Componentização e Reúso de Software

Fundamentos de Componentes e Design

Visão Externa

André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Junho de 2018

Problema

Complexidade do Hospital

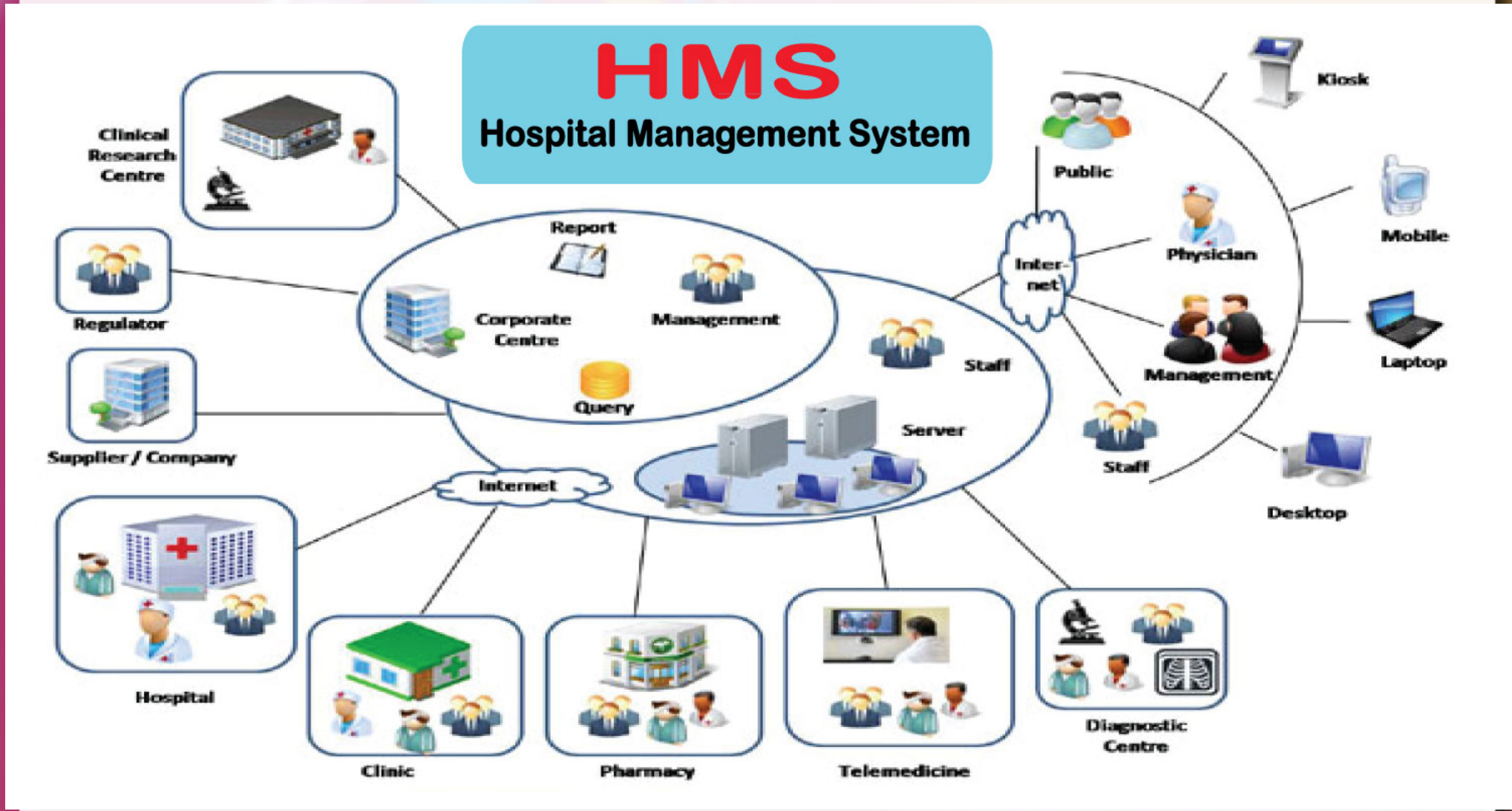
Hospital Tycoon
http://store.steampowered.com/app/11590/Hospital_Tycoon/

Hospital Havoc 2
https://itunes.apple.com/us/app/hospital-havoc-2/id437134954?mt=8

Exercício 1

- Elabore uma lista das funcionalidades, blocos ou módulos que deve ter um sistema de gerenciamento de hospital.

HMS



Innovanza Solutions
http://innovanza.co/hospital-management-system/

Quanta Modules

PATIENT REGISTRATION	WARD / ROOM MANAGEMENT	EMERGENCY/ CASUALTY	DOCTORS WORKBENCH	IPD PATIENT MODULE
LABORATORY INFORMATION SYSTEM	RADIOLOGY INFORMATION SYSTEM	OPERATION THEATER MANAGEMENT	CERTIFICATES ISSUE	ROSTER MODULE
COMMUNICATION MODULE	INVENTORY & FIXED ASSET	MEDICAL STORE MANAGEMENT	WARD PHARMACY/ INVENTORY MANAGEMENT	PHYSIOTHERAPY MODULE
DIET MODULE	HOUSE KEEPING MODULE	PAY ROLL MODULE	FINANCIAL ACCOUNTING MODULE	DENTAL MODULE
BLOOD BANK	MEDICAL RESEARCH	ICD – 10 DIAGNOSIS MODULE	DONATION MODULE	BILLING MODULE
AMBULANCE MODULE	USER MANAGEMENT	ADDRESS DIRECTORY	BACKUP/ RESTORE MODULE	MIS REPORTS
EQUIPMENT MAINTENANCE	CENTRAL STERILIZED SUPPLY DEPARTMENT	ENERGY AUDIT	COSTING	LIBRARY MANAGEMENT
BUDGETS	HELP DESK	FACILITY MANAGEMENT	WASTE MANAGEMENT	PACS & TELE RADIOLOGY
KITCHEN	EMR MODULE			

www.birlamedisoft.com

https://www.slideshare.net/birlam/ppt-hospital-management-system-quantahis

Dilema

Fazer Tudo x Comprar Tudo

- Extremos no desenvolvimento de software tradicional:
  - Desenvolver projeto da estaca zero
  - Comprar sistema pronto (configurado)

(Szyperski, 2002)

Implementar Tudo

- Vantagens:
  - se adapta as necessidades do usuário
  - explora conhecimentos e práticas domésticas
  - diferencial ➔ vantagem competitiva

(Szyperski, 2002)

Implementar Tudo

- Desvantagens:
  - caro
  - soluções sob medida geralmente são localizadas
  - difícil de acompanhar o estado da arte (ex.: acesso Web)
  - barreiras de interoperabilidade
  - pode chegar “muito tarde”

(Szyperski, 2002)

Comprar Tudo

- Vantagens
  - custo pode ser pré-contratado
  - software pré-fabricado diminui tempo de implantação
  - estado da arte e interoperabilidade ➔ tarefa de quem vende

(Szyperski, 2002)

Comprar Tudo

- Desvantagens
  - adaptação dos negócios ao software
  - sem diferencial ➔ sem vantagem competitiva
  - não se adapta rapidamente a novas necessidades

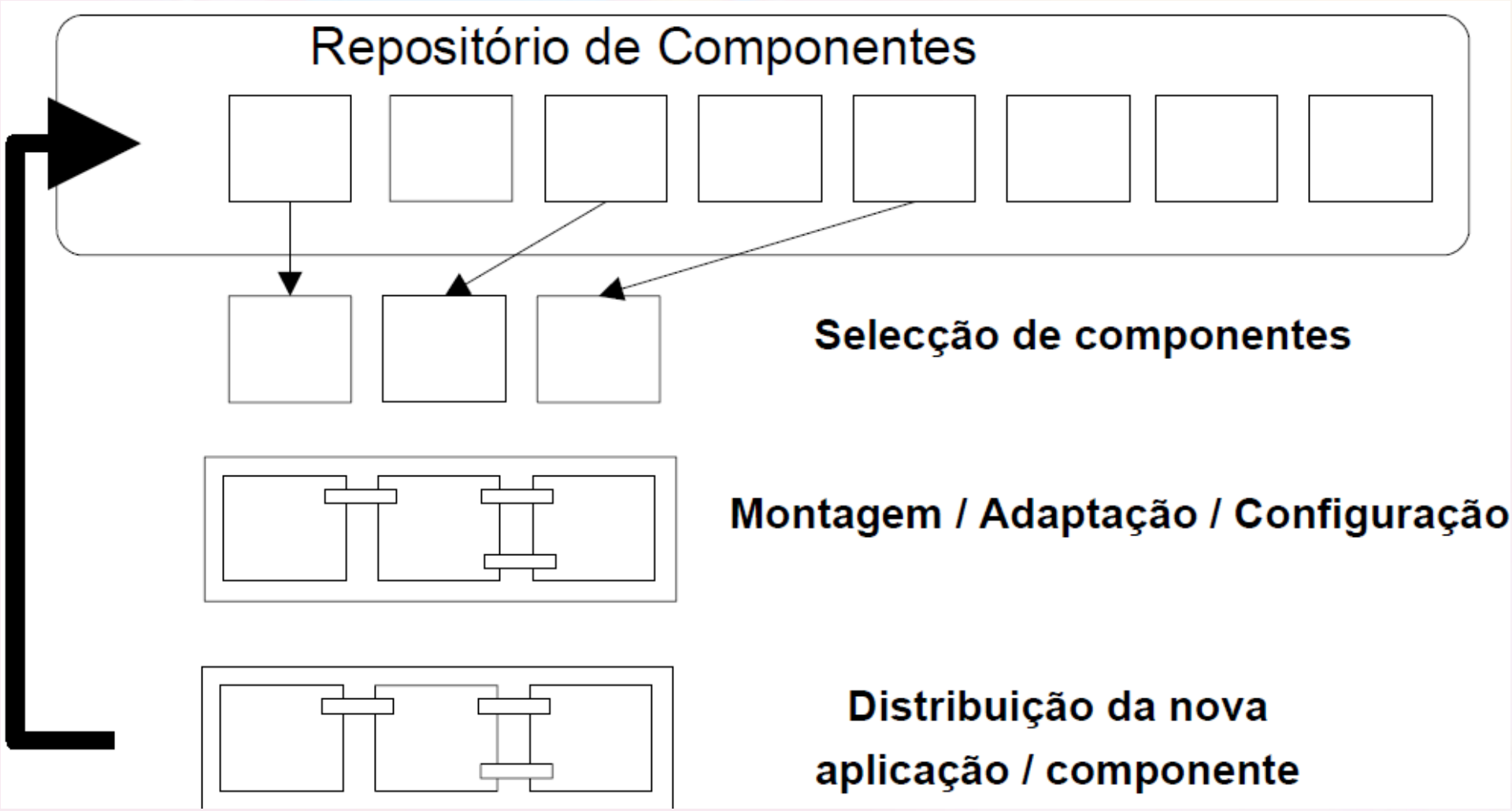
(Szyperski, 2002)

Component Software
Via Intermediária

- Component Software (Software de Componentes)
  - sistema feito de componentes de software
- "O conceito de *component* software representa uma via intermediária que pode resolver este problema." (Szyperski, 2002)

Tradução do original feita pelo autor: "The concept of component software represents a middle path that could solve this problem." (Szyperski, 2002)

Programação por Componentes (Composição)



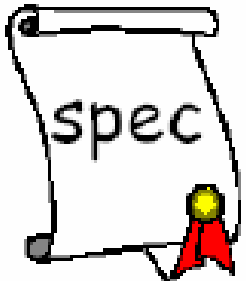
(Caires, 2002)

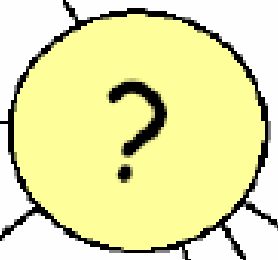
O que é um componente?

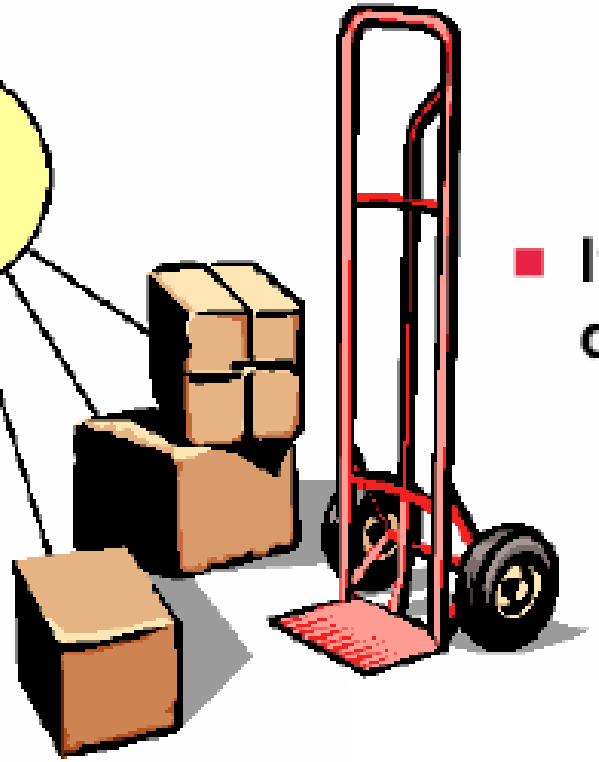
(Cheesman & Daniels, 2000)

- It has a specification
- It has an implementation

```
for (int i=0;
i<limit; i++)
{ list[i] = …
}; ……
```







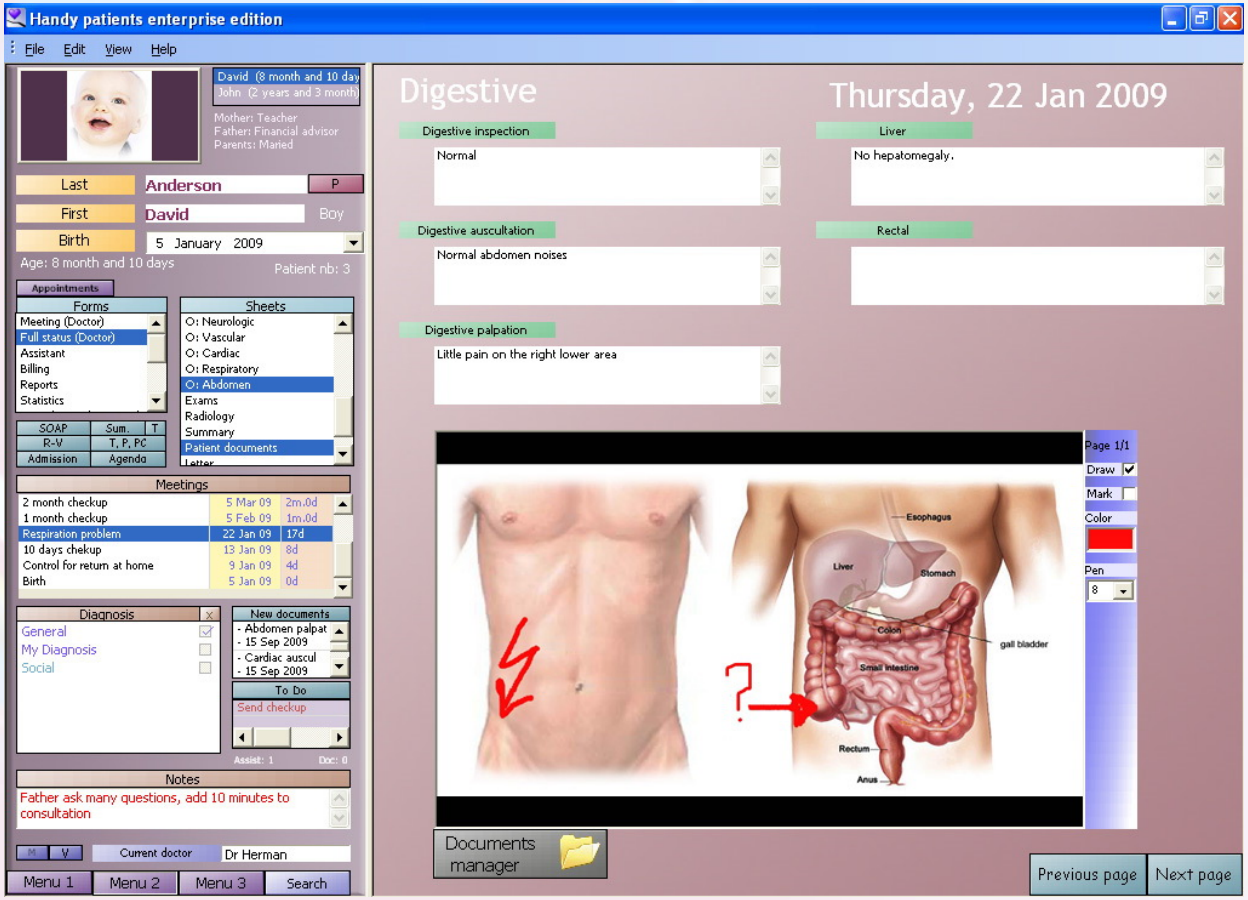
- It can be deployed
- It conforms to a standard

- It can be packaged into modules

[Chesman & Daniels]

Desafio

- Montar componentes para suporte à decisão em um Prontuário Eletrônico.



By Oguntoye patients electronic medical record (free open source version), GPL, https://commons.wikimedia.org/w/index.php?curid=8894074

Componente em UML

- "[…] sistemas de software de tamanho e complexidade arbitrários."[1]
- Componente[2]:
  - unidade modular
  - com interfaces bem definidas
  - substituível dentro do ambiente

---

1. "[…] software systems of arbitrary size and complexity" (Cook, 2015)
2. "[…] Component as a modular unit with well-defined Interfaces that is replaceable within its environment." (Cook, 2015)

Especificação UML

- **OMG Unified Modeling Language (OMG UML) - version 2.5.1**
  OMG
  (2017)
  https://www.omg.org/spec/UML/2.5.1/
- Seção 11.6 - Components

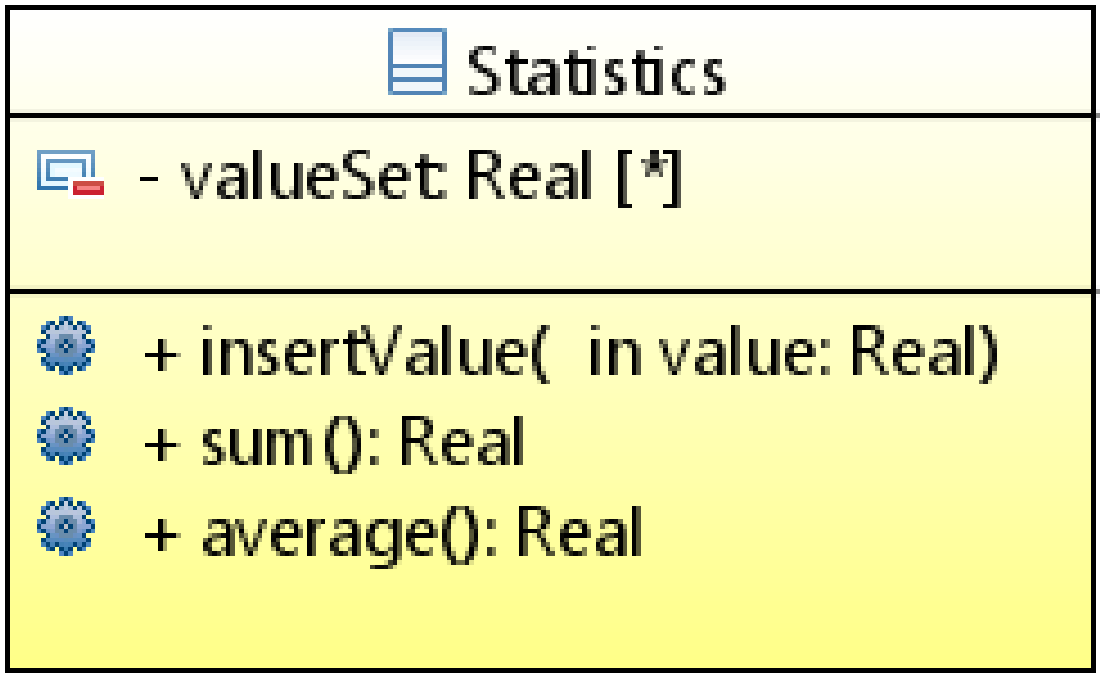
Componente Estatístico

Componente Estatístico

Objetivo

- Registrar um conjunto de números e calcular a soma e média destes números.

Passo 1: Classe Estatística

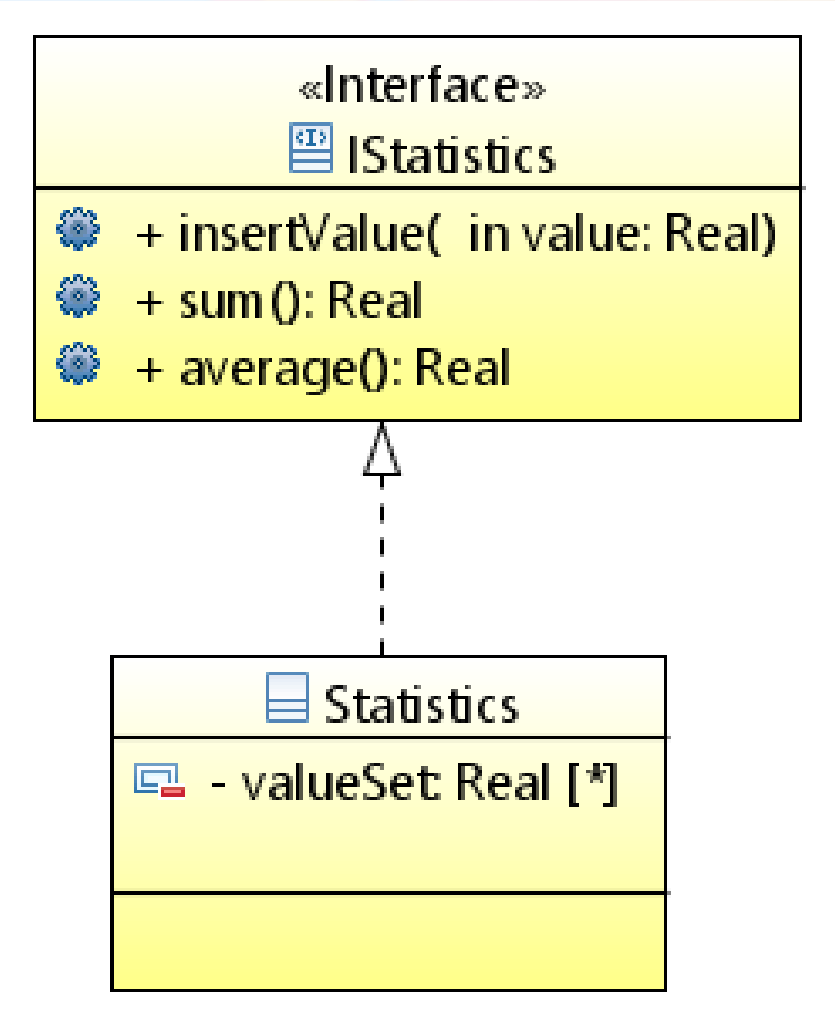


`pt.c08componentes.s10statistics.s01class`

Dependency Inversion Principle (DIP)

- “Depender das Abstrações. Não depender das Concretizações.” (Martin, 2000)

