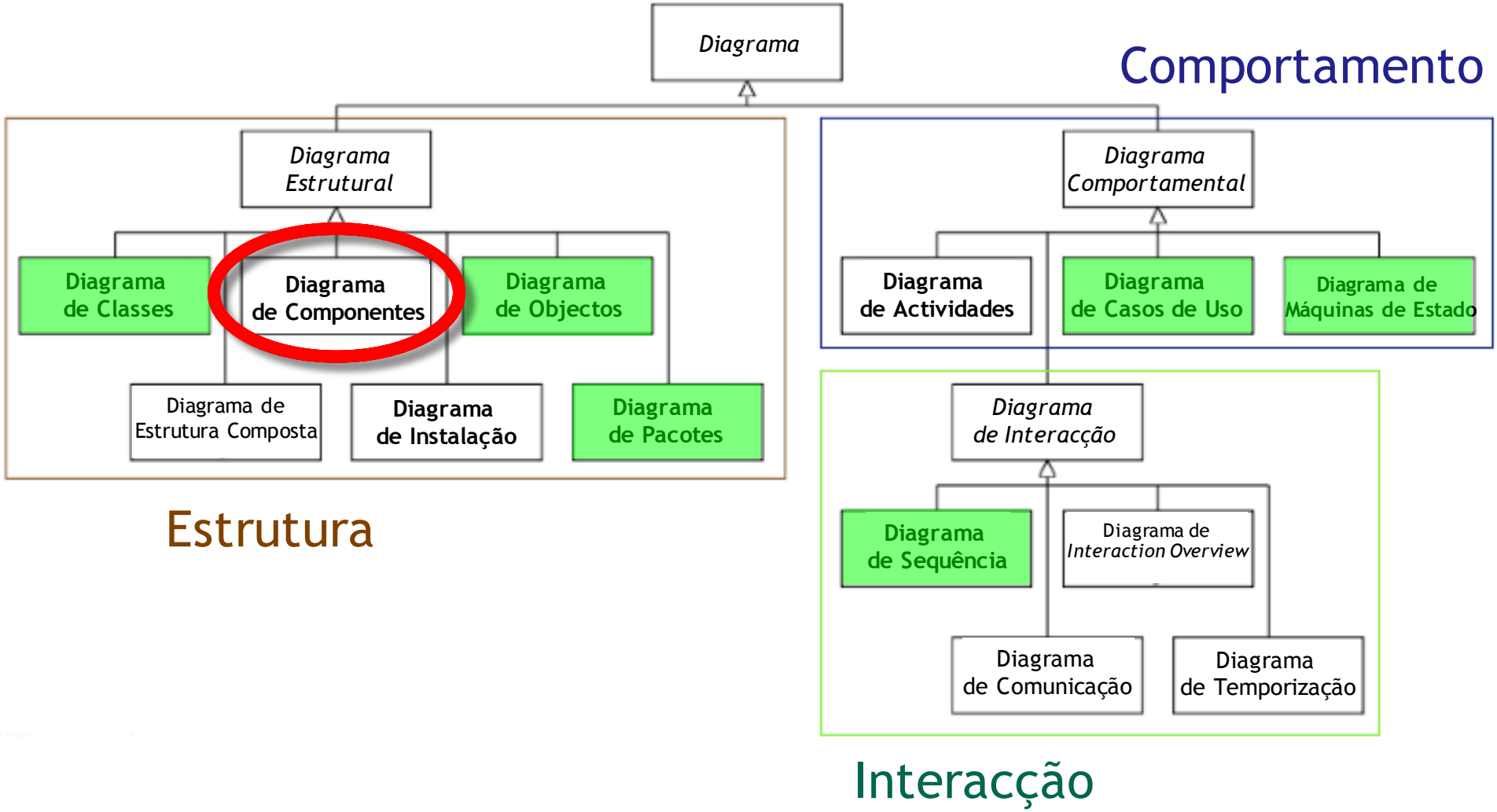
Desenvolvimento de Sistemas Software

Aula Teórica 20: Modelação Estrutural

Desenvolvimento de Sistemas Software
José Creissac Campos / António Nestor Ribeiro

