Projeto de arquitetura

David Garlan and Mary Shaw

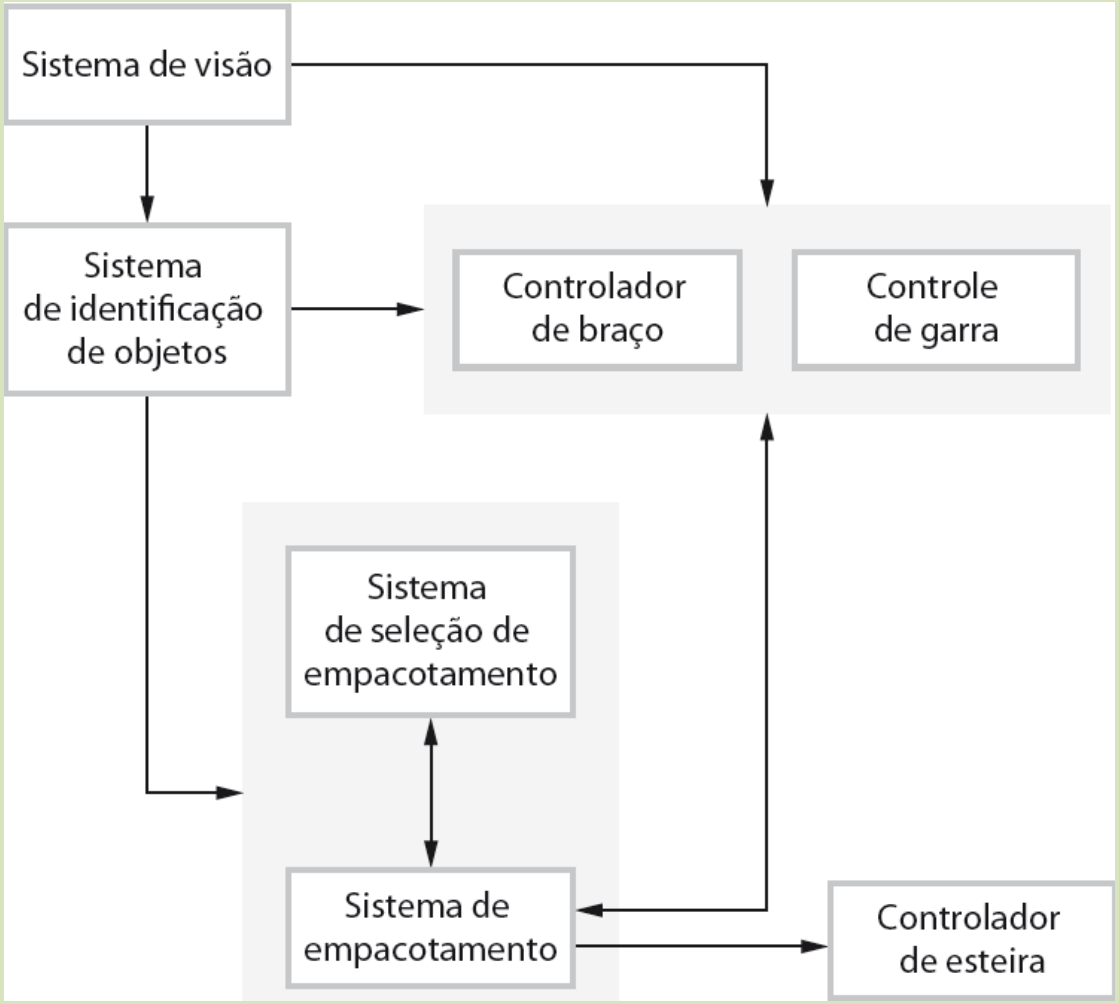
Arquiteturas de software

- O processo de projeto para identificar os **módulos** que compõem um sistema e suas inter-relações é o projeto de arquitetura.

- A arquitetura de um sistema específico pode ser vista como uma coleção de componentes computacionais — ou simplesmente **componentes** — juntamente com uma descrição das interações entre esses componentes — os **conectores**.

- **Conectores** podem representar interações tão variadas quanto chamadas de procedimento, transmissão de eventos, consultas a bancos de dados e dutos (pipes).

A arquitetura de um sistema de controle robotizado de empacotamento



Abstração sobre a arquitetura

- **Arquitetura em pequena escala** está preocupada com a arquitetura dos programas individuais.
  - Nesse nível, estamos preocupados com a maneira como um programa individual é decomposto em componentes.
- **Arquitetura em grande escala** preocupa-se com a arquitetura de sistemas corporativos complexos que incluem outros sistemas, programas e componentes do programa.
  - Esses sistemas empresariais estão distribuídos em diferentes computadores, que podem ser geridos por diferentes empresas.

Vantagens de se ter uma "arquitetura explícita"

- Comunicação de *stakeholders*
  - ✓ A arquitetura pode ser usada como um foco de discussão pelos *stakeholders* do sistema.

- Análise de sistemas
  - ✓ Podem ser feitas análises a respeito da possibilidade **do sistema atender a requisitos não-funcionais**.

- Reuso em larga escala
  - ✓ Pode-se reusar não apenas código, fonte ou classes, mas toda a arquitetura do sistema.

Decisões Importantes durante o Projeto da Arquitetura

- Existe uma arquitetura genérica de aplicação que possa ser usada?
- Como o sistema será distribuído?
- Quais **estilos de arquitetura** são apropriados?
- Como o sistema pode ser decomposto em módulos?
- Qual estratégia de fluxo de controle deve ser usada?
- Como o projeto de arquitetura será **avaliado**?
- Como a arquitetura deve ser **documentada**?

Reúso de arquitetura

- Muitas vezes sistemas de mesmo domínio têm arquiteturas similares que refletem os **conceitos do domínio.**
  - **Exemplo : Sistemas Adaptativos, Gerenciamento de Recursos de Negócio, etc**

- A arquitetura de um sistema pode ser projetada em torno de um ou mais padrões ou "**estilos** " de arquitetura.

  ✓ Esses estilos capturam a essência de uma arquitetura e podem ser instanciados de diferentes maneiras.

Atributos de Qualidade
x
Arquitetura do Sistema

- Desempenho
  - ✓ Agrupe operações críticas e **minimize as comunicações**. Use **componentes de alta granularidade ao invés de baixa granularidade**.
- Proteção
  - ✓ Nas camadas internas, use uma arquitetura em camadas que encapsula dados críticos.
- Segurança
  - ✓ Concentre atributos de segurança crítica em um pequeno número de subsistemas.
- Disponibilidade
  - ✓ Incluem **componentes redundantes** e mecanismos de tolerância a defeitos.
- Manutenibilidade
  - ✓ Use **componentes de alta coesão e baixo acoplamento**

Visões de Arquitetura

- Que pontos de vista ou perspectivas são úteis ao fazer o projeto e documentar a arquitetura de um sistema?

- Quais notações devem ser usadas para descrever os modelos de arquitetura?

  - ✓ Cada modelo de arquitetura mostra **apenas um ponto de vista** ou **perspectiva** do sistema.

  - ✓ Visões arquiteturais podem mostrar:
    - ✓ como um sistema é **decomposto em módulos**,
    - ✓ como os processos **interagem** em **tempo de execução** ou
    - ✓ diferentes formas em que os **componentes do sistema são distribuídos** em uma rede.

Visões de Arquitetura

- Uma visão **lógica**, que mostra as principais abstrações do sistema.
- Uma visão de **processo**, que mostra como o sistema funciona em **tempo de execução**
- Uma visão de **desenvolvimento**, que mostra como o sistema é decomposto durante o desenvolvimento, de forma que cada equipe se concentre em uma parte
- Uma visão **física**, que mostra o hardware do sistema e como os componentes do software são distribuídos entre os processadores do sistema.

Padrões de arquitetura ou Estilos Arquiteturais

- Padrões são um meio de representar, partilhar e reusar conhecimento.

- Um padrão de arquitetura é uma descrição das boas práticas de projeto, que tem sido experimentadas e testadas em diferentes ambientes.

Padrões de arquitetura ou Estilos Arquiteturais

- Um estilo arquitetural define uma família de sistemas em termos de um padrão de organização estrutural.

- Mais especificamente, um estilo arquitetural determina o vocabulário de **componentes** e **conectores** que podem ser usados em instâncias desse estilo, juntamente com um **conjunto de restrições** sobre como eles podem ser combinados. Essas restrições podem incluir **restrições topológicas** nas descrições arquiteturais (por exemplo, ausência de ciclos).

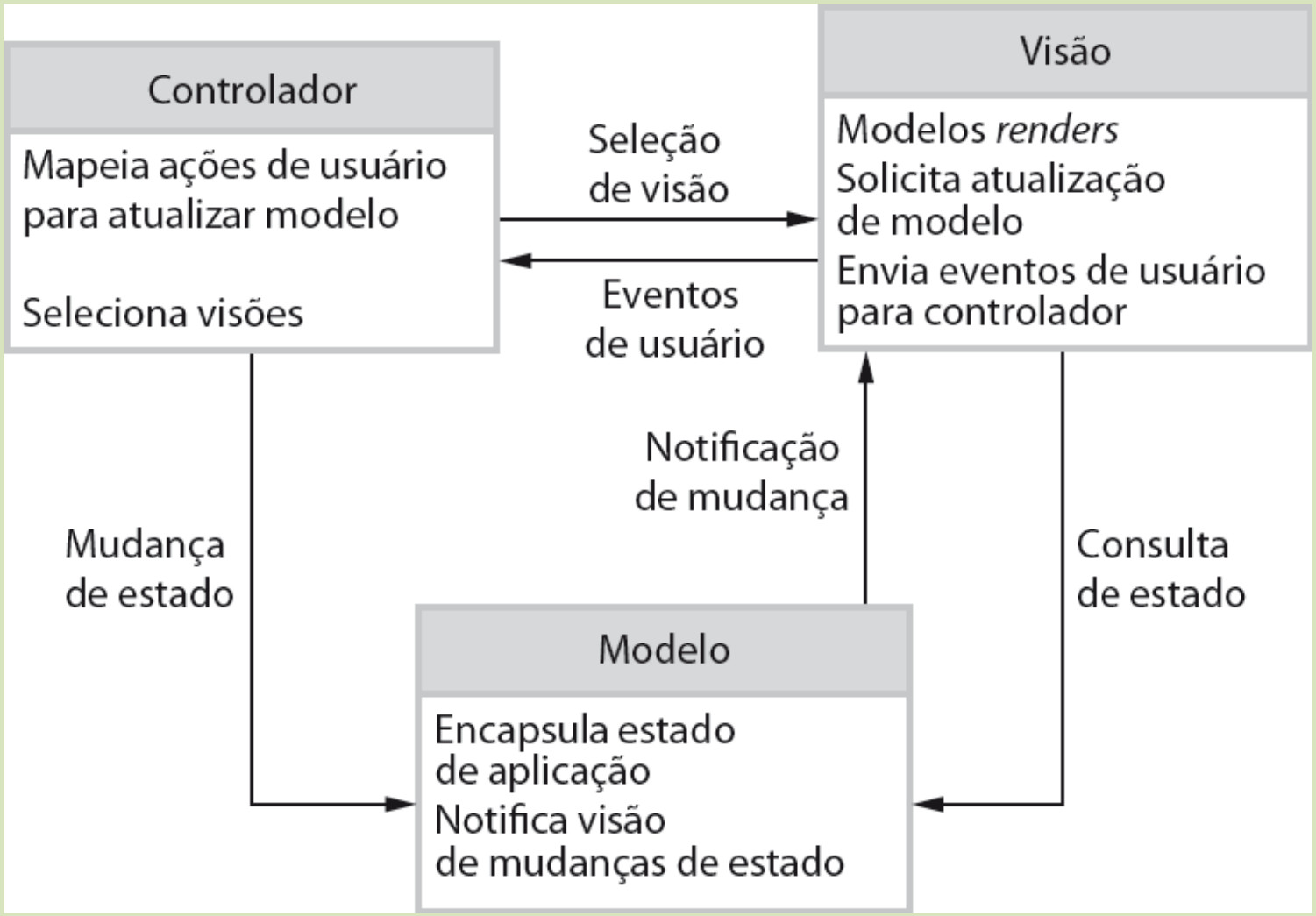
Padrões de arquitetura ou
Estilos Arquiteturais

- O que é o padrão estrutural?
- Quais são os invariantes essenciais de um estilo?
- Quais são as vantagens e desvantagens de usar um estilo específico?

