Indice

[Relatório Trabalho Prático LP1 1](#_Toc8831459)

[Introdução 1](#_Toc8831460)

[Método de Trabalho adotado 2](#_Toc8831461)

[Estrutura de Ficheiros 2](#_Toc8831462)

[Diagrama de Objectos 3](#_Toc8831463)

[Classes 3](#_Toc8831464)

[Item 3](#_Toc8831465)

[Inventário 5](#_Toc8831466)

[Cliente 5](#_Toc8831467)

[Clientes 5](#_Toc8831468)

[Config 5](#_Toc8831469)

[Exceptions 6](#_Toc8831470)

[Bibliografia / Referências 6](#_Toc8831471)

# 

# Relatório Trabalho Prático LP1

A aplicação a ser desenvolvida no decorrer deste trabalho prático tem como finalidade a gestão de parques informáticos de uma ou várias entidades.

É a realização deste trabalho realizar uma abordagem de Aprendizagem de técnicas e métodos de Programação Orientada por Objectos,  
assim como a utilização de ferramentes e métodos de trabalho colaborativo.

## Introdução

A escolha do tema a desenvolver no trabalho prende-se com a necessidade cada vez maior das empresas de  
desenvolverem metodologias para o controle do seu espólio material e virtual.

* A aplicação deverá:
  + Ser abstrata o suficiente para a reutilização de codigo para outros setores/aplicações.
  + Ter uma visão Macro e Micro de todo o inventário presente na Entidade.
    - Listagem de equipamentos
    - Listagem por tipo de Equipamentos
    - Listagem por ano de fabrico
    - Listagem por fabricante
    - …
  + Adição/Remoção/Edição de Items no Inventário

Foi pensado em criar um Interface de utilizador por Windows/Consola no entanto ainda será um grande desafio aprender WPF

Os resultados a obter com o desenvolvimento desta solução é criação de uma aplicação versátil e adaptativa às  
necessidades das diferentes entidades e deverá potencializar uma melhor organização do parque informáticos.

## Método de Trabalho adotado

Para o desenvolvimento deste trabalho, utilizamos uma metodologia que permite  
a todos os membros do grupo desenvolvam simultaneamente a aplicação e que haja um  
controle de versões desenvolvidas através do Git.

Permite ainda dividir a carga de trabalho pois permite que cada elemento do grupo  
desenvolva uma classe específica da aplicação.

## Estrutura de Ficheiros

* **./ITGEstao/**
  + Projecto (VS 2017) de um dll ITGestao.dll que pode ser reutilizada
* **./Tests/**
  + Projecto de testes unitários
* **./UI.Cli/**
  + Interface de Utilizador (Linha de Comandos)
* **./UI.Win/**
  + Interface Utilizador (Windows)
* **./Utils/**
  + Vários Utilitários a serem usados
* **ITgestao.sln**
  + Solução (VS 2017)
* **diagram.png**
  + Diagrama de classes do projecto ITGestao
* [**readme.md**](http://readme.md)
  + Relatório
* **LICENSE**
  + Licença de Utilização

## Diagrama de Objectos

Diagrama de Classes

## 

## Classes

* Item
  + ClasseName:Item (*Extensões de Item*)
  + Generico:Item
  + Rede:Item
  + Computador:Item
  + …
* Inventario
* Localizacao
* Localizacoes
* Cliente
* Clientes
* Config
* Exceptions

### Item

A classe Item é a classe abstrata que define e cria os objetos do tipo Item.  
É a classe “pai” responsável por armazenar e tratar um objeto do tipo Item o mais genérico possível.  
Esta classe têm dois métodos **static** muito importantes de forma a que qualquer filho possa informar a classe base  
que existe mais um tipo de child.

1. Atributos
   * (***id***) - Valor único que identifica o item (id do item);
   * (***idCliente***) - Valor identifica o cliente ao qual pertence o item;
   * (***idLocalizacao***) - Valor que identifica a localização do item.
2. Métodos
   * (***AddAuthorizedType(object \_obj)***) - Permite adicionar um novo tipo de item autorizado;
     + Este metodo utiliza a classe ***Config*** para armazenar na aplicação transversalmente o estado dos items autorizados.
   * (***Type AuthorizedType(object \_obj)***) - Devolve o tipo do objecto colocado como parametro se fôr um objecto autorizado.
     + Este método utiliza a classe ***Config*** para verificar se o objecto está na lista dos autorizados.

