embarcadero

Embarcadero Conference

Um único esforço, uma única base de código, múltiplas plataformas, múltiplos dispositivos

embarcadero

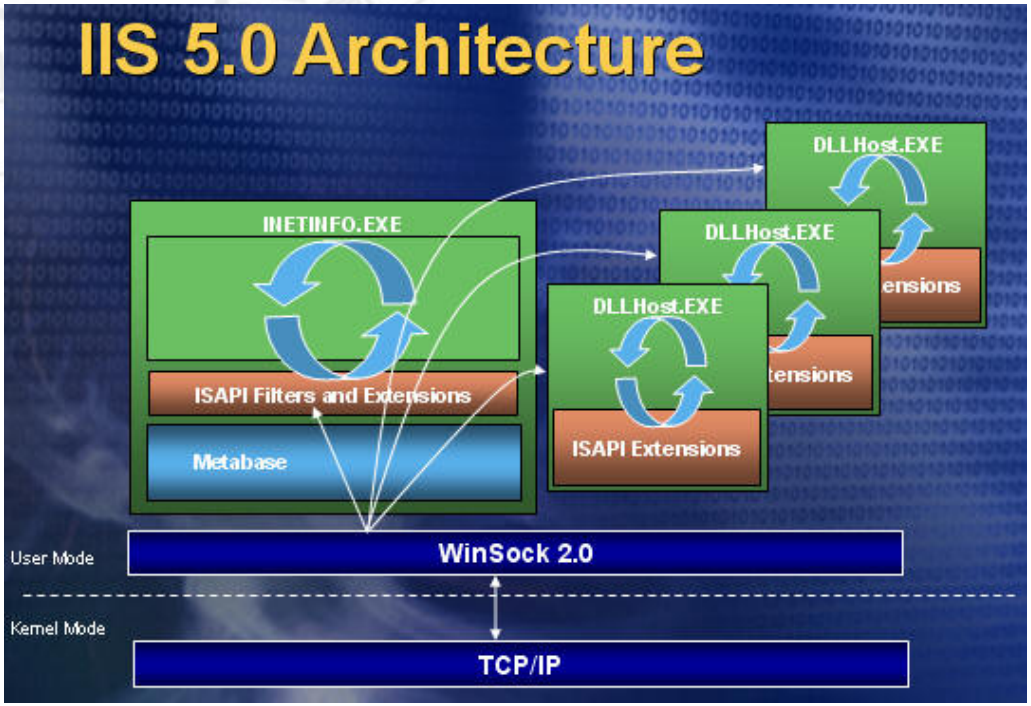
Wagner Landgraf

Distribuindo Servidores HTTP e HTTPS na Nuvem

Objetivo / Resumo

- Aplicações multi-camadas: servidor de aplicação
- Em geral, servidores usam protocolo HTTP
- Vantagens HTTP:
  - protocolo padrão
  - amplamente adotado
  - Uso via browsers
  - portas HTTP geralmente abertas para o lado cliente
- Servidor na rede local (LAN) – cenário comum
- Servidor na internet (WAN) – como fazer, de forma simples e baixo custo?
- Protegendo a comunicação: HTTP Seguro (HTTPS) e certificado digital