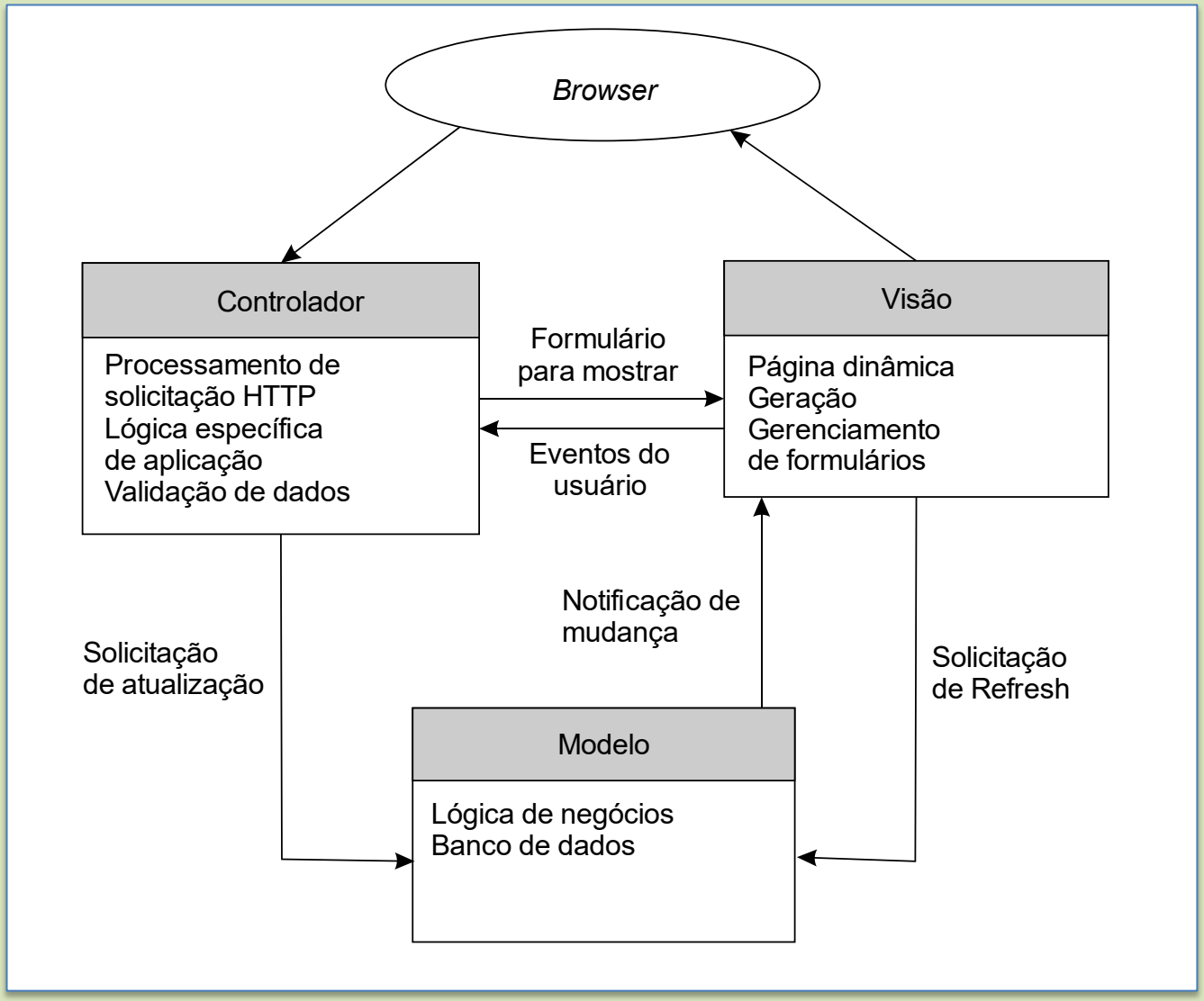
O padrão do Modelo-Visão-Controlador (MVC)

Nome	MVC (Modelo-Visão-Controlador)
Descrição	Separa a apresentação e a interação dos dados do sistema. O sistema é estruturado em três componentes lógicos que interagem entre si. O componente Modelo gerencia o sistema de dados e as operações associadas a esses dados. O componente Visão define e gerencia como os dados são apresentados ao usuário. O componente Controlador gerencia a interação do usuário (por exemplo, teclas, cliques do mouse etc.) e passa essas interações para a Visão e o Modelo. Veja a Figura 6.2.
Exemplo	A Figura 6.3 mostra a arquitetura de um sistema aplicativo baseado na Internet, organizado pelo uso do padrão MVC.
Quando é usado	É usado quando existem várias maneiras de se visualizar e interagir com dados. Também quando são desconhecidos os futuros requisitos de interação e apresentação de dados.
Vantagens	Permite que os dados sejam alterados de forma independente de sua representação, e vice-versa. Apoia a apresentação dos mesmos dados de maneiras diferentes, com as alterações feitas em uma representação aparecendo em todas elas.
Desvantagens	Quando o modelo de dados e as interações são simples, pode envolver código adicional e complexidade de código.

A organização do MVC



Exemplo de arquitetura MVC



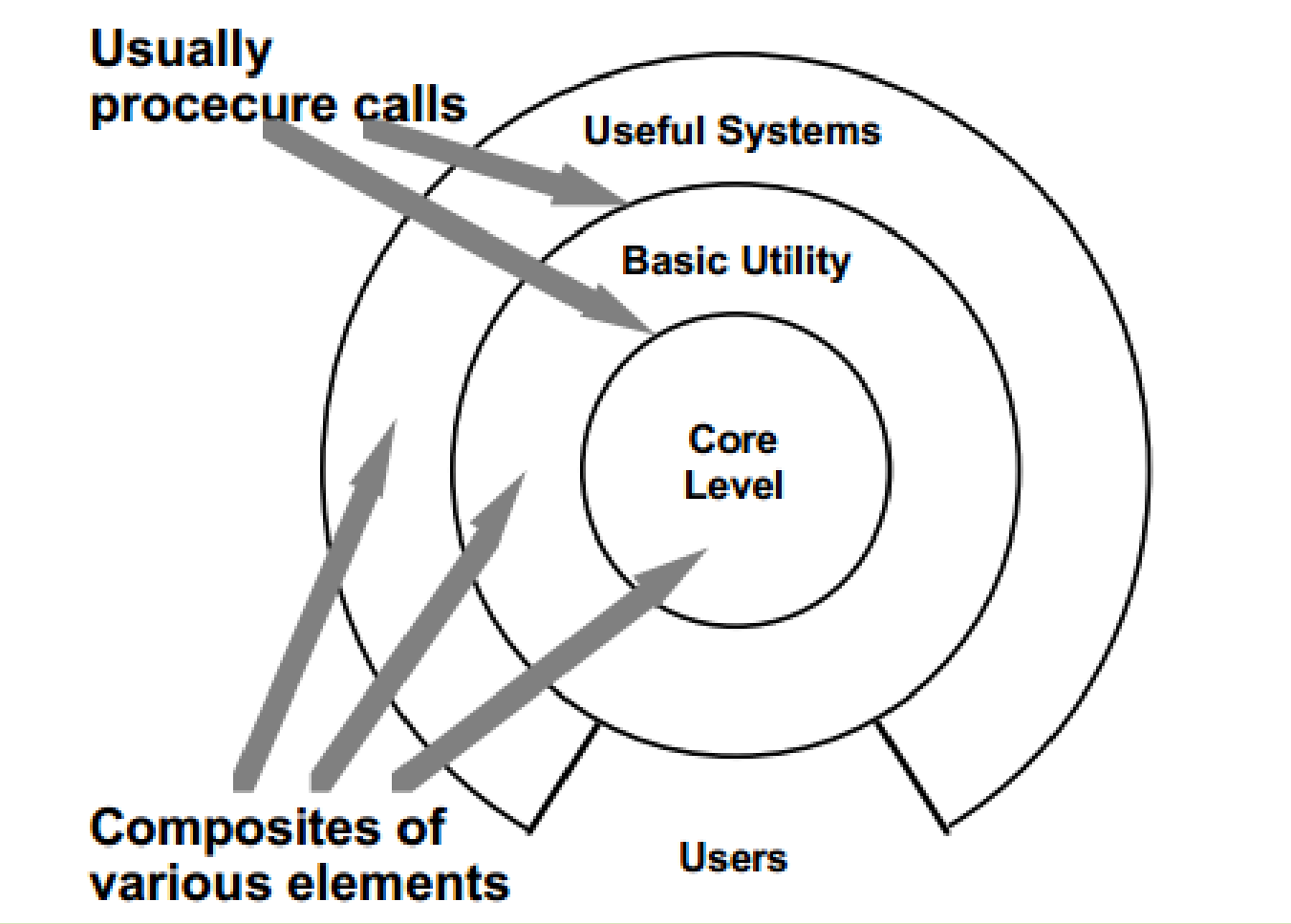
Arquitetura em camadas

- Um sistema em camadas é organizado hierarquicamente, com cada camada fornecendo serviços para a camada acima dela e servindo como cliente para a camada abaixo.

- As restrições topológicas incluem a limitação de interações às camadas adjacentes.

- Apoia o desenvolvimento incremental de subsistemas. Quando a interface de uma camada muda, apenas a camada adjacente é afetada.

