

**Ciências
ULisboa**

Faculdade
de Ciências
da Universidade
de Lisboa

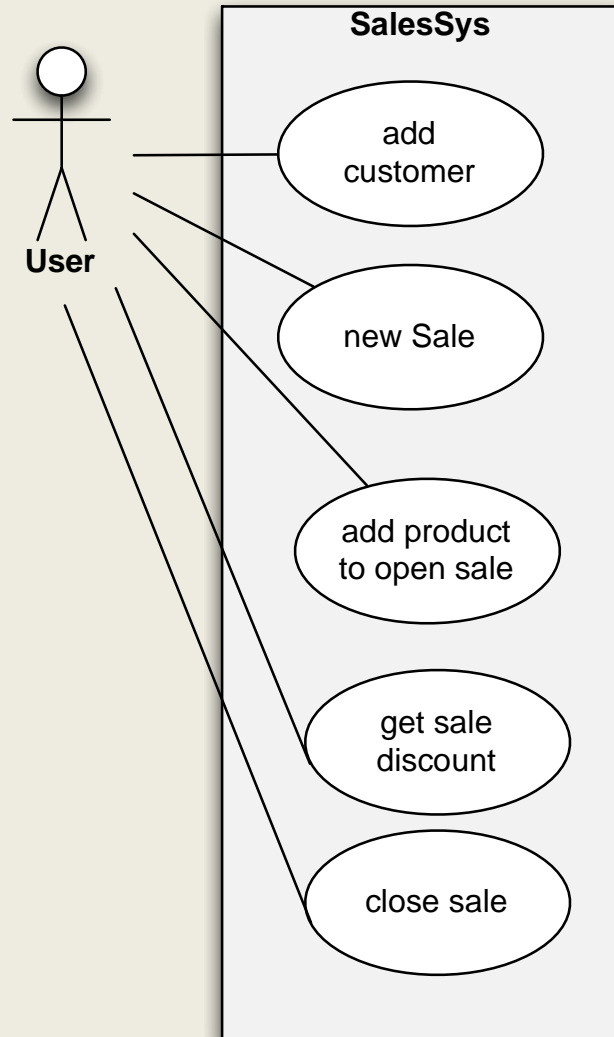
2018/2019

Construção de Sistemas de Software

SalesSys

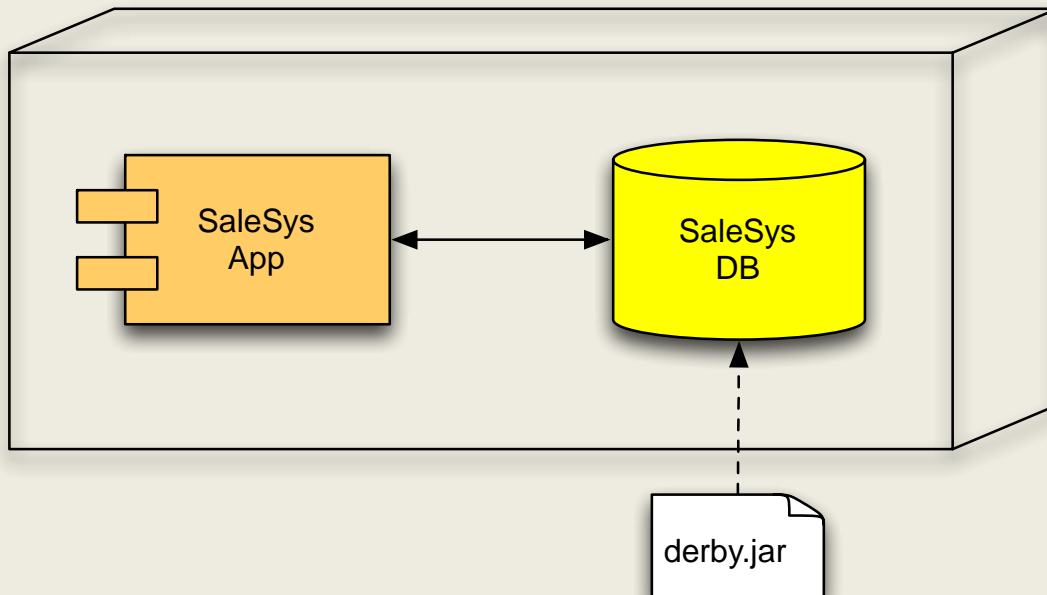
Table Module

SalesSys: Casos de Uso



- Criar cliente
- Criar nova venda
- Adicionar produto a venda
- Obter o valor do desconto de uma venda
- Fechar a venda

