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

CHRISTYANO WESLEY ROMANO

COMPUTAÇÃO EM NUVEM COM FERRAMENTAS DO GOOGLE

ASSIS
2011

Av. Getúlio Vargas, 1200 – Vila Nova Santana – Assis – SP – 19807-634
Fone/Fax: (0XX18) 3302 1055 homepage: www.fema.edu.br

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

COMPUTAÇÃO EM NUVEM COM FERRAMENTAS DO GOOGLE

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

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Área de Concentração: Sistemas de Banco de Dados

ASSIS
2011

Av. Getúlio Vargas, 1200 – Vila Nova Santana – Assis – SP – 19807-634
Fone/Fax: (0XX18) 3302 1055 homepage: www.fema.edu.br

FICHA CATALOGRÁFICA

ROMANO, Christyano Wesley

Computação em Nuvem com ferramentas do Google / Christyano Wesley Romano. Fundação Educacional do Município de Assis – FEMA – Assis, 2011.

42p.

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

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

1.Computação em Nuvem. 2.Banco de Dados. 3.Novas Tecnologias. 4.Google. 5.Internet. 6.Aplicações Web

CDD:001.6

Biblioteca da FEMA.

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

COMPUTAÇÃO EM NUVEM COM FERRAMENTAS DO GOOGLE

CHRISTYANO WESLEY ROMANO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de Curso de Bacharelado em Ciência da Computação, analisado pela seguinte comissão examinadora:

.

Orientador: Dr. Alex Sandro Romeo de Souza Poletto

Analisador: Dr. Almir Rogério Camolesi

ASSIS
2011

DEDICATÓRIO

Dedico este trabalho em primeiro lugar a Deus, aos meus pais Marcos e Eliane, e uma pessoa muito especial Rafaela que me deu muita força forças nesses últimos quatro anos da minha vida, e aos amigos e professores e meu orientador que puderam compartilhar momentos de tristeza e alegria, sempre mostrando que posso ser um grande sonhador.

RESUMO

O conceito de Computação em Nuvem vem crescendo muito em grandes e pequenas empresas, e também em usuários comuns. Esse crescimento vem acontecendo pelo fato de facilitar na questão de utilizar determinados serviços de qualquer lugar e independente da plataforma, com apenas o acesso a Internet sem que as aplicações estejam instaladas em seus computadores. Como essa nova tecnologia vem crescendo, a Google disponibilizou ferramentas gratuitas de desenvolvimento de aplicações para Web, com isso os objetivos do trabalho será mostrar conceitos de Computação em Nuvem, e realizar uma pesquisa sobre as ferramentas disponibilizadas pelo Google, com o intuito de realizar um estudo de caso, desenvolvendo assim uma aplicação.

Palavras-chaves: Computação em Nuvem, Novas Tecnologias, Google, Internet, Aplicações Web**.**

ABSTRACT

The concept of Computation in Cloud comes very growing in great and small companies, and also in common users. These growths comes happening for the fact to facilitate in the question to use definitive services of any independent place and of the platform, with only the access the Internet without the applications are installed in its computers. As this new technology comes growing, the Google available gratuitous tools of development of applications for Web, with this the objectives of the work will be to show concepts of Computation in Cloud, and to carry through a research on the tools available for the Google, with intention to carry through a case study, thus developing an application.

Keywords: Computation in Cloud, New Technologies, Google, Internet, Web Applications.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS E ABREVIATURAS

TI	Tecnologia da Informação.
SDK	Software Development Kit.
JDO	Java Data Objects.
JPA	Java Persistence API.
JSP	Java Server Pages.
APP	Application.

SUMÁRIO

1. INTRODUÇÃO

Hoje em dia com o desenvolvimento da sociedade humana, somos muito dependentes de serviços de utilidades públicas como luz, água, telefone, gás dentre outros, que facilitam nossas vidas. Para a utilização desses serviços é necessário pagá-los, e a mesma ideia vem sendo utilizada na área da Informática. Com o avanço da tecnologia a ideia de vender recursos computacionais esta se tornando cada vez mais comum, hoje vem se falando muito em “Computação em Nuvem”, um tipo de serviço na área da Informática com esse mesmo conceito, pagar para ser utilizado.

Segundo Taurion (2009),

> “pode-se dizer que a Computação em nuvem é um termo para descrever um ambiente da computação baseado em uma imensa rede de servidores, sejam virtuais ou físicos. Uma definição simples pode ser então, o conjunto de recursos como capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na Internet”.

A Figura 1 ilustra uma visão geral de uma nuvem computacional.

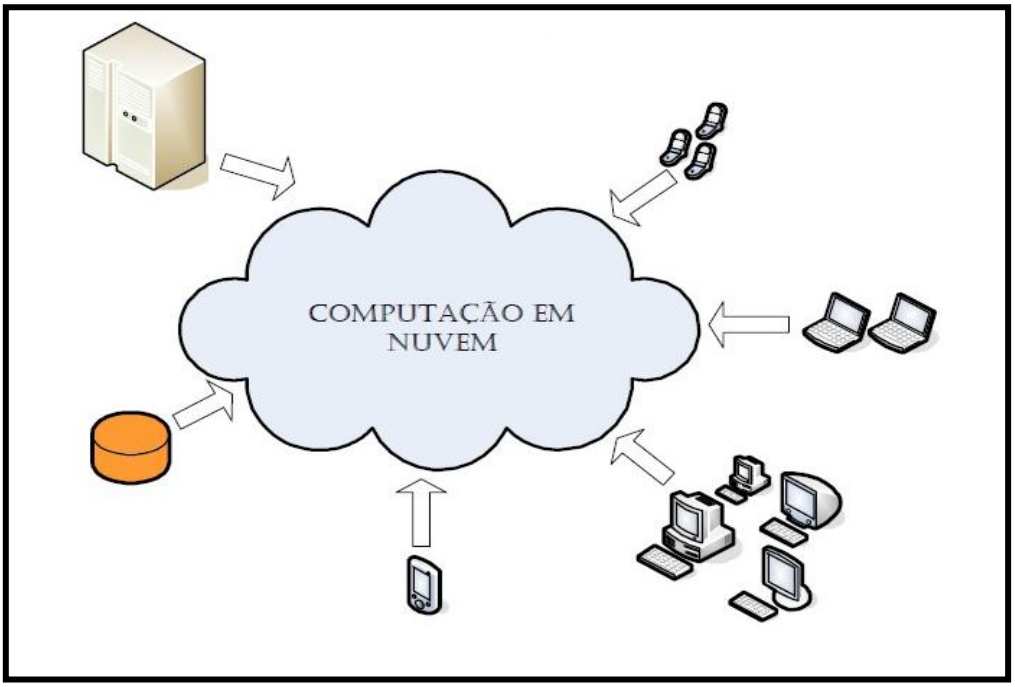

Figura 1. Visão de uma nuvem computacional (Rushel; Zanotto; Da Mota, 2010) melhorada.

Na Figura 1 são ilustrados vários dispositivos que poderão ter acesso a várias nuvens. Com esse acesso, os usuários poderão enviar seus dados e aplicações nas nuvens, e acessar de qualquer dispositivo conectado à Internet.

Outro fato importante é que os usuários de *Cloud Computing* (Computação em Nuvem) não só poderão enviar dados e aplicações, mais também utilizar aplicativos e softwares em nuvem, ou seja, não será necessário ter nada instalado em seu dispositivo, apenas o acesso à internet.

