Ciências ULisboa
Faculdade de Ciências da Universidade de Lisboa

2019/2020

Construção de Sistemas de Software

Table Module

Ciências ULisboa | Informática

CSS (2019/2020)
Caderno de Exercícios
Folha 2

Padrões para organização da camada de negócio

1. Suponha que se pretende desenhar a camada de negócio do SalesSys recorrendo ao padrão *Table Module*. Esta solução deve basear-se numa camada de acesso aos dados desenhada de acordo com o padrão *Table Data Gateway*, implementada de encontro o modelo de dados concebido anteriormente (folha 1). Especificamente, vai-se considerar o caso em que esta camada trata da persistência numa base de dados embebida *Derby* utilizando o *JDBC*.

   (a) Com a ajuda de um ou mais diagramas, explique a diferença entre a solução pretendida e uma solução em que a camada de acesso aos dados não persiste os dados mas apenas os mantém em memória.

   (b) Explique em que consiste o padrão *Table Module* para a organização da camada de negócio de uma aplicação e quais são as vantagens e desvantagens decorrentes da sua aplicação.

   (c) Elabore um diagrama de interação que defina o comportamento da operação addCustomer de acordo com o padrão *Table Module*.

   (d) Repita o que fez anteriormente para as operações:

   - newSale,
   - addProductToSale,
   - getSaleDiscount,
   - closeSale.

   (e) Elabore um diagrama de classes da solução obtida. Apresente as classes da camada de negócio e também as classes da camada de acesso a dados das quais esta camada depende.

   (f) Implemente os casos de uso de acordo com a solução de desenho que concebeu.

2. Suponha que se pretende desenhar a camada de negócio do BoardGamesCafé de acordo com o padrão *Table Module*.

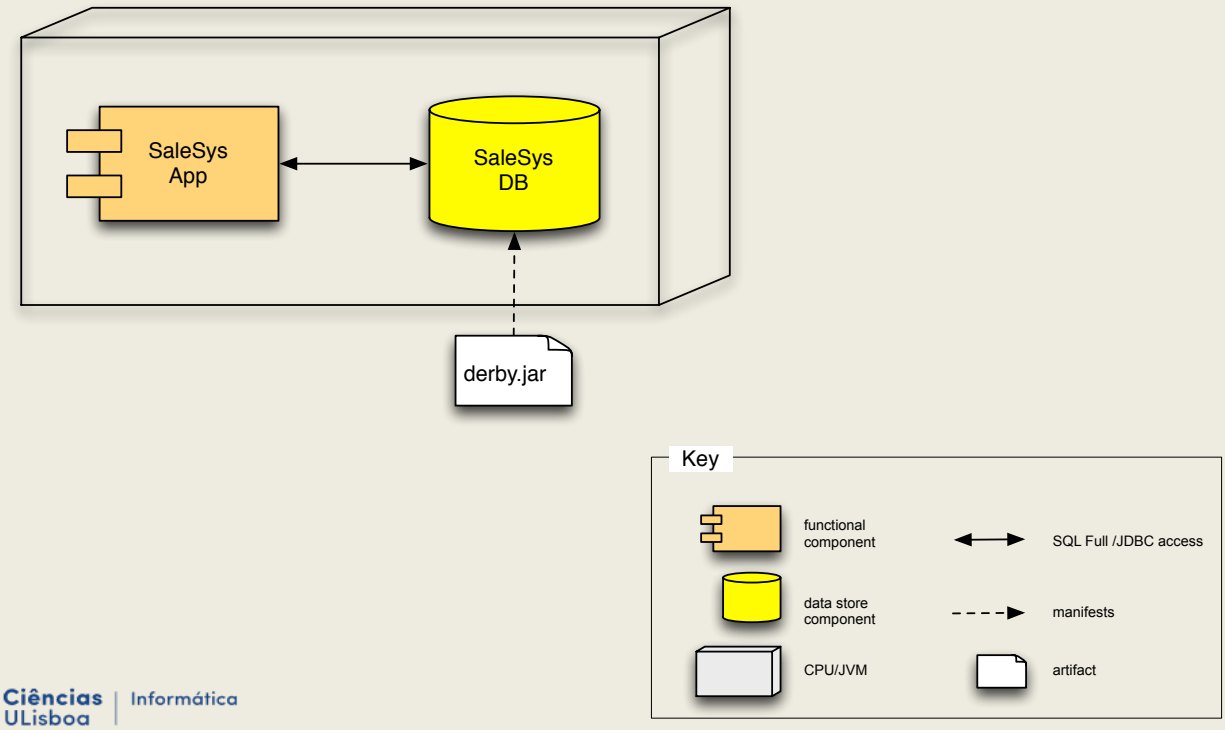
   (a) Elabore um diagrama de interação que defina o comportamento da operação requisitarJogo de acordo com o padrão *Table Module*.

