Serviços Web

Tecnologias de Internet

Prof Vitor Almeida dos Santos

Conteúdo

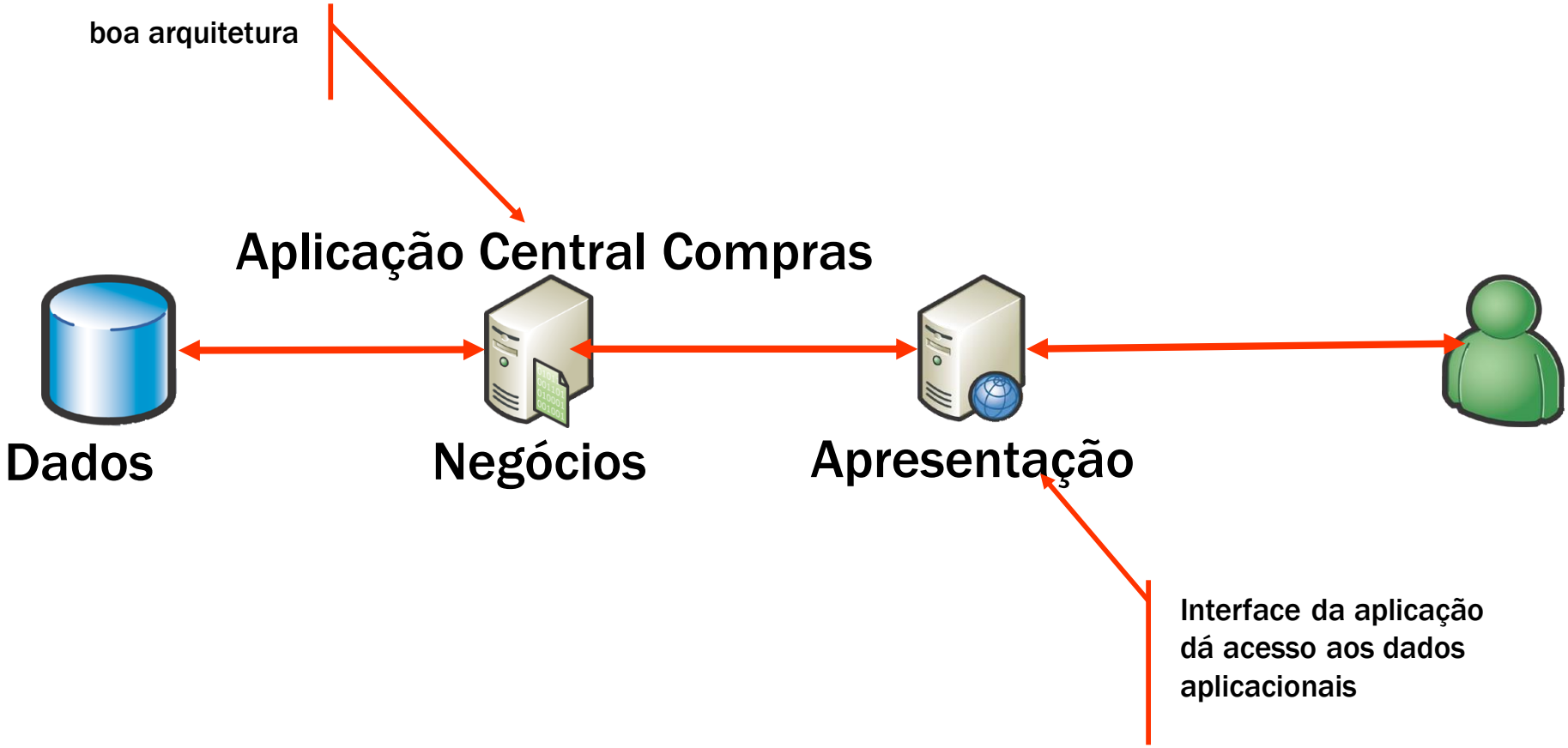
- Serviços web: aspectos teóricos
  - Introdução;
  - Arquitetura Orientada a Serviços;
- Desenvolvendo Serviços Web REST com .NET API

