RELATÓRIO

**Gestão de Clínicas de Informáticas**



Trabalho realizado por:

**Filipe Fonseca Nº108005**

**Emanuel Vitorino Nº107981**

**Turma 351**

Índice

Conteúdo

[Índice de Figuras 2](#_Toc391978091)

[Resumo 4](#_Toc391978092)

[Notação e Glossário 5](#_Toc391978093)

[Organização do Relatório 6](#_Toc391978094)

[1ºParte 6](#_Toc391978095)

[2ºParte 6](#_Toc391978096)

[3ºParte 6](#_Toc391978097)

[Introdução 7](#_Toc391978098)

[Proposta do Projeto 10](#_Toc391978099)

[Tecnologias e Recursos Utilizados 12](#_Toc391978100)

[Realização do Projeto 13](#_Toc391978101)

[1- Planeamento, Proposta e pontos de situação 13](#_Toc391978102)

[I. Tema do projeto 13](#_Toc391978103)

[II. Objetivos 13](#_Toc391978104)

[III. Planeamento das fases de desenvolvimento 14](#_Toc391978105)

[IV. Proposta 14](#_Toc391978106)

[V. Pontos de situação 14](#_Toc391978107)

[2- Análise 15](#_Toc391978108)

[I. Planeamento do modelo de dados 15](#_Toc391978109)

[3- Implementação 18](#_Toc391978110)

[I. Implementação da Interface 18](#_Toc391978111)

[II. Implementação das classes DAL e BLL 21](#_Toc391978112)

[III. Implementação da Autenticação 23](#_Toc391978113)

[Versões 25](#_Toc391978114)

[0.1.0.0 25](#_Toc391978115)

[0.3.2.0 25](#_Toc391978116)

[0.5.4.0 25](#_Toc391978117)

[0.7.3.0 25](#_Toc391978118)

[1.0.0.0 25](#_Toc391978119)

[Conclusões 26](#_Toc391978120)

[Dificuldades 27](#_Toc391978121)

[Agradecimentos 28](#_Toc391978122)

[Apreciação Final 29](#_Toc391978123)

# Índice de Figuras

[Figura 1 - Cronograma 11](file:///C:\Users\108005\Documents\GitHub\ClinicaInformatica\Documentos%20PT\Relatório%20PT.docx#_Toc391975406)

[Figura 2 - Modelo de Dados E.A do nosso projeto 17](file:///C:\Users\108005\Documents\GitHub\ClinicaInformatica\Documentos%20PT\Relatório%20PT.docx#_Toc391975407)

[Figura 3 - Interface Principal MDI Parent do nosso Projeto 18](file:///C:\Users\108005\Documents\GitHub\ClinicaInformatica\Documentos%20PT\Relatório%20PT.docx#_Toc391975408)

[Figura 4 - ViewForm ou Janela de Visualização de Dados - MDIChild 19](#_Toc391975409)

[Figura 5 - Janela de Operações dos Clientes - MDIChild 20](#_Toc391975410)

[Figura 6 Figura ilustrativa sobre a BLL, DAL e a Base-de-Dados 22](file:///C:\Users\108005\Documents\GitHub\ClinicaInformatica\Documentos%20PT\Relatório%20PT.docx#_Toc391975411)

[Figura 7 - A BLL do nosso projeto 23](file:///C:\Users\108005\Documents\GitHub\ClinicaInformatica\Documentos%20PT\Relatório%20PT.docx#_Toc391975412)

[Figura 8 - LoginForm ou janela de Início de Sessão - MDIChild 24](file:///C:\Users\108005\Documents\GitHub\ClinicaInformatica\Documentos%20PT\Relatório%20PT.docx#_Toc391975413)

# Resumo

No 2º ano do curso de TGPSI, todos os alunos devem realizar um projeto representativo das competências adquiridas.

Foi pedido aos alunos a realização de um projeto no âmbito da disciplina de Programação de Sistemas Informáticos, com tema livre no qual seja implementado uma aplicação em vb.net. Todo o projeto deveria apresentar a complexidade necessária para abranger grande parte dos conteúdos lecionados. Entre eles, o desenvolvimento de uma base de dados em Microsoft SQL *Server*.

Decidimos, para o nosso Projeto Tecnológico do ano letivo 2013/2014, desenvolver uma aplicação para gestão de uma clínica informática.

A razão da nossa escolha foi devido a uma proposta pelo nosso professor de Sistema Operativos, para realizar o projeto da clínica, e aceitámos, dado que seria uma aplicação que pode ser usada numa empresa. Como achámos que seria uma boa oportunidade, nós aproveitámos para ser o nosso projeto.

# Notação e Glossário

**VB.NET** – *Visual Basic* na plataforma .*net framework*;

**SQL** – *Structured Query Language*;

**DAL** – *Data Access Layer*;

**BLL** – *Business Logic Layer*;

**IDE** – *Integrated Development Environment*;

# Organização do Relatório

Este relatório está divido por 3 partes. A introdução, onde é abordado tudo o que tem a ver com o planeamento, a proposta e os pontos de situação. A análise, que aborda tudo o que tem a ver com a interface, desenhos da Base de dados, dos modelos relacionais. E por fim, A realização do projeto, que aborda detalhadamente todas as fases do desenvolvimento do projeto em relação ao código, à interface e aos testes.

Mais especificamente:

**1ºParte:** Introdução (Planeamento, Proposta e Ponto de situação)

Nesta parte são relatadas as situações relativas à fase de planeamento e proposta, assim como todos os pontos de situações desenvolvidos e entregues nas várias datas.

**2ºParte:** Análise (Base de dados e Interface)

Nesta parte é abordada a fase da análise, tudo o que tenha a ver com a base de dados, modelo relacional e a interface da aplicação.

