INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Software configurável de processamento de dados heterogéneos para geração de documentos PDF

MÓNICA DIAS CORREIA
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Software configurável de processamento de dados heterogéneos para geração de documentos PDF

Mónica Dias Correia

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Prof. Doutor Nuno Silva

Dedico ao meu namorado, à minha família e aos meus amigos, por sempre acreditarem que sou capaz de alcançar os meus objetos e me apoiarem em tudo.

Resumo

Existem empresas que efetuam a impressão e/ou envelopagem de documentos, efetuando muitas vezes serviços de grandes quantidades (centenas ou até mesmo milhares de impressões e envelopagens). Os serviços destas empresas são contratados, por exemplo, pelas câmaras municipais ou pelas respetivas empresas responsáveis pela água dos vários municípios do país, para todos os meses enviarem para casa dos seus clientes as faturas de água (podendo também enviar avisos de corte, avisos de leitura, entre outros). Estas empresas de impressão e envelopagem são as responsáveis pelo processamento (geração, impressão e envio) das faturas.

Partindo de um software já existente de uma empresa que efetua a criação de faturas de água (documentos PDF) de algumas câmaras municipais, o principal objetivo deste trabalho consiste em dar resposta à necessidade dos clientes (e.g. câmaras municipais, companhias de água) de lidar com o processamento de vários tipos de ficheiros com diferentes formatações de dados e documentos.

Com tal objetivo, será desenvolvido um novo software capaz de suportar o processamento de diferentes e emergentes tipos de documentos sem a intervenção da equipa de desenvolvimento, passando aos colaboradores da empresa a responsabilidade e capacidade de configurar o processamento desses novos e diferentes tipos de ficheiros, o que potenciará o aumento do número de clientes e do aumento de negócio.

Palavras-chave: geração de PDF, envelopagem, configurabilidade, manutenibilidade

Abstract

There are companies that perform the printing and/or envelopage of documents, usually large quantities of services (hundreds or even thousands of impressions and envelopage). The services of these companies are contracted, for example, by the City Halls or by the respective companies responsible for the water of the various City Halls in the country, for every month send the water bills to their customers' homes (you can also send cutting warnings, reading notices, etc.). These printing and envelopage companies are responsible for the processing (generation, printing and sending) of invoices.

Starting from an existing software of a company that makes the creation of water invoices (PDF documents) of some City Halls, the main objective of this work is to respond to the needs of customers (e.g. City Halls, water companies) to handle the processing of some types of files with different data formats and documents.

With such a goal, new software will be developed capable of supporting the processing of different and emerging types of documents without the intervention of the development team, to be of the company's collaborators the responsibility and ability to configure the processing of these new and different types of files, which will increase the number of customers and increase business.

Keywords: PDF generation, envelopage, configurability, maintainability

Agradecimentos

Em primeiro lugar e de forma especial, quero agradecer a todas as pessoas que fazem parte da minha vida pessoal e que contribuem de alguma forma para o meu bem-estar e que me apoiam em todas as minhas decisões, por estarem presentes nos vários momentos da minha vida.

A todas as pessoas da instituição de ensino a que pertenço (ISEP), especialmente ao Departamento de Engenharia Informática (DEI) pelos conhecimentos partilhados e pelo apoio e incentivo por parte de todos os docentes que me ajudaram de forma a que os meus objetivos académicos fossem atingidos.

Ao professor Doutor Nuno Silva por me ter orientado e por me mostrar sempre as melhores formas de atingir bons resultados, efetuando sempre críticas construtivas, ajudando-me a melhorar, mostrando-se sempre disponível para me acompanhar e ajudar.

Deixo um especial agradecimento a toda a minha família e namorado, que sempre me apoiaram e nunca me deixaram desanimar, dando-me sempre motivação para continuar e não desistir.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
PDF	Portable Document Format
JSP	JavaServer Pages
API	Application Programming Interface
AJP	Apache JServ Protocol
UI	User Interface
NCD	New Concept Development
AHP	Analytic Hierarchy Process
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
DOM	Document Object Model
FOP	Formatting Objects Processor
RAM	Random Access Memory
SAX	Simple API for XML
JAXB	Java Architecture for XML Binding
CPU	Central Processing Unit
MVC	Model-View-Controller
WAR	Web Application Resource
Java EE	Java Platform, Enterprise Edition
BPMN	Business Process Model and Notation
GUI	Graphical User Interface
AWT	Abstract Windows Toolkit
HTML	HyperText Markup Language

GoF	Gang of Four
UML	Unified Modeling Language
BIRT	Business Intelligence e Reporting Tools
TDD	Test Driven Development
AAA	Arrange, Act and Assert

1 Introdução

Neste capítulo são expostas as motivações que levaram à elaboração deste documento, apresentando uma introdução ao contexto de trabalho, e uma breve descrição dos problemas encontrados.

Serão também apresentados os objetivos que pretendem ser atingidos e a abordagem para os resolver, salientando as várias contribuições da solução encontrada.

Por fim, será apresentada a estrutura do presente documento, facilitando a navegação no documento por parte do leitor.

1.1 Contexto

Existem empresas que efetuam a impressão e/ou envelopagem de documentos, efetuando muitas vezes serviços de grandes volumes de dados (centenas ou até mesmo milhares de impressões e envelopagens). Os serviços destas empresas são contratados pelas câmaras municipais ou pelas respetivas empresas responsáveis pela água dos vários municípios do país, uma vez que é necessário que todos os meses enviem para casa dos seus clientes as faturas de água (podendo também enviar avisos de corte, avisos de leitura, entre outros). Estas empresas de impressão e envelopagem são as responsáveis pelo processamento (geração, impressão e envio) das faturas.

Uma destas empresas (doravante designada Empresa) responsável pelo processamento das faturas criou um software específico, denominado de ProcessaDados, desenvolvido para uso interno, com o objetivo de dar resposta às necessidades desse processamento, efetuando a criação dos vários documentos PDF, que são posteriormente impressos, envelopados e enviados por correio.

A integração entre os sistemas das câmaras municipais e o ProcessaDados é feita através de ficheiros fonte de dados. Estes ficheiros são criados por empresas terceiras, responsáveis pela leitura/aquisição, controlo e gestão dos dados dos consumidores, e contêm toda a informação que vai dar origem às várias faturas (ou outro tipo de documento). Este ficheiro é depois enviado às várias câmaras municipais, que posteriormente o enviam para a Empresa para ser efetuada a criação dos vários documentos PDF, sua impressão, envelopagem e envio.

O ProcessaDados recebe o ficheiro fonte de dados, convertendo-o num ficheiro XML de acordo com um modelo canónico interno, denominado de ficheiro de dados normalizado. Este é transformado através de XSLT em vários documentos PDF correspondentes às faturas. Um ficheiro XSL, denominado de ficheiro de design PDF, contém o design e estrutura do documento PDF a ser usado pelo XSLT para transformação.

A interação do utilizador com o ProcessaDados ocorre na introdução de alguns parâmetros necessários para o seu correto funcionamento, nomeadamente o ficheiro fonte de dados e o cliente (e.g. câmara municipal) a que pertence o ficheiro.

O ProcessaDados foi desenvolvido utilizando o estilo arquitetural monolítico, implementado maioritariamente em Java, Java Servlet e JavaServer Pages (JSP), e implantado num servidor aplicacional Tomcat.

1.2 Problema

Como existem várias empresas a efetuar a criação dos ficheiros fontes de dados, acabam por existir ficheiros com diferentes formatações e especificações, sendo necessário efetuar leituras de diferentes formas, consoante a empresa que criou o ficheiro e o tipo de dados (faturas de água, avisos de corte, cartas de leitura, entre outros).

Atualmente, a Empresa tem apenas dois clientes (duas câmaras municipais), que usam o mesmo tipo de ficheiro, que contratam o serviço da Empresa para criarem as suas faturas de água. No entanto, é subcontratada por outras empresas responsáveis pelo processamento, para esta efetuar apenas a impressão e envelopagem dos documentos PDF, pois tem uma maior capacidade de impressão e envelopagem. Contudo, o valor acrescentado pelo processo de impressão e envelopagem é reduzido por comparação com o de geração do PDF.

O ProcessaDados está preparado para o tratamento de um único tipo de ficheiro de uma empresa terceira (faturas de água da empresa AIRC). Esta limitação de adaptabilidade e/ou

configurabilidade do software impede o aumento do número de clientes (câmaras municipais, companhias de água) e reduz o valor acrescentado do negócio da Empresa.

Outra limitação do ProcessaDados é não processar grandes ficheiros de fonte de dados, colocando em causa a execução do processo, assim como o elevado tempo de execução e memória utilizada, provocando por vezes atrasos nos envios.

1.3 Objetivo

Porque é importante que o software responda às necessidades do negócio, surgiu a necessidade de disponibilizar um software que seja configurável e adaptável. Dessa forma, pretende-se que os colaboradores da Empresa configurem os campos e especificações de leitura dos mesmos, de forma a que não haja a necessidade de efetuar uma alteração ao software sempre que seja necessário a leitura de um novo tipo de ficheiro.

Assim, o projeto descrito nesta dissertação tem como objetivo o desenvolvimento de um novo software que responda à necessidade de ser capaz de processar/gerar documentos/correspondência de variadas fontes de dados, melhorando também significativamente a sua eficácia e eficiência, sendo capaz de processar documentos que o software atual não é capaz, reduzindo bastante o tempo de processamento.

Portanto, é desejável que o novo software:

- Permita que o utilizador configure os campos e especificações de leitura dos ficheiros;
- Seja mais eficaz, de forma a processar ficheiros maiores, não colocando em causa a execução do processo em ficheiros que o ProcessaDados colocava;
- Seja mais eficiente, diminuindo significativamente o tempo de execução e a memória utilizada pelo novo software em comparação com o ProcessaDados;
- Adote abordagens de design e tecnologias mais atuais, potenciadoras de maior configurabilidade, adaptabilidade, evolutibilidade e manutenibilidade.

1.4 Abordagem

Será efetuada uma extensa análise a todos os processos do ProcessaDados, analisando e avaliando todas as ações efetuadas, de forma a perceber as melhorias que poderão ser implementadas, efetuando uma análise aos vários tipos de ficheiros fontes de dados e formas

de efetuar a configuração dos campos e especificações dos mesmos. Será também efetuada uma análise e comparação dos tempos de execução do software, assim como a memória utilizada, melhorando consideravelmente a eficácia e eficiência do mesmo, de forma a evitar que a execução do processo seja colocada em causa.

Será efetuada uma análise extensa, analisando todo o código do ProcessaDados, efetuando um esforço de sistematização de caraterísticas, identificando as vantagens e desvantagens para cada solução encontrada para os vários requisitos do software, analisando as várias soluções concorrentes ou semelhantes, assim como os estilos, padrões, boas práticas de design e a tecnologia de implementação.

Em função dessa análise, serão analisados e avaliados o contexto e valor de negócio da solução de negócio e informática, e avaliadas abordagens concetuais permitidas e potenciadas pelas boas práticas, permitindo, assim, sugerir direções concetuais para o resto do desenvolvimento.

1.5 Contribuições

Sucintamente, a geração dos documentos PDF usando o ficheiro de design para a sua criação deixa de ser utilizada. Por contraponto, será utilizada uma ferramenta com uma UI própria, denominada Jasper, permitindo aos colaboradores efetuar o design gráfico do documento, não sendo necessário elaborar código de software como até agora acontece no ProcessaDados. O ficheiro de design gráfico em formato Jasper é lido pela aplicação Tibco Jaspersoft Studio para configuração na geração dos ficheiros PDF.

Esta solução tem várias vantagens para a Empresa, câmaras municipais e para a sociedade.

Para a Empresa, as contribuições passam por passar a ter autonomia para efetuar a configuração do software, não dependendo de meios externos para o fazer, potenciando assim a resposta às necessidades, aumentando potencialmente os clientes (câmaras municipais) e tipos de documentos (para além de faturas de água, os avisos de corte, cartas de leitura, entre outros).

Também é uma potencial contribuição para as câmaras municipais, uma vez que todos os anos existem concursos onde as empresas apresentam os seus preços para estes tipos de serviço (criação dos documentos PDF), sendo uma mais valia para as câmaras terem mais uma empresa a entrar no concurso, podendo efetuar preços mais baixos para as mesmas.

1.6 Estrutura do Documento

O capítulo atual – Introdução – apresenta de forma breve um resumo do que é pretendido com este trabalho, o contexto da presente dissertação, o problema detetado no software existente, o objetivo a atingir ao desenvolver um novo software, a abordagem a ter para os problemas encontrados e as contribuições da solução encontrada.

No capítulo da Análise será efetuada uma análise extensa ao software existente, efetuando a análise de negócio, apresentando o conceito de negócio, assim como os vários processos e intervenientes do mesmo, identificando e analisando a oportunidade, referindo depois o valor para o cliente e valor percebido, terminando com o modelo Canvas.

Neste capítulo é também efetuada a análise da arquitetura do software existente, efetuando a descrição da arquitetura utilizando o modelo 4+1, assim como são identificadas e analisadas as tecnologias utilizadas no desenvolvimento e na implantação do mesmo, analisando a eficácia e eficiência. É também efetuada a analise aos vários ficheiros fonte de dados, sistematizando depois os vários requisitos funcionais e não funcionais do mesmo.

No capítulo do Estado da Arte são apresentadas as várias soluções existentes para os problemas encontrados nos vários requisitos, analisando e avaliando as várias soluções, identificando as vantagens e desvantagens de cada uma das mesmas.

No capítulo do Design serão descritas as alternativas arquiteturais, utilizando o Modelo New Concept Development, identificando os seus cinco elementos chave, efetuando também uma análise hierárquica, assim como é apresentado o design arquitetural do novo software, descrevendo a interação entre os vários componentes do software e demonstrando as interações e os tipos de comunicações utilizados pelo software utilizando o modelo 4+1.

Na Implementação são descritas as tecnologias utilizadas, os padrões e regras utilizadas no desenvolvimento do novo software, assim como uma breve abordagem dos testes unitários e de sistema desenvolvidos, sendo esta uma parte indispensável no desenvolvimento de software uma vez que nos permite testar o software com vista a reduzir o número de bugs o que automaticamente leva o utilizador final a ter uma melhor experiência de utilização do mesmo.

No capítulo das Experiências e Avaliação, contém a abordagem e preparação da mesma, assim como os testes funcionais (também conhecidos como testes de sistema), uma vez que estes exercitam uma parte completa do sistema, testando os vários requisitos funcionais dos vários softwares, analisando os resultados, efetuando uma comparação com os resultados obtidos no ProcessaDados, efetuando a sua avaliação.

Por fim, o capítulo Sumário, onde são apresentados os objetivos alcançados, apresentando também quais as limitações encontradas durante todo o desenvolvimento deste projeto, referindo o trabalho a desenvolver, assim como uma apreciação final e pessoal.

2 Análise

Este capítulo tem três objetivos:

- Efetuar uma análise de negócio, identificando e caraterizando os conceitos, processos e intervenientes no negócio;
- Analisar o software ProcessaDados, de forma a descrever de forma detalhada o software existente, identificando e caraterizando as funcionalidades, arquitetura e desempenho (eficácia e eficiência) do mesmo, analisando consequentemente quais os problemas que impedem o suporte adequado ao negócio;
- Alterações ao negócio, nomeadamente a necessidade de processar diversos tipos de dados fonte e de geração de documentos configurados a partir destes.

2.1 Análise de negócio

O objetivo principal desta secção é identificar e descrever os conceitos de negócio mais importantes, descrevendo os vários processos e respetivos intervenientes. Também será identificada e analisada a oportunidade de negócio, analisando o valor de mercado tanto para a organização como para os clientes, sendo também apresentado o modelo Canvas, de forma a ter uma visão geral do negócio.

2.1.1 Conceitos de negócio

Como muitas das câmaras municipais ou as respetivas empresas responsáveis pela água dos vários municípios do país necessitam de criar documentos (e.g. faturas de água, avisos de corte, cartas de leitura, entre outros) baseados nas leituras que fazem de contadores e outros

instrumentos, para enviarem para casa dos seus clientes, surgiu a necessidade de criar um software que fizesse a geração dos documentos PDF a partir dos dados guardados.

Uma vez que existem empresas que efetuam a impressão e/ou envelopagem de documentos, estas empresas são também contratadas pelas câmaras municipais ou as respetivas empresas responsáveis pela água dos vários municípios do país, sendo as responsáveis pelo processamento (geração, impressão e envio) das faturas.

O ProcessaDados foi criado com o objetivo de criar os documentos PDF de faturação para envio aos consumidores.

Esta Empresa trabalha com duas câmaras municipais, efetuando todos os meses a conversão dos dados recebidos no ficheiro fonte de dados em vários documentos PDF representativos das faturas de água dos residentes do município. Todo este processo é efetuado pelo ProcessaDados, sendo este executado pelo utilizador, introduzindo alguns parâmetros necessários (o ficheiro fonte de dados e a câmara municipal a que pertence o ficheiro).

2.1.2 Processos e intervenientes

Esta secção descreve todos os intervenientes envolvidos no processo de geração de faturas, assim como as respetivas tarefas, apresentando a explicação generalizada sobre a elaboração das funções que cada elemento está incumbido a fazer. Estas funções são as descritas na Figura 1.

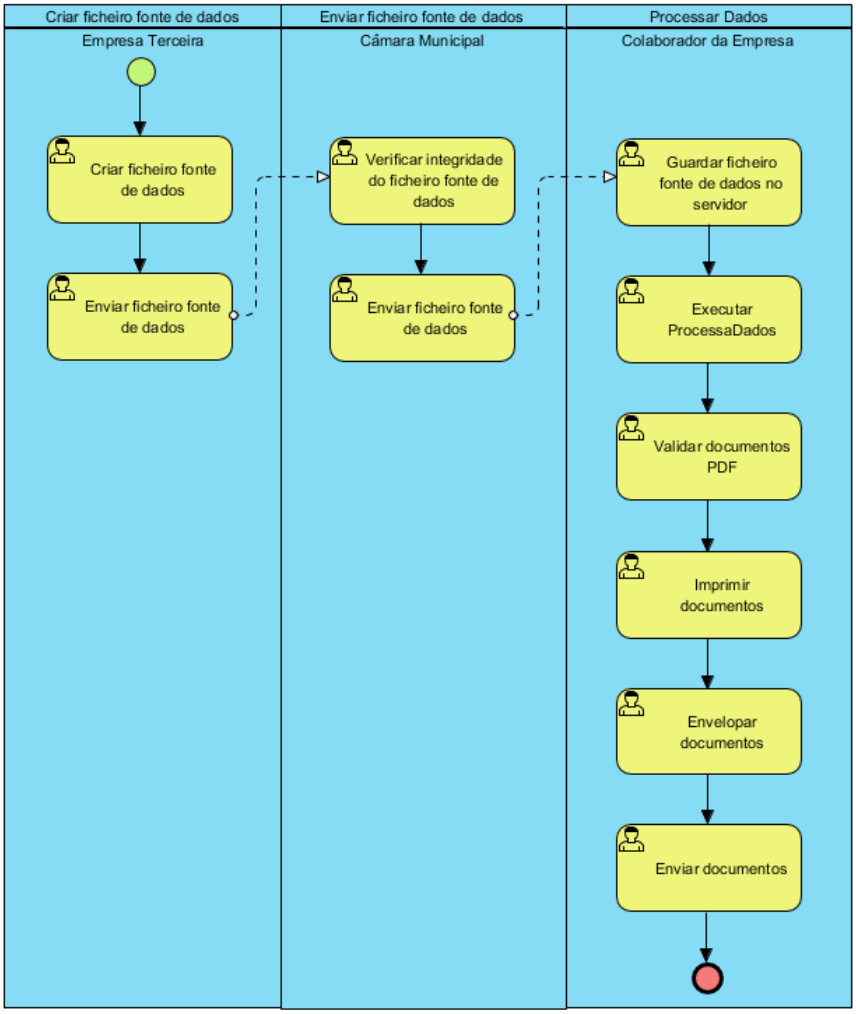


Figura 1 – Diagrama BPMN do ProcessaDados

2.1.2.1 Empresas Terceiras

Conforme referido anteriormente, e é possível verificar na Figura 1, as empresas terceiras (e.g. AIRC, Medidata, entre outras) são responsáveis pela criação do ficheiro fonte de dados, uma vez que também são as responsáveis pela gestão de dados de consumo para a criação das faturas e dos outros vários documentos, sendo enviado de seguida o ficheiro para a câmara municipal.

2.1.2.2 Câmaras municipais

Conforme é possível verificar na Figura 1, as câmaras municipais que sejam clientes da Empresa são responsáveis por verificar a integridade do ficheiro fonte de dados recebido pela empresa terceira, e pela validação da estrutura do mesmo, não sendo efetuada nenhuma validação do ficheiro por parte da Empresa que criou o ProcessaDados. Por fim, envia o ficheiro recebido por email à Empresa.

2.1.2.3 Colaboradores da Empresa

Conforme referido anteriormente, e também é possível verificar na Figura 1, os colaboradores da Empresa são responsáveis por guardar o ficheiro fonte de dados recebido por email na respetiva pasta da câmara municipal no servidor, efetuando de seguida a execução do ProcessaDados, que irá aceder a este ficheiro.

No fim da execução do ProcessaDados, os colaboradores são responsáveis pela validação dos documentos PDF criados, verificando apenas o número destes, sendo depois efetuada a impressão e envelopagem dos mesmos, sendo enviados posteriormente.

2.1.3 Identificação da oportunidade

Cada vez mais é necessário que as empresas trabalhem com softwares adaptáveis, de fácil manutenibilidade e também que sejam eficazes e eficientes, dando resposta às necessidades dos seus clientes.

A Empresa reconheceu que este estava com problemas nestes vários pontos, sendo cada vez mais difícil dar uma resposta rápida às necessidades dos seus clientes, falhando por vezes o cumprimento de prazos de entrega, assim como é incapaz de dar resposta a ficheiros com novos formatos, impedindo a Empresa de crescer.

Posto isso, surgiu a ideia de efetuar um novo software, tendo em conta as necessidades da Empresa e do cliente, sendo esta oportunidade aproveitada para permitir configurar os campos e especificações de leitura do ficheiro, melhorando também a adaptabilidade, manutenibilidade, eficácia e eficiência do software.

2.1.4 Análise da oportunidade

De forma a efetuar a análise da oportunidade referida anteriormente, foram surgindo questões bastante importantes, como:

- O software utiliza métodos eficazes e eficientes de leitura e escrita de ficheiros?
- O software utiliza métodos eficazes e eficientes de criação de documentos PDF?
- O software utiliza as ferramentas e/ou tecnologias adequadas?
- O software foi desenvolvido utilizando a arquitetura adequada para potenciar a adaptabilidade e manutenibilidade requeridas?

- O software permite a leitura de novos tipos de ficheiros?
- O software permite a configuração dos campos e especificações de leitura?
- Existem soluções concorrentes?

2.1.5 Definição de Conceito

No presente projeto, a definição de conceito deu início ao desenvolvimento de um novo software, que permita a configuração dos campos e especificações de leitura, ao mesmo tempo que melhora:

- Eficácia, entendido como sendo a capacidade de alcançar o efeito esperado ou desejado através da realização de uma ação [10];
- Eficiência, entendido como sendo a capacidade de dispor de algo para conseguir um efeito determinado [11];
- Configurabilidade, entendido como sendo a capacidade de configurar algo;
- Adaptabilidade, entendido como sendo a capacidade de se adaptar às necessidades [26];
- Manutenibilidade, entendido como sendo a facilidade que se pode realizar a manutenção de um determinado componente, ou sistema [46].

2.1.6 Valor para o cliente e valor percebido

A análise de valor é um estudo organizado, que se concentra em perceber de que forma o produto pode ser melhorado a um custo baixo, não sacrificando a sua fiabilidade e qualidade.

O valor de um produto nem sempre está associado a uma redução de custo, uma vez que em alguns casos o valor pode ser aumentado, aumentando a sua funcionalidade [54].

Existem dois aspetos que caracterizam o valor para o cliente:

- O valor desejado, é o que o cliente deseja de um produto ou serviço;
- O valor percebido, é o benefício que o cliente acredita que recebe desse produto ou serviço.

Pode-se então concluir que o valor percebido é a opinião que o cliente tem sobre um produto ou serviço, tendo em vista a satisfação das necessidades e exigências consideradas pelo mesmo.

Com o novo software a Empresa prevê vários benefícios, sendo estes os seguintes:

- Configuração – É expectável que o software permita efetuar a configuração dos campos e especificações de leitura dos vários tipos de ficheiro, de forma a não ser necessário nenhum desenvolvimento ou alteração ao software existente, permitindo que a Empresa seja capaz de conseguir criar os documentos PDF de outros tipos de ficheiro;

- Eficácia – É expectável que o software efetue as suas funcionalidades corretamente, não colocando em causa a sua execução, de forma a satisfazer as necessidades do cliente, cumprindo os prazos exigidos pelos mesmos;

- Eficiência – É expectável que o software responda ao esperado de forma rápida, demorando o menor tempo possível na sua execução;

- Adaptabilidade e manutenibilidade – É expectável que o software seja adaptável às necessidades (atuais ou novas) do cliente, sendo mais fácil a sua modificação e manutenção, aumentando a fiabilidade do mesmo, potenciando o aumento de clientes.

Embora não haja custos associados ao uso do software, uma vez que este é usado apenas internamente pela Empresa, também haverá sacrifícios:

- Dificuldades de migração – A Empresa pode ter dificuldades em efetuar a migração do software, uma vez que a mesma não contém nenhum colaborador com capacidades para efetuar a migração, podendo mesmo ser necessário contratar uma equipa/empresa para o efetuar;

- Custos – A Empresa deve estar preparada para suportar alguns investimentos, caso seja necessário efetuar alterações no software, sendo necessário contratar uma equipa de desenvolvimento para efetuar as alterações.

Por fim, é apresentada na Tabela 1 os benefícios e sacrifícios do software.

Benefícios	Sacrifícios
Configuração Eficácia Eficiência Adaptabilidade e manutenibilidade	Migração Custos

Tabela 1 – Tabela de benefícios e sacrifícios

2.1.7 Modelo Canvas

O modelo de Canvas é uma ferramenta prática e versátil que permite visualizar aspetos de um modelo de negócios [18].

Este modelo possui nove partes que ajudam a conhecer melhor o negócio:

- Parcerias principais – são as atividades-chave realizadas por terceiros assim como os recursos principais adquiridos fora da Empresa;
- Atividades-chave – são as atividades mais relevantes para que seja possível entregar a proposta de valor;
- Recursos principais – são os recursos necessários para realizar as atividades-chave;
- Proposta de valor – são baseadas nos produtos ou serviços que criam valor para um determinado segmento de clientes;
- Relacionamento com os clientes – é o que define como é que a organização se relaciona com determinado segmente de clientes;
- Canais – é o que define como é que o cliente compra ou recebe o produto ou serviço;
- Segmento de clientes – define quais os segmentos de clientes que estão na visão da Empresa;
- Estrutura de custos – são os custos necessários para que a estrutura que foi proposta funcione;
- Fontes de receita – define como obter receitas a partir da proposta de valor.

No modelo Canvas (Tabela 2) é possível observar que não existem parcerias principais, uma vez que o software é apenas para uso interno da Empresa. Também é possível observar que são os desenvolvedores que implementam o novo software para que seja possível que a proposta de valor seja atingida.

O segmento de clientes da Empresa são as câmaras municipais e as empresas responsáveis pela água dos vários municípios, sendo o departamento comercial a se relacionar com estes.

É também possível verificar que a fonte de receita da Empresa é obtida através dos clientes que contratem o serviço efetuado pelo software, assim como os custos da Empresa, que são apenas os custos com os desenvolvedores do software.

<table>
<tr><td colspan="5">Reengenharia de aplicação monolítica</td></tr>
<tr><td rowspan="2">Parcerias Principais
Não existe.</td><td>Atividades Principais
Desenvolvimento do software</td><td rowspan="2">Proposta de Valor
Efetuar o desenvolvimento do novo software, cumprindo os requisitos (configuração, eficácia, eficiência, adaptabilidade e manutenibilidade))</td><td>Relacionamento com o Clientes
Departamento Comercial</td><td rowspan="2">Segmento de Clientes
Câmaras Municipais e empresas responsáveis pela água dos vários municípios</td></tr>
<tr><td>Recursos Principais
Desenvolvedores</td><td>Canais
UI e Aplicação web</td></tr>
<tr><td colspan="3">Estrutura de Custos
Desenvolvedores</td><td colspan="2">Fontes de receita
Clientes que contratem o serviço do software</td></tr>
</table>

Tabela 2 – Modelo Canvas

2.2 Arquitetura do Software existente

O objetivo principal desta secção é efetuar a análise da arquitetura do software existente, tanto na granularidade de sistema como na de aplicação, permitindo a perceção da interação dos vários ficheiros utilizados, identificando os requisitos fundamentais do software ProcessaDados.

A arquitetura é muito importante porque define a estrutura que atende a todos os requisitos, tendo em consideração os utilizadores do ProcessaDados, o sistema e as metas de negócios.

Após efetuar uma análise extensa ao ProcessaDados, foi efetuada a análise da arquitetura do software, identificando requisitos fundamentais para o funcionamento deste:

- Leitura de ficheiros fonte de dados;
- Escrita de ficheiros de dados normalizados;
- Leitura dos ficheiros de dados normalizados;
- Criação dos documentos PDF.

A descrição da arquitetura será efetuada utilizando o modelo 4+1 (Staveley, 2011), e em particular as vistas (i) lógica, (ii) de processo, (iii) de implementação e (iv) de implantação.

2.2.1 Granularidade de sistema

Nesta secção é apresentada a arquitetura de software numa granularidade de sistema, apresentando a vista (i) lógica, (ii) de processo e (iii) de implantação.

2.2.1.1 Vista lógica

A vista lógica demonstra as partes que integram o sistema, assim como as várias interações concetuais/lógicas. De forma a compreender melhor a estrutura/arquitetura do sistema desenvolvido foi criado o diagrama de componentes representado na Figura 2, retratando as partes funcionais do sistema.

Um diagrama de componentes retrata as partes funcionais envolvidas, e são normalmente utilizados para modelar sistemas de alto nível ou para mostrar componentes que pertencem a um nível mais baixo. Um componente representa um agrupamento físico de elementos relacionados logicamente, como é o caso dos componentes representados na Figura 2:

- Browser, sendo este qualquer um browser disponível, que seja utilizado pelo utilizador para efetuar os pedidos ao ProcessaDados;
- ProcessaDados, sendo este o software utilizado pela Empresa, estando o mesmo disponível num servidor aplicacional Tomcat como uma aplicação web;
- Cliente1Service, é um web Service, que disponibiliza uma API, nunca sendo utilizada.

Figura 2 – Vista lógica do sistema (diagrama de componentes em UML)

Esta última observação mostra que o software foi mal desenvolvido, tendo falhas significativas. De agora em diante, este último componente não será mais considerado.

2.2.1.2 Vista de processo

O diagrama de sequência apresentado na Figura 3 representa as mensagens, pedidos e interações entre os diversos intervenientes do projeto no sistema. Conforme verificado, o funcionamento explica-se da seguinte forma:

- O colaborador da Empresa efetua o pedido no browser, inserindo os dados necessários;
- O browser envia o pedido ao software ProcessaDados com os dados;
- O ProcessaDados efetua uma separação dos campos recebidos, efetuando a leitura do ficheiro fonte de dados, e de seguida a conversão deste para o ficheiro de dados normalizado;
- O ProcessaDados efetua a leitura do ficheiro de dados normalizado, e por cada elemento, cria os documentos PDF;
- O ProcessaDados retorna a informação para o browser, para que este mostre ao colaborador da Empresa.

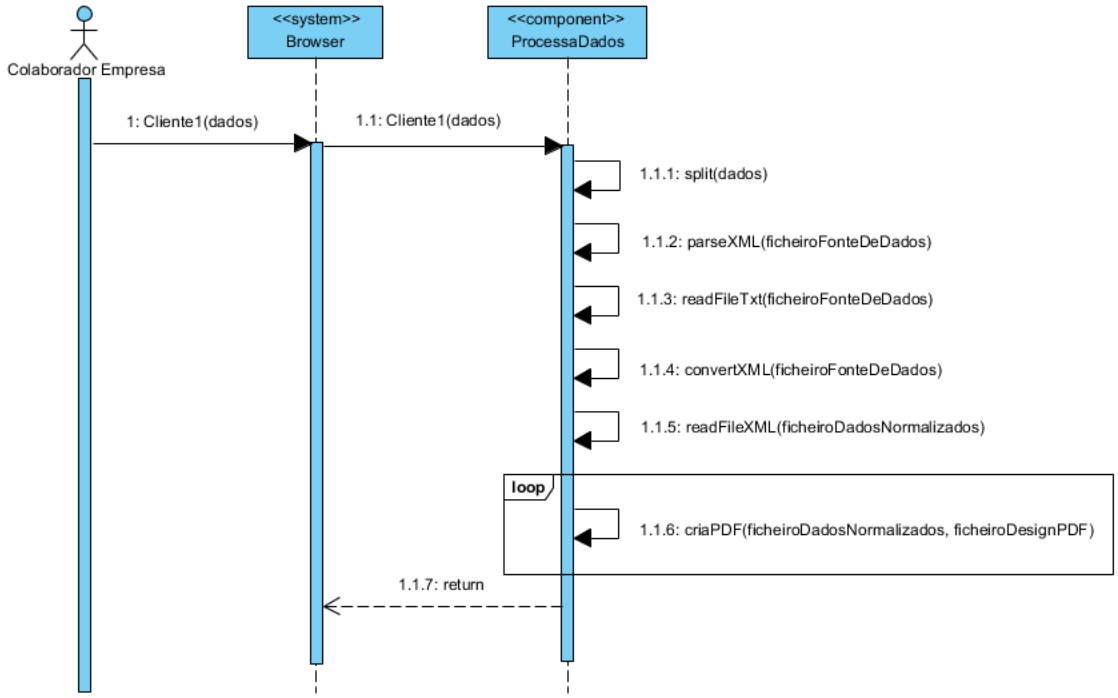


Figura 3 – Vista de processo do sistema (diagrama de sequência em UML)

2.2.1.3 Vista de implantação

Como é possível observar na Figura 4, no diagrama de implantação do sistema, o browser está instalado numa máquina cliente, enquanto o componente Apache e Tomcat estão a correr num servidor havendo comunicação entre estes através de AJP, sendo que o ProcessaDados está no componente Tomcat, havendo comunicação entre o cliente e o servidor através de um pedido HTTP.