Passo 2: Interface Estatística

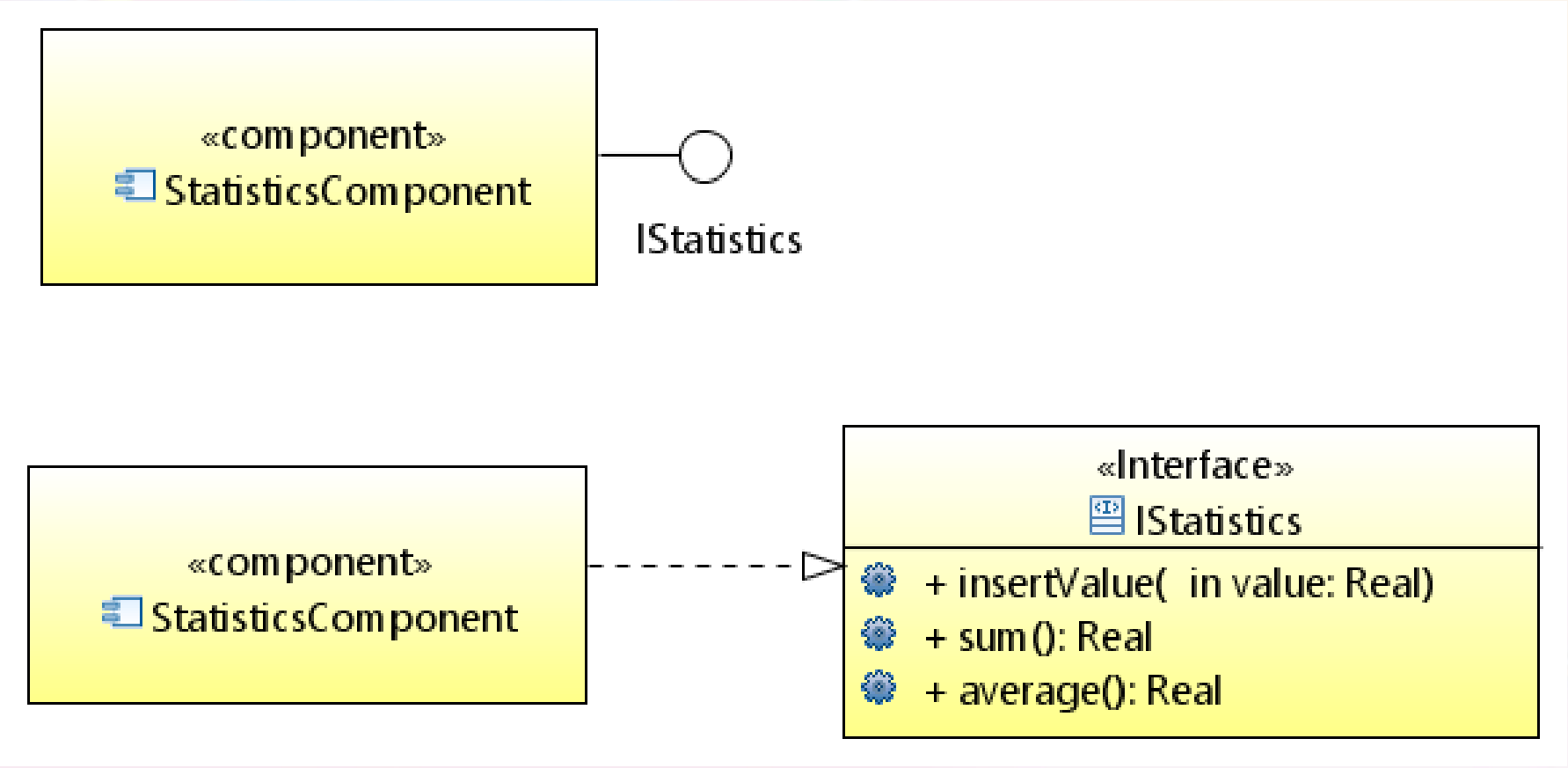


`pt.c08componentes.s10statistics.s02interface`

Passo 3: Transformando em um Componente

Notação Blackbox

- Esconde os detalhes internos da representação

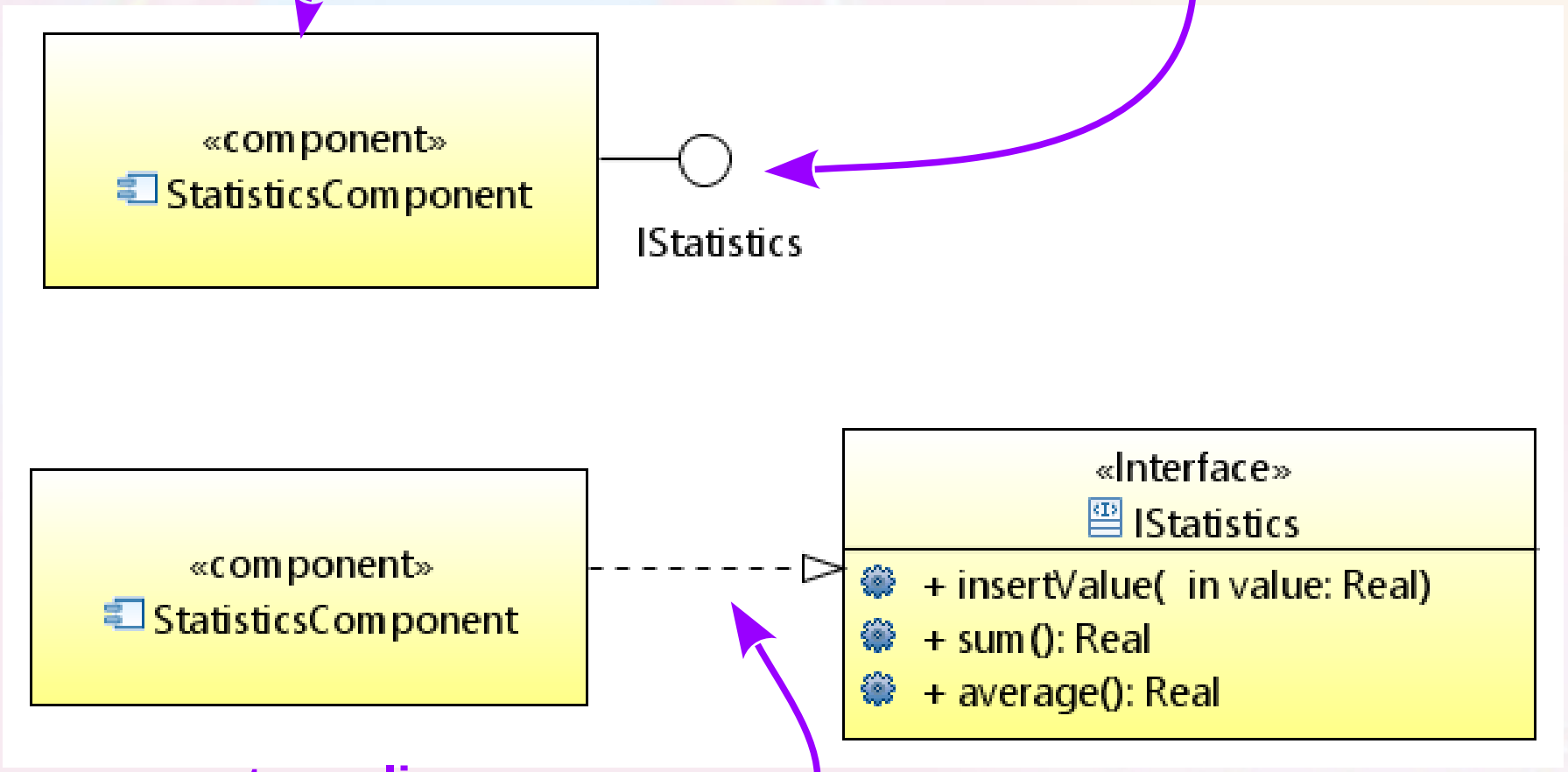


pt.c08componentes.s10statistics.s03component

Notação Blackbox

componente em notação blackbox

componente realiza interface (representação lollipop)

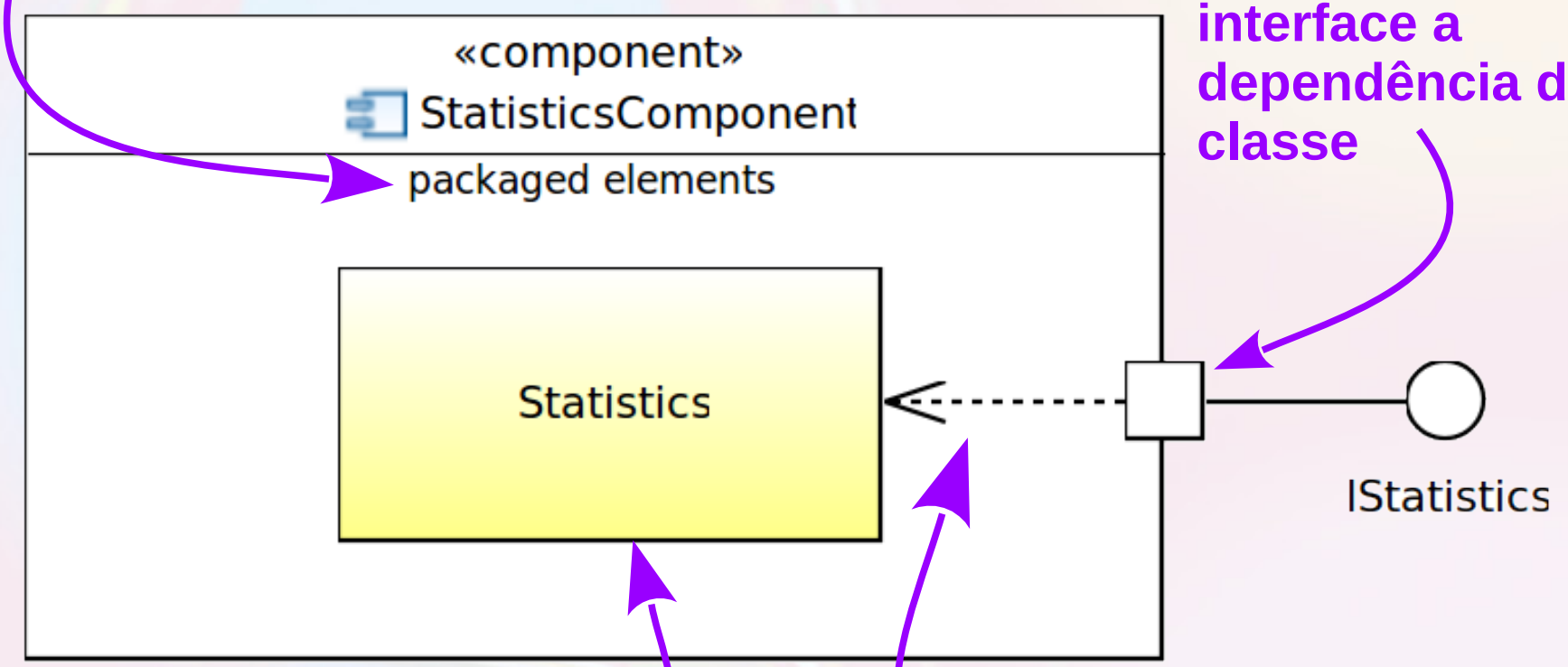


componente realiza interface (representação explícita)

Realizando o Componente

compartimento opcional que mostra elementos que são parte do componente

porta que associa interface a dependência de classe

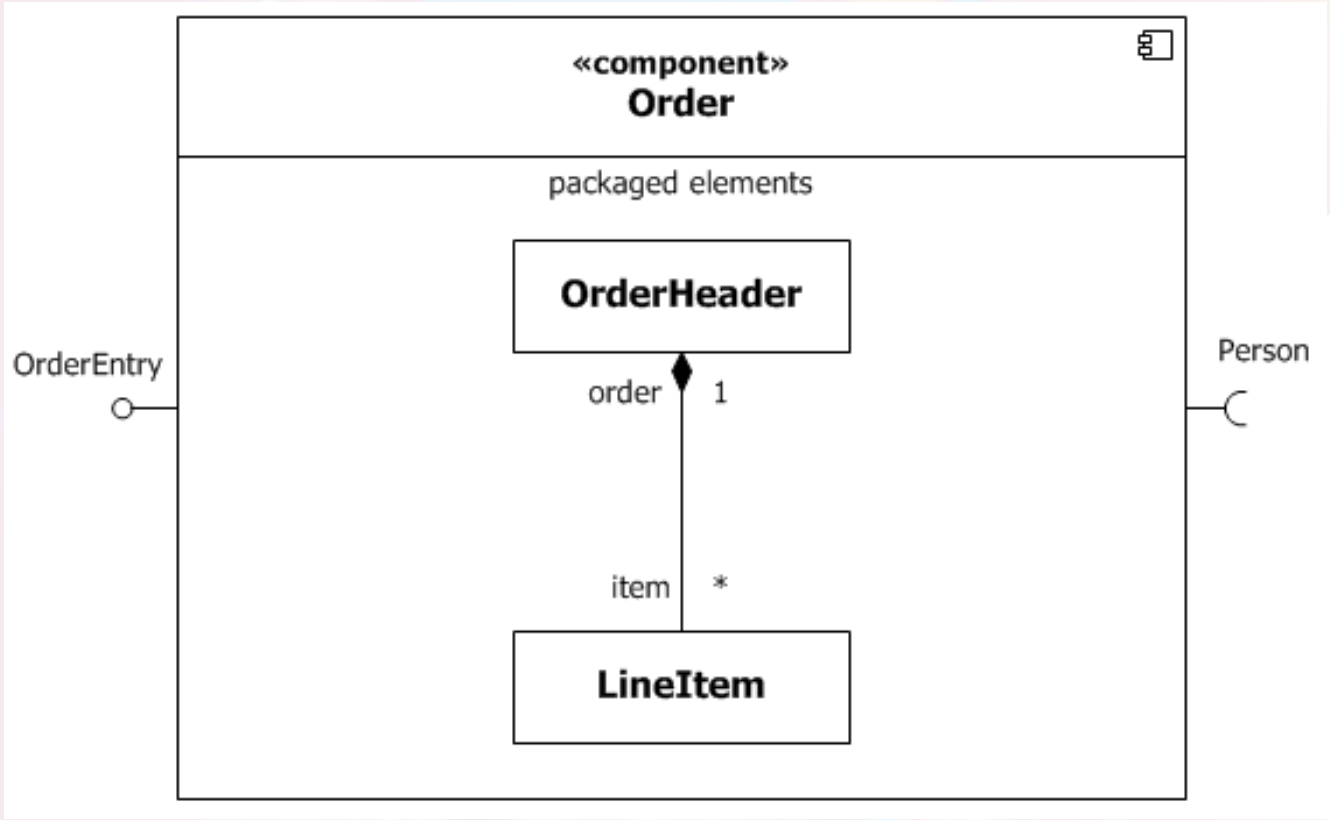


classe que realiza o componente

dependência da porta/interface com a classe que a realiza

`pt.c08componentes.s10statistics.s03component`

Duas perspectivas de Whitebox
OMG UML (Cook et al., 2015)



«component»
Order

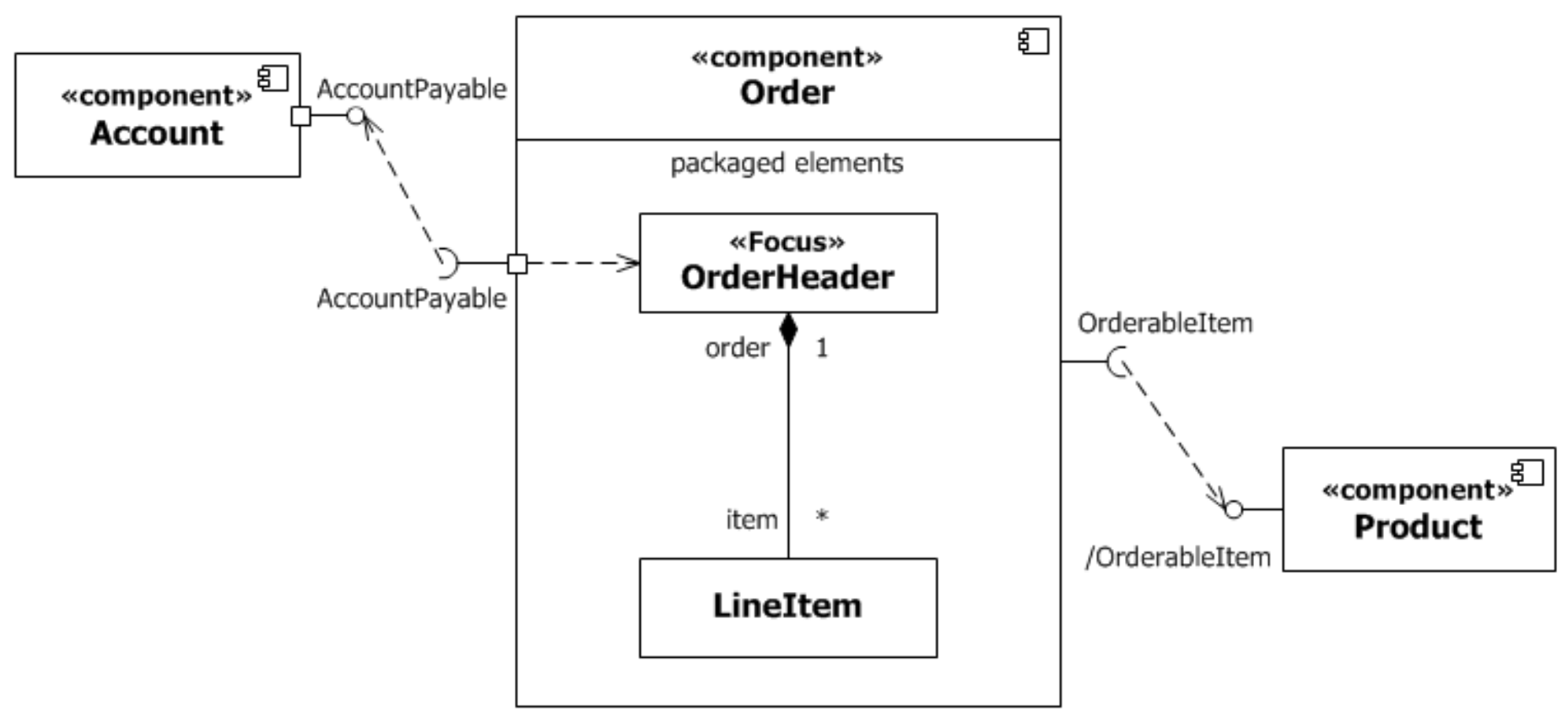
provided interfaces
ItemAllocation
Tracking

required interfaces
Person
Invoice
OrderableItem

realizations
OrderHeader
LineItem

artifacts
Order.jar

Whitebox com portas
OMG UML (Cook et al., 2015)



Visão Externa

- Visão Externa (esta aula)
  - Foco: blackbox
  - Abstração das funcionalidades de um componente vendo-o externamente através de suas interfaces
  - Uso de componentes → Composição
- Visão Interna (próxima aula)
  - Foco: whitebox
  - Como um componente é implementado internamente

Componente JavaBean

- Componentes são unidades de software auto-contidas e reusáveis que podem ser compostas visualmente em componentes compostos, applets, aplicações, e servlets usando ferramentas visuais de construção de aplicações.” (Sun, 2006)

Tradução do Inglês: “Components are self-contained, reusable software units that can be visually assembled into composite components, applets, applications, and servlets using visual application builder tools.” (Sun, 2006)

JavaBeans

- Beans – componentes em Java

Características:

  - Construtor sem argumentos
  - Propriedades
  - Introspecção
  - Customização
  - Persistência
  - Eventos

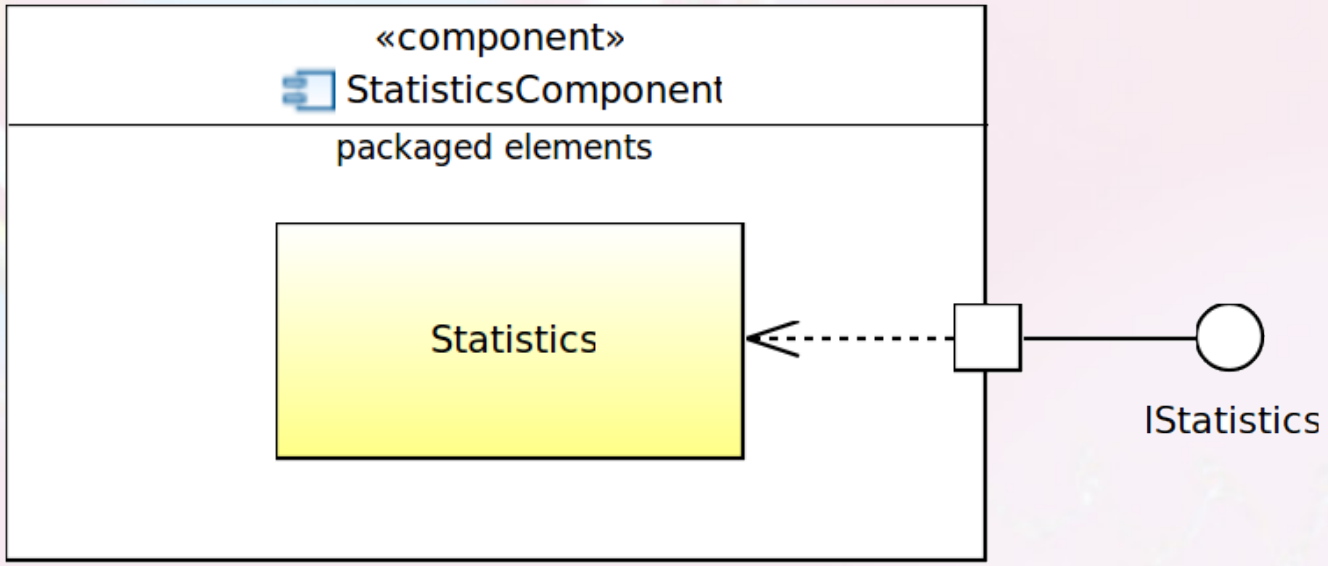
Perspectiva Orientada a Objetos de Componentes

- Componentes são associados a classes
  - São instanciados como objetos
    - Não é um consenso
- Propriedades externamente observáveis
  - Customizam a instância do componente
    - Não é um consenso

Construtor sem Argumentos

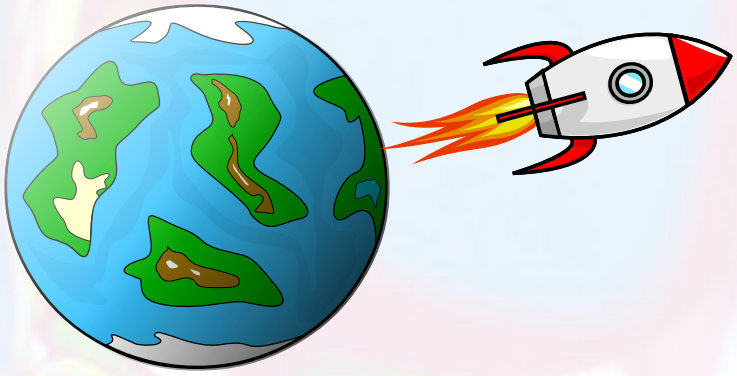
- Permite a criação automática do componente
- Construtor com ação padrão

```
IStatistics stat = new Statistics();
```

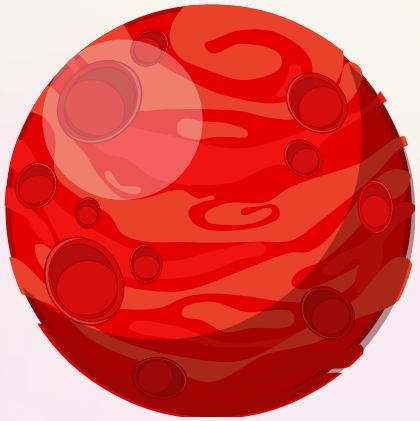


`pt.c08componentes.s10statistics.s03component.v01`

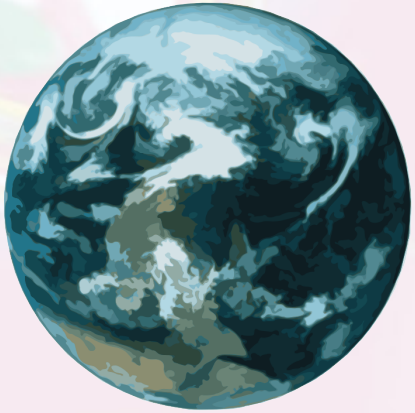
Exercício 2
Mercante Interplanetário



Pindora



Bantor



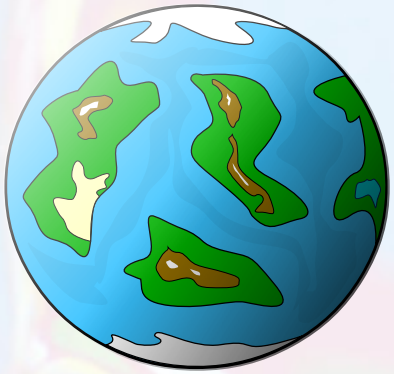
Zeta

Exercício 2

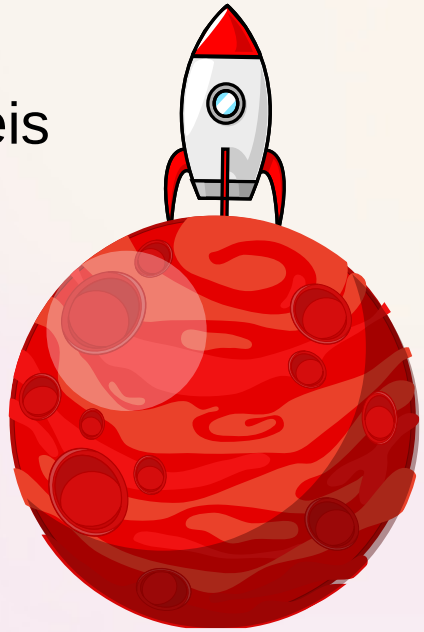
Componente Mercado Planeta

Operações:

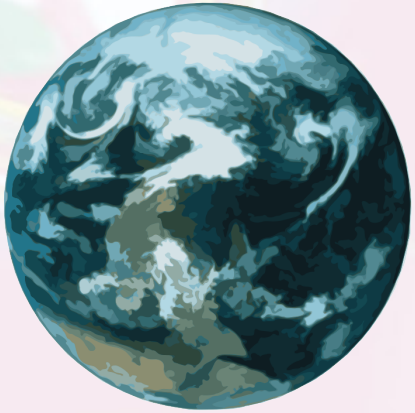
- Verificar mercadorias disponíveis
- Comprar mercadorias
- Vender mercadorias



Pindora



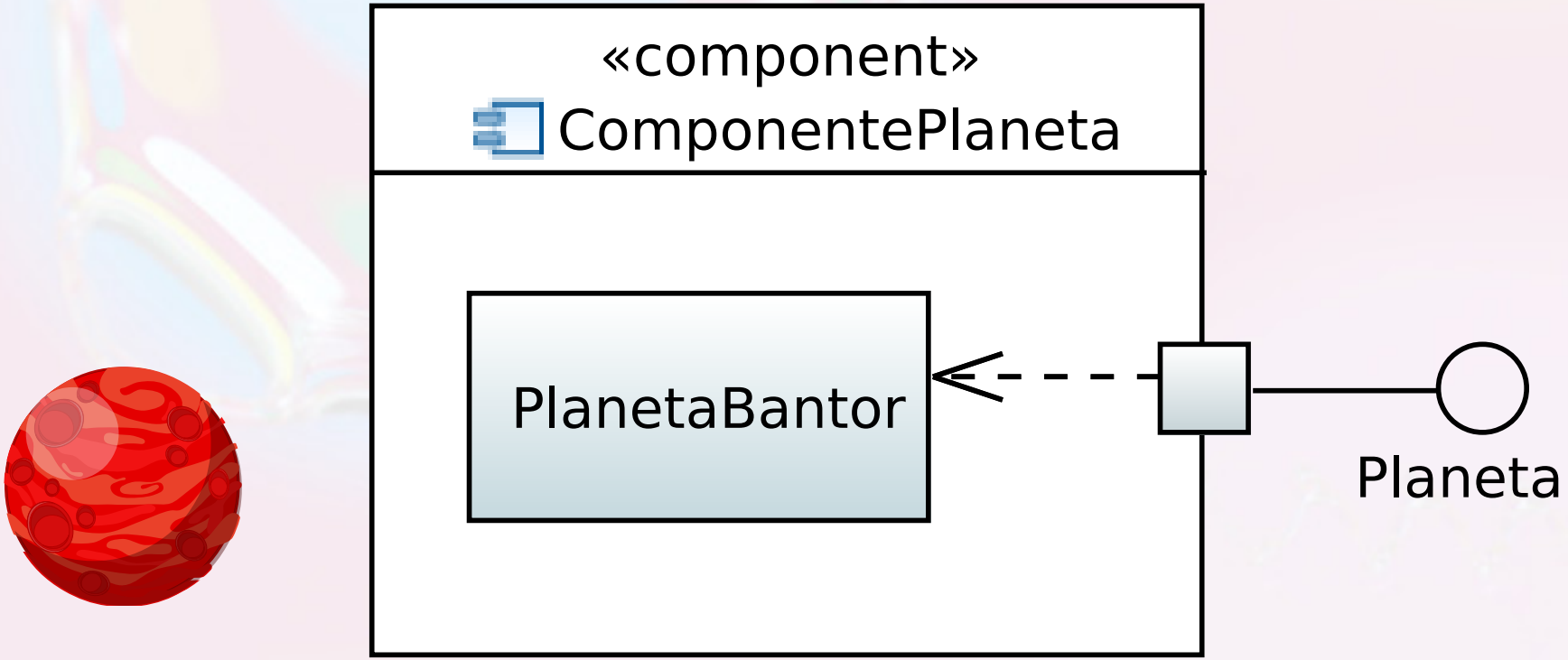
Bantor



Zeta

Exercício 2

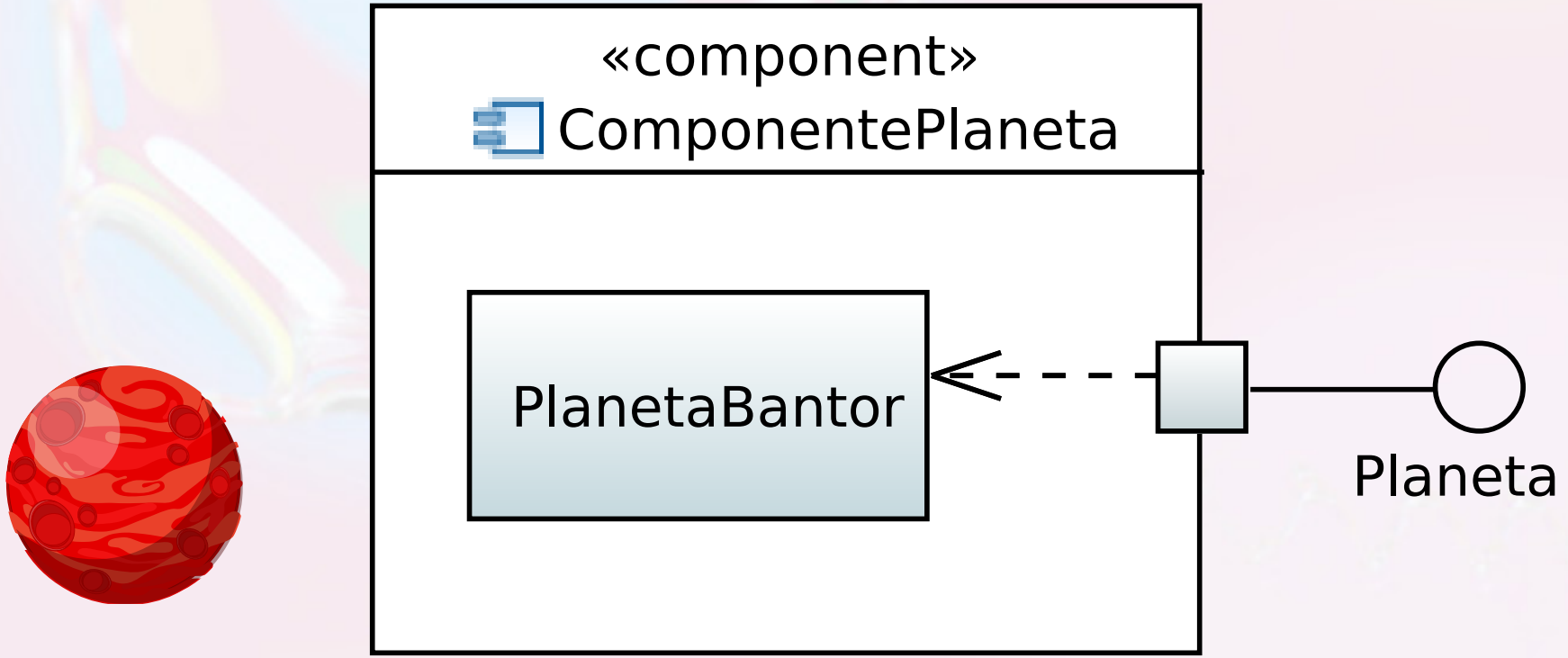
- Instancie o componente a seguir e atribua a uma variável.



