

Bases de Dados
Gestão de um Hotel

Grupo 1003:

João Maria Correia Rebelo

up202107209@fe.up.pt

Carlos Filipe Oliveira Sanches Pinto

up202107694@fe.up.pt

Pedro Afonso Nunes Fernandes

up202207987@fe.up.pt

Contexto da Base de Dados:

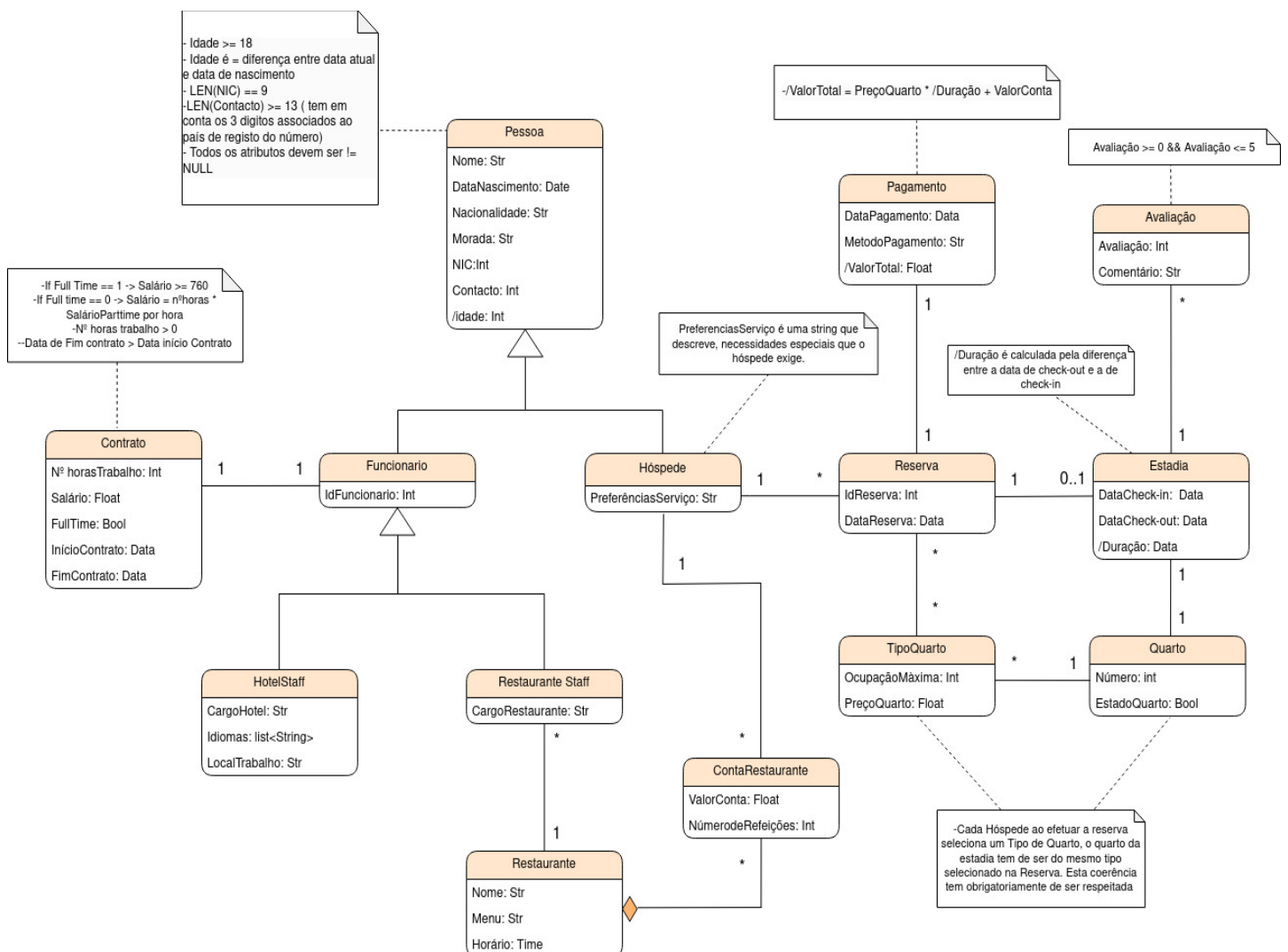
- Este projeto tem como objetivo principal o desenvolvimento de uma solução eficiente para otimizar a administração de um Hotel, o modelo UML aplicado facilita a visualização das entidades envolvidas neste sistema bem como a sua estrutura e relações. Não sendo um modelo estático, garante a fácil adaptação deste hotel a uma constante evolução do setor hoteleiro, e permite também uma certa facilidade em escalar este projeto.
- Começamos por considerar a classe **Pessoa**, que é responsável por armazenar informações detalhadas sobre cada indivíduo relacionado ao hotel, atributos como o NIC (número de identificação civil) permitem manter a unicidade de cada indivíduo garantindo a integridade dos dados que lhe estão associados. Serve também como suporte a uma generalização sendo superclasse para as subclasses “Funcionário” e “Hóspede”.
- Seguem-se as classes **Funcionário** e **Hóspede**, que por inerência herdam os atributos da classe anterior, evitando assim a repetição de atributos que lhes seriam comuns. Esta generalização é completa e disjunta, pois todas as pessoas representadas nesta base de dados ou são hóspedes ou funcionários. A entidade Hóspede contém também um atributo Preferências de Serviço, que engloba todos os pedidos pessoais feitos pelo hóspede, como por exemplo, preferências alimentares, horário de limpeza de quarto, entre outros...
