Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2020/2021

Sistema de Gestão de Hotéis

André Martins (a89586), António Rodrigues (a89585),
José Ferreira (a89572), Rui Vieira (a89564)

novembro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Sistema de Gestão de Hóteis

André Martins (a89586), António Rodrigues (a89585),

José Ferreira (a89572), Rui Vieira (a89564)

novembro, 2020

Resumo

Neste relatório vamos desenvolver um Sistema de Gestão de Hotéis, apresentando as diferentes fases da elaboração e justificando as escolhas tomadas durante a produção do projeto. Por ser Portugal um país tão fortemente ligado ao turismo, este tema despertou o nosso interesse em relacionar esse setor com o conteúdo lecionado nesta unidade curricular.

A primeira parte deste projeto refere-se à definição do sistema, onde fundamentamos a implementação de uma base de dados relacional.

Na segunda parte apresentamos os requisitos de descrição, exploração e controlo por nós definidos.

Na terceira parte é apresentado o modelo concetual para a base de dados que queremos implementar. São identificadas as entidades, bem como os seus atributos e as relações entre as diferentes entidades.

Na quarta parte passamos para a implementação do modelo lógico, obtido através do modelo concetual previamente definido.

Na quinta e última parte “pusemos a mão na massa” e tratamos da implementação física da nossa base de dados, bem como do seu povoamento. Foram definidos, também, procedures, function e views, resultantes de traduzir as interrogações do utilizador em código SQL.

Área de Aplicação: Arquitetura de Sistemas de Bases de Dados.

Este documento retrata de forma detalhada a organização e execução de uma arquitetura de bases de dados de um Sistema de Gestão de Hotéis. Numa fase inicial do relatório serão apresentadas as fases do projeto – Contextualização do tema do relatório, Motivação e Objetivos, Análise de Requisitos e Estrutura do Projeto. Inicialmente serão identificadas as entidades envolvidas no projeto, assim como os respetivos atributos e relacionamentos. De seguida, vamos identificar as chaves candidatas a serem chaves primárias de cada identidade. Após conclusão do modelo conceptual, vamos prosseguir para o modelo lógico onde será implementada toda a estrutura da Base de Dados. Por último, vamos analisar e validar os dados através do método de normalização das relações existentes.

Palavras-Chave: Bases de Dados Relacionais, modelos conceptuais e lógicos, análise de requisitos, entidades, atributos, relacionamentos.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Neste relatório vamos apresentar uma análise, um planeamento, uma estruturação e, por fim, a implementação de um sistema de base de dados relacional que sirva de suporte e sustente à gestão hoteleira. À partida, e mesmo antes de nos focarmos no problema em mãos, sabemos que a nossa base de dados terá de ser um sistema de gestão de informação, sendo que vai guardar todos os dados da empresa e relacioná-los entre si, de modo a que possamos extrair informação relevante. O sistema de gestão de hotéis centra-se na base de dados implementada e, por isso, um bom planeamento e estruturação são cruciais para a eficácia do sistema.

Neste primeiro capítulo, vamos introduzir brevemente o nosso projeto. Assim sendo, é necessário definir o contexto no qual se desenvolve o caso de estudo, seguido da sua descrição.

1.1. Contextualização de aplicação do sistema

O Hotel Apúlia Praia é um pequeno hotel na Vila da Apúlia, Esposende, Braga, situado no coração da Vila da Apúlia e perto da praia, banhada pelo oceano Atlântico.

Durante a situação pandêmica da COVID-19 e devido às restrições de saída do país, teve uma afluência superior à esperada. Entre os meses de junho e setembro o hotel estava esgotado e, dado o número de pré-reservas feitas para o verão de 2021, espera-se um cenário idêntico no verão do mesmo ano. Além da afluência surpreendente na época de verão, o hotel costuma receber turistas nos restantes meses do ano. Isto porque o hotel oferece toda uma visita guiada da Vila e uma apresentação e convivência com a sua cultura e estilo de vida.

O hotel tem vindo a crescer e começa a ser reconhecido como uma das melhores opções de hospedagem a curto prazo na costa da região minhota.

O hotel possui vários tipos de quartos diferentes e o preço de reserva por noite varia conforme o Tipo de Quarto pretendido.

1.2. Apresentação do Caso de Estudo

A estratégia de aproximação do turista à Vila, faz do hotel Apúlia Praia um ponto de entrada turística para a Vila e para Esposende. Com isto em mente, os dirigentes do hotel decidiram dar seguimento ao crescimento do hotel e criar a cadeia de Hotéis Esposende Praia. O objetivo será cobrir a área costeira entre a Praia da Ramalha e a Praia de Antas - praias mais a Sul e a Norte de Esposende, respectivamente - mantendo a oferta da aproximação cultural aos turistas. De momento, a cadeia é composta por 3 hotéis, havendo a possibilidade de expansão e criação de novos hotéis. Os hotéis Apúlia Praia, Esposende Praia e Ofir Praia têm-se revelado um sucesso e são a prova de que o investimento na expansão da cadeia foi uma aposta ganha pela direção da mesma.

Com esta expansão, a direção vê como crucial a criação de um sistema para a organização de reservas, serviços, funcionários e clientes para poder responder de forma rápida e eficaz às exigências dos clientes, funcionários e gestores. Aqui encontra-se a fundamentação para o nosso projeto. Foi proposta uma criação de uma base de dados que suporte um sistema de Gestão de Hotéis, apresentando para isso uma criteriosa análise de requisitos, a sua modelação e, por fim, a implementação física da base de dados.

1.3. Análise de Viabilidade do Processo

O ano de 2020 e todas as restrições aplicadas às saídas e entradas em Portugal na altura do verão, viram muitos portugueses optar pelas praias lusitanas para passar as férias. Verificou-se um grau de satisfação acima da média e é esperado para os próximos anos um aumento da afluência dos portugueses às praias do próprio país.

O aumento visível do turismo na região costeira minhota, motivou o desenvolvimento de um novo sistema que permitisse dar resposta imediata à reserva de quartos e serviços associados a cada hotel na nova cadeia de hotéis. Dado o foco deste trabalho na modelação de dados, encontram-se de seguida enumeradas as motivações que obtivemos para a realização deste projeto:

- Modelação de dados do funcionamento de um hotel
    - Gestão de Clientes
    - Gestão de Funcionários
    - Gestão de Quartos
    - Gestão de Tipos de Quarto
    - Gestão de Reservas

O sistema desenvolvido tem o objetivo de servir de suporte ao crescimento da cadeia de Hotéis Esposende Praia. A curto prazo deve substituir de forma eficaz o sistema antiquado do hotel Apúlia Praia e a longo prazo deve servir de sustento por muito tempo, evitando a sua substituição num curto espaço de tempo, deve servir os requisitos do sistema e garantir que o sistema se mantém funcional e otimizado durante todo o processo de crescimento da cadeia de hotéis.

2. Levantamento e análise de Requisitos

Neste capítulo do relatório de projeto, vamos explorar o levantamento de requisitos e a análise feita aos resultados desse levantamento.

2.1. Método de levantamento e de análise de requisitos adotado

O levantamento de requisitos para o sistema de base de dados foi feito através de entrevistas aos administrativos, funcionários e gerentes do hotel Apúlia Mar e à administração da cadeia Esposende Mar. Além destas entrevistas, aproveitamos o senso comum e a nossa própria experiência com hotelaria para tentar implementar um sistema completamente capaz de lidar com as necessidades relacionadas com a gestão e funcionamento de cada hotel de uma cadeia de hotéis.

2.2. Requisitos levantados

Através da análise da informação recolhida, procedemos à definição dos requisitos que a base de dados deverá suportar. Procuramos dotar a informação de uma maior estruturação, delimitando o papel de cada uma das componentes que formam o ambiente de negócio da empresa. Deste processo resultou a

diferenciação entre o ponto de vista do cliente e a perspetiva do administrador do sistema. Isto já era espectável, uma vez que a visão e objetivos que têm sobre a mesma base de dados são diferentes.

2.2.1. Requisitos de descrição

Nesta secção vamos apresentar os requisitos relacionados com a descrição das várias componentes do sistema.

2.2.1.1. Hotel

A cadeia onde vai ser implementado o sistema é uma cadeia de Hotéis.

Na caracterização de cada um dos hotéis devemos incluir a informação que nos permita identificar o hotel em questão. Os dados guardados sobre cada hotel incluem o seu nome, o endereço de morada, de quantas estrelas é o hotel, um URL para o seu site e um número único que o permita identificar. Estes dados são fornecidos ao sistema aquando do registo de um novo Hotel pelo administrador do sistema.

Acedendo ao sistema com as credenciais de Administrador, este pode aceder a toda a informação de cada Hotel e alterá-la caso seja necessário, podendo também adicionar um hotel ao sistema com todas as informações acima referidas.

2.2.1.2. Funcionário

Um Funcionário é uma componente essencial para o bom funcionamento do Hotel que o emprega. É, então, importante que seja armazenada informação relativa aos funcionários de cada hotel.

Os dados armazenados sobre cada funcionário incluem um contacto de email e telemóvel, o seu nome, morada, número de cartão de cidadão, o seu cargo e turno no hotel, e por fim um número único que o identifica. Estes dados são fornecidos ao sistema pelo administrador aquando do processo de registo de um novo funcionário.

