

E-Commerce

Definição do Modelo Conceptual

Grupo 706:

- Allan Borges de Souza – up201800149

- Bianca Sequeira da Mota – up201800169

- Carolina Rosemback Guilhermino – up201800171

**Índice**

Tema do trabalho ...................................................................................3

Diagrama UML .......................................................................................4

Classes ..................................................................................................5

Client ............................................................................................................5

Product ……………………………………………………………………………5

Order ………………………………………………………………………………5

Payment info………………………………………………………………………5

Pay meth…………………………………………………………………………..5

Address……………………………………………………………………………5

District……………………………………………………………………………..5

Country ...…………………………………………………………………………6

Quantity…………………………………………………………………………...6

Rating...……………………………………………………………………….…..6

Category……………………………………………………………………….….6

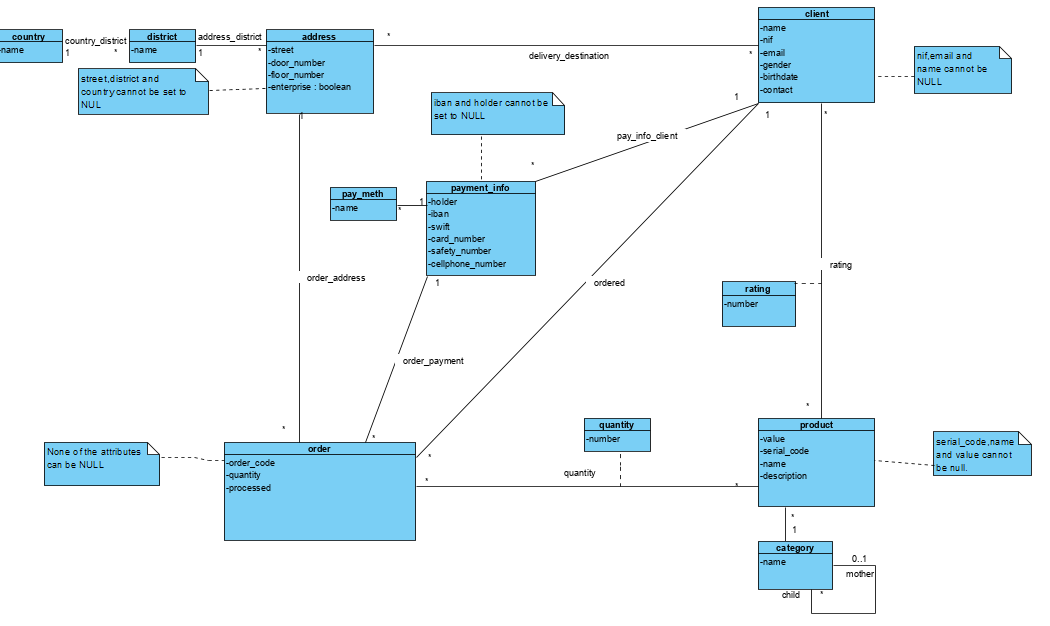
Esquema relacional, Dependências Funcionais e Formas Normais…..7

Restrições…………………………………………………………………...10

**Tema do trabalho**

O tema proposto foi o de um e-commerce. Dessa forma, poderemos utilizar dos conceitos de associação entre classes, agregação, chaves primárias entre outros. A idéia é que modelamos a base de dados com vários tipos de produtos diferentes (classes derivadas), associações entre as classes, como "order", que é uma associação direta entre uma "account" e um "product". O tema em questão foi escolhido pela relação direta com aquilo que está em alta no mercado de trabalho, visto que e-commerces se encontram cada vez mais na linha principal de compra e vendas de produtos.

**Diagrama UML**



**Classes**

* **Client**

A classe Cliente terá atributos como nome, NIF, e-mail, genêro, data de nascimento e contato. Essa classe representará o cliente que pode fazer compras no e-commerce. São clientes todos que possuem classe bancária armazenada no sistema.

* **Product**

A classe produto terá atributos como: código de série, nome, descrição e valor. Essa classe representa todos os produtos que um cliente pode comprar.

* **Order**

A classe order possui atributos como código da ordem, código de série e uma informação dizendo se a ordem já foi processada pelo e-commerce. A classe ordem corresponde a todos os produtos que um cliente deseja comprar, uma ordem só pode se relacionar com um cliente, mas pode ter vários produtos.

* **Payment info**

A classe Bank Info possui atributos como titular da conta, iban, swift, número do cartão, número de segurança e número de telefone. Essa classe foi criada pois o cliente pode desejar navegar pelo e-commerce sem inserir suas informações bancárias, nesse caso não seria criado nenhuma instância dessa classe e memória seria economizada.

* **Pay meth**

Essa classe foi criada para que pudéssemos saber qual método de pagamento o usuário irá escolher e quais atributos são necessários para tal método.

* **Address**

A classe Address possui atributos como rua, número da porta, andar e um booleano informando se a morada corresponde a uma empresa ou não. Essa classe foi criada pois um cliente possui determinada morada, mas pode fazer uma ordem e entregá-la em outra morada.

* **District**

A classe District foi criada pois vários endereços podem estar em um distrito, logo assim economizamos memória.

* **Country**

A classe Country foi criada pois vários distritos podem estar em um país, logo assim economizamos memória.

* **Quantity**

A classe quantity é uma classe associação entre order e product que mostra a quantidade de cada produto na classe order.

* **Rating**

A classe rating é uma classe associação entre cliente e produto que diz a classificação que cada cliente da a um produto.

