

# PLATAFORMA COMO SERVIÇO: UM ESTUDO DE CASO UTILIZANDO O GOOGLE APP ENGINE (GAE)

**SAMUEL JOSE PINOTTI** – pinotti.samuel@gmail.com

**Trabalho orientado pelo Prof. Me. Rodrigo Malara**

**Resumo:** Este artigo apresenta o *SMK System*, um sistema desenvolvido para a *web* que possibilita ao usuário controlar o estoque, cadastrar produtos e clientes, realizar vendas e gerar relatórios. Através desse estudo de caso será demonstrado na prática como utilizar o conceito de Plataforma como Serviço (PaaS), no caso a plataforma Google App *Engine*, também conhecida pela sigla GAE. O sistema foi desenvolvido de forma objetiva, utilizando conceitos de Engenharia de *Software*, como a linguagem de modelagem UML, a linguagem de programação Java e para o armazenamento dos dados, o gerenciador de bancos de dados MySQL.

**Palavras-chave:** Plataforma, PaaS, Google, App, *Engine*, GAE, Java.

**Abstract:** This article presents *SMK System*, a developed web system that allows users to control the inventory, register products and clients, make sales and generate reports. Through this case study will be demonstrated in practice how to use the concept of Platform as a Service (PaaS), in this case the Google App Engine platform, also known by acronym GAE. The system was developed in an objective way, using concepts of Software Engineering, as UML modeling language, Java programming language and for storage, MySQL database manager.

**Keywords:** Platform, PaaS, Google, App, *Engine*, GAE, Java.

## 1 INTRODUÇÃO

A nuvem transformou a forma de criar softwares, acelerando a maneira como os desenvolvedores trabalham, seja em grandes corporações ou pequenas fábricas de *software*. Os desenvolvedores estão enfrentando cronogramas e orçamentos cada vez mais apertados, o que os levam a adotar a nuvem.

Para um desenvolvedor, nuvem é um conjunto de tecnologias fundamentais que permitem

novas formas de construções de outras tecnologias. Então, desenvolvedores estão migrando para Plataforma como Serviço, podendo realizar seu trabalho mais rápido e melhor, ou seja, mais rápido por que gastam menos tempo com configurações e gerenciamento de servidores, e melhor, pois permite implementar melhores prática. (Carlson, 2013).

A computação em nuvem tem contribuído para otimizar o tempo hábil dos desenvolvedores, pois provê ambientes completos de desenvolvimento com hardware, sistema operacional e por vezes, plataformas completas pré-configuradas. Além disso, a computação em nuvem pode oferecer *softwares* prontos para utilização, atendendo também usuários finais.

Este trabalho apresenta a implementação de um sistema de controle de estoque e vendas utilizando um dos conceitos da computação em nuvem, a Plataforma como Serviço, também conhecida pela sigla PaaS. Esse conceito fornece aos desenvolvedores ambientes de desenvolvimento pré-configurados, reduzindo problemas de configurações como *hardware* e sistema operacional, além de tornar a aplicação facilmente escalável.

## 1.1 Objetivo

O objetivo deste trabalho é abordar a utilização da plataforma como serviço, através do desenvolvimento do sistema *SMK System*, construído sobre a plataforma *Google App Engine*, mostrando algumas das tecnologias suportadas por esse ambiente, suas vantagens e desvantagens.

## 1.2 Justificativa

Um Microempreendedor Individual apresentou a necessidade de otimizar o controle de estoque e vendas de seus produtos de forma compartilhada e com custo reduzido.

Após uma análise rápida de algumas soluções disponíveis no mercado para hospedagem de aplicações, o conceito de Plataforma como Serviço apresentou-se bastante interessante, principalmente pelas facilidades na configuração de ambientes de desenvolvimento, e praticamente nenhuma configuração necessária nos ambientes de produção.

Uma nova análise foi realizada, agora voltada para as soluções disponíveis no mercado de Plataformas como Serviço. Optou-se pela plataforma *Google App Engine* ou GAE, principalmente pelas facilidades propostas no *deploy* de aplicações, forma de cobrança e gerenciamento de disponibilidade.

### 1.3 Problema e Hipótese da Pesquisa

Ter acesso às informações relevantes do seu negócio em tempo hábil se tornou um fator importante para manter a competitividade em qualquer seguimento de mercado. Por isso, existe hoje uma grande necessidade de se obter agilidade nas entregas de aplicações, sejam elas corporativas ou de outros seguimentos.

O estudo proposto nesse trabalho, mostrará na prática um pouco do conceito de Plataforma como Serviço, e como este pode contribuir para a otimização do tempo do desenvolvedor, fazendo com que esse fique focado em sua atividade principal, que é desenvolver a aplicação propriamente dita.

Os reais benefícios dessa abordagem são muito relativos, pois dependem da escolha das tecnologias empregadas e da proficiência dos desenvolvedores. O que será mostrado, é uma implementação utilizando exemplos de algumas das inúmeras tecnologias suportadas por essa plataforma.

### 1.4 Metodologia de Pesquisa

Este estudo pode ser considerado de natureza tecnológica, com foco na apresentação prática de conceitos de Plataforma como Serviço.

Pesquisas bibliográficas foram realizadas, referenciando nomes como, Desenvolvendo Aplicações com UML 2.0 (Ana Cristina Melo), livros da série Use a Cabeça!, *Programming for PaaS* (Lucas Carlson), *Maven: The Definitive Guide (Sonatype<sup>1</sup>)* além de documentações do *Apache Maven Project<sup>2</sup>* e *Google App Engine<sup>3</sup>*.

Nesse trabalho foram analisados alguns benefícios obtidos em utilizar a plataforma *Google App Engine*. Isso poderá contribuir em decisões futuras para a escolha desta ou de outra plataforma.

Para o início da construção da aplicação, houve uma etapa inicial de levantamento de requisitos, que ajudou a direcionar o processo de desenvolvimento.

## 2 REVISÃO BIBLIOGRÁFICA

Nesta seção, serão descritos os conceitos e tecnologias empregadas para construção do

---

<sup>1</sup> <https://www.sonatype.com/>

<sup>2</sup> <https://maven.apache.org/what-is-maven.html>

<sup>3</sup> <https://cloud.google.com/appengine/docs/java/runtime>

sistema e desenvolvimento de todo o trabalho.

## 2.1 UML

Desde os primeiros conceitos de orientação a objeto, surgiram vários métodos de representação gráfica para a comunidade. Segundo Melo (2004), mais de 50 métodos foram apresentados entre os anos de 1989 e 1994, porém, a grande maioria errou ao tentar estender os métodos estruturados, prejudicando assim os usuários, que não encontravam uma completa satisfação utilizando o que havia disponível. Aproximadamente em meados dos anos 90 surgiram novas abordagens focadas em trabalhar mais ativamente na ideia do novo paradigma. Então, em 1996 surgiu a UML (*Unified Modeling Language*) ou Linguagem Unificada de Modelagem, a partir da união dos métodos de três dos nomes mais importantes da época, James Rumbaugh, Grady Booch e Ivar Jacobson.

Ainda segundo Melo (2004), a UML não é um método, e sim uma linguagem de modelagem que, através de sua estrutura, conduz a criação e leitura de seus modelos, mas não determina quais e nem quando esses modelos precisam ser criados, pois essa é uma responsabilidade do processo de desenvolvimento. A UML é uma linguagem para especificação, visualização, construção e documentação de artefatos do sistema.

A UML oferece diversos diagramas que podem ser utilizados em qualquer tipo de sistema. Para o *SMK System* foi utilizado o Diagrama de Casos de Uso que apresenta os casos de uso, atores e seus relacionamentos, que demonstram as funcionalidades do sistema. Também foram utilizadas algumas técnicas apresentadas na UML para levantamento de requisitos. Isso originou um documento de requisitos e um documento de casos de uso textuais.

