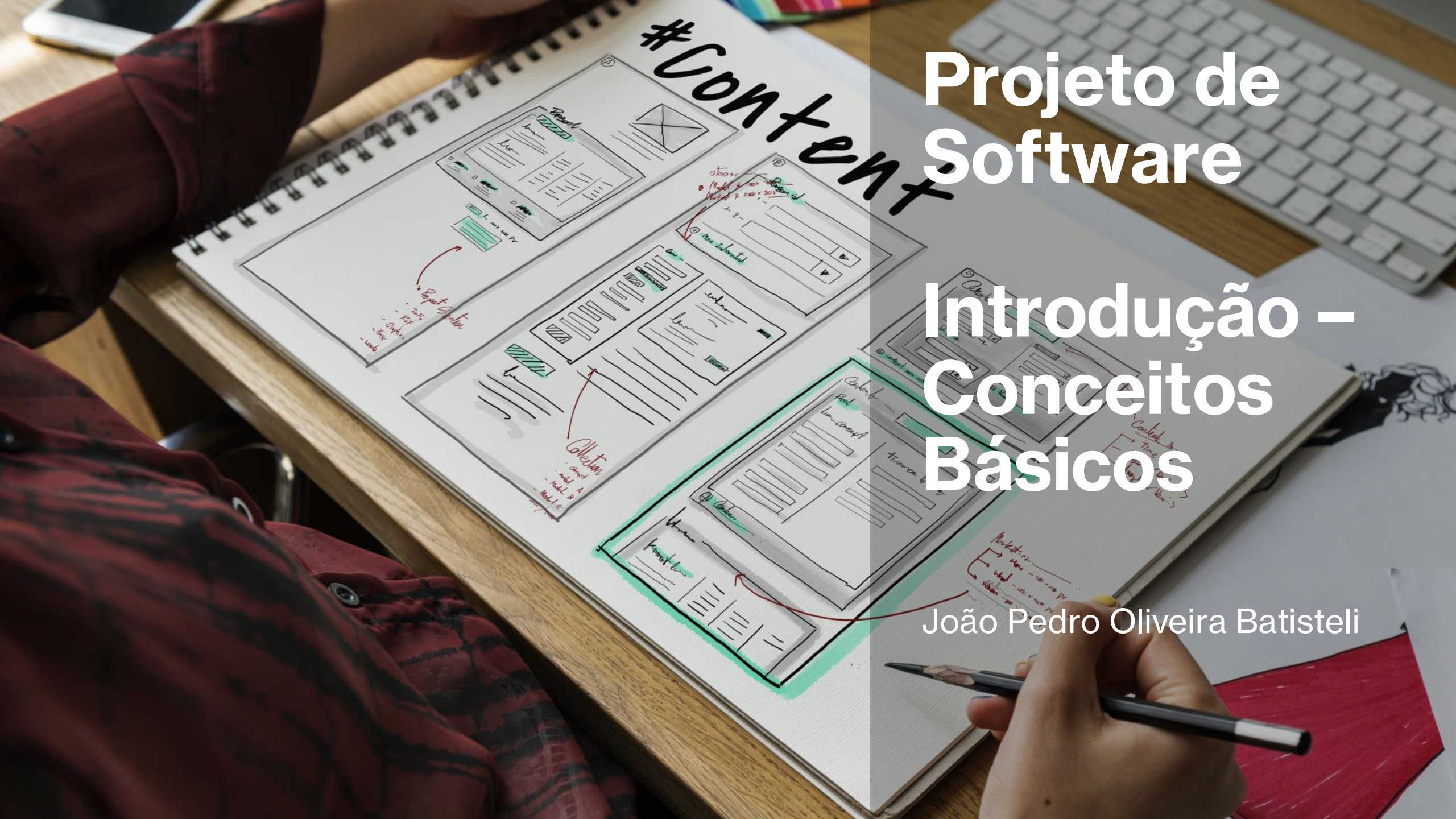


Projeto de Software

Introdução – Conceitos Básicos

João Pedro Oliveira Batisteli

Conceitos Básicos Aplicados a Projetos

- Abstração
- Arquitetura
- Padrões
- Separação por interesse
- Modularidade
- Ocultação de Informação

Abstração

“Ação de abstrair, de analisar isoladamente um aspecto, contido num todo, sem ter em consideração sua relação com a realidade.”

[Filosofia] *”Operação mental através da qual elementos e aspectos são isolados, somente no pensamento, sendo que (na totalidade) não existem isoladamente; resultado dessa operação.”*