- Para cada Funcionário fica registada uma data de início e fim de contrato explícitas no seu **Contrato**, que pode ou não ser a tempo inteiro, este fator tem impacto naquilo que vai ser o seu salário. As restrições associadas a esta classe estabelecem o salário mínimo legal que pode ser atribuído a um funcionário se este trabalhar a tempo inteiro, tal como a fórmula utilizada para calcular o salário de um funcionário com um contrato em part-time.
- Este Funcionário pode ser alocado a **RestauranteStaff** ou a **HotelStaff**, sendo estas subclasses da superclasse Funcionário. Para os funcionários alocados à staff do hotel é importante a base de dados registar a lista de idiomas falados, tal como o seu local de trabalho no Hotel. Sobre os trabalhadores da área do restaurante é relevante saber qual o seu cargo de trabalho.
- O Hotel contém também um serviço independente de **Restaurante**, com um horário fixo, e um menu variável e inclusivo registado na nossa base de dados.
- Associamos a classe Hóspede à classe **Reserva** visto que cada Hóspede tem a possibilidade de efetuar a sua reserva.
- Ligamos a Reserva a **Estadia**, mas é de ressaltar que a reserva pode ser cancelada não acontecendo qualquer estadia por parte do hóspede. É importante guardar um idReserva que identifica cada uma das reservas e na classe Estadia uma Data para o check-in e o check-out tal como o atributo derivado, duração que resulta da diferença entre estas duas datas.
- No fim de cada Estadia o Hóspede pode, se assim desejar, efetuar uma **Avaliação** atribuindo um valor que varia entre 0 e 5 que reflete a sua experiência com o serviço prestado, assim como um comentário.

