CENTRO UNIVERSITÁRIO FEEVALE

INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Plataforma RAD para aplicações locais ou distribuídas

por

GUSTAVO HENRIQUE CERVI
gustavo@overstep.com.br

Anteprojeto de Estágio Supervisionado

Profª. Ms. Marta Rosecler Bez el Boukhari
martabez@feevale.br

Novo Hamburgo, abril de 2004

Sumário

1. Dados de identificação.....3
2. Resumo.....4
3. Motivação.....5
4. Objetivos.....8
5. Metodologia.....9
6. Cronograma.....10
7. Referências Bibliográficas.....11

1. Dados de Identificação

Área de Estudo: Desenvolvimento, framework
Título provisório do trabalho: Plataforma para aplicações distribuídas
Orientador(a): Profª. Ms. Marta Rosecler Bez el Boukhari
Identificação do aluno:

Nome: Gustavo Henrique Cervi

Telefones:

Celular: 9122.7232
Residencial: 595.2372
Comercial: 582.6514

E-mail: gustavo@overstep.com.br

2. Resumo

Atualmente existe uma forte demanda por sistemas distribuídos, o modelo de desenvolvimento WEB é utilizado em larga escala para aplicações, mesmo sendo idealizado para documentos ancorados de forma estática. O paradigma formado em volta desta rede já é amplamente trabalhado, mas possui algumas deficiências.

Em face ao grande avanço das aplicações distribuídas, a utilização do *browser*[1] para interfaceamento com o usuário é muito comum, porém muito limitada quando se entra no território de sistemas integrados ou sistemas que requerem um nível de interação muito maior com o usuário. Alguns conceitos como sistemas *Three Tier*[2] e *Middlewares*[3] resolvem parte dos problemas de alta complexidade, mas complicam nas aplicações mais simples.

Um *framework*[4] para aplicações de média complexidade, distribuídas, multi-plataforma e multi-usuário pode ser desenvolvido utilizando os recursos das linguagens interpretadas aliadas aos conjuntos de bibliotecas gráficas disponíveis para elas. As linguagens interpretadas possuem características importantes como a facilidade de dinamizar o código fonte em tempo de execução. Esta dinâmica pode ser feita via rede, e isto é a base da solução proposta.

Sendo assim, este projeto tem como principal objetivo, propor uma idéia de como disponibilizar ao público, uma plataforma de desenvolvimento de aplicações utilizando recursos importantes encontrados nas linguagens interpretadas e suas bibliotecas.

1 Navegador de internet, software utilizado para visualização e interação de sites

2 Três camadas

3 Camada intermediária entre servidor e cliente

4 Plataforma ou suporte para desenvolvimento e execução de aplicações

3. Motivação

As linguagens interpretadas são, normalmente, consideradas como ferramentas para soluções provisórias ou protótipos de sistemas. Raramente são consideradas como recurso definitivo para desenvolvimento. O motivo para isso, é o fato de não gerarem código binário nativo do sistema operacional ou arquitetura da máquina. Este fato é muito importante e "pesa" muito na hora de se optar por uma linguagem e desenvolver uma aplicação (LUTZ, 2001).

Por outro lado, estas linguagens possuem, na maioria dos casos, um vasto conjunto de bibliotecas prontas e uma flexibilidade muito grande na hora de implementar o código. Estas características auxiliam muito o desenvolvimento do software. Como exemplo de linguagem interpretada, Python[5] é considerada por alguns autores como uma linguagem RAD[6] (LUTZ, 2001; MARTELLI, 2003; CATUNDA, 2001).

Na API[7] da linguagem Python, existe um conjunto de bibliotecas para interfaceamento com o usuário chamado Tkinter[8]. Este conjunto possibilita o desenvolvimento rápido de aplicações em ambiente gráfico e é extremamente maduro, herdando as características e trechos de código da linguagem Tcl/Tk[9].

A característica das linguagens interpretadas que mais interessa neste contexto, é a dinâmica dos procedimentos em tempo de execução. Observando a linguagem Python, uma vez

5 Linguagem interpretada desenvolvida por Guido Van Rossum em 1990 (LUTZ, 2001. p. 3)

6 *Rapid Application Development*: conceito dado às linguagens que permitem o desenvolvimento rápido

7 *Application Programm Interface*: Conjunto de rotinas para desenvolvimento de aplicações

8 Biblioteca gráfica para desenvolvimento de interfaces com o usuário.

9 *Tool Command Language*: Linguagem interpretada com sintaxe semelhante a linha de comando.

que o programa iniciou, ele pode sofrer alterações em seu código, ou ainda instanciar objetos que não existiam no código original, no momento da execução. Este processo ocorre porque o interpretador pode incluir partes de código no momento em que for necessário.

O suporte à rede, fornecido pela linguagem Python, é suficientemente desenvolvido para ser utilizada em um software que trafegue objetos e dados pela rede de maneira eficiente, em conjunto com isso, recursos de multi-programação fazem parte do núcleo da linguagem, tornando o desenvolvimento de um servidor de aplicação muito simples e rápido de se implementar.

A proposta deste trabalho surge quando é observado que um programa pode gerar outro programa (código interpretado) que pode ser enviado (suporte à rede) ao usuário e executado na máquina dele. Este procedimento adiciona uma possibilidade para o programador: um código que gera outro código sob medida que é executado na mesma ou em outra máquina.