Abstração

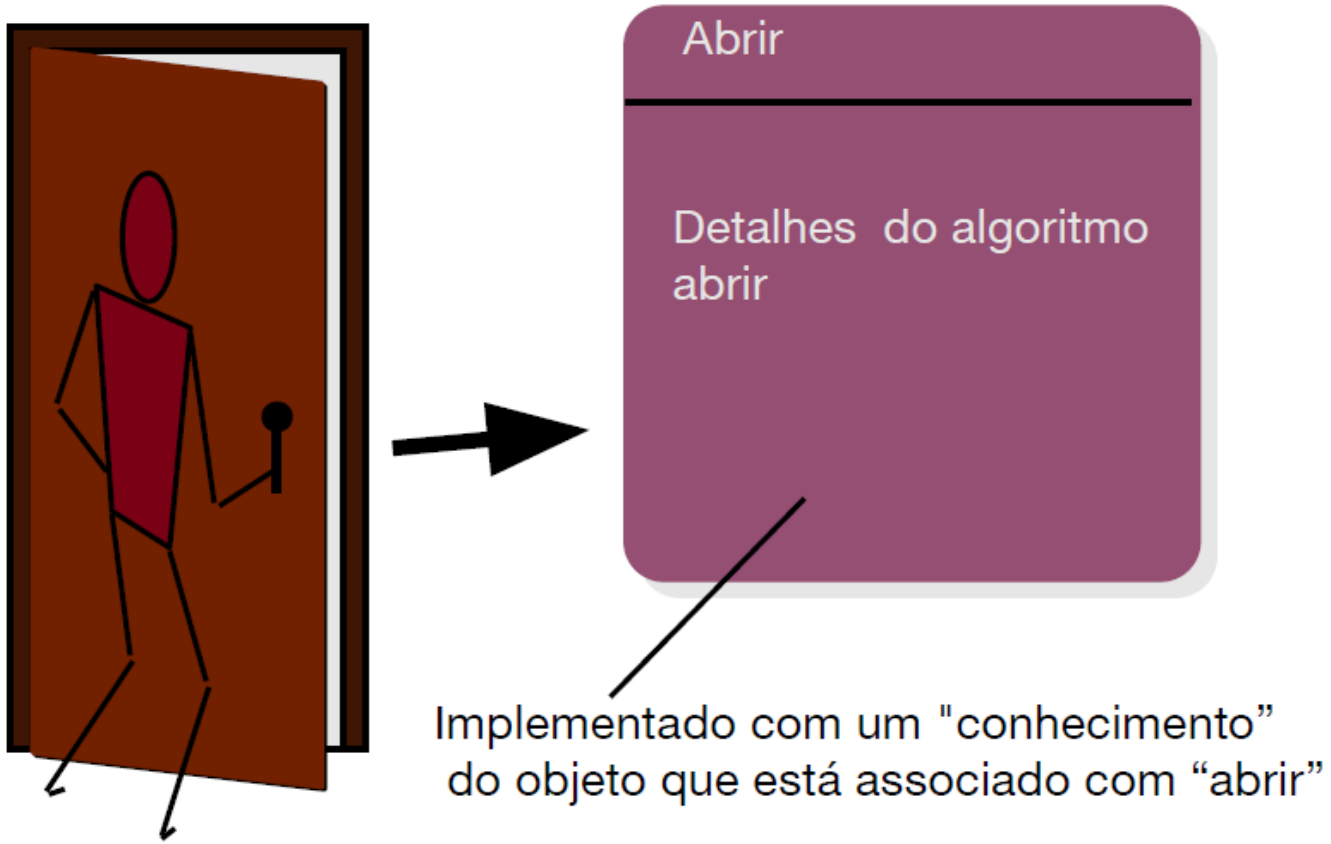
- Abstração é a capacidade de enxergar um problema ou sistema em um nível de generalização, concentrando-se apenas nas informações relevantes para o momento, sem se preocupar com detalhes de implementação.
- Ao projetar soluções modulares, surgem diferentes níveis de abstração, cada um ajudando a quebrar a complexidade do problema em partes mais simples.
- Cada etapa no desenvolvimento de um sistema representa um refinamento no nível de abstração, aproximando a solução de algo executável sem perder a clareza.

