

Relatório de trabalho prático 1

Sistema de Registo de Auditorias V.2

Carlos Santos

Aluno nº 19432

Rúben silva

Aluno nº 19433

Trabalho realizado sob a orientação de:

Luís Ferreira

**Linguagens de Programação II**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, maio de 2020

Índice

[1 Implementação 1](#_Toc41051029)

[1.1 Descrição do Problema 1](#_Toc41051030)

[1.2 Solução e atualização 1](#_Toc41051031)

[2 Estado de Arte 2](#_Toc41051032)

[3 Bibliotecas 3](#_Toc41051033)

[3.1 a1\_TrabalhoPratico 3](#_Toc41051034)

[3.2 a2\_RegrasNegocio 4](#_Toc41051035)

[3.3 a3\_DadosClasses 4](#_Toc41051036)

[3.4 c1\_ObjetosNegocio 8](#_Toc41051037)

[3.5 c2\_Validacoes 12](#_Toc41051038)

[4 Conclusão 13](#_Toc41051039)

[Bibliografia 14](#_Toc41051040)

Lista de Figuras

Figura 1: a1\_TrabalhoPratico *variáveis............................................................................................3*

Figura 2: a1\_TrabalhoPratico *saves..................................................................................................3*

Figura 3: a1\_TrabalhoPratico *gravar................................................................................................3*

Figura 4: a2\_RegrasNegocio Função com Verificação ..................................................................... 4

Figura 5: a2\_DadosClasses *ExisteAuditoria(...)*.................................................................................4

Figura 6: a2\_DadosClasses *QuantidadeAuditorias(...).....................................................................5*

Figura 7: a2\_DadosClasses *AdicionaVulnerabilidade(...)..................................................................5*

Figura 8: a2\_DadosClasses *EditaAuditoria(...)..................................................................................5*

Figura 9: a2\_DadosClasses *ApresentaAuditoriasColaborador(...)....................................................6*

Figura 10: a2\_DadosClasses *ApresentaAuditoriasOrdenadasVuln(...).............................................6*

Figura 11: a2\_DadosClasses *GuardarAuditorias(...).........................................................................7*

Figura 12: a2\_DadosClasses *VerificarAtividade(...)..........................................................................7*

Figura 13: a2\_DadosClasses *TornarColaboradorInativo(...).............................................................7*

Figura 14: a2\_DadosClasses *EquipamentosCompleto......................................................................8*

Figura 15: a2\_DadosClasses *AuditoriaCompleta .............................................................................8*

Figura 16: c1\_ObjetosNegocio *Auditoria.........................................................................................9*

Figura 17: c1\_ObjetosNegocio *Colaborador.....................................................................................9*

Figura 18: c1\_ObjetosNegocio *Equipamento.................................................................................10*

Figura 19: c1\_ObjetosNegocio *Vulnerabilidades............................................................................10*

Figura 20: c1\_ObjetosNegocio *Pessoa...........................................................................................11*

Figura 21: c2\_Validacoes *TryCatchInt.............................................................................................12*

# Implementação

## Descrição do Problema

Necessidade de criar um programa que auxilie colaboradores de uma empresa na criação de auditorias internas. Sendo que este deve oferecer uma vasta quantidade de funcionalidades para o utilizador, tais como, a inserção e edição de colaboradores e vulnerabilidades, a remoção de colaboradores, a edição das vulnerabilidades de uma auditoria, entre outras. Para mostrar resultados e dados, o programa deve também dar a possibilidade de apresentar diferentes listas, como uma lista onde as auditorias estão ordenadas pela quantidade de vulnerabilidades ou então numa auditoria, estas vulnerabilidades estão agrupadas por nível de impacto. Também devem ser apresentados os detalhes de um colaborador mostrando as auditorias realizadas por este, e também mostrar as vulnerabilidades identificadas num equipamento.

Por fim, deve também ser apresentado uma tabela geral onde devem ser comparadas todas as auditorias mostrando a que tem mais e a que tem menos vulnerabilidades, a quantidade de auditorias e a média de vulnerabilidade em todas as auditorias registadas.

## Solução e atualização

Para solucionar o problema, na segunda fase da realização do programa, foi-lhe implementada a estrutura final de modo a corresponder ao estilo de programação por camadas Ntier, corrigidas e adaptadas as classes com modificações como adição de atributos e *enums,* foi também criada a interface de cada classe.