**3ºParte:** Implementação (Código, Interface e testes)

Nesta parte são relatadas as situações relativas à fase da implementação: à base de dados, onde contém o modelo de dados, a interface, onde contém o planeamento da interface, ao código, onde contém tudo em relação ao código e por fim os testes ao código e à base de dados.

# Introdução

Este relatório permite ver como foi a criação do nosso Projeto Tecnológico, os obstáculos que enfrentámos, como os resolvemos, todas as ferramentas usadas e todas as ajudas que tivemos. No âmbito da disciplina de Programação de Sistemas Informáticos, fizemos este Projeto Tecnológico para demonstrar os nossos conhecimentos e para termos uma experiência de como é a apresentação de um produto criado por nós ao público. Como tal, fomos propostos a fazer um programa de Gestão de Clínicas Informáticas e sendo um novo desafio para nós, aceitámos.

Ao longo desta Introdução, serão especificados os seguintes aspetos:

**- Objetivos do projeto;**

**- Proposta do projeto;**

**- Tecnologias e recursos utilizados;**

**- Contributos do projeto;**

**- Organização deste relatório;**

Objetivos do Projeto

Dado que o tema nos foi proposto, e sabendo que iria ser para uso comercial, nós sabíamos que tínhamos que dar tudo por tudo, por isso estivemos a planear fazer algo muito bom.

Os objetivos deste projeto são:

* **Criação de um programa de gestão de uma clínica informática;**
* Este programa serve para controlar todas as reparações que um cliente queira efetuar, gerir o tempo de reparação e o custo da reparação.
* **Gestão dos utilizadores, quer sejam administradores ou utilizadores normais, gestão dos técnicos no qual são utilizadores também, gestão dos Clientes e dos seus componentes (equipamentos elétricos), e por fim gestão das reparações desses componentes;**
* **Facilitar o manuseamento do programa, mais fácil de inserir, editar, eliminar um cliente, técnico, componente ou reparação e a procura dos mesmos;**
* **Controlo das Empresas participantes nesta clínica no qual tem os seus utilizadores que controlam, precisamente, a clínica informática;**
* **‘Back Office’ simples para gestão dos utilizadores e das empresas;**
* Inserir, alterar, remover empresas participantes na clínica;
* Alterar permissões, eliminar, inserir e alterar alguns utilizadores das empresas.
* **Melhorar as capacidades de criação de programas em Visual Basic;**
* Aprender certos códigos, desconhecidos para nós;
* Aprender como é ter um programa para uso comercial.

# Proposta do Projeto

Na proposta do projeto, foi apresentado os objetivos principais do projeto, como dito anteriormente (introdução).

Tendo em conta os objetivos, foram feitas previsões de datas de início e fim de cada fase

A proposta do projeto foi entregue no dia 04 de Abril de 2014.

|  |  |  |  |
| --- | --- | --- | --- |
| *Fase* | *Data de Início* | *Data de Fim* | *Horas/Dias* |
| Planeamento | 20-03-2014 | 25-03-2014 | 120/5 |
| Proposta | 01-04-2014 | 04-04-2014 | 72/3 |
| Análise | 01-04-2014 | 04-04-2014 | 72/3 |
| Implementação | 16-04-2014 | 06-07-2014 | 1944/81 |
| Documentação | 04-04-2014 | 30-06-2014 | 2088/87 |
| Apresentação | 21-07-2014 | 25-07-2014 | 96/4 |

A tabela 1, mostra as previsões das datas de início e de fim de cada fase do projeto, mostrando também as horas e os dias necessários para ser feito.

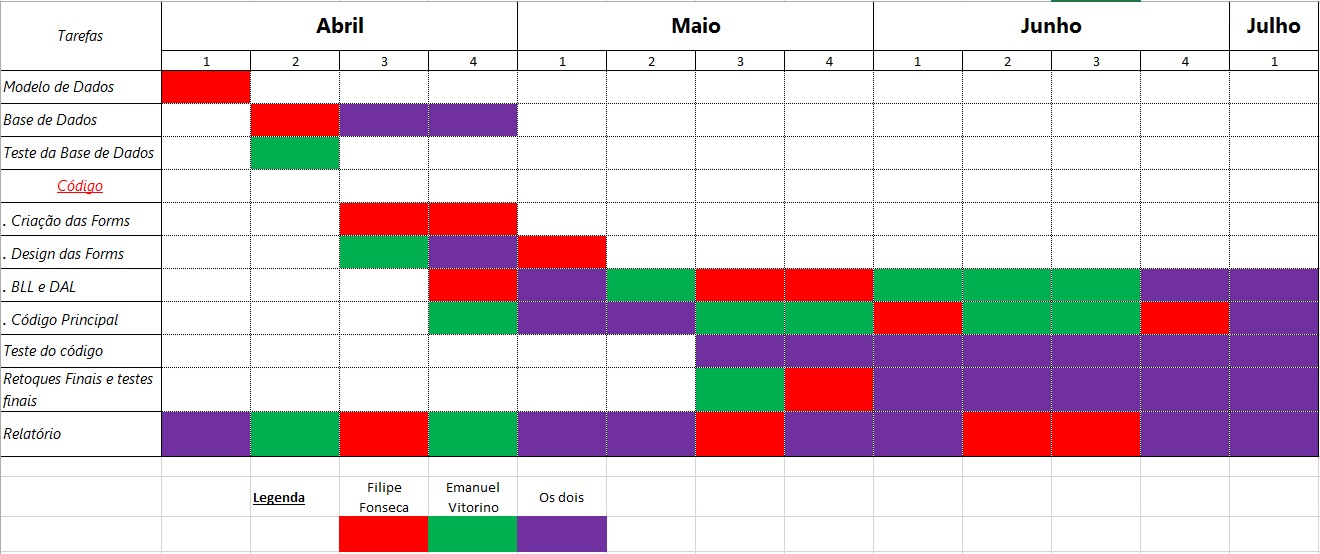
Mais detalhadamente temos aqui o cronograma:

Figura 1 - Cronograma

Este cronograma sofreu algumas alterações desde o último. Pois não estávamos a segui-lo e por isso acabou por ficar fora do tempo previsto. As alterações foram muito simples, dado que investimos mais tempo do que o normal na BLL, foi alterado a data final da BLL e adiámos o código todo 1 semana tal como o relatório.