O Tomcat é um servidor aplicacional, sendo também capaz de atuar como um servidor web, ou integrado num servidor web dedicado (e.g. Apache ou o IIS), sendo neste caso integrado com o Apache.

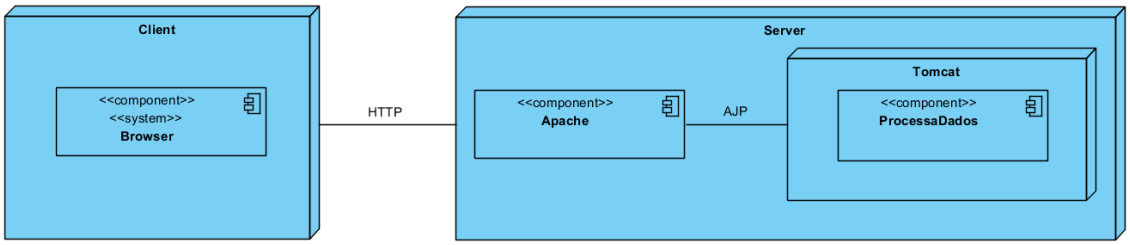


Figura 4 – Vista de implantação do sistema (diagrama de nós em UML)

2.2.2 Granularidade de aplicação

Nesta secção é apresentada a arquitetura de software relativamente à granularidade de aplicação, apresentando as vistas (i) lógica, (ii) de implementação e (iii) de processo. A vista de implantação nesta granularidade é a mesma que a apresentada para a granularidade de sistema.

2.2.2.1 Vista lógica

O diagrama de componentes apresentado na Figura 5 representa a estrutura interna do ProcessaDados.

É possível observar na Figura 5 que o ProcessaDados tem quatro componentes implantados, sendo estes o control, services, model e utils. Também é possível observar que o ProcessaDados recebe um pedido REST, sendo o pedido enviado ao componente control, comunicando com o componente services, que este por sua vez irá comunicar com o componente model e com o componente utils.

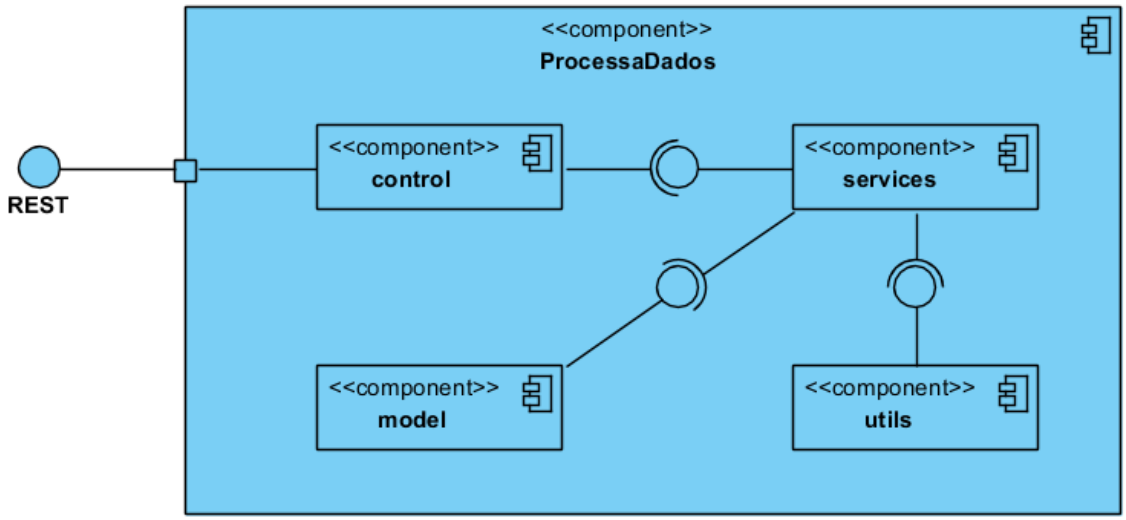


Figura 5 – Vista lógica do ProcessaDados (diagrama de componentes em UML)

2.2.2.2 Vista de implementação

Pela análise do software existente, foi elaborado um diagrama de classes agregadas em pacotes, representativo da estrutura e relações das várias classes do ProcessaDados (Figura 6).

Na Figura 6 são apresentados os quatro pacotes que constituem a aplicação:

- Control, que contém as classes Cliente1Servlet e Cliente2Servlet, sendo que cada uma destas contém o servlet para o respetivo cliente da Empresa. Estas são as classes responsáveis por receber os pedidos efetuados pelo colaborador da Empresa;

- Services, que contém as classes Cliente1Service e Cliente2Service responsáveis por efetuar as chamadas às restantes classes;

- Model, que contém a classe Documento, que representa o modelo utilizado no projeto, sendo criado um objeto, que contém a informação geral necessária para a criação do documento PDF (nome do documento e caminho da pasta onde será gravado o documento);

- Utils, que contém as classes TXTToXMLAIRCV1 e TXTToXMLAIRCV2, sendo as responsáveis pela transformação do ficheiro fonte de dados no ficheiro de dados normalizado, cada uma destas para a sua respetiva câmara municipal, assim como a classe CreataPdfService, que é responsável pela criação dos documentos PDF.

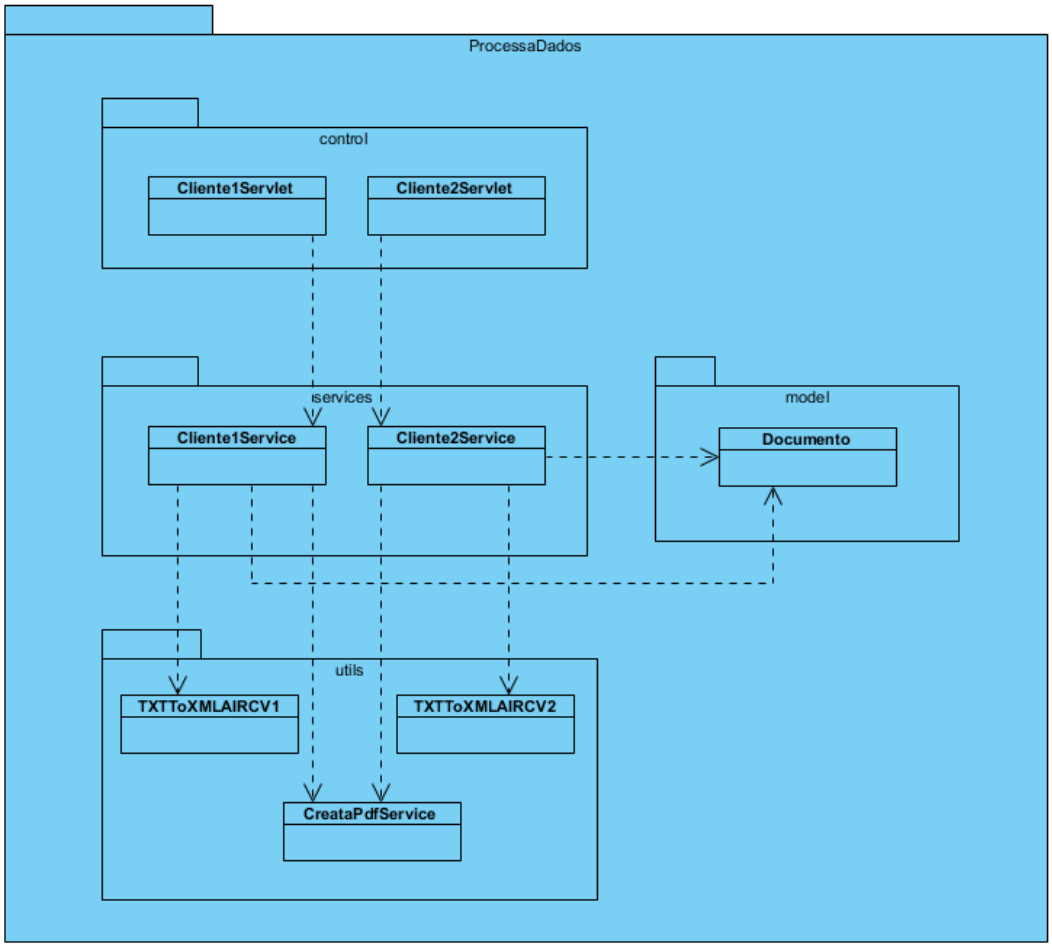


Figura 6 – Vista de implementação do ProcessaDados (diagrama de classes em UML)

2.2.2.3 Vista de processo

O diagrama de sequência apresentado na Figura 7 representa as mensagens, pedidos e interações com o ProcessaDados. Conforme verificado, o funcionamento explica-se da seguinte forma:

- O colaborador da Empresa efetua o pedido ao ProcessaDados com os dados necessários;
- É enviado um pedido ao Cliente1Servlet com os dados inseridos;
- O Cliente1Servlet efetua uma separação dos dados recebidos, efetuando de seguida a chamada ao Cliente1Service, enviando o nome do ficheiro fonte de dados ao servidor;
- O Cliente1Service efetua a leitura do ficheiro fonte de dados, enviando para o TXTToXMLAIRCV1 a informação lida, para que seja efetuada a conversão para XML;
- O TXTToXMLAIRCV1 efetua a conversão do ficheiro fonte de dados para o ficheiro de dados normalizado, guardando o mesmo, retornando para o Cliente1Service o nome do ficheiro de dados normalizado criado;
- O Cliente1Service efetua a leitura do ficheiro de dados normalizado, efetuando para cada documento, a chamada ao CreateaPdfService, enviando a informação de um documento, e o ficheiro de design PDF para efetuar a criação do documento;
- O CreateaPdfService efetua a criação do documento, utilizando o ficheiro de dados normalizado e o ficheiro de design PDF recebido, e retorna o nome do ficheiro PDF;
- O Cliente1Service retorna a informação ao Cliente1Servlet, que por sua vez apresenta-a ao colaborador da Empresa.

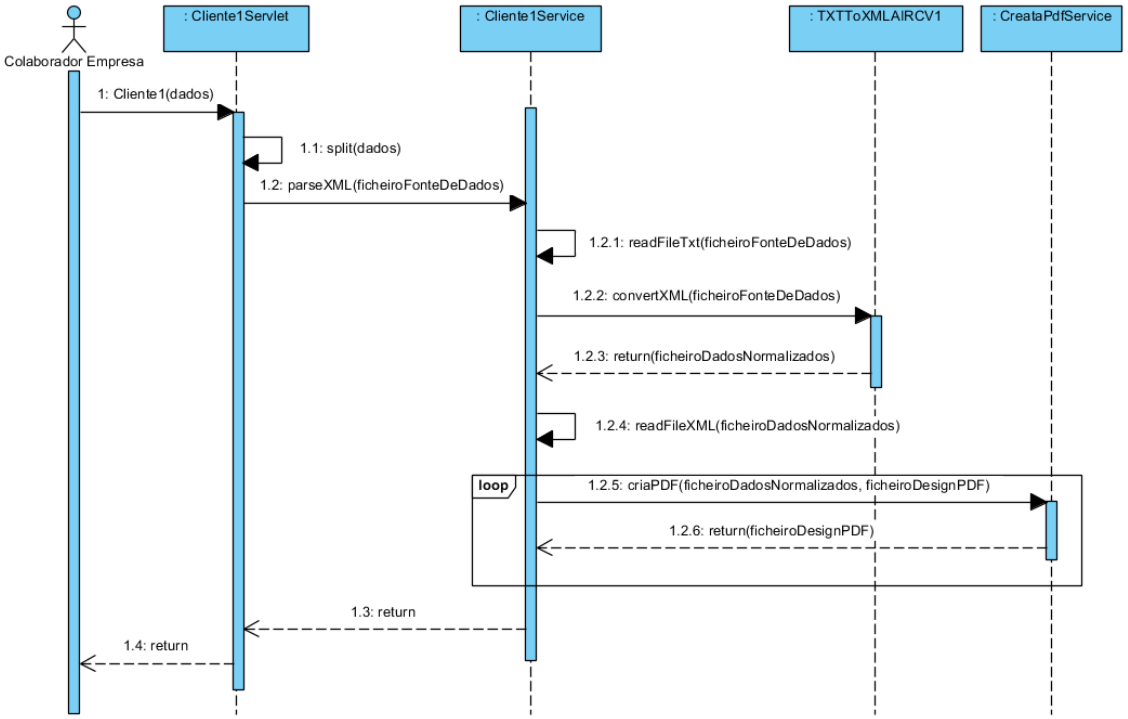


Figura 7 – Vista do processo do ProcessaDados (diagrama de sequência em UML)

2.3 Análise detalhada do software

Atualmente, o ProcessaDados é capaz de lidar com um tipo de ficheiro fonte de dados, sendo este composto por várias linhas conforme apresentado na Figura 8.

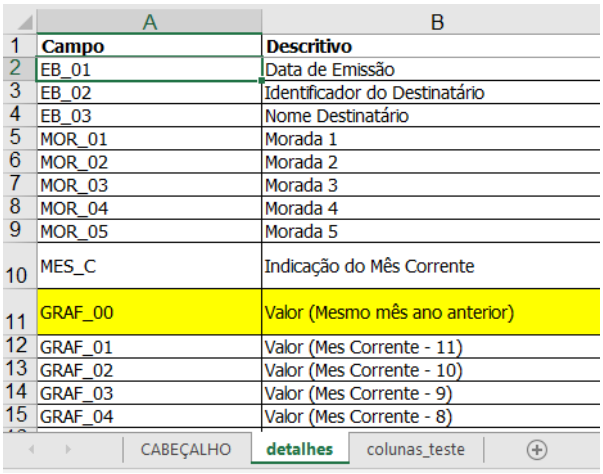
```
6|20180917|CM
2018/09/17|22866|NOME1|RUA 1|LOCALIDADE
1|3080|111|LOCALIDADE|08|17|15|12|14|12|12|12|12|11|11|11|12|11|15|97548/18|2018/09/17|2018/10/10|22866|15,33|0,36|NOME|DADOS1|DADOS2
|NIF1|DOMÉSTICO|Leitura
CM|1001|100|7674|00338358|2|34|38|2018/08/29|4|4|1|4|0,5900|||0,7375|||0,9219|||1,1523|||||||||26|Tarifa fixa -
Água|3,63|6|0,22|32,0000|0,1133|Dias|32d - 29-07-2018 a 29-08-2018|1|11|Tarifa variável - Água - 1.º Esc
[1-5m3/30dias]|2,36|6|0,14|4,0000|0,5900|m3|62d - 29-06-2018 a 29-08-2018|1|27|TRH - água|0,25|Não sujeito- Art.º 2
CIVA|0,00|32,0000|0,0078|Dias|62d - 29-06-2018 a 29-08-2018|1|23|Tarifa fixa - San.|2,72|Não sujeito- Art.º 2
CIVA|0,00|32,0000|0,0850|Dias|32d - 29-07-2018 a 29-08-2018|2|15|Tarifa variável - San. - 1.º Esc [1-5m3/30dias]|1,33|Não
sujeito- Art.º 2 CIVA|0,00|3,0000|0,4425|m3|62d - 29-06-2018 a 29-08-2018|2|24|Tarifa fixa - RSU|4,68|Não sujeito- Art.º 2
CIVA|0,00|32,0000|0,1463|Dias|32d - 29-07-2018 a
29-08-2018|3|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||BANCO|NIB|104562|10002286650|0,36|||Agosto/2018|||4||SGA000001|1|0|1|2018/06/28|||||||||||||||||||||||||||||||||||||
|||||||Y|N|38|2|15
MM|11131|240|||0,00|10,77|2018/08/16|-10,77|2018/09/10|15,33|||2018/07/29|2018/08/29|32|62|34|38|4|2018/08/29|Leitura
CM|34||0|2018/07/28|Estimativa|32|34|2|2018/06/28|Leitura CM|32||0|2018/05/30|Estimativa|32|32|0|2018/04/30|Leitura
CM|32||0|2018/03/31|Estimativa|0,06|0,00|||1081|TC8X|||||9|4|6|3|3|1|0|0|0|0|2|0|4||Fatura|14,97|B|DOMÉSTICO
(BASE)|TPL|5,99||8,98|1|-1|1|1|agosto/2018|0,00|2018/06/28|2018/08/29|34|0|||||001|||2018/09/27|2018/09/29|2018/10/10|97548|DADO
S BANCO|||0,00|2014||0|2014||0|2014||0|6,24|4,05|4,68|0,00|15,33|PORTUGAL|0,00|
2018/09/17|22845|NOME2|RUA 2|LOCALIDADE
```

Figura 8 – Exemplo de um ficheiro fonte de dados

Sistematizando a sua estrutura, tem uma primeira linha de cabeçalho com informação geral do ficheiro, informando qual a data do mesmo e a câmara municipal que o envia. Cada uma das restantes linhas do ficheiro fonte de dados é composta por vários pipes (“|”), sendo que cada elemento entre estes representa um campo.

De forma a que os colaboradores da Empresa possam perceber a estrutura do ficheiro fonte de dados, e posteriormente possam efetuar a configuração dos campos e especificações de leitura do ficheiro, é enviado à Empresa um ficheiro de especificações (Figura 9) em Excel pela empresa terceira que criou o ficheiro fonte de dados, onde é possível identificar os campos e especificações do ficheiro.

É possível verificar na Figura 9, que o primeiro campo tem a denominação de EB_01, e representa a data de emissão, enquanto o segundo campo tem a denominação de EB_02, e representando o identificador do destinatário, e assim sucessivamente, para cada campo da linha do ficheiro.



	A	B
1	**Campo**	**Descritivo**
2	EB_01	Data de Emissão
3	EB_02	Identificador do Destinatário
4	EB_03	Nome Destinatário
5	MOR_01	Morada 1
6	MOR_02	Morada 2
7	MOR_03	Morada 3
8	MOR_04	Morada 4
9	MOR_05	Morada 5
10	MES_C	Indicação do Mês Corrente
11	GRAF_00	Valor (Mesmo mês ano anterior)
12	GRAF_01	Valor (Mes Corrente - 11)
13	GRAF_02	Valor (Mes Corrente - 10)
14	GRAF_03	Valor (Mes Corrente - 9)
15	GRAF_04	Valor (Mes Corrente - 8)

CABEÇALHO | detalhes | colunas_teste

Figura 9 – Ficheiro de especificações com a informação do ficheiro

As linhas de detalhe do ficheiro fonte de dados (Figura 8) correspondem à informação das faturas, correspondendo cada uma das linhas a um elemento “documento” do ficheiro de dados normalizado (Figura 10). Cada campo da linha do ficheiro fonte de dados representa um subelemento de “documento” (conforme a Figura 9, que contém a denominação dada aos campos).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<factura>
    <documento>
        <EB_01>2018/09/17</EB_01>
        <EB_02>22866</EB_02>
        <EB_03>NOME1</EB_03>
        <MOR_01>RUA 1</MOR_01>
        <MOR_02>LOCALIDADE 1</MOR_02>
        <MOR_03>3080</MOR_03>
        <MOR_04>111</MOR_04>
        <MOR_05>LOCALIDADE</MOR_05>
        <MES_C>08</MES_C>
```

Figura 10 – Exemplo de um ficheiro de dados normalizado criado pelo ProcessaDados

Cada um dos elementos “documento” do ficheiro de dados normalizado, representa uma fatura, que dará origem a um documento PDF, como exemplificado na Figura 11.

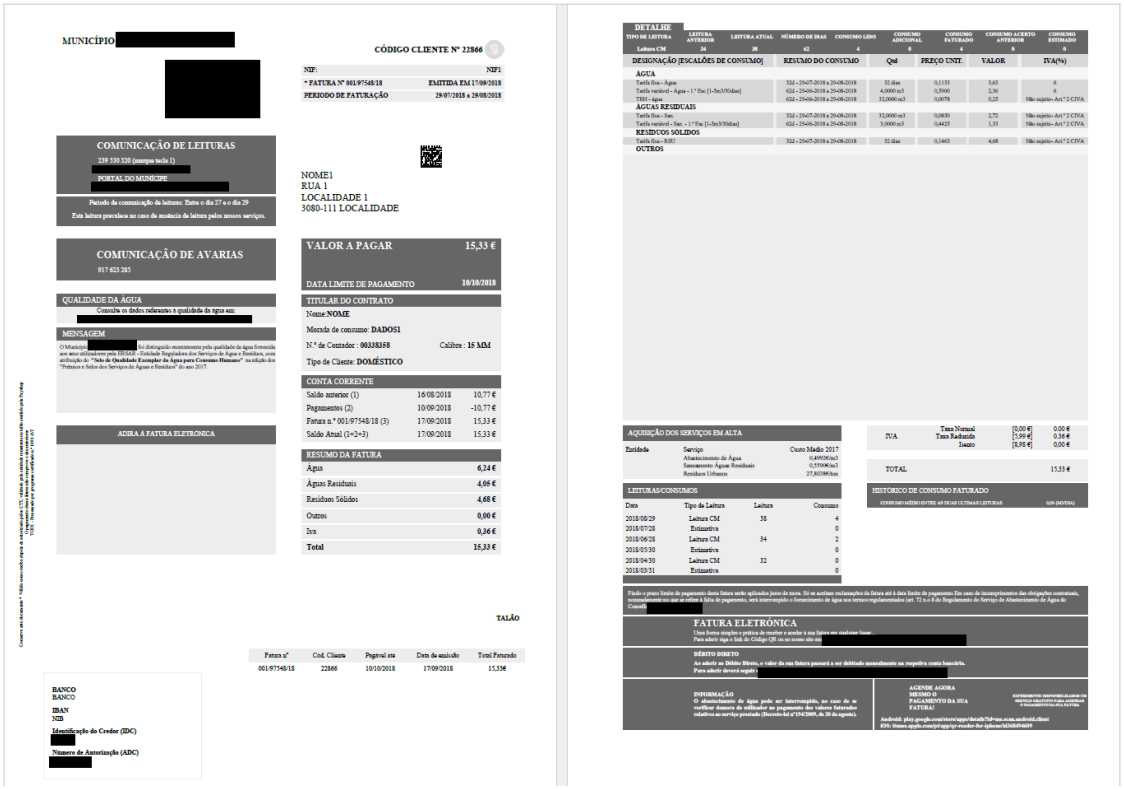


Figura 11 – Exemplo de documento PDF criado pelo ProcessaDados

De forma resumida, um ficheiro fonte de dados é transformado num ficheiro de dados normalizado, e um ficheiro de dados normalizado dá origem a um ou vários documentos PDF (Figura 12).

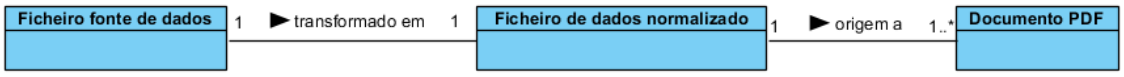


Figura 12 – Modelo de domínio em UML dos ficheiros utilizados

De forma a ser possível uma melhor perceção e conhecimento dos métodos implementados para os vários requisitos, será efetuada uma análise na secção seguinte, permitindo analisar e identificar possíveis falhas dos métodos.

2.3.1 Métodos implementados

Após efetuar a análise da arquitetura do software conforme descrito na secção 2.2, foi efetuada a análise dos métodos implementados para os vários requisitos.

Para efetuar a leitura de ficheiros fontes de dados, sendo esta a primeira etapa executada pelo ProcessaDados, conforme descrito na secção 2.2.2.3, é executada no método convertXML. Após analisar a forma utilizada para efetuar a leitura, verificou-se que o software utiliza um InputStreamReader, que é criado a partir de um FileInputStream, identificando a codificação do ficheiro, criando um fluxo de entrada, sendo depois passado para um BufferedReader para uma melhor eficiência, efetuando a leitura por linha (Figura 13). O InputStreamReader é a ponte entre o fluxo de bytes e de caracteres, uma vez que efetua a leitura de bytes, decodificando-os em caracteres, utilizando um conjunto de caracteres especificado.

```
public String convertXML(String fileTXT) {
    BufferedReader in = null;
    try {
        in = new BufferedReader(new InputStreamReader(new FileInputStream(fileTXT), "ISO-8859-1"));
        int ext = fileTXT.lastIndexOf(".");
        out = new StreamResult(fileTXT.substring(0, ext) + ".xml");
        initXML();
        String str;
        while ((str = in.readLine()) != null) {
            if (str.startsWith("20")) {
                String[] doc = str.split("\\|");
                process(doc);
            }
        }
```

Figura 13 – Código da leitura do ficheiro fonte de dados

De seguida, para efetuar a escrita de ficheiros de dados normalizados, sendo esta a segunda etapa executada pelo ProcessaDados, conforme descrito na secção 2.2.2.3. Após analisar a forma utilizada para efetuar a escrita, verificou-se que o software utiliza um SAXTransformerFactory (Figura 14), criando um objeto TransformerHandler, que inicia os vários elementos da árvore indicando o nome do elemento, o respetivo valor, e os seus atributos, terminando depois o elemento, criando uma estrutura em árvore. O AttributesImp é criado para definir os atributos dos elementos. Por fim, utiliza o StreamResult para criar o ficheiro de dados normalizados utilizando os vários elementos criados em árvore.

```
private void initXML() throws TransformerConfigurationException, SAXException {
    SAXTransformerFactory tf = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
    th = tf.newTransformerHandler();
    Transformer serializer = th.getTransformer();
    serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    serializer.setOutputProperty(OutputKeys.STANDALONE, "no");
    serializer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    serializer.setOutputProperty(OutputKeys.INDENT, "yes");
    th.setResult(out);
    th.startDocument();
    atts = new AttributesImpl();
    th.startElement("", "", "factura", atts);
}

@SuppressWarnings("empty-statement")
private void process(String[] doc) throws SAXException {
    int i;
    th.startElement("", "", "documento", atts);
    String data_emissao[] = doc[0].split("/");
    String dataEm = data_emissao[2] + "/" + data_emissao[1] + "/" + data_emissao[0];
    th.startElement("", "", "EB_01", atts);
    th.characters(dataEm.toCharArray(), 0, dataEm.length());
    th.endElement("", "", "EB_01");
    th.startElement("", "", "EB_02", atts);
    th.characters(doc[1].toCharArray(), 0, doc[1].length());
    th.endElement("", "", "EB_02");
    th.startElement("", "", "EB_03", atts);
    th.characters(doc[2].toCharArray(), 0, doc[2].length());
    th.endElement("", "", "EB_03");
```

Figura 14 – Código da escrita do ficheiro de dados normalizados

A leitura de ficheiros de dados normalizados é a terceira etapa executada pelo ProcessaDados, conforme descrito também na secção 2.2.2.3. Após analisar a forma utilizada para efetuar a leitura do ficheiro de dados normalizados, verificou-se que o software utiliza o DOM (Document Object Model), recebendo uma estrutura em árvore com os artefactos XML (elemento, nó, atributo, entre outros) do ficheiro, carregando toda a informação do ficheiro na memória, podendo não ser aconselhável para ficheiros muito grandes (Figura 15).

```
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(file);
doc.getDocumentElement().normalize();
NodeList nodes = doc.getElementsByTagName(tagInicio);
for (int i = 0; i < nodes.getLength(); i++) {
    xslFile = serverPath + "\\rotinas\\" + tagempresa + "\\" + "FS.xsl";
    Node node = nodes.item(i);
    StringBuilder xml = getStringXmlString(node);
```

Figura 15 – Código da leitura do ficheiro de dados normalizados

O método getStringXmlString é executado para cada um dos nós principais da árvore, manipulando os dados dos elementos e atributos de cada nó, inserindo-os num StringBuilder (Figura 16).

```
public static StringBuilder getStringXmlString(Node node) {
    StringBuilder doc = new StringBuilder();
    String root = node.getNodeName();
    doc.append("<").append(root);
    if (node.getAttributes().getLength() > 0) {
        doc = getAttributes(doc, node);
    }
    doc.append(">");
    for (Node child = node.getFirstChild(); child != null; child = child.getNextSibling()) {
        if (child instanceof Element) {
            Element childValue = (Element) node;
            doc.append("<").append(child.getNodeName());
            if (child.getAttributes().getLength() > 0) {
                doc = getAttributes(doc, child);
            }
            doc.append(">");
            if (child.getChildNodes().getLength() <= 1) {
                doc.append(getValue(child.getNodeName(), childValue));
            } else {
                getChildsString(child, doc);
            }
            doc.append("</").append(child.getNodeName()).append(">");
        }
    }
    doc.append("</").append(root).append(">");
    return doc;
}
```

Figura 16 – Código da manipulação de dados dos nós principais

Por fim, para efetuar a última etapa, sendo esta a criação dos documentos PDF, conforme também descrito na secção 2.2.2.3, é utilizado o FOP (Formatting Objects Processor), que é um formatador de impressão orientado por objetos de formatação XSL, utilizando o ficheiro de design PDF (Figura 17), desenhando as páginas resultantes para uma saída especificada, neste caso, para PDF. Para isso é utilizado o TransformerFactory, que é usado para criar o Transformer (Figura 18).