Abstração

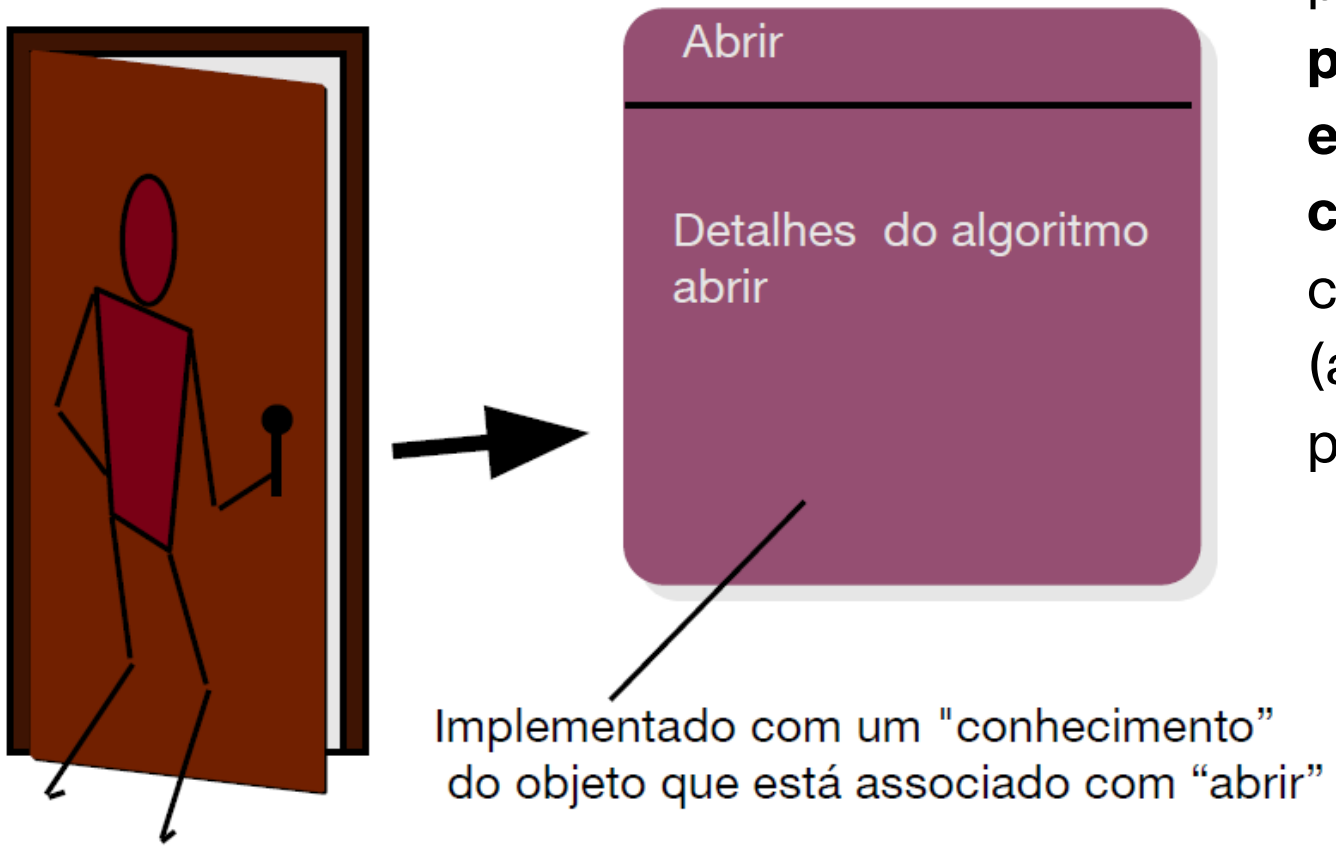
Tipos de abstração

1. **Abstração Procedural (funcional):** “esconde” como uma tarefa/funcionalidade é realizada, expondo apenas o que ela faz (ex.: funções, procedimentos ou métodos).
2. **Abstração de Dados: c**onsiste em ocultar como os dados são representados ou estruturados internamente, expondo apenas operações para manipulá-los.

