

Loja Electrónica

A aplicação Loja Electrónica foi dividida, em termos lógicos, em 4 subsistemas. A figura 1 ilustra os subsistemas e as relações de dependência existentes entre eles:

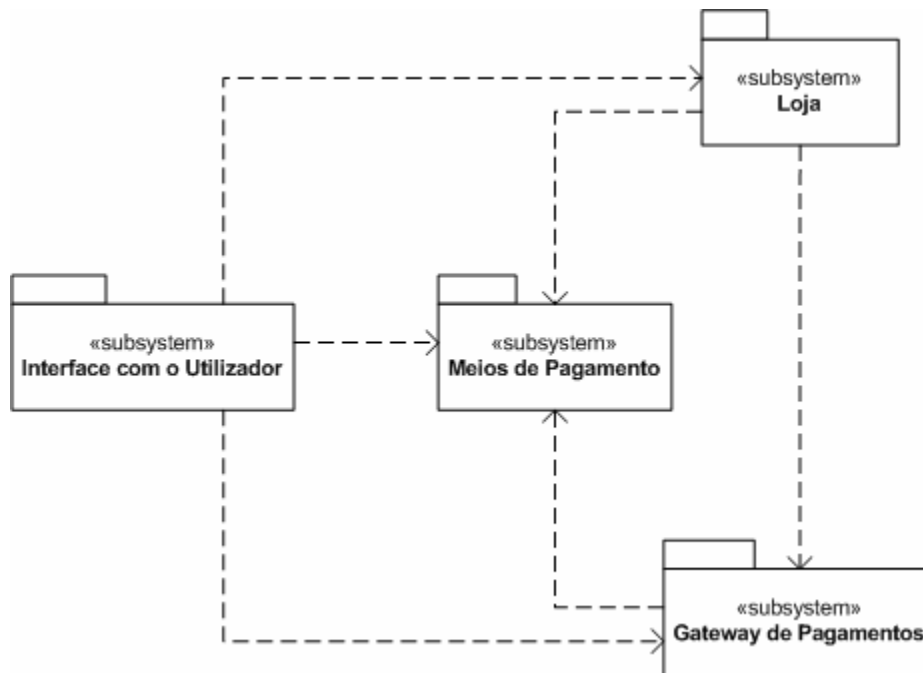


Fig 1: Os 4 subsistemas do sistema Loja Electrónica e respectivas dependências

O subsistema *Meios de Pagamento* é central à solução, uma vez que todos os outros subsistemas fazem uso de objectos da família MeioPagamento.

O subsistema *Interface com o Utilizador* é cliente dos subsistemas Loja (nas encomendas) e Gateway de Pagamentos (nos pagamentos).

O subsistema *Loja* utiliza os serviços do *Gateway de Pagamentos* de forma a assegurar o pagamento de cada encomenda.

A figura 2 mostra a família de classes *MeioPagamento*. A classe abstracta *MeioPagamento* é a classe base da qual derivam as classes *TransferenciaBancaria*, *PagamentoServicos* e *CartaoCredito*, que representam os diferentes meios de pagamento disponíveis na aplicação.

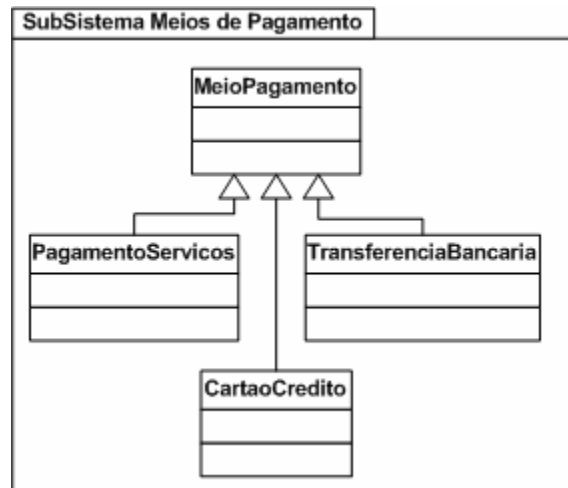


Fig 2: A hierarquia de classes da família MeioPagamento

A utilização da classe *MeioPagamento* permite um tratamento genérico de objectos diferentes (instâncias de diferentes classes derivadas). Por exemplo, para associar a uma encomenda um meio de pagamento, definiu-se a seguinte relação entre as classes:



Fig 3: Associação entre a classe Encomenda e a classe MeioPagamento

Se o modo de pagamento de uma encomenda for Pagamento de Serviços, então existirá a seguinte relação entre dois objectos:



Fig 4: Objecto da classe Encomenda associado a um objecto da classe PagamentoServicos

Se o modo de pagamento de uma encomenda for Transferência Bancária, então existirá a seguinte relação entre dois objectos:



Fig 5: Objecto da classe Encomenda associado a um objecto da classe TransferenciaBancaria

Para se saber se um determinado objecto da família MeioPagamento corresponde ao pagamento de uma determinada Encomenda, a classe Encomenda possui o método EstaPaga:

```
public bool EstaPaga( MeioPagamento mp )
{
    return this.mp.Equals( mp );
}
```

O método Equals¹ foi redefinido nas classes derivadas da família MeioPagamento. O método Equals a ser executado dependerá da classe do objecto referenciado por [this.mp](#). Isto é **Polimorfismo**.

As classes que constituem o subsistema **Gateway de Pagamentos** e as relações entre elas estão ilustradas na figura seguinte:

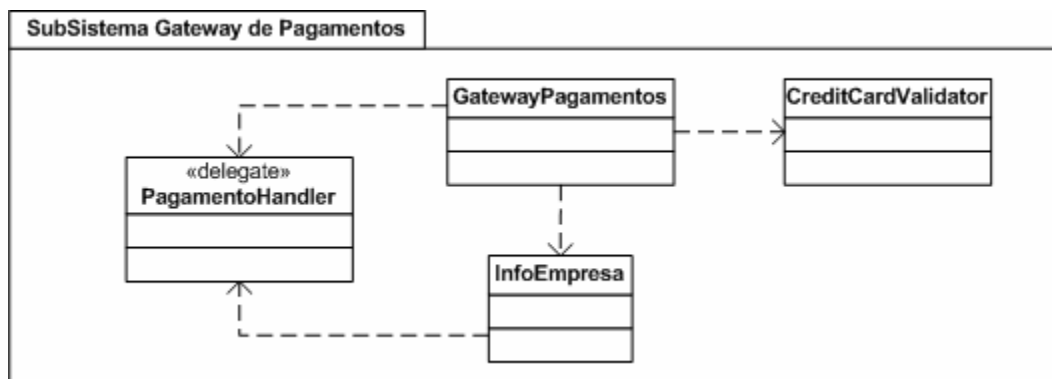


Fig 6: As classes constituintes do subsistema Gateway de Pagamentos

A classe *CreditCardValidator* serve de proxy para aceder ao Web Service que valida os cartões de crédito. A classe *GatewayPagamentos* permite a realização de pagamentos através dos diversos meios de pagamento descritos anteriormente. Além disso, possui um **serviço de notificação de pagamentos** para as empresas registadas no Gateway.

Parte-se do princípio que o Gateway presta serviço ao público em geral, e também às empresas. No caso das empresas (incluindo lojas) existe um serviço adicional de notificação dos pagamentos efectuados a seu favor. A solução escolhida para implementar o serviço de notificação passa pela utilização de **delegates com a função de callbacks**. Existe uma distinção subtil, descrita na figura 7, entre a solução escolhida e a solução alternativa de aplicar um evento.