Exercício 2

- Instancie o componente a seguir e atribua a uma variável:

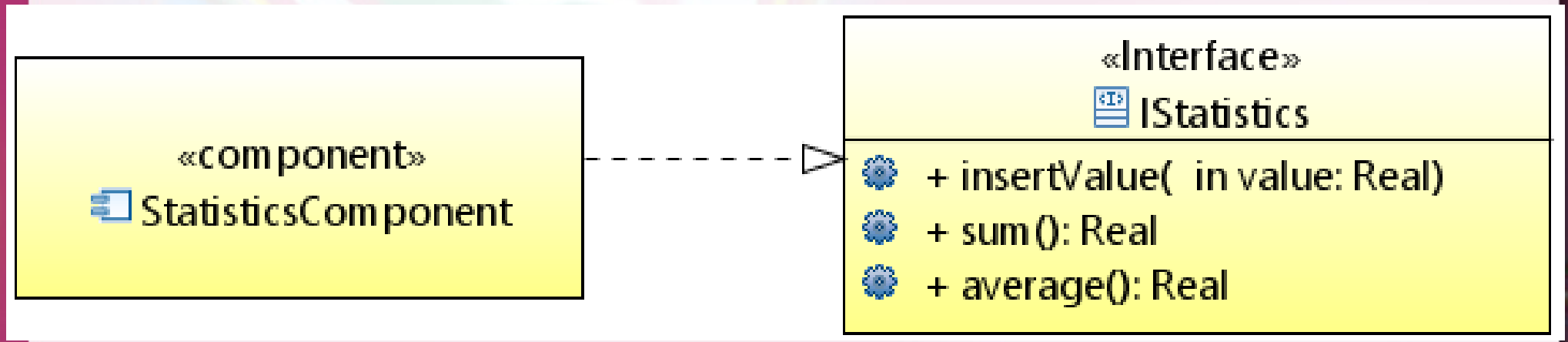
```
Planeta p = new PlanetaBantor();
```



Usando Serviços do Componente

Acessando os Serviços pela Interface

```
stat.insertValue(50.0f);
stat.insertValue(70.0f);
stat.insertValue(30.0f);
System.out.println("somatorio: " + stat.sum());
System.out.println("media: " + stat.average());
```



Exercício 2.5

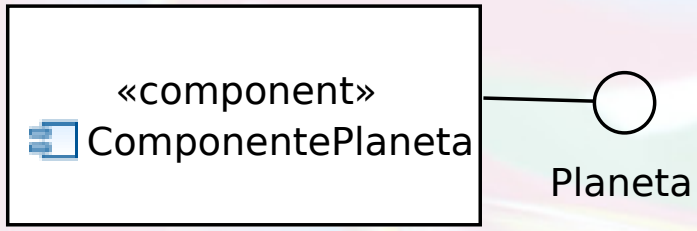
- Escreva duas linhas de código para listar todas as mercadorias e verificar o preço de compra da mercadoria “Pistola Xist Paralisante”.

Planeta

Operações:
- Verificar mercadorias disponíveis
- Verificar preço de compra de uma mercadoria
- Verificar preço de venda de uma mercadoria

Exercício 2.5

- Escreva duas linhas de código para listar todas as mercadorias e verificar o preço de compra da mercadoria “Pistola Xist Paralisante”.
- System.out.println(p.listaMercadorias());
  System.out.println(p.precoCompraMercadoria(“Pistola Xist ...”));



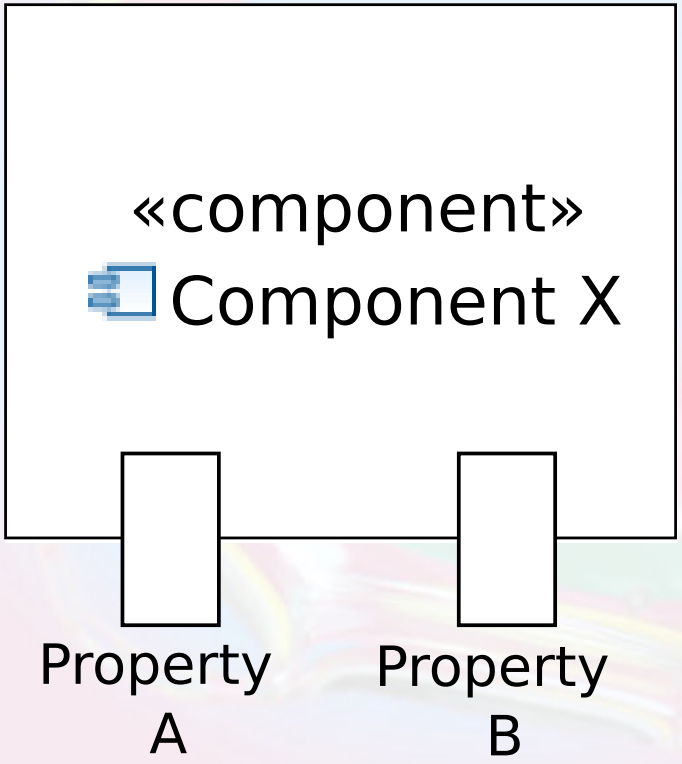
Operações:
- Verificar mercadorias disponíveis
- Verificar preço de compra de uma mercadoria
- Verificar preço de venda de uma mercadoria

Propriedades

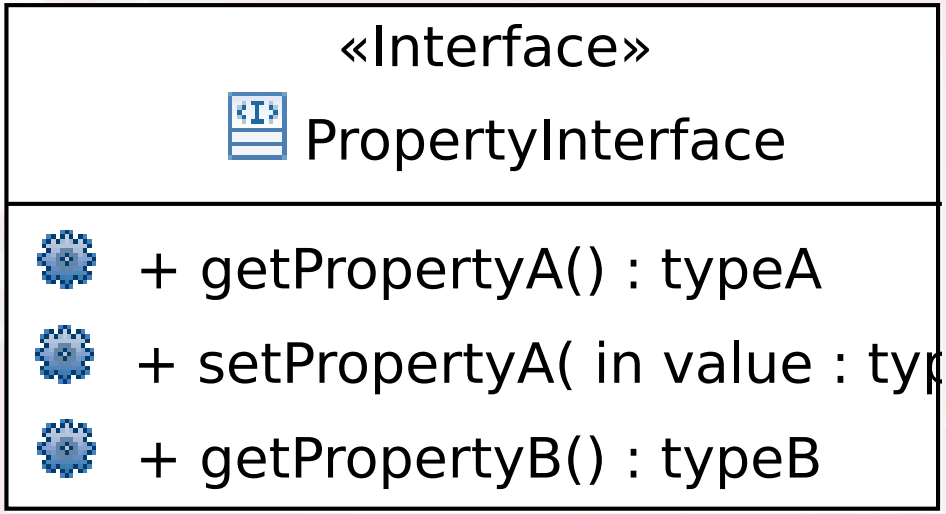
Propriedades

- Campos com valores que podem ser consultados externamente e eventualmente modificados.
- Permitem a customização externa do componente.
- Ligadas ao princípio de instância de componente e componente *stateful*.

Propriedades
Notação CORBA Component Model



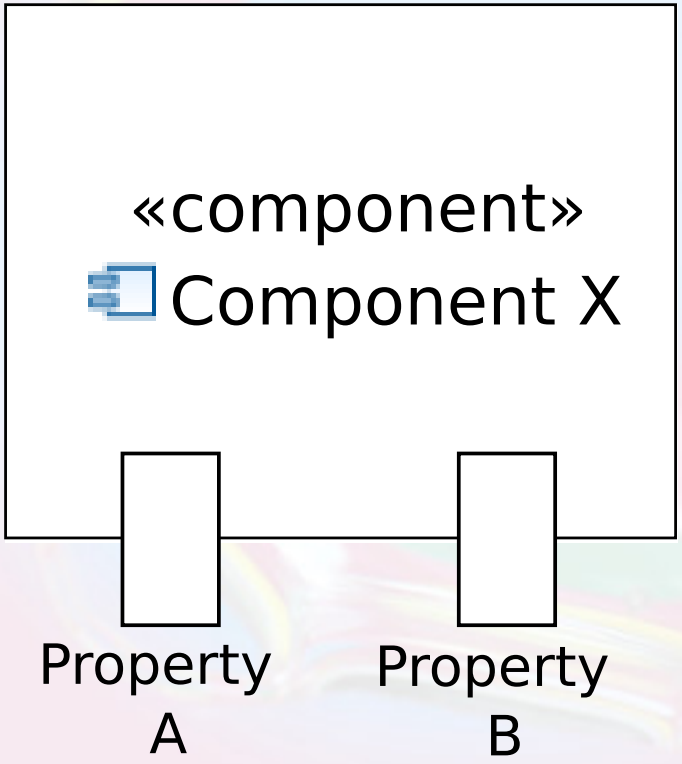
PropertyB é somente leitura



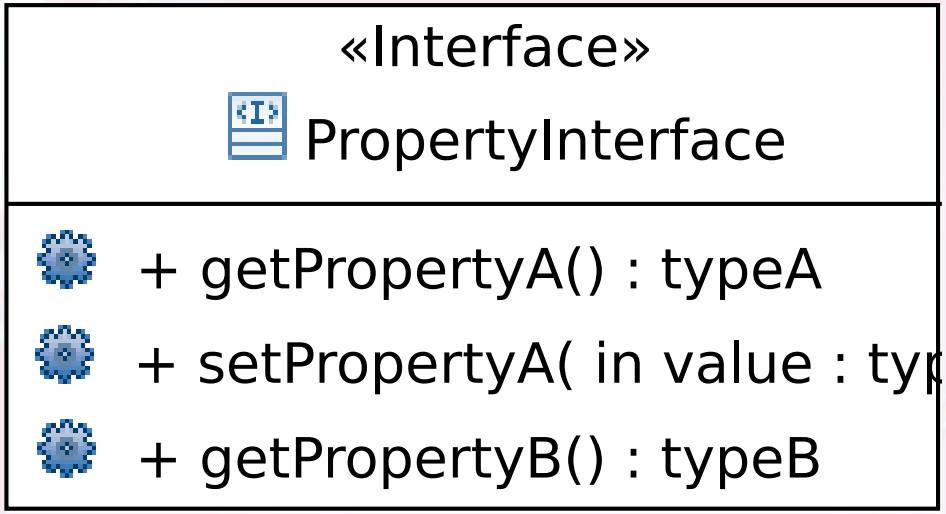
Propriedades em Java

- Expostas através de métodos:
  - prefixo “get” → leitura
  - prefixo “set” → modificação
- Somente leitura
  - não têm método “set”

Propriedades
Notação CORBA Component Model

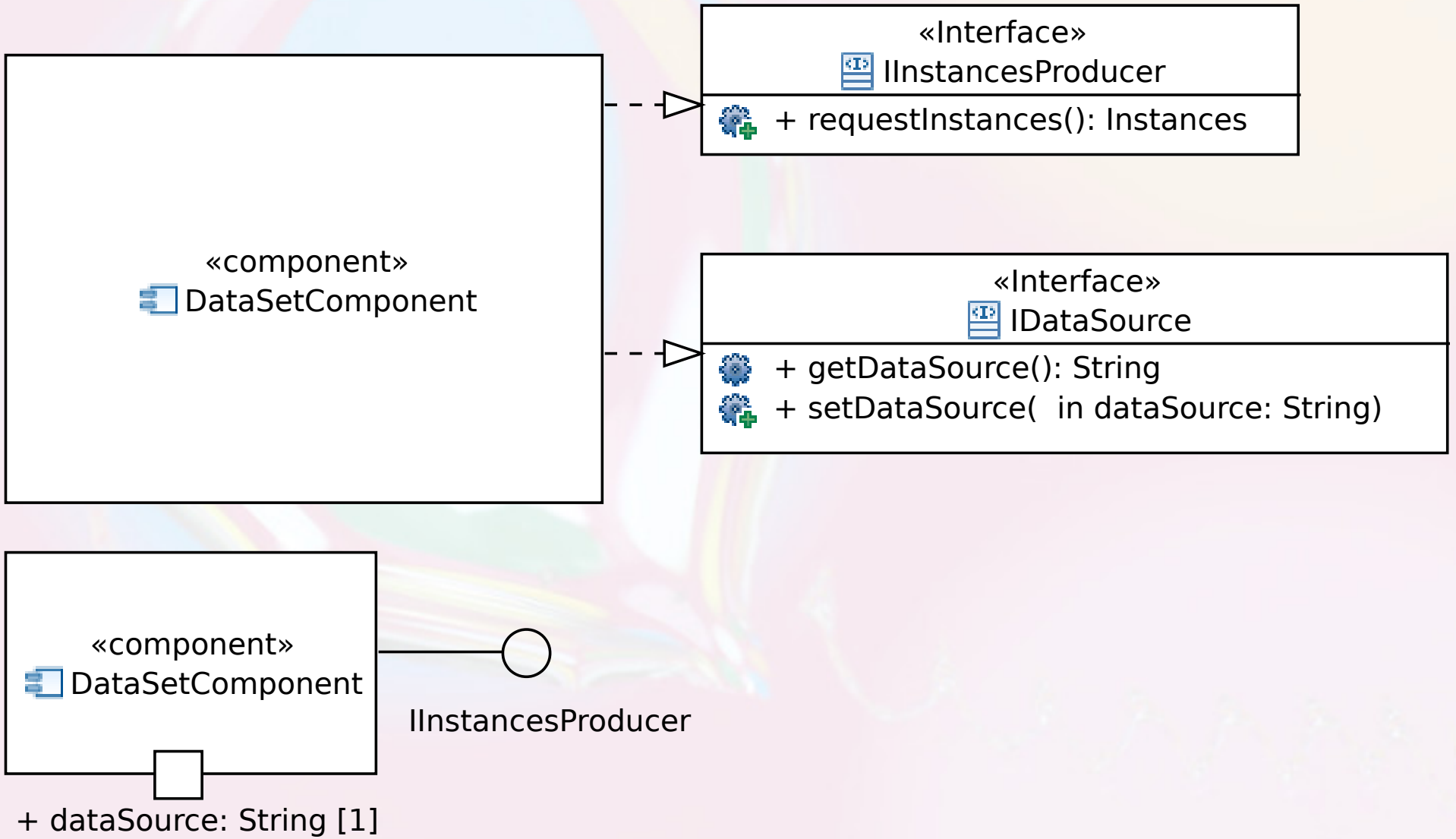


PropertyB é somente leitura

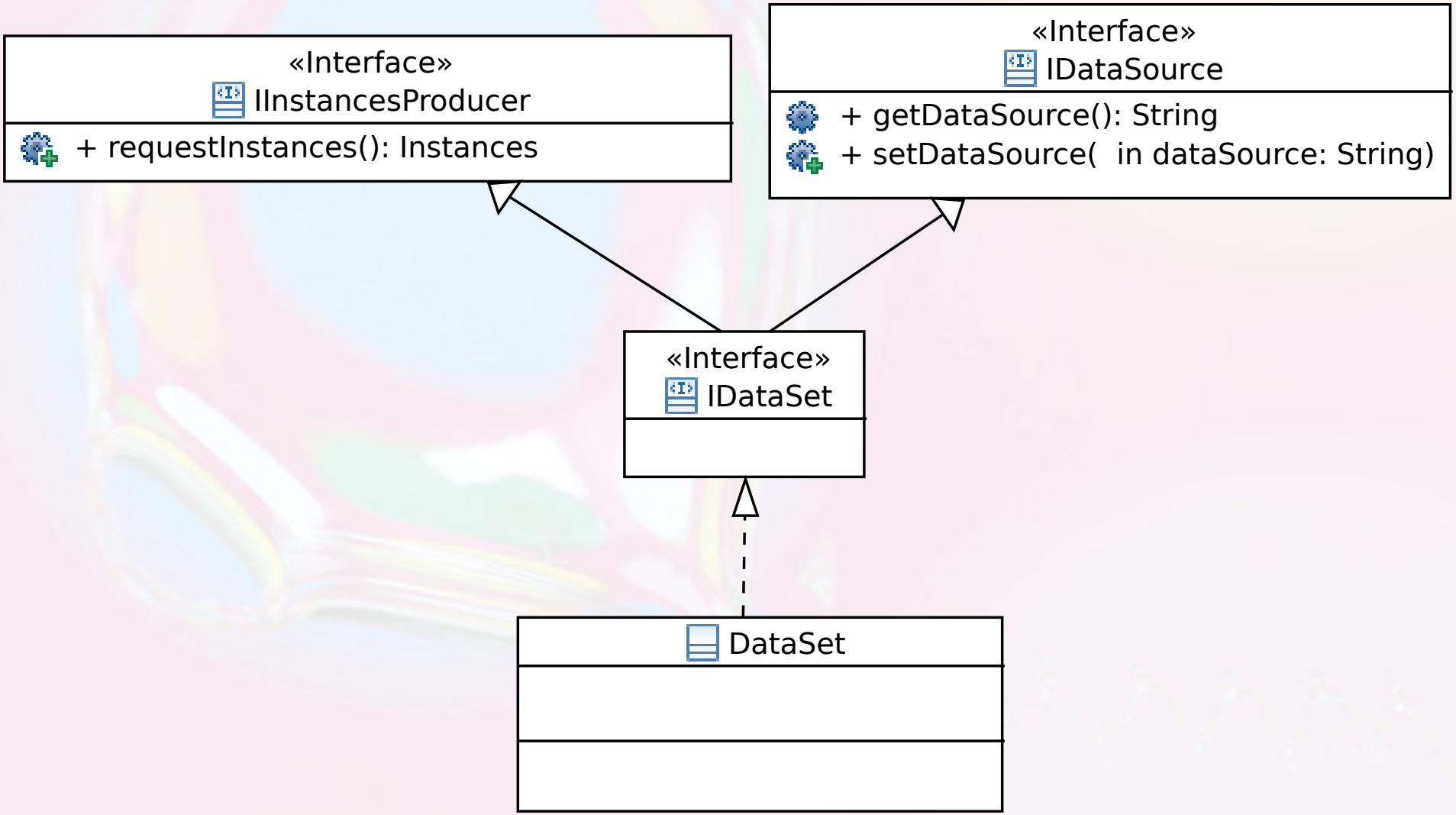


Estudo Caso

Componente DataSet com Propriedade dataSource

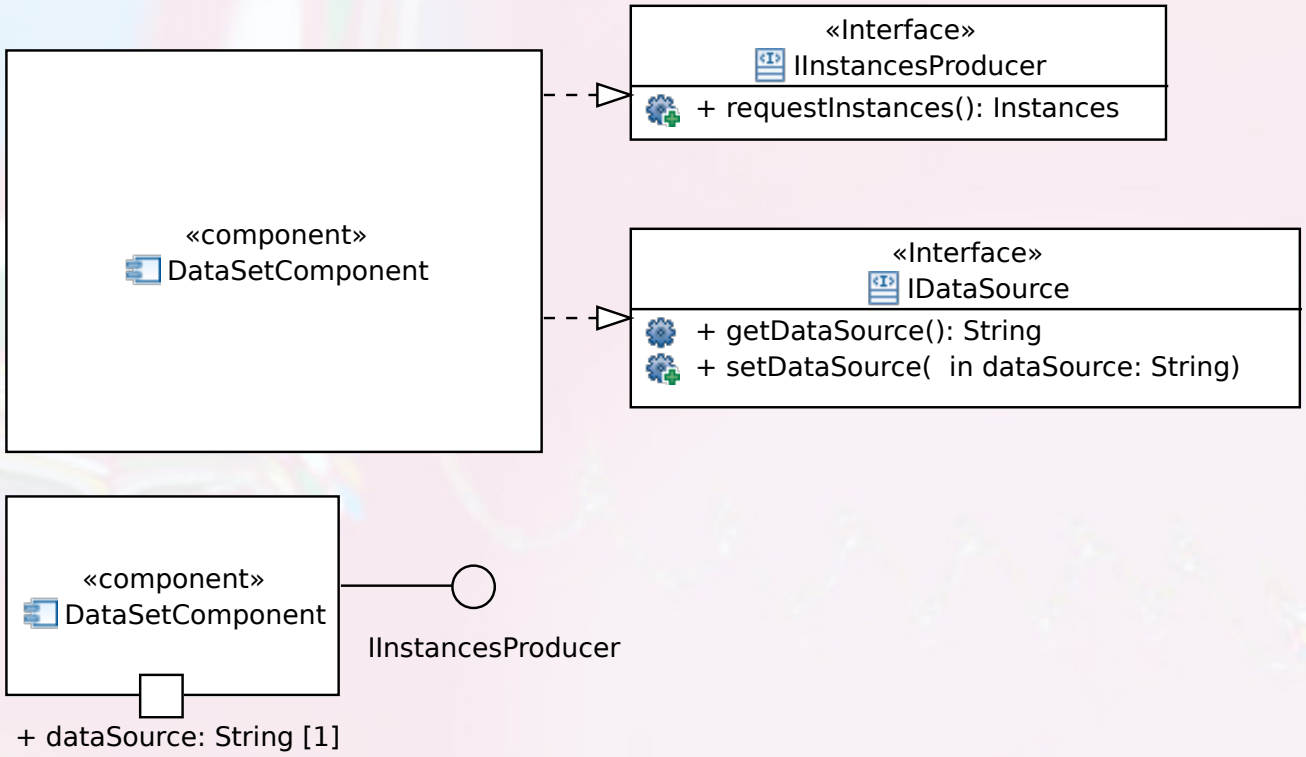


DataSet
Estrutura de Classes



Acessando os Serviços pela Interface

```
IDataSet ds = new DataSetComponent();
ds.setDataSource("....csv");
System.out.println(ds.requestInstances());
```