Abstração Procedural

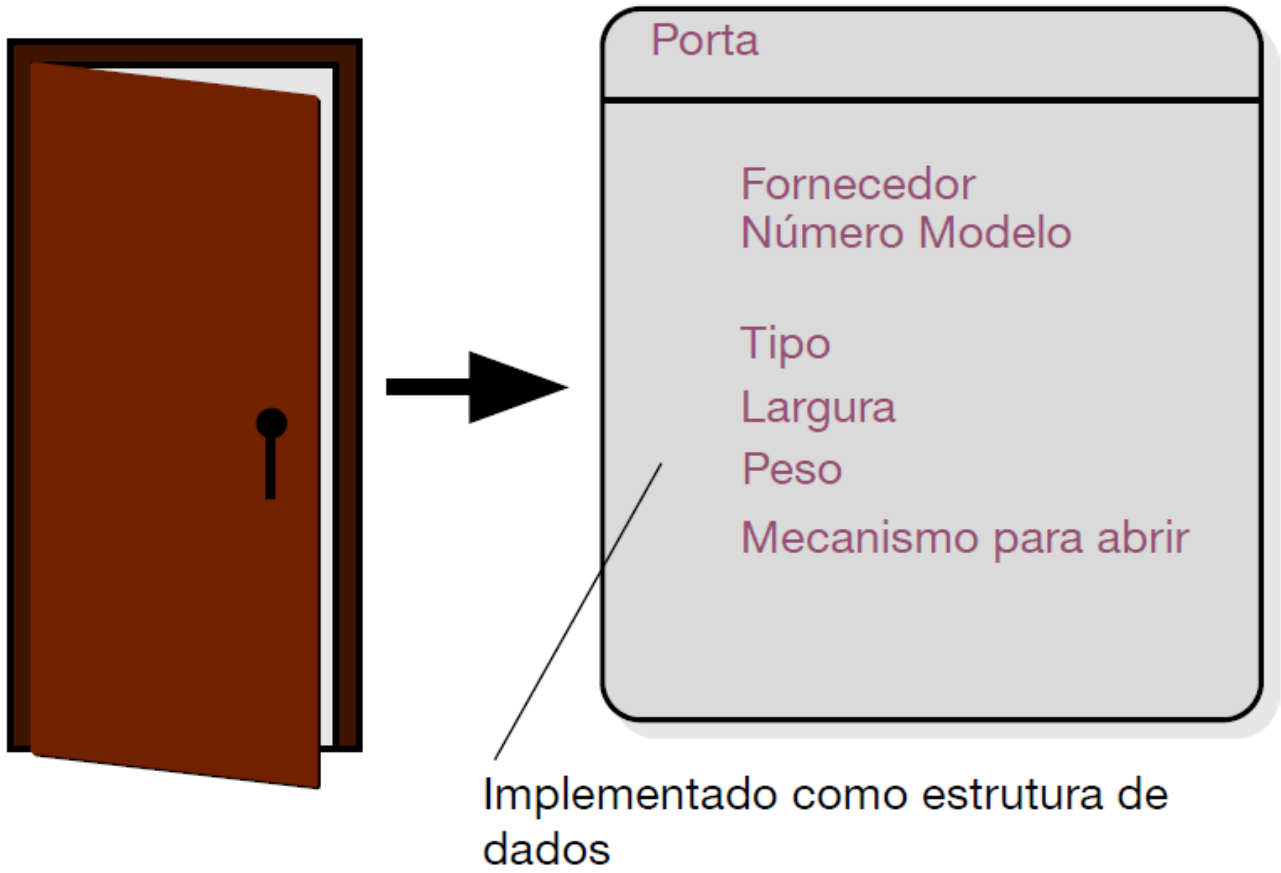


Abstração Procedural

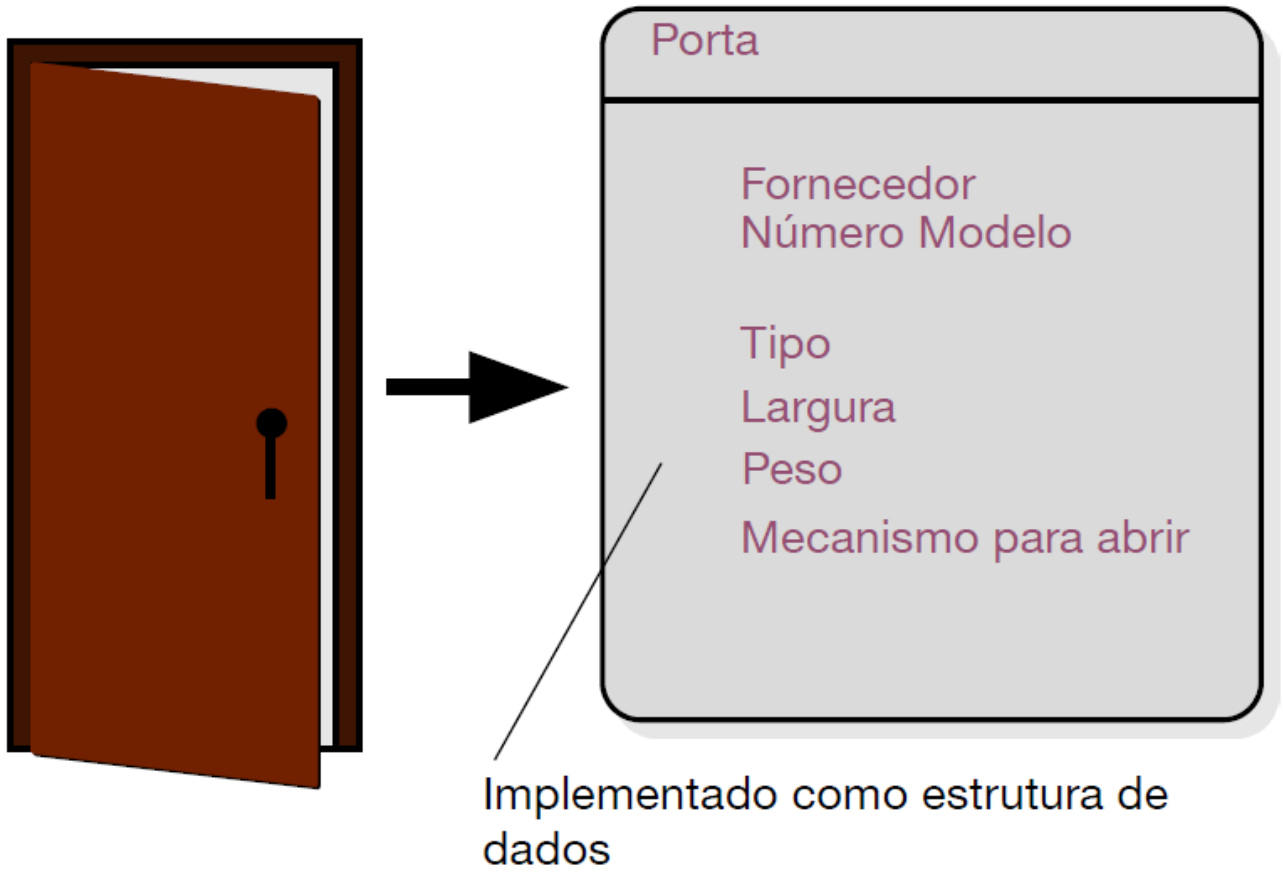


- Quem usa o procedimento (**ator**) **só precisa saber que existe uma operação chamada “abrir”**, e como chamá-la (assinatura da função, parâmetros, etc.).

Abstração de Dados



Abstração de Dados



- Especificar **quais informações** são necessárias para descrever e manipular um objeto.
- Ocultar detalhes sobre **como** esses dados são armazenados fisicamente na memória ou transmitidos.

Arquitetura

- “Organização geral do software aos modos pelos quais disponibiliza integridade conceitual para um sistema”.

Arquitetura

- “Organização geral do software aos modos pelos quais disponibiliza integridade conceitual para um sistema”.

Para entender melhor:

- "**Organização geral do software**" → significa como o sistema está dividido em partes (como módulos, camadas ou serviços).
- “**Integridade conceitual**" → quer dizer que todas as partes do sistema seguem a mesma lógica, princípios e objetivos, tornando o sistema mais coerente, compreensível e fácil de manter.

Arquitetura

- É a estrutura ou a organização de componentes de programas, a maneira através da qual esses componentes interagem e a estrutura de dados que são usadas pelos componentes.
- A **arquitetura de software** trata da organização de um sistema em um nível de abstração mais alto do que aquele que envolve classes ou construções semelhantes.

Padrões

- Padrões de projeto são inspirados em uma ideia proposta por Cristopher Alexander, um arquiteto (de construções civis e não de software) e professor da Universidade de Berkeley.
- Alexander lançou um livro chamado *A Patterns Language*, no qual ele documenta diversos padrões para construção de cidades e prédios. Segundo Alexander:

> "*Cada padrão descreve um problema que sempre ocorre em nosso contexto e uma solução para ele, de forma que possamos usá-la um milhão de vezes.*"

Padrões

- Padrões de projeto são inspirados em uma ideia proposta por Cristopher Alexander, um arquiteto (de construções civis e não de software) e professor da Universidade de Berkeley.
- Alexander lançou um livro chamado *A Patterns Language*, no qual ele documenta diversos padrões para construção de cidades e prédios. Segundo Alexander:

> "*Cada padrão descreve um problema que sempre ocorre em nosso contexto e uma solução para ele, de forma que possamos usá-la um milhão de vezes.*"

- Padrão é parte de um conhecimento consolidado já existente que transmite a essência de uma solução comprovada para um problema recorrente em certo contexto.