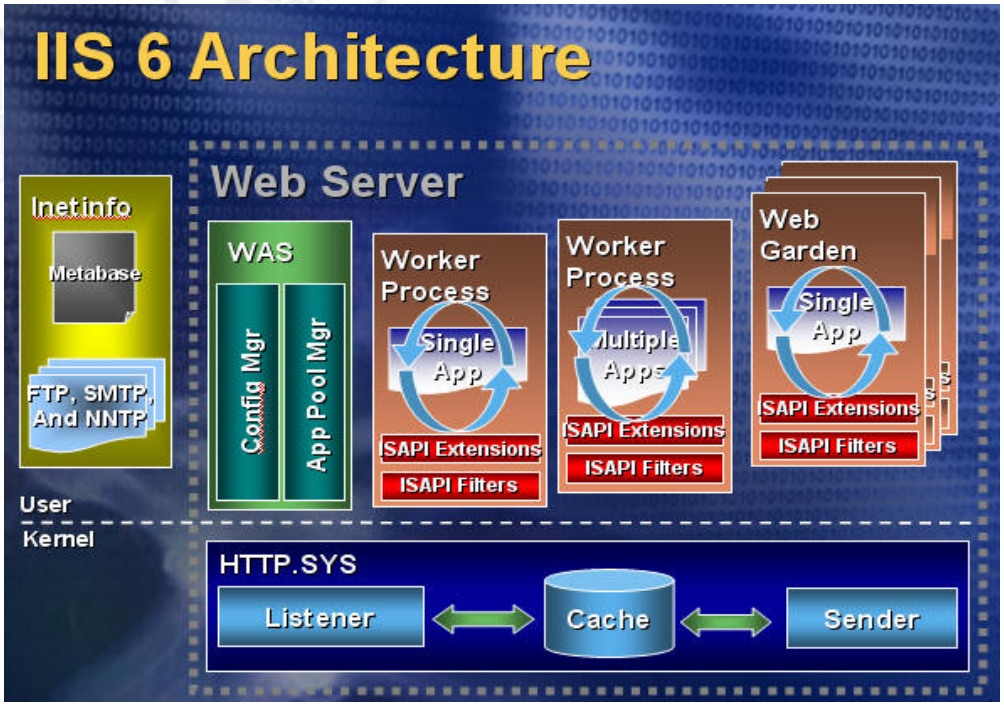
Servidor HTTP usando sockets



- Processamento manual das requisições via socket
- Cada aplicação/processo tem uma porta exclusiva
- Compatilhamento porta 80 via módulos IIS
- Compatível com Windows XP SP1 e anteriores

Origem da imagem: http://www.winsvr.org/info/info.php?sessid=&infoid=37

HTTP Stack – http.sys



- HTTP Listener no nível do sistema operacional (kernel)
- Windows trata: requisições e encaminhamento, cache, conexão segura, etc.
- Primeira versão: Windows XP SP2 e Windows Server 2003
- Versão 2.0 a partir do Windows Vista e Windows Server 2008
- Outras aplicações (.exe) podem registrar-se para receber requisições

Servidor HTTP usando TMS Sparkle

- TMS Sparkle – framework HTTP
- Premissa: usar o que há de mais moderno e pronto no sistema operacional
- Servidor:
  - usa http.sys (Windows trata boa parte do processamento, deixando mais seguro e mais rápido)
- Cliente: multiplataforma (Windows, Mac, Android, iOS)
  - Também usa API nativa em cada plataforma
  - Usa configurações do sistema operacional
  - Velocidade e performance
  - Conexão segura nativa (sem uso de bibliotecas externas como OpenSSL)

Servidor HTTP Exemplo

Demonstração:

Construindo um servidor HTTP “Hello World” usando TMS Sparkle

Distribuição do servidor na nuvem

Como distribuir o servidor na internet?

Virtual Private Server (Servidor Virtual Privado)

- É um servidor virtual em ambiente compartilhado que possui acesso root (administrador) e seu próprio sistema operacional independente, de modo que você pode configurá-lo livremente e instalar qualquer software que rode no sistema operacional em questão
- O VPS é uma solução muito mais barata para pessoas ou empresas com necessidades específicas que não podem ser atendidas por uma hospedagem comum (Wikipedia)
- Útil para instalar aplicações/servidores escritos em Delphi para rodar na nuvem, com acesso constante à internet, IP fixo e disponibilidade permanente, a um baixo custo

Soluções de VPS Hosting/Cloud Hosting

- Amazon EC2 (http://aws.amazon.com/ec2)
- Windows Azure (http://www.windowsazure.com)
- Google Compute Engine (http://cloud.google.com/compute)
- Digital Ocean (http://www.digitalocean.com)
- Slicehost/Rackspace (http://www.rackspace.com)
- GoGrid (http://www.gogrid.com)
- Vários outros

