CENTRO PAULA SOUZA

Faculdade de Tecnologia de Americana

Curso de Bacharelado em Análise de Sistemas e Tecnologia da Informação

AMANDA CAROLINE DIANNI

DESENVOLVIMENTO DE APLICAÇÕES WEB UTILIZANDO O GOOGLE APP ENGINE

Americana, SP

2014

CENTRO PAULA SOUZA

Faculdade de Tecnologia de Americana

Curso de Bacharelado em Análise de Sistemas e Tecnologia da Informação

AMANDA CAROLINE DIANNI

DESENVOLVIMENTO DE APLICAÇÕES WEB UTILIZANDO O GOOGLE APP ENGINE

Projeto desenvolvido em cumprimento curricular da disciplina Projeto de Graduação do Curso de Bacharelado em Análise de Sistemas e Tecnologia da Informação da FATEC – Americana, sob orientação da Prof. Me. Maria Elizete Luz Saes.

Área: Desenvolvimento

Americana, SP

2014

Dianni, Amanda Caroline

D529d Desenvolvimento de aplicações *Web* utilizando o *Google App Engine.* / Amanda Caroline Dianni. – Americana: 2014.

61f.

Monografia (Bacharelado em Análise de Sistemas e Tecnologia da Informação). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.

Orientador: Prof. Me. Maria Elizete Luz Saés

1. Computação em nuvem 2. Desenvolvimento de software I. Saés, Maria Elizete Luz II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.518

681.3.05

Amanda Caroline Dianni

DESENVOLVIMENTO DE APLICAÇÕES WEB UTILIZANDO O GOOGLE APP ENGINE

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Bacharel em Análise e Desenvolvimento de Sistemas e Tecnologia da Informação pelo CEETEPS/Faculdade de Tecnologia – Fatec/ Americana.
Área de concentração: Desenvolvimento

Americana, 01 de Dezembro de 2014.

Banca Examinadora:

Prof°. Me. Elizete Luz Saes
Professor Mestre
Faculdade de Tecnologia de Americana - FATEC

Prof°. Me. Luciene Maria Garbuio Castello Branco
Professor Mestre
Faculdade de Tecnologia de Americana - FATEC

Prof°. Me. Maria Cristina Luz Fraga Moreira Aranha
Professor Mestre
Faculdade de Tecnologia de Americana - FATEC

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus pelas infinitas bênçãos que derramou sobre minha vida durante toda a jornada acadêmica.

Ao meu marido, ao carinho e amor que sempre demonstrou.

A minha família pelo incentivo, o auxílio, a ajuda e por estarem sempre ao meu lado.

A Faculdade de Tecnologia de Americana, pelo ensino de extrema qualidade, pelo profissionalismo e incentivo ao estudo.

A todos os excelentes professores que tive ao longo do curso e suas aulas que tanto somaram ao meu crescimento pessoal e profissional.

A todos meus colegas e amigos de faculdade, pela ajuda e companheirismo em todo o curso.

DEDICATÓRIA

Ao meu amado marido.

RESUMO

O presente trabalho conceitua a plataforma Google App Engine, e apresenta detalhes sobre seu funcionamento e a maneira como a plataforma trabalha, bem como suas funcionalidades, destacando seus aspectos positivos e negativos. É realizada uma breve análise dos serviços semelhantes ao App Engine, oferecidos pelas principais empresas concorrentes. Conceitua, também de forma breve, os outros serviços oferecidos pela plataforma do Google na nuvem, que podem ser integrados com o Google App Engine, para expandir suas funcionalidades. O trabalho apresenta, de forma sucinta, o desenvolvimento de uma aplicação simples, para demonstrar a viabilidade da utilização da plataforma, utilizando alguns dos recursos disponibilizados por ela.

Palavras Chave: Plataforma de Aplicações; Google App Engine; Computação em Nuvem; Desenvolvimento de Aplicações; Estudos de Viabilidade.

ABSTRACT

The present work conceptualizes the Google App Engine platform and presents details about its operation and the way that the platform works, as its functionalities, highlighting the positive and negative aspects. It also brings a brief analysis of the services that are similar to the App Engine services offered by the main competitors. This study also conceptualizes, in brief, the other services provided by Google cloud platform, which may be integrated to Google App Engine, to expand its features. This work also shows, concisely, the development of a simple application, in order to demonstrate the practicability of this platform, by using some of the resources provided by it.

Keywords: *Application Platform; Google App Engine; Cloud Computing; Development of Applications; Practicability*.

SUMÁRIO

ÍNDICE DE IMAGENS

LISTA DE SIGLAS

IaaS - Infraestrutura como Serviço

PaaS - Plataforma como Serviço

SaaS - *Software* como Serviço

GAE - *Google App Engine*

HTTP - Protocolo de Transferência de Hipertexto

URL - Localizador Padrão de Recursos

CPU - Unidade Central de Processamento

CSS - Folhas de Estilo em Cascata

HTML - Linguagem de Marcação de Hipertexto

XML - Linguagem de Marcação Extensível

SDK - Kit de Desenvolvimento de *Software*

API - Interface de Programação de Aplicações

XMPP - Protocolo Extensível de Mensagem e Presença

WAR - Arquivo de Aplicação Web

JDO - Objeto de Dados Java

JPA - API de Persistência Java

IDE - Ambiente Integrado de Desenvolvimento

JVM - Máquina Virtual Java

ID - Número de Identificação

NoSQL - Banco de Dados Não-Relacional

SQL - Linguagem de Consulta Estruturada

1. INTRODUÇÃO

A computação em nuvens é uma tecnologia que vem ganhando cada vez mais espaço no mercado de desenvolvimento e gerenciamento de aplicações *web*. Considerando este crescimento, as grandes empresas de serviços *online*, como Google, IBM, Microsoft e Amazon, criaram serviços específicos para que os desenvolvedores pudessem elaborar e hospedar suas aplicações.

Dessa forma, surge o Google App Engine, que é uma plataforma do Google na nuvem, utilizada tanto para o desenvolvimento como para a hospedagem das aplicações.

O objetivo geral deste trabalho é conceituar o Google App Engine, apresentando ao leitor os aspectos positivos e negativos da plataforma, de forma a auxiliá-lo a entender se a escolha do App Engine é a mais indicada para o desenvolvimento de suas aplicações.

O presente trabalho apresenta também os detalhes do funcionamento da plataforma, destacando os serviços oferecidos por ela e as facilidades e dificuldades de utilização, bem como as alternativas disponíveis das grandes empresas concorrentes.

Este estudo demonstra, ainda, como é feita a criação de uma aplicação simples, utilizando uma das linguagens suportadas pela plataforma, o Java. E, por fim, com a aplicação desenvolvida, apresenta alguns casos reais de utilização dos serviços disponibilizados pelo Google App Engine.

O trabalho foi estruturado em seis capítulos, sendo que o primeiro foi reservado para esta introdução, o segundo conceitua a plataforma Google App Engine, e um pouco da computação em nuvens, enquanto o terceiro apresenta os aspectos positivos e negativos da plataforma, o quarto mostra os outros serviços oferecidos pela plataforma do Google na nuvem, os quais podem ser utilizados de forma integrada ao App Engine. O quinto descreve o desenvolvimento de uma

aplicação, utilizando a plataforma. O sexto e último capítulo foi reservado para as considerações finais, com base no desenvolvimento do trabalho.

Após a análise dos aspectos positivos e negativos do Google App Engine, e do desenvolvimento da aplicação, é possível avaliar a viabilidade da utilização desta plataforma para o desenvolvimento das aplicações.

2. DESENVOLVIMENTO DE APLICAÇÕES BASEADAS NA WEB

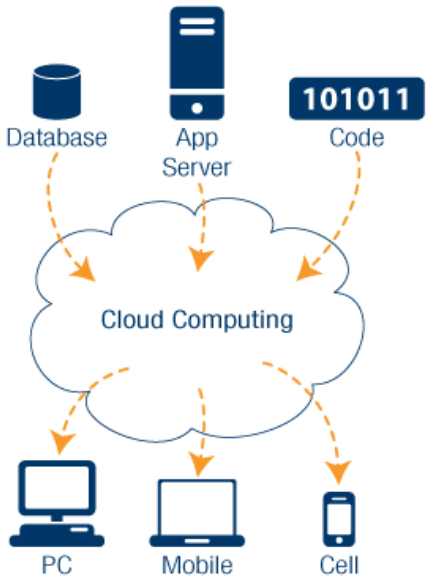
Aplicações baseadas na *Web* são aplicações ou serviços projetados para utilização através da *Web*, normalmente por navegadores, como redes sociais, jogos multijogadores, aplicações móveis, entre outros. O desenvolvimento desse tipo de aplicações está relacionado principalmente à necessidade de simplificar a utilização, atualização e manutenção das mesmas, pois são mantidas em um único local, podendo ser acessadas por diferentes usuários em diversas localidades, sem a necessidade de uma instalação para cada usuário.

2.1. COMPUTAÇÃO EM NUVEM

Computação em Nuvem (ou *Cloud Computing*, em inglês), é uma expressão utilizada para descrever uma variedade de diferentes tipos de conceitos de computação que envolvem um grande número de computadores conectados através de um canal de comunicação em tempo real, mais comumente a Internet.

A estrutura de uma nuvem computacional pode ser vista na Figura 2-1 – Estrutura de uma Nuvem Computacional, na qual a base de dados, o servidor, e todo o código de uma aplicação ficam dentro da nuvem, e os dispositivos acessam essa aplicação diretamente na nuvem, sempre que fazem uma chamada a ela.

Figura 2-1 – Estrutura de uma Nuvem Computacional



Fonte: disponível em http://www.infoescola.com/informatica/computacao-em-nuvem/

Na computação em nuvem, os serviços são divididos em três classes, que levam em consideração o nível de abstração do recurso provido, e o modelo de serviço do provedor. “O nível de abstração pode ser visto como a camada de arquitetura onde os serviços das camadas superiores podem ser compostos pelos serviços das camadas inferiores.” Os modelos de serviços são: Infraestrutura como Serviço, *Software* como Serviço e Plataforma como Serviço (PEDROSA; NOGUEIRA, 2011).

O modelo Infraestrutura como Serviço (IaaS, do inglês, *Infrastructure as a Service*), consiste no serviço de prover soluções de *hardware* com grande escalabilidade, disponibilidade, redundância e segurança (GONÇALVES, 2013).

Já no modelo Plataforma como Serviço (PaaS, do inglês, *Platform as a Service*), o provedor gera a plataforma necessária para o usuário hospedar aplicações criadas por ele ou adquiridas de desenvolvedores que não estão relacionadas ao provedor (GONÇALVES, 2013).

No caso do modelo *Software* como Serviço (SaaS, do inglês, *Software as a Service*), o *software* e todas suas dependências são hospedados em um servidor remoto. O usuário usa uma aplicação desenvolvida pelo provedor do serviço que

está disponível para uso por diversos clientes simultaneamente (GONÇALVES, 2013).

2.2. INTRODUÇÃO AO GOOGLE APP ENGINE

O Google App Engine, ou simplesmente GAE, é uma plataforma de desenvolvimento e hospedagem de aplicações *Web* que utiliza a infraestrutura do Google. É uma tecnologia no modelo Plataforma como Serviço (PaaS, do inglês, *Platform as a Service*). A utilização do Google App Engine dispensa a necessidade de manter servidores (GOOGLE, 2014).

Google App Engine é desenhado para hospedar aplicações que recebem vários usuários simultaneamente. Aplicações que conseguem suportar vários acessos simultaneamente sem que seu desempenho seja afetado são chamadas de escaláveis. Aplicações escritas para o GAE escalam automaticamente: quanto mais acessos a aplicação tiver, mais recursos serão alocados a ela pelo App Engine, e é ele quem vai gerenciar esses recursos. A aplicação em si não tem a necessidade de saber nada sobre os recursos que está consumindo (GOOGLE, 2014).

Abaixo, na Figura 2-2 – Logo Google App Engine, é apresentado o logo da plataforma.

Figura 2-2 – Logo Google App Engine

Fonte: disponível em https://cloud.google.com/products/

2.2.1. O AMBIENTE DE EXECUÇÃO

Uma aplicação App Engine responde a requisições *web*. Uma requisição *web* começa quando um cliente, normalmente um navegador *web* de um usuário, faz contato com a aplicação com uma requisição feita através do Protocolo de Transferência de Hipertexto (HTTP, do inglês, *Hypertext Transfer Protocol*), para entrar em uma página web através de um Localizador Padrão de Recursos (URL, do inglês, *Uniform Resource Locator*). Quando o App Engine recebe a requisição, ele identifica a aplicação pelo nome de domínio do endereço, seja pelo subdomínio *.appspot.com*, que é provido gratuitamente para todas as aplicações, ou seja por um nome de domínio customizado que o desenvolvedor tenha registrado e configurado com o *Google Apps* – serviço do Google que oferece versões de vários de seus produtos, que podem ser personalizados de forma independente com o nome de domínio do cliente. O App Engine seleciona um servidor de vários servidores possíveis para tratar a requisição, fazendo esta seleção baseado em que servidor é o mais susceptível a providenciar uma resposta rápida. Ele então chama a aplicação com o conteúdo da requisição HTTP, recebe os dados de resposta da aplicação, e retorna a resposta ao cliente.

Da perspectiva da aplicação, o ambiente de execução passa a existir quando a manipulação da requisição começa, e desaparece quando termina. App Engine oferece pelo menos dois métodos de armazenamento de dados que persistem entre as requisições, mas estes mecanismos vivem fora do ambiente de execução. Por não reter estado no ambiente de execução entre as requisições, App Engine consegue distribuir o tráfego entre quantos servidores forem necessários para prover a todas as requisições o mesmo tratamento, independente de quanto tráfego estiver tratando de uma só vez.

O código da aplicação não pode acessar o servidor em que está rodando no sentido tradicional. Uma aplicação pode ler seus próprios arquivos do sistema de arquivos, mas não pode escrever nestes arquivos, e não pode ler arquivos que pertencem a outras aplicações. Uma aplicação pode ver as variáveis de ambientes configuradas pelo App Engine, mas a manipulação dessas variáveis não necessariamente persiste entre as requisições. Uma aplicação não pode acessar os

recursos de rede do servidor de *hardware*, apesar de poder realizar operações de serviços usando serviços.

Em resumo, cada requisição permanece em seu próprio *sandbox*[1]. Isso permite que o App Engine trate uma requisição com o servidor que vai providenciar a resposta mais rápida. Não existe uma maneira de garantir que o mesmo servidor de *hardware* trate duas requisições, mesmo que estas requisições venham do mesmo cliente, e cheguem relativamente rápido.

Por causa do *sandbox*, o App Engine pode executar múltiplas aplicações no mesmo servidor, sem que o comportamento de uma aplicação afete outra. Para completar a limitação de acesso ao sistema operante, o ambiente de execução também limita a quantidade de tempo, uso da Unidade Central de Processamento (CPU, do inglês, *Central Processing Unit*) e memória que uma requisição pode utilizar. O App Engine mantém esses limites flexíveis, e aplica limites mais rígidos às aplicações que usam mais recursos, para proteger os recursos compartilhados de aplicações "fugitivas".

Uma requisição tem até 30 segundos para retornar uma resposta ao cliente. Mesmo parecendo uma quantidade de tempo bem confortável para uma aplicação *web*, o App Engine é otimizado para aplicações que respondem em menos de um segundo. Também, se uma aplicação usa muitos ciclos de CPU, o App Engine pode desacelerá-la, de forma que a aplicação não sobrecarregue o processador de uma máquina que está servindo múltiplas aplicações. Um manipulador de requisições com intensidade no uso da CPU pode demorar mais tempo para completar do que demoraria se tivesse uso exclusivo do processador, e o tempo pode variar a medida que o App Engine detecta padrões no uso da CPU e aloca recursos de acordo.

O Google App Engine oferece quatro ambientes de execução: Java, Python, PHP e Go. Estes ambientes usam o mesmo modelo de servidor de aplicação: uma requisição é encaminhada para um servidor, a aplicação é iniciada nele - se necessário - e invocada para manipular a requisição para gerar uma resposta, e essa resposta é retornada ao cliente. Cada ambiente executa seu interpretador com

[1] O conceito do ***Sandbox*** é semelhante ao de criar uma máquina virtual, com foco em segurança. Todos os registros e danos causados pelo que quer que seja executado dentro dele são imediatamente apagados assim que a máquina for reiniciada (GUILHERME, 2012).

restrições de *sandbox*, de forma que qualquer tentativa de usar um recurso da linguagem ou uma biblioteca que requer acesso fora deste *sandbox* vai falhar com uma exceção.

Ao mesmo tempo em que usar um servidor diferente para cada requisição apresenta vantagens na escalação, também apresenta consumo de tempo para iniciar uma nova instância da aplicação para cada requisição. O App Engine minimiza estes custos mantendo a aplicação em memória em um servidor de aplicação o tanto quanto for possível e reusando os servidores de maneira inteligente. Quando um servidor precisa recuperar seus recursos, ele limpa a aplicação menos recentemente usada. Todos os servidores de aplicação têm o ambiente de execução pré-carregado antes que a requisição alcance o servidor, então apenas a aplicação em si precisa ser carregada em um servidor novo.

As aplicações podem explorar o comportamento de *cache* da aplicação para armazenar os dados diretamente no servidor de aplicação utilizando variáveis globais estáticas. Como uma aplicação pode ser despejada entre duas requisições quaisquer, e não há garantias de que as requisições de um determinado usuário vão ser manipuladas pelo mesmo servidor, variáveis globais são muito úteis para armazenar recursos de inicialização, como arquivos de configuração interpretados (SANDERSON, 2010).

2.2.2. OS SERVIDORES DE ARQUIVOS ESTÁTICOS

A maioria dos *websites* têm recursos que são entregues aos navegadores que não são alterados durante uma operação regular do site. As imagens e arquivos de Folhas de Estilo em Cascata (CSS, do inglês, *Cascading Style Sheets*) que definem a aparência do site, os códigos *JavaScript* que rodam no navegador, e arquivos de Linguagem de Marcação de Hipertexto (HTML, do inglês, *HyperText Markup Language*) para páginas sem componentes dinâmicos são alguns exemplos deste tipo de recurso, coletivamente conhecidos como "arquivos estáticos". Desde que a entrega destes arquivos não envolva o código da aplicação, é desnecessário e ineficiente distribui-los dos servidores de aplicação.