Estes dados permitem ao administrador distribuir os funcionários pelos turnos de funcionamento do hotel, assim como evitar faltas de pessoal nas várias áreas de serviço do hotel como a cozinha, recepção, segurança, etc. Permitem ainda que os dados dos funcionários sejam alterados/atualizados e que sejam registados novos funcionários.

Cada funcionário é empregado de um e apenas um hotel.

2.2.1.3. Quarto

O principal serviço prestado pelos hotéis é o aluguer de quartos. Assim, estes assumem um papel essencial no desenvolvimento de um sistema feito para gestão hoteleira.

Os dados armazenados incluem um número único que o permite distinguir todos os quartos de forma singular e o estado do quarto.

Estes dados permitem ao administrador distinguir os quartos entre si e permitem ainda perceber se um dado quarto está livre ou ocupado, assim como verificar a ocupação total do hotel. Permitem ainda a atualização/alteração de informação de quartos já existentes no sistema e a adição de um quarto novo ao sistema.

Da perspetiva do cliente, estes dados permitem saber se um quarto está disponível para alugar ou se está ocupado nas datas pretendidas.

2.2.1.4. Tipo de Quarto

Dada a grande quantidade e variedade de quartos disponíveis (em termos de serviço e capacidade) nos hotéis da cadeia, é necessário organizá-los por tipos de quartos.

Os dados armazenados incluem uma designação para o tipo de quarto (que é única para cada tipo e nos permite distinguir entre os tipos de quartos diferentes), um preço associado ao aluguer de um quarto desse tipo por noite e a capacidade dos quartos desse mesmo tipo.

Todos os quartos têm um e um só tipo de quarto associado. Este é responsável pelo preço de aluguer de um quarto, juntamente com a duração da estadia.

Estes dados permitem ao administrador alterar informação relativa aos tipos de quartos, assim como adicionar novos tipos. Da perspetiva do cliente, permite-lhe perceber as diferenças entre os diferentes tipos de quartos que cada hotel tem para oferecer e o preço de alugar um quarto de um tipo de quarto em função da duração da estadia pretendida.

A seguinte tabela descreve a variação dos preços segundo o Tipo do Quarto.

Tipo de Quarto	Capacidade	Preço associado
Singular	1	50€
Duplo	2	90€
Triplo	3	150€
Quádruplo	4	200€
Suíte Dupla	2	350€

Tabela 1 - Preço por Tipo de Quarto

2.2.1.5. Reserva

As reservas são feitas pelos clientes que pretendem alugar um ou mais quartos num determinado hotel. Os dados associados a cada reserva são o seu custo, a data em que a reserva foi feita, data de início e final da reserva e um número único que identifica e distingue cada reserva das demais.

O custo é calculado através do preço do Tipo do Quarto associado aos quartos da reserva e da duração da reserva.

Estes dados permitem ao administrador do sistema ver informação sobre datas de maior afluência, tipos de quartos e quartos mais requisitados e recolher dados para formular a faturação do hotel. Permitem também ao cliente efetuar reservas, cancelar reservas, visualizar as reservas feitas num certo hotel, os gastos em reservas em cada hotel e visualizar informações específicas a cada reserva.

2.2.1.6. Cliente

Os clientes são registados no sistema quando efetuam uma reserva num hotel. Nesse momento são pedidas algumas informações ao cliente para serem armazenadas no hotel no registo individual de cada cliente.

Os dados armazenados incluem o nome do cliente, o seu número de CC, a sua morada, contacto de email e telemóvel ou telefone e o seu país de origem, juntamente com um número único que permite identificar cada cliente unicamente.

Estes dados permitem ao administrador recolher informações sobre os clientes de cada hotel, assim como ver o histórico das reservas feitas por cada cliente. Ao cliente é permitida a alteração dos seus dados pessoais

2.2.2. Requisitos de exploração

Do ponto de vista do cliente, deve ser possível:

- (RE1) - Ver quartos livres por hotel
- (RE2) - Consultar o preço de alugar um quarto de um certo tipo numa determinada data.
- (RE3) - Consultar a reserva de maior valor num determinado espaço de tempo
- (RE4) - Visualizar o histórico de reservas feitas entre duas datas

Do ponto de vista do administrador, deve ser possível:

- (RE8) - Visualizar as informações referentes a cada funcionário.
- (RE9) - Consultar a disponibilidade de um quarto.
- (RE10) - Consultar a ocupação total do hotel.
- (RE11) - Consultar quais as datas de maior afluência.
- (RE13) - Consultar o total de faturação do hotel.

2.2.3. Requisitos de controlo

Do ponto de vista do cliente deve ser possível:

- (RC1) - Registar uma nova reserva.
- (RC2) - Cancelar o registo de uma reserva.
- (RC3) - Inserir os dados para o registo.
- (RC4) - Atualizar as informações inseridas aquando do registo.

Do ponto de vista do administrador deve ser possível:

- (RC5) - Registar um hotel.
- (RC6) - Atualizar as informações inseridas aquando do registo de um hotel.
- (RC7) - Registar um funcionário.
- (RC8) - Atualizar as informações inseridas aquando do registo de um funcionário.
- (RC9) - Registar um quarto.

- (RC10) - Atualizar as informações inseridas aquando do registo de um quarto.
- (RC11) - Registar um novo tipo de quartos.
- (RC12) - Atualizar as informações inseridas aquando do registo de um tipo de quarto.

2.3. Análise e validação geral dos requisitos

A proposta de requisitos foi apresentada à administração da cadeia Esposende Mar.

Os requisitos de exploração registados revelaram ser bastante descritivos do modo de funcionamento pretendido do sistema.

Os requisitos de controlo demonstram as ações possíveis ao administrador e ao cliente do sistema pretendidas pela administração.

Considerando os requisitos de exploração e controlo, torna-se evidente o papel de cada um dos utilizadores da base de dados e o que se pretende desta.

Pudemos então concluir que os requisitos apresentados identificam corretamente os dados que serão guardados na base de dados e o que se pretende atingir com o sistema.

3. Modelo Conceptual

Concluída a fase de levantamento de requisitos e aprovados os resultados da mesma, tendo-os já analisado detalhadamente, passamos à próxima etapa da criação do sistema de base de dados - a modelação conceptual do mesmo.

O modelo conceptual é independente dos detalhes de implementação e procura representar de forma fidedigna o modelo de informação pretendido pela cadeia Esposende Praia. Para facilitar a definição e interpretação do modelo, este é acompanhado de documentação, composta por um diagrama ER e uma explicação sobre as entidades, relações e atributos identificados.

3.1. Apresentação da abordagem de modelação realizada

A abordagem de modelação utilizada na realização do modelo baseia-se em 6 passos:

1. Identificação e caracterização das entidades.
2. Identificação e caracterização dos relacionamentos.
3. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos.
4. Detalhe ou generalização das entidades.
5. Apresentação e explicação do Diagrama ER.
6. Validação do modelo de dados produzido.

Assim, foi possível definir o modelo conceptual da maneira que apresentamos de seguida.

3.2. Identificação e caracterização das entidades

A primeira tarefa necessária para a construção do nosso modelo conceptual passa por se identificarem as entidades do problema, ou seja, os principais componentes para a estruturação da nossa empresa. Para tal, é necessária uma observação cuidadosa do problema em questão pois embora tenhamos um objeto com uma importância significativa para o modelo, pode não ser necessariamente uma entidade.

Neste capítulo iremos apresentar as entidades do problema, o seu significado e também a justificação da sua ocorrência.

3.2.1. Hotel

O **Hotel** é, claramente, uma entidade essencial e básica na implementação de uma base de dados sobre o próprio **Hotel**.

A entidade **Hotel** vai conter toda a informação referente a um e um só hotel da cadeia Esposende Mar. Todas as informações guardadas nesta entidade são essenciais para permitir estudos estatísticos (preferência de localidade e hotel, por exemplo) e distinguir os hotéis dentro da base de dados e do sistema.

3.2.2. Quarto

O **Quarto** é uma entidade fulcral no nosso trabalho, uma vez que é o principal serviço prestado pelo **Hotel**. Com a informação relativa a cada quarto, garantimos que nenhum quarto é reservado para dois clientes ao mesmo tempo e que as datas de reserva não se sobrepõem.

3.2.3. Tipo de Quarto

O **Tipo de Quarto** é uma entidade essencial na capacidade de oferta de serviços e informações aos clientes.

Esta entidade representa as várias opções de alojamento disponíveis nos hotéis, sendo que são essas que diferenciam os preços das reservas.

Ao tratar o **Tipo de Quarto** como uma entidade, qualquer alteração a ser feita num dado **Tipo de Quarto** será muito menos custosa do que seria se o tratássemos como um atributo de cada **Quarto**.

Assim, aumentamos a eficiência do nosso sistema e também a sua organização, uma vez que é muito mais simples procurar um quarto específico no seu tipo do que numa lista que contém todos os quartos de cada hotel.

3.2.4. Funcionário

O **Funcionário** é retratado como uma entidade, uma vez que a distribuição de horários e tarefas pelos funcionários do hotel são funcionalidades essenciais na visão da administração da cadeia.

Esta entidade abrange todos os funcionários, desde os gerentes aos funcionários de limpeza, cozinha e técnicos do hotel. Apenas assim conseguimos garantir que a entidade não necessita de uma total reestruturação se o sistema for expandido. Garantimos a longevidade do sistema ao prepará-lo para receber muita e variada informação a partir de uma entidade genérica que abrange muitos casos possíveis (cargos e horários, por exemplo).

