



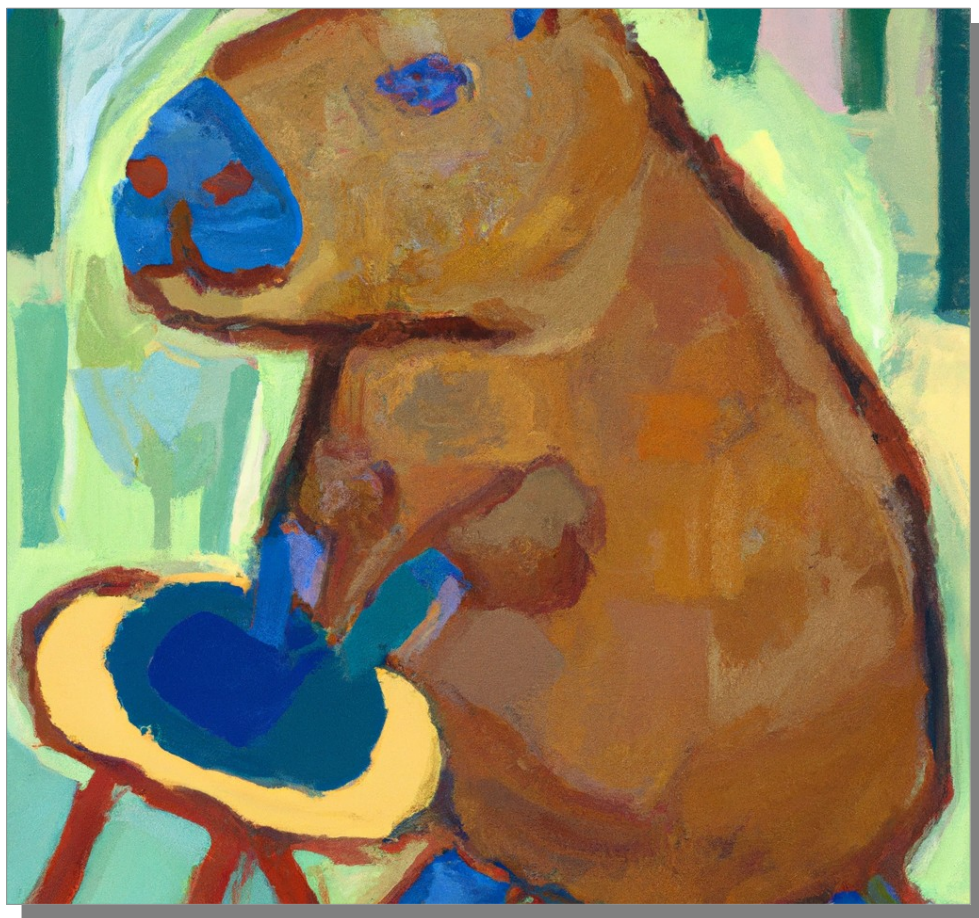
Universidade de Brasília

Departamento de Ciência da Computação



Bancos de Dados

CIC0097



Prof. Pedro Garcia Freitas

<https://pedrogarcia.gitlab.io/>

pedro.garcia@unb.br

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciências da Computação



Este conjunto de slides não deve ser utilizado ou republicado sem a expressa permissão do autor.

This set of slides should not be used or republished without the author's express permission.