# Tecnologias e Recursos Utilizados

***Software:***

O Projeto todo foi feito utilizando o *software* ***Microsoft Visual Studio***. Quanto à base de dados, foi utilizado o *software* **DIA** para o Modelo de Dados e a base de dados em si foi feita utilizando o ***Visual Studio*** e o ***Microsoft SQL Server****.* Por fim, as imagens usadas no programa, foram feitas pelo *software* ***Adobe Photoshop CS6 e GIMP*** e os ícones foram transferidos de um website com o nome de ***Icon Finder***. Por fim, para guardar o projeto e permitir trabalhar em conjunto com maior eficácia utilizámos uma cloud (sistema de armazenamento online) de código com o nome de GitHub.

* Microsoft *Visual Studio* 2012/2013 Professional;
* DIA (DIA *Diagram Editor*);
* Adobe *Photoshop* CS6 (*Creative Suite* 6);
* GIMP (GNU *Image Manipulation Program*);
* Icon Finder (<https://www.iconfinder.com>);
* GitHub (www.github.com).

# Realização do Projeto

Nesta parte do relatório iremos falar sobre o desenvolvimento do projeto.

## **Planeamento, Proposta e pontos de situação**

Esta fase foi constituída, por ordem de realização, nas seguintes subfases:

1. Tema do projeto;
2. Objetivos;
3. Planeamento do desenvolvimento;
4. Proposta;
5. Pontos de situação;

### **Tema do projeto**

Decidimos, para o nosso Projeto Tecnológico do ano letivo 2013/2014, desenvolver uma aplicação de gestão de uma clínica informática.

A razão da nossa escolha foi devido a uma proposta pelo nosso professor de Sistema Operativos, para realizar o projeto da clínica, e aceitámos dado que seria uma aplicação para ser usada numa empresa. Como achámos que seria uma boa oportunidade, nós aproveitámos para ser o nosso projeto.

### **Objetivos**

Para os objetivos mais detalhados, consulte a página 6.

Razões para a escolha de algumas funcionalidades:

* **Multiempresas:**
  + Permite que o programa seja usado em várias empresas e não apenas uma e permite que uma dessas empresas possa ter alunos como técnicos (ex. Escolas).
* **Relatórios de reparações e técnicos:**
* Permite ao administrador do programa saber o tempo real das reparações, os técnicos que participam nelas, a qualidade das reparações, a satisfação dos clientes e o tipo de reparação (ex. Hardware e Software).

### **Planeamento das fases de desenvolvimento**

Para os cronogramas com as datas planeadas, consulte as páginas 10 e 11.

### **Proposta**

A proposta do projeto foi entregue, pelo *moodle* da escola numa secção criada fundamentalmente para o projeto tecnológico, no dia 4 de Abril de 2014.

A proposta abrange os Objetivos e as Funcionalidades do projeto.

### **Pontos de situação**

Durante o projeto foi solicitado vários pontos de situação, que permite controlar o desenvolvimento do projeto ao longo do tempo. Um ponto de situação é um pequeno relatório que descreve o estado do projeto e tudo o que foi realizado no espaço de tempo que delimita cada ponto de situação.

**Foram apontados 4 pontos de situação, para os dias:**

* 2 de Maio de 2014;
* 16 de Maio de 2014;
* 2 de Junho de 2014;
* 16 de Junho de 2014.

## **Análise**

Nesta fase da análise existem duas situações, o planeamento do modelo de dados e o planeamento de algumas funcionalidades.

### **Planeamento do modelo de dados**

**Antes do planeamento da interface, foi necessário planear a base de dados.**

As entidades principais da clínica são:

* Clientes;
* Componentes;
* Reparações;
* Técnicos;
* Utilizadores;
* Empresas;

- **Clientes**

* Cada cliente tem de ter as suas informações pessoais que serão úteis à clínica de modo a poderem seguir a reparação do seu componente (nome, morada, contacto, etc.)
* Cada cliente pode ter **vários** componentes e pode realizar **várias** reparações a esse mesmo componente;
* **Vários** clientes pertencem apenas a **uma** empresa.

- **Componentes**

* Os componentes devem conter as informações necessárias para esclarecer os técnicos (marca, modelo, observações, etc.)
* Os componentes têm somente **um** dono (cliente)
* Um componente possui apenas uma marca.
* Os componentes podem ter **várias** reparações.
* **Vários** componentes pertencem a **uma** empresa.

**-** **Reparações**

* As reparações tem as informações necessárias para manter atualizado o cliente do seu componente (Tempo real da reparação, Data inicio e fim da reparação, etc.)
* As reparações contém apenas **um** componente de cada vez, no qual é reparado por **um ou mais** técnicos.
* As reparações são divididas por tipos, Hardware e Software pois cada componente tem o seu tipo.
* **Várias** reparações pertencem a **uma** empresa.

- **Técnicos**

* Os técnicos tem as suas informações para poderem ser contactados e a sua identidade (nome, contacto, etc.)
* Os técnicos podem participar em **várias** reparações.
* Cada técnico é **um** utilizador, pode ser administrador ou normal.
* **Vários** técnicos pertencem a **uma** empresa.

- **Utilizadores**

* Os utilizadores tem as informações necessárias para o seu login no programa (Nome Utilizador, Palavra-Passe, etc.)
* Os utilizadores são **vários** técnicos
* Os utilizadores estão divididos em três tipos:
* Administrador Geral – que controla as empresas e tudo o que lhes pertence;
* Administrador – que controla a empresa a que lhe foi atribuído;
* Utilizador normal – que apenas gerem as suas reparações e os clientes.
* **Vários** utilizadores pertencem a **uma** empresa.