```
<?xml version='1.0' encoding='UTF-8' ?>
<xsl:stylesheet version="2.0"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:barcode="http://barcode4j.krysalis.org/ns">
    <xsl:output method="xml" version="1.0" indent="yes" encoding="UTF-8"/>
    <xsl:template match="documento">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
                 xmlns:fox="http://xmlgraphics.apache.org/fop/extensions">
            <fo:layout-master-set>
                <fo:simple-page-master master-name="all-pages" page-height="297mm" page-width="210mm"
                                       margin-bottom="5mm" margin-left="5mm"
                                       margin-right="5mm"
                                       margin-top="10mm">
                    <fo:region-body region-name="xsl-region-body"/>
                </fo:simple-page-master>
                <fo:page-sequence-master master-name="default-sequence">
                    <fo:repeatable-page-master-reference master-reference="all-pages"/>
                </fo:page-sequence-master>
            </fo:layout-master-set>
            <fo:page-sequence master-reference="all-pages">
                <fo:flow flow-name="xsl-region-body">
                    <fo:block-container font-family="arial" position="absolute" left="-2.5mm" width="12mm" height="300mm" top="109mm">
                        <fo:block text-align="right">
                            <fo:block-container reference-orientation="90" font-size="4.9pt" display-align="center" width="125mm" >
                                <fo:block text-align="left">
                                    Conserve este documento * Válido como recibo depois de autorizado pelos CTT, validado pela entidade
                                    remetente ou talão emitido pela Payshop
                                </fo:block>
                                <fo:block text-align="center">
                                    O pagamento duma fatura não comprova o das anteriores
                                </fo:block>
                                <fo:block text-align="center">
                                    <xsl:value-of select="CERT2"/>
                                    - Processado por programa certificado n.º <xsl:value-of select="CERT1"/> /AT
                                </fo:block>
                            </fo:block-container>
                        </fo:block>
                    </fo:block-container>
```

Figura 17 – Código do ficheiro de design PDF

```
public static void createPdf(String xml, String xslFile, Documento dadosDocumento, String serverPath) {
    try {
        // Set up FOP
        String filepath = serverPath + "\\WEB-INF\\conf\\fopconfig.xml";
        String xmlEnconding = xml.replaceAll("&", "&amp;");
        xmlEnconding = xmlEnconding.replaceAll("&amp;lt;", "&lt;");
        xmlEnconding = xmlEnconding.replaceAll("&amp;gt;", "&gt;");
        xmlEnconding = xmlEnconding.replaceAll("&amp;quot;", "&quot;");
        org.apache.fop.apps.FopFactory fopFactory = org.apache.fop.apps.FopFactory.newInstance();
        fopFactory.setBaseURL(serverPath);
        OutputStream out = new BufferedOutputStream(new FileOutputStream(new File(dadosDocumento.getFile())));
        fopFactory.setUserConfig(new File(filepath));
        Fop fop = fopFactory.newFop(MimeConstants.MIME_PDF, out);
        TransformerFactory tFactory = TransformerFactory.newInstance();
        Source xsltSrc = new StreamSource(new File(xslFile));
        Transformer transformer = tFactory.newTransformer(xsltSrc);
        String dataMatrix = dadosDocumento.getNumero();
        transformer.setParameter("dataMatrix", dataMatrix);
        FOUserAgent userAgent = fopFactory.newFOUserAgent();
        userAgent.setCreator("Creator");
        userAgent.setAuthor("Autor");
        userAgent.setTitle("PDF generation");
        Result res = new SAXResult(fop.getDefaultHandler());
        Source src = new StreamSource(new StringReader(xmlEnconding));
        transformer.transform(src, res);
        out.close();
```

Figura 18 – Código da criação dos documentos PDF

2.3.2 Eficácia e eficiência

Para uma melhor perceção do comportamento do ProcessaDados relativamente à sua eficácia e eficiência foram efetuadas testes-experiências no processamento de vários ficheiros de tamanhos diferentes e em condições de execução distintas.

O objetivo é, para o processamento desejado, medir:

- Tempo de execução;
- Quantidade de memória necessária.

Para efetuar os testes e a análise referida anteriormente, foi utilizada uma máquina Windows, com 16GB de RAM e com um CPU i7-7700HQ com 2.80GHz. Nesta máquina foi efetuada a instalação do servidor Tomcat, de forma a efetuar o mesmo processo que a Empresa.

O Tomcat foi instalado, sendo inicialmente efetuada uma análise das suas configurações de memória predefinidas de instalação (Figura 19).

Figura 19 – Configuração de memória inicial do Tomcat

Após a análise das configurações, foi possível concluir que seria necessário alterar as mesmas, de forma a ser possível ter mais memória no Tomcat, sendo então alteradas as propriedades do Tomcat (Figura 20).

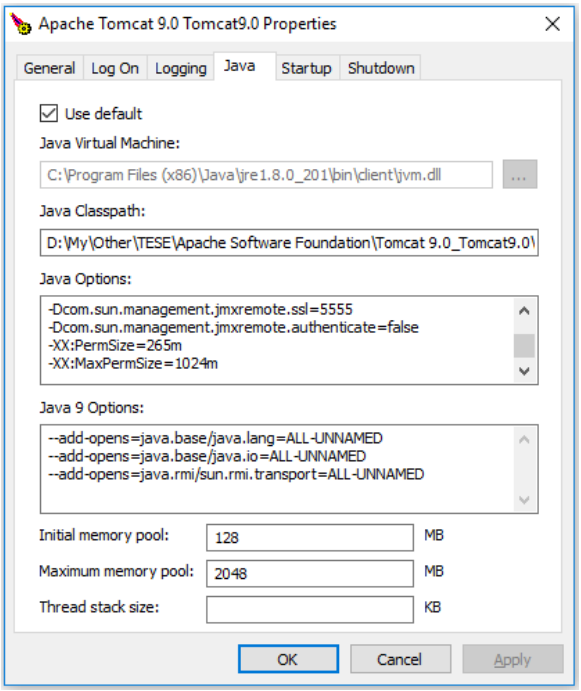


Figura 20 – Configuração de memória do Tomcat

Após efetuar a alteração, o Tomcat foi reiniciado, de forma a verificar as alterações efetuadas, podendo depois confirmar os valores no gestor do mesmo (Figura 21).

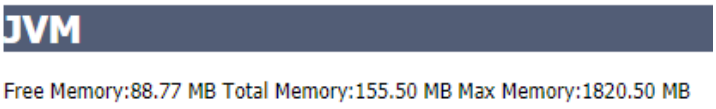


Figura 21 – Configuração de memória alterada do Tomcat

Para identificar os problemas do ProcessaDados, foram utilizados nove ficheiros fontes de dados de diferentes tamanhos (2,30MB, 3,56MB, 4,30MB, 6,53MB, 8,01MB, 10,9MB, 24MB, 27MB e 31,1MB), de forma a perceber e analisar qual o comportamento do software para os diferentes ficheiros.

Para obter os tempos de execução e memória utilizada pelo software, foi utilizado o programa Java VisualVM. Este programa obtém a informação de cada execução efetuada pelo software, indicando os milissegundos do tempo de execução, assim como os bytes da memória utilizada.

Após a execução da aplicação para os vários ficheiros, os resultados obtidos foram os apresentados na Tabela 3, indicando o tamanho do ficheiro e os respetivos valores.

Tamanho ficheiro (MB)	**Quantidade de documentos PDF**	**Tempo de execução (min)**	**Memória Utilizada (GB)**
2,3 (Ficheiro 1)	992	8,96	37,19
3,56 (Ficheiro 2)	1 536	13,56	57,80
4,3 (Ficheiro 3)	1 856	16,51	69,47
6,53 (Ficheiro 4)	2 816	25,10	105,39
8,01 (Ficheiro 5)	3 456	31,34	129,41
10,9 (Ficheiro 6)	4 736	42,34	177,34
24 (Ficheiro 7)	11 014	141,97	411,19
27 (Ficheiro 8)	12 294	189,19	460,09
31,1 (Ficheiro 9)	14 054	Erro de execução	Erro de execução

Tabela 3 – Tabela com os valores do tempo de execução e memória utilizada

De forma a perceber e comparar o tempo de execução dos vários ficheiros, foi criado o gráfico apresentado na Tabela 4. Com este gráfico podemos observar que o tempo de execução sofreu um crescimento linear relativo ao tamanho do ficheiro, já não sendo possível executar o ProcessaDados para um ficheiro com 31,1MB.

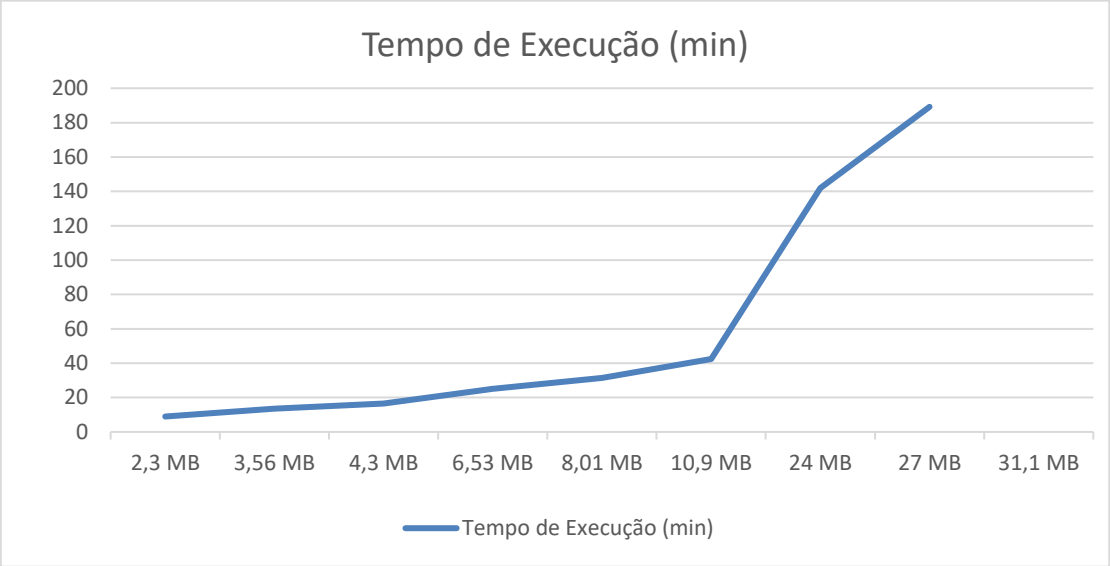


Tabela 4 – Gráfico do tempo de execução (min)

Também de forma a perceber e comparar a memória utilizada dos vários ficheiros, foi criado o gráfico apresentado na Tabela 5. Com este gráfico podemos observar que a quantidade de memória usada também sofreu um crescimento linear relativo ao tamanho do ficheiro, já não sendo possível executar o ProcessaDados para um ficheiro com 31,1MB.

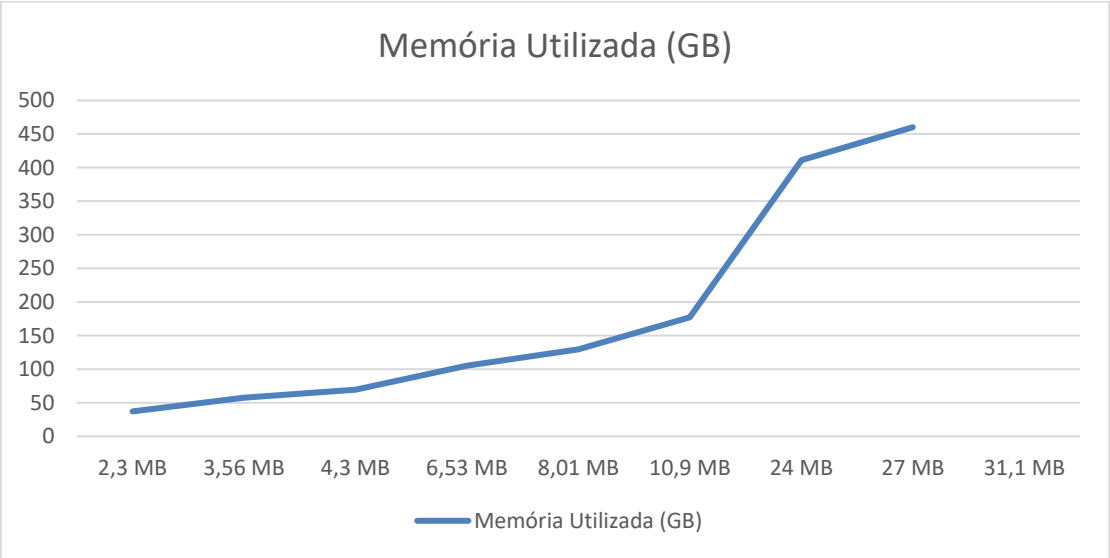


Tabela 5 – Gráfico da memória utilizada (GB)

Após analisar os vários ficheiros, e o seu comportamento a nível de tempo de execução e memória utilizada, foi possível concluir que o não processamento do último ficheiro teve origem na falta de memória, sendo esta a informação obtida pelo log do software.

As causas básicas mais comuns de falta de memória são [4]:

- Tamanho da heap ser muito pequeno;
- Mais threads do que o sistema permite;
- Software com muitas recursões;
- Software que carrega um ficheiro muito grande em memória;
- Mais pedidos de execução de software do que o sistema permite.

No ProcessaDados, o problema deve-se ao carregamento de um ficheiro muito grande em memória, uma vez que este é um dos requisitos do software, causando o não processamento do ficheiro.

Após esta análise foi possível concluir que o ProcessaDados tem:

- Eficácia em ficheiros até 27MB;
- Eficiência em tempo e memória linear.

2.4 Novos ficheiros de dados fonte

De forma a analisar os vários ficheiros fonte de dados, as várias empresas responsáveis pelo controlo e gestão dos dados dos consumidores disponibilizaram sete tipos de ficheiros de teste, havendo mais tipos de ficheiros que não foram disponibilizados.

Estes ficheiros disponibilizados foram analisados e foram encontradas inúmeras diferenças entre os mesmos, sendo que para cada um dos ficheiros fonte de dados, existe um ficheiro de especificações criado pela empresa terceira de forma a perceber a estrutura do mesmo, tendo sido também disponibilizados estes ficheiros.

Após efetuar a análise, foi possível verificar que o primeiro, segundo e terceiro ficheiro fonte de dados disponibilizado e o respetivo ficheiro de especificações são muito idênticos, conforme apresentado na Figura 22, Figura 23 e Figura 24, onde é possível ver o identificador

da linha, a descrição, o início, comprimento e fim de cada campo do ficheiro fonte de dados, variando apenas estes campos, contendo a mesma estrutura de informação.

	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	datafícheiro	3	6	8
3	00	Identificador do Destinatário	9	30	38
4	01	Tipo documento	3	1	3
5	01	Fatura eletronica	4	2	5
6	02	Saldo anterior	5	15	19
7	02	Pagamentos	22	15	36
8	02	Saldo actual	39	15	53

Figura 22 - Ficheiro de especificações com a informação do primeiro ficheiro fonte de dados

Linha ID	Descrição	Inicio	Comprimento	Fim
00	datafícheiro	3	8	10
02	Tipo documento Identificador	3	1	3
02	Fatura eletronica	4	2	5
02	Email	7	40	46
04	Tipo documento (fatura / nota de crédito)	73	20	92
04	nº fatura	93	30	122

Figura 23 - Ficheiro de especificações com a informação do segundo ficheiro fonte de dados

	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	datafícheiro	3	8	10
3	02	Identificador ficheiro	5	20	24
4	02	Tipo documento	27	40	66
5	04	Id consumidor	71	14	84
6	04	Nº consumidor	86	7	92
7	08	Destinatario	53	40	92
8	09	Morada	53	40	92

Figura 24 - Ficheiro de especificações com a informação do terceiro ficheiro fonte de dados

Ao efetuar a análise dos respetivos ficheiros fonte de dados disponibilizados, conforme se pode verificar na Figura 25, Figura 26 e Figura 27, é possível observar que a primeira linha contém um cabeçalho, contendo nas seguintes linhas a informação dos documentos, sendo que nos dois primeiros caracteres está representado o identificador da linha, contendo os respetivos campos num determinado intervalo de caracteres, sendo que existe um determinado identificador que corresponde ao início de um documento.

O segundo ficheiro fonte de dados tem a particularidade de poder ter o identificador nos dois primeiros caracteres, ou nos três primeiros, sendo que no início do documento o identificador são os dois primeiros caracteres, e quando o identificador é “99”, as leituras seguintes passam a ter o identificador nos três primeiros caracteres.

Estes três ficheiros são idênticos, variando apenas os identificadores, e o tipo de informação.

```
00300419SMLAMEGO

01D

02            0,00            0,00           10,03

04  FTR 01-11111              60          60          5

05                                                                    2019/04/29

06                                                                  Abril de 2019

07                                                                             1        1

08                                                                    2019/05/24

09  4810            Tagus           15

10NOME NOME NOME

11111111111

12RUA TESTE                                NOME NOME NOME                      1

13CAMBRES                                  RUA TESTE

14CAMBRES                                  PENELAS
```

Figura 25 – Exemplo do primeiro ficheiro fonte de dados

```
00190319

02D

04                                                          *FATURA/RECIBO N.° FTR 01/11111

05

14                                         NOME NOME NOME

15NOME NOME NOME                           R TESTE

16R TESTE                                  3700-000 SAO JOAO DA MADEIRA

17111111111

18  32109            172
```

Figura 26 - Exemplo do segundo ficheiro fonte de dados

```
00060519


02  DGCL 10 - V3
AVISO


04                                                          Consumidor n.°
11

08                                         NOME NOME
NOME

09                                         R TESTE,
50

10                                         S JOAO
MADEIRA

11                                         3700-000 SAO JOAO DA
MADEIRA
```

Figura 27 - Exemplo do terceiro ficheiro fonte de dados

Ao efetuar a análise do quarto ficheiro fonte de dados disponibilizado e o respetivo ficheiro de especificações, conforme apresentado na Figura 28, é possível ver o identificador da linha, a descrição, o início, comprimento e fim de cada campo do ficheiro fonte de dados, conforme nos ficheiros analisados anteriormente.

	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	Nome do ficheiro	3	9	11
3	00	Data	12	10	21
4	00	Nº sequencial	22	2	23
5	00	Nome do ficheiro antigo	24	23	46
6	00	Código Entidade SIBS	47	5	51
7	11	NUD	3	16	18
8	11	Tipo Executado	19	3	21
9	11	NIF	22	9	30

Figura 28 - Ficheiro de especificações com a informação do quarto ficheiro fonte de dados

Conforme se pode verificar na Figura 29, é possível observar que a primeira linha contém um cabeçalho, contendo nas seguintes linhas a informação dos documentos, sendo que nos dois primeiros caracteres está representado o identificador da linha, contendo os respetivos campos num determinado intervalo de caracteres, sendo que uma linha corresponde a um documento.

```
00SEFCTT_PP2019-04-0301SEFCTT_PP_2019-03-05_0121179
1120190403000004144NIF111111111NOME NOME NOME
DIAS
                                    RUA TESTE
5
                                4150-000PORTO                                    03 de Abril     de
2019000208/2018015002316030620901848080802018000266400000001904300002080157000000000026.642019/04/30
1120190403000004145NIF111111111NOME NOME NOME
MOUTINHO
                                 BAIR TESTE
C21
                             4100-000PORTO                                  03 de Abril     de
2019000632/2018015002316148620901848080802018000315900000001904300006320153000000000031.592019/04/30
```

Figura 29 - Exemplo do quarto ficheiro fonte de dados

Ao efetuar a análise do quinto ficheiro fonte de dados disponibilizado e o respetivo ficheiro de especificações, conforme apresentado na Figura 30, é possível ver como é composta uma linha do ficheiro fonte de dados, verificando que o primeiro caracter identifica se o campo é para ser colocado no documento como Normal ou Bold, os quatro caracteres seguintes correspondem ao nome do campo, e o restante da informação corresponde à informação.

	Local	Campo
2	01	Normal ou Bold
3	2-5	Nome do Campo
4	6-..	Valor do Campo
5		

Figura 30 - Ficheiro de especificações com a informação do quinto ficheiro fonte de dados

Conforme se pode verificar na Figura 31, não existe nenhuma linha de cabeçalho, sendo que quando a linha é iniciada pelo caracter "#" corresponde ao início de um documento, contendo nas restantes linhas a informação dos documentos, sendo que o primeiro caractere corresponde a identificar se o texto será escrito com a formatação Normal ou Bold, sendo os quatro caracteres seguintes o nome do campo, e os restantes caracteres o texto correspondente. Neste tipo de ficheiro podem existir linhas em que o texto contém o caracter ";", sendo este utilizado para efetuar uma separação de subcampos, sendo utilizado por exemplo no detalhe da fatura, separando a descrição, da quantidade, do valor do iva, do valor gasto, entre outros.

Este tipo de ficheiro fonte de dados é muito parecido com o primeiro ficheiro fonte de dados, mudando apenas os identificadores e o início, comprimento e fim dos campos.

```
#1
NADC1PT50101269
NADC200005147398
NCCL1201920103881
NCLT100051473
NCNS122 m3
NCNT1CONVERSAO
NCNT270/00000055859
NCNT350
NCNT42010.08.12
NCOD012692740073E0101708C
NDAT12019.03.22
NDAT22019.04.17
NDES1Valor a receber em 2019.03.22       98.09
NDES2A incluir no próximo processamento
BDET1Contas de Água;;;42,41;
NDET1  Escalão Único  ;22,0 m3;1,1700;25,74;6,00
```

Figura 31 - Exemplo do quinto ficheiro fonte de dados

Ao efetuar a análise do sexto ficheiro fonte de dados disponibilizado e o respetivo ficheiro de especificações, conforme apresentado na Figura 32, é possível ver o identificador da linha, a descrição, o início, comprimento e fim de cada campo do ficheiro fonte de dados, sendo idêntico aos três primeiros ficheiro de especificações apresentados nesta secção.

1	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	Nome ficheiro	3	6	8
3	00	Data	9	10	18
4	00	Nº sequencial	19	2	20
5	00	Código entidade	21	5	25
6	00	Código entidade ctt	26	5	30
7	11	Nº avisos SIBS	3	9	11
8	11	Forma citação	12	1	12
9	11	Nº executado	13	9	21
10	11	NIF	22	9	30
11	11	Tipo executado	31	1	31
12	11	Nome	32	160	191
13	11	Morada	191	161	351
14	11	Código postal	352	8	359

Figura 32 - Ficheiro de especificações com a informação do sexto ficheiro fonte de dados

Conforme se pode verificar na Figura 33, é possível observar que a primeira linha contém um cabeçalho, contendo nas seguintes linhas a informação dos documentos, sendo que nos dois primeiros caracteres está representado o identificador da linha, contendo os respetivos campos num determinado intervalo de caracteres, sendo que existe um determinado identificador que corresponde ao início de um documento.

Este tipo de ficheiro fonte de dados também é muito parecido com os três primeiros ficheiros fonte de dados, mudando apenas os identificadores e o início, comprimento e fim dos campos.

```
1  00SEFCTT2019-03-13012117990184
2  11000011111111178324620466135800ME NOME
   NOME
                              RUA
   TESTE
                                    4100-000PORTO                                    1312201900147494N
3  121312201900147494  ÁguasPortoUBS                    00000000020.74
4  12  1312201900147494              172548700000000004.13
5  12  1312201900147494              172548800000000001.29
6  1300000000005.4200000000020.740022019-04-200018621746209018419001474940001579000000019042010178324660000000015.79Quinze
   Euros e Setenta e Nove
   Cêntimos

7  24131220190014749410 1-ÁguasPortoUBS   172548700012746132           2018-12 2019-01Tar. Disponibil.I/C/T/O-CJM     RUA
   TESTE                                      4100-000 PORTO                                        00000000002.89
8  24                                                                  2018-12 2019-01IVA
   I/C/T/O-SJM
   00000000000.23
9  24                                                                  2018-12 2019-01Saneamento
   I/C/T/O-CJM
   00000000001.01
```

Figura 33 - Exemplo do sexto ficheiro fonte de dados

O último ficheiro corresponde ao ficheiro já processado pelo ProcessaDados, já tendo sido efetuada esta análise na secção 2.3.

Após efetuar a análise destes sete tipos de ficheiros fontes de dados, e sabendo da existência de outros não disponibilizados, foi efetuada uma pesquisa de forma a analisar as várias soluções existentes para a configuração dos campos e especificações de leitura, sendo

possível concluir que não existia nenhum software nem ferramenta capaz de efetuar o pretendido, sendo necessário efetuar a criação de um software para este requisito.

2.5 Sistematização de Requisitos

Esta secção tem dois objetivos:

- Sistematizar os requisitos funcionais;
- Sistematizar os requisitos não funcionais;

enunciados nos capítulos anteriores.

2.5.1 Requisitos funcionais

Os requisitos funcionais descrevem as funcionalidades que se espera que o software disponibilize, de uma forma completa e consistente. São as funcionalidades que o colaborador da Empresa efetua e as que espera que o sistema ofereça, atendendo aos propósitos para o qual o sistema será desenvolvido. Estes requisitos abordam o que o software deve fazer e as funcionalidades que deve ser possível realizar.

Esses requisitos são os descritos na Figura 34 em forma de casos de uso:

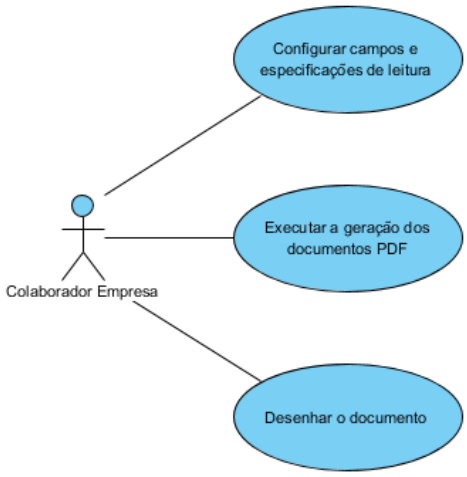


Figura 34 – Diagrama de casos de uso do ProcessaDados em UML

De forma a compreender melhor os requisitos da Figura 34, foi criado um diagrama BPMN de forma a retratar o fluxo existente, conforme apresentado na Figura 35.

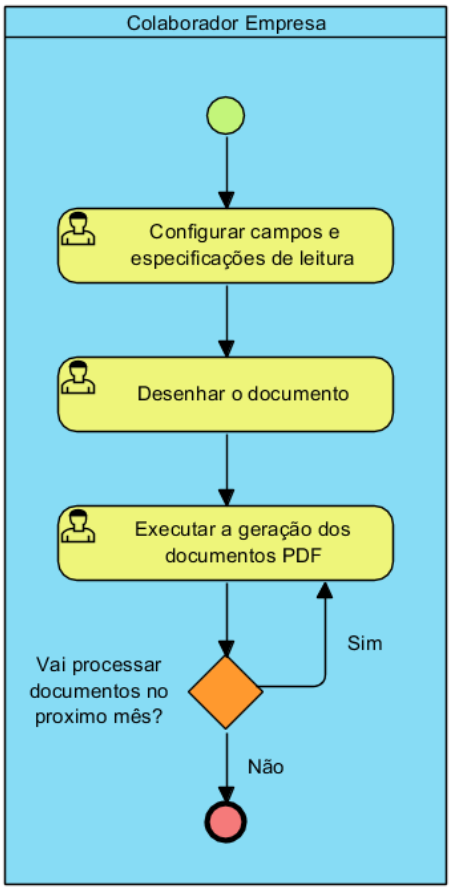


Figura 35 – Diagrama BPMN do fluxo dos requisitos do ProcessaDados

Mediante as figuras acima, é possível usufruir da aplicação com o seguinte fluxo:

- Efetuar a configuração dos campos e as especificações de leitura;
- Efetuar o desenho do documento;
- Efetuar a geração dos documentos PDF, sendo que este requisito pode ser repetido, caso no mês seguinte exista também o processamento dos documentos.

2.5.2 Requisitos não funcionais

Os requisitos não funcionais da aplicação, que já têm vindo a ser descritos no decorrer do documento, estão classificados segundo o modelo de FURPS+. Esses requisitos são os apresentados na tabela seguinte (Tabela 6), contendo a referência à sua classificação no modelo de FURPS+:

Classificação FURPS+	Requisito não funcional
P -> Performance	Menor tempo de execução.
P -> Performance	Menor tamanho de memória utilizada.
F -> Funcionalidade	Processamento de ficheiros superior a 27MB.
S -> Suportabilidade	Adaptabilidade e configurabilidade.
S -> Suportabilidade	Manutenibilidade, na medida em que é possível reutilizar o componente de forma fácil.
+ -> Design e Implementação	Adoção de boas práticas.

Tabela 6 – Requisitos não funcionais seguindo o modelo de FURPS+

2.6 Síntese

Neste capítulo foi efetuada a análise de negócio, sendo feita a descrição do conceito de negócio, dos vários processos e intervenientes, assim como foi identificada e analisada a oportunidade, e apresentado o modelo Canvas.

Foi também efetuada a arquitetura do software existente, efetuando a descrição da arquitetura utilizando o modelo 4+1 tanto para a granularidade de sistema como de aplicação, sendo depois identificadas e analisadas as tecnologias utilizadas no desenvolvimento e na implantação deste, analisando a eficácia e eficiência.

Por fim, foram sistematizado os requisitos do sistema, tanto funcionais como não funcionais.

3 Estado da Arte

Em função da análise ao sistema existente e dos requisitos de negócio, este capítulo apresenta o estado da arte relacionado com o software e objetivos, apresentando várias soluções existentes para os problemas identificados:

- Problema 1, configuração dos campos e especificações de leitura do ficheiro fonte de dados (leitura e escrita dos ficheiros de configurações e especificações);
- Problema 2, leitura de ficheiros fonte de dados e escrita de ficheiros de dados normalizado;
- Problema 3, design dos documentos PDF;
- Problema 4, criação de documentos PDF;
- Problema 5, leitura de ficheiros de dados normalizados e criação de documentos PDF.

Os seguintes capítulos apresentam, analisam e comparam soluções e abordagens para a resolução dos vários problemas detetados no ProcessaDados.

3.1 Problema 1: configuração de leitura de ficheiros

De acordo com a análise efetuada na secção 2.4, foi possível concluir que é necessário efetuar a criação de um novo software. Para a criação deste novo software, concluiu-se que o melhor método seria o desenvolvimento de uma UI, permitindo ao colaborador da Empresa configurar todos os campos necessários e as especificações de leitura, sendo criado um ficheiro XML denominado de ficheiro de configurações e especificações com essa informação.

De acordo com a pesquisa e de forma a analisar as várias soluções existentes para efetuar a escrita e leitura do ficheiro de configurações e especificações, analisaram-se vários métodos, descritos nas secções seguintes.

3.1.1 DOM

O DOM (Document Object Model) é uma convenção para a representação e interação com objetos em documentos HTML, XHTML e XML, criando uma árvore de objetos que representa o conteúdo e a organização dos dados no documento, carregando todo o ficheiro em memória para processamento, podendo ser desadequado para ficheiros muito grandes.

3.1.2 SAX

O SAX (Simple API for XML), é uma API para XML, em que o analisador começa no início do documento e passa cada parte do mesmo na sequência em que o encontra, acedendo ao conteúdo através de eventos. Esta API foi realizada em Java, embora atualmente esteja disponível para outras linguagens mesmo sem um padrão.

As principais vantagens do SAX são as seguintes:

- Menor gasto de memória, uma vez que não guarda todo o documento na memória, apenas mantém em memória as tags externas à que está a ser lida;
- Processamento em fluxo, uma vez que é o ideal para leituras contínuas no disco ou através da rede.

A sua principal desvantagem é a potencial impossibilidade de validar o documento através de uma definição de tipo de documento, uma vez que em certos casos só pode ser realizada se o documento estiver em memória.

3.1.3 JAXB

O JAXB (Java Architecture for XML Binding) que fornece uma API e ferramentas que automatizam o mapeamento entre documentos XML e objetos Java, permitindo armazenar e recuperar dados na memória em qualquer formato XML, sem a necessidade de implementar um conjunto específico de rotinas de leitura e escrita de XML para a estrutura de classes Java.

As principais vantagens do JAXB são as seguintes:

- Fornece uma forma eficiente de mapeamento;

- Fornece um padrão de mapeamento;
- Permite aceder aos dados em ordem não sequencial;
- Usa a memória de maneira eficiente, uma vez que a árvore de objetos é mais eficiente em termos de uso de memória do que as árvores baseadas em DOM.

3.2 Problema 2: leitura de ficheiros extensos

De acordo com a pesquisa e de forma a analisar as várias soluções existentes para efetuar a leitura de ficheiros extensos de fonte de dados, foram analisados vários métodos de leitura de ficheiro de texto linha a linha, descritos nas secções seguintes.

A análise das soluções existentes para a escrita de ficheiros de dados normalizados são as descritas nas secções 3.1.1 e 3.1.2.

3.2.1 ReadBufferReader

O ReadBufferReader é um método de leitura que efetua a mesma através de um fluxo de entrada de caracteres, fazendo um buffer (BufferedReader) para uma leitura eficiente de caracteres, matrizes e linhas.

3.2.2 ReadLine

O ReadLine é um método que efetua a leitura de todas as linhas do ficheiro, garantindo que o mesmo é fechado quando todos os bytes forem lidos ou ocorra um erro em tempo de execução. Os bytes do ficheiro são decodificados em caracteres usando o conjunto de caracteres especificado (e.g. UTF_8, entre outros).

3.2.3 ReadScanner

O ReadScanner é um método que efetua a leitura utilizando um scanner de texto simples, através de um inputStream. O scanner interrompe a entrada usando um delimitador, que por padrão corresponde ao espaço em branco.

3.2.4 ReadSoftware

O ReadSoftware é um método que efetua a leitura utilizando buffer (BufferedReader), InputStreamReader e FileInputStream, sendo este o serviço utilizado pelo software existente.

3.2.5 Análise das soluções

De acordo com a pesquisa e de forma a analisar as várias soluções existentes para a leitura de ficheiros extensos de fonte de dados descritas nas secções anteriores (3.2.1, 3.2.2, 3.2.3 e 3.2.4) e para a escrita de ficheiros de dados normalizados descritas nas secções 3.1.1 e 3.1.2, foi efetuada uma aplicação com o intuito de testar as várias soluções, de forma a analisar qual a mais eficaz e eficiente.

Esta aplicação é composta pela combinação dos quatro métodos de leitura, e os dois de escrita, resultando em oito serviços, efetuando a leitura um ficheiro de 10,9MB (o Ficheiro 6, utilizado na secção 2.3.2), sendo executada a aplicação seis vezes para cada um dos serviços criados, de forma a analisar e comparar os resultados obtidos.

Esta aplicação é executada a partir do Tomcat, utilizando o Java VisualVM para obter os tempos de execução e memória utilizada para os respetivos serviços, efetuando todos eles a leitura por linha, escrevendo a linha lida num ficheiro de logs.

O método WriteXMLusingDOM efetua a escrita através de DOM, conforme a secção 3.1.1, e o método WriteXMLusingSoftware efetua a escrita através de SAX, conforme a secção 3.1.2.

O primeiro serviço é denominado de ReadBufferReaderWriteXMLusingDOMService (Figura 36), e utiliza os métodos ReadBufferReader para leitura, e o WriteXMLusingDOM para escrita.

```java
public static void read(String fileName, String fileNameXML) throws FileNotFoundException, IOException,
        ParserConfigurationException, TransformerConfigurationException, TransformerException {
    try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
        String s;
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.newDocument();
        Element rootElement = doc.createElement("teste");
        doc.appendChild(rootElement);
        while ((s = br.readLine()) != null) {
            log.info(s);
            String[] lineSplit = s.split("\\|");
            Element documento = doc.createElement("documento");
            rootElement.appendChild(documento);
            for (int i = 0; i < lineSplit.length; i++) {
                String nameLine = "linha" + i;
                Element linha = doc.createElement(nameLine);
                linha.appendChild(doc.createTextNode(lineSplit[i]));
                documento.appendChild(linha);
            }
        }
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.METHOD, "xml");
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(fileNameXML));
        transformer.transform(source, result);
    }
}
```

Figura 36 – Serviço ReadBufferReaderWriteXMLusingDOMService

O segundo é denominado de ReadBufferReaderWriteXMLusingSoftwareService (Figura 37), e utiliza os métodos ReadBufferReader para leitura, e o WriteXMLusingSoftware para escrita.

```java
public static void read(String fileName, String fileNameXML) throws FileNotFoundException,
        IOException, TransformerConfigurationException, SAXException {
    try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
        String s;
        StreamResult out = new StreamResult(fileNameXML);
        SAXTransformerFactory tf = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
        TransformerHandler th = tf.newTransformerHandler();
        Transformer serializer = th.getTransformer();
        serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        serializer.setOutputProperty(OutputKeys.STANDALONE, "no");
        serializer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
        serializer.setOutputProperty(OutputKeys.INDENT, "yes");
        th.setResult(out);
        th.startDocument();
        AttributesImpl atts = new AttributesImpl();
        th.startElement("", "", "teste", atts);
        while ((s = br.readLine()) != null) {
            log.info(s);
            String[] lineSplit = s.split("\\|");
            th.startElement("", "", "documento", atts);
            for (int i = 0; i < lineSplit.length; i++) {
                String nameLine = "linha" + i;
                th.startElement("", "", nameLine, atts);
                th.characters(lineSplit[i].toCharArray(), 0, lineSplit[i].length());
                th.endElement("", "", nameLine);
            }
            th.endElement("", "", "documento");
        }
        th.endElement("", "", "teste");
        th.endDocument();
    }
}
```

Figura 37 – Serviço ReadBufferReaderWriteXMLusingSoftwareService

O terceiro método é denominado de ReadLineWriteXMLusingDOMService (Figura 38), e utiliza os métodos ReadLine para leitura, e o WriteXMLusingDOM para escrita.

```
public static void read(String fileName, String fileNameXML) throws FileNotFoundException,
        IOException, TransformerConfigurationException, TransformerException, ParserConfigurationException {
    List<String> lines = Files.readAllLines(Paths.get(fileName), StandardCharsets.UTF_8);
    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    Document doc = dBuilder.newDocument();
    Element rootElement = doc.createElement("teste");
    doc.appendChild(rootElement);
    lines.forEach((s) -> {
        log.info(s);
        String[] lineSplit = s.split("\\|");
        Element documento = doc.createElement("documento");
        rootElement.appendChild(documento);
        for (int i = 0; i < lineSplit.length; i++) {
            String nameLine = "linha" + i;
            Element linha = doc.createElement(nameLine);
            linha.appendChild(doc.createTextNode(lineSplit[i]));
            documento.appendChild(linha);
        }
    });
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    transformer.setOutputProperty(OutputKeys.METHOD, "xml");
    transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(new File(fileNameXML));
    transformer.transform(source, result);
}
```

Figura 38 – Serviço ReadLineWriteXMLusingDOMService

O quarto é denominado de ReadLineWriteXMLusingSoftwareService (Figura 39), e utiliza os métodos ReadLine para leitura, e o WriteXMLusingSoftware para escrita.

