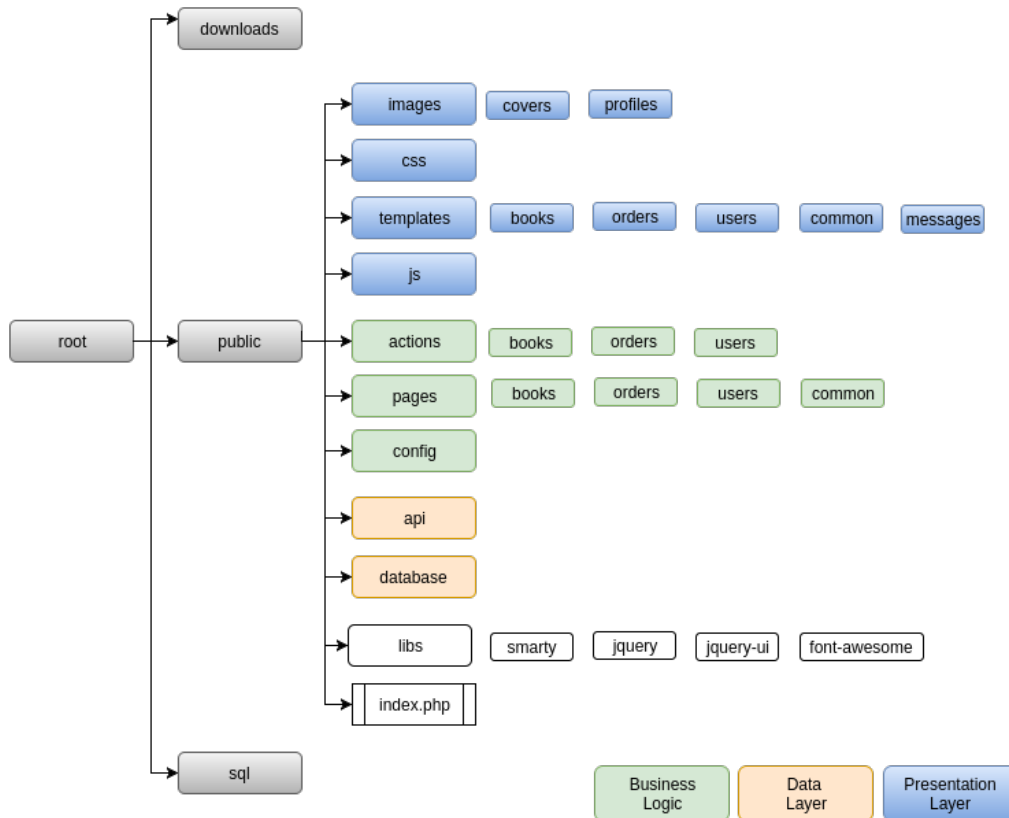


Figura 2: Organização das diferentes diretorias

Quanto às diretorias inseridas na camada *Presentation Layer*, estas contêm todos os ficheiros que o utilizador vê e interage:

- **images:** contém as fotos de perfil dos utilizadores (identificados como *\*id do utilizador\*.png* e capas dos livros, dentro da sub pasta, *covers*, identificadas como *\*referencia do livro\*.png*;
- **css:** folha de estilo;
- **templates:** todos os ficheiros *tpl* e que resultam no output da camada de lógica do negócio;
- **js:** para os *scripts*.

Por outro lado, à camada *Business Layer* só pertencem ficheiros que tratam da manipulação dos dados. Por esta razão, estes ficheiros não incluem explicitamente código HTML ou SQL. No que diz respeito às diretorias:

- **actions:** contém todo o tipo de ações relativas aos utilizadores (como o *login*, *logout* e registo), aos produtos em si (adicionar e editar) e às encomendas (alterar estado e adicionar produto ou fazer *checkout* do carrinho de compras);
- **pages:** corresponde também a código php (assim como *actions*) e essencialmente tratam das operações e testes necessários para o fornecimento de informação aos *templates* da camada *Presentation Layer*;
- **config:** contém essencialmente ficheiros para configuração e inicialização das chamadas à base de dados.
- **api:** tem como objetivo devolver código JSON para chamadas AJAX com o objetivo de dispor informação sem a necessidade de fazer *reload* à página

Por último, *Database Layer* é a camada de acesso aos dados. Os ficheiros presentes na diretoria *database* contém apenas funções de *query* à base de dados. Estas funções são fundamentalmente usadas pela camada *Business Layer*, nomeadamente pelos ficheiros das diretorias *actions* e *pages*.