1.1 OBJETIVO

O principal objetivo do trabalho é de preparar um material, com exemplos práticos sobre Computação em Nuvem, bem como obter um maior conhecimento sobre essa nova tendência. Para tal, será feito um estudo dos conceitos gerais de Computação em Nuvem, da Google APP, bem como o desenvolvimento de uma aplicação para demonstração prática do uso dessa tecnologia.

1.2. JUSTIFICATIVA

A justificativa pela escolha deste assunto se dá pelo fato de ser uma nova tendência para a área de desenvolvimento de aplicações, que está sendo implantada em muitas empresas de grande e pequeno porte, além de oferecer uma redução nos gastos de aplicações e manutenções nas empresas.

1.3. MOTIVAÇÕES

A motivação de realizar essa pesquisa vem da necessidade de obter conhecimentos de novas tecnologias na área da Computação, além também, de ser uma tecnologia muito utilizada atualmente, como os conceitos de computação em nuvem, bem como de utilizar as ferramentas de desenvolvimento, oferecidas pelo Google.

1.4. ESTRUTURAS DO TRABALHO

Este trabalho está organizado em cinco capítulos, sendo o primeiro, esta Introdução.

No segundo capítulo, serão apresentadas as fundamentações teóricas sobre Computação em Nuvem.

No terceiro capítulo, será apresentada a ferramenta Google APP ENGINE.

No quarto capítulo, será apresentado um estudo de caso sobre a tecnologia realizando assim uma aplicação simples mostrando como funcionam as ferramentas.

No quinto capítulo a conclusão do estudo realizando uma analise da tecnologia vantagens desvantagens.

2. COMPUTAÇÃO EM NUVEM

A grande ideia do termo *Cloud Computing* ou Computação em Nuvem é mudar a forma de comercialização e desenvolvimento de softwares e aplicativos, porem isso vem causando muitas divergências por ser uma tecnologia nova no mercado.

Um dos motivos que a tecnologia gera divergência é que grandes aplicações já implantadas em empresas estão seguras, porem muitas empresas no mercado de hoje tem receio da tecnologia, em questão da segurança de dados, outro fator muito importante é pelo fato da tecnologia de computação em nuvem ser muito dependente da Internet.

A Computação na Nuvem ou *Cloud Computing* é um novo modelo de Computação que permite ao usuário final, acessar uma grande quantidade de aplicações e serviços em qualquer lugar e independe da plataforma, bastando para isso ter um terminal conectado à “nuvem”. (PEDROSA; NOGUEIRA, 2011).

O principal conceito de computação em nuvem é pagar para se utilizar, em melhores palavras são serviços e produtos de TI sobre demanda, também conhecida como *Utility Computing* (SOUSA; MOREIRA; MACHADO, 2010).

A estrutura de Computação em nuvem é dividida em três classes ou modelos de serviços, cada classe ou modelo é dependente da outra com isso formando a estrutura da nuvem.

As três classes de serviços são nomeadas da seguinte forma: Infraestrutura como Serviço (IaaS), camada inferior; Plataforma como Serviço (PaaS), camada intermediaria e Software como Serviço (SaaS), camada superior. (PEDROSA; NOGUEIRA, 2011).

As classes citadas acima fazem parte da estrutura da Computação em Nuvem, já que eles definem um padrão arquitetural para soluções de computação em nuvens.

A Figura 2 mostrará as classes ou modelos de serviços.

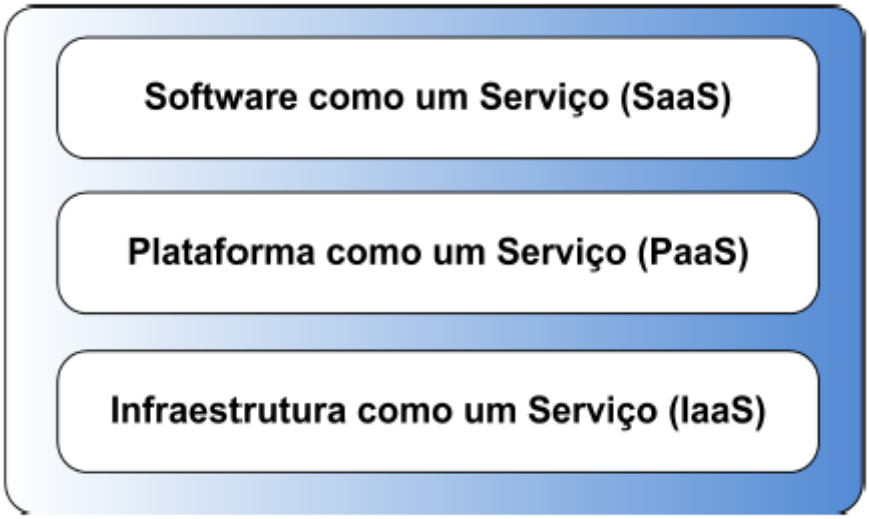

Figura 2. Modelos de serviços (Sousa; Moreira; Machado, 2010).

2.1 SaaS - SOFTWARE COMO SERVIÇO.

SaaS é a camada mais alta da arquitetura da Computação em Nuvem, é um software em forma de prestação de serviço. O software como serviço é executado em um servidor, ou seja, o usuário não precisa ter o mesmo instalado em seu computador, basta estar na Internet, para utilizar o serviço disponibilizado.

Como o SaaS são softwares prontos para serem executados, a maior vantagem é que múltiplos usuários podem utilizar o mesmo em vários locais distintos.

Em melhores palavras, o SaaS é destinado para o usuário final, que precisa do aplicativo ou aplicação pronta para usar.

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 usa-lo para beneficio exclusivo, disponibiliza o mesmo a um custo baixo a uma grande quantidade de usuários. (NOGUEIRA; PIEZZI, 2009).

Pode-se dizer, que o SaaS, representa os serviços de mais alto nível disponibilizados em uma nuvem. Esses serviços representam as aplicações completas que são oferecidas aos usuários (RUSHEL; ZANOTTO; DA MOTA, 2010).

2.2 PaaS - PLATAFORMA COMO SERVIÇO

PaaS é a camada intermediária onde o serviço de desenvolvimento de aplicações nas nuvens, assim como SaaS, também trabalha de forma de prestação de serviço.

Este conceito é associado a *Cloud Computing* e significa prover toda uma plataforma de desenvolvimento de software como um serviço. Ou seja, desenvolver, compilar, *debugar*, *deploy*, *test* em uma aplicação (PACHECO, 2011).

Plataformas de desenvolvimento como Google EngineApp, trabalha com conceitos PaaS.

2.3 IaaS (INFRAESTRUTURA COMO UM SERVIÇO)

O IaaS é a camada primaria da estrutura da nuvem e traz os serviços oferecidos na camada de infraestrutura, nestes serviços pode-se incluir servidores, roteadores, sistemas de armazenamento e outros recursos de computação. Também é responsável por prover toda a infraestrutura necessária para a SaaS e o PaaS. (RUSHEL; ZANOTTO; DA MOTA, 2010).

Segundo Sousa (2010, pg.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, pode-se aproveitar os recursos disponíveis e adicionar novos servidores virtuais à infraestrutura existente de forma dinâmica”.