```
public static void read(String fileName, String fileNameXML) throws FileNotFoundException,
        IOException, TransformerConfigurationException, SAXException {
    List<String> lines = Files.readAllLines(Paths.get(fileName), StandardCharsets.UTF_8);
    StreamResult out = new StreamResult(fileNameXML);
    SAXTransformerFactory tf = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
    TransformerHandler th = tf.newTransformerHandler();
    Transformer serializer = th.getTransformer();
    serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    serializer.setOutputProperty(OutputKeys.STANDALONE, "no");
    serializer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    serializer.setOutputProperty(OutputKeys.INDENT, "yes");
    th.setResult(out);
    th.startDocument();
    AttributesImpl atts = new AttributesImpl();
    th.startElement("", "", "teste", atts);
    lines.forEach((s) -> {
        try {
            log.info(s);
            String[] lineSplit = s.split("\\|");
            th.startElement("", "", "documento", atts);
            for (int i = 0; i < lineSplit.length; i++) {
                String nameLine = "linha" + i;
                th.startElement("", "", nameLine, atts);
                th.characters(lineSplit[i].toCharArray(), 0, lineSplit[i].length());
                th.endElement("", "", nameLine);
            }
            th.endElement("", "", "documento");
        } catch (SAXException ex) {
            log.error("read - SAXException: " + ex);
        }
    });
    th.endElement("", "", "teste");
    th.endDocument();
}
```

Figura 39 – Serviço ReadLineWriteXMLusingSoftwareService

O quinto serviço é denominado de ReadScannerWriteXMLusingDOMService (Figura 40), e utiliza os métodos ReadScanner para leitura, e o WriteXMLusingDOM para escrita.

```
public static void read(String fileName, String fileNameXML) throws FileNotFoundException, IOException,
        TransformerException, ParserConfigurationException {
    try (InputStream is = new FileInputStream(fileName); Scanner s = new Scanner(is, StandardCharsets.UTF_8.name())) {
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.newDocument();
        Element rootElement = doc.createElement("teste");
        doc.appendChild(rootElement);
        while (s.hasNextLine()) {
            String line = s.nextLine();
            log.info(line);
            String[] lineSplit = line.split("\\|");
            Element documento = doc.createElement("documento");
            rootElement.appendChild(documento);
            for (int i = 0; i < lineSplit.length; i++) {
                String nameLine = "linha" + i;
                Element linha = doc.createElement(nameLine);
                linha.appendChild(doc.createTextNode(lineSplit[i]));
                documento.appendChild(linha);
            }
        }
        s.close();
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.METHOD, "xml");
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(fileNameXML));
        transformer.transform(source, result);
    }
}
```

Figura 40 – Serviço ReadScannerWriteXMLusingDOMService

O sexto é denominado de ReadScannerWriteXMLusingSoftwareService (Figura 41), e utiliza os métodos ReadScanner para leitura, e o WriteXMLusingSoftware para escrita.

```
public static void read(String fileName, String fileNameXML) throws FileNotFoundException, IOException,
        TransformerConfigurationException, SAXException {
    try (InputStream is = new FileInputStream(fileName); Scanner s = new Scanner(is, StandardCharsets.UTF_8.name())) {
        StreamResult out = new StreamResult(fileNameXML);
        SAXTransformerFactory tf = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
        TransformerHandler th = tf.newTransformerHandler();
        Transformer serializer = th.getTransformer();
        serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        serializer.setOutputProperty(OutputKeys.STANDALONE, "no");
        serializer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
        serializer.setOutputProperty(OutputKeys.INDENT, "yes");
        th.setResult(out);
        th.startDocument();
        AttributesImpl atts = new AttributesImpl();
        th.startElement("", "", "teste", atts);
        while (s.hasNextLine()) {
            String line = s.nextLine();
            log.info(line);
            String[] lineSplit = line.split("\\|");
            th.startElement("", "", "documento", atts);
            for (int i = 0; i < lineSplit.length; i++) {
                String nameLine = "linha" + i;
                th.startElement("", "", nameLine, atts);
                th.characters(lineSplit[i].toCharArray(), 0, lineSplit[i].length());
                th.endElement("", "", nameLine);
            }
            th.endElement("", "", "documento");
        }
        s.close();
        th.endElement("", "", "teste");
        th.endDocument();
    }
}
```

Figura 41 – Serviço ReadScannerWriteXMLusingSoftwareService

O sétimo serviço é denominado de ReadSoftwareWriteXMLusingDOMService (Figura 42), e utiliza os métodos ReadSoftware para leitura, e o WriteXMLusingDOM para escrita.

```
public static void read(String fileName, String fileNameXML) throws FileNotFoundException,
        IOException, ParserConfigurationException, TransformerException {
    BufferedReader br;
    br = new BufferedReader(new InputStreamReader(new FileInputStream(fileName), "UTF-8"));
    String s;
    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    Document doc = dBuilder.newDocument();
    Element rootElement = doc.createElement("teste");
    doc.appendChild(rootElement);
    while ((s = br.readLine()) != null) {
        log.info(s);
        String[] lineSplit = s.split("\\|");
        Element documento = doc.createElement("documento");
        rootElement.appendChild(documento);
        for (int i = 0; i < lineSplit.length; i++) {
            String nameLine = "linha" + i;
            Element linha = doc.createElement(nameLine);
            linha.appendChild(doc.createTextNode(lineSplit[i]));
            documento.appendChild(linha);
        }
    }
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    transformer.setOutputProperty(OutputKeys.METHOD, "xml");
    transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(new File(fileNameXML));
    transformer.transform(source, result);
}
```

Figura 42 – Serviço ReadSoftwareWriteXMLusingDOMService

Por fim, o último é denominado de ReadSoftwareWriteXMLusingSoftwareService (Figura 43), e utiliza os métodos ReadSoftware para leitura, e o WriteXMLusingSoftware para escrita.

```
public static void read(String fileName, String fileNameXML) throws FileNotFoundException,
        IOException, TransformerConfigurationException, SAXException {
    BufferedReader br;
    br = new BufferedReader(new InputStreamReader(new FileInputStream(fileName), "UTF-8"));
    String s;
    StreamResult out = new StreamResult(fileNameXML);
    SAXTransformerFactory tf = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
    TransformerHandler th = tf.newTransformerHandler();
    Transformer serializer = th.getTransformer();
    serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    serializer.setOutputProperty(OutputKeys.STANDALONE, "no");
    serializer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
    serializer.setOutputProperty(OutputKeys.INDENT, "yes");
    th.setResult(out);
    th.startDocument();
    AttributesImpl atts = new AttributesImpl();
    th.startElement("", "", "teste", atts);
    while ((s = br.readLine()) != null) {
        log.info(s);
        String[] lineSplit = s.split("\\|");
        th.startElement("", "", "documento", atts);
        for (int i = 0; i < lineSplit.length; i++) {
            String nameLine = "linha" + i;
            th.startElement("", "", nameLine, atts);
            th.characters(lineSplit[i].toCharArray(), 0, lineSplit[i].length());
            th.endElement("", "", nameLine);
        }
        th.endElement("", "", "documento");
    }
    th.endElement("", "", "teste");
    th.endDocument();
}
```

Figura 43 – Serviço ReadSoftwareWriteXMLusingSoftwareService

Após executar cada um dos serviços seis vezes, foi possível obter os tempos de execução apresentados na Tabela 7. Com os valores obtidos, é possível observar que existem três serviços que foram os mais rápidos, com tempos muito semelhantes.

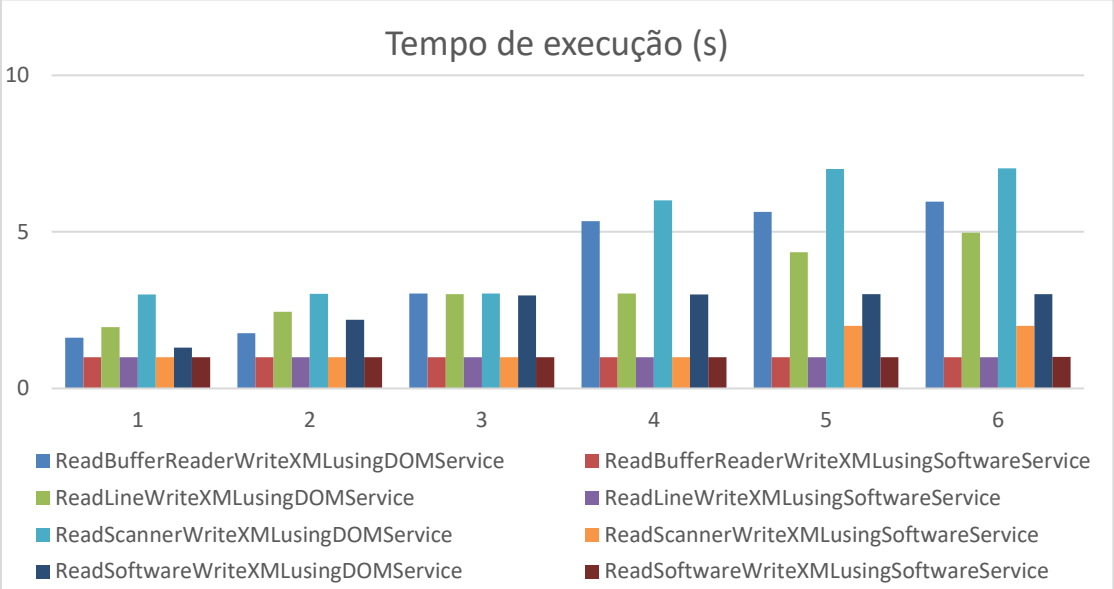


Tabela 7 – Gráfico do tempo de execução dos serviços

De forma a ser mais percetível a diferença entre os três serviços mais rápidos, foi criada a Tabela 8, podendo-se concluir que o serviço mais rápido foi o ReadBufferReaderWriteXMLusingSoftwareService, obtendo sempre o menor tempo de execução.

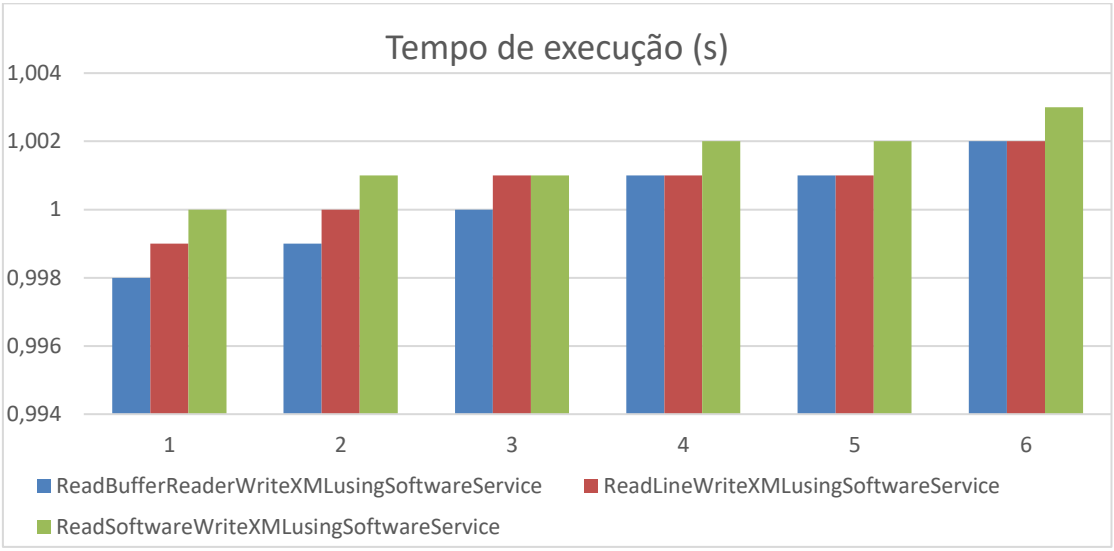


Tabela 8 - Gráfico do tempo de execução dos três serviços mais rápidos

Após a execução referida anteriormente, foi também possível obter a memória utilizada apresentada na Tabela 9. Com os valores obtidos, é possível observar que também existem três serviços que utilizam menos memória, com valores muito semelhantes.

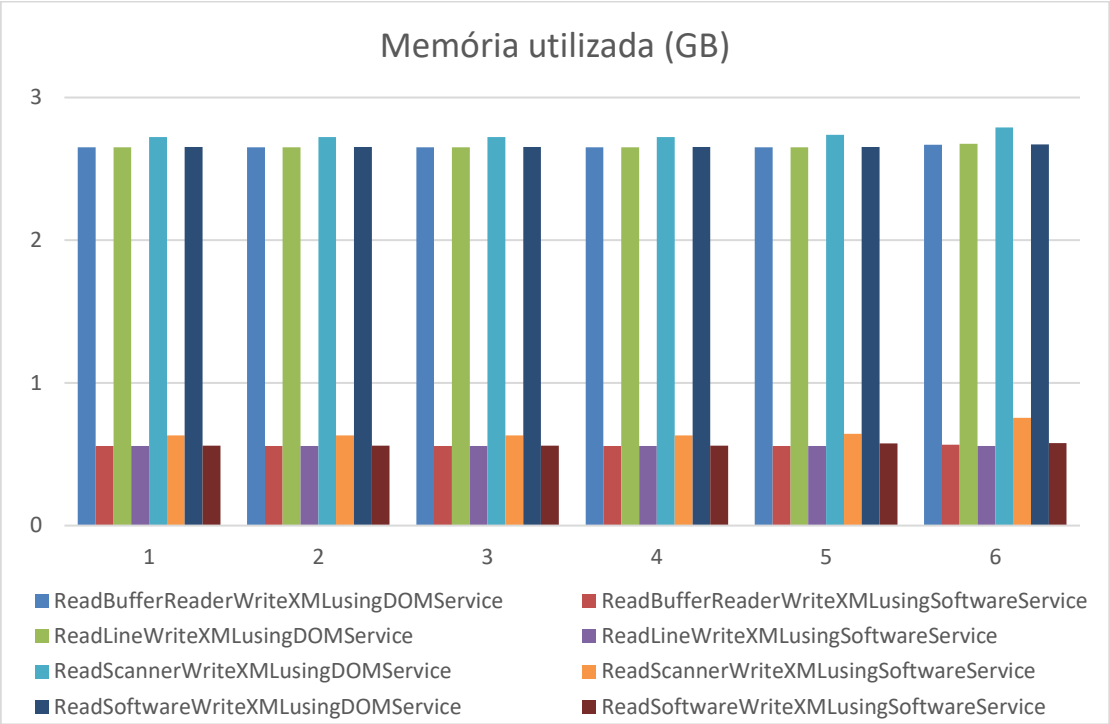


Tabela 9 – Gráfico da memória utilizada dos serviços

De forma a ser mais percetível a diferença entre os três serviços com menos memória utilizada, foi criada a Tabela 10, podendo-se concluir que os serviços ReadBufferReaderWriteXMLusingSoftwareService e ReadLineWriteXMLusingSoftwareService foram os que menos memória utilizaram, sendo a diferença mínima entre eles.

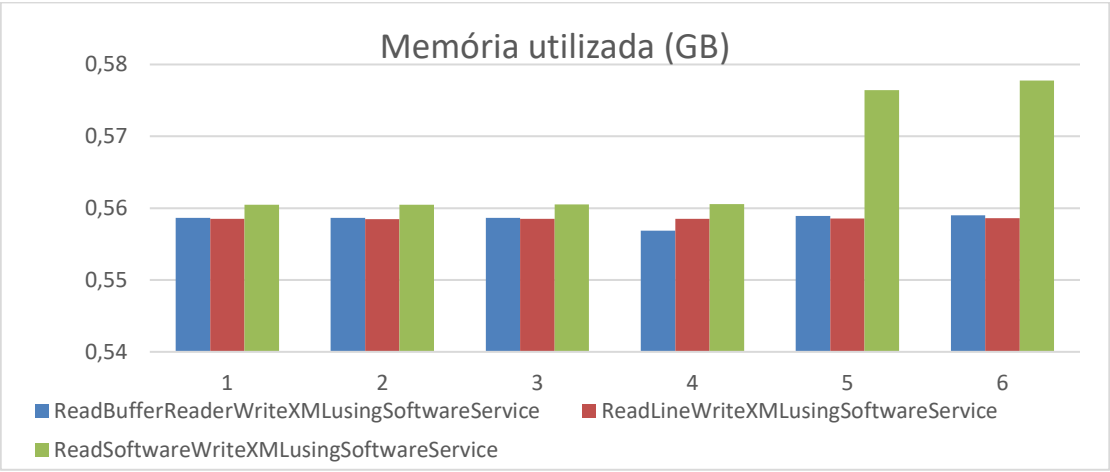


Tabela 10 - Gráfico da memória utilizada dos três serviços com menos memória utilizada

Após esta análise, foi possível concluir que o serviço que deverá ser utilizado será o ReadBufferReaderWriteXMLusingSoftwareService, uma vez foi sempre o mais rápido, sendo que as diferenças de memória utilizada com o serviço ReadLineWriteXMLusingSoftwareService são mínimas.

3.3 Problema 3: design de documentos PDF

De acordo com a pesquisa e de forma a analisar as várias soluções existentes para efetuar o design dos documentos, analisaram-se várias tecnologias, descritas das secções seguintes.

3.3.1 BIRT

O BIRT (Business Intelligence e Reporting Tools) é uma ferramenta *open source*, que permite efetuar o desenho de documentos, sendo um projeto de software de alto nível da fundação Eclipse [57]. O BIRT possui um componente de desenho de documentos dentro do IDE do Eclipse (Figura 44).

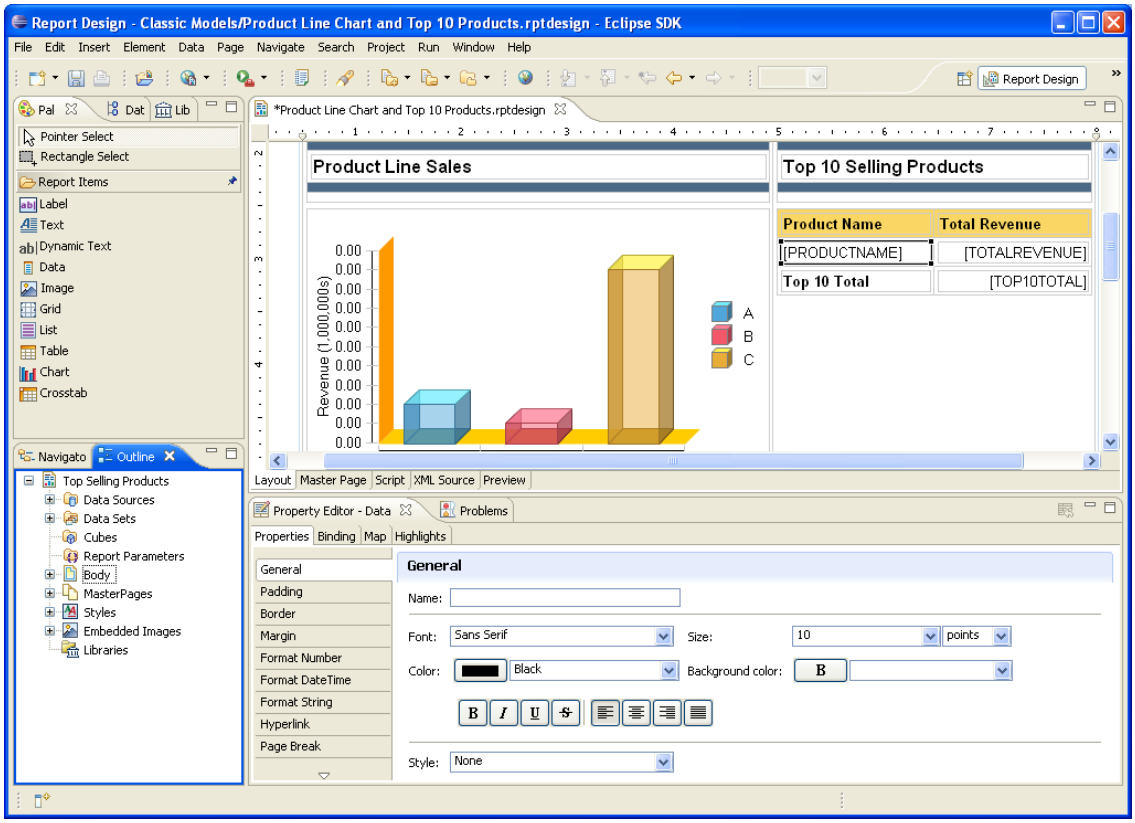


Figura 44 – UI do Eclipse para efetuar o desenho de documentos

O BIRT possui um componente de tempo de execução para gerar relatórios que podem ser implementados em qualquer ambiente Java.

O BIRT também permite que seja efetuada a criação dos documentos, podendo gravar em vários formatos incluindo Microsoft Excel, PDF, PostScript, Microsoft Word ou Microsoft PowerPoint.

O desenho dos documentos BIRT são mantidos num esquema XML que determinam como aceder a diferentes tipos de fontes de dados, incluindo JDO, JFire Scripting Objects, POJO, SQL, Web Services e XML.

3.3.2 Jasper Reports

O Jasper Reports [43] é uma ferramenta *open source* e *standalone*, que permite efetuar o desenho de documentos, definindo a localização dos campos do ficheiro de input, sendo criado um ficheiro Jasper. Para efetuar o design pode ser utilizado o software TIBCO Jaspersoft Studio [15] (Figura 45), doravante designada JasperUI, sendo este uma UI desenvolvida para efetuar o design de Jasper Reports, criando o respetivo ficheiro Jasper.

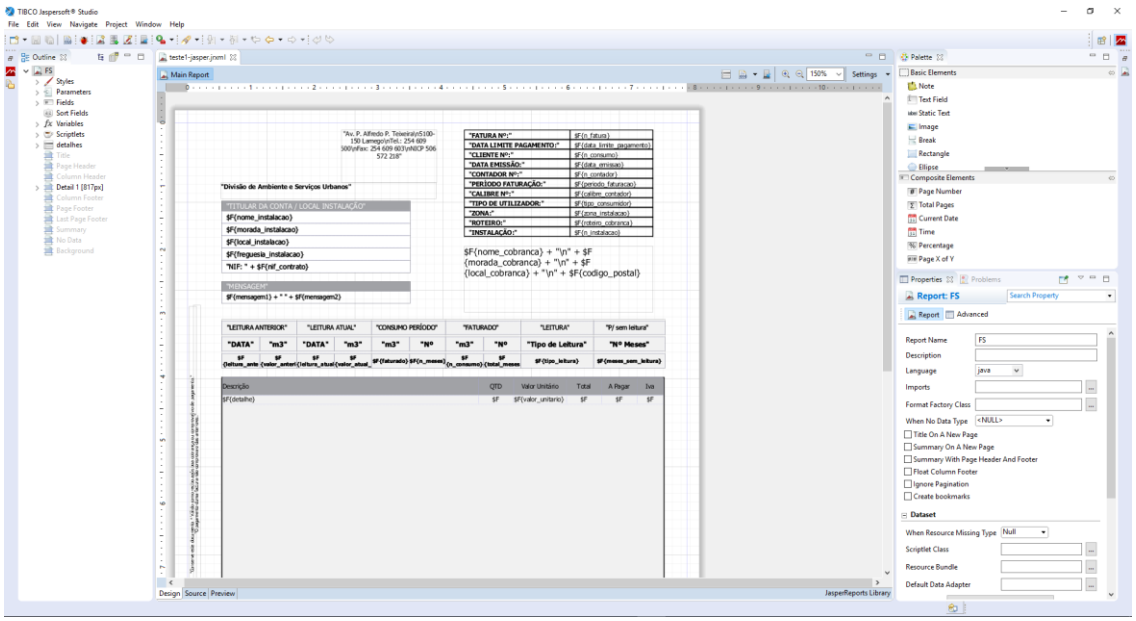


Figura 45 – UI do JasperUI para efetuar o desenho de documentos

Este software permite projetar e executar modelos de relatórios, escrever expressões complexas, contendo componentes visuais de layout, dispondo de mais de 50 tipos de gráficos, mapas, tabelas, visualizações personalizadas, entre outros. O JasperUI também permite que seja efetuada a criação dos documentos, podendo gravar em vários formatos incluindo screen, printer, PDF, HTML, Microsoft Excel, RTF, ODT, CSV ou XML.

O JasperUI consegue aceder a diferentes tipos de fontes de dados, incluindo CSV, Hibernate, Domínio Jaspersoft, JavaBeans, JDBC, JSON, NoSQL, XML ou a sua própria fonte de dados personalizada.

Também existe disponível como um plug-in do Eclipse ou uma aplicação independente, de forma a ser utilizado noutro IDE.

3.3.3 Análise das tecnologias

De acordo com a pesquisa e de forma a analisar as várias soluções existentes para efetuar o design dos documentos descritas nas secções anteriores (3.3.1 e 3.3.2), foram comparadas as duas tecnologias.

Ao efetuar a comparação entre o BIRT e o Jasper Repors foi possível verificar que o Jasper Reports é *open source* e *standalone* (enquanto o BIRT é apenas *open source*), conseguindo aceder a mais tipos de fontes de dados (enquanto o Jasper Reports acede a nove, o BIRT acede apenas a seis), assim como exportar para mais formatos, sendo que o BIRT pode demorar entre 4 a 8 vezes mais do que o JasperReports a executar [1].

Assim, a tecnologia mais adequada para esta situação é o Jasper Reports, principalmente porque o BIRT pode ser entre 4 a 8 vezes mais lento, sendo esta uma dimensão muito importante neste projeto.

3.4 Problema 4: criação de documentos PDF

De acordo com a pesquisa e análise do problema 3 na secção 3.3, foi concluído que o melhor design de documentos PDF é o Jasper Reports, sendo nesta secção analisada a criação dos documentos utilizando-o.

Ao efetuar a análise ao JasperUI, foi possível perceber que este permite a exportação para vários formatos, assim como efetua a criação de um único ficheiro com todos os documentos.

De forma a perceber como é efetuada a criação dos documentos, foi possível verificar que quando é selecionada a guia de visualizar (Figura 46), é efetuada inicialmente a compilação do ficheiro de design, validando todos os elementos e expressões utilizadas, sendo de seguida carregada a informação da fonte de dados para o design, sendo depois apresentado o documento com os respetivos dados.

Após o documento ser apresentado no ecrã, é possível efetuar a exportação do ficheiro para o formato pretendido, conforme apresentado na Figura 46.

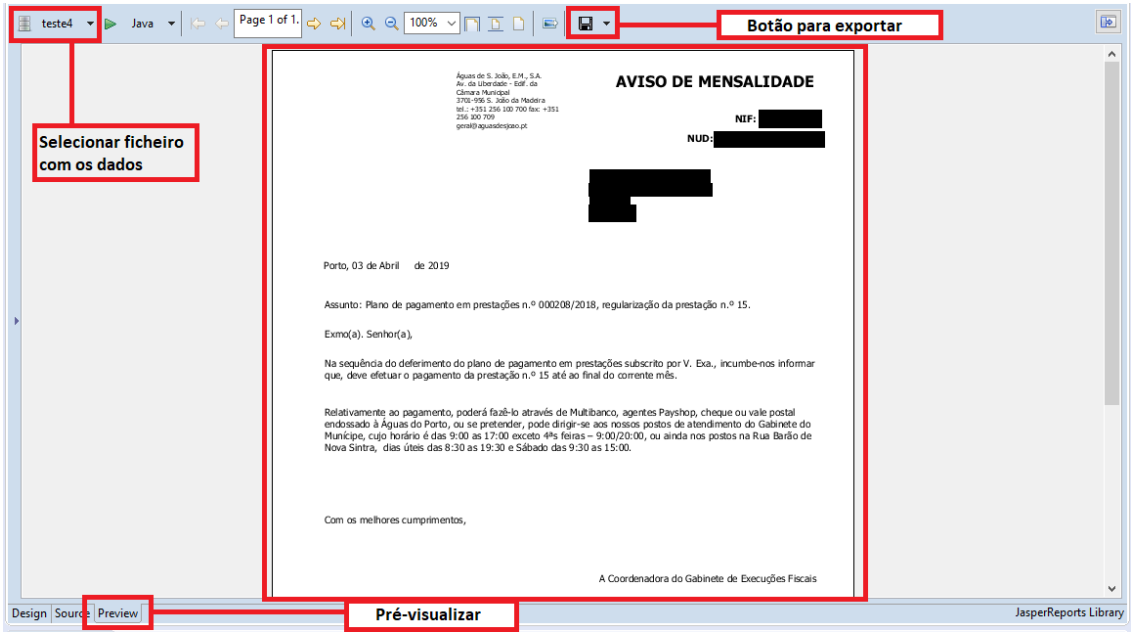


Figura 46 – Ecrã de exportação do JasperUI

3.5 Problema 5: leitura de ficheiros e criação de documentos PDF

A análise das soluções existentes para a leitura de ficheiros de dados normalizados são as descritas nas secções 3.1.1 e 3.1.2, sendo que para efetuar a criação dos documentos PDF utilizando o novo software desenvolvido será utilizado o Jasper Reports descrito na secção 3.3.2, uma vez que esta foi a melhor solução encontrada para o problema 3, sendo utilizada a libraria disponibilizada, doravante denominada de JasperReportsLibrary.

Quanto à leitura de ficheiros, e de acordo com a pesquisa e de forma a analisar as várias soluções existentes, foi efetuada uma aplicação com o intuito de testar as várias soluções, de forma a analisar qual a mais eficaz e eficiente.

Esta aplicação é composta pela combinação dos dois métodos de leitura, e o de criação de documentos PDF, resultando em dois serviços. É efetuada a leitura de um ficheiro com 60MB de tamanho (o ficheiro de dados normalizados gerado pelo Ficheiro 6, utilizado na secção 2.3.2), sendo executado cada serviço seis vezes, de forma a analisar e comparar os resultados obtidos. Os dois serviços efetuam a leitura do ficheiro de dados normalizados, efetuando de seguida a criação dos respetivos documentos PDF (utilizando a JasperReportsLibrary).

O primeiro serviço é denominado de ReadXMLusingSoftwareService (Figura 47), utilizando o DOM para efetuar a leitura, sendo que este recebe uma estrutura em árvore com os nós do ficheiro, carregando toda a informação na memória, sendo aconselhável e possível para processamento de ficheiros pequenos. Este é o método utilizado pelo software existente.

```
public static void read(String fileName) throws FileNotFoundException, IOException, ParserConfigurationException, SAXException, COSVisitorExceptior
    String recordpath = "/documento";
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
    Document document = documentBuilder.parse(fileName);
    document.getDocumentElement().normalize();
    NodeList nodeList = document.getElementsByTagName("documento");
    String ficheiros = "C:\\Users\\NB24245\\OneDrive - Instituto Superior de Engenharia do Porto\\TESE\\Cliente1\\ReadXMLusingSoftware\\";
    for (int index = 0; index < nodeList.getLength(); index++) {
        Node node = nodeList.item(index);
        StringBuilder xml = getStringXmlString(node);
        String numero = String.valueOf(index + 1);
        try {
            JRXmlDataSource xml1 = new JRXmlDataSource(XMLUtils.getDocumentFromString(xml.toString()), recordpath);
            String jasper = "C:\\Users\\NB24245\\OneDrive - Instituto Superior de Engenharia do Porto\\TESE\\teste7-jasper.jasper";
            JasperPrint jp1 = JasperFillManager.fillReport(jasper, new HashMap<>(), xml1);
            JRExporter exporter1 = new net.sf.jasperreports.engine.export.JRPdfExporter();
            exporter1.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, ficheiros + File.separator + numero + ".pdf");
            exporter1.setParameter(JRExporterParameter.JASPER_PRINT, jp1);
            exporter1.exportReport();
        } catch (JRException ex) {
            System.out.println("Erro no documento - " + numero);
            java.util.logging.Logger.getLogger(ReadXMLusingSaxService.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Figura 47 – Serviço ReadXMLusingSoftwareService

O outro serviço é denominado de ReadXMLusingSaxService (Figura 48), utilizando o SAXParser para efetuar a leitura, usando manipuladores de eventos, não carregando toda a informação na memória. Como é lido passo a passo, é aconselhável e possível para processamento de ficheiros grandes.

```
public static void read(String fileName) throws FileNotFoundException, IOException, ParserConfigurationException, SAXException {
    String recordpath = "/documento";
    SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
    SAXParser saxParser = saxParserFactory.newSAXParser();
    DefaultHandler defaultHandler;
    String ficheiros = "C:\\Users\\NB24245\\OneDrive - Instituto Superior de Engenharia do Porto\\TESE\\Cliente1\\ReadXMLusingSax\\";
    File pasta = new File(ficheiros);
    if (!pasta.exists()) {
        pasta.mkdir();
    }
    defaultHandler = new DefaultHandler() {
        StringBuffer xml;
        boolean convert = false;
        int numero = 1;

        @Override
        public void startElement(String uri, String localName, String qName, Attributes attr) throws SAXException {
            if (!qName.equalsIgnoreCase("documentos")) {
                if (qName.equalsIgnoreCase("documento")) {
                    convert = true;
                    xml = new StringBuffer();
                }
                xml.append("<").append(qName);
                if(attr != null && attr.getLength() > 0){
                    for(int i=0; i<attr.getLength(); i++){
                    xml.append(" ").append(attr.getQName(i)).append("=\"").append(attr.getValue(i)).append("\"");
                    }
                }
                xml.append(">");
            }
        }

        @Override
        public void characters(char chars[], int start, int len) {
            if (convert) {
                String texto = decodeTexto(new String(chars, start, len));
                xml.append(texto);
            }
        }

        @Override
        public void endElement(String uri, String localName, String qName) throws SAXException {
            if (!qName.equalsIgnoreCase("documentos")) {
                xml.append("</").append(qName).append(">");
                if (qName.equalsIgnoreCase("documento")) {
                    try {
                        JRXmlDataSource xml1 = new JRXmlDataSource(XMLUtils.getDocumentFromString(xml.toString()), recordpath);
                        String jasper = "C:\\Users\\NB24245\\OneDrive - Instituto Superior de Engenharia do Porto\\TESE\\teste7-jasper.jasper";
                        JasperPrint jp1 = JasperFillManager.fillReport(jasper, new HashMap<>(), xml1);
                        JRExporter exporter1 = new net.sf.jasperreports.engine.export.JRPdfExporter();
                        exporter1.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, ficheiros + File.separator + numero + ".pdf");
                        exporter1.setParameter(JRExporterParameter.JASPER_PRINT, jp1);
                        exporter1.exportReport();
                        numero++;
                    } catch (JRException ex) {
                        System.out.println("Erro no documento - " + numero);
                        numero++;
                        java.util.logging.Logger.getLogger(ReadXMLusingSaxService.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }
        }
    };
    saxParser.parse(fileName, defaultHandler);
}
```

Figura 48 – Serviço ReadXMLusingSaxService

Após executar cada um dos serviços seis vezes, foi possível obter os tempos de execução apresentados na Tabela 11. Através dos valores obtidos, é possível observar que os serviços são equivalentes, uma vez que ambos foram três das seis vezes o mais rápido, embora o ReadXMLusingSoftwareService inicialmente tenha demorado cerca de mais um minuto que o outro serviço, o que sugere a existência de uso de memória cache.

