

# Relatório Trabalho Prático LP1

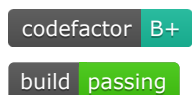
---

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 ferramentas e métodos de trabalho colaborativo.

## Autores do Trabalho:

- **Daniel Torres** ([a17442@alunos.ipca.pt](mailto:a17442@alunos.ipca.pt))
- **Oscar Silva** ([a14383@alunos.ipca.pt](mailto:a14383@alunos.ipca.pt))



## 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 abstracta o suficiente para a reutilização de código para outros sectores/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

Não tendo sido possível a aprendizagem de WPF e ao aprofundamento de outras técnicas à frente faladas

o projecto UI.Win apenas foi deixado como prova de conceito, assim como não foi muito desenvolvido o UI.Cli.

Foi primordialmente focado o desenvolvimento de capacidades de trabalho em grupo e automatizadas, assim como a funcionalidade de uma livreria (DLL) que poderá ser reutilizada em outros contextos.

No entanto todos os métodos e classes foram testados usando testes unitários que podem ser vistos no projecto de Tests.

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 adoptado

---

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.

O GitHub em conjunto com outras ferramentas que serão abordadas mais à frente no tópico de Ferramentas/Serviços de Desenvolvimento trazem uma enorme mais valia à produção de Software.

Algumas ferramentas utilizadas:

- Travis-ci
  - Travis CI é um serviço de integração contínua
- Codefactor
  - Analisador automático de syntax de código

## Estrutura de Ficheiros

---

- **./Libs/Folder/...**
  - Pasta que contem exemplos de items, esta pasta irá gerar dll's a serem utilizadas pelo ITGestão
- **./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)
    - Apenas serve como prova de conceito da forma que foi dividido o UI da parte Lógica
- **./Utils/**
  - Vários Utilitários a serem usados
- **ITgestao.sln**
  - Solução (VS 2017)
- **diagram.png**
  - Diagrama de classes do projecto ITGestao
- [readme.md](#)
  - Relatório
- **.travis.yml**
  - Ficheiro de compilação automática e Testes GIT
- **LICENSE**
  - Licença de Utilização

## Ferramentas / Serviços utilizados

---

### Travis-CI

Em resumo o *Travis-CI* é um serviço no qual nos é permitido correr uma instância de software (ex docker) de forma a que sejam feitos testes de compilação e testes unitários à solução.

Travis-CI (Continuous Improvement) é um serviço de melhoria continua de código e análise do mesmo.

Para a utilização deste serviço com C# tivemos de mudar a framework de testes de [MSTest](#) para a ferramenta [NUnit Framework - Unit Testing](#) pois a execução de testes em ambiente Linux/MacOsx é incompatível com a ferramenta MSTest.

### Ficheiro de Configuração .travis.yml

```
language: csharp
solution: ITgestao.sln
```

```
install:
- sudo apt-get install nunit-console
- nuget restore ITgestao.sln
- nuget install NUnit.framework
- nuget install NUnit
- nuget install NUnit3TestAdapter
- nuget install NUnit.Runners -Version 3.7.0 -OutputDirectory testrunner

script:
- rm -rf ./UI.Win
- xbuild /p:Configuration=Release ITgestao.sln
- mono ./testrunner/NUnit.ConsoleRunner.3.7.0/tools/nunit3-console.exe ./Tests/bin/
```

A configuração do *Travis-CI* é feita através de um ficheiro que é deixado na raiz do projecto (**.travis.yml**) de forma a informar ao agent do *Travis-CI* quais as configurações necessárias para correr/testar a nossa solução.

O ficheiro de configuração do nosso trabalho utiliza Linux e mono para compilar e testar a aplicação, de notar que de momento não é possível realizar testes em WPF de forma que o subprojecto WPF é removido do travis antes de iniciar a execução.

O *Travis-CI* é capaz de correr os mais variados ambientes quer de sistemas operativos quer de pacotes já feitos [MultiOS](#) / [Docker](#).

### Funcionamento:




O *Travis-CI* efectua testes de forma configurável pelo administrador do repositório.

Preferimos utilizar o *Travis-CI* para confirmar os pull requests para o branch master e para testar a compilação do branch master.