Ao invés disso, o App Engine fornece um conjunto separado de servidores dedicados à entrega dos arquivos estáticos. Estes servidores são otimizados para que tanto a arquitetura interna quanto a topologia da rede possam manipular requisições de arquivos estáticos. Para o cliente, arquivos estáticos se parecem com quaisquer outros recursos distribuídos pela aplicação.

O usuário pode configurar vários aspectos de como os arquivos estáticos são distribuídos, incluindo as URLs para os arquivos estáticos, tipos de conteúdo, e instruções para os navegadores manterem cópias dos arquivos em cache por um determinado período de tempo para reduzir o tráfego e acelerar o carregamento da página (SANDERSON, 2010).

2.2.3. O ARMAZENAMENTO DE DADOS

A maioria das aplicações *web* mais úteis precisam guardar informações durante a manipulação de uma requisição para serem recuperadas durante uma requisição posterior. A organização típica para um *website* pequeno envolve um único servidor de banco de dados para o site inteiro, e um ou mais servidores *web* que conectam com o bando de dados para armazenar e recuperar os dados. Usar um único servidor central de banco de dados faz com que seja fácil ter uma representação regular dos dados, então múltiplos usuários, acessando múltiplos servidores *web* verão as mesmas informações, e mais recentes. Mas um servidor central é difícil de escalar uma vez que atinge sua capacidade por causa de conexões simultâneas.

O tipo mais popular de sistemas de armazenamento de dados para aplicações web na década passada tem sido o banco de dados relacional, com tabelas de linhas e colunas organizadas para eficiência de espaço e concisão, e com índices e poder de computação bruto para realizar consultas, especialmente consultas de junção, que podem tratar múltiplos registros relacionados como uma unidade consultável. Outros tipos de sistemas de armazenamento de dados incluem armazenamento de dados hierárquicos (sistemas de arquivos, banco de dados de Linguagem de Marcação Extensível – XML, do inglês, *eXtensible Markup Language*)

e banco de dados orientados a objetos. Cada tipo de banco de dados tem seus prós e contras, e cada um é mais adequado para uma determinada aplicação dependendo da natureza dos dados da aplicação e como eles são acessados. E cada tipo de banco de dados tem suas próprias técnicas para crescer no servidor.

O sistema de banco de dados do Google App Engine se assemelha mais a um banco de dados orientado a objetos; não é um banco de dados relacional. Assim como o ambiente de execução, o desenho do armazenamento de dados do App Engine (chamado de *datastore*) é uma abstração que permite ao App Engine manipular os detalhes da distribuição e escalação da aplicação, desta maneira o código da aplicação pode focar em outras coisas (SANDERSON, 2010).

2.2.3.1. ENTIDADES E PROPRIEDADES

Uma aplicação App Engine armazena seus dados como uma ou mais entidades do armazenamento de dados. Uma entidade tem uma ou mais propriedades, cada uma contendo um nome e um valor de um dos tipos primitivos de valores. Cada entidade é de um determinado tipo, para categorizar a entidade para o propósito de consultas.

A primeira vista, parece muito similar a um banco de dados relacional: entidades de um tipo são como linhas em uma tabela, e propriedades são como colunas - ou campos. Entretanto, há duas diferenças principais entre entidades e linhas. A primeira: não é obrigatório que uma entidade de um determinado tipo tenha as mesmas propriedades de outras entidades do mesmo tipo. A segunda: uma entidade pode ter uma propriedade de mesmo nome que outra entidade, mas com um tipo diferente de valor. Dessa maneira, entidades do armazenamento de dados são *schemaless* [2](sem esquema, em uma tradução livre). Este modelo fornece tanto flexibilidade como alguns desafios de manutenção.

[2] "***Schemaless*** é um recurso oferecido por sistemas gerenciadores de banco de dados que fazem uso da abordagem NoSQL para armazenar dados sem que seja necessário se importar com a normalização dos mesmos." (COSTA, 2013)

Outra diferença entre uma entidade e uma linha de tabela é que uma entidade pode ter múltiplos valores para uma única propriedade.

Todas as entidades do armazenamento de dados tem uma chave única, que pode ser fornecida pela aplicação, ou gerada pelo App Engine; fica a critério do desenvolvedor. Diferente de um banco de dados relacional, a chave não é um campo ou uma propriedade, mas um aspecto independente da entidade. Uma entidade por ser buscada rapidamente se o usuário souber sua chave, e consultas podem ser efetuadas em cima de valores da chave.

Uma chave de entidade não pode ser alterada depois que a entidade é criada, e nem o tipo da entidade. O App Engine usa o tipo da entidade e sua chave para ajudar a determinar aonde a entidade será armazenada, em um grande acervo de servidores - embora nem a chave e nem o tipo da entidade garantem que duas entidades serão armazenadas no mesmo servidor (SANDERSON, 2010).

2.2.3.2. CONSULTAS E ÍNDICES

Uma consulta no armazenamento de dados retorna zero ou mais entidades de um único tipo. Pode também retornar apenas as chaves das entidades que seriam retornadas por uma consulta. Uma consulta pode realizar filtros baseados em condições que devem ser atendidas pelos valores de uma propriedade da entidade, e podem retornar entidades ordenadas pelos valores de uma entidade. Uma consulta também pode realizar filtros e ordenação usando chaves.

Em um banco de dados relacional comum, consultas são planejadas e executadas em tempo real nas tabelas de dados, que são armazenadas da maneira que o desenvolvedor desenhou. O desenvolvedor pode também dizer ao banco de dados para produzir e manter índices ou certas colunas para acelerar certas consultas.

Já o App Engine faz uma coisa de maneira drasticamente diferente. Com o App Engine, todas as consultas têm um índice correspondente mantido pelo armazenamento de dados. Quando a aplicação realiza uma consulta, o

armazenamento de dados procura pelo índice daquela consulta, examina até encontrar a primeira linha que corresponde à consulta, e então retorna a entidade para cada linha consecutiva do índice, até a primeira linha que não corresponde à consulta.

Mas isso requer que o App Engine saiba com antecedência as consultas a aplicação irá realizar. Não é necessário que ele saiba os valores dos filtros antecipadamente, mas é necessário que saiba o tipo de entidade a ser consultada, as propriedades pelas quais vão ser realizados filtros ou ordenações, e os operadores dos filtros e a ordem das classificações.

O App Engine fornece um conjunto de índices para consultas simples por padrão, baseado em quais propriedades existem nas entidades de um tipo. Para consultas mais complexas, uma aplicação precisa incluir especificações de índices em sua configuração. O SDK do App Engine ajuda a produzir este arquivo de configuração, verificando quais consultas são realizadas enquanto o desenvolvedor testa sua aplicação usando o servidor web de desenvolvimento fornecido em seu próprio computador. Quando o desenvolvedor subir sua aplicação no App Engine, o armazenamento de dados sabe criar os índices para todas as consultas que a aplicação realizou durante os testes. A configuração destes índices pode ser realizada manualmente também.

Quando a aplicação cria novas entidades e atualiza as existentes, o armazenamento de dados atualiza todos os índices correspondentes. Isto faz as consultas serem muito rápidas – pois cada consulta é uma simples varredura em uma tabela –, a custo de atualização de entidades – pois possivelmente várias tabelas precisam ser atualizadas por causa de uma única mudança. De fato, o desempenho de uma consulta indexada não é afetado pelo número de entidades armazenadas, apenas o tamanho do conjunto de resultados (SANDERSON, 2010).

2.2.3.3. TRANSAÇÕES

Quando uma aplicação tem muitos clientes tentando ler ou escrever os mesmos dados simultaneamente, é indispensável que os dados sempre estejam em um estado consistente. Um usuário nunca deve ver dados escritos pela metade ou dados que não fazem sentido porque a ação de outro usuário não está completa.

Quando uma aplicação atualiza as propriedades de uma única entidade, o App Engine garante ou que as atualizações da entidade sejam todas bem sucedidas de uma vez, ou que a atualização inteira falhe e a entidade permaneça do jeito que estava antes do início da atualização. Outros usuários não veem nenhum efeito da mudança, a menos que ela tenha sido bem sucedida.

Em outras palavras, uma atualização de uma única entidade ocorre em uma *transação*. Cada transação é atômica: a transação ou é bem sucedida ou falha completamente, e não pode ser sucedida ou falhar em pedaços menores (SANDERSON, 2010).

Uma aplicação pode ler ou atualizar múltiplas entidades em uma única transação, mas ela deve dizer ao App Engine quais entidades serão atualizadas juntas na criação das entidades. A aplicação faz isso através da criação das entidades em grupos de entidades. O App Engine usa grupos de entidades para controlar como as entidades são distribuídas pelos servidores, dessa maneira pode garantir que uma transação em um grupo seja bem sucedida ou falhe completamente. Em termos de banco de dados, o armazenamento de dados do App Engine tem suporte nativo para transações locais.

Quando uma aplicação chama a Interface de Programação de Aplicações (API, do inglês, *Application Programming Interface*) do armazenamento de dados para atualizar uma entidade, o controle não retorna para a aplicação até que a transação seja sucedida ou falhe, e a chamada retorna com o conhecimento de sucesso ou falha. Para atualizações, isso significa que a aplicação espera por todas as entidades e índices serem atualizados antes de fazer qualquer outra coisa.

Se um usuário tentar atualizar uma entidade enquanto a atualização da entidade de outro usuário está em andamento, o armazenamento de dados retorna imediatamente com uma exceção de falha de concorrência. É muitas vezes apropriado para a aplicação tentar executar uma transação devolvida várias vezes antes de declarar a condição de erro, normalmente recuperando os dados que podem ter sido alterados dentro de uma transação antes de calcular os novos valores e atualizá-los. Em termos de banco de dados, o App Engine usa controle de concorrência otimista.

Ler a entidade nunca falha devido a concorrência; a aplicação apenas enxerga a entidade em seu estado estável mais recente.

Na maioria dos casos, a recuperação de uma transação em uma entidade contestada vai ser bem sucedida. Mas se uma aplicação é projetada de modo que muitos usuários podem atualizar uma única entidade, quanto mais a aplicação se tornar popular, com mais frequência os usuários vão receber falhas de transações concorrentes. É importante desenhar grupos de entidades para evitar falhas de concorrência mesmo com um grande número de usuários.

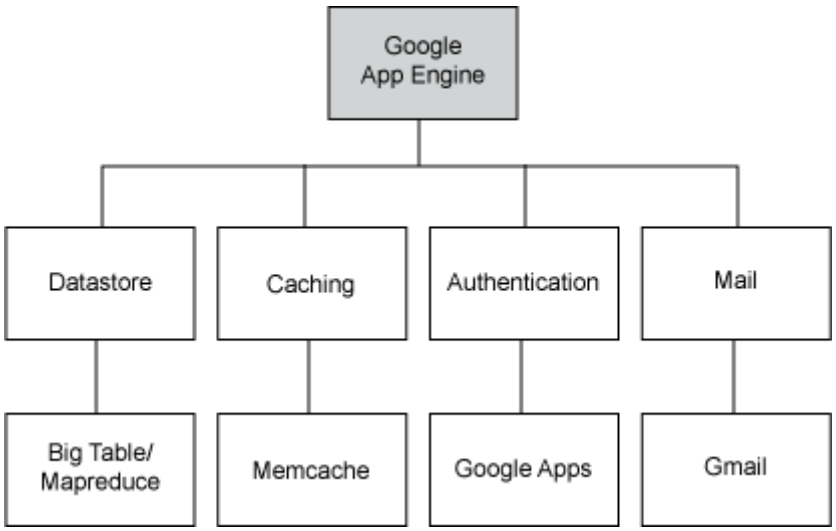
Uma aplicação pode empacotar múltiplas operações de armazenagem de dados em uma única transação. Por exemplo, a aplicação pode começar uma transação, ler uma entidade, atualizar o valor de uma propriedade com base no último valor lido, salvar a entidade, e então efetivar a transação. Se existir um conflito e a aplicação quiser tentar novamente, ela deve recuperar a transação inteira: ler a entidade, que possivelmente foi atualizada, novamente, usar o novo valor para o cálculo, e tentar atualizar novamente.

Com índices e com o controle de concorrência otimista, o armazenamento de dados do App Engine é projetado para aplicações que precisam ler dados rapidamente, garantir que os dados vistos estão em uma forma consistente, e escalar o número de usuários e o tamanho dos dados automaticamente. Enquanto estes objetivos são um tanto diferentes daqueles de um banco de dados relacional, eles são especialmente adequados para aplicações *web* (SANDERSON, 2010).

2.2.4. OS SERVIÇOS

O relacionamento do armazenamento de dados com o ambiente de execução é o de um serviço: a aplicação usa uma API para acessar um sistema separado que gerencia cada uma de suas próprias necessidades separadamente do ambiente de execução. Google App Engine inclui inúmeros outros serviços auto escaláveis úteis para aplicações web. O organograma destes serviços pode ser observado na

Figura 2-3 – Serviços do Google App Engine



Fonte: disponível em http://www.ibm.com/developerworks/br/opensource/library/os-cloud-virtual2/

O serviço de cache de memória, ou *memcache*, é um serviço de armazenamento de chave-valor de curto prazo. Sua vantagem principal sobre o armazenamento de dados é que é rápido, muito mais rápido para armazenamento e recuperação simples. O *memcache* guarda valores em memória ao invés de no disco para acesso mais rápido. É distribuído igual ao armazenamento de dados, e dessa forma todas as requisições veem o mesmo conjunto de chaves e valores. Entretanto, não é persistente como o armazenamento de dados: se um servidor cai, como em uma falha de energia, por exemplo, a memória é apagada. Ele também tem um senso de atomicidade e transacionalidade mais limitado que o armazenamento de dados. Como o nome sugere, o serviço de *memcache* é mais bem usado como um *cache* para os resultados de consultadas realizadas com

frequência ou cálculos. A aplicação busca por um valor em *cache*, e se o valor não está lá, ela realiza a consulta ou o cálculo e armazena o valor no *cache* para usos posteriores.

As aplicações App Engine podem acessar outros recursos *web* usando o serviço de busca de URL - *Fetch URL*. O serviço faz requisições HTTP para outros servidores na Internet, como para recuperar páginas ou interagir com serviços web. Como servidores remotos podem ser lentos para responder, a API de busca de URL suporta a busca de URLs no segundo plano enquanto um processador de requisições faz outras coisas, mas em todos os casos, a busca deve iniciar e terminar junto com o tempo de vida que o processador de requisições. A aplicação pode também estabelecer um prazo, depois do qual a chamada é cancelada se o hospedeiro remoto ainda não respondeu.

Aplicações App Engine podem mandar mensagens usando o serviço de *e-mail*. As mensagens podem ser enviadas do lado da aplicação ou do lado do usuário que fez a requisição que está enviando o *e-mail* (se a mensagem é do usuário). Muitas aplicações *web* usam *e-mails* para notificar usuários, confirmar ações de usuários, e validar informações de contato.

Uma aplicação pode também receber mensagens de *e-mail*. Se uma aplicação é configurada para receber *e-mail*, uma mensagem que é enviada para seu endereço é encaminhada para o serviço de *e-mail*, que entrega a mensagem para a aplicação em forma de uma requisição HTTP para um manipulador de requisições.

Aplicações App Engine podem enviar e receber mensagens instantâneas de e para serviços de bate-papo que suportam o Protocolo Extensível de Mensagem e Presença (XMPP, do inglês, *eXtensible Messaging and Presence Protocol*), inclusive o Google Talk (serviço de bate-papo do Google). Uma aplicação envia uma mensagem de bate-papo XMPP, através da chamada do serviço XMPP. Assim como o recebimento de *e-mail*, quando alguém envia uma mensagem para o endereço da aplicação, o serviço XMPP a entrega para a aplicação chamando um manipulador de requisições.

O serviço de processamento de imagens pode realizar transformações leves de imagens, como para fazer miniaturas de fotos carregadas. As tarefas de processamento de imagens são executadas usando a mesma infraestrutura que o Google usa para processar imagens de outros de seus produtos, portanto o resultado retorna rapidamente (SANDERSON, 2010).

2.2.5. CONTAS DO GOOGLE

O App Engine possui integração com o Google Accounts, que é o sistema de conta de usuários usado pelas aplicações do Google, como o Google Mail (*e-mail* do Google), Google Drive (armazenamento de arquivos do Google) e Google Calendar (calendário do Google). O desenvolvedor pode usar o Google Accounts como o sistema de conta de usuários de sua aplicação, e dessa forma ele não precisa criar um sistema próprio. Desta maneira, se os usuários da aplicação já possuírem uma conta do Google, eles poderão acessar a aplicação usando suas contas existentes, sem a necessidade de criar uma nova conta apenas para utilizar esta aplicação. Mas o desenvolvedor também pode optar por criar seu próprio sistema de contas, usando o provedor OpenID.

A utilização do Google Accounts como o sistema de conta de usuários é especialmente útil para desenvolver aplicações para uma empresa que utiliza o Google Apps (serviço do Google para uso, por uma empresa, de domínios próprios em diversos produtos oferecidos pelo Google) (SANDERSON, 2010).

2.2.6. FILAS DE TAREFAS E TAREFAS AGENDADAS

Uma aplicação web tem que responder a requisições web de forma rápida, normalmente em menos de um segundo e preferivelmente em apenas algumas dúzias de milissegundos, para oferecer uma experiência suave ao usuário. Isso não dá à aplicação muito tempo para realizar seu trabalho. Às vezes há mais trabalho a fazer do que tempo para fazê-lo. Nestes casos normalmente é aceitável se o

trabalho é feito em alguns segundos, minutos, ou horas ao invés de imediatamente, enquanto o usuário espera por uma resposta do servidor. Mas o usuário precisa de uma garantia de que o trabalho será feito.

Para este tipo de trabalho, o App Engine usa filas de tarefas (chamadas de *Task Queues*). As filas de tarefas permite que os manipuladores de requisições designem trabalhos para serem executados posteriormente, fora do escopo de uma requisição *web*. As filas garantem que todas as tarefas sejam executadas eventualmente. Se uma tarefa falhar, a fila repete a tentativa até que a tarefa seja executada com sucesso. O desenvolvedor pode configurar o intervalo em que as filas serão processadas para espalhar a carga de trabalho durante o dia.

