COMUNICAÇÃO SMTP EM APLICAÇÕES GENEXUS – UM COMPARATIVO ENTRE A LINGUAGEM NATIVA E A NÃO NATIVA

https://doi.org/10.5281/zenodo.15571842

NICOLAU, Vandré Felipe de Oliveira, Especialista*
* Faculdade de Tecnologia de Praia Grande
CEETEPS - Centro Estadual de Educação Tecnológica Paula Souza
Pça. 19 de Janeiro, 144, Boqueirão, Praia Grande / SP, CEP: 11700-100
Fone (13) 3591-1303
vandrefelipe@gmail.com

OLIVEIRA, Douglas Hamilton, Mestre**
** Faculdade de Tecnologia de Americana
Rua Emílio de Menezes, s/nº, Vila Amorim, Americana / SP, CEP: 13469-111
Fone (19) 3406-3297 / 3406-5776
profdouglasoliveira@hotmail.com

RESUMO

O artigo visa demonstrar a comunicação de aplicações GENEXUS com servidores de *e-mails* utilizando o protocolo SMPT que é responsável pelo envio de mensagens de correio eletrônico, para isso serão comentados duas formas de envio: a forma nativa do GENEXUS com a utilização de variáveis que, na verdade, são objetos instanciados de classes especificas do GENEXUS que tratam do protocolo SMTP e a outra forma usando linguagem externa não nativa que, no caso, será a linguagem da Microsoft C#.NET no framework 3.5. Para tanto, o artigo desenvolve um estudo de caso onde uma solução Genexus com recursos de envio de mensagens aplicado em um ERP acadêmico sofreu uma atualização sendo necessária a introdução de um *web service* que passou a gerenciar exclusivamente os processos de envio de mensagens.

PALAVRAS-CHAVE: SMTP, correio eletrônico, *e-mail*, GENEXUS.

ABSTRACT

The article aims to demonstrate shortly GENEXUS communicati on applications with mail servers using the SMTP protocol that is responsible for sending electronic mail messages that are being viewed and discussed in two forms of transmission. The native form of GENEXUS with use of variables that are actually instantiated objects of specific classes of GENEXUS dealing with the SMTP protocol and how using external language using non native languag e of the case that will be Microsoft C #. NET on Framework 3.5.

KEY-WORDS: *SMTP, e-mail, GENEXUS.*

INTRODUÇÃO

As relações comerciais do século XXI utilizam meios digitais em sua maioria e a comunicação entre as empresas com seus clientes ou mesmo com outras empresas se faz através de mensagens eletrônicas ou, ao menos, é uma tendência que se consolida cada vez mais e torna-se necessário que os desenvolvedores disponibilizem recursos de envio e recepção de mensagens em suas soluções de *software* independente do tamanho da aplicação.

Uma aplicação atual que não possui esse recurso tende a não atender as necessidades dos usuários por isso o conhecimento do protocolo SMTP é por parte dos desenvolvedores é bastante apreciada pelos gerentes de TI quando solicitam um profissional em seu departamento de Recursos Humanos.

1 ORIGEM

Até 1977, a Arpanet utilizava vários padrões informais de mensagens de texto (*e-mail*) enviadas entre computadores de seu *host* (hospedeiro). Foi constatada a necessidade de codificar essas práticas e oferecer um padrão. O resultado desse esforço foi o padrão *Request for Comments* (RFC) # 733, “Norma para o formato de ARPANET mensagem de texto". A especificação tentou evitar grandes alterações

nos *softwares* da época, permitindo ao mesmo tempo vários novos recursos (CROCKER, 1982).

Essa é a origem do protocolo SMTP que é o cerne para o foco central desse artigo que é o GENEXUS mandando *e-mails* com linguagem nativa e não nativa.

O objetivo do *Simple Mail Transfer Protocol (*SMTP) é a transferência de *e-mail* de modo confiável e eficiente. SMTP é independente do subsistema de transporte especial e requer apenas um canal de fluxo confiável de dados ordenados (POSTEL, 1982).[1]

2 DESENVOLVENDO APLICAÇÕES QUE SE UTILIZAM DE SMTP

A aplicação descrita no artigo possui conceitos de envio de mensagens de *e-mail* e apresenta a tecnologia SMTP descrevendo um modelo prático adotado comumente por aqueles que utilizam a plataforma Genexus para desenvolvimento de soluções.

