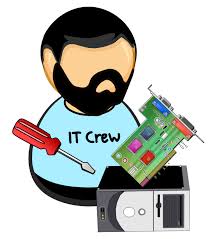
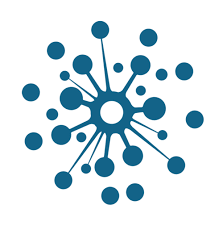
Relatório do Projeto Final

Gestão Incidentes



Índice

[Introdução 2](#_Toc483000508)

[Objetivo 2](#_Toc483000509)

[Glossário 2](#_Toc483000510)

[Desenvolvimento 3](#_Toc483000511)

[Arquitetura do Programa 3](#_Toc483000512)

[Classes 3](#_Toc483000513)

[Estrutura da Classe BDTicktes 3](#_Toc483000514)

[Diagrama de Classes 4](#_Toc483000515)

[Diagrama de Heranças 5](#_Toc483000516)

[Base de Dados 6](#_Toc483000517)

[Exemplo da Normalização: 6](#_Toc483000518)

[Diagrama do Modelo Relacional 7](#_Toc483000519)

[Conclusão 8](#_Toc483000520)

# Introdução

Objetivo

Este projeto tem como objetivo desenvolver uma aplicação á medida para a resolução de incidentes de IT.

Glossário

O Projeto envolve um conjunto de conceitos e termos que podem não ser do conhecimento geral.

1. Incidente – Um Serviço, Equipamento ou Programa não está a funcionar ou não está a funcionar ao nível esperado.
2. Colaborador – Funcionários da empresa que reportam os incidentes á equipe de IT Support.
3. Técnico – Funcionários da empresa que resolvem incidentes.
4. Ticket – Processo de tratamento do incidente, é criado por um colaborador e resolvido por um técnico, um ticket tem estados de resolução, uns são finais outros não.
5. Requisição de material – Um Ticket pode necessitar de equipamento, software ou serviços adicionais que a equipe de suporte não pode disponibilizar diretamente, a requisição serve para registar essas necessidades adicionais, cada requisição só suporta um item, pelo que será necessária uma requisição por item, Ex: um PC tem duas peças avariadas, serão necessárias duas requisições. Este processo poderá ter uma evolução futura para registar mais que um item, mas só será implementado se for verificada a necessidade por parte do departamento.
6. Prioridade, definida pelo utilizador, serve para organizar os incidentes por prioridade, permitindo uma melhor gestão do tempo dos técnicos e dos Incidentes, visto que uma maior prioridade vai determinar o seu tempo de execução.
7. Processo humano, qualquer programa implica um conjunto de regras de utilização, neste programa o processo humano é importante para que o sucesso da aplicação, determinado pela organização que o usa, o processo humano deve acompanhar as regras de utilização da interface da aplicação. Ex: cada requisição só lida com um item, ora se no campo da descrição forem colocados mais que um item, obviamente a aplicação não se vai queixar, mas deve o processo humano intervir aí e determinar que essa utilização do programa não é aconselhada.
8. Inventário – Todos os equipamentos da empresa devem estar registados no inventário, isso ajuda o programa determinar o tipo de intervenção, no registo dos equipamentos deve ser o mais detalhado possível e todos os campos devem ser preenchidos, os equipamentos têm peças. Ex: um computador pode ser registado com um único numero ou ser dividido em caixa do PC, monitor, programas instalados, inclusive mesmo o teclado e o rato podem ser registados em separado, deixando de ser peças de um PC para passar a ser um equipamento.

# Desenvolvimento

Arquitetura do Programa

O Programa é desenvolvido em C# com uma base de dados Local em SQL Server, pode ser usado num computador local ligado a um servidor de Base de Dados local, ou ligado a um servidor de base de dados remoto.

Devido ás exigências do cliente expressas ao analista este programa não permite usar outro tipo de bases de dados, no entanto a sua arquitetura permite a migração para outro tipo de bases de dados com pouca alteração de código, que será explicado nas escolhas feitas para as classes.

O programa tenta observar o melhor possível as regras de Programação orientada por objetos, OOP, a reutilização de código é uma regra constante e tenta-se evitar a redundância ao máximo.