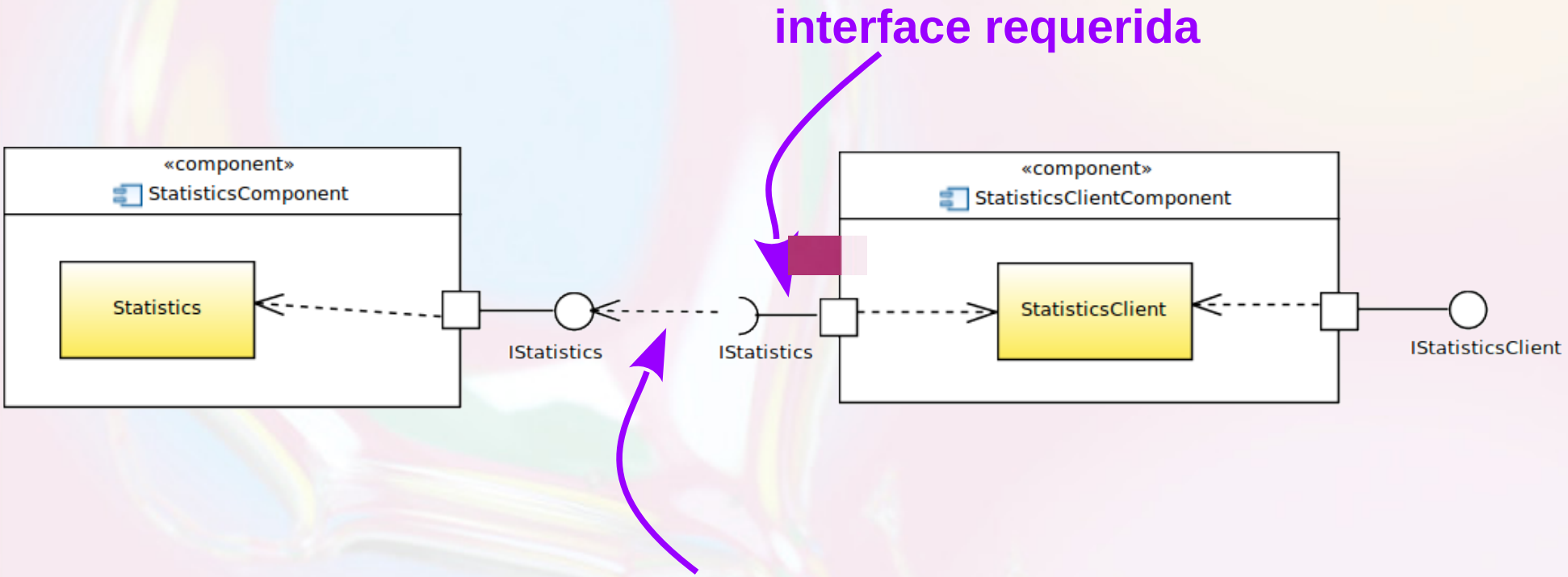
Interface Requerida

Interface Requerida

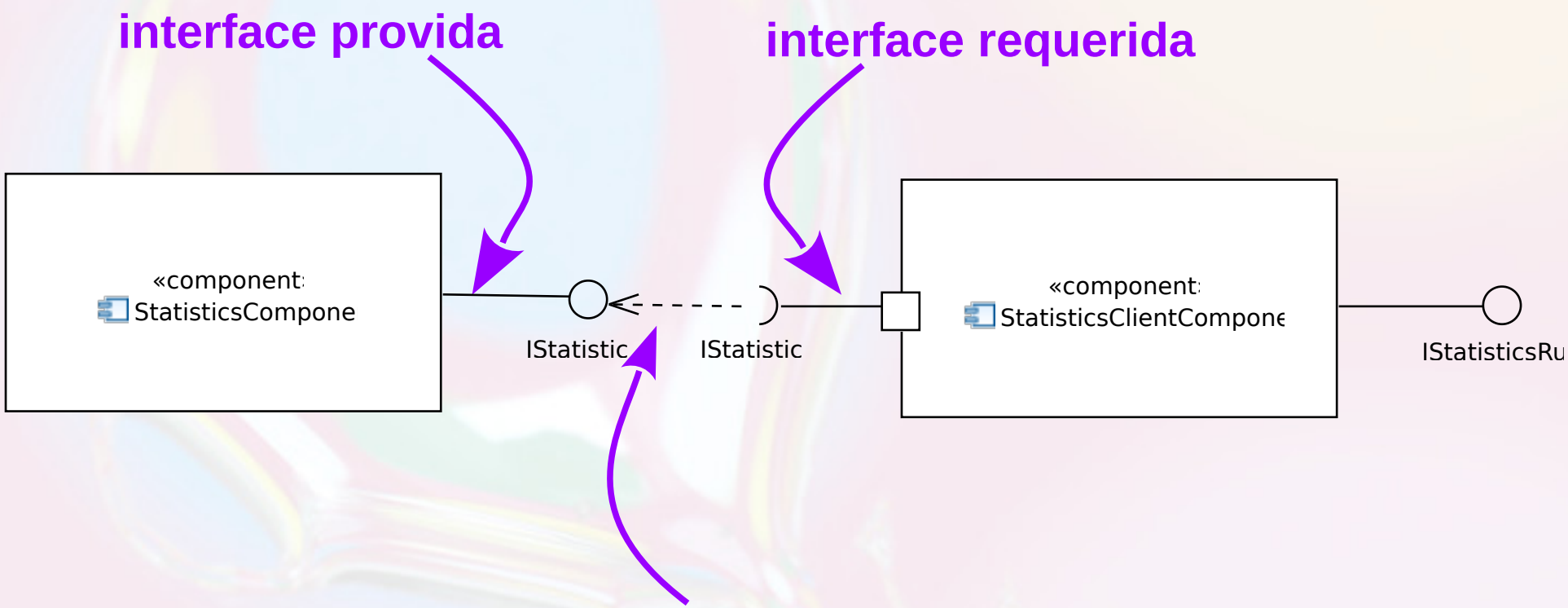
- Explicita a dependência de um componente por uma interface de outro componente.
- Interface X requerida por A = A requer um componente que oferece interface X

Interfaces Providas e Requeridas tornam explícitas todas as dependências

Interface Provida e Requerida



Interface Provida e Requerida
Componente Client (blackbox)

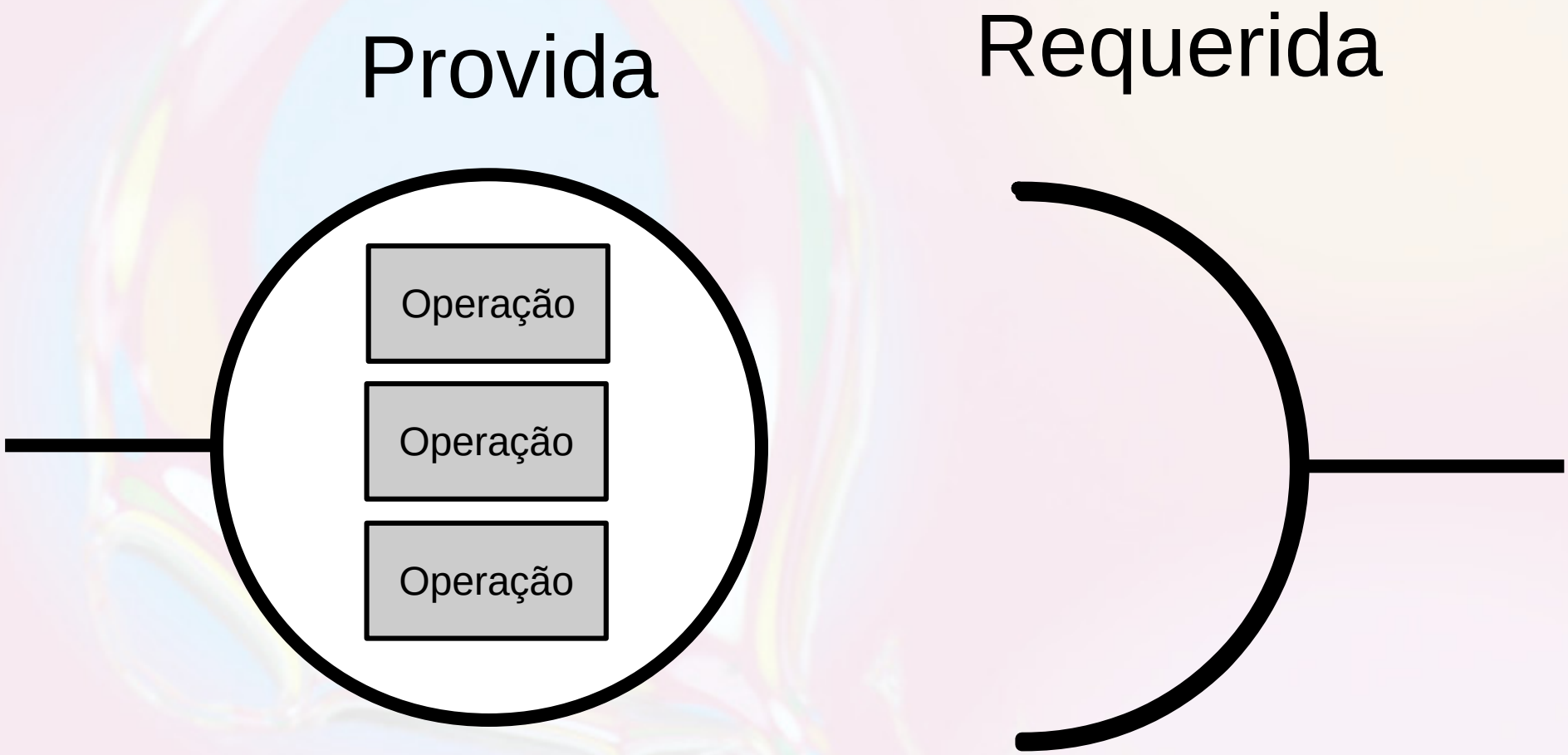


Interface Provida e Requerida

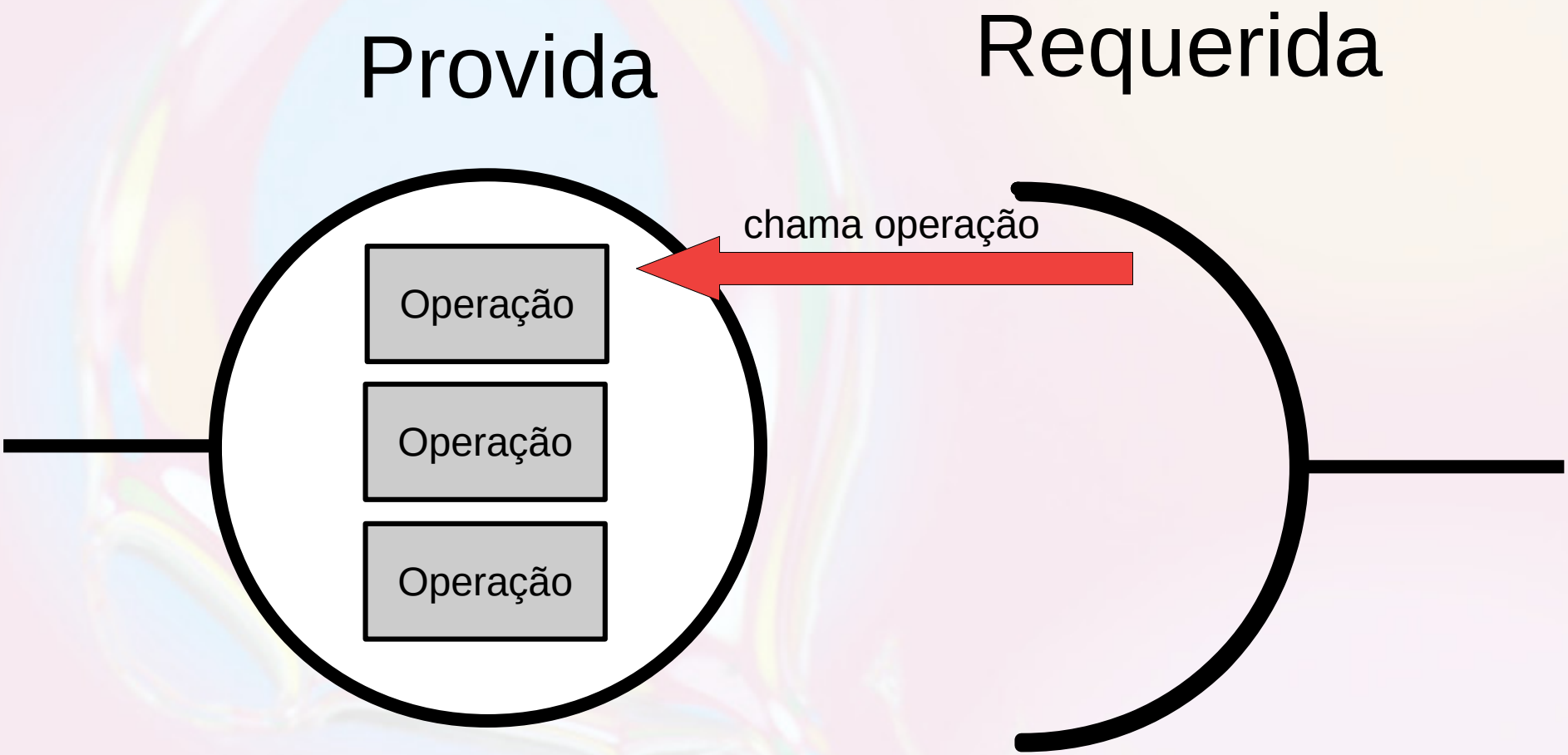
Provida

Requerida

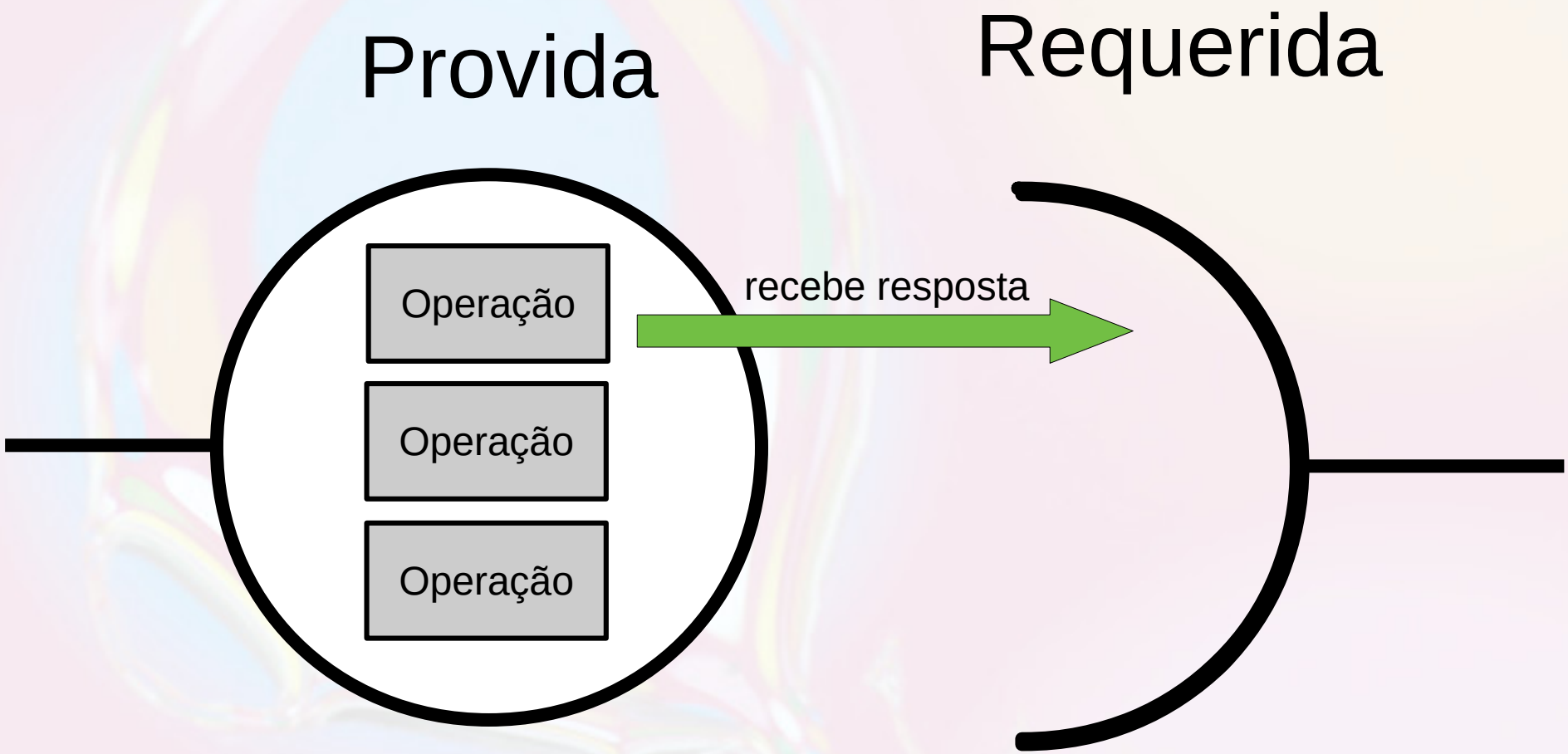
Interface Provida e Requerida



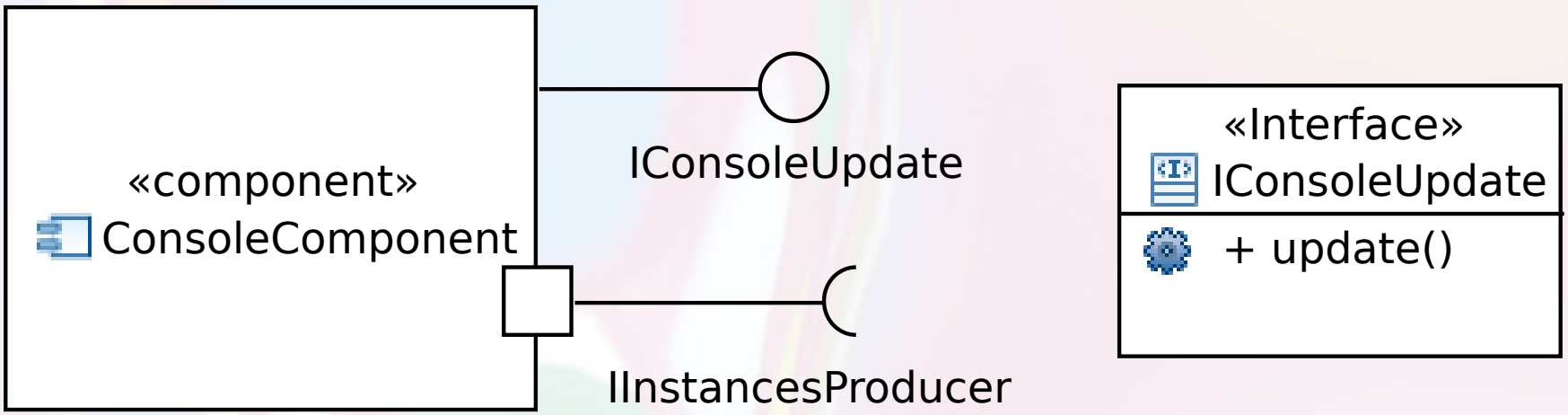
Interface Provida e Requerida



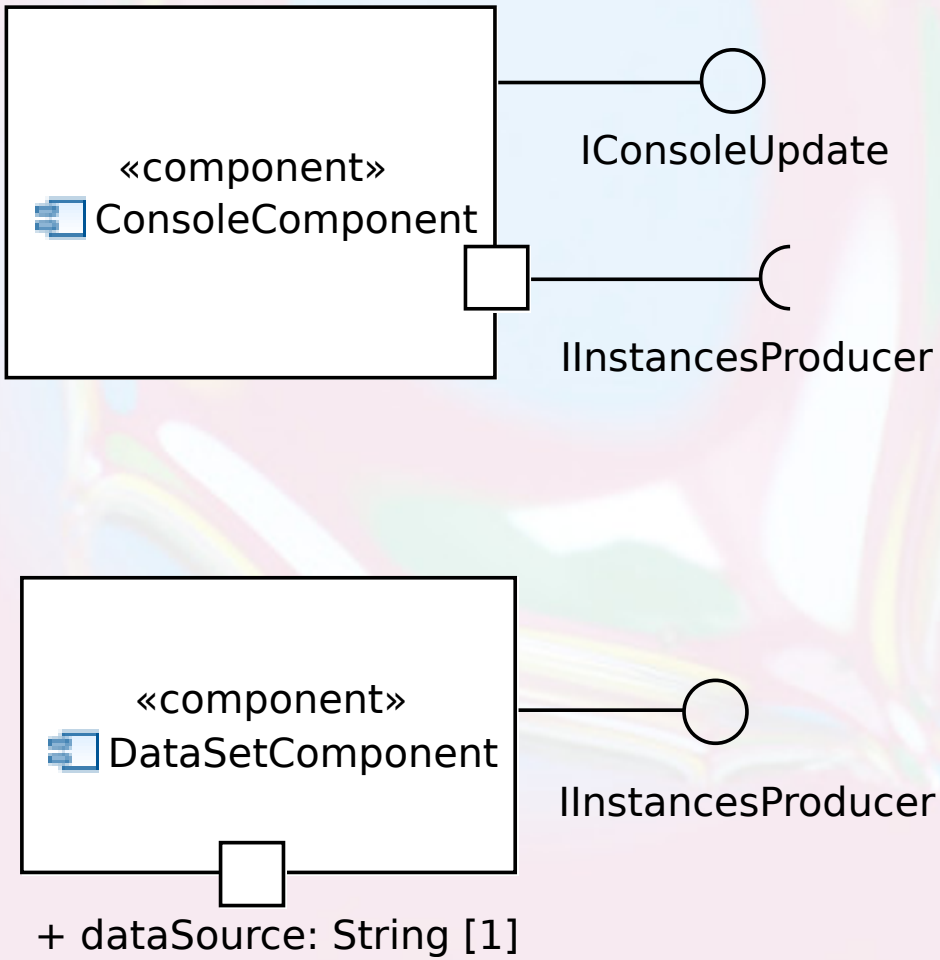
Interface Provida e Requerida



Componente Console

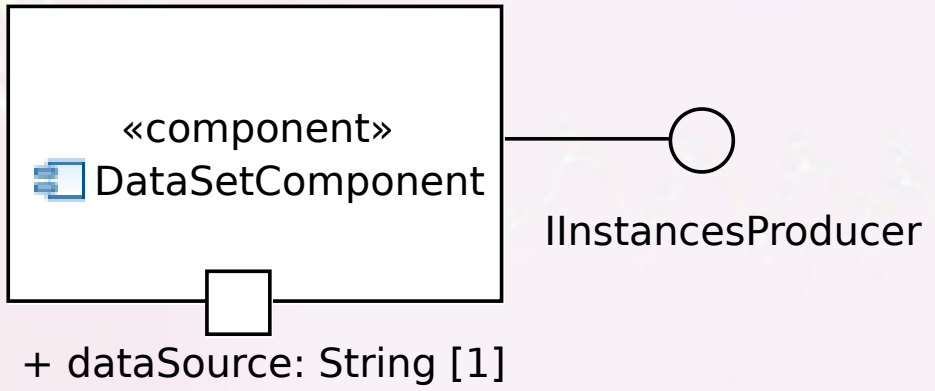


Conectando Componentes



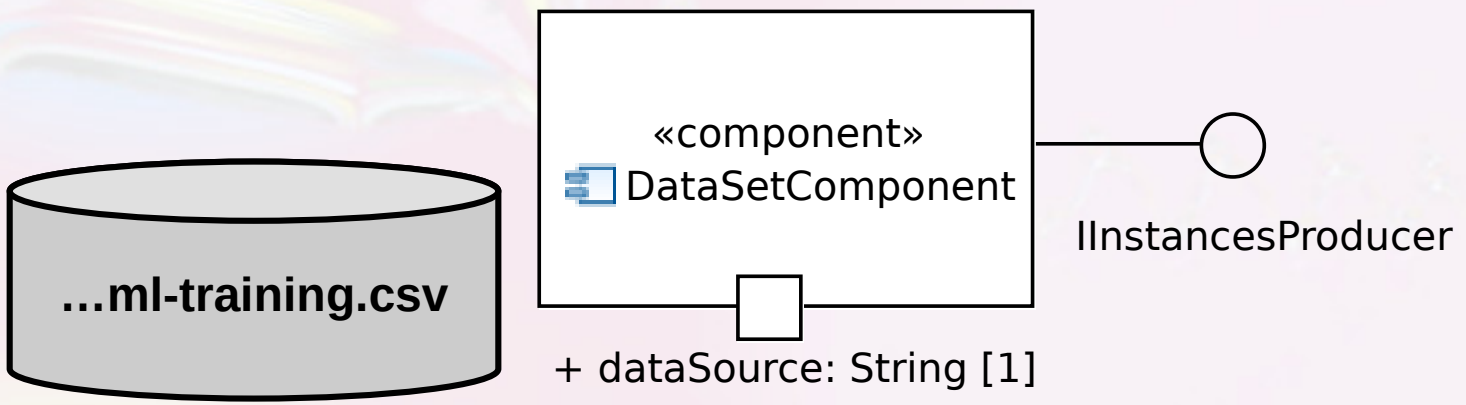
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();
```



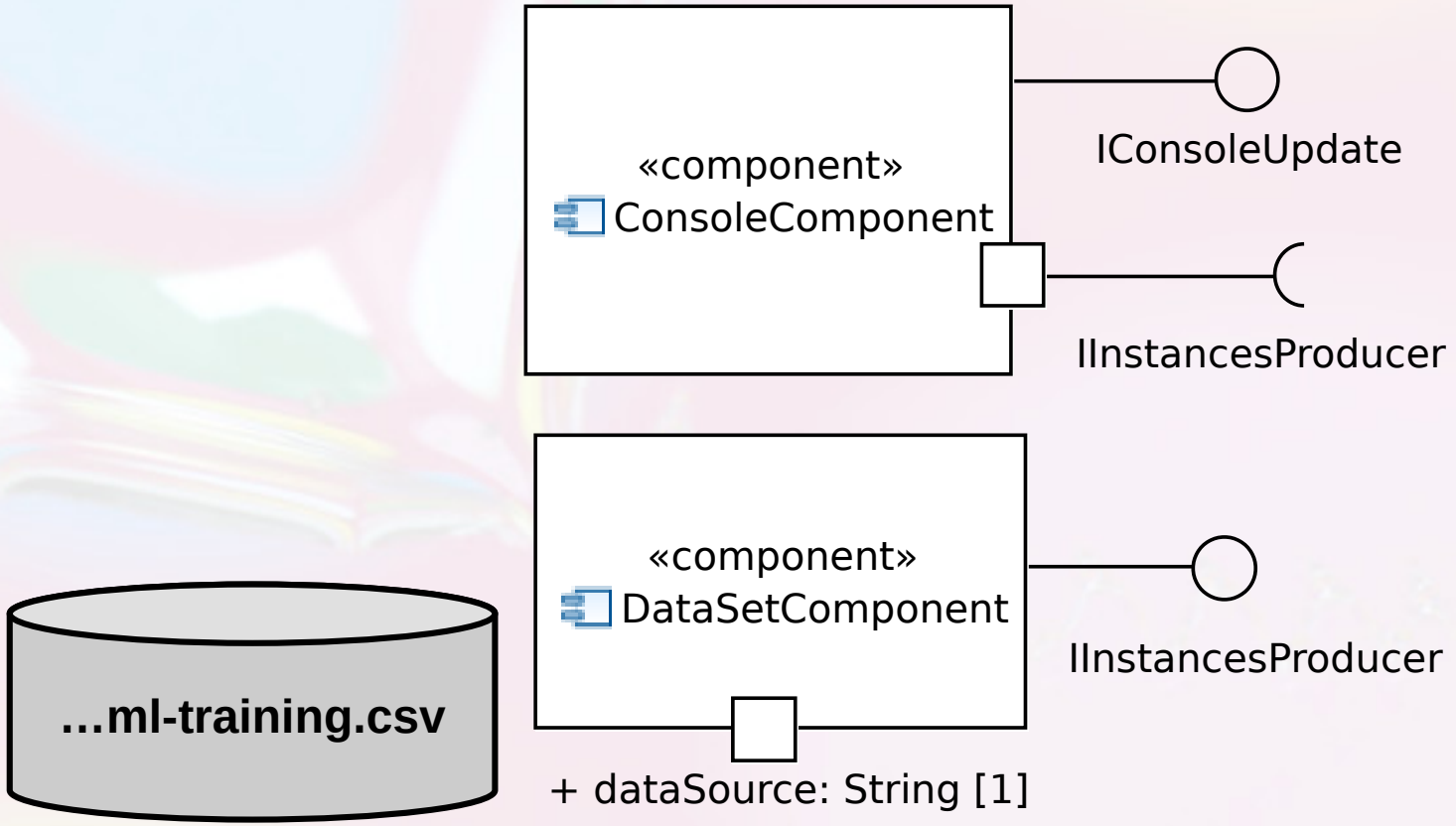
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();
dataset.setDataSource("…ml-training.csv");
```