SaleSys: Arquitetura



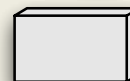
Key



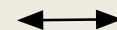
functional component



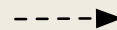
data store component



CPU/JVM



SQL Full /JDBC access

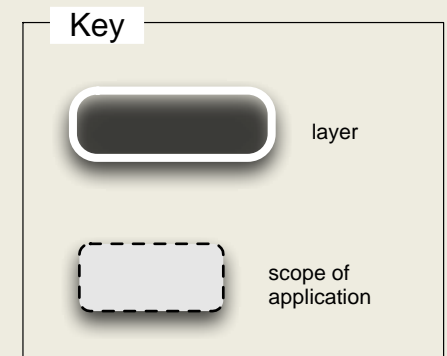
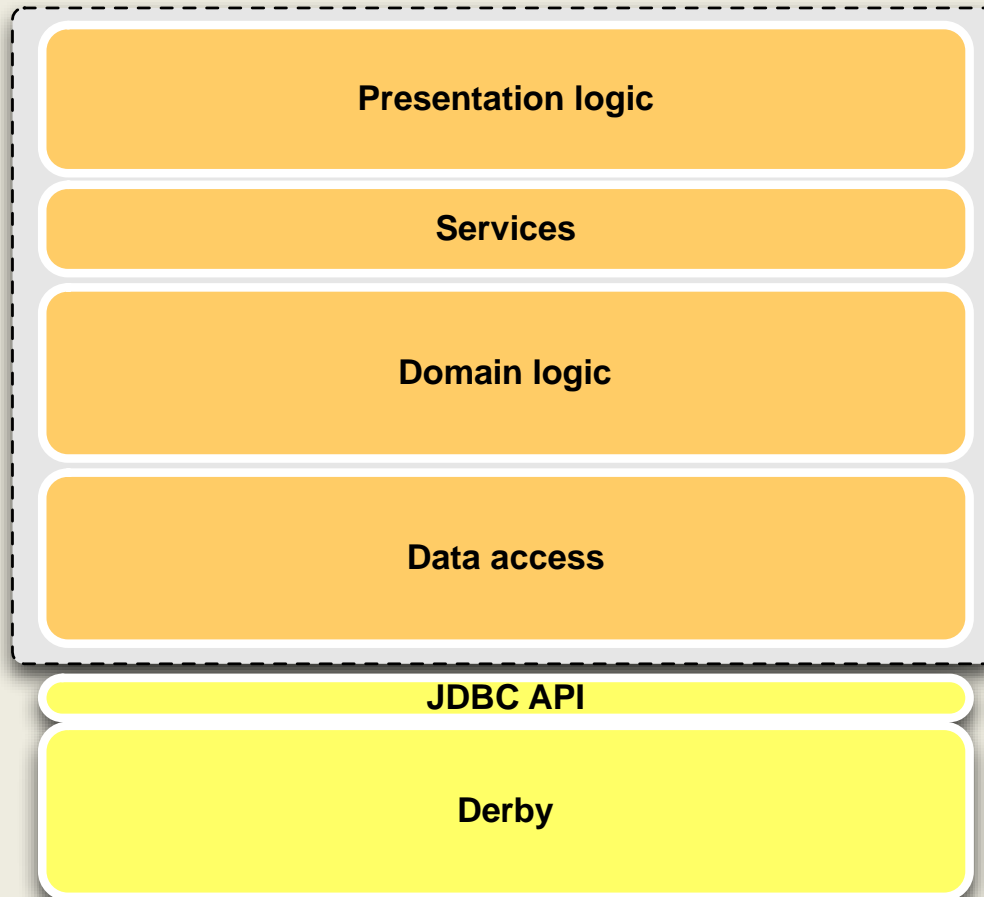