Classes

Este Programa tem um Conjunto de Classes que expõe a base de dados à Interface Gráfica, essas classes estão divididas em classes de Entidades e Classe de Interface, não confundir esta interface com a interface gráfica, esta Classe de Interface é a responsável pela comunicação entre a base de dados e o programa, chamou-se a esta Classe BDTickes, será feita uma descrição mais detalhada no Diagrama de Classes

As Entidades identificadas foram: Tickets, Perfis, MaterialRequests, Equipamento, Material, Status, Habilitacao, Colaborador, Tecnico, SuporteEquipamento, SuporteSoftware, SuporteRede, Area.

Estas classes definem as varias entidades da aplicação e permitem a interação entre elas através da classe BDTicktes.

A Classe BDTicktes herda o Interface aplicacional IBDTickets, que permite a expansão futura da aplicação para uma solução com diferentes interfaces Gráficos, conforme a necessidade.

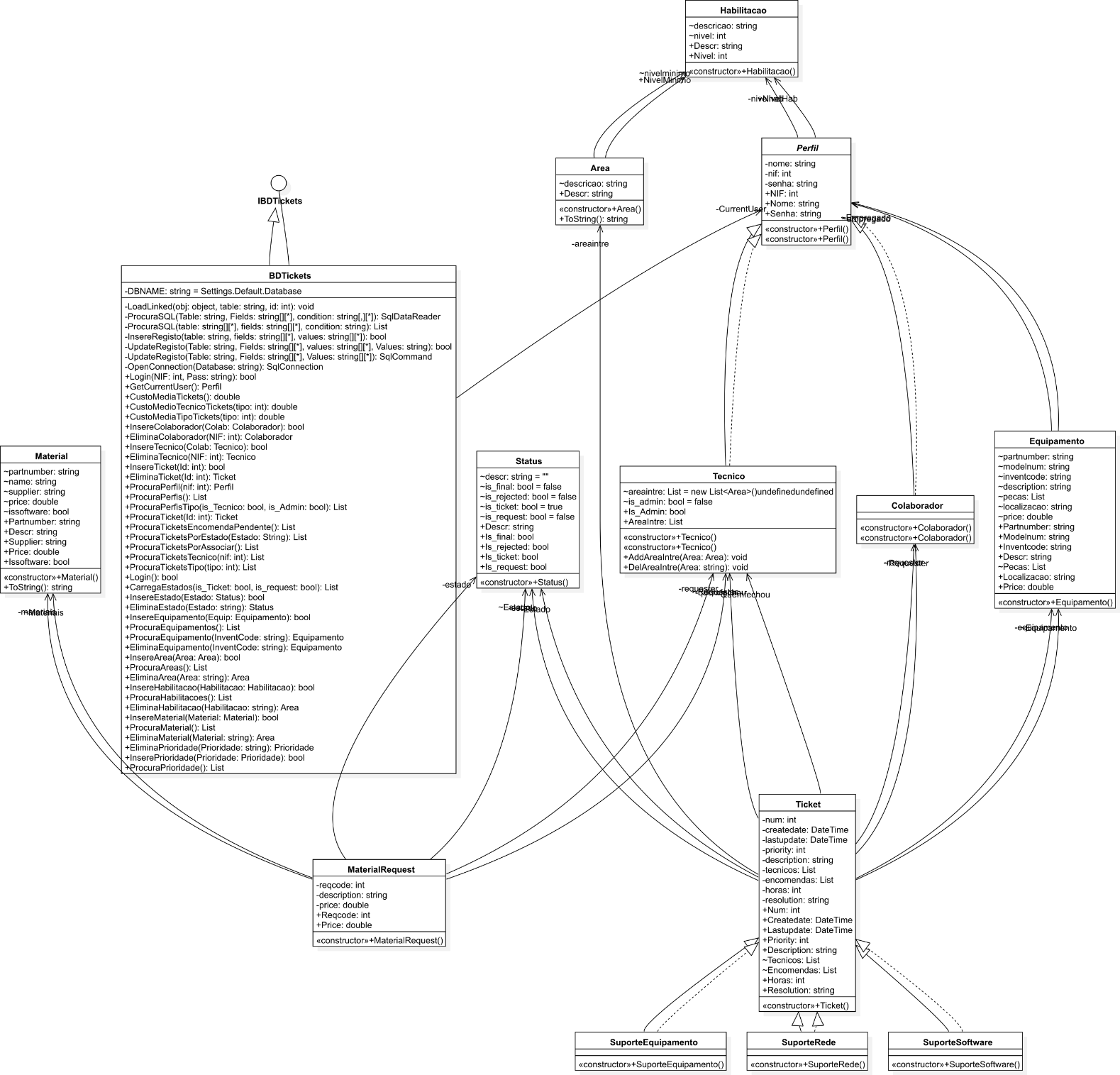
Estrutura da Classe BDTicktes

Como já deve ter percebido a classe BDTicktes é a principal do programa, estando previsto na segunda Fase do Projeto a implementação desta classe como um serviço de Windows para poder servir clientes remotos.