- **Empresas**

* As empresas tem as suas informações a fim de estarem sempre disponíveis (nome, contacto, etc.)
* As empresas podem ou não ter **vários** clientes, componentes, reparações, técnicos e utilizadores.

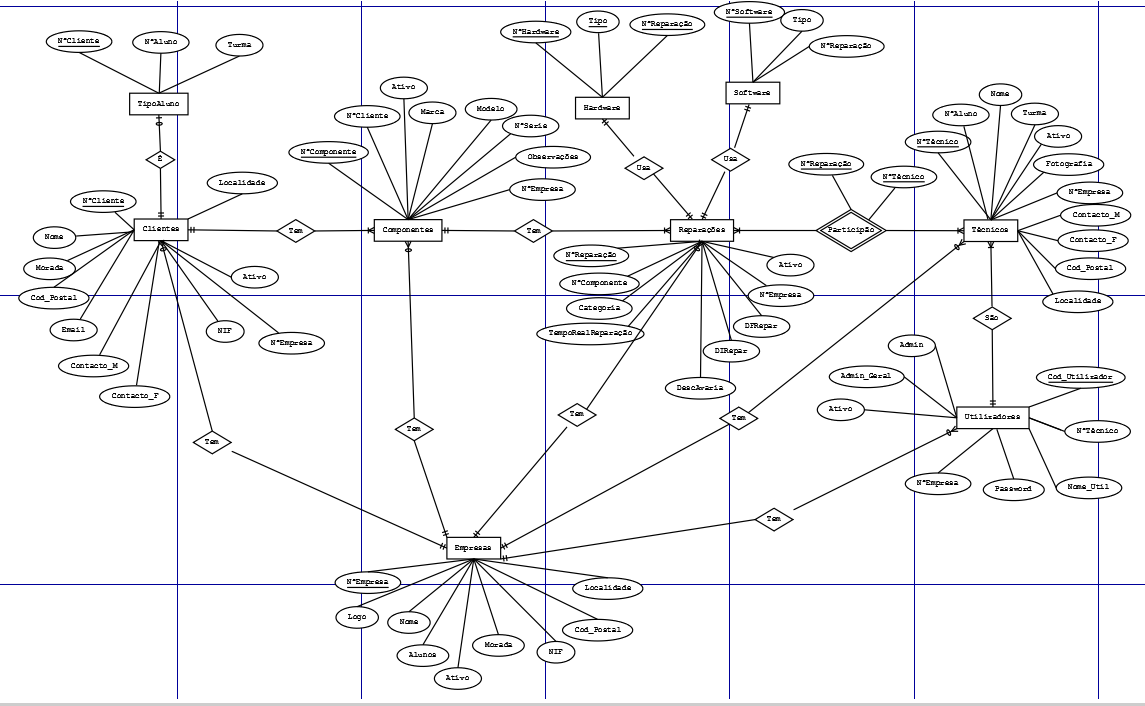
Para mais detalhes dos atributos e das relações, temos aqui o modelo de dados:

Figura 2 - Modelo de Dados E.A do nosso projeto

**Esta é a versão final do modelo E.A**

## **Implementação**

Nesta fase, dá-se a implementação do código na aplicação. Esta implementação tem como base mostrar o planeamento e realização de cada parte do código.

Esta fase divide-se em:

1. **Implementação da Interface**;
2. **Implementação das classes DAL e BLL**;
3. **Implementação da autenticação**;

### **Implementação da Interface**

Para a interface, nós optamos por uma interface simples e de fácil manuseamento. Por isso, implementámos *Forms* genéricas, ou seja, *forms* que permitem fazer tudo numa só janela, (ex. podemos inserir, editar, remover, procurar clientes, tudo na *form* dos Clientes) e que nos permitem usar a mesma *form* para várias tabelas.

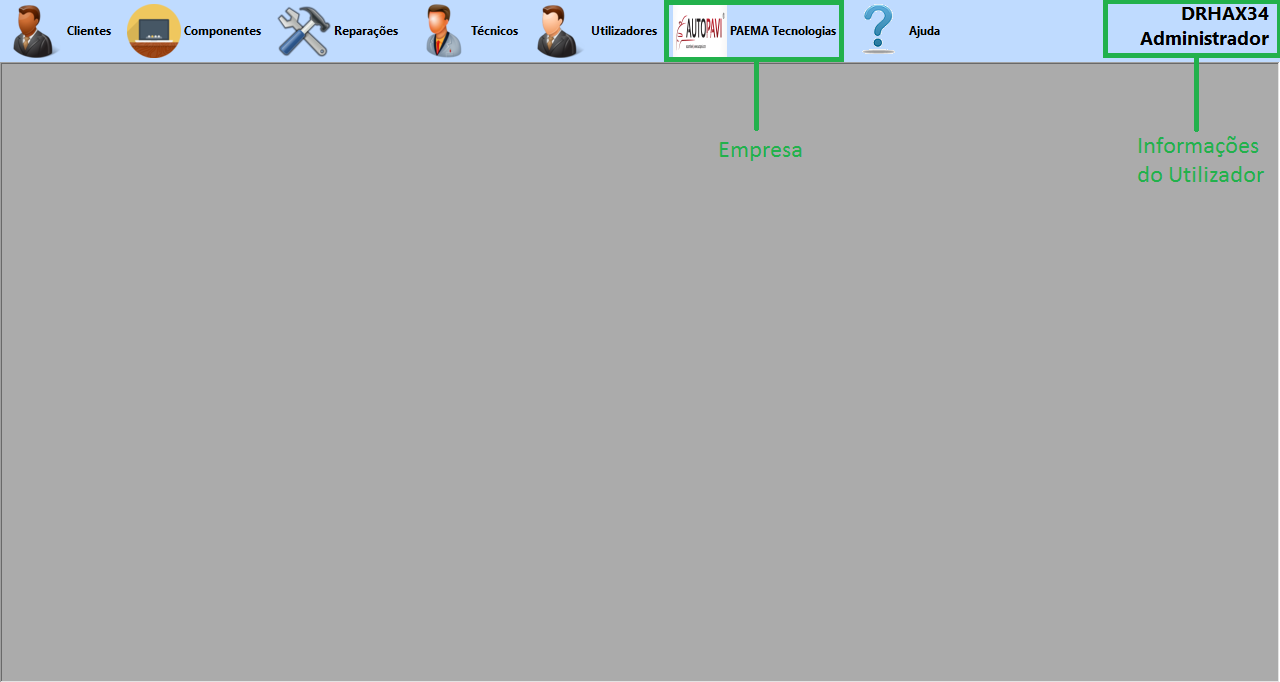
****A parte do *design* da interface, criámos uma imagem no Adobe Photoshop CS6 e implementámos no programa como *SplashScreen*, os ícones de cada entidade (clientes, componentes, reparações, etc.) foram retiradas de um site (para mais informações consulte a página 12, Tecnologia e Recursos utilizados). Aqui temos uma imagem a mostrar a interface principal do programa.