O programa conta com opções de adição, edição, remoção e apresentação de auditorias, colaboradores, equipamentos e vulnerabilidades.

# Estado de Arte

O Regulamento Geral sobre a Proteção de Dados entrou em vigor em 25 de maio de 2018 e é um regulamento do direito europeu sobre privacidade e proteção de dados pessoais, aplicável a todos os indivíduos na União Europeia, regulamenta também os dados pessoais exportados para fora da União Europeia.

Este regulamento obriga a informar acerca da base legal para o tratamento de dados, prazo de conservação dos mesmos e a sua transferência.

O regulamento obriga a controlar as circunstâncias em que foi obtido o consentimento dos titulares quando isso for base legal do tratamento dos dados pessoais. Existem um conjunto de exigências para obtenção desse consentimento e o seu não cumprimento obriga à obtenção de um novo consentimento.

O regulamento obriga a um grande controlo do risco associado ao possível roubo de informação. Este controlo de risco deverá passar a ser garantido por medidas de segurança efetivas que garantam a confidencialidade, a integridade dos dados e que previnam a destruição, ou a divulgação/acesso não autorizado de dados.

Existem DPO’s – Data Protection Officer – (Delegado de Proteção de Dados) que são um canal de comunicação entre uma determinada empresa e uma agência de Proteção de Dados e os titulares dos dados pessoais e têm como função auxiliar a empresa no processo de adaptação e estruturação de um programa com foco na proteção de dados, que tem de se manter a par de todas as obrigações do RGPD. Estes podem usar várias aplicações que os ajudam nesta tarefa, como por exemplo, a GDRP App.

# Bibliotecas

As bibliotecas deste trabalho foram implementadas de forma a respeitar a programação por camadas Ntier, e por isso existem 3 camadas que foram apelidadas de “*ax*\_” como prefixo para facilitar a sua identificação. Assim existe *“a1\_TrabalhoPratico”* que se trata da camada mais superficial, *Presentation Layer,* de seguida, *“a2\_RegrasNegocio”* que tem como objetivo impor regras, segurança e validações, *Business Rules Layer.* Por fim, existe a camada mais profunda *“a3\_DadosClasses”* que tem como objetivo manusear os dados das classes, *Data Layer.*

Existem também duas camadas auxiliares que tiveram como prefixo “*cx*\_” que são “*c1\_ObjetosNegocio”* que contém todas as classes que servem de objeto de negócio, e também *“c2\_Validacoes”* que contém as exceptions necessárias.

## a1\_TrabalhoPratico

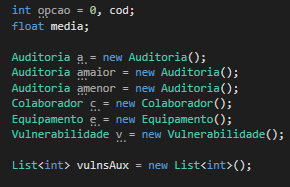
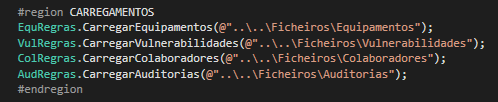
****Nesta parte final é onde se encontra o main e onde são implementados todos os métodos que o programa oferece. Aqui serão criadas asalgumas variáveis auxiliares para obter informação**:**

Figura 1: a1\_TrabalhoPratico *variáveis*

Ao arranque da aplicação são carregados os **ficheiros binários (saves)**:

Figura 2: a1\_TrabalhoPratico *saves*

Ao encerrar a aplicação todas as modificações são guardadas.

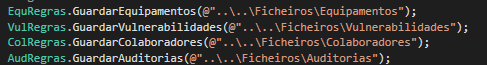
****

Figura 3: a1\_TrabalhoPratico *gravar*

## a2\_RegrasNegocio

Uma imagem com captura de ecrã

Descrição gerada automaticamente Figura 4: a2\_RegrasNegocio Função com Verificação

Quase todas as funções presentes nesta biblioteca são responsáveis pela verificação de todas as regras e exceções antes das variáveis prosseguirem para a camada dos dados. Estas atuam nas funções de a2\_RegrasNegocio: das auditorias, dos colaboradores, dos equipamentos e das vulnerabilidades.

## a3\_DadosClasses

Nesta biblioteca, todos os dados são manipulados e não acessíveis pelo utilizador.