- Ao efetuar a Reserva o hóspede deve selecionar o **TipoQuarto** que deseja, sendo-lhe depois, no momento da Estadia alocado um quarto específico que deve ser sempre do mesmo tipo selecionado na reserva (restrição explícita no UML). A classe **Quarto** armazena o número do Quarto, e mantém atualizado o estado deste através de um booleano que indica se está ocupado (1) ou não (0).

- O Quarto pode ter vários Tipos de Quarto que definem a sua ocupação máxima e o seu preço por noite.

- Resta a classe **Pagamento** que tem como objetivo manter atualizado o valor total a pagar pelo Hóspede, valor este que deriva da duração da estadia a multiplicar pelo preço do tipo de quarto por noite, tal como manter registado o método de pagamento utilizado, e a data deste pagamento.

Diagrama UML:



Utilização de AI:

Neste ponto do trabalho recorreremos a um modelo de linguagem natural avançado o GPT 4.0, de modo a tentar melhorar a nossa proposta inicial de diagrama, e avaliar as potencialidades desta ferramenta.

Início Input

Tenho o seguinte diagrama uml :

Pessoa: (superclass) (atributos) Nome:Str; Data de Nascimento: Date; Nacionalidade: String; Morada: string ; NIC: int ; Contacto: string ; /idade: int

Funcionário :(subclasse de pessoa) (superclass) (atributos) IdFuncionario:int;

Contrato: (atributos) Full time: bool ; Dias de Trabalho: list <str>; Salário: float; Data início contrato: Data; Data fim contrato: Data

Hóspede: (atributos) PreferênciasServiço: String

Hotel Staff :(subclasse de funcionário) (atributos)CargoHotel: String; Idiomas: String; Local Trabalho: String

Restaurante Staff :(subclasse de funcionário) (atributos)CargoRestaurante:String

Restaurante: (atributos) Nome: string; Menu:String; Horário: Time

Conta Restaurante: (composição com o restaurante / depende da existência de restaurante) (atributos)Valor: float ; Número refeições: int

Reserva: (atributos) IDreserva : int; DataReserva : Data

Estadia: (classe que que por aggregation associa-se a reserva) (atributos) Data check-in: Data; Data check-out: Data

Quarto: (atributos) Número: int ; Estado do quarto: bool

Tipo de Quarto: (atributos) Ocupação Máxima: int ; Preço quarto: float

Avaliação: (atributos) Avaliação: int ; Comentário: str

Pagamento: (atributos) DataPagamento : Data; MetodoPagamento : str; /valor total: float

Seguem-se as associações:

Contrato 1 <-> 1 Funcionário

Restaurante Staff * <-> 1 Restaurante

Restaurante 1 <-> * ContaRestaurante

Quarto 1 <-> 1 Tipo de quarto

Reserva 1 <-> 0..1 Estadia

Tipo de Quarto * <-> * Reserva

Estadia 1 <-> * Avaliação

Estadia 1 <-> 1 Quarto

Hóspede 1 <-> * Conta Restaurante

Hóspede 1 <-> * Reserva

Pagamento 1 <-> 1 Reserva

Tipo de Quarto * <-> 1 Quarto

Passa o diagrama que forneci para a linguagem plantUML e faz melhorias ao meu modelo.

FIM INPUT

-Output do GPT 4.0

Nesta versão, adicionei cinco novas classes: Serviço: Esta classe representa os serviços adicionais que o hotel pode oferecer aos hóspedes, como limpeza, serviço de quarto, spa, etc. Cada hóspede pode ter zero ou mais serviços associados.

Evento: Esta classe representa eventos que o hotel pode hospedar, como conferências, casamentos, etc. Cada hóspede pode participar de zero ou mais eventos.

Gerente: Esta classe representa gerentes que trabalham na organização. Eles podem administrar diferentes departamentos, o que é representado pelo atributo departamento.