Figura 3 - Interface Principal MDI Parent do nosso Projeto

Optámos por uma interface simples dado que o que acontece frequentemente nos programas de Gestão é o difícil acesso às funcionalidades, o que faz perder tempo. A nossa interface é de manuseamento fácil e rápido, o que faz poupar tempo. A *form* principal é uma janela MDI Parent, o que significa que as janelas onde vamos introduzir dados, etc não poderá sair da janela principal e também será mostrada como parte da janela principal.

Implementámos também um assistente de primeiro uso para uma configuração mais fácil do programa em que apenas temos que introduzir a empresa inicial (caso haja mais), configurar o Administrador Geral (para gerir as empresas e os utilizadores) e o Administrador (para gerir a empresa a que pertence).

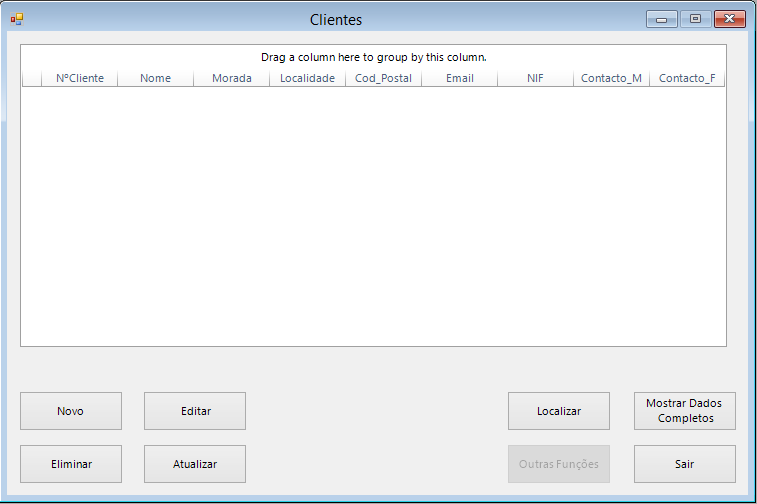
**De seguida, a interface de visualização de dados (*Form* genérica):**

Figura - ViewForm ou Janela de Visualização de Dados - MDIChild

Aqui mostramos um exemplo de uma *Form* Genérica, podemos executar todas as operações a partir desta *Form*, ou janela. Esta é também a mesma janela que vai ser usada para todas as tabelas e/ou informações. Podemos localizar os dados dependendo da tabela que está a ser mostrada (no caso dos Clientes: Nome,NIF, NºCliente). Para localizar, apenas é preciso carregar no botão localizar e irá aparecer uma *GroupBox* com o campo e os *Radiobuttons* para introduzir a procura. Esta é uma das ChildForms, ou seja, será mostrada dentro de uma MDI Parent (neste caso a janela principal).

**Para terminar esta secção, temos aqui a interface para inserir/editar/eliminar dados (Vamos usar a janela dos Clientes para este caso.):**

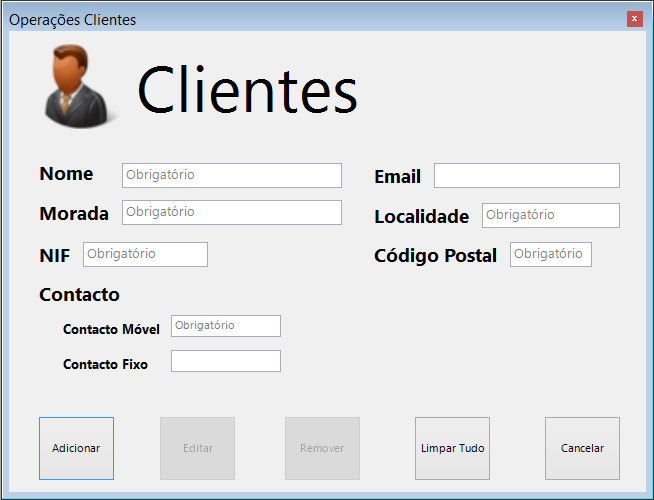
****

Figura - Janela de Operações dos Clientes - MDIChild

Como podemos ver, cada campo necessário está marcado como obrigatório. Para mostrar marcas de água nos campos de texto (ou TextBox) usámos o método *Graphics,* criámos uma classe com o nome *WaterMarkTextBox* e *WaterMarkMaskedTextBox* e colocámos na *form* dado que ao executar, o *Visual Studio* coloca à disposição o objeto que acabámos de criar na interface *Designer*. Cada Campo está validado, ou seja, no campo para introduzir o Nome, Morada e Localidade apenas podem se introduzem letras enquanto no campo para introduzir os Contactos, NIF e Código Postal apenas podem se introduzir Números ou (no caso dos Contactos) “+”.