Como o termo IaaS oferece serviços de infraestrutura para as camadas acima da nuvem, como PaaS e SaaS, é necessário ter uma infraestrutura IaaS em funcionamento, ou seja, as camadas PaaS e SaaS são totalmente dependentes da IaaS. Existem vários maneiras de se utilizar as camadas citadas, formando assim tipos de implantações diferentes em uma nuvem, no próximo tópico serão apresentados os tipos de implantações.

2.4 MODELOS DE IMPLEMENTAÇÃO DE UMA NUVEM

Um modelo de implementação trabalha sobre os padrões de cada classe apresentada nos capítulos anteriores, tudo irá depender da necessidade da aplicação que a nuvem vai fornecer.

A implementação da nuvem irá depender da necessidade da aplicação a ser oferecida e do tipo de contrato de prestação de serviço. Apesar da aparência dos serviços serem disponibilidades de forma pública, onde qualquer usuário tem acesso a todo o conteúdo da nuvem, os modelos de negócios tem promovido o desenvolvimento de modelos de implementação que garantem um adequado nível de controle da informação a ser disponibilizada (tipo e conteúdo) e visibilidade da nuvem. (PEDROSA; NOGUEIRA, 2011).

Hoje esses modelos são divididos em três categorias: nuvem pública, nuvem privada e nuvem híbrida.

A Figura 3 apresenta os tipos de nuvens.

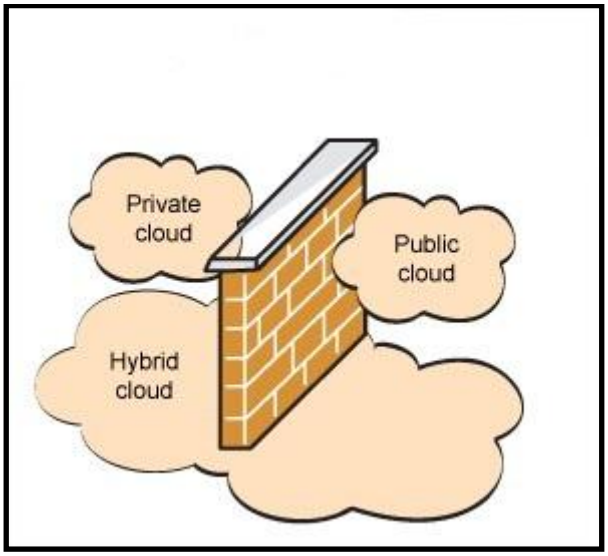

Figura 3. Tipos de Nuvens (Sousa; Moreira; Machado, 2010)

2.4.1 Nuvem Pública.

A nuvem pública pode ser utilizada por diversos usuários, basta o usuário conhecer o local de acesso da nuvem para utilizar seus recursos. Porém, os usuários que utilizam esse tipo de nuvem não têm acesso a infraestrutura do serviço.

Para este modelo de implantação as restrições de acessos não podem ser aplicadas, quando ao gerenciamento de redes, a aplicação de técnicas de autenticação e autorização também não será possível. (RUSHEL; ZANOTTO; DA MOTA, 2010).

2.4.2 Nuvem Privada.

Nuvens privadas são utilizadas apenas por um único usuário, em melhores palavras, um órgão ou empresa. Porém, o usuário que utiliza esse tipo de nuvem tem acesso total a infraestrutura, podendo assim implantar suas regras de negócio a nuvem.

Para esse modelo de implantação são empregados políticas de acesso aos serviços, tais como: Gerenciamento de Redes, Configurações dos Provedores de Serviços e a utilização de Tecnologias de Autenticações e Autorização, que são as principais características deste modelo (RUSHEL; ZANOTTO; DA MOTA, 2010).

2.4.3 Nuvem Hibrida.

A nuvem híbrida apresenta conceitos de nuvem privada e nuvem publica, podendo assim ter acesso à Infraestrutura ou não, disponibilizando serviços a vários usuários ou apenas um.

Nuvens híbridas é uma combinação de nuvens públicas e privadas. Essas nuvens seriam geralmente criadas pela empresa e as responsabilidades de gerenciamento seriam divididas entre a empresa e o provedor de nuvem pública. A nuvem híbrida usa serviços que estão no espaço público e no privado. (AMRHEIN; QUINT, 2009).

2.5 Papeis da Computação em Nuvem

Com os serviços citados, os papeis são importantes para definir onde cada tipo de usuário pode utilizar os modelos de serviços.

Para entender melhor a computação em nuvem, é preciso classificar os atores dos modelos de acordo com os papéis desempenhados (SOUSA; MOREIRA; MACHADO, 2010).

A Figura 4 destaca estes papéis.

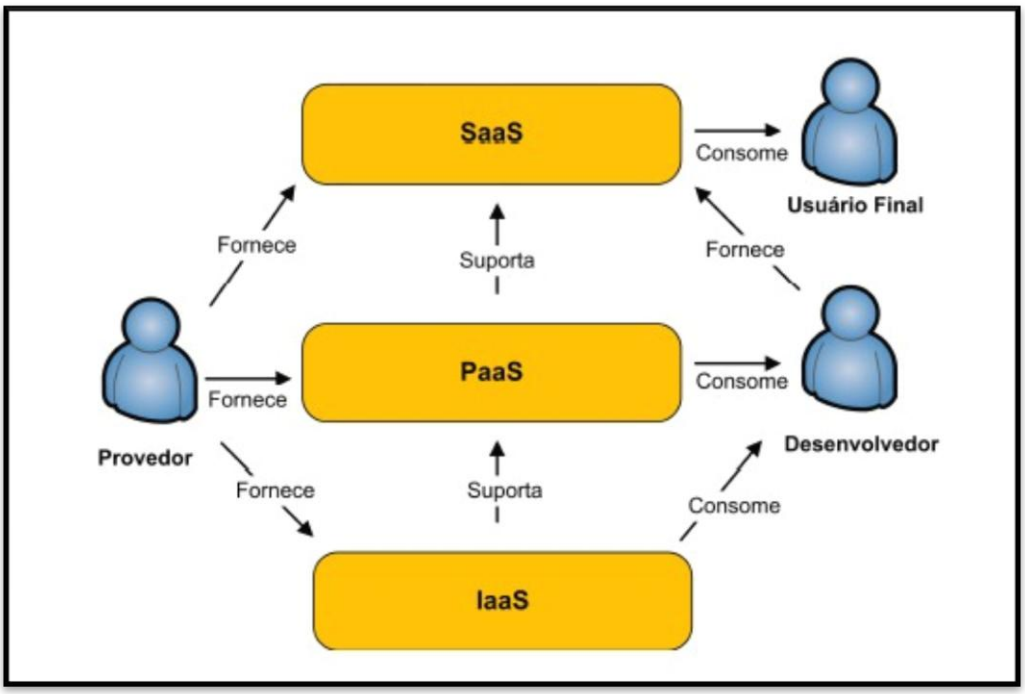

Figura 4. Papéis de Computação em Nuvem (Sousa; Moreira; Machado, 2010).

O provedor é responsável por disponibilizar, gerenciar e monitorar toda a estrutura para a solução de computação em nuvem, deixando o desenvolvedor e o usuário final sem esse tipo de responsabilidade e fornecendo serviços nos três modelos de serviços. Os desenvolvedores utilizam os recursos fornecidos e disponibilizam serviços para os usuários finais. (SOUSA; MOREIRA; MACHADO, 2010).

