Índice

[1. Notas 2](#_Toc132546213)

[2. Componentes 3](#_Toc132546214)

[3. Ideias 4](#_Toc132546215)

# Notas

**Workflows:** pensar em actividades de Workflow versus perfis e permissões. Ao usar perfis e permissões, é necessário criar um perfil e permissões para cada tipo de actividade. Por exemplo, “gestor”, “validar aprovação de documento”. Se, num Workflow, houver vários passos de aprovação diferentes, deverão haver tantas permissões quantos passos (diferenciados) e associar essas permissões a cada utilizador (e não ao perfil, senão tinha que ser criado um perfil também). Em Workflows, como no FileNet, existem tarefas, e cada uma tem que ser associada a uma pessoa / grupo de pessoas, ao invés das permissões. Qual é melhor?

**Hierarquias de pastas:** em procura de elementos (ficheiros, etc.) há duas formas de fazer as procuras, cada uma mais adequada à sua situação: global, e por contexto. Por exemplo: na nossa estrutura de pastas da WorkShare, é conveniente ter os ficheiros organizados por processo quando estamos a tratar de várias coisas relativas a um processo (ex: Z:\Financeira\CTR\_01\Processo, WF Templates, Backup PEPs, Test Scripts, Quick Guides, Interface SAP, Migração). Mas, quando estamos a trabalhar para, por exemplo, fazer os Quick Guides, é mais conveniente ter a divisão em áreas (ex: Quick Guides\CTR\_01, GPN\_01, CTP\_04, TSO\_02…).

**Modelação:** partir sempre de casos particulares (use-cases) para uma generalização. O modelo a que se chega (de processos, workflow, etc.) é sempre uma generalização que facilita o trabalho de implementação. O objectivo é pegar nos casos particulares (que é, ao fim e ao cabo, o que pretendemos implementar/suportar) e definir um modelo em que se reaproveita ao máximo os caminhos possíveis, de forma a reaproveitar. Mas é sempre necessário ter a visão dos casos particulares, tanto para perceber a generalização como para não perder o rasto ao objectivo inicial.

Os use-cases podem voltar a ser utilizados quando chegarmos à fase de testes. Uma vez que as situações particulares são aquelas que são validadas com o cliente antes da implementação e terão que ser validadas depois, o mais lógico é que os testes a efectuar sejam guiados pelos use-cases.

**Programação:** objectos “Worker” vs “Data”: há objectos que são tipicamente para trabalhar, e o que interessa são os métodos que têm  serviços que disponibilizam. Depois, há os objectos Data, que são apenas repositórios de dados, estáticos, e que não fazem absolutamente nada.

**Escolha do nível de generalização:** para o utilizador, generalidade não é bom: requer um nível de abstracção prejudicial. É muito melhor ter um serviço concreto para trabalhar. Já para a implementação desse serviço, é bom usar generalidade, porque permite que se reusem partes da implementação.

# Componentes

**Renderer:** Fazer um \*Renderer, para transformar um determinado objecto em HTML. Exemplo: ServicoRenderer:

Table serviço = …;

ServicoRenderer renderer = new ServicoRenderer(servico);

renderer.render(response.getWriter());

(alias, ver se isto compensa…)

**Table 🡪 Objecto *data***

##

Table deveria ser um interface, que pode ter várias implementações diferentes. Ex: SQLTable, HTMLTable, etc. Importante é ter um interface comum para aceder à informação que tem um formato de tabela.

Estes objectos podem implementar Map (as keys são os nomes das colunas), e as Rows/Columns List.

ModifiableTable: possibilidade de adicionar colunas, modificar colunas, somar duas tabelas...

Querier: fazer um insert ou update usando uma Table / Row.

**Document:**

interface ImmutableDocument implements Iterable: implementação com ArrayList e String

String getLine(int line);

int countLines();

ImmutableIterator iterator();

interface Document extends ImmutableDocument: implementação com LinkedList e StringBuffer

StringBuffer getLineBuffer(int line);

Iterator iterator();

self addLine(int pos, String line);

# Ideias

🡪 **Formato global** de transferência de dados (semelhante ao EDI), mas em XML. Exemplo: facturas – definir qual a informação que todas as facturas devem ter: NIF, nome da empresa, data, valor, etc. Se uma determinada empresa enviasse as facturas neste formato para outra e o sistema dela o reconhecesse, então esta estaria directamente disponível em ambos os sistemas. Não havia, por exemplo, a necessidade de usar soluções com OCR como na PT PRO.

🡪 **Browser** que fosse capaz de manter as páginas num repositório local. Cada site teria a sua pasta (ex: Java.sun.com), e o utilizador ficaria com um registo de navegação offline. A cada vez o utilizador fosse à Internet, o registo ficaria actualizado. Seria então possível navegar na Internet offline.

🡪 **Sistema de Modelação** - Modelo de processos visual em que se consigam ver:

- Processos

- Actividades

- Tarefas, divididas por actividade e por processo

- FAQ

- Lista de Actividades/Tarefas em Excel

🡪 **Workflows:** caso prático de utilização de workflows dinâmicos / de gestão pessoal:

Assunto: Procura de Emprego 2004

1- Primeira fase de candidaturas: incluir e-mails enviados, documento de registo de progresso, URLs importantes, 1ª versão da carta de apresentação

2- Rearranjar currículo: 2ª versão do curriculo, 2ª versão da carta de apresentação, URL do InfoJobs, alguns textos retirados do InfoJobs para referência futura (requisito: criar documentos de texto automaticamente, ou talvez até outros, baseados em templates)

3- Processo de selecção da Accenture: mails (da empresa e ao pessoal)

4- Curso de Formação (módulos terminados e não)

5- Procura de apartamento (mails, contactos telefónicos, agências, URLs de agências e do sapo, fotos de casas vistas, ...)