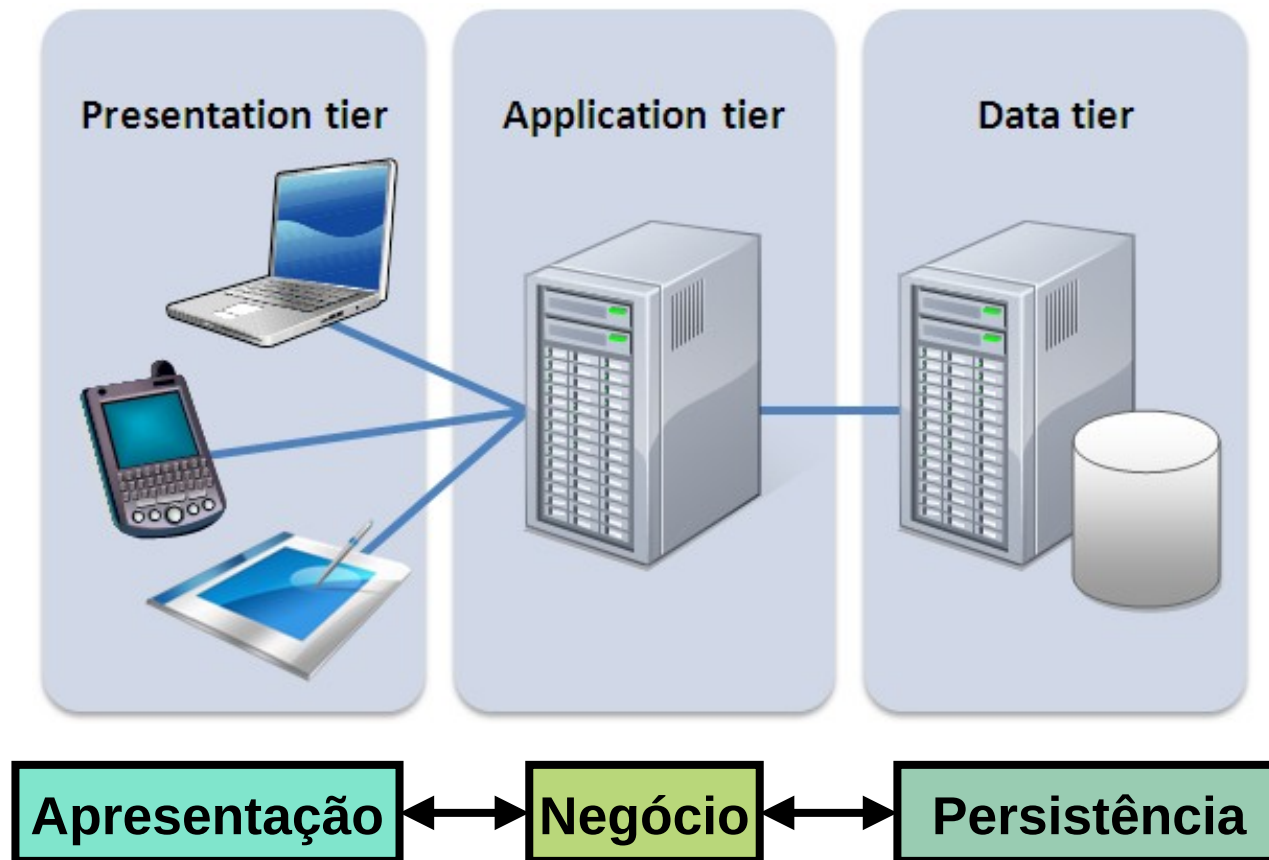


Módulo 17

Camadas de Software e Arquitetura para Desenvolvimento

**CIC0097/2023.1
T1/T2**

1. Arquitetura para Desenvolvimento



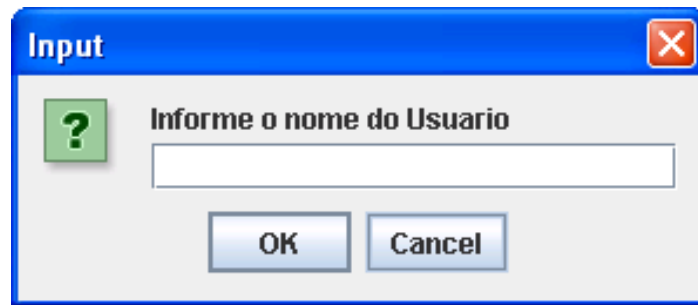
1. Arquitetura para Desenvolvimento



**Arquitetura em três camadas
(*three tier architecture*) é
o padrão de desenvolvimento
para aplicativo !!!**



1. Arquitetura para Desenvolvimento



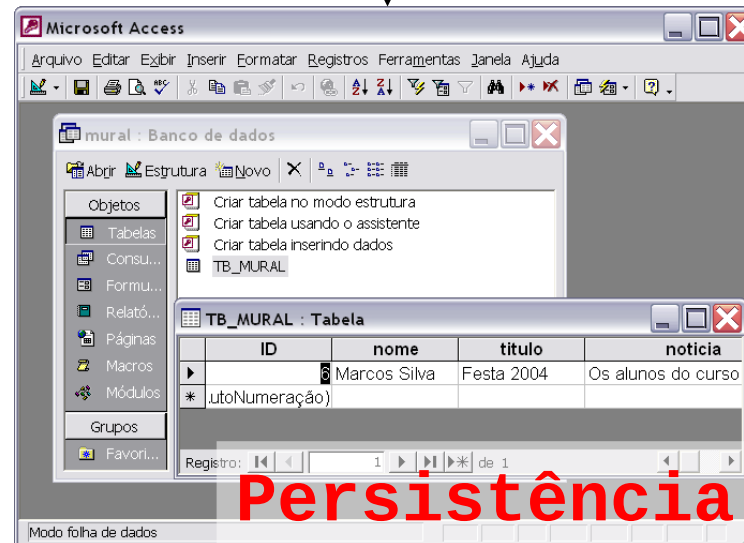
Input

Informe o nome do Usuario

OK Cancel

Apresentação

Negócio/Aplicação



Microsoft Access

mural : Banco de dados

Objetos

- Tabelas
- Consu...
- Formu...
- Relató...
- Páginas
- Macros
- Módulos
- Grupos
- Favori...

Novo

- Criar tabela no modo estrutura
- Criar tabela usando o assistente
- Criar tabela inserindo dados
- TB_MURAL

TB_MURAL : Tabela

ID	nome	titulo	noticia
1	Marcos Silva	Festa 2004	Os alunos do curso
* (autoNumeração)			

Registro: 1 de 1

Modo folha de dados

Persistência



1. Arquitetura para Desenvolvimento

A arquitetura de três camadas (tiers) é uma arquitetura de aplicativo de software estabelecida que organiza aplicativos em três camadas de computação física e lógica:

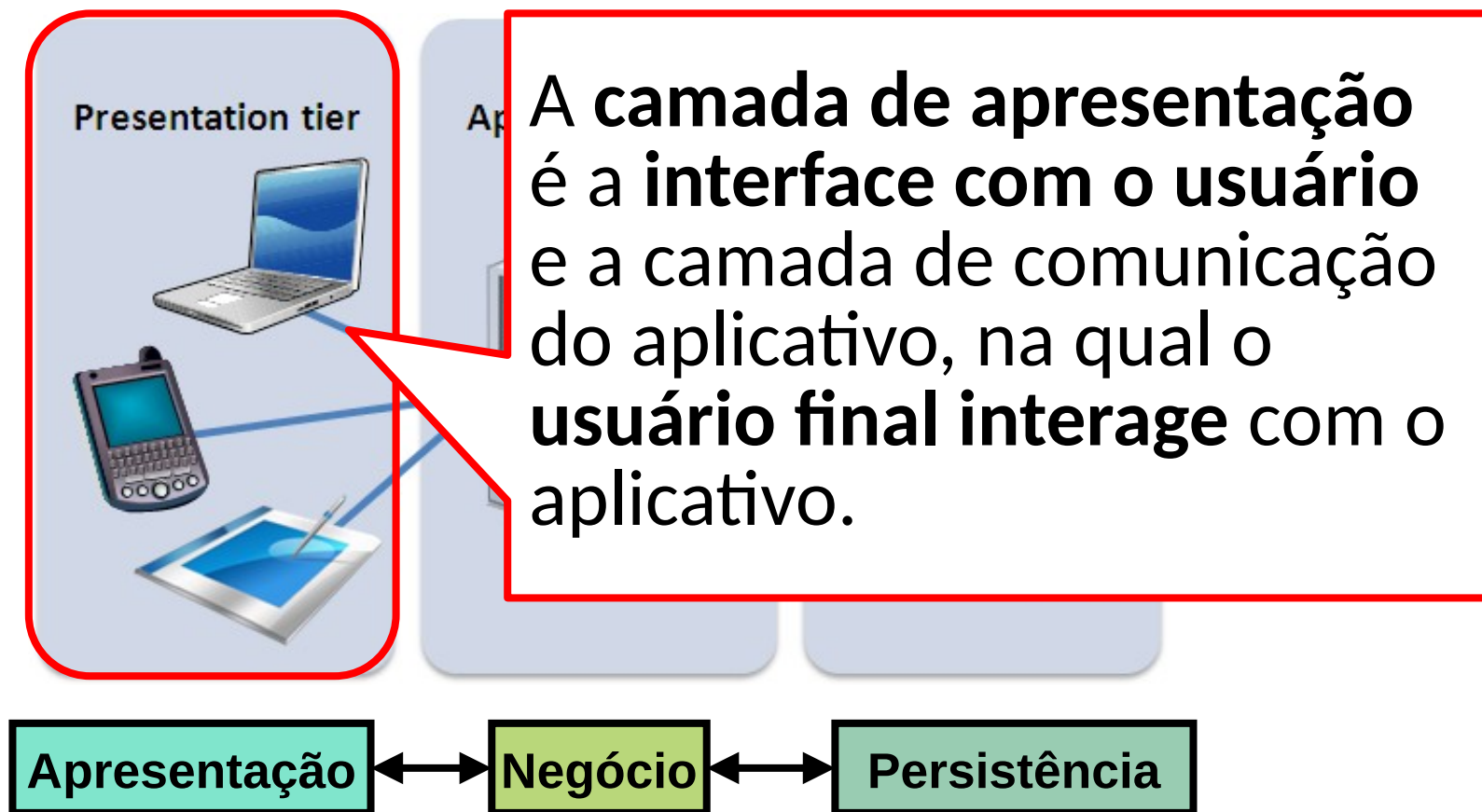
- a camada de apresentação ou a interface com o usuário;
- a camada do aplicativo, na qual os dados são processados;
- e a camada de dados, na qual os dados associados ao aplicativo são armazenados e gerenciados.

