U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

E-Commerce

Definição do Modelo Conceptual

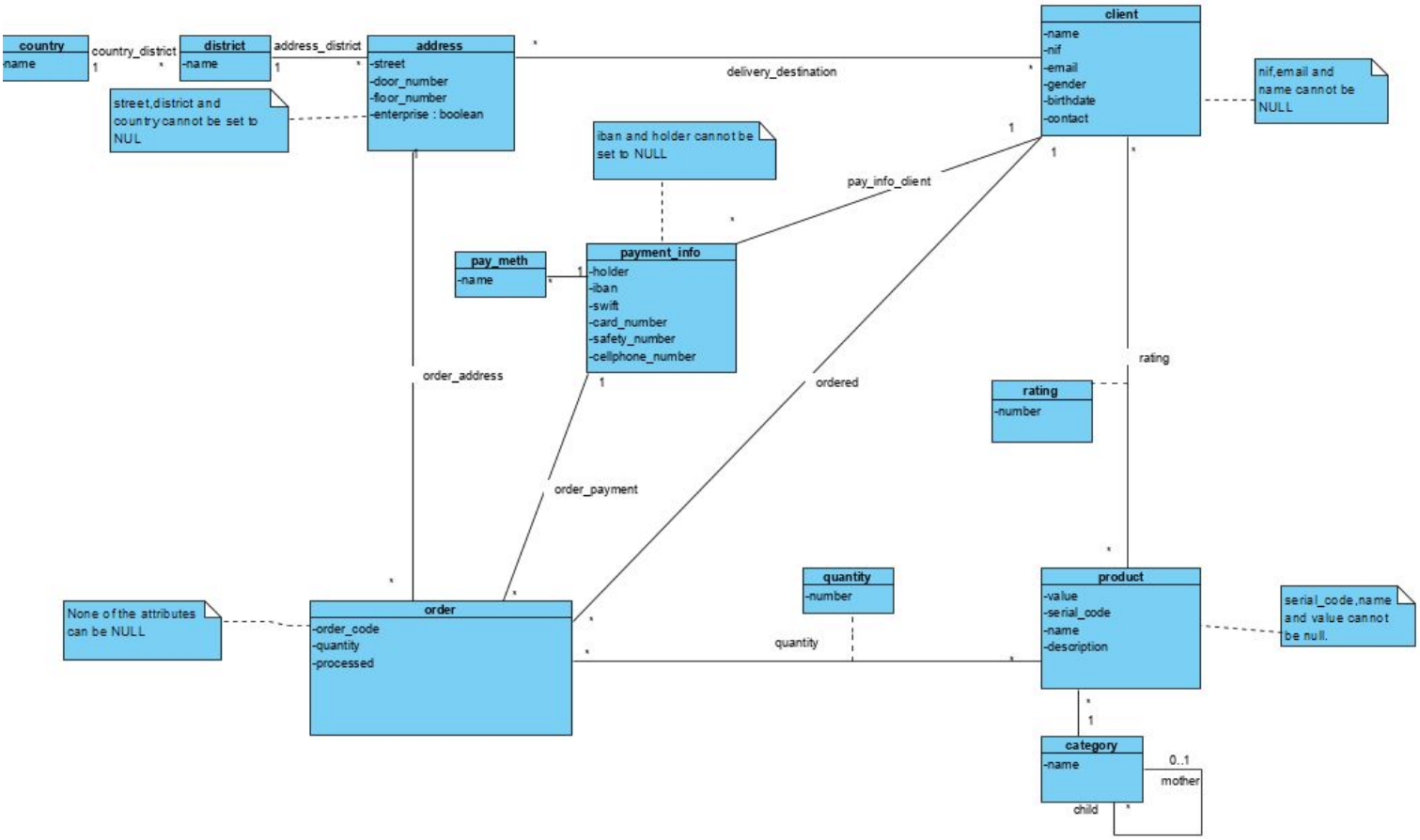
Grupo 706:
- Allan Borges de Souza – up201800149
- Bianca Sequeira da Mota – up201800169
- Carolina Rosemback Guilhermino – up201800171

Índice

Tema do trabalho

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 (nif, name, email, birthdate, gender, contact)

  Dependências funcionais:

  ➢ nif -> name, email, birthdate, gender, contact

  3.ª Forma Normal ✔
  BCNF ✔

- address (street, door_number, floor_number, enterprise, dName->district)

  Dependências funcionais:

  ➢ street, door_number, floor_number -> enterprise, dName

  3.ª Forma Normal ✔
  BCNF ✔

- district (dName)

  -

- country (cName)

  -

- pay_meth (pName)

  -

- order (order_code, processed, street->address, door_number->address, floor_number->address, nif->client, iban->payment_info)

  Dependências funcionais:

  ➢ order_code -> processed, street, door_number, floor_number, nif, iban

  3.ª Forma Normal ✖
  nif depende diretamente de iban, porque a relação de payment_info com cliente é de muitas para uma. Logo, ocorre uma dependência transitiva.

- category (cat_name)

  -

- payment_info (iban, holder, swift,card_number, safety_number, cellphone_number)

  Dependências funcionais:

  ➢ iban -> holder, swift,card_number, safety_number, cellphone_number

  3.ª Forma Normal ✔
  BCNF ✔

- product (serial_code, value, pName, description)

  Dependências funcionais:

  ➢ serial_code -> value, pName, description

  3.ª Forma Normal ✔
  BCNF ✔

- quantity (order_code, serial_code, quant)

  Dependências funcionais:

  ➢ order_code, serial_code -> quant

  3.ª Forma Normal ✔
  BCNF ✔

- rating (nif->client, serial_code->product, rate)

  Dependências funcionais:

  ➢ nif, serial_code -> rate

  3.ª Forma Normal ✔
  BCNF ✔

- delivery_destination (nif->client, street->address, door_number->address, floor_number->address)

  -

- ordered (nif->client, order_code->order)

  -

- country_district (dName->district, cName->country)

  Dependências funcionais:

  ➢ dName -> cName

  3.ª Forma Normal ✔
  BCNF ✔

- pay_info_client (iban->payment_info, nif->client)

  Dependências funcionais:

  ➢ iban -> nif

  3.ª Forma Normal ✔️
  BCNF ✔️

Restrições

cliente	email não pode ser uma variável null
address	street não pode ser null, todo endereço deve possuir uma rua floor_number>=0, o número da porta deve ser positivo door_number>=0, o número do andar deve ser positivo
district	dName !=' ', o nome do distrito não pode ser uma string vazia
country	cName != ' ', o nome do país não pode ser uma string vazia
pay_meth	pName != ' ', o nome do método de pagamento não pode ser uma string vazia
orderr	order_code>=0, o codigo de série não pode ser um número negativo.
category	o nome da categoria não pode ser null
payment_info	safety_number não pode ser null e deve ser positivo iban>=0, iban não pode ser um número negativo
product	prod_name não pode ser null serial_code não pode ser null
quantity	quantity>=0, quantidade não pode ser negativa
rating	rate>=0, rate não pode ser negativo
delivery_destination	street não pode ser null