Os atores podem assumir o papel de desenvolvedor ou usuário final, ocupando a camada intermediária ou a alta da nuvem; já o provedor, irá fornecer os serviços para todos os níveis de camadas da nuvem, tanto de infraestrutura quanto de desenvolvimentos, ou aplicações, para opção e interesse de cada ator, ou usuário final ou desenvolvedor, como apresentado na Figura 4.

A próxima seção destaca a infraestrutura dos servidores da Google (Google App), com o intuito de realizar um estudo de caso com as ferramentas disponíveis para o desenvolvimento. Foram escolhidas as ferramentas do Google pela facilidade que a ferramenta trabalha com a linguagem Java, e por Google ser conhecida mundialmente.

3. GOOGLE APPLICATION ENGINE

Existem muitas ferramentas de desenvolvimento como o Google Application Engine. A empresa Amazon fornece o mesmo tipo de serviço de desenvolvimento que o Google dentre ela existem outras como o Microsoft Azure, um serviço totalmente voltado a parte de banco de dados *PL\SQL*, todas utilizando conceitos de nuvem.

O Google Application Engine é um conjunto de ferramentas para desenvolvimento e serviços disponibilizados pela Google.

Trata-se de um modelo de PaaS, que diferentemente de seu conceito original, em que todo o ambiente responsável pelas etapas de desenvolvimento e publicação do software se dá através de ferramentas disponibilizadas via Web, disponibiliza um ambiente desktop completo e de fácil configuração para esta finalidade (MÜLLER, 2010).

Com a ferramenta, é possível criar aplicações e enviá-las para a nuvem da Google, com isso utilizando também a plataforma de infraestrutura, ou seja, o desenvolvedor não precisa ter servidor para desenvolver seus aplicativos, a Google fornecerá. Com o envio das aplicações, o usuário final pode executá-las.

Para este propósito o Google Application Engine possui suporte a aplicativos criados com o uso das linguagens de programação Python e Java, e através desta última, várias baseadas na máquina virtual Java (MÜLLER, 2010).

Como o Google Application Engine trabalha no conceito de computação em nuvem tem um custo para o armazenamento, esse custo é muito acessível.

Para se utilizar gratuitamente a Google Application Engine basta não ultrapassar seu limite de armazenamento. Todos os aplicativos podem usar até 500 MB de armazenamento e CPU e largura de banda suficiente para suportar um aplicativo eficiente que oferece cerca de cinco milhões de visualizações de página por mês, totalmente grátis. Ao ativar o faturamento para o seu aplicativo, os limites gratuitos aumentam e você paga somente pelos recursos que ultrapassam os níveis gratuitos (GOOGLE, 2010).

Na próxima seção será mostrado o funcionamento da Application Engine com Java.

3.1. GOOGLE APP ENGINE E JAVA

Com o avanço da Internet, hoje em dia é muito comum o desenvolvimento de aplicativos via Web. Esses aplicativos são desenvolvidos em várias linguagens, como, JAVA, PHP, Asp.net dentre outras. Porém, todas utilizam o mesmo conceito de programação em Web, ou seja, todas possuem um servidor de aplicações onde nele terá todas as informações do aplicativo desejado. Com o GOOGLE APP ENIGNE não é muito diferente disso, por isso foi disponibilizado para trabalhar com a linguagem Java.

Com a popularidade da linguagem JAVA, e do compilador Eclipse, por sua ampla IDE de desenvolvimento, o Google disponibilizou ferramentas para o desenvolvimento de aplicações Web, com isso é possível criar aplicativos Web utilizando as tecnologias Java padrão e executá-los nas infraestruturas do Google.

Segundo Muller (2010, p. 81),

> "O ambiente Java do App Engine executa em uma JVM (Máquina Virtual Java) da versão 6 do Java, com suporte a servlets, biblioteca Java padrão, armazenamento de dados e serviço do App Engine. Apesar de executar os programas usando a versão 6 do Java, pode-se usar classes compiladas utilizando qualquer versão anterior. O suporte as bibliotecas padrões facilita o desenvolvimento de aplicações já que não requer grandes mudanças no desenvolvimento para o App Engine em relação aos servidores usuais."

A ferramenta Application Engine, também oferece para o compilador Eclipse uma ferramenta de total integração com sua IDE através de um *plugin*, contendo sua SDK (Software Development Kit), ou seja, um kit de desenvolvimento para aplicação com o nome de Google Application Engine. Com isso traz as ferramentas necessárias para o desenvolvimento.

3.1.1. Sistema de Armazenamento

Como Java utiliza padrões de armazenamento e persistência de dados, o Google implementou, através de um software livre, padrões para o armazenamento de dados.

Os serviços de armazenamento de dados suportado pela AppEngine são o JDO (Java Data Objects) ou JPA (Java Persistence API).

O padrão JDO é o mais comum para as aplicações desenvolvidas pelo AppEngine, pelo fato da implementação do seu padrão que é conhecida como *Data Nucleus*.

Estes padrões para o armazenamento de dados são implementados pelo Application Engine com o uso do Data Nucleus Access Platform, a implementação de software livre escolhida pelo AppEngine para dar suporte a estes padrões (MÜLLER, 2010).

3.1.2. JDO (Java Data Objects)

JDO (Java Data Objects) é uma interface padrão para armazenar objetos que contem dados em um banco de dados, ou seja, o JDO é responsável por gravar os dados da aplicação.

Java Data Objects é uma forma padrão de acesso a dados persistente em banco de dados. (JDO, 2011).

Como foi citado anteriormente, foi desenvolvido o *Data Nucleus* quando é criada uma aplicação com Application Engine o *Data Nucleus* é responsável pelo armazenamento dos dados da aplicação.

3.1.3. Sandbox.

Como a ferramenta trabalha sobre as camadas de computação em nuvem, onde os processamentos e o armazenamento são distribuídos em vários servidores, a Google fornece um ambiente virtual seguro, chamado de “sandbox”. Em melhores palavras, como os aplicativos contem dados, é necessário ter segurança na aplicação. A Sandbox é responsável pela mesma, ou seja, quando a aplicação é enviada para a nuvem, a mesma é distribuída em vários servidores, quando um usuário acessa a aplicação a Sandbox entra em ação, responsável por criar um ambiente virtual onde armazena toda aplicação e durante sua execução.

A Sandbox tem a finalidade de garantir que um aplicativo não interfira na execução de outro, além de, por se tratar de um ambiente distribuído, servir como uma forma de virtualização de um sistema operacional (MÜLLER, 2010).

4. ESTUDO DE CASO

O intuito do trabalho é pesquisar sobre as novas ferramentas de computação em nuvem, disponibilizadas pelo Google para desenvolver sobre os padrões Java.

Com isso a infraestrutura da aplicação será de toda responsabilidade do Google, ou seja, quando a aplicação é enviada para os servidores do Google que esta nas nuvens a aplicação não ira mais depender de um Servidor local para ser executada, para usá-la basta estar na Internet.

A aplicação será um estudo de caso sobre as ferramentas citadas nas seções anteriores. Utilizando as ferramentas do Google junto ao compilador *Eclipse*.

No próximo tópico será apresentada uma aplicação onde, mostrara os códigos desenvolvidos focando na parte de persistência dos dados e das configurações necessárias para o armazenamento nos servidores, com isso elaborando o estudo de caso da nova tecnologia.