manifests

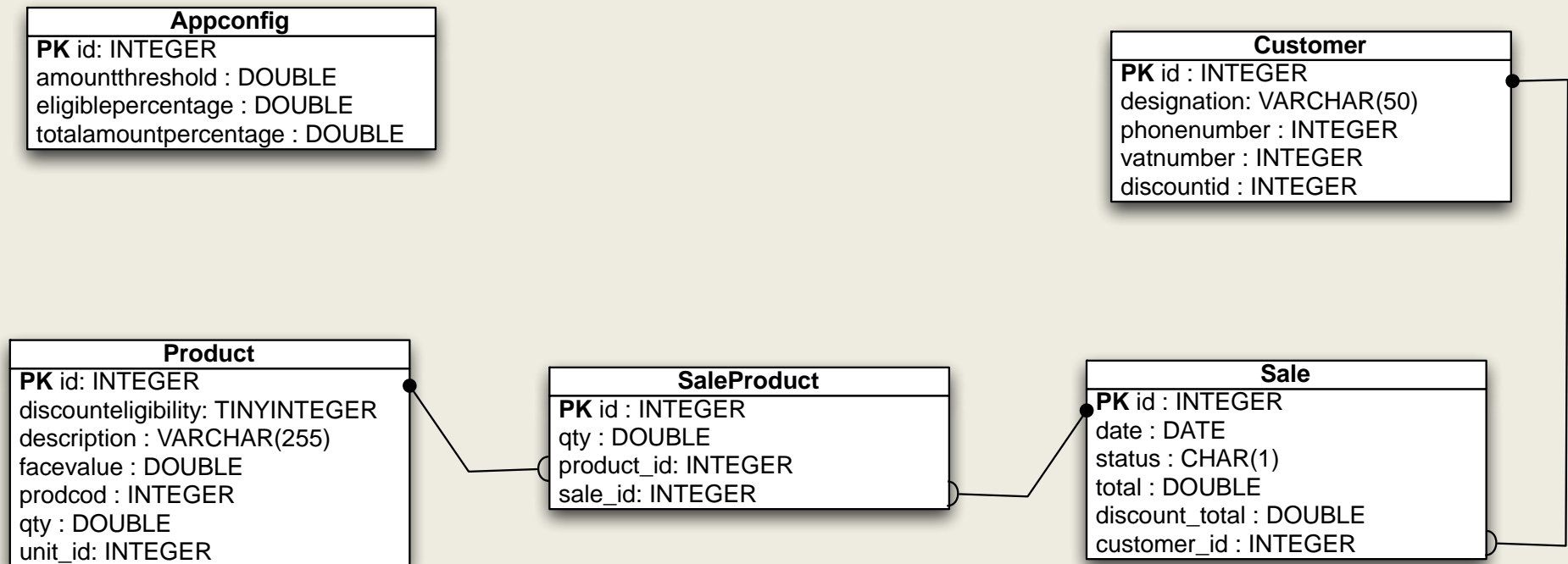


artifact

SalesSysApp. Organização em camadas (do código)



SalesSys: Modelo de Dados



SalesSys: esquema da base de dados

CREATE TABLE CUSTOMER (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1) PRIMARY KEY NOT NULL,
DESIGNATION VARCHAR(50) NOT NULL,
PHONENUMBER INTEGER,
VATNUMBER INTEGER NOT NULL UNIQUE,
DISCOUNT_ID INTEGER NOT NULL

)

CREATE TABLE PRODUCT (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1) PRIMARY KEY NOT NULL,
DESCRIPTION VARCHAR(255),
DISCOUNTELIGIBILITY SMALLINT DEFAULT 0,
FACEVALUE DOUBLE,
PRODCOD INTEGER ,
QTY DOUBLE,
UNIT_ID INTEGER

)

SalesSys: esquema da base de dados

```
CREATE TABLE PRODUCT (
```

```
...
```

```
)
```

```
CREATE TABLE SALEPRODUCT (
```

```
...
```

```
)
```

```
CREATE TABLE SALE (
```

```
...
```

```
)
```

```
CREATE TABLE APPCONFIG (
```

```
    ID INTEGER PRIMARY KEY NOT NULL,
```