1. Arquitetura para Desenvolvimento

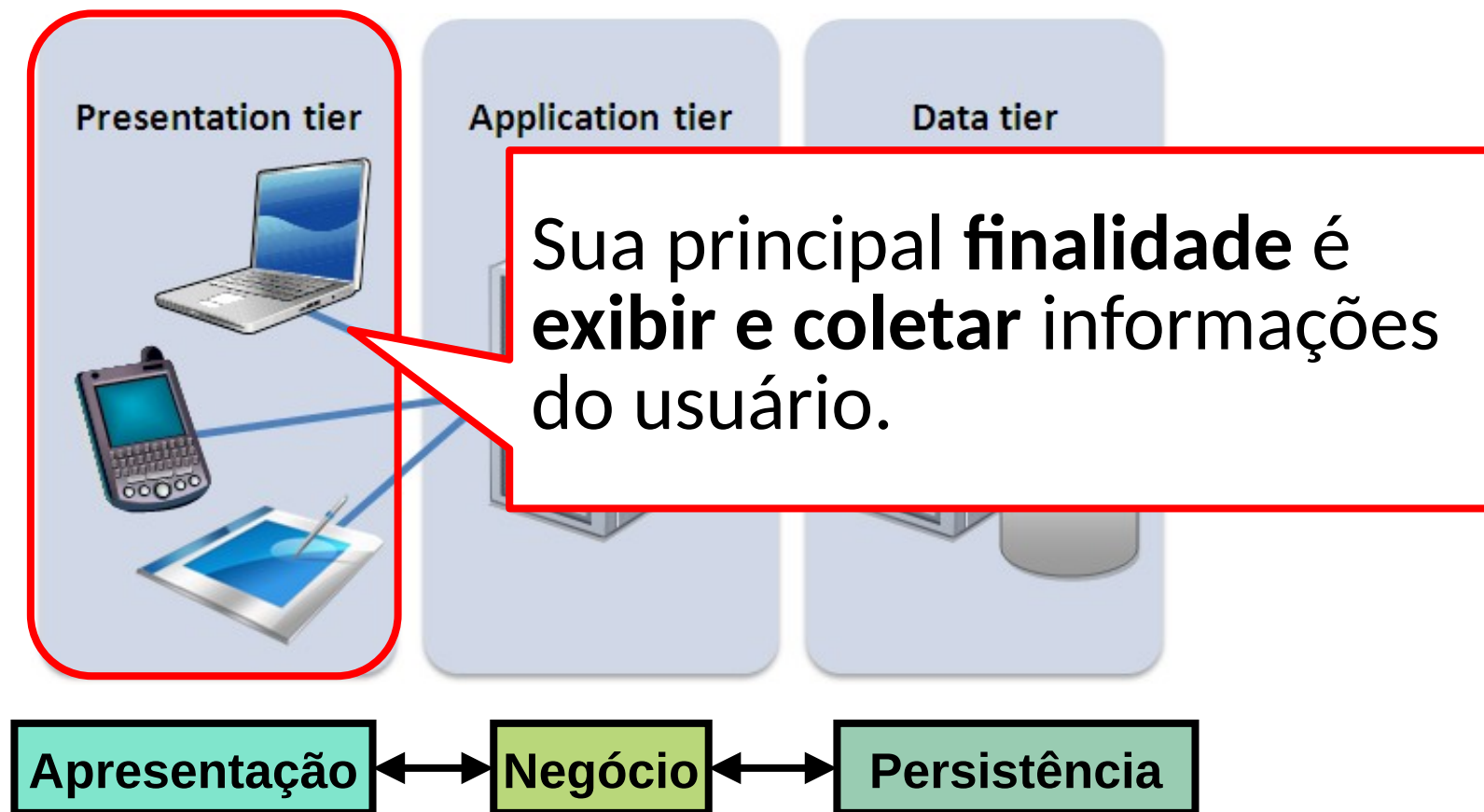
O principal **benefício** da arquitetura de três camadas é que devido ao fato de cada camada executar sua própria infraestrutura, cada camada pode ser **desenvolvida simultaneamente** por uma equipe de desenvolvimento separada e pode ser atualizada ou **ajustada conforme necessário** sem impactar as outras camadas.