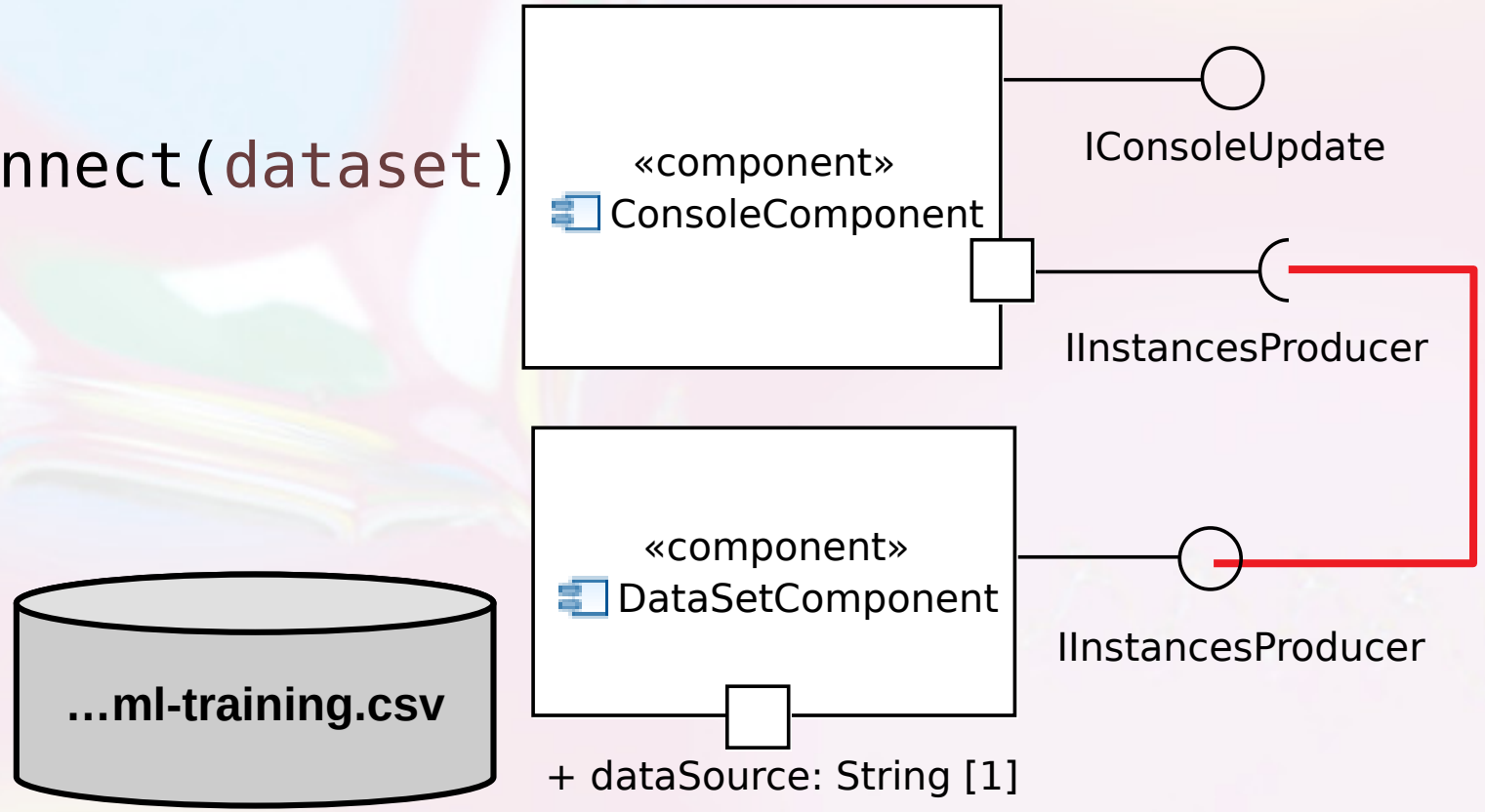
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();
dataset.setDataSource("….csv");
IConsole console = new ConsoleComponent();
```



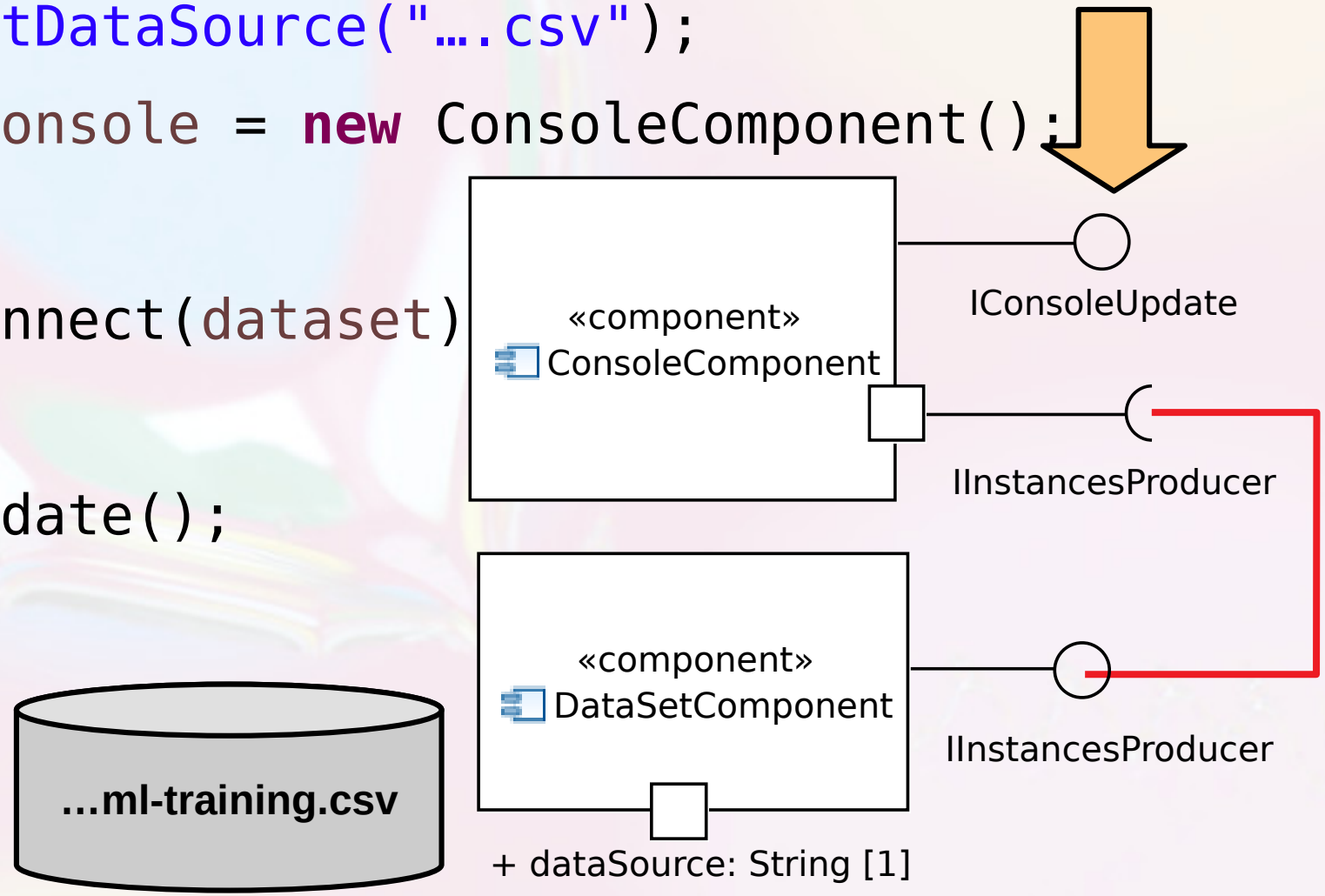
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();
dataset.setDataSource("….csv");
IConsole console = new ConsoleComponent();

console.connect(dataset)
```

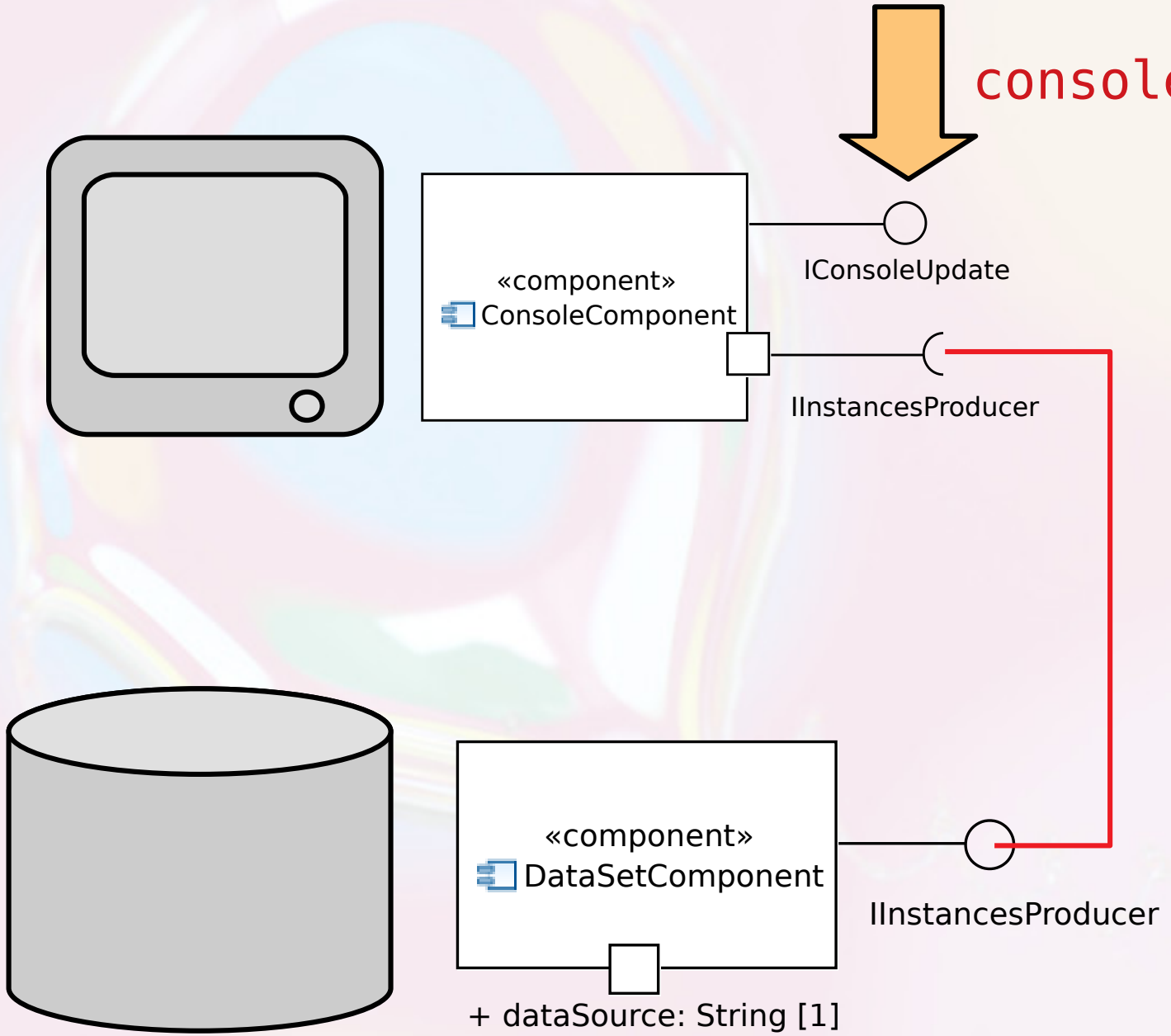


Conectando Componentes

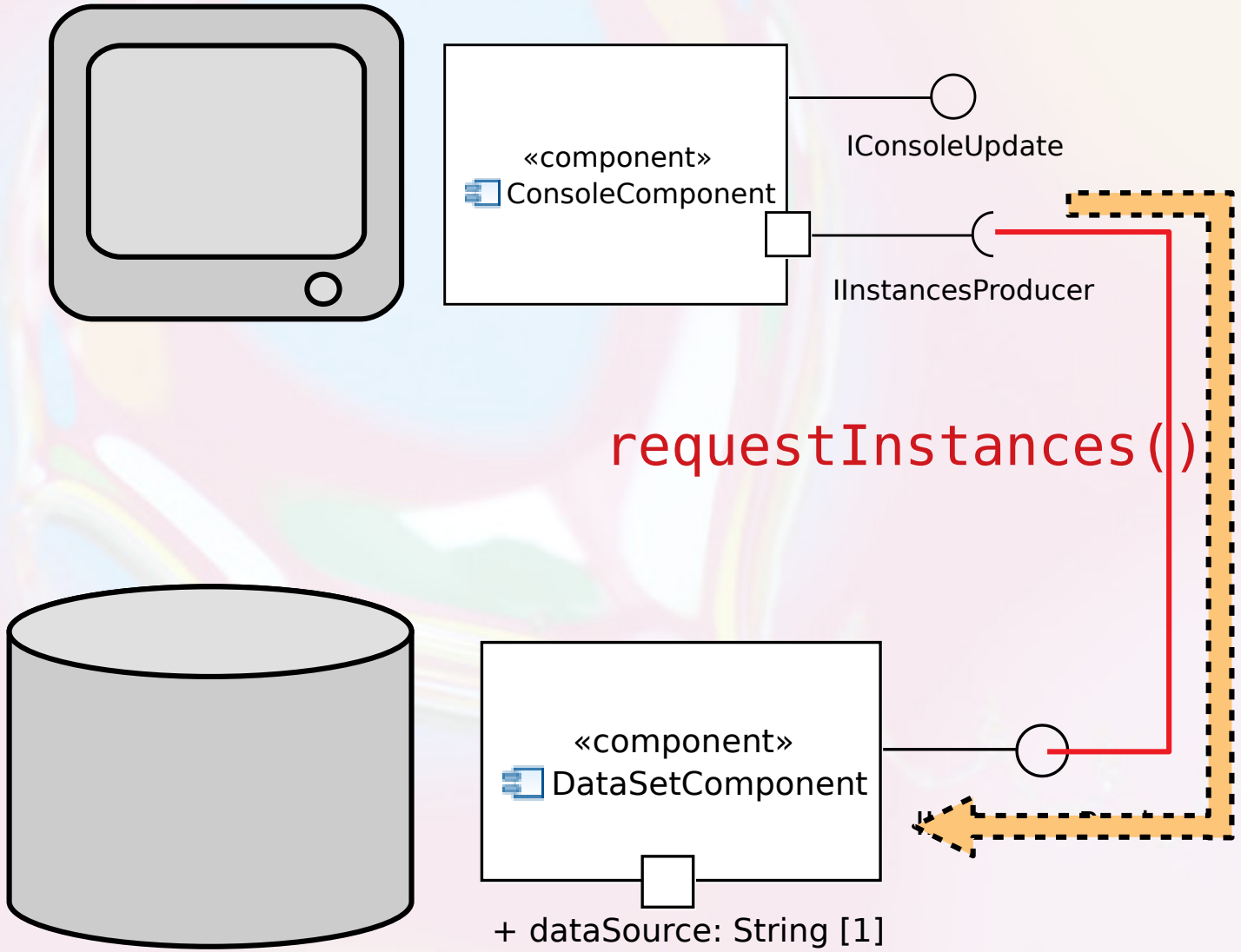
```
IDataSet dataset = new DataSetComponent();
dataset.setDataSource("….csv");
IConsole console = new ConsoleComponent();

console.connect(dataset)

console.update();
```