## 2 Base de dados

A figura seguinte representa o modelo relacional adotado no projeto, tanto para o segundo como para o terceiro trabalho prático.

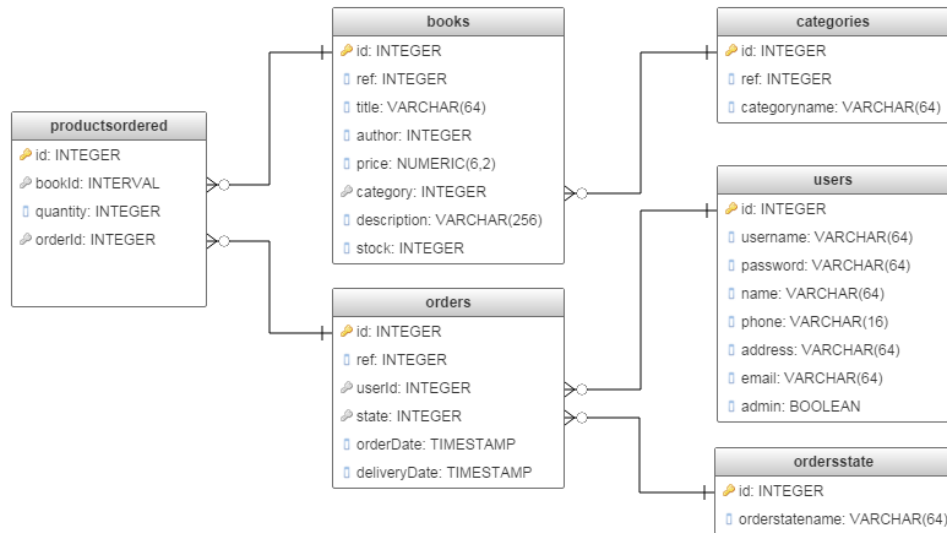


Figura 3: Modelo da Base de dados

### 3 Trabalho Laboratorial 2

Os esquemas que se seguem ilustram as possíveis transições entre as páginas do *website* de acordo com a entidade em questão: visitante, cliente ou administrador.

Nota: Na representação utilizada, setas contínuas representam transições tomadas por iniciativa do utilizador (por exemplo, *click* num botão). Por outro lado, representações a tracejado representam transições "involuntárias", desencadeadas por ações, por exemplo.