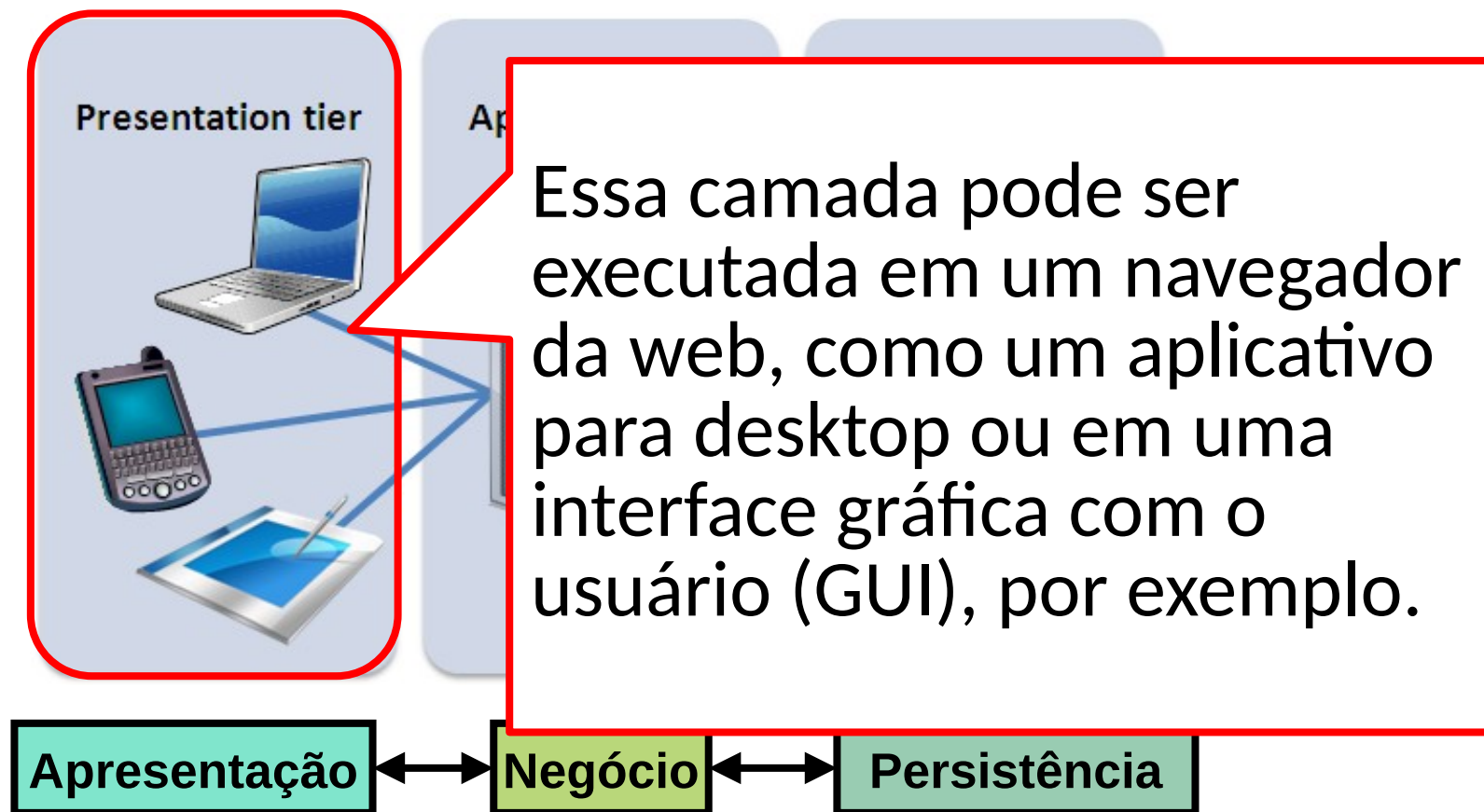
1. Arquitetura para Desenvolvimento



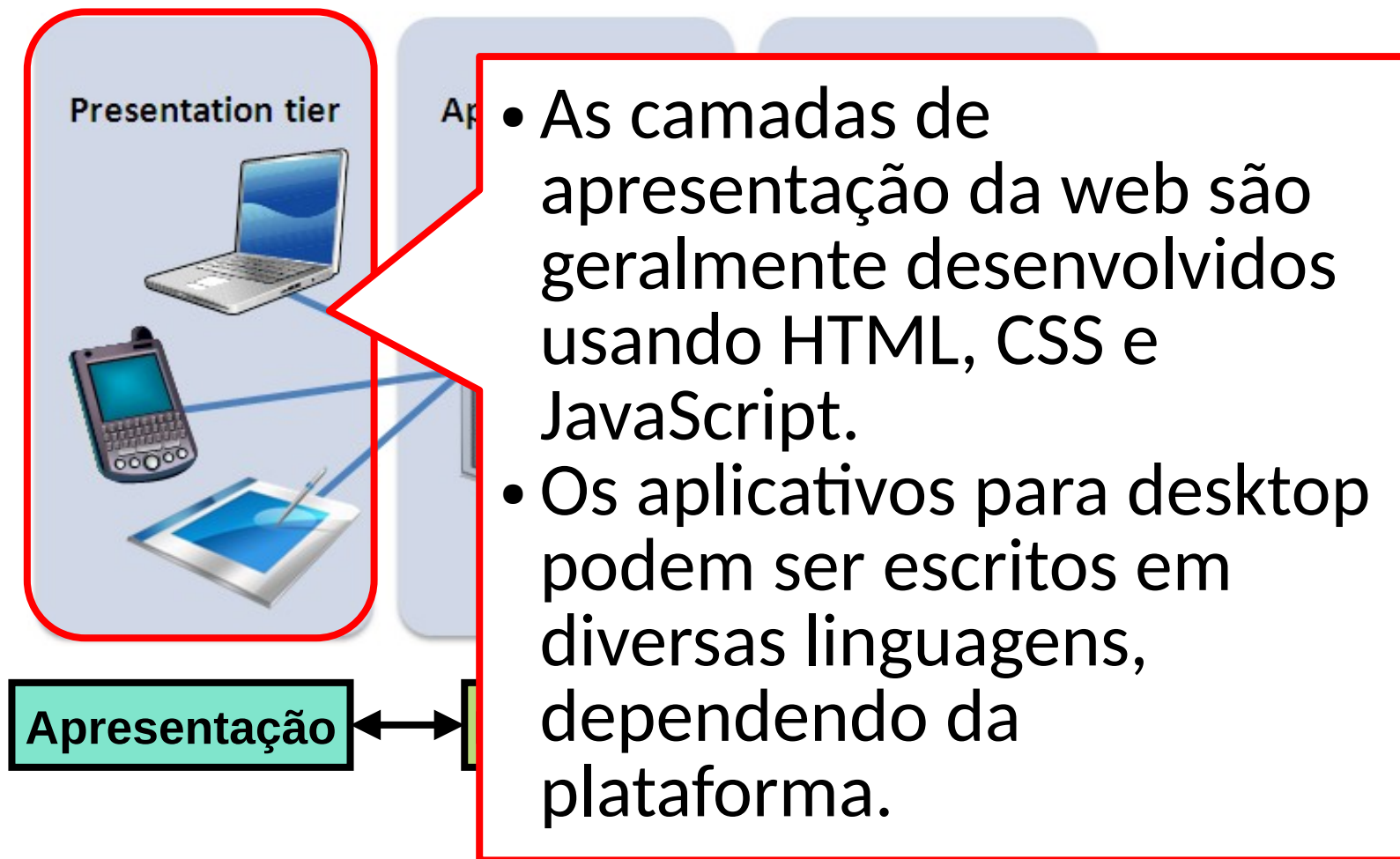
1. Arquitetura para Desenvolvimento



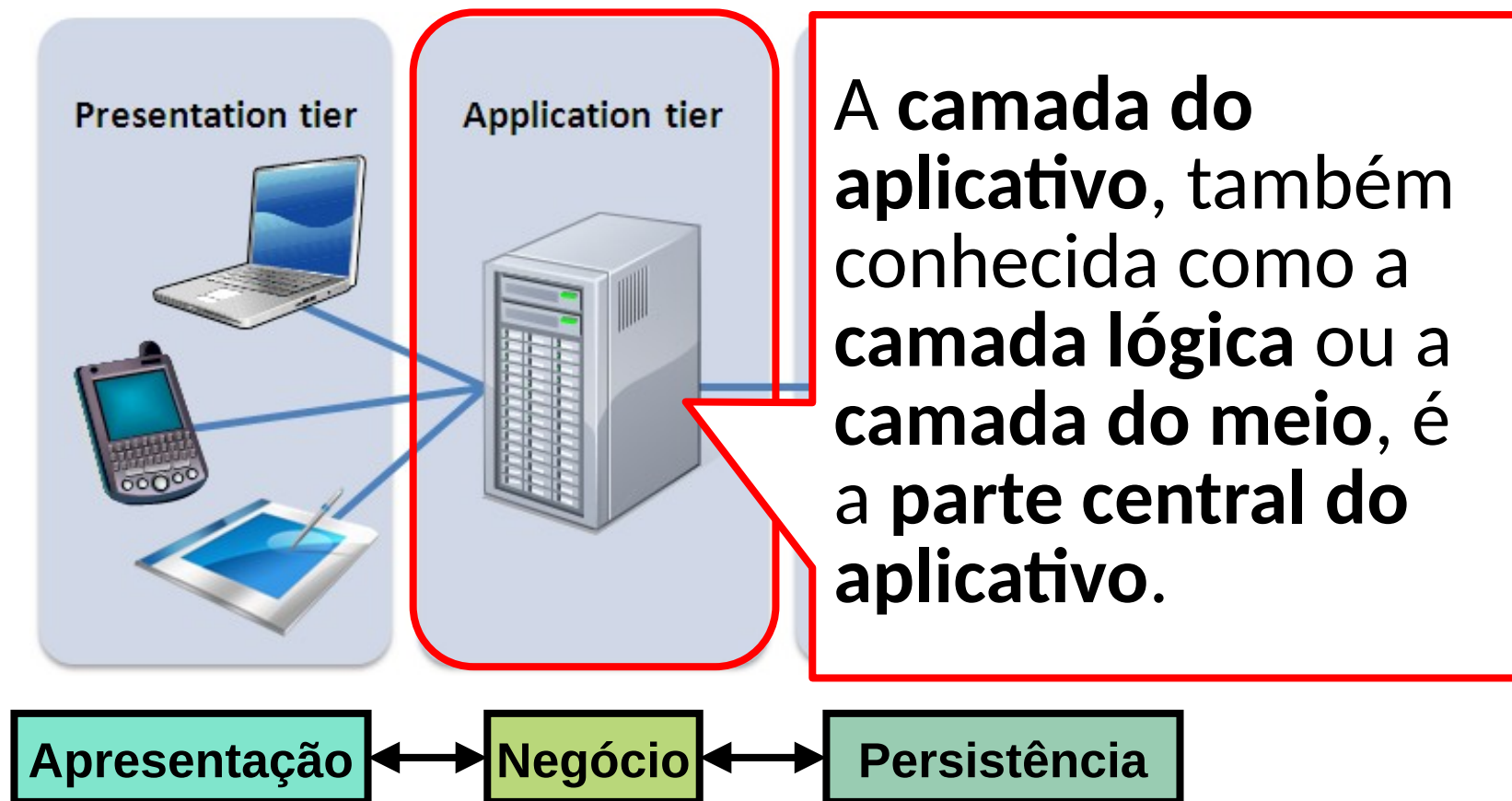
1. Arquitetura para Desenvolvimento



1. Arquitetura para Desenvolvimento



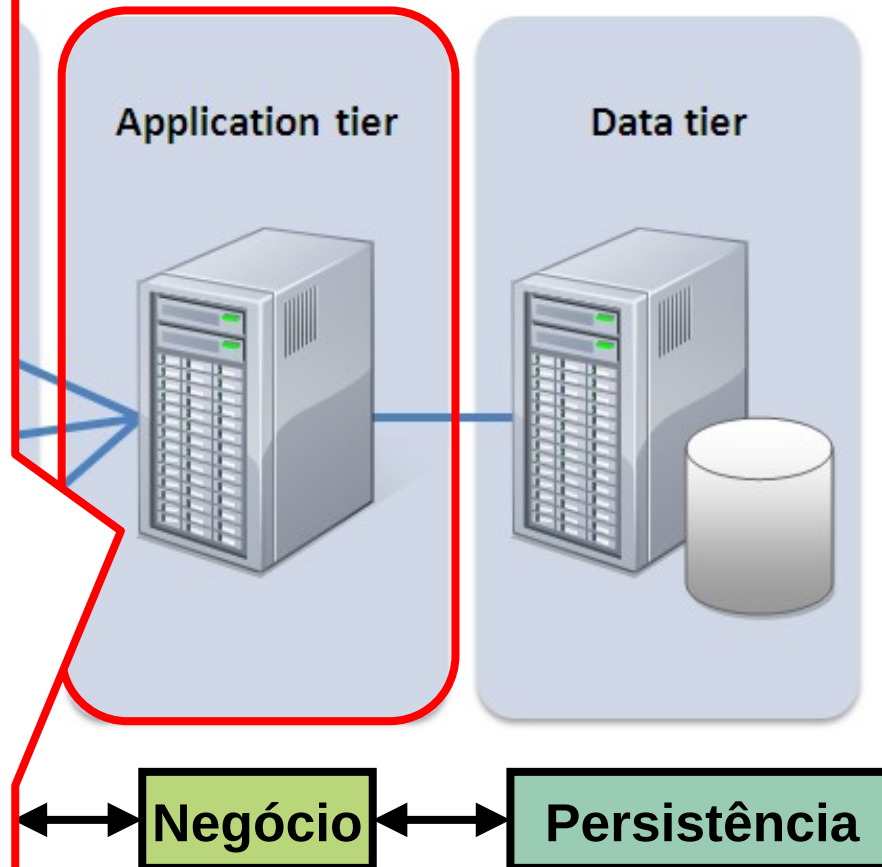
1. Arquitetura para Desenvolvimento



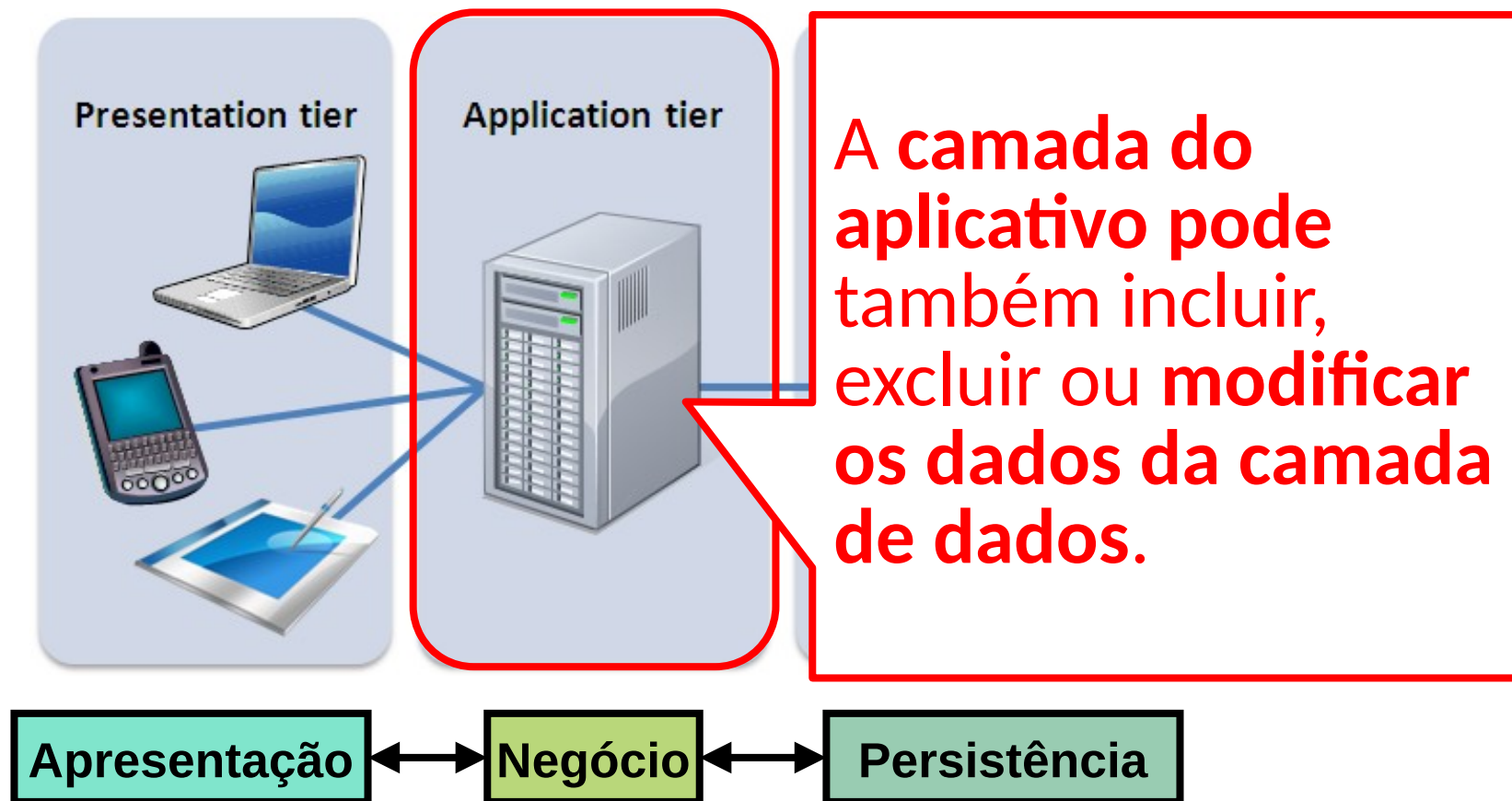


Nessa camada, as **informações** coletadas na camada de apresentação **são processadas**, algumas