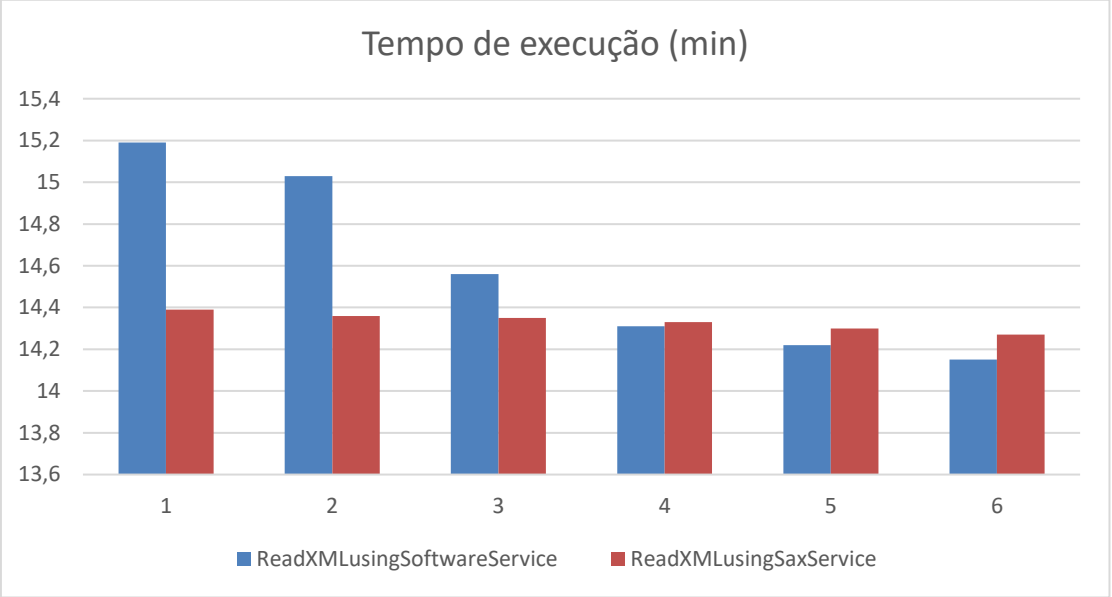


Tabela 11 – Gráfico do tempo de execução dos serviços

Após a execução referida anteriormente, foi também possível calcular a memória utilizada apresentada na Tabela 12. Com os valores obtidos, é possível observar que o serviço que utilizou menos memória foi o ReadXMLusingSaxService, como seria de esperar.

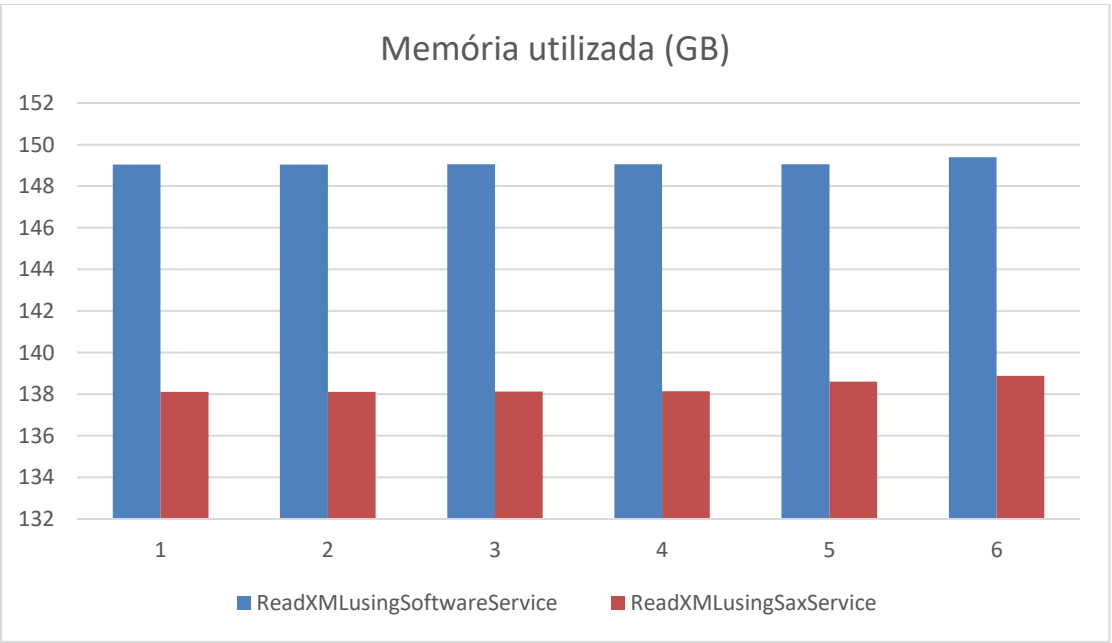


Tabela 12 – Gráfico da memória utilizada nos serviços

Após esta análise, foi possível concluir que o melhor método de leitura de ficheiros XML é o ReadXMLusingSaxService, uma vez que nos seis testes realizados, sempre foi o que utilizou, consistentemente menos memória. Quanto ao tempo de execução e mesmo que se ignore o

muito maior tempo de execução nas primeiras 3 corridas por parte do ReadXMLusingSoftwareService, nas outras três a vantagem deste é mínima em relação a ReadXMLusingSaxService, o que não compensa a desvantagem na dimensão de memória consumida.

3.6 Síntese

Neste capítulo foram identificados cinco problemas no software ProcessaDados. O objetivo no problema 1 é resolver o problema da configuração dos campos e especificações, podendo verificar que não existe nenhuma solução para o pretendido, sedo necessário a criação de um software que efetue o pretendido.

O objetivo no problema 2 é encontrar a melhor forma de efetuar a leitura dos ficheiros fonte de dados, e da escrita de ficheiros de dados normalizado.

O objetivo no problema 3 é encontrar a melhor forma de efetuar o design dos documentos PDF, podendo verificar que a melhor solução encontrada foi o Jasper Reports, utilizando o JasperUI.

Uma vez que no problema 3 já foi definida uma solução a ser utilizada, o problema 4 tem como objetivo a análise de como efetuar a criação dos documentos PDF utilizando o JasperUI.

Por fim, o objetivo no problema 5 é encontrar a melhor forma de efetuar a leitura de ficheiros de dados normalizados, efetuando a criação dos documentos PDF utilizando a JasperReportsLibrary. É importante analisar o comportamento destes 2 requisitos juntos, uma vez que a criação do PDF é efetuada enquanto é efetuada a leitura do documento.

4 Design

"O desenho arquitetural é das primeiras atividades que compõe a área do conhecimento em desenho de software. Devido à complexidade dos projetos, a parte comum a qualquer engenharia para se construir um artefacto complexo, um website acessível neste caso, é construir o projeto de acordo com um plano. Por outras palavras, desenho arquitetural é projetar um sistema antes deste ser construído para evitar falhas e se construir um plano/caminho a seguir" (Staveley, 2011).

Esta secção tem como principal objetivo apresentar as alternativas arquiteturais, descrevendo a geração, enriquecimento e seleção de ideias, assim como apresentar o design arquitetural deste projeto.

4.1 Alternativas concetuais

Esta secção tem como objetivo a apresentação das alternativas arquiteturais, utilizando o modelo NCD (New Concept Development), de forma a perceber quais são as necessidades dos clientes, as ideias geradas, e destas, quais as selecionadas. De forma a permitir o uso de critérios qualitativos e quantitativos no processo da avaliação das ideias e desenvolvimento, facilitando a compreensão e avaliação do projeto, é utilizado o método AHP.

4.1.1 Modelo New Concept Development

Segundo Peter Koen, o modelo NCD (New Concept Development) é composto por três partes essenciais (Figura 49):

- O motor representa a liderança, cultura, e estratégia empresarial da organização que orienta os cinco elementos chave do modelo, sendo controlado pela organização.

- Os cinco elementos de atividade:

    - Identificação da oportunidade;

    - Análise da oportunidade;

    - Geração de ideias;

    - Seleção de ideias;

    - Desenvolvimento do conceito.

- Os fatores de influência do ambiente externo, sendo algo que a organização dificilmente consegue controlar, e que afeta todo o processo de inovação até à comercialização. Estes fatores são as capacidades organizacionais, estratégia de negócio e mundo exterior (canais de distribuição, clientes e concorrentes).

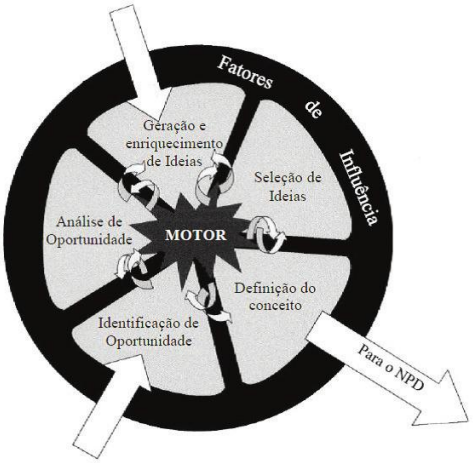


Figura 49 – Modelo NCD (New Concept Development)

A figura acima (Figura 49) ilustra o modelo NCD, podendo-se observar que possui um formato circular, com o intuito de sugerir que é expectável que as ideias e conceitos fluam, circulem e interajam entre os cinco elementos. Conforme se observa, existem duas setas que apontam para o interior do modelo, representando os pontos de partida e indicando que os projetos e soluções devem começar pela identificação de oportunidades ou geração de ideias e enriquecimento das mesmas. A seta de saída para o exterior, representa que os conceitos resultantes do modelo, entram no novo processo de desenvolvimento de produto (NPD).

4.1.2 Geração e enriquecimento de ideias

A geração da ideia está relacionada com os processos evolucionais de idealização e de enriquecimento de ideias, uma vez que estas são construídas, modificadas e poderão ser abandonadas.

Quando surgiu a decisão de resolver o problema do software existente, alterando, criando ou eliminando os processos existentes, surgiram algumas ideias que podiam ser alternativas, nomeadamente:

- Ideia 1: Fazer alterações pontuais para cumprimento dos requisitos de eficácia e eficiência;
- Ideia 2: Fazer alterações pontuais para cumprimento dos requisitos de eficácia, eficiência, adaptabilidade e manutenibilidade;
- Ideia 3: Permitir a configuração dos campos e especificações de leitura do ficheiro, dando resposta aos requisitos de eficácia, eficiência, adaptabilidade e manutenibilidade.

A ideia 1 efetua apenas pequenas melhorias, não cobrindo todos os problemas encontrados no mesmo, impedindo a configuração dos campos e especificações de leitura, e dificultando assim a adaptabilidade e manutenibilidade do software.

A ideia 2 efetua também pequenas melhorias, impedindo também a configuração dos campos e especificações de leitura, mas já dando resposta à adaptabilidade e manutenibilidade do software.

A ideia 3, já melhora significativamente a eficácia e eficiência do software, permitindo a configuração dos campos e especificações de leitura, e facilitando a adaptabilidade e manutenibilidade do software.

4.1.3 Seleção de ideias

A seleção de ideias é crítica para o sucesso da mesma. O método de decisão multicritério, que será usado para determinar qual a solução gerada que deve ser selecionada é o método de análise hierárquica (AHP - Analytic Hierarchy Process). Este método permite o uso de critérios qualitativos e quantitativos no processo de avaliação, tendo como ideia principal dividir o problema de decisão em níveis hierárquicos, facilitando a sua compreensão e avaliação.

Conforme referido anteriormente (secção 2.1.3), a proposta de valor para este projeto é efetuar um novo software. De acordo com as ideias geradas (secção 4.1.2), foi construída a árvore hierárquica de decisão (Figura 50).

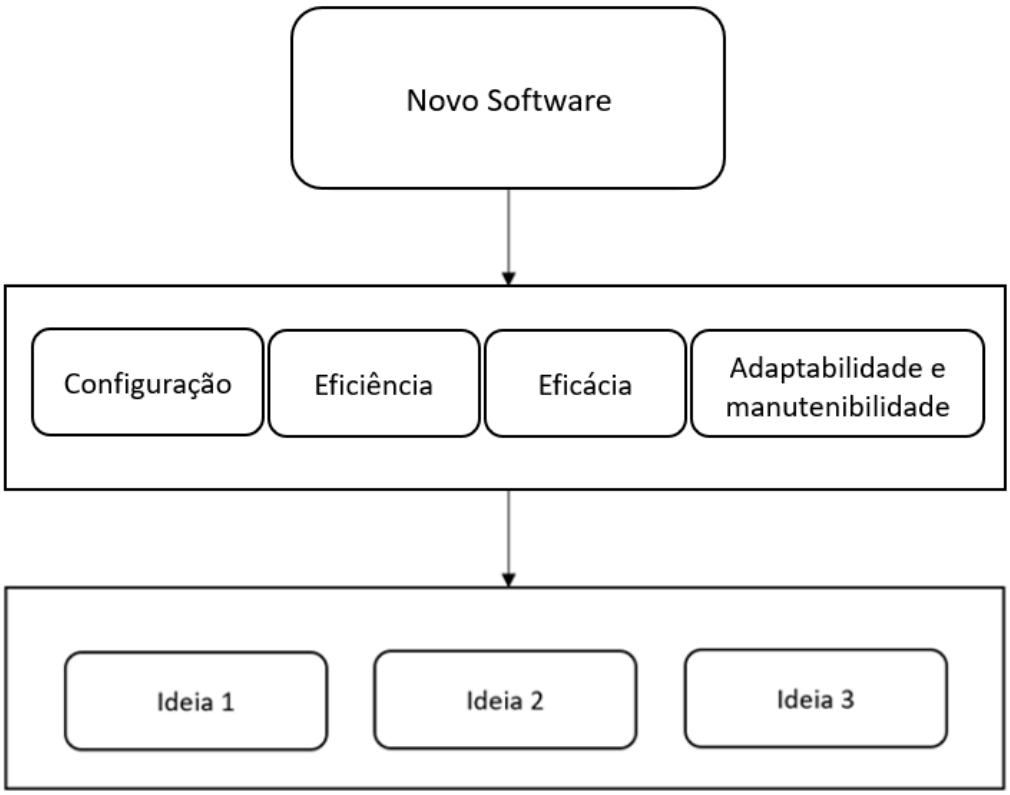


Figura 50 - Árvore hierárquica de decisão

Na Figura 50 é possível observar que a primeira camada representa o objetivo principal deste projeto (novo software). Para isso, foram envolvidos alguns critérios, de forma a avaliar qual a melhor ideia para chegar à solução, sendo estes os seguintes:

- Configuração, permitindo a configuração dos campos e especificações de leitura do ficheiro;
- Eficiência, de que forma é que a solução cumpre os seus objetivos;
- Eficácia, se a solução cumpre os seus objetivos;
- Adaptabilidade e manutenibilidade, se a solução é adaptável às necessidades do utilizador, e se é fácil a sua modificação e manutenção.

Com estes critérios, é possível então avaliar qual a melhor ideia para atingir o principal objetivo, começando por comparar as alternativas e critérios. Foi utilizada a escala fundamental

definida por Satty (Tabela 13), sendo possível estabelecer prioridades entre os elementos para cada nível da hierarquia, descrevendo a matriz normalizada com os pesos finais, para cada critério (Tabela 14).

Nível de importância	**Definição**	**Explicação**
1	Igual importância	As duas atividades contribuem igualmente para o objetivo
3	Fraca importância	A experiência e o julgamento favorecem levemente uma atividade em relação à outra
5	Forte importância	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra
7	Muito forte importância	Uma atividade é muito fortemente favorecida em relação a outra
9	Importância absoluta	A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza
2, 4, 6, 8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições

Tabela 13 – Escala fundamental de Satty

Critérios de Avaliação	**Configuração**	**Eficácia**	**Eficiência**	**Adaptabilidade e manutenibilidade**
Configuração	1	2	5	3
Eficácia	1/2	1	4	2
Eficiência	1/4	1/4	1	1/3
Adaptabilidade e manutenibilidade	1/3	1/2	3	1
Total	25/12	15/4	13	19/3

Tabela 14 – Tabela de avaliação AHP

A configuração dos campos e especificações de leitura do ficheiro e a eficácia são os dois critérios com mais importância na seleção da ideia a implementar, sendo de média importância a adaptabilidade e manutenibilidade, e menos importante a eficiência.

Critérios de Avaliação	**Configuração**	**Eficácia**	**Eficiência**	**Adaptabilidade e manutenibilidade**
Configuração	0.4801	0.5333	0.3846	0.4737
Eficácia	0.24	0.2667	0.3077	0.3158
Eficiência	0.12	0.0667	0.0769	0.0526
Adaptabilidade e manutenibilidade	0.16	0.1333	0.2308	0.1579
Total	1	1	1	1

Tabela 15 - Matriz normalizada do método de avaliação HP

Critérios de Avaliação	**Pesos**
Configuração	0.4679
Eficácia	0.2826
Eficiência	0.0791
Adaptabilidade e manutenibilidade	0.1441

Tabela 16 - Pesos associados aos critérios de avaliação hierárquica

Após analisar as tabelas acima (Tabela 14, Tabela 15 e Tabela 16) e das várias ideias geradas, a que permite efetuar a configuração dos campos e as especificações de leitura do ficheiro, a mais eficaz, e com maior adaptabilidade e manutenibilidade, é a ideia 3.

De acordo com a avaliação hierárquica, para este projeto a ideia 3 será a selecionada para responder às necessidades do negócio, uma vez que é a ideia que vai de encontro aos requisitos de adaptabilidade e manutenibilidade, permitindo a configuração dos campos e especificações de leitura, permitindo assim que a Empresa efetue o tratamento de outros tipos

de ficheiro para além do único possível atualmente, dando também resposta aos requisitos de eficácia e eficiência, sendo esta a que transparece maior valor.

4.2 Design arquitetural

Esta secção tem como objetivo a descrição da arquitetura proposta, utilizando o modelo 4+1 (Staveley, 2011), e em particular as vistas (i) lógica, (ii) de processo, (iii) de implementação e (iv) de implantação. Além disso, adotar-se-á uma representação em dois níveis de abstração: de sistema (secção 4.2.1) e de componente (secção 4.2.2 e 4.2.3).

4.2.1 Design arquitetural de sistema

Nesta secção é apresentado o design arquitetural numa granularidade de sistema, apresentando a vista (i) lógica, (ii) de processo, (iii) de implementação e (iv) de implantação.

4.2.1.1 Vista lógica

O software a desenvolver irá fazer uso de 3 principais processos, constituídos por várias tarefas. De forma a compreender melhor a estrutura/arquitetura do sistema desenvolvido foi criado um diagrama de componentes (Figura 51) e um diagrama BPMN (Figura 52) que retratam essa perspetiva.

É possível observar na Figura 51 que o sistema é composto por três componentes:

- ConfiguraDados, que contém uma UI de forma a interagir com o colaborador da Empresa. Este componente é o responsável pela configuração dos campos e especificações de leitura, sendo utilizado um ficheiro de especificações disponibilizado pela mesma empresa terceira que criou o ficheiro fonte de dados, onde é possível identificar os campos e especificações do ficheiro. Este componente também efetua a criação do ficheiro de configurações e especificações, contendo toda a informação inserida pelo colaborador da Empresa;

- JasperUI, que também disponibiliza uma UI. Este componente permite ao responsável pelo desenho criar o documento PDF a gerar, efetuando a criação do ficheiro de design PDF;

- GeraPDF, que disponibiliza um REST, sendo o responsável pela leitura do ficheiro fonte de dados recebido pelas várias Câmaras Municipais, gerando os vários documentos PDF. Este componente utiliza o ficheiro de configurações e especificações criado pelo

ConfiguraDados aplicando as configurações no ficheiro fonte de dados, e o ficheiro de design PDF criado pelo JasperUI para a criação dos documentos PDF.

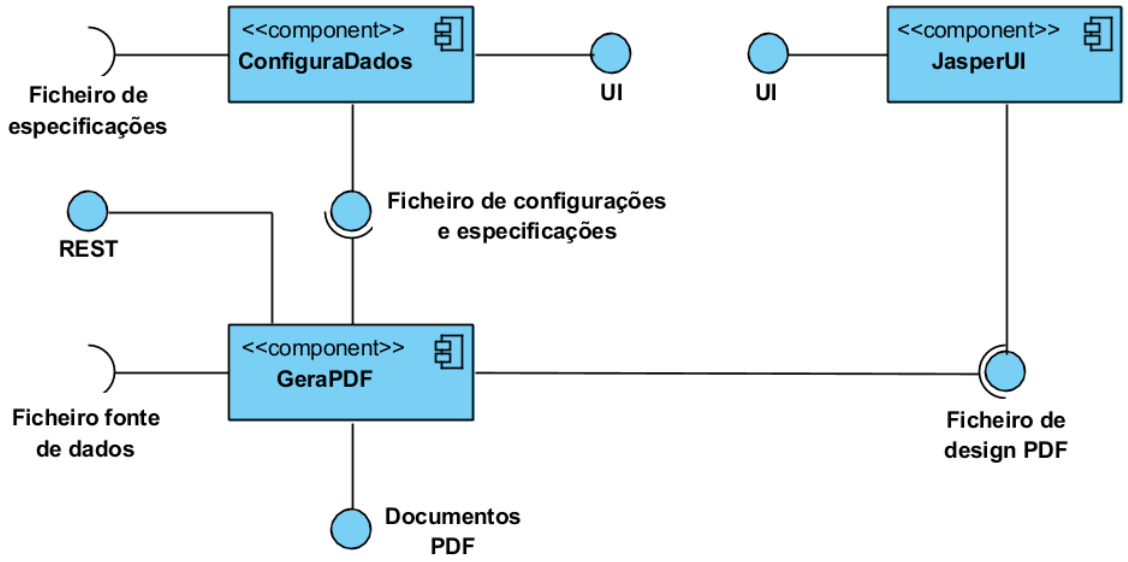


Figura 51 – Vista lógica de componentes do sistema (diagrama de componentes em UML)

O BPMN especifica o processo de negócio num diagrama, facilitando a sua leitura tanto para utilizadores técnicos como para os utilizadores de negócios, uma vez que é um diagrama intuitivo, conforme apresentado na Figura 52.

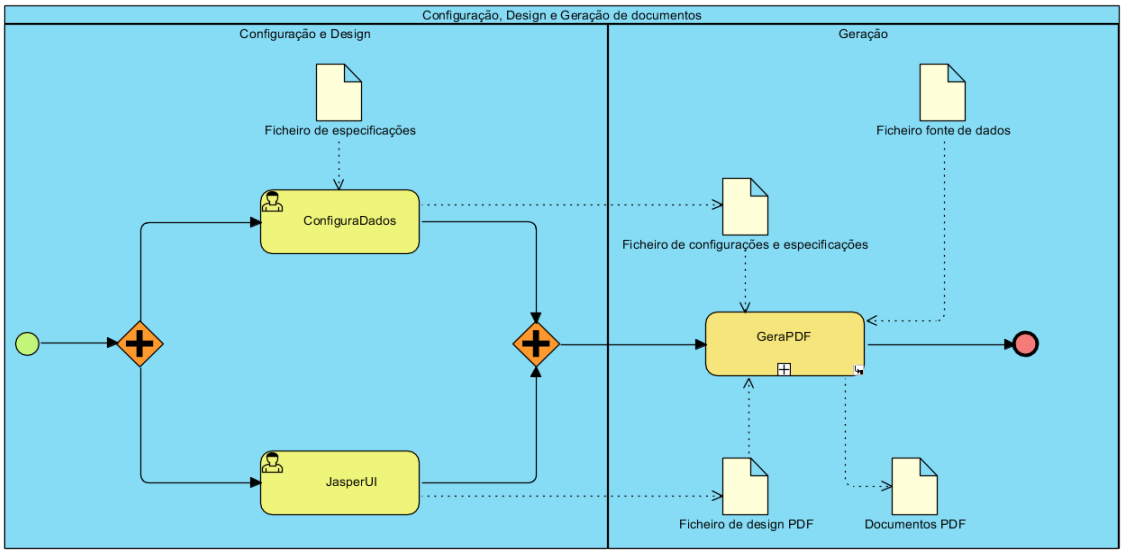


Figura 52 – Vista lógica do sistema (diagrama BPMN)

4.2.1.2 Vista de processo

A vista de processo descreve a dinâmica das partes que constituem o sistema, na realização das tarefas/responsabilidades que lhe estão atribuídas.

O diagrama de sequência apresentado na Figura 53 representa as mensagens, pedidos e interações entre os diversos componentes do sistema. O funcionamento explica-se da seguinte forma:

- (1) o colaborador da Empresa utiliza o ConfiguraDados, inserindo as especificações, criando e guardando a informação no ficheiro de configurações e especificações;
- (2) o colaborador da Empresa cria design dos ficheiros PDF a imprimir e envelopar, gerando o ficheiro de design PDF;
- (3) o colaborador da Empresa cria os ficheiros PDF, usando o ficheiro de dados como fonte a interpretar segundo o ficheiro de configurações e especificações, e o ficheiro de design PDF. A conjugação destes com a JasperReportsLibrary gera os documentos PDF a imprimir e envelopar.

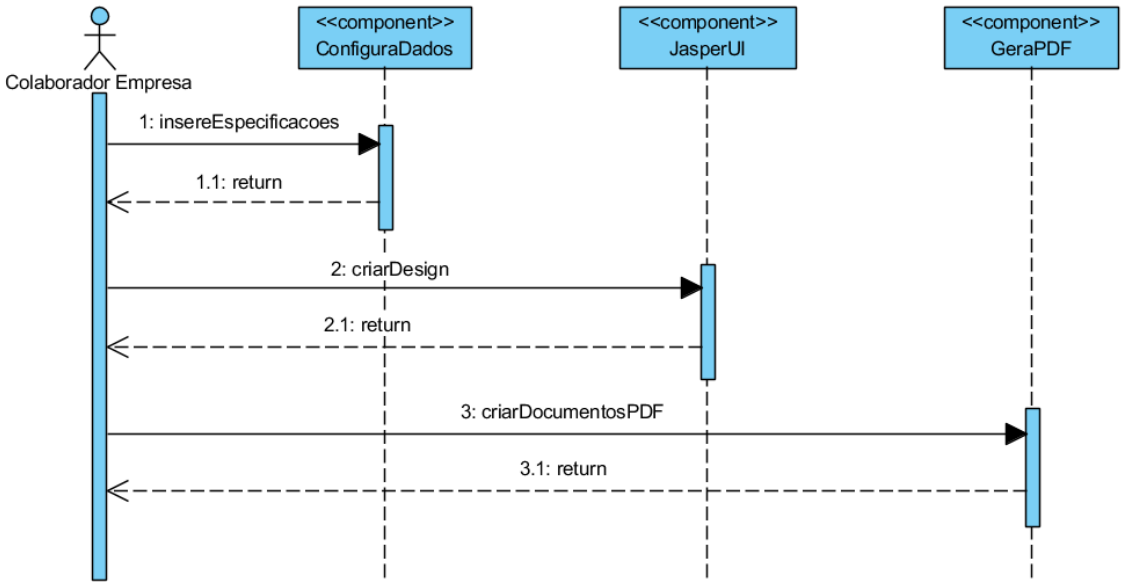


Figura 53 – Vista de 3 processos (diagrama de sequência em UML)

4.2.1.3 Vista de implementação

A vista de implementação tem como objetivo mostrar a estrutura dos projetos de software. Conforme é possível observar no empacotamento apresentado na Figura 54, onde são apresentados os três pacotes que constituem o sistema, sendo que o GeraPDF é dependente do ConfiguraDados e do JasperUI.

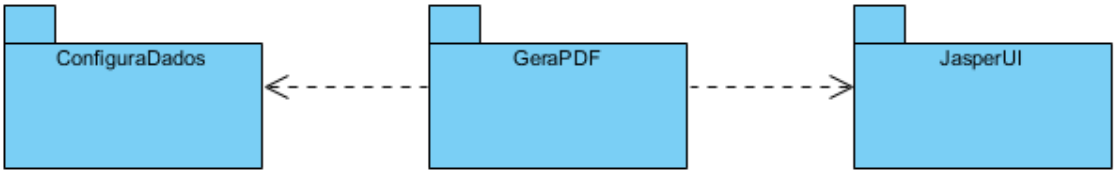


Figura 54 – Vista de implementação do sistema (Diagrama de componentes em UML)

4.2.1.4 Vista de implantação

A vista de implantação tem como objetivo mostrar a estrutura de hardware na qual o software é executado, contendo a parte física do sistema e a conexão entre estes.

Como é possível observar na Figura 55, no diagrama de implantação do sistema, o componente JasperUI e o ConfiguraDados estão instalados numa máquina cliente, enquanto o componente Apache e Tomcat estão a correr num servidor havendo comunicação entre estes através de AJP, sendo que o GeraPDF está no componente Tomcat, havendo comunicação entre o cliente e o servidor através de um pedido HTTP.

O Tomcat é um servidor aplicacional, sendo neste caso integrado com o Apache.

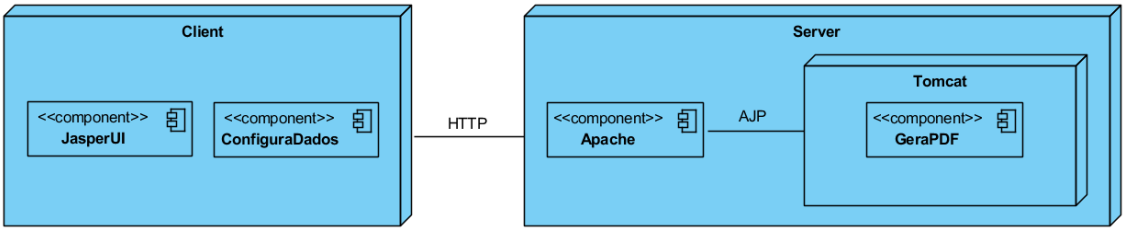


Figura 55 – Vista de implantação do sistema (diagrama de nós em UML)

4.2.2 Design arquitetural de componente GeraPDF

Nesta secção é apresentado o design arquitetural do componente GeraPDF, apresentando as vistas (i) lógica, (ii) de processo e (iii) de implementação.

A vista de implantação nesta granularidade é a mesma que a apresentada para a granularidade de sistema.

4.2.2.1 Vista lógica

De forma a compreender melhor a estrutura/arquitetura do GeraPDF foi criado um diagrama de componentes (Figura 56) e um diagrama BPMN (Figura 57) que retratam essa perspetiva.

É possível observar na Figura 56 que as responsabilidades do componente GeraPDF também são divididas em dois subcomponentes:

- Transformacao, que disponibiliza um REST, e é responsável por aplicar a configuração e especificações do ficheiro de configurações e especificações ao ficheiro fonte de dados recebido pelas Câmaras Municipais, sendo criado um ficheiro de dados normalizado, sendo posteriormente executado o Geracao;

- Geracao, que é responsável por criar os vários documentos PDF, utilizando o ficheiro de dados normalizado e o ficheiro de design PDF.