¹ Herdado da classe [object](#)

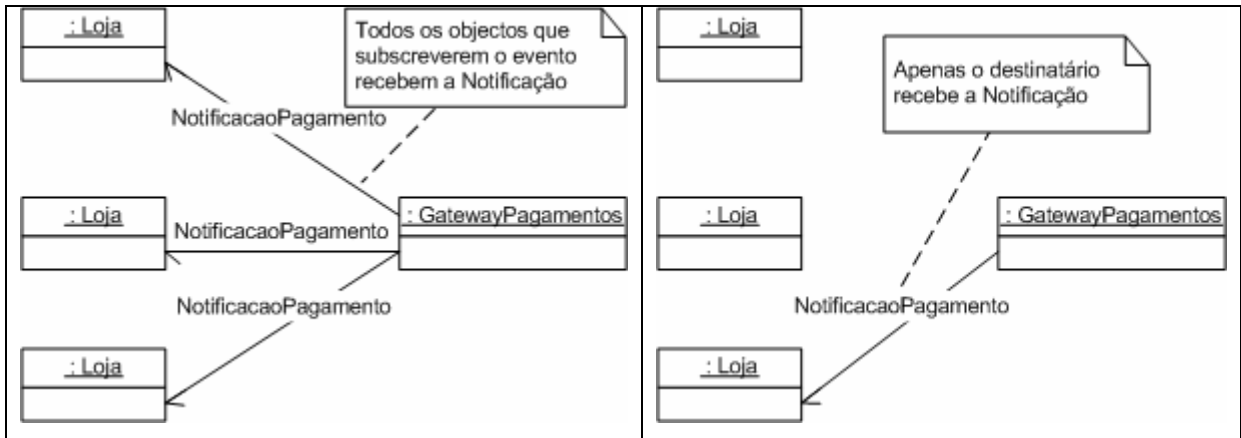


Fig 7: A notificação de um pagamento através de um Evento e através de um Callback

As classes que constituem o subsistema **Loja** e as relações entre elas estão ilustradas na figura seguinte:

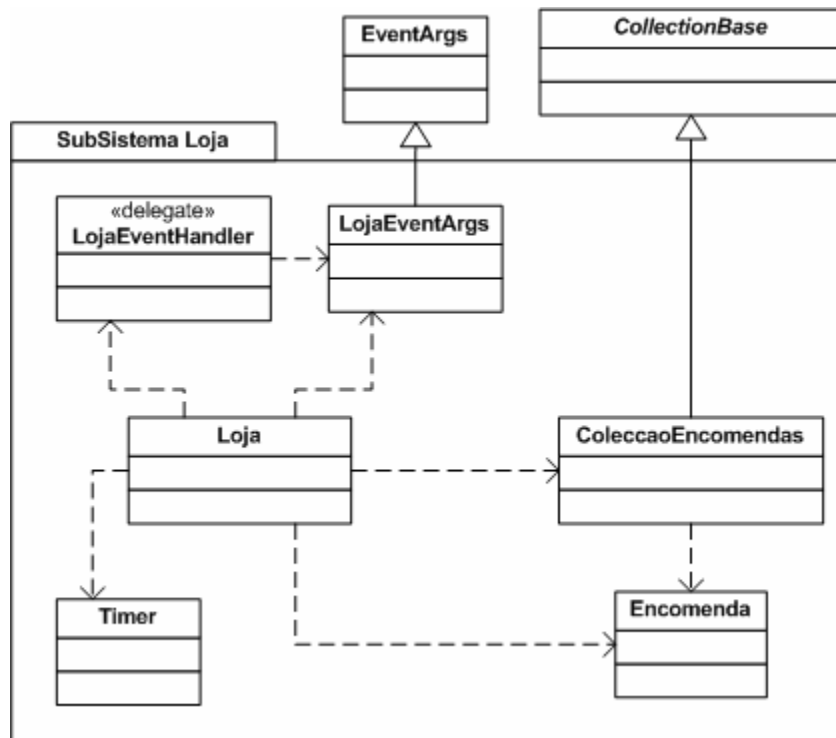


Fig 8: As classes que constituem o subsistema Loja

Um objecto da classe *ColeccaoEncomendas* destina-se a armazenar as encomendas pendentes, à espera de pagamento.