## 2.2 MySQL

De acordo com MySQL - MySQL 5.7 *Reference Manual* - 1.3.1 *What is MySQL?* (2016), um banco de dados é um conjunto de dados estruturados, podendo ser desde uma lista de compras presa na geladeira, até um grande volume de informações de uma rede corporativa. Para adicionar, acessar e processar os dados armazenados em um banco de dados é necessário um sistema de gerenciamento de banco de dados como o MySQL, que foi a ferramenta escolhida para este trabalho. Ele é o mais popular gerenciador de banco de dados SQL *Open Source*,

distribuído e apoiado pela *Oracle Corporation*<sup>4</sup>. MySQL é um banco de dados relacional que armazena dados em tabelas separadas, tornando o ambiente de desenvolvimento mais flexível, também é *Open Source*, que quer dizer que qualquer pessoa pode utilizar e modificar o *software*. Por isso, o MySQL pode ser baixado e instalado sem ter que pagar licença. Se desejar, você ainda pode estudar o código fonte e alterá-lo.

Por ser desenvolvido nas linguagens C e C++, o MySQL é facilmente portátil a vários sistemas e plataformas. Também suporta vários tipos de dados como *float*, *double*, *char*, *varchar*, *binary*, *varbinary*, *text*, *blob*, *date*, *time*, *datetime*, *timestamp*, *year*, *set* e *enum*. Sobre escalabilidade e limites, existem usuários que utilizam o MySQL *Server* com mais de 200.000 tabelas e mais de 5.000.000.000 de linhas. Ele suporta até 64 índices por tabela, podendo cada índice consistir de 1 a 16 partes de colunas ou colunas. Para o desenvolvimento, ele ainda fornece ferramentas de linha de comando como *mysqldump*<sup>5</sup> e *mysqladmin*<sup>6</sup> além da IDE MySQL *Workbench*<sup>7</sup>. (MySQL - MySQL 5.7 Reference Manual - 1.3.2 *The Main Features of MySQL*, 2016).

### 2.3 Apache Maven

Segundo *Sonatype* (2008) a definição de Maven depende da perspectiva do usuário. A grande maioria dos usuários dizem que Maven é uma ferramenta de construção, usada para construir implementações de códigos fonte. Engenheiros e gerentes de projetos referem-se ao Maven de forma mais abrangente, como uma ferramenta de gerenciamento de projetos. A questão é que, assim como o *Ant*<sup>8</sup>, o Maven é capaz de compilar, empacotar, realizar testes e distribuir projetos, mas, além disso, também é capaz de executar relatórios gerando um *web site* para facilitar a comunicação entre os membros do projeto. Por isso, uma melhor definição para o Apache Maven é, uma ferramenta de gerenciamento de projetos que engloba um conjunto de padrões, um ciclo de vida do sistema, gerenciamento de dependências e cria uma lógica para a execução das metas dentro do ciclo de vida do sistema.

Maven - *Introduction* (2016), lista alguns dos benefícios em utilizar Maven em seu projeto:

---

<sup>4</sup> <https://www.oracle.com/index.html>

<sup>5</sup> <http://dev.mysql.com/doc/refman/5.7/en/mysqldump.html>

<sup>6</sup> <http://dev.mysql.com/doc/refman/5.7/en/mysqladmin.html>

<sup>7</sup> <http://dev.mysql.com/doc/refman/5.7/en/workbench.html>

<sup>8</sup> <http://ant.apache.org/>

- **Torna o processo de compilação fácil:** apesar de não eliminar a necessidade de saber sobre os mecanismos subjacentes, Maven prevê uma série de blindagens de alguns detalhes;
- **Fornecer um sistema de construção uniforme:** permite construir projetos usando um Modelo de Objetos do Projeto (POM) e um conjunto de *plugins* compartilhados com todos os projetos Maven. Isso proporciona um sistema de construção uniforme.
- **Fornecer informações sobre a qualidade do projeto:** o Maven pode oferecer muitas informações úteis sobre o projeto como, registro de alterações criadas diretamente a partir do controle de origem, listas de discussões, listas de dependências, além de relatórios de testes unitários.
- **Fornecer diretrizes para melhorar o desenvolvimento com boas práticas:** o Maven reúne os principais princípios de boas práticas para auxiliar no desenvolvimento do projeto com qualidade, como por exemplo, mantém uma espécie de árvore separada para testes, porém sincronizada com as versões de produção, usa convenções de nomenclaturas para buscar e executar testes e seu ambiente já possui configuração de caso de testes, sem depender de preparação para compilação.

## 2.4 Java

A linguagem de programação Java foi criada pelo *Green Team*, time criado pela *Sun Microsystems*<sup>9</sup>, com a intenção de desenvolver inovações tecnológicas. Esse time era liderado por James Gosling, considerado o pai do Java. A ideia era criar um interpretador para pequenos dispositivos, facilitando a reescrita de *softwares* para aparelhos eletrônicos. Após o fracasso em buscarem parcerias com grandes fabricantes de aparelhos eletrônicos, a *Sun* aproveitou o advento da *web* e percebeu que a ideia criada em 1992 poderia ser utilizada para rodar pequenas aplicações dentro do browser. A grande vantagem estaria no fato de haver grande quantidade de sistemas operacionais e *browsers* na Internet, e mesmo assim poderiam programar em uma única linguagem. A partir daí surgiu o Java 1.0. (Caelum, 2016).

É a *Java Virtual Machine* (JVM) que permite ao Java cumprir seu propósito inicial que é:

---

<sup>9</sup> <https://www.oracle.com/sun/index.html>

Escreva uma vez, execute em qualquer lugar. Diferente dos compiladores que geram códigos específicos para cada sistema operacional, a JVM possui uma camada a mais entre a aplicação e o sistema operacional, que é responsável por “traduzir” *bytecodes* Java e realizar as chamadas necessárias para o sistema operacional. A figura 1 ilustra como funciona a JVM. Além disso, a JVM também gerencia memória, pilha de execução, etc, e sua aplicação roda sem envolvimento com o sistema operacional, conversando apenas com a JVM.

Estima-se<sup>10</sup> que hoje exista no mundo mais de 9 milhões de desenvolvedores, 97% de *desktops* corporativos e mais de 3 bilhões de celulares executando Java.

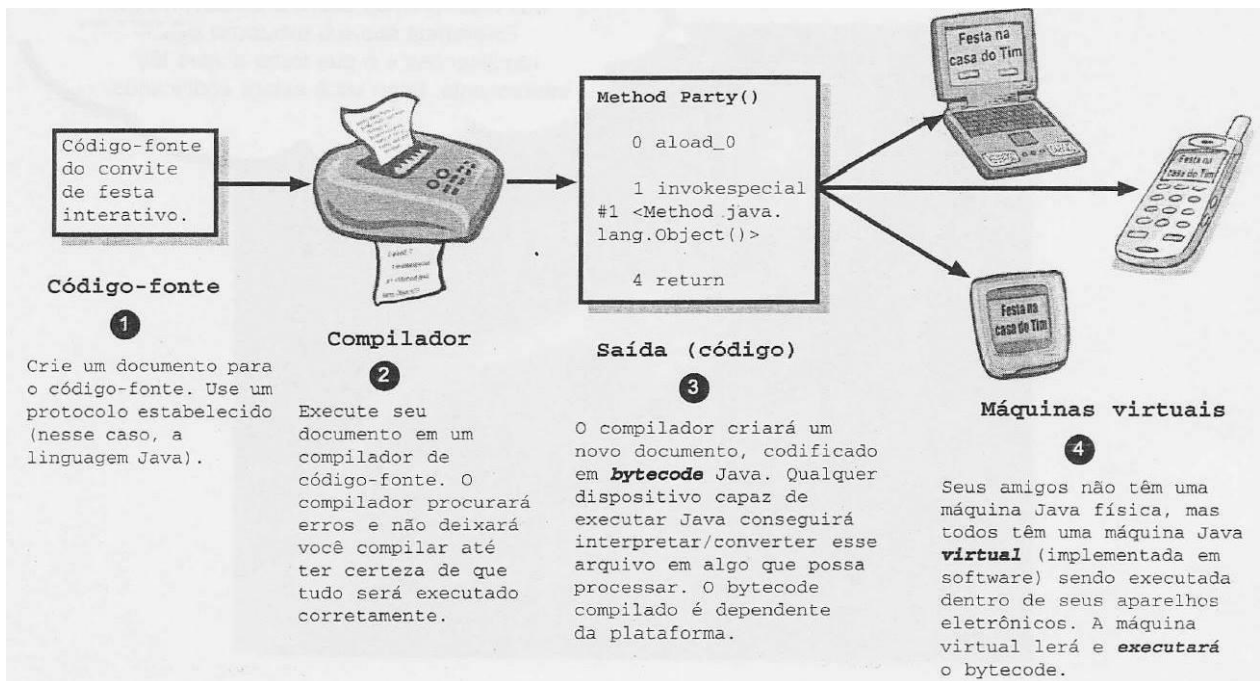


Figura 1 – Funcionamento da JVM (Sierra e Bert, 2009)