Diagramas da UML 2.x



Diagramas de Componentes

- Como definir quais os componentes software do sistema?
  - Modelo em camadas?
  - Utilização de bibliotecas e serviços externos?
- Um Diagrama de Componentes descreve
  - Os componentes do sistema
  - As dependências entre eles
- Pode ser desenhado a diferentes níveis
  - código fonte
  - componentes binários (e.g. DLLs)
  - componentes executáveis
- Permite identificar, em cada nível, o que é necessário para construir o sistema

Diagrama de Componentes

- O que é um componente?
  - Um pedaço de software reutilizável, bem encapsulado e "*facilmente*" substituível.
  - São blocos (peças) que combinados constroem o sistema pretendido.
  - A dimensão dos componentes não é homogénea, existindo num mesmo sistema, componentes de diferentes dimensões.

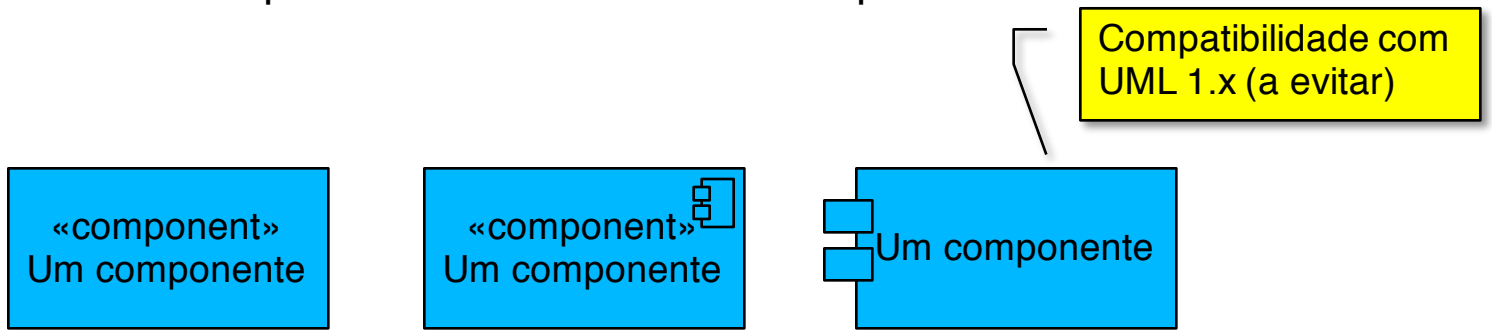
- Quais são os bons candidatos a serem componentes do sistema?
  - Items que desempenham uma funcionalidade que é utilizada recorrentemente no sistema
    - Exemplos: componentes de *logging*, parsers de XML, componentes de gestão de carrinhos de compra (*shopping carts*), etc.

- Em UML um componente pode efectuar as mesmas funcionalidades que uma classe faz
  - Generalização
  - Associação com outros componentes ou classes
  - Implementação de interfaces

- Um componente representa um empacotamento físico de elementos relacionados logicamente (normalmente classes)

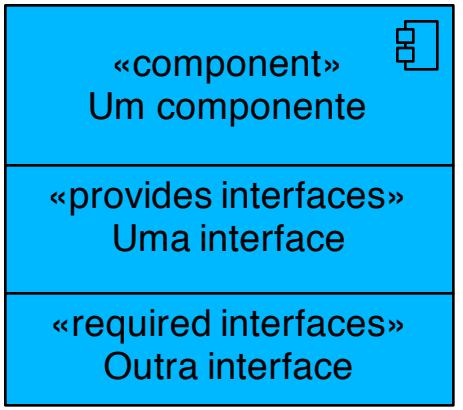
Diagramas de Componentes

- Componente
  - Uma parte modular do sistema
  - Comportamento definido pelas interfaces fornecidas/requiridas
- Notação