Serviços Web
Aspectos Teóricos

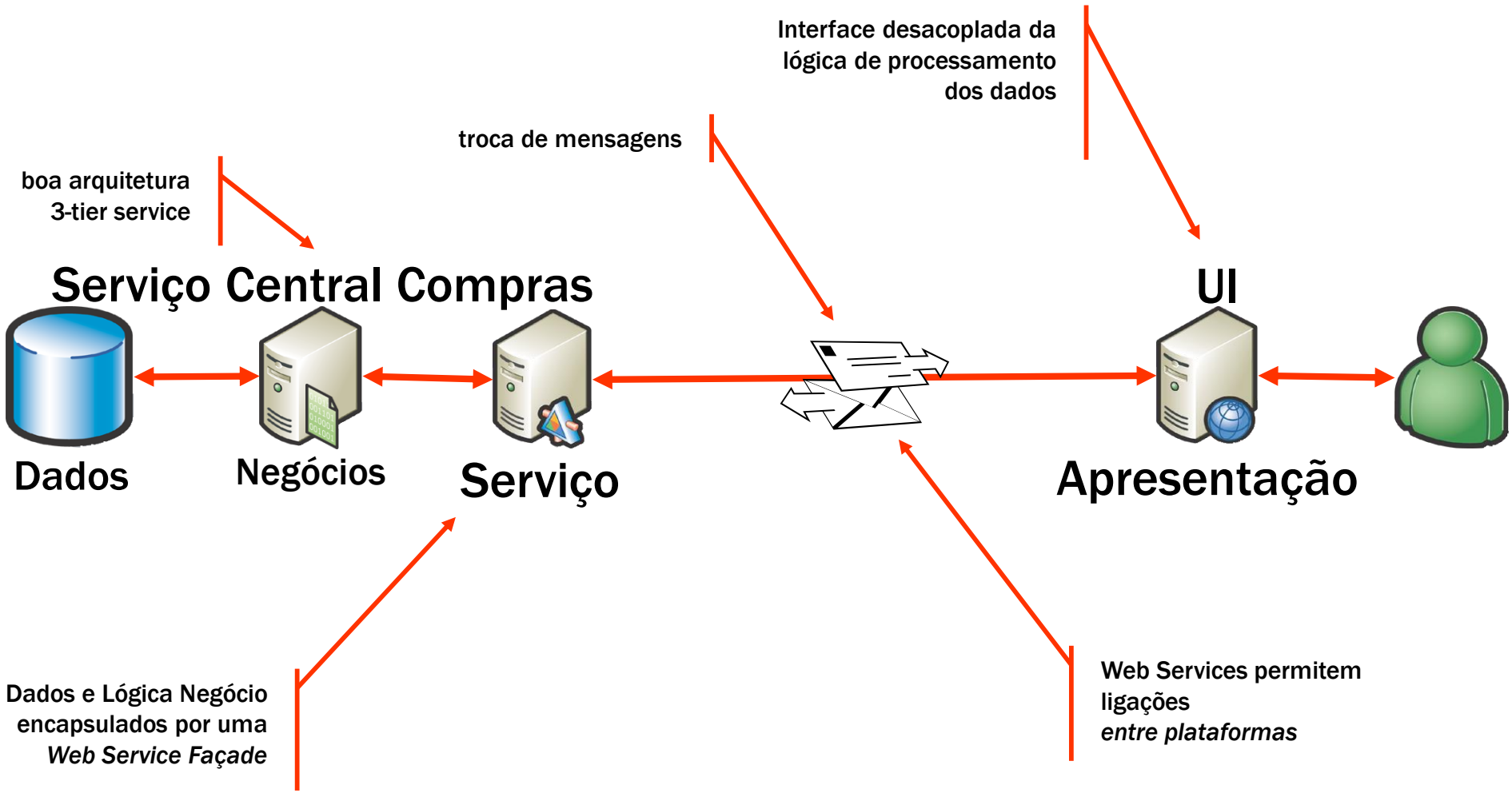
Introdução

- Qual o contexto das aplicações distribuídas comerciais hoje em dia?
  - Diversidade de **aplicações**: de grandes pacotes comerciais a aplicações desenvolvidas sob medida;
  - Diversidade de **fornecedores** de softwares;
  - Diversidade de **tecnologias**: cliente-servidor a multicamadas;
  - Diversidade de **plataformas**: mainframes, Unix, Windows etc.