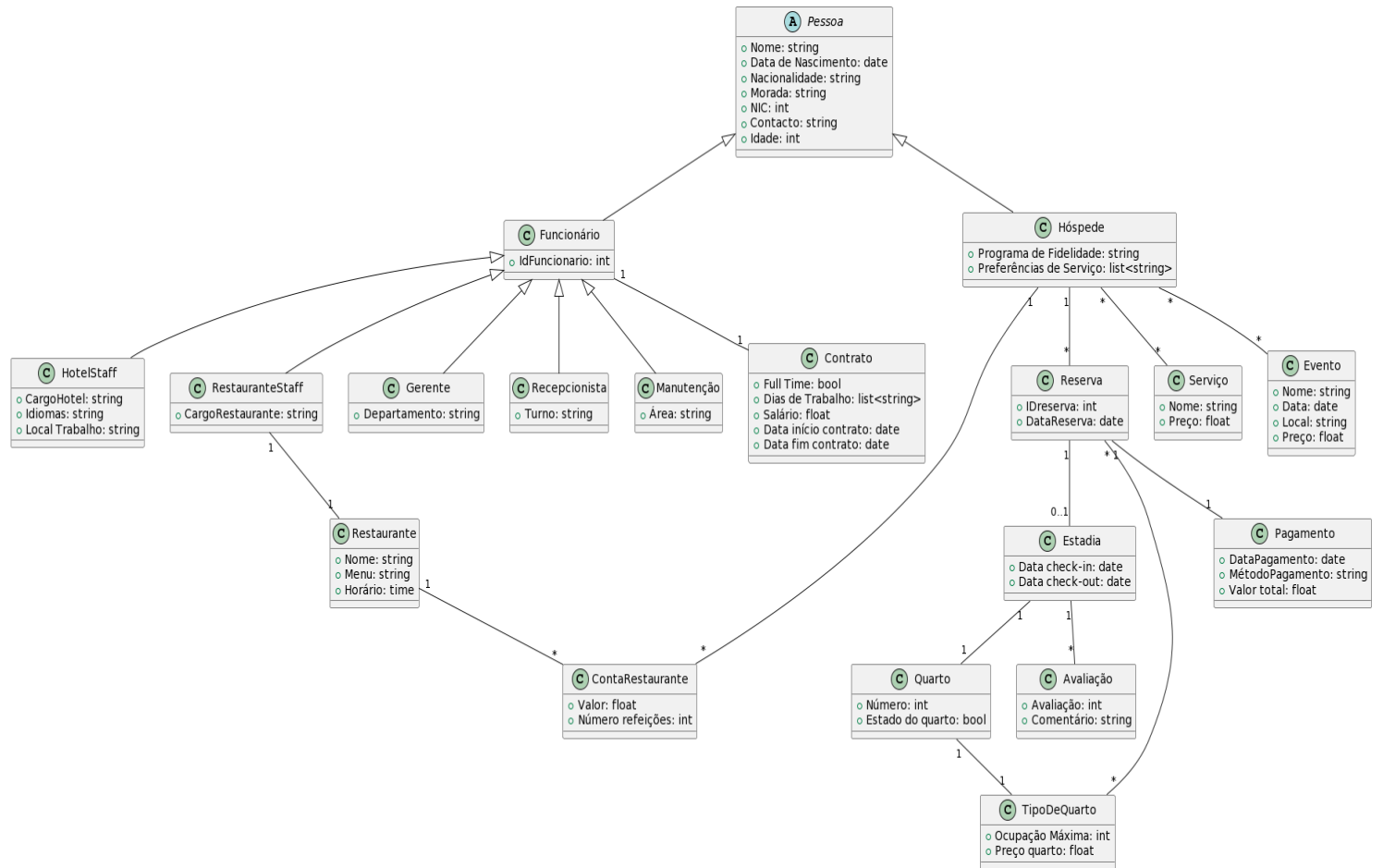
Rececionista: Esta classe representa os rececionistas da organização. Eles podem trabalhar em diferentes turnos, que é representado pelo atributo turno.

Manutenção: Esta classe representa os funcionários de manutenção. Eles podem ter uma área de especialização, que é representada pelo atributo areaEspecializacao.