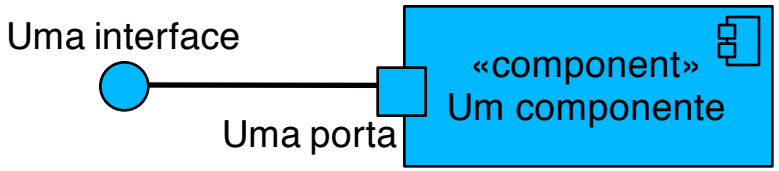


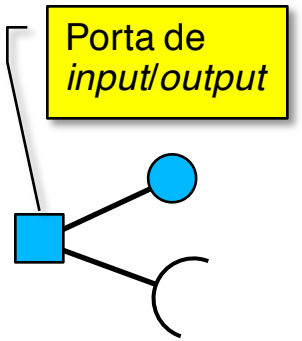
  - Alguns estereótipos de Componente:
    - «subsystem» - decomposição hierarquica do sistema global
    - «process» - componente transacional
    - «service» - componente funcional sem estado

Diagramas de Componentes



- Interfaces
  - Indicam os serviços requeridos / fornecidos pelo componente
- Portas (*ports*)
  - Identificam pontos de interacção com o componente
- Componente fornece (implementa) interface (porta de *output*)





- Componente requer (utiliza) interface (porta de *input*)

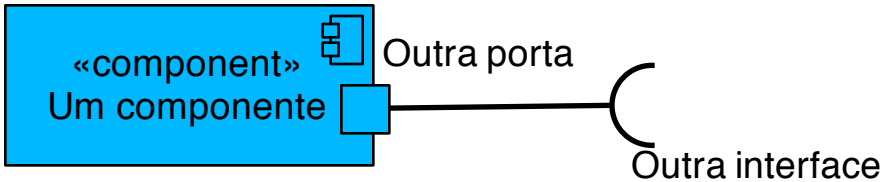
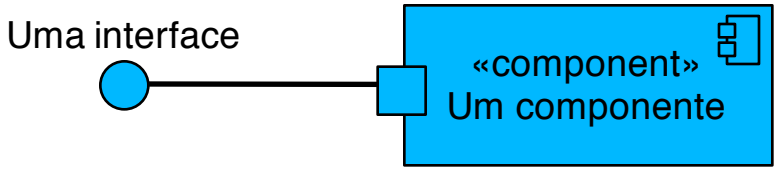
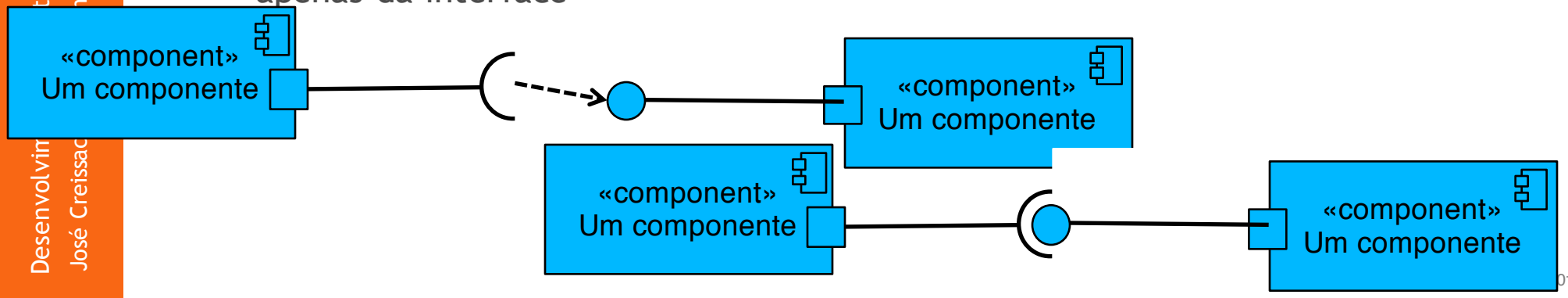


Diagrama de Componentes

- **Relação de concretização** (*realization*): um componente pode concretizar (implementar os serviços de) uma ou mais interfaces
  - Normalmente quer dizer que tem classes que implementam esses interfaces
  - Diz-se que as interfaces são exportadas
  - Um componente poder ser substituído por outro componente que implementa as mesmas interfaces