* **Category**

A classe category é uma classe com uma auto-associação pois pode haver várias outras categorias dentro de uma só.

**Esquema relacional, Dependências Funcionais e Formas Normais**

* client (id, nif, name, email, birthdate, gender, contact)
  + Keys: {id}, {nif}, {email}, {contact}
* address (id, street, door\_number, floor\_number, enterprise, d\_id->district)
  + Keys: {id}
* district (id, dName, country\_id->country)
  + Keys: {id}, {dName}
* country (id, cName)
  + Keys: {id}, {cName}
* pay\_meth (id, pName)
  + Keys: {id}, {pName}
* orderr (order\_code, processed, client\_id->client)
  + Keys: {order\_code}
* category (id, catename, mother\_id->category)
  + Keys: {id}, {catename}
* payment\_info (id, iban, holder, swift, card\_number, safety\_number, cellphone\_number, client\_id->client)
  + Keys: {id}, {iban}
* product (serial\_code, value, prod\_name, description, cat\_id->category)
  + Keys: {serial\_code}, {prod\_name}
* quantity (order\_code, s\_code, quant)
  + Keys: {order\_code}, {s\_code}
* rating (client\_id>client, serial\_code->product, rate)
  + Keys: {client\_id}, {serial\_code}´
* delivery\_destination (client\_id->client, address\_id->address)
  + Keys: {client\_id}, {address\_id}
* order\_address(order\_code->orderr, address\_id ->address)
  + Keys: {order\_code}, {address\_id}
* order\_payment (order\_code->order, pay\_id->payment\_info)
  + Keys: {order\_code}, {pay\_id}

**Restrições**

|  |  |
| --- | --- |
| client | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY NOT NULL. * nif: inteiro que não pode ser nulo. Implementação: constraint NOT NULL. * email:text que não pode ser nulo. Implementação: constraint NOT NULL, check(email != ‘ ‘). |
| address | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY. * street: text que não pode ser nulo. Implementação: constraint NOT NULL, check(street != ‘ ‘). * floor\_number: inteiro maior ou igual a zero. Implementação: floor\_number >= 0. * door\_number: inteiro maior ou igual a zero. Implementação: door\_number >= 0. * d\_id: text que não pode ser nulo. É uma foreign key que referencia district id. Implementação: FOREIGN KEY (d\_id) REFERENCES district(id) ON DELETE CASCADE. |
| district | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY NOT NULL. * dName: text não nulo. Implementação:constraint NOT NULL. * country\_id: inteiro não nulo e foreign key que referencia country id. Implementação: FOREIGN KEY(country\_id) REFERENCES country(id) ON DELETE CASCADE. |
| country | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY. * cname: text não nulo. Implementação: constraint NOT NULL |
| pay\_meth | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY. * pname: text não nulo. Implementação:constraint NOT NULL. |
| orderr | * order\_code: primary key, então é único e não nulo, deve se maior ou igual a zero. Implementação: PRIMARY KEY e check(order\_code >= 0)). * client\_id: inteiro não nulo e foreign key que referencia client id. Implementação: FOREIGN KEY(client\_id) REFERENCES client(id) ON DELETE CASCADE ON UPDATE CASCADE. |
| category | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY. * catename: text, único e não nulo. Implementação: constraints UNIQUE e NOT NULL. * mother\_id: inteiro não nulo e foreign key que referencia category id. Implementação: FOREIGN KEY(mother\_id) REFERENCES category(id) ON DELETE CASCADE. |
| payment\_info | * id: primary key, então é único e não nulo. Implementação: PRIMARY KEY. * iban: inteiro único. Implementação: constraint unique. * swift: inteiro único. Implementação: constraint unique. * safety\_number: inteiro não nulo. Implementação: constraint NOT NULL. * client\_id: inteiro não nulo e foreign key que referencia client id. Implementação: FOREIGN KEY(client\_id) REFERENCES client(id) ON DELETE CASCADE ON UPDATE CASCADE. |
| product | * serial\_code: primary key, então é único e não nulo. Implementação: PRIMARY KEY NOT NULL. * prod\_name: text não nulo. Implementação: constraint NOT NULL. check(prod\_name != ‘ ’) * cat\_id: inteiro não nulo, foreign key que referencia category id. Implementação: FOREIGN KEY(cat\_id) REFERENCES category(id) |
| quantity | * order\_code: inteiro que referencia orderr, é uma primary key. Implementação: references. e PRIMARY KEY * s\_code:inteiro que referencia product, é uma primary key. Implementação: references. * quant: inteiro que deve ser maior que zero. Implementação: check(quant > 0). |
| rating | * client\_id: inteiro, primary key e referencia client. Implementação: references e PRIMARY KEY. * serial\_code: inteiro, primary key e referencia product. Implementação: references e PRIMARY KEY. * rate: inteiro que deve ser maior ou igual a zero. Implementação: check(rate >= 0). |
| delivery\_destination | * client\_id: inteiro, primary key e referencia client. Implementação: references e PRIMARY KEY. * address\_id: inteiro, primary key e referencia address. Implementação: references e PRIMARY KEY. |
| order\_address | * order\_code:inteiro, primary key e referencia orderr. Implementação: references e PRIMARY KEY. * address\_id: inteiro, primary key, referencia address e não nulo. Implementação: references, PRIMARY KEY e constraint NOT NULL. |
| order\_payment | * order\_code:inteiro, primary key e referencia orderr. Implementação: references e PRIMARY KEY. * pay\_id: order\_code:inteiro, primary key e referencia payment\_info. Implementação: references e PRIMARY KEY. |