Desta forma sempre que for efectuado um pull request ao master do repositório o *Travis-CI* automaticamente inicia os testes como podem ver na imagem abaixo:


## Some checks haven't completed yet

1 queued, 1 in progress, and 1 successful checks

●		Travis CI - Pull Request	Queued — Build ...	<a href="#">Details</a>
●		Travis CI - Branch	In progress — Build Star...	<a href="#">Details</a>
✓		CodeFactor	— 1 issue fixed.	<a href="#">Details</a>


Após a finalização dos testes é dado o “**ok**” para se efectuar o merge no repositório, cabe sempre a quem efectua a manutenção do repositório a decisão final se quer efectuar um merge que falhou à sua responsabilidade.

Add more commits by pushing to the **oscar** branch on **nargotik/trabalho1-lp2**.




✓ All checks have passed  
3 successful checks

[Hide all checks](#)

✓  CodeFactor


— 1 issue fixed.

[Details](#)

✓  Travis CI - Branch

Successful in 8m — Build Passed

[Details](#)

✓  Travis CI - Pull Request

Successful in 8m — Build Passed

[Details](#)

✓ This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

No dashboard do *Travis-CI* podemos ver o log assim como o tempo utilizado nos testes:

✓ oscar	Insert correction on UI.cli	→ #61 passed	⌚ 8 min 59 sec	Ⓢ
○ oscardsilva1998		→ c73a057 ↗	📅 5 minutes ago	

Log dos testes:

```
$ mono ./testrunner/NUnit.ConsoleRunner.3.7.0/tools/nunit3-console.exe ./Tests/bin/Release/Tests.dll
NUnit Console Runner 3.7.0
Copyright (c) 2017 Charlie Poole, Rob Prouse

Runtime Environment
  OS Version: Linux 4.4.0.101
  CLR Version: 4.0.30319.42000

Test Files
  ./Tests/bin/Release/Tests.dll

Run Settings
  DisposeRunners: True
  WorkDirectory: /home/travis/build/nargotik/trabalho1-lp2
  ImageRuntimeVersion: 4.0.30319
  ImageTargetFrameworkName: .NETFramework,Version=v4.6.1
  ImageRequiresX86: False
  ImageRequiresDefaultAppDomainAssemblyResolver: False
  NumberOfTestWorkers: 2

Test Run Summary
  Overall result: Passed
  Test Count: 19, Passed: 19, Failed: 0, Warnings: 0, Inconclusive: 0, Skipped: 0
  Start time: 2019-05-29 09:23:53Z
  End time: 2019-05-29 09:24:10Z
  Duration: 16.862 seconds
```

Log com exemplo de falhas nos testes.

```
9) Failed : Tests.ItemsTests.GenericItemDelfromInventory
  Expected: True
  But was: False
  at Tests.ItemsTests.GenericItemDelfromInventory () [0x00000] in <45b8909ab3b64df187ab06bb260a0196>:0

10) Failed : Tests.ItemsTests.GenericItemDelfromInventoryById
  Expected: True
  But was: False
  at Tests.ItemsTests.GenericItemDelfromInventoryById () [0x00000] in <45b8909ab3b64df187ab06bb260a0196>:0

Run Settings
  DisposeRunners: True
  WorkDirectory: /home/travis/build/nargotik/trabalho1-lp2
  ImageRuntimeVersion: 4.0.30319
  ImageTargetFrameworkName: .NETFramework,Version=v4.6.1
  ImageRequiresX86: False
  ImageRequiresDefaultAppDomainAssemblyResolver: False
  NumberOfTestWorkers: 2

Test Run Summary
  Overall result: Failed
  Test Count: 19 Passed: 9, Failed: 10, Warnings: 0, Inconclusive: 0, Skipped: 0
  Failed Tests - Failures: 8, Errors: 2, Invalid: 0
  Start time: 2019-05-29 10:57:26Z
  End time: 2019-05-29 10:57:27Z
  Duration: 1.132 seconds

Results (nunit3) saved as TestResult.xml
The command "mono ./testrunner/NUnit.ConsoleRunner.3.7.0/tools/nunit3-console.exe ./Tests/bin/Release/Tests.dll" exited with 10.
```

O *Travis-CI* fornece ainda uma pequena imagem dinâmica que está presente na introdução do nosso trabalho que indica se o projecto está com os testes em “passing” ou “fail”.