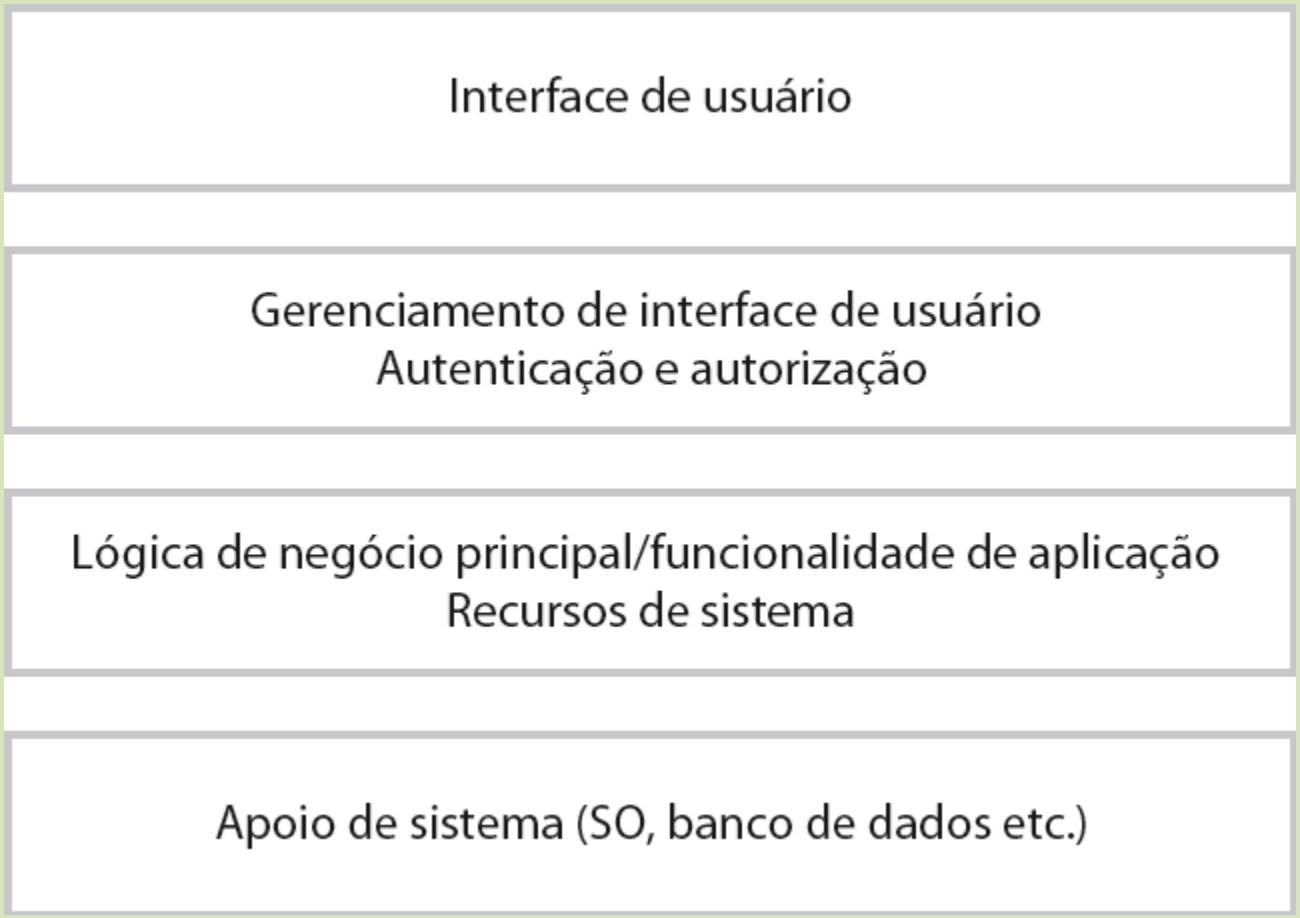
Arquitetura em camadas



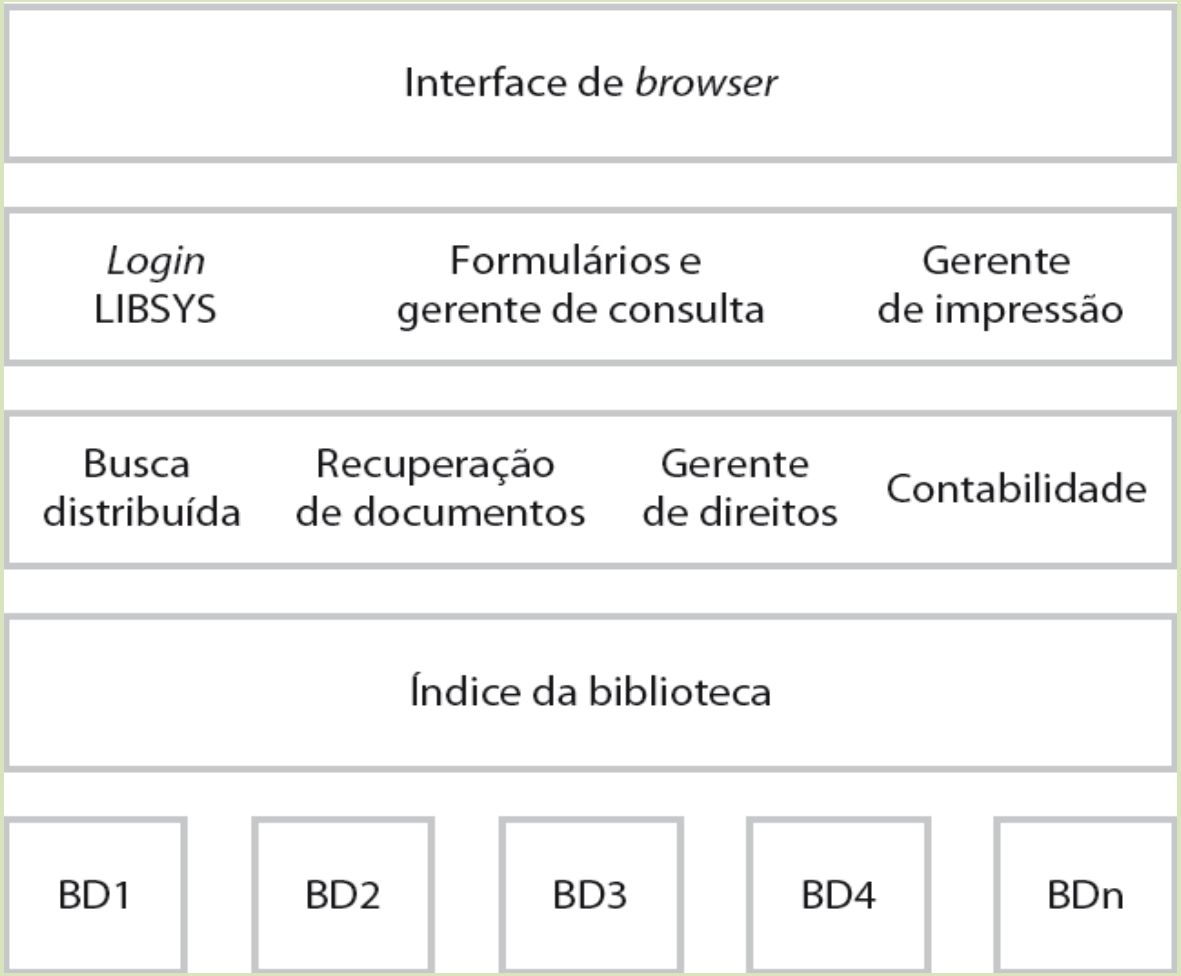
O padrão de arquitetura em camadas

Nome	Arquitetura em camadas
Descrição	Organiza o sistema em camadas com a funcionalidade relacionada associada a cada camada. Uma camada fornece serviços à camada acima dela; assim, os níveis mais baixos de camadas representam os principais serviços suscetíveis de serem usados em todo o sistema. Veja a Figura 6.4.
Exemplo	Um modelo em camadas de um sistema para compartilhar documentos com direitos autorais, em bibliotecas diferentes, como mostrado na Figura 6.5.
Quando é usado	É usado na construção de novos recursos em cima de sistemas existentes; quando o desenvolvimento está espalhado por várias equipes, com a responsabilidade de cada equipe em uma camada de funcionalidade; quando há um requisito de proteção multinível.
Vantagens	Desde que a interface seja mantida, permite a substituição de camadas inteiras. Recursos redundantes (por exemplo, autenticação) podem ser fornecidos em cada camada para aumentar a confiança do sistema.
Desvantagens	Na prática, costuma ser difícil proporcionar uma clara separação entre as camadas, e uma camada de alto nível pode ter de interagir diretamente com camadas de baixo nível, em vez de através da camada imediatamente abaixo dela. O desempenho pode ser um problema por causa dos múltiplos níveis de interpretação de uma solicitação de serviço, uma vez que são processados em cada camada.

Uma arquitetura genérica em camadas



A arquitetura do sistema LIBSYS



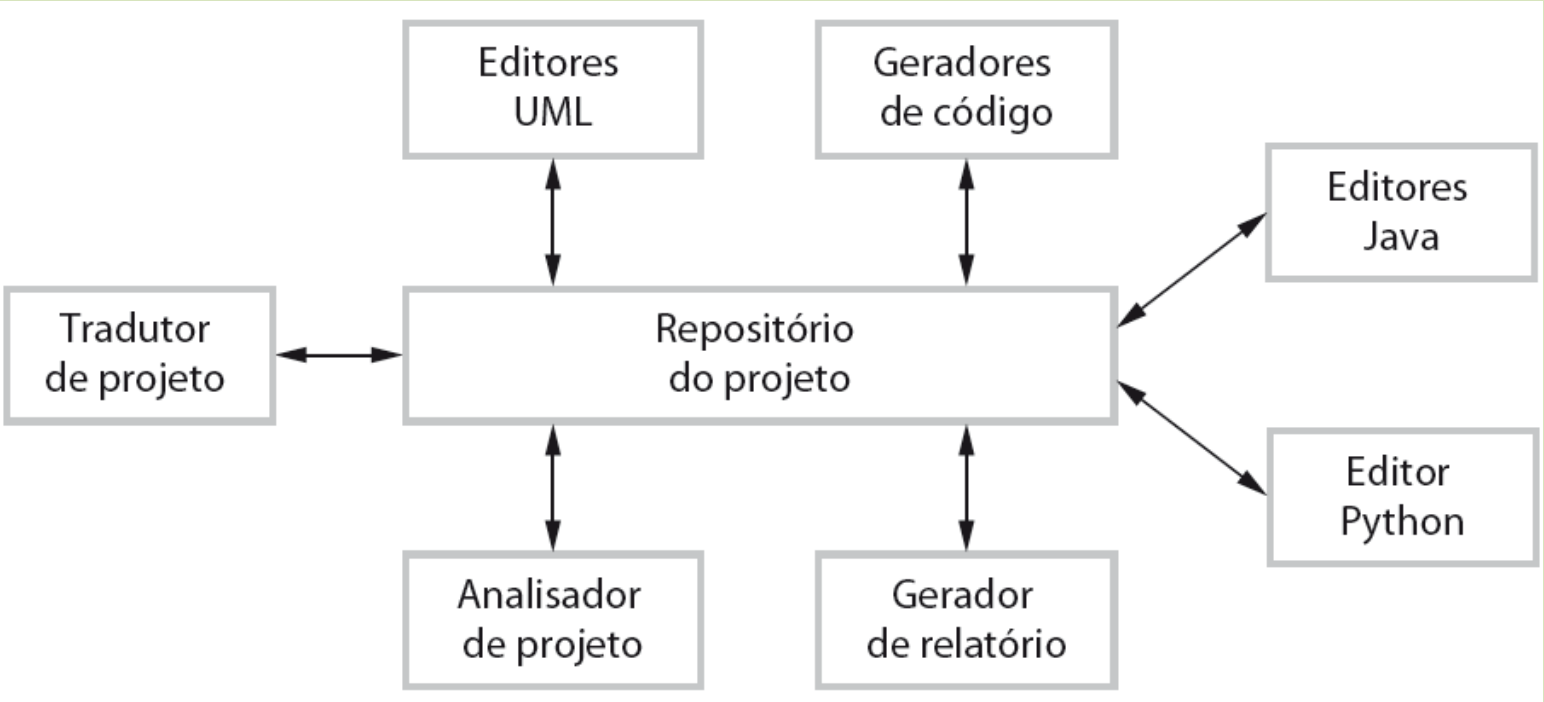
Arquitetura de repositório

- Subsistemas devem trocar dados. O que pode ser feito de duas maneiras:
  - ✓ Dados compartilhados são guardados em um repositório central e podem ser acessados por todos os subsistemas;
  - ✓ Cada subsistema mantém seu próprio repositório e transmite dados explicitamente para outros subsistemas.
- Quando grandes quantidades de dados devem ser compartilhadas, é mais comum o uso do modelo de repositório compartilhado pois esse é um eficiente mecanismo de compartilhamento de dados.