Uma fila executa uma tarefa chamando um manipulador de requisições. Pode incluir uma carga útil de dados fornecida pelo código que criou a tarefa, entregue para o manipulador da tarefa como uma requisição HTTP. O manipulador da tarefa é sujeito aos mesmos limites que os outros manipuladores de requisições, inclusive ao limite de 30 segundos.

Uma ferramenta especialmente poderosa das filas de tarefas é a habilidade de enfileirar uma tarefa dentro de uma transação do armazenamento de dados. Isso garante que a tarefa será enfileirada apenas se o restante da transação do armazenamento de dados for bem sucedido. Pode-se usar tarefas transacionais para executar operações de armazenamento de dados adicionais que precisam ser consistentes com a transação eventualmente, mas que não precisam das fortes garantias de consistência das operações locais do armazenamento de dados.

O App Engine tem outro serviço para executar tarefas em horários específicos do dia. Tarefas agendadas são também conhecidas como *cron jobs*, nome emprestado de uma funcionalidade similar do sistema operacional Unix. O serviço de tarefas agendadas pode invocar um manipulador de requisições em um horário específico do dia, semana, ou mês, baseado em um agendamento que o desenvolvedor fornece quando sobe sua aplicação para o App Engine. Esse tipo de tarefas é muito útil para realização de manutenções regulares ou para envio de mensagens periódicas (SANDERSON, 2010).

2.2.7. FERRAMENTAS DO DESENVOLVEDOR

O Google fornece ferramentas gratuitas para o desenvolvimento de aplicações App Engine em Java, Python, PHP e Go. O *download* dos SDKs, está disponível no site do Google.

Cada SDK inclui um servidor de desenvolvimento *web*, que roda a aplicação no computador local do desenvolvedor, e simula o ambiente de execução, o armazenamento de dados e os serviços do Google App Engine. O servidor de desenvolvimento detecta automaticamente alterações feitas nos arquivos de código e os recarrega conforme necessário; dessa maneira, o desenvolvedor pode manter o servidor rodando enquanto desenvolve a aplicação.

A versão de desenvolvimento do armazenamento de dados pode gerar automaticamente as configurações para os índices de consulta, conforme a aplicação executa essas consultas.

O servidor de desenvolvimento inclui uma aplicação *web* embutida para inspecionar o conteúdo do armazenamento de dados simulado. Essa interface pode ser usada para criar novas entidades para testes (SANDERSON, 2010).

2.2.8. O CONSOLE DE ADMINISTRAÇÃO

Quando a aplicação está pronta para ser publicada, o desenvolvedor deve criar uma conta de administrador e configurar a aplicação no App Engine. Essa conta deve ser usada para criar e gerenciar a aplicação, ver os acessos e as estatísticas de utilização dos recursos e as mensagens de registros, e tudo isso utilizando uma interface baseada na *web*, chamada de Console de Administração.

O acesso ao Console de Administração é realizado através de uma conta do Google. O desenvolvedor pode optar por utilizar uma conta existente, ou criar uma específica para a aplicação, que pode ser usada posteriormente como o endereço de remetente dos *e-mails* enviados por ela, por exemplo. Após criar uma aplicação

usando o Console de Administração, outras contas do Google podem ser adicionadas como administradoras: qualquer administrador pode acessar o Console e subir uma nova versão da aplicação.

O Console fornece acesso em tempo real às informações sobre como a aplicação está sendo usada, bem como acesso aos registros de dados emitidos pela aplicação. O armazenamento de dados também pode ser acessado, utilizando uma interface web, as consultas podem ser realizadas e o estado dos índices pode ser consultado (SANDERSON, 2010).

2.3. ALTERNATIVAS DISPONÍVEIS

Existem inúmeras outras plataformas que oferecem serviços de forma similar ao Google App Engine. Aqui serão apresentadas as mais conhecidas.

AMAZON EC2

O *Amazon Elastic Compute Cloud*, ou simplesmente Amazon EC2, é parte da plataforma da Amazon.com na nuvem computacional, Amazon Web Services, ou simplesmente AWS. Ele permite que os usuários executem programas de aplicação em ambiente de computação AWS.

Para utilizar o EC2, o assinante realiza uma série de configurações, descritas no site da AWS. Após a conclusão dessas configurações, o assinante pode solicitar as máquinas virtuais conforme sua necessidade. A capacidade pode ser aumentada ou diminuída em tempo real de uma a mais de 1.000 máquinas virtuais simultaneamente. A cobrança é feita de acordo com os recursos de computação e rede consumidos.

Da mesma forma que o Google App Engine pode ser integrado com outros serviços da plataforma do Google na nuvem, o EC2 também pode ser integrado com outros serviços do AWS (AMAZON, 2014).

O Amazon EC2 se enquadra na categoria de Infraestrutura como Serviço (IaaS).

MICROSOFT AZURE

O Microsoft Azure, ou simplesmente Azure, é uma plataforma de nuvem, que pode ser classificada como Infraestrutura como Serviço (IaaS) e Plataforma como Serviço (PaaS). É um serviço totalmente hospedado e controlado pela Microsoft.

Com o Azure, é possível obter máquinas virtuais, e também criar e implantar aplicativos *web* (MICROSOFT, 2014).

3. VANTAGENS E DESVANTAGENS

O Google App Engine possui aspectos positivos e negativos que devem ser considerados na escolha de utilizá-lo ou não. Abaixo, os principais de ambos os lados serão brevemente apresentados e conceituados. Um resumo destes aspectos pode ser observado na Figura 3-1 – Aspectos Positivos e Negativos.

Figura 3-1 – Aspectos Positivos e Negativos

Fonte: elaborada pela autora

3.1. ASPECTOS POSITIVOS

O Google App Engine apresenta grande facilidade de utilização, pois ele é um conjunto completo de desenvolvimento que utiliza tecnologias familiares para construir e hospedar aplicativos da *web*. Com o Google App Engine, o desenvolvedor pode criar o código do aplicativo, utilizando o kit de desenvolvimento disponibilizado pelo Google em seu site, testá-lo no próprio computador local e enviá-lo para o Google com apenas um clique ou com um *script* de linha de comando. Depois de enviado para o Google, ele hospeda e escala o aplicativo para o desenvolvedor, e este não precisa mais se preocupar em administrar o sistema,

produzir instâncias novas do aplicativo, particionar o banco de dados ou comprar máquinas. O Google cuida de toda a manutenção necessária (GOOGLE, 2014).

Como já visto anteriormente, aplicações escritas para o GAE escalam automaticamente dependendo da quantidade de acessos simultâneos esta aplicação tiver. Em momento algum o usuário desenvolvedor da aplicação tem a necessidade de se preocupar com os recursos utilizados por ele, e se estes recursos serão suficientes para lidar com tantas requisições: é o App Engine quem vai fazer isso automaticamente. Ele irá alocar mais recursos para a aplicação quando necessário, e desalocá-los quando não houver mais a necessidade (GOOGLE, 2014).

O Google é famoso por ser altamente confiável e por sua infraestrutura de alto desempenho. Com o Google App Engine, o desenvolvedor pode se beneficiar dos 10 anos de conhecimento que o Google possui em executar sistemas altamente escalonáveis e orientados para o desempenho. As mesmas políticas de segurança, privacidade e proteção de dados que o Google tem para seus aplicativos se aplicam também a todos os aplicativos do Google App Engine (GOOGLE, 2014).

O GAE tem um baixo custo, pois, diferente de dos servidores de hospedagem ou gerenciamento tradicionais, o usuário paga apenas pelos recursos que utiliza. A criação de um aplicativo é gratuita. Um aplicativo de conta gratuita pode usar até 500MB de armazenamento, e até 5 milhões de visualizações de página por mês. O usuário pode optar, mais tarde, ativar o faturamento, através da definição de um orçamento máximo diário, e alocar este orçamento para cada recurso de acordo com as necessidades de sua aplicação. Cada aplicativo recebe recursos dentro de limites, as chamadas "cotas" (GOOGLE, 2014).

3.2. ASPECTOS NEGATIVOS

Nos parágrafos seguintes, são explorados os principais aspectos negativos da utilização do Google App Engine.

Atualmente, o Google App Engine oferece suporte, por enquanto, para poucas linguagens: Java e Python. Mais recentemente, o GAE tem oferecido suporte

também às linguagens PHP e Go. Porém, o Google App Engine é uma plataforma que ainda está em crescimento e desenvolvimento, então é provável que essa restrição de linguagem seja temporária (GOOGLE, 2014).

Alguns recursos impõem limites não relacionados a cotas para proteger a estabilidade do sistema. Por exemplo, quando um aplicativo é chamado para servir uma solicitação da *web*, ele deve emitir uma resposta em até 30 segundos. Se o aplicativo demorar muito, o processo será encerrado e o servidor retornará um código de erro ao usuário. O tempo de espera de solicitação é dinâmico e pode ser reduzido para poupar os recursos caso um manipulador de solicitação chegue ao tempo limite com muita frequência (GOOGLE, 2014).

Algumas aplicações têm a necessidade de oferecer disponibilidade constante para o acesso dos usuários. Como já visto, com o App Engine, o desenvolvedor só paga pelo que usa. Já foi visto também que o App Engine trabalha com instâncias das aplicações, e sempre que uma instância nova é criada durante o acesso de um usuário, o tempo de resposta é maior. O número de instâncias grátis do Google App Engine é baixo para aplicações que necessitam de disponibilidade constante. Portanto, aplicações que apresentam essa necessidade, precisam de um número maior de instâncias, o que vai gerar um alto custo mensal para o desenvolvedor (GOOGLE, 2014).

4. OUTROS SERVIÇOS DO GOOGLE NA NUVEM

A plataforma do Google na nuvem, também chamada de *Google Cloud Platform*, é um conjunto de produtos e serviços de computação em nuvens fornecidos pelo Google, que utilizam a mesma infraestrutura que o próprio Google utiliza em seus produtos internos para clientes finais. O objetivo destes produtos e serviços para os desenvolvedores é que eles foquem na escrita e desenvolvimento do código, e não na infraestrutura de suas aplicações (GOOGLE, 2014).

A seguir, na Figura 4-1 – Logo *Google Cloud Platform*, o logo da plataforma pode ser visualizado.

Figura 4-1 – Logo *Google Cloud Platform*

Fonte: disponível em https://cloud.google.com/

Além do Google App Engine, a plataforma do Google da nuvem inclui vários outros produtos e serviços. Eles podem ser combinados para atender às necessidades de cada aplicação, inclusive podem ser combinados com o próprio App Engine. Os produtos e serviços incluem:

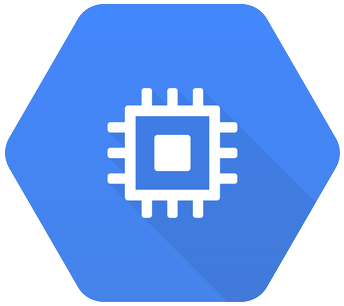
COMPUTE ENGINE

Google Compute Engine é um serviço que fornece máquinas virtuais que rodam na infraestrutura do Google. Com esse serviço, o usuário pode rodar milhares de CPUs virtuais na infraestrutura do Google, que oferece escala, desempenho e valor. Ele tem a capacidade de criar máquinas virtuais com uma grande variedade de configurações, manter e guardar dados com armazenamento em blocos, gerenciar redes de acesso às máquinas virtuais e usar uma variedade de

ferramentas de autenticação para que o usuário gerencie suas máquinas virtuais (GOOGLE, 2014).

O logo deste serviço é apresentado na Figura 4-2 – Logo *Google Compute Engine*.

Figura 4-2 – Logo *Google Compute Engine*



Fonte: disponível em https://cloud.google.com/products/

CLOUD STORAGE

Google Cloud Storage é um serviço da Internet para armazenar dados na nuvem do Google. Ele permite o armazenamento e recuperação de qualquer quantidade de dados a qualquer hora, com uma interface de programação simples, também utilizando as vantagens da infraestrutura do Google, de maneira segura (GOOGLE, 2014).

Na Figura 4-3 – Logo *Google Cloud Storage* é apresentado o logo deste serviço.

Figura 4-3 – Logo *Google Cloud Storage*

Fonte: disponível em https://cloud.google.com/products/

CLOUD DATASTORE

Como já visto anteriormente, o *Datastore*, que é o armazenamento de dados do Google App Engine, é um banco de dados não relacional, que armazena os objetos de dados como entidades, que contêm propriedades (GOOGLE, 2014).

O logo do serviço de armazenamento de dados do Google App Engine é mostrado na Figura 4-4 – Logo *Google Cloud Datastore*.

Figura 4-4 – Logo *Google Cloud Datastore*

Fonte: disponível em https://cloud.google.com/products/

CLOUD SQL

Google Cloud SQL é um banco de dados MySQL que existe na nuvem do Google. Tem todas as capacidades e funcionalidades do MySQL, com alguns

recursos adicionais. Ele está disponível para as aplicações do App Engine, e é uma alternativa de armazenamento de dados para aplicações em que o *Datastore* não supre as necessidades de armazenamento (GOOGLE, 2014).

A seguir, na Figura 4-5 – Logo *Google Cloud SQL* pode ser visto o logo do banco de dados MySQL do Google na nuvem.

Figura 4-5 – Logo *Google Cloud SQL*

Fonte: disponível em https://cloud.google.com/products/

GOOGLE BIGQUERY

Google BigQuery é um serviço *web* que possibilita a análise interativa de grandes conjuntos de dados em massa, trabalhando em conjunto com o *Google Cloud Storage*. As consultas dos dados são feitas usando consultas do tipo SQL (GOOGLE, 2014).

Na Figura 4-6 – Logo *Google BigQuery*, a seguir, é possível visualizar o logo deste serviço.

Figura 4-6 – Logo *Google BigQuery*

Fonte: disponível em https://cloud.google.com/products/

5. DESENVOLVIMENTO DE APLICAÇÕES

Os aplicativos desenvolvidos para o Google App Engine podem ser desenvolvidos em quatro ambientes: Java, Python, Go e PHP.

O modelo de desenvolvimento de uma aplicação no App Engine é simples, e envolve:

I. Criar a aplicação;
II. Testá-la no ambiente local, usando o *software* de servidor *web* incluso junto com o kit de desenvolvimento do App Engine;
III. Subir a aplicação finalizada no App Engine.

Neste trabalho, o desenvolvimento de uma aplicação será feito utilizando a linguagem Java, através do SDK Java.

Todas as ferramentas e bibliotecas necessárias para o desenvolvimento de uma aplicação são inclusas no SDK do App Engine. Os SDKs funcionam em qualquer plataforma, incluindo Windows, Mac OS e Linux.

Como visto anteriormente, uma aplicação App Engine responde a requisições *web*. Ela faz isso chamando os manipuladores de requisições, que são rotinas que aceitam parâmetros de requisições e retornam respostas. O *App Engine* determina qual manipulador de requisições será usado para tratar uma determinada requisição da URL de requisições, usando um arquivo de configurações incluído na aplicação que mapeia as URLs para os manipuladores.

Uma aplicação pode também incluir arquivos estáticos, como imagens, folhas de estilo CSS, e *JavaScript*. O App Engine fornece esses arquivos diretamente para os clientes em resposta a requisições de URLs correspondentes, sem a necessidade de invocar nenhum código. As configurações da aplicação especificam quais arquivos são estáticos, e quais URLs usar para estes arquivos.

As configurações da aplicação incluem metadados sobre ela, como o Número de Identificação (ID) da aplicação e número da versão. Quando a aplicação é implantada no App Engine, todos os seus arquivos, incluindo código, arquivos de

configuração, e arquivos estáticos, são carregados e associados com o ID da aplicação, e número de versão, mencionados na configuração. Uma aplicação também pode ter arquivos de configuração específicos para os serviços, como índices do armazenamento de arquivos, filas de tarefas, e tarefas agendadas. Estes arquivos são associados com a aplicação em geral, e não com uma versão específica da aplicação.

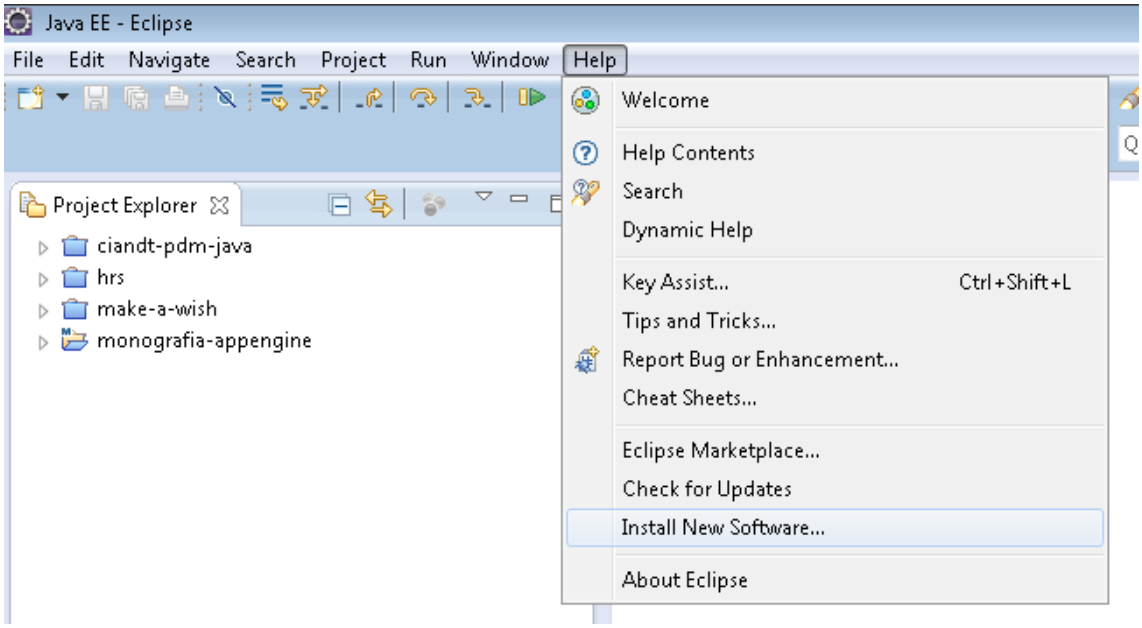
5.1. DESENVOLVENDO UMA APLICAÇÃO JAVA

Conforme apresentado anteriormente, neste trabalho será desenvolvida uma aplicação utilizando a linguagem Java, na versão 7. Para tanto, foi instalado o Java 7 e o IDE Java – mas que também pode ser utilizada para desenvolvimento em outras linguagens – Eclipse, versão Juno.