### **Implementação das classes DAL e BLL**

Para aceder à base de dados de uma forma organizada, dividimos o processo de acesso por camadas.

A camada DAL (Data Access Layer) é a camada responsável por fazer a ligação à base de dados, executar o comando SQL e carregar os dados.

A camada BLL (Business Logic Layer) é a camada intermediária que é responsável pela troca de dados entre a aplicação e a DAL.



Figura 6 Figura ilustrativa sobre a BLL, DAL e a Base-de-Dados

**A nossa BLL apresenta seis grandes classes:**

* Admin\_only (Apenas disponível para Administradores)
* Login
* Clientes
* Componentes
* Reparações
* Técnicos

Dentro de cada uma dessas classes existem várias funções como a função de procurar (alguns por ID, outros por nome, entre outros), inserir, editar, remover e outras funções importantes para cada classe.

Aqui está uma imagem para poderem verificar melhor as classes existentes.

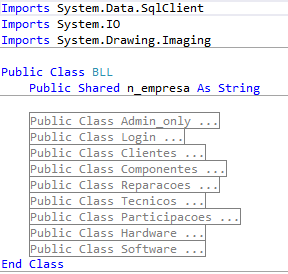


Figura 7 - A BLL do nosso projeto

Dentro de cada classe na BLL (à exceção da Classe Participações, Hardware e Software que só tem carregar, inserir e remover) temos as funções de procurar, inserir, alterar, carregar, remover e reativar dados. Dentro da classe “*Admin\_only*” existem as classes de “Empresas” e “*Login*” e existe também uma função para remover permanentemente uma empresa, também temos uma função dentro da classe Empresas para desativar uma empresa. Apenas temos acesso a esta classe (Empresas) se declararmos no Assistente de Primeiro Uso que temos apenas uma empresa.

Aqui mostramos a função para eliminar permanentemente uma empresa:

Shared Sub eliminar\_empresa(ByVal nempresa As Integer)0

Dim p As New ArrayList

p.Add(New SqlParameter("@n\_empresa", nempresa))

DAL.ExecuteNonQuery("Delete FROM Utilizadores where NºEmpresa=@n\_empresa", p)

DAL.ExecuteNonQuery("Delete FROM Técnicos where NºEmpresa=@n\_empresa", p)

DAL.ExecuteNonQuery("Delete FROM Clientes where NºEmpresa=@n\_empresa", p)

DAL.ExecuteNonQuery("Delete FROM Dispositivos where NºEmpresa=@n\_empresa", p)

DAL.ExecuteNonQuery("Delete FROM Reparações where NºEmpresa=@n\_empresa", p)

DAL.ExecuteNonQuery("Delete FROM Empresas where NºEmpresa=@n\_empresa", p)

End Sub

Como podemos ver, a classe é composta por várias operações para eliminar permanentemente em cada tabela os Clientes, Utilizadores, etc que pertencem à empresa especificada e eliminar também da tabela Empresas a Empresa correspondente.

Um exemplo da função “inserir” em cada classe:

Shared Function inserir(ByVal NIF As String, ByVal localidade As String, ByVal nome As String, ByVal morada As String, ByVal cod\_postal As String, ByVal email As String, ByVal ativo As Boolean, ByVal contacto\_mov As String, ByVal contacto\_fix As String)

Dim p As New ArrayList

Dim c As New ArrayList

Dim a As New ArrayList

Dim check(1) As Integer

p.Add(New SqlParameter("@Nome", nome))

p.Add(New SqlParameter("@Morada", morada))

p.Add(New SqlParameter("@NIF", NIF))

p.Add(New SqlParameter("@Cod\_Postal", cod\_postal))

p.Add(New SqlParameter("@Localidade", localidade))

p.Add(New SqlParameter("@Email", email))

p.Add(New SqlParameter("@Ativo", ativo))

p.Add(New SqlParameter("@Ativo", ativo))

p.Add(New SqlParameter("@contacto\_m", contacto\_mov))

p.Add(New SqlParameter("@contacto\_f", contacto\_fix))

p.Add(New SqlParameter("@n\_empresa", BLL.n\_empresa))

return DAL.ExecuteNonQuery("Insert into Clientes(Nome,Morada,NIF,Localidade,Cod\_Postal,Email,Ativo,NºEmpresa,Contacto\_M,Contacto\_F) VALUES (@nome, @morada, @NIF,@Localidade, @cod\_postal,@email,1,@n\_empresa,@contacto\_m,@contacto\_f)", p)

End Function

Esta função (que no exemplo pertence aos Clientes) insere os dados que especificámos na base-de-dados e devolve um valor que determina se inseriu corretamente ou não (0 ou -1). O código pertencente a este tuplo (neste caso, cliente) é gerado automaticamente.

Exemplo da função “alterar”:

Shared Function alterar\_aluno(ByVal numcliente As Integer, ByVal localidade As String, ByVal NIF As String, ByVal nome As String, ByVal morada As String, ByVal cod\_postal As String, ByVal email As String, ByVal ativo As Boolean, ByVal contacto\_m As String, ByVal contacto\_f As String)

Dim p As New ArrayList

Dim c As New ArrayList

Dim check(1) As Integer

p.Add(New SqlParameter("@numcliente", numcliente))

p.Add(New SqlParameter("@Nome", nome))

p.Add(New SqlParameter("@Morada", morada))

p.Add(New SqlParameter("@NIF", NIF))

p.Add(New SqlParameter("@Localidade", localidade))

p.Add(New SqlParameter("@Cod\_Postal", cod\_postal))

p.Add(New SqlParameter("@Email", email))

p.Add(New SqlParameter("@Ativo", ativo))

p.Add(New SqlParameter("@Contacto\_m", contacto\_m))

p.Add(New SqlParameter("@Contacto\_f", contacto\_f))

return DAL.ExecuteNonQuery("Update Clientes set Localidade=@Localidade, Contacto\_M = @contacto\_m, Contacto\_F = @contacto\_f, nome = @nome, morada = @morada, NIF= @NIF, Cod\_Postal= @cod\_postal, Email= @email, Ativo=@Ativo where NºCliente=@numcliente", p)

End Function

Nesta função altera os dados que especificámos na base-de-dados e mais uma vez devolve o valor para determinar se as alterações foram feitas com sucesso. O código pertencente a este tuplo não pode ser alterado.