vezes em relação a outras informações da camada de dados, **usando a lógica de negócios** que é um conjunto específico de **regras de negócios**.

para Desenvolvimento

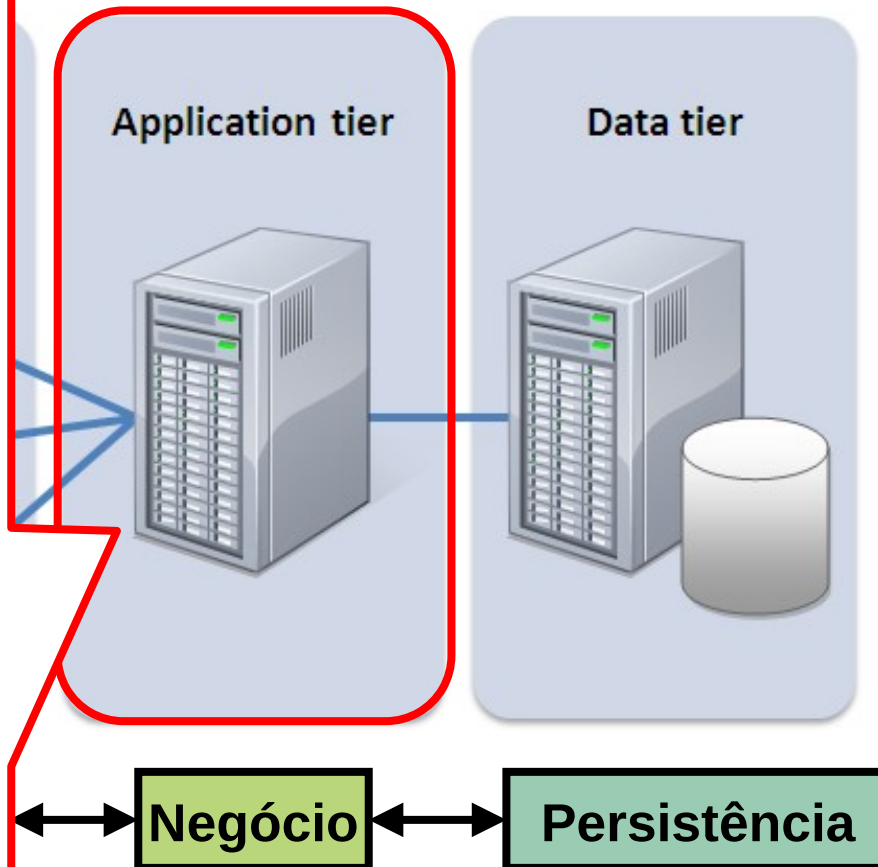


1. Arquitetura para Desenvolvimento



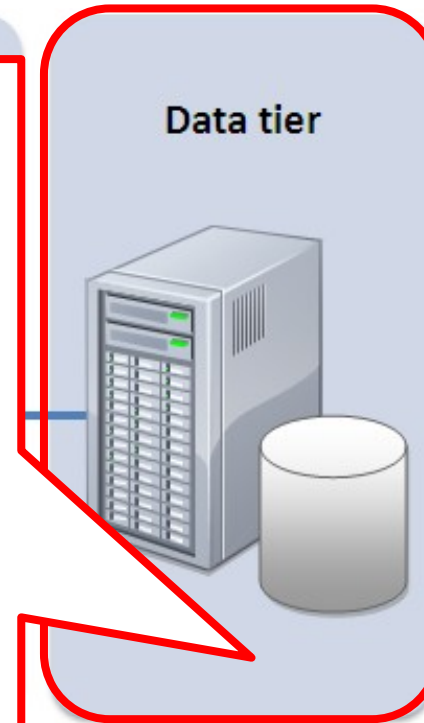
1. Arquitetura para Desenvolvimento

A camada do aplicativo é geralmente desenvolvida usando **Python, Java, Perl, PHP, Javascript** ou Ruby e se comunica com a camada de dados usando chamadas de API.