Padrões

- Em 1995, Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides lançaram um livro adaptando as ideias de Alexander para o mundo de desenvolvimento de software.

- Os autores propuseram um catálogo com soluções para resolver problemas recorrentes em projeto de software. Eles deram o nome de **Padrões de Projeto** às soluções propostas no livro.

- "*Padrões de projeto descrevem objetos e classes que se relacionam para resolver um problema de projeto genérico em um contexto particular.*"

Separação por Interesse

- **Ideia central:** Um sistema complexo torna-se mais compreensível e gerenciável quando é dividido em partes menores e com responsabilidades bem definidas.
- “A complexidade percebida de dois problemas, quando estes são combinados, normalmente é maior do que a soma da complexidade percebida quando cada um deles é tomado individualmente”.
- **Aplicação na Arquitetura de Software:**
  - Cada parte do sistema (módulo, camada ou componente) deve se concentrar em **um único conjunto de responsabilidades**.
  - Isso facilita o desenvolvimento, testes, manutenção e evolução do sistema.
  - Exemplo: Separar lógica de negócio, interface com o usuário e acesso a dados.

Modularidade

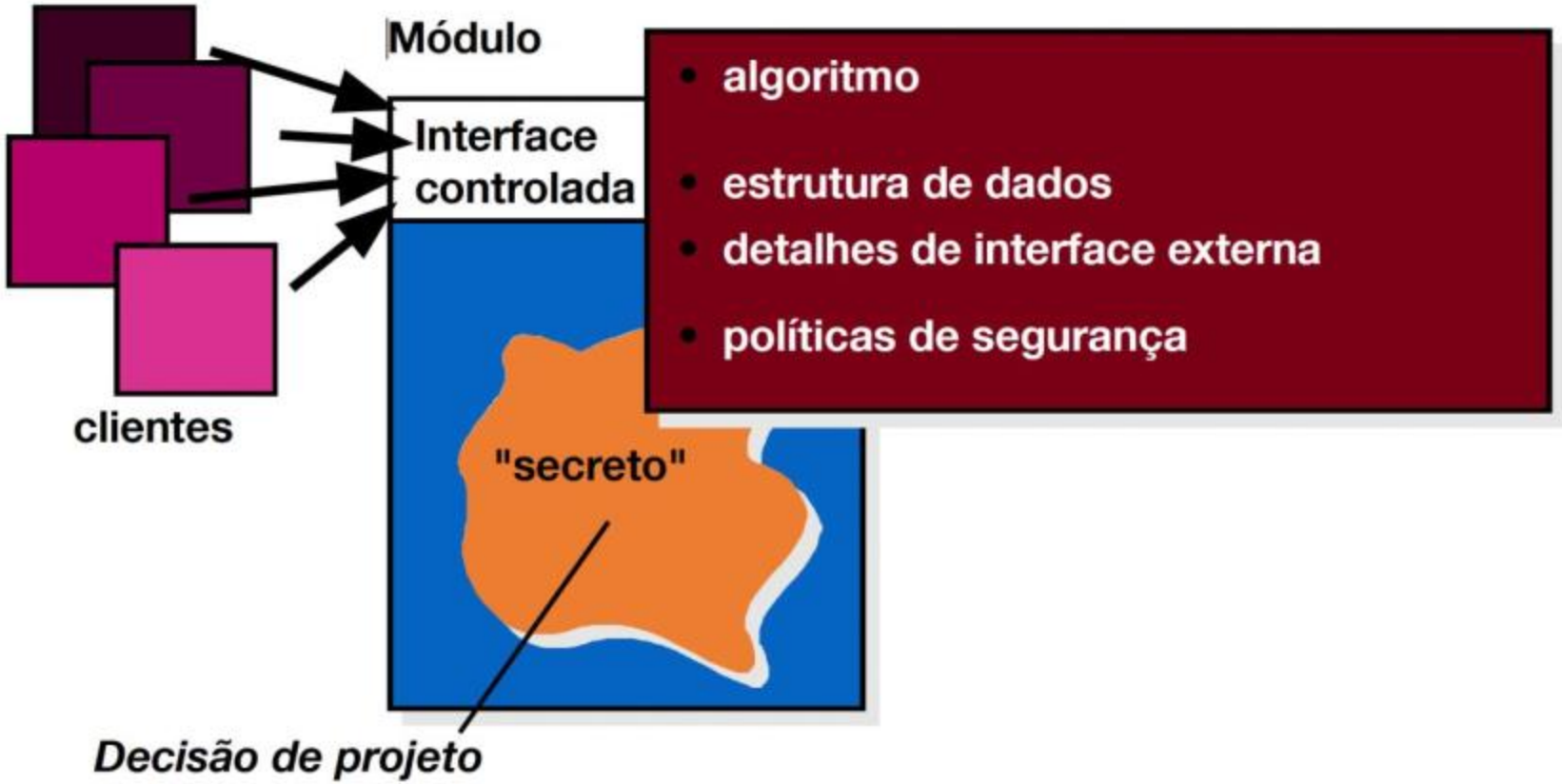
- A modularidade é uma **forma prática de aplicar a separação de interesses**, dividindo um sistema complexo em **módulos independentes**, cada um responsável por uma parte específica da funcionalidade.
- **Um paralelo com a neurociência**: A mente humana tem limitações cognitivas e não consegue compreender sistemas muito grandes como um todo de forma eficiente.
- A divisão em módulos permite focar em partes menores, compreensíveis e gerenciáveis do sistema.
- Modularidade 🤝 abstração:

“A abstração nos permite focar no propósito de cada módulo sem nos preocupar com seus detalhes internos.”