O padrão Repositório

Nome	Repositório
Descrição	Todos os dados em um sistema são gerenciados em um repositório central, acessível a todos os componentes do sistema. Os componentes não interagem diretamente, apenas por meio do repositório.
Exemplo	A Figura 6.6 é um exemplo de um IDE em que os componentes usam um repositório de informações sobre projetos de sistema. Cada ferramenta de software gera informações que ficam disponíveis para uso por outras ferramentas.
Quando é usado	Você deve usar esse padrão quando tem um sistema no qual grandes volumes de informações são gerados e precisam ser armazenados por um longo tempo. Você também pode usá-lo em sistemas dirigidos a dados, nos quais a inclusão dos dados no repositório dispara uma ação ou ferramenta.
Vantagens	Os componentes podem ser independentes — eles não precisam saber da existência de outros componentes. As alterações feitas a um componente podem propagar-se para todos os outros. Todos os dados podem ser gerenciados de forma consistente (por exemplo, *backups* feitos ao mesmo tempo), pois tudo está em um só lugar.
Desvantagens	O repositório é um ponto único de falha, assim, problemas no repositório podem afetar todo o sistema. Pode haver ineficiências na organização de toda a comunicação através do repositório. Distribuir o repositório através de vários computadores pode ser difícil.

Uma arquitetura de repositório para um IDE



Arquitetura cliente-servidor

- É um modelo de sistema distribuído que mostra como os **dados** e **processamento** são distribuídos através de uma série de componentes.

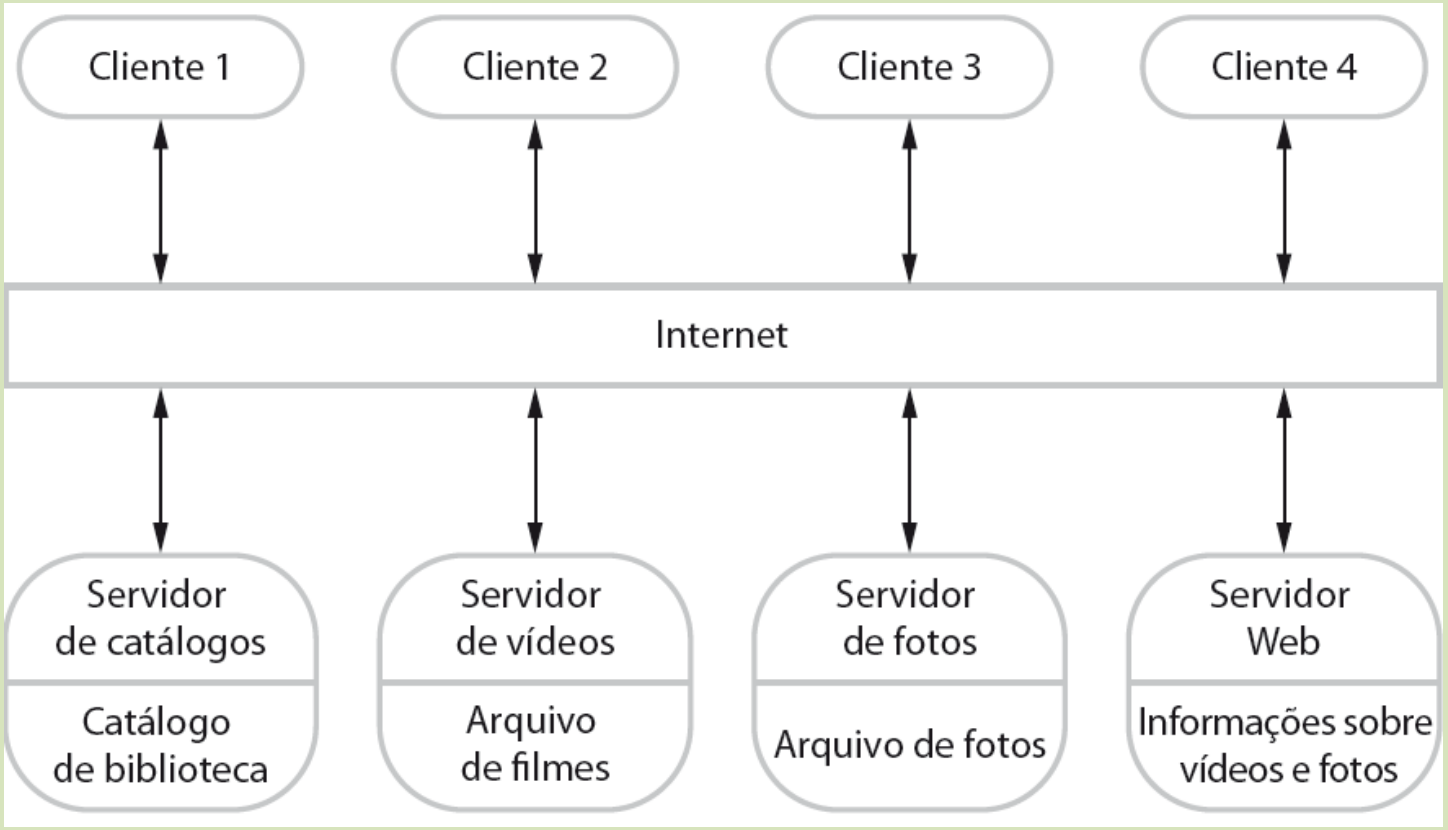
  ✓ Pode ser implementado em um único computador.

- Especificações:
  - Um conjunto de servidores autônomos que prestam serviços específicos, tais como impressão, gerenciamento de dados, etc.
  - Um conjunto de clientes que solicitam estes serviços.
  - Rede que permite aos clientes acessar os servidores.
  - Chamadas de métodos remotos (CORBA, web-services)

O padrão cliente-servidor

Nome	Cliente-servidor
Descrição	Em uma arquitetura cliente-servidor, a funcionalidade do sistema está organizada em serviços — cada serviço é prestado por um servidor. Os clientes são os usuários desses serviços e acessam os servidores para fazer uso deles.
Exemplo	A Figura 6.7 é um exemplo de uma biblioteca de filmes e vídeos/DVDs, organizados como um sistema cliente-servidor.
Quando é usado	É usado quando os dados em um banco de dados compartilhado precisam ser acessados a partir de uma série de locais. Como os servidores podem ser replicados, também pode ser usado quando a carga em um sistema é variável.
Vantagens	A principal vantagem desse modelo é que os servidores podem ser distribuídos através de uma rede. A funcionalidade geral (por exemplo, um serviço de impressão) pode estar disponível para todos os clientes e não precisa ser implementada por todos os serviços.
Desvantagens	Cada serviço é um ponto único de falha suscetível a ataques de negação de serviço ou de falha do servidor. O desempenho, bem como o sistema, pode ser imprevisível, pois depende da rede. Pode haver problemas de gerenciamento se os servidores forem propriedade de diferentes organizações.

A arquitetura cliente-servidor para uma biblioteca de filmes



Arquitetura de duto e filtro

- Em um estilo de pipe (duto) e filtro, cada componente possui um conjunto de entradas e um conjunto de saídas.

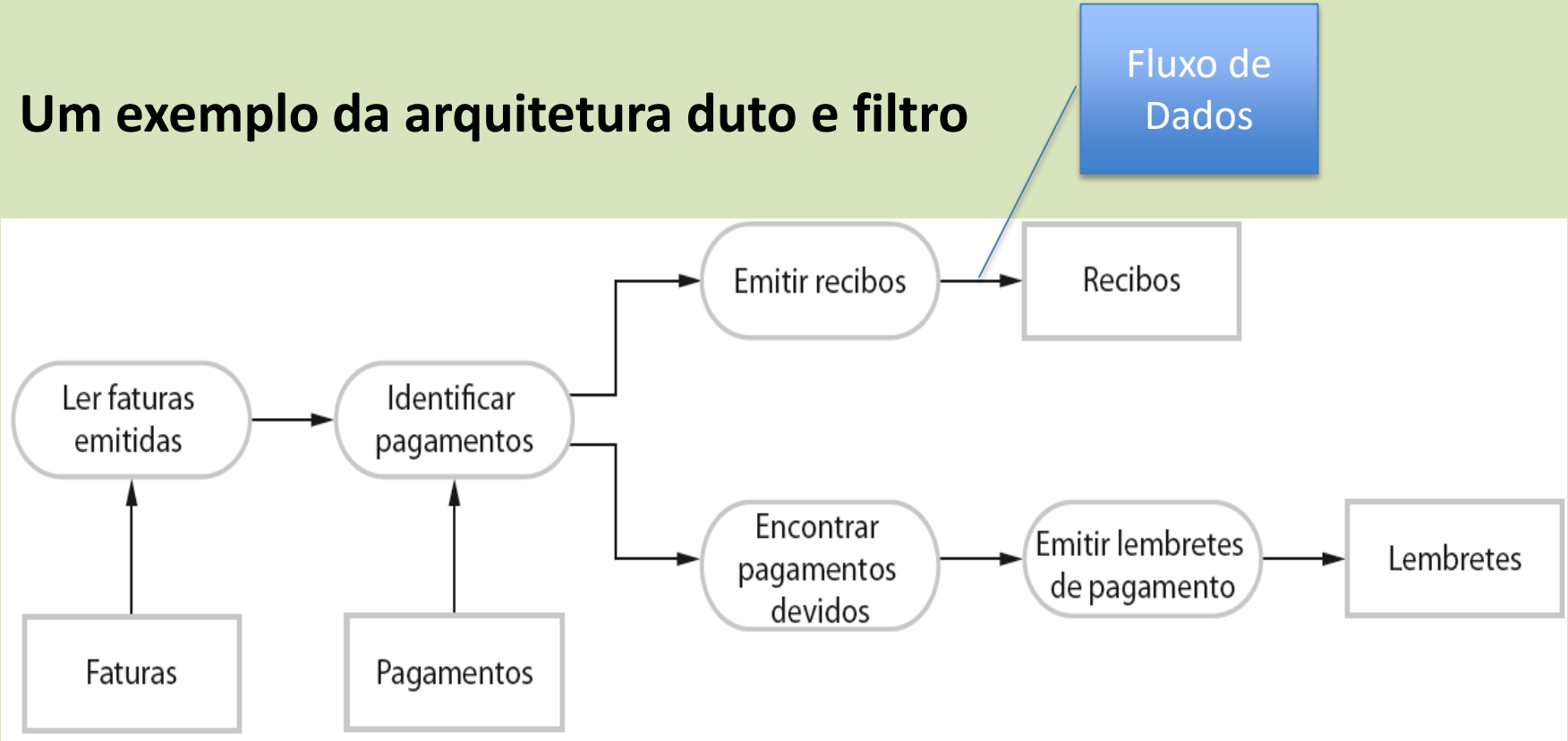
- Os **conectores** desse estilo servem como condutores para os fluxos, transmitindo as saídas de um filtro para as entradas de outro. Por isso, os conectores são chamados de 'pipes'.

- Invariantes:
  - Os filtros devem ser entidades independentes; em particular, não devem compartilhar estado com outros filtros.
  - Os filtros não conhecem a identidade de seus filtros upstream (anteriores) e downstream (posteriores).
  - Suas especificações podem restringir o que aparece nos pipes de entrada ou oferecer garantias sobre o que aparece nos pipes de saída.

O padrão duto e filtro

Nome	Duto e filtro
Descrição	O processamento dos dados em um sistema está organizado de modo que cada componente de processamento (filtro) seja discreto e realize um tipo de transformação de dados. Os dados fluem (como em um duto) de um componente para outro para processamento.
Exemplo	A Figura 6.8 é um exemplo de um sistema de duto e filtro usado para o processamento das faturas.
Quando é usado	Comumente, é usado em aplicações de processamento de dados (tanto as baseadas em lotes como as baseadas em transações) em que as entradas são processadas em etapas separadas para gerarem saídas relacionadas.
Vantagens	O reúso da transformação é de fácil compreensão e suporte. Estilo de *workflow* corresponde à estrutura de muitos processos de negócios. Evolução por adição de transformações é simples. Pode ser implementado tanto como um sistema sequencial quanto concorrente.
Desvantagens	O formato para transferência de dados tem de ser acordado entre as transformações de comunicação. Cada transformação deve analisar suas entradas e gerar as saídas para um formato acordado. Isso aumenta o *overhead* do sistema e pode significar a impossibilidade de reúso de transformações funcionais que usam estruturas incompatíveis de dados.