## 2.5 BootStrap

O *Bootstrap* foi criado originalmente no *Twitter*<sup>11</sup> em meados de 2010 por Mark Otto e Jacob Thornton. Antes de se tornar o mais conhecido *framework front-end* e o maior projeto de código do aberto do mundo, o *Bootstrap* era conhecido como *Twitter Blueprint*. Com poucos meses de desenvolvimento, o *Twitter* realizou seu primeiro *Hack Week*<sup>12</sup>. Foi onde o projeto

<sup>10</sup> [https://www.java.com/pt\\_BR/about/](https://www.java.com/pt_BR/about/)

<sup>11</sup> <https://twitter.com/>

<sup>12</sup> <https://blog.twitter.com/2010/hack-week>

explodiu entre os desenvolvedores de todos os níveis, tornando-se um guia de desenvolvimentos internos na empresa por mais de um ano antes do seu lançamento oficial e assim continua até hoje. (História - *Bootstrap*, 2016).

*Bootstrap* é na verdade uma coleção empacotada de vários elementos e funções que podem ser personalizadas, baseadas em combinações HTML, CSS e JavaScript.

Apesar de utilizar o CSS tradicional, o código fonte do *Bootstrap* utiliza os dois pré-processadores mais populares, *Less*<sup>13</sup> e *Sass*<sup>14</sup>. (*Bootstrap*, 2016).

## 2.6 Plataforma como Serviço (Paas)

O conceito Plataforma como Serviço pode ser definido como uma camada de *hardware* virtual, oferecida como serviço. Isso quer dizer que PaaS oferece diversos tipos de serviços como, sistemas operacionais, serviços de mensagem e banco de dados, além de fornecer ambientes de desenvolvimento de *software* pré configurados, facilitando o desenvolvimento e a implantação, minimizando a complexidade de gerenciamento de *hardware* e *software* necessários ao ambiente.

De acordo com BORGES, H. P., et al (2016), as plataformas de nuvens computacionais são diferenciadas pelo alto nível de abstração que oferecem em cada uma de suas camadas. Se compararmos com IaaS (Infra-estrutura como serviço), que fornece um conjunto de máquinas virtuais que devem ser configuradas e os componentes de aplicação devem ser implantados, PaaS oferece aos usuários uma forma para implantação de suas aplicações em um repositório aparentemente com recursos ilimitados, eliminando a complexidade de implantação e configuração de infra-estrutura.

Isso traz algumas vantagens ao utilizar plataforma como serviço:

- Menor risco ao negócio, pois não é necessário investimento com infra-estrutura e *softwares* básicos como sistemas operacionais;
- Novas funcionalidades são disponibilizadas em tempo real, sem a necessidade de especialistas para isso;
- Problemas encontrados na aplicação podem ser rapidamente corrigidos e implantados de forma transparente aos usuários;
- A empresa não se preocupará com a escolha e manutenção da infra-estrutura;

---

<sup>13</sup> <http://lesscss.org/>

<sup>14</sup> <http://sass-lang.com/>



- Aumento na segurança e disponibilidade dos dados;
- O sistema é facilmente escalável de acordo com o crescimento.

## 2.7 Google App Engine (GAE)

O Google App *Engine* é a plataforma como serviço do Google. Hoje essa plataforma suporta as linguagens de programação *Python*<sup>15</sup>, Java, PHP<sup>16</sup>, e Go<sup>17</sup>. Para esse trabalho, a linguagem escolhida foi Java.

O GAE, como é conhecido, executa o aplicativo *web* usando a Java *Virtual Machine* 7<sup>18</sup> em um ambiente seguro, chamado *sandbox*. Seu funcionamento é basicamente através da chamada de classes *servlets* da aplicação para processar pedidos e enviar respostas. Ele utiliza a versão 2.5 da especificação *servlet*<sup>19</sup>. O *sandbox* é uma espécie de caixa que isola as requisições e garante que os aplicativos só executem ações que não interfiram no desempenho e escalabilidade de outros aplicativos. Esse isolamento permite que o GAE distribua as requisições de aplicações para diversos servidores *web*, sem que um aplicativo interfira em outro. Isso garante segurança e escalabilidade, fazendo com que vários servidores sejam acionados de acordo com a necessidade. (Java *Runtime Environment*, 2016).

Para atender as requisições, o GAE recebe as solicitações da *web* para o aplicativo, invocando o *servlet* correspondente a URL definida no descritor de implantação<sup>20</sup> *web.xml*. Ele executa várias instâncias da aplicação tendo um servidor *web* para cada instância tratar as requisições. Qualquer requisição pode ser enviada para qualquer instância, mesmo sendo requisições consecutivas. O número de instância é ajustado automaticamente de acordo com o tráfego. (*How Requests are Handled*, 2016).

## 3 METODOLOGIA E DESENVOLVIMENTO

### 3.1 IDE

Existem ferramentas utilizadas no desenvolvimento de aplicações para auxiliar o desenvolvedor em algumas tarefas. As IDEs (*Integrated Development Environment*) como são

---

<sup>15</sup> <https://www.python.org/>

<sup>16</sup> <https://secure.php.net/>

<sup>17</sup> <https://golang.org/>

<sup>18</sup> <http://docs.oracle.com/javase/7/docs/>

<sup>19</sup> <https://jcp.org/aboutJava/communityprocess/mrel/jsr154/index.html>

<sup>20</sup> <https://cloud.google.com/appengine/docs/java/config/webxml>

conhecidas essas ferramentas, possuem pacotes, plug-ins e bibliotecas que otimizam o processo de desenvolvimento. Ainda podemos encontrar compiladores, console de logs, gerenciadores de dependência, editores de códigos e tudo o que a ferramenta pode oferecer para tornar o desenvolvimento mais produtivo.

As IDEs utilizadas neste artigo foram, o Eclipse<sup>21</sup>, pois é uma das ferramentas mais completas oferecendo diversas funcionalidades e atalhos, o MySQL *Workbench*<sup>22</sup> que também possui muitas funcionalidades que auxilia no desenvolvimento de scripts SQL e diagramas de banco de dados (DER) e por último, foi utilizada a ferramenta *Astah Community*<sup>23</sup> para construção dos diagramas da UML, pois essa ferramenta possui modelos para todos os diagramas no padrão da UML 2.0.

### 3.2 Desenvolvimento

Para o desenvolvimento do *SMK System* o primeiro passo foi o levantamento de requisitos para identificar as funcionalidades e características que o sistema deveria possuir. Foram levantados requisitos funcionais, como cadastro de produtos, cadastro de clientes, realização de vendas e geração de relatórios. Foram identificados também alguns requisitos de qualidade do sistema. Esses requisitos podem ser vistos no apêndice “A”.

Após o levantamento de requisitos, foi construído o diagrama de caso de uso. Ele contribuiu para a visualização do sistema do ponto de vista do usuário final, fornecendo uma melhor descrição das principais funcionalidades e interações com os atores do sistema. Esse diagrama pode ser visto no apêndice “B”.

O próximo passo foi a elaboração do documento de casos de uso textuais, estabelecendo os fluxos principais e alternativos, identificando atores e precondições para cada funcionalidade do sistema e descrevendo os principais cenários. O documento de casos de uso textuais pode ser visto no apêndice “C”.

Houve também uma análise para a escolha das tecnologias a serem utilizadas. Optou-se pela linguagem de programação Java para servir de experiência para trabalhos futuros, sendo essa uma das principais linguagens sob o conceito de Orientação a Objetos e facilita a organização da estrutura do projeto, utilizando o modelo MVC (*Model View Controller*). O MySQL também foi

---

<sup>21</sup> <https://eclipse.org/>

<sup>22</sup> <http://dev.mysql.com/doc/refman/5.7/en/workbench.html>

<sup>23</sup> <http://astah.net/editions/community>

o escolhido por se tratar de uma ferramenta *open source* e ser o mais popular gerenciador de banco de dados na comunidade.