Em vez de se utilizar a classe `ArrayList`, criou-se uma classe especializada² que funciona do mesmo modo que um `ArrayList`, mas que só coleciona objectos da classe `Encomenda`.

Um objecto da classe `Timer` serve para despistar, em intervalos regulares, as encomendas cujo prazo de pagamento expirou, e proceder ao seu cancelamento.

A classe `LojaEventArgs` define a informação transmitida nos eventos disparados pela Loja. Uma vez que se abdicou de implementar a comunicação com os clientes da Loja via correio electrónico³, o evento `Mensagem` da classe `Loja` simula o disparo de uma mensagem de correio electrónico.

A classe `Loja` possui três assinaturas do método `Encomendar`⁴, uma para cada meio de pagamento. São três formas distintas de processar um pedido de uma encomenda, de acordo com as regras definidas na Loja para cada meio de pagamento.

O seguinte diagrama de sequência descreve o processamento de uma encomenda realizada através de cartão de crédito:

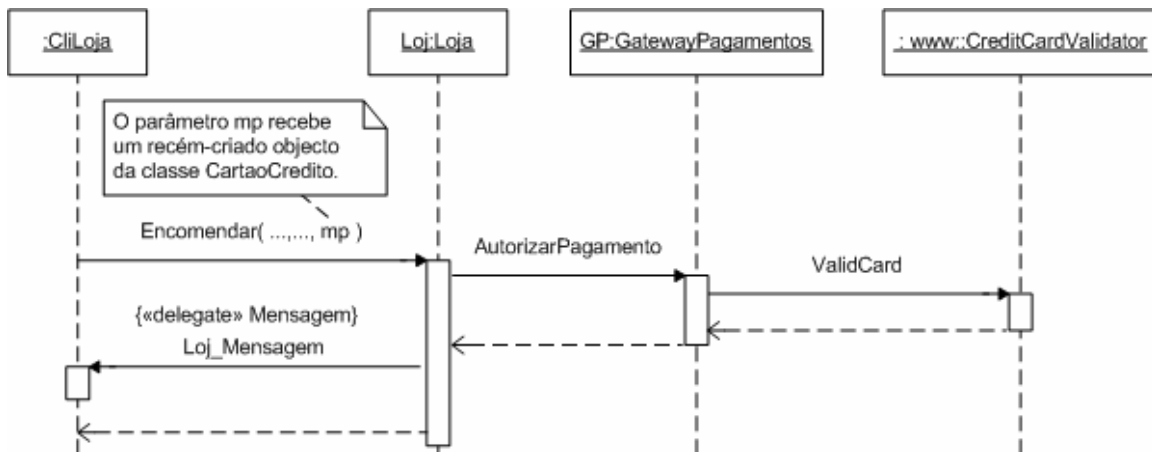


Fig 9: Diagrama de Sequência do encomendar por meio de cartão de crédito

Casos os dados do cartão de crédito sejam válidos, o pagamento é autorizado e a encomenda enviada para o cliente.

² "Strongly Typed Collection", obtida através da Herança da classe `CollectionBase`. Consultar artigo no endereço www.ftponline.com/vsm/2002_10/magazine/features/balena/default.aspx

³ pretendida no enunciado

⁴ overloading

As encomendas cujo meio de pagamento seja pagamento de serviços ou transferência bancária ficam pendentes até a Loja receber a notificação do seu pagamento ou até o seu prazo de pagamento ser ultrapassado.