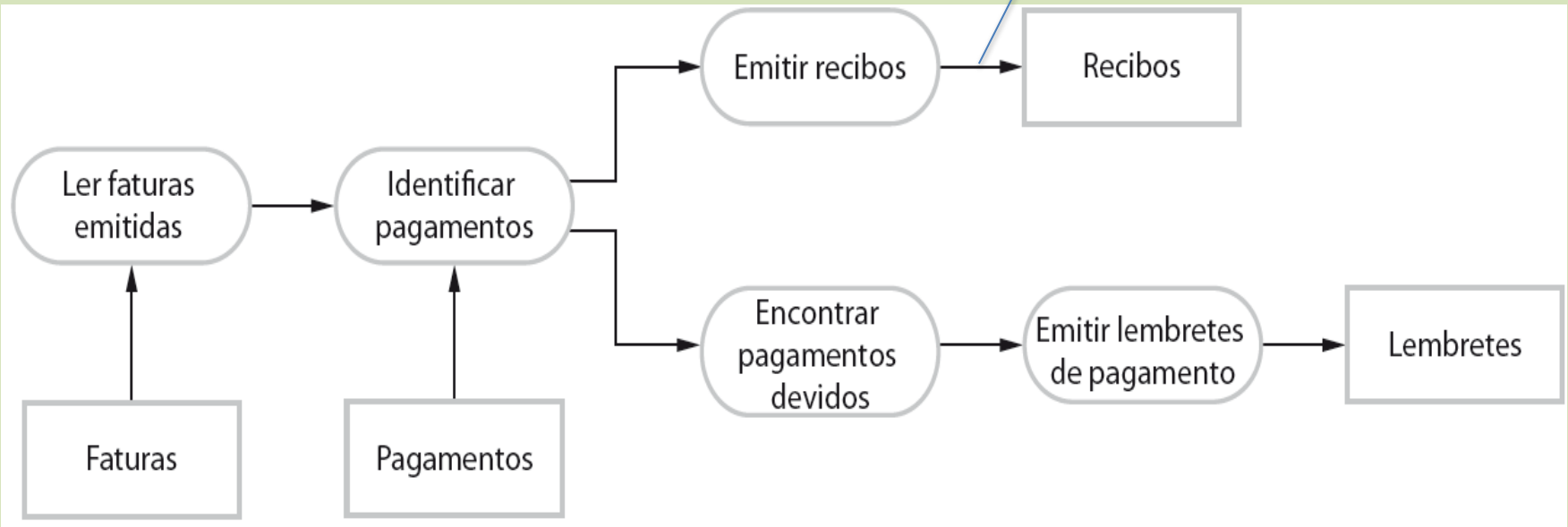
Um exemplo da arquitetura duto e filtro



As especializações comuns desse estilo incluem:

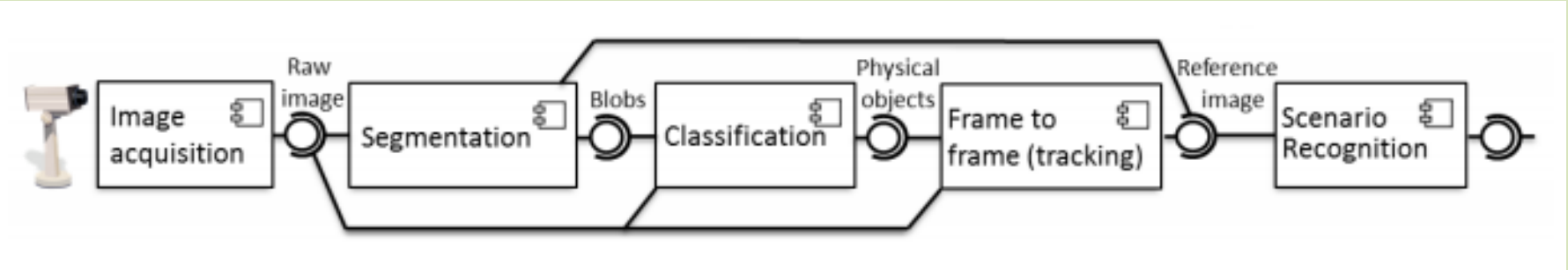
- **Pipelines**, que restringem as topologias e as sequências lineares de filtros;
- **Pipes limitados**, que restringem a quantidade de dados que podem residir em um pipe; e
- **Pipes tipados**, que exigem que os dados transmitidos entre dois filtros tenham um tipo bem definido.

Um exemplo da arquitetura duto e filtro



Como outro exemplo bem conhecido, compiladores têm sido tradicionalmente vistos como sistemas em pipeline (embora as fases frequentemente não sejam incrementais). As etapas no pipeline incluem análise lexical, análise sintática, análise semântica e geração de código.
Outros exemplos de pipes e filtros ocorrem nos domínios de processamento de sinais, programação funcional e sistemas distribuídos.

Um exemplo da arquitetura duto e filtro



Um exemplo da arquitetura duto e filtro

Sistemas de pipe e filtro possuem diversas propriedades interessantes:

- Primeiramente, eles permitem que o projetista compreenda o comportamento geral de entrada/saída de um sistema como uma composição simples dos comportamentos dos filtros individuais.
- Em segundo lugar, eles suportam a reutilização: quaisquer dois filtros podem ser conectados, desde que concordem com os dados que estão sendo transmitidos entre eles.
- Em terceiro lugar, os sistemas podem ser facilmente mantidos e aprimorados: novos filtros podem ser adicionados aos sistemas existentes e filtros antigos podem ser substituídos por versões melhoradas.
- Por fim, eles naturalmente suportam execução concorrente. Cada filtro pode ser implementado como uma tarefa separada e potencialmente executado em paralelo com outros filtros.

Pontos Importantes

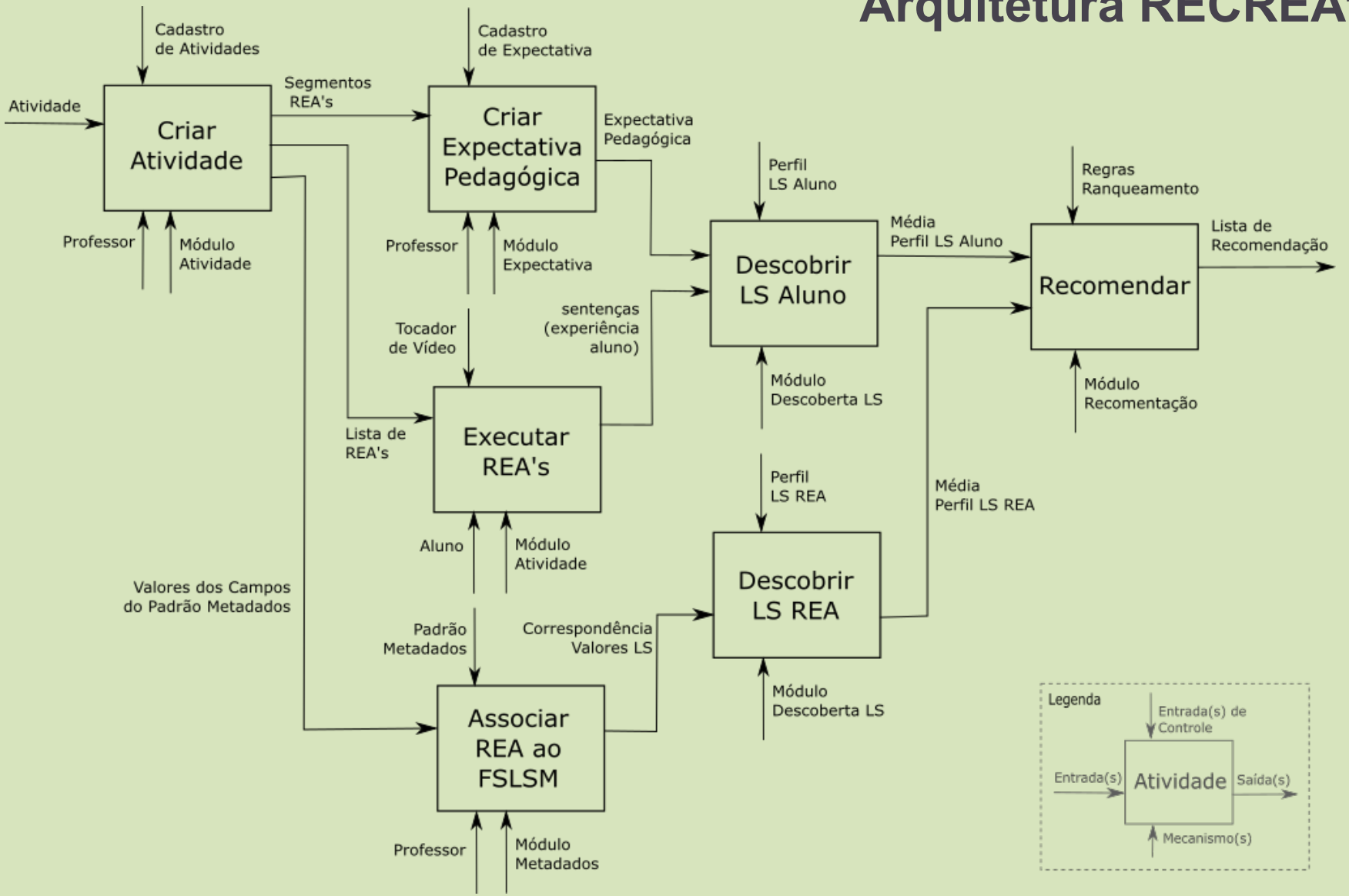
- Uma arquitetura de software é uma descrição de como um sistema de software é organizado.

- Decisões de projeto de arquitetura incluem decisões sobre o tipo de aplicação, a distribuição do sistema, e o estilo de arquitetura a ser usada.