Exemplo da função “remover”:

Shared Sub apagar(ByVal N\_Cliente As String)

Dim p As New ArrayList

p.Add(New SqlParameter("@NºCliente", N\_Cliente))

If N\_Cliente <> Nothing Or N\_Cliente <> "" Then

DAL.ExecuteNonQuery("Update Clientes set Ativo=0 where NºCliente = @NºCliente", p)

DAL.ExecuteNonQuery("Update Componentes set Ativo=0 where NºCliente=@NºCliente", p)

End If

End Sub

Nesta função alteramos um cliente como desativo, ou seja “removido”. Os clientes apenas são removidos permanentemente se removermos a empresa a que pertencem permanentemente.

Exemplo da função “restaurar”:

Shared Sub restaurar(ByVal N\_Cliente As String)

Dim p As New ArrayList

p.Add(New SqlParameter("@NºCliente", N\_Cliente))

If N\_Cliente <> Nothing Or N\_Cliente <> "" Then

DAL.ExecuteNonQuery("Update Clientes set Ativo=1 where NºCliente = @NºCliente", p)

DAL.ExecuteNonQuery("Update Componentes set Ativo=1 where NºCliente=@NºCliente", p)

End If

End Sub

Nesta função apenas declaramos um cliente desativo como ativo.

Exemplo da função “Procurar”:

Shared Function procura\_dados\_numcliente(ByRef num\_cliente As String) As DataTable

Dim p As New ArrayList

p.Add(New SqlParameter("@n\_empresa", BLL.n\_empresa))

p.Add(New SqlParameter("@NºCliente", num\_cliente))

If num\_cliente = "" Then

Return DAL.ExecuteQueryDT("SELECT NºCliente,Nome,Morada,NIF,Localidade,Cod\_Postal,Email,Contacto\_M,Contacto\_F FROM Clientes where Ativo=1 AND NºEmpresa=@n\_empresa", p)

Else

Return DAL.ExecuteQueryDT("SELECT NºCliente,Nome,Morada,NIF,Localidade,Cod\_Postal,Email,Contacto\_M,Contacto\_F FROM Clientes where NºCliente like @NºCliente AND Ativo=1 AND NºEmpresa=@n\_empresa", p)

End If

End Function

Esta função serve para procurar dados na tabela correspondente com base no NºCliente (dado que no exemplo estamos a procurar na tabela Clientes). Em vez de comparar os dados com “=” usamos “like” para a função devolver dados que contenham a palavra que está entre “%”. (Ex. “%teste%”).

Exemplo da função “Carregar dados específicos”:

Shared Function carregar\_dados\_numcliente(ByRef num\_cliente As String) As DataTable

Dim p As New ArrayList

p.Add(New SqlParameter("@n\_empresa", BLL.n\_empresa))

p.Add(New SqlParameter("@NºCliente", num\_cliente))

If num\_cliente = "" Then

Return DAL.ExecuteQueryDT("SELECT NºCliente,Nome,Morada,NIF,Localidade,Cod\_Postal,Email,Contacto\_M,Contacto\_F FROM Clientes where Ativo=1 AND NºEmpresa=@n\_empresa", p)

Else

Return DAL.ExecuteQueryDT("SELECT NºCliente,Nome,Morada,NIF,Localidade,Cod\_Postal,Email,Contacto\_M,Contacto\_F FROM Clientes where NºCliente=@NºCliente AND Ativo=1 AND NºEmpresa=@n\_empresa", p)

End If

End Function

Igual à procura de dados mas em vez de usarmos “like” usamos “=” para devolver apenas os dados que especificámos.

Por fim, exemplo da função “Carregar Tabela”:

Shared Function carregar() As DataTable

Dim p As New ArrayList

p.Add(New SqlParameter("@n\_empresa", BLL.n\_empresa))

Return DAL.ExecuteQueryDT("SELECT NºCliente,Nome,Morada,Localidade,Cod\_Postal,Email,NIF,Contacto\_M,Contacto\_F FROM Clientes where Ativo=1 AND NºEmpresa=@n\_empresa", p)

End Function

Esta função tem como objetivo devolver todos os tuplos da tabela “Clientes” que estejam ativos (“Ativos=1 (1 = true ou verdadeiro) ”) e que pertençam à empresa que está especificada na BLL (ao iniciarmos o programa e selecionarmos a empresa, o programa configura a BLL para utilizar o número da empresa que selecionámos).

### **Implementação da Autenticação**

A nossa autenticação consiste no nome do utilizador, palavra-passe e nº de empresa dado que cada utilizador (exceto os administradores gerais) pertence e gere apenas uma empresa.

Cada palavra-passe é encriptada com um algoritmo que se chama “hash64” que consiste no cálculo de um código hexadecimal juntando a palavra-passe em texto simples com outra frase dada por nós que, por motivos de segurança não iremos escrever neste relatório.