Com as informações levantadas, o próximo passo foi realizar a modelagem do banco de dados através de um Diagrama de Entidade e Relacionamento (DER). Ele apresenta as tabelas do sistema com seus respectivos campos e relacionamentos, contendo chaves primárias e chaves estrangeiras. O DER pode ser visto no apêndice “D”.

Após o término da fase de análise do sistema, iniciou-se o desenvolvimento. Em primeiro lugar foram desenvolvidos os *scripts* para criação das tabelas no banco de dados utilizando a IDE *MySQL Workbench*. Depois, foi instalada a SDK do *Google App Engine* para Java<sup>24</sup>. Em seguida iniciou-se a configuração do Eclipse, instalando o *plugin* *Google App Engine*<sup>25</sup>. Para o desenvolvimento, o Maven também foi configurado<sup>26</sup>. Isso facilitou o início do projeto, pois com o Maven configurado, foi possível iniciar o projeto utilizando um dos *archetypes* pré-configurados. O *archetype* escolhido foi o *appengine-skeleton-archetype*. Essa escolha foi feita seguindo as figuras 2 e 3.

Ao preencher a tela apresentada na figura 4, o Eclipse junto com o Maven geram um projeto padronizado com os diretórios já definidos. Isso pode ser visto na figura 5.

Foi criado também um projeto no console do Google<sup>27</sup>. Para isso é necessário possuir uma conta Google e estar logado. Houve uma última configuração no Eclipse, onde é feita a associação do projeto criado no console com o projeto do Eclipse, conforme mostrado na figura 6. Criou-se também uma instância no Cloud SQL<sup>28</sup>.

---

<sup>24</sup> [https://cloud.google.com/appengine/docs/java/download#java\\_linux](https://cloud.google.com/appengine/docs/java/download#java_linux)

<sup>25</sup> <https://developers.google.com/eclipse/docs/install-eclipse-4.5>

<sup>26</sup> <https://cloud.google.com/appengine/docs/java/tools/maven>

<sup>27</sup> <https://cloud.google.com/appengine/docs/java/quickstart>

<sup>28</sup> <https://cloud.google.com/sql/docs/create-instance>

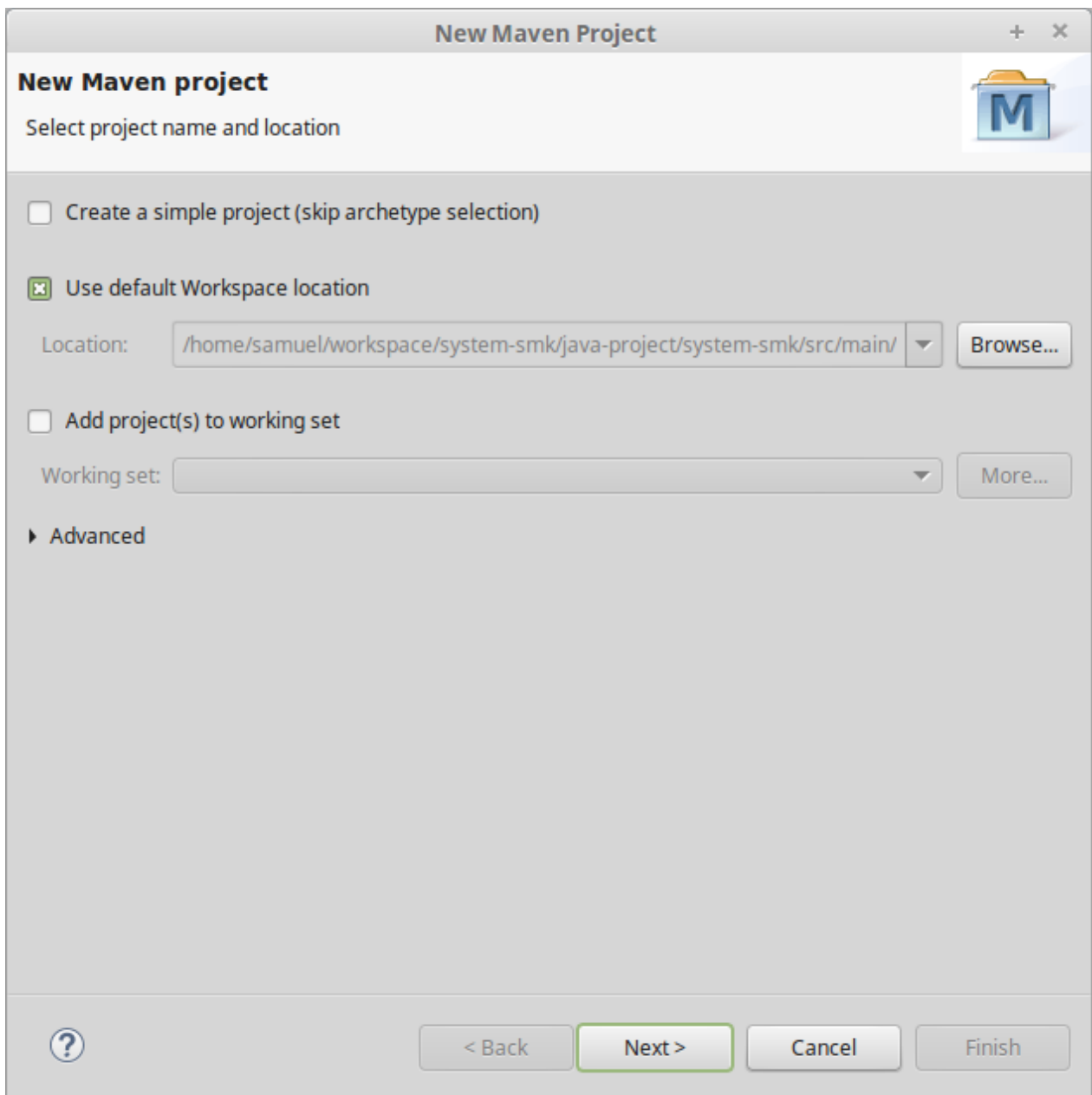


Figura 2 – Escolha do repositório local (o autor)

