BASE DE DADOS

Gestão de aluguer de instrumentos a uma loja de música

**Grupo 503 – turma 2mieic05**

João Álvaro Ferreira, [up201605592@fe.up.pt](mailto:up201605592@fe.up.pt)

João Barbosa, up[201604156](mailto:201604156)@fe.up.pt

Pedro Neto, up[201604420](mailto:201604420)@fe.up.pt

## Descrição

Este projeto baseia-se na gestão do aluguer dos instrumentos musicais, efetuada por uma loja de música. Esta loja tem várias filiais, sendo que estas servem também de armazém para os instrumentos.

Inicialmente criamos uma classe **Entidade**, onde temos toda a sua informação relevante como o nome, o número de identificação fiscal, o telefone e a morada, sendo que uma pessoa envolvida com a loja pode ser um **Cliente** ou um **Empregado.**

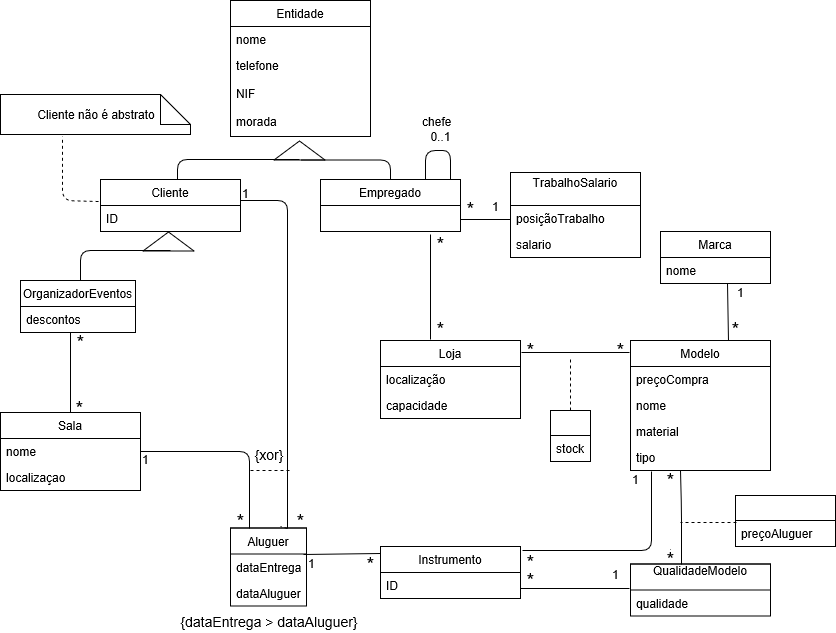
Do lado dos **Clientes,** temos um **Cliente** genérico que contém como atributo o ID que está ligado diretamente à classe **Aluguer** (que contem as informações de data de entrega e de chegada), estabelecendo uma relação *“One to Many”* ; e um **OrganizadorEventos,** derivado da classe **Cliente**, que tem como atributo descontos que possa receber ser um tipo de cliente especial e alugar (em princípio) vários instrumentos de cada vez. Estes **OrganizadorEventos** têm diversos espetáculos que decorrem, possivelmente, em multiplas **Salas** (tendo uma relação *“Many to Many”* devido a isso). Esta **Sala** tem como atributo o seu nome e localização, e tem por sua vez uma ligação ao **Aluguer.**

É crucial ter esta informação sobre a organização de eventos e sobre as **Salas** já que as **Lojas** oferecem o serviço de entrega dos **Instrumentos** às **Salas** de espetáculo (ao contrário dos clientes genéricos que têm de levantar o instrumento à **Loja**) e estes têm de ser os mesmos que foram alugados pelos **OrganizadorEventos**. É importante mencionar que um **Aluguer** é feito ou por um **Cliente** genérico, ou por um **OrganizadorEventos** sendo portanto entregue a uma **Sala¸** daí existir uma relação *“xor”* entre ambas as ligações.