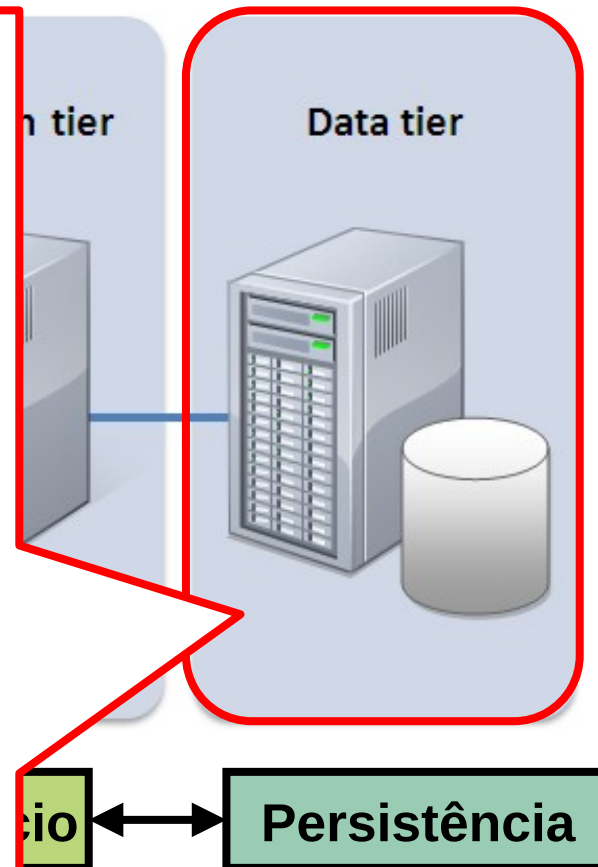
1. Arquitetura para Desenvolvimento

A **camada de dados**, por vezes chamada de camada de banco de dados, **camada de acesso a dados ou back-end**, é na qual as informações processadas pelo aplicativo são armazenadas e gerenciadas.



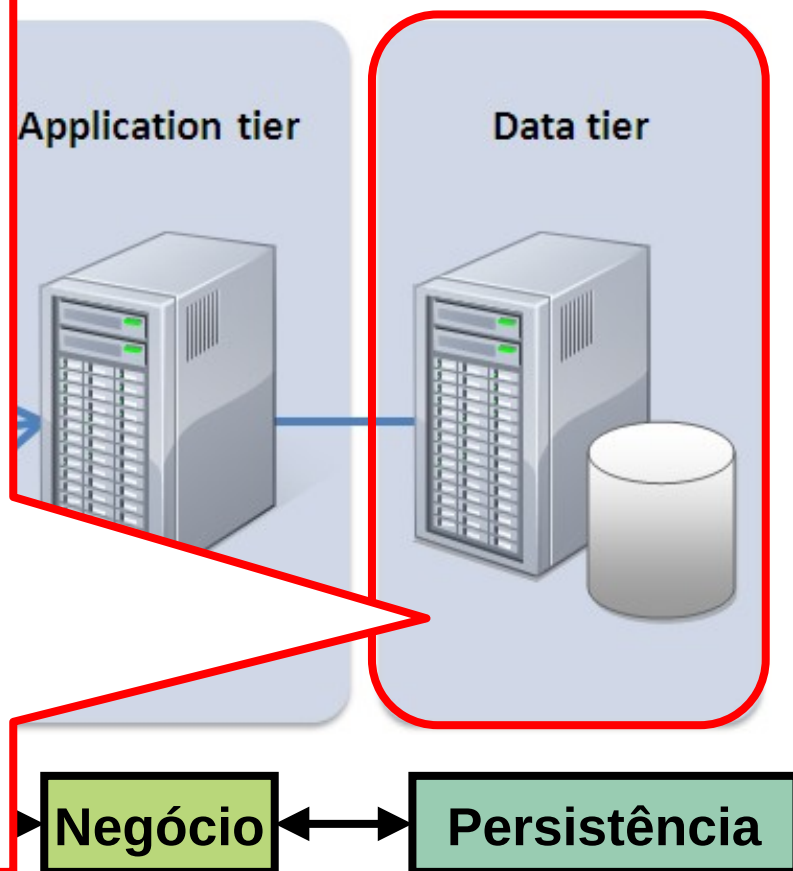
1. Arquitetura para Desenvolvimento

Este pode ser um sistema de gerenciamento de banco de dados relacional, como **PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix** ou **Microsoft SQL Server** ou em um servidor de banco de dados **NoSQL**, como **Cassandra, CouchDB** ou **MongoDB**.



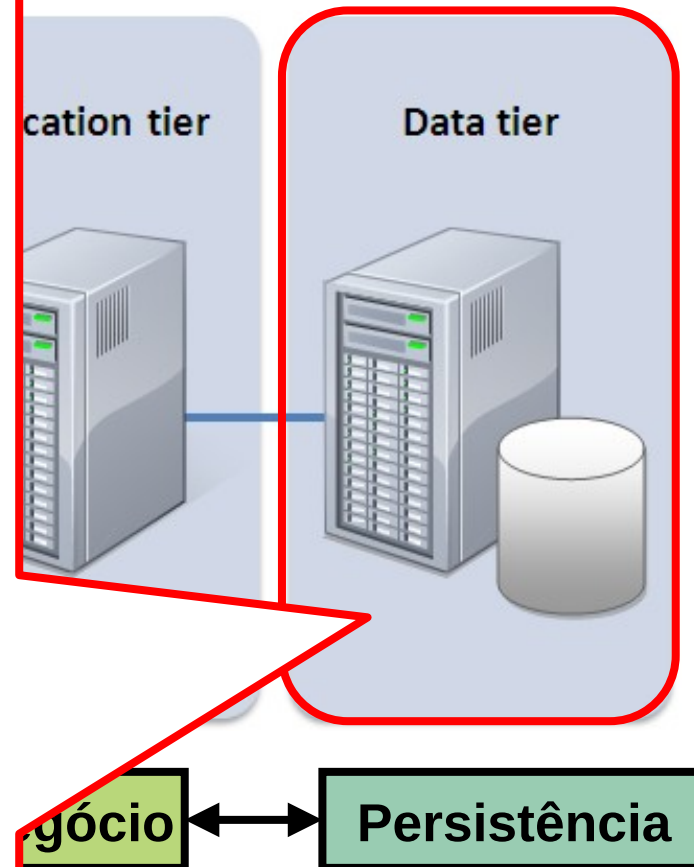
1. Arquitetura para Desenvolvimento

Em um aplicativo de três camadas, **toda a comunicação** passa pela camada do aplicativo. A **camada de apresentação** e a **camada de dados** não podem se comunicar diretamente entre si.



1. Arquitetura para Desenvolvimento

- Portanto, a separação das abstrações de código deve ser a mais desacoplada possível.
- Assim, padrões de projetos são fundamentais para projetar sistemas de forma mais eficiente, reutilizável e de fácil manutenção.





2. Benefícios de três camadas (tiers)

- O principal benefício da arquitetura de três camadas é a separação lógica e física da funcionalidade.
- Cada camada pode ser executada em um sistema operacional e plataforma de servidor diferentes, por exemplo, servidor da web, servidor de aplicativos, servidor de banco de dados, que mais atende aos seus requisitos funcionais.



2. Benefícios de três camadas (tiers)

- E cada camada é executada em pelo menos um hardware de servidor dedicado ou servidor virtual, por isso os serviços de cada camada podem ser **customizados** e **otimizados** sem impactar as outras camadas.



2. Benefícios de três camadas (tiers)

- Outros benefícios incluem:
 - **Desenvolvimento mais rápido:** uma vez que todas as camadas podem ser desenvolvidas simultaneamente por diferentes equipes, uma organização pode lançar o aplicativo no mercado mais rapidamente e os programadores podem usar as linguagens e ferramentas mais recentes e melhores para cada camada.