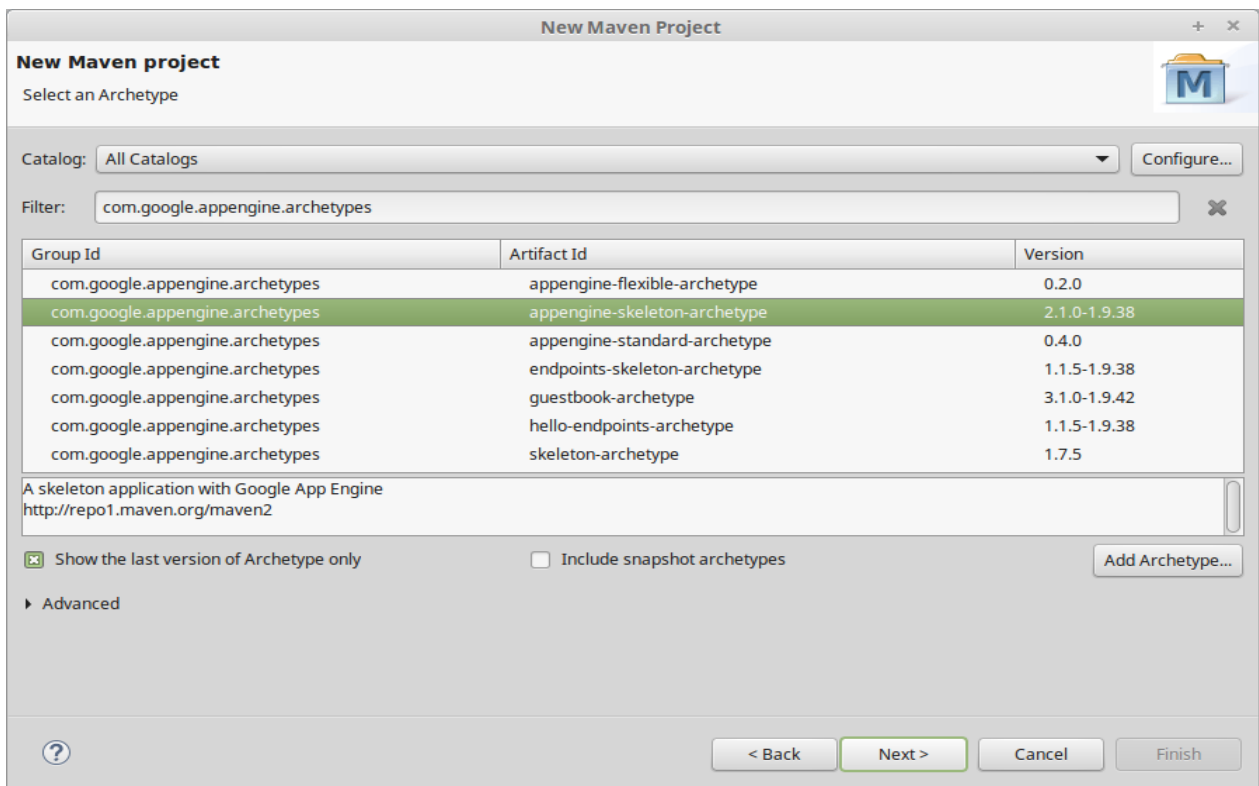
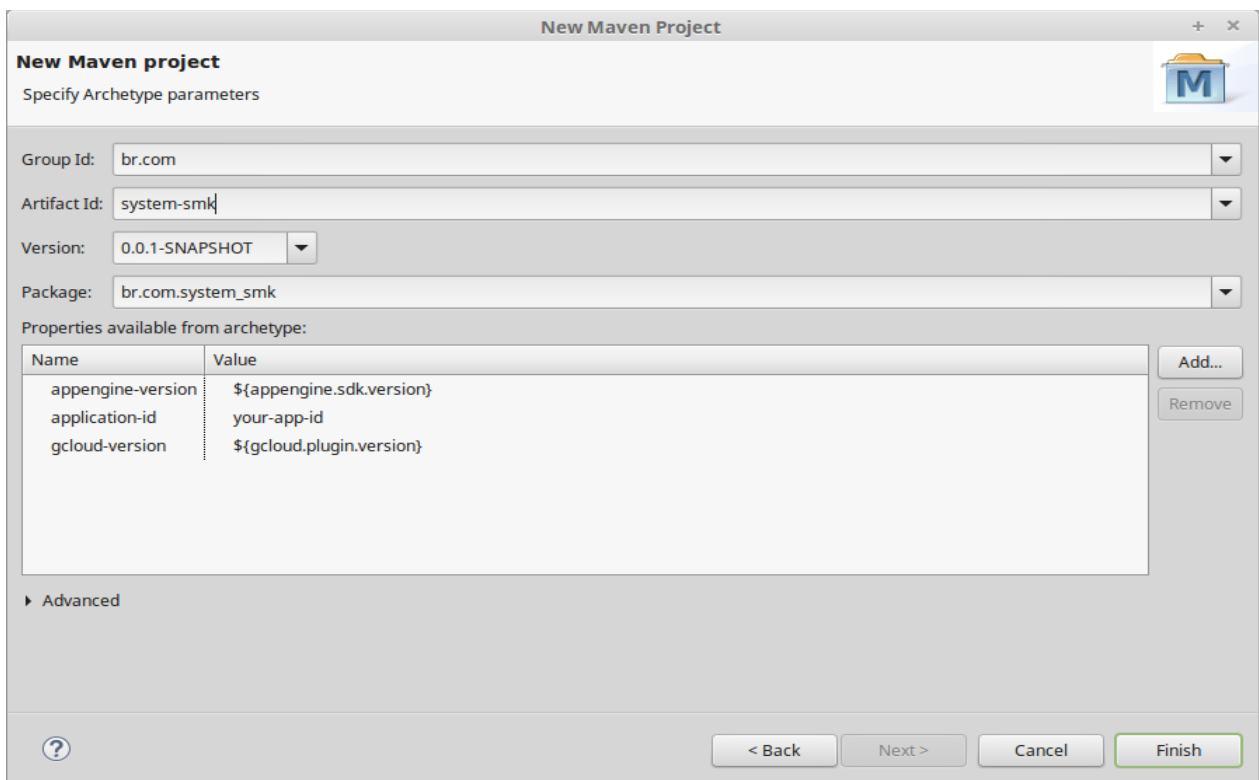
Figura 3 – Escolha do *archetype* (o autor)

Figura 4 – Informações para criação do projeto. (o autor)

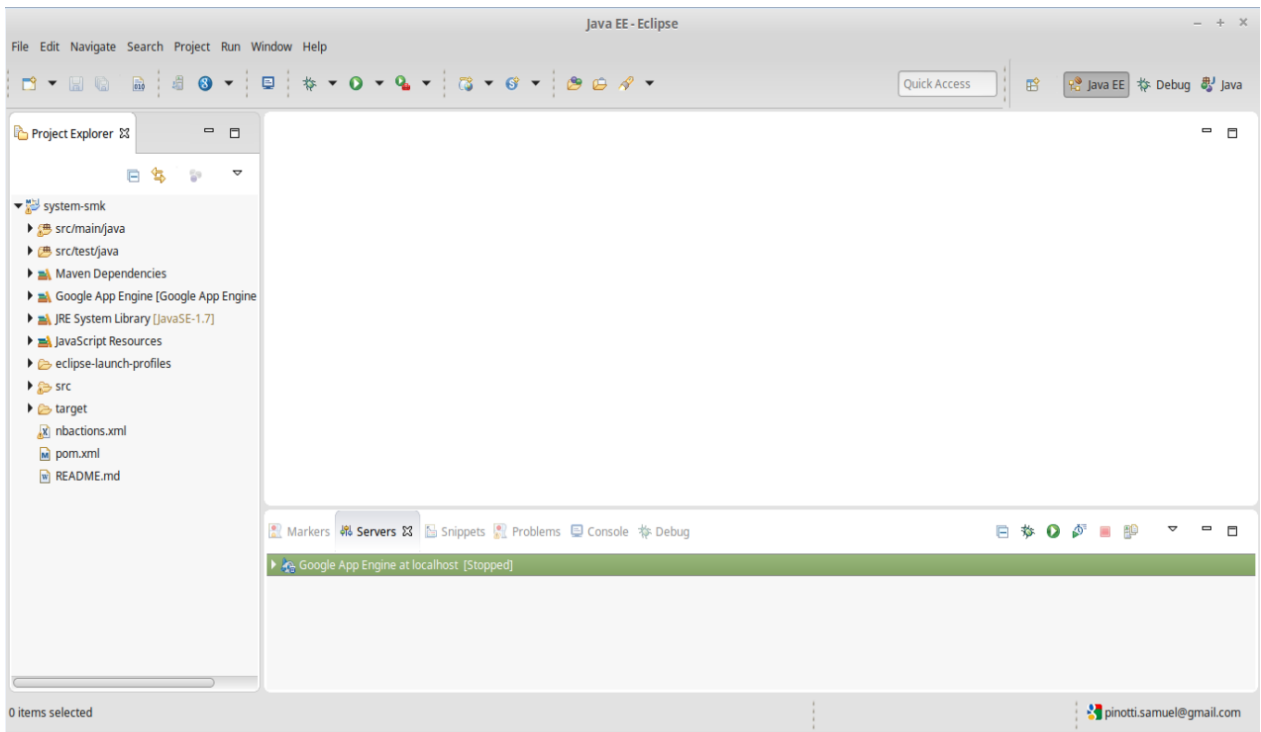


Figura 5 – Estrutura do projeto. (o autor)

