

Relatório de trabalho prático

Gestão de Auditorias

Yuri Serediuk Lemos

Nº 19256

João Pedro Marques Figueiredo

Nº 19337

Trabalho realizado sob a orientação de:

Luís Ferreira

Linguagens de Programação II

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, 25 de Abril de 2020

Índice

Relatório de trabalho prático	1
Gestão de Auditorias	1
1. Introdução	5
1.1. Contextualização	5
1.2. Motivações e objetivos	6
2. Estado da arte	7
3. Implementação	8
3.1. Descrição	8
3.2. Solução	9
3.2.1. Diagrama	9
3.2.2. Criação de Classes	10
3.2.2.1. Classe Pessoa.....	10
3.2.2.2. Classe Funcionário.....	11
3.2.2.3. Classe Funcionários	12
3.2.2.4. Classe Equipamento	13
3.2.2.5. Classe Equipamentos.....	14
3.2.2.6. Classe Vulnerabilidade	15
3.2.2.7. Classe Vulnerabilidades.....	16
3.2.2.8. Classe Ocorrência	17
3.2.2.9. Classe Interface IMetodosGenericos.....	18
3.2.2.10. Classe Program.....	19
4. Conclusão	20
4.1. Lições aprendidas	20
4.2. Apreciação Final	20

Índice de Imagens

FIGURA 1 DIAGRAMA DE CLASSES	9
FIGURA 2 CLASSE PESSOA	10
FIGURA 3 CLASSE FUNCIONÁRIO	11
FIGURA 4 CLASSE FUNCIONÁRIOS	12
FIGURA 5 CLASSE EQUIPAMENTO	13
FIGURA 6 CLASSE EQUIPAMENTOS.....	14
FIGURA 7 CLASSE VULNERABILIDADE	15
FIGURA 8 CLASSE VULNERABILIDADES	16
FIGURA 9 CLASSE OCORRÊNCIA	17
FIGURA 10 CLASSE INTERFACE IMETODOSGENERICOS.....	18
FIGURA 11 CLASSE PROGRAM	19

1. Introdução

Este projeto foi realizado tendo como objetivo a avaliação da compreensão dos alunos para a linguagem de programação orientada a objetos (POO). Nesta primeira parte do projeto desenvolvido mostra a capacidade de compreensão dos alunos dos termos básicos do paradigma desta linguagem.

1.1. Contextualização

O advento da tecnologia fez com que fossem necessários novos arranjos para a sociedade e para o poder público. Não raro, a lei não consegue acompanhar as mudanças velozes do mundo conectado, mas há anos que a demanda por leis mais protetivas aos dados pessoais do cidadão ganha espaço entre especialistas e entusiastas da tecnologia. Isto porque, o cidadão comum, por vezes, não é capaz de reconhecer os riscos e oportunidades criadas para a exploração de seus dados pessoais. Assim, muitas vezes abrem mão de informações pessoais importantes em troca de simples acesso a sites e aplicativos de telemóveis. Não apenas isto, mas toda a troca de dados realizada em estabelecimentos comerciais físicos e em registos quaisquer transações do tipo também deveriam constar no escopo de uma lei geral de proteção de dados. A União Europeia há alguns anos já gozava de tal proteção com a Diretiva 95/46 CE, a qual buscava regular a coleta, uso e tratamento de dados no território europeu. Porém, acompanhando as tendências de uso global, transferência e necessidade de apagamento ou prestação de contas sobre dados pessoais, um novo Regulamento foi elaborado a fim de expandir a proteção ao indivíduo. Sendo assim, após um período de dois anos, entrou em vigor o Regulamento Geral de Proteção de Dados europeu — GDPR. Com texto extenso e uma preocupação notável em abranger as mais diversas possibilidades de transação envolvendo dados, o GDPR sagrou-se um marco na proteção de dados e da proteção à privacidade do utilizador. Instituiu princípios sólidos e claros, a fim de não abrir margem para interpretações diversas. Assim, o documento cria no indivíduo a possibilidade de domínio sobre os próprios dados, reclamando a propriedade destes como algo pessoal e não comercial, pertencente às empresas ou amplamente explorado pelo Poder Público.

