



INSTITUTO POLITÉCNICO  
DO CÁVADO E DO AVE  
ESCOLA SUPERIOR DE TECNOLOGIA

RELATÓRIO DE TRABALHO PRÁTICO

# Sistema de gestão para um *Hostel* em C# (Basics)

---

NELSON CUNHA

A19241

E

EMANUEL GOMES

A18869

Trabalho realizado sob a orientação de:

Luís Ferreira

**Linguagens de Programação II**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, Maio de 2020



## Índice

1	INTRODUÇÃO	1
1.1	Objetivos	1
2	PROJETO: TRABALHO DESENVOLVIDO – FASE 3	3
3	DIAGRAMA DE CLASSES	7
3	CONCLUSÃO	9
	BIBLIOGRAFIA	11
	ANEXOS	13

## Lista de Tabelas

Não foi encontrada nenhuma entrada do índice de ilustrações.

Lista de Figuras

Figura 1 - Visual da aplicação..... 3

Figura 2 - Objetos utilizados no MVC e suas interações..... 4

Figura 3 - Diagrama de classes (versão 2)..... 7



# 1 Introdução

O presente trabalho desenvolvido no âmbito da disciplina de Linguagens de Programação II, referente à avaliação da componente prática da disciplina, é baseado num sistema de gestão para um *Hostel*, usando a linguagem em C#.

A ideia deste trabalho surgiu em seguimento de um encontro com um empresário no ramo hoteleiro no Minho.

Ou seja, para nós este trabalho não é apenas um trabalho académico queremos também tirar proveito de trabalho realizado e se possível aplicá-lo na prática.

Salientamos, que esta versão final do trabalho encontra-se ainda em fase de teste (fase 3 do trabalho de acordo com o cronograma previsto).

A seguir, vamos apresentar brevemente os objetivos para este trabalho.

## 1.1 Objetivos

O trabalho tem como objetivo ser uma aplicação pratica e simples com o intuito de ajudar da melhor forma possível o gerente do hotel na manipulação dos dados dos hotéis.

No entanto, o nosso objetivo, em termos práticos, será de conseguir efetuar reservas de quartos em C# e alcançar os objetivos definidos pela disciplina de Linguagens de Programação II.

Tentamos implementar novos conceitos que nos foram ensinados durante as aulas da disciplina e tentar testar novas metodologias e conceitos no respetivo trabalho.





## 2 Projeto: trabalho desenvolvido – Fase 3

Nesta terceira fase do trabalho, tentamos dar seguimento ao que foi apresentado na 2 fase, dando ênfase ao desenvolvimento da nova estrutura e organização do trabalho por camadas e ainda tentamos adaptar o trabalho com a Framework MVC.

Salientamos que na prática, os funcionários também podem reservar quartos no Hostel a condições preferências em relação aos Hospedes (*guest*), mas optamos por não criar uma classe denominada, por exemplo, por *FuncionarioHospede* onde encontraríamos um método “desconto” à título de exemplo.

Temos de referir que antes de desenvolver as camadas, tínhamos criado uma classe *Hostel.cs* onde colocamos um *array* de quartos, cuja a capacidade (real) do Hostel era exatamente 15 quartos, onde o nº 12 é um quarto de casal e os quartos nº 7, 9 e 10, são quartos do tipo Single, sendo os restantes do tipo dormitório.

Nota-se que eliminamos alguns métodos desenvolvidos na primeira fase do trabalho e reestruturamos outras elaboradas na 2 fase deste trabalho, como por exemplo o caso do pagamento, isto porque nesta fase o nosso objetivo seria de nos dedicar mais na parte das reservas propriamente dito.

E finalmente temos de referir que nesta 3 fase tentamos criar um layout deste projeto usando o *Windows Form App* (.NET Framework) para tentar criar um aspeto mais visual e “*user friendly*” (ver Figura 1) para a inserção, por exemplo, de novos hóspedes (apenas fizemos para essa “categoria” para testar).