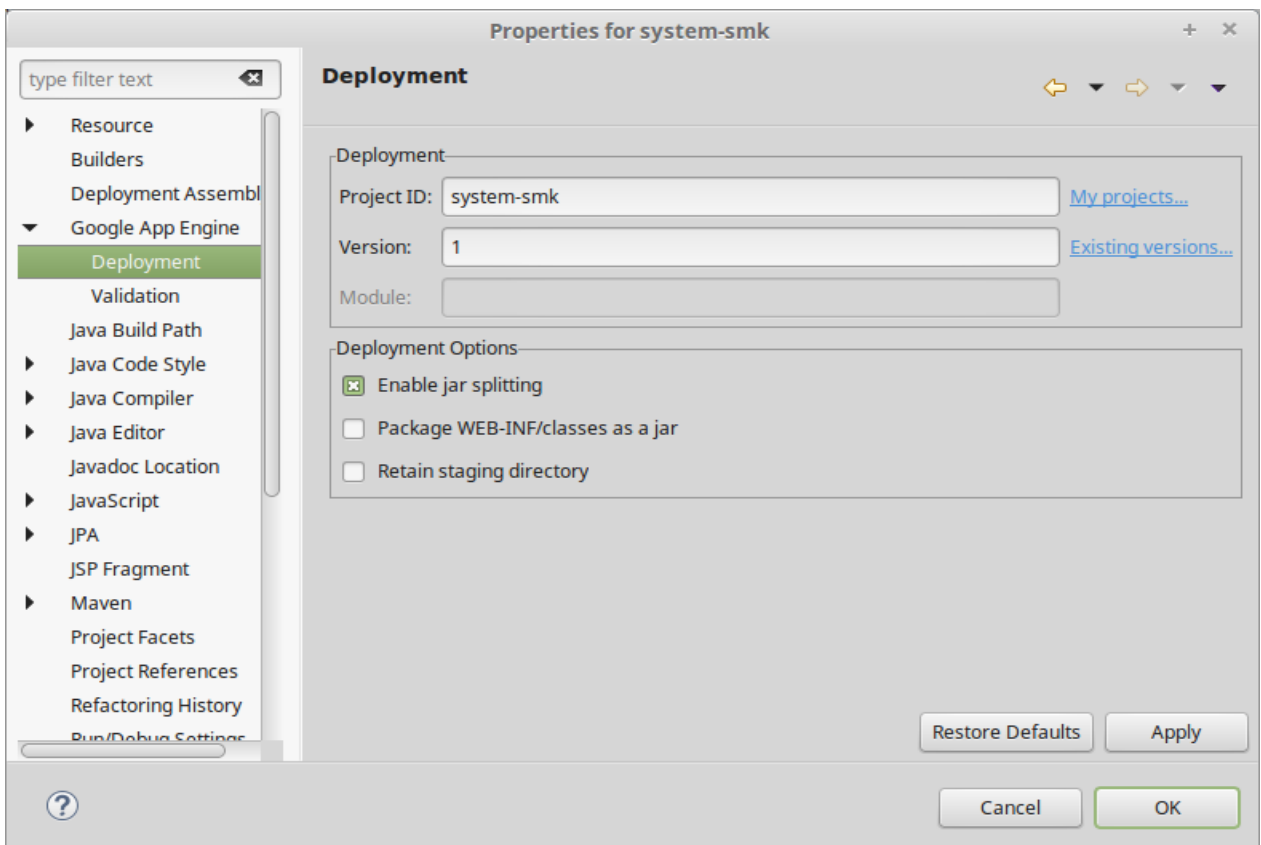


Figura 6 – Associação do projeto ao Eclipse. (o autor)

Com toda a configuração pronta, o desenvolvimento ocorreu de forma natural, da mesma forma de um projeto *web* padrão.

O *deploy* da aplicação foi feito clicando com o botão direito do *mouse* em cima do projeto no Eclipse e escolhendo as opções “Google App Engine WTP > Deploy Project to Remote Server”. O Maven faz o *build* do projeto e em seguida o *deploy* da aplicação no projeto criado anteriormente no console do Google. Por padrão o Google oferece um domínio gratuito da seguinte maneira: [http://<id\\_do\\_projeto>.appspot.com/](http://<id_do_projeto>.appspot.com/). Nesse caso, a aplicação está hospedada sob o domínio <https://system-smk.appspot.com/>, acessível na data da apresentação deste trabalho.

#### 4 CONCLUSÃO

O desenvolvimento desse trabalho ampliou de maneira significativa os conhecimentos sobre tecnologias obtidos no decorrer do curso, além de acrescentar novos conhecimentos, como funções jQuery, requisições Ajax com JavaScript e o conceito de plataforma como serviço.

Do ponto de vista de análise de sistemas, pode se dizer que a UML é essencial para as fases iniciais de qualquer projeto, onde ocorrem todos os estudos e levantamentos junto ao usuário. A UML traduz todas essas definições de forma clara e objetiva para todas as partes envolvidas no projeto.

O resultado desse trabalho é o *SMK System*, um sistema simples, mas que mostra alguns dos principais recursos disponíveis para o desenvolvimento de *software* e hospedagem. Esse sistema está representado no apêndice “E”.

Foi possível notar algumas características marcantes sobre o GAE, a plataforma apresentada nesse trabalho. O GAE se mostrou muito eficiente na parte do *deploy* da aplicação. Tendo o ambiente configurado corretamente no início do projeto, as entregas da aplicação são feitas em praticamente dois clicks. Isso contribui muito para o conceito de entregas contínuas de aplicações e torna o GAE uma ótima escolha para hospedagem de sistemas.

Existem também alguns problemas percebidos durante a utilização dessa plataforma. Por exemplo, o Google está sempre induzindo seus usuários a utilizar as ferramentas disponibilizadas por eles próprios. Exemplo disso é a armazenagem de dados. O Google oferece o Google *Data Store*<sup>29</sup>, sob o conceito de banco de dados não relacional e o Google *Cloud SQL*, que nada mais é do que uma instância MySQL, que foi a utilizada nesse trabalho. Ao optar pela versão do

---

<sup>29</sup> <https://cloud.google.com/datastore/docs/concepts/overview>

MySQL, não existe plano gratuito, apenas uma versão *trial* por alguns dias. Isso pode se tornar inviável escolhendo o GAE, sendo que existem no mercado soluções gratuitas por tempo indeterminado, limitando apenas o tamanho do banco.

Outro ponto negativo percebido foi a parte de documentação. Apesar do Google possuir uma vasta documentação muito rica em detalhes, se optar por utilizar versões que não sejam as mais atuais, por exemplo, o Java 7 em vez do Java 8, podem ocorrer problemas que não são explicados nessas documentações, ou seja, o Google está induzindo os usuários a utilizar as plataformas nas versões mais atualizadas. Por um lado isso é bom, pois força o desenvolvedor a trabalhar sempre com as mais novas tecnologias, mas para aplicações já hospedadas, pode se tornar um problema.

Para trabalhos futuros a intenção é adicionar novas funcionalidades ao sistema, como controle de fluxo de caixa e o cadastro de produtos em consignação, além de melhorar as funcionalidades já desenvolvidas. Pode ser que o sistema seja hospedado em outra plataforma.

## REFERÊNCIAS

CARLSON, L., 2013. *Programming for PaaS*. 1ª ed. Sebastopol: O'Reilly Media.

MELO, A. C., 2004. *Desenvolvendo aplicações com UML 2.0: do conceitual à implementação*. 2ª ed. Rio de Janeiro: Brasport.

MySQL - MySQL 5.7 Reference Manual - 1.3.1 What is MySQL?, 2016. Disponível em: <<http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>>. Acesso em: Novembro de 2016.

MySQL - MySQL 5.7 Reference Manual - 1.3.2 The Main Features of MySQL, 2016. Disponível em <<http://dev.mysql.com/doc/refman/5.7/en/features.html>>. Acesso em: Novembro de 2016.

SONATYPE Company, 2008. *Maven. The Definitive Guide*. 1ª ed. Sebastopol: O'Reilly Media.

**Maven - Introduction**, 2016. Disponível em: <<https://maven.apache.org/what-is-maven.html>>. Acesso em: Novembro de 2016.



CAELUM. **O que é Java - Java e Orientação a Objetos**. Disponível em <<https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/>>. Acesso em: Novembro de 2016.

SIERRA, K.; BERT, B., 2009. **Use a Cabeça! Java**. 2ª ed. Rio de Janeiro: Alta Books.

**História - Bootstrap**. Disponível em <<http://getbootstrap.com.br/v4/about/history/>>. Acesso em: Novembro de 2016.

**Bootstrap**. Disponível em <<http://getbootstrap.com.br/>>. Acesso em: Novembro de 2016.

BORGES, H. P., et al. **Computação em Nuvem**. Disponível em <[http://livroaberto.ibict.br/bitstream/1/861/1/COMPUTA%C3%87%C3%83O%20EM%20NUVE M.pdf](http://livroaberto.ibict.br/bitstream/1/861/1/COMPUTA%C3%87%C3%83O%20EM%20NUVE%20M.pdf)>. Acesso em: Novembro de 2016.

