**Relatório de Projecto**

**Programação |**

**Avaliação Periódica 1**

**Autor(es):**

Mariana Pais Jorge Magalhães a2022147454

**Data:** 01/2024

Resumo

Este projeto consiste na implementação de uma aplicação em Java que adota o paradigma de Programação Orientada a Objetos (POO). O objetivo principal é auxiliar na gestão de uma loja de informática, permitindo o registo, autenticação e gestão de utilizadores e encomendas de hardware. A aplicação suporta três tipos de utilizadores (administradores, técnicos e clientes) e possibilita funcionalidades como a gestão de utilizadores, serviços, equipamentos e fornecedores, além de manter o histórico de ações realizadas. Para garantir a persistência de dados, são utilizados ficheiros de texto e de objetos.

A metodologia empregada baseou-se no desenvolvimento iterativo, priorizando requisitos funcionais e não funcionais, incluindo a validação de dados e a monitorização de acessos. Entre os resultados, destaca-se a implementação de funcionalidades completas para gestão de serviços e equipamentos, a integração de log de acessos e a utilização de persistência de dados. Conclui-se que a aplicação alcançou os objetivos propostos, apresentando uma solução robusta para o problema de gestão descrito.

**Palavras-chave**

Introduzir palavras-chave ou frases em ordem alfabética, separadas por vírgula.

**Índice**

[Resumo ii](#_Toc187775967)

[Lista de Figuras iv](#_Toc187775968)

[Lista de Tabelas v](#_Toc187775969)

[Lista de Acrónimos vi](#_Toc187775970)

[1. Introdução 7](#_Toc187775971)

[2. Objectivos e Metodologias 8](#_Toc187775972)

[2.1. Ferramentas e Tecnologias 9](#_Toc187775973)

[2.2. Planeamento 9](#_Toc187775974)

[3. Trabalho Desenvolvido 10](#_Toc187775975)

[3.1. Requisitos Implementados 10](#_Toc187775976)

[3.2. Classes e *Packages* 10](#_Toc187775977)

[3.3. Algoritmos 10](#_Toc187775978)

[3.4. Estruturas de Dados 10](#_Toc187775979)

[3.5. Armazenamento de Dados 11](#_Toc187775980)

[3.6. Procedimentos de Teste 11](#_Toc187775981)

[4. Conclusões 12](#_Toc187775982)

[4.1. Forças 12](#_Toc187775983)

[4.2. Limitações 12](#_Toc187775984)

[4.3. Trabalho Futuro 12](#_Toc187775985)

[5. Referências 13](#_Toc187775986)

[5.1. Lista de Referências 13](#_Toc187775987)

[5.2. Como Integrar Referências 19](#_Toc187775988)

[6. Anexos 20](#_Toc187775989)

# Lista de Figuras

[Figura 1 – Planeamento 10](#_Toc187786203)

# Lista de Tabelas

[Tabela 1‑1 *– Características dos planetas (adaptado/reproduzido de* (Jet Propulsion Laboratory, 2020)*).* 3](#_Toc62047341)

Lista de Acrónimos

*[Lista ordenada alfabeticamente]*

**OEM** Original Equipment Manufacturer

**POO** Programação Orientada a Objetos

**NIF** Número de Identificação Fiscal

# Introdução

A crescente necessidade de gestão eficiente em pequenas e médias empresas, particularmente no sector da informática, evidencia a importância de soluções tecnológicas que automatizem tarefas, rotinas e otimizem processos. Este trabalho tem como principal motivação o desenvolvimento de uma aplicação que auxilia na gestão de uma loja de informática, abordando a organização de encomendas, equipamentos e serviços associados.

O problema identificado consiste na ausência de um sistema centralizado que permita gerir utilizadores, encomendas e inventário de forma integrada, reduzindo erros e aumentando a produtividade. Para resolver este problema, foi adotada uma abordagem que alia os princípios da Programação Orientada a Objetos (POO) à implementação de estruturas de armazenamento de dados persistentes, garantindo robustez e escalabilidade à solução.

A aplicação proposta foi desenvolvida em linguagem Java e apresenta funcionalidades como o registo e autenticação de utilizadores, a gestão de equipamentos e fornecedores, bem como o acompanhamento de encomendas e serviços. Adicionalmente, foi integrada a monitorização de acessos e a validação rigorosa de dados, assegurando uma experiência segura e eficiente.

Este documento está estruturado da seguinte forma: na próxima secção, são apresentados os objetivos gerais e a metodologia adotada para o desenvolvimento do projeto, incluindo ferramentas e tecnologias utilizadas. A secção de trabalho desenvolvido descreve detalhadamente as funcionalidades implementadas e os resultados alcançados. Por fim, na secção de conclusões, são evidenciadas as principais contribuições do projeto, as suas limitações e as propostas para trabalho futuro.