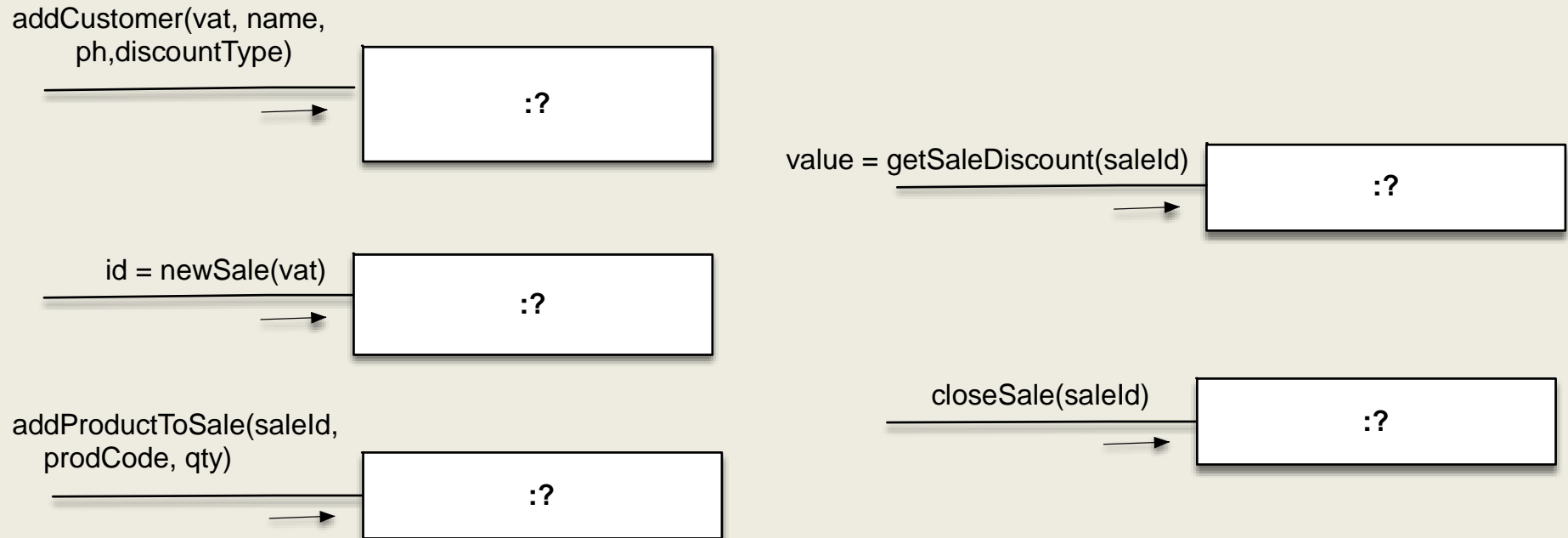
```
    TOTALAMOUNTPERCENTAGE DOUBLE NOT NULL,
```

```
    AMOUNTTHRESHOLD DOUBLE,
```

```
    ELIGIBLEPERCENTAGE DOUBLE NOT NULL
```

```
)
```

SalesSys: Solução com padrão *Table Module*



Padrão *Table Module*

- Desenhado para trabalhar com dados tabulares na forma de um **conjunto de registos** (*Record Set*) os quais podem ser obtidos por uma camada que trata do acesso aos dados (tipicamente através de interrogações SQL à base de dados)
- Um padrão em que o código da lógica de negócio (*domain logic*) é organizado **em torno das tabelas** usadas para persistir os dados
- A organização ditada pelo padrão é
 - ter uma classe por cada tabela (ou vista) da BD com métodos para atuar ou fazer cálculos sobre os dados que ela armazena
 - ter um único objecto dessa classe a lidar com toda a lógica de negócio associada aos dados guardados na tabela