Exercícios

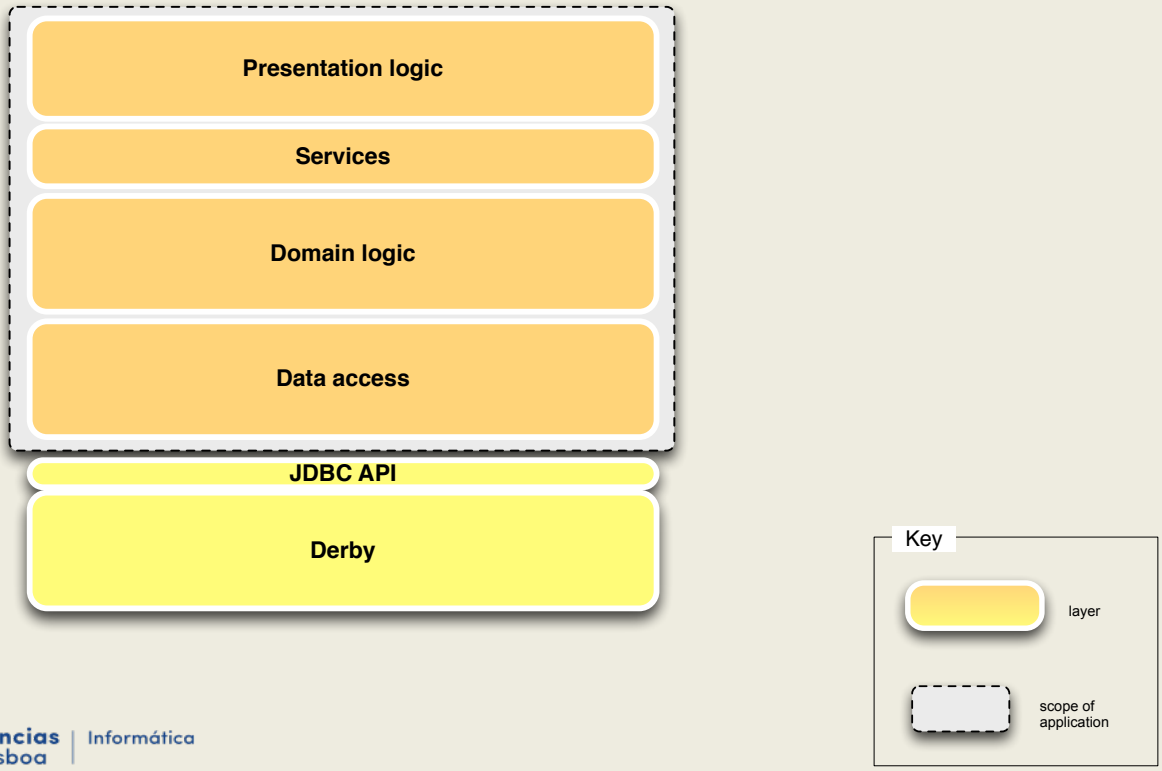
Suponha que se pretende desenhar a camada de negócio do SalesSys recorrendo ao padrão *Table Module*. Esta solução deve basear-se numa camada de acesso aos dados desenhada de acordo com o padrão *Table Data Gateway*, implementada de encontro o modelo de dados concebido anteriormente (folha 1). Especificamente, vai-se considerar o caso em que esta camada trata da persistência numa base de dados embebida *Derby* utilizando o *JDBC*.

(a) Com ajuda de um ou mais diagramas, explique a diferença entre a solução pretendida e uma solução em que a camada de acesso aos dados não os persiste mas apenas os mantém em memória.