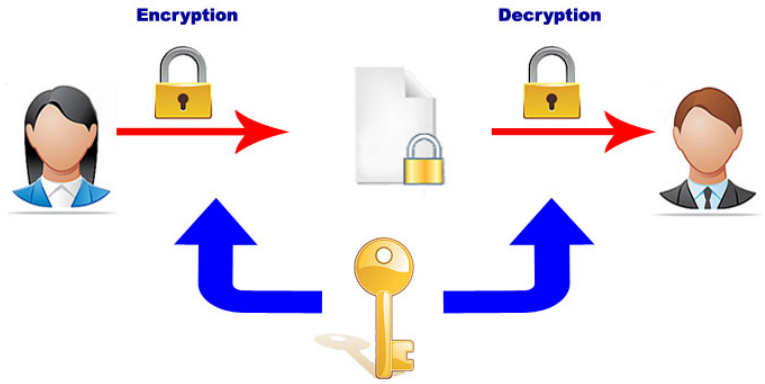
Instalando Certificado Digital no Servidor

Demonstração:

Distribuindo o servidor HTTP/HTTPS Delphi usando Amazon EC2

Como melhorar a segurança da comunicação usando HTTPS

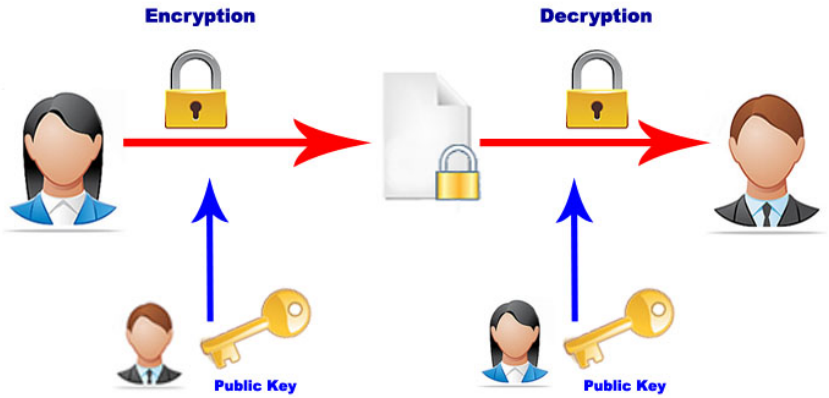
Chave simétrica



- Algoritmos rápidos
- Teoricamente a mensagem é enviada criptografada e não pode ser descoberta
- Problema: Chave precisa ser compartilhada (ambos deve conhecê-la).
- Ao enviar a chave de uma ponta a outra (sem criptografia), a chave pode ser descoberta e toda a comunicação fica comprometida

Origem da imagem: http://en.kryptotel.net/encryption.html

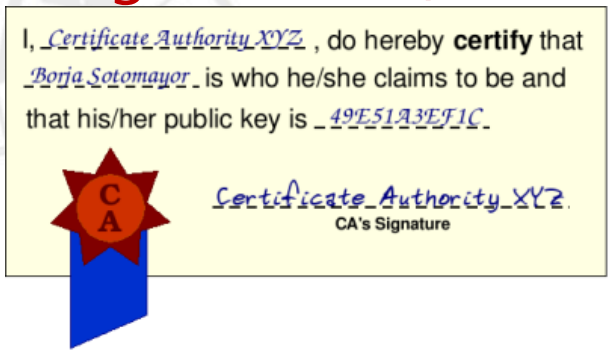
Criptografia de Chave Pública



- Chaves assimétricas: criptografa com uma chave e descriptografa com a outra
- Uma ponta mantém sua chave privada em segredo e distribui a chave pública livremente
- Não é computacionalmente possível descobrir uma chave a partir da outra
- Problema: Como garantir que a chave pública recebida veio da pessoa certa? Um ataque Man-In-the-Middle pode trocar a chave pública

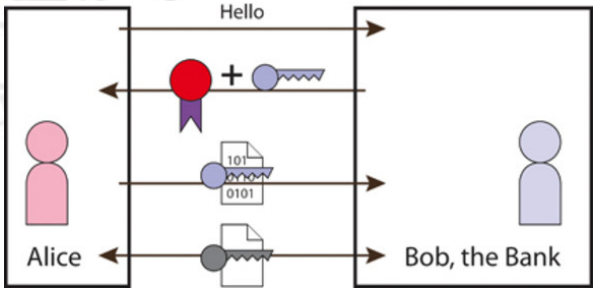
Origem da imagem: http://en.kryptotel.net/encryption.html