O projeto foi realizado com base em fontes de informação diversas, que incluem documentação técnica e recursos online, devidamente referenciados no capítulo de referências. Adicionalmente, diagramas técnicos e outros documentos complementares encontram-se disponíveis na secção de anexos.

# Objectivos e Metodologias

O principal objetivo deste trabalho é desenvolver uma aplicação completa em linguagem Java, com base no paradigma de Programação Orientada a Objetos (POO), para apoiar a gestão de uma loja de informática. A aplicação deve oferecer funcionalidades que facilitem o registo e autenticação de utilizadores, a gestão de equipamentos e fornecedores, a organização de encomendas, e o acompanhamento de serviços, assegurando uma utilização eficiente e segura. Além disso, deve ser garantida a persistência de dados e a monitorização das ações realizadas pelos utilizadores.

Os objetivos específicos incluem:

1. Criar um sistema robusto de autenticação e autorização, adaptado a três tipos de utilizadores (administradores, técnicos e clientes).
2. Implementar funcionalidades para a gestão de inventário, incluindo categorias, fornecedores e equipamentos.
3. Desenvolver um módulo de acompanhamento de serviços, permitindo a criação, aprovação, atribuição e conclusão de tarefas associadas a encomendas.
4. Garantir a validação de dados e a integridade do sistema, com registos de atividades e persistência de dados por meio de ficheiros.
5. Fornecer um ambiente de utilização intuitivo e eficaz, utilizando uma interface baseada em texto.

*Para alcançar os objetivos estabelecidos, seguiu uma abordagem metodológica iterativa e incremental, com características próximas do modelo de desenvolvimento em cascata, adaptado às necessidades específicas do projeto. Etapas principais de trabalho:*

1. *Análise de Requisitos*

* *Estudo detalhado dos requisitos funcionais definidos no documento de especificação*
* *Identificação das principais funcionalidades a implementar*
* *Priorização dos requisitos com base na complexidade e importância*

1. *Implementação Faseada*

* *Desenvolvimento incremental, focando inicialmente nos requisitos fundamentais*
* *Implementação progressiva dos módulos de utilizadores, autenticação e gestão de equipamentos*

1. *Estratégia de Implementação*

* *Gestão de persistência mediante ficheiros de objetos e texto*
* *Implementação de mecanismos de serialização para armazenamento de dados*

1. *Desenho da Arquitetura*

* *Definição da estrutura de classes orientada a objetos*
* *Planeamento da persistência de dados*
* *Desenho dos mecanismos de gestão de utilizadores e autenticação*

*Principais desafios metodológicos:*

* *Garantir a modularidade do código*
* *Implementar controlos de acesso e autenticação robustos*
* *Gerir a persistência de dados de forma eficiente*
* *Validar e tratar adequadamente as entradas do utilizador*

## Ferramentas e Tecnologias

*Para o ambiente de desenvolvimento integrado (IDE) foi utilizado o visual studio code, com suporte extensivo para Java através de extensões, integração com Git e depuração integrada.*

*Em termos de linguagem de programação utilizei a versão mais recente do java, 23.0.1.*

*Para ter uma melhor gestão de código utilizei o git, por ter um controlo de versões distribuído, facilita na colaboração e rastreamento de alterações, tem a possibilidade de ramificação e experimentação de código e por ter integração com plataformas de repositório como GitHub/GitLab, para o caso a plataforma utilizada foi o github.*

*Foi utilizado um ficheiro de markdown para a documentação do requisito, neste caso o markdown foi por uma questão de ter uma sintaxe simples e legível, facilidade de conversão para múltiplos formatos e integração com ferramentas de controlo de versões.*

*A escolha destas ferramentas, teve a ver com uma combinação que maximiza a eficiência de desenvolvimento, a qualidade do código e a facilidade de manutenção, alinhadas com os requisitos específicos do projeto de gestão de loja de informática.*

## Planeamento

*O planeamento do projecto foi estruturado para garantir uma gestão eficiente do tempo e dos recursos disponíveis, tendo como objectivo principal cumprir os requisitos estabelecidos e entregar uma solução funcional dentro do prazo. O trabalho foi desenvolvido de forma individual e dividido em etapas sequenciais e iterativas, permitindo uma validação contínua dos resultados obtidos em cada fase.*

O cronograma abaixo apresenta uma visão geral da distribuição das tarefas ao longo do tempo