SalesSys: Arquitetura



SalesSysApp: Organização em camadas (do código)



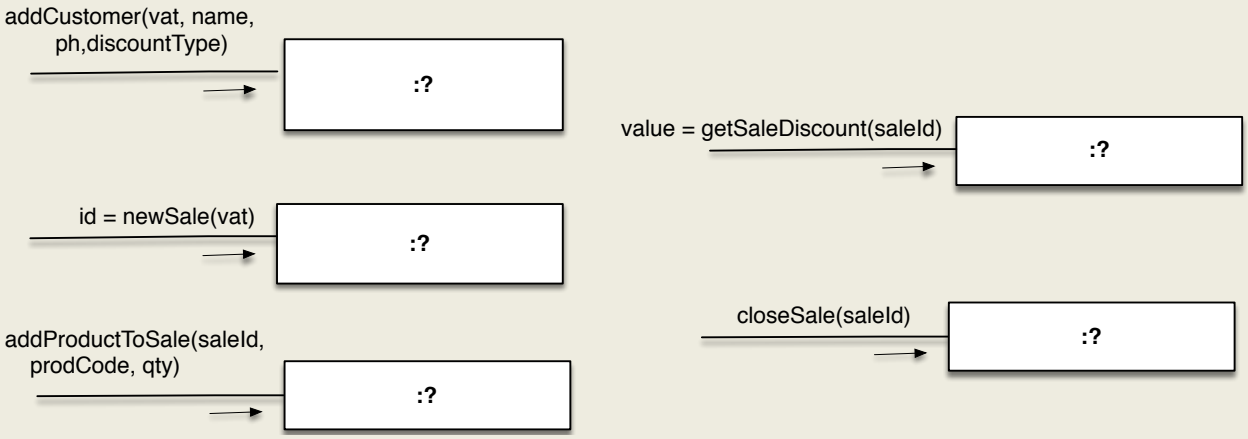
Exercícios

(c) Elabore um diagrama de interação que defina o comportamento da operação addCustomer de acordo com o padrão *Table Module*.

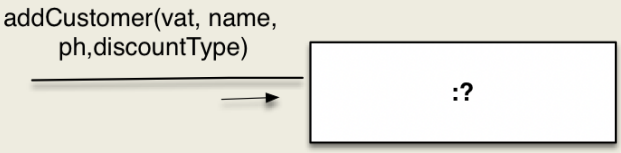
(d) Repita o que fez anteriormente para as operações:

- newSale,
- addProductToSale,
- getSaleDiscount,
- closeSale .

SalesSys: Solução com padrão *Table Module*

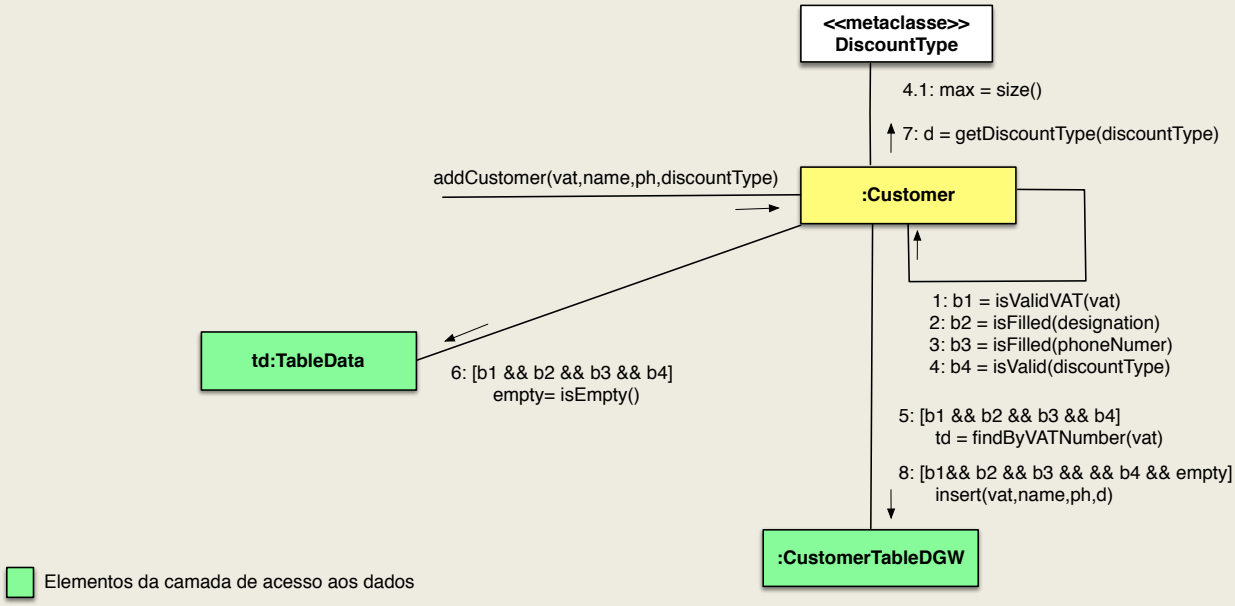


SalesSys: Solução com *Table Module*



- **Validação**
  - Verificar se código do tipo de desconto é válido
  - Verificar se o vat é válido
  - Verificar se o nome e o telefone estão preenchidos
  - Verificar se já há um cliente com o vat dado (*to fail fast*)
- **Processamento**
  - Assegurar que o novo cliente é guardado de forma persistente