Estes dados estão todos guardados em estruturas de dados *Collections Generic* do tipo *List<Type>.*

Assim, nesta biblioteca, estão presentes funções focadas em:

**Verificar a existência de algum objeto:**

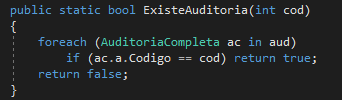


Figura 5: a2\_DadosClasses *ExisteAuditoria(...)*

Neste caso, a função percorre a lista de objetos até encontrar algum com o mesmo código, se encontrar, retornará true, se não, retornará false.

**Obter quantidade de objetos numa lista:**

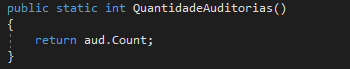


Figura 6: a2\_DadosClasses *QuantidadeAuditorias(...)*

Esta função simplesmente retorna a quantidade de objetos que existe na lista “aud”.

**Adicionar/Remover algum objeto:**

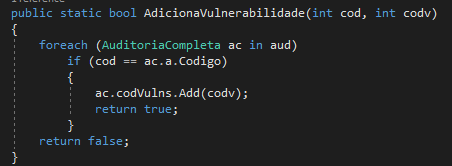


Figura 7: a2\_DadosClasses Auditorias.*AdicionaVulnerabilidade(...)*

Esta função recebe como parâmetro um código de uma auditoria e também um código de uma vulnerabilidade. Depois percorre toda a lista de objetos e adiciona a vulnerabilidade a uma auditoria quando esta for encontrada na lista.

**Editar algum dado:**

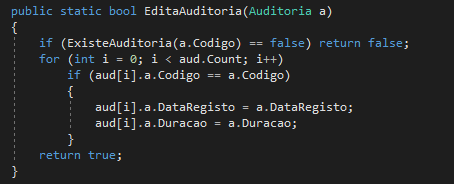


Figura 8: a2\_DadosClasses *EditaAuditoria(...)*

Neste tipo de função, a lista é percorrida até encontrar alguma com o mesmo código do objeto que recebeu como parâmetro, se encontrar alguma com o mesmo código, os dados serão editados, senão, retornará false;

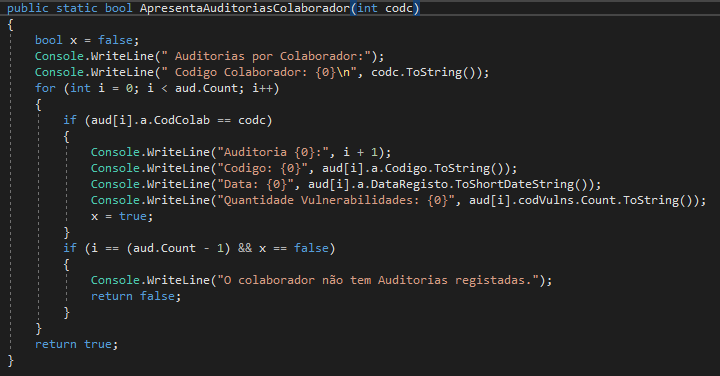
**Apresentar algum dado: **

Figura 9: a2\_DadosClasses *ApresentaAuditoriasColaborador(...)*

Neste tipo de função, será apresentado ao utilizador as informações que este pretender previamente. A função irá percorrer a lista de dados e se esta encontrar a pretendida pelo utilizador, irá mostrar na consola os dados.

Existe também a variação deste código que apresentará os dados ordenados e tabelados, os máximos e mínimos, e a média:

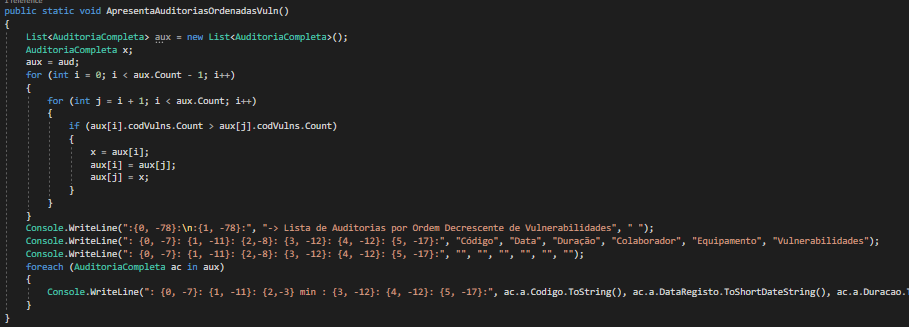


Figura 10: a2\_DadosClasses *ApresentaAuditoriasOrdenadasVuln(...)*

Nesta função, para se ordenar os dados, é usado o famoso **bubble** **sort**, e posteriormente uma apresentação tabelada da informação.