2.1 APLICAÇÃO NATIVA GENEXUS SMTP

Genexus é uma ferramenta de desenvolvimento de sistemas que permite a construção de *software* com grande capacidade tecnológica, por meio de seus geradores de código, que produz código fonte nas linguagens Java, C# e Ruby, a partir da programação na própria linguagem Genexus (GONDA, 1988).

Esse mecanismo permite que a aplicação Genexus fique desvinculada da tecnologia nativa, podendo migrar para uma nova tecnologia emergente apenas pela execução de um novo gerador de código, eventualmente que venha ser utilizado. Ou seja, uma aplicação Genexus que seja gerada em C#, por exemplo, pode naturalmente ser reconstruída em Java, sem que isso implique qualquer mudança no sistema (GOMDA, 2003).

O que se cita, como qualquer mudança, é a reescrita do código da linguagem original A para a nova linguagem B, sendo que a principal vantagem do Genexus é que ele tem seu próprio código de comandos internos não precisando ser reescrito. O que de fato acontece é que o

[1] Ver Anexo A.

Genexus usa um compilador de uma linguagem de mercado para gerar o seu código baseado os códigos internos do genexus descritos na base de conhecimento.

A base de conhecimento é o projeto do Genexus propriamente dito e possue todos os parâmetros, descrições de entidades (Transações), regras de negócio e linhas de códigos internos e tudo isso é traduzido para o compilador.

Para produzir um pequeno sistema que envie *e-mails* no protocolo SMTP em Genexus, pode-se realizá-lo de duas maneiras, a primeira a ser apresentada a seguir, utiliza recursos da própria ferramenta para enviar *e-mails:*

A→ crie uma aplicação GENEXUS utilizando a versão X do produto que pode ser baixado no site da empresa indicado no final desse artigo.

B→ além do Genexus é necessária a instalação dos seguintes softwares para a configuração correta do ambiente de desenvolvimento: Microsoft Windows XP SP2, ou superior, Microsoft .NET Framework 3.5 ou superior, Microsoft SQL Server 2005 Express (Free) ou 2008, Microsoft Internet Explorer 6.0 SP1 ou superior.

C→ a aplicação deverá possuir apenas um objeto *webpanel* onde toda a aplicação funcionará, conforme figura 1 da lista de variáveis:

Figura 1 – Variáveis utilizadas no sistema
Fonte: Genexus X Evolution 1 versão 10.1.38483 U4

D→ veja que existem variáveis do tipo SMTP, *MailMessage* e *MailRecipient* que são as principais desta aplicação. A *interface* gráfica que será programada nesse exemplo é montada como um painel *Web (WebPanel),* e terá o formato descrito na figura 2:

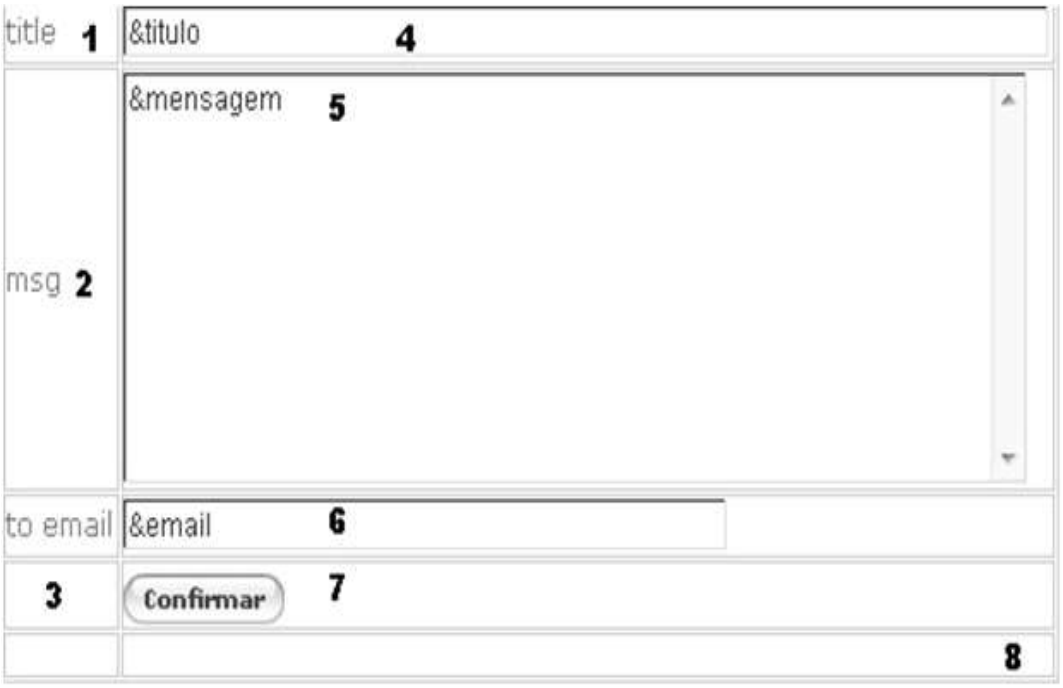

Figura 2 – Tela da Aplicação de envio de Emails Genexus Nativo
Fonte: Genexus X Evolution 1 versão 10.1.38483 U4

1, 2 e 3 → são controles do tipo *TextBlock (labels).*
4,5 e 6 → são as variáveis criadas no item C.
7→ um controle do tipo *Button.*
8→ tabela com 2 colunas e 5 linhas.

E→ o evento programado no CONFIRMAR é descrito a seguir, e na sequência as linhas são explicadas:

```
1  Event Enter
2    // servidor
3    &smtpsession.Authentication    = 1
4    &smtpsession.Host              = "smtp.gmail.com"
5    &smtpsession.Port              = 587
6    &smtpsession.UserName          = "vandrefelipe@gmail.com"
7    &smtpsession.Password          = "maiqueldiequisson"
8    &smtpsession.Sender.Address    = "vandrefelipe@gmail.com"
9    &smtpsession.Sender.Name       = "vandrefelipe@gmail.com"
10   &smtpsession.Login()
11   If &smtpsession.ErrCode<>0
12     &msg = &smtpsession.ErrDescription
13   Else
14     &mailmessage.HTMLText   = &mensagem
15     &mailmessage.Subject    =&titulo
16     // lista de emails para enviar
17     &mailrecipient.Address  = &email
18     &mailmessage.To.Add(&mailrecipient)
19     &smtpsession.Send(&mailmessage)
20     If &smtpsession.ErrCode<>0
21       &Retorno = 0
22     Else
23       &Retorno = 1
24     Endif
25   Endif
26   If &retorno = 1
27     msg('enviada com sucesso')
28   Else
29     msg('Nao enviada')
30   Endif
31 EndEvent
```

Figura 3 – Código do Programa de envio de Emails em GENEXUS

Fonte: Genexus X Evolution 1 versão 10.1.38483 U4

Linhas 3 até 9→ a variável SMTPSession (simbolizada no Genexus através do prefixo de um "&") é uma variável do tipo SMTPSession e tem a função de carregar os dados do lado servidor de *email.* Para este exemplo é utilizado uma conta do provedor Gmail (*www.gmail.com*) e as especificações necessárias são: o *host,* que no caso do *gmail,* é *smtp.gmail.com.* A porta também é determinada no *site* do *gmail* e as demais informações são referentes a usuário, senha, endereço e nome da origem do *e-mail.*

Linha 10→ o método *login* faz o efetivo acesso ao servidor de *e-mail* no caso do *gmail.*

Linhas 10 a 12→ caso algum erro aconteça é possível tratar sendo que o erro diferente de zero é algum problema obtido no *login* com o servidor de *e-mail.*

Linhas 14 e 15→ a variável *MailMessage* é responsável pela mensagem propriamente dita e nela são registrados os métodos HTMLtext contendo a mensagem propriamente dita e o método *subject* que contem o título. Ambas devem ser do tipo texto.

Linha 17→ o endereço de destino pode ser um ou vários deles separados por ponto e vírgula e por isso existe um tipo de objeto específico para tal tratamento que é o *MailRecipient* em seu método *address.*

Linha 18→ de volta à variável *MailMessage* adiciona-se o método *add* para adicionar o objeto contendo a coleção de *e-mails* a serem enviados.

Linha 19→ marca o fim do processo com o método send. Ele envia o *e-mail* ao(s) destino(s).

Linhas 20 a 30→ faz-se o tratamento de erros caso alguma coisa tenha dado errado. No final existe uma lista de status de mensagens de SMTP.