SalesSys: Solução com *Table Module*

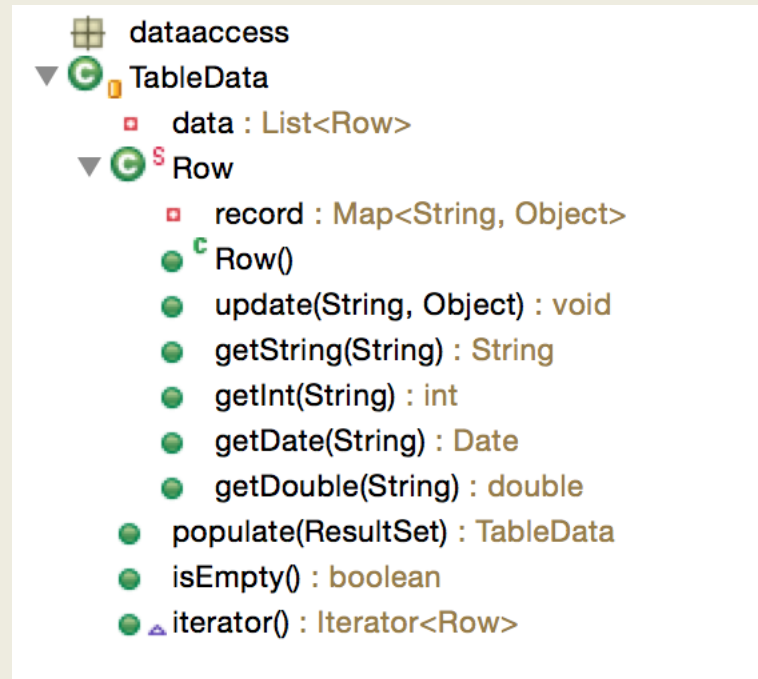
- Para a solução do SalesSys com o *Table Module* considera-se que a camada de acesso aos dados segue o padrão *Table Data Gateway*
- Mais precisamente considera-se que existe uma classe **Persistence** que
 - conhece as fontes de dados (no SalesSys, apenas uma BD relacional e que trata da interação com a base de dados através de um conector JDBC)
 - dá suporte a transações através de métodos
 - **beginTransaction()**
 - **commit()**
 - **rollback()**

SalesSys: Solução com *Table Module*

- Considera-se ainda que, para cada tabela **XXX**, existe uma classe **XXXTableDataGateway**
 - com um construtor que recebe um objeto **Persistence**
 - com métodos para fazer pesquisas que retornam conjuntos de registos (*RecordSet*) na forma de um objeto do tipo **TableData**, o qual é subtipo de **Iterable<Row>**
 - com métodos específicos para extrair os valores das várias colunas de uma **Row**
 - com métodos para fazer inserir novas entradas (**insert**) na tabela, fazer alterações (**updateXYZ**) e apagar entradas (**delete**)
- A classe **Persistence** expõe o objeto do tipo **XXXTableDataGateway** que representa cada tabela **XXX**