**Java Runtime Environment / App Engine standard environment for Java / Google Cloud Platform**. Disponível em <<https://cloud.google.com/appengine/docs/java/runtime>>. Acesso em: Novembro de 2016.

**How Requests are Handled / App Engine standard environment for Java / Google Cloud Platform**. Disponível em <<https://cloud.google.com/appengine/docs/java/how-requests-are-handled>>. Acesso em: Novembro de 2016.

## APÊNDICE A – DOCUMENTO DE REQUISITOS

### 1 Visão geral do sistema

O principal objetivo do sistema é auxiliar os usuários na realização de vendas, controle de estoque e visualização de relatórios. Os usuários serão funcionários capacitados para realização de funções administrativas e vendas. O sistema deve ser objetivo e intuitivo em suas ações.

### 2 Requisitos Funcionais

#### 2.1 Requisitos Funcionais de Entrada

1. O sistema deve permitir o cadastro, alteração e exclusão de novos usuários.
2. O sistema deve permitir o cadastro de novos usuários, solicitando as informações necessárias, que são: Nome, E mail, Telefone e Celular.
3. O sistema deve gerar uma senha automática no momento do cadastro de novos usuários e solicitar a troca durante o primeiro acesso.
4. O sistema deve permitir a alteração de qualquer informação do usuário, desde que esse exista na base de dados.
5. O sistema deve permitir a exclusão de qualquer usuário, desde que esse exista na base de dados.
6. O sistema deve permitir o cadastro, alteração e exclusão de novos clientes.
7. O sistema deve permitir o cadastro de novos clientes, solicitando ao usuário as informações necessárias que são: Nome, CPF, E mail, Telefone, Celular, Endereço, Bairro, Número, Cidade, Estado e Observação.
8. O sistema deve permitir ao usuário alterar qualquer informação do cliente, desde que esse exista na base de dados.
9. O sistema deve permitir ao usuário excluir qualquer cliente, desde que esse exista na base de dados.
10. O sistema deve permitir o cadastro, alteração e exclusão de novos produtos.
11. O sistema deve permitir o cadastro de novos produtos, solicitando ao usuário as informações necessárias que são: Código, Descrição, Tamanho, Cor, Valor de Compra, Valor de Venda, Quantidade e Data de Compra.
12. O sistema deve permitir ao usuário alterar qualquer informação do produto, desde que esse exista na base de dados.

13. O sistema deve permitir ao usuário excluir qualquer produto, desde que esse exista na base de dados.

## **2.2 Requisitos Funcionais de Processamento**

14. O sistema deve permitir a alteração da senha do usuário quando for o primeiro acesso.
15. O sistema deve permitir a consulta de usuários cadastrados através do nome.
16. O sistema deve permitir a consulta de produtos cadastrados através do código.
17. O sistema deve permitir a consulta de clientes cadastrados através do nome.
18. O sistema deve permitir a realização de vendas.
19. O sistema deve permitir a consulta das vendas realizadas, filtrando por determinado período.

## **2.3 Requisitos de Saída**

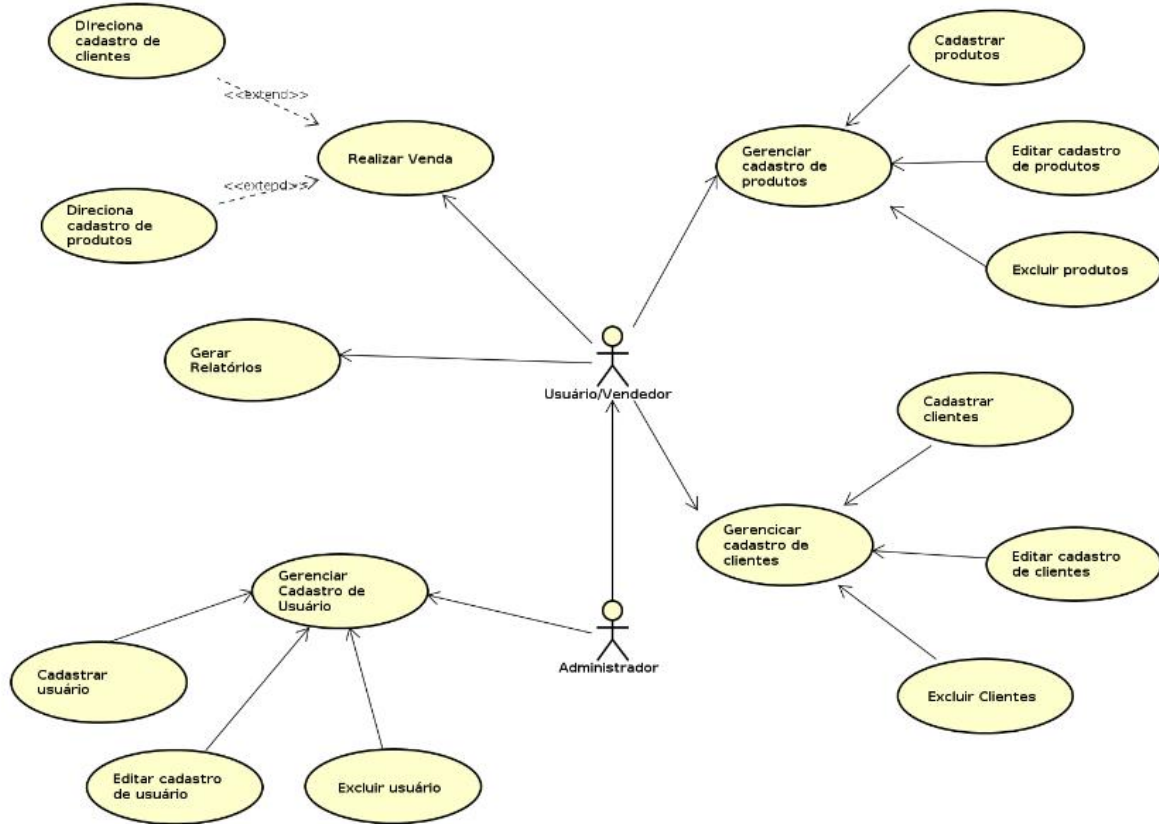
20. O sistema deve emitir relatório com os usuários cadastrados.
21. O sistema deve emitir relatório de produtos em estoque.
22. O sistema deve emitir relatório dos clientes cadastrados.
23. O sistema deve emitir relatório de vendas.

# **3 Requisitos de Qualidade**

## **3.1 Portabilidade**

24. O sistema deve funcionar corretamente nos principais navegadores (ex: Internet Explorer, Mozilla Firefox, Google Chrome, etc).

## APÊNDICE B – DIAGRAMA DE CASO DE USO



## APÊNDICE C – CASO DE USO TEXTUAL

### Caso de Uso: Cadastro de Usuário

**Descrição:** Administrador realiza o cadastro de novos usuários.

**Ator:** Administrador

**Pré-condições:** Administrador deve possuir acesso ao sistema.

#### Cenário Principal:

1. Administrador clica no menu Cadastros em seguida clica em Usuário.
2. O sistema exibe uma tela com um formulário de cadastro.
3. O administrador preenche o formulário com campos: Nome do Usuário, E mail, Telefone e Celular.
4. O administrador clica no botão Salvar.

5. O sistema exibe uma mensagem informando o sucesso do cadastro e senha padrão.
6. O sistema retorna para a tela inicial.

**Cenário Alternativo:**

1. O sistema informa que o e mail informado já está cadastrado.

**Caso de Uso:** Cadastro de Clientes

**Descrição:** Usuário cadastra novos clientes.

**Ator:** Usuário

**Pré-condições:** Usuário deve possuir acesso ao sistema.

**Cenário Principal:**

1. Usuário clica no menu Cadastros e em seguida clica em Clientes.
2. O sistema exibe uma tela com um formulário de cadastro.
3. O usuário preenche o formulário com campos: Nome do Cliente, CPF, E mail, Telefone, Celular, Endereço, Bairro, Número, Cidade, Estado e Observação.
4. O usuário clica no botão Salvar.
5. O sistema exibe uma mensagem informando o sucesso do cadastro.
6. O sistema retorna para a tela inicial.