2. Benefícios de três camadas (tiers)

- Outros benefícios incluem:
 - **Segurança aprimorada:** uma vez que a camada de apresentação e a camada de dados não podem se comunicar diretamente, uma camada do aplicativo bem projetada pode ter a função de uma espécie de firewall interno, impedindo injeções SQL e outros exploradores de vulnerabilidade mal-intencionados.



2. Benefícios de três camadas (tiers)

- Outros benefícios incluem:
 - **Escalabilidade:** qualquer camada pode ser dimensionada independentemente das outras conforme necessário.
 - **Maior confiabilidade:** uma indisponibilidade em uma camada é menos propensa a impactar a disponibilidade ou o desempenho das outras camadas.



3. Camada (tier) vs. nível (layer)

- Nas discussões sobre arquitetura de três camadas, o termo "layer" é frequentemente utilizado erroneamente como sinônimo de "tier", como em "camada de apresentação" ou "camada de lógica de negócios".



3. Camada (tier) vs. nível (layer)

- Eles não são a mesma coisa. Uma "layer" refere-se a uma **divisão funcional do software**, mas um "tier" refere-se a uma **divisão funcional do software que é executada em infraestrutura separada** das outras divisões.
- O aplicativo de Contatos no seu telefone, por exemplo, é uma aplicação de três camadas (*three-layer*), mas é uma aplicação de um único "tier", porque todas as três *layers* são executadas no seu telefone.

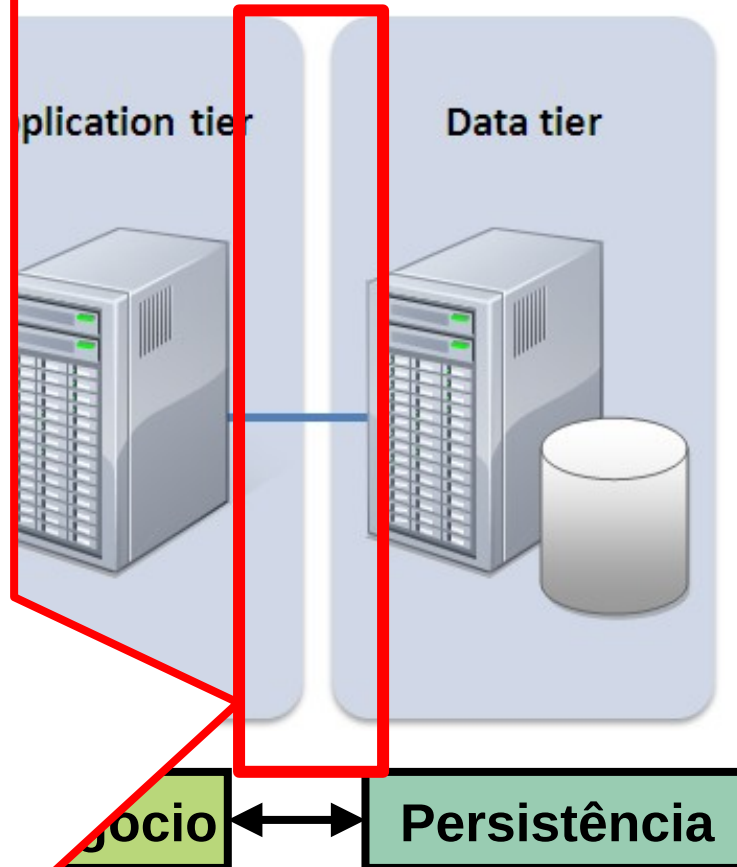


4. Camada de persistência

- A camada de persistência (*persistence layer*), também conhecida como camada de acesso a dados ou camada de persistência de dados, é um componente de um aplicativo de software ou sistema responsável por armazenar e recuperar dados de um armazenamento persistente, como um banco de dados ou sistema de arquivos.

4 Camadas de Persistência

A camada de **persistência** atua como **intermediária** entre a **camada de lógica de negócios do aplicativo** e o **armazenamento de dados subjacente**. Ela **encapsula as operações** e mecanismos necessários para interagir com o armazenamento de dados, incluindo **leitura, escrita, atualização e exclusão de dados**.





4. Camada de persistência

- O **objetivo** principal da camada de persistência é fornecer uma **forma contínua e consistente** de acessar e manipular dados, **abstraindo os detalhes específicos da tecnologia de armazenamento** de dados subjacente.
- Ela garante que o aplicativo possa trabalhar com os dados **sem estar fortemente acoplado à implementação** específica de armazenamento de dados.



4. Camada de persistência

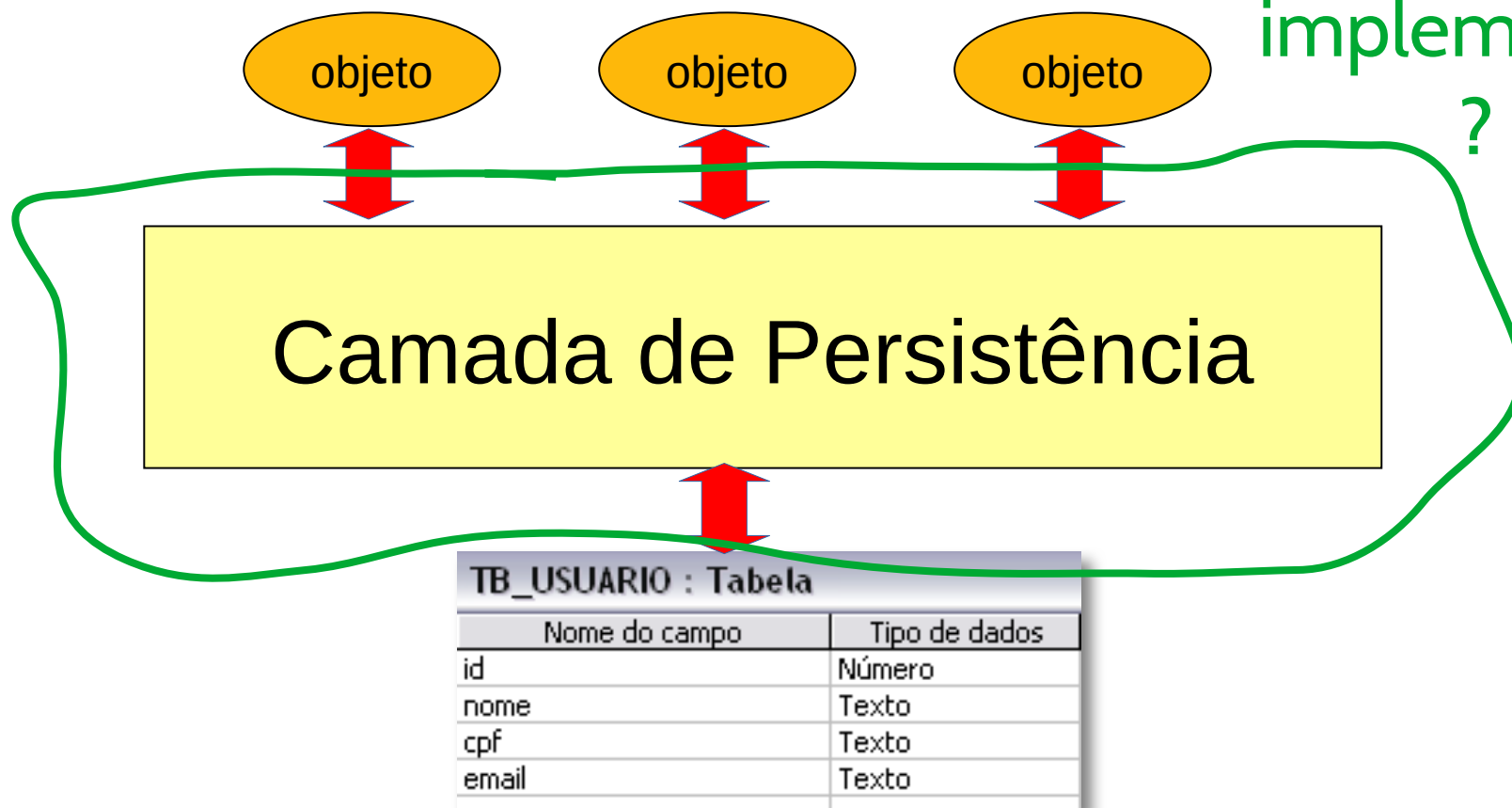
- Ao separar as preocupações de persistência do restante do aplicativo, a **camada de persistência permite um design modular**, facilidade de **manutenção e escalabilidade**.
- Também permite que **diferentes tecnologias de armazenamento** de dados sejam usadas de forma intercambiável, **desde que sigam a mesma interface** fornecida pela camada de persistência.