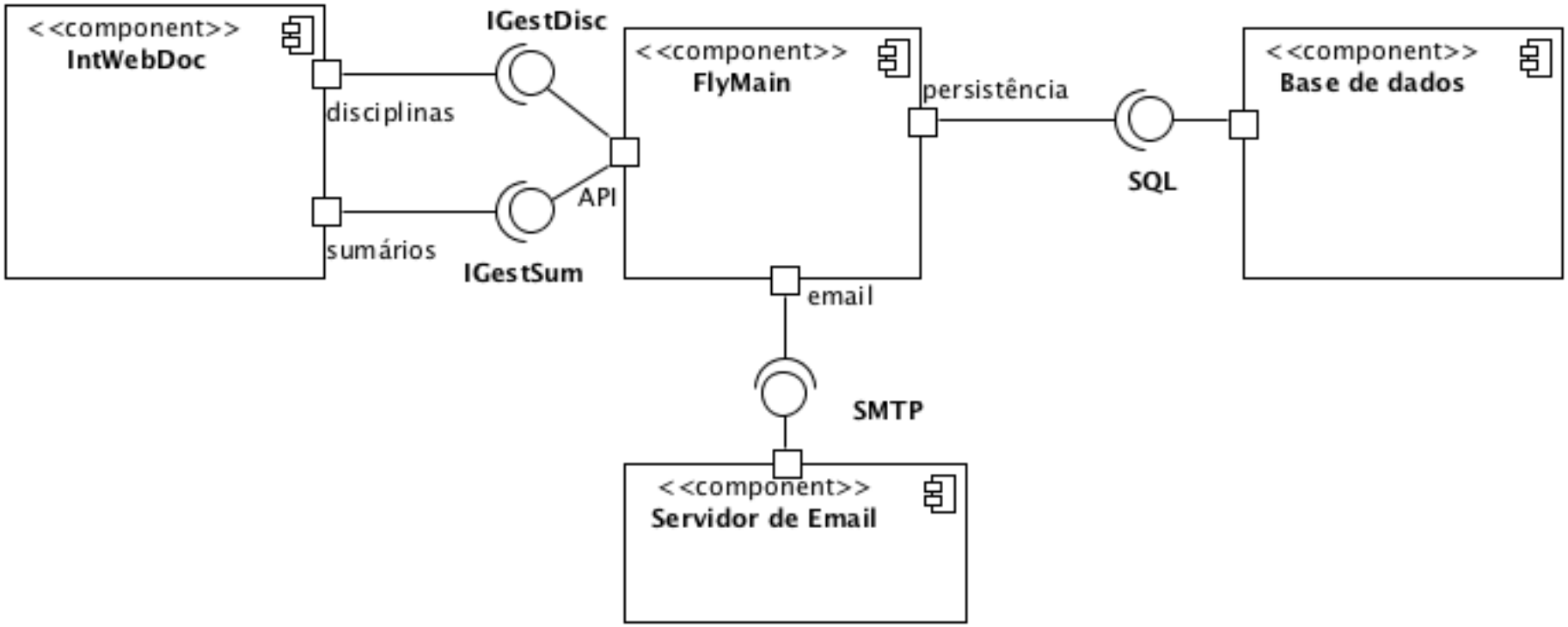


- **Relação de dependência**: um componente pode usar uma ou mais interfaces
  - Diz-se que essas interfaces são importadas
  - Um componente que usa outro componente através de uma interface bem definida, não deve depender da implementação (do componente em si), mas apenas da interface