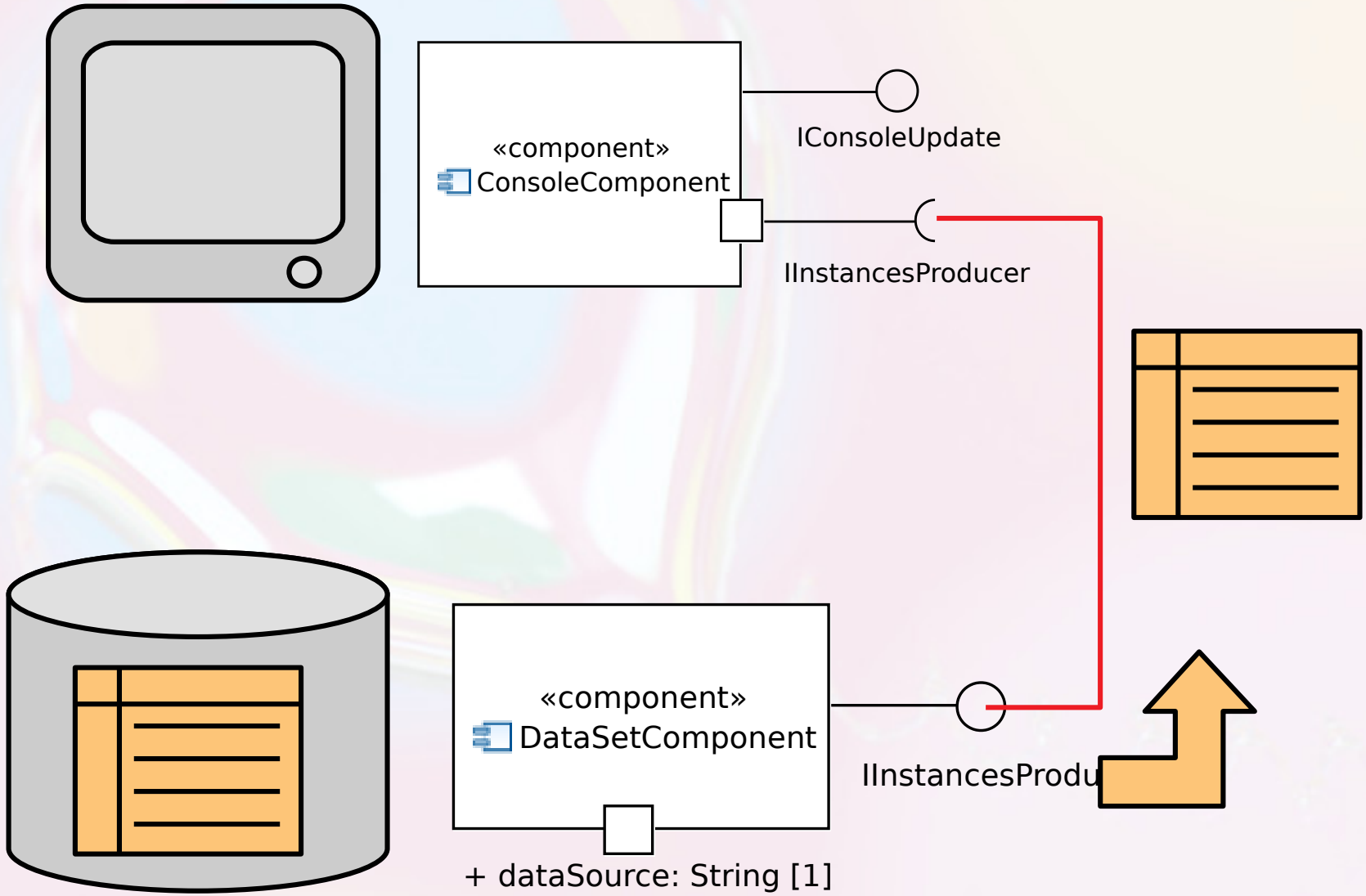
Conectando Componentes



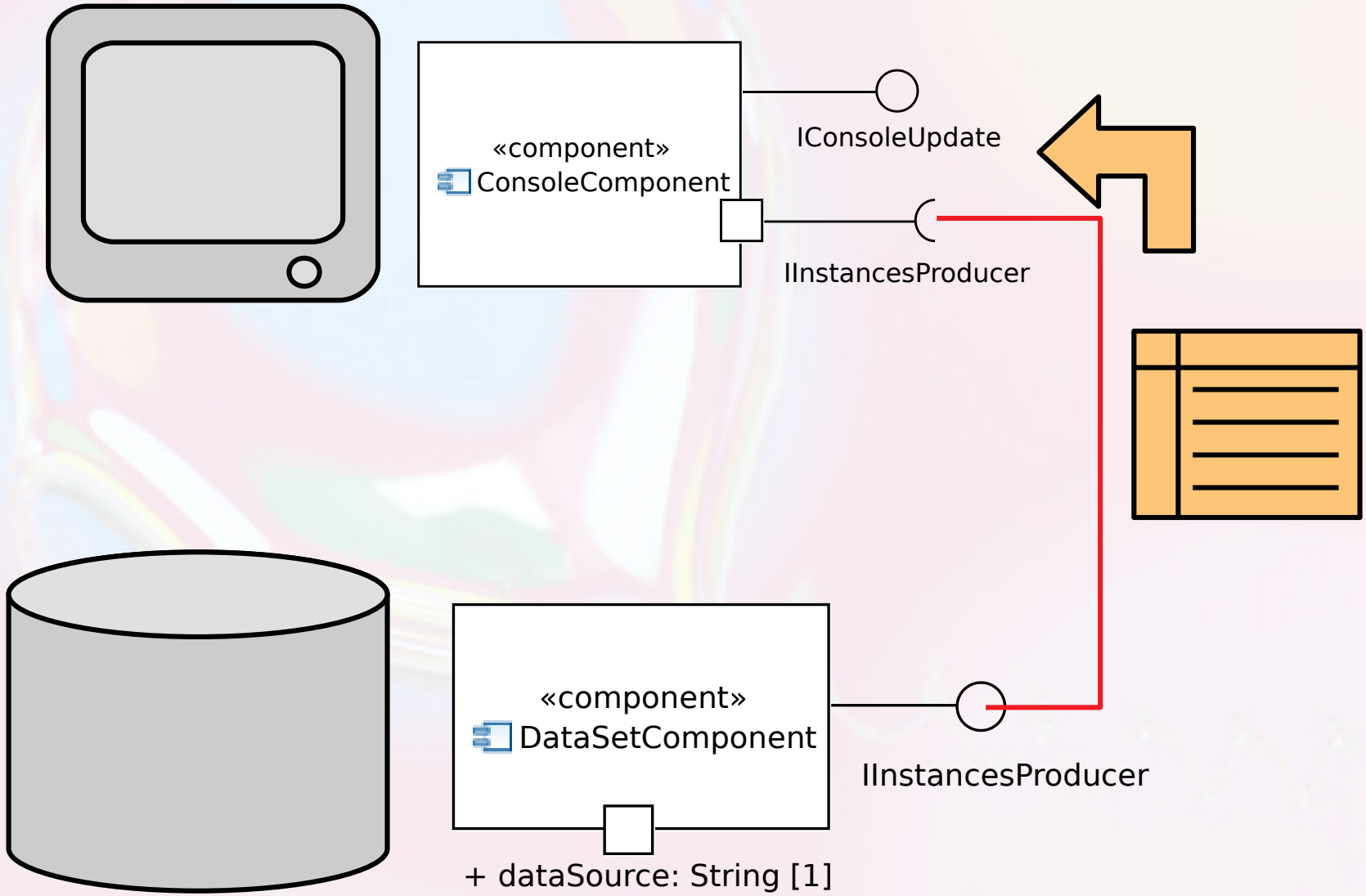
Conectando Componentes



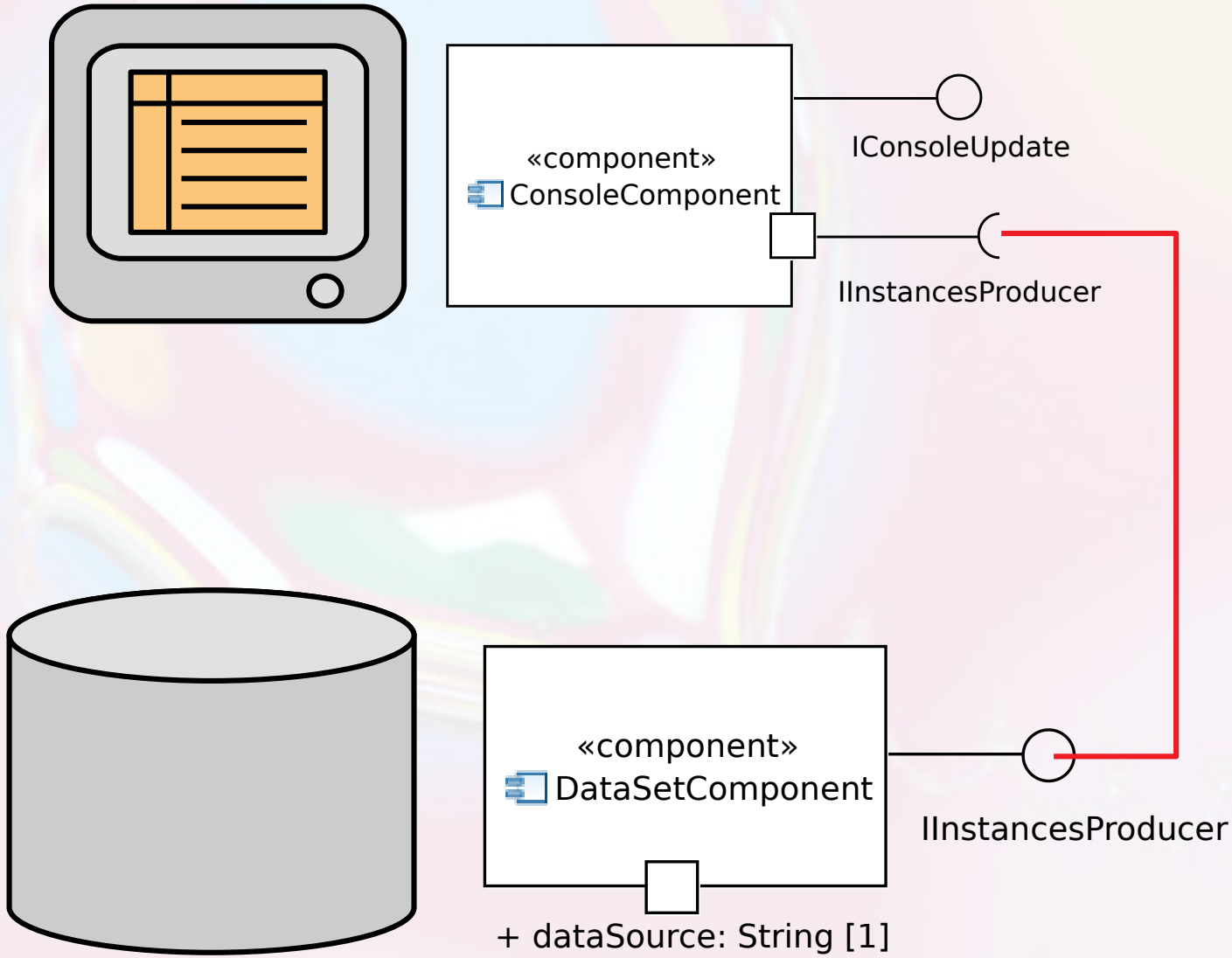
Conectando Componentes



Conectando Componentes



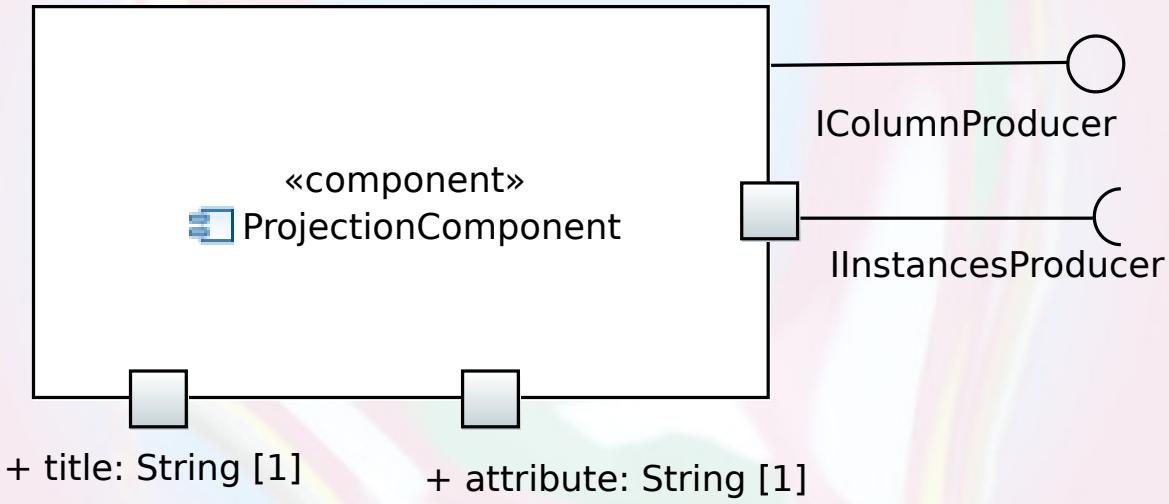
Conectando Componentes

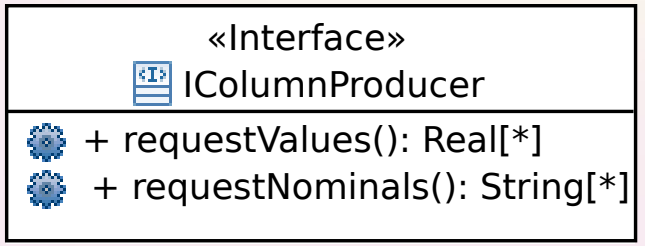


Exercício 3

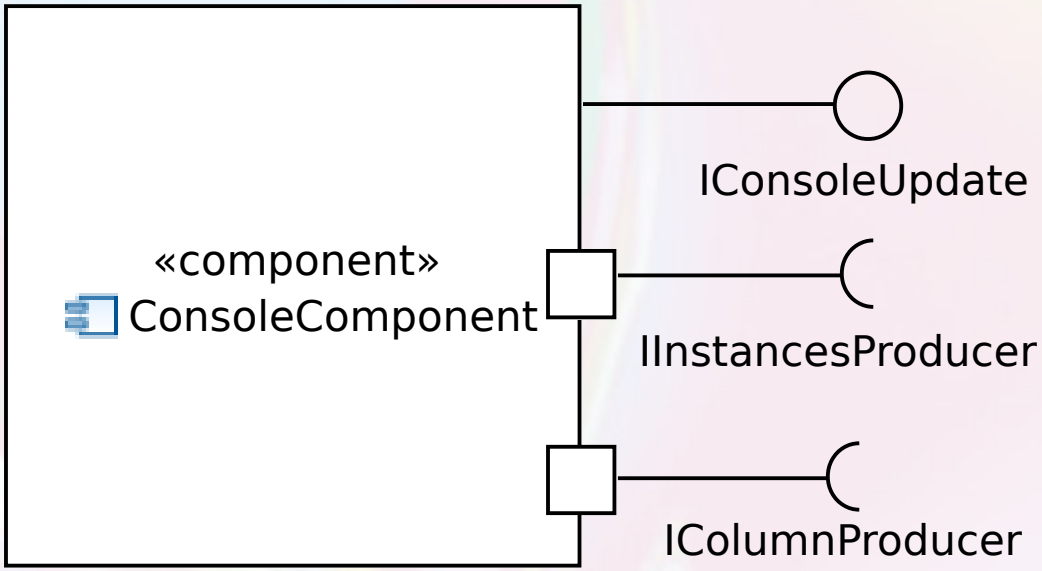
- Realize uma filtragem da coluna de “name” da tabela.