Quanto aos **Instrumentos**, estes têm como atributos o preço de aluguer, a condição (se está em condições de ser alugado e o quão gasto está) e um ID único. Cada **Instrumento** está associado a uma de várias **Lojas** de duas maneiras; uma relação direta, de modo a que facilmente se liste todos os **Instrumentos** e os preços destes para cada **Loja**, e uma relação mediada pelo **Modelo** de modo a facilmente filtrar o **Instrumento** a atribuir pelo modelo que o **Cliente** pretende alugar. O **Modelo** contém informação sobre o nome da **Marca** respetiva, o preço de compra (quando comprado diretamente à marca), o tipo de instrumento (percussão, cordas, sopros, etc…) e uma descrição das suas características (material das peças, número de certos componentes, etc…). Os **Instrumentos** dum **Modelo** estão distribuídos por lojas, cada uma com um stock associado.

Quanto aos **Empregados**, estes trabalham numa **Loja**, sendo que cada **Loja** tem vários **Empregados** que se relacionam uns com os outros hierarquicamente (relação de chefia).

## Diagrama de Classes UML



Atributos

**Entidade:**

-NIF

-nome

-telefone

-morada

**Cliente**:

-ID

**OrganizadorEventos**:

-descontos

**Sala:**

**­**-nome

-localização

**Aluguer:**

-dataEntrega

-dataAluguer

**Empregado**:

-salario

-posiçãoTrabalho

**Loja**:

-localização

-capacidade

**Instrumento**:

-ID

**QualidadeModelo:**

-qualidade

**Marca**:

-nome

**Modelo**:

-nome

-preçoCompra

-material

-tipo

##### Restrições

**Entidade:**

Nome e telefone não podem ser nulls, todos os clientes e empregados têm obrigatoriamente de os indicar ao registarem-se.

**Cliente:**

O ID atribuído ao cliente quando este se regista, para além de ter de existir, é único para cada cliente.

**Empregado:**

Salário e posiçãoTrabalho não podem ser null

**Sala:**

Localização não pode ser null

**Aluguer:**

dataAluguer, dataEntrega não podem ser null e dataAluguer tem de ser anterior a dataEntrega

NIF não pode ser null e o nome ser null, ou o NIF é null e o nome não é null

**Instrumento:**

PreçoAluguer e Modelo não podem ser null

**Modelo:**

preçoCompra, tipo e nomeMarca não podem ser null

**Loja:**

Capacidade não pode ser null

**EmLoja (relação):**

Modelo, loja e stock não pode ser null

### Diagrama Relacional Dependencias Funcionais

Entidade(NIF, nome, telefone, morada)

NIF -> nome, telefone, morada

Cliente(NIF->Entidade, ID)

NIF -> ID

Empregado(NIF->Entidade, salário, posiçãoTrabalho, chefe->Empregado)

NIF - > posiçãoTrabalho, chefe

posiçãoTrabalho -> salário

OrganizadorEventos(NIF->Cliente, descontos)

NIF -> descontos

Sala(nome, localização)

nome -> localização

Aluguer(nAluguer, dataEntrega, dataAluguer, NIF->Cliente, nome->Sala)

nAluguer -> dataEntrega, dataAluguer, NIF, nome

Instrumento(ID, qualidade, preçoAluguer, nome->Modelo, nAluguer->Aluguer)

ID -> qualidade, nome, nAluguer

nome, qualidade -> preçoAluguer

Marca(nome)

Modelo(nome, preçoCompra, material, tipo, marca->Marca.nome)

nome -> preçoCompra, material, tipo, marca

Loja(localização, capacidade)

localização -> capacidade

EmLoja(nome->Modelo, localização->Loja, stock)

EmpregadosDaLoja(localização->Loja, NIF->Empregado)

Todas as relações pertencem à 1ªFN (atributos atómicos) e, à 2ªFN (nenhum elemento não primo é dependente dum subconjunto de chaves candidatas, tendo em conta que, em todos os casos desta base de dados, as chaves candidatas contêm apenas um atributo).

As relações Empregado e Instrumento violam a 3ªFN (e, por sua vez, a Forma Normal de Boyce-Codd) pois contêm uma dependência não trivial em que o lado esquerdo não é uma superchave e o lado direito contém elementos não primos (posiçãoTrabalho -> salário | nome, qualidade -> preçoAluguer).