Antes de criar o projeto, é necessário instalar o *plugin* do Google para o Eclipse. Essa tarefa é facilmente realizada através da ferramenta de atualização de *software* do Eclipse. No próprio site da documentação do Google App Engine é possível encontrar o endereço correto do *plugin* para a versão da IDE que estiver utilizando, como também um passo a passo de como realizar a instalação.

Após abrir a IDE, é necessário acessar a ferramenta de instalação/atualização de *software*, como mostrado na Figura 5-1 – Instalação de *Plugin*, passo 1.

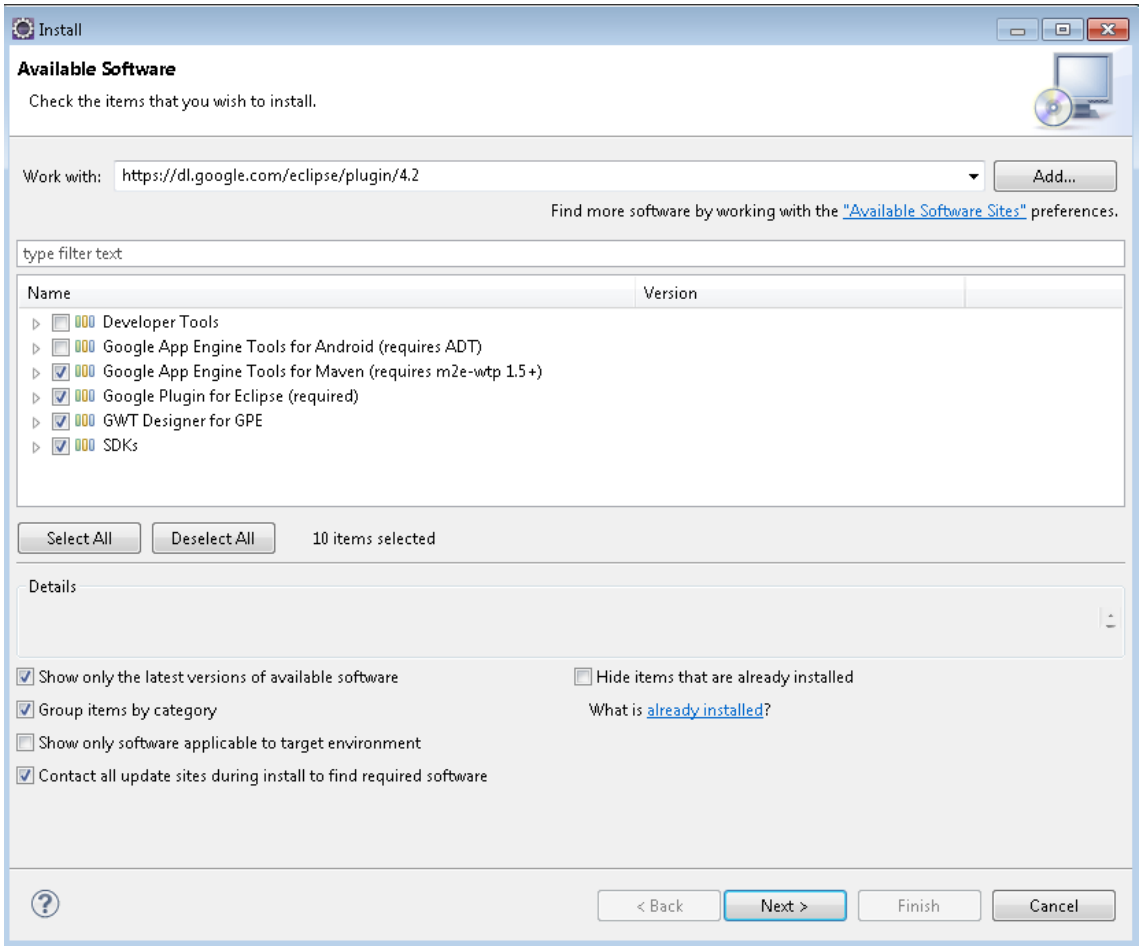
Figura 5-1 – Instalação de *Plugin*, passo 1



Fonte: elaborada pela autora

Isso irá abrir a janela da ferramenta, que exibirá um campo aonde pode ser inserida a URL do *plugin*. Como neste trabalho a versão da IDE é a 4.2 – Juno, o endereço inserido é o correspondente encontrado na documentação do Google App Engine. Após clicar no botão *"Add..."*, a ferramenta de instalação exibirá todos os componentes possíveis de serem instalados. Para o desenvolvimento de uma aplicação simples, não é necessário instalar todos os componentes; portanto, apenas os realmente necessários são selecionados para instalação. Depois de selecionar os componentes, é necessário clicar no botão "*Next*", para passar ao próximo passo. O passo descrito neste parágrafo pode ser observado na Figura 5-2 – Instalação de *Plugin*, passo 2.

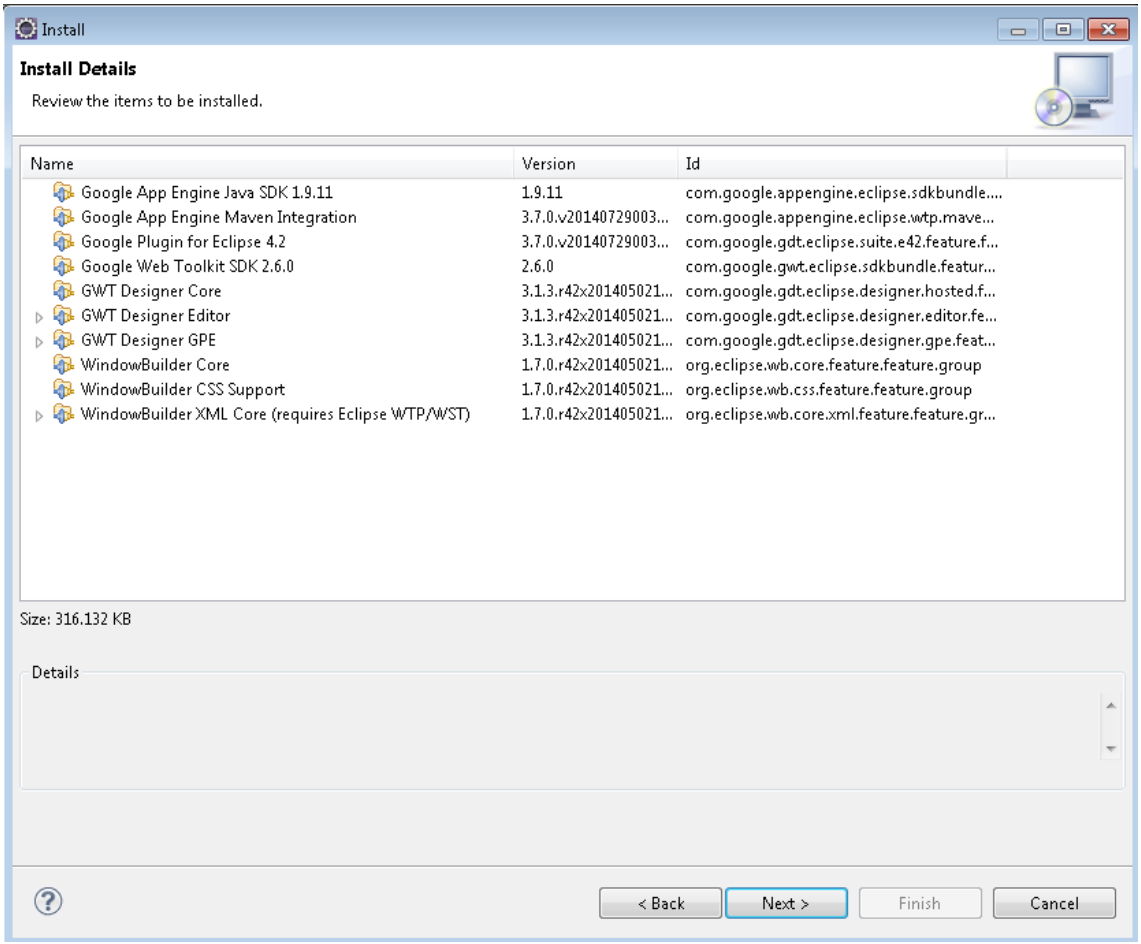
Figura 5-2 – Instalação de *Plugin*, passo 2



Fonte: elaborada pela autora

O passo seguinte é apenas uma janela de confirmação dos componentes a serem instalados. É necessário apenas rever todos os componentes, para se certificar de que não está faltando nenhum componente, ou sendo instalado um componente desnecessário, e clicar no botão "*Next*", para o próximo passo, como pode ser visto na Figura 5-3 – Instalação de *Plugin*, passo 3.

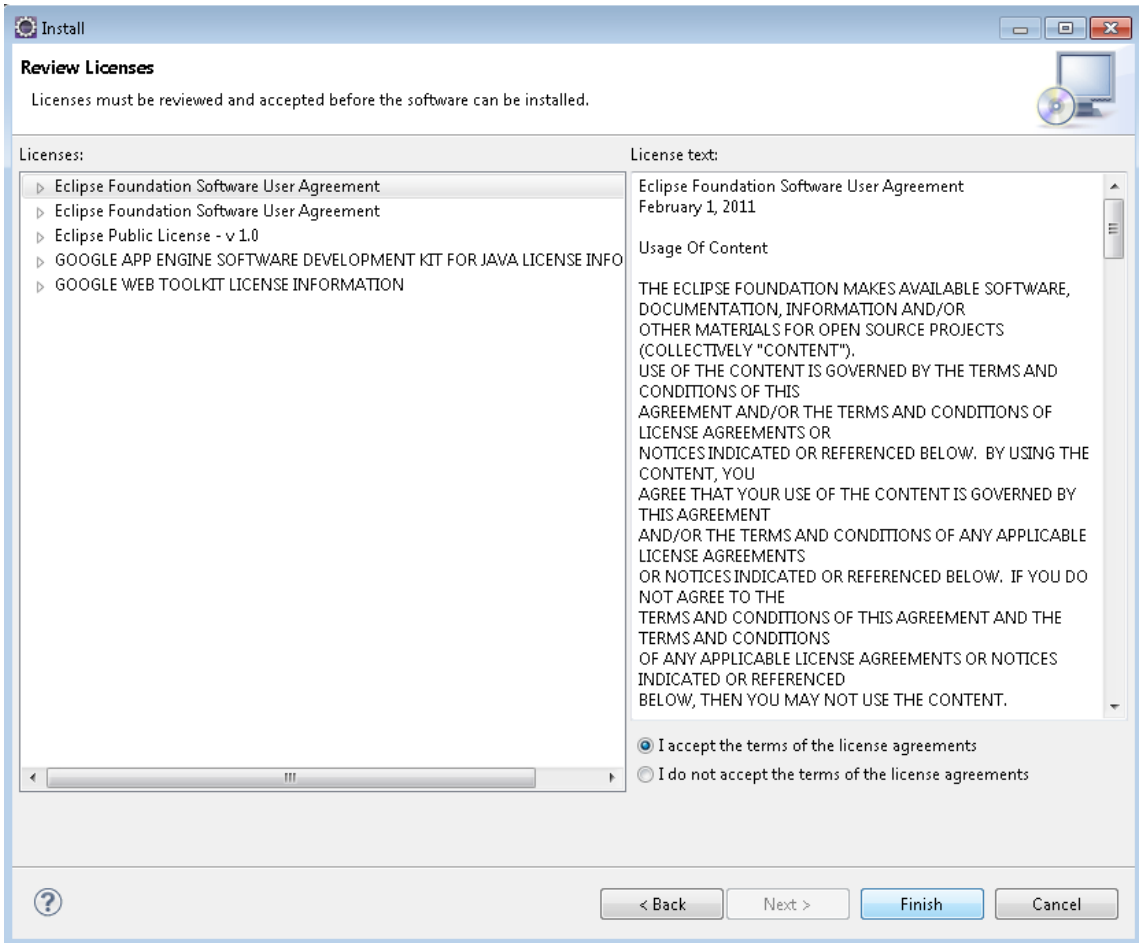
Figura 5-3 – Instalação de *Plugin*, passo 3



Fonte: elaborada pela autora

Em seguida, é necessário ler e aceitar os termos de uso, e clicar em "*Finish*" para a conclusão da instalação do *plugin* e dos componentes selecionados (Figura 5-4 – Instalação de *Plugin*, passo 4).

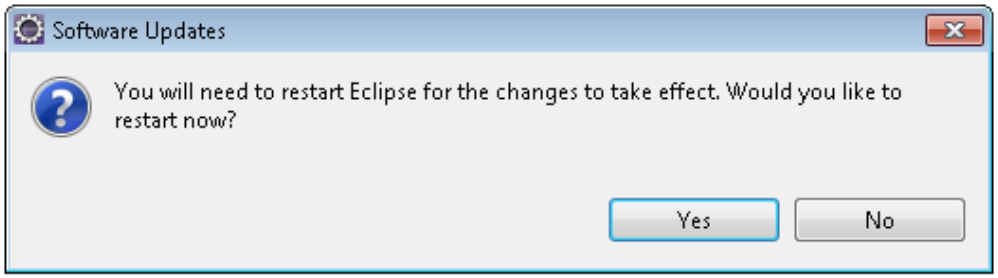
Figura 5-4 – Instalação de *Plugin*, passo 4



Fonte: elaborada pela autora

Por fim, é exibida uma janela pedindo para reiniciar o Eclipse, para que as instalações tenham efeito na IDE, e os componentes do *plugin* instalado possam ser utilizados, conforme Figura 5-5 – Instalação de *Plugin*, passo 5. Após clicar em "*Yes*", a IDE é reiniciada, e a instalação está efetivamente concluída.

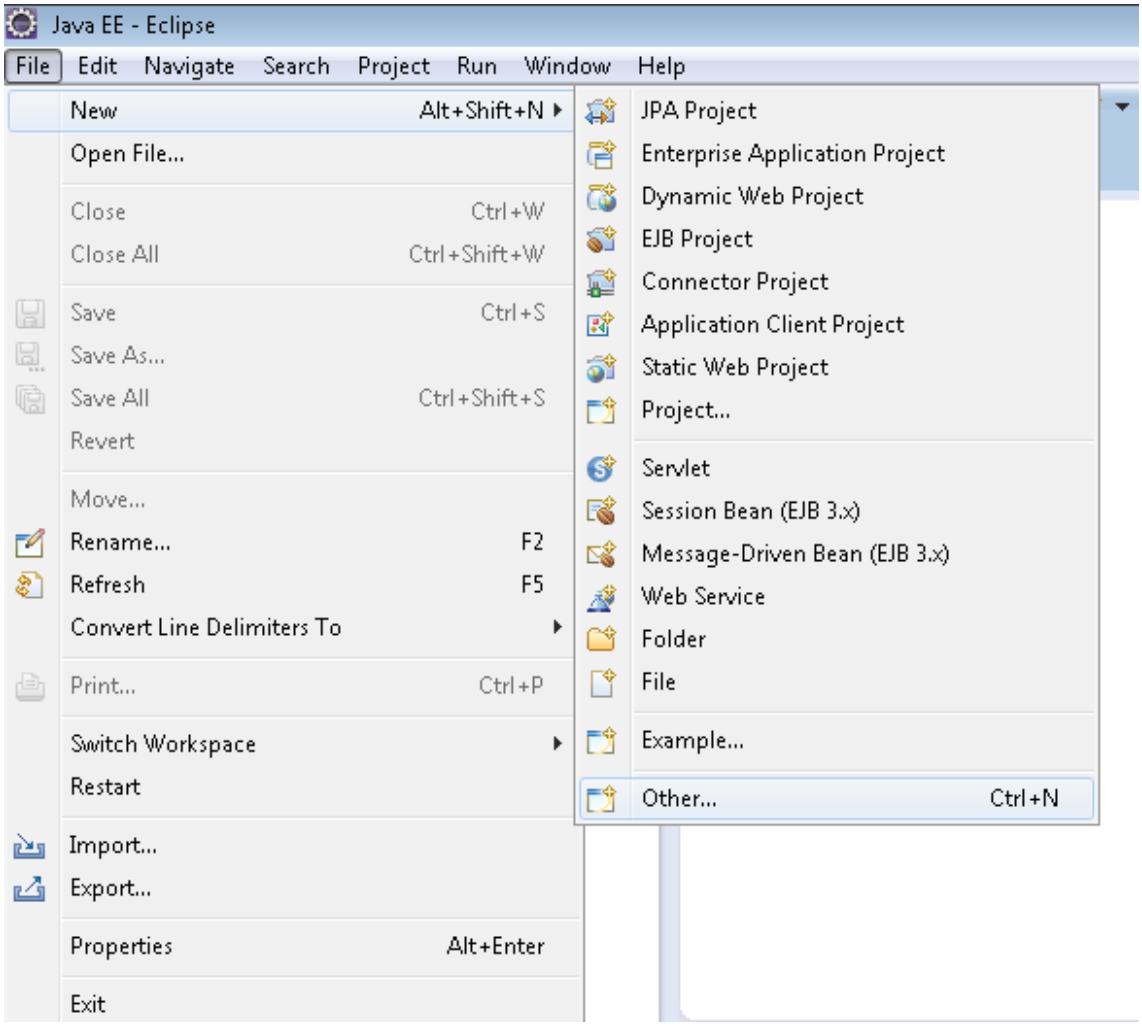
Figura 5-5 – Instalação de *Plugin*, passo 5



Fonte: elaborada pela autora

Com a instalação do *plugin* do Google App Engine concluída, é possível agora criar um novo projeto do Google. Para tanto, primeiramente deve-se acessar a ferramenta de criação de novo projeto, encontrada na IDE. Como não foi criado nenhum projeto do Google nessa IDE ainda, a opção não se encontra na lista principal exibida, então é preciso acessar a lista completa. Isso pode ser visto na Figura 5-6 – Criação de Novo Projeto, passo 1, aonde é selecionada a opção "*Other...*", para que seja aberta a lista completa de opções.