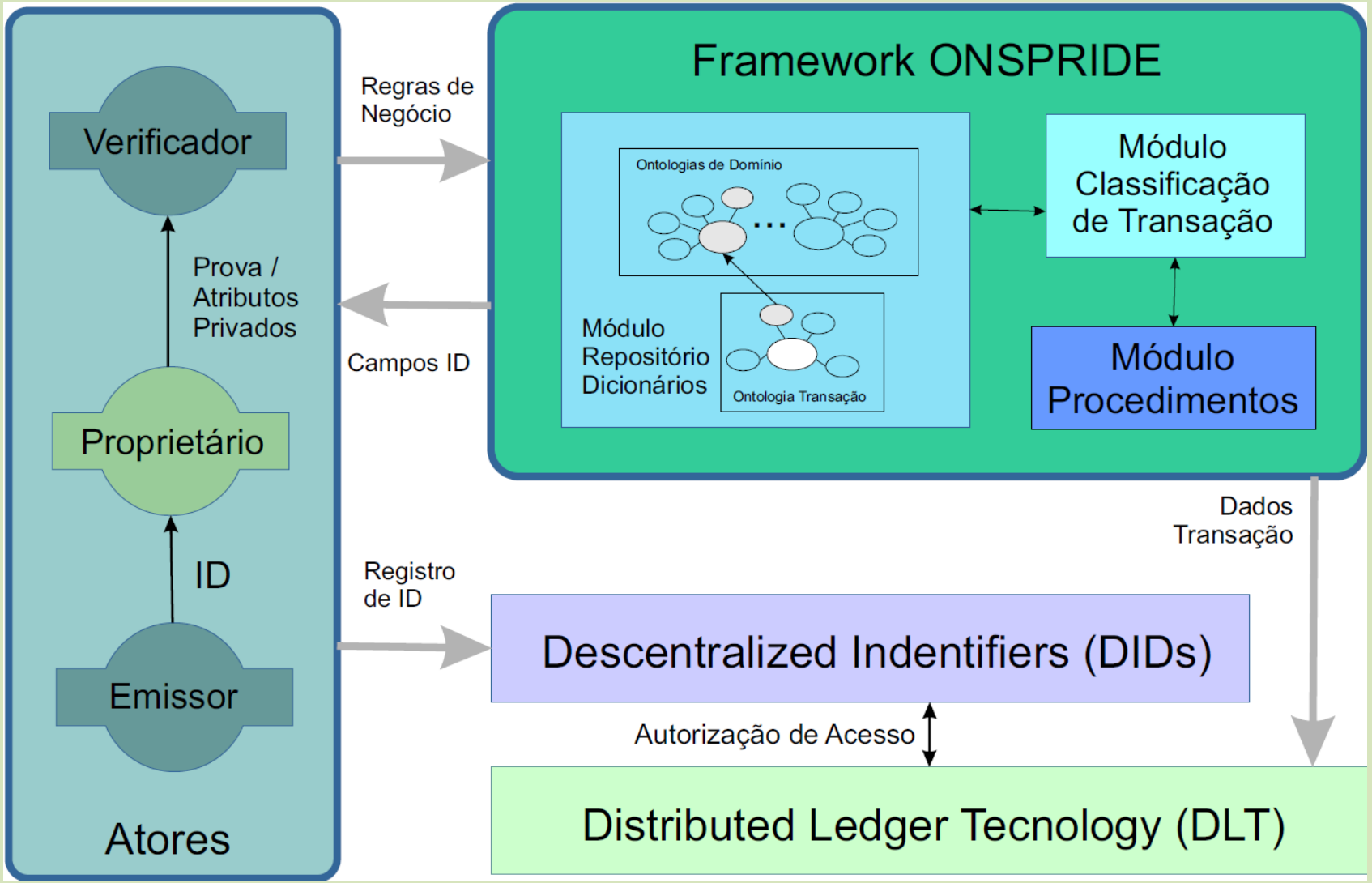
- As arquiteturas podem ser documentadas de várias perspectivas ou visões diferentes tais como uma visão conceitual, uma visão lógica, uma visão de processo, uma visão de desenvolvimento e uma visão física.

- Os padrões de arquitetura são um meio de reusar o conhecimento sobre as arquiteturas genéricas de sistemas. Eles descrevem a arquitetura, explicam quando podem ser usados e descrevem suas vantagens e desvantagens.

