Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis - IMESA

GUILHERME MENDES DOS SANTOS

COMPUTAÇÃO EM NUVEM UTILIZANDO FERRAMENTAS GOOGLE

2012
Assis - SP

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis - IMESA

COMPUTAÇÃO EM NUVEM UTILIZANDO FERRAMENTAS GOOGLE

Trabalho de Conclusão apresentado ao Curso de Bacharelado em Ciência da Computação do Instituto Municipal do Ensino Superior de Assis – IMESA e Fundação Educacional do Município de Assis – FEMA, como requisito para a obtenção do Certificado de Conclusão.

Orientado (a): Guilherme Mendes dos Santos
Orientador (a): Dr. Almir Rogério Camolesi

2012
Assis - SP

FICHA CATALOGRÁFICA

SANTOS, Guilherme Mendes

Computação em Nuvem utilizando ferramentas Google / Guilherme Mendes dos Santos. Fundação Educacional do Município de Assis – FEMA – Assis, 2012.

44 páginas.

Orientador. Dr. Almir Rogério Camolesi.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino de Assis – IMESA.

1. Computação em Nuvem, linguagem de programação, JAVA, Google Web Toolkit, Google App Engine.

CDD: 001.6

COMPUTAÇÃO EM NUVEM UTILIZANDO FERRAMENTAS GOOGLE

GUILHERME MENDES DOS SANTOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora.

Orientador: Dr. Almir Rogério Camolesi
Analisadora: Esp. Diomara Martins Reigato Barros

2012
Assis - SP

DEDICATÓRIA

Dedico este trabalho aos meus pais Iria e Hélio e toda a família que me deram muita força nesses últimos anos.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter sido meu ponto de apoio.

Aos meus pais Iria e Hélio que me deram forças nos momentos mais difíceis.

Ao meu orientador, Dr. Almir Rogério Camolesi e professores que me ensinaram ao longo destes anos.

Aos amigos que sempre me apoiaram e puderam compartilhar os momentos de tristeza e alegria.

RESUMO

Este trabalho apresenta o conceito de Computação em Nuvem utilizando a linguagem Java. A Computação em nuvem é uma tecnologia que vem ganhando espaço ao longo dos anos pela facilidade na utilização de serviços de qualquer plataforma, em qualquer lugar. Outra proposta interessante é a forma de pagamento destes serviços, chamada de Modelo de pagamento por uso. Essa nova solução despertou o interesse da empresa Google que criou ferramentas para o desenvolvimento de aplicativos voltados a Computação em Nuvem. Este trabalho também tem o objetivo de exemplificar através de um estudo de caso um sistema que aborde as características desta tecnologia.

Palavra-chave: Computação em Nuvem; linguagem de programa JAVA; Google Web Toolkit; Google App Engine.

ABSTRACT

This job introduces the concept of cloud computing using Java. The Cloud computing is a technology that has been gaining ground over the years by the ease of use of services from any platform, anywhere. Another interesting proposal is the form of payment for these services, called pay per use model. This new solution aroused the interest of the company that created Google tools for developing applications targeted at cloud computing. This work also aims to illustrate through a case study of a system that addresses the characteristics of this technology.

Keyword: Cloud Computing; program language JAVA, Google Web Toolkit, Google App Engine.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos, com o crescimento na área da Tecnologia da Informação, a internet vem sendo um meio bastante utilizado para enviar e ter acesso as informações, tanto para um usuário simples como empresas de grande porte, o que fez desenvolvedores de sistemas e fornecedores de equipamentos investirem em soluções para simplificar o acesso a estas informações. Entre todas estas tecnologias, uma que vem ganhando destaque é a Computação em nuvem (cloud Computing).
Segundo VELTE (2011, p.03):

> O nome computação em nuvem (cloud computing) é uma metáfora da Internet. Basicamente, a Internet é representada em diagramas de rede como uma nuvem. O ícone da nuvem representa “tudo isso e um poucos mais” que permite que a rede funcione. É como “etc.” para o restante do mapa da solução. Significa também uma área do diagrama ou da solução de qualquer individuo, então por que o diagrama esta fora? Essa noção é provavelmente a que melhor se encaixa ao conceito de cloud computing.

Já para DELIC e WALKER (2008), a nuvem é um complexo de hardwares, softwares, dados e pessoas que provêem serviços on-line. É um dos fundamentos da próxima geração de computação. É um mundo onde a rede é a plataforma para todos os computadores, onde tudo que pensamos como um computador, hoje é apenas um dispositivo que se conecta a um grande computador que estamos construindo.

1.1 OBJETIVOS

O objetivo deste trabalho é apresentar os conceitos e uma aplicação na linguagem Java [Tutorial Java: O que é Java, 2009] para as computações em nuvem, a fim de auxiliar novas pesquisas e incentivar a utilização desta nova tecnologia.

1.2 JUSTIFICATIVA

A tecnologia evolui e surgem novas formas de resolver problemas. A função da computação em nuvem é cortar custos operacionais e simplificar o acesso as informações de forma rápida e também remota, não só igualá-la a internet.

1.3 MOTIVAÇÃO

A computação em nuvem é uma idéia que nos permite utilizar as mais variadas aplicações via internet, em qualquer lugar e independente da plataforma, com transparência e facilidade de tê-las instaladas em nosso próprio computador.

1.4 PERSPECTIVAS DE CONTRIBUIÇÃO

A perspectiva é de que este trabalho divulgue as vantagens da computação em nuvem e sirva de base para estudos e implantação desta tecnologia.

1.5 METODOLIA DE PESQUISA

A metodologia de pesquisa adotada foi experimental. Inicialmente foi realizada uma pesquisa sobre o assunto. Com base nos conceitos adquiridos foi proposta uma aplicação. A aplicação desenvolvida foi um software com ferramentas disponibilizadas pelo Google através do Google App Engine [Porque o Google APP Engine, 2012], Java como linguagem e com auxilio do compilador Eclipse [Introdução á Plataforma Eclipse].

1.6 ESTRUTURAS DO TRABALHO

O trabalho foi estruturado da seguinte forma:

No Capítulo 2 é apresentado o conceito de Computação em nuvem, elementos que compõe uma Computação em nuvem, modelos de serviços e os modelos de implementação.

No Capítulo 3 é apresentado a plataforma de serviço que foi utilizada para a construção da aplicação.

No capítulo 4 é apresentado o Estudo de Caso, a aplicação usando Computação em nuvem.

Por fim, o Capítulo 5 conclui as idéias principais do trabalho.

2. COMPUTAÇÃO EM NUVEM

2.1 INTRODUÇÃO

A computação em nuvem é um dos assuntos mais comentados atualmente na área da informática, embora se pareça tratar de algo novo, o armazenamento denominado nuvem acontece desde o principio da internet.
Segundo BUYYA (2009, p.1):

> Computação em nuvem é uma tendência recente de tecnologia cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso. Tendências anteriores à computação em nuvem foram limitadas a uma determinada classe de usuários ou focadas em tornar disponível uma demanda específica de recursos de TI, principalmente de informática.

O que torna hoje a computação em nuvem uma tecnologia muito interessante é o avanço tecnológico na velocidade das conexões disponíveis e o ilimitado armazenamento de dados.
A figura 1 ilustra em uma visão geral uma nuvem computacional.

Figura 1. Ilustração de uma Computação em Nuvem (MARTIN, 2012).

Acredita-se que futuramente aplicativos e dados não serão armazenados em nossos computadores pessoais. Caberá a empresas fornecerem estes serviços através da internet, seguindo a idéia de Arquitetura Orientada a Serviço (*Service Oriented Architecture – SOA).*

Ou seja, através de um site o usuário cria uma conta, utiliza o aplicativo online e pode salvar todo o trabalho que for feito e acessá-lo em qualquer outro lugar desde que haja uma conexão com a internet.