E→ O teste é simples e, de acordo com a tela seguinte, basta preencher os campos e pressionar o enviar como descrito na figura 4:

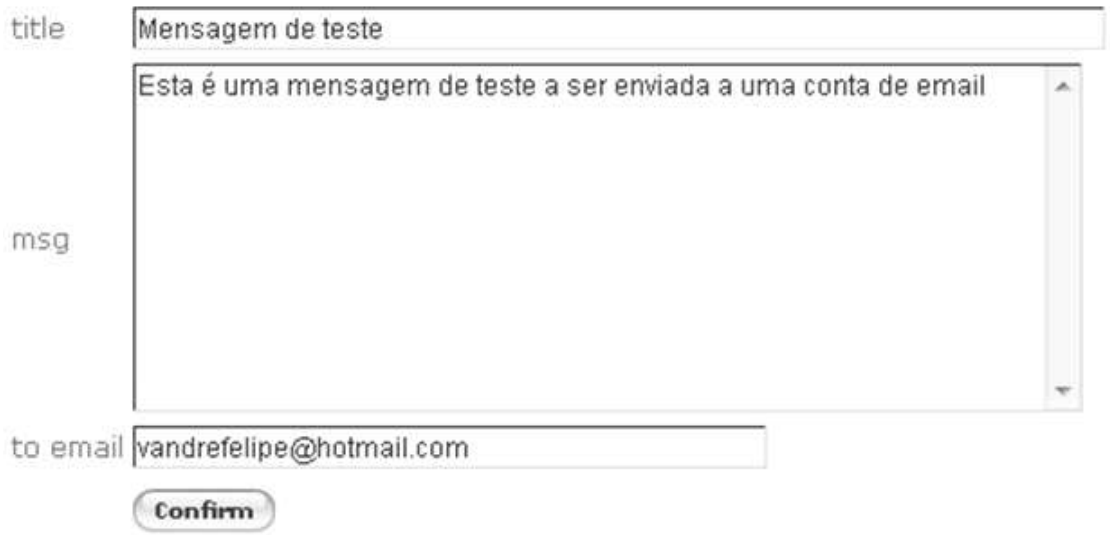

Figura 4 – Tela do Programa de envio de Emails em GENEXUS em execução

Fonte: Genexus X Evolution 1 versão 10.1.38483 U4

2.2 APLICAÇÃO NÃO NATIVA GENEXUS SMTP

Apesar de Genexus manter a postura de independência da tecnologia do compilador escolhido, ainda é possível programar em código da linguagem diretamente na ferramenta, tornando-a flexível o suficiente para executar tudo que a programação do compilador escolhido realiza, e ainda de forma automatizada nos processos repetitivos que podem ser deixados para que o próprio Genexus resolva.

Neste exemplo, o processo de envio de *e-mail* é o mesmo, ou seja, utilizar os recursos do protocolo SMTP para o envio de mensagens de correio eletrônico. A diferença está em demonstrar a utilização de linguagem do compilador que, neste caso, é utilizada a linguagem C#.NET framework 3.5:

A→ crie uma aplicação GENEXUS com um objeto *procedure* e um objeto *webpanel,* sendo que a *procedure* conterá um código nativo em C#, e o *webpanel* é semelhante ao exemplo anterior.

B→ a *procedure* se chamará sendmail_csharp e conterá as variáveis descritas na figura 5:

Figura 5 – Variáveis utilizadas no sistema
Fonte: Genexus X Evolution 1 versão 10.1.38483 U4

C→ os parâmetro de entrada e saída na aba *Rules* é a seguinte: parm(in:&EMailTo, in:&EMailSubject, in:&EMailMessage, out:&Retorno);

Os parâmetros com o *in* são os parâmetros de entrada que no caso são 2: o destino, o titulo e a mensagem e o parâmetro com *out* é o parâmetro de saída.

