INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

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

Lista de Figuras

1 Implementação

1.1 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.

1.2 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.

2 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.

3 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.

3.1 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 as algumas variáveis auxiliares para obter informação**:**

```
int opcao = 0, cod;
float media;

Auditoria a = new Auditoria();
Auditoria amaior = new Auditoria();
Auditoria amenor = new Auditoria();
Colaborador c = new Colaborador();
Equipamento e = new Equipamento();
Vulnerabilidade v = new Vulnerabilidade();

List<int> vulnsAux = new List<int>();
```

Figura 1: a1_TrabalhoPratico *variáveis*

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

```
#region CARREGAMENTOS
EquRegras.CarregarEquipamentos(@"..\..\Ficheiros\Equipamentos");
VulRegras.CarregarVulnerabilidades(@"..\..\Ficheiros\Vulnerabilidades");
ColRegras.CarregarColaboradores(@"..\..\Ficheiros\Colaboradores");
AudRegras.CarregarAuditorias(@"..\..\Ficheiros\Auditorias");
#endregion
```

Figura 2: a1_TrabalhoPratico *saves*

Ao encerrar a aplicação todas as modificações são guardadas.

```
EquRegras.GuardarEquipamentos(@"..\..\Ficheiros\Equipamentos");
VulRegras.GuardarVulnerabilidades(@"..\..\Ficheiros\Vulnerabilidades");
ColRegras.GuardarColaboradores(@"..\..\Ficheiros\Colaboradores");
AudRegras.GuardarAuditorias(@"..\..\Ficheiros\Auditorias");
```

Figura 3: a1_TrabalhoPratico *gravar*

3.2 a2_RegrasNegocio

```
public static bool AdicionaVulnerabilidadeAuditoria(int cod, int codv)
{
    try
    {
        return Auditorias.AdicionaVulnerabilidade(cod, codv);
    }
    catch (Excecoes x)
    {
        Console.WriteLine(x);
    }
    return false;
}
```

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.

3.3 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:

```
public static bool ExisteAuditoria(int cod)
{
    foreach (AuditoriaCompleta ac in aud)
        if (ac.a.Codigo == cod) return true;
    return false;
}
```

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:

```
public static int QuantidadeAuditorias()
{
    return aud.Count;
}
```

Figura 6: a2_DadosClasses *QuantidadeAuditorias(...)*

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

Adicionar/Remover algum objeto:

```
public static bool AdicionaVulnerabilidade(int cod, int codv)
{
    foreach (AuditoriaCompleta ac in aud)
        if (cod == ac.a.Codigo)
        {
            ac.codVulns.Add(codv);
            return true;
        }
    return false;
}
```

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:

```
public static bool EditaAuditoria(Auditoria a)
{
    if (ExisteAuditoria(a.Codigo) == false) return false;
    for (int i = 0; i < aud.Count; i++)
        if (aud[i].a.Codigo == a.Codigo)
        {
            aud[i].a.DataRegisto = a.DataRegisto;
            aud[i].a.Duracao = a.Duracao;
        }
    return true;
}
```

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:

```
public static bool ApresentaAuditoriasColaborador(int codc)
{
    bool x = false;
    Console.WriteLine(" Auditorias por Colaborador:");
    Console.WriteLine(" Codigo Colaborador: {0}\n", codc.ToString());
    for (int i = 0; i < aud.Count; i++)
    {
        if (aud[i].a.CodColab == codc)
        {
            Console.WriteLine("Auditoria {0}:", i + 1);
            Console.WriteLine("Codigo: {0}", aud[i].a.Codigo.ToString());
            Console.WriteLine("Data: {0}", aud[i].a.DataRegisto.ToShortDateString());
            Console.WriteLine("Quantidade Vulnerabilidades: {0}", aud[i].codVulns.Count.ToString());
            x = true;
        }
        if (i == (aud.Count - 1) && x == false)
        {
            Console.WriteLine("O colaborador não tem Auditorias registadas.");
            return false;
        }
    }
    return true;
}
```

