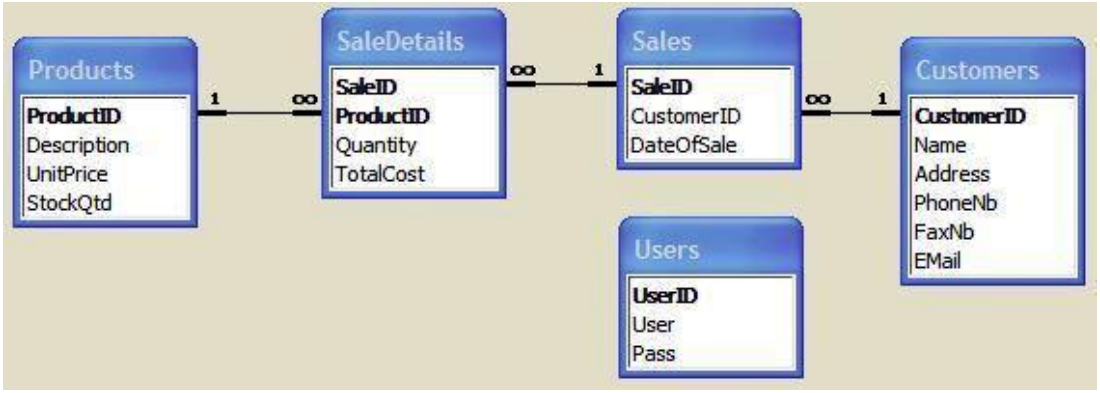
ADAV – 2005/2006

Trabalho de Referência - .NET / ADO.NET

Suponha uma loja de informática que pretende gerir Clientes, Produtos e Vendas. Para isso, necessita de uma aplicação com acesso a uma base de dados com as tabelas indicadas na figura seguinte:



Devem existir as seguintes funcionalidades:

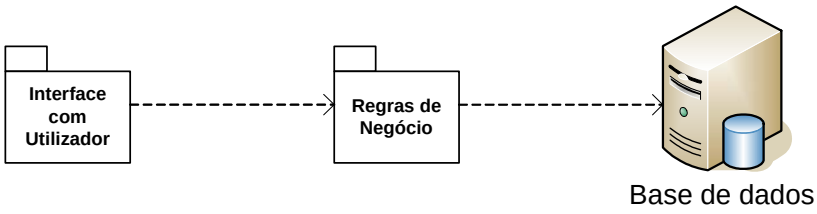
- Validação de utilizador/password;
- Inserir/actualizar clientes;
- Inserir vendas (transacções) com validação da existência dos produtos em *stock* e com actualização da respectiva quantidade;
- Recepcionar material para *stock* – considera-se que o material recepcionado corresponde a produtos já existentes em *stock*.

Alguns comentários:

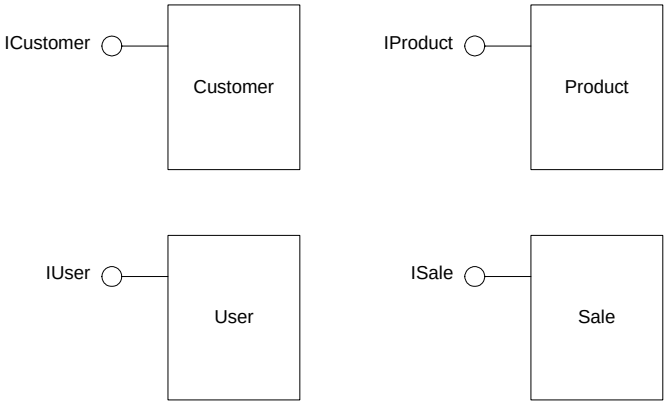
- A solução a implementar deve estar de acordo com o diagrama de packages mostrado em baixo:
- O componente (*package*) das "regras de negócio" não guarda estado o que obriga a passar o *user*/*password* em todos os métodos;
- Deve ser sempre verificado se um determinado cliente/produto existe;
- O método `CreateDetails` dos objectos da classe `Sale` tem por objectivo criar um `Dataset` com uma tabela vazia de acordo com o esquema da tabela `SaleDetails` da base de dados. Este `Dataset` será usado para o

preenchimento dos detalhes de cada venda, sendo depois um parâmetro de entrada para o método `Add` dos objectos da classe `Sale`.