MILLER (2008) destaca que, por se tratar de um novo paradigma, existem muitas contradições. Entretanto, a maioria dos pesquisadores considera que essa nova abordagem deva proporcionar economia de escala, uma vez que possibilitará que usuários domésticos, a partir de um computador com capacidades reduzidas ou até mesmo um televisor de alta definição, possa utilizar serviços especializados, oferecidos por companhias.

Basicamente há três partes que compõe os elementos de uma computação em nuvem: Clientes, *Data Center* e Servidores Distribuídos. Cada elemento desenvolve um papel especifico na solução.

2.2 CLIENTES

Clientes, segundo VELTE (2011), são definidos como os mesmos clientes em uma simples computação em rede local, em computação em nuvem são computadores que estão em nossas mesas, mas também máquinas com capacidade de mobilidade já que a computação em nuvem necessita apenas da internet.

Resumindo, os clientes são os usuários finais e na computação em nuvem são classificados em três categorias:

- Dispositivos móveis: PDAs ou Smartphones, Windows Smartphone ou Iphone.
- Clientes *thin*: Computadores sem disco rígido internos, o servidor faz todo o trabalho e os clientes *thin* exibem as informações.
- Thick: computador convencional onde utiliza o software pelo browser da web.

2.3 DATA CENTER

Data Center é um conjunto de servidores onde são armazenados os aplicativos que são acessados pela internet.
Uma técnica que vem crescendo na área de TI é a virtualização de servidores. O que permite que vários servidores sejam instalados em apenas uma maquina física, vale lembrar que o número de servidores instalados em uma única máquina dependerá do tamanho e da velocidade do servidor e quais aplicações estão funcionando no servidor virtual.
Nesta técnica o software pode ser instalado permitindo que vários servidores virtuais sejam utilizados. VELTE (2011, p.07).

2.4 SERVIDORES DISTRIBUÍDOS

Não há a necessidade dos servidores estarem alocados juntos. Normalmente estes servidores estão espalhados pelo mundo, mas para o usuário, estes servidores trabalham como se estivessem no mesmo local.
Isto proporciona ao fornecedor mais flexibilidade e opções de segurança. Se acontecer a perda de comunicação de algum servidor, o serviço ainda poderá ser acessado de outro servidor disponível. VELTE (2011, p.08).

2.5 MODELOS DE SERVIÇOS

Há muito mais coisas acontecendo por trás das nuvens do que simplesmente igualá-la a internet e armazenamento de informações, atualmente a computação em nuvem é classificada em três modelos de serviços. Software como Serviço (SaaS), Plataforma como um serviço (PaaS) e Infraestrutura como um serviço (IaaS).

2.5.1 SOFTWARE COMO UM SERVIÇO (SAAS)

O Software como Serviço (SaaS) nada mais é do que um aplicativo oferecido como um serviço aos clientes que acessam através da internet. Este tipo de software custa menos que comprar todo o aplicativo, resumindo, quanto menos ele for usado, mais ele será em conta.

Outra característica do Software como Serviço (SaaS) é que o cliente fica livre da obrigação de manter sua infra-estrutura atualizada e funcionando, além de fornecerem uma proteção mais eficiente.

Segundo AULBACH (2009, p.881):

> Um mesmo software pode ser utilizado por múltiplos usuários, sejam pessoas ou empresas. Esse tipo de serviço é executado e disponibilizado por servidores em Data Centers de responsabilidade de uma empresa desenvolvedora, ou seja, o software é desenvolvido por uma empresa que ao invés de vendê-lo ou usá-lo para beneficio exclusivo, disponibiliza-o a um custo baixo a uma grande quantidade de usuários.

A figura 2 ilustra um Software como um Serviço (SaaS).

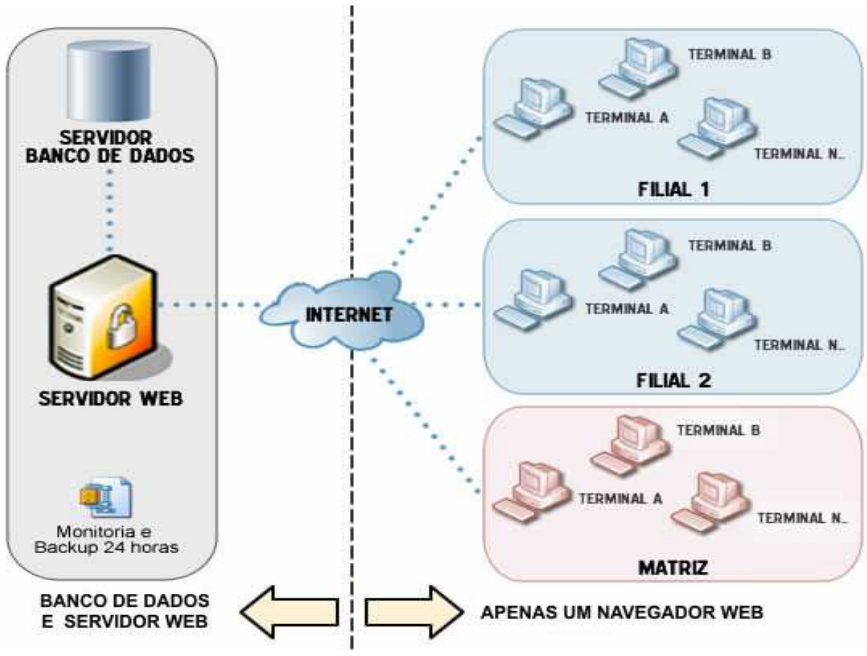

Figura 2. Representação do modelo de Software como um Serviço (WEBSOFTWARE BRASIL, 2012).

2.5.2 PLATAFORMA COMO UM SERVIÇO (PAAS)

A Plataforma como um Serviço (PaaS) bem como os SaaS é outro modelo de aplicação, na Plataforma como um Serviço (PaaS) é fornecido todos os recursos e ferramentas necessárias para que desenvolvedores de software construa novos aplicativos e serviços diretamente da internet sem precisar instalá-las em seu computador pessoal.

O fator de definição que torna PaaS exclusiva é que permite que desenvolvedores desenvolvam e implementem aplicativos da Web em uma infraestrutura hospedada. Ou seja, PaaS permite aproveitar os recursos de computação aparentemente infinitos de uma infraestrutura de nuvem.(ORLANDO, 2011).

A figura 3 mostra uma relação das classificações de computação em nuvem e os elementos de Plataforma como um Serviço (PaaS).

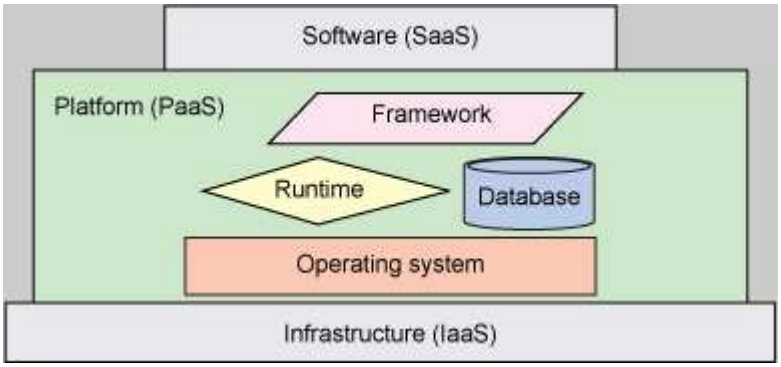

Figura 3. Uma interpretação gráfica do relacionamento entre classificações de computação em nuvem e os elementos de PaaS. (ORLANDO, 2011).

2.5.3 INFRAESTRUTURA COMO UM SERVIÇO (IAAS)

Enquanto os SaaS e PaaS oferecem aplicações aos clientes, a Infraestrutura como um Serviço (IaaS) fornece apenas recursos de Hardware, geralmente na forma de virtualização, podem ser fornecidos recursos como:

- Espaço Físico para servidor;
- Memória;
- Equipamentos de rede;
- Ciclos de CPU;

- Espaço de Armazenamento.