SalesSys: Diagramas de Interação



Ciências ULisboa | Informática

Table Data

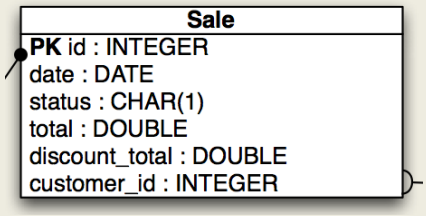
- `TableData` é uma classe cujos objetos representam dados tabulares

```
dataaccess
TableData
    data : List<Row>
    Row
        record : Map<String, Object>
        Row()
        update(String, Object) : void
        getString(String) : String
        getInt(String) : int
        getDate(String) : Date
        getDouble(String) : double
    populate(ResultSet) : TableData
    isEmpty() : boolean
    iterator() : Iterator<Row>
```

Ciências ULisboa | Informática

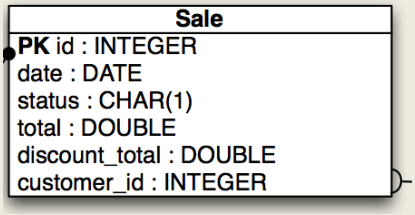
SalesSys: Solução com *Table Module*

- Validação
  - **Recorrendo aos serviços prestados pela camada de dados,** verificar se há um cliente com aquele vat
- Processamento
  - **Recorrendo aos serviços prestados pela camada de dados,** registar uma nova venda com da data corrente na base de dados; notar que isto implica ter de obter a chave do registo do cliente com o vat dado
  - Devolver o número de registo da venda