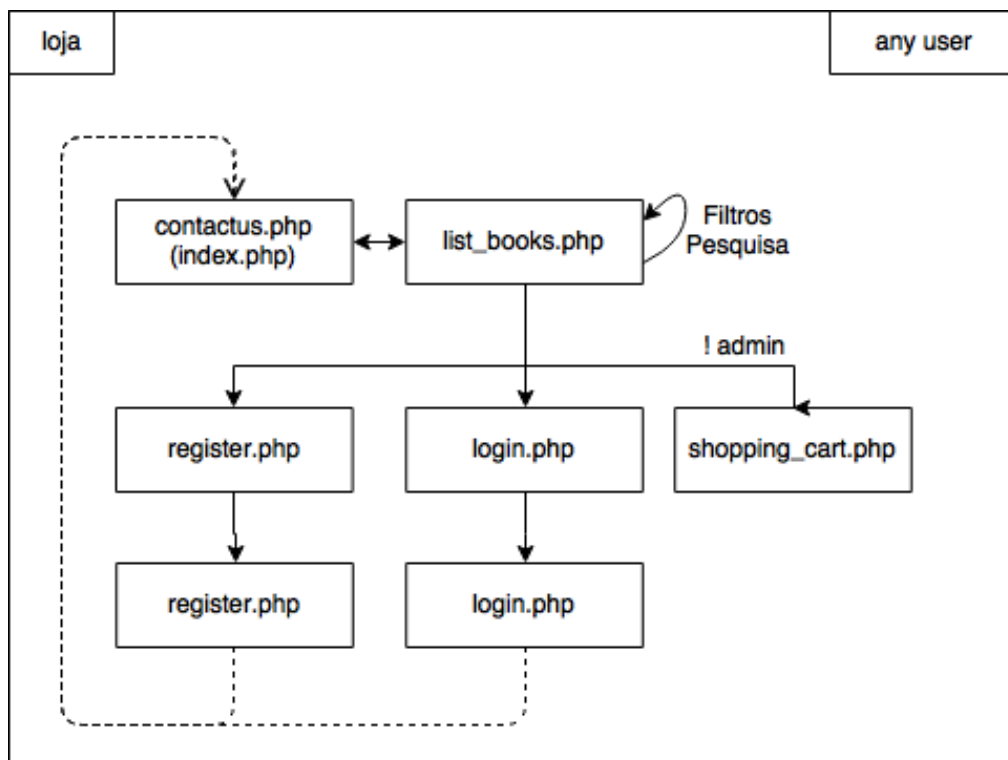


Figura 4: Modelo de Navegação na loja para os três tipos de utilizadores

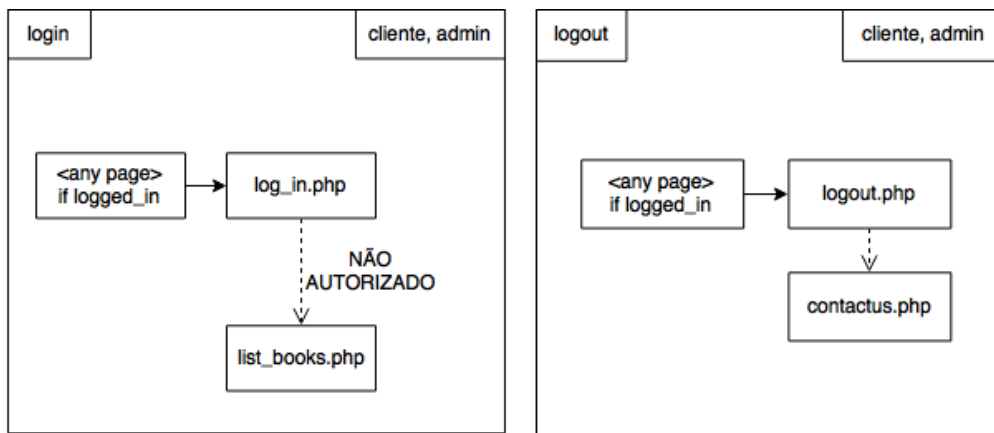


Figura 5: Modelo de Navegação de login/logout para clientes e administradores

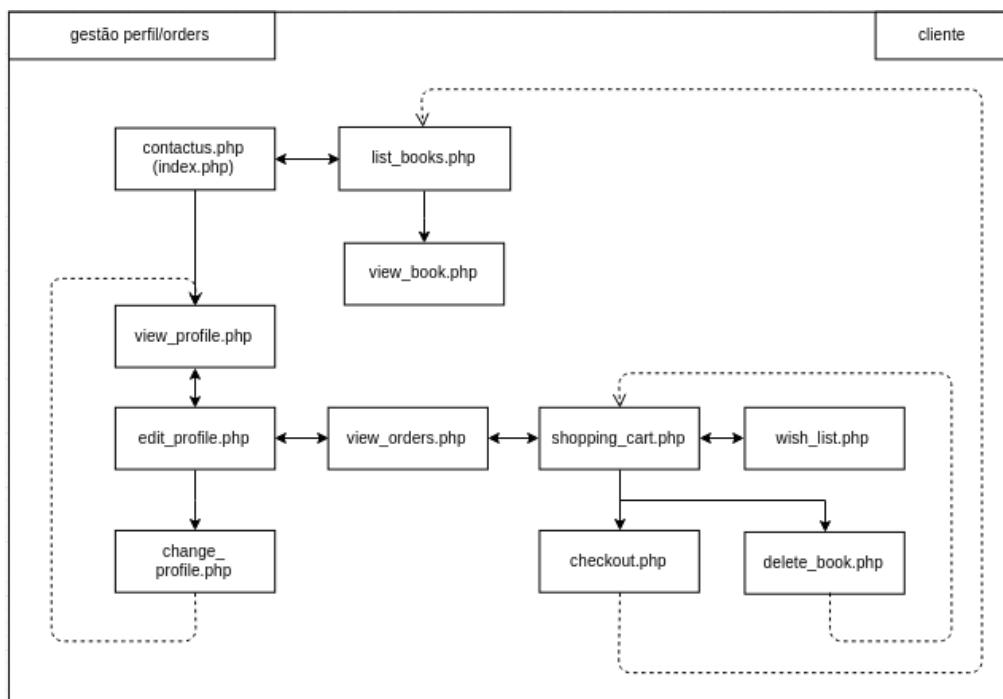


Figura 6: Modelo de Navegação para cliente - gestão perfil/orders



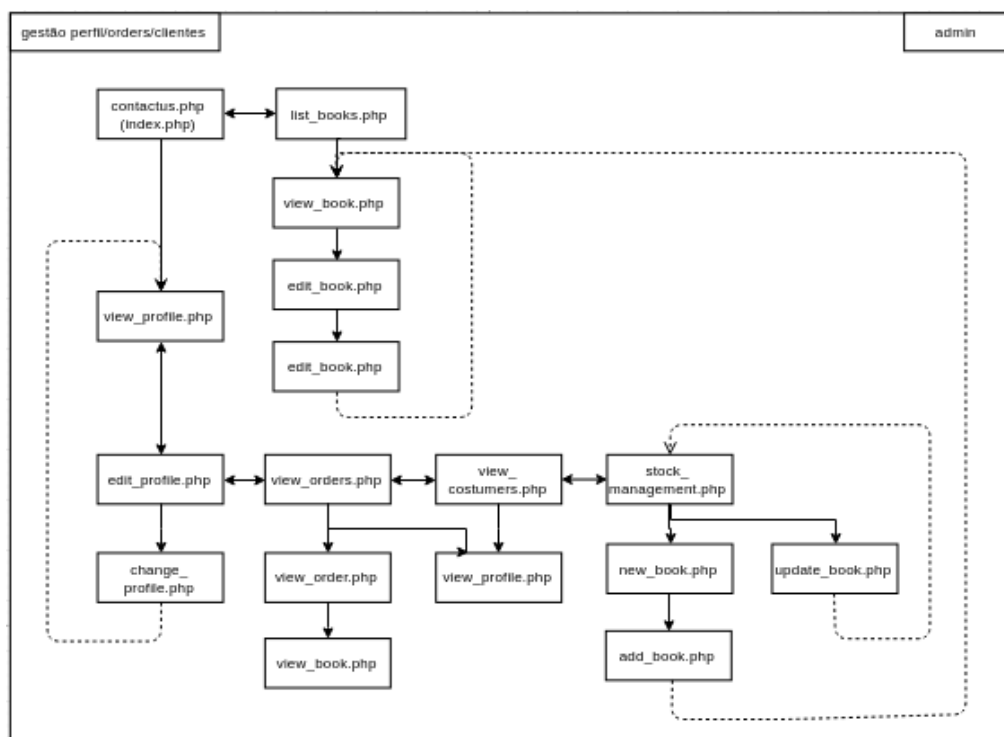


Figura 7: Modelo de navegação para admin - gestão de perfil/encomendas/clientes

## 4 *Use Cases*

Tal como ilustrado acima, os utilizadores têm diferentes permissões consoante se trata de um visitante, administrador ou cliente. De seguida são enunciadas as operações disponíveis para cada uma das entidades.

### **Visitante:**

- *Login* ou registo;
- Visualizar/filtrar/pesquisar produtos;
- Adicionar/eliminar produtos do carrinho de compras.

### **Cliente:**

- Ver/editar perfil
- Fazer *upload* de uma foto de perfil;
- Visualizar/filtrar/pesquisar produtos;
- Adicionar/remover um produto à *wishlist*;
- Adicionar/eliminar produtos do carrinho de compras e fazer *checkout*
- Alterar o estado das encomendas (de enviado para recebido).

### **Administrador:**

- Ver/editar perfil;
- Fazer *upload* de uma foto de perfil;
- Visualizar/filtrar/pesquisar produtos;
- Alterar o estado das encomendas;