Ocultação de Informação

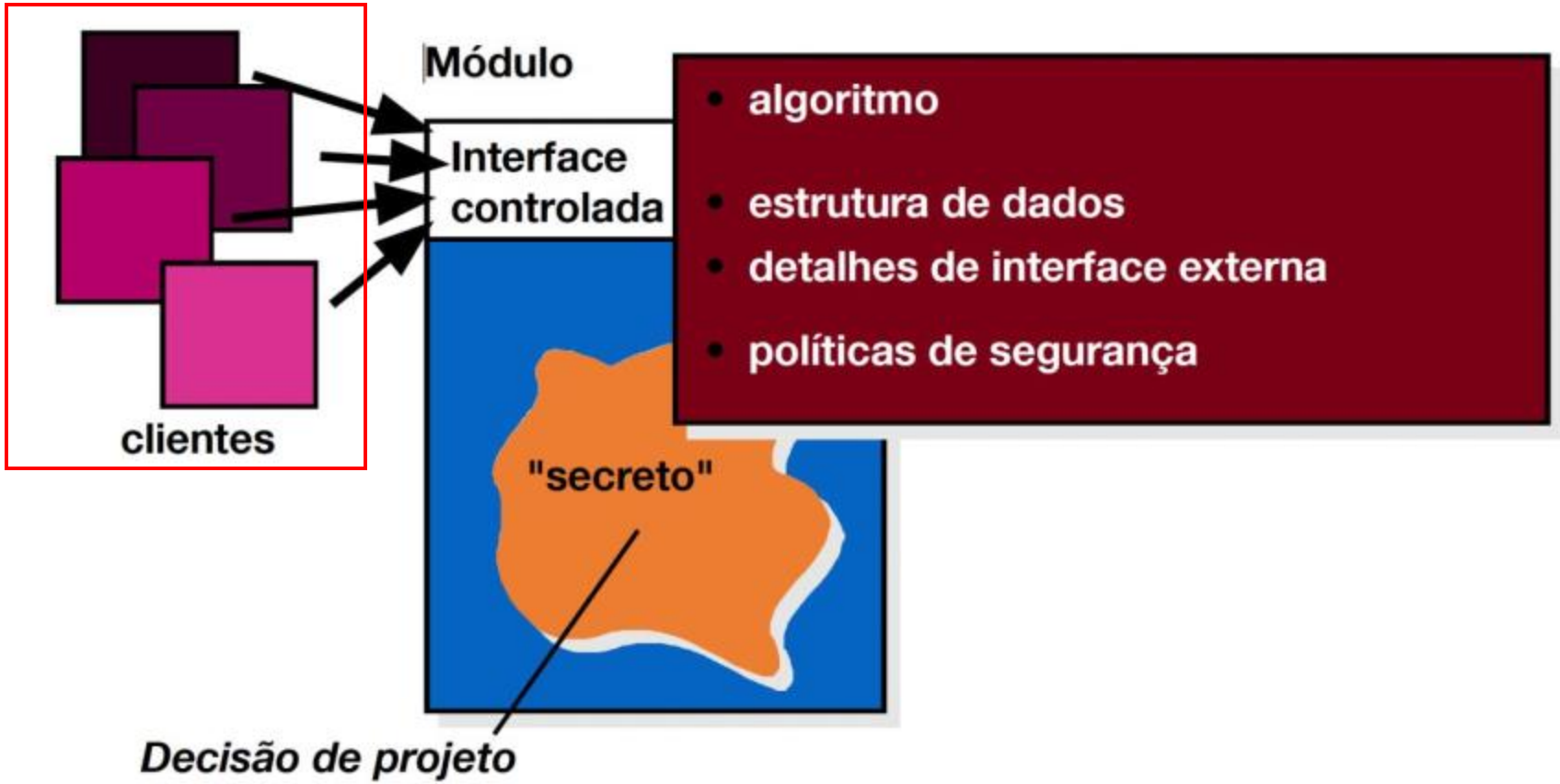
- A ocultação de informação é um princípio fundamental tanto na **Programação Orientada a Objetos (POO)** quanto no **projeto de software em geral**.
- Esse princípio afirma que **detalhes internos de um módulo devem ser escondidos (encapsulados)** dos demais módulos.
- Cada módulo deve esconder suas decisões de projeto internas, expondo apenas o que for necessário para que outros módulos interajam com ele.
- Módulos se comunicam entre si apenas por interfaces bem definidas (métodos, APIs, contratos).
- Os detalhes de como um módulo faz algo (estrutura interna, algoritmos, formatos) não devem ser visíveis nem acessíveis diretamente por outros módulos.