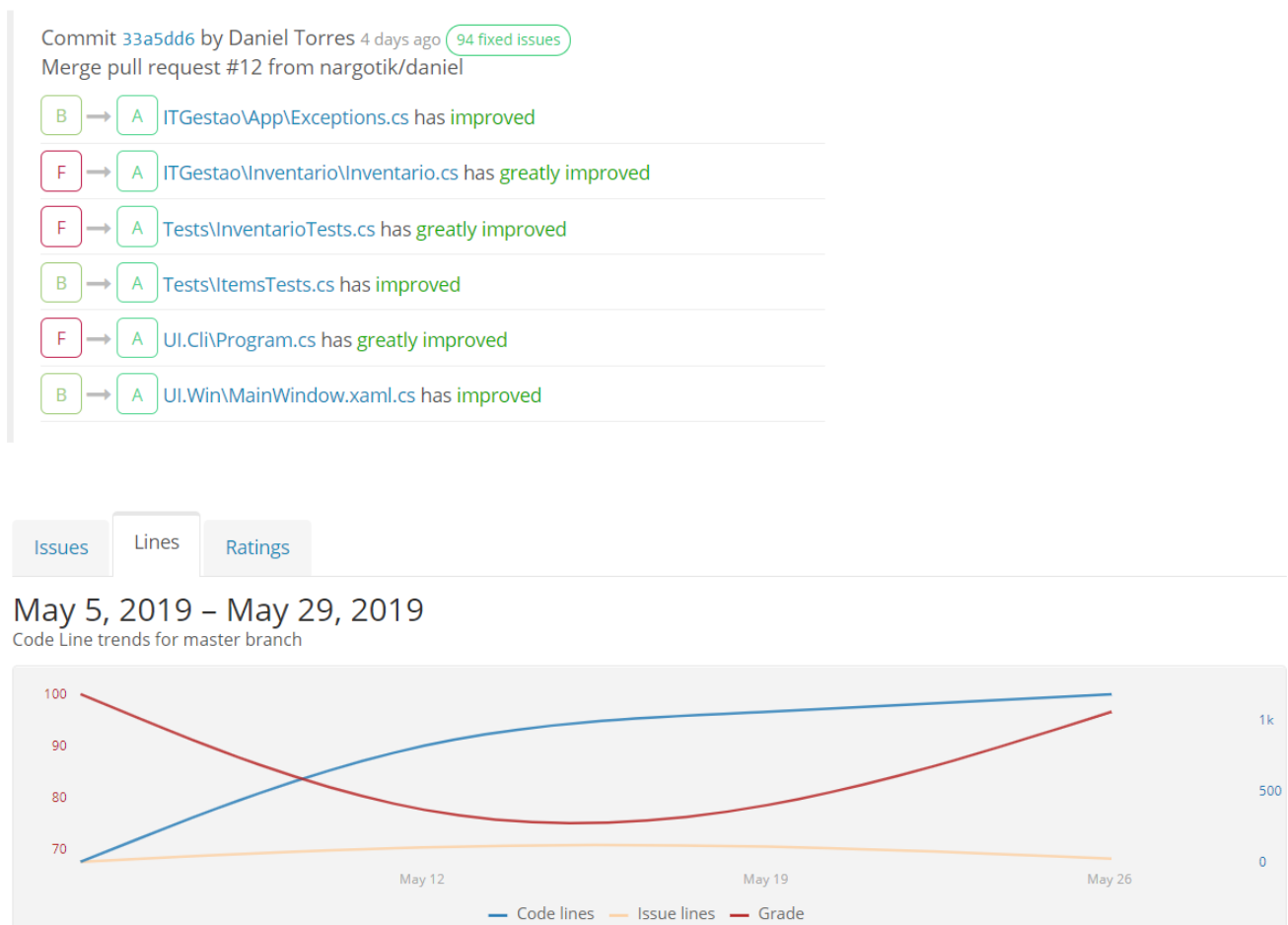
## CodeFactor

O CodeFactor é um serviço de análise de syntax de código de forma a que sejam cumpridas algumas regras básicas de realização de código.

