BASE DE DADOS

Gestão de aluguer de instrumentos a uma loja de música

GRUPO 503 – TURMA 2MIEIC05

João Álvaro Ferreira, up201605592@fe.up.pt

João Barbosa, up201604156@fe.up.pt

Pedro Neto, up201604420@fe.up.pt

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

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.**

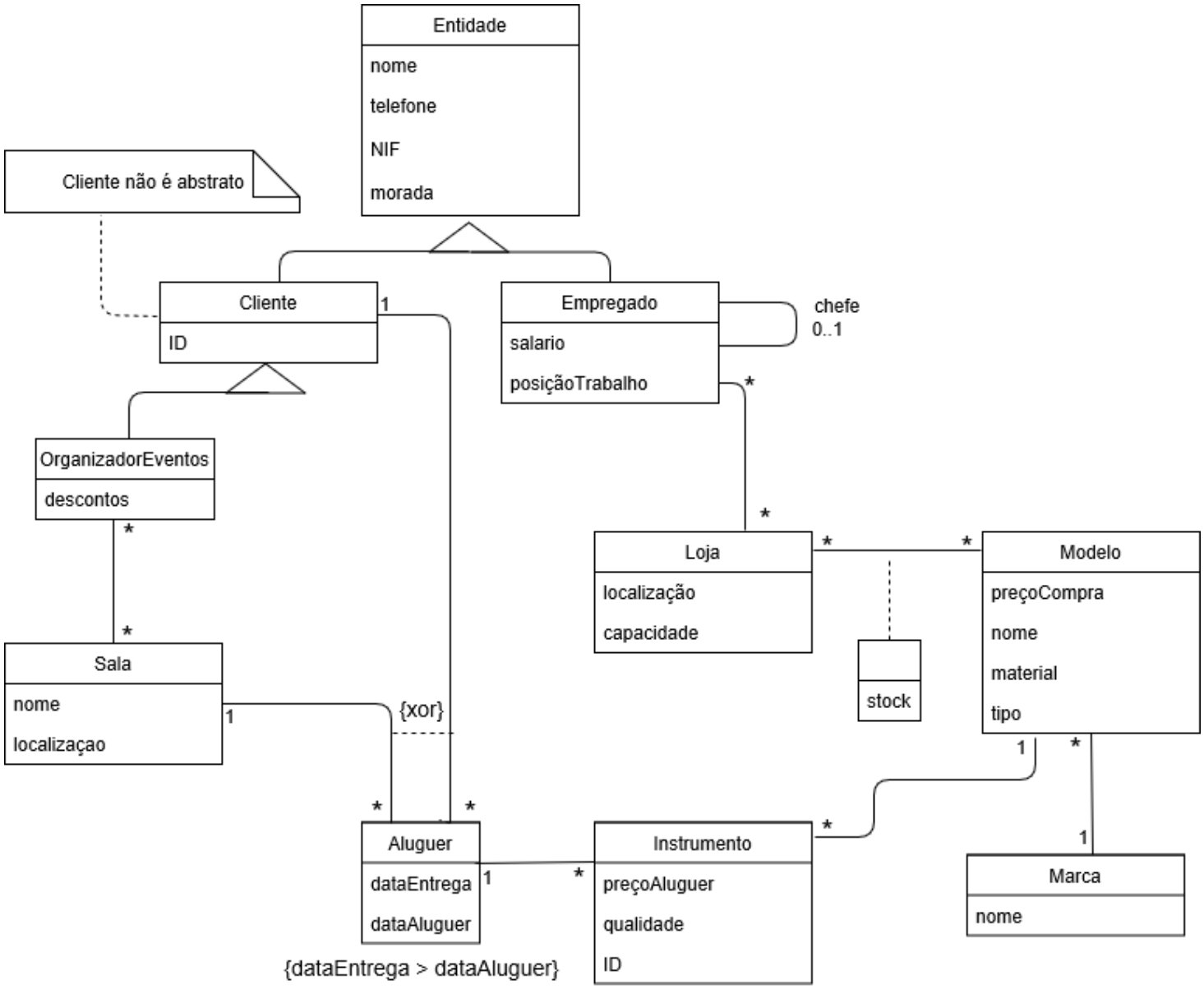
Do lado dos **Clientes,** temos um **Cliente** genérico que contém como atributo o ID que está ligado diretamente à classe **Aluguer** (que contem as informações de data de entrega e de chegada), estabelecendo uma relação *"One to Many"* ; e um **OrganizadorEventos,** derivado da classe **Cliente**, 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

-preçoAluguer

-condição

Marca:

-nome

Modelo:

-nome

-preçoCompra

-descrição

-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

MarcouSala (relação):

OrganizadorEventos e Sala não podem 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

MarcouSala(NIF->OrganizadorEventos, nome->Sala)

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.