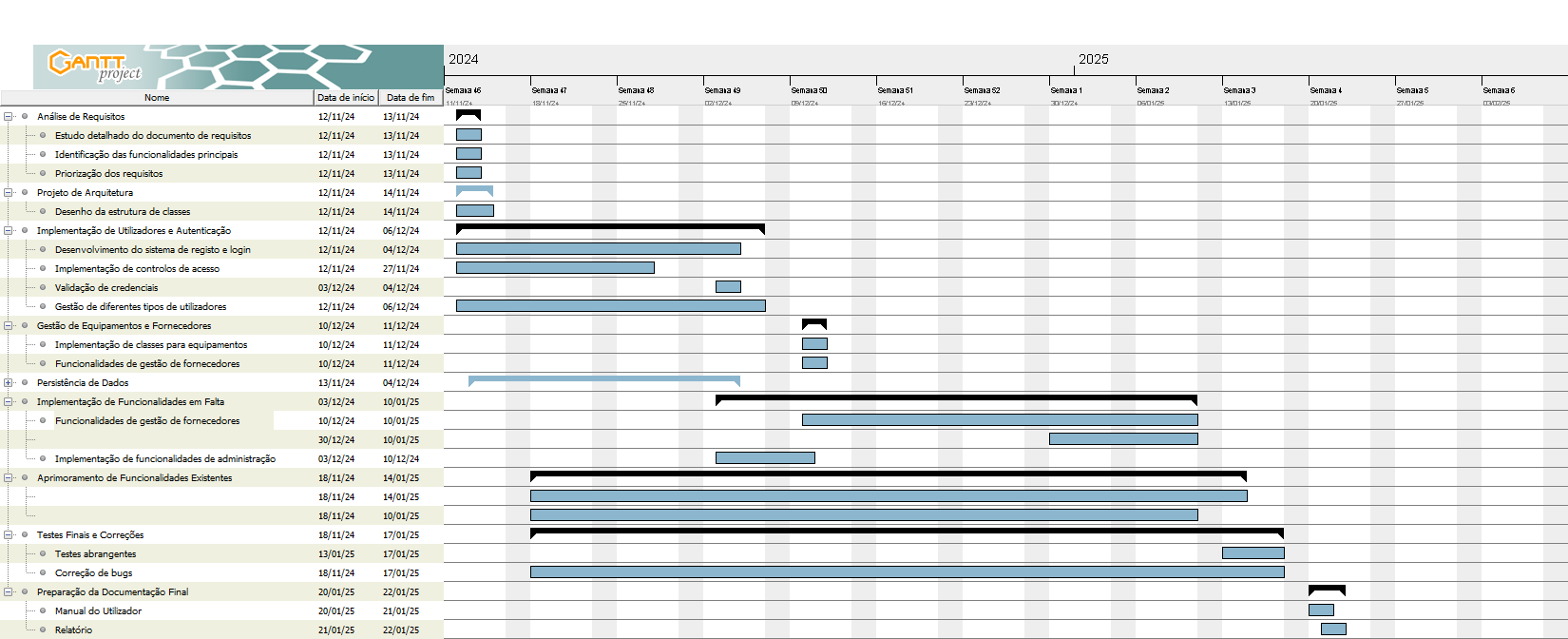


Figura – Planeamento

É importante salientar que algumas tarefas apresentaram uma duração maior no cronograma, não devido à sua complexidade, mas porque o trabalho no projeto não foi realizado de forma contínua. Por exemplo, a fase de listagens aparenta ter demorado um mês, mas o tempo efetivo de trabalho foi significativamente menor, uma vez que o desenvolvimento foi realizado conforme a necessidade, após a conclusão de etapas prévias como o registo e a autenticação de utilizadores.

Este modelo de trabalho permitiu avançar com listagens à medida que outras funcionalidades eram finalizadas, otimizando o fluxo de desenvolvimento. Da mesma forma, o desenvolvimento de funcionalidades relacionadas aos equipamentos e serviços foi realizado de forma eficiente, com cada etapa ocupando apenas alguns dias de trabalho efetivo. Apesar disso, a organização no cronograma reflete um período mais extenso para acomodar possíveis ajustes e validações, assegurando a qualidade do resultado.

# Trabalho Desenvolvido

## Requisitos Implementados

*A seguinte tabela mostra todos os requisitos, indicando o seu estado (implementado ou não implementado) e o tempo dedicado a cada tarefa.*