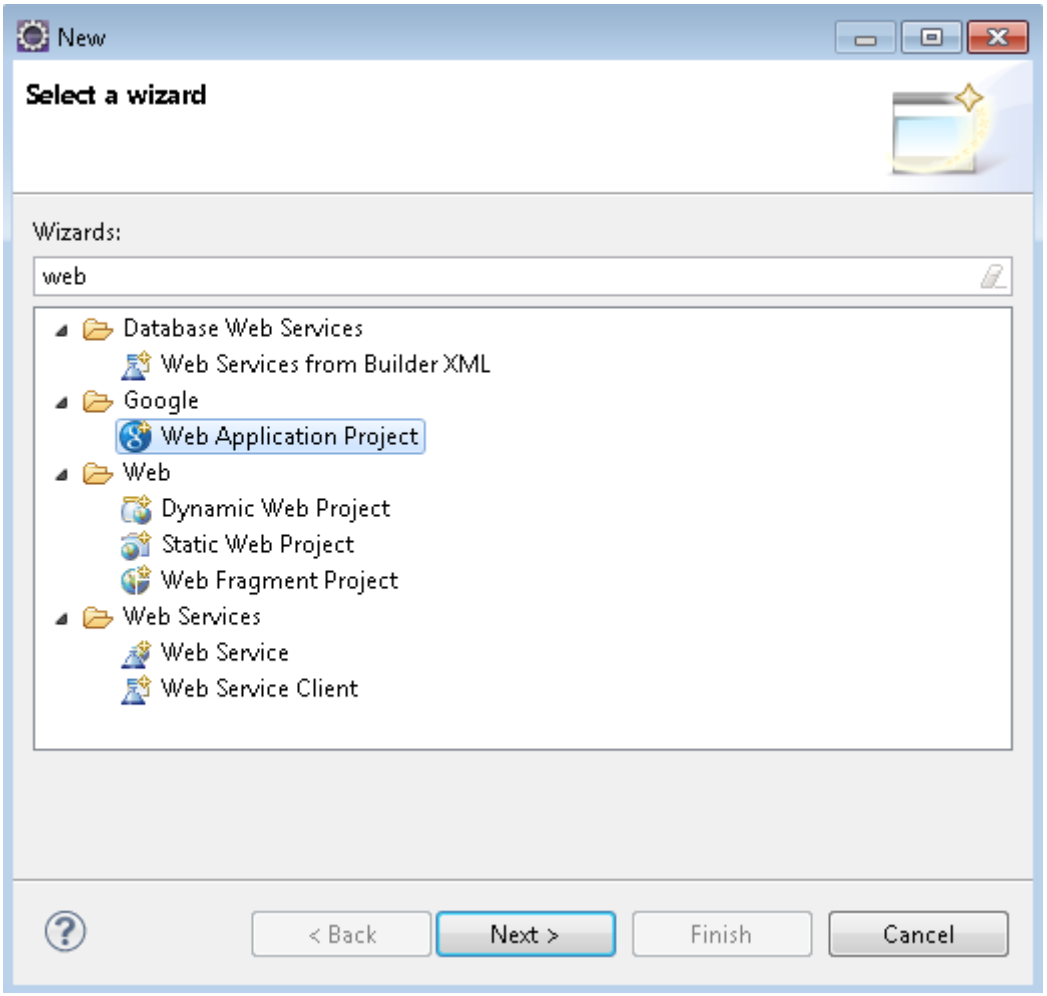
Figura 5-6 – Criação de Novo Projeto, passo 1



Fonte: elaborada pela autora

Após clicar nessa opção, é aberta uma janela com a lista completa de opções de projetos e arquivos que podem ser criados. Como a lista é grande, para facilitar, pode ser digitada a palavra "*web*" no campo de filtro existente. Isso faz com que a lista seja limitada apenas às opções que contém essa palavra no nome. A opção desejada para criação do projeto Google, é a encontrada dentro do item "Google": "*Web Application Project*" (Projeto de Aplicação *Web*, em tradução livre), conforme mostrado na Figura 5-7 – Criação de Novo Projeto, passo 2. Após a seleção, é necessário clicar no botão "*Next*", para prosseguir com a criação.

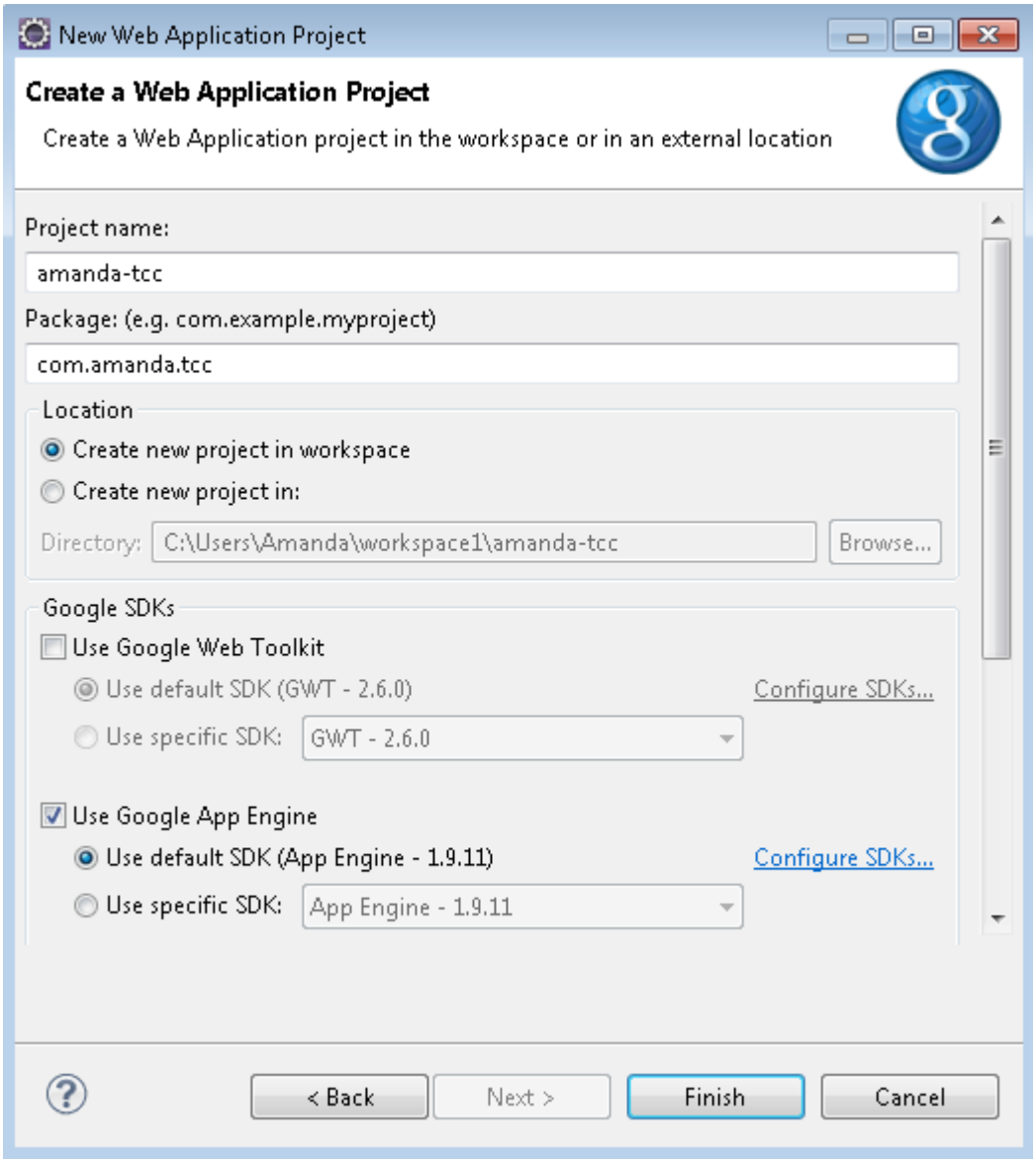
Figura 5-7 – Criação de Novo Projeto, passo 2



Fonte: elaborada pela autora

Quando essa opção é selecionada, é aberta uma janela na qual é necessário entrar com as configurações desejadas para o projeto. As mais importantes são o nome do projeto, o pacote principal em que os arquivos .java ficarão localizados. É importante também, certificar-se de que a opção de usar o Google App Engine está selecionada. Esse passo pode ser visto na Figura 5-8 – Criação de Novo Projeto, passo 3. Em seguida, é preciso clicar no botão "*Finish*" para concluir a criação.

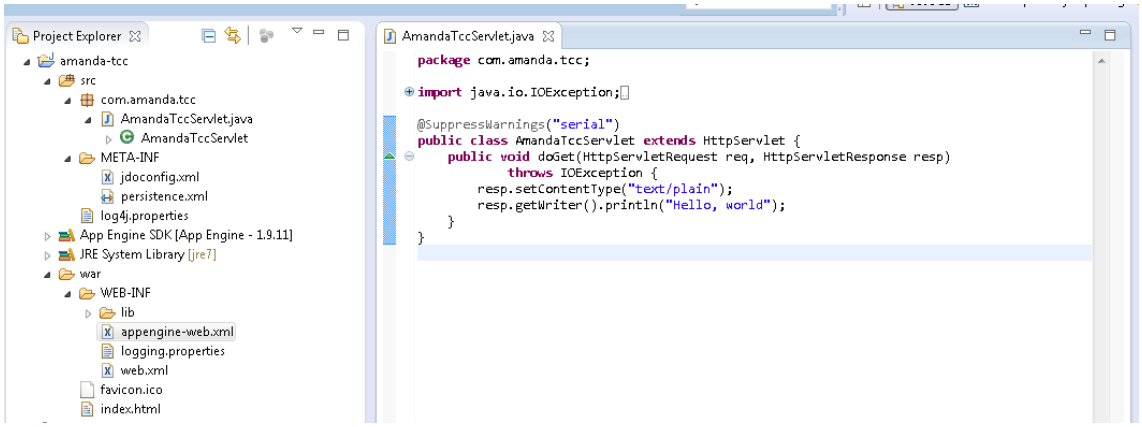
Figura 5-8 – Criação de Novo Projeto, passo 3



Fonte: elaborada pela autora

Finalmente, o projeto é criado. Como pode ser observado a seguir, na Figura 5-9 – Criação de Novo Projeto, passo 4, o *plugin* cria toda a estrutura necessária do projeto, além de um *Servlet* inicial, de exemplo, que exibe o tão conhecido "*Hello, world*" (traduzindo do Inglês, Olá, mundo) na tela, ao ser executado.

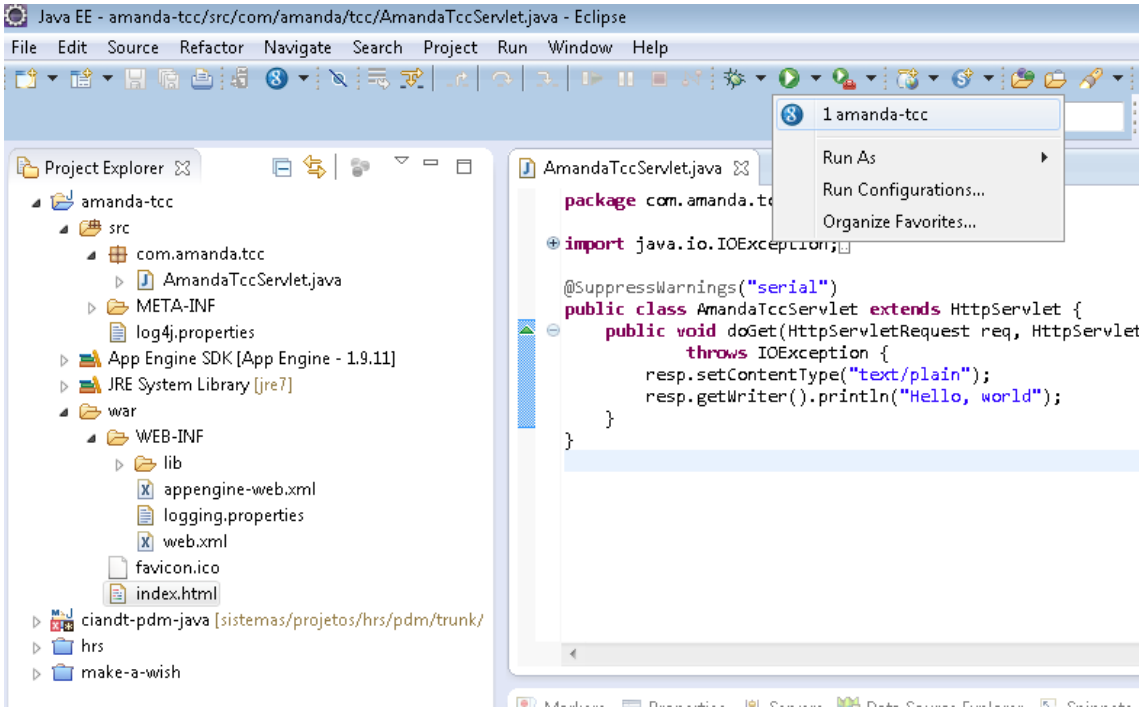
Figura 5-9 – Criação de Novo Projeto, passo 4



Fonte: elaborada pela autora

Para execução da aplicação localmente, na máquina do desenvolvedor, é só clicar na opção de execução, encontrada na barra superior da IDE, conforme pode ser visto na Figura 5-10 – Execução da Aplicação em Ambiente Local, passo 1.

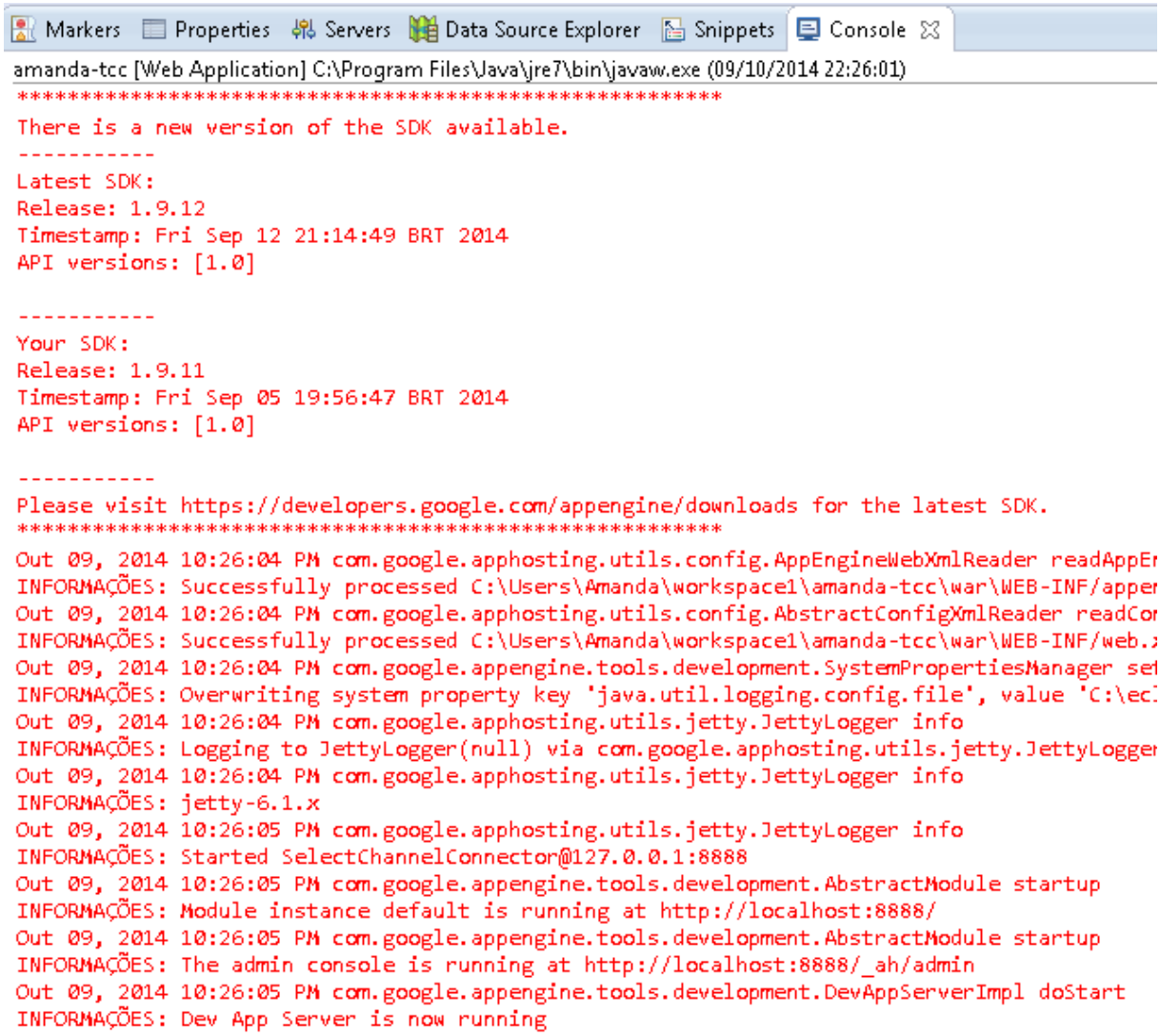
Figura 5-10 – Execução da Aplicação em Ambiente Local, passo 1



Fonte: elaborada pela autora

Ao ser executada, a aplicação exibe algumas informações, no console da IDE, como a versão instalada do SDK e a última disponível, e informações pertinentes à aplicação executada em si: se os arquivos de configuração são processados corretamente, o endereço para acessar a aplicação, e o endereço do armazenamento de dados. Neste caso, o endereço para acessar a aplicação é http://localhost:8888/. As informações do console podem ser observadas na Figura 5-11 – Execução da Aplicação em Ambiente Local, passo 2.