Componente Projection

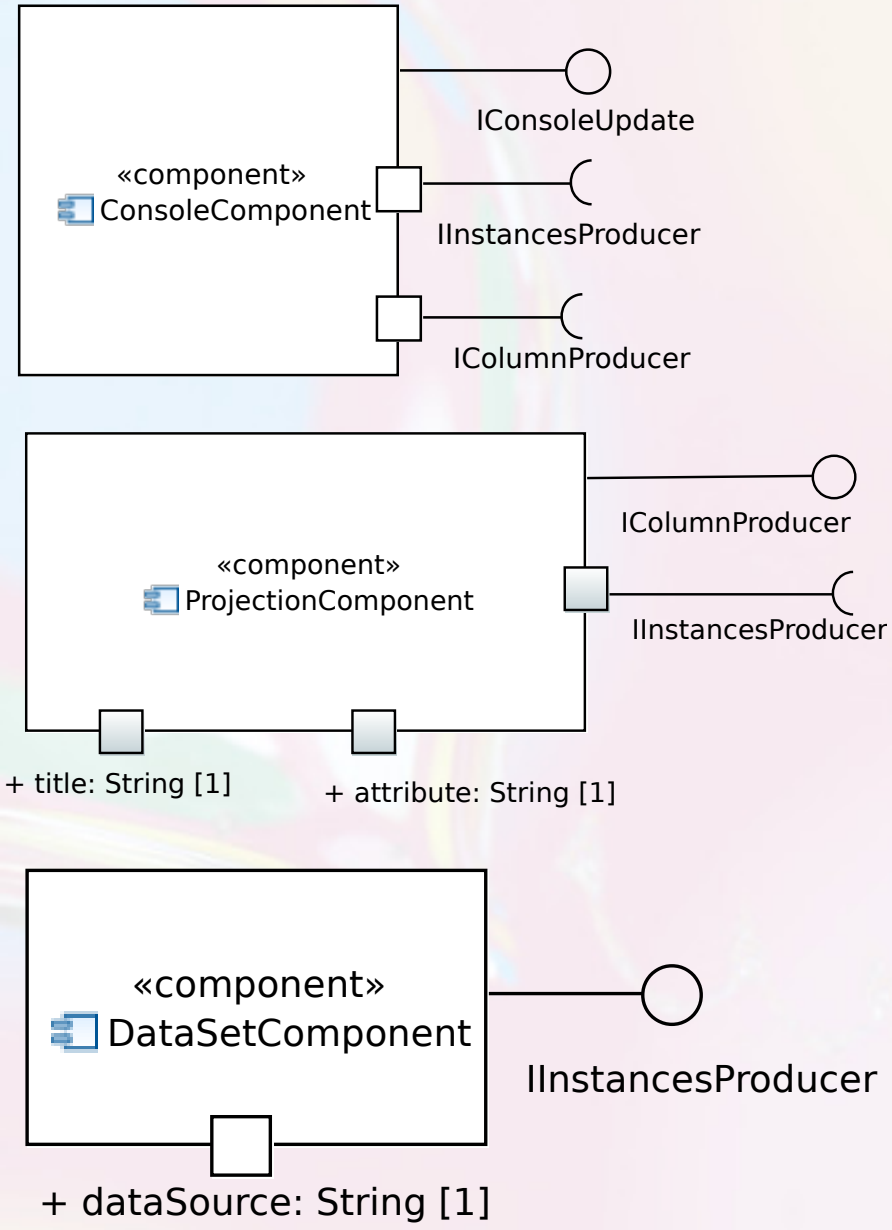




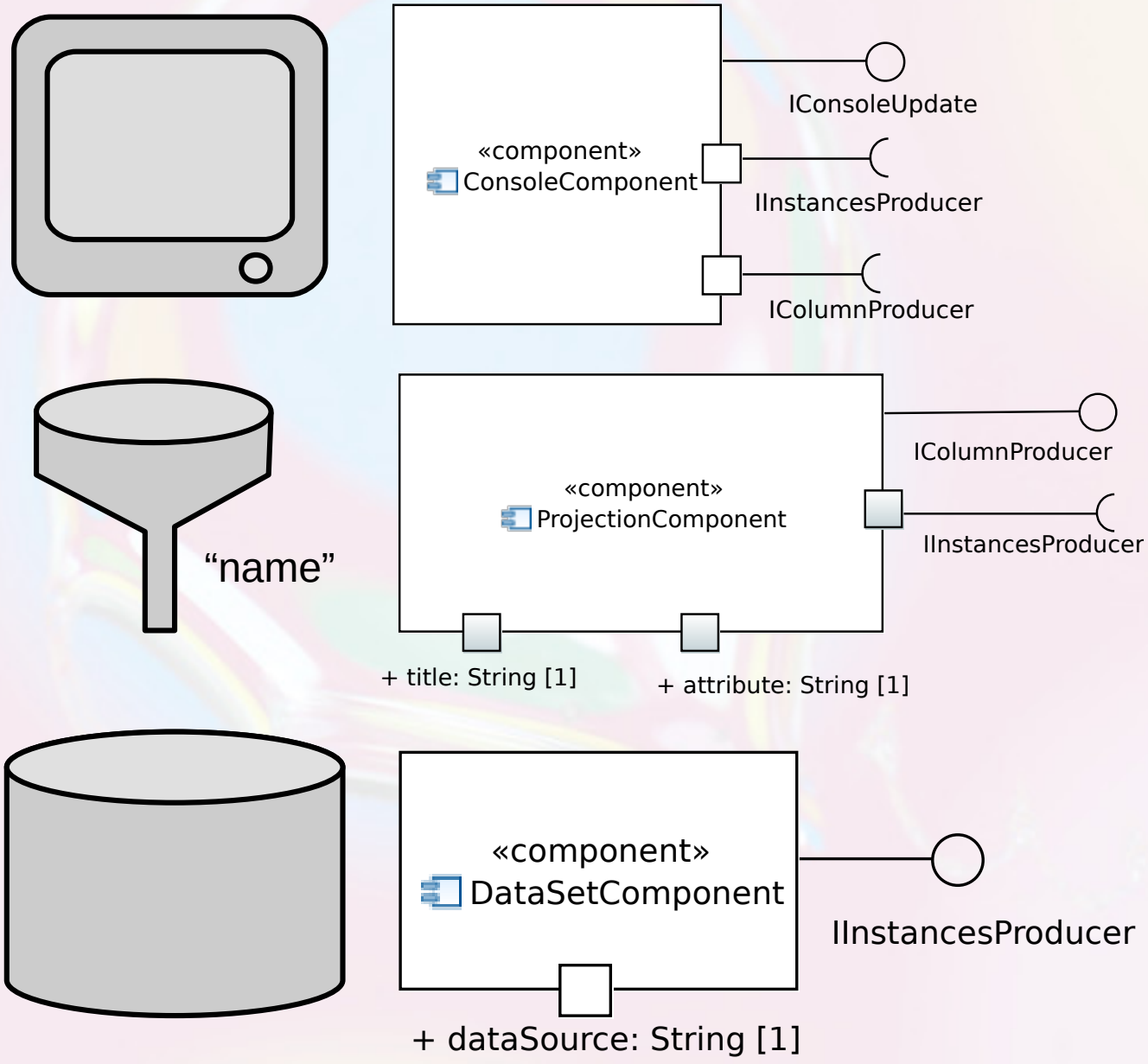
Componente Console
Terceira Interface



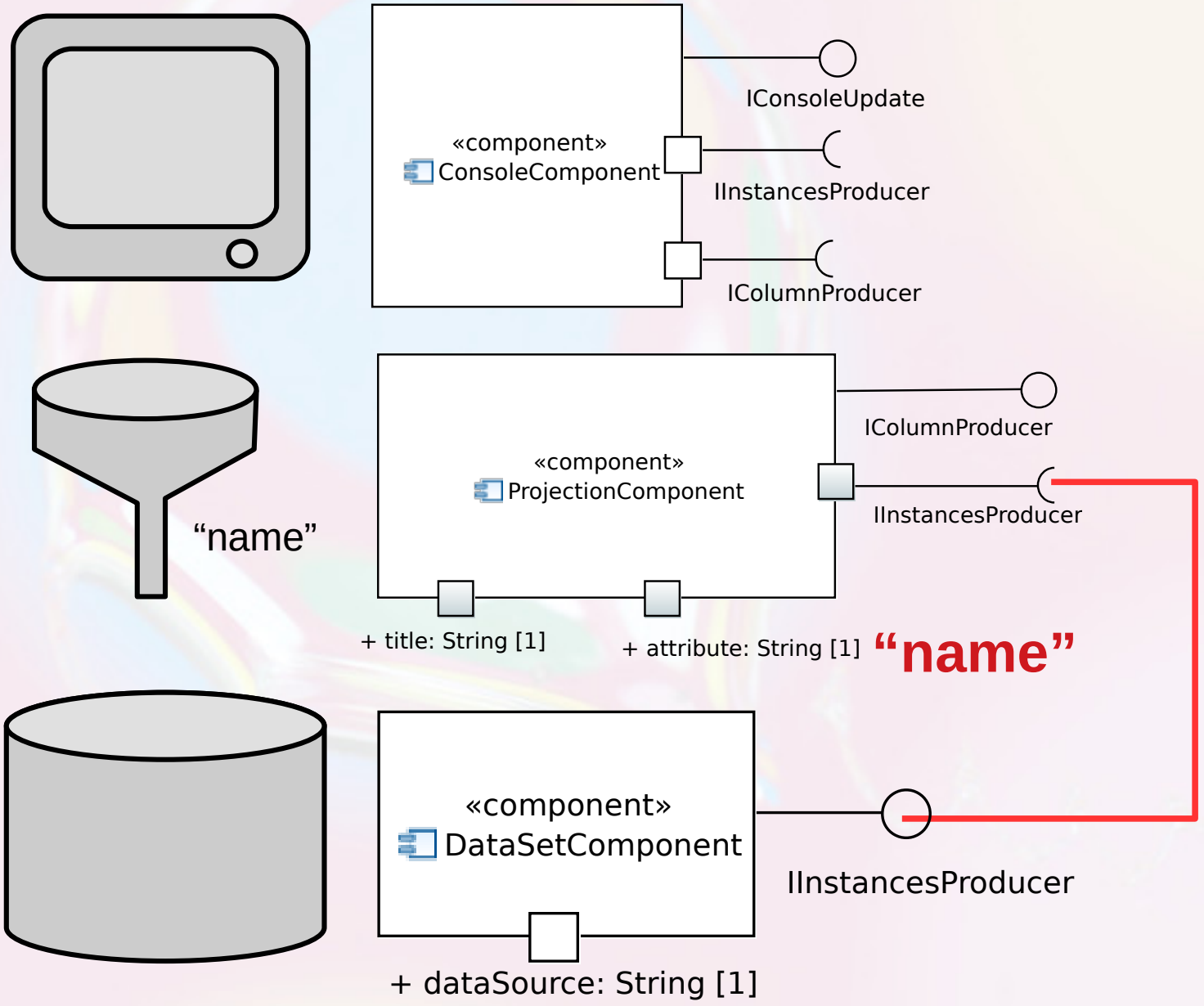
Conectando Três Componentes



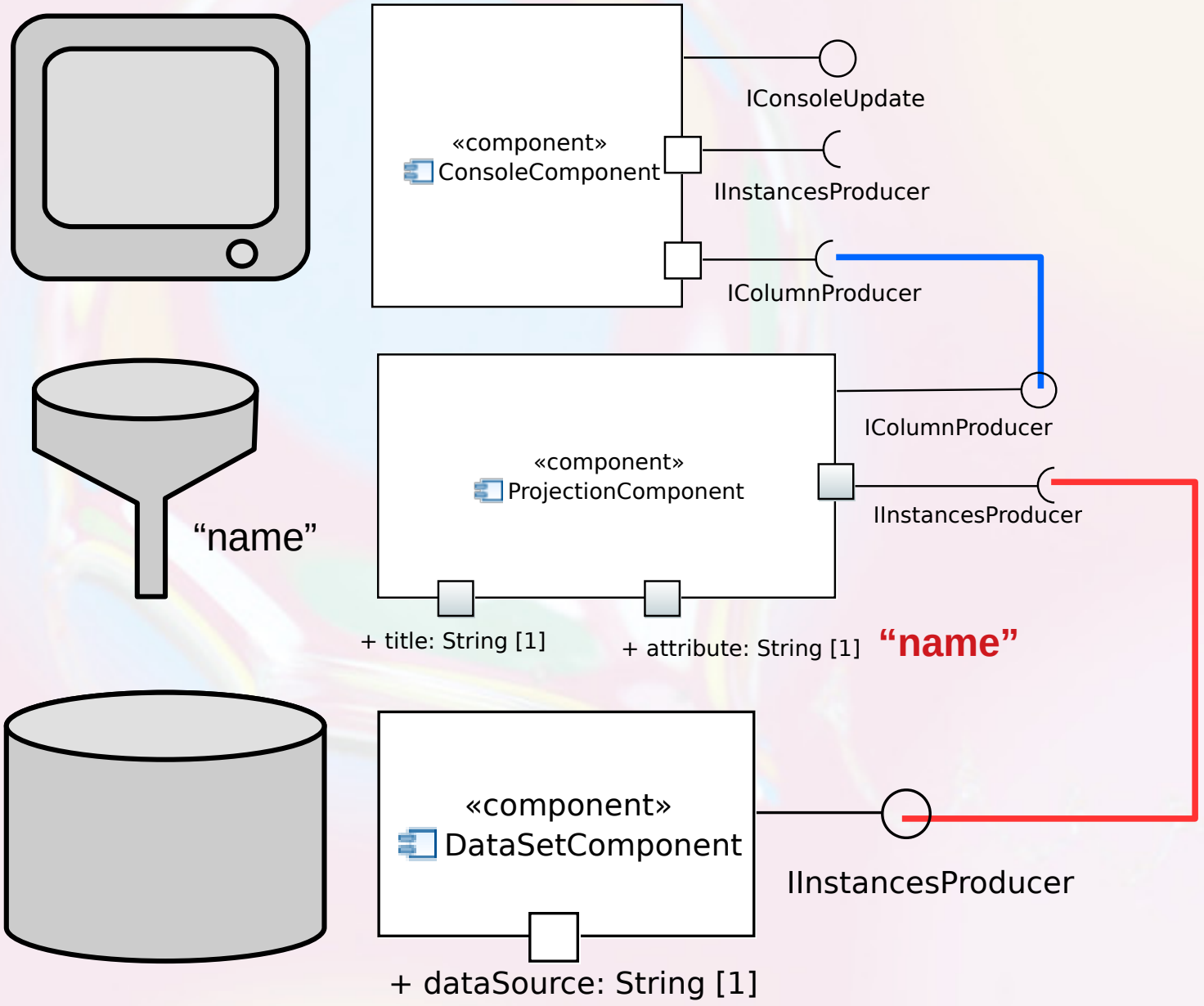
Conectando Três Componentes



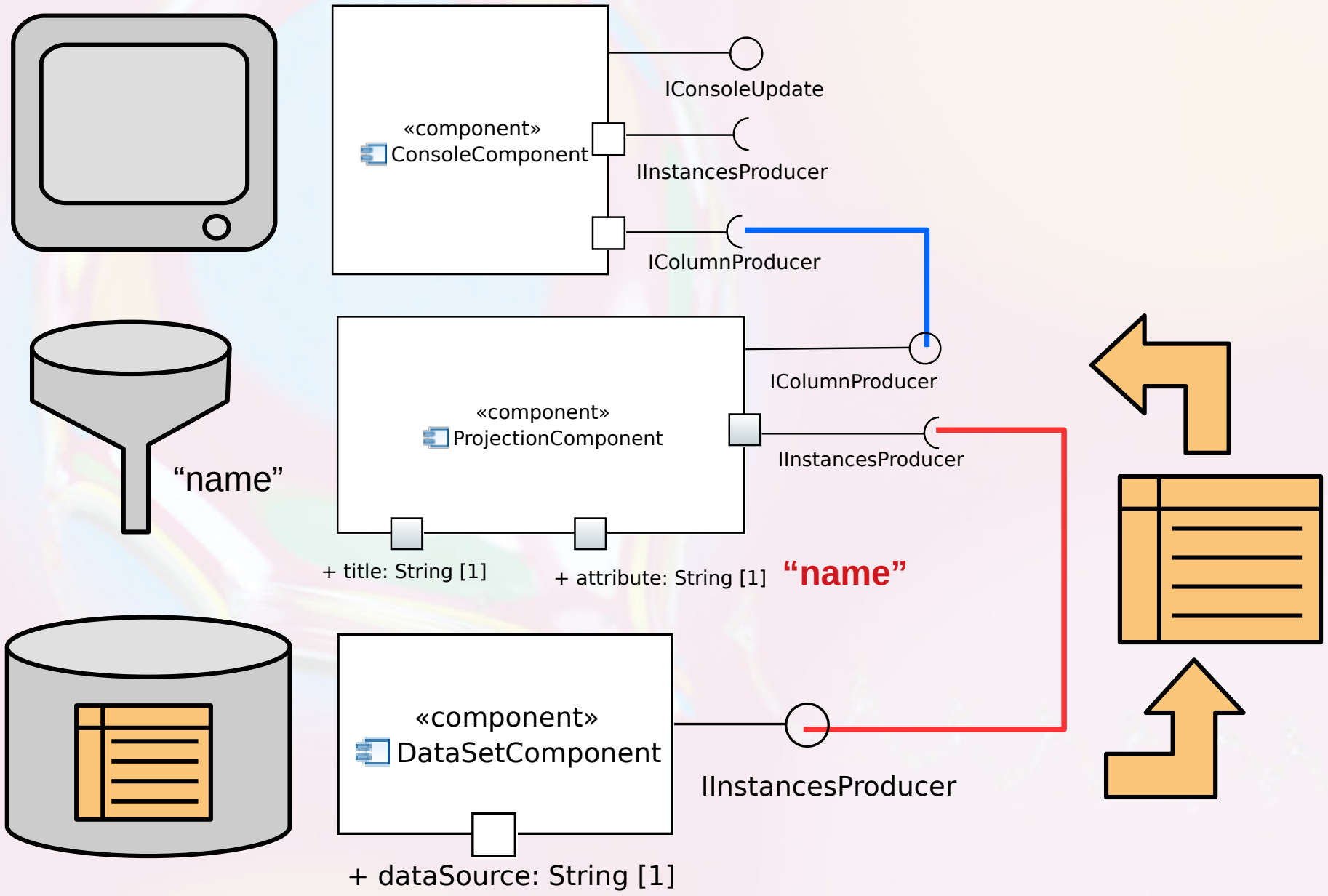
Conectando Três Componentes



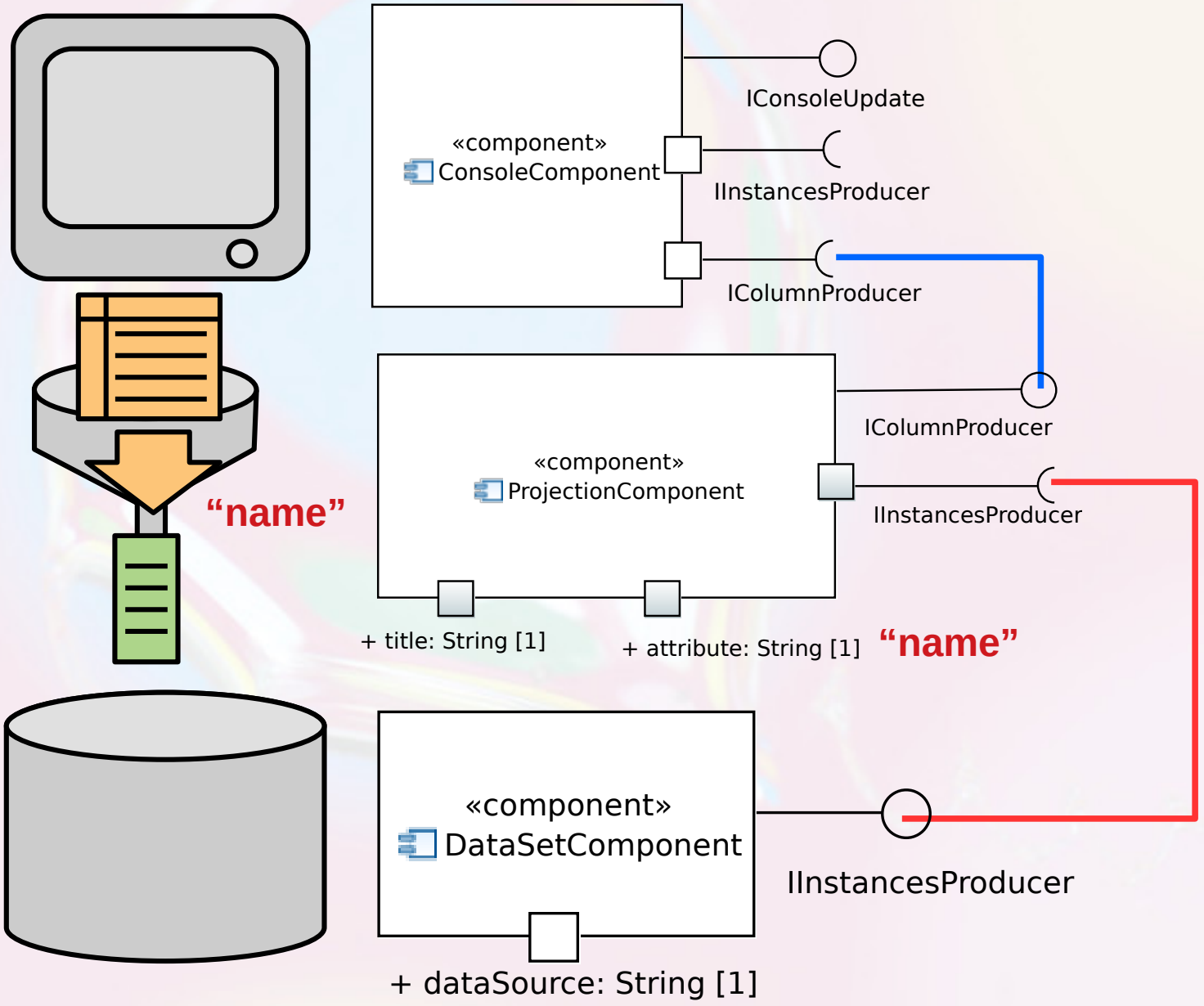
Conectando Três Componentes



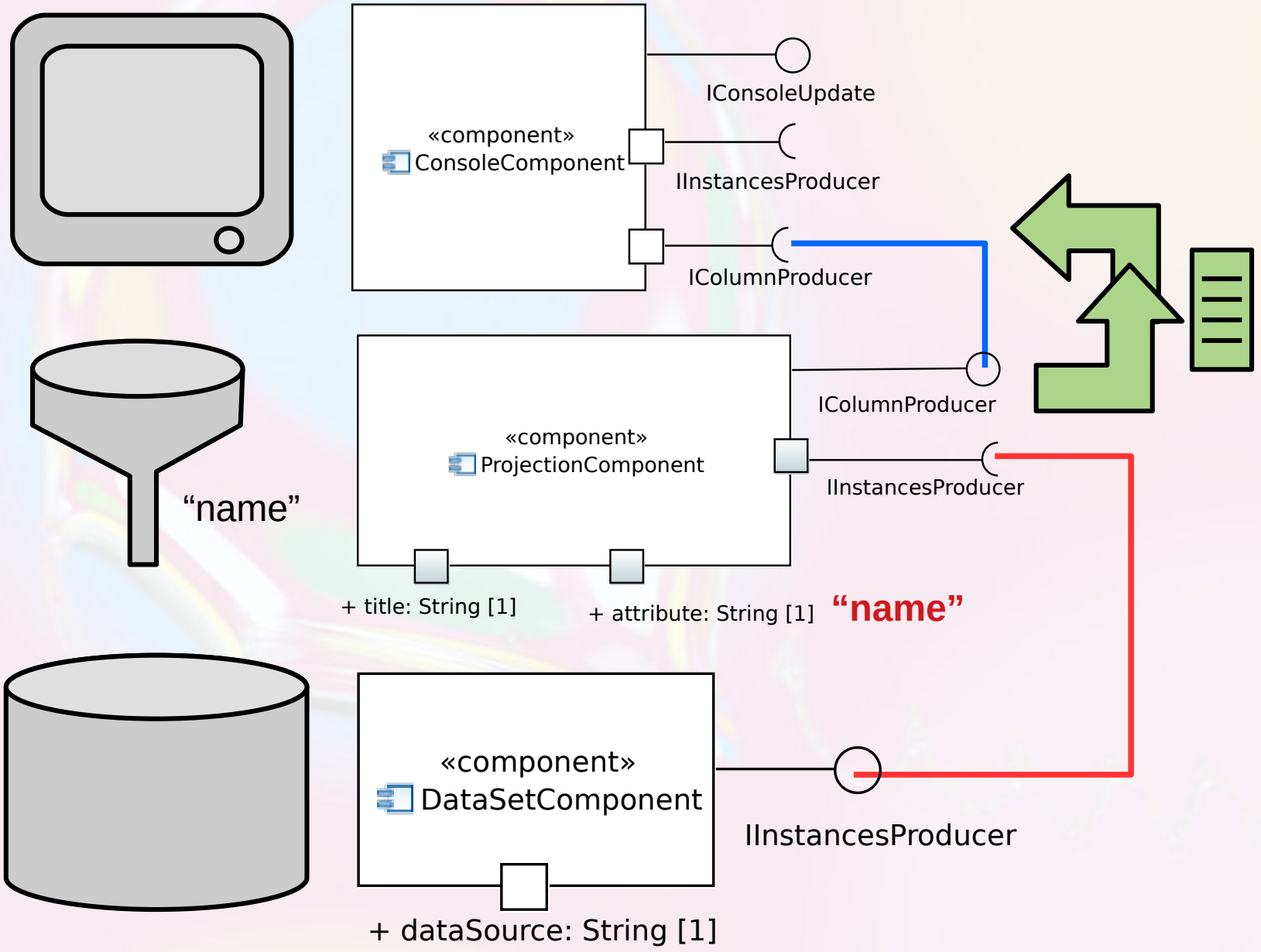
Conectando Três Componentes



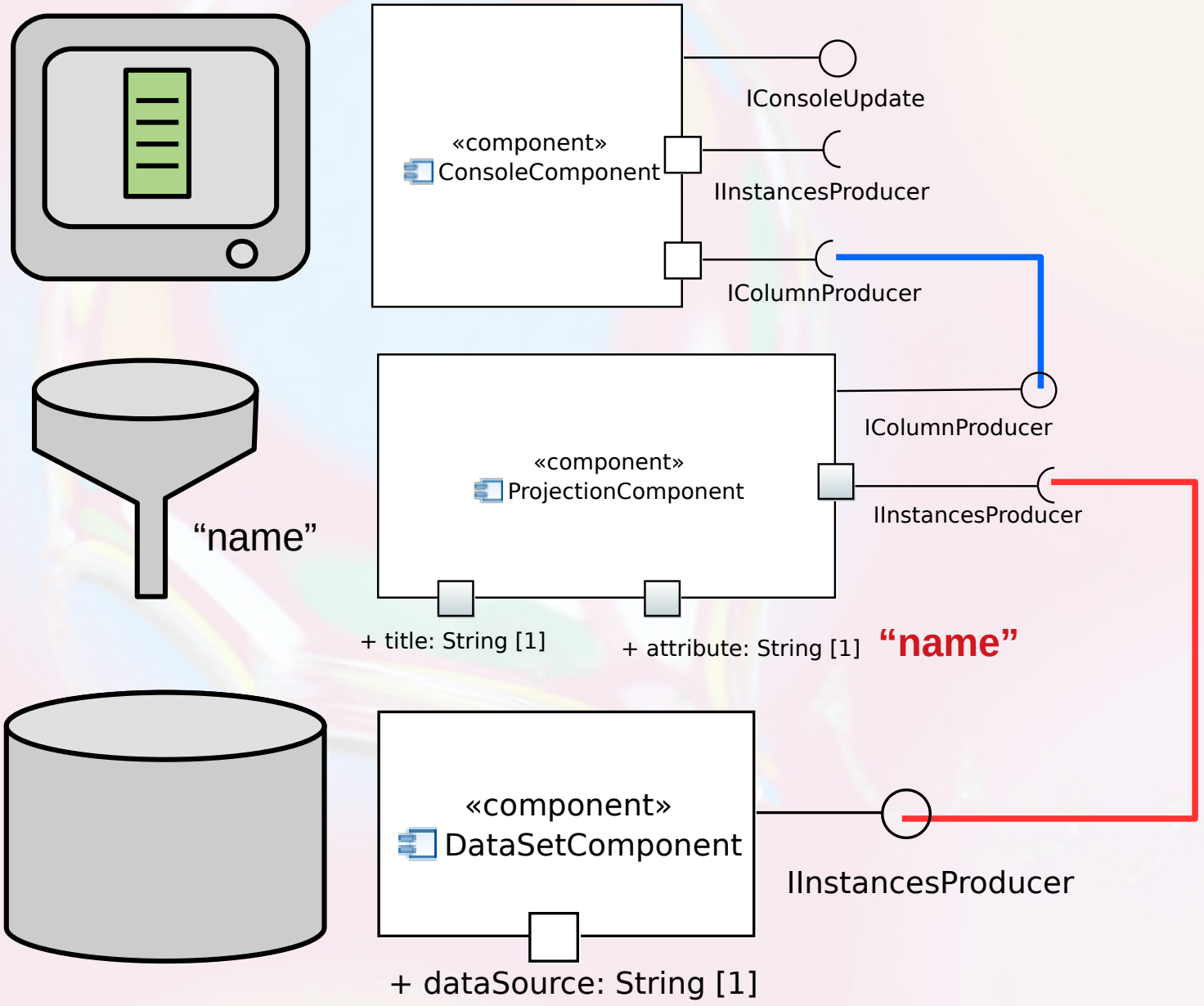
Conectando Três Componentes



Conectando Três Componentes



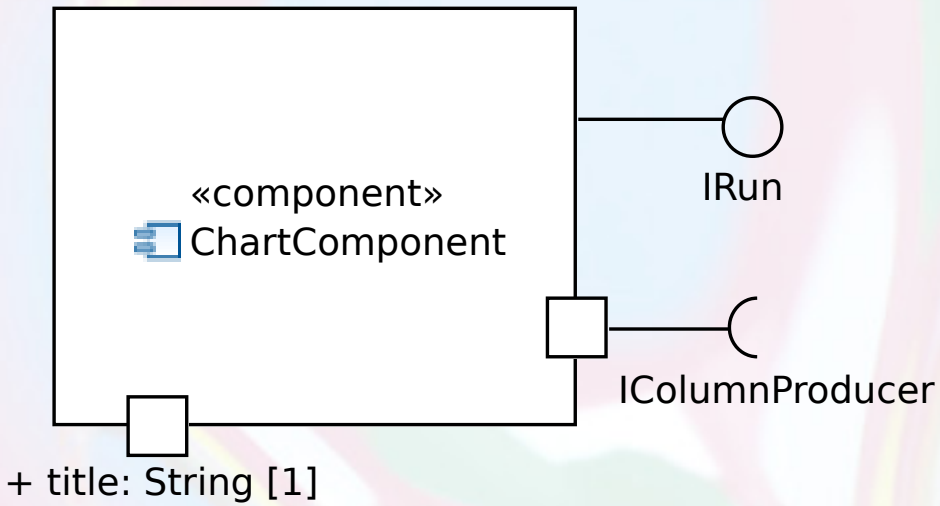
Conectando Três Componentes

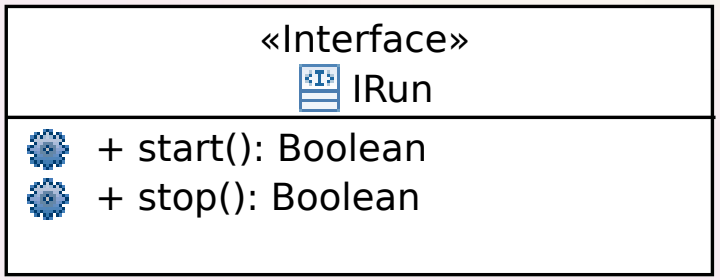


Exercício 4

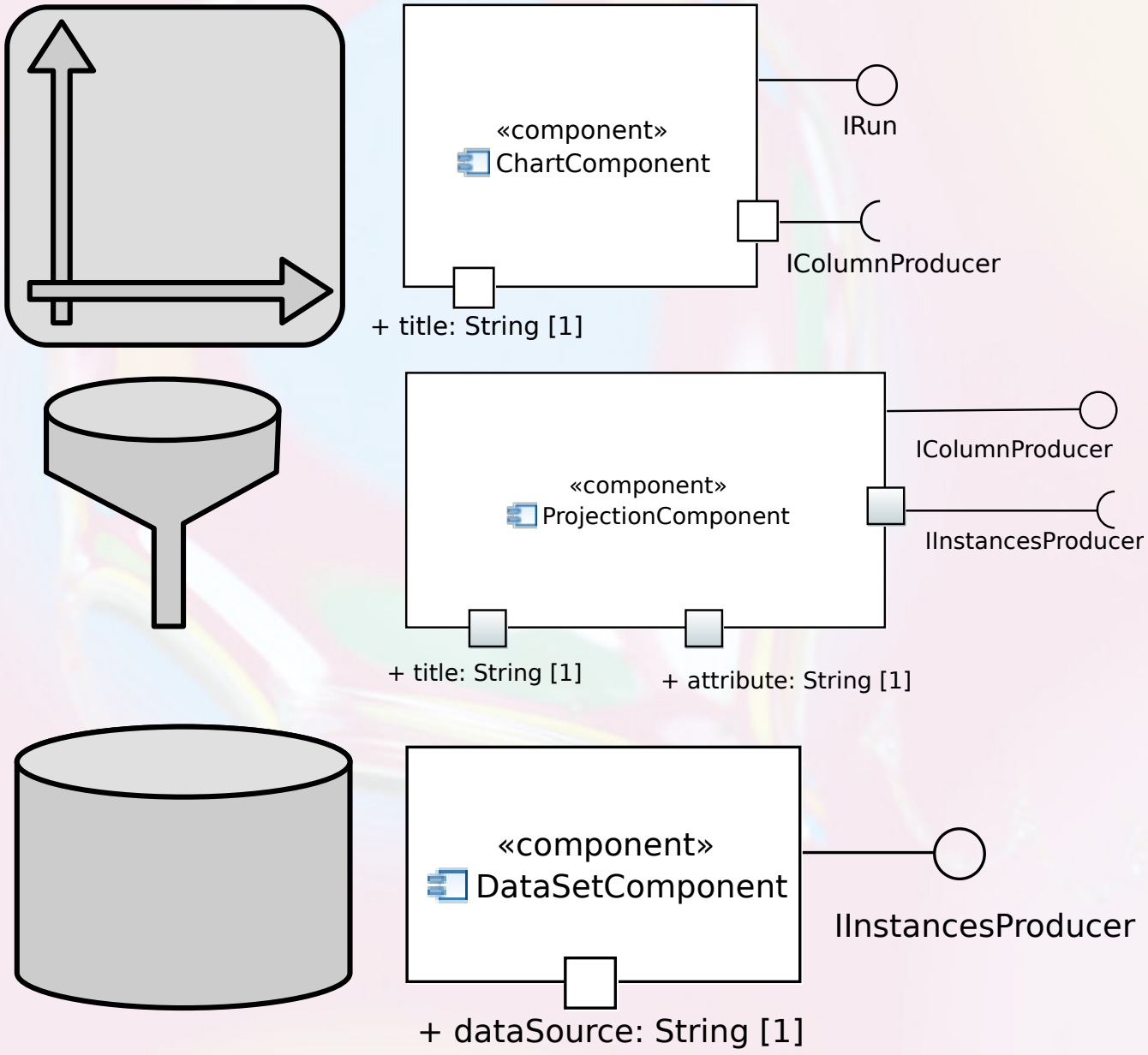
- Apresente um gráfico comparativo entre idade e tempo de recuperação, indicando os diferentes diagnósticos.

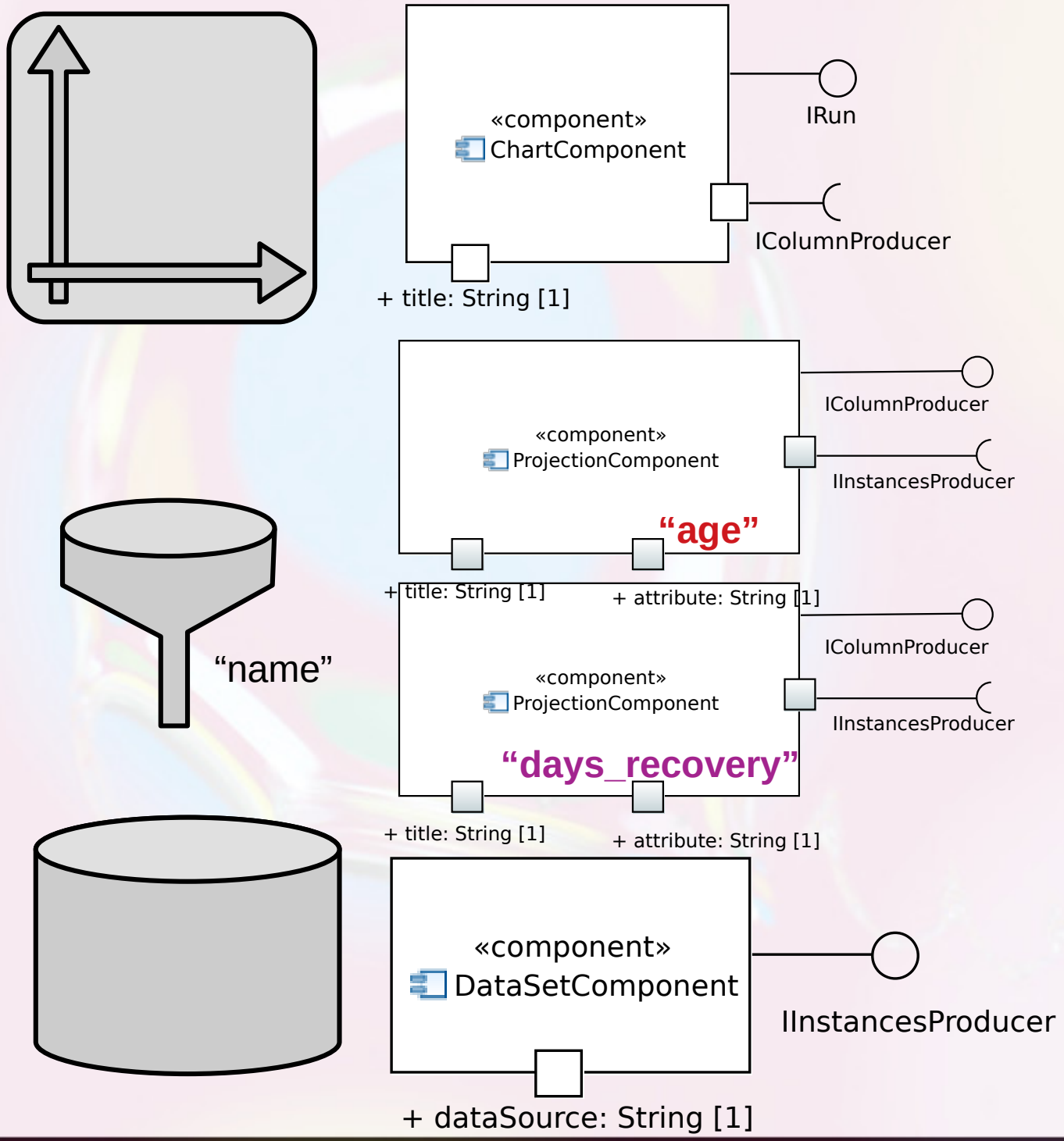
Componente Chart

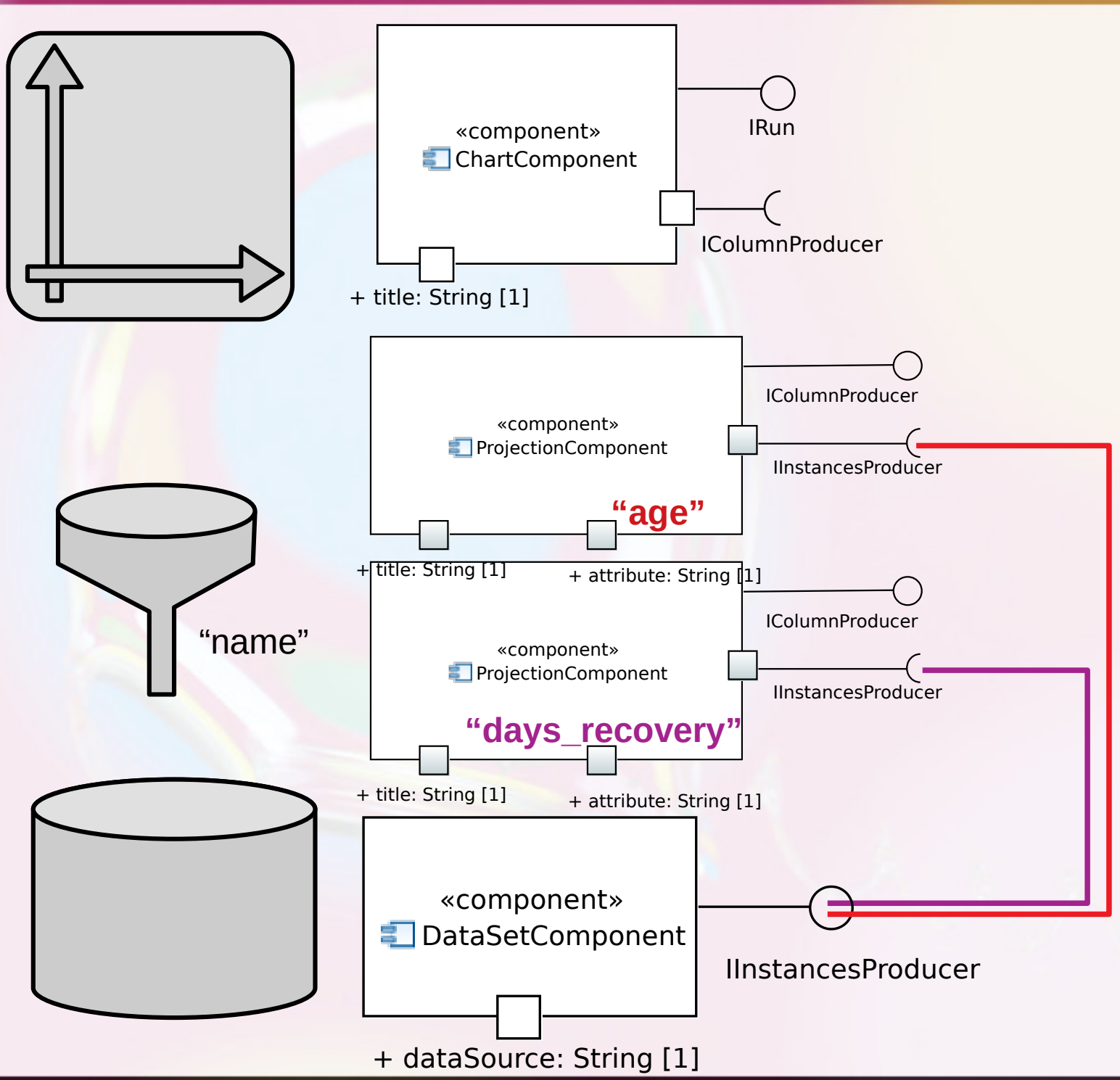


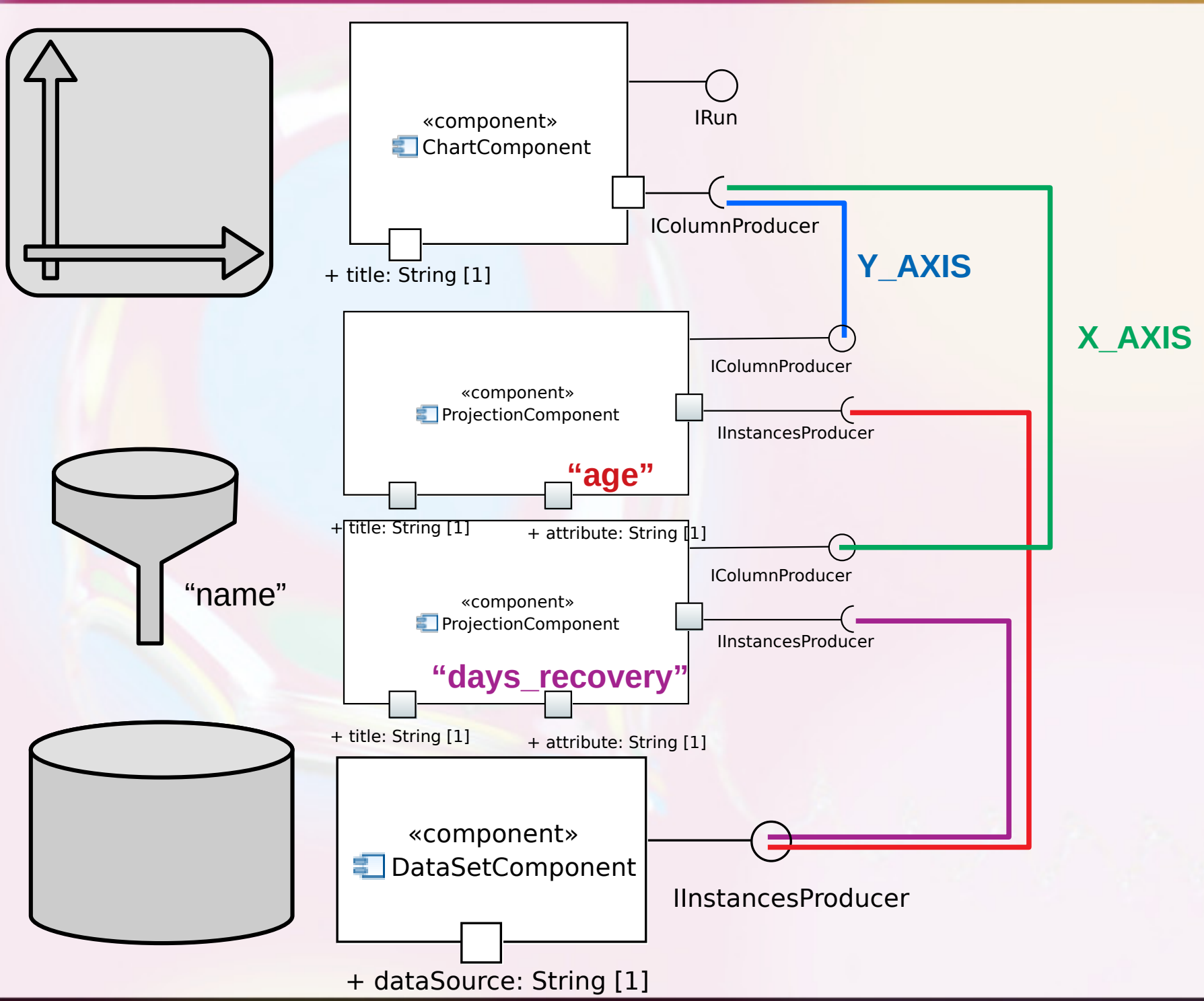


Duas Instâncias de um Componente




«component»
ChartComponent
IRun
IColumnProducer
+ title: String [1]
«component»
ProjectionComponent
IColumnProducer
IInstancesProducer
“age”
+ title: String [1]
+ attribute: String [1]
“name”
«component»
ProjectionComponent
IColumnProducer
IInstancesProducer
“days_recovery”
+ title: String [1]
+ attribute: String [1]
«component»
DataSetComponent
IInstancesProducer
+ dataSource: String [1]