|  |  |  |  |
| --- | --- | --- | --- |
| Código do Requisito | Descrição do Requisito | Estado | Tempo Dedicado/Estimado (horas) |
| R1 | Permitir os utilizadores registarem-se e autenticarem-se na aplicação. | Implementado | 10 |
| R2 | Permitir o acesso à aplicação por 3 tipos de utilizadores: administradores, técnicos e clientes. | Implementado | 8 |
| R3 | Os utilizadores são caracterizados pelos atributos login, password, nome, estado (ativo/inativo), email e tipo (administradores, técnicos ou clientes). | Implementado | 12 |
| R4 | Cada utilizador apenas pode alterar a sua própria informação, não podendo alterar ou visualizar dados de outros utilizadores, ou criar utilizadores. | Implementado | 6 |
| R5 | O login e email devem ser únicos. | Implementado | 4 |
| R6 | Os clientes e técnicos caracterizam-se adicionalmente por número de identificação fiscal (NIF), morada e contacto telefónico. | Implementado | 8 |
| R7 | Caso não existam utilizadores criados, a aplicação deve solicitar a criação de um utilizador (administrador). | Implementado | 1 |
| R8 | Após a autenticação, a aplicação deve apresentar a mensagem “Bem-vindo [nome utilizador]”. | Implementado | 0.25 |
| R9 | Quando a aplicação estiver a encerrar, deve apresentar a mensagem “Adeus [nome utilizador]”. | Implementado | 0.25 |
| R10 | Os administradores aprovam os pedidos de registo dos utilizadores. Todos os pedidos devem ser aprovados antes de poderem ser usados para autenticação. | Implementado | 6 |
| R11 | Os administradores aprovam e encerram os pedidos de serviço (encomenda), bem como associam um técnico responsável a cada serviço. | Implementado | 3 |
| R12 | Os técnicos introduzem equipamentos, categorias e fornecedores. | Implementado | 16 |
| R13 | Os clientes realizam pedidos de compras de equipamentos. | Implementado | 10 |
| R14 | Os administradores podem gerir todas as vendas realizadas. | Implementado | 1 |
| R15 | Os técnicos apenas podem gerir as suas vendas. | Implementado | 1 |
| R16 | Os clientes apenas podem gerir as suas compras | Implementado | 1 |
| R17 | O serviço de compra inicia com o cliente a solicitar uma encomenda, indicando os produtos pretendidos. | Não Implementado | 10 |
| R18 | O administrador aprova o serviço e associa um técnico responsável. | Implementado | 5 |
| R19 | Após ser designado para um serviço, o técnico responsável regista a informação do processo, nomeadamente pode criar sub-tarefas e associar outros técnicos adicionais às sub-tarefas, e gere o processo até à sua conclusão. | Implementado | 10 |
| R20 | Um serviço de compra (encomenda), além de ter associado um técnico responsável, inclui a listagem dos produtos incluídos (equipamentos de hardware, podendo definir a quantidade), uma data de realização do pedido, uma descrição, o estado e o tempo que demorou a ser processado o valor total da venda. | Implementado | 0.50 (1/2) |
| R21 | Adicionalmente, cada serviço possui um código identificador único e sequencial. | Implementado | 0.25 (1/4) |
| R22 | O estado do serviço apresenta um dos seguintes valores: a) se o serviço está *submetido*; b) se a serviço foi *aceite*; c) ou se já foi *concluído*. | Implementado | 1 |
| R23 | Quando um técnico conclui um serviço deve indicar o tempo despendido no processamento do pedido. O valor total da venda deve ser calculado automaticamente. | Implementado | 3 |
| R24 | A aplicação deve permitir introduzir e gerir equipamentos de hardware. | Implementado | 10 |
| R25 | O código de um equipamento deve ser único. | Implementado | 4 |
| R26 | Cada equipamento deve ter associado uma lista até 6 fornecedores, podendo ser introduzido um novo fornecedor a qualquer momento. | Implementado | 6 |
| R27 | Cada fornecedor é caracterizado por nome, morada e contacto telefónico. | Implementado | 4 |
| R28 | Cada equipamento deve apresentar uma lista até 4 categorias, podendo ser introduzida uma nova categoria a qualquer momento. | Implementado | 6 |
| R29 | Cada categoria é caracterizada pela sua designação e família (atributos alfanuméricos). | Implementado | 4 |
| R30 | Cada série possui uma geração e sequência (atributos numéricos). | Implementado | 4 |
| R31 | Cada versão é caracterizada pela unidade, valor alfa e valor beta (atributos numéricos). | Implementado | 4 |
