



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 2	3
3	DIAGRAMA DE CLASSES	7
3	CONCLUSÃO	9
	BIBLIOGRAFIA	11
	ANEXOS	13

Lista de Tabelas

Tabela 1:Hospedes3

Tabela 2: Funcionários.....3

Lista de Figuras

Figura 1 - Visual da aplicação	5
Figura 2 - Diagrama de classes (fase 1)	7
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 do trabalho encontra-se ainda em fase de desenvolvimento (fase 2 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.

Pretendemos ainda conseguir 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 2

Nesta segunda fase do trabalho, tentamos desenvolver novas classes e melhorar as classes que foram apresentadas na primeira fase.

Tentamos ainda melhorar a estrutura e a organização da melhor forma possível nesse projeto.

Criamos algumas classes que serão apresentadas no diagrama no capítulo seguinte (ver Figura 3), onde por exemplo pretendemos que certos campos de informação estejam disponíveis, por exemplo na tabela 1 que apresenta informação importante para o *Hostel*.

Tabela 1: Hospedes

Nome	Morada	Tel.	Mail	Nº Contribuinte

Na tabela abaixo, apresentamos outro exemplo de classe criada que é a classe funcionário, onde se destaca os campos que entendemos ser mais pertinentes para este projeto.

Tabela 2: Funcionários

Nome	Morada	Contacto	Mail

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.

Para a classe **Quartos.cs** (e não *Quarto.cs*) tentamos desenvolver uma *List<Quarto> quartos*, ou seja criar uma lista.

Salientamos ainda que corrigimos uns problemas detetadas na primeira fase relativamente às heranças entre a classe Pai Pessoa e os filhos (*Funcionario e Hospede*) por exemplo. Em relação a classe *Quarto.cs*, onde já pensamos (na 1 fase) num método para verificar se um determinado quarto podia

alojar mais que um determinado número de pessoas, e ainda um método que verificasse se um determinado quarto já foi registado com o mesmo número via uma função booleana (ver em anexo).

Outro exemplo que está ainda em fase de estudo seria de criar, talvez, uma classe *Data.cs* (propriamente dito) para gerir o “calendário” e aceder as datas com as respetivas reservas e estado dos quarto do hostel, mas ainda está em fase de elaboração.

Salientamos, em termos de modificação nesta fase do trabalho, que acrescentamos o nº de contribuinte (uma *string* contribuinte) na classe *hospede.cs* que foi mencionado na primeira fase do trabalho que seria uma especie de id para o hóspede.

Podemos referir que na classe *Program.cs* apenas queremos acrescentar os métodos (das futuras camadas).

No caso da classe *Booking.cs* (reserva), pretendemos desenvolver um método uma lista de reservas (ainda em fase de elaboração) e criar um método com uma *string* com informação da reserva. Mas esta classe ainda requer algum trabalho.

Salientamos 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 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 fase tentamos criar um “novo” 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).

The screenshot shows a Windows application window titled "Form1". It contains three main sections: "Hóspede", "Funcionário", and "Booking". Each section has a set of input fields and "Gravar" (Save) and "Cancel" buttons. The "Hóspede" section has fields for Nome, Morada, Telemovel, Email, and Contribuinte. The "Funcionário" section has fields for Nome, Morada, Telemovel, and Email. The "Booking" section has fields for Hospede, Quarto (a dropdown menu), CheckIN (22 de maio de 2020), and CheckOUT (22 de maio de 2020). Below these sections is a large gray rectangular area. To the right of this area are three buttons: "Inserir Funcionário", "Inserir Hospede", and "Inserir Booking".

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 nesta segunda fase ao desenvolvimento e correção das classes que é uma fase crucial para bem estruturar o nosso trabalho e tentar criar as **camadas Business Layer – BL, Business Object – BO** e Dados (ainda em fase de elaboração na presente versão deste trabalho).

Ou seja, nesta fase tentamos “relacionar” 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*).

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*.

3. Diagrama de classes

Neste capítulo, vamos apresentar em primeiro lugar as classes que foram desenvolvidas na primeira fase (ver Figura abaixo).