Nesta função, a lista é percorrida e sempre que o valor chave for superior/inferior (depende do pretendido), esse será guardado e apresentado no final.

**Guardar/Carregar algum dado em ficheiro binário:**

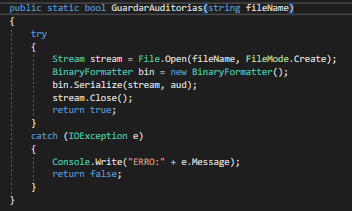


Figura 11: a2\_DadosClasses *GuardarAuditorias(...)*

Esta função, neste caso, guarda toda a informação que se encontra armazenada na lista “aud”, recebendo como argumento o diretório de onde o ficheiro deve ser guardado.

**Verificar/alterar a atividade de um colaborador:**

Um colaborador ativo, em momento algum, deve ser removido dos registos da aplicação, para isso foi implementado duas funções, uma para verificar atividade, e a outra para alterar a sua atividade:

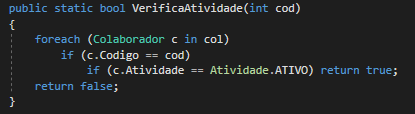


Figura 12: a2\_DadosClasses *VerificarAtividade(...)*

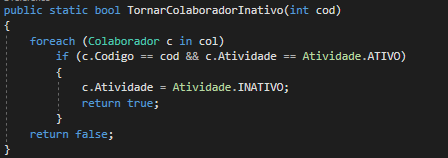


Figura 13: a2\_DadosClasses *TornarColaboradorInativo(...)*

Estas duas funções manipulam o *enum* relativo à classe Colaborador. Recebem como parâmetro o código do colaborador e verificam o estado do *enum*, no caso da segunda função altera o estado do enum para Inativo.

Nesta biblioteca nomeadamente nos ficheiros *Equipamentos.cs* e *Auditorias.cs* existem também duas classes auxiliares “AuditoriaCompleta” e “EquipamentoCompleto”, que servem para complementar as classes singulares “Auditoria” e “Equipamento” com uma lista de inteiros com códigos de vulnerabilidades.

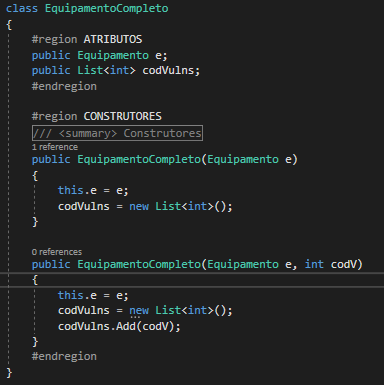
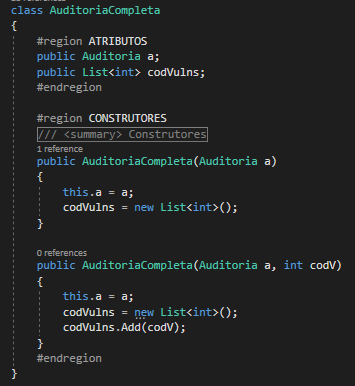


Figura14: a2\_DadosClasses *EquipamentosCompleto*

Figura 15: a2\_DadosClasses *AuditoriaCompleta*

Estas listas são usadas para armazenar a informação quando existe uma auditoria ou um equipamento com todos os dados, isto acontece para que ao serem criados estes objetos não existam atributos criados, mas sem informação e para que não sejam manipuladas listas a partir da camada mais superficial.

## c1\_ObjetosNegocio

Nesta biblioteca do programa são criadas todas as classes designadas por objetos de negócio. Cada classe tem o seu construtor por defeito e construtor com valores vindos do exterior criados, e também cada atributo destas classes tem a sua propriedade criada.

São criadas também as interfaces relativas a cada classe. A classe “Colaborador” tem como herança a classe Pessoa.