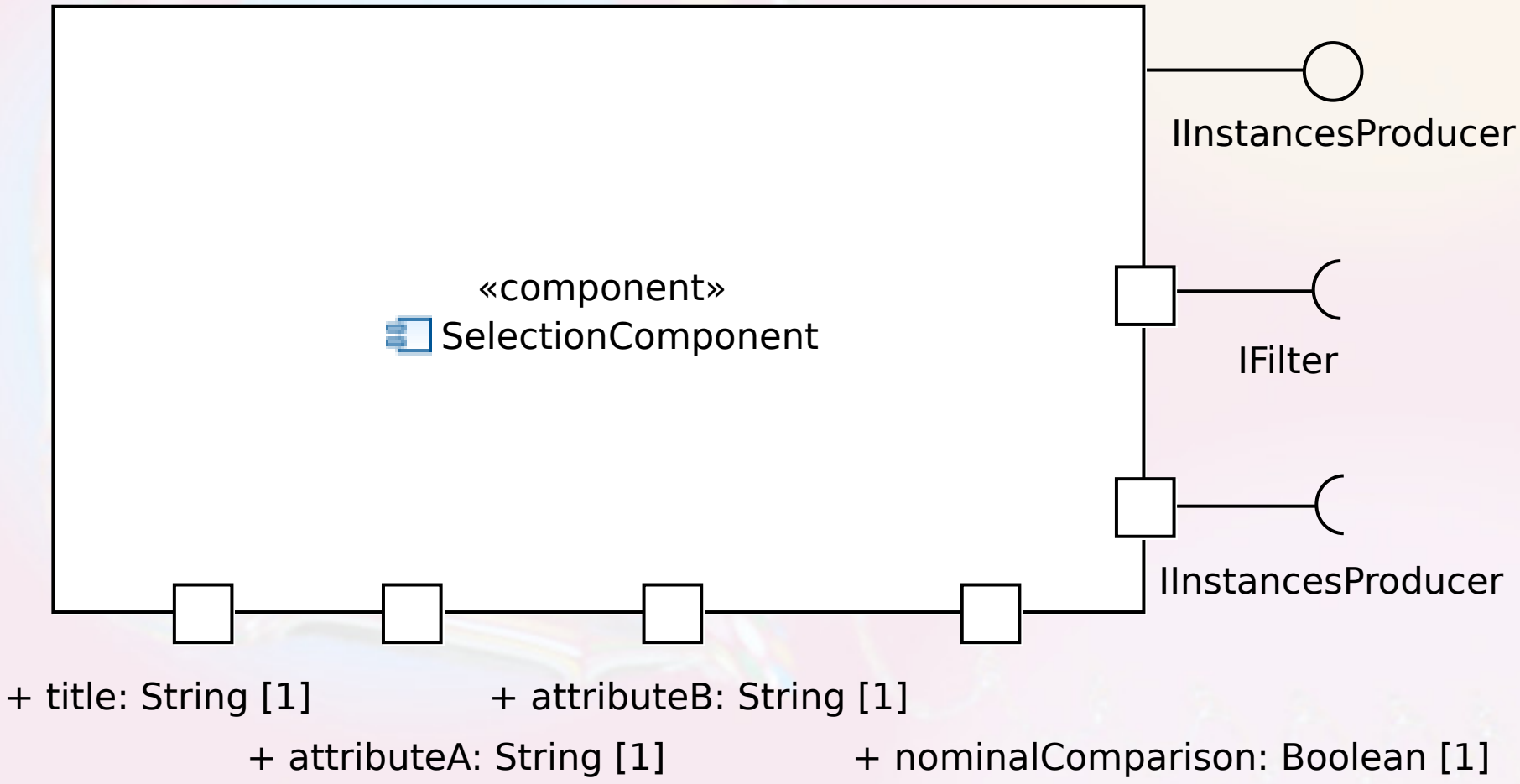

«component»
ChartComponent
IRun
IColumnProducer
+ title: String [1]
«component»
ProjectionComponent
IColumnProducer
IInstancesProducer
“age”
+ title: String [1]
+ attribute: String [1]
“name”
«component»
ProjectionComponent
IColumnProducer
IInstancesProducer
“days_recovery”
+ title: String [1]
+ attribute: String [1]
«component»
DataSetComponent
IInstancesProducer
+ dataSource: String [1]


«component»
ChartComponent
IRun
IColumnProducer
+ title: String [1]
Y_AXIS
X_AXIS
«component»
ProjectionComponent
IColumnProducer
IInstancesProducer
“age”
+ title: String [1]
+ attribute: String [1]
“name”
IColumnProducer
«component»
ProjectionComponent
IInstancesProducer
“days_recovery”
+ title: String [1]
+ attribute: String [1]
«component»
DataSetComponent
IInstancesProducer
+ dataSource: String [1]

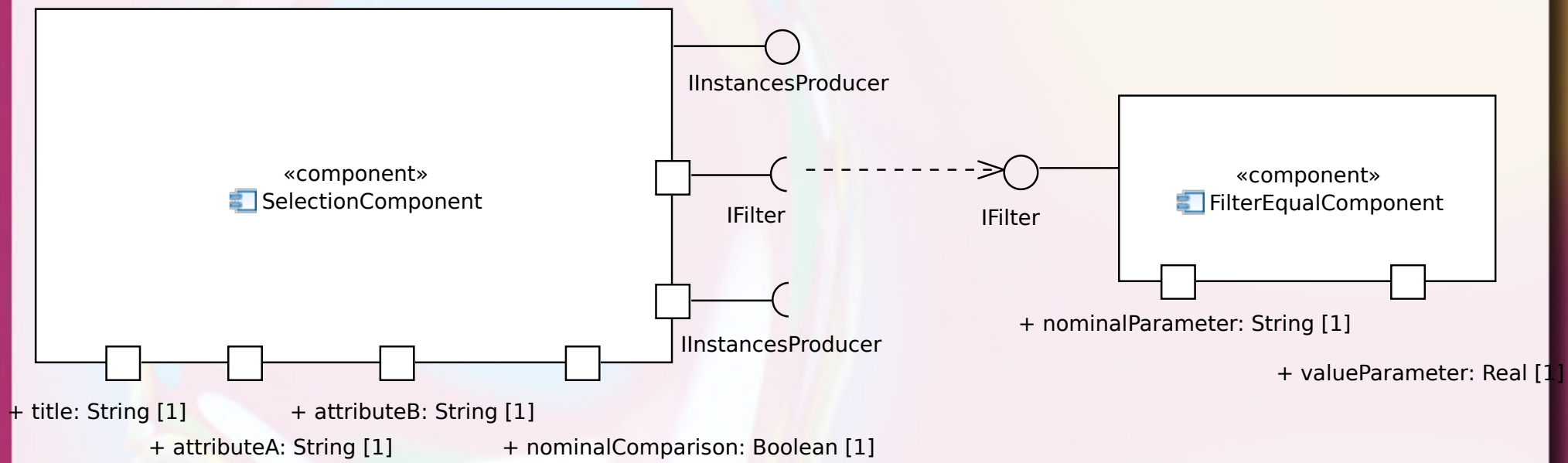
Exercício 5

- Apresente o DataSet filtrando a doença “bacterial_infection”.

Componente Selection



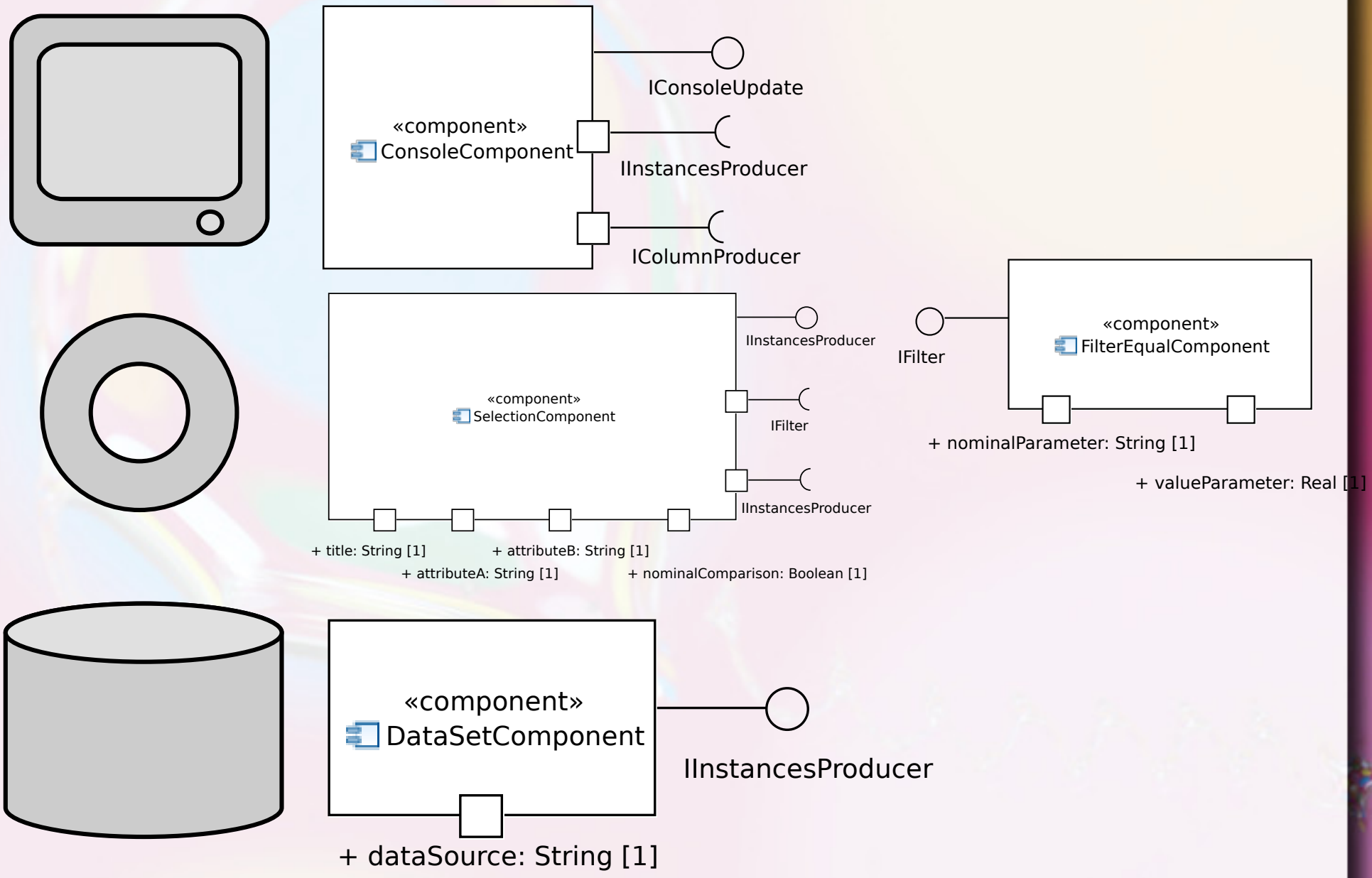
Componente Filter



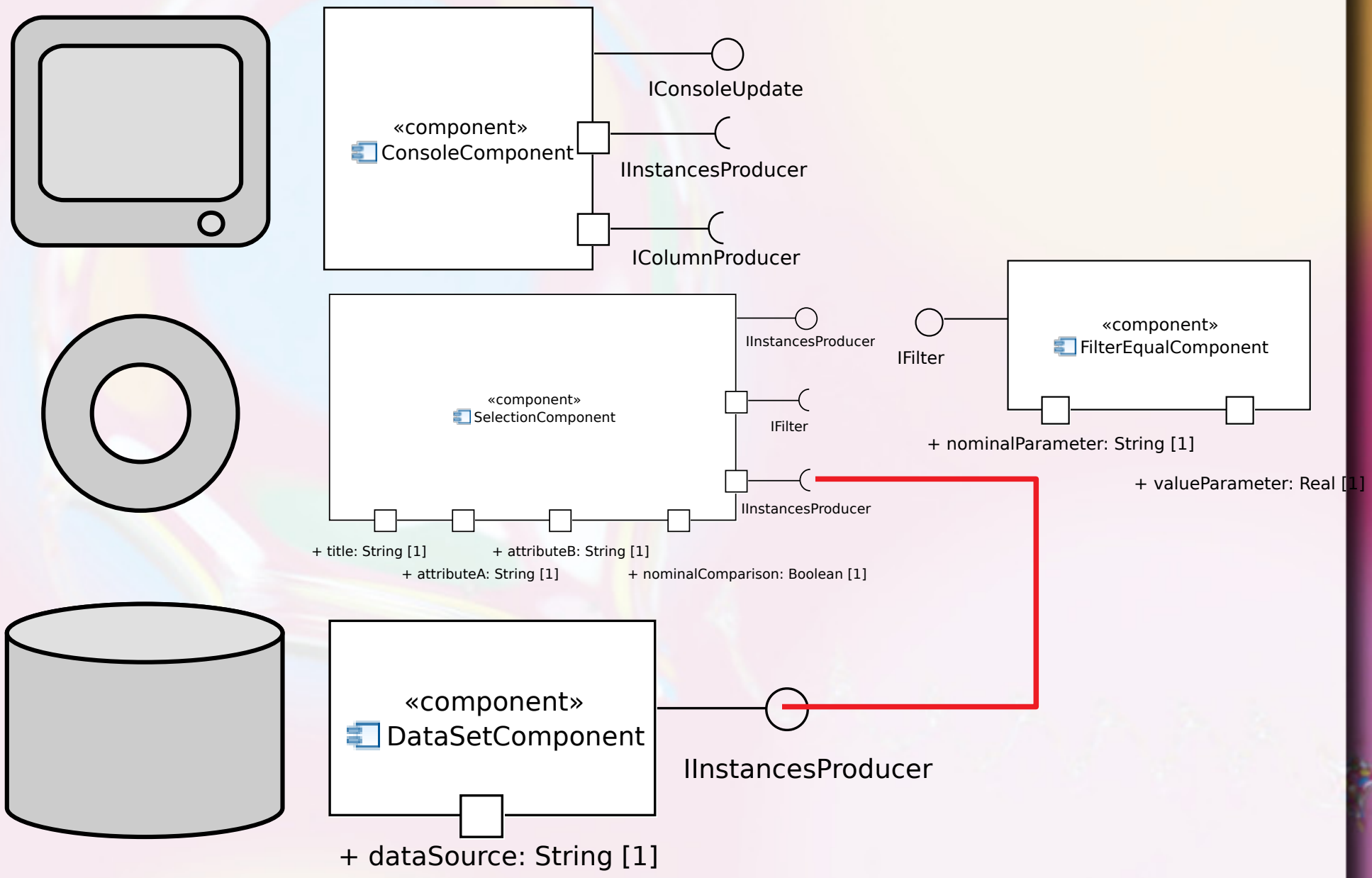
«Interface»
IFilter

+ filter(in toEvaluate: String): Boolean
+ filter(in toEvaluate: Real): Boolean
+ filter(in toCompare1: String, in toCompare2: String): Boolean
+ filter(in toCompare1: Real, in toCompare2: Real): Boolean

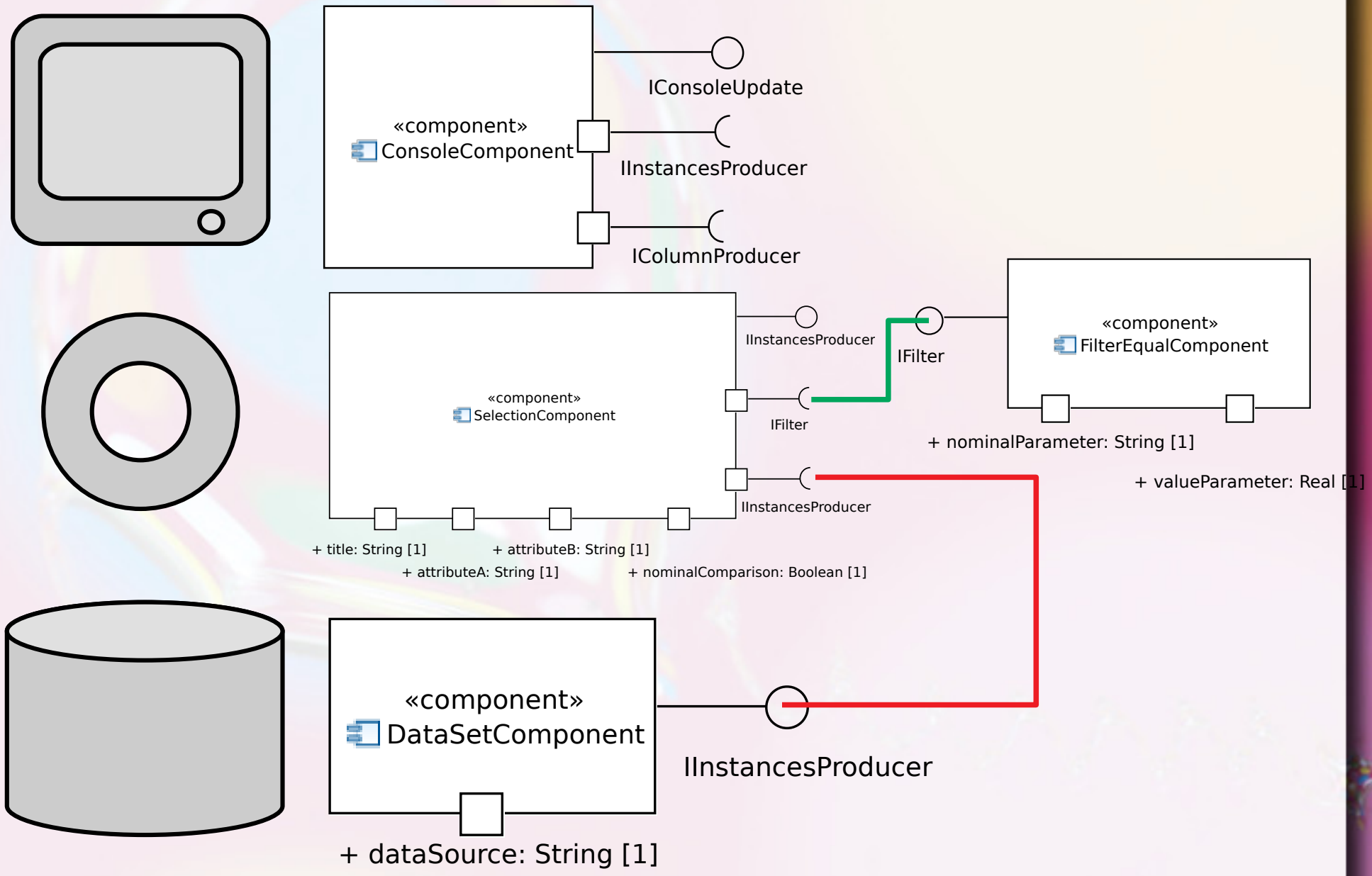
Conectando Quatro Componentes



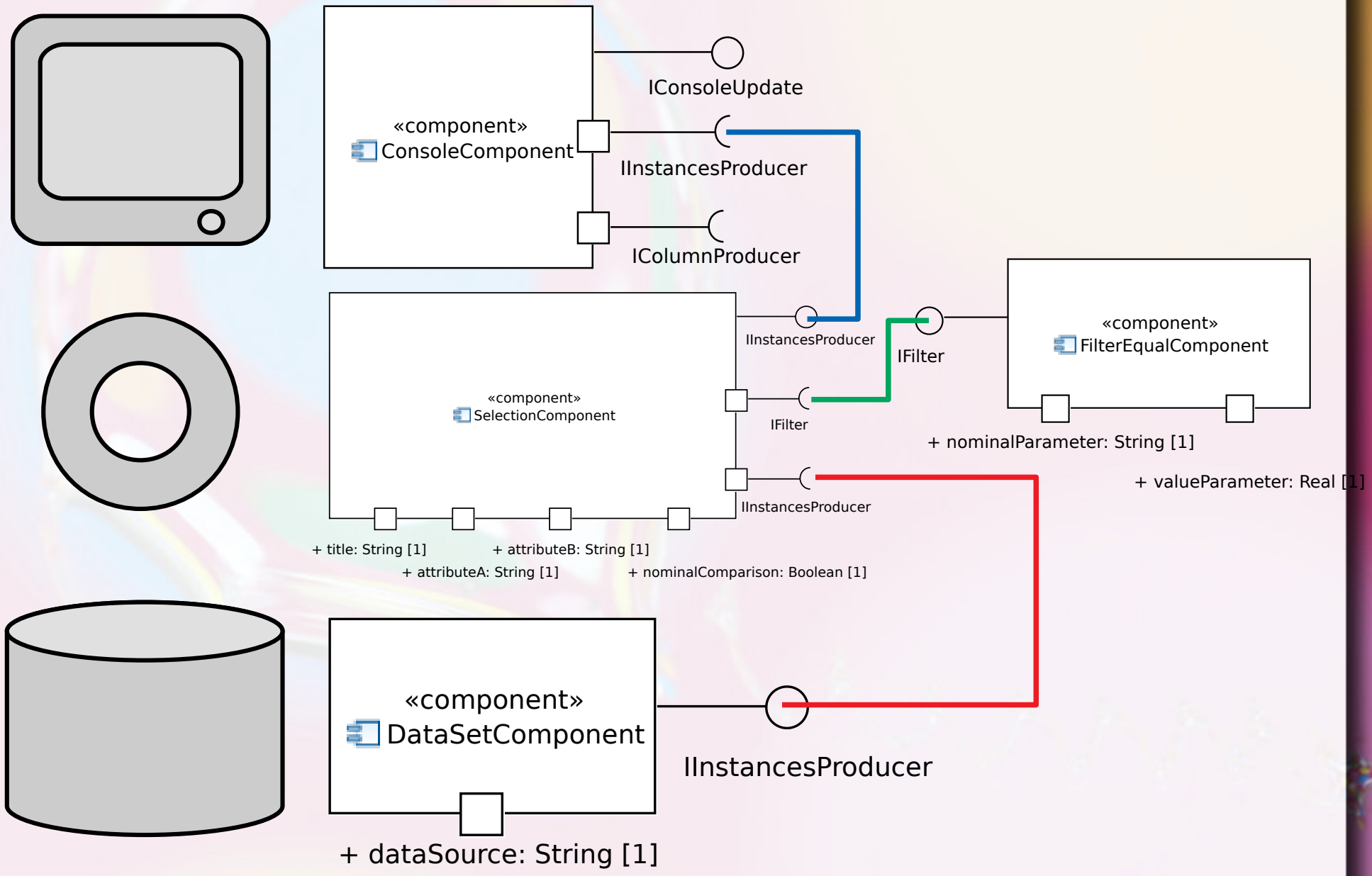
Conectando Quatro Componentes



Conectando Quatro Componentes

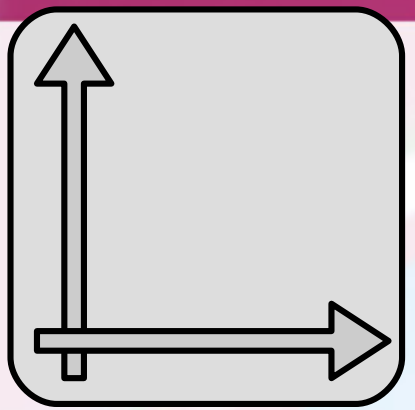


Conectando Quatro Componentes



Exercício 6

- Apresente um gráfico comparativo entre idade e tempo de recuperação, filtrando apenas a doença “bacterial_infection”.

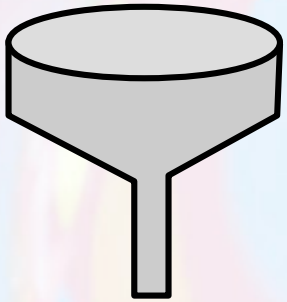


«component»
ChartComponent

IRun

IColumnProducer

+ title: String [1]



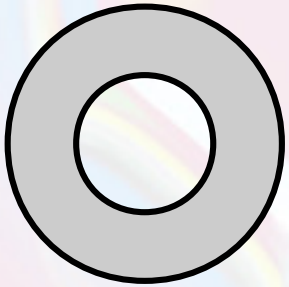
«component»
ProjectionComponent

IColumnProducer

IInstancesProducer

+ title: String [1]

+ attribute: String [1]



«component»
SelectionComponent

IInstancesProducer

IFilter

IInstancesProducer

+ title: String [1]

+ attributeB: String [1]

+ attributeA: String [1]

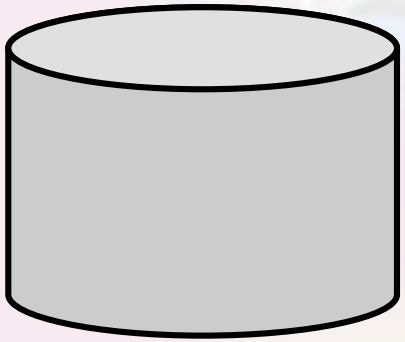
+ nominalComparison: Boolean [1]

IFilter

«component»
FilterEqualComponent

+ nominalParameter: String [1]

+ valueParameter: Real [1]



«component»
DataSetComponent

IInstancesProducer

+ dataSource: String [1]

Visão Externa

- Visão Externa (esta aula)
  - Foco: blackbox
  - Abstração das funcionalidades de um componente vendo-o externamente através de suas interfaces
  - Uso de componentes → Composição
- Visão Interna (próxima aula)
  - Foco: whitebox
  - Como um componente é implementado internamente

Referências

- Caires, Luis. **Fundamentos e Tecnologias de Componentes (slides)**. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2002.
- Cheesman, J., & Daniels, J. (2000). **UML Components: A simple process for specifying component-based software**. Addison-Wesley.
- Cook, S., Bock, C., Rivett, P., Rutt, T., Seidewitz, E., Selic, B., & Tolbert, D. (2015). **OMG Unified Modeling Language (OMG UML) - version 2.5**. Needham. Retrieved from http://www.omg.org/spec/UML/2.5/
- Szyperski, C. **Component Software: Beyond Object-Oriented Programming**. Addison-Wesley Longman Publishing Co., Inc., 2002.

Referências

- Comella-Dorda, S. **Component Object Model (COM), DCOM, and Related Capabilities**. Carnegie Mellon University, março de 2001.
- Cook, S., Bock, C., Rivett, P., Rutt, T., Seidewitz, E., Selic, B., & Tolbert, D. (2015). **OMG Unified Modeling Language (OMG UML) - version 2.5**. Needham. Retrieved from http://www.omg.org/spec/UML/2.5/
- Gamma, E. Helm, R. Johnson, R. Vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.
- Martin, R. C. **Design Principles and Design Patterns**. Object Mentor, 2000.
- Parrish, R. **XPCOM Part 1: An introduction to XPCOM**. DeveloperWorks, fevereiro de 2001, on-line: http://www.ibm.com/developerworks/webservices/library/co-xpcom.html
- Willliams, S. & Kindel, C. **The Component Object Model: A Technical Overview**. Microsoft Corporation, 1994

André Santanchè
http://www.ic.unicamp.br/~santanche

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link: http://creativecommons.org/licenses/by-nc-sa/3.0/

- Agradecimento a Doug Wheller [http://www.flickr.com/photos/doug88888/] por sua fotografia “Water drop” usada na capa e nos fundos, disponível em [http://www.flickr.com/photos/doug88888/7032440831/] vide licença específica da fotografia.