Na forma de virtualização, vários usuários podem utilizar os recursos de uma única máquina simultaneamente.

O pagamento para a utilização destes recursos é de acordo com a quantidade utilizada, ou seja, ela é ajustada de acordo com a necessidade do cliente.

Segundo SOUSA (2010, p.8):

> O termo IaaS se refere a uma infraestrutura computacional baseada em técnicas de virtualização de recursos de computação. Esta infraestrutura pode escalar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. Do ponto de vista de economia e aproveitamento do legado, ao invés de comprar novos servidores e equipamentos de rede para a ampliação de serviços, podem-se aproveitar os recursos disponíveis e adicionar novos servidores virtuais à infraestrutura existente de forma dinâmica.

A figura 4 ilustra uma Infraestrutura como um Serviço (IaaS).

Figura 4. Representação de Infraestrutura como um serviço (FLAGSHIP NETWORKS INC., 2012).

2.6 MODELOS DE IMPLANTAÇÃO DE UMA NUVEM

A implementação de solução de Computação em nuvem depende do objetivo a ser alcançado e da necessidade para cada solução computacional, atualmente a Computação em nuvem é dividido em categorias, são elas:

- Nuvem Pública
- Nuvem Privada
- Nuvem Comunitária
- Nuvem Híbrida

2.6.1 NUVEM PÚBLICA

Uma nuvem pública é o modelo padrão de computação em nuvem, na qual um prestador de serviços fornece recursos como aplicativos, plataformas e infraestrutura para o público global por meio da web. Estes serviços na nuvem pública podem ser gratuitos ou em modelo onde é pago de acordo com o uso destes recursos.
Segundo PEREIRA (2012, p. 03):

> As infraestruturas estão disponíveis para o uso do público em geral e são gerenciadas pelas empresas fornecedoras que disponibilizam serviços em nuvem de forma que o usuário paga pelo que usar. As infraestruturas são instaladas nas empresas fornecedoras e, muitas vezes, os usuários não possuem conhecimento da localização dos dados.

2.6.2 NUVEM PRIVADA

Uma nuvem privada é o modelo construído exclusivamente para um único usuário, como uma empresa por exemplo. A infraestrutura utilizada da ao usuário o total controle da nuvem, criando suas próprias regras de negócio.
Segundo PEREIRA (2012, p. 03):

> As infraestruturas são criadas para o uso exclusivo de uma organização e são gerenciadas pelo departamento de TI, terceirizadas ou de forma combinada. Esses recursos são geralmente utilizados para fornecer serviços internos ou utilizados como mecanismo de regras de negócio da organização. A infraestrutura pode estar localizada dentro ou fora da organização (exemplo Data Centers). Uma nuvem privada pode proporcionar maior controle sobre os dados.

2.6.3 NUVEM COMUNITÁRIA

Nuvem comunitária nada mais é que uma nuvem privada compartilhada entre algumas organizações. Um exemplo seria uma nuvem compartilhada entre vários campi de universidades.

Segundo PEREIRA (2012, p.03):

> As infraestruturas são compartilhadas por várias organizações que possuem propósitos e interesses comuns (política, missão, requisitos de seguranças). Essas infraestruturas podem ser gerenciadas pelo departamento de TI das organizações da comunidade, terceirizadas ou de forma combinada. A infraestrutura pode estar localizada dentro ou fora das organizações (exemplo Data Centers).

2.6.4 NUVEM HÍBRIDA

Nuvem híbrida é a combinação de dois ou mais modelos de implantação, um ambiente que está sendo preferido pelas empresas permitindo que uma empresa estabeleça a melhor formação para seu modelo de negócios. Isso faz com que a empresa controle aplicativos que não queira deixar público e desprotegido, mas continua utilizando os recursos públicos de uma nuvem quando necessário para a empresa.

Segundo PEREIRA (2012, p.03):

> As infraestruturas são compostas por dois ou mais modelos de implantação (nuvem privada, nuvem comunitária, nuvem pública). Em uma nuvem híbrida, os modelos de implantação trabalham como se fossem uma única nuvem. Organizações utilizam esse modelo para ter maior controle sobre os dados podendo, por exemplo, utilizar um modelo de implantação privado para controlar os dados internos da organização e um modelo de implantação público para as regras de negócios.

A figura 5 ilustra um modelo de Nuvem Híbrida.

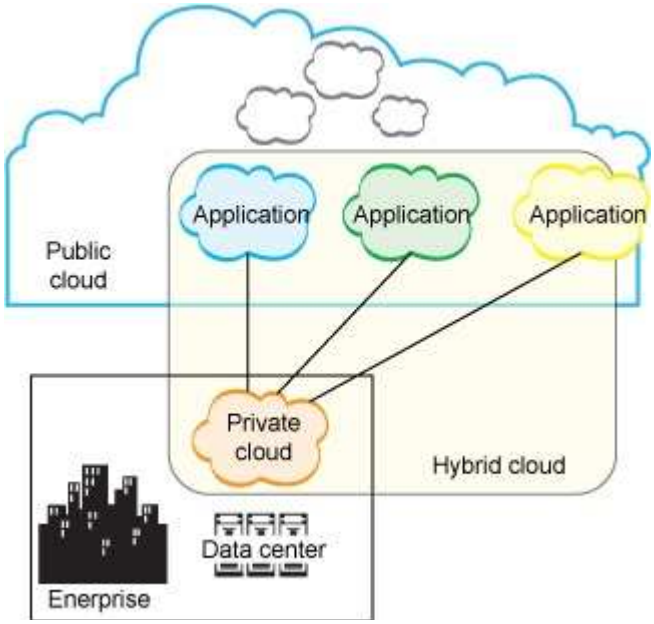

Figura 5. Três nuvens públicas utilizadas com uma privada para formar uma híbrida (WALKER, 2012).

3. GOOGLE E NUVEM

Não existe nada nos dias de hoje que a Google não tenha envolvimento, não poderia ser diferente com a computação em nuvem que está sendo um dos maiores serviços que a Google está fornecendo aos seus clientes. O Google já trabalha no desenvolvimento de computação em nuvem há mais de dez anos e disponibiliza uma Plataforma como um Serviço (PaaS) para o desenvolvimento e hospedagem de aplicações web de forma gratuita até determinado consumo. Este conjunto de ferramentas criado inicialmente em Abril de 2008 é chamado de Google App Engine.

3.1 GOOGLE APP ENGINE

Segundo VELTE (2011), o Google App Engine é uma Plataforma como um Serviço (PaaS) que permite que você execute seus aplicativos pela web, na infraestrutura do Google. Com o Google App Engine, os desenvolvedores podem realizar as seguintes tarefas:

- Criar o código de uma vez e implantá-lo: O Google App Engine torna fácil a implementação, pois fornece recursos de computação de acordo com a necessidade. Os desenvolvedores apenas criam seus códigos e o Google App Engine cuida do resto.
- Absorver picos de tráfego: Um problema que acontece em todos os aplicativos web é o volume de tráfego quando sua aplicação se torna popular, o que aumenta desastrosamente o acesso a ela, o Google App Engine faz uma reprodução automática e balanceamento de carga, facilitando a escala de um usuário para um milhão de usuários, levando vantagem sobre outros componentes da infraestrutura escalável do Google.
- Fácil integração com outros serviços do Google: O desenvolvedor que utiliza o Google App Engine não precisa criar componentes de autenticação e email toda vez para cada novo pedido. Os desenvolvedores têm acesso a todos os componentes que acompanham o aplicativo e biblioteca externa do Google

de APIs.Com o Google App Engine você cria o código de seu aplicativo, e testa localmente e depois com apenas um clique enviá-lo ao Google.

A criação de um aplicativo na ferramenta Google é gratuita, porém se exceder 500MB de armazenamento ou 5 milhões de visualizações de pagina por mês é possível ativar o faturamento, definir um orçamento máximo diário e distribuir seu orçamento para cada recurso de acordo com suas necessidades. (GOOGLE, 2012).