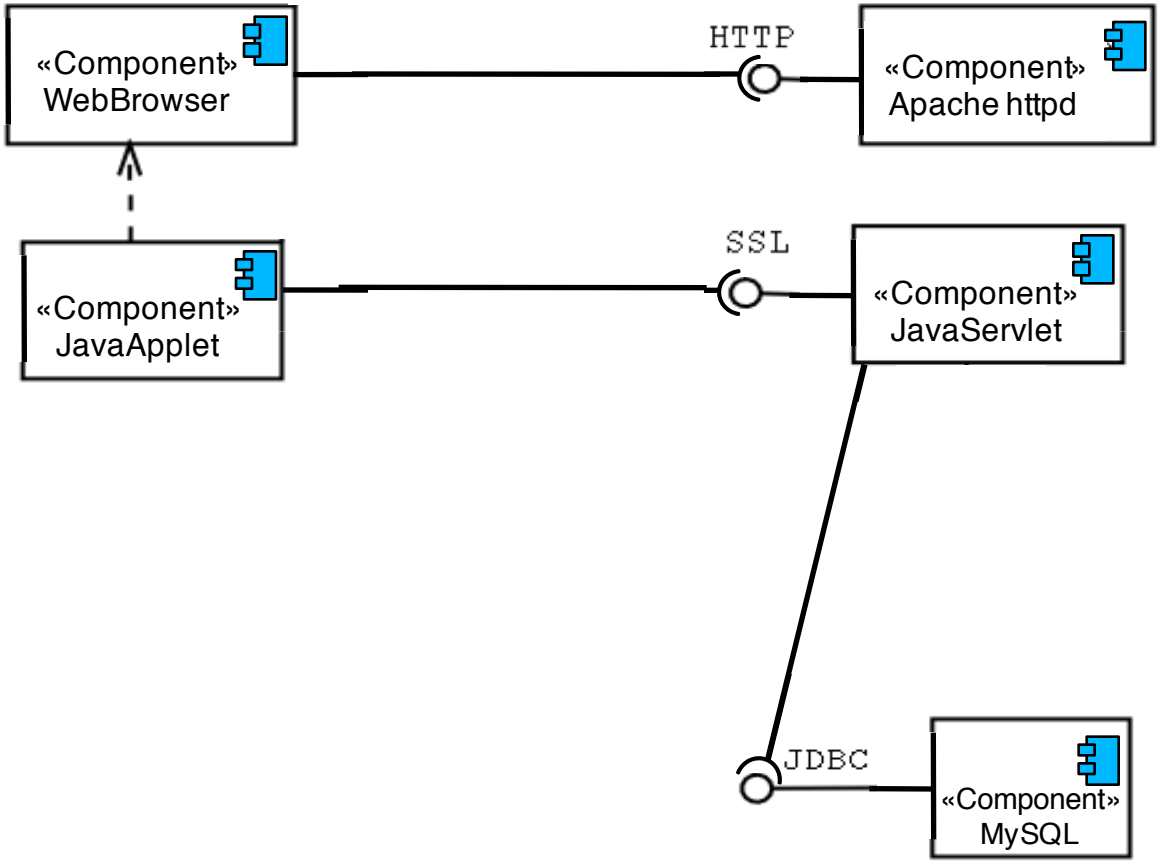


Diagramas de Componentes

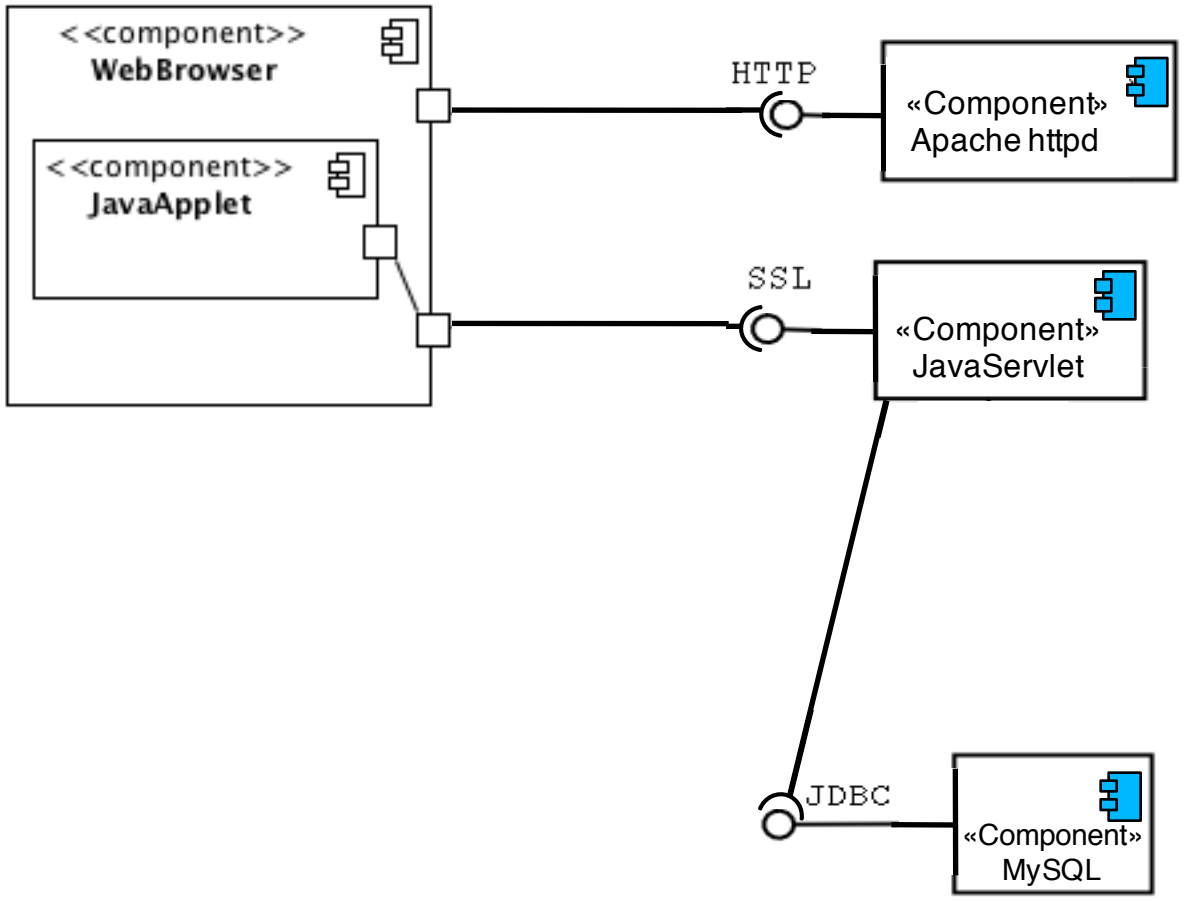
- Exemplo: 3 camadas?



Diagramas de Componentes

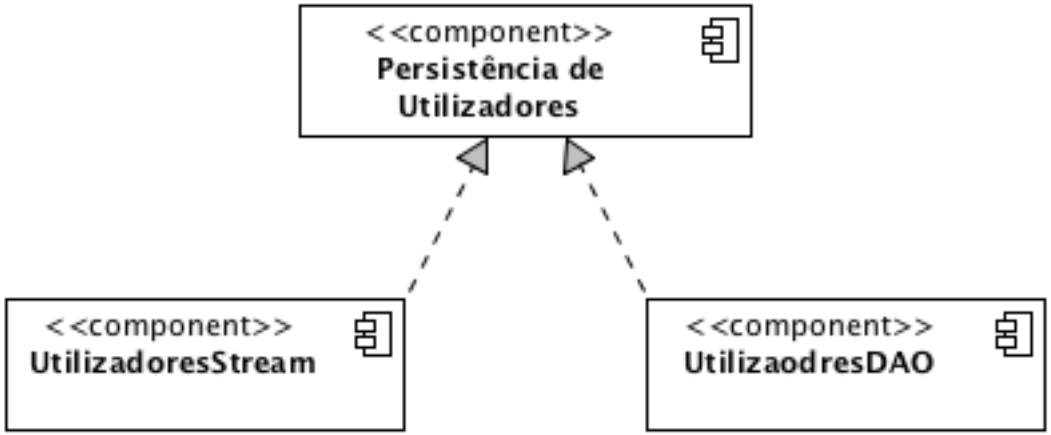


Diagramas de Componentes



Diagramas de Componentes

- Realização de componentes



Modelação Estrutural/Modelação Comportamental

Sumário

- Modelação Estrutural
  - Diagramas de Objectos
  - Diagramas de Componentes