1.2. Motivações e objetivos

O propósito do trabalho foi montar de forma criativa, consoante ao RGPD um programa de fácil acessibilidade utilizando a linguagem C#, com isso pode-se então ter uma melhor percepção das falhas ao RGPD numa determinada empresa.

2. Estado da arte

O Regulamento Geral sobre a Proteção de Dados (RGPD), aprovado no Parlamento Europeu em abril de 2016, visa regular a proteção das pessoas singulares no que diz respeito ao tratamento de dados pessoais e à livre circulação desses dados, revogando assim a Diretiva 95/46/CE. Este Regulamento introduziu alterações significativas às regras mantidas em matérias de proteção de dados, impondo novas e rigorosas obrigações e severas coimas como punição ao seu não cumprimento.

O RGPD proporcionou um período de transição de 2 anos para que as organizações implementassem um plano de transformação que proporcione o cumprimento de todos os requisitos necessários à aplicabilidade do RGPD, em 25 de maio de 2018. Em Portugal, a 8 de agosto de 2019, foi publicada a nova Lei nº 58/2019 que assegura a execução, na ordem jurídica nacional, do RGPD, e que revoga a anterior Lei nº 67/98.

O mundo virtual está em crescimento todos os dias, e a proteção dos dados é essencial para que a confidencialidade de uma companhia não seja comprometida. E para que os dados sejam resguardados de forma segura judicialmente, surgiu o *Data Protection Officer* (DPO) ou, em português, diretor de Proteção de Dados.

De acordo com a lei LGPD (Lei Geral de Proteção de Dados – 13.709/2018), a profissão entrará em vigor em agosto de 2020. O especialista será responsável pelo aconselhamento sobre as normas vigentes relativas à proteção de dados pessoais, o monitoramento do cumprimento do GDPR (*General Data Protection Regulation*) pela entidade a que está vinculado e a cooperação com as autoridades públicas supervisoras da norma.

3. Implementação

3.1. Descrição

As auditorias internas podem ser efetuadas por vários colaboradores. Cada auditoria deve ter em conta um código (ano/numero), quem foi o colaborador responsável, a data, quantidade de vulnerabilidade identificadas. Depois de inseridos os dados acerca das auditorias, quando terminar, a aplicação deverá mostrar:

- Uma relação das auditorias efetuadas, indicando o código e o colaborador responsável.
- Qual a auditoria onde foram detetadas mais e menos vulnerabilidades, indicando o seu código, data e quantidade de vulnerabilidades.

A aplicação deverá ter também uma estrutura de colaboradores, equipamentos, vulnerabilidades e um registo de ocorrência