| R32 | É possível alterar a quantidade em stock de um equipamento. | Implementado | 2 |
| R33 | Sempre que um equipamento é vendido, a quantidade em stock deve ser atualizada. | Implementado | 0.50 (1/2) |
| R36 | Deve ser possível ordenar utilizadores por ordem alfabética do nome. | Implementado | 0.75 |
| R37 | Deve ser possível ordenar equipamentos por designação. | Implementado | 0.75 |
| R39 | Deve ser possível listar todos os utilizadores. | Implementado | 0.50 (1/2) |
| R40 | Deve ser possível listar utilizadores por tipo. | Implementado | 0.75 |
| R41 | Deve ser possível listar todos os serviços. | Implementado | 0.75 |
| R42 | Deve ser possível listar todos os serviços associados a um cliente. | Implementado | 0.75 |
| R43 | Deve ser possível listar todos os serviços por estado. | Implementado | 0.75 |
| R44 | Deve ser possível listar serviços com tempo despendido superior a um determinado limite (introduzido pelo utilizador no momento de pesquisa). | Implementado | 0.75 |
| R45 | Deve ser possível listar todos os equipamentos. | Não Implementado | -- |
| R46 | Deve ser possível listar equipamentos que sejam (ou não) OEM. | Não Implementado | -- |
| R47 | Deve ser possível listar equipamentos com uma quantidade de stock abaixo de um determinado limite (introduzido pelo utilizador no momento de pesquisa). | Não Implementado | -- |
| R49 | Deve ser possível pesquisar utilizadores por login ou nome. | Não Implementado | -- |
| R50 | Deve ser possível pesquisar serviços por código ou uma palavra que surja na descrição. | Não Implementado | -- |
| R51 | Deve ser possível pesquisar equipamentos por marca ou código. | Não Implementado | -- |
| R52 | Deve ser possível pesquisar equipamentos que possuem uma determinada categoria. | Não Implementado | -- |
| R53 | Deve ser possível realizar pesquisas avançadas, ou seja, apresentar todos os registos que apresentem um termo de pesquisa, mesmo que parcialmente. | Não Implementado | -- |
| R54 | Os clientes podem listar e pesquisar os seus serviços que realizaram. | Não Implementado | -- |
| R55 | Os técnicos podem listar e pesquisar os seus serviços que processaram. | Não Implementado | -- |
| R56 | Os administradores podem listar e pesquisar todos os serviços. | Não Implementado | -- |
| R57 | O acesso à aplicação deve ser restringido com credenciais (login/password), informação que deverá ser armazenada num ficheiro de texto “credenciais\_acesso.txt”. | Implementado | 8 |
| R58 | Durante o encerramento da aplicação, os dados devem ser automaticamente guardados num ficheiro de objetos “dados\_apl.dat”. | Implementado | 6 |
| R59 | A aplicação, no arranque, deve automaticamente ler os dados do ficheiro de objectos “dados\_apl.dat”, caso este exista, e informar o utilizador que os dados foram lidos com sucesso. | Implementado | 10 |
| R61 | Disponibilizar uma interface em modo texto onde o utilizador possa interagir e controlar a aplicação. | Não Implementado | -- |
| R62 | Todas as acções dos utilizadores deverão ser guardadas num ficheiro de texto denominado “log.txt”. Este ficheiro deverá ser escrito de forma sequencial, apresentando as acções mais recentes no início do ficheiro. | Não Implementado | -- |
| R63 | Deve ser possível consultar o log de acções através da aplicação. | Não Implementado | -- |
| R64 | Deverá ser registado o número total (até ao momento) de execuções do sistema e o username do último utilizador que acedeu à aplicação. Esta informação deverá ser guardada num ficheiro de objectos e actualizada sempre que a aplicação é executada, apresentando os dados ao utilizador. | Não Implementado | -- |
| R65 | A aplicação deve estar corretamente estruturada, tendo em conta o paradigma Orientado a Objetos e recorrendo à linguagem Java. | Não Implementado | - |
| R66 | Implemente as estruturas de armazenamento necessárias, procurando otimizar os recursos utilizados. | Não Implementado | - |
| R67 | Valide todas as leituras de dados do utilizador (e.g. verifique se os nomes são únicos). | Não Implementado | - |
| R68 | Sempre que necessário, apresentar ao utilizador mensagens informativas adequadas. Quando um utilizador realizar uma ação sobre a aplicação, esta deve informar se a ação foi realizada com sucesso ou insucesso. | Não Implementado | - |