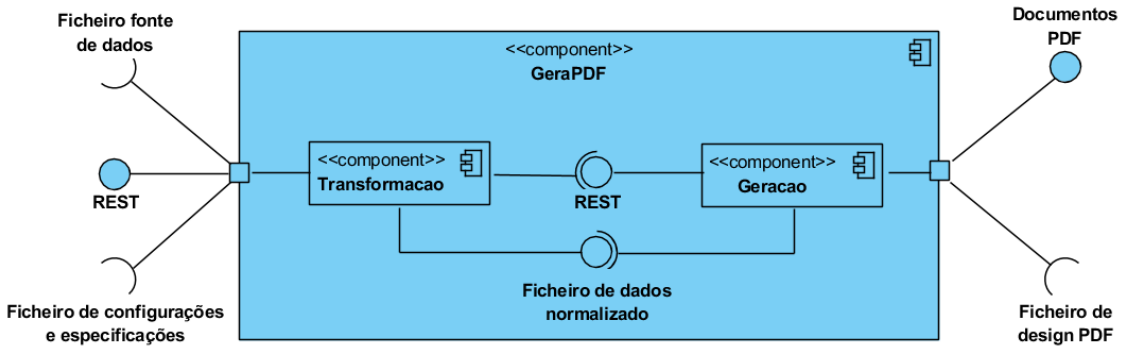


Figura 56 – Vista lógica do GeraPDF (diagrama de componentes em UML)

Mimetizando as atividades/responsabilidade descritas no diagrama anterior, e de forma a compreender melhor a estrutura/arquitetura o GeraPDF foi criado um diagrama em BPMN que especifica o processo de negócio (Figura 57):

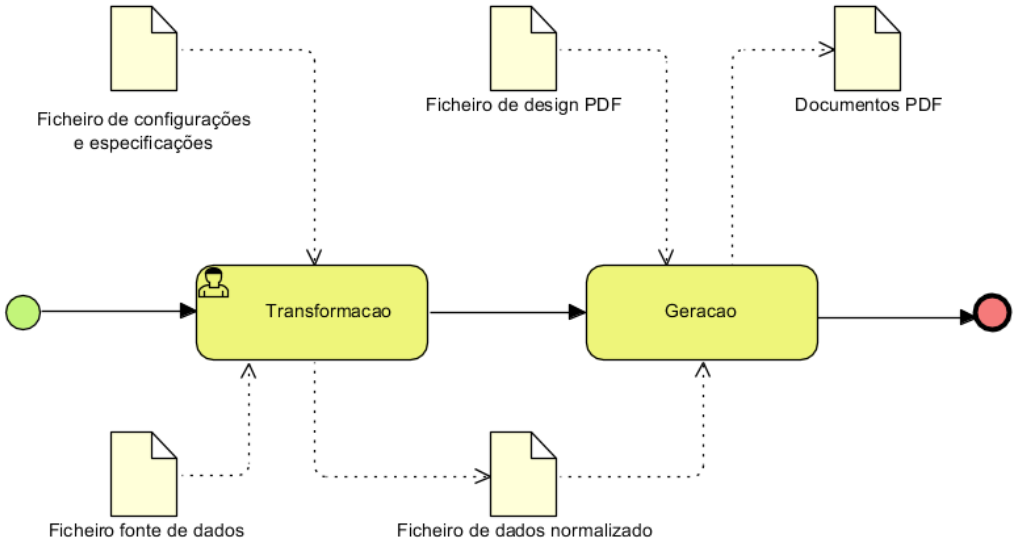


Figura 57 – Vista lógica do GeraPDF (diagrama BMPN)

O diagrama de componentes apresentado na Figura 58 representa a estrutura interna do componente Transformacao.

É possível observar na Figura 58

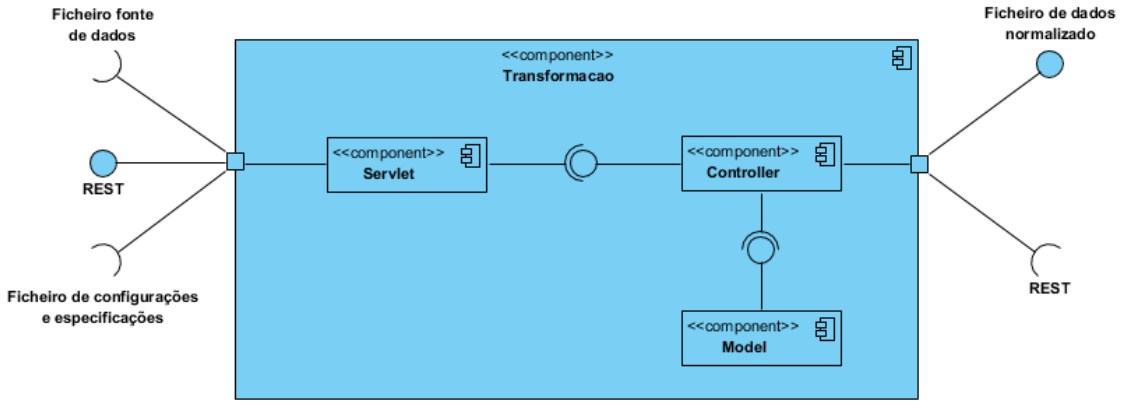


Figura 58 que o Transformacao tem três componentes, sendo estes o Servlet, Controller e Model. Também é possível observar que o Transformacao recebe um pedido, sendo o pedido enviado ao componente Servlet, comunicando com o componente Controller, que este por sua vez irá comunicar com o componente Model.

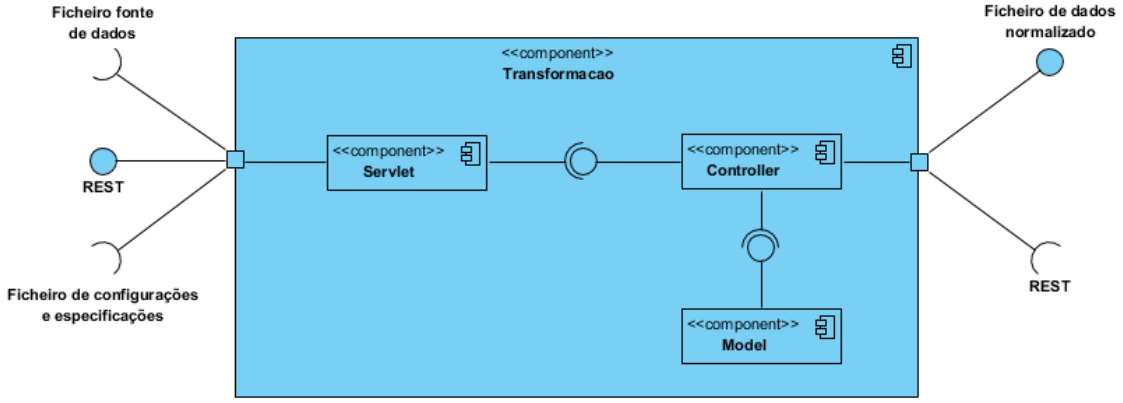


Figura 58 – Vista lógica do Transformacao (diagrama de componentes em UML)

Por fim, o diagrama de componentes apresentado na Figura 59 representa a estrutura interna do componente Geracao.

É possível observar na Figura 59 que o Geracao tem também três componentes, sendo estes o Servlet, o Controller e o JasperReportsLibrary. Também é possível observar que o Geracao recebe um pedido, sendo o pedido enviado ao componente Servlet, que irá comunicar com o componente Controller, e este por sua vez irá comunicar com o JasperReportsLibrary para efetuar a criação dos documentos PDF.

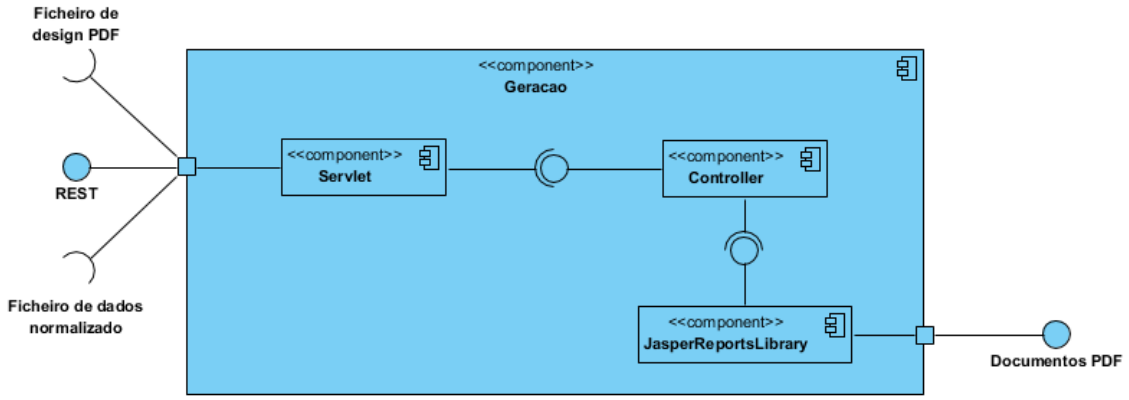


Figura 59 – Vista lógica do Geracao (diagrama de componentes em UML)

4.2.2.2 Vista de processo

A vista de processo descreve a dinâmica das partes que constituem a aplicação, na realização das tarefas/responsabilidades que lhe estão atribuídas.

O diagrama apresentado na Figura 60 representa as mensagens, pedidos e interações entre os diversos componentes. O funcionamento explica-se da seguinte forma:

- O colaborador da Empresa envia o pedido ao Transformacao, sendo aplicadas as configurações dos campos e especificações de leitura do ficheiro de configurações e especificações criado pelo ConfiguraDados;
- O Transformacao envia o pedido ao Geracao para a criação dos documentos PDF, utilizando o ficheiro de dados normalizado criado pelo Transformacao, e o ficheiro de design PDF criado pelo JasperUI;
- O Geracao retorna a informação ao Transformacao, que por sua vez apresenta-a ao colaborador da Empresa.

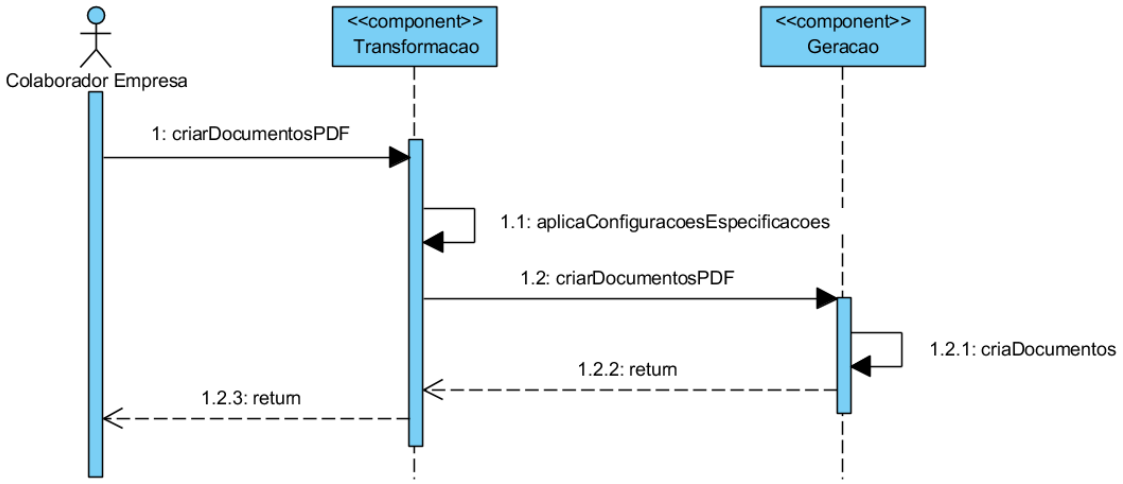


Figura 60 – Vista de processo do GeraPDF (diagrama de sequência em UML)

4.2.2.3 Vista de implementação

A vista de implementação tem como objetivo mostrar as partes que constituem o software sob um ponto de vista arquiteturalmente significativo. Foi elaborado um diagrama de classes agregadas em pacotes, representativo da estrutura e relações das várias classes do Transformacao (Figura 61).

Na Figura 61 são apresentados os três pacotes que constituem a aplicação:

- Servlet, que contém a classe TransformacaoServlet, sendo esta responsável por receber os pedidos efetuados pelo colaborador da Empresa;
- Controller, que contém as classes ExecutaGeracao e TransformacaoXML, sendo que a primeira é a responsável por executar a aplicação Geracao, e a segunda é responsável por efetuar a criação do ficheiro de dados normalizado utilizando o ficheiro de configurações e especificações e o ficheiro fonte de dados;
- Model, que contém as classes Transformacao, CampoComposto, CampoSimples e Id, que representam os modelos utilizados, sendo criado um objeto Transformacao pelo TransformacaoXML no pacote Controller, contendo este os outros objetos necessários.

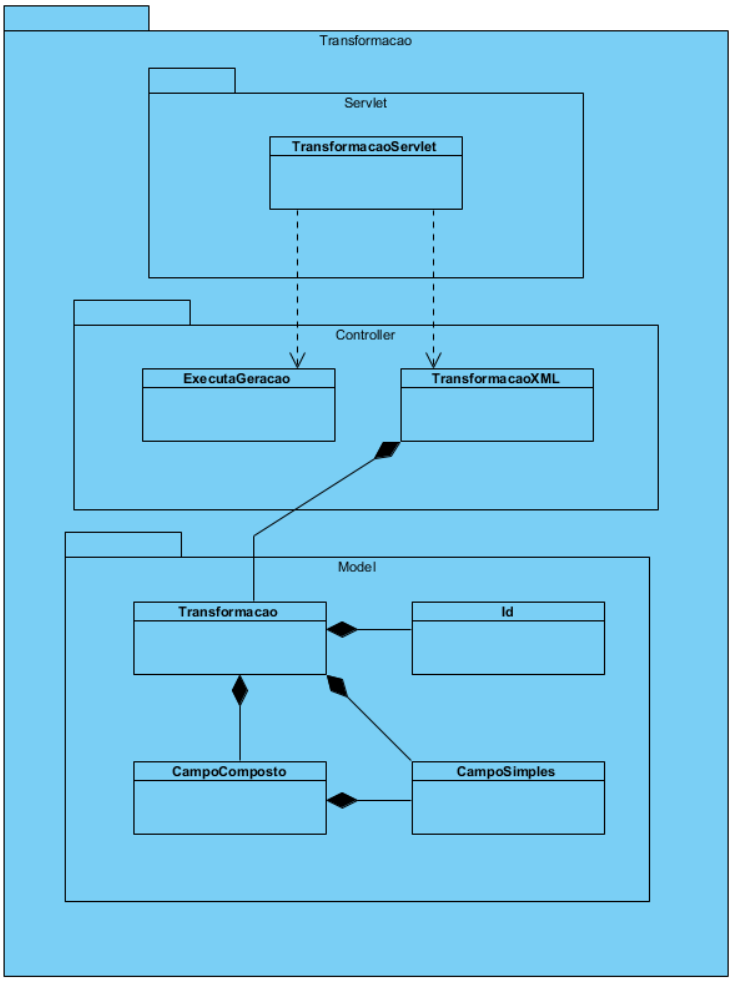


Figura 61 - Vista de implementação do Transformacao (diagrama de classes em UML)

Por fim, foi elaborado um terceiro diagrama de classes agregadas em pacotes, representativo da estrutura e relações das várias classes do Geracao (Figura 62).

Na Figura 62 são apresentados os dois pacotes que constituem a aplicação:

- Servlet, que contém a classe GeracaoServlet, sendo esta responsável por receber os pedidos efetuados pelo Transformacao;
- Controller, que contém a classe ExecutaGeracao, sendo esta responsável por criar os vários documentos PDF.

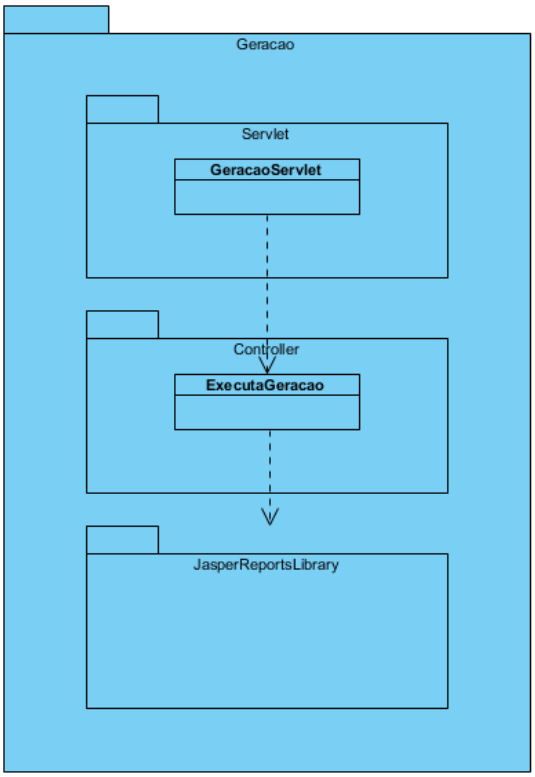


Figura 62 - Vista de implementação do Geracao (diagrama de classes em UML)

4.2.3 Design arquitetural de componente ConfiguraDados

Nesta secção é apresentado o design arquitetural do componente ConfiguraDados, apresentando as vistas (i) lógica e (ii) de implementação.

A vista de implantação nesta granularidade é a mesma que a apresentada para a granularidade de sistema.

Este componente é responsável por efetuar a criação do ficheiro de configurações e especificações, sendo um componente muito importante.

4.2.3.1 Vista lógica

O diagrama de componentes apresentado na Figura 63 representa a estrutura interna do componente ConfiguraDados.

É possível observar na Figura 63 que o ConfiguraDados tem três componentes, sendo estes o Controller, Model e View. Também é possível observar que o ConfiguraDados recebe um pedido, sendo o pedido enviado ao componente View, que este por sua vez irá comunicar com o componente Controller e com o componente Model.

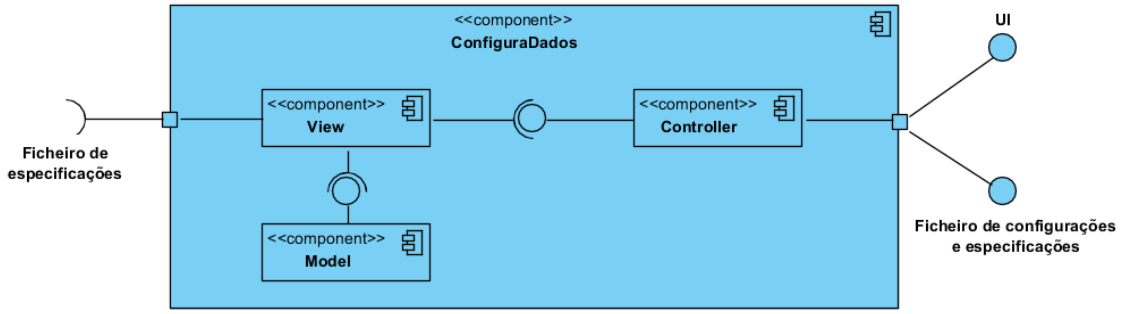


Figura 63 – Vista lógica do ConfiguraDados (diagrama de componentes em UML)

4.2.3.2 Vista de implementação

Este componente será construído adotando o estilo *standalone*/*desktop*, e será desenvolvido em Java, sendo que a UI adotará o framework Swing.

Foi elaborado um diagrama de classes agregadas em pacotes, representativo da estrutura e relações das várias classes do ConfiguraDados (Figura 64).

Na Figura 64 são apresentados os três pacotes que constituem a aplicação:

- View, que contém a classe ConfiguraDados responsável pela UI e por efetuar as chamadas às restantes classes;
- Controller, que contém a classe CreateXML responsável pela criação do ficheiro de configurações e especificações, a partir dos objetos criados utilizando os modelos;
- Model, que contém as classes Transformacao, CampoComposto, CampoSimples e Id, que representam os modelos utilizados, sendo criado um objeto Transformacao pelo ConfiguraDados no pacote da View, contendo este os outros objetos necessários.

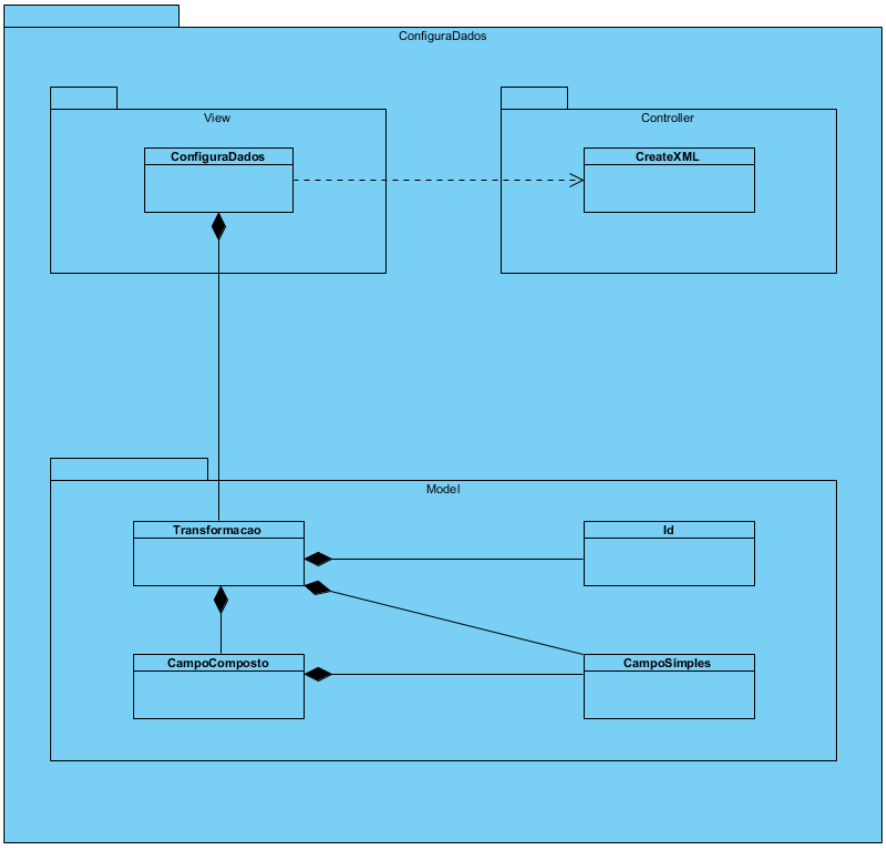


Figura 64 - Vista de implementação do ConfiguraDados (diagrama de classes em UML)

4.3 Sumário

Este capítulo descreveu a geração, enriquecimento e seleção de ideias, assim como a arquitetura proposta para o desenvolvimento do novo software, sendo possível obter um melhor conhecimento sobre o que é pretendido.

Todos os conhecimentos adquiridos através deste design arquitetural vão ser tidos em conta durante as atividades de implementação, implantação, teste e experiências do projeto.

De seguida será descrita a implementação do novo software, abordando todos os pontos importantes, como a tecnologia utilizada, padrões, regras e os seus testes efetuados.

5 Implementação

Este capítulo apresenta de forma detalhada a implementação/construção do projeto, os padrões aplicados, algoritmos e metodologias, descrevendo o ambiente de desenvolvimento e as tecnologias utilizadas para cada um dos softwares desenvolvidos.

Para a implementação, foram analisadas as várias soluções na secção 3, tendo sido adotadas as seguintes soluções:

- Problema 1, utilizando JAXB para efetuar a leitura e escrita dos ficheiros de configurações e especificações, uma vez que fornece um padrão de mapeamento dos objetos e usa a memória de maneira eficiente;
- Problema 2, utilizando o ReadBufferReader para efetuar a leitura de ficheiros fonte de dados e o SAX para efetuar a escrita de ficheiros de dados normalizado;
- Problema 3, utilizando o Jasper Reports para efetuar o design dos documentos PDF;
- Problema 4, utilizando o JasperUI para efetuar a criação dos documentos PDF;
- Problema 5, utilizando o SAX para efetuar a leitura de ficheiros de dados normalizados e a JasperReportsLibrary para efetuar a criação de documentos PDF.

Toda a implementação foi realizada sobre o NetBeans IDE uma vez que este oferece suporte abrangente para as tecnologias e melhorias de especificações Java mais recentes, antes de outros IDE, sendo o primeiro IDE gratuito a oferecer suporte a JDK 8, JDK 7, Java EE 7, entre outros [30].

5.1 Padrões e Regras

Um padrão é uma descrição ou modelo de como resolver um problema que pode ser usado em muitas situações diferentes, mas que têm um conjunto de caraterísticas em comum que permite e promovem a adoção de tal padrão.

Para o desenvolvimento doConfiguraDados e Transformacao foi aplicado o padrão de arquitetura de software MVC (Model-View-Controller), separando a parte de apresentação/visualização (View) da parte de negócio (Model) através de controladores (Controller).

O padrão MVC separa os componentes maiores possibilitando a reutilização de código e desenvolvimento paralelo de maneira eficiente. Este padrão começou por ser aplicado em aplicações com interfaces gráficas de utilizador (GUI) *standalone*, mas tornou-se popular no desenvolvimento de aplicações web e até mesmo para aplicações móveis, para desktop e para outros clientes [29].

A parte de apresentação/visualização é a de interface com o utilizador, sendo utilizada para receber a entrada de dados e apresentar visualmente o resultado. Não se preocupa em saber como o conhecimento foi obtido ou de onde, apenas o apresenta ao utilizador através de HTML, ASP, XML, Applets ou outros.

A parte de modelo é responsável por tudo o que a aplicação faz a partir do controlador num ou mais elementos de dados, uma vez que representa o domínio/negócio que se está a resolver/informatizar, nomeadamente os dados e o comportamento por detrás do negócio.

A parte de controlador é responsável por interpretar as ações de entrada pelo utilizador, enviando essas ações para o Modelo e para a janela de visualização onde serão realizadas as operações necessárias.

Para o desenvolvimento do Geracao apenas foi criada a vista e controlador, uma vez que este software apenas cria os documentos PDF utilizando o ficheiro de dados normalizado e o ficheiro de design PDF, não havendo nenhum modelo necessário.

Para o desenvolvimento do ConfiguraDados e Transformacao foi aplicado o padrão Builder, que pertence à classe dos padrões de criação dos padrões de desenho do GoF (Gang of Four), uma vez que o padrão Builder é um padrão de projetos de software comum, usado para encapsular a lógica de construção de um objeto. Este padrão facilita a criação dos objetos e é frequentemente utilizado quando a construção de um objeto é considerada complexa, ou quando se trata da construção de representações múltiplas de uma mesma classe.

Este padrão foi utilizado nestes softwares, uma vez que estes contêm os objetos denominados de Transformacao, CampoComposto, CampoSimples e Id (conforme apresentado na secção 4.2.2.3), permitindo variar a representação interna dos objetos, encapsular o código entre a construção e a representação, assim como controlar o processo de construção.

O Transformacao e o Geracao são compilados, gerando um ficheiro WAR, sendo posteriormente implantado num servidor aplicacional Tomcat da Empresa, para ser utilizado também pelos seus colaboradores através de um browser.

5.2 ConfiguraDados

O ConfiguraDados foi implementado em Java, sendo compilado, gerando um executável, sendo este instalado nas diversas máquinas da Empresa, de forma a ser utilizado pelos colaboradores da mesma para efetuar a criação dos ficheiros de configurações e especificações (Figura 65), uma vez que este contém uma UI gráfica (Graphical User Interface, Interface Gráfica com Utilizador em português), conforme apresentado na Figura 66 e Figura 67.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>02</inicio>
    <negritoNormal></negritoNormal>
    <separador></separador>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>2</fim>
    </listaId>
    <listaId>
        <tipo>REGRAS</tipo>
        <inicio>1</inicio>
        <fim>3</fim>
        <regra>99</regra>
    </listaId>
    <listaCampoSimples>
        <nome>tipo_documento_id</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>3</inicio>
        <fim>3</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>fe</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>4</inicio>
        <fim>5</fim>
    </listaCampoSimples>
    <listaCampoComposto>
        <nome>mensagens</nome>
        <repete>3</repete>
        <posicao>0</posicao>
        <id>52</id>
        <listaCampoSimples>
            <nome>mensagem</nome>
            <posicao>0</posicao>
            <id>52</id>
            <inicio>3</inicio>
            <fim>102</fim>
        </listaCampoSimples>
    </listaCampoComposto>
```

Figura 65 – Exemplo de um ficheiro de configurações e especificações de dados

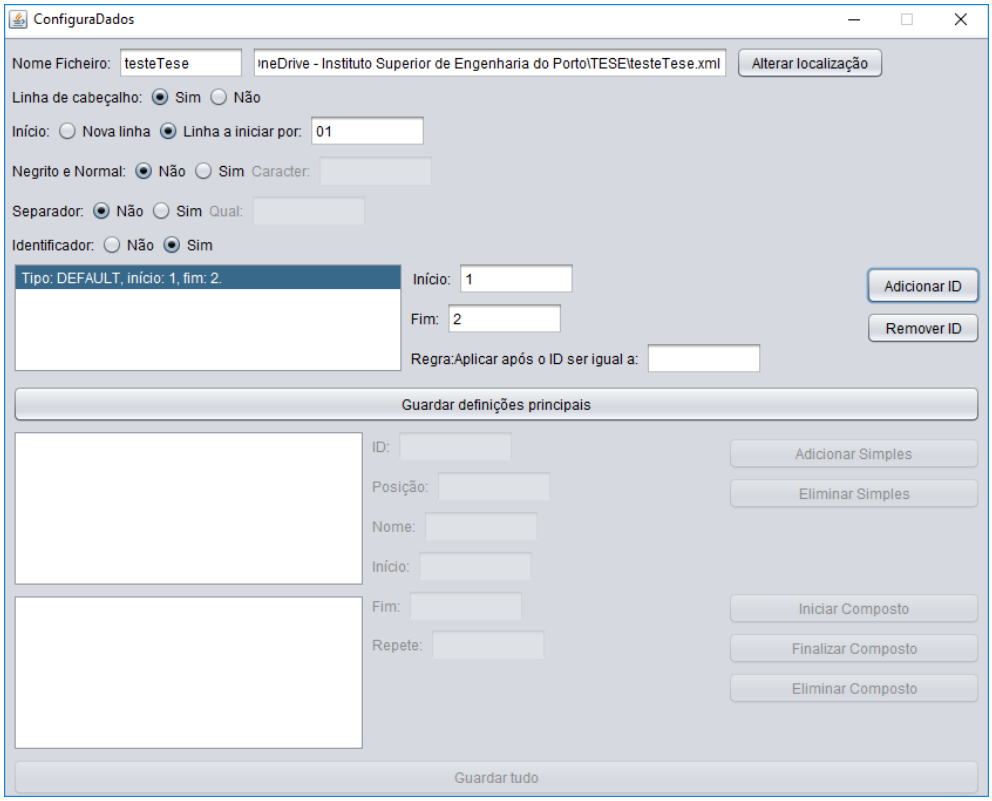


Figura 66 – UI da aplicação ConfiguraDados

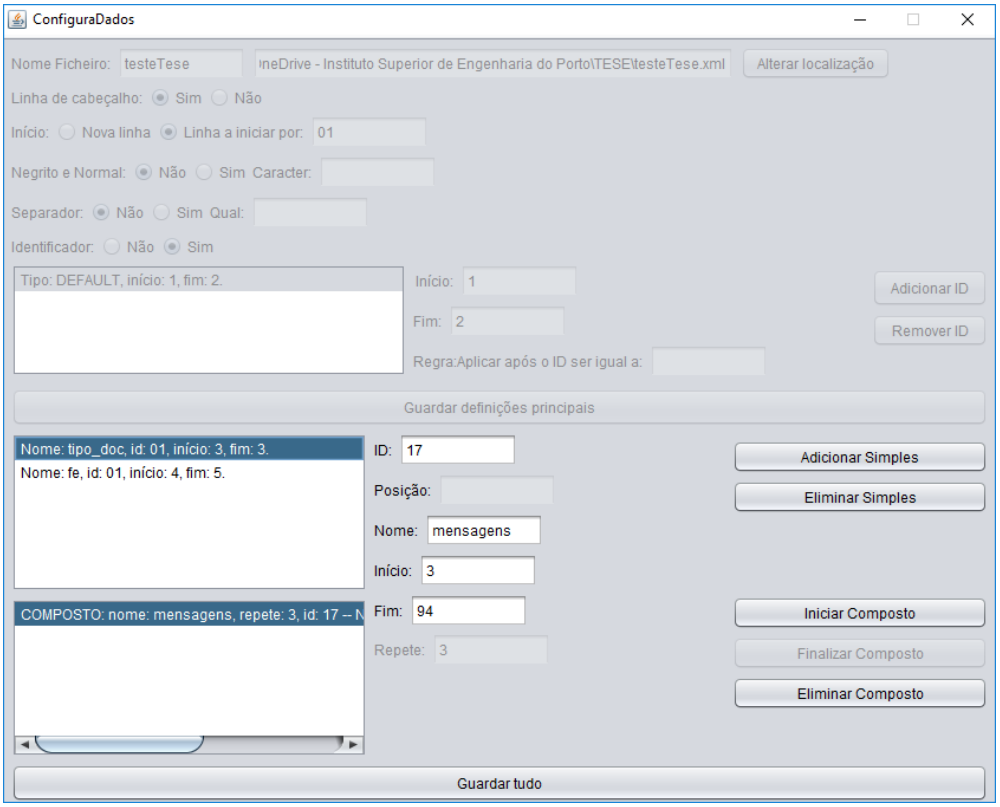


Figura 67 – UI da aplicação ConfiguraDados

Esta interface é formada através de componentes GUI, sendo estes os objetos que fazem a interação com o utilizador por dispositivos que venham a servir para entrada de dados. No desenvolvimento de aplicações *standalone/desktop*, existem dois *frameworks* de construção de interfaces gráficas: Swing ou AWT (Abstract Windows Toolkit).

Existem algumas diferenças entre o Swing e AWT. Para além dos componentes do AWT demorarem mais tempo a carregar e consumirem mais recursos do sistema do que os componentes Swing, também existem diferenças na aparência e comportamento dos componentes, uma vez que quando é um componente AWT, a sua aparência e comportamento diferem consoante a plataforma, enquanto que quando é um componente Swing, a sua aparência e comportamento é sempre o mesmo para todas as plataformas.

O Swing também contém uma lista de componentes maior do que o AWT, sendo necessário menos código no Swing para efetuar a mesma ação do que no AWT. O Swing utiliza o padrão de desenho MVC, e o AWT não [12].

Os componentes possuem classes que definem quais serão os seus estados e comportamentos, existindo uma hierarquia dessas classes conforme apresentado na Figura 68.

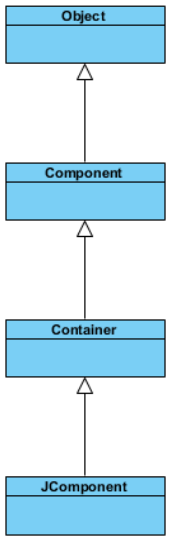


Figura 68 - Hierarquia das classes dos componentes GUI

O JComponent é do tipo Swing, e é uma subclasse de Container, sendo que todos os componentes Swing são Containers. O Container é do tipo AWT, e é uma subclasse de Component. Os Containers são janelas que podem ser usadas para organizar e mostrar outros componentes.

Alguns recursos comuns do JComponent incluem teclas de atalho, dicas de ferramenta, suporte para tecnologias de acessibilidade e suporte para localização de interface com o utilizador, sendo esta a classe que cobre os principais componentes que são usados nas janelas.

5.3 JasperUI

O JasperUI foi instalado nas diversas máquinas da Empresa, de forma a ser também utilizado pelos seus colaboradores na criação dos ficheiros de design PDF, uma vez que este contém uma UI, conforme apresentado na Figura 69.

Esta UI permite efetuar a criação dos ficheiros de design PDF, contendo componentes visuais de layout, com variados gráficos, tabelas, mapas, entre outros, permitindo também escrever expressões complexas.

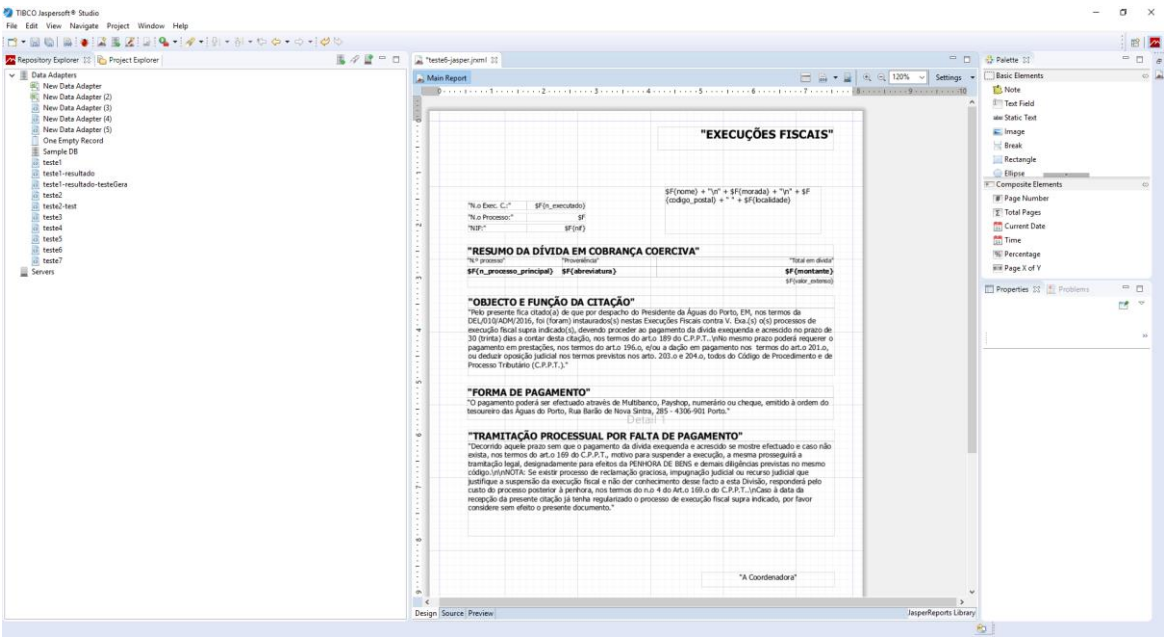


Figura 69 – UI da aplicação JasperUI

5.4 GeraPDF

O GeraPDF, e consequentemente os subcomponentes Transformacao e Geracao, são aplicações web em Java, chamados quando o colaborador da Empresa efetua um pedido ao servidor, efetuando a execução da aplicação.

O Transformacao efetua a criação do ficheiro de dados normalizado (Figura 70), através dos ficheiros fonte de dados (Figura 71) e do ficheiro de configurações e especificações. O

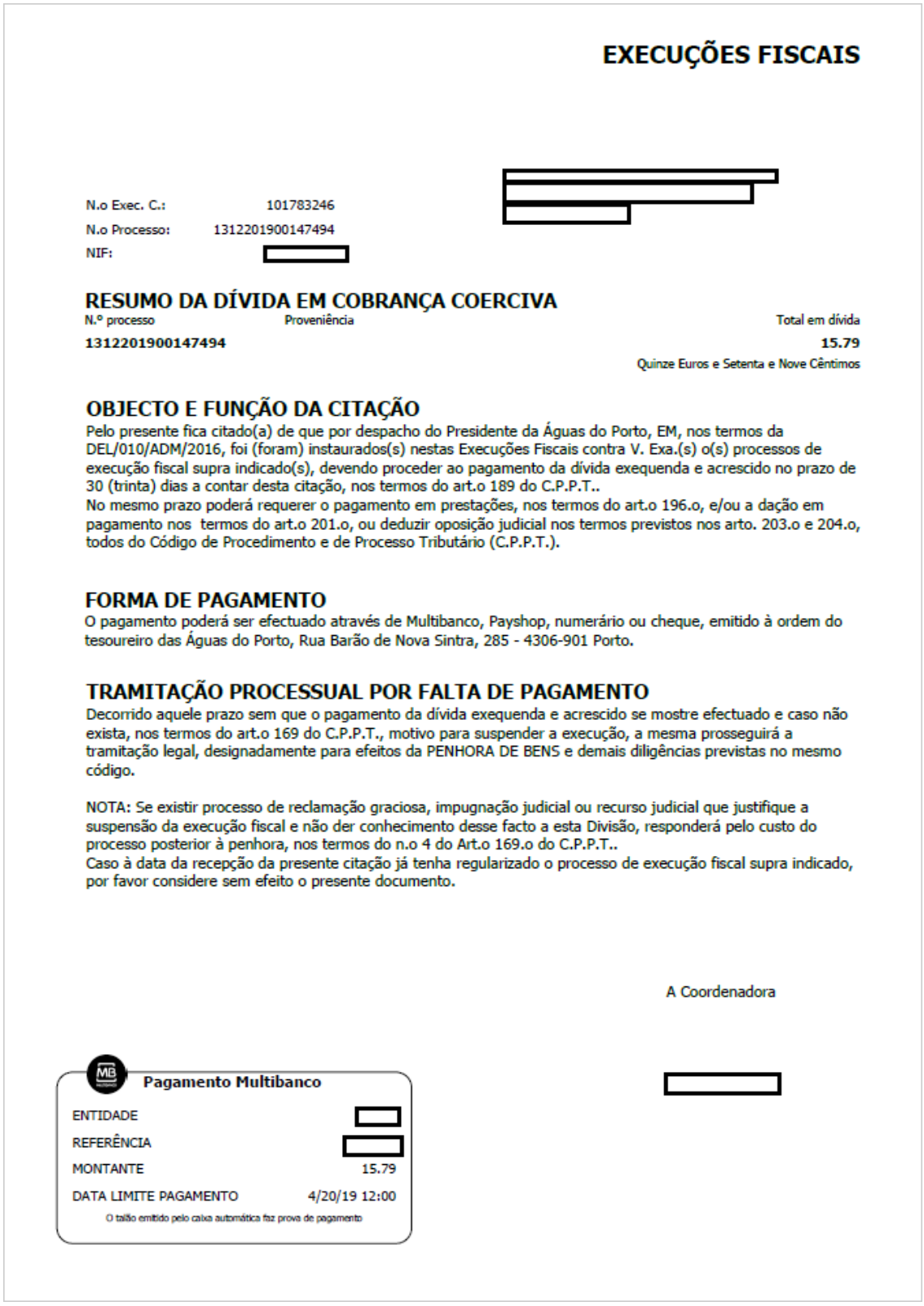
Geração efetua a criação dos documentos PDF (Figura 72) utilizando o ficheiro de dados normalizado.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<documentos>
    <documento>
        <LADC1>N</LADC1>
        <ADC1>PT50101269</ADC1>
        <LADC2>N</LADC2>
        <ADC2>00001579932</ADC2>
        <LCCL1>N</LCCL1>
        <CCL1>201920156584</CCL1>
        <LCLT1>N</LCLT1>
        <CLT1>00015799</CLT1>
        <LCNS1>N</LCNS1>
        <CNS1>1 m3</CNS1>
        <LCNT1>N</LCNT1>
        <CNT1>ATLANTIS</CNT1>
        <LCNT2>N</LCNT2>
        <CNT2>31/00000034131</CNT2>
        <LCNT3>N</LCNT3>
        <CNT3>15</CNT3>
        <LCNT4>N</LCNT4>
        <CNT4>1996.09.06</CNT4>
        <LCOD0>N</LCOD0>
        <COD0>10000360000E0154173R</COD0>
        <LCTT1>N</LCTT1>
        <CTT1>00157991925565581706 98 917 000012697  14</CTT1>
        <LDAT1>N</LDAT1>
        <DAT1>2019.05.23</DAT1>
        <LDAT2>N</LDAT2>
        <DAT2>2019.06.17</DAT2>
```