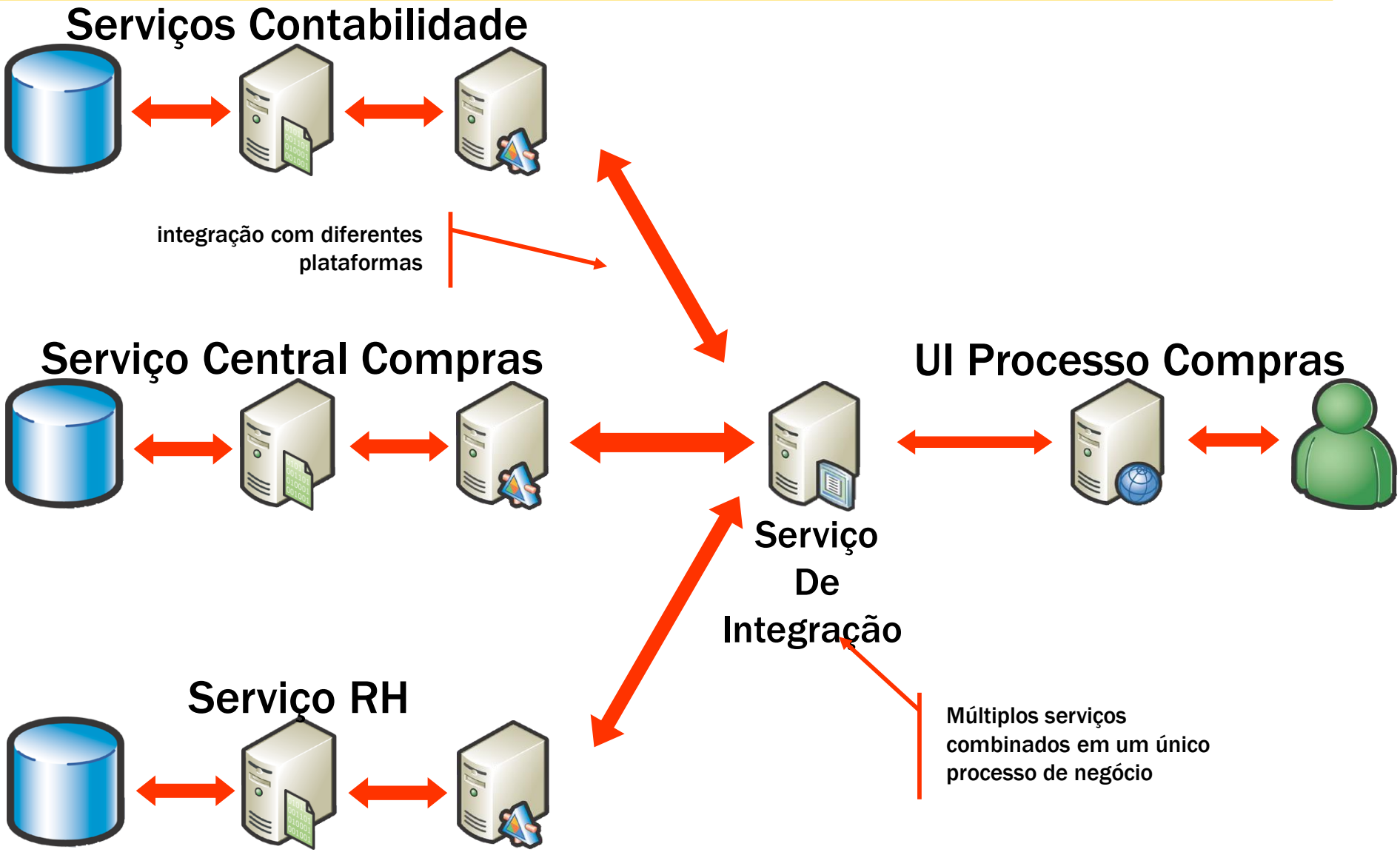
Arquiteturas de Software



Arquiteturas orientadas a serviços



Arquiteturas orientadas a serviços



Serviços Web – Definição

Um **serviço web** utiliza **padrões abertos** de comunicação e **definição de dados** comumente utilizados na web com o intuito de disponibilizar funcionalidades remotamente; um serviço web pode ser desenvolvimento com **qualquer tecnologia**, desde que siga tais padrões abertos.