4.1. Aplicação

A aplicação que será desenvolvida para o estudo de caso, apesar de simples, demonstra o uso dessa nova tecnologia. É uma aplicação em que o usuário poderá ou não fazer *login* e postar uma mensagem de saudação, com o objetivo de apresentar a ferramenta Application Engine SDK, disponibilizada pela Google para desenvolver em Java com o compilador Eclipse.

A ferramenta não vem instalada na versão do Eclipse que é utilizado, porém é preciso instalar a mesma. A ferramenta pode ser instalada utilizando o recurso de Software Update (Atualização de Software), que esta na opção Help, do Eclipse. A Figura 5 mostra o caminho para a atualização.

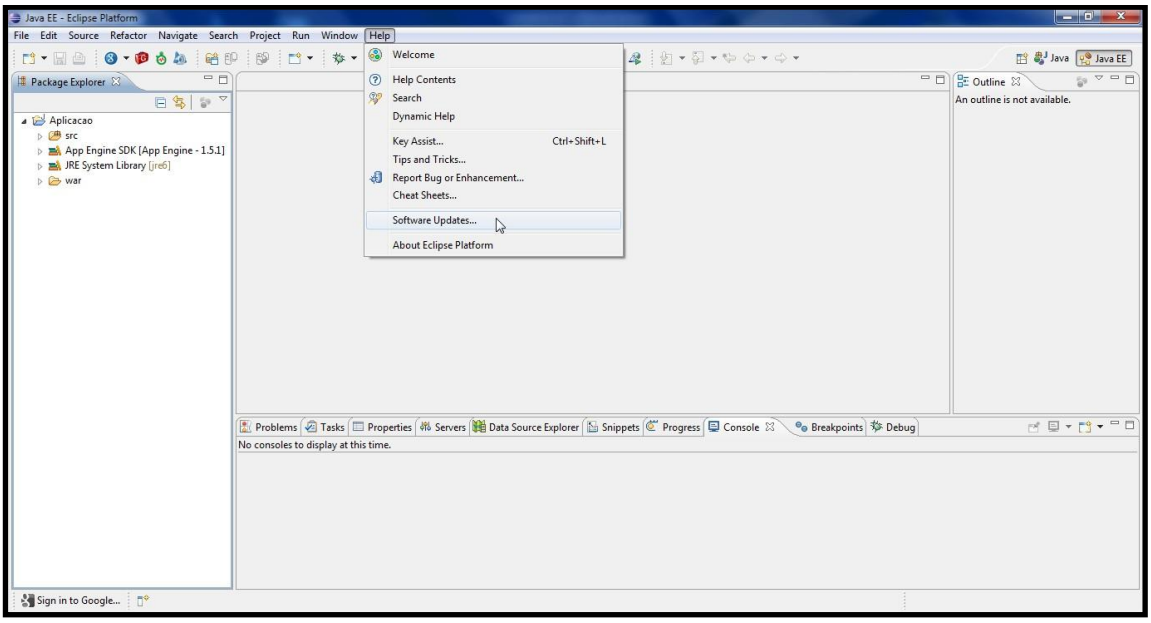

Figura 5. Instalação da ferramenta SDK.

Depois basta adicionar a url http://dl.google.com/eclipse/plugin/3.4 para o *download* automático da instalação. Quando terminar a instalação repare que na barra superior do Eclipse é adicionado o ícone  onde é possível criar aplicação utilizando as ferramentas para desenvolver.

4.2. Criando o Projeto.

Como toda a aplicação desenvolvida em Java, quando é criado um projeto utilizando o *plugin* do Google, cria-se um diretório, para a aplicação a ser desenvolvida nome do projeto será *Aplicacao/*. Quando é criado o projeto automaticamente é criado subdiretórios, como o *src/* que contem códigos fontes em Java e o *war/* que contem aplicativos que serão compilados. Alem da criação dos subdiretórios são automaticamente importadas todas as configurações necessárias da Application Engine SDK. Na Figura 6 é apresentado o projeto criado com os seguintes diretórios citados.

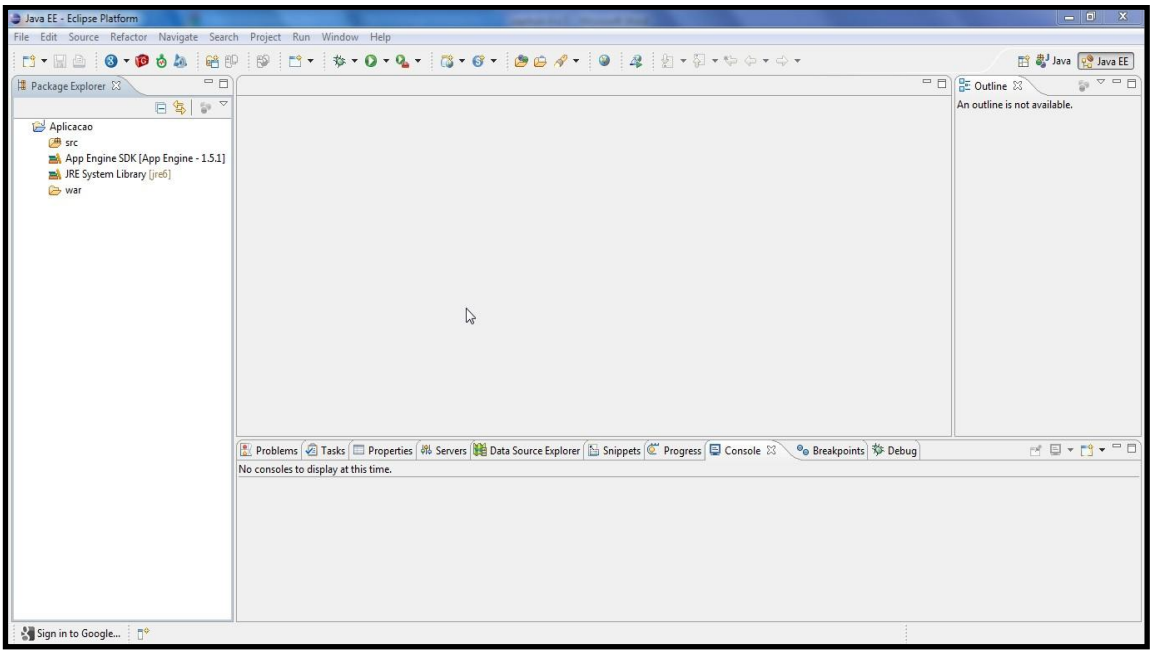

Figura 6. Criação do Projeto Aplicação.

4.3. Trabalhando com Servlet.

Como a aplicação segue os padrões de programação em Java, trabalha com *servlet*. Quando é criada a aplicação dentro do diretório src/ é criado automaticamente um *servlet* que é responsável por enviar respostas e requisições para o servidor, com isso interagindo com o servidor.

Outro fato importante é a manipulação do *servlet* que é feito no diretório WEB-INF, que também foi criado automaticamente junto a aplicação, nele se encontra o arquivo web.xml responsável por mapear *servlets* no servidor.

Será utilizado no *servlet* o serviço de *Usuário* fornecido pela ferramenta do Google, uma classe que permite o usuário utilizar contas do Google para operar no aplicativo. Como mostra a Figura 7.