3.1.1 SANDBOX

Consiste na idéia que quando se permite um programa executar em uma máquina, deseja-se provar um meio no qual o programa possa executar com segurança e confiança, mas neste meio possui seus limites. Assim permite-se ao programa interagir com certos recursos, contudo tem a certeza que este programa não vai ultrapassar seus limites, ou seja, utilizar recursos que não sejam permitidos.

No sandbox o aplicativo é isolado em seu próprio ambiente seguro e confiável, independentemente de hardware, sistema operacional e localização física do servidor da web.

Segundo MÜLLER (2010, p.32).

> O App Engine, por se tratar de um sistema baseado em computação em nuvem, em que o processamento e armazenamento do aplicativo são distribuídos entre vários servidores, necessita de um ambiente virtual seguro para cada aplicativo. Este ambiente chamado de sandbox fornece acesso limitado ao sistema operacional, tal limitação possibilita que o Google App Engine distribua as solicitações de web da aplicação entre diversos servidores, podendo iniciar ou interromper os servidores para atender às demandas de tráfego e também que cada aplicação possua uma área isolada segura e confiável independente de hardware, sistema operacional e localização física do servidor, garantindo que uma aplicação não influencie no funcionamento das demais aplicações. Este método de virtualização além de possibilitar a distribuição na execução do aplicativo, evita o chamado efeito slashdot, onde em um ambiente compartilhado, o uso abusivo de recursos por uma aplicação afeta o desempenho das demais.

3.1.2 ARMAZENAMENTO DE DADOS

O Google App Engine oferece um poderoso sistema de armazenamento de dados distribuídos que contém um mecanismo de consulta e transações. Da mesma forma que o trafego de um serviço da web cresce, o armazenamento de dados cresce conforme o volume de dados aumente.

O armazenamento de dados do Google App Engine suporta um conjunto fixo de tipos de valores para as propriedades nas entidades de dados, porem o armazenamento de dados não é um banco de dados relacional tradicional e também não possuem esquemas. As entidades têm um tipo e um conjunto de propriedades e sua estrutura é fornecida e aplicada pelo código de seu aplicativo.

A atualização das entidades é feita usando “grupos de entidade”, ou seja, quando é feita uma transação é manipulado entidades dentro de um único grupo. Quando as entidades são do mesmo grupo, elas são armazenadas juntas tornando uma execução de transações eficiente. (GOOGLE, 2012).

3.2 GOOGLE WEB TOOLKIT

GOOGLE (2012) explica que *O Google App Engine suporta aplicativos criados em várias linguagens de programação. O ambiente de execução em Java do Google App Engine permite criar o seu aplicativo usando tecnologias Java padrão, incluindo JVM, servlets Java e a linguagem de programação Java, ou qualquer outra linguagem que usa um interpretador ou compilador com base na JVM, como JavaScript.*

Porém o Google já afirmou que pretende apoiar mais idiomas no futuro, e que o Google App Engine foi escrito para ser independente de linguagem.

VELTE (2011, p.44) destaca que, Google Web Toolkit inclui suporte em linguagem Java, esta capacidade inclui genéricos Java, tipos enumerados, anotações, autoboxing, lista de parâmetro variável e muito mais.

No desenvolvimento de uma aplicação, usaremos a ferramenta Google Web Toolkit, um kit de desenvolvimento de software para construção e otimização de aplicativos

baseados em navegador que permite escrever aplicativos Java e depois compilar o código fonte para JavaScript que funciona em todos os navegadores, incluindo os móveis. Também será utilizado o Eclipse IDE, instalado com o Google Plugin para Eclipse que adiciona assistentes de novos projetos e configurações de depuração para o IDE.

3.2.1 INSTALAÇÃO

A instalação ocorre nos seguintes procedimentos:

a) Obter através do site oficial o software Eclipse para Desenvolvedor Java EE, a versão utilizada será o Eclipse 3.7 (Índigo).
b) Obter através do site oficial o Google Plugin para Eclipse 3.7 (Índigo).
c) Inicie o Eclipse, em seguida, selecione ***Help > Install new Software*****.** Na caixa de diálogo que aparece, digite a URL do site de atualização para o ***Work with:*** http://dl.google.com/eclipse/plugin/3.7 e pressione a tecla *"Enter"*. Note que aparecerá os plugins e SDKs, selecione a caixa de seleção ao lado de ambos, isso ira instalar o plugin. Selecione o botão "***Next".***
d) Ao final do processo de instalação selecione a opção ***"Restar Now"***.

3.2.2 CRIAÇÃO DE UM PROJETO

Terminada a instalação, o Eclipse IDE esta pronto para criar projetos de aplicações Web como ilustrada na figura 6.

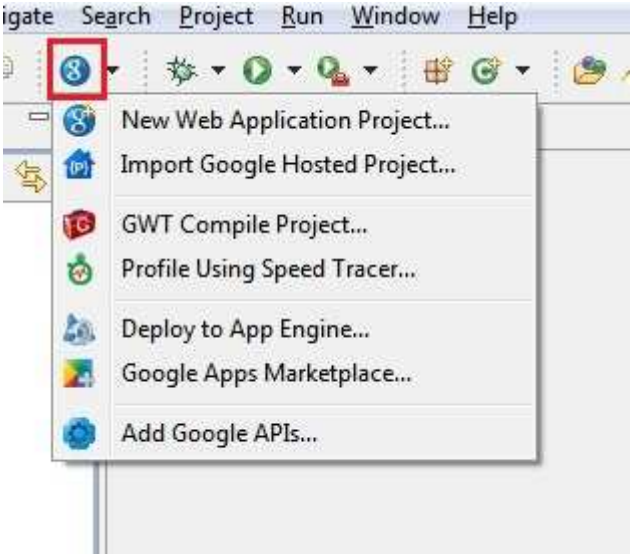

Figura 6. Novo Projeto.

3.2.3 ENVIAR UM APLICATIVO PARA O GOOGLE APP ENGINE

Primeiramente é necessário registrar uma conta App Engine pelo site: https://accounts.google.com/ServiceLogin?service=ah&passive=true&continue=https://appengine.google.com/_ah/conflogin%3Fcontinue%3Dhttps://appengine.google.com/&ltmpl=ae. Depois de criada a conta, você terá uma tela de controle de suas aplicações. Por padrão a Google disponibiliza gratuitamente a criação de no Máximo 10 aplicações.

O próximo passo é criar uma aplicação como é mostrado na figura 7.