3.2.5. Reserva

A **Reserva** é a entidade que representa cada aluguer de um **Quarto**. É uma entidade essencial, uma vez que é o registo dos serviços passados, presentes e futuros.

A partir da reserva vamos conseguir retirar informação dos clientes da cadeia, assim como informação sobre a faturação de cada hotel, entre outras.

Sendo a entidade que representa o serviço principal da cadeia e de cada um dos seus hotéis, assume um papel central no nosso sistema, sendo o ponto de ligação entre o **Hotel** e **Cliente** e entre **Quarto** e **Cliente**.

3.2.6. Cliente

Decidimos tratar o **Cliente** como uma entidade, uma vez que, aquando das entrevistas, a administração da cadeia se mostrou interessada em oferecer aos clientes sugestões com base nas suas visitas prévias. Por exemplo, se um cliente for habitual num dado hotel numa certa altura do ano, ser-lhe-á recomendada a renovação de reserva para a mesma data no ano seguinte, no mesmo hotel. De igual modo, se um cliente estiver a percorrer os vários hotéis da cadeia, ser-lhe-á recomendado experimentar um hotel novo com experiências diferentes.

3.3. Identificação e caracterização dos relacionamentos

Depois de termos identificado todas as entidades que constituem o nosso modelo, procedemos agora à identificação dos relacionamentos que existem entre elas.

Funcionário e Hotel:

O relacionamento “emprega” entre as entidades **Hotel** e **Funcionário**, caracteriza-se pela cardinalidade de (1,1) para (1,N), uma vez que um **Hotel** tem de ter obrigatoriamente um ou mais **Funcionários** mas cada **Funcionário** trabalha num e um só **Hotel**, não havendo nenhum **Funcionário** registado que não tenha um Hotel associado.

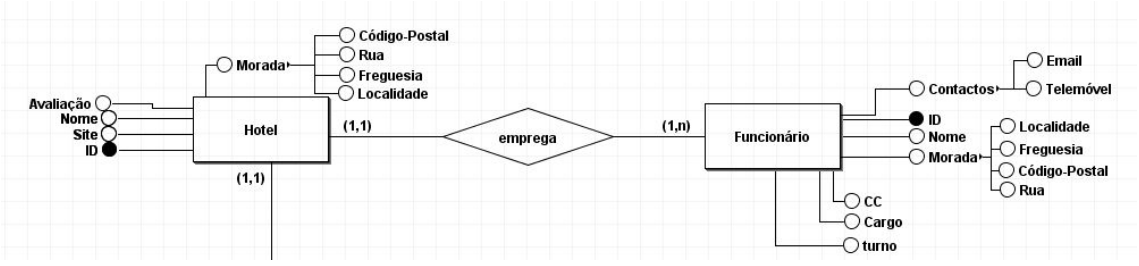

Figura 1 - Relacionamento entre Hotel e Funcionário

Hotel e Quarto:

O relacionamento “pertencem a” entre as entidades **Hotel** e **Quarto**, caracteriza-se pela cardinalidade de (1,1) para (1,N),uma vez que um **Hotel** tem obrigatoriamente um ou mais **Quartos** mas cada **Quarto** pertencem obrigatoriamente a um e um só **Hotel**.

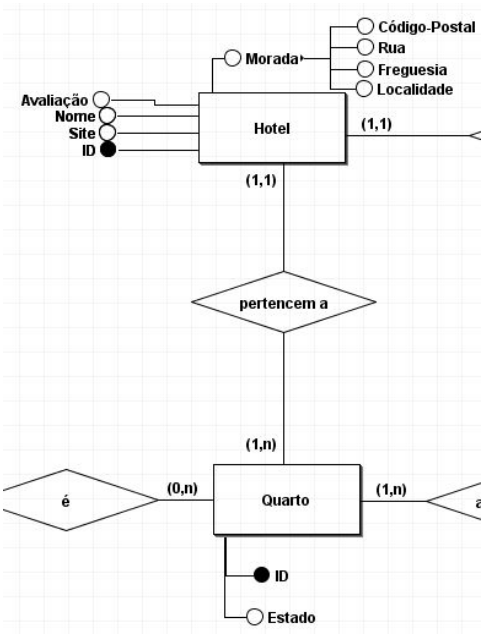

Figura 2 - Relacionamento entre Hotel e Quarto

Tipo de Quarto e Quarto:

O relacionamento "é" entre as entidades **Tipo de Quarto** e **Quarto**, caracteriza-se pela cardinalidade de (1,1) para (0,N), uma vez que cada **Quarto** tem obrigatoriamente um e um só tipo, enquanto cada **Tipo de Quarto** pode, ou não, ter um ou mais **Quartos** desse mesmo **Tipo**.

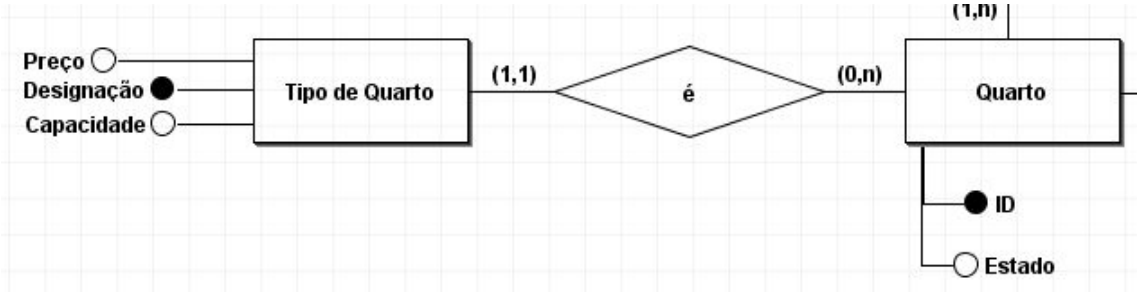

Figura 3 - Relacionamento entre Tipo de Quarto e Quarto

Quarto e Reserva:

O relacionamento "associada a" entre as entidades **Quarto** e **Reserva**, caracteriza-se pela cardinalidade de (1,N) para (0,N), uma vez que cada **Quarto** pode ou não ter uma ou mais **Reservas** associadas a si próprio, enquanto cada **Reserva** tem obrigatoriamente um ou mais **Quartos** associados.

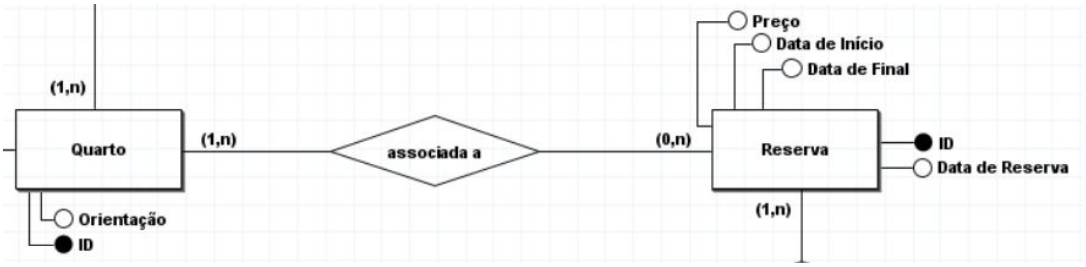

Figura 4 - Relacionamento entre Reserva e Quarto

Reserva e Cliente:

O relacionamento “feito por” entre as entidades **Reserva** e **Cliente**, caracteriza-se pela cardinalidade de (1,N) para (1,1), uma vez que cada **Reserva** é feita obrigatoriamente por um e um só **Cliente**, enquanto um **Cliente** pode ter uma ou mais **Reservas** associadas a si próprio.

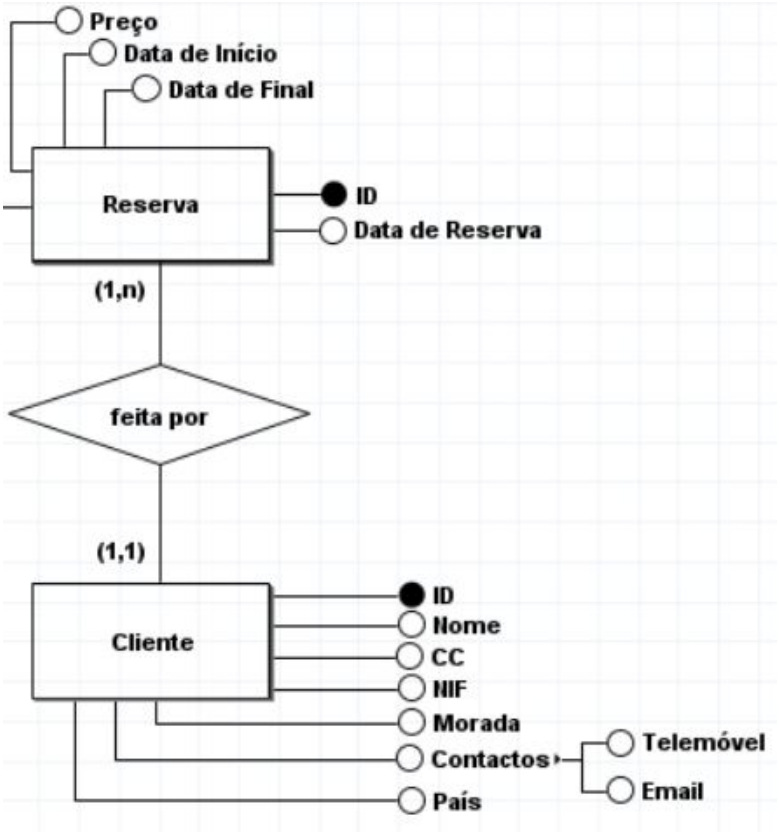

Figura 5 - Relacionamento entre Cliente e Reserva

De modo a garantir uma leitura mais simples dos relacionamentos do nosso modelo, apresentamos o Dicionário de Relacionamentos do mesmo.