Certificado Digital e CA (Certification Authorities)



- Certificado contém a chave pública e a identificação de quem a possui.
- É assinado (criptografado) com a chave privada de um CA
- O certificado pode ser descriptografado com a chave pública do CA (garantindo que foi o CA quem o emitiu)
- Problema: Como garantir que a chave pública do CA não é também falsa?
- Resposta: Trusted Root CA – Sistemas operacionais e browsers já vem com os certificados dos Root CA embutidos

TLS/SSL – Transport Layer Security/Secure Sockets Layer



1. Cliente solicita comunicação segura com servidor
2. Servidor envia certificado (contendo a chave pública e a identificação do servidor)
3. Cliente (browser) verifica se certificado corresponde ao site sendo acessado, não está expirado, etc.
4. Cliente cria uma chave simétrica, temporária, para uso na comunicação, e envia ao servidor, criptografada usando a chave pública
5. Servidor descriptografa a chave simétrica com sua chave privada e inicia comunicação usando a chave simétrica

TLS/SSL – Benefícios

- Confidencialidade
  - Os dados trafegados não podem ser monitorados/lidos por terceiros
- Integridade
  - A informação recebida é exatamente a mesma que foi enviada – não houve alterações
- Autenticidade
  - Pode-se confiar que quem enviou a informação é realmente quem diz ser.
- Confiança do cliente

Empresas que emitem certificado digital

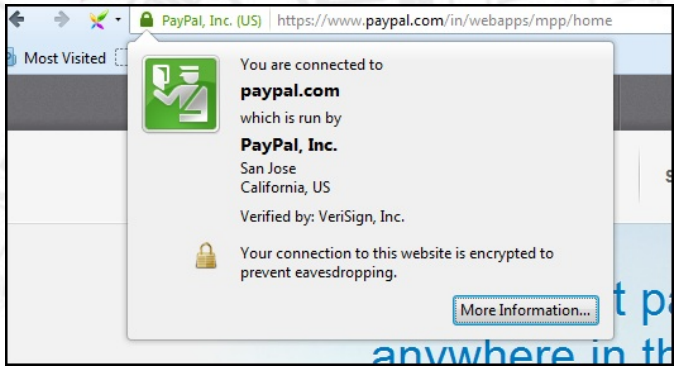
- Comodo (www.comodo.com)
- Digicert (www.digicert.com)
- Entrust (www.entrust.com)
- GeoTrust (www.geotrust.com)
- GlobalSign (www.globalsign.com)
- GoDaddy (www.godaddy.com)
- Network Solutions (www.networksolutions.com)
- SSL (www.ssl.com)
- StartCom (www.startssl.com)
- Symantec – antiga Verisign (www.verisign.com)
- Thawte (www.thawte.com)
- Trustwave (www.trustwave.com)