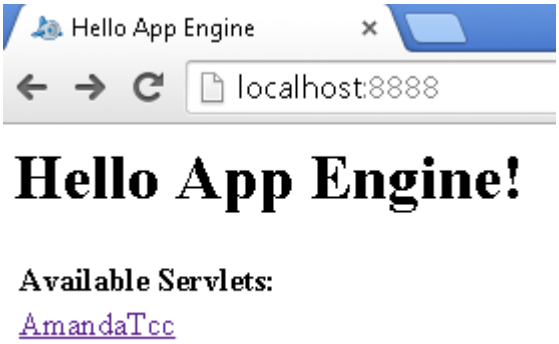
Figura 5-11 – Execução da Aplicação em Ambiente Local, passo 2



Fonte: elaborada pela autora

Ao acessar o endereço da aplicação local no navegador, é exibida a página vista na Figura 5-12 – Execução da Aplicação em Ambiente Local, passo 3. O conteúdo da página foi gerado pelo *plugin* de criação da aplicação, assim como o *Servlet* comentado anteriormente. Nesta página, são exibidos *links* para acessar os *Servlets* disponíveis; no caso, apenas o *Servlet* criado de exemplo é exibido, pois ele é o único existente.

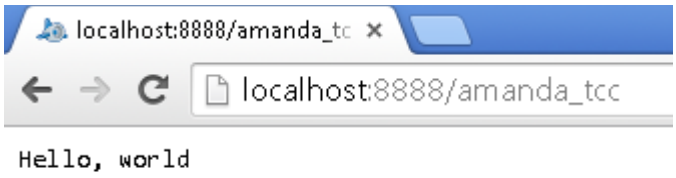
Figura 5-12 – Execução da Aplicação em Ambiente Local, passo 3



Fonte: elaborada pela autora

Ao clicar no *link* do *Servlet* disponível, a página é redirecionada para o conteúdo do *Servlet*, que apenas exibe a mensagem "*Hello, world*".

Figura 5-13 – Execução da Aplicação em Ambiente Local, passo 4

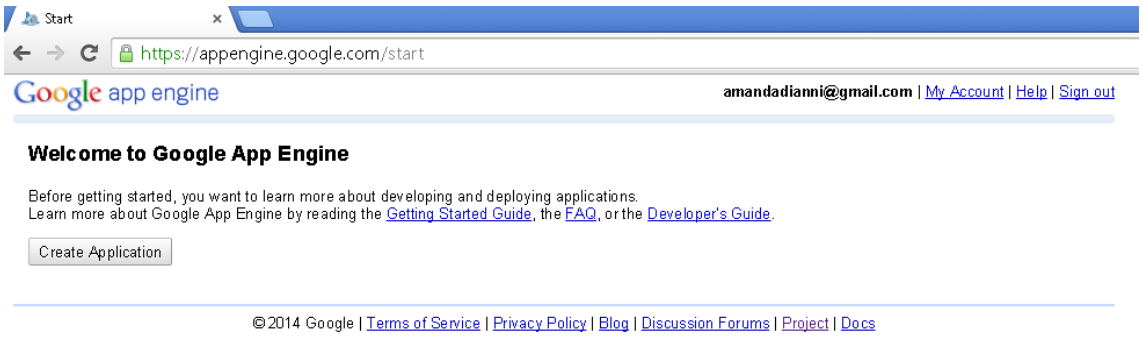


Fonte: elaborada pela autora

Outro passo muito importante na criação do projeto é a criação de uma aplicação no Google App Engine, pois é nela que a aplicação local será implantada. Para fazer isso, é preciso acessar o endereço https://appengine.google.com/, e

entrar com um usuário e senha de uma conta do Google, ou do Google Apps. Ao acessar esse endereço, é apresentada uma mensagem do Google com links úteis da documentação do App Engine, e um botão para criação da aplicação, vide Figura 5-14 – Criação de uma Aplicação no Google App Engine, passo 1.

Figura 5-14 – Criação de uma Aplicação no Google App Engine, passo 1



Fonte: elaborada pela autora

Ao clicar no botão "*Create Application*" (traduzindo do Inglês, Criar Aplicação), a página das figuras Figura 5-15 – Criação de uma Aplicação no Google App Engine, passo 2 e Figura 5-16 – Criação de uma Aplicação no Google App Engine, passo 3 é apresentada.

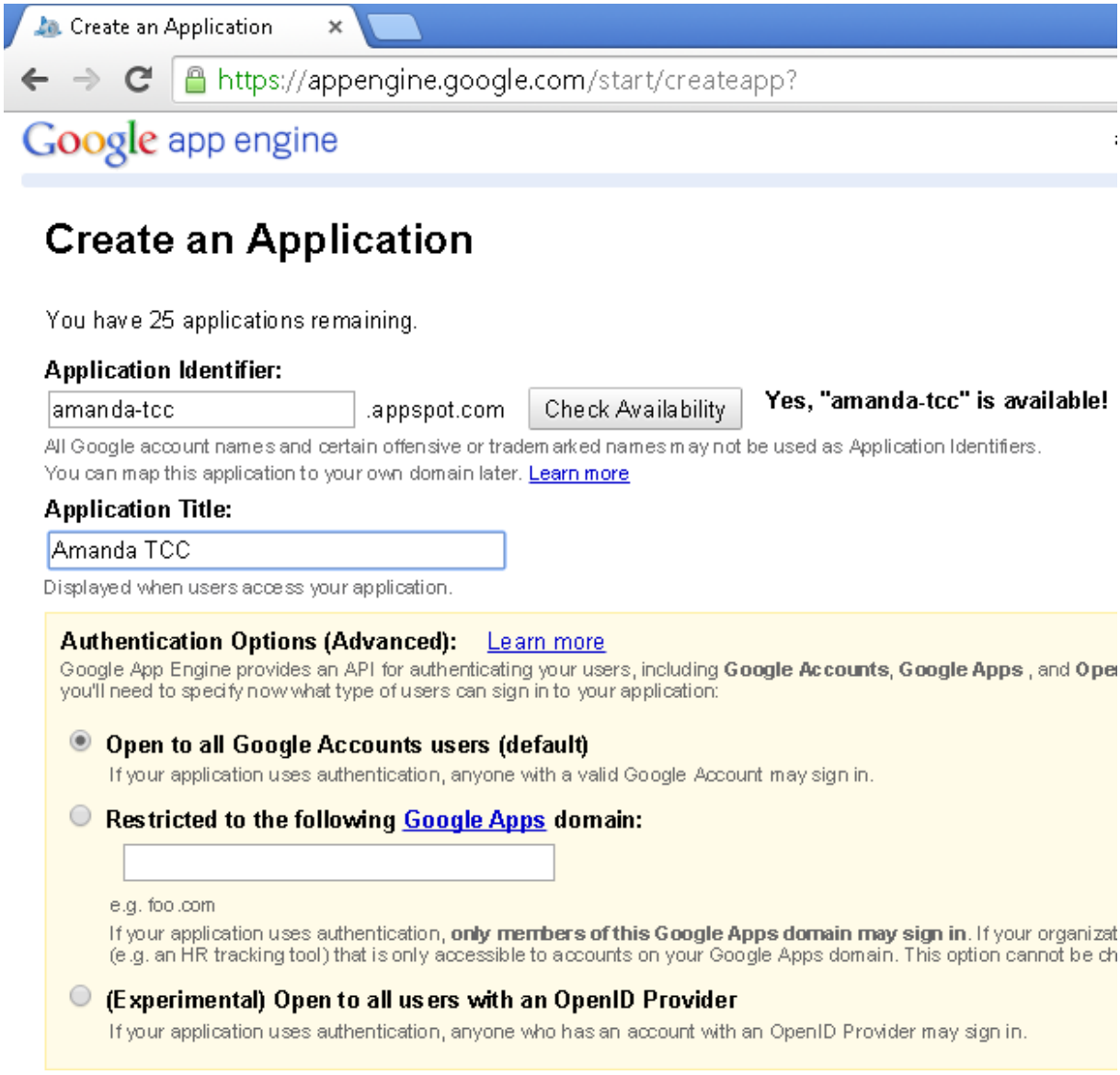
Na primeira parte são apresentados os campos para configuração da aplicação. No campo "*Application Identifier*" (traduzindo do Inglês, Identificador da Aplicação), deve ser colocado um identificador único para a aplicação. Esse identificador também fará parte do endereço da aplicação, seguido do mencionado domínio *.appspot.com*. Logo em frente a esse campo, existe um botão para verificar a disponibilidade do identificador. No caso deste trabalho, o identificador escolhido foi *amanda-tcc*, e como pode ser visto na imagem, este identificador está disponível. É importante ressaltar que o identificador não poderá ser alterado após a criação da aplicação.

O campo seguinte, "*Application Title*" (traduzindo do Inglês, Título da Aplicação), deve ser inserido o título que o desenvolvedor deseja que seja

apresentado quando os usuários acessarem sua aplicação. Este título pode ser alterado posteriormente.

Em seguida, é necessário escolher o tipo de autenticação para acesso à aplicação. Como visto anteriormente, os tipos existentes são: aberto para todos os usuários de contas do Google, restrito a um domínio do Google Apps, ou utilizando o provedor OpenID. Para este trabalho, o padrão será mantido, que é a opção de aberto para todos os usuários de contas do Google.

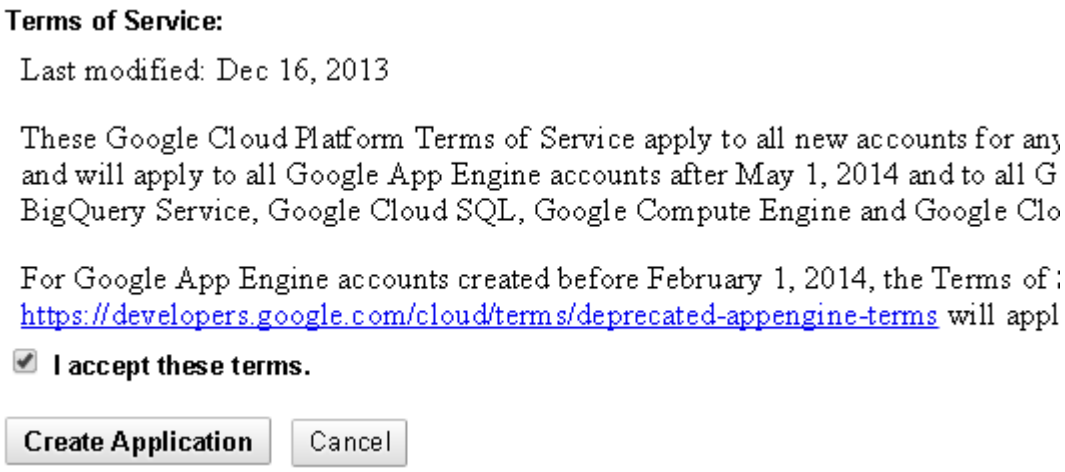
Figura 5-15 – Criação de uma Aplicação no Google App Engine, passo 2



Fonte: elaborada pela autora

E, finalmente, é necessário ler e aceitar os termos de serviço do Google App Engine, como mostrado abaixo, e clicar no botão “*Create Application*”.

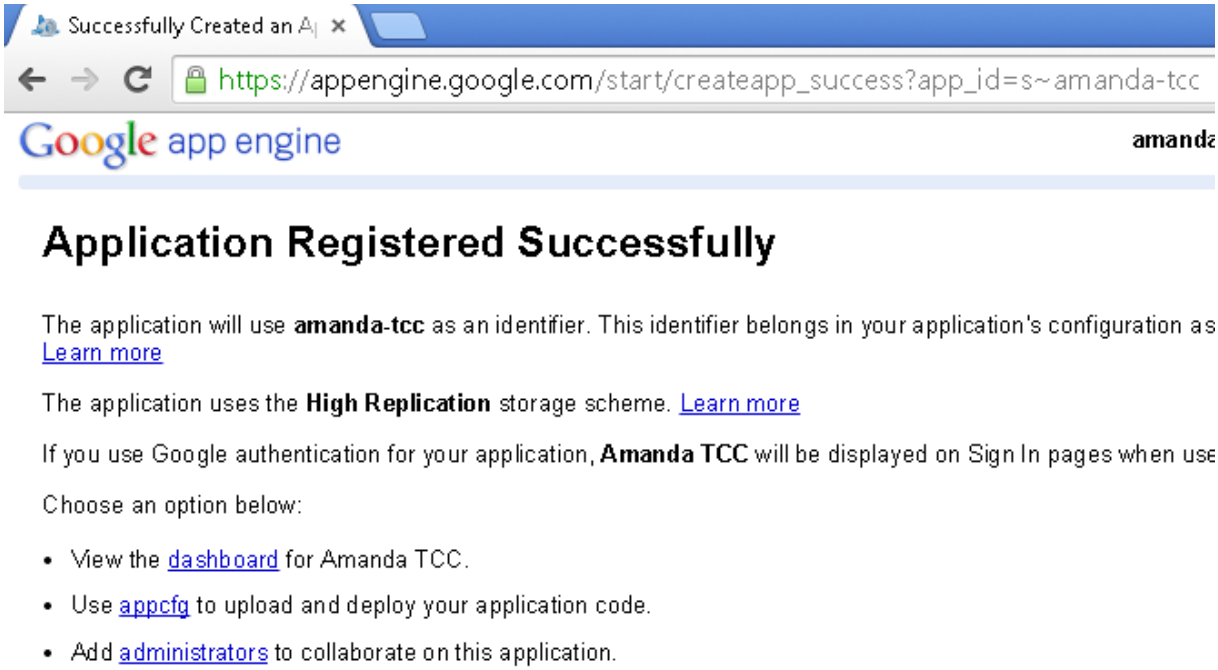
Figura 5-16 – Criação de uma Aplicação no Google App Engine, passo 3



Fonte: elaborada pela autora

Após isso, a aplicação é criada, e o navegador é redirecionado para uma página com uma mensagem de que a aplicação foi registrada com sucesso, e com diversos *links* úteis, como para o consolo de administração, e links para a documentação, etc.

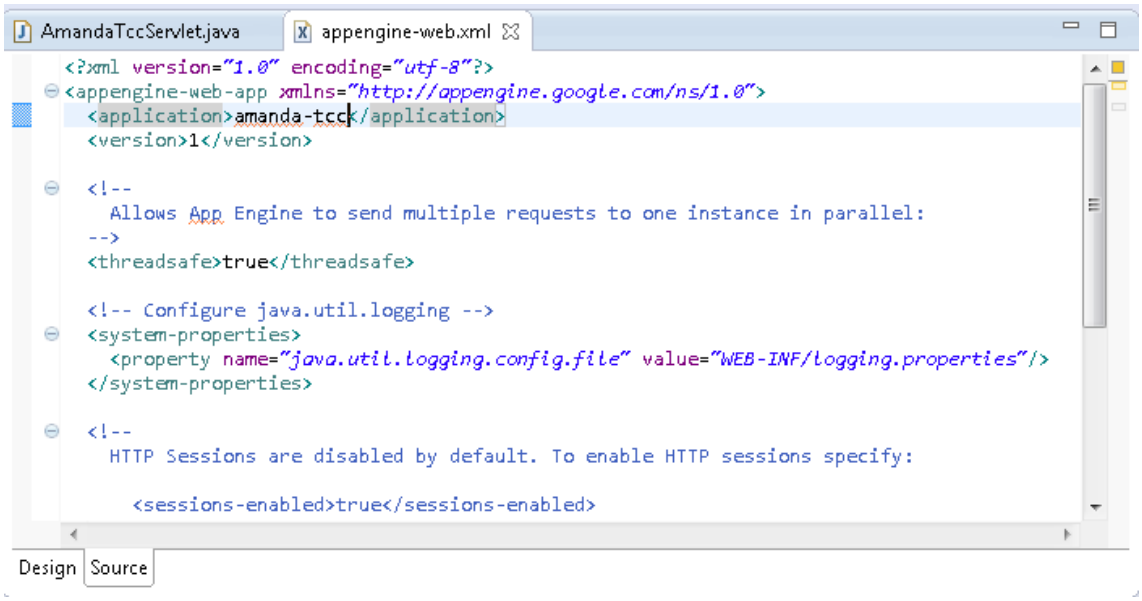
Figura 5-17 – Criação de uma Aplicação no Google App Engine, passo 4



Fonte: elaborada pela autora

Agora que a aplicação já foi criada em ambiente local, e foi registrada uma aplicação no Google App Engine, é necessário implantar a aplicação local no ambiente do Google. Para que isso seja feito, o primeiro passo é abrir o arquivo de configuração chamado *appengine-web.xml*, e alterar o parâmetro chamado "*application*". Neste parâmetro deve ser inserido o identificador da aplicação criado anteriormente; no caso, o identificador é amanda-tcc. O parâmetro seguinte, "*version*", corresponde à versão em que a aplicação será implantada, pois podem existir várias versões de uma mesma aplicação no Google App Engine. A versão 1 já vem na configuração por padrão, e será mantida. A configuração do arquivo pode ser observada na Figura 5-18 – Implantação da aplicação no Google App Engine, passo 1.