Figura 70 – Exemplo de um ficheiro de dados normalizado

```
00000teste
#1
NADC1PT50101269
NADC200001579932
NCCL1201920156584
NCLT100015799
NCNS11 m3
NCNT1ATLANTIS
NCNT231/00000034131
NCNT315
NCNT41996.09.06
NCOD010000360000E0154173R
NCTT1001579919255655817069 8 917 000012697  14
NDAT12019.05.23
NDAT22019.06.17
NDES1Valor a pagar em 2019.05.23       12.69
NDES2Juros de Mora para Dívida Vencida
BDET1Contas de Água;;;4,75;
NDET1  1º Escalão;1,0 m3;0,5800;0,58;6,00
NDET1  1º Escalão a Deduzir ;0,0 m3;0,5800;0,00;6,00
NDET1  Qt. Disponibilidade(de 2019.04.20 a 2019.05.20);31 dias;0,1333;4,14;6,00
NDET1  Taxa Recursos Hídricos - Água;0,8 m3;0,0312;0,02;6,00
```

Figura 71 – Exemplo de um ficheiro fonte de dados



EXECUÇÕES FISCAIS

[redacted]

N.o Exec. C.: 101783246
N.o Processo: 1312201900147494
NIF: [redacted]

RESUMO DA DÍVIDA EM COBRANÇA COERCIVA

N.º processo	Proveniência	Total em dívida
1312201900147494		**15.79**

Quinze Euros e Setenta e Nove Cêntimos

OBJECTO E FUNÇÃO DA CITAÇÃO

Pelo presente fica citado(a) de que por despacho do Presidente da Águas do Porto, EM, nos termos da DEL/010/ADM/2016, foi (foram) instaurados(s) nestas Execuções Fiscais contra V. Exa.(s) o(s) processos de execução fiscal supra indicado(s), devendo proceder ao pagamento da dívida exequenda e acrescido no prazo de 30 (trinta) dias a contar desta citação, nos termos do art.o 189 do C.P.P.T..
No mesmo prazo poderá requerer o pagamento em prestações, nos termos do art.o 196.o, e/ou a dação em pagamento nos termos do art.o 201.o, ou deduzir oposição judicial nos termos previstos nos arto. 203.o e 204.o, todos do Código de Procedimento e de Processo Tributário (C.P.P.T.).

FORMA DE PAGAMENTO

O pagamento poderá ser efectuado através de Multibanco, Payshop, numerário ou cheque, emitido à ordem do tesoureiro das Águas do Porto, Rua Barão de Nova Sintra, 285 - 4306-901 Porto.

TRAMITAÇÃO PROCESSUAL POR FALTA DE PAGAMENTO

Decorrido aquele prazo sem que o pagamento da dívida exequenda e acrescido se mostre efectuado e caso não exista, nos termos do art.o 169 do C.P.P.T., motivo para suspender a execução, a mesma prosseguirá a tramitação legal, designadamente para efeitos da PENHORA DE BENS e demais diligências previstas no mesmo código.

NOTA: Se existir processo de reclamação graciosa, impugnação judicial ou recurso judicial que justifique a suspensão da execução fiscal e não der conhecimento desse facto a esta Divisão, responderá pelo custo do processo posterior à penhora, nos termos do n.o 4 do Art.o 169.o do C.P.P.T..
Caso à data da recepção da presente citação já tenha regularizado o processo de execução fiscal supra indicado, por favor considere sem efeito o presente documento.

A Coordenadora

[redacted]

Pagamento Multibanco

ENTIDADE	[redacted]
REFERÊNCIA	[redacted]
MONTANTE	15.79
DATA LIMITE PAGAMENTO	4/20/19 12:00

O talão emitido pelo caixa automática faz prova de pagamento

Figura 72 – Exemplo de um documento PDF gerado

O Transformacao e o Geracao utilizam o “servlet”, vindo do inglês, e dá uma ideia de um servidor pequeno (servidorzinho), sendo este uma classe Java usada para estender as funcionalidades de um servidor, sendo que a aplicação é executada num servidor, recebendo os pedidos HTTP, processando-os, e enviando depois uma resposta, podendo ser em HTML, imagem, ou outro, tendo o seguinte funcionamento (Figura 73) [9]:

- Os clientes enviam o pedido para o servidor Apache através do browser;
- O servidor Apache recebe o pedido e envia-o ao servidor Tomcat;
- O servidor Tomcat recebe o pedido e envia-o ao servlet correspondente;
- O servlet processa o pedido e envia a resposta ao servidor Tomcat;
- O servidor Tomcat recebe a resposta e envia-a de volta ao servidor Apache;
- O servidor Apache recebe a resposta e envia-a de volta ao browser, que a apresenta ao cliente.

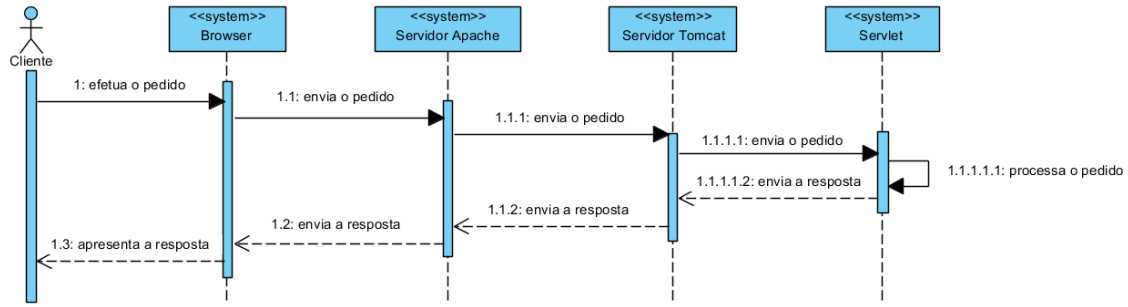


Figura 73 – Execução do Servlet

5.5 Testes

Os testes automáticos (e.g. programados) são bastante importantes, uma vez que não só asseguram que o correto comportamento do software se mantém quando é utilizado, como também permite que o desenvolvedor consiga melhorar o código, garantindo que não está a danificar a funcionalidade ao atualizar o software. Assim sendo, os principais fatores que motivam a sua utilização são os seguintes [28]:

- Previne o aparecimento de bugs devido a código mal desenvolvido;
- O código testado é mais confiável;

- Testa situações de sucesso e de falha;
- Gera e preserva um “conhecimento” sobre as regras de negócios do projeto;
- A sua execução é ilimitada (pelo computador e não pelo ser-humano).

As secções seguintes descrevem os testes desenvolvido no âmbito da implementação do projeto, nomeadamente testes unitários e de sistema.

5.5.1 Testes unitários

Os testes unitários são métodos de teste ao código de produção (i.e. de negócio), onde as unidades individuais de código são testadas de forma a verificar se estão aptos para uso, cumprindo as funcionalidades mediante os argumentos recebidos e os resultados gerados [28].

Os testes unitários foram desenvolvidos utilizando a abordagem Test-Driven Development (TDD) [48]. Trata-se duma abordagem evolutiva para o desenvolvimento de testes, em que primeiro é desenvolvido o teste e só depois é desenvolvido o código de produção (e.g. de negócio).

Neste projeto, tal como em vários dos projetos que utilizam Java, a framework utilizada para realizar testes unitários é JUnit, uma vez que esta fornece uma completa API para construir os testes, tendo as seguintes vantagens [48]:

- Possibilidade de verificar se cada unidade de código funciona da forma esperada;
- Facilita a criação e execução automática de testes, assim como a apresentação dos resultados;
- É orientada a objetos;
- É gratuita.

Para o ConfiguraDados e Transformacao foram criados testes para a criação dos vários objetos, validando se os objetos são criados e alterados corretamente.

Para todos os softwares foram criados testes aos vários métodos existentes, utilizando o padrão AAA (Arrange, Act and Assert), sugerindo que o método de teste seja dividido em três secções: organizar, atuar e afirmar. O organizar tem como objetivo a criação e inicialização dos objetos. O atuar tem como objetivo invocar o método a testar. Por fim, o afirmar tem como objetivo verificar se a ação do método a testar se comporta conforme o esperado.

O ConfiguraDados contém um método que é responsável pela criação do ficheiro de configurações e especificações, tendo sido criados testes a este método, conforme o código da Figura 74. Neste código, de acordo com o padrão AAA:

- Organizar – É efetuada a criação dos vários objetos (da linha 32 até à linha 75);
- Atuar – É efetuada a chamada ao método (linha 77);
- Afirmar:
    - É verificado se a resposta do método é a esperada (linha 79);
    - É validado se o ficheiro de configurações e especificações foi criado, verificando se este existe (linha 80);
    - É efetuada a leitura do conteúdo do ficheiro (da linha 82 até à linha 84);
    - É validado se o texto esperado é o texto do ficheiro (linha 86);
    - O ficheiro é eliminado (linha 87);
    - É validado se o ficheiro foi eliminado (linha 88).

```java
public void testCreate() throws FileNotFoundException, IOException {
    //Arrange
    CampoSimplesBuilder campoSimplesBuilder1 = new CampoSimplesBuilder();
    campoSimplesBuilder1.setPosicao(0);
    campoSimplesBuilder1.setNome("nome1");
    CampoSimples campoSimples1 = campoSimplesBuilder1.create();
    List<CampoSimples> listaCampoSimples1 = new ArrayList<>();
    listaCampoSimples1.add(campoSimples1);

    CampoSimplesBuilder campoSimplesBuilder2 = new CampoSimplesBuilder();
    campoSimplesBuilder2.setPosicao(0);
    campoSimplesBuilder2.setNome("nome2");
    CampoSimples campoSimples2 = campoSimplesBuilder2.create();
    List<CampoSimples> listaCampoSimples2 = new ArrayList<>();
    listaCampoSimples2.add(campoSimples2);

    CampoCompostoBuilder campoCompostoBuilder = new CampoCompostoBuilder();
    campoCompostoBuilder.setNome("nome");
    campoCompostoBuilder.setPosicao(1);
    campoCompostoBuilder.setRepete(1);
    campoCompostoBuilder.setListaCampoSimples(listaCampoSimples2);
    CampoComposto campoComposto = campoCompostoBuilder.create();
    List<CampoComposto> listaCampoComposto = new ArrayList<>();
    listaCampoComposto.add(campoComposto);

    TransformacaoBuilder transformacaoBuilder = new TransformacaoBuilder();
    transformacaoBuilder.setCabecalho("Sim");
    transformacaoBuilder.setInicio("Linha");
    transformacaoBuilder.setListaCampoSimples(listaCampoSimples1);
    transformacaoBuilder.setListaCampoComposto(listaCampoComposto);
    Transformacao transformacao = transformacaoBuilder.create();

    String textoEsperado = "<?xml version=\"1.0\" encoding=\"UTF-8\" "
            + "standalone=\"yes\"?><transformacao><cabecalho>Sim</cabecalho>"
            + "<inicio>Linha</inicio><negritoNormal>0</negritoNormal>"
            + "<listaCampoComposto><nome>nome</nome><repete>1</repete>"
            + "<posicao>1</posicao><listaCampoSimples><nome>nome2</nome>"
            + "<posicao>0</posicao><inicio>0</inicio><fim>0</fim>"
            + "</listaCampoSimples></listaCampoComposto><listaCampoSimples>"
            + "<nome>nome1</nome><posicao>0</posicao><inicio>0</inicio>"
            + "<fim>0</fim></listaCampoSimples></transformacao>";
    String caminho = "teste.xml";
    File file = new File(caminho);
    String texto = "";
    String s;
    CreateXML instance = new CreateXML();
    //Act
    boolean result = instance.create(caminho, transformacao);
    //Assert
    assertEquals(true, result);
    assertEquals(true, file.exists());
    BufferedReader br = new BufferedReader(new FileReader(caminho));
    while ((s = br.readLine()) != null) {
        texto = texto + s.trim();
    }
    br.close();
    assertEquals(textoEsperado, texto);
    file.delete();
    assertEquals(false, file.exists());
}
```

Figura 74 – Exemplo de teste efetuado na aplicação ConfiguraDados

O Transformacao contém um método responsável pela criação do ficheiro de dados normalizado, sendo efetuados os mesmos tipos de testes que foram efetuados para a aplicação ConfiguraDados, conforme o código da Figura 75. Neste código, de acordo com o padrão AAA:

- Organizar – É efetuada a criação dos vários objetos (da linha 37 até à linha 53);
- Atuar – É efetuada a chamada ao método (linha 55);
- Afirmar:
    - É verificado se a resposta do método é a esperada (linha 57);
    - É validado se o ficheiro de dados normalizado foi criado, verificando se este existe (linha 58);
    - É efetuada a leitura do conteúdo do ficheiro (da linha 60 até à linha 62);
    - É validado se o texto esperado é o texto do ficheiro (linha 64);
    - O ficheiro é eliminado (linha 65);
    - É validado se o ficheiro foi eliminado (linha 66).

```
34      public void testTransforma() throws JAXBException, IOException,
35              FileNotFoundException, TransformerConfigurationException, SAXException {
36          //Arrange
37          String ficheiroTransformacao = "teste-transformacao.xml";
38          String ficheiroTxt = "teste-ficheiro.txt";
39          String ficheiroResultado = "teste-resultado.xml";
40          File file = new File(ficheiroResultado);
41          String textoEsperado = "<?xml version=\"1.0\" encoding=\"UTF-8\" "
42                  + "standalone=\"no\"?><documentos><documento><nud>20190403000041144</nud>"
43                  + "<tipo_executado>NIF</tipo_executado><nif>149547773</nif>"
44                  + "<nome>FERNANDA CALISTA  DIAS</nome><morada>RUA LUIS CRUZ NR128 C 5</morada>"
45                  + "<codigo_postal>4150-467</codigo_postal><localidade>PORTO</localidade>"
46                  + "<data_emissao>03 de Abril     de 2019</data_emissao><acordo>000208/2018</acordo>"
47                  + "<n_prestacao>015</n_prestacao><referencia>002316030</referencia>"
48                  + "<payshop>62090184808080201800026640000000190430000020801157</payshop>"
49                  + "<montante>00000000026.64</montante>"
50                  + "<data_limite_pagamento>2019/04/30</data_limite_pagamento></documento></documentos>";
51          String texto;
52          texto = "";
53          String s;
54          //Act
55          boolean result = TransformacaoXML.transforma(ficheiroTransformacao, ficheiroTxt, ficheiroResultado);
56          //Assert
57          assertEquals(true, result);
58          assertEquals(true, file.exists());
59          BufferedReader br = new BufferedReader(new FileReader(ficheiroResultado));
60          while ((s = br.readLine()) != null) {
61              texto = texto + s.trim();
62          }
63          br.close();
64          assertEquals(textoEsperado, texto);
65          file.delete();
66          assertEquals(false, file.exists());
67      }
```

Figura 75 – Exemplo de teste efetuado na aplicação Transformacao

O Geracao contém um método responsável pela criação dos documentos PDF, sendo efetuados os mesmos tipos de testes que foram efetuados para as aplicações ConfiguraDados e Transformacao, conforme o código da Figura 76. Neste código, de acordo com o padrão AAA:

- Organizar – É efetuada a criação dos vários objetos (da linha 25 até à linha 29);
- Atuar – É efetuada a chamada ao método (linha 31);
- Afirmar:
    - É verificado se a resposta do método é a esperada (linha 33);
    - É validado se o documento PDF foi criado, verificando se este existe (linha 34);
    - O ficheiro é eliminado (linha 35);
    - É validado se o ficheiro foi eliminado (linha 36);
    - A pasta é eliminada (linha 37);
    - É validado se a pasta foi eliminada (linha 38).

```
23      public void testExecuta() throws Exception {
24          //Arrange
25          String ficheiroXML = "teste-resultado.xml";
26          String ficheiros = "teste-ficheiros";
27          String ficheiroJasper = "teste-jasper.jasper";
28          File pasta = new File(ficheiros);
29          File file1 = new File(ficheiros + File.separator + "1.pdf");
30          //Act
31          boolean result = ExecutaGeracao.executa(ficheiroXML, ficheiros, ficheiroJasper);
32          //Assert
33          assertEquals(true, result);
34          assertEquals(true, file1.exists());
35          file1.delete();
36          assertEquals(false, file1.exists());
37          pasta.delete();
38          assertEquals(false, pasta.exists());
39      }
```

Figura 76 – Exemplo de teste efetuado na aplicação Geracao

Todos estes testes são muito importantes para o bom funcionamento das aplicações, uma vez que garantem que as funcionalidades dos métodos são as esperadas, testando a criação dos vários objetos e ficheiros, verificando se os métodos efetua a criação dos ficheiros com sucesso, validando também o seu conteúdo.

5.5.2 Testes de sistema

O teste de sistema é a fase do processo de teste de software e de hardware em que o sistema completo (integrado) é testado num ambiente que simula o ambiente de produção verificando os seus requisitos. Estes testes não testam só requisitos funcionais, mas também requisitos não funcionais, testando se os mesmos estão de acordo com a expectativa do cliente [49].

Estes testes foram efetuados e estão descritos na secção 6.1, onde é efetuado a abordagem dos testes efetuados, efetuando depois uma análise e avaliação dos mesmos.

5.6 Síntese

Este capítulo descreveu as tecnologias utilizadas na implementação, assim como os padrões e regras utilizados no desenvolvimento do novo software, sendo possível obter um melhor conhecimento sobre estes.

Este capítulo também descreve os testes, sendo este um tema muito importante, efetuando uma descrição dos testes unitários e de sistema.

De seguida serão descritas as experiências e avaliações efetuadas ao novo software.

6 Experiências e Avaliação

Este capítulo descreve o processo de avaliação do software e servirá para apurar a eficácia, eficiência, manutenibilidade e configurabilidade do software desenvolvido.

A avaliação terá em conta os interesses dos colaboradores da Empresa, uma vez que são estes os utilizadores das aplicações desenvolvidas, conseguindo usufruir das aplicações de forma fácil e intuitiva.

6.1 Abordagem e Preparação

Para a avaliação do software desenvolvido, foi dado foco à avaliação dos resultados obtidos em diversos cenários, efetuando vários testes às partes do sistema.

6.1.1 ConfiguraDados

Para a avaliação do ConfiguraDados, uma vez que este tem uma interface com o utilizador, foram realizadas várias execuções do software, efetuando a criação de vários ficheiros de configurações e especificações, assim como foram efetuados testes funcionais. Os testes funcionais têm como objetivo testar as funcionalidades do mesmo.

Para efetuar estas experiências foi utilizada a mesma máquina referida na secção 2.3.2, sendo que o ConfiguraDados apenas foi instalado na máquina, ficando logo disponível para utilização.

De forma a avaliar o ConfiguraDados foi realizado um teste funcional para cada um dos ficheiros fonte de dados disponível, permitindo configurar todos os campos e especificações de leitura dos ficheiros necessários. Foram testados vários casos de testes, inserindo campos

válidos e inválidos, de forma a analisar se o comportamento seria o adequado, obtendo mensagens para o utilizador quando os campos inseridos são inválidos.

Este software foi apresentado aos cinco colaboradores da Empresa, para que estes pudessem também testar as suas funcionalidades. No final da apresentação foram efetuadas algumas questões relativamente ao software, sendo estas:

- "Na vossa opinião, o software é intuitivo?", sendo que os cinco colaboradores responderam que sim;
- "Na vossa opinião, o software é de fácil perceção?", sendo que os cinco colaboradores responderam que sim;
- "Na vossa opinião, o software contém mensagens de erro explicitas?", sendo que os cinco colaboradores também responderam que sim;
- "Na vossa opinião, o software vai de encontro às necessidades da empresa?", sendo que os cinco colaboradores também responderam que sim;
- "Na vossa opinião, o que poderá ser melhorado?", sendo que um dos colaboradores referiu que o aspeto da UI não era muito bom, sendo que posteriormente seria pedido alterações relativamente ao aspeto dos componentes, os restantes colaboradores concordaram, não tendo mais a acrescentar.

6.1.2 JasperUI

De forma a avaliar o JasperUI foi realizado um teste funcional para cada um dos ficheiros disponibilizado, de forma a perceber se era possível efetuar todas as configurações dos campos e dados.

Este software foi também apresentado aos cinco colaboradores da Empresa, para que estes pudessem também testar as suas funcionalidades. No final da apresentação foram também efetuadas algumas questões relativamente ao software, sendo estas:

- "Na vossa opinião, o software é intuitivo?", sendo que os cinco colaboradores responderam que sim;
- "Na vossa opinião, o software é de fácil perceção?", sendo que os cinco colaboradores também responderam que sim, tendo acrescentado que pode haver dificuldades para quem não tiver conhecimentos a nível de Inglês, podendo não conseguir perceber a descrição das funções existentes no software;

- “Na vossa opinião, o software vai de encontro às necessidades da empresa?”, sendo que os cinco colaboradores também responderam que sim, tendo sido acrescentado que tem funcionalidades que atualmente não são utilizadas, mas que futuramente poderão ser necessárias, sendo uma mais valia.

6.1.3 GeraPDF

Para o Transformacao e Geracao foram realizadas várias execuções do software, efetuando testes funcionais e não funcionais, avaliando tempos de resposta e memória utilizada para diferentes ficheiros fonte de dados. Os testes não funcionais têm como objetivo testar a confiabilidade, eficiência, eficácia, configurabilidade, usabilidade, manutenibilidade, adaptabilidade e portabilidade.

Para efetuar estas experiências foi utilizada a mesma máquina referida na secção 2.3.2, na qual os componentes Transformacao e Geracao são executados no servidor aplicacional Tomcat já instalado. Os ficheiros utilizados são os mesmos utilizados na secção 2.3.2, de forma a ser possível comparar os tempos de execução e memória utilizada, utilizando o Java VisualVM de forma a obter os tempos de execução e memória utilizada.

6.2 Resultados

6.2.1 ConfiguraDados e JasperUI

De forma a ser possível avaliar a configurabilidade, foram efetuados vários testes funcionais ao JasperUI e ao ConfiguraDados, utilizando para o segundo os diferentes tipos de ficheiros fonte de dados, de forma a verificar se este efetua o que é desejado.

Foram utilizados os sete ficheiros de especificações disponibilizados pela Empresa, de forma a testar o ConfiguraDados, criando o ficheiro de configurações e especificações de cada um.

Para efetuar a criação do primeiro ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 77, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 78. Neste ficheiro é possível verificar que o ficheiro contém cabeçalho, e que a leitura é efetuada utilizando um identificador (as regras do identificador estão descritas na listaId).

	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	dataficheiro	3	6	8
3	00	Identificador do Destinatário	9	30	38
4	01	Tipo documento	3	1	3
5	01	Fatura eletronica	4	2	5
6	02	Saldo anterior	5	15	19
7	02	Pagamentos	22	15	36
8	02	Saldo actual	39	15	53
9	04	nº fatura	8	22	29
10	04	Zona de cobrança	32	10	41
11	04	Zona de cobrança	44	10	53
12	04	Roteiro de cobrança	56	10	65

Figura 77 – Primeiro ficheiro de especificações

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>01</inicio>
    <negritoNormal/>
    <separador/>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>2</fim>
    </listaId>
    <listaCampoSimples>
        <nome>data_ficheiro</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>3</inicio>
        <fim>8</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>id_destinatario</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>9</inicio>
        <fim>38</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>tipo_doc</nome>
        <posicao>0</posicao>
        <id>01</id>
        <inicio>3</inicio>
        <fim>3</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>fe</nome>
        <posicao>0</posicao>
        <id>01</id>
        <inicio>4</inicio>
        <fim>5</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>saldo_anterior</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>5</inicio>
        <fim>19</fim>
    </listaCampoSimples>
```

Figura 78 – Primeiro ficheiro de configurações e especificações

Para efetuar a criação do segundo ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 79, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 80. Neste ficheiro é possível verificar que o ficheiro contém cabeçalho, e que a leitura é efetuada utilizando um identificador, sendo possível verificar que este contém duas regras (uma vez que contém dois listaId).

Linha ID	Descrição	Inicio	Comprimento	Fim
00	dataficheiro	3	8	10
02	Tipo documento Identificador	3	1	3
02	Fatura eletronica	4	2	5
02	Email	7	40	46
04	Tipo documento (fatura / nota de crédito)	73	20	92
04	nº fatura	93	30	122

Figura 79 – Segundo ficheiro de especificações

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>02</inicio>
    <negritoNormal></negritoNormal>
    <separador></separador>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>2</fim>
    </listaId>
    <listaId>
        <tipo>REGRAS</tipo>
        <inicio>1</inicio>
        <fim>3</fim>
        <regra>99</regra>
    </listaId>
    <listaCampoSimples>
        <nome>data_ficheiro</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>3</inicio>
        <fim>10</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>tipo_documento_id</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>3</inicio>
        <fim>3</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>fe</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>4</inicio>
        <fim>5</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>email</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>7</inicio>
        <fim>46</fim>
    </listaCampoSimples>
```

Figura 80 – Segundo ficheiro de configurações e especificações

Para efetuar a criação do terceiro ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 81, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 82. Neste ficheiro é possível verificar que o ficheiro contém cabeçalho, e que a leitura é efetuada utilizando também um identificador.