D→ o código é em linguagem C#, que no GENEXUS são predescedidos pela palavra *csharp*, na frente da frase e as variáveis Genexus utilizadas vem entre colchetes e pontos de exclamação. As explicações estão na figura 6:

```
1  &SMTPEmailPassword = 'maiqueldiequisson'
2  &SMTPEmailUser     = 'vandrefelipe@gmail.com'
3  &SMTPUserName      = 'vandrefelipe@gmail.com'
4  &SMTPHost          = 'smtp.gmail.com'
5  &SMTPPort          = 587
6  csharp try
7  csharp {
8  csharp System.Net.Mail.MailMessage mM = new System.Net.Mail.MailMessage();
9  csharp mM.From = new System.Net.Mail.MailAddress([!&SMTPUserName!]);
10 csharp mM.To.Add([!&EMailTo!]);
11 csharp mM.Subject = [!&EMailSubject!];
12 csharp mM.Body    = [!&EMailMessage!];
13 csharp mM.IsBodyHtml = true;
14 csharp mM.Priority = System.Net.Mail.MailPriority.High;
15 csharp System.Net.Mail.SmtpClient sC = new System.Net.Mail.SmtpClient
16 csharp string strId;
17 csharp string strPassword;
18 csharp strId       = [!&SMTPUserName!];
19 csharp strPassword = [!&SMTPEmailPassword!];
20 csharp sC.Credentials = new System.Net.NetworkCredential(strId,strPassword);
21 csharp  sC.Send(mM);
22 csharp  [!&statusEnvio!] =  1;
23 csharp }
24 csharp catch (System.Exception ex)
25 csharp {
26 csharp  [!&statusEnvio!] =  0;
27 csharp }
28 &Retorno = &statusEnvio
```

Figura 6 – Código em CSHARP inserido no programa GENEXUS

Fonte: Genexus X Evolution 1 versão 10.1.38483 U4.

Linhas 1 até 5 preenchem dados referentes a conta servidora, a conta de *e-mail* que enviará a mensagem, semelhante ao exemplo apresentado anteriormente na linguagem nativa.

Linha 8 Instancia um objeto da classe MailMessage chamado mm.

Linha 9 preenche a origem da mensagem.

Linha 10 adiciona a lista de emails destinatários

Linha 12 preenche a mensagem.

Linha 13 determina se a mensagem é no formato HTML.

Linha 14 determina o grau de prioridade de envio da mensagem.

Linha 15 Instancia um cliente de SMTP para envio da mensagem.

Linhas 16 e 17 cria variáveis de texto para auxiliar no envio.

Linhas 18 e 20 preenchem os dados de usuário e senha e instancia o objeto com essas informações.

Linha 21 envia a mensagem.

Linhas 21 a 21 Tratam erros.

E→ para chamar essa procedure basta criar um *webpanel* contendo as seguintes variáveis:

Figura 7 – Variáveis utilizadas pelo programa com código em CSHARP inserido no programa GENEXUS
Fonte: Genexus X Evolution 1 versão 10.1.38483 U4.

F→ o *webpanel* conterá a tela descrita na figura 7:

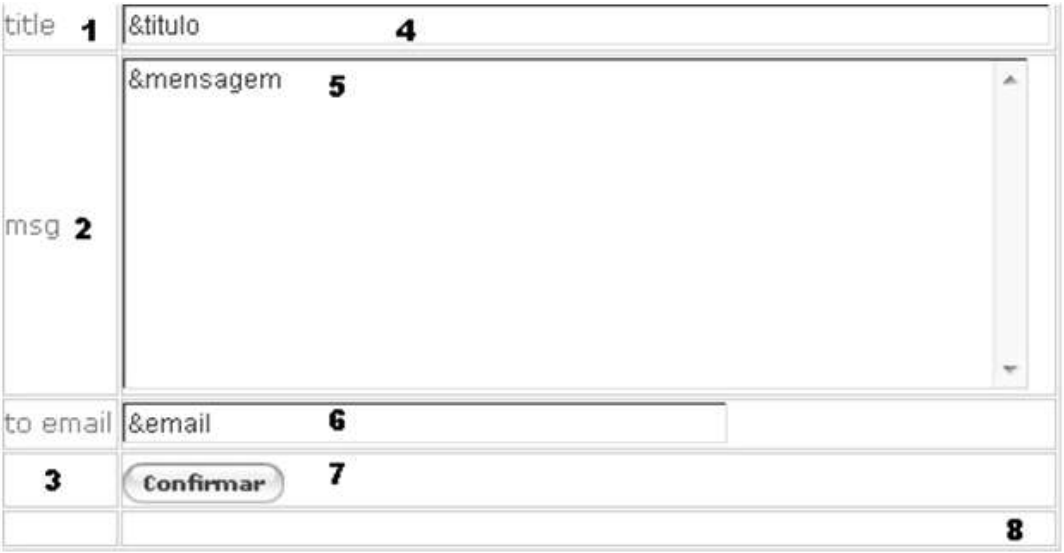

Figura 8 – Tela do programa com código em CSHARP inserido no programa GENEXUS
Fonte: Genexus X Evolution 1 versão 10.1.38483 U4.