Ocultação de Informação



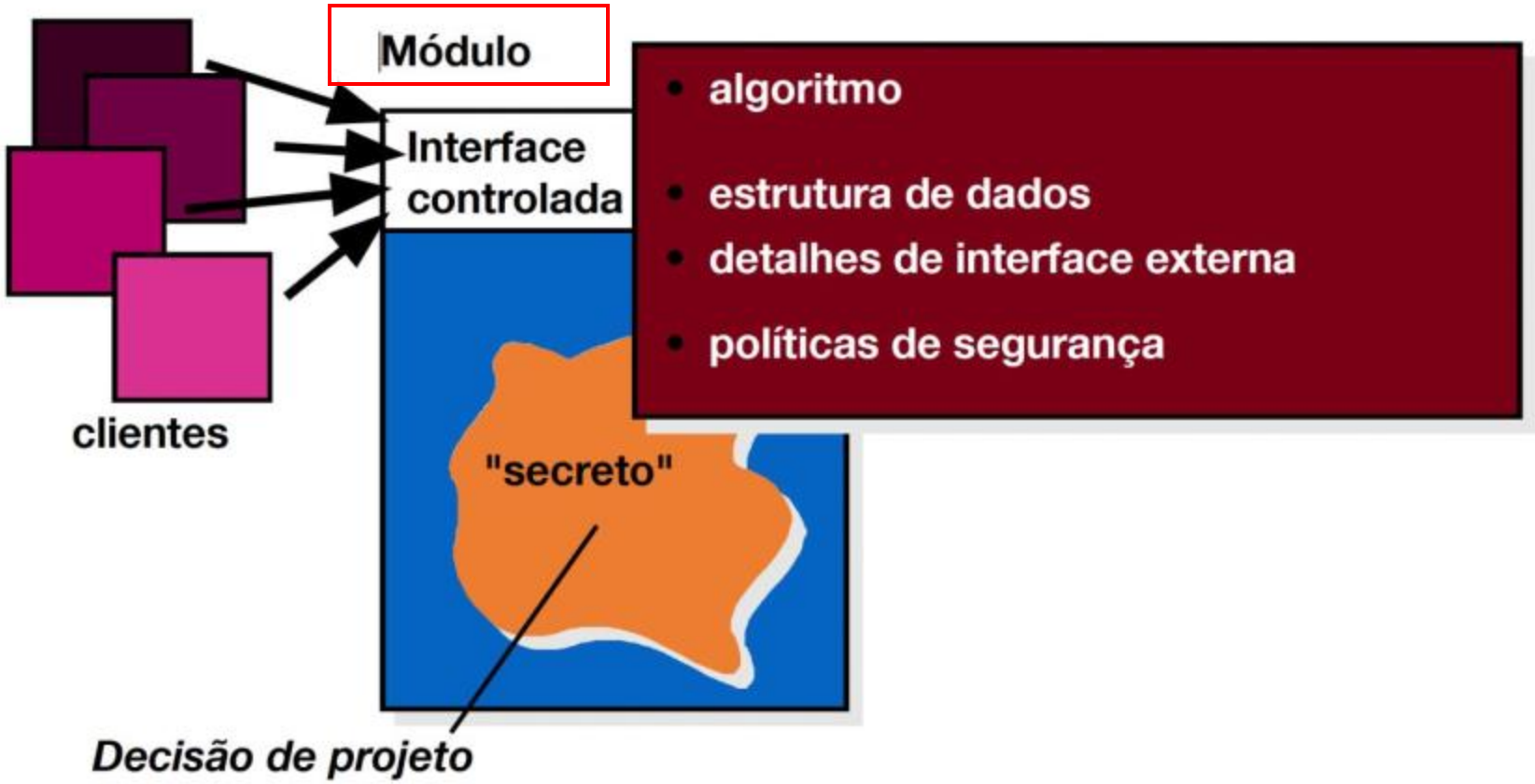
Ocultação de Informação

Módulos, classes ou partes externas do sistema que precisam utilizar os serviços oferecidos pelo módulo principal.