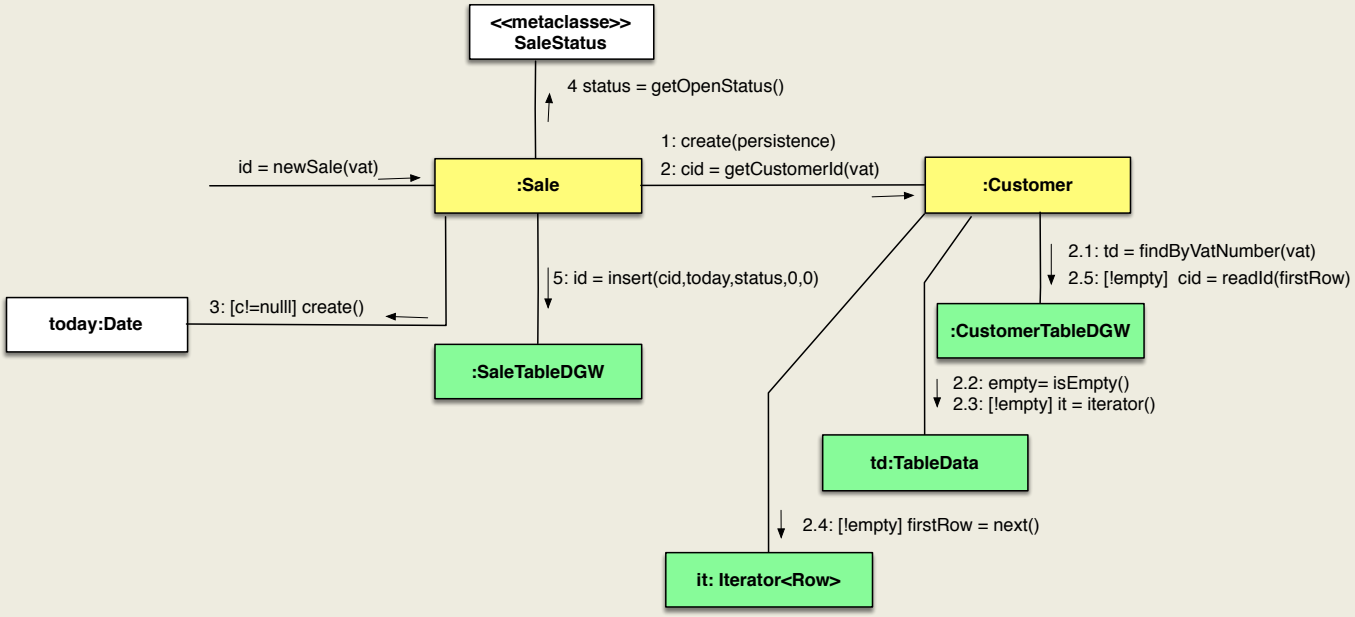


SalesSys: Solução com *Table Module*

- Responsabilidades mais importantes dependem de dados guardados na tabela *Sale* pelo que são atribuídas ao *Sale TableModule*
  - Verificar se há um cliente com aquele vat e obter a respetiva chave do registo: dependem de dados guardados na tabela *Customer* pelo que são atribuídas ao *Customer TableModule*
  - Registar uma nova venda com a data corrente na base de dados e devolver o número de registo da venda



SalesSys: Diagramas de Interação



SalesSys: Solução com *Table Module*

Para esta solução do SalesSys considera-se que na camada de acesso aos dados

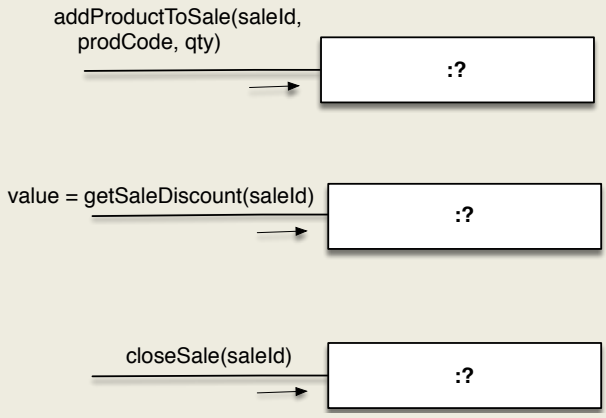
- existe uma classe `Persistence` que
  - conhece as fontes de dados — no SaleSys, apenas uma BD relacional
  - trata da interação com a base de dados através de um conector JDBC
  - dá suporte a transações através de métodos

    `beginTransaction()`

    `commit()`

    `rollback()`
- os dados são devolvidos na forma de **conjunto de registos** (***Record Set***) como objetos da classe `TableData`