**Cenário Alternativo:**

1. O sistema alerta que o CPF informado já está cadastrado.

**Caso de Uso:** Cadastro de Produtos

**Descrição:** Usuário cadastra novos produtos.

**Ator:** Usuário

**Pré-condições:** Usuário deve possuir acesso ao sistema.

**Cenário Principal:**

1. Usuário clica no menu Cadastros e em seguida clica em Produtos.
2. O sistema exibe uma tela com um formulário de cadastro.
3. O usuário preenche o formulário com campos: Código, Descrição, Tamando, Cor, Valor de Compra, Valor de Venda, Quantidade e Data de Compra.
4. O usuário clica no botão Salvar.

5. O sistema exibe uma mensagem informando o sucesso do cadastro.
6. O sistema retorna para a tela inicial.

**Caso de Uso:** Realização de Novas Vendas

**Descrição:** Usuário realiza novas vendas.

**Ator:** Usuário

**Pré-condições:** Usuário deve possuir acesso ao sistema.

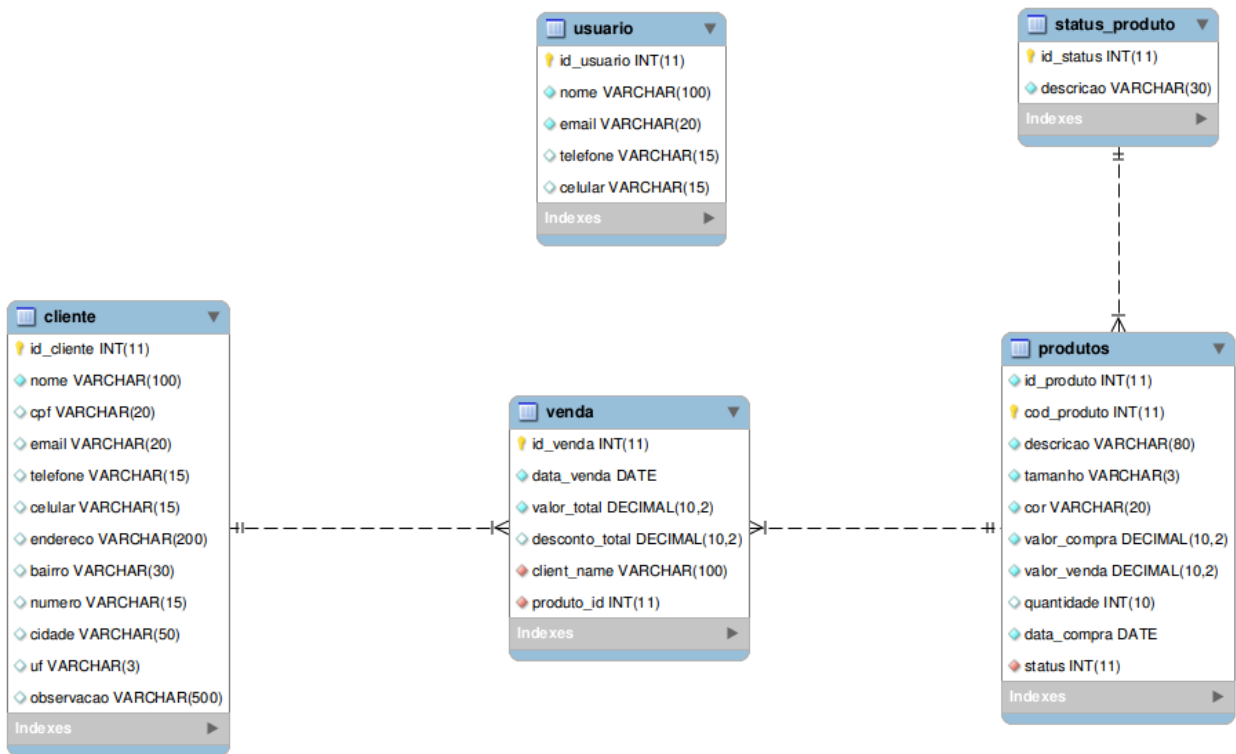
**Cenário Principal:**

1. Usuário clica no menu Vendas.
2. O sistema exibe uma tela com um formulário de vendas.
3. O usuário inicia o preenchimento do campo Nome do Cliente, e o sistema sugere nomes de acordo com o que existe na base de dados.
4. O usuário preenche o campo Código.
5. O sistema preenche automaticamente os campos: Descrição e Valor.
6. O usuário preenche o campo: Data.
7. O usuário clica no botão Salvar.
8. O sistema exibe uma mensagem informando o sucesso da realização da venda.
9. O sistema retorna para a tela inicial.

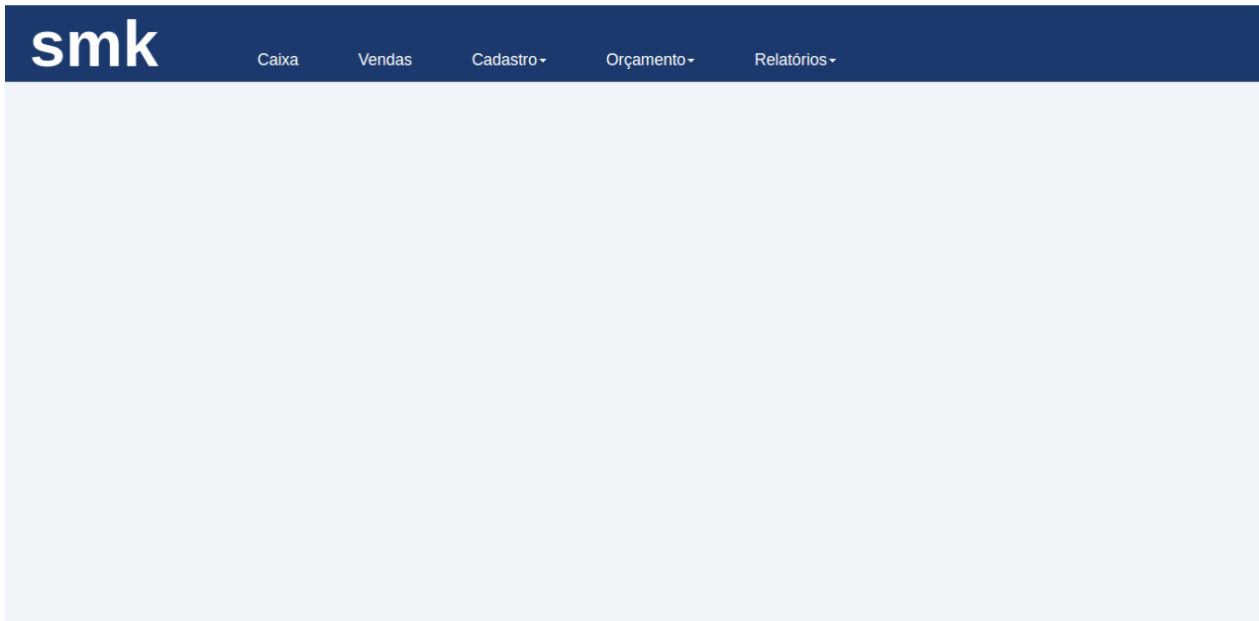
**Cenário Alternativo:**

1. O sistema alerta que o cliente informado não consta cadastrado e direciona para a tela de cadastro de clientes.
2. O sistema alerta que o produto informado não consta cadastrado e direciona para tela de cadastro de clientes.

## APÊNDICE D – DIAGRAMA DE ENTIDADE E RELACIONAMENTO (DER)



## APÊNDICE E – REPRESENTAÇÃO DO SISTEMA



**smk** Caixa Vendas Cadastro- Orçamento- Relatórios-

Código \* Descrição \*

Tamanho Cor

Valor de Compra Valor de Venda

Quantidade Data de Compra

Salvar

**smk** Caixa Vendas Cadastro- Orçamento- Relatórios-









Nome do cliente \*

Código \* Descrição \*

Valor Data

Salvar



Código	Nome	Email	Telefone	Celular	Endereço	Bairro	Número	Observação	Ação
12	Teste1	teste@teste.com	14789632	147896325	teste	teste	123	teste	 
13	Teste2								 
14	Teste3	teste@teste.com	1612345678	1612345678	Rua Jose Bonifacio	Vila Santa Cruz	123	teste	 
15	Teste4								 
16	Teste5								