3.2. Solução

3.2.1. Diagrama

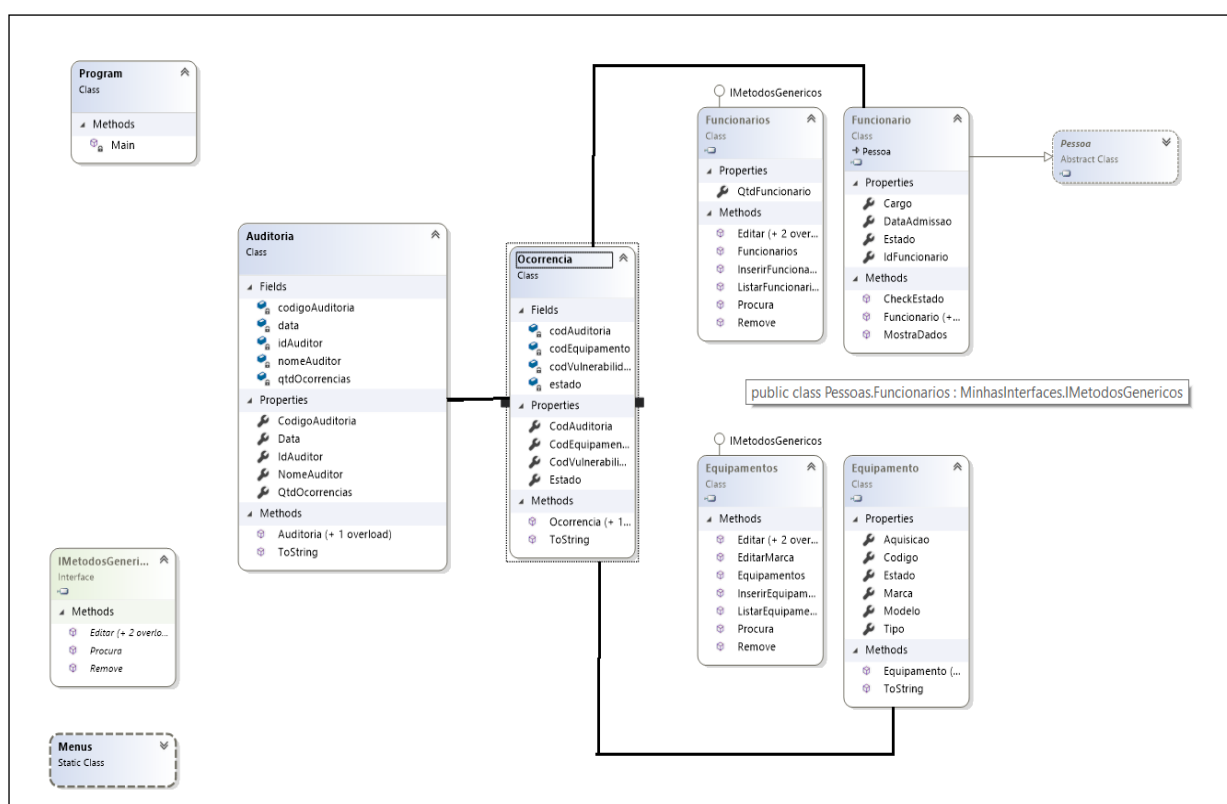


Figura 1 Diagrama de classes

3.2.2. Criação de Classes

3.2.2.1. Classe Pessoa

Esta classe representa uma Pessoa, nesta classe contém:

- Nome
- Bilhete de identidade
- Contacto
- Morada
- Número Contribuinte

```
//-----  
// <copyright file="Program.cs" company="IPCA">  
// Copyright (c) IPCA. All rights reserved.</copyright>  
//-----  
// <author>Yuri Lemos</author>  
// <author>João Figueiredo</author>  
// <desc> This program do the basics of C#</desc>  
// <Date> 4 / 4 / 2020 </Date>  
// <version>1.0</version>  
-----  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Pessoas  
{  
    /// <summary>  
    /// Esta classe representa uma Pessoa  
    /// <code>Type: Classe Abstrata</code>  
    /// </summary>  
    1 reference  
    public abstract class Pessoa  
    {  
        #region Atributos  
        /// <summary>  
        /// Os valores recebidos aqui são:  
        /// *BI - Numero do bilhete de identidade  
        /// *Nome - Nome da pessoa  
        /// *nContribuinte - Nº Contribuinte  
        /// *morada - Morada da pessoa  
        /// *contacto - Contacto da pessoa  
        /// </summary>  
        string bi;  
        string nome;  
        int nContribuinte;  
        string morada;  
        int contacto;  
        #endregion  
    }  
}
```

Figura 2 Classe Pessoa

3.2.2.2. Classe Funcionário

Esta classe representa um Funcionário, ela é uma classe abstrata e por isso não poderá ser criada instâncias a partir desta classe. Nesta classe contém os seguintes atributos:

- Cargo
- Data Admissão
- Estado
- Identificação do funcionário

```
//-----
// <copyright file="Program.cs" company="IPCA">
// Copyright (c) IPCA. All rights reserved.</copyright>
//-----
// <author>Yuri Lemos</author>
// <author>João Figueiredo</author>
// <desc> This program do the basics of C#</desc>
// <Date> 4 / 4 / 2020 </Date>
// <version>1.0</version>

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Pessoas
{
    /// <summary>
    /// Esta Classe representa uma instancia de Funcionario
    /// <code>Herança: Pessoa</code>
    /// </summary>
    [reference]
    public class Funcionario : Pessoa
    {
        #region Atributos
        /// <summary>
        /// Os valores recebidos aqui são:
        /// *idFuncionario - Numero de identificação do funcionario na empresa
        /// *cargo - Qual o cargo ou função que ocupa
        /// *dataAdmissao - A data em que foi admitido na empresa
        /// *estado - Estado da pessoa na empresa ( Se ainda trabalha para a empresa ou não)
        /// </summary>
        int idFuncionario;
        string cargo;
        DateTime dataAdmissao;
        bool estado;

        #endregion

        [Constructor]
        [Propriedades]
```