Entidade	Cardinalidade	Relacionamento	Cardinalidade	Entidade
Hotel	(1,1)	emprega	(1,N)	Funcionário
Hotel	(1,1)	pertencem a	(1,N)	Quarto
Tipo de Quarto	(1,1)	é	(0,N)	Quarto
Quarto	(1,N)	associada a	(0,N)	Reserva
Reserva	(1,N)	feito por	(1,1)	Cliente

Tabela 2 - Dicionário de Relacionamentos do Modelo Conceptual

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

Depois de termos identificado todas as entidades do nosso problema e de as termos relacionado entre si, passamos agora ao momento de identificarmos todas as suas propriedades/atributos que consideramos relevantes para o problema em causa.

Entidade	Atributo	Descrição	Tipo de Atributo
Hotel	ID	Código único de cada Hotel	Chave primária
	Nome	Nome do Hotel	Simples
	Site	URL para o site do Hotel	Simples
	Avaliação	Avaliação de 0 a 5 estrelas do Hotel	Simples
	Morada(1)	Endereço de morada do Hotel	Composto

(1) - Apresentam-se na Tabela abaixo os sub-atributos do atributo Morada

Tabela 3 - Atributos da Entidade Hotel

Atributo	Sub-atributo	Descrição	Tipo de atributo
Morada[1]	Código-Postal	Código-Postal do Hotel	Simples
	Rua	Rua do Hotel	Simples
	Freguesia	Freguesia do Hotel	Simples
	Localidade	Localidade do Hotel	Simples

Tabela 4 - Sub-atributos do Atributo Morada

Entidade	Atributo	Descrição	Tipo de Atributo
Funcionário	ID	Código único de cada Funcionário	Chave Primária
	Nome	Nome do Funcionário	Simples
	CC	Número de Identificação Civil do Funcionário	Simples
	Cargo	Cargo do Funcionário no Hotel	Simples
	Turno	Turno de trabalho do Funcionário	Simples
	Contactos[2]	Contactos do Funcionário	Composto
	Morada[3]	Morada do Funcionário	Composto

[2],[3]- Apresentam-se na Tabela abaixo os sub-atributos dos atributos Contactos e Morada

Tabela 5 - Atributos da entidade Funcionário

Atributo	Sub-atributo	Descrição	Tipo de atributo
Contactos	Email	Endereço de e-mail do Funcionário	Simples

	Telemóvel	Número de telemóvel do Funcionário	Simples
Morada	Código-Postal	Código-Postal da morada do Funcionário	Simples
	Rua	Rua da morada do Funcionário	Simples
	Freguesia	Freguesia da morada do Funcionário	Simples
	Localidade	Localidade da morada do Funcionário	Simples

Tabela 6 - Sub-atributos dos Atributos Contactos e Morada

Entidade	Atributo	Descrição	Tipo de Atributo
Tipo de Quarto	Designação	Nome dado ao Tipo de Quarto	Chave Primária
	Preço	Preço por noite do aluguer de um Quarto de determinado Tipo de Quarto	Simples
	Capacidade	Capacidade máxima do Quarto (em pessoas adultas)	Simples

Tabela 7 - Atributos da entidade Tipo de Quarto

Entidade	Atributo	Descrição	Tipo de Atributo
Quarto	ID	Código único de cada Quarto	Chave Primária
	Estado	Diz-nos se um Quarto está ocupado ou livre	Simples

Tabela 8 - Atributos da entidade Quarto

Entidade	Atributo	Descrição	Tipo de Atributo
Reserva	ID	Código único da Reserva	Chave Primária
	Data da Reserva	Data em que a reserva foi efetuada	Simples
	Data de Início	Data de início de aluguer (check-in)	Simples
	Data de Final	Data de final de aluguer (check-out)	Simples
	Preço	Preço da Reserva (Tipo de Quarto * nº de noites)	Derivado

Tabela 9 - Atributos da entidade Reserva

Entidade	Atributo	Descrição	Tipo de Atributo
Cliente	ID	Código único de cada Cliente	Chave Primária
	Nome	Nome do Cliente	Simples
	CC	Número de Identificação Civil do Cliente	Simples
	NIF	Número de Identificação Fiscal do Cliente	Simples (Opcional)
	Morada	Morada do Cliente	Simples
	País	País de origem do Cliente	Simples
	Contactos[4]	Contactos do Cliente	Composto

[4]- Apresentam-se na Tabela abaixo os sub-atributos do atributo Contactos

Tabela 10 - Atributos da entidade Cliente

Atributo	Sub-atributo	Descrição	Tipo de atributo
Contactos	Email	Endereço de e-mail do Cliente	Simples
	Telemóvel	Número de telemóvel do Cliente	Simples

Tabela 11 - Sub-atributos do atributo Contactos

3.5. Detalhe ou generalizações das entidades

Uma vez que as entidades apresentadas representam objetos diferentes no nosso sistema, não existe a necessidade de proceder ao detalhe ou generalização das mesmas.

Não é necessário utilizar a generalização, uma vez que os atributos associados a cada entidade variam. Isto é, não existem duas entidades com o mesmo conjunto de atributos. De igual modo, não é necessário recorrer ao detalhe de entidades, uma vez que as ocorrências das entidades de um mesmo tipo terão características semelhantes, acontecerão sobre contextos idênticos e, acima de tudo, representarão objetos também eles idênticos.

3.6. Apresentação e explicação do diagrama ER

Apresentadas todas as entidades, os relacionamentos entre elas e os atributos correspondentes, passamos à apresentação do Diagrama ER.

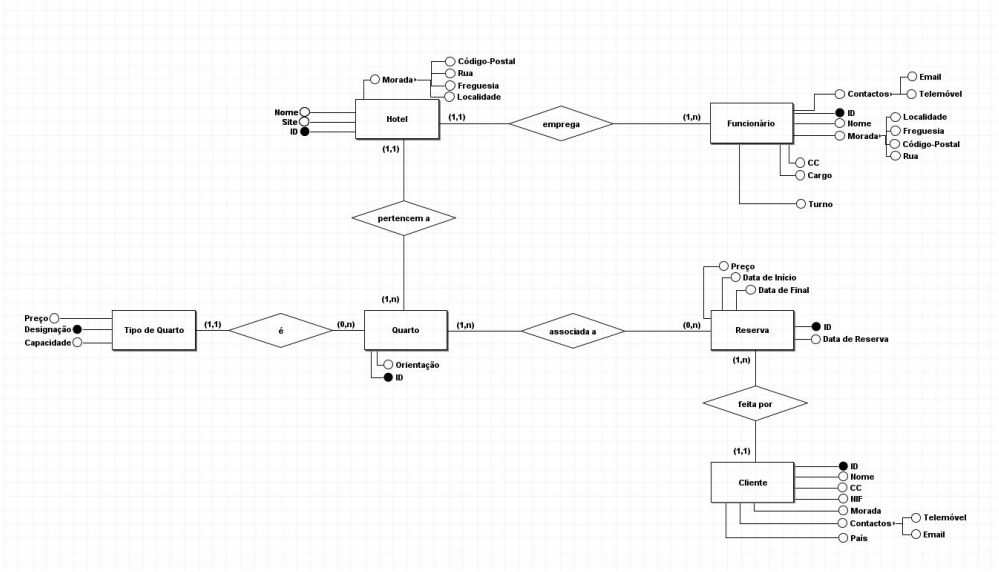

Figura 6 - Modelo Conceptual

Analisando o Diagrama apresentado, constatamos que as entidades e os atributos que lhes correspondem se encontram em conformidade com os Requisitos enunciados no Capítulo 2 deste documento. A justificação para a escolha das entidades, dos relacionamentos e dos atributos encontram-se nos subcapítulos acima.

Na entidade **Hotel**, o atributo "Morada" é único de **Hotel** para **Hotel** e seria candidato a Chave Primária, no entanto, dado que este atributo é composto, achamos por bem ter um atributo mais simples de representar para ser a Chave Primária da entidade **Hotel**. Assim, escolhemos o atributo "ID" como Chave Primária da entidade **Hotel**.

Na entidade **Funcionário**, o atributo "CC" seria candidato a Chave Primária, no entanto, a sua definição encontra-se fora do domínio do sistema e, em comparação com o atributo "ID", o seu valor numérico será certamente superior. Uma vez que por cada **Funcionário** é guardado um email e um número de telemóvel, estes atributos também seriam candidatos a Chave Primária, contudo o **Funcionário** pode alterar os contactos, o que afasta estes atributos da seleção para Chave Primária da entidade **Funcionário**. Assim, escolhemos o atributo "ID" como Chave Primária da entidade **Funcionário.**

Na entidade **Quarto**, o atributo Estado varia apenas entre os valores "Livre" e "Ocupado", podendo vários **Quartos** ter o mesmo valor para este atributo. Assim, escolhemos o atributo "ID" como Chave Primária da entidade **Quarto**.