Esta classe tem em si uma arquitetura por camadas, os métodos privados lidam diretamente com a base de dados de uma maneira genérica, ficando a decisão de qual a tabela ou tabelas usadas e os respetivos campos do lado das classes publicas.

Os métodos privados usam arrays de strings para carregar os dados e os campos e mesmo as condições necessárias, isto reduz o tamanho do código e aplica a prática de reutilização de código.

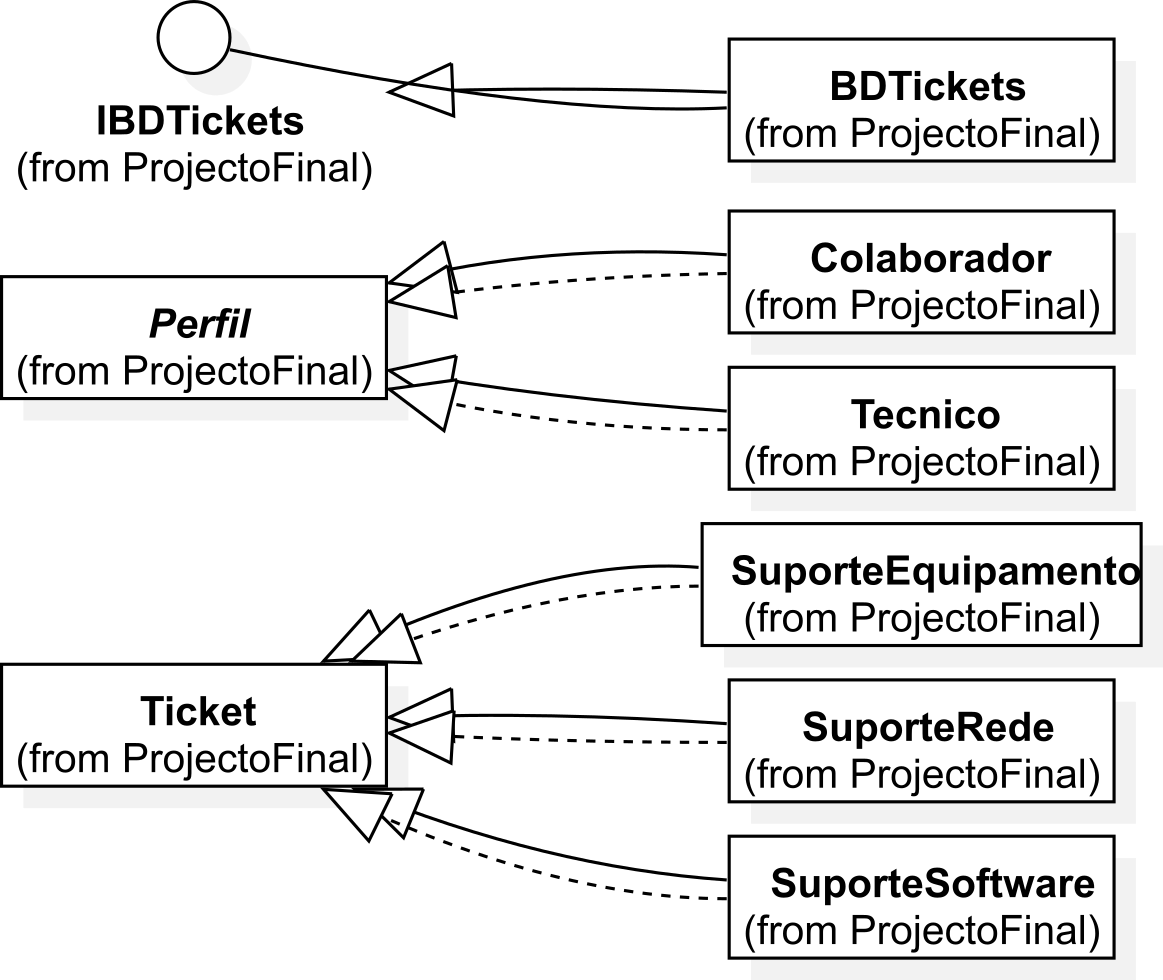
As chamadas de SQL usam todas SQL Parameters, conforme as boas práticas de desenvolvimento de SGBDS de modo a evitar problemas de [SQL Injection](https://www.w3schools.com/sql/sql_injection.asp).