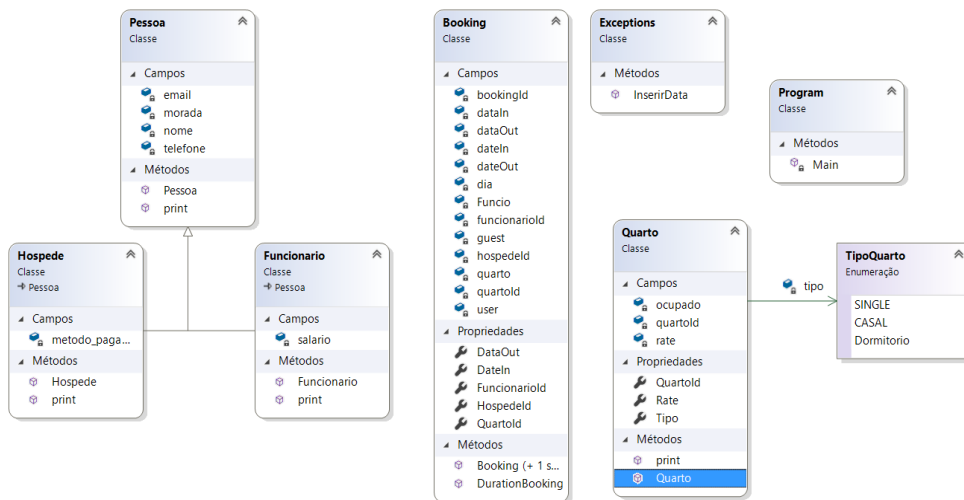


Figura 2 - Diagrama de classes (fase 1)

A seguir, vamos apresentar a versão “atual” e modificada das classes criadas. É de notar que esse diagrama foi “tirado” antes de passar por camadas¹.

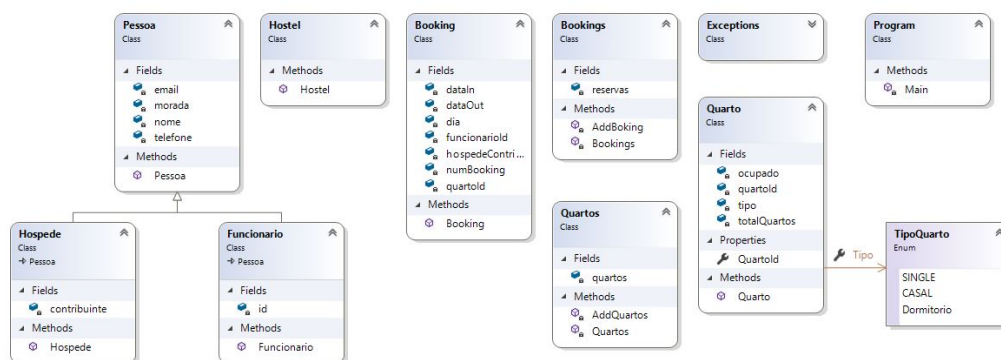


Figura 3 - Diagrama de classes (versão 2)

¹ Sendo neste momento, tínhamos de ter um diagrama por “tipo de camada” (e.g BL, BO..).

Salientamos que existe ainda algumas dúvidas quanto à necessidade de criar ainda mais uma ou até mais duas classes para este trabalho. No entanto, tentamos criar novas classes cujo objetivos são apenas dedicar a um “*purpose*” (Stiefel & Oberg, 2001).

3 Conclusão

Apesar deste trabalho estar ainda em desenvolvimento nesta segunda fase, tentamos corrigir alguns erros detetados na primeira fase, e respeitar os critérios do *CLS compliance* (boas praticas)

Neste momento, tentamos criar 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.

Ainda temos algumas dúvidas quanto à necessidade de criar certas classes nomeadamente a classe Data (calendário) e de conseguir interligar alguns dos conceitos vistos na disciplina da cadeira.

Mas acreditamos que para a terceira e última fase deste trabalho, conseguiremos desenvolver um aplicativo simples e funcional que possa ser implementado.

Bibliografia

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

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;  
}
```