Serviços Web – Principais Modelos

- Há dois modelos principais de Serviços Web:

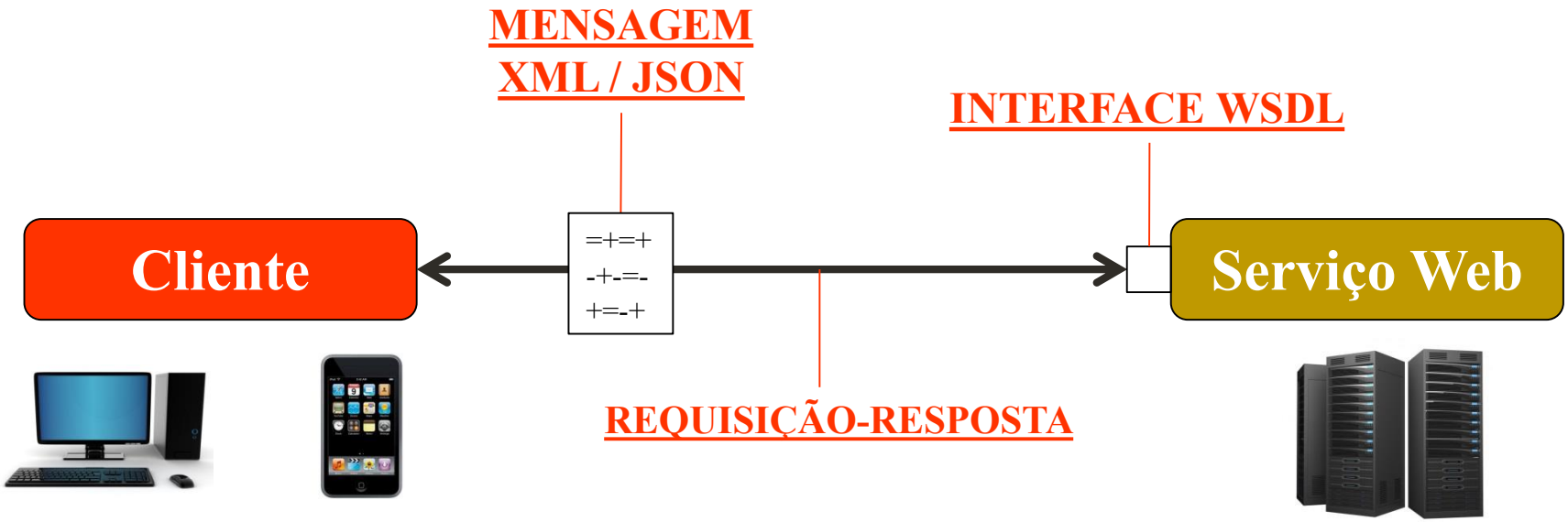
SOAP

- **Mensagens XML**
- **Possui contrato;**
- **Requisições GET/POST;**
- **Padrões de endereçamento, segurança, orquestração etc**

REST

- **Mensagens JSON puras;**
- **Não possui contrato;**
- **Requisições segundo métodos HTTP: GET, POST, PUT e DELETE;**
- **Sem outros padrões;**