Na entidade **Tipo de Quarto**, os atributos "Preço" e "Capacidade" não assumem um valor único para cada **Tipo de Quarto**. Assim, escolhemos o atributo "Designação" como Chave Primária da entidade **Tipo de Quarto**.

Na entidade **Reserva**, os atributos "Data de Início", "Data de Final", "Data de Reserva" e "Preço" não assumem valores únicos para cada **Reserva**. Assim sendo, escolhemos o atributo "ID" como Chave Primária da entidade **Reserva**.

Na entidade **Cliente**, tanto o atributo "CC" como o atributo "NIF" seriam candidatos a Chave Primária, no entanto, a sua definição encontra-se fora do domínio do sistema e, em comparação com o atributo "ID", o seu valor numérico será certamente superior. Uma vez que por cada **Cliente** é guardado um email e um número de telemóvel, estes atributos também seriam candidatos a Chave Primária, contudo o **Cliente** pode alterar os seus contactos, o que afasta estes atributos da seleção para Chave Primária da entidade **Cliente**. Assim, escolhemos o atributo "ID" como Chave Primária da entidade **Cliente.**

3.7. Validação do modelo de dados produzido

Uma vez concluído o processo de modelação conceptual da base de dados, procedemos à apresentação do modelo de dados à administração da cadeia Esposende Praia de forma a perceber se o modelo abrange os requisitos pedidos. Neste encontro, explicamos passo a passo a nossa estratégia para responder a cada um dos requisitos previamente levantados na fase Levantamento de Requisitos.

- É possível distinguir os vários hotéis da cadeia pela informação representada como atributos da entidade **Hotel**.
- É possível aceder à informação de todos os Funcionários de um Hotel em função do seu cargo e/ou turno para escalonamento dos mesmos.
- É possível aceder à informação de um Funcionário de um Hotel caso seja necessário contactá-lo por algum motivo. Este contacto é garantido em 3 meios pela informação guardada: Correios, Correio Eletrónico e por Telemóvel.
- É possível aceder à informação referente a todos os Quartos de um certo Hotel de modo a perceber quantos são de cada Tipo de Quarto e quantos estão ocupados ou livres a cada momento.
- É possível, a partir das Reservas associadas a cada Quarto, impedir que duas reservas sejam feitas para o mesmo quarto em datas coincidentes.
- É possível, a partir das Reservas, perceber em que datas são feitas mais reservas, assim como quais as datas de maior procura.
- É possível, a partir das Reservas, ter a informação referente à faturação do Hotel, através do atributo “Preço”. Isto permite-nos também ver quanto capital foi gerado por Quarto, Funcionário, entre outras medidas estatísticas importantes na gestão de um Hotel.
- É possível aceder à informação do Cliente que fez cada uma das Reservas de modo a perceber os países de origem de turistas que mais procuram a cadeia e os seus hotéis.
- É também possível ver as Reservas feitas por um Cliente num certo Hotel.

4. Modelação Lógica

Neste capítulo do relatório de projeto, vamos explorar a modelação lógica para o sistema idealizado para a implementação na cadeia de hotéis.

4.1. Construção e validação do modelo de dados lógico

A partir da derivação de todas as relações que representam os relacionamentos, as entidades e os atributos identificados anteriormente no Modelo Conceptual iniciamos a criação do Modelo Lógico.

4.2. Desenho do modelo lógico

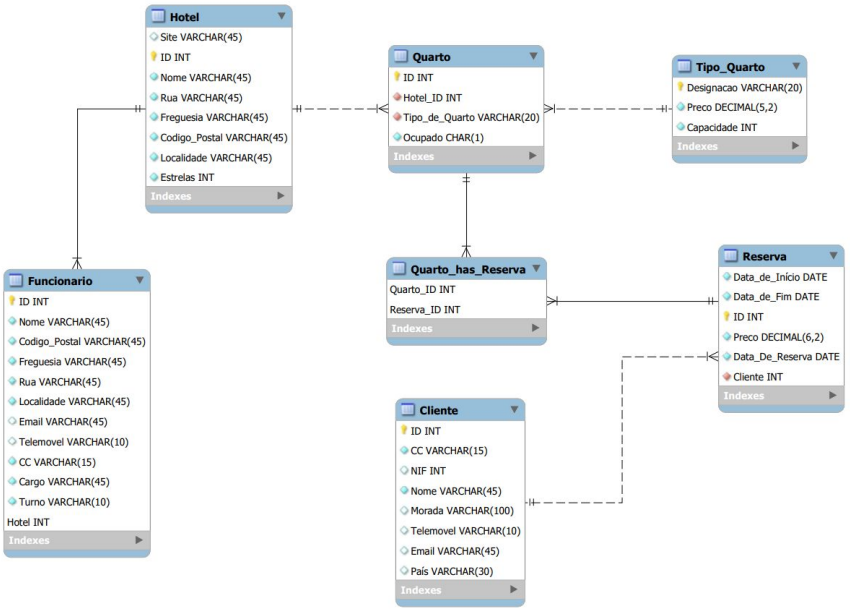

Figura 7 - Modelo Lógico

4.3. Validação do modelo através da normalização

O modelo relacional é um modelo de dados representativo (ou de implementação), adequado a ser o modelo subjacente de um Sistema Gerenciador de Banco de Dados (SGBD), que se baseia no princípio de que todos os dados estão armazenados em tabelas (ou, matematicamente falando, relações). Toda sua definição é teórica e baseada na lógica de predicados e na

teoria dos conjuntos. A partir deste modelo é possível construir um Modelo Lógico minimamente redundante e com uma estrutura consistente.

Relativamente à normalização de banco de dados, baseia-se num conjunto de regras que visa, principalmente, a organização de um projeto de banco de dados para reduzir a redundância dos mesmos, tal como foi dito anteriormente, aumentar a integridade de dados e o desempenho. Para normalizar o banco de dados, deve-se examinar as colunas (atributos) de uma entidade e as relações entre entidades (tabelas), com o objetivo de se evitar anomalias observadas na inclusão, exclusão e alteração de registros.

Começamos com a identificação das dependências funcionais entre os atributos de forma a proceder à validação do modelo por via da normalização.

Dependências Funcionais de Funcionario

ID - Contactos, Nome, Codigo_Postal, Freguesia, Rua, Localidade, Email, Telemovel, CC, Cargo, Turno

Dependências Funcionais de Hotel

ID - Site, Nome, Rua, Freguesia, Codigo_Postal, Localidade, Estrelas

Dependências Funcionais de Quarto

ID - Hotel_ID, Tipo_de_Quarto, Ocupado

Dependências Funcionais de Reserva

ID - Data_de_Reserva, Data_de_Início, Data_de_Fim, Preco, Cliente

Dependências Funcionais de Cliente

ID - Nome, CC, NIF, Morada, País, Telemóvel, Email

Dependências Funcionais de Tipo_Quarto

Designacao, Preco, Capacidade

Dependências Funcionais de Quarto_has_Reserva

Quarto_ID, Reserva_ID

Para verificar o nosso modelo através de normalização temos de verificar se o mesmo respeita a 1ª Forma Normal, a 2ª Forma Normal e, por fim, a 3ª Forma Normal.

1. Como nenhuma relação apresenta atributos multivalorados ou grupos repetidos, estas estão na 1ª Forma Normal, isto é, a interseção de cada coluna com cada linha apresenta apenas um valor.
2. Todos os atributos das relações são totalmente dependentes da chave primária (sem dependências parciais). O modelo respeita assim a 2ª Forma Normal.
3. Como todas as relações estão na 1ª e 2ª Forma Normal e nenhum dos atributos apresenta dependências transitivas, bem como, nenhum atributo de nenhuma relação depende de outro sem ser a chave primária, também a 3ª Forma Normal é respeitada por este modelo.

Concluímos assim que este modelo está normalizado, pois todas as relações estão em acordo com a 3ª Forma Normal.

4.4. Validação do modelo com interrogações do utilizador

Após a construção do modelo lógico e da sua normalização, necessitamos de verificar se este era capaz de responder aos requisitos do utilizador, testando assim a validação através de interrogações por parte deste.

É possível consultar a quantidade de quartos livres por hotel?

Começamos por selecionar da tabela dos hotéis, o hotel pretendido. De seguida verificamos a tabela de quartos do nosso hotel e finalmente selecionamos na tabela apenas os quartos que se encontram livres, obtendo assim a resposta pretendida à interrogação.

É possível visualizar o histórico de reservas feitas entre duas datas?

Acendendo à tabela de reservas, selecionamos aquelas que estão no intervalo temporal entre as duas datas pretendidas, chegando à informação necessária para responder à query.

É possível visualizar as informações referentes a cada funcionário?

Para obter a resposta a esta questão apenas é necessário selecionar a tabela dos funcionários.

É possível consultar a disponibilidade de um quarto?

Selecionando o quarto pretendido da lista dos quartos e verificando a sua ocupação temos a resposta.

É possível consultar a ocupação total do hotel?

Começando por selecionar o hotel da lista dos hotéis, vamos depois selecionar a lista de quartos do mesmo e, por fim, consultar quais destes estão ocupados, chegando à resposta à questão.

É possível consultar o total de faturação do hotel?

Tal como na questão anterior, começamos por selecionar o hotel da lista dos hotéis. De seguida vamos consultar a lista de reservas do mesmo, verificamos o preço de cada e adicionando-os, obtemos a solução.

É possível fazer uma simulação do preço de uma reserva?