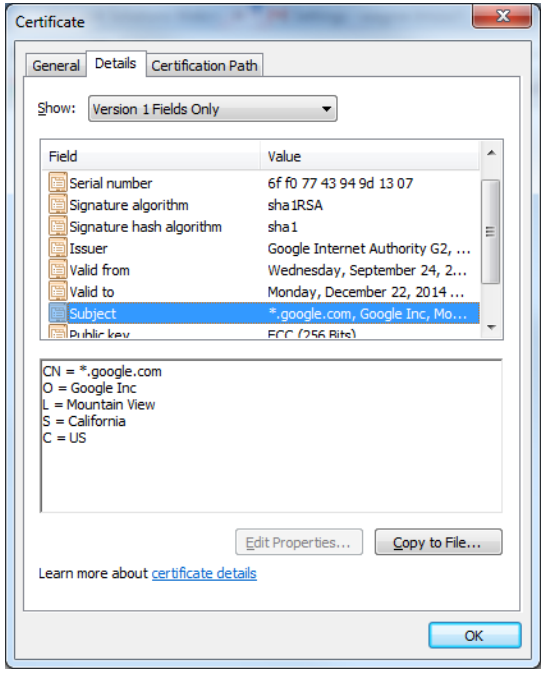
Tipos de certificado digital

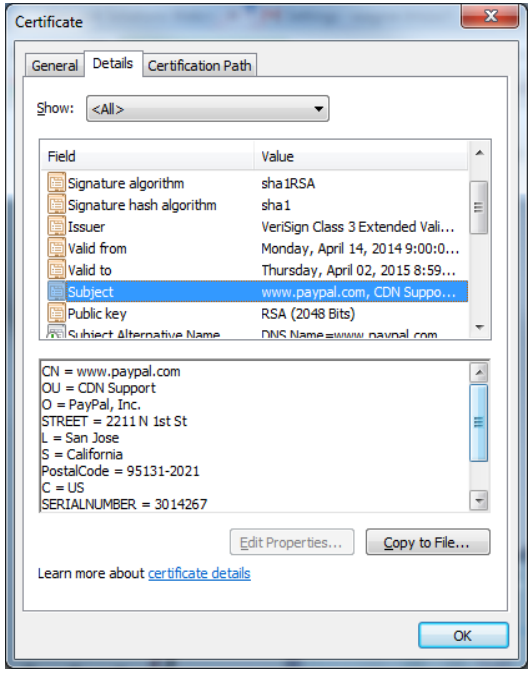
- Preço: Existem versões gratuitas. As versões pagas podem variar de R$ 80 a R$ 3.000 / ano.

- Diferenças principais:
  - Domínios suportados
    - Single-domain: www.meusite.com
    - Multiple-domains: app.meusite.com, www.meusite.com, dev.meusite.com
    - Wildcard certificates: *.meusite.com
  - Assurance level (que informação o certificado garante)
    - Domain-only: confirma apenas que o certificado é do domínio “meusite.com”
    - Organization: confirma o domínio (meusite.com) e a empresa (Meu Site Ltda.)
    - Extended Validation: processo complexo de identificação, garante o domínio, a empresa, endereço, cidade, entre outros. “Barra verde” no browser
  - “Nome” da empresa CA

Tipos de certificado digital







Obtendo um certificado digital

- Crie um Certificate Signing Request – CSR
- Contém o domínio, nome da empresa, país, chave pública
- É gerado do lado do cliente – a chave privada fica restrita ao cliente, somente a chave pública é enviada à CA
- Existem diversas ferramentas para isso: openssl (multiplataforma), certreq.exe (Windows) e diversas ferramentas disponibilizadas pelas CA.
- Enviar o CSR à CA, aguardar e-mail de confirmação deles ao domínio desejado (exemplo: se você está solicitando um certificado para meusite.com, um e-mail será enviado ao "dono" do domínio no whois – geralmente webmaster@meusite.com).

Instalando Certificado Digital no Servidor

Demonstração:

Instalando o Certificado Digital para uso em um servidor HTTPS

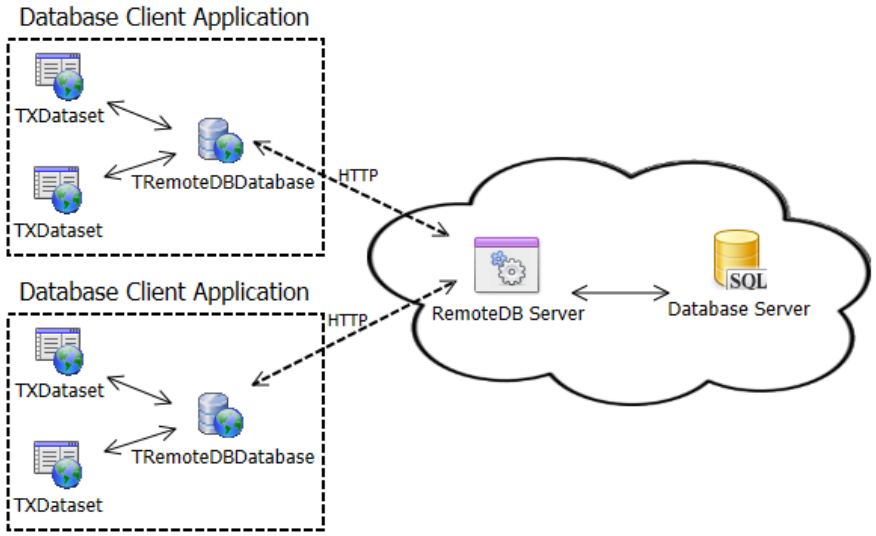
Instalando o servidor como serviço

Demonstração:
Instalando o servidor como serviço
(log-off do servidor)