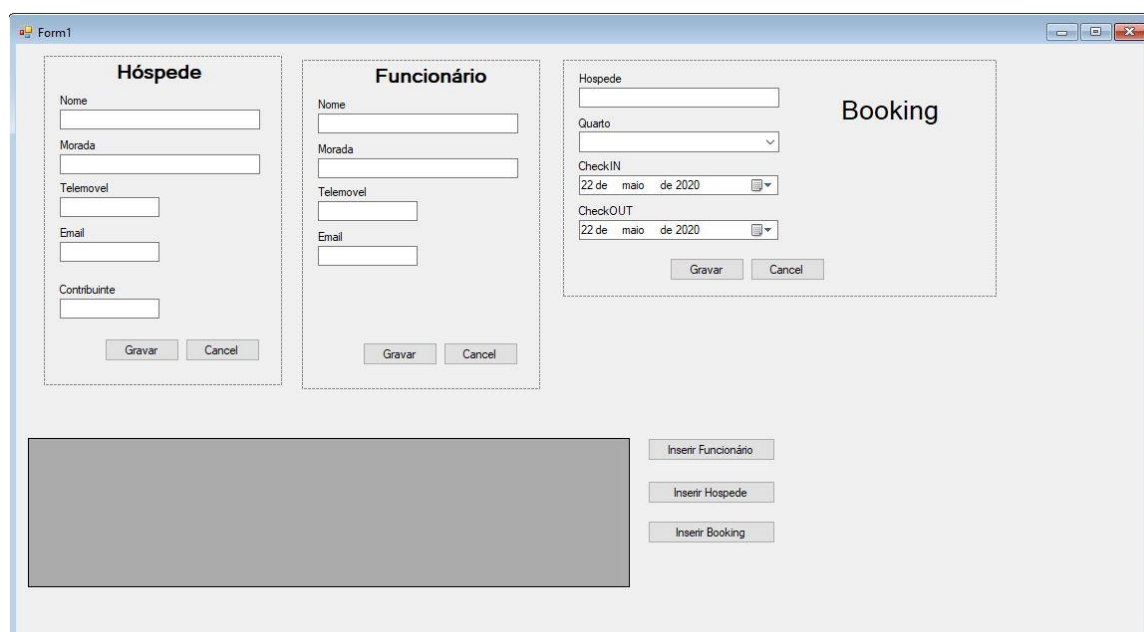


Figura 1 - Visual da aplicação

Temos de referir ainda que já desenvolvemos uma parte do código que não consta no código enviado, isto porque tentamos dar mais ênfase ao desenvolvimento da parte dos hóspedes e das reservas e adaptar o projeto com as **camadas** *Business Layer* – BL, *Business Object* – BO e Dados.

Ou seja, “relacionamos” as nossas classes com as respetivas camadas, isto é: temos a 1ª camada Hostel que liga a BL onde temos uma classe “Funções” que permite gravar os hóspedes (*Windows Form*) por exemplo. No entanto, tivemos de colocar o BL em “comentário”, isto devido a atualização feita no projeto ao tentar introduzir o framework MVC, abordar a seguir.

E finalmente no BO, temos por exemplo as classes: *Quarto*, *Hospede*, *Funcionario*, *Booking* entre outros. E na camada Dados, temos a classe *Quartos* e *Bookings*.

No entanto, temos de referir que neste projeto com a introdução e implementação (pelo menos a tentativa) da arquitetura MVC (Model-View-Controller) (ver Figura 2). Basicamente, o conceito principal do modelo MVC é utilizar uma solução já definida para separar as partes distintas do projeto reduzindo as suas dependências ao máximo.

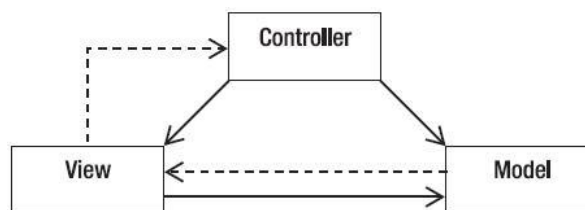


Figura 2 - Objetos utilizados no MVC e suas interações (Fonte: <https://www.devmedia.com.br/>)

Ou seja, o MVC é utilizado em muitos projetos devido à arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o *Model* (tratar por exemplo da realização da operação matemática – não abordamos essa parte neste trabalho), o *Controller* e a *View* (que é basicamente a nossa interface gráfica), executa o que lhe é definido e nada mais do que isso.