1, 2 e 3 → são controles do tipo text box (labels).
4,5 e 6 → são as variáveis criadas no item anterior.
7→ um controle to tipo Button.
8→ tabela com 2 colunas e 5 linhas.
G→ o código a seguir executa uma procedure passando parâmetros e recebendo um retorno:

```
Event Enter
        &retorno = sendmail_csharp.
Udp(&email,&titulo,&mensagem)
        if &retorno      = 1
                msg('mensagem enviada')
        else
                msg('mensagem nao enviada')
        endif

EndEvent
```

3 COMPARAÇÕES

O ambiente de desenvolvimento Genexus possui algumas peculiaridades que o destacam com relação a outras ferramentas de desenvolvimento. A sua principal característica é não ser exatamente uma linguagem de programação e sim uma ferramenta de protótipos incrementais que permite que o desenvolvimento fique focado na plena análise de sistemas e levantamento de requisitos e a parte da programação propriamente dita e geração de código seja feita pela ferramenta. Por esta razão, o exemplo SMTP descrito, como forma nativa de fato, utiliza classes internas do GENEXUS para manipulação de objetos SMTP, mas, quando o projeto sai do protótipo e vira código fonte, esse processo utiliza um compilador previamente escolhido pelo desenvolvedor, podendo ser Java, .NET, COBOL entre outros; logo, o GENEXUS gera, na prática, código em alguma linguagem de programação de mercado. Vide no item problemas de compatibilidade entre as versões do Genexus com relação às classes para SMTP.

Posto isso, é importante observar que mesmo assim ele permite

que código não nativo seja utilizado como no caso visto com linguagem csharp.NET. Nesse caso, o prefixo CSHARP é colocado em todas as linhas e com isso o GENEXUS "sabe" que, neste trecho, o compilador da linguagem em questão será acionado.

O GENEXUS possui esse recurso pela seguinte razão: caso algum recurso existente em alguma linguagem não possua equivalente em GENEXUS é possível incorporar esse trecho de código em num projeto genexus e consumir esse recurso sem prejuízo para a aplicação. A vantagem fica por conta do GENEXUS poder se adaptar e utilizar recursos externos, e o contra é que caso alguma modificação tenha que ser feita nesse recurso esse processo necessita ser feito de forma manual e não automatizada pelo GENEXUS como ocorre com seus objetos nativos.

Portanto, quando algum recurso está disponível no GENEXUS utilize-o mesmo possuindo uma versão pronta dele em alguma outra linguagem. Usando o GENEXUS, garante-se a automação do processo de geração de código por conta da ferramenta.

4 PROBLEMAS DE COMPATIBILIDADE

Ao longo dos *releases* das últimas versões do Genexus para plataforma Windows Vista e Windows Server, para 32 bits e 64 bits, observam-se alguns problemas de compatibilidade no processo de gerenciamento do envio das mensagens, sendo que, em algumas oportunidades, a necessidade de se colocar soluções Genexus no ar forçou desenvolvedores a fazer aplicações híbridas com relação a plataformas de sistema operacional, versão de ferramentas Genexus e utilização de *webservices.*

Conforme um caso real enfrentado pela equipe de desenvolvimento responsável pelo EPR acadêmico do Centro Paula Souza, que vem sendo paulatinamente implantada nas FATECs (Faculdade de Tecnologia do Estado de São Paulo), desde 2009. É apresentado o problema e a solução adotada que funciona bem até hoje. Para entender o problema, é necessário conhecer alguns detalhes técnicos da aplicação no momento de sua implantação no segundo semestre de 2009 até a sua atualização, quando se notou o problema em maio de 2010.