Table Data

- **TableData** é uma classe cujos objetos representam dados tabulares

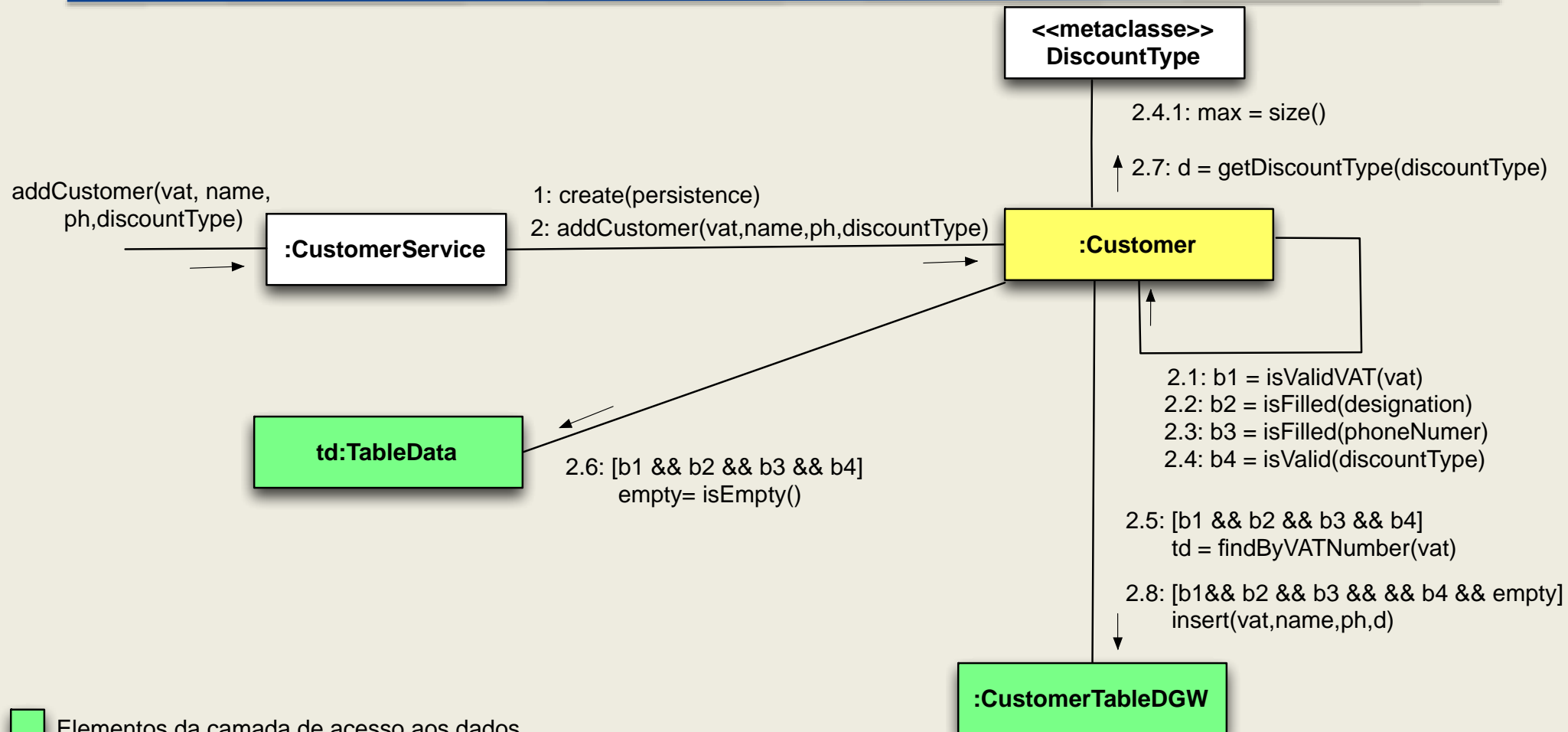



- É um tipo semelhante a **java.sql.ResultSet** mas mais abstrato


```
public interface ResultSet
    extends Wrapper, AutoCloseable
```