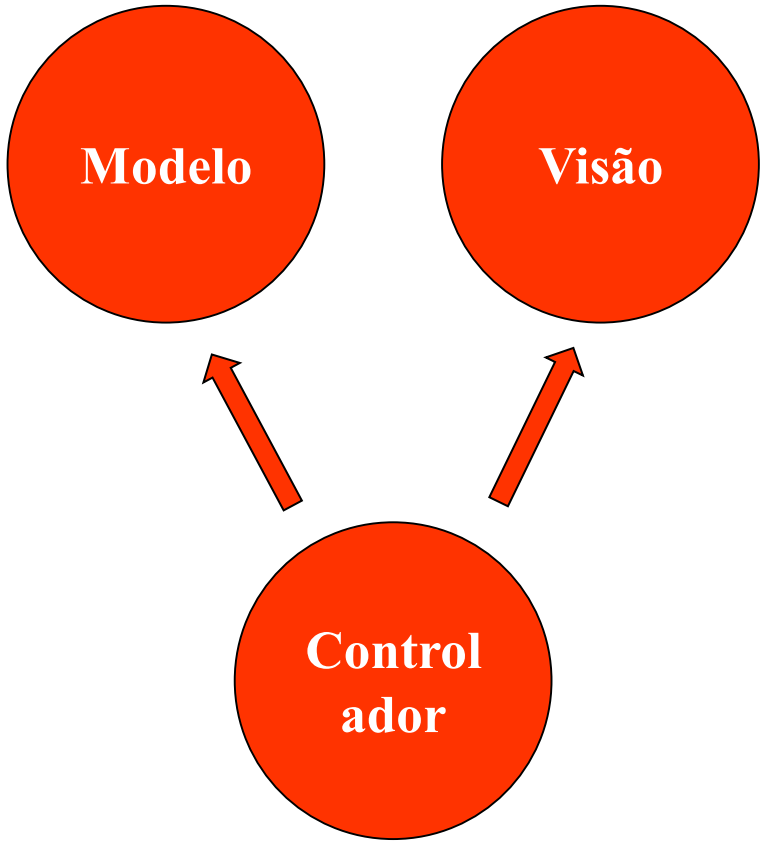
Serviços Web – SOAP



Serviços Web REST
com
.NET Web API

O Modelo MVC

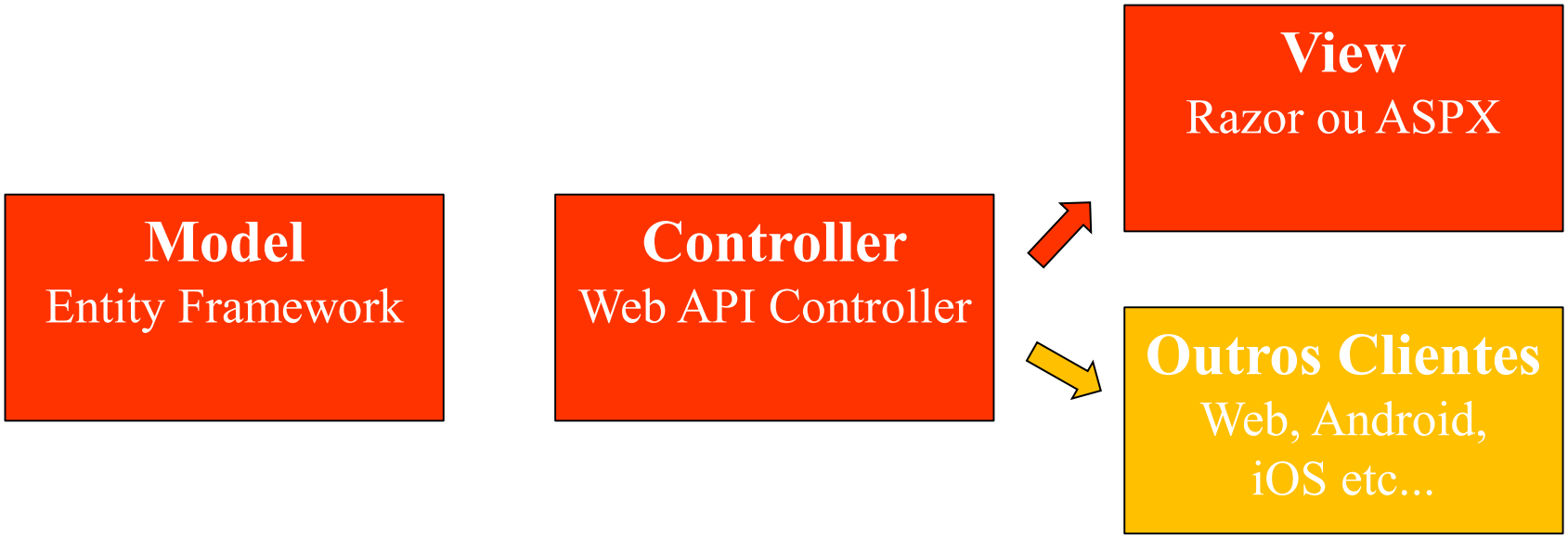
- Modelo
  - Representação do domínio de dados
  - Lógica de Negócios
  - Persistência
- Visão
  - Interface de usuário
  - Visualização do modelo
- Controlador
  - Intermediário
  - Recebe requisições, vincula dados à visão.