4.1 CENÁRIO DA IMPLANTAÇÃO DO SISTEMA EM JULHO DE 2009

Conforme descrito no quadro 1, o cenário em julho de 2009 era o seguinte:

Tipo de Aplicação	Web (http://www.projetocps.pro.br/login.aspx)
Servidor	Windows Server 2003 – 32 Bits
Versão Genexus	Genexus X Release U2 – 64 Bits[2]
Arquitetura e Sitema Operacional das máquinas dos desenvolvedores	HP HDX18 - Intel centrino Quad 64 Bits-Windows Vista – 64 Bits - 4GB RAM
Gerador	.NET framework 3.5
Banco de Dados	Microsoft SQL-Server 2008 Enterprise
Uso de Web Services	Não

Quadro 1 – Cenário da Implantação do Sistema em julho de 2009

Neste cenário a aplicação continha código nativo de Genexus como descrito no item 2.2, e as funcionalidades que necessitavam de envio de *email,* eram feitas através de procedimentos semelhantes aos descritos como, por exemplo, o envio de mensagens para alunos e professores das unidades de ensino, envio de chave aos professores para o lançamento de notas, mensagens de confirmação de criação de conta de usuário entre outras funções.

[2] A equipe de desenvolvimento trabalha decentralizada em locais geograficamente distantes e se utilizam do Genexus Server para atualizar o projeto genexus no servidor supracitado.

4.2 CENÁRIO DA ATUALIZAÇÃO DO SISTEMA EM MAIO DE 2010 E O PROBLEMA GERADO

Em maio de 2010, o cenário era (quadro 2):

Tipo de Aplicação	Web (http://www.projetocps.pro.br/login.aspx)
Servidor	Windows Server 2003 – 32 Bits
Versão Genexus	Genexus X Release U2 – 64 Bits Windows 7 [3]
Arquitetura e Sitema Operacional das máquinas dos desenvolvedores	HP HDX18 - Intel centrino Quad 64 Bits- Windows 7 – 64 Bits - 4GB RAM
Gerador	.NET framework 3.5
Banco de Dados	Microsoft SQL-Server 2008 Enterprise
Uso de Web Services	Não

Quadro 2 – Cenário da Atualização do Sistema em maio de 2010

Foram feitos vários testes na aplicação e verificando-se que estava tudo correto, recorreu-se ao suporte da própria distribuidora do Genexus (Artech), que solicitou que se fizessem testes gerando máquinas virtuais com outras versões de Genexus e com outras versões de sistema operacionais não 64 bits como o próprio Win 7 32 bits, o Win Vista 32 bits e o Win XP para 32 bits também.

No final, foi visto que a aplicação funcionava perfeitamente sendo desenvolvida no Windows 7 - 64 Bits mas as funcionalidades de envio de mensagens não e não tendo como voltar o sistema para a versão anterior foi criado um WebService que disponibiliza funções de envio de email que seriam consumidas pelo sistema todo.

[3] A equipe de desenvolvimento trabalha decentralizada em locais geograficamente distantes e se utilizam do Genexus Server para atualizar o projeto genexus no servidor supracitado mas agora com a versão 64 do Windows 7. O problema apresentado depois da atualização do sistema de Windows Vista 64 Bits para Windows 7 64 Bits foi que os emails passaram a não mais conseguir chegar em seu destinatário mas sem mensagem de retorno. Simplesmente a mensagem não chegava.

4.3 CENÁRIO DA ADAPTAÇÃO DO SISTEMA EM JUNHO DE 2010 PARA SOLUCIONAR O PROBLEMA

Por fim, em junho de 2010, o cenário era (quadro 2):

Tipo de Aplicação	Web (http://www.projetocps.pro.br/login.aspx)
Servidor	Windows Server 2003 – 32 Bits
Versão Genexus	Genexus X Release U2 – 64 Bits Windows 7
Arquitetura e Sitema Operacional das máquinas dos desenvolvedores	HP HDX18 - Intel centrino Quad 64 Bits-Windows 7 – 64 Bits - 4GB RAM
Gerador	.NET framework 3.5
Banco de Dados	Microsoft SQL-Server 2008 Enterprise
Uso de Web Services	Sim – Foi gerada uma aplicação para executar as funções de envio de email em uma aplicação para ser executada no próprio servidor em versão 32 bits valendo-se de código csharp sendo que a aplicação principal consome esse WebService para funções de envio de email