Através da subtração entre a data do final da "reserva" e a data inicial da mesma, vamos chegar ao número de dias de "reserva". Depois, consultando a lista de quartos e selecionando o quarto pretendido, verificamos o preço e multiplicamos pelo número de dias, obtendo assim uma simulação do preço de uma "reserva".

É possível consultar a reserva de maior valor num determinado espaço de tempo?

Organizando a lista de reservas que obtemos após filtrar a lista de reservas pelo determinado espaço temporal pretendido decrescentemente em relação ao preço e acedendo ao primeiro , obtemos a resposta a esta query.

4.5. Revisão do modelo lógico produzido

Após todo o processo realizado foi marcada uma reunião com a direção da cadeia de hóteis com o objetivo de rever os requisitos através das interrogações e transações. Como foi considerado pela direção e por nós que o trabalho estava

bem elaborado e a cumprir todos os requisitos pretendidos avançámos e começámos a elaborar o Modelo Físico para este projeto.

5. Implementação física

Para darmos por concluída a construção do sistema de base de dados, passamos à fase da implementação física, seguida do povoamento da base de dados, exploração e monitorização de toda a informação armazenada.

Primeiro, começamos por fazer a conversão do modelo lógico, apresentado no capítulo 4, para o SGBD (Sistema de Gestão de Base de Dados) escolhido. Uma vez feita a implementação e o povoamento, procedemos à definição em SQL de algumas ações de exploração e monitorização dos dados guardados.

Neste capítulo, além da descrição dos passos referidos acima, também fazemos uma estimativa do espaço em disco que a nossa base de dados irá ocupar.

5.1. Seleção do sistema de gestão de bases de dados

Para desenvolver o esquema físico da nossa base de dados tivemos que escolher qual o melhor SGBD a utilizar. Uma vez que estamos no modelo de dados relacional, optamos por utilizar o MySQL. Primeiro, por uma questão de familiarização com esta ferramenta, uma vez que foi a utilizada durante as aulas práticas. Depois, por ter mecanismos bastante intuitivos e "user friendly" e bastante úteis. Sendo este, também, um dos SGBD mais utilizados pela comunidade, teríamos também bastante apoio caso fosse necessário corrigir algum erro que possa ter surgido durante o desenvolvimento deste projeto.

5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhidos em SQL

No processo da implementação física, definimos as relações, retiradas do esquema lógico com o qual trabalhamos. Assim sendo, temos as seguintes relações:

- **Hotel**

  Domínio ID Hotel: Inteiro

  Domínio Site: String de comprimento variável, comprimento 45

  Domínio Nome: String de comprimento variável, comprimento 45

Domínio Rua: String de comprimento variável, comprimento 45

Domínio Freguesia: String de comprimento variável, comprimento 45

Domínio Código Postal: String de comprimento variável, comprimento 45

Domínio Localidade: String de comprimento variável, comprimento 45

Domínio Estrelas: Inteiro

Hotel(

ID	ID Hotel	NOT NULL,
Site	Site	NOT NULL,
Nome	Nome	NOT NULL,
Rua	Rua	NOT NULL,
Freguesia	Freguesia	NOT NULL,
Codigo_Postal	Código Postal	NOT NULL,
Localidade	Localidade	NOT NULL,
Estrelas	Estrelas	NOT NULL,

CHAVE PRIMÁRIA (ID)

);

Em código SQL:

```
-- -----------------------------------------------------
-- Table `Hotel`.`Hotel`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Hotel` (
  `Site` VARCHAR(45) NULL,
  `ID` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Freguesia` VARCHAR(45) NOT NULL,
  `Codigo_Postal` VARCHAR(45) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  `Estrelas` INT NOT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;
```

Figura 8 - Código SQL para criar tabela Hotel

- **Funcionário**

  Domínio ID Funcionário: Inteiro

  Domínio Nome: String de comprimento variável, comprimento 45

  Domínio Código Postal: String de comprimento variável, comprimento 45

Domínio Freguesia: String de comprimento variável, comprimento 45
Domínio Rua: String de comprimento variável, comprimento 45
Domínio Email: String de comprimento variável, comprimento 45
Domínio Telemóvel: String de comprimento variável, comprimento 10
Domínio Cartão de Cidadão: String de comprimento variável, comprimento 15
Domínio Cargo: String de comprimento variável, comprimento 15
Domínio Turno: String de comprimento variável, comprimento 10
Domínio Hotel: Inteiro

Funcionario

(ID	ID Funcionário	NOT NULL,
Nome	Nome	NOT NULL,
Codigo_Postal	Código Postal	NOT NULL,
Freguesia	Freguesia	NOT NULL,
Rua	Rua	NOT NULL,
Email	Email	NULL,
Telemovel	Telemovel	NULL,
CC	Cartão de Cidadão	NOT NULL,
Cargo	Cargo	NOT NULL,
Turno	Turno	NOT NULL,

CHAVE PRIMÁRIA (ID),
CHAVE ESTRANGEIRA (Hotel) REFERENCIA Hotel(ID)
ON UPDATE NO ACTION
ON DELETE NO ACTION);

Em código SQL:

```sql
-- -----------------------------------------------------
-- Table `Hotel`.`Funcionario`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Funcionario` (
  `ID` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Codigo_Postal` VARCHAR(45) NOT NULL,
  `Freguesia` VARCHAR(45) NOT NULL,
  `Rua` VARCHAR(45) NOT NULL,
  `Localidade` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NULL,
  `Telemovel` VARCHAR(10) NULL,
  `CC` VARCHAR(15) NOT NULL,
  `Cargo` VARCHAR(45) NOT NULL,
  `Turno` VARCHAR(10) NOT NULL,
  `Hotel` INT NOT NULL,
  PRIMARY KEY (`ID`, `Hotel`),
  INDEX `Hotel_idx` (`Hotel` ASC) VISIBLE,
  CONSTRAINT `Hotel`
    FOREIGN KEY (`Hotel`)
    REFERENCES `Hotel`.`Hotel` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

Figura 9 - Código SQL para criar tabela Funcionario

- **Tipo de Quarto**

  Domínio Designação: String de comprimento variável, comprimento 20

  Domínio Preço: Valor monetário, entre 000.00 e 999.99

  Domínio Capacidade: Inteiro

  Tipo_Quarto

(Designacao	Designação	NOT NULL,
Preco	Preço	NOT NULL,
Capacidade	Capacidade	NOT NULL,

  CHAVE PRIMÁRIA (Designacao));

  Em código SQL:

```
-- -----------------------------------------------------
-- Table `Hotel`.`Tipo_Quarto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Tipo_Quarto` (
  `Designacao` VARCHAR(20) NOT NULL,
  `Preco` DECIMAL(5,2) NOT NULL,
  `Capacidade` INT NOT NULL,
  PRIMARY KEY (`Designacao`))
ENGINE = InnoDB;
```

Figura 10 - Código SQL para criar tabela Tipo_Quarto

- **Quarto**

  Domínio ID Quarto: Inteiro

  Domínio ID Hotel: Inteiro

  Domínio Tipo de Quarto: String de comprimento variável, comprimento 20

  Domínio Ocupado: Caracter (‘1’ ou ‘0’)

  Quarto
  (ID ID Quarto NOT NULL,
  Hotel_ID ID Hotel NOT NULL,
  Tipo_de_Quarto Tipo de Quarto NOT NULL,
  Ocupado Ocupado NOT NULL,
  CHAVE PRIMÁRIA(ID)
  CHAVE ESTRANGEIRA (Hotel_ID) REFERENCIA Hotel(ID)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION,
  CHAVE ESTRANGEIRA (Tipo_De_Quarto) REFERENCIA Tipo_Quarto(Designacao) ON UPDATE NO ACTION
  ON DELETE NO ACTION);

  Em código SQL:

```
-- -----------------------------------------------------
-- Table `Hotel`.`Quarto`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Quarto` (
  `ID` INT NOT NULL,
  `Hotel_ID` INT NOT NULL,
  `Tipo_de_Quarto` VARCHAR(20) NOT NULL,
  `Ocupado` CHAR(1) NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `Hotel_idx` (`Hotel_ID` ASC) VISIBLE,
  INDEX `Tipo_Quarto_idx` (`Tipo_de_Quarto` ASC) VISIBLE,
  CONSTRAINT `Hotel_ID`
    FOREIGN KEY (`Hotel_ID`)
    REFERENCES `Hotel`.`Hotel` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Tipo_Quarto`
    FOREIGN KEY (`Tipo_de_Quarto`)
    REFERENCES `Hotel`.`Tipo_Quarto` (`Designacao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 11 - Código SQL para criar tabela Quarto

- **Cliente**

  Domínio ID Cliente: Inteiro

  Domínio Cartão de Cidadão: String de comprimento variável, comprimento 15

  Domínio Número de Identificação Fiscal: Inteiro

  Domínio Nome: String de comprimento variável, comprimento 45

  Domínio Morada: String de comprimento variável, comprimento 100

  Domínio Telemóvel: String de comprimento variável, comprimento 10

  Domínio Email: String de comprimento variável, comprimento 45

  Domínio País: String de comprimento variável, comprimento 30

  Cliente

(ID Cliente	ID Cliente	NOT NULL,
CC	Cartão de Cidadão	NOT NULL,
NIF	Nº Idenficação Fiscal	NULL,

Nome	Nome	NOT NULL,
Morada	Morada	NOT NULL,
Telemovel	Telemóvel	NOT NULL,
Email	Email	NOT NULL,
País	País	NULL,
CHAVE PRIMÁRIA (ID),		
);		