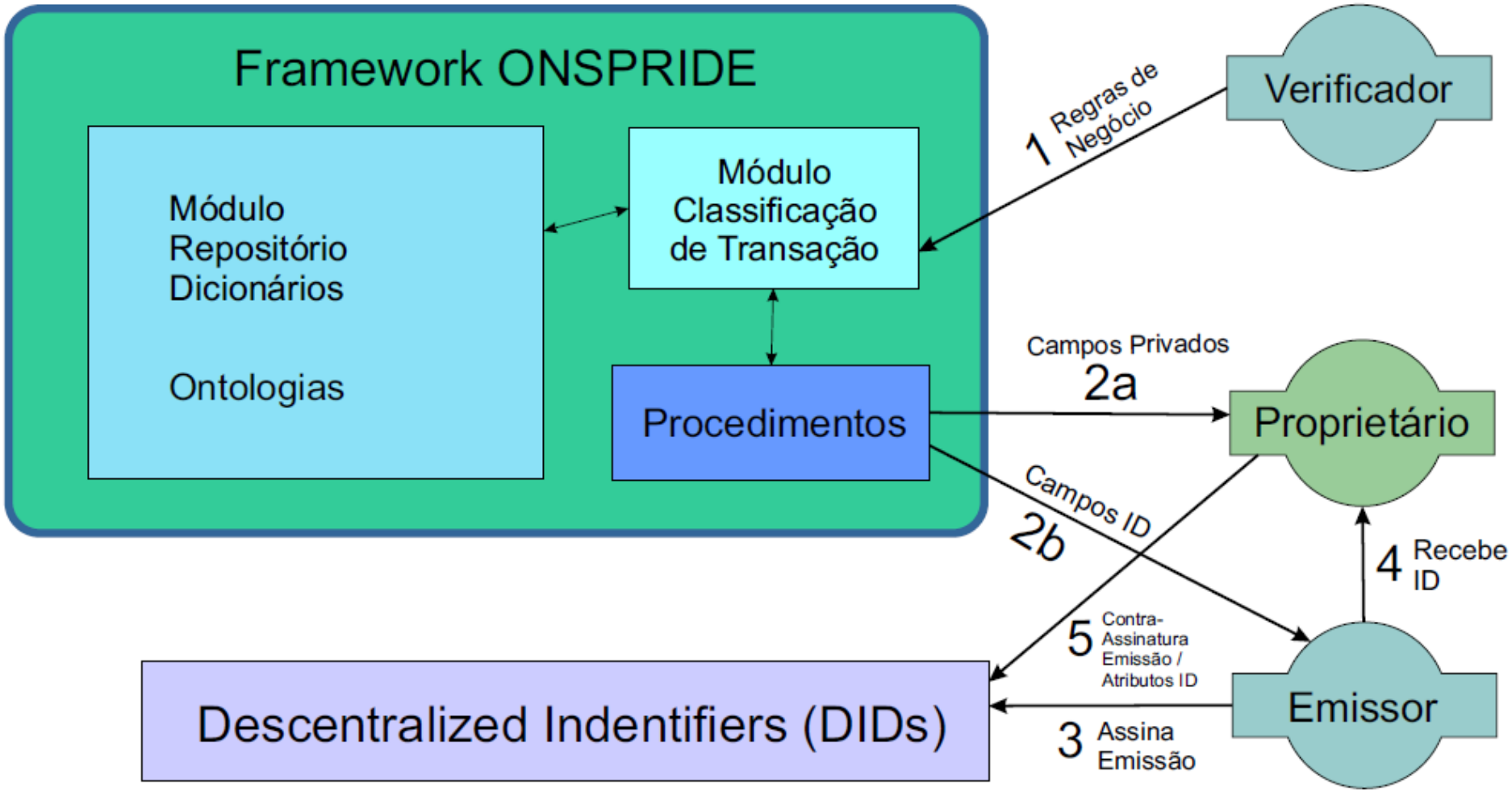
Arquitetura RECREAtE



Arquitetura da Solução ONSPRIDE

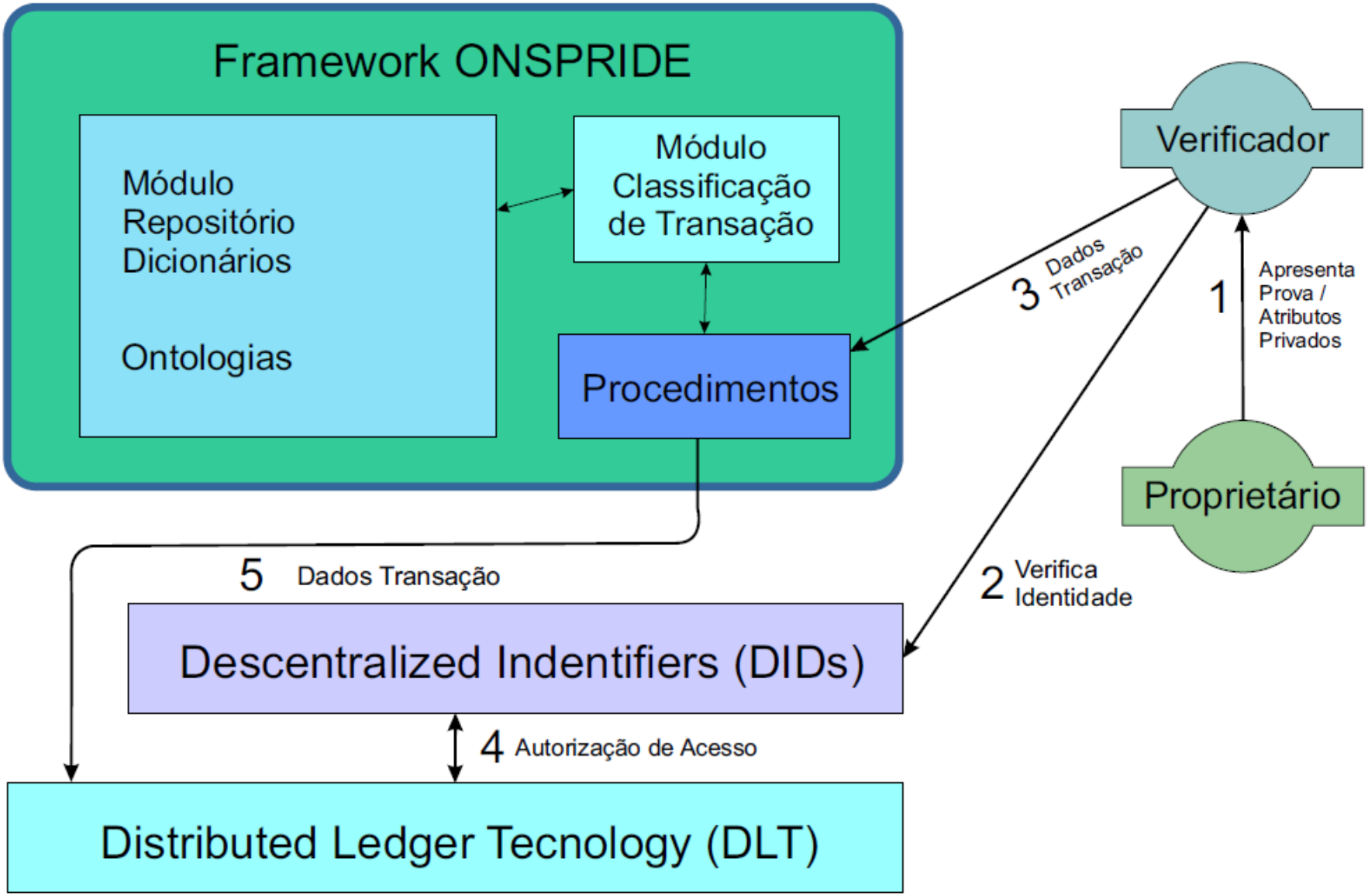


Cenário de Uso



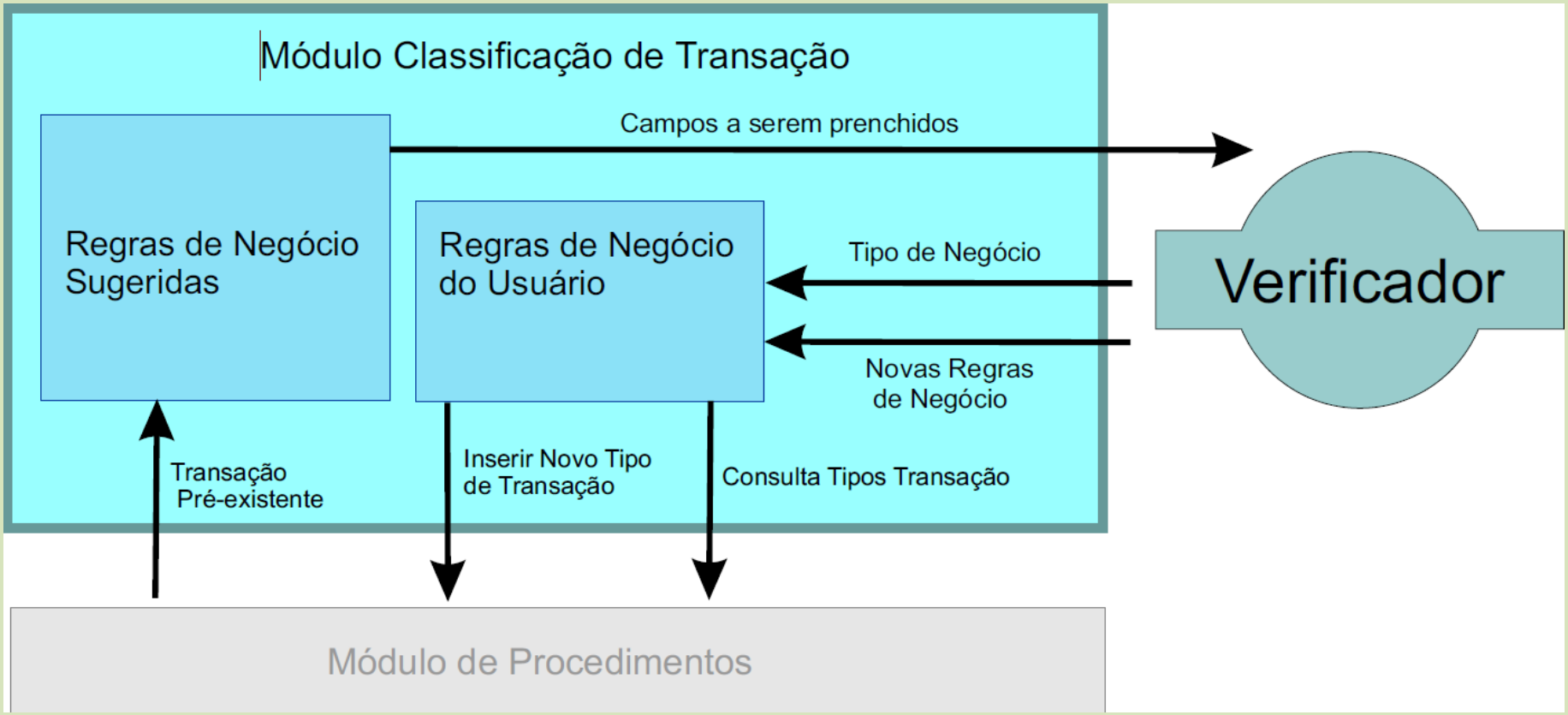
Fonte: Elaborada pelo autor.

Figura 16 – Cenário de Uso da solução com o framework ONSPRIDE - Apresentação de ID e acesso

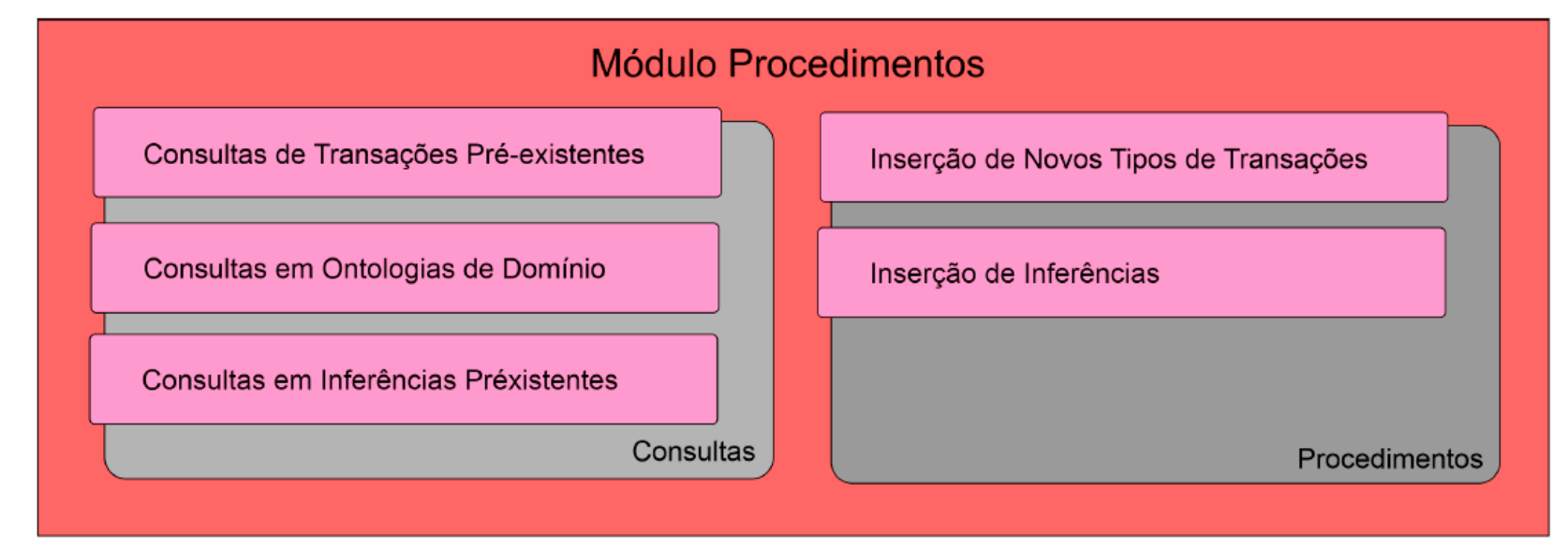


Fonte: Elaborada pelo autor.