A figura seguinte mostra o diagrama de seqüência de uma encomenda realizada com pagamento de serviços:

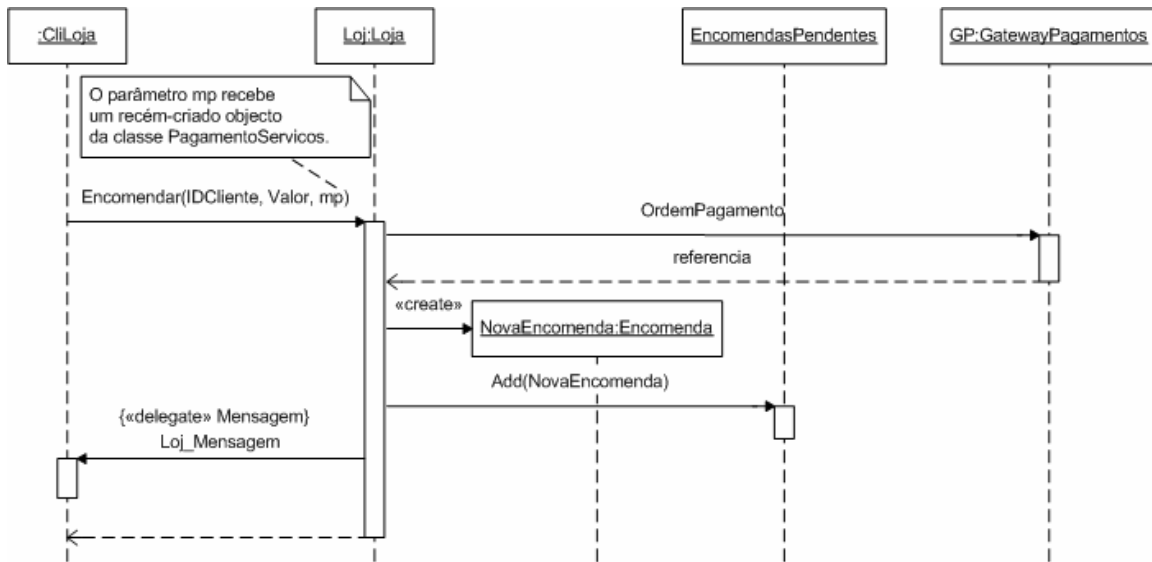


Fig 10: Diagrama de Sequência do encomendar com pagamento de serviços

A figura seguinte mostra o diagrama de seqüência do pagamento de uma encomenda através do pagamento de serviços:

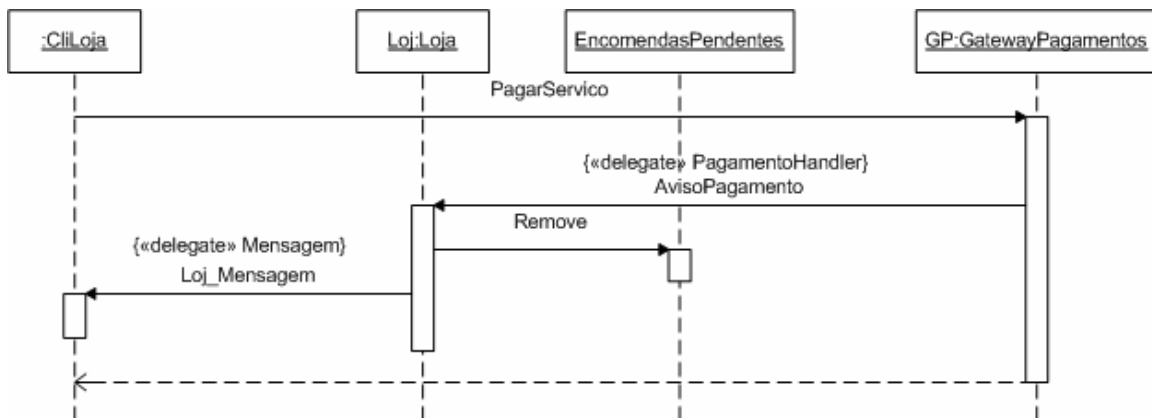


Fig 11: Pagamento de uma encomenda através do Pagamento de Serviços

No subsistema *Interface com o Utilizador* foram utilizadas algumas novidades:

- Control Splitter Bar
- Docking
- Validação da entrada dos dados com:
 - o Evento Validating
 - o Control ErrorProvider