Diagrama de Classes

Como já havia descrito a Classe BD Tickets é a principal classe da aplicação, no entanto a classe Ticket que serve principalmente como estrutura de objetos, é a classe que tem uma relação mais profunda com as outras classes de objetos.

Devido ao facto de uma boa parte do código ainda não estar escrito há várias relações entre classes que não estão definidas.

Diagrama de Heranças



O esquema de heranças que foi Definido Inicialmente pode não chegar a ser usado na fase dois do projeto mas será implementado na primeira fase como “Proof of Concept”

Base de Dados

A base de dados foi estruturada conforme o modelo relacional, respeitando as três principais regras da normalização de classes.

Exemplo da Normalização:



Os dados Brutos que queríamos guardar não estão de todo utilizáveis nem normalizados.



Após a utilização da primeira Forma Normal concluímos que os dados provocam repetições pelo que é necessário aplicar a 2ª e a 3ª forma Normal

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tabela Perfil 1 - 3FN | | | | | | |
| **NIF** | **nome** | **senha** | **Habilitação** | **Nivel** | **Cargo** | **Areas** |
| 1 | nuno | \*\*\*\*\*\* | Curso Especialista de Redes | 5 | 1 | 1,2,3 |
| 2 | manuel | \*\*\*\*\* | Curso Superior Informática | 6 | 2 | 2 |
| 3 | Luisa | \*\*\*\*\* | Mestrado em Psicologia aplicada aos Ambientes de Trabalho | 8 | 3 | 3 |

Após a 2ª e a 3ª Forma Normal, os dados já parecem estar mais normalizados usando-se índices em vez de referencias em texto para se estruturar os dados que não são dependentes do índice da tabela perfil, as tabelas auxiliares do cargo e das áreas fazem parte da normalização:

|  |  |
| --- | --- |
| **Cargos** | ID |
| Tecnico | 1 |
| Admin | 2 |
| Colaborador | 3 |

|  |  |
| --- | --- |
| **Areas de Intrevenção** | ID |
| Software | 1 |
| Hardware | 2 |
| Redes | 3 |

No entanto a normalização ainda não está concluída pois identificou-se mais um conjunto de dados que não dependem do índice da tabela perfil, formam asim um grupo de dados á parte, estou a falar do nível de Escolaridade e as Habilitações, que foram, na estrutura de dados final da base de dados, passadas para uma tabela própria.

Diagrama do Modelo Relacional



Como se pode ver a Tabela Perfis é a única que guarda dados de utilizadores da aplicação, tem três campos Is\_qualquercoisa, estes campos são do tipo BIT para poderem ser usados como boolean em C#, em vez de ter mais uma tabela para validar o tipo de utilizador, visto que estes tipos são pré-definidos e não vão mudar, poupa-se espaço em disco e facilita a logica do programa, visto não ser preciso mais joins para chegar ao mesmo resultado.

A Tabela AreaIntre, EquipPart e a TicketTec existem para criar relações Manny to Manny entre as tabelas a que estão ligadas.

Existe o caso especial da tabela States que define estados tanto para os Tickets como para os MatRequests, visto que estes estados tem uma parametrização igual, pode haver estados repetidos que deste modo são partilhados pelas duas tabelas, ficando ao código da aplicação decidir o que fazer com a parametrização da tabela States.

# Conclusão

O Projeto ainda está na fase de análise de Requisitos, pelo que pode sofrer alterações tanto a nível de estrutura de dados como de classes.

O Manual de Utilizador será apresentado após a conclusão da aplicação, num documento á parte.

Pretende-se com este projeto demostrar o poder do desenvolvimento OPP usando a Hereditariedade e o Polimorfismo como grandes armas para o rápido desenvolvimento da aplicação.

Esta aplicação vai permitir a Gestão de Incidentes, do Inventário e das Requisições associadas aos Tickets de Incidentes.

A Aplicação terá numa Segunda Fase uma interface gráfica só para colaboradores remotos, essa aplicação usará o protocolo TCP/IP e dados encriptados para transmitir e receber informações com a aplicação servidora.