Figura 3 Classe Funcionário

3.2.2.3. Classe Funcionários

Esta classe representa vários funcionários, nesta classe contém:

- Lista de funcionários
- Quantidade de funcionários

```
//-----  
// <copyright file="Program.cs" company="IPCA">  
// Copyright (c) IPCA. All rights reserved.</copyright>  
//-----  
// <author>Yuri Lemos</author>  
// <author>João Figueiredo</author>  
// <desc> This program do the basics of C#</desc>  
// <Date> 4 / 4 / 2020 </Date>  
// <version>1.0</version>  
//-----  
  
using MinhasInterfaces;  
using System;  
using System.Collections.Generic;  
  
namespace Pessoas  
{  
    /// <summary>  
    /// Esta Classe representa uma instancia de Funcionarios  
    /// <code>Implementa: IMetodosGenericos</code>  
    /// </summary>  
  
    3 references  
    public class Funcionarios : IMetodosGenericos  
    {  
        #region Atributos  
        /// <summary>  
        /// Os valores recebidos aqui são:  
        /// *funcionarios - Criação de uma lista do tipo Funcionarios  
        /// *qtdFuncionarios - Quantidade de funcionarios que existe na lista  
        /// </summary>  
        List<Funcionario> funcionarios;  
        int qtdFuncionarios;  
        #endregion  
  
        Construtor  
        Propriedades  
        Metodos  
    }  
}
```

Figura 4 Classe Funcionários

3.2.2.4. Classe Equipamento

Esta classe representa um equipamento, nesta classe contém:

- Código
- Marca
- Modelo
- Data aquisição
- Estado
- Tipo de equipamento

```
//-----
// <copyright file="Program.cs" company="IPCA">
// Copyright (c) IPCA. All rights reserved.</copyright>
//-----
// <author>Yuri Lemos</author>
// <author>João Figueiredo</author>
// <desc> This program do the basics of C#</desc>
// <Date> 4 / 4 / 2020 </Date>
// <version>1.0</version>

using System;

namespace Aparelhos
{
    /// <summary>
    /// Esta Classe representa uma instancia de Equipamento
    /// </summary>
    9 references
    public class Equipamento
    {
        #region Atributos
        /// <summary>
        /// Os valores recebidos aqui são:
        /// *codigoEqui - Numero de identificação do equipamento na empresa
        /// *tipo - Qual a categoria do equipamento (Eg: Impressora, Portatil, etc...)
        /// *marca - Qual a marca do equipamento da empresa
        /// *modelo - Qual o modelo do equipamento
        /// *aquisicao - Data de compra do equipamento pela Empresa
        /// *estado - Qual a situação d o equipamento na empresa (se foi para o lixo)
        /// </summary>

        int codigoEqui;
        string tipo;
        string marca;
        string modelo;
        DateTime aquisicao;
        bool estado;
        #endregion

        Construtor
        Propriedades
    }
}
```

Figura 5 Classe Equipamento

3.2.2.5. Classe Equipamentos

Esta classe representa vários equipamentos, nesta classe contém:

- Lista de equipamentos
- Quantidade de Equipamentos

```
//-----
// <copyright file="Program.cs" company="IPCA">
// Copyright (c) IPCA. All rights reserved.</copyright>
//-----
// <author>Yuri Lemos</author>
// <author>João Figueiredo</author>
// <desc> This program do the basics of C#</desc>
// <Date> 4 / 4 / 2020 </Date>
// <version>1.0</version>
using MinhasInterfaces;
using System;
using System.Collections.Generic;

namespace Aparelhos
{
    /// <summary>
    /// Esta Classe representa uma instancia de Equipamentos
    /// <code>Implementa: IMetodosGenericos</code>
    /// </summary>

    3 references
    public class Equipamentos : IMetodosGenericos
    {
        #region Atributos
        /// <summary>
        /// Os valores recebidos aqui são:
        /// *equipamentos - Criação de uma lista do tipo Equipamento
        /// *qtdEquipamentos - Quantidade de equipamentos que existe na lista
        /// </summary>
        List<Equipamento> equipamentos;
        int qtdEquipamentos;
        #endregion

        Construtor
        Metodos
    }
}
```

Figura 6 Classe Equipamentos

3.2.2.6. Classe Vulnerabilidade

Esta classe representa uma vulnerabilidade, nesta classe contém:

- Código
- Descrição
- Nível de impacto

```
//-----  
// <copyright file="Program.cs" company="IPCA">  
// Copyright (c) IPCA. All rights reserved.</copyright>  
//-----  
// <author>Yuri Lemos</author>  
// <author>João Figueiredo</author>  
// <desc> This program do the basics of C#</desc>  
// <Date> 4 / 4 / 2020 </Date>  
// <version>1.0</version>  
//-----  
  
namespace Defeitos  
{  
    /// <summary>  
    ///  
    /// </summary>  
    10 references  
    public class Vulnerabilidade  
    {  
        /*  
        * Vulnerabilidades Contém:  
        *   - Código da vulnerabilidade  
        *   - Descrição  
        *   - Nível de impacto que são eles:  
        *       + Elevado  
        *       + Médio  
        *       + Baixo  
        */  
  
        #region Atributos  
        int codigo;  
        string descricao;  
        string nivelImpacto;  
  
        #endregion  
  
        [Constructor]  
  
        [Propriedades]  
  
        [Metodos]  
    }  
}
```

Figura 7 Classe Vulnerabilidade

3.2.2.7. Classe Vulnerabilidades

Esta classe representa uma lista de vulnerabilidades, nesta classe contém:

- Lista de vulnerabilidades
- Quantidade de vulnerabilidades

```
//-----  
// <copyright file="Program.cs" company="IPCA">  
// Copyright (c) IPCA. All rights reserved.</copyright>  
//-----  
// <author>Yuri Lemos</author>  
// <author>João Figueiredo</author>  
// <desc> This program do the basics of C#</desc>  
// <Date> 4 / 4 / 2020 </Date>  
// <version>1.0</version>  
//-----  
  
using System;  
using System.Collections.Generic;  
using MinhasInterfaces;  
namespace Defeitos  
{  
    /// <summary>  
    ///  
    /// </summary>  
    3 references  
    public class Vulnerabilidades : IMetodosGenericos  
    {  
        /*  
        * Vulnerabilidades Contém:  
        * - Uma lista de vulnerabilidades  
        * - Quantidade de vulnerabilidades  
        */  
        #region Atributos  
        List<Vulnerabilidade> vulnerabilidades;  
        int qtdVulnerabilidades;  
        #endregion  
        Constructor  
        Propriedades  
        Metodos  
    }  
}
```

Figura 8 Classe Vulnerabilidades

3.2.2.8. Classe Ocorrência

Esta classe representa uma ocorrência, nesta classe contém:

- Código da auditoria
- Código do equipamento
- Código da vulnerabilidade
- Estado

```
//-----  
// <copyright file="Program.cs" company="IPCA">  
// Copyright (c) IPCA. All rights reserved.</copyright>  
//-----  
// <author>Yuri Lemos </author>  
// <author>João Figueiredo</author>  
// <desc> This program do the basics of C#</desc>  
// <Date> 4 / 4 / 2020 </Date>  
// <version>1.0</version>  
//-----  
  
namespace Auditorias  
{  
    /// <summary>  
    ///  
    /// </summary>  
    2 references  
    class Ocorrencia  
    {  
        #region Atributos  
        /// <summary>  
        /// Os valores recebidos aqui são:  
        /// *  
        /// *dataAdmissao - A data em que foi admitido na empresa  
        /// *estado - Estado da pessoa na empresa ( Se ainda trabalha para a empresa ou não)  
        /// </summary>  
        int codAuditoria;  
        int codVulnerabilidade;  
        int codEquipamento;  
        bool estado;  
        #endregion  
        Construtor  
        Propriedades  
        Metodos  
    }  
}
```