1	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	dataficheiro	3	8	10
3	02	Identificador ficheiro	5	20	24
4	02	Tipo documento	27	40	66
5	04	Id consumidor	71	14	84
6	04	Nº consumidor	86	7	92
7	08	Destinatario	53	40	92
8	09	Morada	53	40	92

Figura 81 – Terceiro ficheiro de especificações

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>02</inicio>
    <negritoNormal></negritoNormal>
    <separador></separador>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>2</fim>
    </listaId>
    <listaCampoSimples>
        <nome>data_ficheiro</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>3</inicio>
        <fim>10</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>id_ficheiro</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>5</inicio>
        <fim>24</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>tipo_documento</nome>
        <posicao>0</posicao>
        <id>02</id>
        <inicio>27</inicio>
        <fim>66</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>id_consumidor</nome>
        <posicao>0</posicao>
        <id>04</id>
        <inicio>71</inicio>
        <fim>84</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>n_consumidor</nome>
        <posicao>0</posicao>
        <id>04</id>
        <inicio>86</inicio>
        <fim>92</fim>
    </listaCampoSimples>
```

Figura 82 – Terceiro ficheiro de configurações e especificações

Para efetuar a criação do quarto ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 83, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 84. Neste ficheiro é possível verificar que o ficheiro contém cabeçalho, e que a leitura é efetuada utilizando também um identificador.

1	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	Nome do ficheiro	3	9	11
3	00	Data	12	10	21
4	00	Nº sequencial	22	2	23
5	00	Nome do ficheiro antigo	24	23	46
6	00	Código Entidade SIBS	47	5	51
7	11	NUD	3	16	18
8	11	Tipo Executado	19	3	21
9	11	NIF	22	9	30

Figura 83 – Quarto ficheiro de especificações

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>11</inicio>
    <negritoNormal></negritoNormal>
    <separador></separador>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>2</fim>
    </listaId>
    <listaCampoSimples>
        <nome>nome_ficheiro</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>3</inicio>
        <fim>11</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>data</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>12</inicio>
        <fim>21</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>n_sequencial</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>22</inicio>
        <fim>23</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>nome_ficheiro_antigo</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>24</inicio>
        <fim>46</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>codigo_entidade_sibs</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>47</inicio>
        <fim>51</fim>
    </listaCampoSimples>
```

Figura 84 – Quarto ficheiro de configurações e especificações

Para efetuar a criação do quinto ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 85, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 86. Neste ficheiro é possível verificar que o ficheiro não contém cabeçalho, e que a leitura é efetuada utilizando um identificador e um separador, havendo também a distinção de negrito e normal.

1	Local	Campo
2	01	Normal ou Bold
3	2-5	Nome do Campo
4	6-..	Valor do Campo
5		

Figura 85 – Quinto ficheiro de especificações

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Não</cabecalho>
    <inicio>#</inicio>
    <negritoNormal>1</negritoNormal>
    <separador>;</separador>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>1</fim>
    </listaId>
    <listaId>
        <tipo>REGRAS</tipo>
        <inicio>2</inicio>
        <fim>5</fim>
        <regra>#</regra>
    </listaId>
    <listaCampoSimples>
        <nome>CNT1</nome>
        <posicao>0</posicao>
        <id>CNT1</id>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>CNT2</nome>
        <posicao>0</posicao>
        <id>CNT2</id>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>CNT3</nome>
        <posicao>0</posicao>
        <id>CNT3</id>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>CNT4</nome>
        <posicao>0</posicao>
        <id>CNT4</id>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
```

Figura 86 – Quinto ficheiro de configurações e especificações

Para efetuar a criação do sexto ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 87, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 88. Neste ficheiro é possível verificar que o ficheiro contém cabeçalho, e que a leitura é efetuada também utilizando um identificador.

	Linha ID	Descrição	Inicio	Comprimento	Fim
2	00	Nome ficheiro	3	6	8
3	00	Data	9	10	18
4	00	Nº sequencial	19	2	20
5	00	Código entidade	21	5	25
6	00	Código entidade ctt	26	5	30
7	11	Nº avisos SIBS	3	9	11
8	11	Forma citação	12	1	12
9	11	Nº executado	13	9	21
10	11	NIF	22	9	30
11	11	Tipo executado	31	1	31
12	11	Nome	32	160	191
13	11	Morada	191	161	351
14	11	Código postal	352	8	359

Figura 87 – Sexto ficheiro de especificações

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>11</inicio>
    <negritoNormal>0</negritoNormal>
    <separador></separador>
    <listaId>
        <tipo>DEFAULT</tipo>
        <inicio>1</inicio>
        <fim>2</fim>
    </listaId>
    <listaCampoSimples>
        <nome>nome_ficheiro</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>3</inicio>
        <fim>8</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>data</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>9</inicio>
        <fim>18</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>n_sequencial</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>19</inicio>
        <fim>20</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>codigo_entidade</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>21</inicio>
        <fim>25</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>codigo_entidade_ctt</nome>
        <posicao>0</posicao>
        <id>00</id>
        <inicio>26</inicio>
        <fim>30</fim>
    </listaCampoSimples>
```

Figura 88 – Sexto ficheiro de configurações e especificações

Por fim, para efetuar a criação do sétimo ficheiro de configurações e especificações foi utilizado o ficheiro de especificações apresentado na Figura 89, sendo criado o respetivo ficheiro de configurações e especificações apresentado na Figura 90. Neste ficheiro é possível verificar que o ficheiro contém cabeçalho, que a leitura é efetuada utilizando um separador, e que uma nova linha representa um novo documento.

	A	B
1	**Campo**	**Descritivo**
2	EB_01	Data de Emissão
3	EB_02	Identificador do Destinatário
4	EB_03	Nome Destinatário
5	MOR_01	Morada 1
6	MOR_02	Morada 2
7	MOR_03	Morada 3
8	MOR_04	Morada 4
9	MOR_05	Morada 5
10	MES_C	Indicação do Mês Corrente
11	GRAF_00	Valor (Mesmo mês ano anterior)
12	GRAF_01	Valor (Mes Corrente - 11)
13	GRAF_02	Valor (Mes Corrente - 10)
14	GRAF_03	Valor (Mes Corrente - 9)
15	GRAF_04	Valor (Mes Corrente - 8)

CABEÇALHO | detalhes | colunas_teste

Figura 89 – Sétimo ficheiro de especificações

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<transformacao>
    <cabecalho>Sim</cabecalho>
    <inicio>Linha</inicio>
    <negritoNormal>0</negritoNormal>
    <separador>|</separador>
    <listaCampoSimples>
        <nome>EB_01</nome>
        <posicao>0</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>EB_02</nome>
        <posicao>1</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>EB_03</nome>
        <posicao>2</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>MOR_01</nome>
        <posicao>3</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>MOR_02</nome>
        <posicao>4</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>MOR_03</nome>
        <posicao>5</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
    <listaCampoSimples>
        <nome>MOR_04</nome>
        <posicao>6</posicao>
        <inicio>0</inicio>
        <fim>0</fim>
    </listaCampoSimples>
```

Figura 90 – Sétimo ficheiro de configurações e especificações

Foram criados os ficheiros de design PDF para cada um dos sete ficheiros de especificações disponibilizados pela Empresa, de forma a testar o JasperUI, criando o ficheiro PDF com vários documentos.

6.2.2 GeraPDF

Após a execução do GeraPDF, utilizando os ficheiros de design PDF e os ficheiros de configurações e especificações criados na secção anterior, e os ficheiros fonte de dados disponibilizados, os resultados obtidos foram os apresentados na Tabela 17, indicando o tamanho do ficheiro, a quantidade de documentos PDF criados, e os respetivos valores.

Tamanho ficheiro (MB)	**Quantidade de documentos PDF**	**Tempo de execução (min)**	**Memória Utilizada (GB)**
2,3 (Ficheiro 1)	992	2,47	30,71
3,56 (Ficheiro 2)	1 536	3,42	47,37
4,3 (Ficheiro 3)	1 856	4,02	57,22
6,53 (Ficheiro 4)	2 816	6,46	86,91
8,01 (Ficheiro 5)	3 456	7,72	106,64
10,9 (Ficheiro 6)	4 736	10,56	146,14
24 (Ficheiro 7)	11 014	25,12	338,07
27 (Ficheiro 8)	12 294	27,32	377,56
31,1 (Ficheiro 9)	14 054	31,62	431,66

Tabela 17 – Tabela com os valores do tempo de execução e memória utilizada

De forma a perceber e comparar o tempo de execução dos vários ficheiros, foi criado o gráfico apresentado na Tabela 18. Com este gráfico podemos observar que o tempo de execução obteve um crescimento linear relativo ao tamanho do ficheiro, executando todos os ficheiros de fonte de dados.

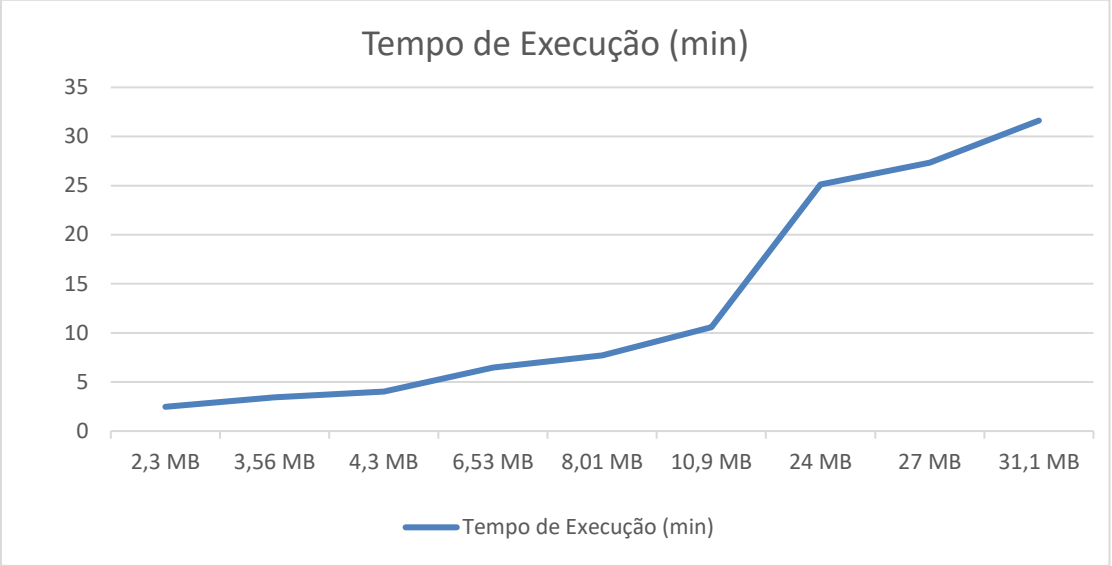


Tabela 18 – Gráfico do tempo de execução (min)

Também de forma a perceber e comparar a memória utilizada dos vários ficheiros, foi criado o gráfico apresentado na Tabela 19. Com este gráfico podemos observar que a quantidade de memória usada também sofreu um crescimento linear relativo ao tamanho do ficheiro, executando todos os ficheiros de fonte de dados.

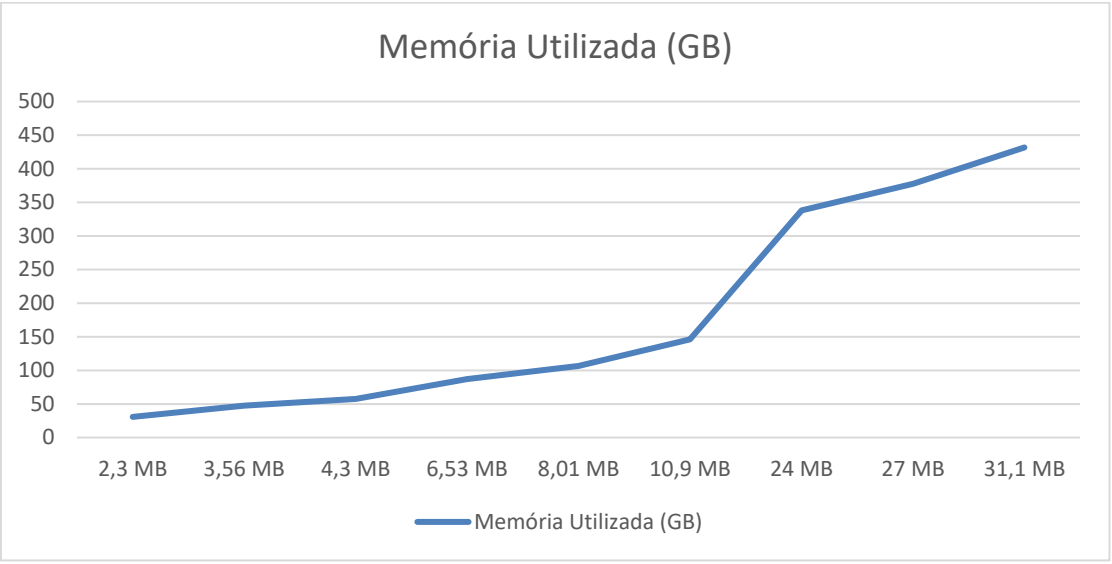


Tabela 19 – Gráfico da memória utilizada (GB)

6.3 Avaliação comparativa: ProcessaDados vs. GeraPDF

Nesta secção serão comparados os resultados obtidos relativos ao ProcessaDados com o GeraPDF (Transformacao e Geracao).

De forma a perceber e avaliar o tempo de execução dos vários ficheiros, comparando o ProcessaDados com o GeraPDF foi criado o gráfico apresentado na Tabela 20 comparando os tempos de execução.

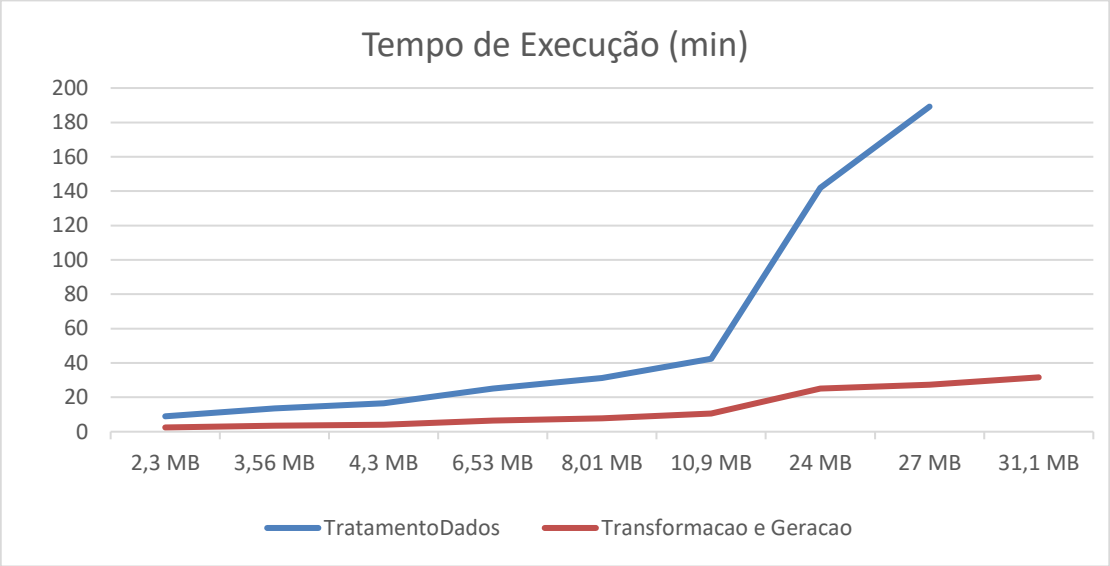


Tabela 20 - Gráfico do tempo de execução (min)

Com estes gráficos é possível observar a enorme diferença entre os dois softwares, sendo que o novo reduz significativamente o tempo de execução e é sempre relativamente baixo.

Analisando o ficheiro 1, com 2,3MB pode-se observar que o GeraPDF demorou 2,47 minutos, enquanto o ProcessaDados demorou 8,96 minutos (cerca de 4 vezes mais).

Analisando o ficheiro 8, com 27MB é possível observar que o GeraPDF demorou 27,32 minutos, enquanto o ProcessaDados demorou 189,19 minutos (cerca de 7 vezes mais).

É possível também observar que o ficheiro 9 com 31,1MB foi executado com sucesso pelo GeraPDF, tendo demorado apenas 31,62 minutos, algo que não era possível no ProcessaDados.

De forma a perceber e avaliar a memória utilizada pelos vários ficheiros, comparando o ProcessaDados com o GeraPDF foi criado o gráfico apresentado na Tabela 21 comparando a memória utilizada.

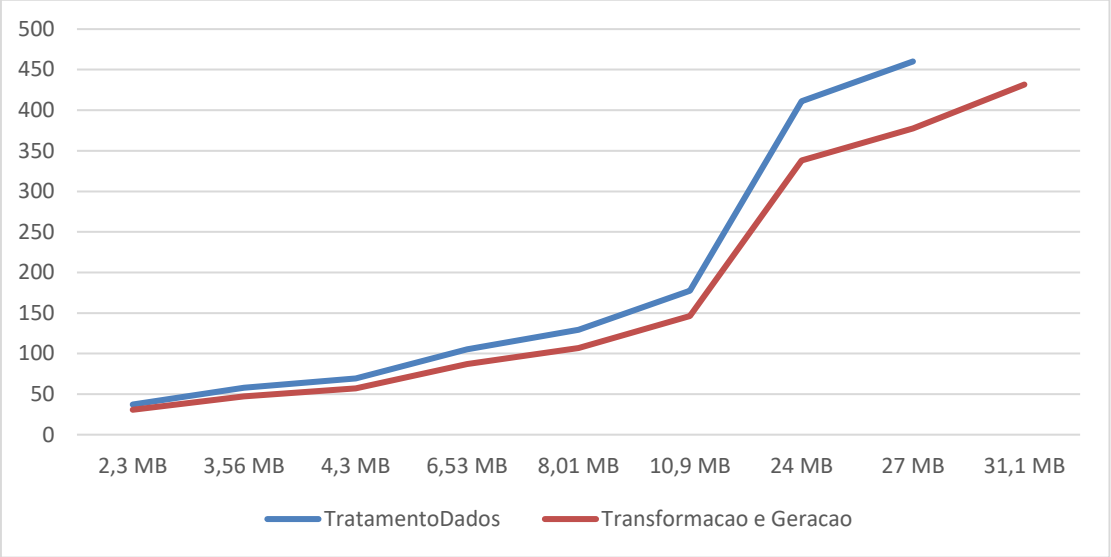


Tabela 21 - Gráfico da memória utilizada (GB)

Com estes gráficos é possível observar uma pequena diferença entre os dois softwares, sendo que o novo reduz a utilização de memória.

Analisando o ficheiro 1, com 2,3MB pode-se observar que o ProcessaDados utilizou 37,19GB, enquanto o GeraPDF utilizou 30,7GB, podendo-se concluir que o GeraPDF reduziu cerca de 20% da memória utilizada em relação ao ProcessaDados.

Analisando o ficheiro 8, com 27MB é possível observar que o ProcessaDados utilizou 460,09GB, enquanto o GeraPDF utilizou 377,56GB, podendo-se concluir que o GeraPDF reduziu também cerca de 20% da memória utilizada em relação ao ProcessaDados.

Por fim, é possível também observar que o ficheiro 9 com 31,1MB foi executado com sucesso pelo GeraPDF, tendo utilizado apenas 431,66GB.

6.4 Apreciação crítica

Após os resultados obtidos e da avaliação efetuado na secção anterior, é possível concluir que o GeraPDF cumpre os seus requisitos, havendo uma enorme melhoria em relação ao tempo de execução, e havendo uma melhoria significativa na memória utilizada.

O JasperUI disponibiliza inúmeras funcionalidades que possivelmente não serão utilizadas pela Empresa, uma vez que atualmente não são necessárias, mas que poderão vir a ser, sendo uma mais valia a utilização deste software, uma vez que é muito completo.

O ConfiguraDados dá resposta às necessidades dos clientes, permitindo a configuração dos campos e especificações de leitura dos ficheiros pelo colaborador da Empresa, o que era impossível com o software original.

Por fim, é possível verificar que o JasperUI e o ConfiguraDados são configuráveis e adaptáveis, sendo possível efetuar todas as funcionalidades necessárias.

7 Sumário

Neste capítulo são apresentadas as conclusões acerca do projeto realizado, assim como uma descrição de todos os objetivos atingidos neste projeto, referindo também as limitações e trabalho futuro, visando algumas melhorias na aplicação desenvolvida.

Por fim, é realizada uma apreciação global acerca do projeto e a aprendizagem obtida.

7.1 Objetivos alcançados

O grande objetivo deste projeto foi o desenvolvimento de um software que seja configurável e adaptável, permitindo que os colaboradores da Empresa configurem os campos e especificações de leitura dos vários ficheiros fontes de dados, processando/gerando os documentos PDF, melhorando também significativamente a sua eficácia e eficiência.

De forma a cumprir o objetivo, foi necessário efetuar uma pesquisa para os vários requisitos, de forma a analisar e avaliar cada uma das soluções encontradas, efetuando também uma pesquisa de ferramentas de desenho de relatórios/documentos, de forma a analisar qual a melhor a ser utilizada. Após esta análise foi desenvolvido o novo software aplicando os requisitos pretendidos, adotando boas práticas de desenvolvimento, efetuando também testes unitários e de sistema de forma a garantir o seu bom funcionamento.

Por fim, foram efetuados testes funcionais e não funcionais, obtendo os tempos de execução e memória utilizada, realizando uma avaliação aos resultados obtidos, efetuado uma comparação com os valores obtidos no ProcessaDados, podendo-se concluir que os tempos de execução do novo software foram bastante reduzidos em comparação com o ProcessaDados, tendo melhorado a eficácia e eficiência do software, cumprindo também os restantes requisitos.

O documento resultante sistematiza tanto os pressupostos tomados como as decisões, podendo ser uma mais-valia para a Empresa, uma vez que o documento contém esclarecimentos sobre tecnologias, decisões e implementação das funcionalidades.

7.2 Limitações

Uma vez que este projeto necessita de ações por parte do utilizador, sendo este um colaborador da Empresa, é necessário que todos os colaboradores da mesma tenham uma formação, de forma a ganharem conhecimento sobre o funcionamento do GeraPDF, do ConfiguraDados e do JasperUI, podendo haver desinteresse por parte dos colaboradores, ou até mesmo dificuldades em utilizar os softwares, ou compreender as várias funções disponibilizadas pelos mesmos.

É muito importante que a Empresa garanta que existem pelo menos dois colaboradores que sejam capazes de executar as ações necessárias nos softwares, garantindo assim que existe sempre alguém capaz de efetuar essas ações para o bom funcionamento da mesma.

7.3 Trabalho futuro

Todas as melhorias sugeridas fruto dos testes e das apresentações efetuadas aos colaboradores da Empresa devem ser tomadas em consideração. No que diz respeito à satisfação dos utilizadores, esta foi feita junto dos colaboradores da Empresa.

Ao efetuar a apresentação do novo software aos colaboradores da Empresa foi referido pelos mesmos que o aspeto da UI do ConfiguraDados não era muito bom, sendo necessário efetuar algumas alterações relativamente ao aspeto dos componentes. Este será um trabalho futuro, tornando os componentes mais apelativos visualmente.

7.4 Apreciação final e pessoal

Considero que ao longo do projeto, o meu conhecimento sobre o negócio tenha aumentado, assim como de engenharia de software e métodos de trabalho que desconhecia.

A nível pessoal sinto que melhorei bastante tanto a capacidade de trabalho individual como de pesquisa. Foram também desenvolvidas e melhoradas as capacidades de elaboração de documentos relativos a um projeto.

Sobre o software, sinto que este irá trazer várias vantagens à Empresa, uma vez que poderá expandir o seu negócio e aumentar o número de clientes, já conseguindo dar resposta às necessidades dos mesmos, não sendo necessário alterações ou implementações no software, sendo possível os colaboradores da mesma efetuarem as tarefas necessárias para o processamento necessário.

A Empresa passará a ter vantagem sobre as suas empresas concorrentes, uma vez que esta é capaz de efetuar melhores preços aos seus clientes, e contém uma enorme capacidade de impressão e envelopagem de documentos.

8 Referências bibliográficas

[1] Adhi, B. P. (Maio de 2019). *Performance comparison of reporting engine birt, jasper report, and crystal report on the process business intelligence*. Obtido de Researchgate: https://www.researchgate.net/publication/332831027_Performance_comparison_of_reporting_engine_birt_jasper_report_and_crystal_report_on_the_process_business_intelligence

[2] Aniche, M. (1 de Julho de 2014). *Unidade, integração ou sistema? Qual teste fazer?* Obtido de Caelum: https://blog.caelum.com.br/unidade-integracao-ou-sistema-qual-teste-fazer/

[3] *Apache Tomcat*. (s.d.). Obtido de Apache Tomcat: http://tomcat.apache.org/

[4] *Avoiding Apache Tomcat Out Of Memory Errors*. (s.d.). Obtido de MuleSoft: https://www.mulesoft.com/tcat/tomcat-oome-out-of-memory-error

[5] Bodnar, J. (17 de dezembro de 2017). *Reading text files in Java*. Obtido de zetcode: http://zetcode.com/articles/javareadtext/

[6] Bodnar, J. (3 de fevereiro de 2018). *Java DOM tutorial*. Obtido de zetcode: http://zetcode.com/java/dom/

[7] Carvalho, E. (30 de Janeiro de 2017). *TDD — Como não ser um amador*. Obtido de Medium: https://bar8.com.br/tdd-abap-sap-f8b5c1780049

[8] *Chapter 1: What is Software Architecture?* (14 de janeiro de 2010). Obtido de Microsoft: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658098%28v%3dpandp.10%29

[9] Chauhan, A. (s.d.). *Introduction to Java Servlets*. Obtido de GeeksforGeeks: https://www.geeksforgeeks.org/introduction-java-servlets/

[10]*Conceito de eficácia*. (12 de Janeiro de 2011). Obtido de Conceito: https://conceito.de/eficacia

[11]*Conceito de eficiência*. (30 de Março de 2011). Obtido de Conceito: https://conceito.de/eficiencia

[12]*Difference between Swing and AWT in Java – AWT vs Swing*. (6 de Abril de 2018). Obtido de My Easy Tuts: https://myeasytuts.com/difference-between-swing-and-awt-in-java/

[13]Douglas, A. (2012). *Introdução a Servlets em Java*. Obtido de Devmedia: https://www.devmedia.com.br/space/allan-douglas

[14]Dourado, L. S. (2013). *Introdução ao desenvolvimento de aplicações web*. Obtido de Devmedia: https://www.devmedia.com.br/introducao-ao-desenvolvimento-de-aplicacoes-web/29798

[15]*Estúdio Jaspersoft*. (s.d.). Obtido de Jaspersoft Community: https://community.jaspersoft.com/project/jaspersoft-studio

[16]*Exporting reports with Jaspersoft Studio*. (s.d.). Obtido de Jaspersoft Community: https://community.jaspersoft.com/wiki/exporting-reports-jaspersoft-studio

[17]Gomes, P. (9 de Setembro de 2017). *Unit Testing and the Arrange, Act and Assert (AAA) Pattern*. Obtido de Medium: https://medium.com/@pjbgf/title-testing-code-ocd-and-the-aaa-pattern-df453975ab80

[18]GUBERT, G. (23 de Julho de 2014). *Canvas: O Que é o Business Model Canvas e Como Funciona*. Obtido de Arsenal Empreendedor: http://arsenalempreendedor.com.br/canvas-o-que-e-o-canvas-e-como-funciona/

[19]Humberto, C. (s.d.). *Design Patterns: Padrões "GoF"*. Obtido de Devmedia: https://www.devmedia.com.br/design-patterns-padroes-gof/16781

[20]Jenkov, J. (12 de setembro de 2018). *Java IO: InputStreamReader*. Obtido de Jenkov: http://tutorials.jenkov.com/java-io/inputstreamreader.html

[21]Kokemuller, N. (12 de fevereiro de 2019). *What Is Customer Perceived Value?* Obtido de Chron: https://smallbusiness.chron.com/customer-perceived-value-23692.html

[22]Krill, P. (21 de Setembro de 2017). *Java EE 8 is here: What you need to know*. Obtido de InfoWorld: https://www.infoworld.com/article/3226777/java-ee-8-is-here-what-you-need-to-know.html

[23]Kumar, P. (s.d.). *Different ways of Reading a text file in Java*. Obtido de geeksforgeeks: https://www.geeksforgeeks.org/different-ways-reading-text-file-java/

[24]Kumar, P. (s.d.). *How to write XML file in Java (DOM Parser)*. Obtido de journaldev: https://www.journaldev.com/1112/how-to-write-xml-file-in-java-dom-parser

[25]*Lesson: Formatting Objects Basics*. (s.d.). Obtido de Webucator: https://www.webucator.com/tutorial/learn-xsl-fo/formatting-objects-basics.cfm

[26]Liu, E. (s.d.). *Seis Elementos de um Grande Software num Portal Web para Clientes*. Obtido de Computerworld: https://www.computerworld.com.pt/itd/seis-elementos-de-um-grande-software-num-portal-web-para-clientes/

[27]Longen, A. (22 de Janeiro de 2019). *O que é Apache? Uma visão aprofundada do servidor Apache*. Obtido de Hostinger: https://www.hostinger.com.br/tutoriais/o-que-e-apache

[28]Manoel. (2009). *JUnit Tutorial*. Obtido de DevMedia: https://www.devmedia.com.br/junit-tutorial/1432

[29]Massari, J. (s.d.). *Padrão MVC | Arquitetura Model-View-Controller*. Obtido de PortalGSTI: https://www.portalgsti.com.br/2017/08/padrao-mvc-arquitetura-model-view-controller.html

[30]*NetBeans IDE - A Forma Mais Inteligente e Rápida de Codificar*. (s.d.). Obtido de Netbeans: https://netbeans.org/features/index_pt_BR.html

[31]Nicola, S. (s.d.). *Análise_Valor_Aula_4_20NOV_2018_1hora_AHP.pdf*. Obtido de Moodle: https://moodle.isep.ipp.pt/mod/resource/view.php?id=137680

[32]Oracle. (s.d.). *Java Architecture for XML Binding (JAXB)*. Obtido de Oracle: https://www.oracle.com/technetwork/articles/javase/index-140168.html#introjb

[33]Palmeira, T. V. (2012). *Introdução a Interface GUI no Java*. Obtido de Devmedia: https://www.devmedia.com.br/introducao-a-interface-gui-no-java/25646

[34]Paul, J. (5 de julho de 2016). *10 Examples to read a text file in Java*. Obtido de Javarevisited: https://javarevisited.blogspot.com/2016/07/10-examples-to-read-text-file-in-java.html

[35]Piekarski, A. E., & Quináia, M. A. (3 de maio de 2000). *Reengenharia de software: o que, por quê e como*. Obtido de academia: https://www.academia.edu/10710292/Reengenharia_de_software_o_que_por_qu%C3%AA_e_como

[36]Pinto, P. (9 de Fevereiro de 2019). *Sabe o que é uma API (Application Programming Interface)?* Obtido de Pplware: https://pplware.sapo.pt/high-tech/sabe-o-que-e-uma-api-application-programming-interface/

[37]Poyias, A. (28 de Janeiro de 2019). *Design Patterns — A quick guide to Builder pattern.* Obtido de Medium: https://medium.com/@andreaspoyias/design-patterns-a-quick-guide-to-builder-pattern-a834d7cacead

[38]Predrag. (4 de novembro de 2018). *An MVC Example with Servlets and JSP*. Obtido de Baeldung: https://www.baeldung.com/mvc-servlet-jsp

[39]PROGRAMMING, & Sridhar, J. (21 de dezembro de 2017). *How to Read and Write XML Files With Code*. Obtido de makeuseof: https://www.makeuseof.com/tag/read-write-xml-file-code/

[40]Research, T., & Robie, J. (s.d.). *What is the Document Object Model?* Obtido de W3: https://www.w3.org/TR/WD-DOM/introduction.html

[41]Richardson, C. (s.d.). *Pattern: Monolithic Architecture*. Obtido de Microservice Architecture: https://microservices.io/patterns/monolithic.html

[42]Saini, S. (12 de fevereiro de 2018). *How To Read XML File In JAVA*. Obtido de oodles Technologies: https://www.oodlestechnologies.com/blogs/How-To-Read-XML-File-In-JAVA

[43]Sharma, A. (16 de Junho de 2016). *JasperReports – Open Source Reporting Tool*. Obtido de DZone: https://dzone.com/articles/jasper-reports-open-source-reporting-tool

[44]Silva, F. R. (2010). *Introdução: Design Pattern*. Obtido de Devmedia: https://www.devmedia.com.br/introducao-design-pattern/18838

[45]Staveley, A. (28 de Dezembro de 2011). *The "4+1" View Model of Software Architecture*. Obtido de Dzone: https://dzone.com/articles/%E2%80%9C41%E2%80%9D-view-model-software

[46]Teles, J. (s.d.). *Manutenibilidade: O que é e como ela pode te ajudar.* Obtido de Engeteles: https://engeteles.com.br/manutenibilidade/

[47]TEREK, H. (5 de julho de 2017). *Configure tomcat memory usage*. Obtido de Programmer Gate: https://www.programmergate.com/tomcat-memory-usage/

[48]*Test-driven development*. (s.d.). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Test-driven_development

[49]*Teste de sistema*. (s.d.). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Teste_de_sistema

[50]*The Apache Tomcat Connectors - AJP Protocol Reference*. (s.d.). Obtido de Apache Tomcat: https://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html

[51]Toumi, A. (11 de setembro de 2012). *Application Architecture [tuto 1 J2EE]*. Obtido de Best Practices Software engineering: http://abdennour-insat.blogspot.com/2012/09/application-architecture-tuto-1-j2ee.html

[52]Tyson, M. (29 de Janeiro de 2019). *WHAT IS JAVA?* Obtido de JavaWorld: https://www.javaworld.com/article/3336161/what-is-jsp-introduction-to-javaserver-pages.html

[53]Umesh. (21 de fevereiro de 2018). *How to Read Large File in Java*. Obtido de JavadevJournal: https://www.javadevjournal.com/java/how-to-read-large-file-in-java/

[54]*Value Analysis and Function Analysis System Technique*. (s.d.). Obtido de NPD Solutions: http://www.npd-solutions.com/va.html

[55]Ventura, J. A. (2014). *Teste de integração na prática*. Obtido de DevMedia: https://www.devmedia.com.br/teste-de-integracao-na-pratica/31877

[56]Vogel, L. (22 de fevereiro de 2018). *Java and XML - Tutorial*. Obtido de vogella: http://www.vogella.com/tutorials/JavaXML/article.html

[57]*What is BIRT?* (s.d.). Obtido de Eclipse: https://www.eclipse.org/birt/about/

[58]Woodruff, R. B. (março de 1997). *Customer value: The next source for competitive advantage*. Obtido de Springer Link: https://link.springer.com/article/10.1007%2FBF02894350