**Auditoria:**

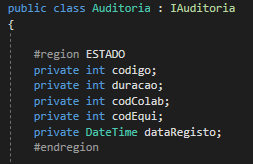


Figura 16: c1\_ObjetosNegocio *Auditoria*

*codigo*-> Código da auditoria;

*duracao*-> Duração da auditoria;

*codColab*-> Código do colaborador responsável;

*codEqui*-> Código do Equipamento em questão;

*dataRegisto*-> Data da realização da auditoria.

**Colaborador:**

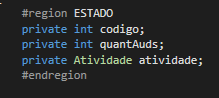


Figura 17: c1\_ObjetosNegocio *Colaborador*

*codigo*-> Código do colaborador;

*quantAuds*-> Quantidade de Auditorias;

*atividade*-> Estado de atividade do colaborador (Ativo/Inativo);

**Equipamento:**

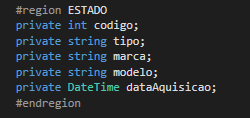


Figura 18: c1\_ObjetosNegocio *Equipamento*

*codigo*-> Código do Equipamento;

*tipo*-> Tipo de Equipamento;

*marca*-> Marca do Equipamento;

*modelo*-> Modelo do Equipamento;

*dataAquisicao*-> Data de aquisição do Equipamento;

**Vulnerabilidades:**

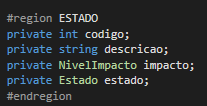


Figura 19: c1\_ObjetosNegocio *Vulnerabilidades*

*codigo-> Código da Vulnerabilidade;*

*descricao*-> Descrição da Vulnerabilidade;

*impacto*-> Nível de impacto da Vulnerabilidade;

*estado*-> Estado de resolução da Vulnerabilidade;

**Pessoa:**

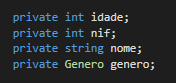


Figura 20: c1\_ObjetosNegocio *Pessoa*

Esta classe serve apenas de herança para a classe Colaborador.

*idade*-> Idade do colaborador;

*nif*-> Nif do colaborador;

*nome*-> Nome do colaborador;

*genero*-> Género do colaborador;

## c2\_Validacoes

Nesta libraria existem dois ficheiros onde foram criadas validações e exceções que são usadas de forma recorrente.

**Validações (TryCatchInt):**

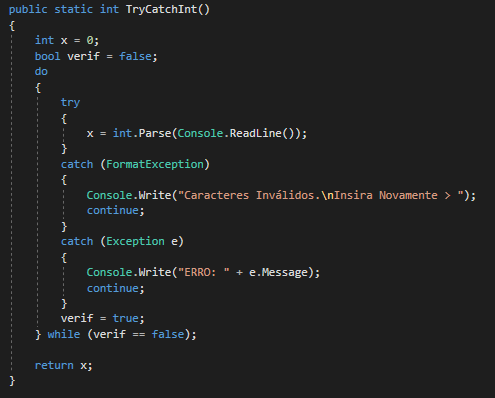


Figura 21: c2\_Validacoes *TryCatchInt*

Esta função serve para receber um inteiro do utilizador verificando se este correto ou não através de uma exception. Caso o input do utilizador seja um inteiro a função devolve esse inteiro, caso não seja, a exception apresenta o erro e pede novamente o inteiro.

# Conclusão

A construção do programa nesta segunda fase encontra-se perto da sua conclusão, apresenta todos os requisitos e encontra-se visualmente agradável e de forma organizada. O código tem os comentários necessários para que qualquer pessoa consiga facilmente perceber do que cada parte se trata.

Com esta parte do trabalho prático, as nossas capacidades relativas ao paradigma POO, aos ficheiros, às *Collections Generics*, ao Ntier, às Interfaces e às Exceptions foram aprimoradas.

O programa encontra-se alojado no seguinte repositório GitHub: <https://github.com/carlosantos2103/19432_19433_LP2>

# Bibliografia

[**https://mydataprivacy.eu/o-regulamento/**](https://mydataprivacy.eu/o-regulamento/)

[**https://eur-lex.europa.eu/legal-content/PT/TXT/HTML/?uri=OJ:L:2016:119:FULL**](https://eur-lex.europa.eu/legal-content/PT/TXT/HTML/?uri=OJ:L:2016:119:FULL)

[**https://gdprapp.com/**](https://gdprapp.com/)