Tabela - Requesitos implementados

## Classes e *Packages*

*Apresentar e descrever todas as packages e classes do projecto. Os principais métodos de cada classe (e que não sejam óbvios) devem ser igualmente explicados.*

*Recomenda-se que sejam disponibilizados* ***diagramas de classes*** *(ou outros diagramas UML, sempre que se justifique) para melhor compreensão da estrutura do projecto.*

## Algoritmos

*Descrever com brevidade os principais algoritmos, essenciais para o funcionamento do projecto.*

## Estruturas de Dados

*Descrever com brevidade as estruturas de dados usadas para manipular dados no projecto (e.g. Vectors, ArrayLists, LinkedLists, Hashtables, Maps, Binary Trees), justificando as opções.*

## Armazenamento de Dados

*Descrever com brevidade os métodos de armazenamento de dados usados para garantir persistência de dados.*

*Caso o projecto manipule uma base de dados deve ser explicado o* ***Diagrama de Entidade-Relacionamento (ER)*** *e o* ***Diagrama Físico****.*

*Deve ser descrita e explicada a estrutura em árvore do sistema de ficheiros e pastas necessários ao funcionamento do projecto.*

## Procedimentos de Teste

*Descrever com brevidade os procedimentos de teste adoptados para garantir a funcionalidade e qualidade do código.*

# Conclusões

*O capítulo final deve descrever sumariamente o trabalho desenvolvido, indicando as forças e limitações. Deve-se resumir detalhadamente o objectivo do trabalho e as tarefas concretizadas.*

*É importante descrever as forças do trabalho, ou seja, as metas concretizadas com sucesso, bem como as limitações ou fraquezas do mesmo, indicando o que poderia ou deveria ser desenvolvido de outra forma.*

*Enquanto o documento deve ser técnico, este capítulo permite que sejam realizadas anotações mais pessoais. Assim, enquanto o restante texto deve ser escrito na terceira pessoa, neste capítulo é possível usar a primeira pessoa.*

## Forças

*Descrever as forças do trabalho, frisando o que o distingue das restantes abordagens e demonstrando como foi solucionado o problema inicial.*

## Limitações

*Descrever detalhadamente as limitações do trabalho, ou seja, de que forma não é possível solucionar partes do problema.*

## Trabalho Futuro

*Descrever as melhorias futuras, apresentando uma proposta para os próximos passos a desenvolver para atingir uma solução mais completa ou mais robusta, ou como expandir e diversificar a solução.*

# Referências

## Lista de Referências

Works Cited

Alura. “Como Converter String Para Date Em Java.” *Alura*, 4 Jan. 2017, www.alura.com.br/artigos/como-converter-string-para-date-em-java. Accessed 14 Jan. 2025.

---. “Regex Em Java: Validando Dados Com Expressões Regulares.” *Alura*, 21 Mar. 2016, www.alura.com.br/artigos/validando-dados-com-expressao-regular-em-java?srsltid=AfmBOooBzHoJHM3mpgda8W\_C\_ETDlPMOC7QJuYETUfIn2STxRvwxo7u1. Accessed 11 Dec. 2024.

baeldung. “Baeldung.” *Baeldung*, 29 Nov. 2022, www.baeldung.com/map-ofentries-and-map-of. Accessed 10 Dec. 2024.

Ballem. “Java 8! Filtrando Dados Com Stream.” *MBallem | Programando Com Java*, 3 Aug. 2015, mballem.com/post/java-8-filtrando-dados-com-stream/. Accessed 9 Dec. 2024.

Batra, Yatin. “Java Stream Contains, ContainsAny and ContainsAll Examples.” *Java Code Geeks*, 24 June 2024, www.javacodegeeks.com/java-stream-contains-containsany-and-containsall-examples.html#google\_vignette. Accessed 18 Dec. 2024.

“BigDecimal Class in Java.” *GeeksforGeeks*, 27 Jan. 2018, www.geeksforgeeks.org/bigdecimal-class-java/. Accessed 9 Dec. 2024.

“Como Você Escolhe Entre Fluxo Java E Coleta Para Processamento de Dados?” *Linkedin.com*, 2023, pt.linkedin.com/advice/1/how-do-you-choose-between-java-stream-collection-data-processing?lang=pt. Accessed 9 Dec. 2024.

daniel\_silva. “Ordenar Por Data Em ArrayList.” *Portugal-a-Programar*, 15 Apr. 2020, www.portugal-a-programar.pt/forums/topic/78216-ordenar-por-data-em-arraylist/. Accessed 14 Jan. 2025.

de. “O SerialVersionUID de Tem Que Ser O Mesmo Em Todas as Classes “Serializáveis”?” *Stack Overflow Em Português*, 6 Sept. 2019, pt.stackoverflow.com/questions/408183/o-serialversionuid-de-tem-que-ser-o-mesmo-em-todas-as-classes-serializ%C3%A1veis. Accessed 9 Dec. 2024.

Divyesh Dabhi. “Baeldung.” *Baeldung*, 29 Jan. 2021, www.baeldung.com/java-map-entry. Accessed 10 Dec. 2024.

“Entendendo Serialização de Objetos E O SerialVersionUID | Oracle Brasil.” *Oracle.com*, 2019, www.oracle.com/br/technical-resources/articles/java/serialversionuid.html. Accessed 9 Dec. 2024.

GeeksforGeeks. “How to Read and Write Files Using the New I/O (NIO.2) API in Java?” *GeeksforGeeks*, 18 Mar. 2024, www.geeksforgeeks.org/read-and-write-files-using-the-new-i-o-nio-2-api-in-java/. Accessed 9 Dec. 2024.

---. “Java Streams Counting() Method with Examples.” *GeeksforGeeks*, 8 Mar. 2018, www.geeksforgeeks.org/java-util-stream-collectors-counting-method-examples/. Accessed 9 Dec. 2024.