- Gerir *stock*;
- Ativar/descontinuar produto;
- Alterar/adicionar produtos;

- Aceder ao perfil dos clientes;

Pelo contrário algumas operações deixaram de ser disponibilizadas aos utilizadores, por achar-se que não fariam muito sentido:

- Remover clientes;
- Remover produtos - agora substituído pelo estado de “descontinuado”;

## 5 Melhorias em relação ao trabalho anterior

Para além das novas operações disponibilizadas aos utilizadores apresentadas anteriormente, foram também melhorados outros aspetos relativamente ao trabalho anterior:

- Adicionadas *strip tags* antes de adicionar qualquer novo elemento à base de dados para evitar que um utilizador mal intencionado introduza *tags* html na base de dados;
- Redirecionamento para uma das páginas principais (por exemplo, para a página `index.php`) quando um utilizador tenta aceder a uma página ou tenta realizar uma operação para a qual não tem permissões (inclui forçar o URL para qualquer página e/ou *action*);
- Paginação da página principal (`listbooks.php`) por ajax ao invés de se passar o id da página por URL;
- *Display* dos livros e filtragem por AJAX;
- *Autocomplete* na pesquisa de livros;
- Criadas algumas animações para tornar mais interativo;
- Criados *scripts* para melhorar a experiência do utilizador.