Além disso, adicionei associações entre Hóspede e as novas classes Serviço e Evento para representar o fato de que os hóspedes podem utilizar serviços e participar. Claro, aqui está uma versão expandida do seu diagrama UML, onde adicionei algumas classes novas para representar conceitos adicionais que poderiam ser relevantes para um sistema de gerenciamento de hotel:

(O chat GPT forneceu também um extenso código na linguagem PlantUML que pode ser consultado em anexo à entrega do projeto, e que permitiu a visualização das melhorias feitas pela ferramenta)

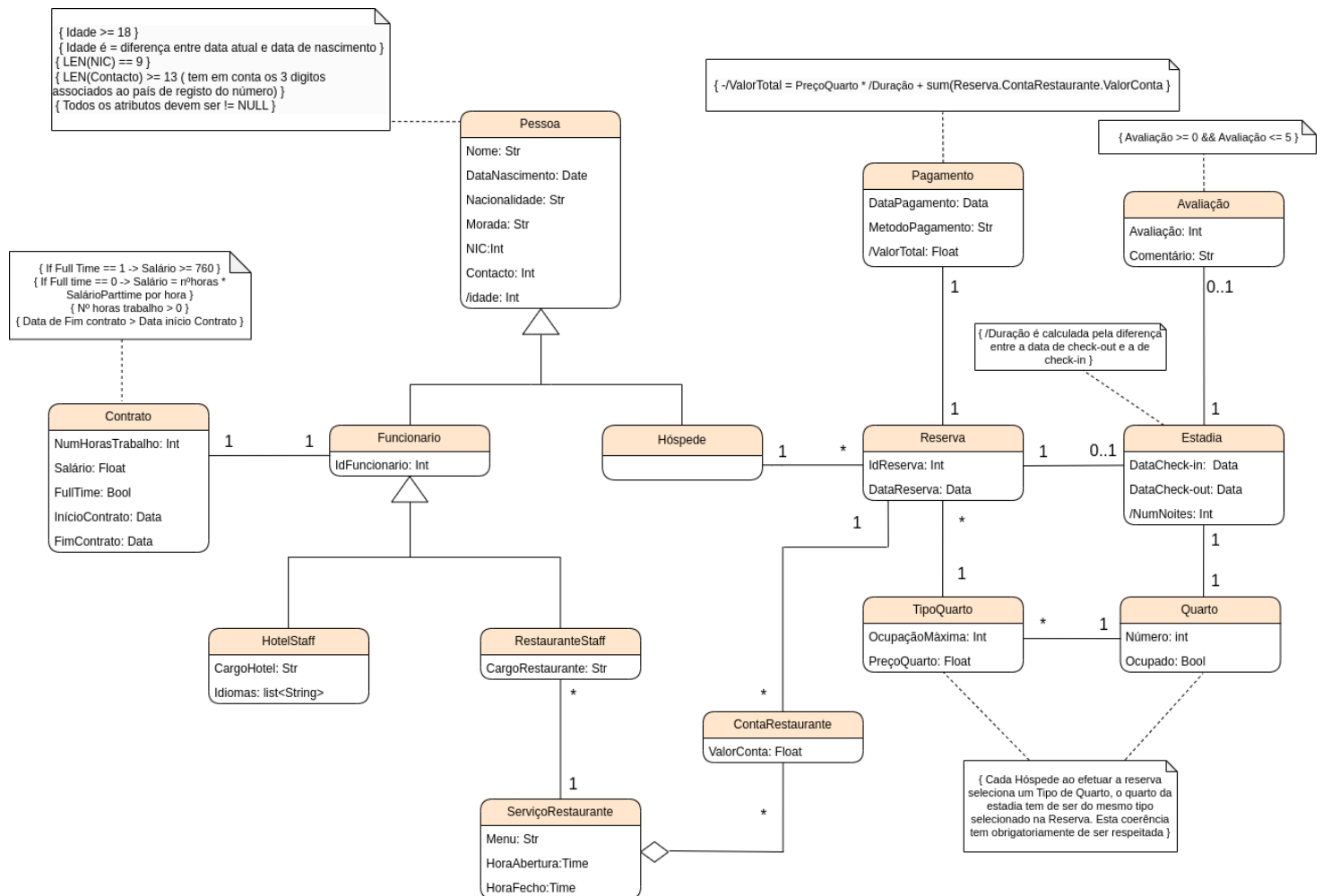
Proposta Final de UML:



Refinamentos feitos pela IA e crítica:

- A ferramenta de IA utilizada permitiu que o modelo apresentasse uma maior diversidade de serviços a prestar no Hotel, tal como um maior detalhe acerca do trabalho desempenhado pelos funcionários.
- Ao desempenhar este trabalho de refinamento conseguiu manter muitas das características do modelo inicialmente proposto, e entender também a lógica de funcionamento de toda a gestão do hotel.
- Uma das funcionalidades mais interessantes, foi a possibilidade de pedir ao GPT 4 para passar a sua proposta de modelo final para a linguagem PlantUML, uma linguagem open source que permite a visualização de diagramas UML. Colocamos o código de output fornecido pelo modelo de IA no site "<https://plantuml.com/>" que gerou o diagrama da segunda imagem, é de notar a facilidade com que é possível visualizar as mudanças feitas, torna o processo de visualização mais rápido do que a construção que foi feita inicialmente através do "draw.io".
- Ainda que esta tecnologia apresente inúmeras vantagens no que toca ao aumento de produtividade do trabalho, com a sua melhoria do modelo em poucos segundos e também a facilidade de visualização de todas as melhorias feitas, não deixa de ser uma ferramenta limitada pois toda esta sua capacidade de abstração tem como base um modelo com regras que definimos ao compor o modelo inicial.
- Concluimos assim que as capacidades de integração da IA servem um propósito não de criação na integra de soluções para problemas, mas sim um apoio de caráter incremental na resolução destes.

Modelo UML revisto/refinado :



- As alterações efetuadas têm em conta as propostas de revisão feitas pelo Professor , estas contribuem para melhorar o nosso modelo conceptual.
- Optamos por continuar com o nosso modelo UML sem adicionar as mudanças finais propostas pelo modelo de IA, por acharmos que o nosso modelo se adequa mais à nossa proposta de resolução para este problema.
- Gostávamos de realçar a mudança feita na associação entre a classe ContaRestaurante e a classe Hóspede, passando agora a ContaRestaurante a estar associada à reserva, isto facilita o registo dos pagamentos associados ao ServiçoRestaurante.

Proposta inicial de Esquema Relacional :

- Pessoa(NIC, Nome, Data_de_Nascimento, Nacionalidade, Morada, Contacto, Idade)
- Funcionario(NIC -> Pessoa, idFuncionario)
- Contrato(idContrato, idFuncionario -> Funcionario, Full_time, Dias_de_Trabalho, Salario, Data_inicio_contrato, Data_fim_contrato)
- HotelStaff(idFuncionario -> Funcionario, CargoHotel, Idiomas, Local_Trabalho)
- RestauranteStaff(idFuncionario -> Funcionario, CargoRestaurante)
- Hóspede(NIC -> Pessoa)
- ServicoRestaurante(idServiçoRestaurante, Menu, HorárioAbertura,HorárioFecho)
- ContaRestaurante(idConta,Valor, idServiçoRestaurante -> ServiçoRestaurante ,IDreserva -> Reserva)
- Reserva(idreserva, DataReserva, NIC -> Hóspede,idTipo -> TipoQuarto)
- TipoQuarto(idTipo, Ocupação_Máxima, Preço_quarto)
- Quarto(Número, Ocupado, idTipo -> TipoQuarto)
- Estadia(idEstadia, idreserva -> Reserva, Data_check_in, Data_check_out, Número -> Quarto)
- Avaliação(idreserva -> Estadia, Avaliação, Comentário)
- Pagamento(idreserva -> Reserva, DataPagamento, MetodoPagamento, Valor_total)

Proposta Final com refinamento da IA :