A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

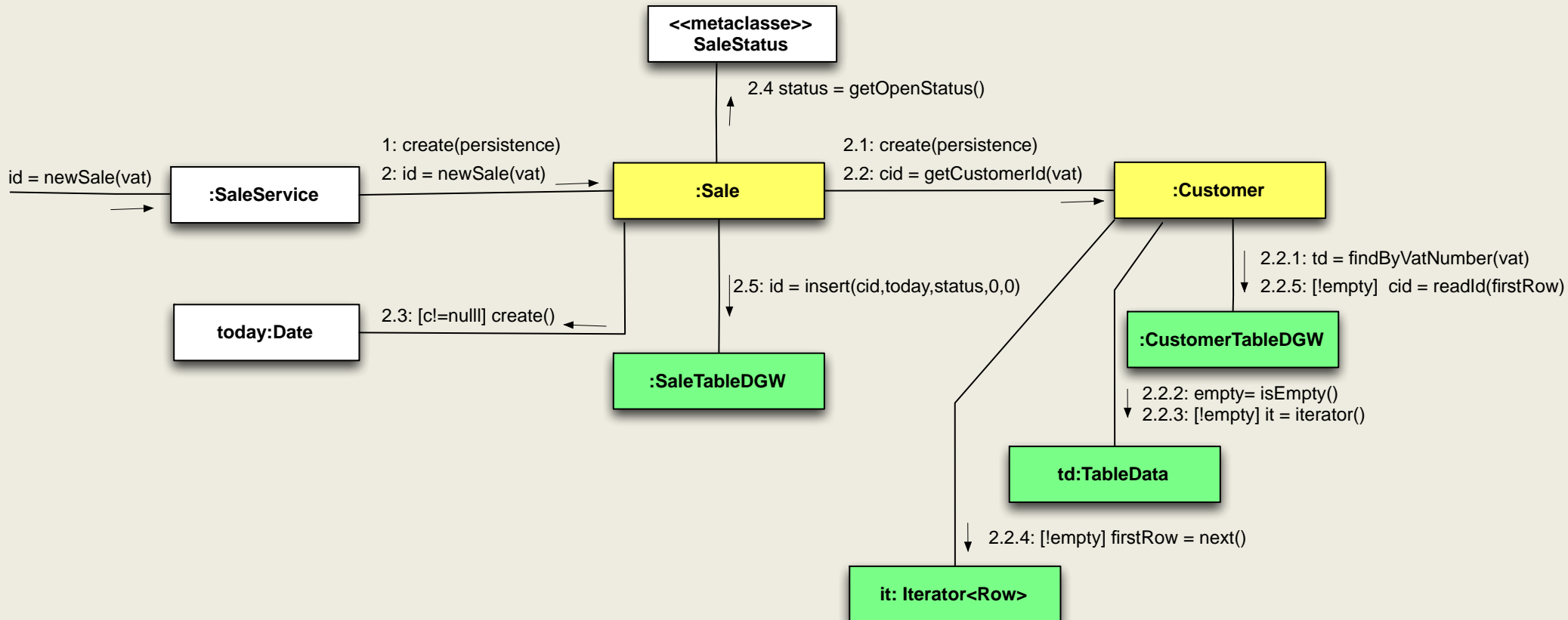
SalesSys: Diagramas de Interação



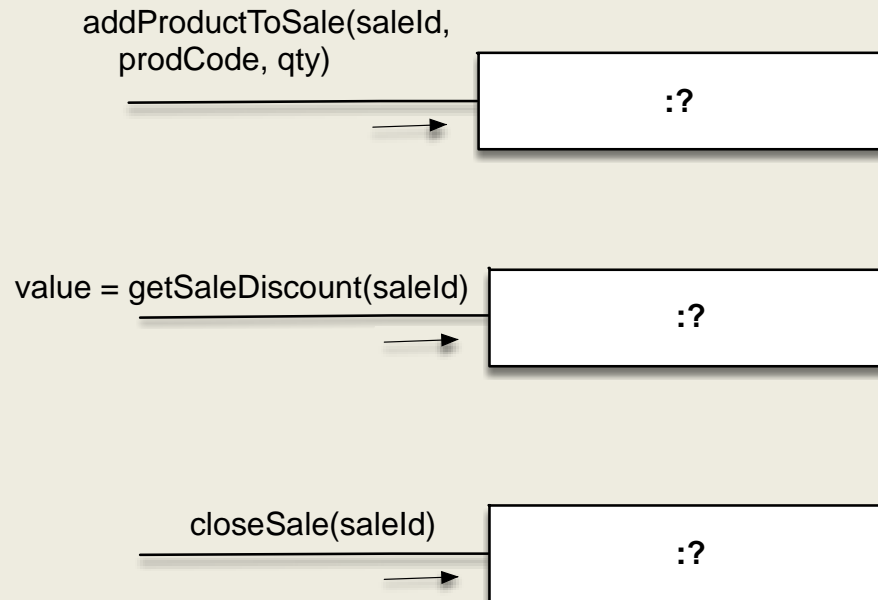
 Elementos da camada de acesso aos dados