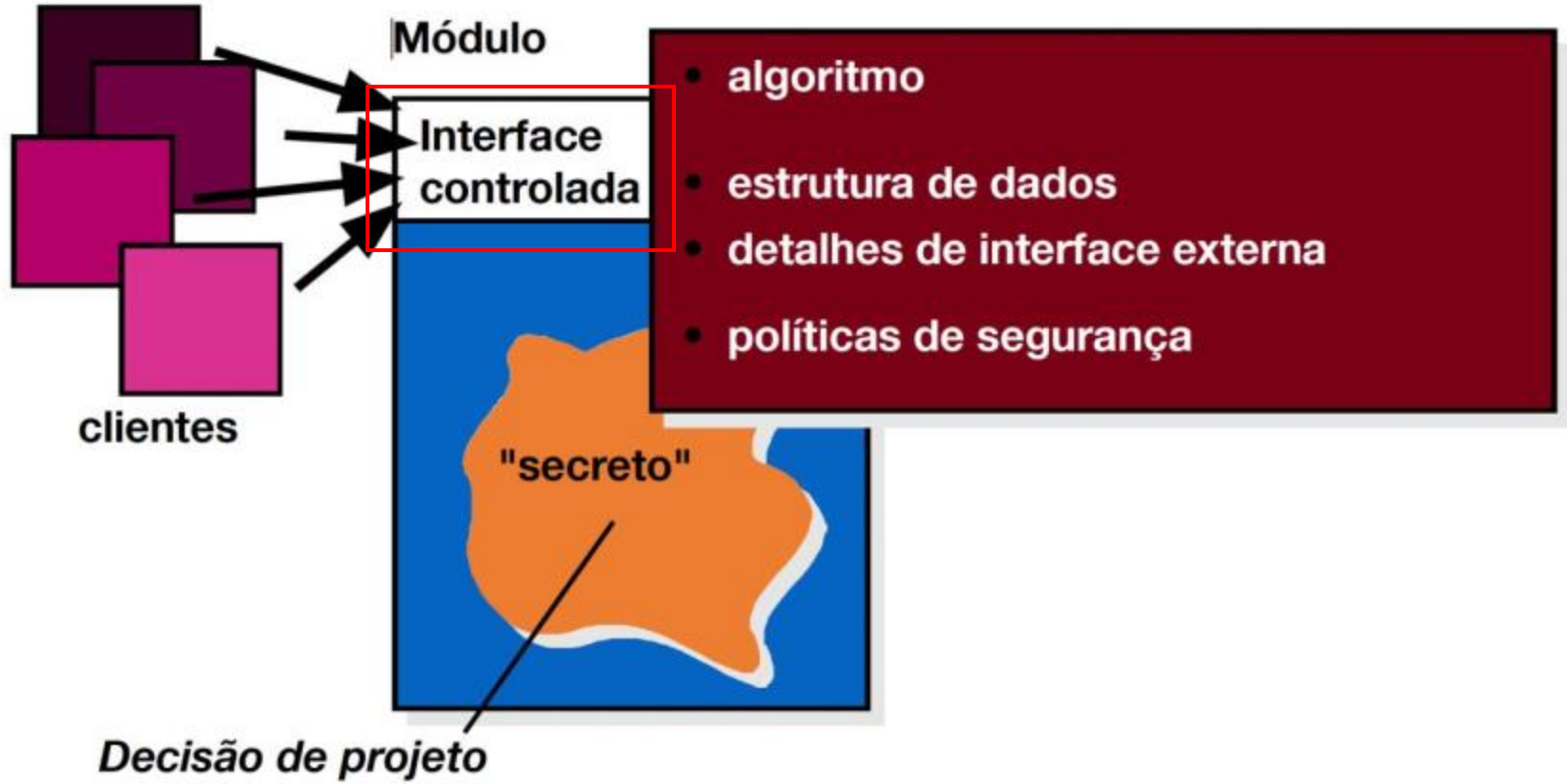
Ocultação de Informação

Componente que encapsula uma parte do sistema, responsável por alguma funcionalidade específica.

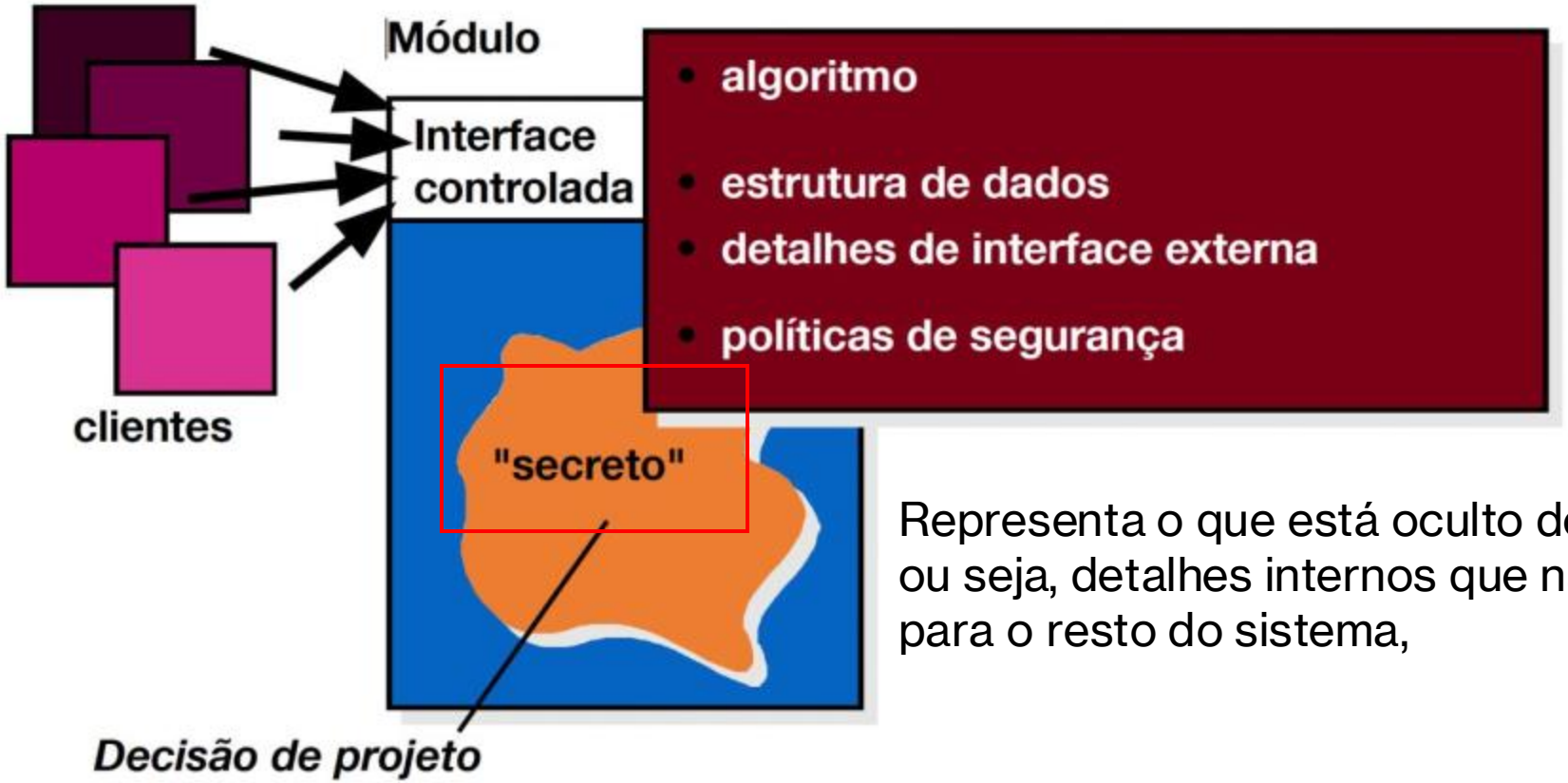


Ocultação de Informação

Unica maneira pela qual os clientes podem interagir com o módulo. A interface define os métodos, funções ou contratos públicos que o módulo oferece.



Ocultação de Informação



Representa o que está oculto dentro do módulo, ou seja, detalhes internos que não são expostos para o resto do sistema,