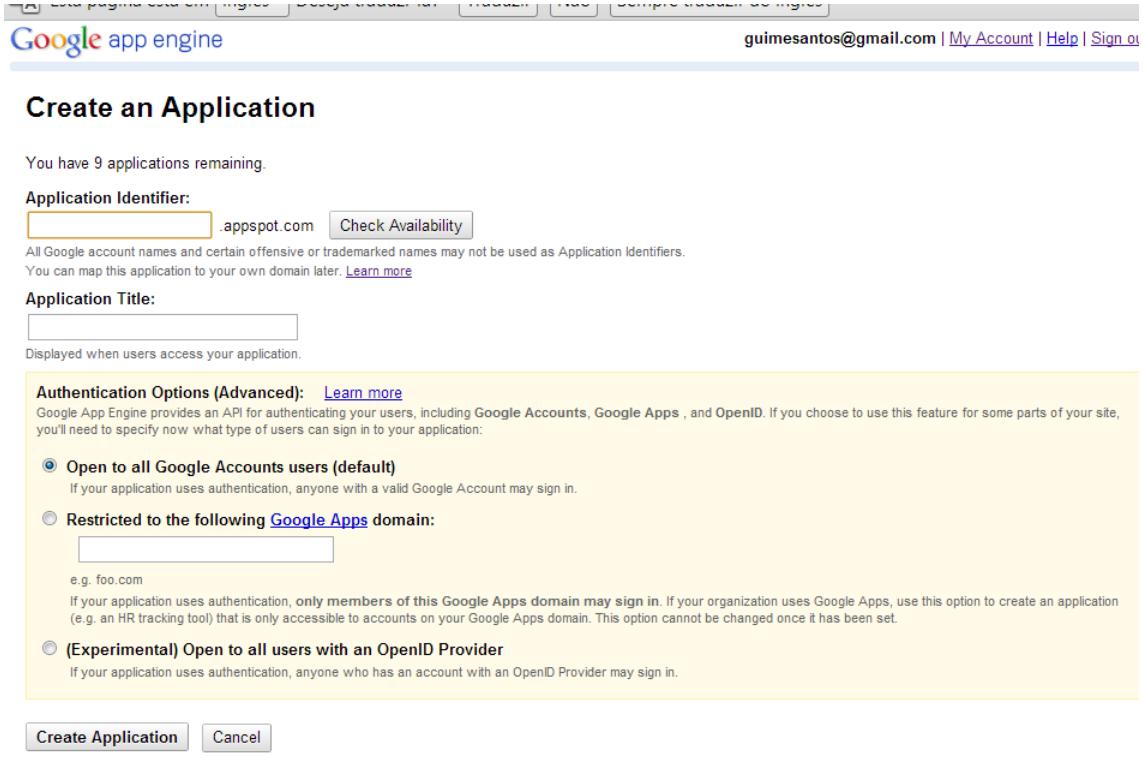

Figura 7. Criando uma aplicação.

Nesta tela é criado o *Application Identifier*, que será um nome único para obter acesso a sua aplicação pelo navegador. Criado seu identificador você terá acesso a configurações e controle de suas aplicações como Dados, Administração, Faturamento entre outras opções como mostra a figura 8.

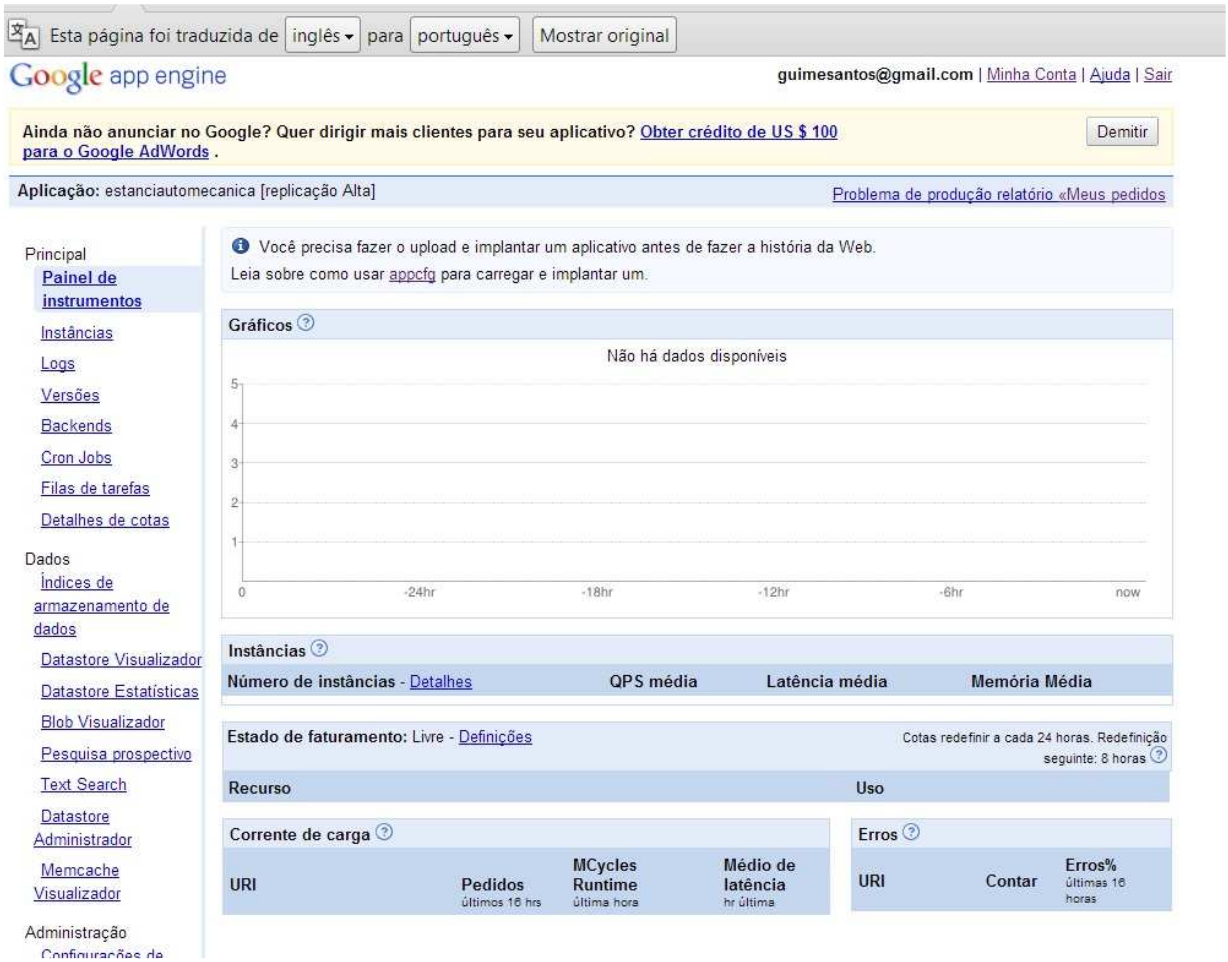

Figura 8. Painel de Controle de aplicações.

Depois de escrito e compilado, o código ainda precisa ser enviado para as nuvens, que passa por um procedimento simples, basta clicar em um botão com a descrição "*Deploy to App Engine...*" como mostra a figura 10.

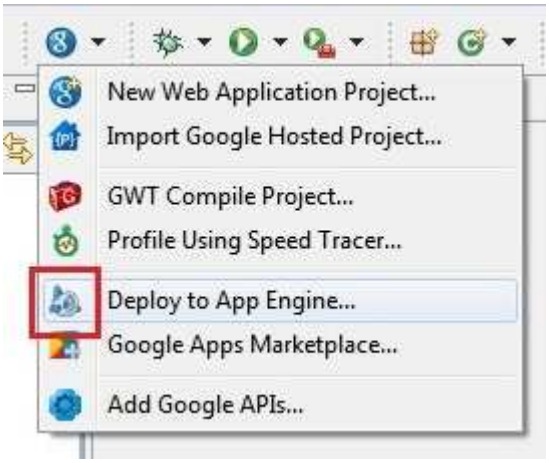

Figura 9. Construindo o projeto.

Agora a aplicação esta inteiramente funcionando na nuvem e pode ser acessada pelo *browser* como o exemplo: `http://application-id.appspot.com/` onde *application-id* é o ID do aplicativo App Engine criado anteriormente.

4. ESTUDO DE CASO

4.1 INTRODUÇÃO

Com base no estudo realizado sobre a tecnologia Google em Nuvem, será desenvolvida uma aplicação capaz de utilizar os aspectos apresentados anteriormente.

Trata-se de um sistema que desempenha a documentação de serviços de uma oficina mecânica desenvolvida para web com o auxilio da ferramenta Google Web Toolkit, no qual o software documenta serviços prestados pela firma, e auxilia na prevenção de troca de itens como óleos e filtros. Tudo isso sendo executado diretamente pela estrutura de nuvem da Google App Engine.

Para a elaboração do projeto foi utilizada a tecnologia Java na versão 6, como linguagem de programação, o plugin contendo o kit de desenvolvimento Google App Engine Java e a ferramenta Google Web Toolkit SDK 2.4.0 no ambiente de desenvolvimento Eclipse.