Figura 9 Classe Ocorrência

3.2.2.9. Classe Interface IMetodosGenericos

Esta classe é uma interface contendo vários métodos abstratos, estes métodos são:

- Editar
- Procura
- Remove

Estes métodos podem ser utilizados por qualquer classe desde que esta interface seja chamada e que todos seus métodos inseridos na classe que está a implementá-lo.

```
// <copyright file="Program.cs" company="IPCA">
// Copyright (c) IPCA. All rights reserved.</copyright>
//-----
// <author>Yuri Lemos</author>
// <author>João Figueiredo</author>
// <desc> This program do the basics of C#</desc>
// <Date> 4 / 4 / 2020 </Date>
// <version>1.0</version>

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MinhasInterfaces
{
    /// <summary>
    /// Isso é uma interface com métodos genéricos
    /// </summary>
    3 references
    public interface IMetodosGenericos
    {
        15 references
        int Procura(int id);

        6 references
        bool Remove(int id);

        3 references
        bool Editar(int id, string nome);
        6 references
        bool Editar(int id, int numero);
        3 references
        bool Editar(int id, DateTime data);
    }
}
```

Figura 10 Classe Interface IMetodosGenericos

3.2.2.10. Classe Program

Esta classe é onde executa o método principal para inicialização do programa, aqui podem ser criados as instâncias das classes e executar os métodos das mesmas.

```
// <copyright file="Program.cs" company="IPCA">
// Copyright (c) IPCA. All rights reserved.</copyright>
//-----
// <author>Yuri Lemos</author>
// <author>João Figueiredo</author>
// <desc> This program do the basics of C#</desc>
// <Date> 4 / 4 / 2020 </Date>
// <version>1.0</version>

using System;
using Pessoas; // Bibliotecas Externas
using Aparelhos;
using Defeitos;

namespace Auditorias
{
    class Program
    {
        static void Main(string[] args)
        {
            bool exit = true;
            int id; // Id para procura de qualquer objeto
            /**
             * Criação de instancias das classes Funcionario e Funcionarios
             */

            Funcionario f1 = new Funcionario("Abf2886392", "José Antônio", 123423, "Eng.Informático", new DateTime(2001, 08, 18));
            Funcionario f2 = new Funcionario("QUSD7282", "Pedro Henriques", 1878924, "Técnico Informático", new DateTime(2005, 07, 28));

            Equipamento e1 = new Equipamento("Laptop", "HP", "Omen", new DateTime(2019, 02, 12));
            Equipamento e2 = new Equipamento("Laptop", "Lenovo", "IdeaPad", new DateTime(2019, 9, 20));

            Vulnerabilidade v1 = new Vulnerabilidade(1, "Virus", "Elevado");
            Vulnerabilidade v2 = new Vulnerabilidade(2, "Trojan", "Elevado");

            Funcionarios listaFuncionarios = new Funcionarios();
            Equipamentos listaEquipamnetos = new Equipamentos();
        }
    }
}
```

Figura 11 Classe Program

4. Conclusão

4.1. Lições aprendidas

Os conhecimentos mais técnicos e teóricos sobre RGPD como algumas normas, legislações e como deve ser implementado ajudou muito a compreender e adquirir mais conhecimento sobre esta área que é a privacidade e proteção de dados pessoais. Isso ajudou muito a ter um olhar mais voltado ao mercado e à área de tecnologia com suas revoluções. Na esfera da prática, as habilidades e conhecimentos ao desenvolver o programa, revisando e implementando novos cálculos foram aperfeiçoadas.

4.2. Apreciação Final

A experiência propiciou a real prática das teorias vivenciadas em sala de aula, atribuindo mais gosto pelas disciplinas além de evoluir o raciocínio para adaptar propostas ao desenvolvimento.