Em SQL:

```
-- -----------------------------------------------------
-- Table `Hotel`.`Cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Cliente` (
  `ID` INT NOT NULL,
  `CC` VARCHAR(15) NOT NULL,
  `NIF` INT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Morada` VARCHAR(100) NULL,
  `Telemovel` VARCHAR(10) NULL,
  `Email` VARCHAR(45) NULL,
  `País` VARCHAR(30) NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;
```

Figura 12 - Código SQL para criar tabela Cliente

- **Reserva**

  Domínio Data de Início: Temporal

  Domínio Data de Fim: Temporal

  Domínio ID Reserva: Inteiro

  Domínio Preço: Valor monetário entre 0000.00 e 9999.99

  Domínio Data da Reserva: Temporal

  Data Cliente: Inteiro

Reserva		
(Data_de_Início	Data de Início	NOT NULL,
Data_de_Fim	Data de Fim	NOT NULL,
ID	ID Reserva	NOT NULL,

Preco	Preço	NOT NULL,
Data_de_Reserva	Data de Reserva	NOT NULL,

CHAVE PRIMÁRIA (ID),

CHAVE ESTRANGEIRA (Cliente) REFERENCIA Cliente(ID)
ON UPDATE NO ACTION
ON DELETE NO ACTION);

Em SQL:

```
-- -----------------------------------------------------
-- Table `Hotel`.`Reserva`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Reserva` (
  `Data_de_Início` DATE NOT NULL,
  `Data_de_Fim` DATE NOT NULL,
  `ID` INT NOT NULL,
  `Preco` DECIMAL(6,2) NOT NULL,
  `Data_De_Reserva` DATE NOT NULL,
  `Cliente` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `Cliente_idx` (`Cliente` ASC) VISIBLE,
  CONSTRAINT `Cliente`
    FOREIGN KEY (`Cliente`)
    REFERENCES `Hotel`.`Cliente` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 13 - Código SQL para criar tabela Reserva

- **Quarto_has_Reserva**

  Domínio Quarto ID: Inteiro

  Domínio Reserva ID: Inteiro

  (

Quarto_ID	Quarto ID	NOT NULL,
Reserva_ID	Reserva ID	NOT NULL,

  CHAVE PRIMÁRIA (Quarto_ID, Reserva_ID),
  CHAVE ESTRANGEIRA (Quarto_ ID) REFERENCIA Quarto(ID)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION,

CHAVE ESTRANGEIRA (Reserva_ID) REFERENCAI Reserva(ID)
ON UPDATE NO ACTION
ON DELETE NO ACTION
);

Em SQL:

```
-- -----------------------------------------------------
-- Table `Hotel`.`Quarto_has_Reserva`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Hotel`.`Quarto_has_Reserva` (
  `Quarto_ID` INT NOT NULL,
  `Reserva_ID` INT NOT NULL,
  PRIMARY KEY (`Quarto_ID`, `Reserva_ID`),
  INDEX `fk_Quarto_has_Reserva_Reserva1_idx` (`Reserva_ID` ASC) VISIBLE,
  INDEX `fk_Quarto_has_Reserva_Quarto1_idx` (`Quarto_ID` ASC) VISIBLE,
  CONSTRAINT `fk_Quarto_has_Reserva_Quarto1`
    FOREIGN KEY (`Quarto_ID`)
    REFERENCES `Hotel`.`Quarto` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Quarto_has_Reserva_Reserva1`
    FOREIGN KEY (`Reserva_ID`)
    REFERENCES `Hotel`.`Reserva` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 14 - Código SQL para criar tabela Quarto_has_Reserva

5.3. Tradução das interrogações do utilizador para SQL (alguns exemplos)

Nesta secção, serão apresentados exemplos de código SQL que permitem obter as respostas às interrogações do utilizador, que são expressas pelos requisitos de exploração.

(RE1) Ver quartos livres

```sql
-- (RE1) Ver quartos livres.
DELIMITER $$
CREATE PROCEDURE disponibilidade_quartos(IN id_hotel INT)
BEGIN
    SELECT * FROM QUARTO AS q
    WHERE q.ocupado = '0' AND q.hotel_ID = id_hotel
    ORDER BY q.id ASC;
END $$
```

Figura 15 - Código SQL para RE1

(RE2) Simular o preço de uma reserva

```sql
-- (RE2) Simular o preço de uma reserva
DELIMITER $$
CREATE FUNCTION simular_reserva(data_inicio DATE, data_fim DATE, id_quarto INT)
        RETURNS FLOAT DETERMINISTIC
BEGIN
        RETURN (SELECT (SELECT Preco FROM Tipo_Quarto AS tq INNER JOIN Quarto q
                                        ON q.Tipo_de_Quarto = tq.Designacao
                                        WHERE q.id = id_quarto) * (SELECT DATEDIFF(data_fim, data_inicio)));
END $$
```

Figura 16 - Código SQL para RE2

(RE3) Consultar a reserva de maior valor num determinado espaço de tempo

```sql
-- (RE3) Consultar a reserva de maior valor num determinado espaço de tempo
DELIMITER $$
CREATE PROCEDURE maior_reserva(IN data_inicio DATE, data_fim DATE)
BEGIN
    SELECT r.Data_de_início, r.Data_de_fim, r.Preco, c.Nome, h.Nome
    FROM Reserva AS r INNER JOIN Cliente c
                      ON c.id = r.cliente
                      INNER JOIN quarto_has_reserva qr
                      ON qr.Reserva_ID = r.ID
                      INNER JOIN Quarto q
                      ON q.ID = qr.Quarto_ID
                      INNER JOIN Hotel h
                      ON q.Hotel_ID = h.ID
    WHERE r.data_de_início >= data_inicio AND r.data_de_fim <= data_fim
    ORDER BY Preco DESC
    LIMIT 1;
END$$
```

Figura 17 - Código SQL para RE3

(RE4) Visualizar o histórico de reservas feitas entre duas datas

```sql
-- (RE4) Visualizar o histórico de reservas feitas entre duas datas
DELIMITER  $$
CREATE PROCEDURE consulta_reservas (IN id_cliente INT, data_inicio DATE, data_fim DATE)
BEGIN
        SELECT * FROM Reserva AS r INNER JOIN Cliente AS c ON r.cliente = c.id
        WHERE r.cliente = id_cliente AND r.data_de_início >= data_inicio AND r.data_de_fim <= data_fim
    ORDER BY r.data_de_início ASC;

END $$
```

Figura 18 - Código SQL para RE4

(RE8) Visualizar as informações referentes a cada funcionário

```sql
-- (RE8)  Visualizar as informações referentes a cada funcionário.
 DELIMITER $$
CREATE PROCEDURE consulta_Funcionarios(IN id_hotel INT)
BEGIN
    SELECT * FROM FUNCIONARIO AS f
    WHERE f.hotel = id_hotel;
END $$
```

Figura 19 - Código SQL para RE8

(RE9) Consultar disponibilidade de um Quarto

```sql
-- (RE9) Consultar disponibilidade de um Quarto
DELIMITER $$
CREATE PROCEDURE disponibilidade(IN id_quarto INT)
BEGIN
    SELECT Ocupado FROM QUARTO AS q
    WHERE q.id = id_quarto;
END $$
```

Figura 20 - Código SQL para RE9

(RE10) Consultar a ocupação total do hotel

```
-- (RE10) Consultar a ocupação total do hotel.

DELIMITER $$
CREATE PROCEDURE Ocupacao(IN id_hotel INT)
BEGIN
    SELECT COUNT(*) AS 'Quartos Ocupados' FROM QUARTO AS q
    WHERE q.hotel_id = id_hotel AND q.ocupado = '1';
END $$
```

Figura 21 - Código SQL para RE10

(RE13) Consultar o total de faturação do hotel

```
-- (RE13) Consultar o total de faturação do hotel.
DELIMITER $$
CREATE FUNCTION consulta_Lucro(id_hotel INT)
        RETURNS FLOAT DETERMINISTIC
BEGIN
    RETURN (SELECT SUM(PRECO) AS Lucro FROM Quarto_has_Reserva as qr INNER JOIN Quarto as q
                                        ON qr.quarto_ID = q.id
                                        INNER JOIN Reserva as r
                                        ON r.ID = qr.Reserva_ID

    WHERE q.hotel_ID = id_hotel);
END $$
```

Figura 22 - Código SQL para RE13

5.4. Escolha, definição e caracterização de índices em SQL (alguns exemplos)

Tendo respondido às interrogações dos utilizadores, é necessário pensar na melhor forma do nosso sistema de gestão de hotéis ser mais rápido e mais eficiente. Uma maneira de tornar as nossas queries mais rápidas é através do uso de índices. Após analisarmos alguns dos problemas e interrogações que eram colocadas, percebemos que as datas das reservas eram bastante solicitadas.

Posto isto, decidimos que seria uma boa estratégia que as colunas que guardam a informação da data de início e de fim de uma dada reserva fossem acedidas de uma maneira mais célere. Para tal, foram-lhes atribuído um índice

```
CREATE INDEX idx_data_inicio ON Reserva(data_de_início);
CREATE INDEX idx_data_fim ON Reserva(data_de_fim);
```

Figura 23 - Índices em SQL da data de início e de fim de uma dada reserva

5.5. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Seguidamente, iremos apresentar uma estimativa do espaço em disco que a nossa base de dados irá ocupar. Para termos uma estimativa mais segura, e de maneira a precaver qualquer cenário possível, os atributos VARCHAR terão o tamanho máximo declarado.