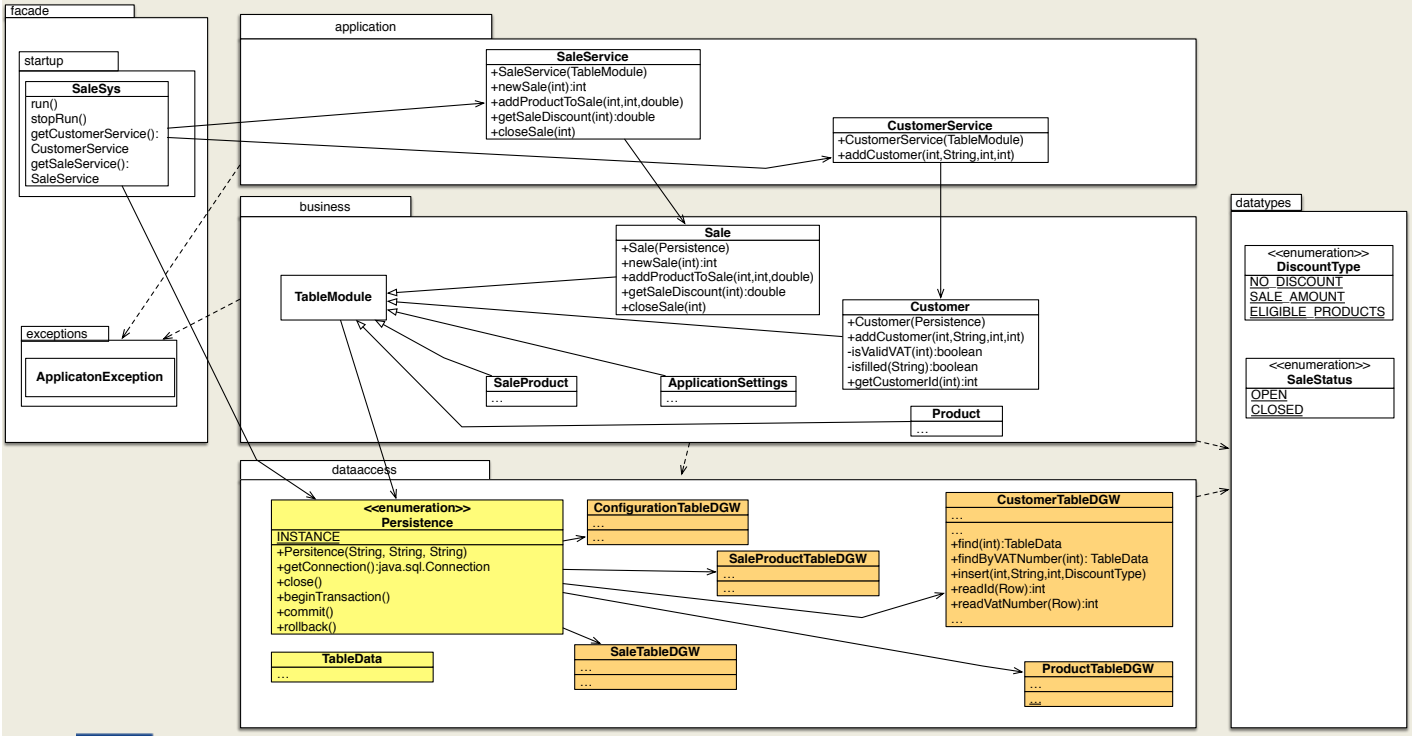
SalesSys: Diagramas de Interação



SalesSys: Detalhes de Implementação

```
public void addProductToSale(int saleId, int productCode, double qty)
        throws ApplicationException {
    // Business rule: products can be only added to open sales
    if (isClosed (saleId))
        throw new ApplicationException("Cannot add products to a closed sale.");
    // take units from the stock
    int productId = 0;
    try {
        persistence.beginTransaction();
        ProductModule product = new ProductModule(persistence);
        productId = product.getProductId(productCode);
        product.takeFromStock (productId, qty);
        // add the product to the sale
        new SaleProductModule(persistence).addProductToSale(saleId, productId, qty);
        persistence.commit();
    } catch (ApplicationException|PersistenceException e) {
        try {
            persistence.rollback();
        } catch (PersistenceException e1) {
            throw new ApplicationException("Internal error with selling product id " + productId, e1);
        }
        throw new ApplicationException("Internal error with selling product id " + productId, e);
    }
}
```

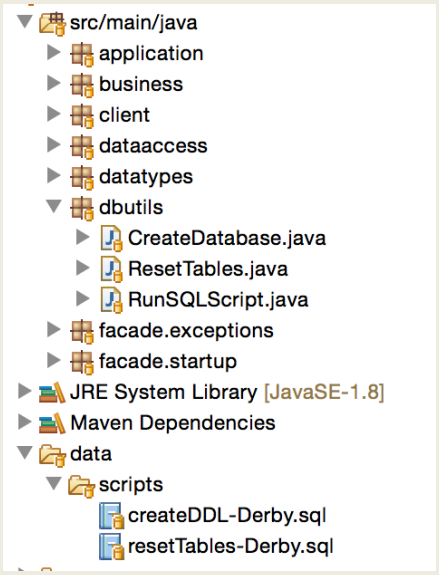
SalesSys: Diagrama de Classes



SalesSys: Detalhes de Implementação do *Startup*

```
public class SaleSys {

    private CustomerService customerService;
    private SaleService saleService;
    private Persistence persistence;

    public static final String DB_CONNECTION_STRING = "jdbc:derby:data/derby/cssdb";

    public void run() throws ApplicationException {
        // Connects to the database
        try {
        persistence = new dataaccess.Persistence(DB_CONNECTION_STRING + ";create=false", "SaleSys", "");
            customerService = new CustomerService(new CustomerModule(persistence));
            saleService = new SaleService(new SaleModule(persistence));
        } catch (PersistenceException e) {
            throw new ApplicationException("Error connecting database", e);
        }
    }

    public void stopRun() {
        // Closes the database connection
        persistence.close();
    }
```

SalesSys: Organização do Projecto




```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w
  <modelVersion>4.0.0</modelVersion>
  <groupId>pt.ulisboa.ciencias.di</groupId>
  <artifactId>table-module-v1</artifactId>
  <version>2.1</version>
  <description>An example of the application of the table module pattern
    using table data gateway to a Derby database
  </description>
  <dependencies>
    <dependency>
      <groupId>org.apache.derby</groupId>
      <artifactId>derby</artifactId>
      <version>10.12.1.1</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```