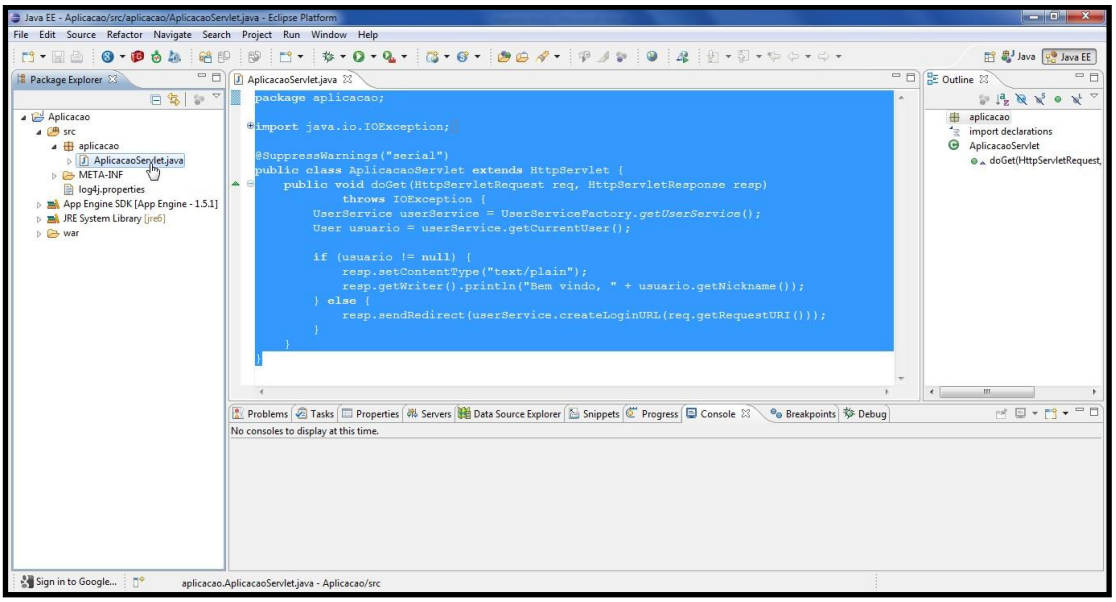

Figura 7. Servlet Usuário.

O serviço de usuário é importado para o *servlet* podendo assim utilizar seus recursos. Como mostrara no código a seguir.

```
package aplicacao;

import java.io.IOException;

import javax.servlet.http.*;

import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

@SuppressWarnings("serial")
public class AplicacaoServlet extends HttpServlet {
        public void doGet(HttpServletRequest req, HttpServletResponse resp)
                        throws IOException {
                UserService userService = UserServiceFactory.getUserService();
      User usuario = userService.getCurrentUser();

      if (usuario != null) {
         resp.setContentType("text/plain");
         resp.getWriter().println("Bem vindo, " + usuario.getNickname());
      } else {
         resp.sendRedirect(userService.createLoginURL(req.getRequestURI()));
```

```
    }
      }
}
```

A finalidade do *servlet* é criar um usuário, se o usuário for diferente de nulo, ele armazena o nome que será digitado na variável usuário e manda para o servidor, e se ele for nulo, será redirecionado para URL informada, no caso a do Google.

4.4. Armazenando Dados Com JDO.

O principal objetivo do uso do JDO é recuperar os dados e armazená-los na aplicação, ou seja, funcionar como um banco de dados, com isso o Data Nucleus fornece o uso da JDO para o armazenamento de dados nas aplicações desenvolvidas para Application Engine do Google. Quando o projeto é criado automaticamente o arquivo de configuração do armazenamento dos dados é criado, chamado de jdoconfig.xml encontrado dentro do diretório WEB-INF. O código abaixo mostrará o conteúdo do arquivo jdoconfig.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<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.DatastoreJDOPersistenceManagerFact
ory"/>
    <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"/>
  </persistence-manager-factory>
</jdoconfig>
```

Com o arquivo de configuração criado, pode ser criada uma classe na aplicação para fazer o armazenamento de dados. A classe que será apresentada a seguir tem o nome de Saudacao.jar, uma classe em Java que importara toda a persistência de dados utilizando JDO, para armazenar os objetos da classe em dados.

```
package aplicacao;
import java.util.Date;
import javax.jdo.annotations.IdGeneratorStrategy;
import javax.jdo.annotations.IdentityType;
import javax.jdo.annotations.PersistenceCapable;
import javax.jdo.annotations.Persistent;
import javax.jdo.annotations.PrimaryKey;
import com.google.appengine.api.users.User;
@PersistenceCapable(identityType = IdentityType.APPLICATION)
public class Saudacao {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    private Long codigo;
    @Persistent
    private User autor;
    @Persistent
    private String conteudo;
    @Persistent
    private Date data;
    public Saudacao(User autor, String conteudo, Date data) {
        this.autor = autor;
        this.conteudo = conteudo;

        this.data = data;

    }
        public Long getCodigo() {
                return codigo;
        }
        public void setCodigo(Long codigo) {
                this.codigo = codigo;
        }
        public User getAutor() {
                return autor;
        }
        public void setAutor(User autor) {
                this.autor = autor;
        }
        public String getConteudo() {
```

```
            return conteudo;
        }
        public void setConteudo(String conteudo) {
            this.conteudo = conteudo;
        }
        public Date getData() {
            return data;
        }
        public void setData(Date data) {
            this.data = data;
        }
}
```

Na classe Saudação tem três variáveis: o autor, o conteúdo e a data. Nas três variáveis tem a notação @Persistent, que serve para informar para o DataNucleus que as variáveis serão gravadas como dados da aplicação.

Outro fator importante no armazenamento dos dados é a manipulação dos objetos que serão gravados, para isso é criado uma classe, chamada de Persistence Manager Factory (*PMF*), ou seja, a cada vez que a aplicação transformar os objetos em dados é requisitado Manager Factory (*MF*). Com isso, a criação da classe facilita de não ter mais que ficar requisitando MF, ou seja, é construído um método na classe PMF responsável por toda vez que precisar de uma manipulação será utilizado sem precisar requisitar a cada persistência feita. A implementação da classe PMF é:

```
package aplicacao;
import javax.jdo.JDOHelper;
import javax.jdo.PersistenceManagerFactory;
public final class PMF {
    private static final PersistenceManagerFactory pmfInstance =
        JDOHelper.getPersistenceManagerFactory("transactions-optional");
    private PMF() {}
    public static PersistenceManagerFactory get() {
```

```
        return pmfInstance;
    }
}
```

Com o acesso a dados e a Saudacao criada, é criado um novo *servlet* para escrever as mensagens postadas. Esse *servlet* terá o nome de EscreverAplicacaoServlet e seu objetivo é responder e requisitar os dados no servidor, pegando o usuário que vai interagir com o sistema, e gravar os dados na aplicação. Segue o seguinte código do novo *servlet*.