As restantes relações pertencem à Forma Normal Boyce-Codd (e, por sua vez, à 3ªFN) pois ou não contêm nenhuma dependência funcional, ou contêm uma dependência funcional cujo lado esquerdo corresponde a uma dependência cujo lado esquerdo é a chave primária.

Interrogações

Para testar o bom funcionamento da base de dados, utilizamos uma lista de queries que julgamos ser relevantes às específicas ligações desenvolvidas. Estas são:

1. **Devolver os nomes de todas as entidades associadas à loja.** Isto é, retornar o nome de todos os seus empregados e gerentes.
2. **Mostrar o nome de todos os modelos de uma marca específica (Fender) mais caros que 500 (preço de compra).** Testa as ligações entre Marca e respetivos Modelos. Foi escolhida a marca Fender devido a ter múltiplos modelos com preço superior a 500, para o que nos servimos da diretiva “*HAVING*”.
3. **Devolve o nome e o NIF de todos os empregados com um salário superior a 1100.** Testa as ligações entre Empregado, Entidade e TrabalhoSalario. Foi escolhido o valor 1100 já que existem diversos postos que garantem um salário superior a 1100, resultando numa resposta variada.
4. **Quantos instrumentos há de cada marca na empresa.** Verifica as ligações entre EmLoja, Modelo e Marca, para o que nos servimos de uma SUM do stock e um GROUPBY nome de marca.
5. **Quais os clientes que alugaram um modelo específico (Les Paul).**  Testa as ligações entre Entidade, Cliente, Aluguer e Instrumento. O modelo em questão foi escolhido devido a, apesar de ser um mesmo modelo, existir na nossa base de dados duas versões deste com qualidades diferentes (Aceitavel e Gasto) que têm a elas associadas não só preços de compra diferentes, como preços de aluguer diferentes, algo que podia ser problemático caso houvesse algum erro nas ligações da base de dados.
6. **Quais os clientes que não tem qualquer instrumento com eles.** Esta query testa a ausência de determinadas entidades em algumas ligações definidas entre Entidade, Cliente, Aluguer e Instrumento, que tinham sido testadas de maneira diferente na query anterior.
7. **Qual o modelo mais alugado e quantas vezes foi alugado.** Novamente a testar outra faceta entre Entidade, Cliente, Aluguer e Instrumento, desta vez servimo-nos da diretiva MAX associada à diretiva COUNT para verificar qual o modelo mais alugado.
8. **Total de dinheiro gasto por cada cliente em alugueres em ordem decrescente.** Testa as ligações entre Entidade, Cliente, Aluguer, Instrumento e PrecoAluguer, usando ORDER BY e DESC.
9. **Alugueres em cada sala de eventos do Porto depois de uma data específica (12/04/2018).** Servindo-se de um inner join entre Sala e Aluguer, este query verifica se a informação se mantem consistente entre ambas as tabelas. O local foi escolhido devido à maioria dos alugueres em salas de eventos serem no Porto, mas não todos, e a data foi escolhida devido a termos muitos eventos antes, durante e depois dessa data, algo que podia causar erros se as ligações tivessem sido mal feitas.
10. **Quanto tempo está alugado cada instrumento de uma marca específica (Fender).** Novamente fazendo uso da marca Fender para testes, nesta query testamos as ligações entre Modelo, Aluguer e Instrumento, servindo-nos da função *strftime* e de um inner join entre Aluguer e Instrumento para verificar que não há inconsistências entre as tabelas.

Gatilhos

Para além das interrogações e restrições, adicionamos também gatilhos que ajudam-nos a manter o bom funcionamento da base de dados.

1. **Gatilho 1** - Assegura que nenhuma loja excede a sua capacidade.
2. **Gatilho 2** - Assegura que quando um instrumento é adicionado, se não existir nenhum preço de aluguer para o par (modelo, qualidade) desse instrumento, então este é criado a partir do preço de compra do modelo.
3. **Gatilho 3 -** Assegura que não existem alugueres concorrentes numa mesma sala

Instruções de Execução

(a fazer)