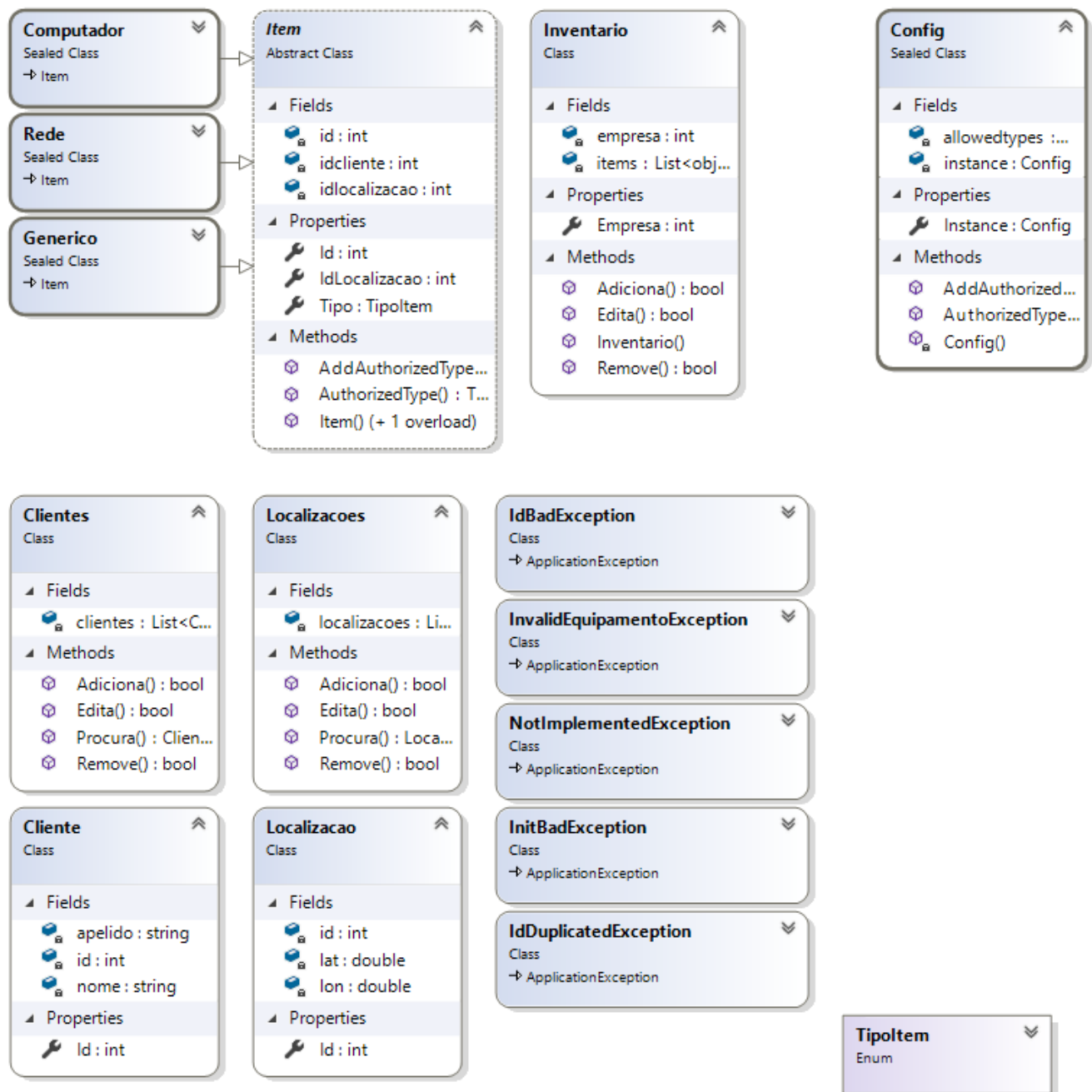
Este serviço é importante, principalmente se quisermos manter um projecto com muitos programadores onde uniformizamos a forma como é colocado o código no projecto, garantindo assim desta forma homogeneidade dessas regras básicas de código.