```
package aplicacao;

import java.io.IOException;

import java.util.Date;
import java.util.logging.Logger;
import javax.jdo.PersistenceManager;
import javax.servlet.http.*;
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;
import aplicacao.Saudacao;
import aplicacao.PMF;
public class EscreverAplicacaoServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        private static final Logger log =
 Logger.getLogger(EscreverAplicacaoServlet.class.getName());
        public void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws IOException {
      UserService userService = UserServiceFactory.getUserService();
      User user = userService.getCurrentUser();
      String conteudo = req.getParameter("conteudo");
```

```
        Date data = new Date();
        Saudacao saudacao = new Saudacao(user, conteudo, data);
        PersistenceManager pm = PMF.get().getPersistenceManager();
        try {
            pm.makePersistent(saudacao);
        } finally {
            pm.close();
        }
        resp.sendRedirect("aplicacao.jsp");

    }
}
```

A última resposta (resp.sendRedirect("aplicacao.jsp");) tem como objetivo pegar toda informação que contem no *servlet* e jogar na classe .jsp mostrando assim ao usuário as mensagens postadas, a classe .jsp será apresentada no seguinte tópico.

4.5. Criando a página em Java Server Pages

Para a interface da aplicação será criada uma página .Jsp (Java Server Pages), a Application Engine do Google suporta os recursos. Com isso todo o recurso visual da página será implementado na aplicação.jsp, que contem o seguinte código:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.util.List" %>
<%@ page import="javax.jdo.PersistenceManager" %>
<%@ page import="com.google.appengine.api.users.User" %>
<%@ page import="com.google.appengine.api.users.UserService" %>
<%@ page import="com.google.appengine.api.users.UserServiceFactory" %>
<%@ page import="aplicacao.Saudacao" %>
<%@ page import="aplicacao.PMF" %>

<%@page import="aplicacao.Saudacao"%>
<html>
        <head>
   <link type="text/css" rel="estilosPagina/" href="/estilosPagina/estiloPrincipal.css"
```

```
/>
        </head>

  <body>
<%
   UserService userService = UserServiceFactory.getUserService();
   User user = userService.getCurrentUser();
   if (user != null) {
%>
<p>Olá, <%= user.getNickname() %>! (Você quer

<a href="<%= userService.createLogoutURL(request.getRequestURI())
%>">Deslogar</a>.)</p>

<%
   } else {
%>
<p>Olá!
<a href="<%= userService.createLoginURL(request.getRequestURI())
%>">Logue</a>

para incluir saudações com seu nome.</p>
<%
   }
%>
<%
   PersistenceManager pm = PMF.get().getPersistenceManager();
String query = "select from " + Saudacao.class.getName() + " order by data desc
range 0,5";
   List<Saudacao> saudacoes = (List<Saudacao>) pm.newQuery(query).execute();
   if (saudacoes.isEmpty()) {
%>
<p>O livro não possui saudações.</p>
<%
   } else {
      for (Saudacao s : saudacoes) {
         if (s.getAutor() == null) {

%>
<p>Pessoa anonima escreveu: </p>
<%
         } else {
%>
<p><b><%= s.getAutor().getNickname() %></b> escreveu: </p>
<%
         }
%>
```

```
<blockquote><%= s.getConteudo() %></blockquote>
<%

      }

   }

   pm.close();

%>

   <form action="/sign" method="post">
     <div><textarea name="conteudo" rows="3" cols="60"></textarea></div>
     <div><input type="submit" value="Postar Saudação" /></div>
   </form>
         <input type="submit" value="Pesquisar">
  </body>
</html>
```

Na página .jsp pode importar o serviço de usuário que foi utilizado no *servlet*, para trabalhar com os dados do usuário do *servlet*. Mas para a página trabalhar com o *servlet* é necessário mapear a mesma, no arquivo web.xml dentro do diretório WEB-INF, adicionando o seguinte código.

```
<welcome-file-list>
      <welcome-file>aplicacao.jsp</welcome-file>
</welcome-file-list>
```

O objetivo do mapeamento é definir que a página criada .jsp será a página principal, ou seja, quando o aplicativo entrar em execução ela será exibida para o usuário.

Na página principal é utilizado todos os recursos já implementado antes, como o serviços de usuário, para definir o usuário conectado ou não, o PMF que é responsável para manipular os dados e consultas no banco, e a classe Saudação utilizando assim os objetos como o autor e conteúdo, todas essas informações serão requisitadas na página aplicação.jsp.

Toda a aplicação é executada no servidor local, ou seja, a aplicação está apenas funcionando na máquina local não tem nada nas nuvens ainda, no próximo tópico será apresentado como é feito o envio da mesma.

4.6. Enviando o Aplicativo.

Para enviar o aplicativo para as nuvens é necessário utilizar uma conta do Google, com isso poderá enviar o mesmo utilizando o botão de *deploy*  encontrado na barra superior do Eclipse, com isso enviando toda aplicação para os servidores do google. Outro arquivo de configuração que é criado junto com o projeto é o appengine-web.xml responsável pelo nome da aplicação nos servidores do Google e contem o seguinte código:

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
	<application></application>
	<version>1</version>

	<!-- Configure java.util.logging -->
	<system-properties>
		<property name="java.util.logging.config.file" value="WEB-
INF/logging.properties"/>
	</system-properties>