Servidores mais complexos

TMS RemoteDB

- Acesso a banco de dados na internet via HTTP
- Sem necessidade de client de banco no cliente
- Sem configuração específica de firewall (usa porta 80)
- Client multiplataforma: Windows, Android, iOS, Mac
- Alta performance, resiliente à instabilidade de conexão
- Multi-componente no servidor: FireDac, ADO, dbExpress, ElevateDB, etc.



Servidores mais complexos

TMS XData

- Servidor de aplicação REST/JSON
- Utiliza TMS Aurelius – mapeamento é reutilizado
- Métodos REST equivalente ao CRUD disponíveis sem linha de código
- Métodos para regra de negócio baseado em interfaces
- Integração com consultas orientadas a objetos do Aurelius
- Atualização parcial de objetos
- Bancos de dados suportados: Firebird, MySQL, MS SQL Server, Interbase, Oracle, PostgreSQL, SQLite, NexusDB, ElevateDB, DB2, Absolute DB
- Componentes suportados: FireDac, dbExpress, ADO, UniDac, SQL-Direct, IBX, NexusDB, ElevateDB, IBObjects, FIBPlus, UIB, Direct Oracle Access

TMS XData - Exemplos

Classe:

```
[Entity, Automapping]
TCustomer = class
strict private
  FId: integer;
  FName: string;
  FTitle: string;
  FBirthday: TDateTime;
  FCountry: TCountry;
public
  property Id: Integer read FId write FId;
  property Name: string read FName write FName;
  property Title: string read FTitle write FTitle;
  property Birthday: TDateTime read FDateTime write FDateTime;
  property Country: TCountry read FCountry write FCountry;
end;
```

TMS XData - Exemplos

Requisição:

```
GET /tms/xdata/Customer(3) HTTP/1.1
```

Resposta Json:

```
{
  "$id": 1,
  "@xdata.type": "XData.Default.Customer",
  "Id": 3,
  "Name": "Maria Anders",
  "Title": "Sales Representative",
  "Birthday": "1980-05-20",
  "Country": null
}
```

Alteração parcial de dados:

```
PATCH /tms/xdata/Customer(1) HTTP/1.1

{
  "Title": "Marketing Manager"
}
```

Consultas:

```
GET /tms/xdata/Customer?$filter=Country/Name eq 'USA'&$orderby=Name&$top=10 HTTP/1.1
```

TMS XData - Exemplos

Regras de negócio via métodos:

```
[ServiceContract]
IMyService = interface(IInvokable)
['{F0BADD7E-D4AE-4521-8869-8E1860B0A4A0}']
  function FindOverduePayments(CustomerId: integer): TList<TPayment>;
end;
```

Implementação do método (servidor):

```
function TMyService.FindOverduePayments(CustomerId: integer): TList<TPayment>;
begin
  Result := TXDataOperationContext.Current.GetManager.Find<TPayment>
    .CreateAlias('Customer', 'c')
    .Where(TLinq.Eq('c.Id', CustomerId) and TLinq.LowerThan('DueDate', Now))
    .List;
end;
```