Desenvolvendo um Serviço Web REST com WEB API

O que é Web API?

- **Framework Microsoft .NET.**
- **Frequentemente usado dentro de um modelo MVC.**



Desenvolvendo um Serviço Web REST com Web API

O que é um *controller* Web API?

- **Uma Classe com métodos para manipulação de dados:**
  - **GET → consulta de um ou mais objetos; retorna um objeto ou uma coleção.**
  - **POST → inserção de um objeto; recebe os campos do objeto.**
  - **PUT → atualização de um objeto; recebe os campos e o ID do objeto.**
  - **DELETE → remoção de um objeto; recebe o ID do objeto.**

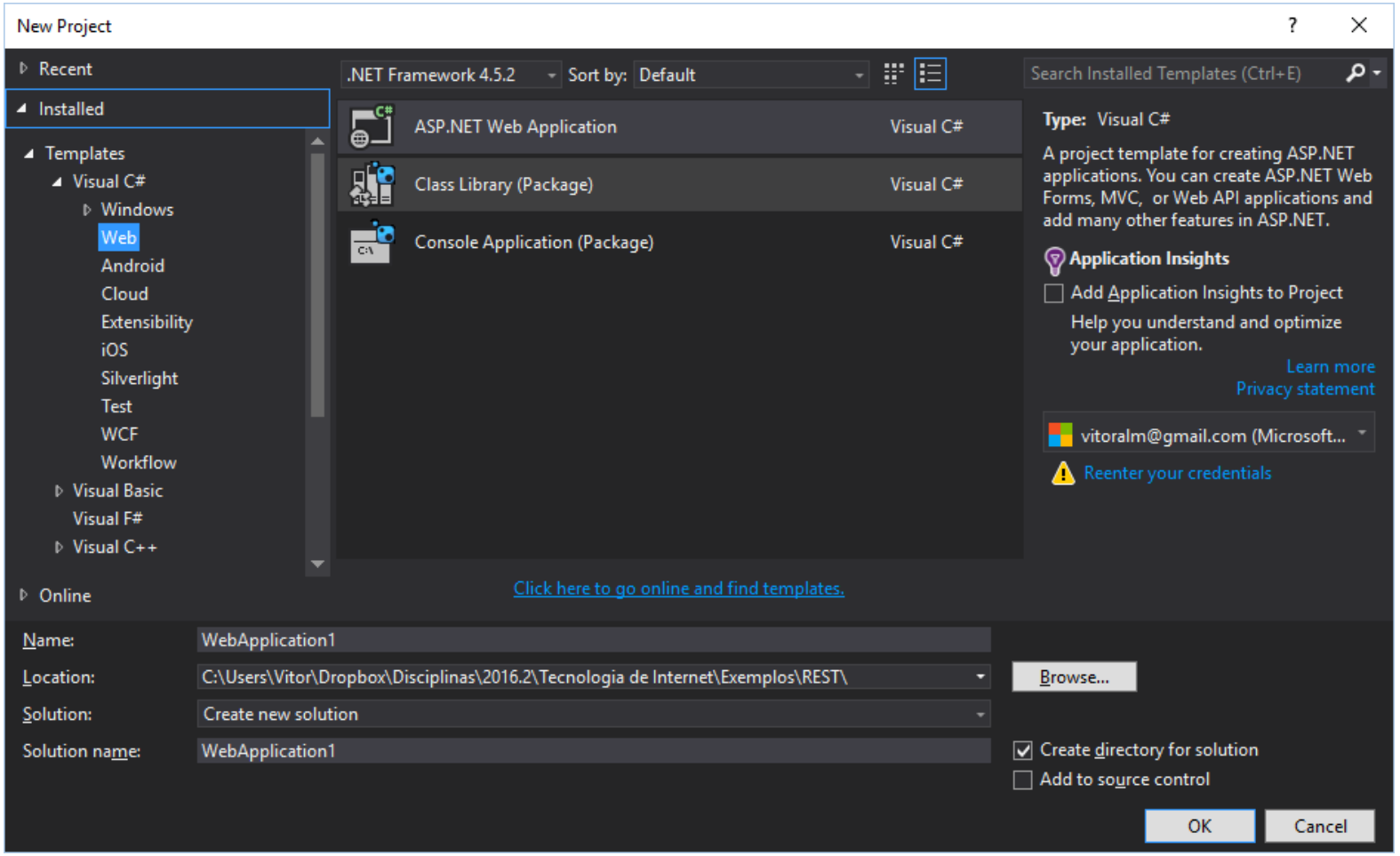
REST e o HTTP

- Usa os métodos do protocolo **HTTP** para manipular dados: POST, GET, PUT e DELETE, em uma analogia com **CRUD** seria: UPDATE, READ, CREATE, DELETE;
  - Requisição GET: consulta um recurso;
  - Requisição POST: insere um recurso;
  - Requisição PUT: atualiza um recurso;
  - Requisição DELETE: remove um recurso;

- Um recurso pode ser um arquivo xml, imagem etc;

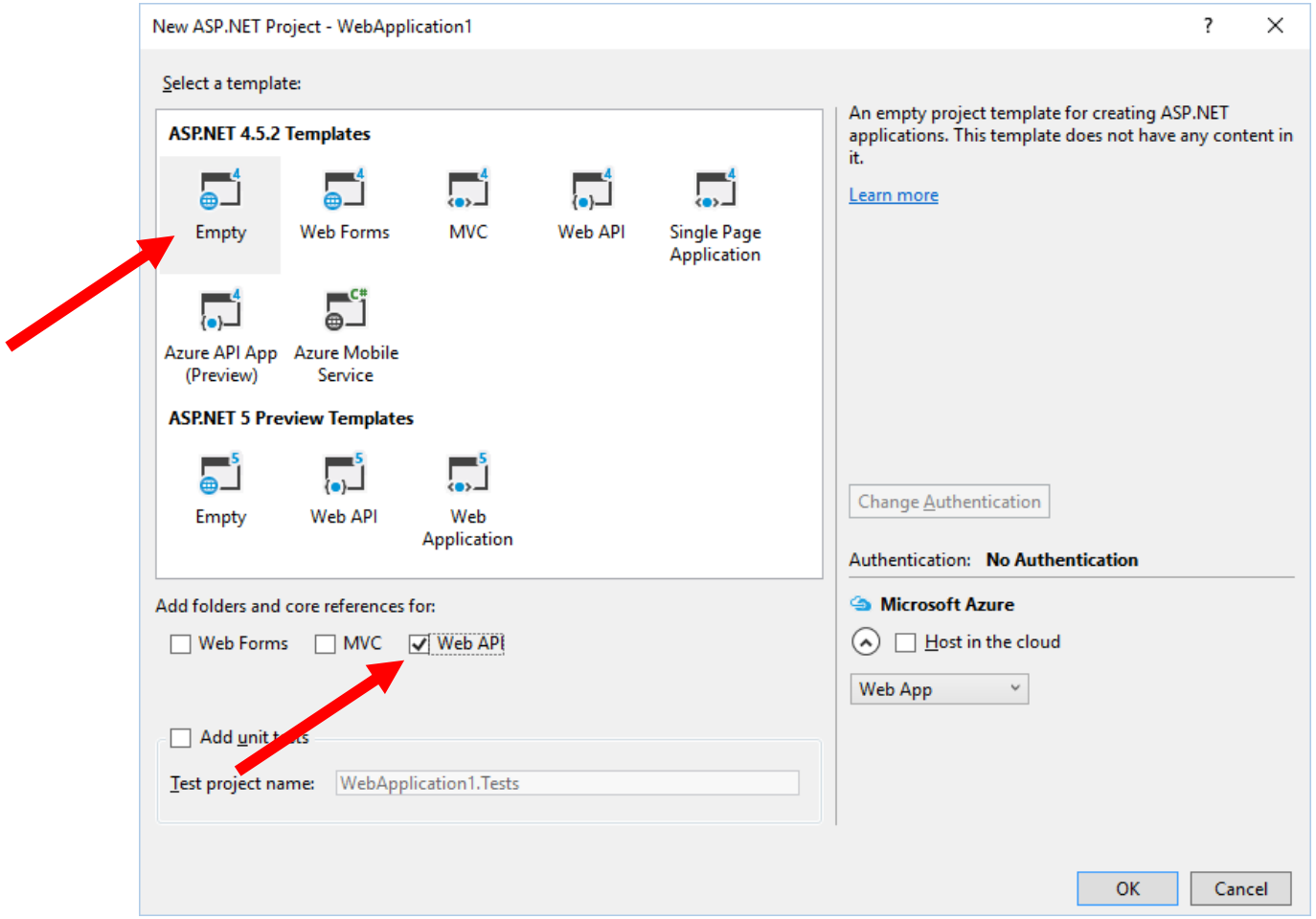
Desenvolvendo um Serviço Web REST com Web API – Parte 1