---. “Java.time.Duration Class in Java.” *GeeksforGeeks*, 6 Mar. 2021, www.geeksforgeeks.org/java-time-duration-class-in-java/. Accessed 9 Dec. 2024.

---. “Stream NoneMatch() Method in Java with Examples.” *GeeksforGeeks*, 8 Mar. 2018, www.geeksforgeeks.org/stream-nonematch-method-java-examples/. Accessed 9 Dec. 2024.

“GraalVM Documentation.” *Graalvm.org*, 2024, www.graalvm.org/21.3/docs/introduction/. Accessed 9 Dec. 2024.

Gupta, Lokesh. “Java Stream Count() Matches with Filter().” *HowToDoInJava*, 4 Sept. 2017, howtodoinjava.com/java8/stream-count-elements-example/. Accessed 9 Dec. 2024.

How. “How to Filter Element from a List Using Java 8 Streams?” *Stack Overflow*, 15 Apr. 2022, stackoverflow.com/questions/71888924/how-to-filter-element-from-a-list-using-java-8-streams. Accessed 9 Dec. 2024.

“Java List Size() Method with Examples - Javatpoint.” *Www.javatpoint.com*, 2021, www.javatpoint.com/java-list-size-method. Accessed 9 Dec. 2024.

“Java List.contains(Object with Field Value Equal to X).” *W3docs.com*, 2024, www.w3docs.com/snippets/java/java-list-contains-object-with-field-value-equal-to-x.html. Accessed 18 Dec. 2024.

“Java® Platform, Standard Edition & Java Development Kit Version 21 API Specification.” *Docs.oracle.com*, 2019, docs.oracle.com/en/java/javase/21/docs/api/index.html. Accessed 9 Dec. 2024.

key ). “Java.util.Map, Melhor Implementação Considerando Apenas Get(Object Key).” *Stack Overflow Em Português*, 18 Aug. 2015, pt.stackoverflow.com/questions/81365/java-util-map-melhor-implementa%C3%A7%C3%A3o-considerando-apenas-getobject-key. Accessed 10 Dec. 2024.

“List Contains() Method in Java with Examples.” *GeeksforGeeks*, 11 Dec. 2018, www.geeksforgeeks.org/list-contains-method-in-java-with-examples/. Accessed 18 Dec. 2024.

“List Size() Method in Java with Examples.” *GeeksforGeeks*, 11 Dec. 2018, www.geeksforgeeks.org/list-size-method-in-java-with-examples/. Accessed 9 Dec. 2024.

Marcos. “Usando Streams No Java - Marcos - Medium.” *Medium*, 4 Sept. 2021, mmarcosab.medium.com/usando-streams-no-java-b85d98d9cf17. Accessed 9 Dec. 2024.

Martins, Rodrigo. “Understanding the Difference between Enum and Stream in Elixir.” *Medium*, 2 Oct. 2023, medium.com/@rrmartins/understanding-the-difference-between-enum-and-stream-in-elixir-ebd8615e0879. Accessed 9 Dec. 2024.

“Overview.” *Graalvm.org*, 2018, www.graalvm.org/jdk21/docs/introduction/. Accessed 9 Dec. 2024.

piotr.minkowski. “Interesting Facts about Java Streams and Collections - Piotr’s TechBlog.” *Piotr’s TechBlog*, 25 Apr. 2024, piotrminkowski.com/2024/04/25/interesting-facts-about-java-streams-and-collections/. Accessed 9 Dec. 2024.

“Processamento de Dados Com Streams Do Java SE 8 - Parte 1 | Oracle Brasil.” *Oracle.com*, 2020, www.oracle.com/br/technical-resources/articles/java/processing-streams-java-se-8.html. Accessed 9 Dec. 2024.

Ramesh Fadatare. “Java Stream Count() Method with Examples - Ramesh Fadatare - Medium.” *Medium*, 27 Sept. 2024, rameshfadatare.medium.com/java-stream-count-method-with-examples-7e65a8edadc0. Accessed 9 Dec. 2024.

“Reading and Writing Text Files - Dev.java.” *Dev.java: The Destination for Java Developers*, 2023, dev.java/learn/java-io/reading-writing/buffered-text/. Accessed 9 Dec. 2024.

“Reading, Writing, and Creating Files (the JavaTM Tutorials >   
 Essential Classes > Basic I/O).” *Oracle.com*, 2019, docs.oracle.com/javase/tutorial/essential/io/file.html. Accessed 9 Dec. 2024.

Rinaldo. “Java 8 Streams: Pare de Usar “For” E Simplifique Seu Código! - @Rinaldodev.” *@Rinaldodev*, 30 Mar. 2019, rinaldo.dev/java-8-streams-pare-de-usar-for-e-simplifique-seu-codigo/. Accessed 9 Dec. 2024.