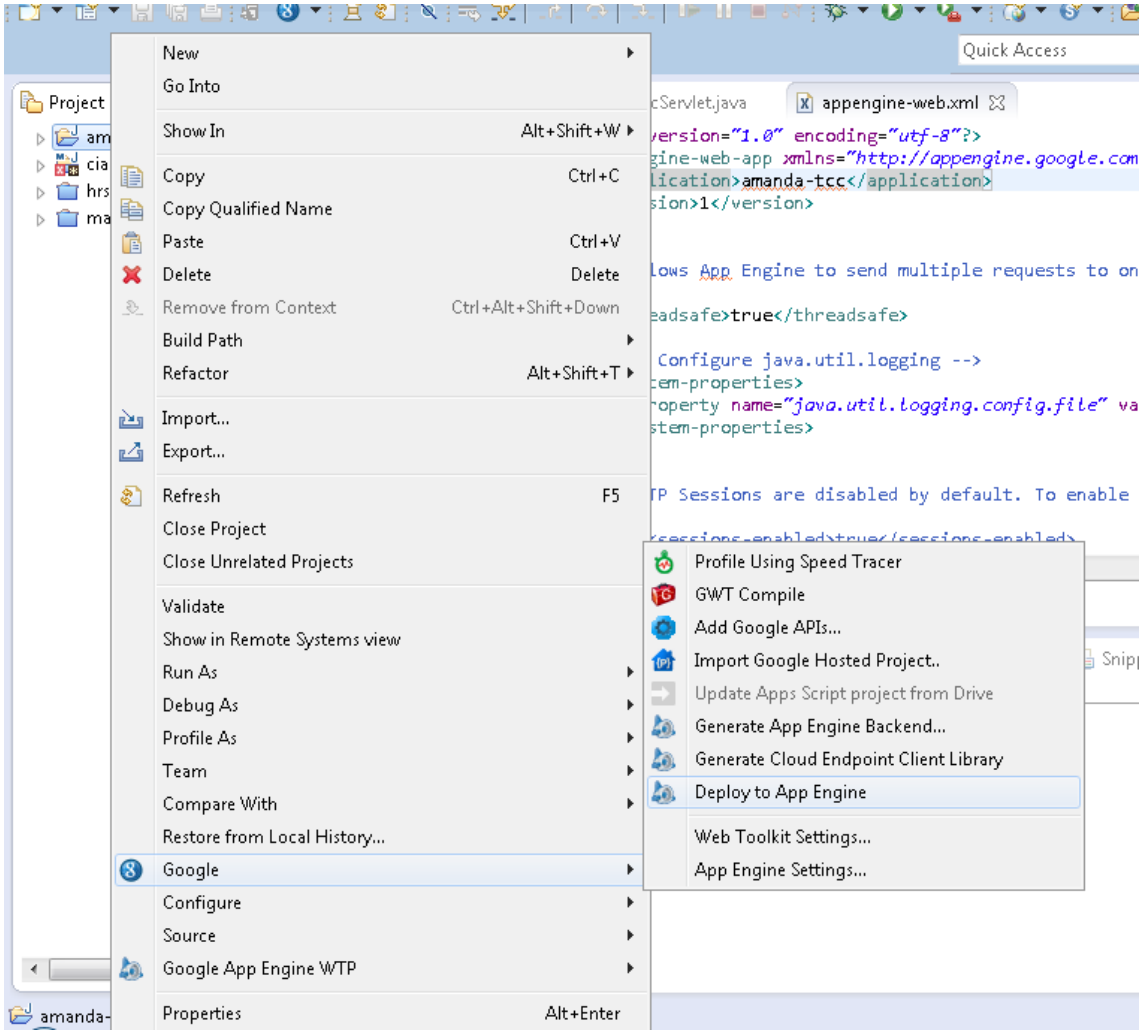
Figura 5-18 – Implantação da aplicação no Google App Engine, passo 1



Fonte: elaborada pela autora

Para efetuar a implantação através da IDE Eclipse, é necessário clicar com o botão direito em cima do diretório do projeto, na barra lateral esquerda, acessar o menu Google, e clicar na opção "*Deploy to App Engine*" (traduzindo do Inglês, Implementar no App Engine), como pode ser visto na Figura 5-19 – Implantação da aplicação no Google App Engine, passo 2.

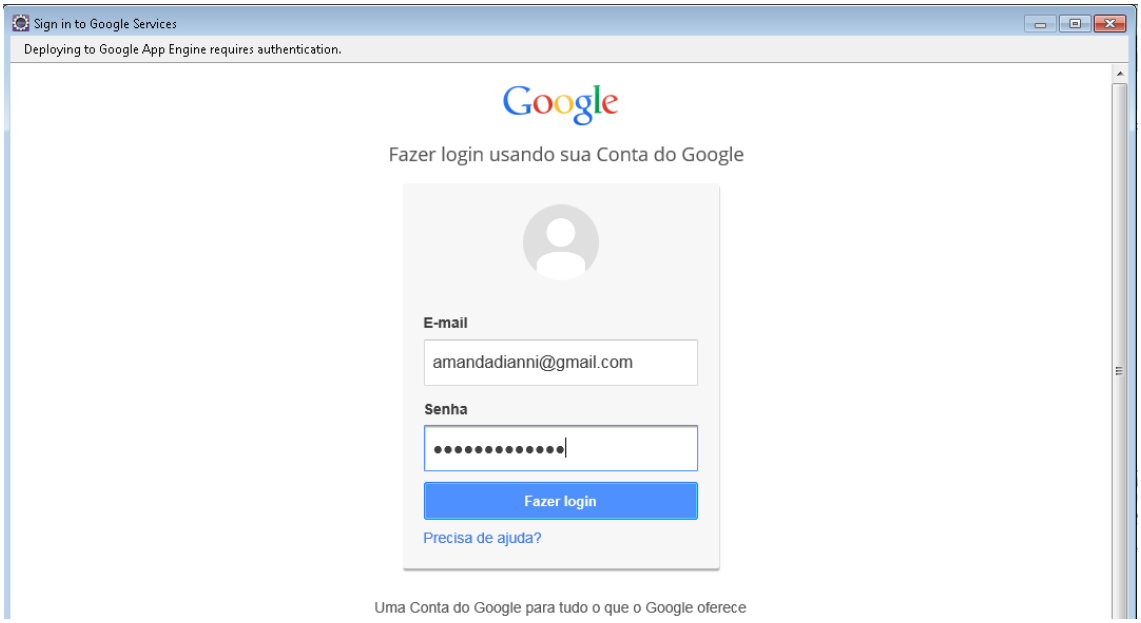
Figura 5-19 – Implantação da aplicação no Google App Engine, passo 2



Fonte: elaborada pela autora

Clicar nessa opção fará com que a IDE abra uma janela para autenticação do Google, para poder fazer a implantação, como mostrado na Figura 5-20 – Implantação da aplicação no Google App Engine, passo 3.

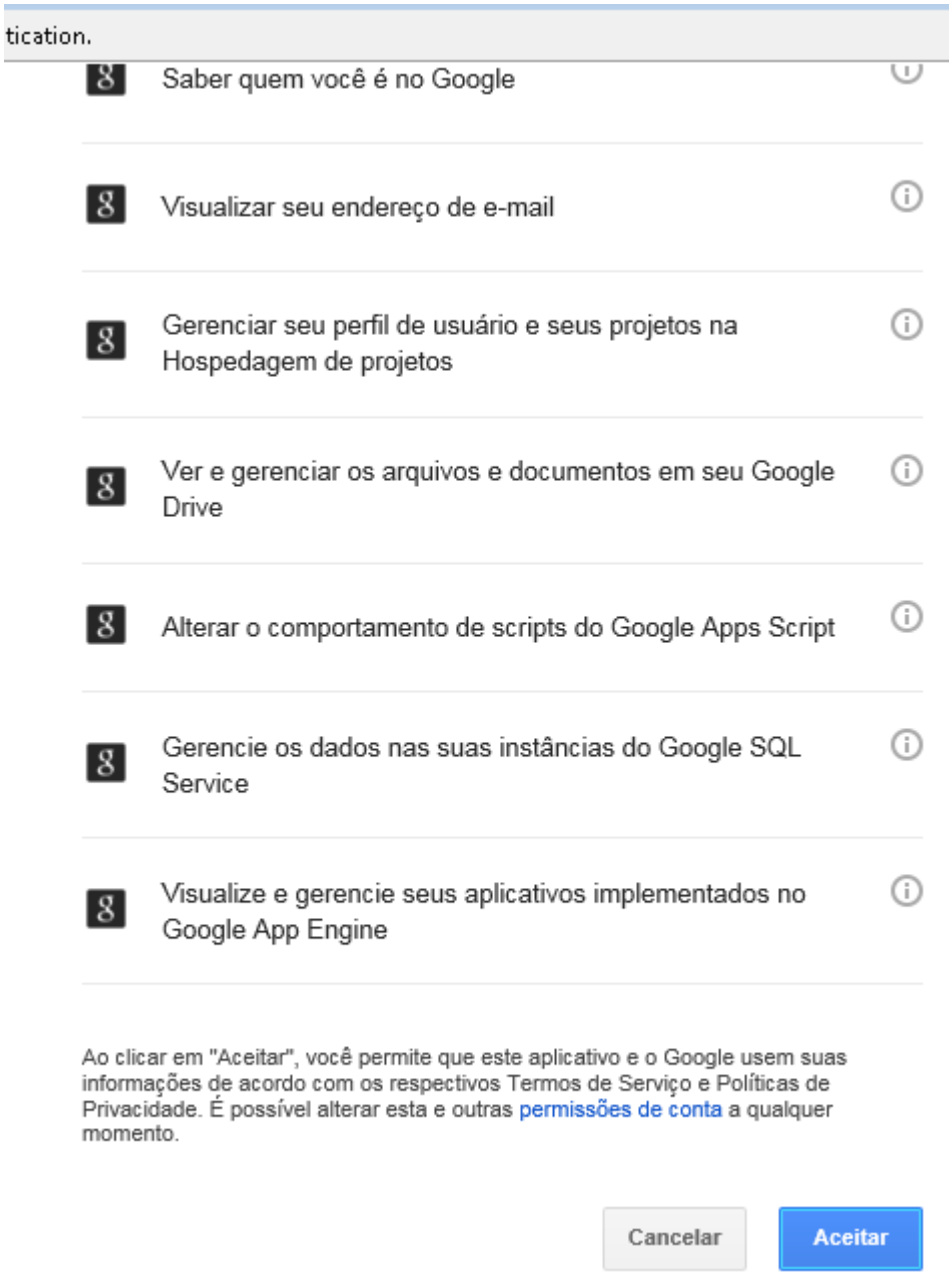
Figura 5-20 – Implantação da aplicação no Google App Engine, passo 3



Fonte: elaborada pela autora

Após entrar com o usuário e senha, o Google informa que é necessário aceitar que o Google e o App Engine usem as informações do usuário listadas. Isso pode ser observado na Figura 5-21 – Implantação da aplicação no Google App Engine, passo 4.

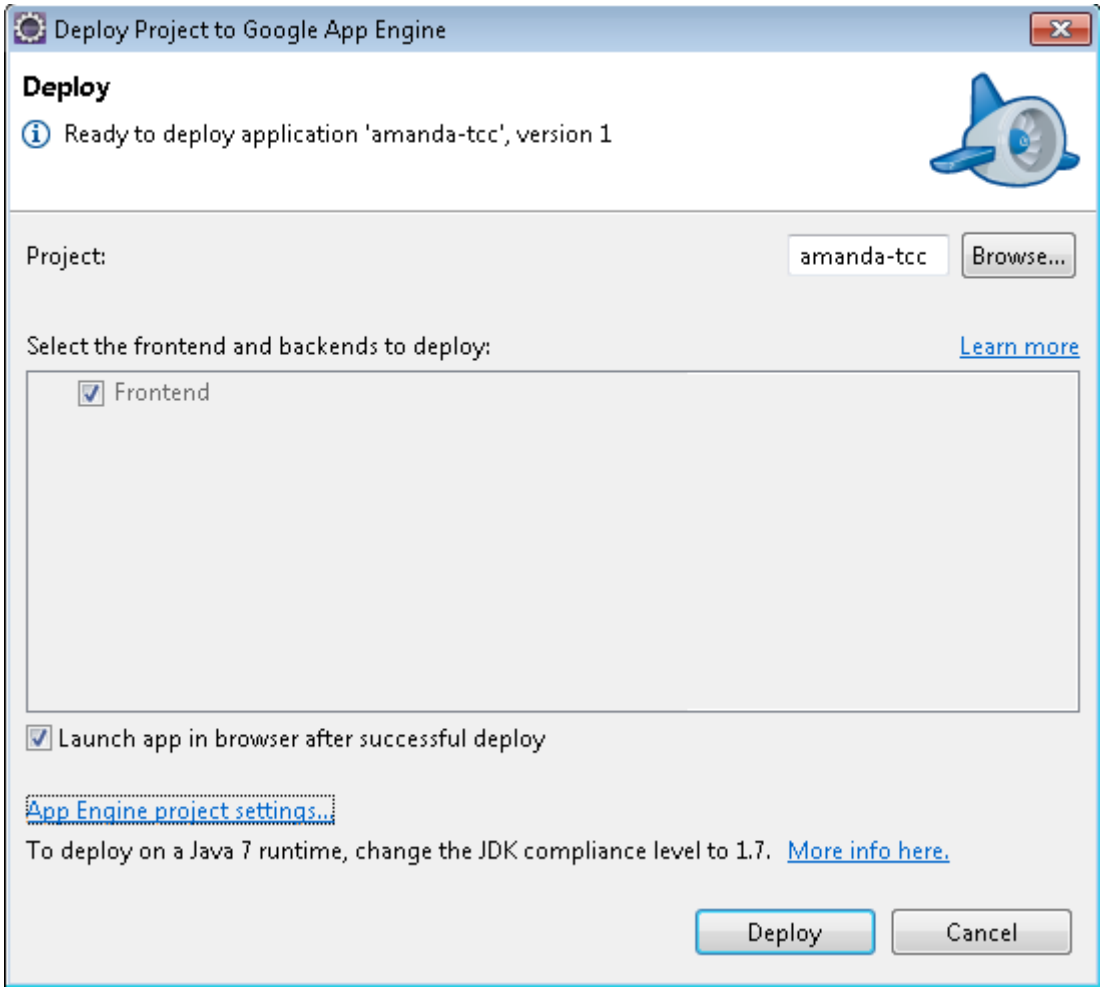
Figura 5-21 – Implantação da aplicação no Google App Engine, passo 4



Fonte: elaborada pela autora

Depois de aceitar os termos, uma janela é aberta, com informações sobre a aplicação a ser implantada, e o ambiente do Google App onde isso irá acontecer, conforme visto na Figura 5-22 – Implantação da aplicação no Google App Engine, passo 5.

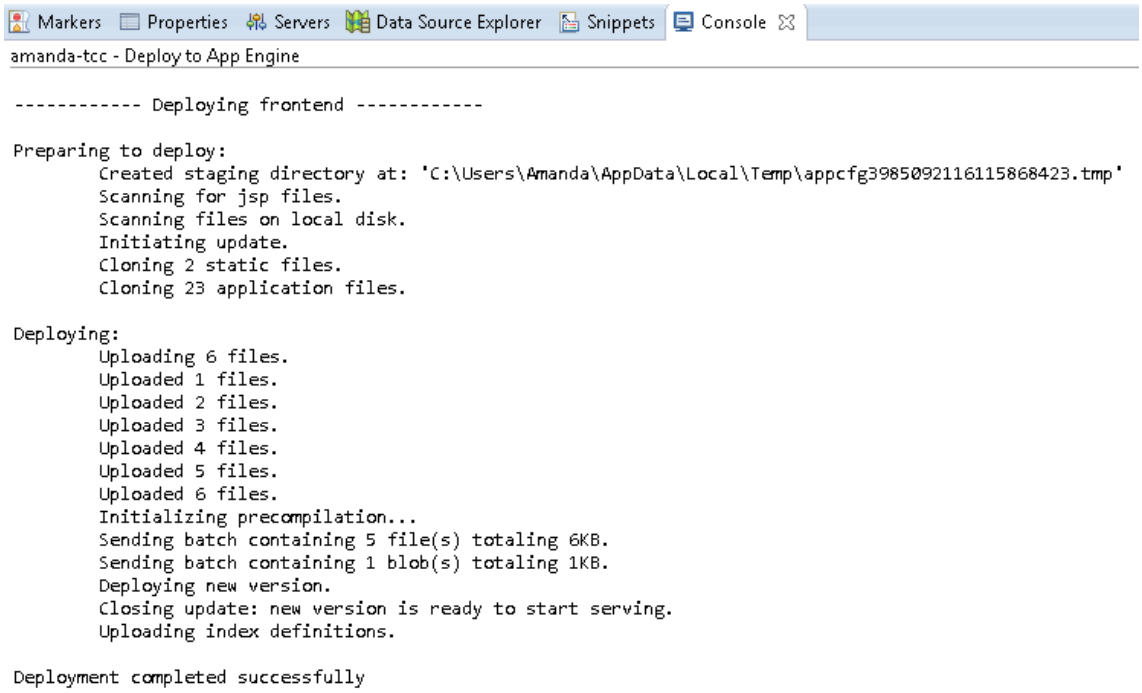
Figura 5-22 – Implantação da aplicação no Google App Engine, passo 5



Fonte: elaborada pela autora

Ao começar a implantação, o progresso da implantação é mostrado no console do Eclipse, como mostrado na Figura 5-23 – Implantação da aplicação no Google App Engine, passo 7.

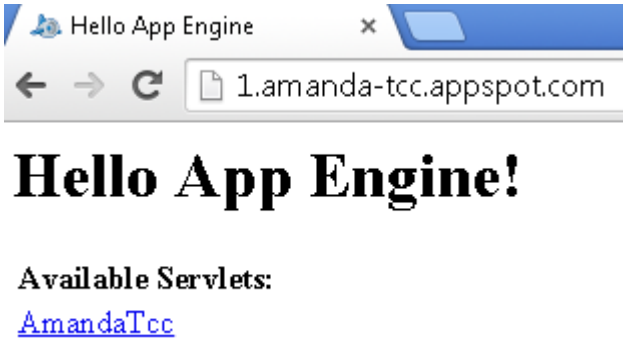
Figura 5-23 – Implantação da aplicação no Google App Engine, passo 7



Fonte: elaborada pela autora

Por fim, ao acessar o endereço da aplicação no Google App Engine, precedido pelo número da versão em que a aplicação local foi implantada, o resultado é uma página exatamente igual à página acessada no ambiente local. Isso pode ser observado na Figura 5-24 – Implantação da aplicação no Google App Engine, passo 8.