- File → New → Project → ASP.NET Web Application



Desenvolvendo um Serviço Web REST com Web API – Parte 2

- Template **Empty** e referência a **Web API**;



Desenvolvendo um Serviço Web REST com Web API – Parte 3

- Botão direito sobre a pasta Controllers → Add → Web API Controller Class.

- **Observe os métodos presentes.**

```
public class ValuesController : ApiController
{
    // GET api/<controller>
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/<controller>/5
    public string Get(int id)
    {
        return "value";
    }

    // POST api/<controller>
    public void Post([FromBody]string value)
    {
    }

    // PUT api/<controller>/5
    public void Put(int id, [FromBody]string value)
    {
    }

    // DELETE api/<controller>/5
    public void Delete(int id)
    {
    }
}
```

Desenvolvendo um Serviço Web REST com Web API – Parte 4

- Manipulação de parâmetros
  - **`[FromBody]`** deve ser um tipo primitivo ou um objeto;
  - **`int id`** representa a chave primária; deve ser parâmetro de consulta (GET) e remoção (DELETE);

Desenvolvendo um Serviço Web REST com Web API – Parte 5

- Códigos de retorno:
  - Padrão: retorno 200 (OK);
  - Para personalizar, use retorno **`IHttpActionResult`** nos métodos; Exemplo:

```
// POST api/<controller>
public IHttpActionResult Post([FromBody]Produto p)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);
    else
    {
        Produto.inserir(p);
        return StatusCode(HttpStatusCode.NoContent);
    }
}
```

Respostas HTTP mais comuns

Resposta	Código	Definição
InternalServerError	500	Erro no servidor.
BadRequest	400	Servidor não conseguiu processar; parâmetros errados possivelmente.
Forbidden	403	Servidor não aceitou a requisição.
NotFound	404	Recurso não existe no servidor.
NoContent	204	Sucesso, mas sem conteúdo de retorno.
OK	200	Sucesso.
Created	201	Recurso criado com sucesso.

Exemplo de uso:

```
return StatusCode(HttpStatusCode.NoContent);
```

Desenvolvendo um Serviço Web REST com Web API – Parte 6

- Configurações Gerais
  - Em WebApiConfig acrescente:

```
config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));
```

Desenvolvendo um Serviço Web REST com Web API – Parte 7

- Baixe o seguinte pacote via NuGet:
  Microsoft.AspNet.WebApi.Cors

- Inclua o seguinte comando no método Register do seu WebAPIConfig:

```
config.EnableCors();
```

- Inclua o seguinte atributo sobre seu controller:

```
[EnableCors(origins: "*",  headers: "*",
methods: "*")]
```

Fontes de Pesquisa

- Web Services
  - Sistemas Distribuídos, George Couloures, Cap 19: Serviços Web
- Serviços REST
  - https://www.infoq.com/br/articles/rest-introduction
  - http://www.restapitutorial.com/
- Web API
  - https://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api
  - https://msdn.microsoft.com/pt-br/library/dn450975.aspx