4.2.1 REQUISITOS FUNCIONAIS

A aplicação exercerá as seguintes funções:

a) O administrador da aplicação terá a possibilidade de criar, excluir, editar e consultar clientes, automóveis e usuários.
b) O administrador também terá a possibilidade de criar, excluir e editar serviços que deverão ser executados pela oficina.
c) Deve ser possível iniciar e finalizar relatório de serviço a executar.

4.2.2 REQUISITOS NÃO FUNCIONAIS

A aplicação conta também com requisitos que não são funcionais listados a seguir:

a) Para ter acesso ao aplicativo, o administrador da aplicação deve se autenticar através da sua conta de usuário.
b) Ao se criar ou editar clientes, carros e usuários, deve realizar a validação dos campos.
c) O modelo de implantação será de Nuvem Privada.
d) A interface gráfica deve ser feita utilizando a ferramenta Google Web Toolkit.

4.2.3 ESTRUTURA DO TRABALHO

Com a análise feita e identificado os interesses principais para a aplicação, pode-se fazer o levantamento dos requisitos funcionais para estruturar a arquitetura do sistema, resumindo, os requisitos fundamentais para que o sistema cumpra com a necessidade do cliente. Esses requisitos funcionais podem-se representar através de um diagrama de caso de uso, como na figura a seguir.

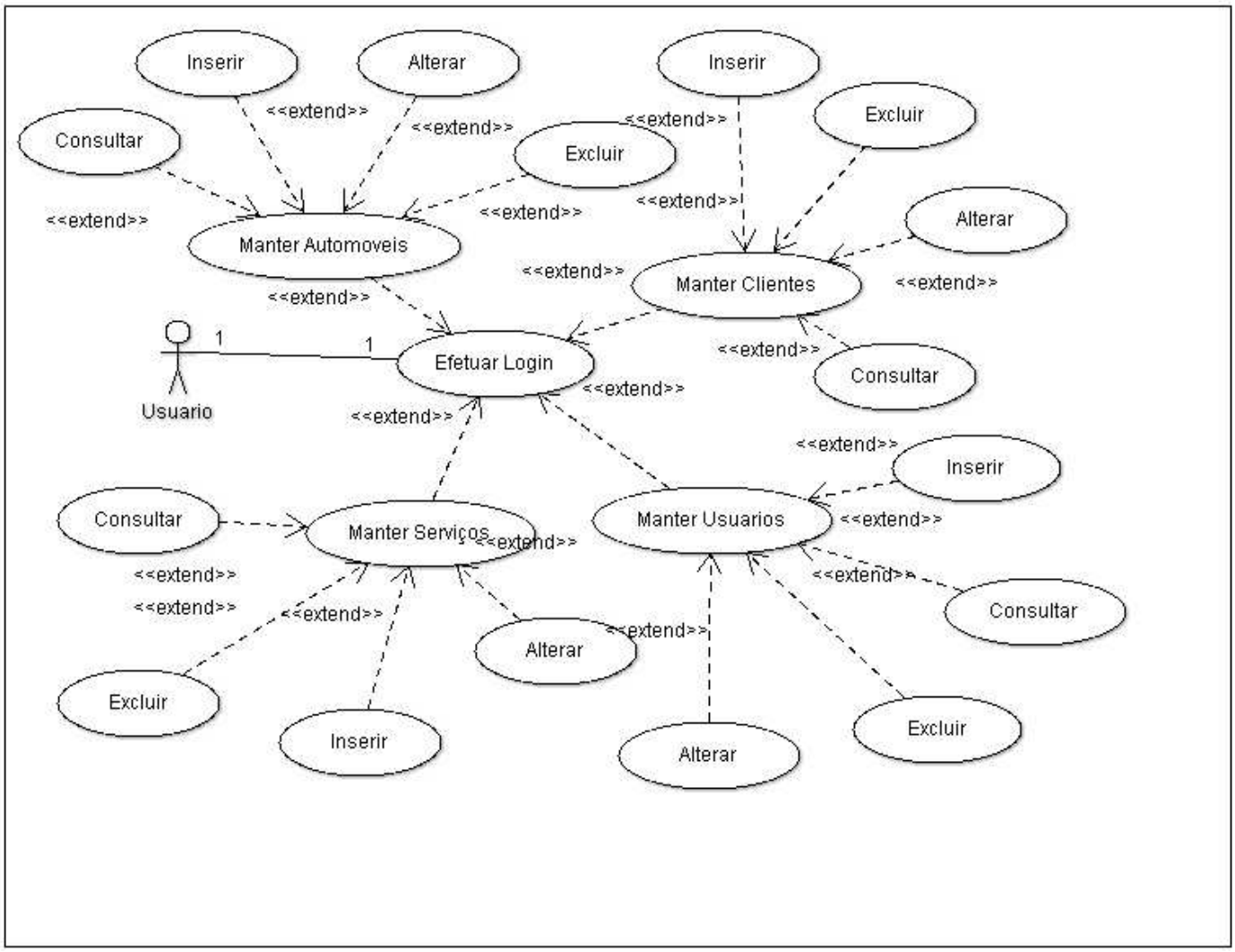

Figura 10. Diagrama de Caso de Uso.

Do mesmo modo, as tabelas necessárias para armazenar os dados da aplicação podem ser representadas no Diagrama de Classes ilustrado na figura a seguir.