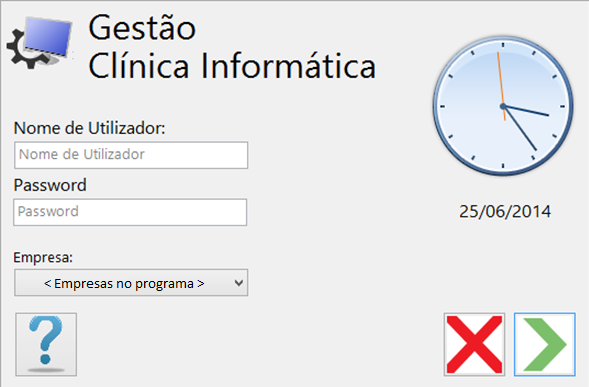
O utilizador insere um nome de utilizador e palavra-passe e escolhe a empresa a que pertence (no caso de o programa ter múltiplas empresas). A palavra-passe introduzida pelo utilizador é encriptada e passamos os dados para uma função existente na BLL que devolve o código de utilizador caso o nome de utilizador e o código resultante da encriptação da palavra-passe sejam iguais (tornámos o SQL nesta função Case-Sensitive com a conversão dos dados a comparar em Código Binário). O programa irá entrar na área do utilizador. Por fim, criámos um sistema de perguntas de segurança para caso o utilizador perder a palavra-passe pode sempre recuperar sem qualquer incómodo.

Figura 8 - LoginForm ou janela de Início de Sessão - MDIChild

# Versões

## 0.1.0.0

* Implementação da BLL, DAL e Base de Dados;
* Interface pouco ou nada funcional;
* Criação das *Forms* de Clientes, Dispositivos, Reparações, Técnicos, Utilizadores;
* Apenas permite uma empresa.

## 0.3.2.0

* Implementação das *Forms MDIParen*t e edição das *Forms* Clientes, Dispositivos, Reparações, Técnicos, Utilizadores para *MDIChild*;
* Implementação da Autenticação;
* Implementação de Multi-Empresa;
* Correção de erros na BLL e na Base-de-Dados;
* Criação da classe *WaterMarkTextBox* para melhor aspeto geral do programa;
* Criação de um fundo para um programa.

## 0.5.4.0

* Criação de um *SplashScreen*;
* Remoção do fundo por motivos de Rapidez do Programa;
* Correção de erros na BLL;
* Todas as *forms* funcionais exceto Utilizadores, Empresas e Reparações.

## 0.7.3.0

* *Forms* dos Utilizadores e Empresas funcionais;
* Modificação da Interface para melhor aspeto geral;
* Correção de erros na BLL;
* Todas as *forms* funcionais;
* Considerada Release-Candidate (ou seja mais próxima para ser a versão final);
* Implementação das Validações nas *TextBox*.

## 1.0.0.0

* Melhorias na interface;
* Correção de Algumas Validações;
* Melhorias nas *Forms*;
* Criação de um Menu Inicial e acabamento dos Relatórios da Aplicação.

# Conclusões

**Objetivos realizados**

* **Criação de um programa de gestão de uma clínica informática;**
* Este programa serve para controlar todas as reparações que um cliente queira efetuar e gerir o tempo de reparação.
* **Gestão dos utilizadores, quer sejam administradores ou utilizadores normais, gestão dos técnicos no qual são utilizadores também, gestão dos Clientes e dos seus componentes (equipamentos elétricos), e por fim gestão das reparações desses componentes;**
* **Facilitar o manuseamento do programa, mais fácil de inserir, editar, eliminar um cliente, técnico, componente ou reparação e a procura dos mesmos;**
* **Controlo das Empresas participantes nesta clínica no qual tem os seus utilizadores que controlam, precisamente, a clínica informática;**
* **‘Back Office’ simples para gestão dos utilizadores e das empresas;**
* Inserir, alterar, remover empresas participantes na clínica;
* Alterar permissões, eliminar, inserir e alterar alguns utilizadores das empresas.
* **Melhorar as capacidades de criação de programas em Visual Basic;**
* Aprender certos códigos, desconhecidos para nós;
* Aprender como é ter um programa para uso comercial.

# Dificuldades

Tivemos dificuldades na estabilização do código da “BLL”, tivemos que estar sempre ao longo do projeto a editar e corrigir o código na classe “BLL”. Também tivemos algumas dificuldades nas opções de imprimir e nas opções de imprimir relatórios sobre tempo de reparação de alguns produtos, eficiência, entre outros.

# Agradecimentos

**Professor António Beirós –** Pelos conhecimentos prestados e pela ajuda prestada quando tínhamos erros no projeto e pelos conhecimentos prestados ao longo do segundo ano.

**Professora Sandra Rodrigues –** Pela ajuda no design da aplicação e ajuda na apresentação e também pelos conhecimentos lecionados ao longo do primeiro e segundo ano.

**Professor João Santos –** Pela oportunidade de sermos nós a criar um programa necessário para ele e ajuda na criação da Base de dados e Relatórios.

**Escola Profissional Gustave Eiffel –** Pelo equipamento que nos providenciou para a execução deste projeto e pelos professores que nos ensinaram ao longo deste tempo.

# Apreciação Final

A nossa apreciação final do projeto tecnológico é satisfatória, pois apresenta aquilo que sempre tivemos em mente.

Nós dedicámos mais tempo do que o esperado no projeto, haviam fases que consumiam bastante tempo.

Relativamente ao cronograma, fomos seguindo-o, porém com algumas partes fora do tempo previsto.

Na parte do desenvolvimento, a implementação dos relatórios foi a que consumiu mais tempo e atrasou ligeiramente o projeto.

No entanto, mesmo com o trabalho que tivemos nos relatórios ficou como nós pretendíamos e estamos contentes com os resultados.

O desenvolvimento do projeto tecnológico consiste na aprendizagem de novos conteúdos e um meio de preparação para a nossa PAP.

O projeto em si não é fácil pois é, basicamente, uma conjunção de toda a matéria que demos. Pode ter alguma pressão, mas só a temos se não houver trabalho, e quem tem trabalho verá que a pressão que existe é pouca.