Figura 5-24 – Implantação da aplicação no Google App Engine, passo 8



Fonte: elaborada pela autora

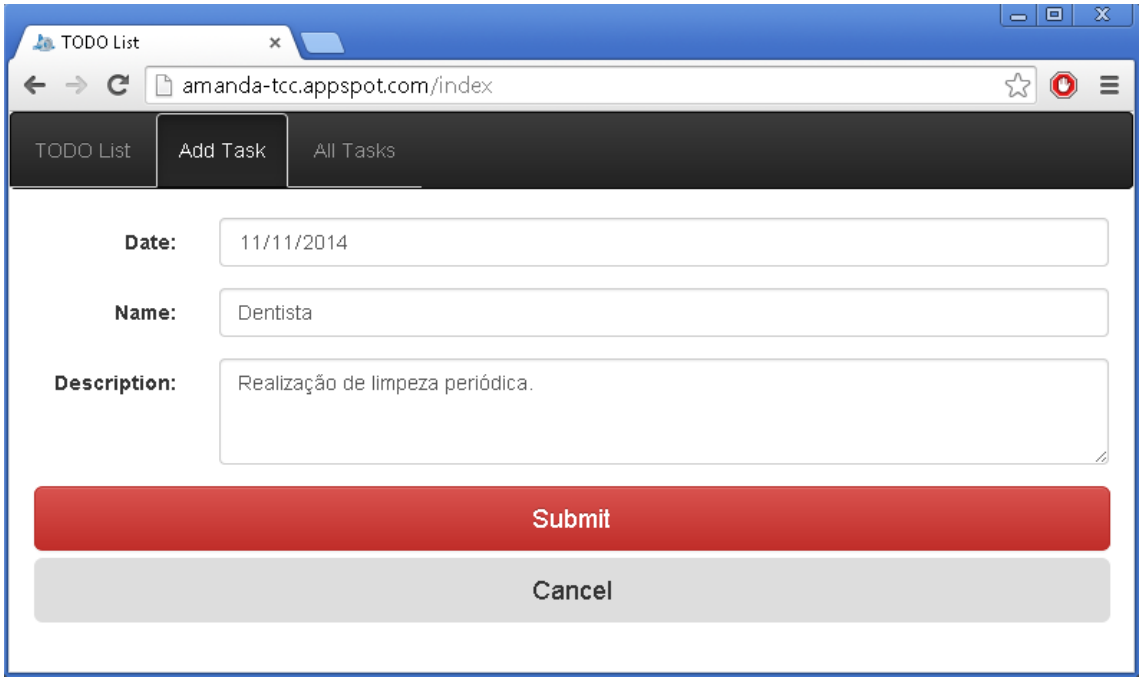
A APLICAÇÃO

Para este trabalho, a aplicação desenvolvida é uma lista diária de tarefas. O objetivo é que os usuários possam cadastrar suas tarefas, com as informações de nome da tarefa, um campo para descrição mais detalhada, e a data para realização da mesma, como mostra a Figura 5-25 – Lista de Tarefas: Inserir tarefa.

Então, a lista das tarefas que o usuário deve realizar no dia vai aparecer de acordo com a data escolhida na hora do cadastro. Nessa lista diária, ele pode marcar se a tarefa foi concluída, e, uma vez concluída, ela deixa de aparecer na lista diária. A lista diária é apresentada na Figura 5-26 – Lista de Tarefas: Tarefas do dia

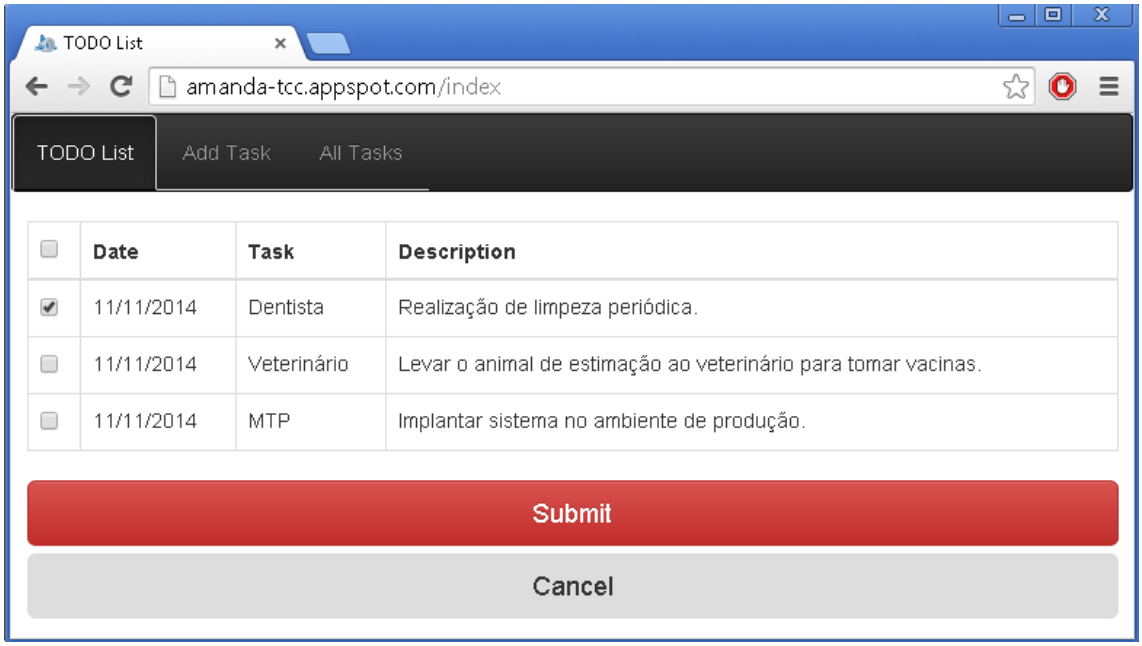
O usuário também consegue acessar uma lista com todas as tarefas já cadastradas por ele, como pode ser observado na Figura 5-27 – Lista de Tarefas: Todas as tarefas.

Figura 5-25 – Lista de Tarefas: Inserir tarefa



Fonte: elaborada pela autora

Figura 5-26 – Lista de Tarefas: Tarefas do dia



Fonte: elaborada pela autora

Figura 5-27 – Lista de Tarefas: Todas as tarefas

Fonte: elaborada pela autora

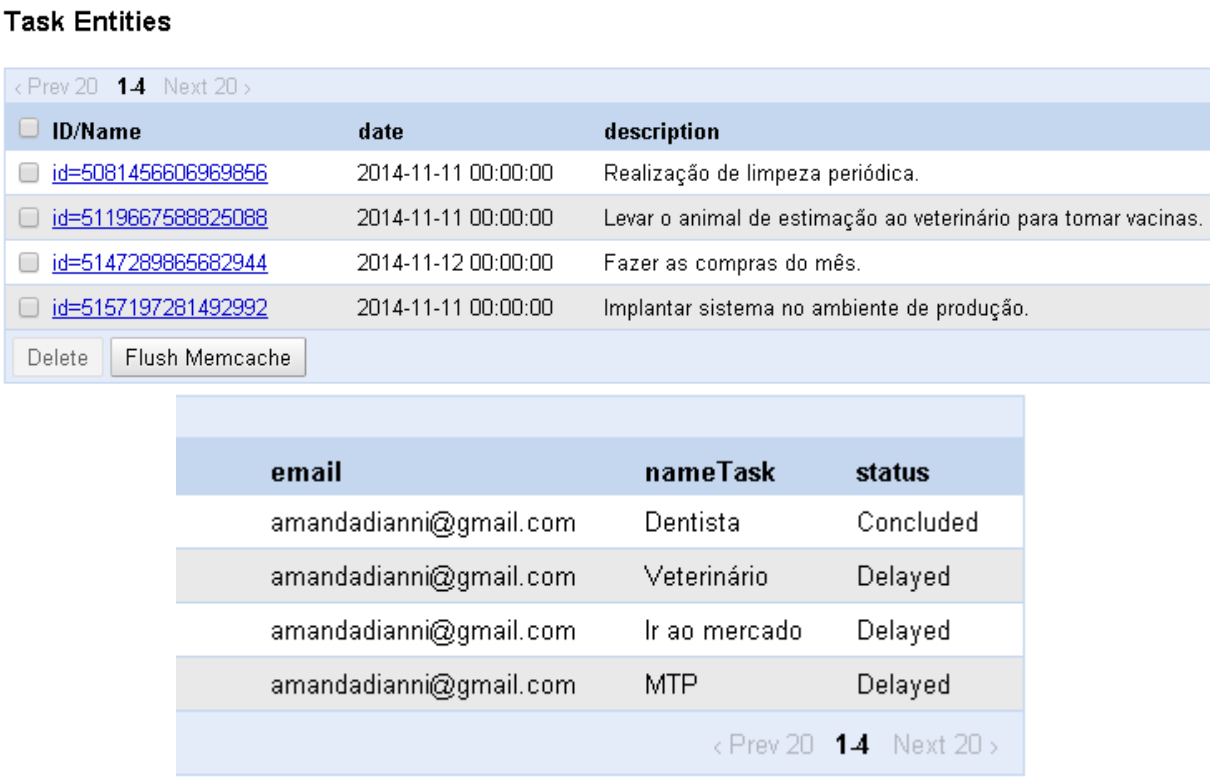
SERVIÇOS DO GOOGLE APP ENGINE UTILIZADOS NA APLICAÇÃO

Alguns dos serviços do Google App Engine vistos anteriormente foram utilizados neste trabalho. São eles:

- **DATASTORE**

  O serviço de armazenamento de dados oferecido pelo App Engine, o *Datastore*, foi utilizado para a persistência dos dados da aplicação. Para esta aplicação, existe apenas um conjunto de entidades, chamado *Task* (do inglês, Tarefa). Em cada entidade de tarefa, são armazenados todos os dados referentes às tarefas criadas pelo usuário: o nome, a descrição, a data, o estado da tarefa, e o endereço de *e-mail* do usuário que a criou. Estas entidades podem ser visualizadas através do console de administração do Google App Engine. Essa visualização é mostrada na Figura 5-28 – Lista de Tarefas: Armazenamento de dados

Figura 5-28 – Lista de Tarefas: Armazenamento de dados



Fonte: elaborada pela autora

- **MEMCACHE**

O serviço de cache de memória do Google App Engine foi utilizado na aba em que são mostradas todas as tarefas do usuário (aba *All Tasks*). Ele funciona da seguinte forma: quando o usuário acessa essa aba, o serviço busca no cache de memória se existe uma entrada no cache referente ao usuário que acessou a aplicação. Se existir, o serviço irá buscar essas tarefas do cache de memória; caso não exista, o serviço irá fazer a busca no armazenamento de dados, e então inserir o resultado da busca no cache de memória. Dessa forma, da próxima vez que o usuário acessar essa aba, o serviço irá encontrar os dados no cache de memória, sem a necessidade de uma nova busca no armazenamento de dados.

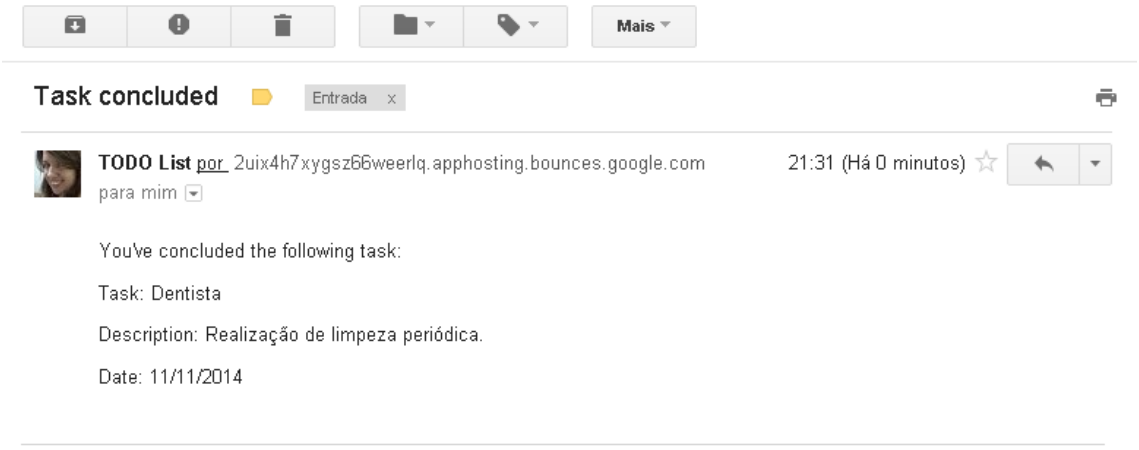
Quando o usuário faz o cadastro de uma nova tarefa, através da aba de cadastro (aba *Add Task*), ou marca a tarefa como concluída, através da aba de tarefas do dia (aba *TODO List*), o serviço que faz a inserção no

armazenamento de dados também irá remover a entrada referente ao usuário do cache de memória, visto que, após a nova inserção ou atualização, os dados em cache estarão desatualizados.

- **SERVIÇO DE E-MAIL**

O serviço de *e-mail* do Google App Engine foi utilizado na conclusão de uma tarefa. Em toda tarefa que o usuário marca como concluída, é enviado um *e-mail* ao usuário, informando que a tarefa foi concluída.

Figura 5-29 – Lista de Tarefas: Email de conclusão



Fonte: elaborada pela autora

- **CRON JOB**

O serviço de *Cron Job*, ou trabalhos agendados, foi utilizado para atualizar as tarefas atrasadas. Quando uma tarefa é criada, o estado padrão dela é *Pendent* (do inglês, Pendente), e após a sua conclusão, o estado passa a ser *Concluded* (do inglês, Concluída). Porém, se uma tarefa perde o prazo de conclusão, ela não deve continuar com o estado de pendente.

Para resolver este problema, um trabalho agendado para ser realizado todos os dias, às 1h, foi criado. Este trabalho agendado chama um serviço que busca no armazenamento de dados todas as tarefas com o estado pendente, mas que a data de realização esteja no passado, e atualiza o estado para *Delayed* (do inglês, Atrasada).

- **TASK QUEUE**

  O serviço de *Task Queue*, ou fila de tarefa, do Google App Engine foi utilizado no envio dos *e-mails*. Ao invés de o serviço que realiza a conclusão de uma tarefa enviar o e-mail diretamente, ele adiciona o *e-mail* em uma fila de tarefas. O serviço de fila de tarefas, por sua vez, irá realizar o envio do e-mail.

  A utilização dessa fila é importante neste caso, pois se ocorrer algum erro na hora de enviar o *e-mail*, ele não é removido da fila. Ele continuará na fila até ser enviado com sucesso, ou até ser removido manualmente da fila.

6. CONSIDERAÇÕES FINAIS

O Google App Engine é uma boa plataforma para o desenvolvimento de aplicações na nuvem, pois com ele as aplicações podem ser facilmente desenvolvidas e hospedadas nos servidores do Google, sem que o desenvolvedor tenha a preocupação com a infraestrutura da hospedagem. Os custos são baseados na utilização dos recursos dos servidores do Google, gerando um custo baixo para aplicações de pequeno porte, e de pouco acesso.

Porém, o desenvolvedor precisa ter em mente que aplicações de grande porte, que recebem um alto número de acessos e requisições, vão gerar um alto custo. Nestes casos, também, é possível que os serviços oferecidos pelo Google App Engine não atendam a todas as necessidades da aplicação, e dessa forma será necessário utilizá-lo integrado com os outros serviços do Google na nuvem, o que também irá gerar mais custo.

O desenvolvimento de aplicações utilizando o Google App Engine é muito simples, o que pode também influenciar na escolha de utilizá-lo ou não. Outras vantagens, como escalabilidade automática, desempenho e segurança são pontos importantes a serem considerados, e influenciam positivamente na tomada de decisão.

Estudos futuros sobre a expansão da utilização do Google App Engine, através da integração dele com outros serviços do Google da nuvem, podem ser realizados para melhor entendimento das aplicações em que a escolha de utilizá-lo é ideal, ou a mais recomendada.

Com base nas pesquisas realizadas para o desenvolvimento deste trabalho, conclui-se que as necessidades da aplicação devem ser consideradas e equiparadas com os serviços disponibilizados pelo Google App Engine, antes de optar por utilizá-lo ou não. As limitações da plataforma devem ser levadas em consideração, pois utilizá-la de maneira incorreta, ou para o desenvolvimento de uma aplicação que ela não atende a todas as necessidades pode gerar frustração ao desenvolvedor. Porém, utilizá-la da maneira correta, e para as aplicações certas será, da mesma forma, gratificante.

7. REFERÊNCIAS BIBLIOGRÁFICAS

AMAZON, 2014. **Amazon EC2**. Desenvolvido por: Amazon. Disponível em: <http://aws.amazon.com/pt/ec2/>. Acesso em: 13 de Novembro de 2014.

Apps Documentation & Support, 2014. **About Google App Engine**. Desenvolvido por: Google. Disponível em: <http://support.google.com/a/bin/answer.py?hl=en&answer=91077>. Acesso em: 05 de Maio de 2014.

COSTA, Marcelo. **PostgreSQL: Armazenamento de dados em formato "schemaless".** Disponível em: <http://www.infoq.com/br/articles/postgresql-schemaless/>. Acesso em: 30 de Setembro de 2014.

GONÇALVES, Antonio Ricardo. **O que é SaaS, IaaS e PaaS em Cloud Computing? (Conceitos básicos)**. Disponível em: <http://antonioricardo.org/2013/03/28/o-que-e-saas-iaas-e-paas-em-cloud-computing-conceitos-basicos/>. Acesso em: 20 de Maio de 2014.

GOOGLE, 2014. **Google Cloud Platform**. Desenvolvido por: Google. Disponível em: <https://cloud.google.com/>. Acesso em: 20 de Outubro de 2014.

GOOGLE DEVELOPERS, 2014. **Google App Engine**. Desenvolvido por: Google. Disponível em: <https://developers.google.com/appengine/>. Acesso em: 05 de Maio de 2014.

GUILHERME, Paulo. **O que é Sandbox?**. Disponível em: <http://www.tecmundo.com.br/spyware/1172-o-que-e-sandbox-.htm/>. Acesso em: 15 de Setembro de 2014.

MICROSOFT, 2014. **Microsoft Azure**. Desenvolvido por: Microsoft. Disponível em: <http://azure.microsoft.com/pt-br/>. Acesso em: 13 de Novembro de 2014.

PEDROSA, Paulo H. C.; NOGUEIRA, Tiago. **Computação em Nuvem**. Disponível em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2011/T2/Artigos/G04-095352-120531-t2.pdf>. Acesso em: 7 de Outubro de 2014.

SANDERSON, Dan. **Programming Google App Engine**. O'Reilly Media, Inc., 2010.

TAURION, Cezar. **Um pouco do GAE (Google Application Engine)**. Disponível em: <http://computingonclouds.wordpress.com/2010/02/23/um-pouco-do-gae-google-application-engine/>. Acesso em: 11 de Junho de 2014.