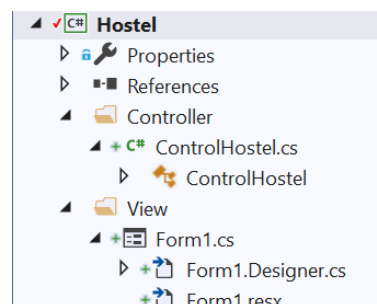


Figura 3 - MVC





### 3. Diagrama de classes

A seguir, vamos apresentar a versão da fase 2 para ter uma ideia da evolução. É de notar que esse diagrama foi “tirado” antes de passar por camadas.

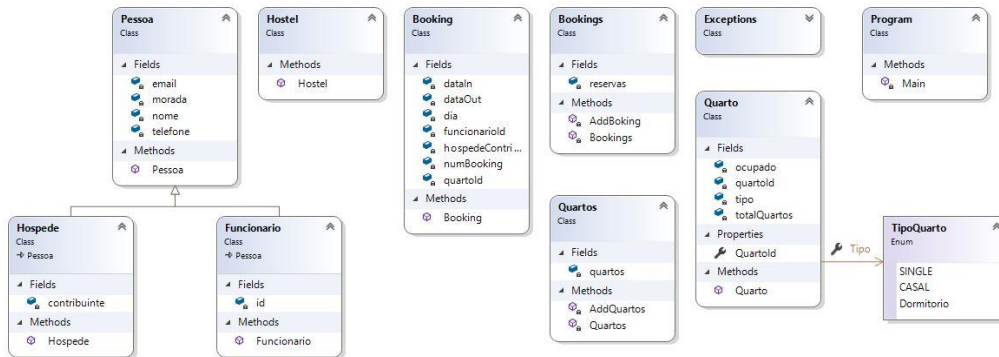


Figura 4 - Diagrama de classes (versão 2)

As classes que foram criadas forma numa ótica e cujo objetivos são apenas dedicar a um “*purpose*” (Stiefel & Oberg, 2001).

Apenas a título informativo, vamos a seguir colocar os respetivos diagramas por camada. Podemos ver na Figura abaixo o diagrama na camada BO.

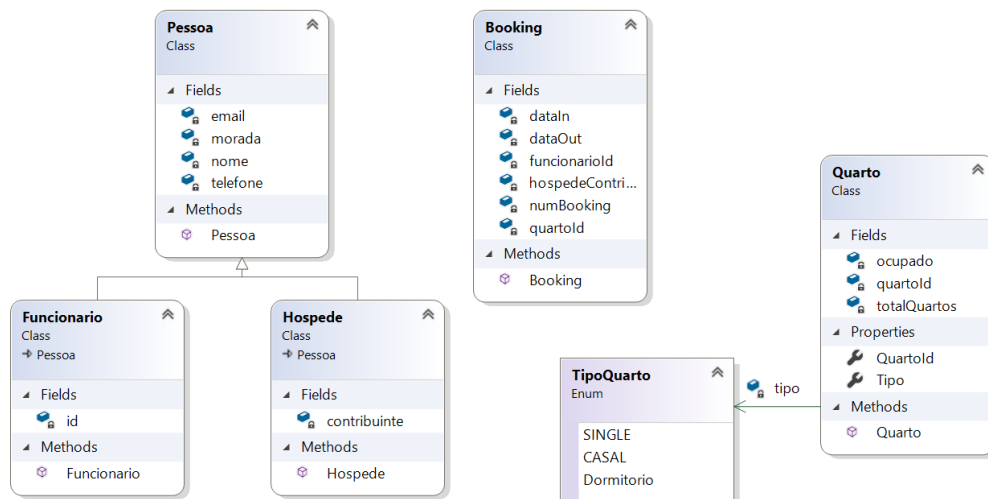


Figura 5 - Diagrama das classes na camada BO

Temos de referir que na Figura 5O, a classe Ficheiro tinha sido usada na fase 2, mas depois deixamos de usar esta classe. Apenas foi deixada para entender a evolução na construção e desenvolvimento do trabalho. Neste momento ao gravar na janela (Windows Form). As informações dos hospedes e reservas são gravadas em ficheiros binários ao clicar em gravar.