 Table Module

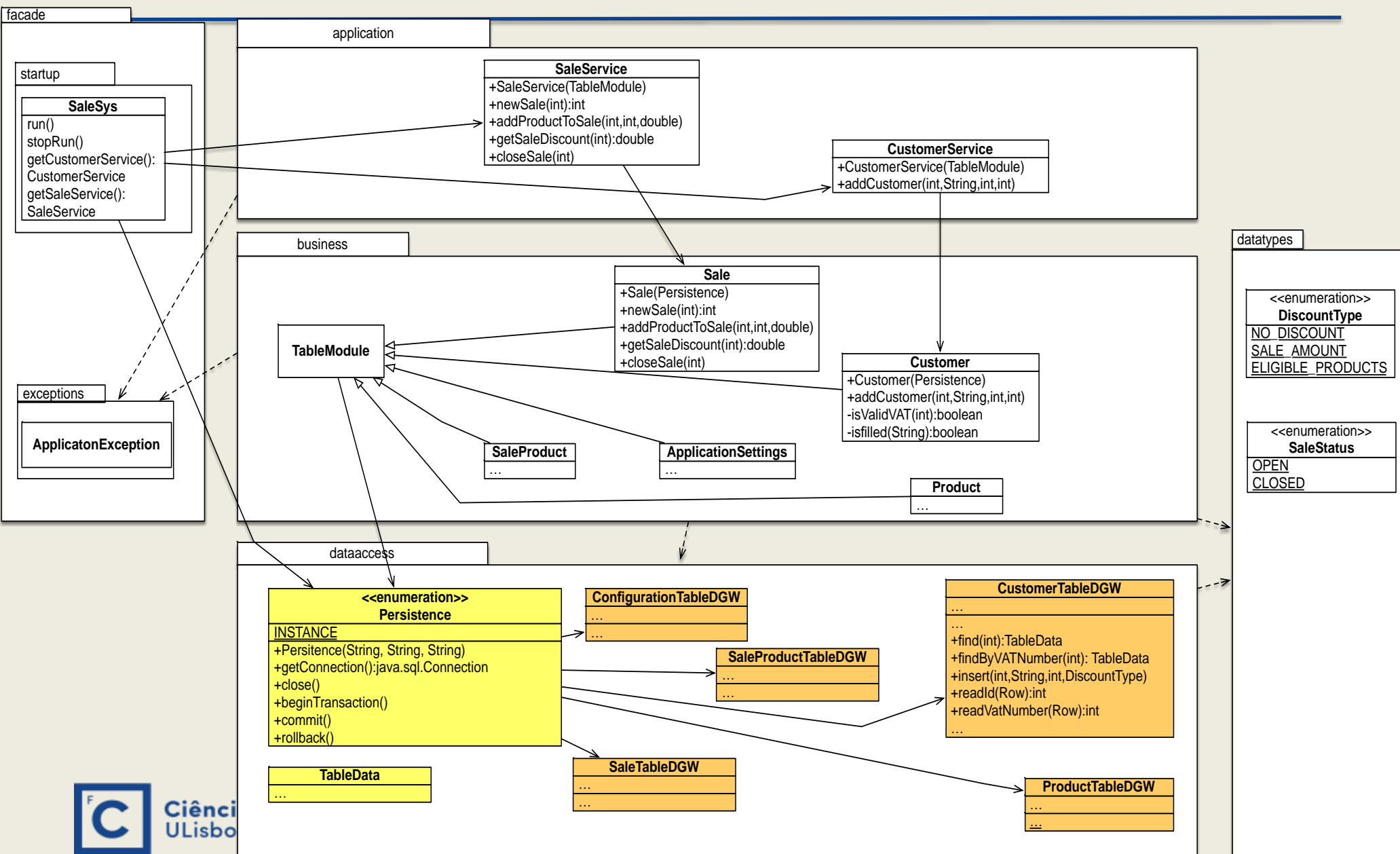
SalesSys: Diagramas de Interação



SalesSys: Diagramas de Interação



SalesSys: Diagrama de Classes



Como escolher?

Transaction Script ?

Table Module ?

Domain Model ?

- É preciso perceber a complexidade da lógica de negócio.
- Se a complexidade é baixa, a escolha óbvia é o *Transaction Script*.
- Há medida que a complexidade aumenta, o custo que advém da utilização do *Transaction Script* ou do *Table Module* torna-se muito elevado — a adição de novas *features* torna-se exponencialmente mais difícil.

Como escolher?

Transaction Script ? Table Module ? Domain Model ?

- E como avaliar a complexidade do *domain logic*?
 - Uma tarefa difícil que deve ser realizada por alguém com experiência, depois de fazer uma análise inicial dos requisitos do sistema
- Outros fatores que podem pesar na escolha
 - a familiaridade da equipa com o *Domain Model* (e a sua qualidade)
 - o apoio do ambiente de desenvolvimento aos *Record Sets* (eg., .NET)