Desta forma, é possível fazer com que aplicações distribuídas sejam enviadas em forma de código fonte, *bytecode*[10], dados ou ambos para um nó da rede, e este, executar os procedimentos programados. Este código pode ser parte de um grande programa e executar um procedimento isolado ou não. Este procedimento pode ser, em exemplo, um formulário, uma consulta, um relatório ou outra forma de aplicação.

Este método possibilita que se faça algo muito próximo ao desenvolvimento de aplicações WEB, mas aproveitando o poder da linguagem interpretada (neste caso, Python). Uma vez que um programa é desenvolvido desta forma, será possível a utilização de todos os recursos disponíveis nesta linguagem. Interações com o usuário partindo do servidor, ou novos componentes gráficos podem ser desenvolvidos sem a utilização de softwares adicionais ou proprietários, basta o código ser implementado e ficar disponível em um servidor ao usuário.

Conforme as especificações publicadas (W3C, 2005), existem grandes limitações que impossibilitam o desenvolvimento de aplicações mais complexas, uma vez que o último tem

10 Arquivo binário ou pré-compilado que é executado em uma máquina virtual ou interpretador

como ambiente (*jail*[11]) o próprio documento onde está inserido, não podendo interagir com outros dados na mesma máquina ou realizar conexões com outros nós da rede.

Soluções comerciais como Borland/Delphi (em modo *three tier*) são fortemente acopladas à sua tecnologia, se tornando difícil de expandir ou modificar partes dos código. Esta tecnologia, além de possuir código-fonte restrito, possui seus direitos defendidos por lei em nome da empresa que o produz, isto acaba por tornar difícil uma das características mais importantes deste modelo que será proposto: a flexibilidade/escalabilidade do código e do *framework*.

11 Um objeto é executado dentro de um ambiente fechado e limitado, não permitindo acesso aos recursos críticos da máquina ou do sistema operacional.

4. Objetivos

Este trabalho tem como objetivo geral, servir como *framework* para desenvolvimento de aplicações, locais ou distribuídas. Auxiliar um programador na criação de *frontends*[12] para as suas aplicações feitas em qualquer linguagem que suporte as dependências deste projeto.

Disponibilizar uma aplicação que sirva como geradora de interfaces com o usuário, montando as janelas com seus componentes e respeitando os comandos recebidos pelo servidor, de forma local ou remota.

Disponibilizar uma aplicação servidor que receba as conexões provenientes dos clientes e as tratem como programado. Responder o que for necessário sempre que solicitado, possibilitar acesso múltiplo.

Possibilitar o desenvolvimento rápido de aplicações gráficas, locais ou distribuídas. Fazer com que este desenvolvimento seja simples, rápido e prático, podendo ainda ser escalável e portável.

Implementar um protocolo de comunicação simples, textual e prático que sirva para a interação entre cliente e servidor de forma transparente ao usuário.

Desenvolver um sistema que permita a internacionalização de mensagens e outras formas de comunicação embarcadas na plataforma.

12 Programa/Interface de apresentação para aplicações

5. Metodologia

I. Levantamento bibliográfico sobre as linguagens, bibliotecas e outros componentes que poderão ser utilizados.

II. Estudo da linguagem e do *toolkit*[13] gráfico que será utilizado no projeto.

III. Especificação do método de transferência de dados via rede e a criação do protocolo base.

IV. Criação de um pequeno servidor que envie os dados ao cliente quando solicitado.

V. Entrega do trabalho

VI. Desenvolvimento do cliente:

1. Classe de protocolo de rede
2. Classe de componentes gráficos
3. Classe de controle de fluxo de informações

VII. Desenvolvimento do servidor:

1. Manipulador de conexões
2. Manipulador de multiprogramação
3. Classe de controle de fluxo de informações

VIII. Entrega e apresentação do trabalho

13 Biblioteca/ferramenta

6. Cronograma

Trabalho de Conclusão I

Etapa / Mes	Abril	Maio	Junho	Julho
I	X	X		
II		X	X	
III			X	X
IV				X
V				X

Trabalho de Conclusão II

Etapa / Mes	Agosto	Setembro	Outubro	Novembro	Dezembro
VI	X	X	X		
V1I		X	X	X	
VIII					X

7. Referências Bibliográficas

LUTZ, Mark. **Programming Python, Second Edition**. EUA: O'Reilly, 2001. 1255p.

MARTELLI, Alex. **Python in a Nutshell**. EUA: O'Reilly, 2003. 636p.

CATUNDA, Marco. **Guia de Consulta Rápida Python**. São Paulo: Novatec, 2001. 128p.

JEPSON, Brian; PECKHAM, Joan; SADASIV, Ram. **Programando Aplicativos de Banco de Dados em Linux**. São Paulo: Makron Books, 2002. 463p.

TANENBAUM, Andrew S. **Computer Networks**. EUA: PTR, 1996. 813p.

TITTEL, Ed et al. **World Wide Web com HTML e CGI:** bíblia do programador. São Paulo: Berkeley, 1996. 524p

W3C. Página de referências sobre o protocolo HTTP. Disponível em: <http://www.w3.org/Protocols/>. Acesso em: 30 mai. 2005.