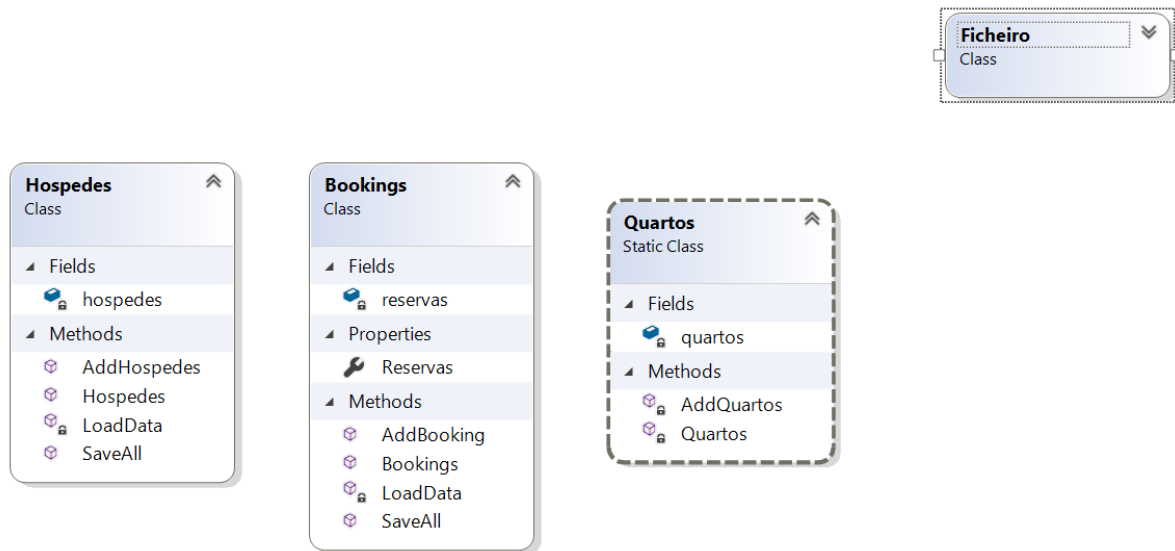


Figura 6 - Diagrama de classe na camada Dados

### 3 Conclusão

Apesar deste trabalho ser uma versão “final”, tentamos corrigir alguns erros detetados na segunda fase, mas optamos por testar e incrementar algumas partes como teste para ver o funcionamento e dinâmica desses novos elementos neste projeto, como por exemplo a criação de um controlador.

Elaboramos este projeto com as camadas BL, BO e a camada Dados, e tentamos criar listas como por exemplo uma lista para os hóspedes e uma lista de reservas por via do Windows Form.

Tentamos também implementar a *framework* MVC, que acreditamos ser algo muito interessante e útil.

Nesta última fase deste trabalho, tentamos implementar (ou pelo menos incorporar como teste) as funcionalidades e estrutura do trabalho que nós foi ensinado durante este disciplina.





## Bibliografia

STIEFEL, M. & OBERG, R.J., (2001), Application Development Using C# and .NET, Prentice Hall;

Sites da internet:

[https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649643\(v=pandp.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649643(v=pandp.10)?redirectedfrom=MSDN)

<https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>



## Anexos

Código ainda em desenvolvimento:

### Classe **Booking**

```
/// Método que realiza o registo de uma reserva.

public void RegistrarReserva(int pessoasPorReserva, DateTime inicial, DateTime final)
{
    for (int i = 0; i < listaReserva.Length; i++)
    {
        if (listaReserva[i] == null)
        {
            listaReserva[i] = new Reserva(pessoasPorReserva, inicial, final);
            return;
        }
    }
}
```

### Classe **Hospede**

inserir uma *string numcontribuinte*

Verificar se os dígitos inseridos do Contribuinte igual a 9:

```
public static bool CheckNumContri(string num)
{
    if(num.Length == 9)
    {
        return true;
    }
    return false;
}
```

### Classe **Quarto**:

```
public int maxPessoas
{
    get { return nmaxPessoas; }
    set { nmaxPessoas = value; }
}

public Quarto(int nquarto, int nmaxPessoas)
{
    this.nQuarto = nquarto;
    this.maxPessoas = nmaxPessoas;
}
```

Método que verifica se um quarto pode albergar mais que um determinado número de pessoas.

```
public bool CheckNumeroPessoas(int nmaximopessoas)
{
    if(nmaximopessoas > this.maxPessoas)
    {

```

```
        return false;  
    }  
    return true;  
}
```