Existem muitos outros serviços deste género, no entanto verificamos que este era de momento o que mais se enquadrava para a linguagem c#.

Este tipo de testes pode também ser efectuado de forma manual no Travis-CI, no entanto seria necessário fazer testes bem mais complexos que os unitários para testar problemas comuns de syntax.



## Diagrama de Objectos



## Classes

- Item
  - `ClassName:Item` (*Extensões de Item*)
  - `Generico:Item`
  - `Rede:Item`
  - `Computador:Item`
  - ...
- Inventario
- Localizacao
- Localizacoes
- Config



- Exceptions
  - Menu
  - MenuOpcao
- 

## Item

A classe Item é a classe abstracta que define e cria os objectos do tipo Item.

É a classe “pai” responsável por armazenar e tratar um objecto 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 método 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 parâmetro se for um objecto autorizado.
  - Este método utiliza a classe **Config** para verificar se o objecto está na lista dos autorizados.

---

## Extensões de Item (ClassName:Item)

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

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

```
public ClassName(...) : base(...)  
{  
    Item.AddAuthorizedType(this);  
}
```

---

## Genérico (Generico:Item)

É um objecto que estenderá **Item**.

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

---

## Computador (Computador:Item)

É um objecto que estenderá **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: ENDEREÇO/TIPO(Switch/router)...

### 1. Métodos/Construtores

- (**Rede(...)**) - Construtor que cria um objeto do tipo Rede.
- 

## Inventário

O Objecto Inventário está encarregue de armazenar e tratar objectos do tipo **Item**.

Foi utilizada padrão singleton com múltiplas instancias pelo id da entidade de inventário.

Para utilizar a class apenas é necessário:

```
// Instancia com entidade = 0 por defeito
Inventario.getInstance().metodo;
...
// Instancia com entidade = 10
Inventario.getInstance(10).metodo;
```

De salientar que o armazenamento dos dados é feito automaticamente quando se adiciona/edita/remove um item com os métodos privados **SaveData()** e **LoadData()**.

O inventário foi desenvolvido de forma a poder ser reutilizado nas mais variadas áreas, sendo a abordagem feita o mais abstracta possível de forma a que o mesmo tanto armazene computadores como fruta ou viaturas.

### 1. Atributos

- (**entidade**) - Valor que identifica uma entidade da instancia do inventário;
- (**itens**) - Lista de objectos do tipo Item;

### 2. Métodos

- **Adiciona(...)** - Método que adiciona um objecto ao inventário;
- **Remove(...)** - Método que elimina um objecto do inventário;
- **GetItemById(...)** - Método que devolve um objecto do inventário pelo ID
- **RemoveItemById(...)** - Método que remove um objecto do inventário pelo ID
- **RemoveAll(...)** - Método que remove todos os itens do inventário
- **Edita(...)** - Método que edita um objecto do inventário

### 3. Testes Unitários

- Foram desenvolvidos vários testes unitários básicos unitários de forma a que qualquer modificação na classe obedeça às regras do desenvolvimento.

---

## 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 possa consumir informação da mesma.

Algumas das informações pensadas a colocar nesta classe futuramente.

- Directório Actual
- Directório 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 excepções já tratadas:

- ID duplicado;
- ID inválido;
- Tentativa de iniciação de objecto com argumentos em falta.

---

## Menu

Foi criada uma classe de forma a podermos agilizar os menus no UI.Cli

Para tal utilizamos também Delegates de forma a que uma opção de um Menu tivesse um acção (Action) atribuída (Callback).

## MenuOpcao

É uma classe que armazena opções do menu e respectivos Actions

## Bibliografia / Referências

---

- [Padrão Singleton](#)
- [Testes Unitários - Unit Testing](#)
- [Windows Presentation Foundation](#)
- [CodeFactor](#)
- [Travis-ci](#)
- [NUnit Framework - Unit Testing](#)
- [GitHub](#)
- [DEVHINTS.IO](#)
- [Docker Hub](#)
- [AppVeyor](#)
- [CoverAlls.IO](#)