

Bases de Dados

*Aluguer de Carros*

*LEIC012 – Grupo 1205*

*(11 de outubro de 2023)*

Leonardo Garcia  **up202200041**@fe.up.pt

Marcel Medeiros **up202200042**@fe.up.pt

Gonçalo Sousa **up202207320**@fe.up.pt

Índice

Introdução.............................….…………………………………………………………2

Contexto da Base de Dados..……….……………………….………………………….2

Diagramas de Classe UML – Versão Original...…….………………………………5

Descrição do Processo de Integração IA………………………………….………..…7

Análise Crítica dos Resultados e Considerações Finais......................­­......................9

Introdução

No âmbito da Unidade Curricular de 2º Ano Bases de Dados, iremos, com este projeto, descrever e implementar a base de dados de uma empresa fictícia de aluguer de carros.

Desta forma, o grupo recorreu a técnicas de representação UML com o intuito de detalhar o modelo conceptual.

 Contexto da Base de Dados

Nosso cliente é uma empresa que quer construir a base de dados para um negócio de aluguer de carros. Com esta base de dados, a empresa pretende armazenar não só dados relativos aos clientes, como também a sua frota de veículos e o ato de alugar o carro.

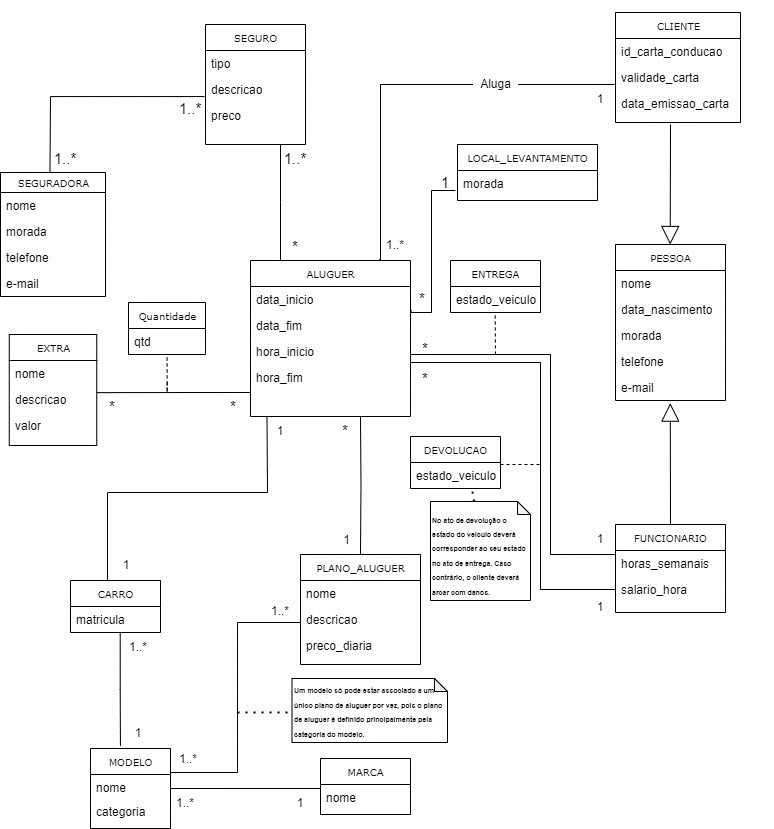
É importante frisar que esta base de dados não descreve uma aplicação de aluguer de carros, ou seja, o nosso foco com este projeto não é definir uma interface que possibilite, através dela, um cliente efetuar o aluguer. É a recolha dos supracitados dados para armazenamento que nos interessa.

Primeiramente, é importante definir a parte humana da base de dados, dessa maneira, a classe **Pessoa** agrega os atributos generalizados para **Cliente e Funcionário.** Cada Cliente é capaz de realizar um **Aluguer** uma ou mais vezes, portanto, a base de dados armazena informações relativas ao inicío e fim de cada aluguer, de forma que o controle de disponibilidade de veiculos seja garantido.