Sayyed, Husain. “ToList() vs. Collectors.toList(): What’s the Difference in Java?” *DEV Community*, 25 Nov. 2024, dev.to/syedrasique718/tolist-vs-collectorstolist-whats-the-difference-in-java-ppi. Accessed 17 Dec. 2024.

Servian, Alexandre. “Regex: Um Guia Pratico Para Expressões Regulares.” *Comunidade XP*, 31 Jan. 2020, medium.com/xp-inc/regex-um-guia-pratico-para-express%C3%B5es-regulares-1ac5fa4dd39f. Accessed 11 Dec. 2024.

“Stream (Java Platform SE 8 ).” *Docs.oracle.com*, docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html. Accessed 9 Dec. 2024.

“Stream Count() Method in Java with Examples.” *GeeksforGeeks*, 5 Mar. 2018, www.geeksforgeeks.org/stream-count-method-java/. Accessed 9 Dec. 2024.

“Stream Filter() in Java with Examples.” *GeeksforGeeks*, 7 Mar. 2018, www.geeksforgeeks.org/stream-filter-java-examples/. Accessed 9 Dec. 2024.

Tea. “What Is the Difference between Streams and Collections in Java 8.” *Stack Overflow*, 11 Sept. 2016, stackoverflow.com/questions/39432699/what-is-the-difference-between-streams-and-collections-in-java-8/39432937. Accessed 9 Dec. 2024.

um, Validar. “Validar Um Texto Com Regex.” *Stack Overflow Em Português*, 26 Mar. 2018, pt.stackoverflow.com/questions/286583/validar-um-texto-com-regex. Accessed 11 Dec. 2024.

Vusal Guliyev. “Writing Files with NIO and IO in Java - Vusal Guliyev - Medium.” *Medium*, 9 Feb. 2024, medium.com/@vusal.guliyev.313/writing-files-with-nio-and-io-in-java-bc60b06a413a. Accessed 9 Dec. 2024.

Yadav, Himanshu. “Java List Size().” *Scaler Topics*, 25 May 2022, www.scaler.com/topics/java-list-size/. Accessed 9 Dec. 2024.

Зепп Бранниган. “Expressões Regulares Em Java (RegEx).” *JavaRush*, 28 Feb. 2021, javarush.com/pt/groups/posts/pt.2013.expresses-regulares-em-java-regex. Accessed 11 Dec. 2024.

## Como Integrar Referências

*Enquanto esta secção lista todas as fontes, no texto do documento estas fontes devem ser referenciadas sempre que algum elemento seja baseado ou replicado de uma fonte externa, de acordo com o formato de referência adoptado (e.g. APA: (Author et al., 2020), IEEE: [1] ).*

*Por exemplo, considere-se a seguinte passagem, onde é realizada uma citação explicita (cópia):*

a abstracção de controlo “*implementa uma forma simples de reutilização de código, ao permitir que um bloco de instruções seja escrito independentemente do programa principal*” **(Martins, 2010)**.

*É realizada uma citação explicita, ou seja, parte do texto é integralmente copiado da fonte, sendo formatado em itálico e referenciado.*

*No exemplo seguinte não é realizada uma citação explícita. O texto é escrito pelo autor mas baseado numa fonte externa:*

Em programação orientada a objectos, um objecto caracteriza-se por uma identidade única, um conjunto de atributos que o caracterizam e um conjunto de acções ou métodos **(Martins, 2010)**.

*O autor usou uma fonte, interpretou e apresentou um texto próprio baseado nessa fonte. Não realizou cópia integral, pelo que não é necessário a sua formatação em itálico, mas uma vez que o conhecimento é baseado num outro autor, continua a ser necessário citar a fonte.*

*Na secção final do documento, Referências, deve então surgir a lista de fontes que suportaram o texto. No exemplo seguinte é usado o formato APA:*

Referências

Martins, F., 2010. “Java 5 e Programação por Objectos”, Editora Lidel Edições Técnicas.

Oracle, 2016. “Java Collections”, disponível em <https://docs.oracle.com/javase/tutorial/collections/index.html>, acedido em: Fevereiro 2016, última actualização: 2015.

# Anexos

*Elementos técnicos devem ser colocados como anexo. A especificação técnica (diagramas conceptual e físico, código de criação da base de dados, diagramas de classe), protótipos, especificação de testes, manuais de utilizadores e outros documentos técnicos podem ser disponibilizados como anexo ao documento.*

*O documento principal descreve os objectivos, metodologia e resultados do trabalho. As componentes técnicas não devem ser abordadas no documento principal, mas disponibilizadas como anexo.*