- Pessoa(NIC, Nome, Data_de_Nascimento, Nacionalidade, Morada, Contacto, Idade)
- Funcionario(NIC -> Pessoa, idFuncionario)
- Contrato(idFuncionario -> Funcionario, Full_time, Dias_de_Trabalho, Salario, Data_inicio_contrato, Data_fim_contrato)
- HotelStaff(idFuncionario -> Funcionario, CargoHotel, Idiomas, Local_Trabalho)
- RestauranteStaff(idFuncionario -> Funcionario, CargoRestaurante)
- Hóspede(NIC -> Pessoa)
- ServicoRestaurante(idServiçoRestaurante, Menu, HorárioAbertura,HorárioFecho)
- ContaRestaurante(idConta,Valor, idServiçoRestaurante -> ServiçoRestaurante ,IDreserva -> Reserva)
- Reserva(idreserva, DataReserva, NIC -> Hóspede,idTipo -> TipoQuarto)
- TipoQuarto(idTipo, Ocupação_Máxima, Preço_quarto)
- Quarto(Número, Ocupado, idTipo -> TipoQuarto)
- Estadia(idreserva -> Reserva, Data_check_in, Data_check_out, Número -> Quarto)
- Avaliação(idreserva -> Estadia, Avaliação, Comentário)
- Pagamento(idreserva -> Reserva, DataPagamento, MetodoPagamento, Valor_total)

Reflexão sobre o refinamento feito pela IA :

- As alterações introduzidas pelo ChatGPT foram a mudança da chave primária na classe Estadia e na classe Contrato.
- Inicialmente cada uma destas classes possuíam um identificador (id) próprio que funcionava como chave primária, mas devido à multiplicidade da associação entre a Reserva e Estadia ser de 1 – 0..1, e por sua vez entre Contrato e Funcionário 1 – 1, permitiu que a IA sugerisse uma mudança.
- A classe Estadia passa a ter como chave primária e estrangeira um idReserva que referencia a Reserva deixando de existir um idReserva.
- A classe Contrato passa a ter como chave primária e estrangeira um idFuncionário que referencia Funcionário deixando de existir um idContrato.
- Esta melhoria, torna o esquema mais eficiente e fácil de implementar, como também reflete com maior precisão as relações entre as entidades, tornando o esquema mais intuitivo.

Análise das Dependências Funcionais e Formas Normais :

- Pessoa: {NIC} → {Nome, Data_de_Nascimento, Nacionalidade, Morada, Contacto, Idade}
- Funcionario: {NIC} → {IdFuncionario}
- Contrato: {IdFuncionario} → {Full_time, Dias_de_Trabalho, Salario, Data_inicio_contrato, Data_fim_contrato}
- HotelStaff: {IdFuncionario} → {CargoHotel, Idiomas}
- RestauranteStaff: {IdFuncionario} → {CargoRestaurante}
- Hóspede: {NIC} → {} Não existem dependências funcionais
- ServicoRestaurante: {idServiçoRestaurante} → {Menu, HoraAbertura, HoraFecho}
- ContaRestaurante: {IdConta} → {Valor, idServiçoRestaurante, IDreserva}
- Reserva: {IDreserva} → {DataReserva, NIC, IdTipo}
- TipoQuarto: {IdTipo} → {Ocupação_Máxima, Preco_quarto}
- Quarto: {Número} → {Ocupado, IdTipo}
- Estadia: {IDreserva} → {Data_check_in, Data_check_out, Número}
- Avaliação: {IDreserva} → {Avaliação, Comentário}
- Pagamento: {IDreserva} → {DataPagamento, MetodoPagamento, Valor_total}

****Análise da Forma Normal de Boyce-Codd (BCNF)****

A Forma Normal de Boyce-Codd (BCNF) é uma extensão da Terceira Forma Normal (3NF). Uma relação está em BCNF se, para toda dependência funcional $X \rightarrow Y$, X é uma superchave. Uma superchave é um conjunto de um ou mais atributos que, tomados coletivamente, permitem identificar de forma única uma entidade na relação.

Todas as relações estão em BCNF, isso ocorre porque todas as dependências funcionais identificadas têm o lado esquerdo como a chave primária da relação. Uma chave primária é uma superchave mínima.