A classe **Aluguer** está diretamente associada com o **Carro** e o **Plano de Aluguer**, esse caracteriza-se pelo **Modelo** do carro que será alugado, uma vez que dependendo da categoria do veículo, o preço do plano irá variar. Outro aspecto necessário nesta classe são seus **Extras**, um extra é aquilo que não é necessário um cliente ter em sua viagem mas que pode optar por adicioná-lo, eles adicionam valor a experiência e o preço final do Aluguer também será ditado por seu tipo e quantidade. Além disso, é importante destacar que um Aluguer deve ter pelo menos um **Seguro** associado, que por sua vez associa-se com pelo menos uma **Seguradora**. Os seguros variam de preço e seus tipos cobrem diferentes vantagens, mas o cliente não poderá realizar o aluguer sem o mesmo.

Seguidamente, achamos importante abordar o aspecto contextual da **Entrega e Devolução** dos carros. Um **Funcionário** da empresa realizará a Entrega de seu carro para o cliente em um dos seus **Locais de Levantamento**, além disso, ele irá armazenar o estado do veículo, para quando a Devolução ocorrer, outro Funcionário inspecionar o mesmo e checar se foi entregue em condições iguais

Diagrama de Classe UML



Descrição do Processo de Integração IA – Modelo Conceitual

A integração IA surge como uma ferramenta para analisarmos as melhorias que poderíamos ter feito no nosso projeto e, de acordo com nossa lógica, julgar se a implementação do ponto de vista da inteligência artificial faria sentido. Para isso usamos a ferramenta **ChatUML** da seguinte maneira:

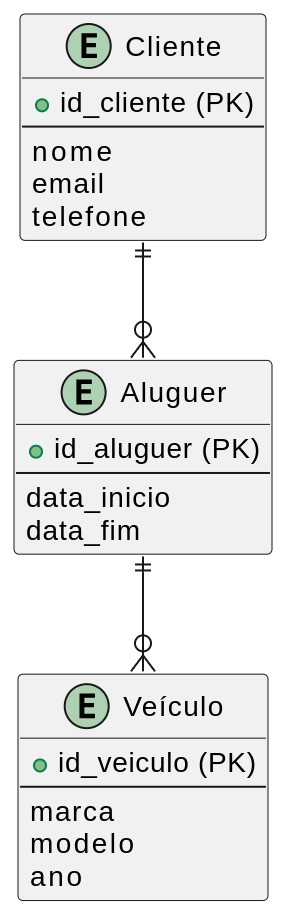
**1º Prompt -**

No âmbito da Unidade Curricular de 2º Ano Bases de Dados, iremos, com este projeto, descrever e implementar a base de dados de uma empresa fictícia de aluguer de carros.

Desta forma, o grupo recorreu a técnicas de representação UML com o intuito de detalhar o modelo conceitual. Como nosso cliente, temos uma empresa que quer construir a base de dados para um negócio de aluguer de carros. Com esta base de dados, a empresa pretende armazenar não só dados relativos aos clientes como também à sua frota de veículos e o ato de alugar o carro.

É importante frisar que esta base de dados não descreve uma aplicação de aluguer de carros, ou seja, o nosso foco com este projeto não é definir uma interface que possibilite, através dela, um cliente efetuar o aluguer. É a recolha dos supracitados dados para armazenamento que nos interessa.

O output que nos foi dado inicialmente foi este:

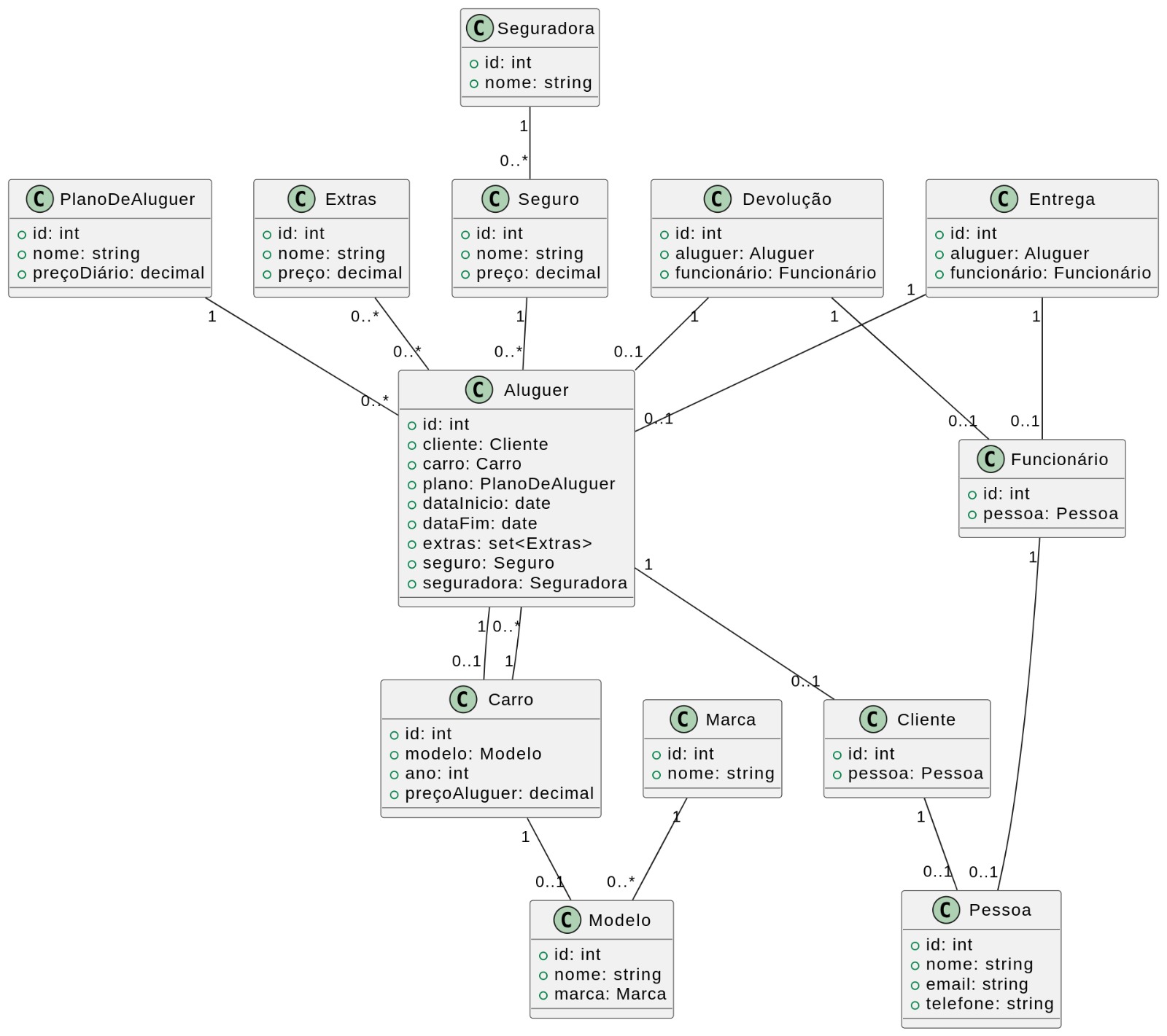


Assim, sentimos necessidade de ser mais expressivos com a IA:

**2º Prompt -**

Ok, mas ficou muito básico para o que temos em mente no nosso projeto. Queremos trabalhar com as seguintes classes: Pessoa, Cliente, Funcionário, Marca, Modelo, Carro, Plano de Aluguer, Extras, Seguro, Seguradora, Aluguer, Entrega e Devolução. Entrega e Devolução devem conter informações sobre o funcionário que realizou essas duas operações.

Então o ChatUML deu-nos um diagrama mais incorpado e com mais classes:



Finalmente, pedimos para a IA implementar o que ela achasse que faria sentido ser

adicionado ao nosso projeto, porém o resultado foi o mesmo do anterior...

Discussão dos Resultados – Modelo Conceitual

Dada a despretensiosidade dos resultados obtidos com o primeiro *input*, achamos desnecessário fazer uma análise sobre o *output*. Embora relativamente bem detalhada, a descrição que fizemos não foi suficiente para obtermos um diagrama com potencial.

O mesmo não acontece com o segundo *input*. Após fazermos uma referência exata das classes que queríamos adicionar, a IA conseguiu fazer um diagrama bastante similar ao nosso. Inclusive adicionou os *Data Types* de cada atributo, algo que não fizemos por agora.