Relação	Atributo	Data Type	Tamanho	Total
Hotel	ID	INT	4 Bytes	278 Bytes
	Site	VARCHAR(45)	45 Bytes	
	Nome	VARCHAR(45)	45 Bytes	
	Rua	VARCHAR(45)	45 Bytes	
	Freguesia	VARCHAR(45)	45 Bytes	
	Codigo_Postal	VARCHAR(45)	45 Bytes	
	Localidade	VARCHAR(45)	45 Bytes	
	Estrelas	INT	4 Bytes	
Funcionário	ID	INT	4 Bytes	358 Bytes
	Nome	VARCHAR(45)	45 Bytes	
	Codigo_Postal	VARCHAR(45)	45 Bytes	
	Freguesia	VARCHAR(45)	45 Bytes	
	Rua	VARCHAR(45)	45 Bytes	
	Localidade	VARCHAR(45)	45 Bytes	
	Email	VARCHAR(45)	45 Bytes	

	Telemovel	VARCHAR(10)	10 Bytes	
	CC	VARCHAR(15)	15 Bytes	
	Cargo	VARCHAR(45)	45 Bytes	
	Turno	VARCHAR(10)	10 Bytes	
	Hotel	INT	4 Bytes	

Tipo_Quarto	Designacao	VARCHAR(20)	20 Bytes	32 Bytes
	Preco	DECIMAL(5,2)	8 Bytes	
	Capacidade	INT	4 Bytes	
Quarto	ID	INT	4 Bytes	36 Bytes
	Hotel_ID	INT	4 Bytes	
	Tipo_de_Quarto	VARCHAR(20)	20 Bytes	
	Ocupado	CHAR(1)	8 Bytes	
Cliente	ID	INT	4 Bytes	257 Bytes
	CC	VARCHAR(15)	15 Bytes	
	NIF	INT	8 Bytes	
	Nome	VARCHAR(45)	45 Bytes	
	Morada	VARCHAR(100)	100 Bytes	
	Telemovel	VARCHAR(10)	10 Bytes	
	Email	VARCHAR(45)	45 Bytes	
	País	VARCHAR(30)	30 Bytes	
Reserva	Data_de_Inicio	DATE	8 Bytes	40 Bytes
	Data_de_Fim	DATE	8 Bytes	
	ID	INT	4 Bytes	
	Preco	DECIMAL(6,2)	8 Bytes	
	Data_de_Reserva	DATE	8 Bytes	
	Cliente	INT	4 Bytes	
Quarto_has_ Reserva	Quarto_ID	INT	4 Bytes	8 Bytes

	Reserva_ID	INT	4 Bytes	

Tabela 12 - Estimativa de ocupação da base de dados no disco

Apesar da entidade Funcionário ser aquela que ocupa mais espaço por ocorrência, não será a que ocupará mais espaço no sistema. A quantidade mais significativa será feita pelo Cliente, uma vez que, para uma utilização máxima de cada hotel, teremos um cliente para cada quarto, e uma reserva para cada um desses clientes.

Pela análise da tabela acima apresentada, teremos a ocorrência de 30 clientes, e, se consideramos que as reservas são de apenas 1 dia e que temos 30 novos clientes todos os dias,gerando 30 reservas novas todos os dias, teremos os seguintes cálculos.

3 Hóteis (3 * 278) + 9 Funcionários (9 * 358) + 30 Quartos (30 * 36) + 4 Tipos de Quarto (4 * 32) + 30 Clientes por dia (30 * 257 * 366) + 30 Reservas por dia (30 * 40 * 366) + 30 Quarto_has_Reserva por dia (30 * 8 * 366), prefazendo um total de **3354164 Bytes/Ano = 3.2 MBytes/Ano.**

5.6. Definição e caracterização das vistas de utilização em SQL (alguns exemplos)

O SQL permite a definição de views, que traduzem o resultado de uma ou várias operações relacionais sobre a nossa base de dados.

Assim sendo, definimos algumas views.

```
CREATE VIEW Reserva_Duracao AS
SELECT r.data_de_início AS 'Data de Início', r.data_de_fim AS 'Data de Fim', c.nome AS 'Nome'
FROM Reserva as r INNER JOIN Cliente as c
                  ON c.id = r.cliente
WHERE r.data_de_início >= now();


CREATE VIEW Reservas_Preco AS
SELECT r.Preco AS 'Preço da Reserva', c.nome AS 'Nome', c.ID 'ID Cliente'
FROM Reserva as r INNER JOIN Cliente as c
                      ON c.id = r.cliente;
```

Figura 24 - Código SQL da definição das views Reserva_Duracao e Reservas_Preco

```sql
CREATE VIEW Quartos_Ocupados_Cliente AS
SELECT c.nome AS 'Nome',  q.ID AS 'Número do Quarto', h.nome AS 'Hotel' ,r.id AS 'Reserva', r.preco AS 'Preço'
FROM Reserva AS r INNER JOIN Cliente c
                  ON c.id = r.cliente
                  INNER JOIN quarto_has_reserva qr
                  ON qr.Reserva_ID = r.ID
                  INNER JOIN Quarto q
                  ON q.ID = qr.Quarto_ID
                  INNER JOIN Hotel h
                  ON q.Hotel_ID = h.ID;
```

Figura 25 - Código SQL da view Quartos_Ocupados_Cliente

```sql
CREATE VIEW Reserva_TipoQuarto AS
SELECT r.ID AS 'Número da Reserva', tq.Designacao AS 'Tipo de Quarto', tq.Capacidade AS 'Capacidade do Quarto', c.nome AS 'Cliente'
FROM Reserva AS r INNER JOIN Cliente c
                  ON c.id = r.cliente
                  INNER JOIN quarto_has_reserva qr
                  ON qr.Reserva_ID = r.ID
                  INNER JOIN Quarto q
                  ON q.ID = qr.Quarto_ID
                  INNER JOIN Tipo_Quarto tq
                  ON q.Tipo_de_Quarto = tq.Designacao;
```

Figura 26 - Código SQL da view Reserva_TipoQuarto

5.7. Revisão do sistema implementado

Ficando assim a última etapa da construção do nosso sistema de base de dados concluída, é possível apresentar este produto ao seu comprador. Explicando cada funcionalidade e o propósito das mesmas, o produto pode ser visto como competente e bastante útil, contudo tem sempre margem para evoluir e para definir novas funcionalidades, não sendo o objetivo da construção deste sistema de base de dados um produto final e inerte, mas sim algo que possa ser melhorado e atualizado a qualquer momento.

6. Conclusões e Trabalho Futuro

O desenvolvimento da base de dados que servirá de suporte à implementação de um sistema de gestão de hotelaria foi feito de forma faseada, o que nos permitiu ter mais tempo para analisar e construir uma solução mais robusta. Contudo, com o decorrer do tempo e conforme fomos desenvolvendo o projeto, fomo-nos apercebendo de alguns erros cometidos durante o processo de desenvolvimento. Nunca se trataram de erros de maior dimensão, tendo sido bastante fácil “voltar atrás” no projeto e corrigir tudo até ao ponto em que nos

encontrávamos. Foi então, necessário em algumas ocasiões retroceder a fases anteriores. Isto seria corrigido se tivéssemos encarado cada uma das fases de implementação com um pouco mais de cuidado.

Olhando para o trabalho desenvolvido e refletindo um pouco sobre as etapas do seu desenvolvimento, podemos afirmar com toda a certeza que os grandes alicerces da solução encontrada são, sem dúvida, as fases de definição do problema e de desenvolvimento do modelo conceptual. Isto é, quanto mais aprofundado e analisado for o estudo do comportamento da empresa, melhor será a qualidade dos requisitos levantados, o que se refletirá em respostas mais assertivas às necessidades e interrogações do utilizador. Por outro lado, quanto mais detalhado for o conhecimento em relação ao que se pretende modular na base de dados e ao modo de operação da empresa, mais fiel será a reprodução dos dados no sistema. É ainda de referir que do modelo conceptual se deriva o modelo lógico, que, posteriormente, dá origem ao modelo físico. O modelo conceptual assume então um papel de base para todo o trabalho que se segue, o que significa que, um modelo conceptual incorreto e descuidado resultaria num sistema obsoleto.

A fase que nos tomou mais tempo para concluir e mais dúvidas nos levantou foi exatamente a modelação conceptual. Isto porque desenvolvemos vários modelos conceptuais diferentes que estavam incorretos devido a uma fraca definição do problema, dos requisitos e objetivos do modelo. Com sucessivos refinamentos dos objetivos que se esperavam alcançar com a implementação do sistema de base de dados, o desenho do modelo conceptual surgiu naturalmente.

No futuro, podíamos incluir na base de dados informação relativa aos demais espaços de cada hotel (restaurantes, piscinas, ginásios, salas de reuniões, etc.). Desta forma, a base de dados possibilitaria também ao administrador analisar o uso destes serviços por parte dos clientes, e aos clientes permitiria uma melhor análise do hotel que escolhem para fazer a sua reserva, uma vez que têm acesso a um catálogo de serviços mais abrangente.

Por fim, concluímos que os objetivos foram cumpridos, sendo o resultado deste processo uma base de dados totalmente funcional e capaz de cumprir com os objetivos e necessidades que levaram à necessidade e/ou vontade por trás da sua implementação.