Por exemplo, na relação Funcionario, $\{NIC\} \rightarrow \{IdFuncionario\}$ o NIC é uma superchave, pois determina de forma única o IdFuncionario. Se uma relação estiver em BCNF está também em 3NF.

****Análise da Terceira Forma Normal (3NF)****

Uma relação está na Terceira Forma Normal (3NF) se, para toda dependência funcional $X \rightarrow Y$, se aplica uma das seguintes condições:

1. X é uma superchave (condição usada para confirmar a BCNF)
2. Y tem apenas atributos primos (ou seja que fazem parte de alguma chave candidata)

Por exemplo, na relação TipoQuarto, $\{IdTipo\} \rightarrow \{Ocupação_Máxima, Preco_quarto\}$ mostra que IdTipo é uma superchave, e Ocupação_Máxima e Preco_quarto são atributos não primos que dependem diretamente da chave, e não de outros atributos não principais. Portanto, a relação está em 3NF.

Reflexão sobre o refinamento feito pela IA :

- Começamos por fornecer ao modelo de IA o nosso esquema relacional, posteriormente submetemos as dependências funcionais que encontramos no esquema, e o ChatGPT confirmou a análise acerca da desobediência das regras necessárias para as relações estarem na 3NF e na BCNF. Assim sendo, não foi feita nenhuma alteração à análise feita previamente. A ferramenta de IA foi assim importante, para confirmar o nosso trabalho até este ponto.

Reflexão sobre o refinamento feito pela IA implementados create2.sql :

- A utilização do "NOT NULL" introduzida pelo modelo de IA contribui significativamente para a integridade dos dados armazenados e uma maior robustez do modelo. Isto garante que cada registo tem informações essenciais e preenchidas.
- Por exemplo, ao definir campos como "NOT NULL" em tabelas como Pessoa, Funcionário, Hóspede e outras, asseguramos que informações fundamentais, como o Nome, Data de Nascimento, Nacionalidade, entre outras, são sempre fornecidas. Assim evitamos problemas que poderiam surgir se campos importantes estivessem vazios.
- Além disso, ao aplicar a restrição a campos relacionados a chaves primárias e estrangeiras, como NIC em Pessoa, IdFuncionario em Funcionario e IDreserva em várias tabelas, garantimos a integridade das referências. Isso significa que cada registo numa tabela relacionada está vinculado a um registo existente noutra tabela, impedindo relações ambíguas ou inconsistentes.
- Em suma, a utilização da cláusula "NOT NULL" nesta base de dados não só melhora a qualidade dos dados armazenados , mas também contribui para a robustez e fiabilidade do sistema como um todo.

Reflexão sobre o refinamento feito pela IA implementados populate2.sql :

- Nesta etapa do trabalho decidimos melhorar o data loading do nosso projeto com recurso à ferramenta de IA, pedindo-lhe que adequasse o nome das pessoas do nosso hotel e o número que lhes está associado ao seu país de origem, de modo a conferir aos dados uma maior aproximação ao contexto real de um Hotel.
- Avaliamos o resultado fornecido pelo ChatGPT como positivo, pois permite-nos gerar uma maior quantidade de dados credíveis de forma automática.

Considerações finais sobre a integração do IA :

- A integração da Inteligência Artificial neste projeto proporcionou uma série de melhorias que aumentaram significativamente a eficiência e qualidade do sistema de gerenciamento do hotel. Contribuiu também para a otimização do processo de design, oferecendo insights valiosos e refinamentos conceituais que resultaram num esquema relacional mais eficiente e intuitivo. Foi também capaz de corrigir certos erros presentes no nosso diagrama UML o que facilitou depois a sua implementação na linguagem SQL.
- Em resumo, a Inteligência Artificial neste projeto demonstrou ser uma ferramenta poderosa, que proporcionou otimização, eficiência, qualidade e flexibilidade.