Embora a semelhança com o nosso diagrama, por consequência da objetividade desta técnica de representação, que não dá grande espaço para “inventar”, achamo-lo um pouco confuso ao início – talvez por não ser da nossa autoria. Outra nota importante de realçar é a referência a relações entre classes como um atributo da classe de partida. Por exemplo, na classe Carro há um atributo modelo do tipo Modelo. Isto dificulta a leitura e interpretação do diagrama, acabando mesmo por ser um pouco redundante, uma vez que essa relação já está representada e é facilmente identificável.

Algo que também nos chamou à atenção foi a forma de representar algumas relações. Associações que seriam do tipo composição estão representadas em inconformidade com o modelo UML. Embora para efeitos práticos a composição esteja subentendida na carnalidade, achamos que seria mais correto recorrer à forma convencionada, tal como nós fizemos.

Tendo em conta também que Cliente e Funcionário são especializações da generalização Pessoa, a relação entre elas deveria estar representada com setas que apontam para a generalização de modo a respeitar o modelo UML.

A carnalidade das relações nem sempre foi a correta. Passo a enunciar algumas relações que deveriam ser retificadas – todas as ‘0..\*’ podem ser substituídas por ‘\*’ dado que ‘\*’ já inclui a possibilidade do 0 ocorrer; da classe Aluguer para a classe Carro, se o aluguer existe então alugar-se-á sempre um carro, ou seja, devemos trocar ‘0..1’ por ‘1’.

A inserção de um atributo ID em todas as classes foi algo que achamos interessante e de certa forma pertinente. De facto, usar um ID dá à classe um atributo único que pode ser usado como chave primária e permite a sua identificação em qualquer circunstância.

Dito isto, podemos concluir que mesmo tendo feito alguns acrescentos relevantes, **o nosso modelo conceitual não sofreu alterações e continua o mesmo.**

Modelo Relacional

PESSOA(id, nome, data\_nascimento, morada, telefone, e-mail)

CLIENTE(id → PESSOA, id\_carta\_conducao, validade carta, data\_emissao\_carta)

FUNCIONARIO(id → PESSOA, horas\_semanais, salario\_hora)

CARRO(id, matricula, modelo → MODELO)

MODELO(nome, categoria, marca → MARCA)

MARCA(nome)

SEGURADORA(nome, morada, telefone, email)

SEGURO(id, tipo, descricao, preco)

ALUGUER\_SEGURO(id\_aluguer → ALUGUER, id\_seguro → SEGURO)

SEGURADORA\_SEGURO(seguradora → SEGURADORA, seguro → SEGURO)

PLANO\_ALUGUER(id,nome,descricao,preço\_diaria)

PLANO DE ALUGUER\_MODELO(aid → PLANo\_ALUGUER,nome → MODELO)

EXTRA(id,nome,descricao,valor)

ALUGUER(id,cliente\_id→CLIENTE,morada→LOCAL\_LEVANTAMENTO,plano\_id → PLANO\_ALUGUER,*data\_inicio,data\_fim,hora\_inicio,hora\_fim)*

*QUANTIDADE(aluguer\_id → ALUGUER, extra\_id → EXTRA, qtd)*

*ENTREGA(aluguer\_id → ALUGUER, funcionário\_id → FUNCIONÁRIO, estado\_veiculo)*

*DEVOLUÇÃO(aluguer\_id → ALUGUER, funcionário\_id → FUNCIONARIO, estado\_veiculo)*

*LOCAL\_LEVANTAMENTO(morada)*

Descrição do Processo de Integração IA – Modelo Relacional

Tendo em vista que a ferramenta ChatUML não nos trouxe resultados proveitosos no processo de integração prévio, desta vez usamos o ChatGPT 3.5 para obter melhores resultados.

-----------------------------------

Discussão dos Resultados – Modelo Relacional

---------------------------------------------

Dependências Funcionais e Análise das Formas Normais

Ao analisar nosso modelo relacional, identificamos as suas dependências funcionais:

**PESSOA(id, nome, data\_nascimento, morada, telefone, e-mail)**

Chaves: {id}, {telefone}, {e-mail}

{id} -> {nome, data\_nascimento, morada, telefone, e-mail}

{e-mail} -> {id, nome, data\_nascimento, morada, telefone}

**CLIENTE(id → PESSOA, id\_carta\_conducao, validade\_carta, data\_emissao\_carta)**

Chaves: {id}, {id\_carta\_conducao}

{id} -> {id\_carta\_conducao, validade\_carta, data\_emissao\_carta}

{id\_carta\_conducao} -> {id, validade\_carta, data\_emissao\_carta}

**FUNCIONARIO(id → PESSOA, horas\_semanais, salario\_hora)**

Chaves: {id}

{id} -> {horas\_semanais,salario\_hora}

**CARRO(id, matricula, modelo → MODELO)**

Chaves: {id}, {matricula}

{id} -> {matricula, modelo}

{matricula} -> {id, modelo}

**MODELO(nome, categoria, marca → MARCA)**

Chaves: {nome}

{nome} -> {categoria, marca}

**SEGURADORA(nome, morada, telefone, e-mail)**

Chaves: {nome}, {telefone}, {e-mail}

{nome} -> {morada, telefone, e-mail}

{e-mail} -> {nome, morada,telefone}

{telefone} -> {nome, morada, e-mail}

**SEGURO(id, tipo, descricao, preco)**

Chaves: {id}, {tipo, descricao}

{id} -> {tipo, descricao, preco}

{tipo, descricao} -> {id, preco}

**PLANO\_ALUGUER(id, nome, descricao, preco\_diaria)**

Chaves: {id}, {nome}

{id} -> {nome, descricao, preco\_diaria}

{nome} -> {id, descricao, preco\_diaria}

**EXTRA(id,nome,descricao,valor)**

Chaves: {id}, {nome}

{id} -> {nome, descricao, valor}

{nome} -> {id, descricao, valor}

**ALUGUER(id,cliente\_id→CLIENTE,morada→LOCAL\_LEVANTAMENTO, plano\_id→PLANO\_ALUGUER,data\_inicio,data\_fim,hora\_inicio, hora\_fim)**

Chaves: {id}

{id} -> {cliente\_id, morada, plano\_id, data\_inicio, data\_fim, hora\_inicio, hora\_fim}

**QUANTIDADE(aluguer\_id→ ALUGUER, extra\_id → EXTRA, qtd)**

Chaves: {aluguer\_id, extra\_id}

{aluguer\_id, extra\_id} -> {qtd}

**ENTREGA(aluguer\_id→ ALUGUER, funcionario\_id → FUNCIONARIO estado\_veiculo)**

Chaves: {aluguer\_id}

{aluguer\_id} -> {funcionario\_id, estado\_veiculo}

**DEVOLUCAO(aluguer\_id→ALUGUER, funcionario\_id → FUNCIONARIO, estado\_veiculo)**

Chaves: {aluguer\_id}

{aluguer\_id} →->{funcionario\_id, estado\_veiculo}

Análise de Violações da Boyce-Codd Normal Form

Para identificar violações da Boyce-Codd Normal Form checamos se no nosso modelo relacional existem dependências funcionais em quais, para cada dependência não-trivial, o seu lado esquerdo é composto por atributos que não são superchave da relação.

O modelo proposto não viola esta forma normal, uma vez que, em todas as dependências funcionais apenas atributos que fazem parte da superchave estão no lado esquerdo. Em relações como **Aluguer e Funcionário,** apenas com seus IDs é que somos capazes á acessar outros atributos da relação, isto pois, em ambos os casos o ID é o único atributo que assumirá valor único. Além disso, em relações com mais de uma dependência funcional, como é o caso de **Pessoa**, atributos que irão assumir valores únicos e fazem parte da superchave como e-mail e id são os únicos que fazem parte do lado esquerdo das dependências funcionais, portanto, também estão na forma normal.

Análise de Violações da Terceira Forma Normal

A seguinte análise foi feita de forma que verificamos em cada relação presente em nosso modelo relacional se a regra da não-transitividade era quebrada. Dessa forma, não identificamos nenhuma quebra da regra mencionada.

Ao que tange as dependências funcionais de cada relação, a correspondência com a Terceira Formal Normal se torna perceptivel, sendo que todas estão na Boyce-Codd Normal Form.

Em conclusão, todas as relações e suas respectivas dependências funcionais propostas inicialmente possuem as propriedades **conservação de dependência e junção sem perdas**. Sendo assim, nenhuma relação precisará ser decomposta e o modelo relacional não sofrerá alterações.

Descrição do Processo de Integração IA – Dependências Funcionais e Análise das Formas Normais