TMS XData - Exemplos

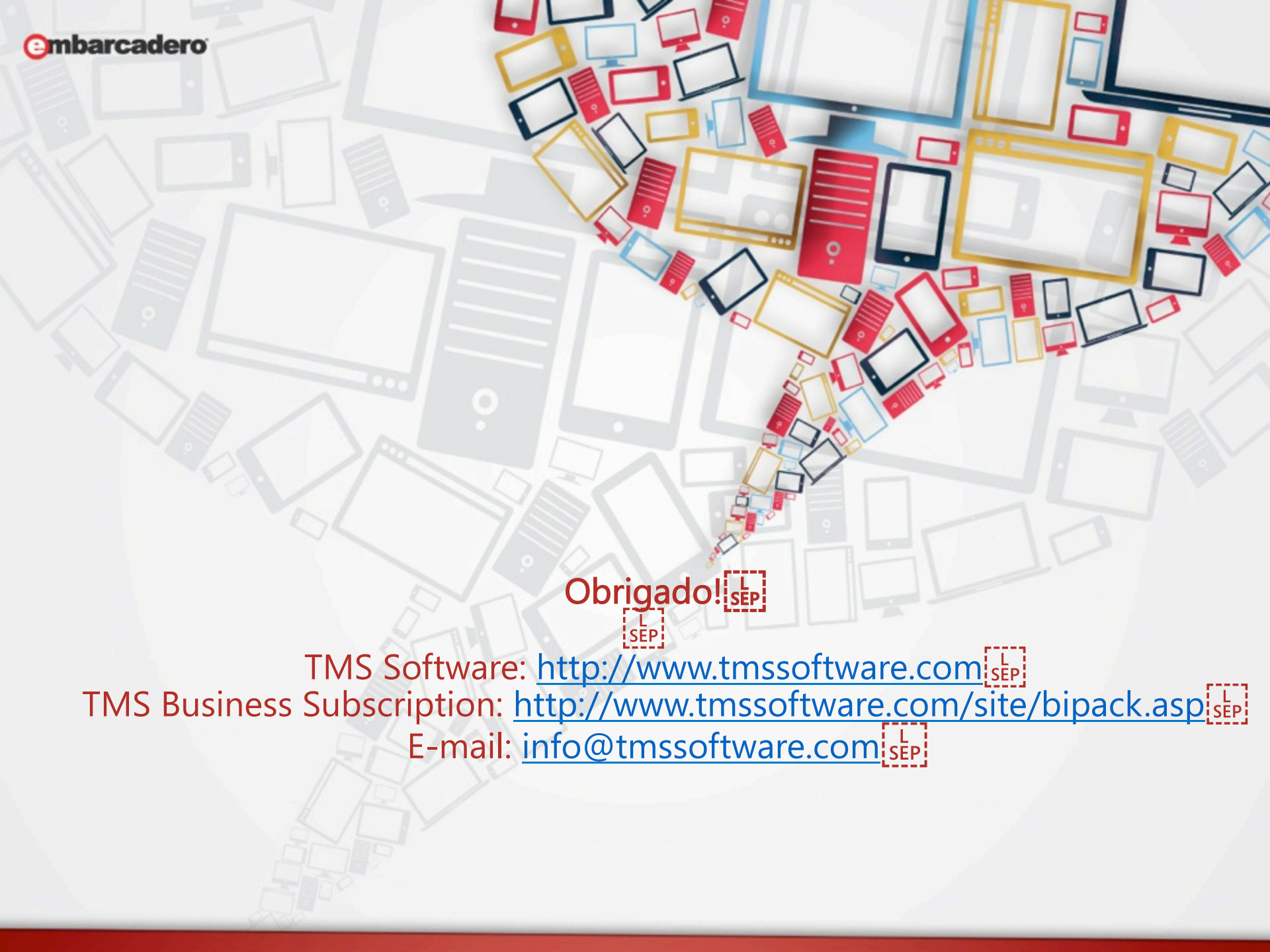
Chamada do método (cliente Delphi):

```
var
  Client: TXDataClient;
  MyService: IMyService;
 Payments: TList<TPayment>;
begin
  Client := TXDataClient.Create;
  Client.Uri := 'https://app.tmssoftware.com.br/tms/xdata';
  MyService := Client.Service<IMyService>;
  Payments := MyService.FindOverduePayments(5142);
  // process payments
end;
```

Chamada do método (cliente HTTP):

```
POST /tms/xdata/MyService/FindOverduePayments HTTP/1.1

{
  "CustomerId": 5142
}
```



Obrigado!

TMS Software: http://www.tmssoftware.com

TMS Business Subscription: http://www.tmssoftware.com/site/bipack.asp

E-mail: info@tmssoftware.com