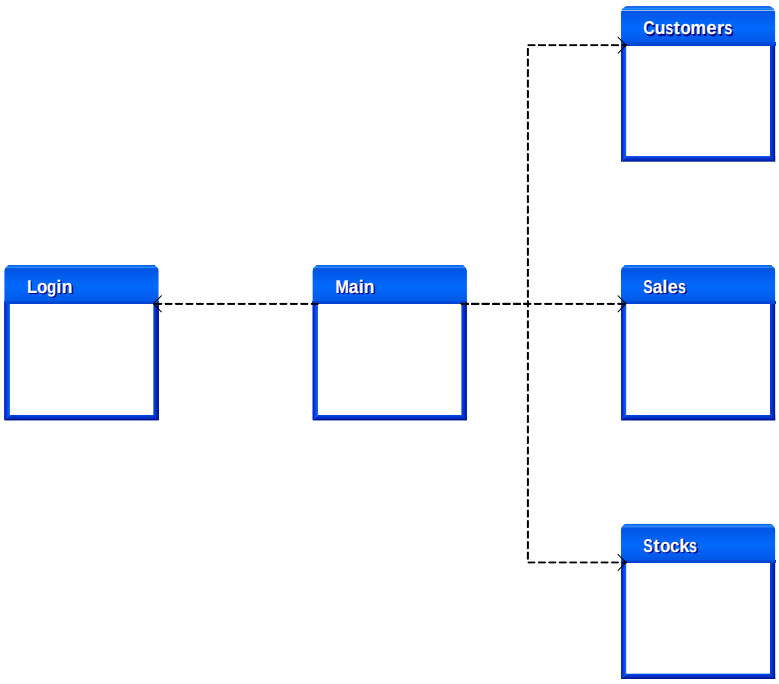
Diagrama de *Packages*



Detalhe “Regras de Negócio”



Detalhe “Interface com Utilizador”



Códigos de ‘Status’ – Retorno de alguns métodos

```
//
// Definição dos códigos de ‘status’ que poderão ser retornados
//
public enum ShopStatusEnum
{
       OK                  =  1,
       NOT_OK              =  0,
       INVALID_LOGIN       = -1,
       INVALID_KEY         = -2,
       INVALID_CUSTOMER_ID = -3,
       INVALID_PRODUCT_ID  = -4,
       INSUFICIENT_STOCK   = -5,
       ERROR               = -6
}
```

User

```
public interface IUser
{
     ShopStatusEnum Validate(string user, string pass);
}
```

Customer

```
public interface ICustomer
{
     //
     // devolve os dados do cliente com um dado id
     //
     DataSet GetByID(string user, string pass, long clientID);
```

```
        //
        // devolve os dados de todos os clientes cujo nome obedeça ao
        // padrão de pesquisa
        //
        DataSet FindByName( string user, string pass,
                            string namePattern);

        //
        // Adiciona um novo cliente e retorna o respectivo ID gerado
        // automaticamente na BD
        //
        long Add(  string user, string pass,
                   string name, string address,
                   string phone, string fax,
                   string email, out ShopStatusEnum status);

        //
        // actualiza os dados do cliente identificado pelo id
        //
        ShopStatusEnum Update(  string user, string pass,
                                long custumerID, string name,
                                string address, string phone,
                                string fax, string eMail);
}
```

Product

```
public interface IProduct
{
        DataSet GetByID(string user, string pass, long productID);

        DataSet FindByName( string user, string pass,
                            string namePattern);

        //
        // Verifica se a quantidade em stock do produto <productID>
        // é igual ou superior <quant>. Devolve o código de 'status' adequado.
        //
        ShopStatusEnum InStock( string user, string pass,
                                long productID, long quant);

        //
        // Devolve a quantidade existente em stock do produto <productID>
        //
        long GetStock( string sUser, string pass,
                       long productID, out ShopStatusEnum status);

        //
        // Adiciona ao stock existente do produto <productID>
        // a quantidade <quant>
        //
        ShopStatusEnum AddStock( string user, string pass,
                                 long productID, long quant);
}
```

Sale

```
public interface ISale
{
        //
        // Cria um <dataset> com uma tabela vazia para posterior preenchimento
        // dos detalhes das vendas. Este <dataset> será depois passado ao
        // método <Add>
        //
        DataSet CreateDetails(string user, string pass);
```

```
        //
        // adiciona uma venda para um dado cliente, especificando no parâmetro
        // dsDetails quais os produtos e respectivas quantidades vendidas
        //
        ShopStatusEnum Add( string user, string pass,
                            long customerID, DateTime date,
                            DataSet dsDetails);
}
```

Tarefas 1ª fase:

- Descarregar Projecto Base deste projecto de referência da página da disciplina de ADAV;
- Abrir o projecto – já contém as interfaces e o esqueleto das classes para implementação;
- Análisar a estrutura de componentes proposta
- Tendo em conta o enunciado do trabalho prático deste ano lectivo, comece a elaborar a estrutura de componentes e classes da sua solução, colocando uma **descrição completa** de cada método da interface (com pre e pós-condições);

Tarefas 2ª fase:

- Descarregar Projecto Base deste projecto de referência da página da disciplina de ADAV;
- Abrir o projecto – já contém algumas classes e métodos implementados;
- Analisar o código que já existe;
- **Implementar** as classes e métodos em falta;
- **Testar** o componente usando – modificando se necessário - uma aplicação para a consola;
- Desenvolver uma **aplicação cliente** do tipo Windows.