#### Extensões de Item (ClasseName:Item)

Todas as classes que extendam de Item devem informar a classe Item que existe um novo tipo de Item que será tratado pelo inventário.

Esta tipo de objetos deverão ser sealed de forma a que não hajam mais extensões do mesmo, a menos que para tal seja necessário extender.

public ClasseName(...) : base(...)  
 {  
 Item.AddAuthorizedType(this);

#### Genérico (Generico:Item)

É um objecto que extenderá ***Item***.

É a classe que trata da informação de um item genérico.

#### Computador (Computador:Item)

É um objecto que extenderá ***Item***.

Esta classe irá criar items do tipo Computador.

Ainda não está definido na altura do desenvolvimento deste relatório os atributos, deverão ser do tipo: RAM/CPU…

1. **Métodos/Construtores**
   * (***Computador(…)***) - Construtor que cria um objeto do tipo Computador.

#### Rede (Rede:Item)

É um objecto que extenderá ***Item***.

Esta classe irá criar items do tipo Rede.

Ainda não está definido na altura do desenvolvimento deste relatório os atributos, deverão ser do tipo: ENDERECO/TIPO(Switch/router)…

1. **Métodos/Construtores**
   * (***Rede(…)***) - Construtor que cria um objeto do tipo Rede.

### Inventário

O Objeto Inventário está encarregue de armazenar e tratar objetos do tipo ***Item***.

1. **Atributos**
   * (***empresa***) - Valor que identifica uma empresa da instancia do inventário;
   * (***itens***) - Lista de objetos do tipo Item;
2. **Métodos**
   * ***Adiciona(…)*** - Método que adiciona um objeto ao inventário;
   * ***Remove(…)*** - Método que elimina um objeto do inventário;
   * ***Edita(…)*** - Método que edita um objeto do inventário

### Cliente

O Objecto Cliente define e cria os objetos do tipo Cliente.

Os atributos deste objeto ainda não foram desenvolvidos na totalidade.

1. **Atributos**
   * (***id***) - Valor único de identifica um cliente;
   * (***nome***) - Primeiro nome do cliente;
   * (***apelido***) - Apelido do cliente.

### Clientes

A classe Clientes é a classe que encarregue de armazenar objetos do tipo ***Cliente***.

1. **Atributos**
   * clientes- Lista de objetos do tipo Cliente;
2. **Métodos**
   * ***Adiciona(…)*** - Método que adiciona um cliente à lista;
   * ***Remove(…)*** - Método que elimina um cliente da lista;
   * ***Edita(…)*** - Método que edita um cliente da lista;

### Config

O Objectivo desta classe é armazenar informação necessária para o runtime da aplicação de forma a que qualquer  
classe que necessite de informação sobre a aplicação posso consumir informação da mesma.

Algumas das informações pensadas a colocar nesta classe futuramente.

* Directorio Actual
* Directorio de Armazenamento dos ficheiros de dados persistentes.
* Tipos de item autorizados pela aplicação ITGestao.dll a serem tratados pelo **Inventário**

Optou-se pela utilização do Padrão Singleton nesta classe de forma que só exista uma só  
instancia da classe config sendo que a mesma é inicializada na primeira vez que é chamada.

public sealed class Config {  
 private Config()  
 {  
 // Código a correr na inicialização da classe  
 (...)  
 }  
 private static Config instance = null;  
 public static Config Instance  
 {  
 get  
 {  
 // A classe não está instanciada  
 if (instance == null)  
 instance = new Config();  
 // Se a classe está instanciada retorna a instancia.  
 return instance;  
 }  
 }  
(...)  
}

### Exceptions

Classe que permite lidar com erros que ocorrem durante a execução da aplicação.

Algumas das exceções já tratadas:

* ID duplicado;
* ID inválido;
* Tentativa de iniciação de objeto com argumentos em falta.

## Bibliografia / Referências

* [Padrão Singleton](https://pt.wikipedia.org/wiki/Singleton)
* [Testes Unitários - Unit Testing](https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019)
* [Windows Presentation Foundation](https://docs.microsoft.com/en-us/dotnet/framework/wpf/)