4. Camada de persistência



4. Camada de persistência

Como
implementar
?





4. Camada de persistência

- Existem várias alternativas para implementar o acesso e persistência de dados em aplicativos de software. Algumas das alternativas comumente usadas incluem:
 - Data Access Object (DAO)
 - Object-Relational Mapping (ORM) frameworks
 - Repository Pattern
 - Active Record Pattern
 - Query Builders



5. Data Access Object (DAO)

	DAO	ORM	Repository Pattern	Active Record Pattern	Query Builders
Definição	Componente intermediário que abstrai o acesso aos dados e fornece métodos padronizados para manipular dados no armazenamento persistente.	Framework que mapeia objetos para tabelas de banco de dados e lida com a conversão entre código orientado a objetos e banco de dados relacionais.	Padrão que abstrai a lógica de acesso aos dados em repositórios, fornecendo uma interface simplificada para consulta e gerenciamento de dados.	Padrão que combina a lógica de acesso aos dados e lógica de negócios em um único objeto, representando uma linha de banco de dados.	Bibliotecas que fornecem uma forma programática de construir consultas de banco de dados usando uma API fluente ou expressiva.



5. Data Access Object (DAO)

	DAO	ORM	Repository Pattern	Active Record Pattern	Query Builders
Responsabilidade	Encapsular as operações e mecanismos de acesso e manipulação de dados.	Mapear objetos para tabelas de banco de dados, gerenciar o ciclo de vida dos objetos e fornecer uma interface de alto nível para acesso e manipulação de dados.	Fornecer uma interface para consulta e manipulação de dados que oculta os detalhes de acesso ao banco de dados.	Representar e manipular dados em um único objeto, combinando a lógica de negócios com as operações de persistência.	Construir consultas de banco de dados de forma programática e conveniente.



5. Data Access Object (DAO)

	DAO	ORM	Repository Pattern	Active Record Pattern	Query Builders
Nível de abstração	Baixo nível	Alto nível	Alto nível	Baixo nível	Médio nível



5. Data Access Object (DAO)

	DAO	ORM	Repository Pattern	Active Record Pattern	Query Builders
Flexibilidade	Flexível na escolha de tecnologias de acesso a dados e manipulação manual de consultas e transações.	Flexível na escolha de tecnologias de mapeamento objeto-relacional e automação de tarefas de persistência.	Flexível na escolha de tecnologias de acesso a dados, manipulação manual de consultas e personalização da lógica de acesso aos dados.	Menos flexível, pois a lógica de acesso aos dados está acoplada ao objeto de negócio.	Flexível na construção de consultas personalizadas e manipulação de consultas de forma programática.



5. Data Access Object (DAO)

	DAO	ORM	Repository Pattern	Active Record Pattern	Query Builders
Complexidade	Requer mais código manual para escrever as consultas e gerenciar as transações.	Reduz a necessidade de escrever consultas SQL manualmente e fornece recursos automatizados para mapeamento objeto-relacional.	Moderada, requer implementação de repositórios para cada entidade e personalização da lógica de acesso aos dados.	Moderada, requer implementação de métodos de persistência em cada objeto de negócio.	Moderada, requer familiaridade com a sintaxe e a API da biblioteca de Builders de Consulta.



5. Data Access Object (DAO)

	DAO	ORM	Repository Pattern	Active Record Pattern	Query Builders
Suporte ao SQL	Permite escrever consultas SQL diretamente.	Abstrai a necessidade de escrever consultas SQL diretamente, mas permite o uso quando necessário.	Geralmente, permite escrever consultas personalizadas em SQL quando necessário.	Não é focado em consultas SQL diretas, mas algumas implementações podem permitir o uso de SQL.	Geralmente, permite escrever consultas personalizadas em SQL quando necessário.



5. Data Access Object (DAO)

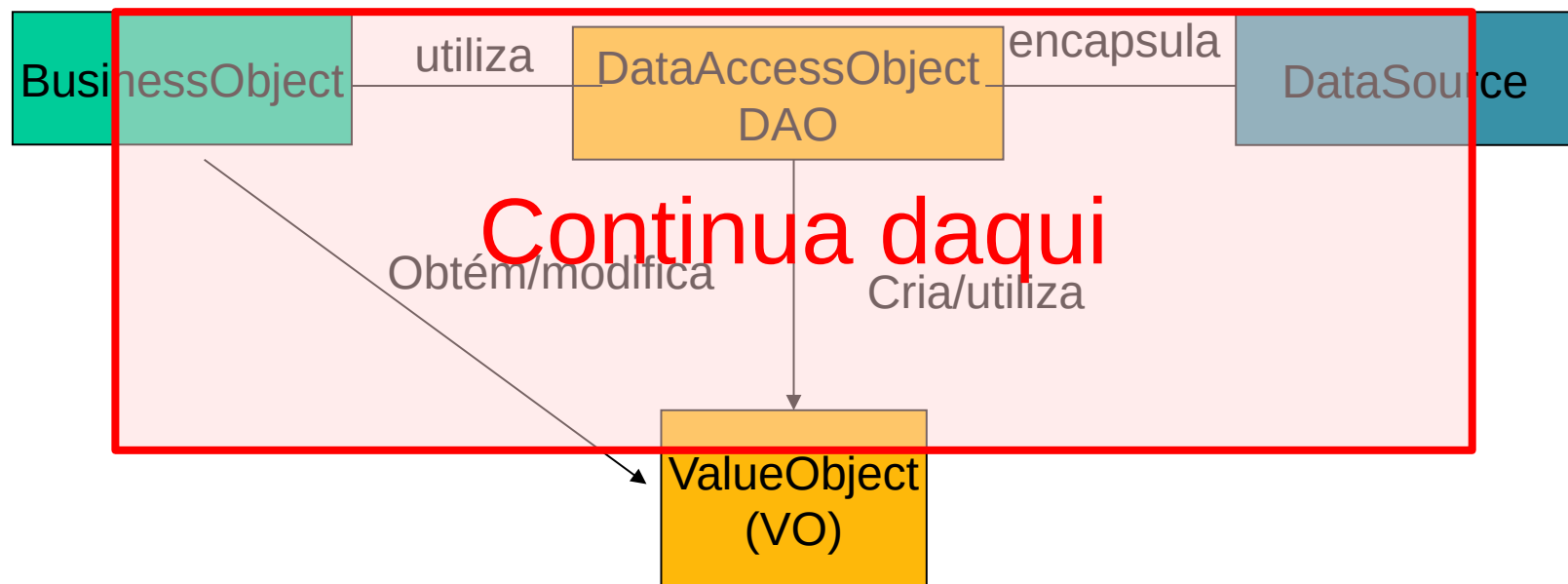
- O objetivo principal de um DAO é **fornecer uma interface simplificada e consistente** para que a aplicação possa interagir com o banco de dados, **ocultando os detalhes de implementação** e fornecendo métodos padronizados para acessar, inserir, atualizar e excluir dados.
- DAO é utilizado para extrair e encapsular todo os acessos a origem dos dados.



5. Data Access Object (DAO)

- O DAO gerencia a conexão com a origem dos dados para manipulação dos dados selecionados.
- Utilizado para construir a camada de persistência.

5. Data Access Object (DAO)





Dúvidas?