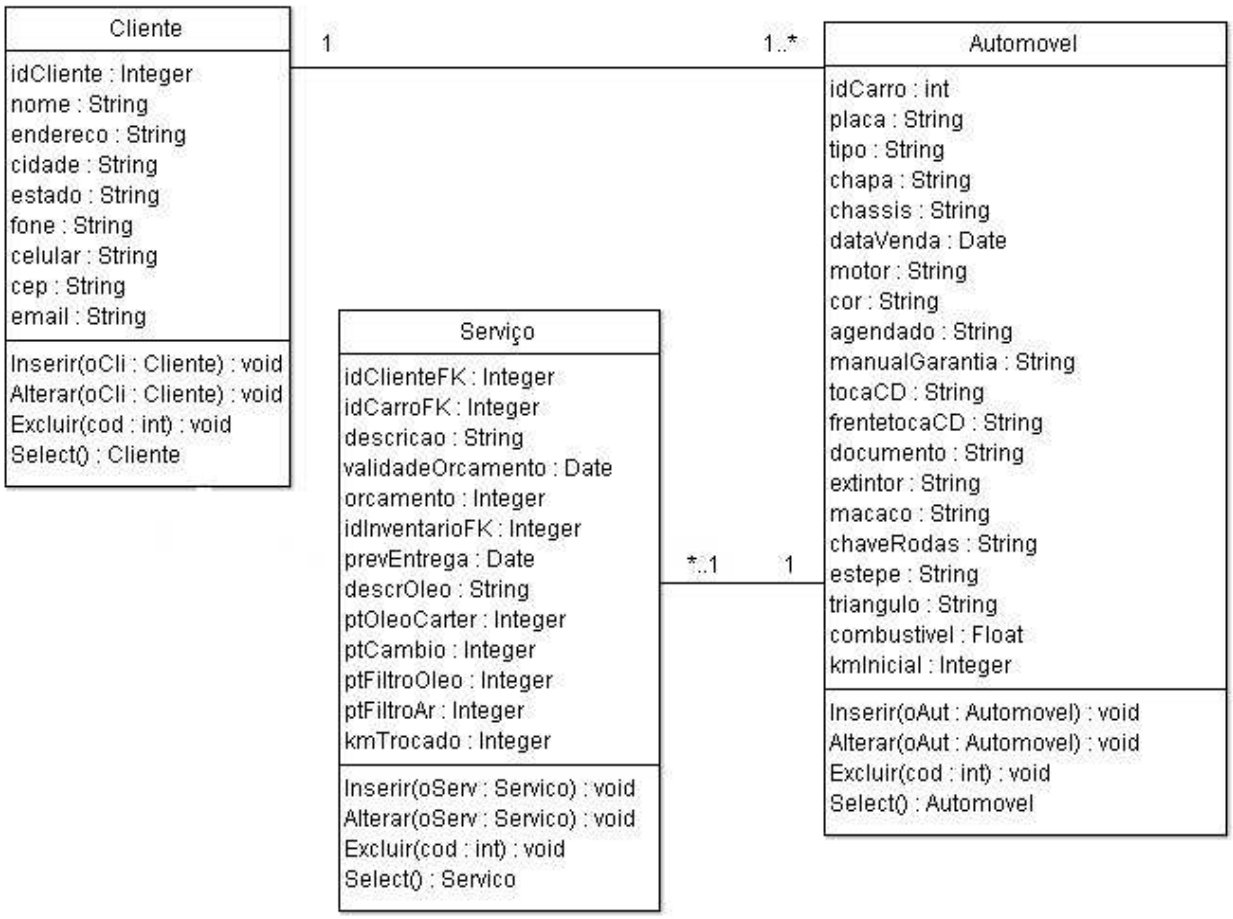

Figura 11. Diagrama de Classes

4.2.4 APLICAÇÃO

4.2.4.1 UTILIZANDO SERVLETS

A Aplicação demonstra o uso dessa nova tecnologia. Como padrão em programação Java este projeto utiliza *servlets* que é responsável por fornecer acesso a recursos do lado do servidor. Ou seja, a comunicação entre cliente-servidor é codificado pelo *servlet* que faz a comunicação com o aplicativo.

Os *Servlets* são encontrados na pasta *WEB-INF* com o nome *web.xml.*

Quando criamos novas classe de implementação pelo cliente, temos que configurar o *Servlet* para que possa haver a comunicação com o servidor como mostra o código abaixo:

```
<servlet>
    <servlet-name>clienteServlet</servlet-name>
    <servlet-class>br.edu.fema.projeto.servlets.ClienteServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>clienteServlet</servlet-name>
    <url-pattern>/mecanica/cliente</url-pattern>
  </servlet-mapping>
```

Com o *servlet* configurado, podemos utilizar seus recursos como mostra o código a seguir.

```
package br.edu.fema.projeto.servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import br.edu.fema.projeto.dao.ClienteDAO;
import br.edu.fema.projeto.vo.ClienteVO;


public class ClienteServlet extends HttpServlet {

        @Override
        protected void doPost(HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException {

                Long codigo = Long.parseLong(req.getParameter("codigo"));
                String nome = req.getParameter("nome");
                String endereco = req.getParameter("endereco");
                String cidade = req.getParameter("cidade");
                String estado = req.getParameter("estado");
                String telefone = req.getParameter("telefone");
                String celular = req.getParameter("celular");
                String cep = req.getParameter("cep");
                String email = req.getParameter("email");

                ClienteVO cliente = new ClienteVO(codigo, nome, endereco,
                cidade, estado, telefone, celular, cep, email);

                ClienteDAO dao = new ClienteDAO();

                try {
                        dao.inserir(cliente);
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        req.getRequestDispatcher("../clienteView.jsp").forward(req, resp);
                }
```

```
        }

        @Override
        protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException {
                doPost(req, resp);
        }

}
```

A finalidade do *servlet* é enviar os dados do cliente para o servidor, porem se falhar é retornado o erro ocorrido.

4.2.4.2 ARMAZENANDO DADOS COM O JDO

GOOGLE (2012) explica que *A JDO (Objetos de dados Java) é uma interface padrão para armazenar objetos que contêm dados em um banco de dados. O padrão define interfaces para anotar objetos Java, recuperando objetos com consultas e interagindo com um banco de dados usando transações. Um aplicativo que usa a interface JDO pode funcionar com vários bancos de dados sem usar qualquer código específico a um banco de dados, incluindo bancos de dados relacionais, bancos de dados hierárquicos e bancos de dados de objetos. Como com outros padrões de interface, JDO simplifica a portabilidade de seu aplicativo entre diferentes soluções de armazenamento.*

Quando criamos um projeto utilizando o Eclipse junto com o plugin do Google App Engine, é criado e configurado automaticamente um arquivo chamado *jdoconfig.xml* que é encontrado no diretório *src/META –INF.* Ele é responsável por informar à JDO que será utilizado o armazenamento de dados do Google App Engine.

O código abaixo mostra o conteúdo do *jdoconfig.xml.*

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<jdoconfig xmlns="http://java.sun.com/xml/ns/jdo/jdoconfig"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="http://java.sun.com/xml/ns/jdo/jdoconfig">

   <persistence-manager-factory name="transactions-optional">
       <property name="javax.jdo.PersistenceManagerFactoryClass"

       value="org.datanucleus.store.appengine.jdo.DatastoreJDOPersistenceManagerFactory"/>
       <property name="javax.jdo.option.ConnectionURL" value="appengine"/>
       <property name="javax.jdo.option.NontransactionalRead" value="true"/>
       <property name="javax.jdo.option.NontransactionalWrite" value="true"/>
```

```
        <property name="javax.jdo.option.RetainValues" value="true"/>
        <property name="datanucleus.appengine.autoCreateDatastoreTxns" value="true"/>
        <property name="datanucleus.appengine.singletonPMFForName" value="true"/>
    </persistence-manager-factory>
</jdoconfig>
```

Seguindo esta configuração, as classes utilizadas para fazer a persistência dos dados foram criadas como o código a seguir.

```
package br.edu.fema.projeto.vo;

import java.io.Serializable;
import javax.jdo.annotations.PersistenceCapable;
import javax.jdo.annotations.Persistent;
import javax.jdo.annotations.PrimaryKey;

@PersistenceCapable(detachable = "true")
public class ClienteVO implements Serializable{

        private static final long serialVersionUID = 1L;
        @PrimaryKey
        @Persistent
        private Long codigo;
        private String nome;
        @Persistent
        private String endereco;
        @Persistent
        private String cidade;
        @Persistent
        private String estado;
        @Persistent
        private String telefone;
        @Persistent
        private String celular;
        @Persistent
        private String cep;
        @Persistent
        private String email;

        public ClienteVO() {
                super();
        }
```

```
                public ClienteVO(Long codigo, String nome, String endereco,
                        String cidade, String estado, String telefone, String celular,
                        String cep, String email) {
                super();
                this.codigo = codigo;
                this.nome = nome;
                this.endereco = endereco;
                this.cidade = cidade;
                this.estado = estado;
                this.telefone = telefone;
                this.celular = celular;
                this.cep = cep;
                this.email = email;
        }

        public Long getCodigo() {
                return codigo;
        }
        public void setCodigo(Long codigo) {
                this.codigo = codigo;
        }
```

```
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getEndereco() {
		return endereco;
	}
	public void setEndereco(String endereco) {
		this.endereco = endereco;
	}
	public String getCidade() {
		return cidade;
	}
	public void setCidade(String cidade) {
		this.cidade = cidade;
	}
	public String getEstado() {
		return estado;
	}
	public void setEstado(String estado) {
		this.estado = estado;
	}
	public String getTelefone() {
		return telefone;
	}
	public void setTelefone(String telefone) {
		this.telefone = telefone;
	}
	public String getCelular() {
		return celular;
	}
	public void setCelular(String celular) {
		this.celular = celular;
	}
	public String getCep() {
		return cep;
	}
	public void setCep(String cep) {
		this.cep = cep;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
}
```

A anotação *@PersistenceCapable* na classe faz com que a mesma seja capaz de ser armazenada e recuperada no armazenamento de dados com JDO. Assim como campos de dados da classe para que possam também ser armazenados precisam de uma anotação chamada *@Persistent* antes de sua declaração.

Para que algum campo seja apontado como chave primária como o campo código da classe Cliente é necessário usar a anotação *@PrimaryKey.*

Outra classe importante para esta aplicação é chamada de *PMF.java* que tem como função criar uma instância com o banco do servidor sem ser preciso fazer requisições a cada manipulação, como é mostrado abaixo.