</appengine-web-app>
```

Nas tags de <application> é definido o nome da aplicação na Internet, mas conhecido como ID no Google a cada conta conectada para aplicações podem ser utilizados até 10 ID's para desenvolvimento. Criando o nome é definido o domínio que é padrão, http://*application-id*.appspot.com/. No lugar de application-id, será o nome da aplicação criada. Com isso a aplicação é encerrada determinando o fim do estudo de caso.

4.7. Dificuldade Encontrada.

Como a aplicação ainda esta sendo executada no servidor local, é necessário enviá-la para os servidores da Google, essa foi a dificuldade encontrada na aplicação.
Para enviar basta esta autenticada no compilador eclipse que esteja com o email do desenvolvedor ativado, como mostra na figura 8.

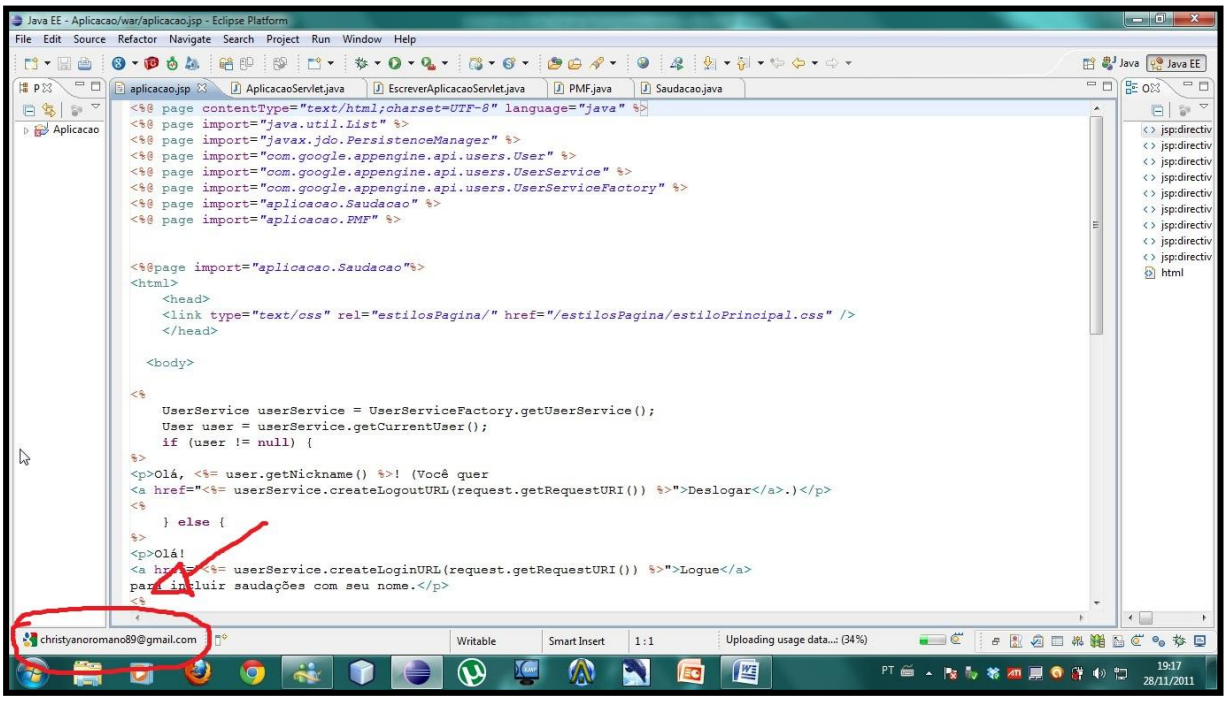

Figura 8. Desenvolvedor autenticado no Google dentro do Eclipse.

Depois de estar autenticado, basta clicar no ícone  , para enviar o aplicativo para os servidores do Google, com isso ele solicitara que informe um numero de telefone celular para enviar um código de segurança, porem o código não foi enviado, sendo assim a dificuldade encontrada no estudo.

CONCLUSÃO

Por ser uma tecnologia considerada nova, a Computação em Nuvem ainda é uma questão muito polêmica. Isso se deve pelo fato de as empresas não utilizarem por insegurança. Muitos empresários de TI acham que por não fazer parte da infraestrutura da sua empresa, ou seja, utilizar servidores não conhecidos pode ocorrer falhas de segurança. Outro fato é que é muito dependente da Internet gerando assim gargalo de usuários deixando lento o acesso a dados, porém, poderá ser muito utilizada, em grandes e pequenas empresas, pelo fato de diminuir os custos das empresas, principalmente na Infraestrutura que com o uso da nova tecnologia não é necessário ter gastos.

A aplicação realizada no trabalho é um bom exemplo de baixo custo apesar de ser apenas um estudo de caso, mas quando é enviada para os servidores da Google não é necessário ter a aplicação na máquina basta apenas acessa-la via internet, por intermédio de uma conta do Google. A tecnologia será representada na Figura 8.

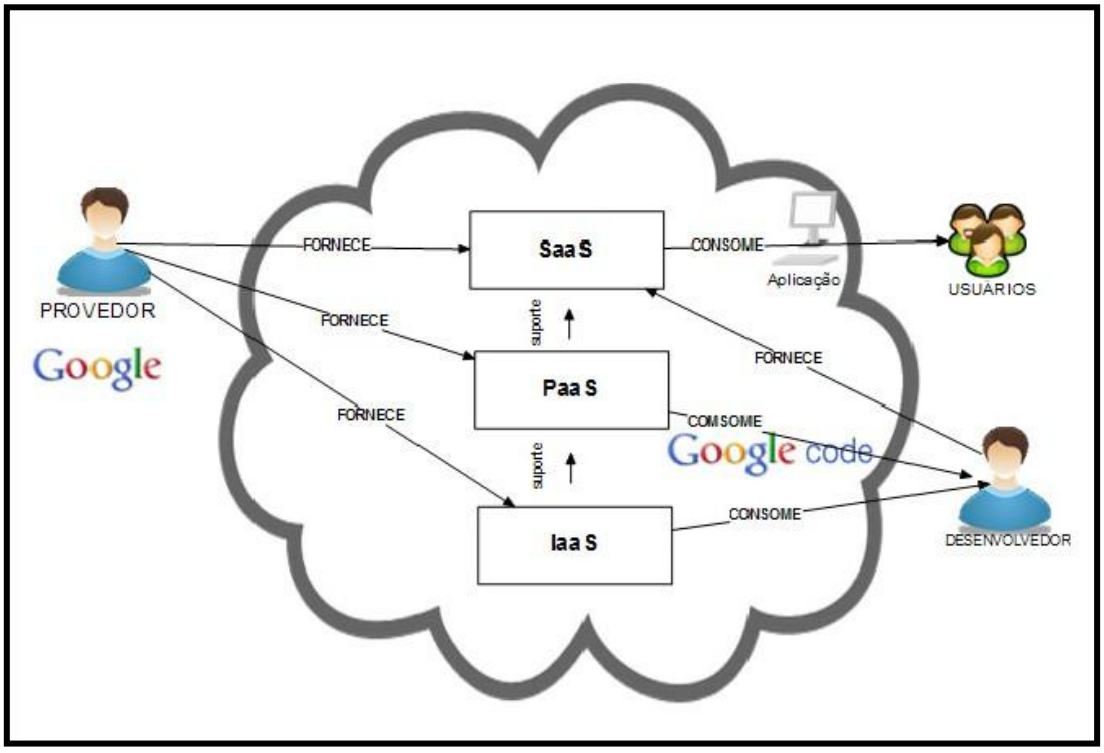

Figura 9. Representação de um projeto em nuvem.

O Google é o provedor que fornece toda a estrutura para o projeto, cada camada é fundamental para o resultado final da aplicação, ou seja, a camada 1 (IaaS), fornece a infraestrutura são os servidores onde esta a aplicação, ou seja, o usuário não precisara se preocupar com a parte física, a camada 2 (PaaS), é de exclusividade do desenvolvedor, ela fornecera toda a ferramenta para o desenvolvimento da aplicação e a camada 3 é a aplicação já enviada para nuvem, para o usuário utilizá-la.

Hoje em dia é fácil afirmar que computação em nuvem cada vez mais ira crescer, claro que muitas empresas adotam outros tipos de serviço por serem seguros, mas é um campo muito amplo em TI que ira facilitar cada vez mais não só empresas como usuários também.

REFERENCIAS

AMRHEIN, Dustin; QUINT, Scott; Computação em Nuvem para Empresa: Parte 1: Capturando a Nuvem. IBM, 2009.

GOOGLE. **Google AppEngine.** Disponível em: http://code.google.com/intl/pt-BR/appengine/. Acessado em: Junho de 2011.

JDO. Java Data Objects. Disponivel em: http://db.apache.org/jdo/ . Acessado em: Outubro de 2011.

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

NOGUEIRA, Matheus Cadori.; PEZZI, Daniel da Cunha. **A computação agora é nas Nuvens.** Cruz Alta, Rio Grande do Sul. UNICRUZ, 2009.

PACHECO, Diego. **PaaS, CloudComputing, Virtualização eo Futuro.** Parte 02. Disponível em: http://imasters.com.br/artigo/14228/cloud/paas_cloud_computing_virtualizacao_e_o_futuro_parte_02/. Acessado em: Junho de 2011.

PEDROSA, Paulo H. C.; NOGUEIRA, Thiago. **Computação em Nuvem**. Campinas, São Paulo. UNICAMP, 2011.

RUSHEL, Henrique.;ZANOTTO, Mariane Suzan.; DA MOTA, WeltonCosta. **Computação em Nuvem**. Paraná. PUC, 2010.

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

TAURION, Cezar. **CloudComputing: computação em nuvem: transformando o mundo da tecnologia da informação**. Rio de Janeiro. Brasport, 2009.