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:

```
public static void ApresentaAuditoriasOrdenadasVuln()
{
    List<AuditoriaCompleta> aux = new List<AuditoriaCompleta>();
    AuditoriaCompleta x;
    aux = aud;
    for (int i = 0; i < aux.Count - 1; i++)
    {
        for (int j = i + 1; i < aux.Count; i++)
        {
            if (aux[i].codVulns.Count > aux[j].codVulns.Count)
            {
                x = aux[i];
                aux[i] = aux[j];
                aux[j] = x;
            }
        }
    }
    Console.WriteLine(":{0, -78}:\n:{1, -78}:", "-> Lista de Auditorias por Ordem Decrescente de Vulnerabilidades", " ");
    Console.WriteLine(": {0, -7}: {1, -11}: {2,-8}: {3, -12}: {4, -12}: {5, -17}:", "Código", "Data", "Duração", "Colaborador", "Equipamento", "Vulnerabilidades");
    Console.WriteLine(": {0, -7}: {1, -11}: {2,-8}: {3, -12}: {4, -12}: {5, -17}:", "", "", "", "", "", "");
    foreach (AuditoriaCompleta ac in aux)
    {
        Console.WriteLine(": {0, -7}: {1, -11}: {2,-3} min : {3, -12}: {4, -12}: {5, -17}:", ac.a.Codigo.ToString(), ac.a.DataRegisto.ToShortDateString(), ac.a.Duracao.
    }
}
```

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:

```
public static bool GuardarAuditorias(string fileName)
{
    try
    {
        Stream stream = File.Open(fileName, FileMode.Create);
        BinaryFormatter bin = new BinaryFormatter();
        bin.Serialize(stream, aud);
        stream.Close();
        return true;
    }
    catch (IOException e)
    {
        Console.Write("ERRO:" + e.Message);
        return false;
    }
}
```

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:

```
public static bool VerificaAtividade(int cod)
{
    foreach (Colaborador c in col)
        if (c.Codigo == cod)
            if (c.Atividade == Atividade.ATIVO) return true;
    return false;
}
```

Figura 12: a2_DadosClasses *VerificarAtividade(...)*

```
public static bool TornarColaboradorInativo(int cod)
{
    foreach (Colaborador c in col)
        if (c.Codigo == cod && c.Atividade == Atividade.ATIVO)
        {
            c.Atividade = Atividade.INATIVO;
            return true;
        }
    return false;
}
```

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.

```
class EquipamentoCompleto
{
    #region ATRIBUTOS
    public Equipamento e;
    public List<int> codVulns;
    #endregion

    #region CONSTRUTORES
    /// <summary> Construtores
    1 reference
    public EquipamentoCompleto(Equipamento e)
    {
        this.e = e;
        codVulns = new List<int>();
    }

    0 references
    public EquipamentoCompleto(Equipamento e, int codV)
    {
        this.e = e;
        codVulns = new List<int>();
        codVulns.Add(codV);
    }
    #endregion
}
```

```
class AuditoriaCompleta
{
    #region ATRIBUTOS
    public Auditoria a;
    public List<int> codVulns;
    #endregion

    #region CONSTRUTORES
    /// <summary> Construtores
    1 reference
    public AuditoriaCompleta(Auditoria a)
    {
        this.a = a;
        codVulns = new List<int>();
    }

    0 references
    public AuditoriaCompleta(Auditoria a, int codV)
    {
        this.a = a;
        codVulns = new List<int>();
        codVulns.Add(codV);
    }
    #endregion
}
```

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.

3.4 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:

```
#region ESTADO
private int codigo;
private int duracao;
private int codColab;
private int codEqui;
private DateTime dataRegisto;
#endregion
```

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:

```
#region ESTADO
private int codigo;
private int quantAuds;
private Atividade atividade;
#endregion
```

Figura 17: c1_ObjetosNegocio *Colaborador*

codigo-> Código do colaborador;

quantAuds-> Quantidade de Auditorias;

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

Equipamento:

```
#region ESTADO
private int codigo;
private string tipo;
private string marca;
private string modelo;
private DateTime dataAquisicao;
#endregion
```

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:

```
#region ESTADO
private int codigo;
private string descricao;
private NivelImpacto impacto;
private Estado estado;
#endregion
```

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:

```
private int idade;
private int nif;
private string nome;
private Genero genero;
```

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;

3.5 c2_Validacoes

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

Validações (TryCatchInt):

```
public static int TryCatchInt()
{
    int x = 0;
    bool verif = false;
    do
    {
        try
        {
            x = int.Parse(Console.ReadLine());
        }
        catch (FormatException)
        {
            Console.Write("Caracteres Inválidos.\nInsira Novamente > ");
            continue;
        }
        catch (Exception e)
        {
            Console.Write("ERRO: " + e.Message);
            continue;
        }
        verif = true;
    } while (verif == false);

    return x;
}
```

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.

4 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://eur-lex.europa.eu/legal-content/PT/TXT/HTML/?uri=OJ:L:2016:119:FULL

https://gdprapp.com/