Quadro 3 – Cenário da Atualização do Sistema em maio de 2010

A solução foi desenvolver uma solução a parte contendo todas as funcionalidades de envio de mensagens em smtp e a sequência (que funciona até hoje) é a seguinte:

a) a aplicação principal quando se precisa enviar uma mensagem acessa essas funcionalidades através de um recurso chamado *External Object* que é a forma com a qual o Genexus acessa recursos de um *web service;*
b) o *External Object* está vinculado a uma outra aplicação que roda num versão de 32 bits (que não apresenta erros) e nessa aplicação estão disponíveis as funções de envio de *e-mail*;
c) os parâmetros de entrada são a mensagem, o assunto, o destinatário sendo que o *e-mail* de origem é fornecido de forma encapsulada pela própria aplicação que envia o *e-mail;*

d) essa aplicação recebe esses dados e processa o *e-mail* retornando um 1 como *true* (foi enviado) ou 0 *false* (não foi enviado);
e) ambas as aplicações rodam no servidor sendo um com máquina virtual para 32 *bits*, ou seja, a que processa os *e-mails.*

5 CONCLUSÃO

No exemplo descrito na pesquisa, foi utilizado um servidor de *e-mail* comercial gratuito, o SMTP do GMAIL, possibilitando o envio de *e-mails* por meio de aplicações externas ao ambiente. Ressalta-se a existência de condições e limitações que o servidor impõe nas questões de segurança e quantidade de *e-mails* a serem enviados. Um servidor próprio de SMTP poderia melhorar certos aspectos, mas requereria muito esforço adicional para o tratamento de SPAMS, Virus e outras pragas existentes na rede.

Este artigo descreveu a maneira de utilizar o protocolo SMTP de servidores existentes na *Web* com Genexus, porém, para futuros trabalhos, poderia ser interessante a programação de um servidor próprio do protocolo SMTP. Em caso de problemas de compatibilidade, as vezes é necessário usar de recursos como, por exemplo, o uso de *Web Services,* como desmonstrado na pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

CROCKER, D. ***Standard for the format of ARPA internet Text messages***. RFC-822, august, 1982.

GENEXUS DOWNLOAD TRIAL VERSION. *Disponível em: <http://www.genexus. com/portal/hgxpp001.aspx?2,61,1055,O,E>. Acesso em 15/05/2010.*

GENEXUS DOWNLOAD TRIAL VERSION. *Disponível em: <http://www.example-code. com/csharp/smtpLastStatus.asp>. Acesso em: 15/05/2010.*

GONDA, B.; Jodal, N. ***Genexus philosophy.*** *Disponível em: <http:// www.genexus. com/portal/agxppdwn.aspx?2,59,1080,O,E,0,22791%3 bE%3b1%3b2315,, 1988>. Acesso em: 23/03/2010.*

GONDA, B. ***Process-oriented or Data-oriented Development***. *Thoughs on the 40th Anniversary of Database Management Systems. Disponível em: <http://www.genexus. com/portal/agxppdwn.aspx?2,59,1080,O,E ,0,22793%3bE%3b1%3b2315,, 2003>. Acesso em: 13/11/2009.*

POSTEL, J. ***Simple mail transfer protocol***. RFC-821, august, 1982.

ANEXO A

```
// Some common SMTP status codes are as follows:
// 211 - System status message.
// 214 - Help message formatted for human reader follows.
// 220 - SMTP service ready.
// 221 - Service/connection closing.
// 250 - Successful request. Action completed.
// 251 - Recipient is not local to the server, but the server
will accept and forward the message.
// 252 - Recipient cant be verified, but the server will accept
the message and attempt delivery.
// 354 - Start message input now, end with .. Indicates the
server is ready to accept a message once youve given it From:
and To: information
// 421 - Service is not available and connection will be
closed.
// 450 - Requested command failed because the recipients
mailbox is unavailable.
// 451 - Command has been aborted due to a server error.
Possibly notify your SysAdmin.
// 452 - Command has been aborted because the server has
insufficient system storage.
// 500 - Server could not recognize the command was due to a
syntax error. (usually due to mail client error)
// 501 - Syntax error was found in command arguments. (usually
due to mail client error)
// 502 - Command was not implemented. (usually due to mail
client error)
// 503 - Server has encounterd a bad command or sequence of
commands. (usually due to mail client error)
// 504 - Command parameter is not implemented. (usually due
to mail client error)
// 550 - Command failed because the users mailbox was
unavailable (or you did not have permissions to send to this
mailbox)
// 551 - Recipient is not local to the server. Server responds
with a fowarding address that should be tried.
// 552 - Action was aborted because storage allocation was
exceeded.
// 553 - Action was aborted because the mailbox name was
invalid.
// 554 - Transaction failed, without a clear reason
```