```
package br.edu.fema.projeto.db;

import javax.jdo.JDOHelper;
import javax.jdo.PersistenceManagerFactory;

public class PMF {

	public static final PersistenceManagerFactory factory =
JDOHelper.getPersistenceManagerFactory("transactions-optional");

	private PMF() {
	}

	public static PersistenceManagerFactory getInstance() {
		return factory;
	}

}
```

4.2.4.3 JAVA SERVER PAGES

Para esta aplicação foi usado *JavaServer Pages(JSPs)* para implementar a interface.

GOOGLE (2012) define *JavaServer Pages* como *um sistema de modelo com interface de usuário projetada e implementada em arquivos separados com espaços reservados e lógica para inserir dados fornecidos pelo aplicativo.*

O código abaixo mostra a criação de uma *JavaServer Page* para a interface de manipulação de dados de clientes.

```
<%@page import="br.edu.fema.projeto.vo.ClienteVO"%>
<%@page import="br.edu.fema.projeto.dao.ClienteDAO"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Cadastro de Cliente</title>
</head>
<body>

<h3>Cadastro de Cliente</h3>

<form action="/mecanica/cliente" method="post">
	<table>
		<tr>
			<td>Codigo</td><td><input type="text" id="codigo" name="codigo" /></td>
```

```
                </tr>
                <tr>
                        <td>Nome</td><td><input type="text" id="nome" name="nome" /></td>
                </tr>
                <tr>
                        <td>Endereco</td><td><input type="text" id="endereco" name="endereco" /></td>
                </tr>
                <tr>
                        <td>Cidade</td><td><input type="text" id="cidade" name="cidade" /></td>
                </tr>
                <tr>
                        <td>Estado</td><td><input type="text" id="estado" name="estado" /></td>
                </tr>
                <tr>
                        <td>Telefone</td><td><input type="text" id="telefone" name="telefone" /></td>
                </tr>
                <tr>
                        <td>Celular</td><td><input type="text" id="celular" name="celular" /></td>
                </tr>
                <tr>
                        <td>CEP</td><td><input type="text" id="cep" name="cep" /></td>
                </tr>
                <tr>
                        <td>E-Mail</td><td><input type="text" id="email" name="email" /></td>
                </tr>

                <tr>
                        <td>
                        <input type="submit" value="Enviar" />
                        </td>
                </tr>
        </table>
</form>

<table>
<%
        ClienteDAO dao = new ClienteDAO();
```

```
        for (ClienteVO cliente :
        dao.selecionarTodos()) {
%>
        <tr>
        <td><%=cliente.getCodigo() %></td>
        <td><%=cliente.getNome() %></td>
        <td><%=cliente.getEndereco() %></td>
        <td><%=cliente.getCidade() %></td>
        <td><%=cliente.getEstado() %></td>
        <td><%=cliente.getTelefone() %></td>
        <td><%=cliente.getCelular() %></td>
        <td><%=cliente.getCep() %></td>
        <td><%=cliente.getEmail() %></td>
        </tr>
<%
        }
%>
</table>

</body>
</html>
```

5. CONCLUSÃO

As vantagens da utilização desta tecnologia é a possibilidade de reutilizar softwares, plataformas para aplicações e infraestrutura, diminuindo custos e proporcionando mobilidade aos usuários. Neste ambiente, todo software desenvolvido tem a capacidade de ser executado em qualquer plataforma de software e hardware, com total transparência ao usuário sem precisar instalar o software em computadores, bastando somente acessar o aplicativo pelo navegador de internet. Outras vantagens citadas anteriormente é deixar para que a empresa fornecedora destes serviços tenha total responsabilidade no cuidado da sua aplicação como na proteção dos dados contra roubos de informações, perda de dados, custos de hardware como servidores, energia, manutenção dos computadores, softwares pagos entre outros fatores diminuindo assim muitas despesas.

A aplicação realizada no trabalho mostra um bom exemplo destes fatores, pois após o aplicativo ser enviado para os servidores da Google, o usuário só tem o trabalho de acessar via *browser* o aplicativo, independente de que o usuário esteja usando um computador de alto desempenho ou até mesmo um aparelho móvel com acesso a internet, já que os processos do aplicativo usarão os recursos de infraestrutura a partir das máquinas da empresa prestadora deste serviço.

Nos dias atuais, a afirmação de que a computação em nuvem ira facilitar o setor de TI cresce cada vez mais, já que muitas empresas já estão adotando os serviços da Computação em Nuvem. Algumas pesquisas informam que até 2015 o tráfego global de Computação em Nuvem irá crescer até 12 vezes.

6. REFERÊNCIAS BIBLIOGRÁFICAS

AULBACH, Stefan; JACOBS, Dean; KEMPER, Alfons; et al. **A Comparison of Flexible Schemas for Software as a Service.** 35th SIGMOD - International Conference on Management of Data, 2009.

BUYYA,R., RANJAN, R., and CALHEIROS, R. N. (2009a). **Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities.** CoRR, abs/0907.4878.

DELIC, K. e WALKER, M. A. **Emergence of the Academic Computing Clouds.** ACM Ubiquity. 9. 2008.

FLAGSHIP NETWORKS INC. **Managed Services.** Disponível em: <http://flagshipnetworks.com/SERVICES/MANAGEDSERVICES/tabid/90/Default.aspx>. Acesso em 10 de jun. 2012.

GALLARDO, David. **Introdução à Plataforma Eclipse.** IBM. Disponível em: http://www.ibm.com/developerworks/br/library/os-eclipse-platform/ . Acesso em: 19 abr. 2012.

GOOGLE. **Bem vindo ao Google App Engine.** Disponível em: <https://appengine.google.com/start>. Acesso em 10 de Abr. 2012.

MARTIN. **What is Cloud Computing?** Disponível em: <http://www.computerservicesforlife.com/1-what-is-cloud-computing/ >. Acesso em 08 jun. 2012.

MILLER, Michael. **Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online**. Que Editor, ISBN: 0789738031, 2008.

MÜLLER, Victor Daniel. **Desenvolvimento de aplicações sob o paradigma da computação em nuvem com ferramentas Google.** Santa Catarina. UFSC, 2010.

ORLANDO, Dan. **Modelos de Serviços de Computação em Nuvem, Parte 2: Plataforma como Serviço. Enterprise RIA Consultant, Vision Media Group.** Disponível em**:** <http://www.ibm.com/developerworks/br/cloud/library/cl-cloudservices2paas/>. Acesso em 05 jun. 2012.

PEREIRA, Rodrigo; OLIVEIRA, Edson A. **Um catálogo de tecnologias e ferramentas para o desenvolvimento de sistemas em nuvem.** Departamento de Informática – UEM – Universidade Estadual de Maringá, 2012.

SOUSA, Flávio R. C.; MOREIRA, Leonardo O. ; MACHADO, Jevam C. **Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios**. UFC, 2010.

Tutoriais Admin. **Tutorial Java: O que é Java**. JavaFree. Disponível em: <http://javafree.uol.com.br/artigo/871498/Tutorial-Java-O-que-e-Java.html>. Acessado em 19 de abr. 2012.

VELTE, Anthony T; VELTE, Toby J.; ELSENPETER, Robert. **Computação em Nuvem.** Rio de Janeiro: Alta Books Editora, 2010.

WALKER, Grace. **Dentro da Nuvem Híbrida, Parte 1: Redefinir os Serviços e Métodos de Fornecimento.** Disponível em: < http://www.ibm.com/developerworks/br/cloud/library/cl-hybridcloud1/>. Acesso em 08 jun. 2012.

WEBSOFTWARE BRASIL. **Virto ERP - O que é, características, benefícios e como funciona.** Disponível em: <http://www.virto.com.br/erp-o-que-e-o-virto.aspx>. Acesso em: 10 de jun. 2012.