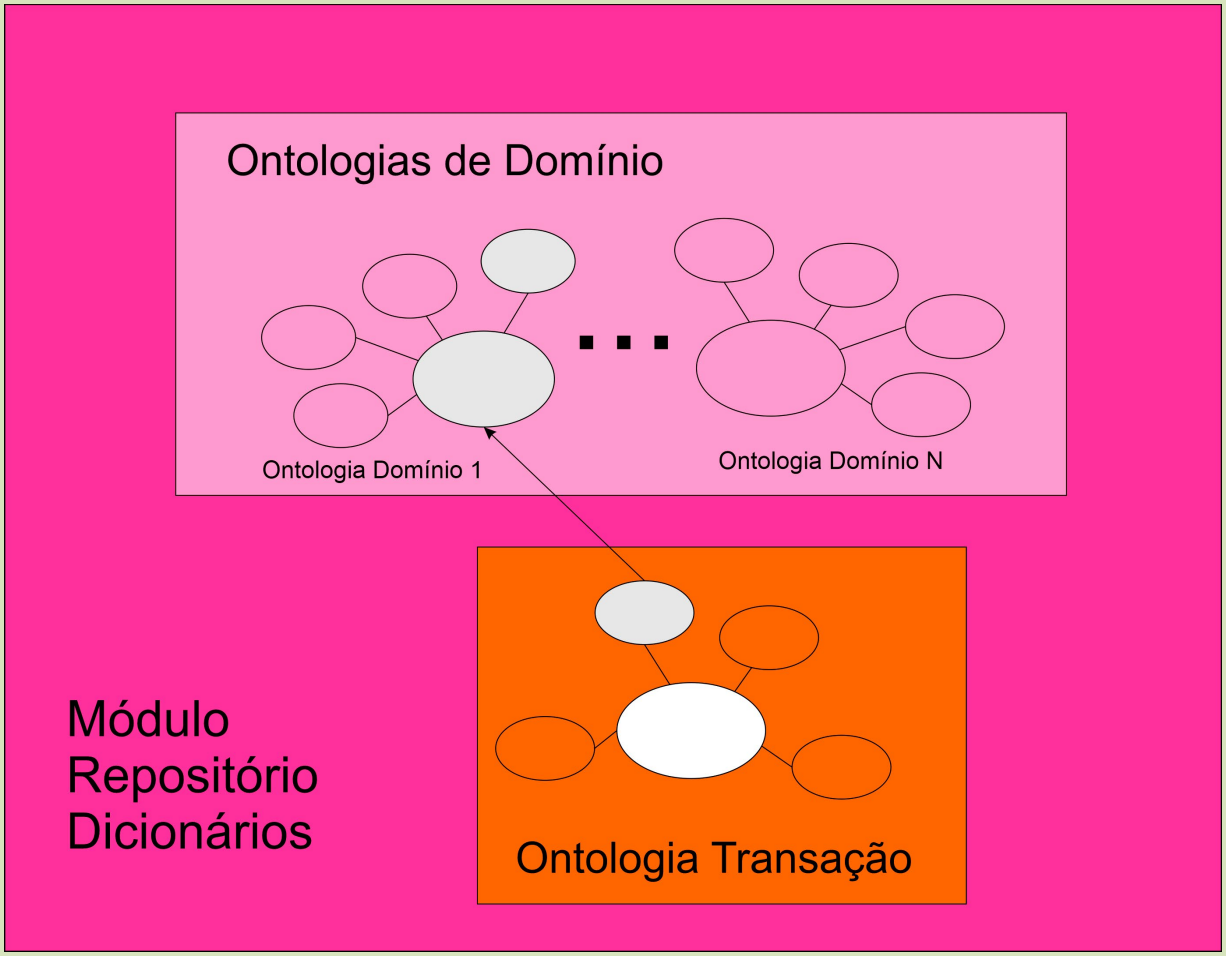
Arquitetura da Solução ONSPRIDE



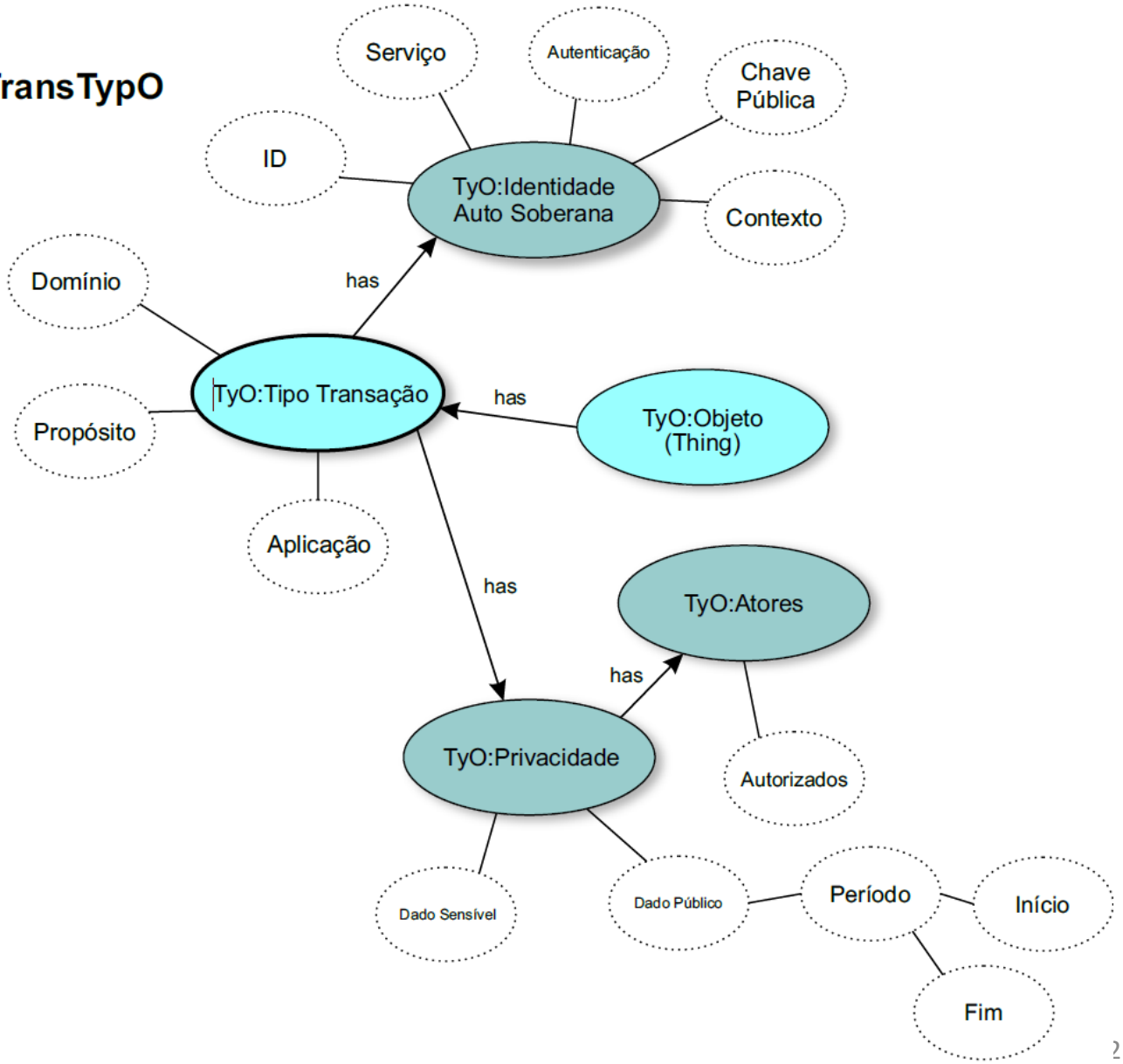
Arquitetura da Solução ONSPRIDE



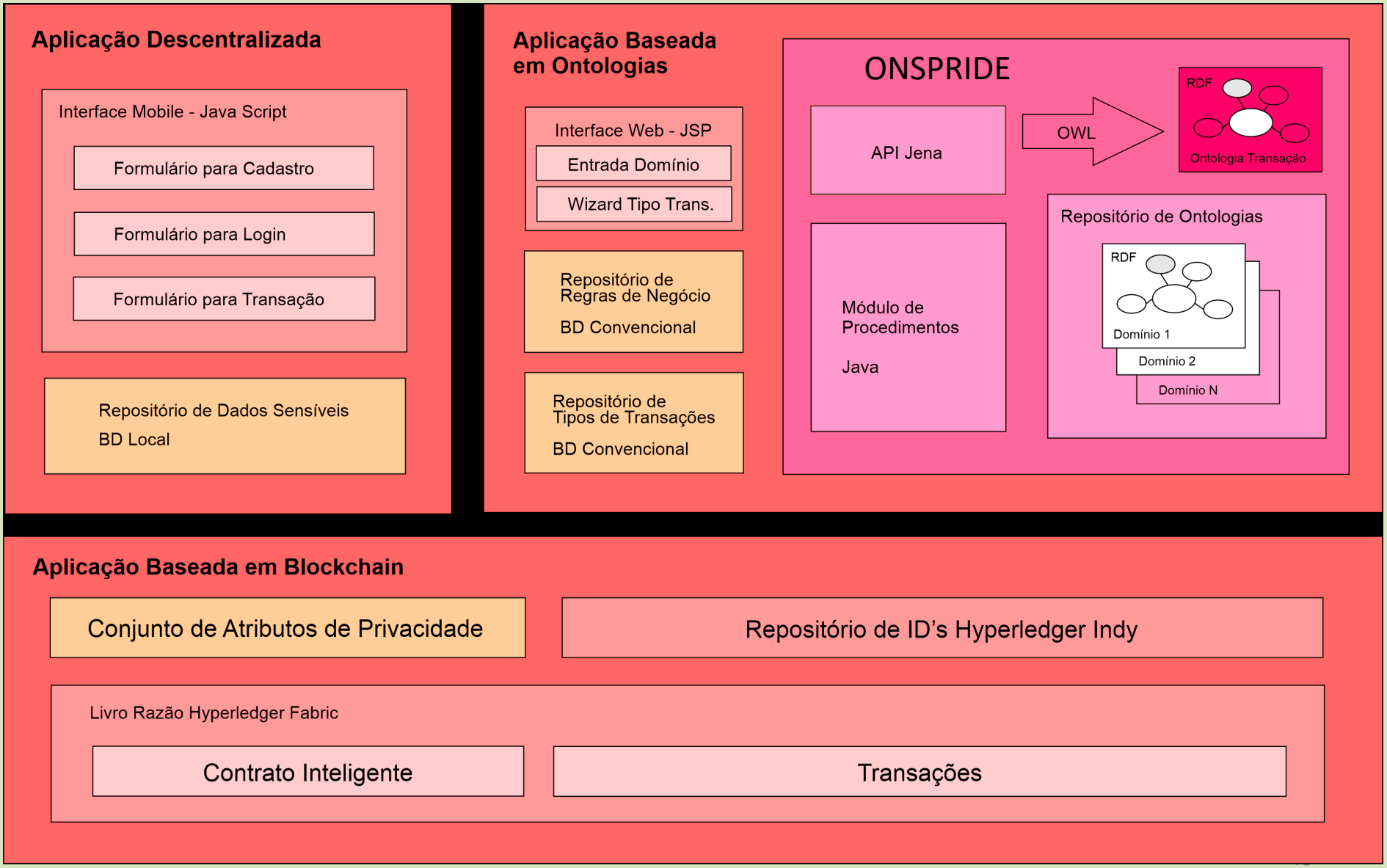
Arquitetura da Solução ONSPRIDE



Ontologia TransTypO



Instância de Implementação da Arquitetura



Instância de Implementação da Arquitetura

Aplicação Descentralizada

Interface Mobile - Java Script

Formulário para Cadastro

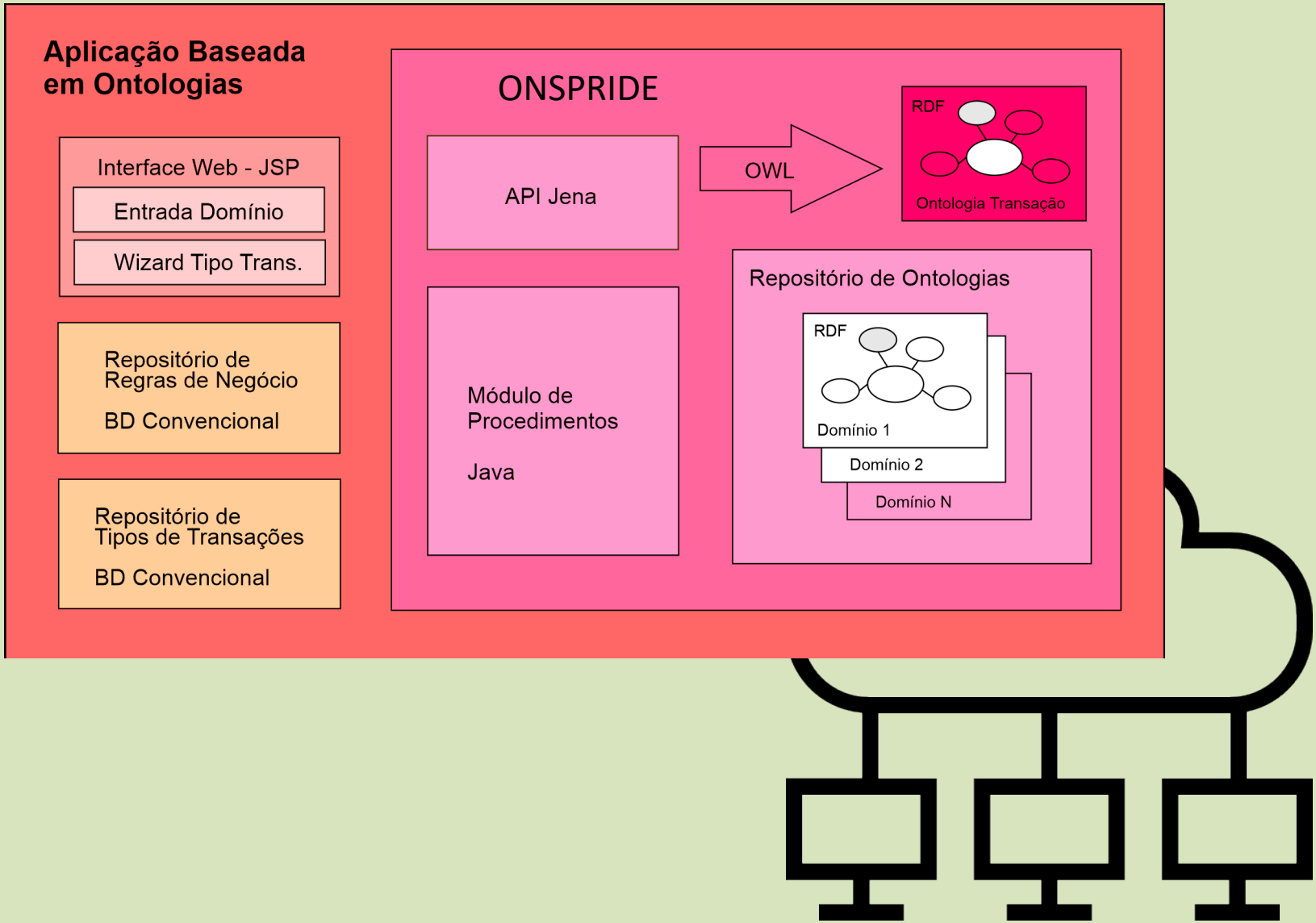
Formulário para Login

Formulário para Transação

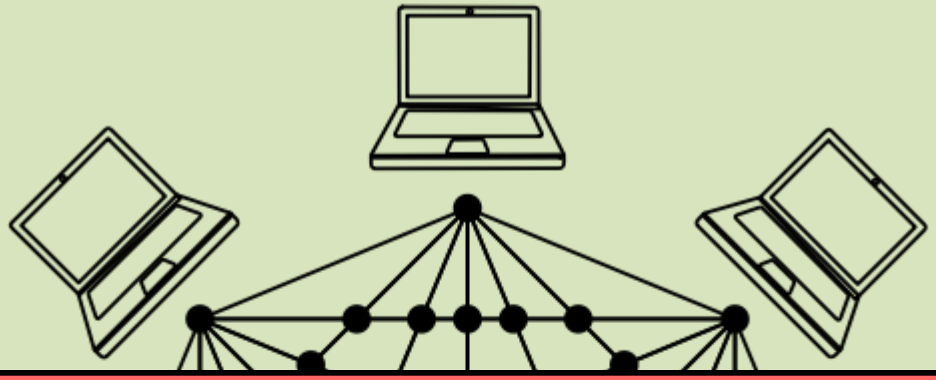
Repositório de Dados Sensíveis

BD Local

Instância de Implementação da Arquitetura



Instância de Implementação da Arquitetura



Aplicação Baseada em Blockchain

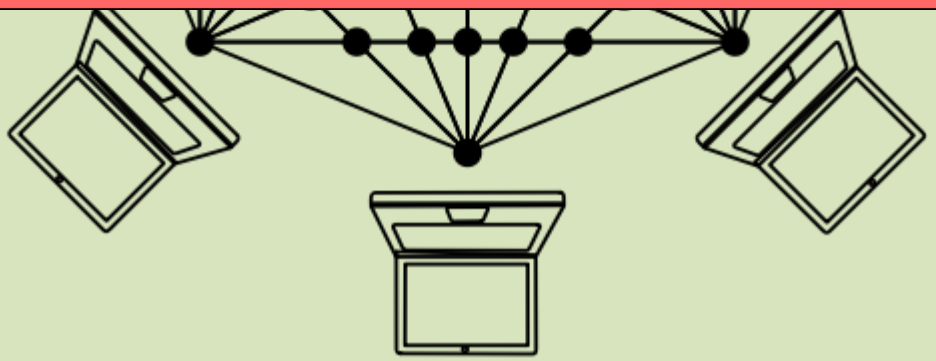
Conjunto de Atributos de Privacidade

Repositório de ID's Hyperledger Indy

Livro Razão Hyperledger Fabric

Contrato Inteligente

Transações



Atividade – Grupo do Projeto

✧ **Atividade para todos - Grupo do Projeto**

✧ Criar uma arquitetura para o software do projeto